Probabilistic Verification of Timing Constraints in Automotive Systems Using UPPAAL-SMC

Eun-Young Kang[1,2(✉)], Dongrui Mu[2], and Li Huang[2]

[1] University of Namur, Namur, Belgium
eykang@unamur.be
[2] School of Data and Computer Science, Sun Yat-Sen University, Guangzhou, China
{mudr,huangl223}@mail2.sysu.edu.cn

Abstract. Modeling and analysis of non-functional properties, such as timing constraints, is crucial in automotive real-time embedded systems. East-adl is a domain specific architectural language dedicated to safety-critical automotive embedded system design. We have previously specified East-adl timing constraints in Clock Constraint Specification Language (Ccsl) and proved the correctness of specification by mapping the semantics of the constraints into Uppaal models amenable to model checking. In most cases, a bounded number of violations of timing constraints in automotive systems would not lead to system failures when the results of the violations are negligible, called Weakly-Hard (WH). Previous work is extended in this paper by including support for probabilistic analysis of timing constraints in the context of WH: Probabilistic extension of Ccsl, called PrCcsl, is defined and the East-adl timing constraints with stochastic properties are specified in PrCcsl. The semantics of the extended constraints in PrCcsl is translated into Uppaal-SMC models for formal verification. Furthermore, a set of mapping rules is proposed to facilitate guarantee of translation. Our approach is demonstrated on an autonomous traffic sign recognition vehicle case study.

Keywords: East-adl · Uppaal-SMC · Probabilistic Ccsl
Weakly-Hard System · Statistical model checking

1 Introduction

Model-driven development is rigorously applied in automotive systems in which the software controllers interact with physical environments. The continuous time behaviors (evolved with various energy rates) of those systems often rely on complex dynamics as well as on stochastic behaviors. Formal verification and validation (V&V) technologies are indispensable and highly recommended for development of safe and reliable automotive systems [3,4]. Conventional V&V, i.e., testing and model checking have limitations in terms of assessing the reliability of hybrid systems due to both the stochastic and non-linear dynamical

C. A. Furia and K. Winter (Eds.): IFM 2018, LNCS 11023, pp. 236–254, 2018.
https://doi.org/10.1007/978-3-319-98938-9_14

features. To ensure the reliability of safety critical hybrid dynamic systems, *statistical model checking (SMC)* techniques have been proposed [11,12,25]. These techniques for fully stochastic models validate probabilistic performance properties of given deterministic (or stochastic) controllers in given stochastic environments.

Conventional formal analysis of timing models addresses worst case designs, typically used for hard deadlines in safety critical systems, however, there is great incentive to include "less-than-worst-case" designs to improve efficiency but without affecting the quality of timing analysis in the systems. The challenge is the definition of suitable model semantics that provide reliable predictions of *system timing*, given the timing of individual components and their compositions. While the standard worst case models are well understood in this respect, the behavior and the expressiveness of "less-than-worst-case" models is far less investigated. In most cases, a bounded number of violations of timing constraints in systems would not lead to system failures when the results of the violations are negligible, called Weakly-Hard (WH) [8,29]. In this paper, we propose a formal probabilistic modeling and analysis technique by extending the known concept of WH constraints to what is called "typical" worst case model and analysis.

East-adl (Electronics Architecture and Software Technology - Architecture Description Language) [5,14], aligned with AUTOSAR (Automotive Open System Architecture) standard [1], is a concrete example of the MBD approach for the architectural modeling of safety-critical automotive embedded systems. A system in East-adl is described by `Functional Architectures (FA)` at different abstraction levels. The `FA` are composed of a number of interconnected *functionprototypes* (f_p), and the f_ps have ports and connectors for communication. East-adl relies on external tools for the analysis of specifications related to requirements. For example, behavioral description in East-adl is captured in external tools, i.e., Simulink/Stateflow[32]. The latest release of East-adl has adopted the time model proposed in the Timing Augmented Description Language (Tadl2) [9]. Tadl2 expresses and composes the basic timing constraints, i.e., repetition rates, end-to-end delays, and synchronization constraints. The time model of Tadl2 specializes the time model of MARTE, the UML profile for Modeling and Analysis of Real-Time and Embedded systems [30]. MARTE provides Ccsl, a time model and a Clock Constraint Specification Language, that supports specification of both logical and dense timing constraints for MARTE models, as well as functional causality constraints [27].

We have previously specified non-functional properties (timing and energy constraints) of automotive systems specified in East-adl and MARTE/Ccsl, and proved the correctness of specification by mapping the semantics of the constraints into Uppaal models for model checking [23]. Previous work is extended in this paper by including support for probabilistic analysis of timing constraints of automotive systems in the context WH: 1. Probabilistic extension of Ccsl, called PrCcsl, is defined and the East-adl/Tadl2 timing constraints with stochastic properties are specified in PrCcsl; 2. The semantics of the extended constraints in PrCcsl is translated into verifiable Uppaal-SMC [2] models for

formal verification; 3. A set of mapping rules is proposed to facilitate guarantee of translation. Our approach is demonstrated on an autonomous traffic sign recognition vehicle (AV) case study.

The paper is organized as follows: Sect. 2 presents an overview of CCSL and UPPAAL-SMC. The AV is introduced as a running example in Sect. 3. Section 4 presents the formal definition of PrCCSL. Section 5 describes a set of translation patterns from CCSL/PrCCSL to UPPAAL-SMC models and how our approaches provide support for formal analysis at the design level. The applicability of our method is demonstrated by performing verification on the AV case study in Sect. 6. Sections 7 and 8 present related work and the conclusion.

2 Preliminary

In our framework, we consider a subset of CCSL and its extension with stochastic properties that is sufficient to specify EAST-ADL timing constraints in the context of WH. Formal Modeling and V&V of the EAST-ADL timing constraints specified in CCSL are performed using UPPAAL-SMC.

Clock Constraint Specification Language (CCSL) [6,27] is a UML profile for modeling and analysis of real-time systems (MARTE) [7,26]. In CCSL, a clock represents a sequence of (possibly infinite) instants. An event is a clock and the occurrences of an event correspond to a set of ticks of the clock. CCSL provides a set of clock constraints that specifies evolution of clocks' ticks. The physical time is represented by a dense clock with a base unit. A dense clock can be discretized into a discrete/logical clock. *idealClock* is a predefined dense clock whose unit is second. We define a universal clock *ms* based on *idealClock*: $ms = idealClock$ `discretizedBy` 0.001. *ms* representing a periodic clock that ticks every 1 ms in this paper. A step is a tick of the universal clock. Hence the length of one step is 1 ms.

CCSL provides two types of clock constraints, *relation* and *expression*: A *relation* limits the occurrences among different events/clocks. Let C be a set of clocks, $c1, c2 \in C$, `coincidence` relation ($c1 \equiv c2$) specifies that two clocks must tick simultaneously. `Precedence` relation ($c1 \prec c2$) delimits that $c1$ runs faster than $c2$, i.e., $\forall k \in \mathbb{N}^+$, where $\mathbb{N}^+$ is the set of positive natural numbers, the k^{th} tick of $c1$ must occur prior to the k^{th} tick of $c2$. `Causality` relation ($c1 \preceq c2$) represents a relaxed version of `precedence`, allowing the two clocks to tick at the same time. `Subclock` ($c1 \subseteq c2$) indicates the relation between two clocks, *superclock* ($c1$) and *subclock* ($c2$), s.t. each tick of the subclock must correspond to a tick of its superclock at the same step. `Exclusion` ($c1 \# c2$) prevents the instants of two clocks from being coincident. An *expression* derives new clocks from the already defined clocks: `periodicOn` builds a new clock based on a *base* clock and a *period* parameter, s.t., the instants of the new clocks are separated by a number of instants of the *base* clock. The number is given as *period*. `DelayFor` results in a clock by delaying the *base* clock for a given number of ticks of a *reference* clock. `Infimum`, denoted `inf`, is defined as the slowest clock that is

faster than both $c1$ and $c2$. `Supremum`, denoted `sup`, is defined as the fastest clock that is slower than $c1$ and $c2$.

UPPAAL-SMC performs the probabilistic analysis of properties by monitoring simulations of complex hybrid systems in a given stochastic environment and using results from the statistics to determine whether the system satisfies the property with some degree of confidence. Its clocks evolve with various rates, which are specified with *ordinary differential equations* (ODE). UPPAAL-SMC provides a number of queries related to the stochastic interpretation of Timed Automata (STA) [12] and they are as follows, where N and *bound* indicate the number of simulations to be performed and the time bound on the simulations respectively:

1. *Probability Estimation* estimates the probability of a requirement property ϕ being satisfied for a given STA model within the time bound: $Pr[bound]\ \phi$.
2. *Hypothesis Testing* checks if the probability of ϕ being satisfied is larger than or equal to a certain probability P_0: $Pr[bound]\ \phi\ \geqslant\ P_0$.
3. *Probability Comparison* compares the probabilities of two properties being satisfied in certain time bounds: $Pr[bound_1]\ \phi_1 \geqslant Pr[bound_2]\ \phi_2$.
4. *Expected Value* evaluates the minimal or maximal value of a clock or an integer value while UPPAAL-SMC checks the STA model: $E[bound; N](min : \phi)$ or $E[bound; N](max : \phi)$.
5. *Simulations*: UPPAAL-SMC runs N simulations on the STA model and monitors k (state-based) properties/expressions $\phi_1, ..., \phi_k$ along the simulations within simulation bound *bound*: $simulate\ N\ [\leqslant bound]\{\phi_1, ..., \phi_k\}$.

3 Running Example: Traffic Sign Recognition Vehicle

An autonomous vehicle (AV) [21,22] application using Traffic Sign Recognition is adopted to illustrate our approach. The AV reads the road signs, e.g., "speed limit" or "right/left turn", and adjusts speed and movement accordingly. The functionality of AV, augmented with timing constraints and viewed as `Functional Design Architecture (FDA)` (`designFunctionTypes`), consists of the following f_ps in Fig. 1: `System` function type contains four f_ps, i.e., the `Camera` captures sign images and relays the images to `SignRecognition` periodically. `Sign Recognition` analyzes each frame of the detected images and computes the desired images (sign types). `Controller` determines how the speed of the vehicle is adjusted based on the sign types and the current speed of the vehicle. `VehicleDynamic` specifies the kinematics behaviors of the vehicle. `Environment` function type consists of three f_ps, i.e., the information of traffic signs, random obstacles, and speed changes caused by environmental influence described in `TrafficSign`, `Obstacle`, and `Speed` f_ps respectively.

We consider the `Periodic, Execution, End-to-End, Synchronization, Sporadic, and Comparison` timing constraints on top of the AV EAST-ADL model, which are sufficient to capture the constraints described in Fig. 1. Furthermore, we extend EAST-ADL/TADL2 with an `Exclusion` timing constraint

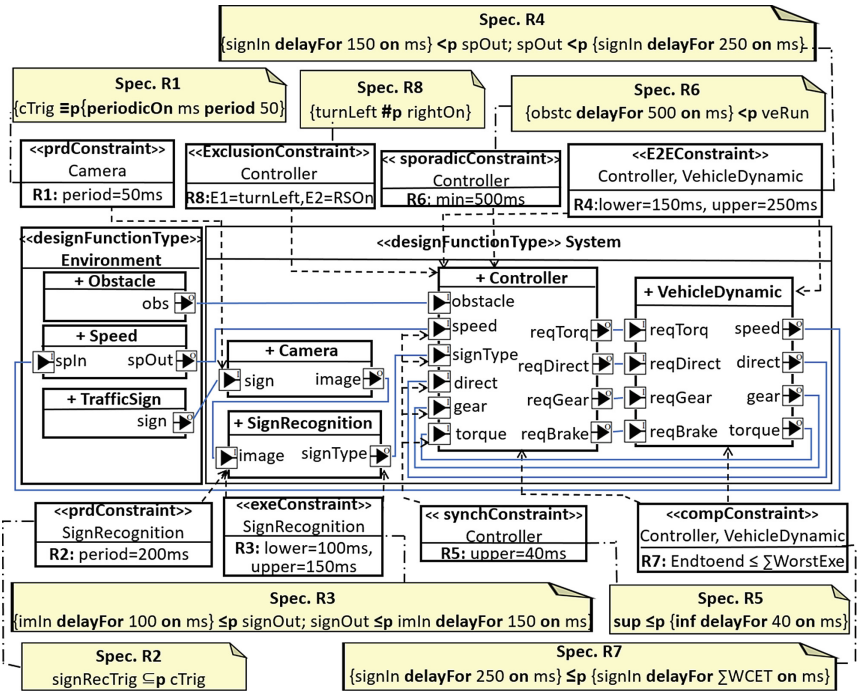


Fig. 1. AV in EAST-ADL augmented with TADL2 constraints (`R. IDs`) specified in PRCCSL (`Spec. R. IDs`)

(R8 in Fig. 1) that integrates relevant concepts from the CCSL constraint, i.e., two events cannot occur simultaneously.

R1. The camera must capture an image every 50 ms. In other words, a `Periodic` acquisition of `Camera` must be carried out every 50 ms.
R2. The captured image must be recognized by an AV every 200 ms, i.e., a `Periodic` constraint on `SignRecognition` f_p.
R3. The detected image should be computed within [100, 150] ms in order to generate the desired sign type, the `SignRecognition` must complete its execution within [100, 150] ms.
R4. When a traffic sign is recognized, the speed of AV should be updated within [150, 250] ms. An `End-to-End` constraint on `Controller` and `VehicleDynamic`, i.e., the time interval from the input of `Controller` to the output of `VehicleDynamic` must be within a certain time.
R5. The required environmental information should arrive to the controller within 40 ms. Input signals (`speed`, `signType`, `direct`, `gear` and `torque` ports) must be detected by `Controller` within a given time window, i.e., the tolerated maximum constraint is 40 ms.
R6. If the mode of AV switches to "emergency stop" due to a certain obstacle, it should not revert back to "automatic running" mode within a specific time

period. It is interpreted as a `Sporadic` constraint, i.e., the mode of AV is changed to `Stop` because of encountering an obstacle, it should not revert back to `Run` mode within 500 ms.
R7. The execution time interval from `Controller` to `VehicleDynamic` must be less than or equal to the sum of the worst case execution time interval of each f_p.
R8. While AV turns left, the "turning right" mode should not be activated. The events of turning left and right considered as exclusive and specified as an `Exclusion` constraint.

`Delay` constraint gives duration bounds (minimum and maximum) between two events *source* and *target*. This is specified using *lower, upper* values given as either `Execution` constraint (R3) or `End-to-End` constraint (R4). `Synchronization` constraint describes how tightly the occurrences of a group of events follow each other. All events must occur within a sliding window, specified by the *tolerance* attribute, i.e., the maximum time interval allowed between events (R5). `Periodic` constraint states that the period of successive occurrences of a single event must have a time interval (R1–R2). `Sporadic` constraint states that *events* can arrive at arbitrary points in time, but with defined minimum inter-arrival times between two consecutive occurrences (R6). `Comparison` constraint delimits that two consecutive occurrences of an event should have a minimum inter-arrival time (R7). `Exclusion` constraint refers that two events must not occur at the same time (R8).

Those timing constraints are formally specified (see as `R. IDs` in Fig. 1) using the subset of clock *relations* and *expressions* (see Sect. 2) in the context of WH. The timing constraints are then verified utilizing Uppaal-SMC and are described further in the following sections.

4 Probabilistic Extension of *Relation* in CCSL

To perform the formal specification and probabilistic verification of East-adl timing constraints (R1–R8 in Sect. 3.), Ccsl *relations* are augmented with probabilistic properties, called PrCcsl, based on WH [8]. More specifically, in order to describe the bound on the number of permitted timing constraint violations in WH, we extend Ccsl *relations* with a probabilistic parameter p, where p is the probability threshold. PrCcsl is satisfied if and only if the probability of *relation* constraint being satisfied is greater than or equal to p. As illustrated in Fig. 1, East-adl/Tadl2 timing constraints (R. IDs in Fig. 1) can be specified (Spec. R. IDs) using the PrCcsl *relations* and the conventional Ccsl *expressions*.

A *time system* is specified by a set of clocks and clock constraints. An execution of the time system is a `run` where the occurrences of events are clock ticks.

Definition 1 (`Run`). *A `run` R consists of a finite set of consecutive steps where a set of clocks tick at each step i. The set of clocks ticking at step i is denoted as $R(i)$, i.e., for all i, $0 \leqslant i \leqslant n$, $R(i) \in R$, where n is the number of steps of R.*

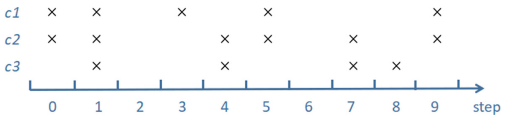


Fig. 2. Example of a `Run`

Figure 2 presents a `run` R consisting of 10 steps and three clocks $c1$, $c2$ and $c3$. The ticks of the three clocks along with steps are shown as "cross" symbols (x). For instance, $c1$, $c2$ and $c3$ tick at the first step, hence $R(1) = \{c1,\ c2,\ c3\}$.

The history of a clock c presents the number of times the clock c has ticked prior to the current step.

Definition 2 (`History`). *For $c \in C$, the* ***history*** *of c in a run R is a function: H_R^c: $\mathbb{N} \rightarrow \mathbb{N}$. For all instances of step i, $i \in \mathbb{N}$, $H_R^c(i)$ indicates the number of times the clock c has ticked prior to step i in run R, which is initialized as 0 at step 0. It is defined as:*

$$H_R^c(i) = \begin{cases} 0, & i = 0 \\ H_R^c(i-1), & c \notin R(i) \wedge i > 0 \\ H_R^c(i-1) + 1, & c \in R(i) \wedge i > 0 \end{cases}$$

Definition 3 (`PrCCSL`). *Let $c1$, $c2$ and R be two logical clocks and a* ***run****. The probabilistic extension of relation constraints, denoted $c1 \sim_p c2$, is satisfied if the following condition holds:*

$$R \vDash c1 \sim_p c2 \Longleftrightarrow Pr(c1 \sim c2) \geqslant p$$

where $\sim \in \{\subseteq, \equiv, \prec, \preceq, \#\}$, $Pr(c1 \sim c2)$ is the probability of the relation $c1 \sim c2$ being satisfied, and p is the probability threshold.

The five CCSL *relations*, `subclock`, `coincidence`, `exclusion`, `causality` and `precedence`, are considered and their probabilistic extensions are defined.

Definition 4 (`Probabilistic Subclock`). *Let $c1$, $c2$ and $\mathcal{M}$ be two logical clocks and a system model. Given k* ***runs*** *$= \{R_1, \ldots, R_k\}$, the probabilistic extension of* ***subclock*** *relation between $c1$ and $c2$, denoted $c1 \subseteq_p c2$, is satisfied if the following condition holds:*

$$\mathcal{M} \vDash c1 \subseteq_p c2 \Longleftrightarrow Pr[c1 \subseteq c2] \geqslant p$$

where $Pr[c1 \subseteq c2] = \frac{1}{k} \sum_{j=1}^{k} \{R_j \models c1 \subseteq c2\}$, $R_j \in \{R_1, \ldots, R_k\}$, *i.e., the ratio of* ***runs*** *that satisfies the* ***subclock*** *relation out of k* ***runs****.*

A `run` R_j satisfies the `subclock` relation between $c1$ and $c2$ "if $c1$ ticks, $c2$ must tick" holds at every step i in R_j, s.t., $(R_j \models c1 \subseteq c2) \Longleftrightarrow (\forall i\ 0 \leqslant i \leqslant n,\ c1 \in$

$R(i) \implies c2 \in R(i))$. "$R_j \models c1 \subseteq c2$" returns 1 if R_j satisfies $c1 \subseteq c2$, otherwise it returns 0.

`Coincidence` relation delimits that two clocks must always tick at the same step, i.e., if $c1$ and $c2$ are coincident, then $c1$ and $c2$ are subclocks of each other.

Definition 5 (`Probabilistic Coincidence`). *The probabilistic* ***coincidence*** *relation between $c1$ and $c2$, denoted $c1 \equiv_p c2$, is satisfied over $\mathcal{M}$ if the following condition holds:*

$$\mathcal{M} \vDash c1 \equiv_p c2 \iff Pr[c1 \equiv c2] \geqslant p$$

where $Pr[c1 \equiv c2] = \frac{1}{k} \sum_{j=1}^{k} \{R_j \models c1 \equiv c2\}$ *is determined by the number of* ***runs*** *satisfying the* ***coincidence*** *relation out of k* ***runs****.*

A `run`, R_j satisfies the `coincidence` relation on $c1$ and $c2$ if the assertion holds: $\forall i,\ 0 \leqslant i \leqslant n,\ (c1 \in R(i) \implies c2 \in R(i)) \wedge (c2 \in R(i) \implies c1 \in R(i))$. In other words, the satisfaction of `coincidence` relation is established when the two conditions "if $c1$ ticks, $c2$ must tick" and "if $c2$ ticks, $c1$ must tick" hold at every step.

The inverse of `coincidence` relation is `exclusion`, which specifies two clocks cannot tick at the same step.

Definition 6 (`Probabilistic Exclusion`). *For all k* ***runs*** *over $\mathcal{M}$, the probabilistic* ***exclusion*** *relation between $c1$ and $c2$, denoted $c1 \#_p c2$, is satisfied if the following condition holds:*

$$\mathcal{M} \vDash c1 \#_p c2 \iff Pr[c1 \# 2] \geqslant p$$

where $Pr[c1 \# c2] = \frac{1}{k} \sum_{j=1}^{k} \{R_j \models c1 \# c2\}$ *is the ratio of the* ***runs*** *satisfying the* ***exclusion*** *relation out of k* ***runs****.*

A `run`, R_j, satisfies the `exclusion` relation on $c1$ and $c2$ if $\forall i,\ 0 \leqslant i \leqslant n$, $(c1 \in R(i) \implies c2 \notin R(i)) \wedge (c2 \in R(i) \implies c1 \notin R(i))$, i.e., for every step, if $c1$ ticks, $c2$ must not tick and vice versa.

The probabilistic extension of `causality` and `precedence` relations are defined based on the history of clocks.

Definition 7 (`Probabilistic Causality`). *The probabilistic* ***causality*** *relation between $c1$ and $c2$ ($c1$ is the cause and $c2$ is the effect), denoted $c1 \preceq_p c2$, is satisfied if the following condition holds:*

$$\mathcal{M} \vDash c1 \preceq_p c2 \iff Pr[c1 \preceq c2] \geqslant p$$

where $Pr[c1 \preceq c2] = \frac{1}{k} \sum_{j=1}^{k} \{R_j \models c1 \preceq c2\}$*, i.e., the ratio of* ***runs*** *satisfying the* ***causality*** *relation among the total number of k* ***runs****.*

A `run` R_j satisfies the `causality` relation on $c1$ and $c2$ if the condition holds: $\forall i$, $0 \leqslant i \leqslant n$, $H_R^{c1}(i) \geqslant H_R^{c2}(i)$. A tick of $c1$ satisfies `causality` relation if $c2$ does not occur prior to $c1$, i.e., the history of $c2$ is less than or equal to the history of $c1$ at the current step i.

The strict `causality`, called `precedence`, constrains that one clock must always tick faster than the other.

Definition 8 (`Probabilistic Precedence`). *The probabilistic* ***`precedence`*** *relation between $c1$ and $c2$, denoted $c1 \prec_p c2$, is satisfied if the following condition holds:*

$$\mathcal{M} \vDash c1 \prec_p c2 \iff Pr[c1 \prec c2] \geqslant p$$

where $Pr[c1 \prec c2] = \frac{1}{k} \sum_{j=1}^{k} \{R_j \vDash c1 \prec c2\}$ *is determined by the number of* ***`runs`*** *satisfying the* ***`precedence`*** *relation out of the k* ***`runs`****.*

A `run` R_j satisfies the `precedence` relation if the condition (expressed as (1)∧(2)) holds: $\forall i$, $0 \leqslant i \leqslant n$,

$$\underbrace{(H_R^{c1}(i) \geqslant H_R^{c2}(i))}_{(1)} \wedge \underbrace{(H_R^{c2}(i) = H_R^{c1}(i)) \implies (c2 \notin R(i))}_{(2)}$$

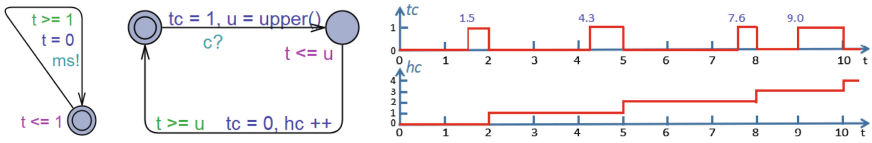
(1) The history of $c1$ is greater than or equal to the history of $c2$; (2) $c1$ and $c2$ must not be coincident, i.e., when the history of $c1$ and $c2$ are equal, $c2$ must not tick.

5 Translating CCSL and PrCCSL into UPPAAL-SMC

To formally verify the EAST-ADL timing constraints given in Sect. 3 using UPPAAL-SMC, we investigate how those constraints, specified in CCSL *expressions* and PrCCSL *relations*, can be translated into STA and probabilistic UPPAAL-SMC queries [12]. CCSL *expressions* construct new clocks and the *relations* between the new clocks are specified using PrCCSL. We first provide strategies that represent CCSL *expressions* as STA. We then present how the EAST-ADL timing constraints defined in PrCCSL can be translated into the corresponding STAs and UPPAAL-SMC queries based on the strategies.

5.1 Mapping CCSL to UPPAAL-SMC

We first describe how the universal clock (TimeUnit *ms*), tick and history of CCSL can be mapped to the corresponding STAs. Using the mapping, we then demonstrate that CCSL *expressions* can be modeled as STAs. The TimeUnit is implicitly represented as a single *step* of time progress in UPPAAL-SMC's *clock* [23]. The STA of TimeUnit (universal time defined as *ms*) consists of one location and one outgoing transition whereby the physical time and the duration



(a) ms (b) Tick and History (c) Simulation of Tick (tc) and History (hc)

Fig. 3. UPPAAL-SMC model of clock tick and history

of TimeUnit *ms* are represented by the *clock* variable t in Fig. 3(a). *clock* resets every time a transition is taken. The duration of TimeUnit is expressed by the invariant $t \leqslant 1$, and guard $t \geqslant 1$, i.e., a single step of the discrete time progress (tick) of universal time.

A clock c, considered as an event in UPPAAL-SMC, and its tick, i.e., an occurrence of the event, is represented by the synchronization channel $c!$. Since UPPAAL-SMC runs in chronometric semantics, in order to describe the discretized steps of `run`s (*R*s), we consider if c ticks in the time range of $[i, i+1)$ ($i+1$ is excluded), c ticks at step i. The STA of tick and history is shown in Fig. 3(b). *hc* is the history of c, and *tc* indicates whether c ticks at the current step. A function *upper*() rounds the time instant (real number) up to the nearest greater integer. When c ticks via $c?$ at the current time step, *tc* is set to 1 prior to the time of the next step ($t < u$). *hc* is then increased by 1 ($hc{+}{+}$) at the successive step (i.e., when $t = u$). For example, when c ticks at $time = 1.5$ (see Fig. 3(c)), *upper*() returns the value of 2 and *tc* becomes 1 during the time interval [1.5, 2), followed by *hc* being increased by 1 at $t = 2$.

Based on the mapping patterns of *ms*, tick and history, we present how `periodicOn`, `delayFor`, `infimum` and `supremum` *expressions* can be represented as UPPAAL-SMC models.

PeriodicOn: $c \triangleq$ `periodicOn` *ms* `period` q, where $\triangleq$ means "is defined as". `PeriodicOn` builds a new clock c based on *ms* and a *period* parameter q, i.e., c ticks at every q^{th} tick of *ms*. The STA of `periodicOn` is illustrated in Fig. 4(a). This STA initially stays in the *loop* location to detect q occurrences (ticks) of *ms*. The value x counts the number of *ms* ticks. When *ms* occurs ($ms?$), the STA takes the outgoing transition and increases x by 1. It "iterates" until *ms* ticks q times ($x == q$), then it activates the tick of c (via $c!$). At the successive step ($ms?$), it updates the history of c ($hc{+}{+}$) and sets $x = 1$. The STA then returns to *loop* and repeats the calculation. This `periodicOn` STA can be used for the translation of EAST-ADL `Periodic` timing constraint (R1 in Fig. 1) into its UPPAAL-SMC model.

DelayFor: $c \triangleq$ $c1$ `delayFor` d `on` $c2$. `delayFor` defines a new clock c based on $c1$ (*base* clock) and $c2$ (*reference* clock), i.e., each time $c1$ ticks, at the d^{th} tick of $c2$, c ticks (each tick of c corresponds to a tick of $c1$). Kang et al. [23] and Suryadevara et al. [33] presented translation rules of `delayFor` into UPPAAL models. However, their approaches are not applicable in the case after $c1$ ticks, and $c1$ ticks again before the d^{th} tick of $c2$ occurs. For example (see Fig. 2),

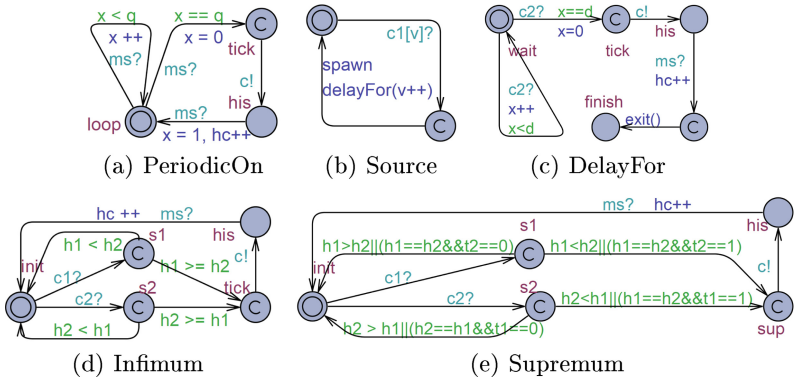


Fig. 4. STA of Ccsl *expressions*

assume that d is 3. After the 1^{st} tick of $c1$ (at step 0) happens, if $c1$ ticks again (at step 2) before the 3^{rd} tick of $c2$ occurs (at step 4), the 2^{nd} tick of $c1$ is discarded in their approaches. To alleviate the restriction, we utilize spawnable STA [12] as semantics denotation of `delayFor` expression and the STA of `delayFor` is shown in Fig. 4(c). As presented in Fig. 4(b), when the v^{th} tick of $c1$ occurs ($c1[v]$?), its `delayFor` STA is spawned by `source` STA. The spawned STA stays in the *wait* location until $c2$ ticks d times. When $c2$ ticks d times ($x == d$), it transits to the *tick* location and triggers c ($c!$). At the next step (ms?), the STA increases hc by 1 and moves to $finish$ location and then becomes inactive, i.e., calculation of the v^{th} tick of c is completed. This `delayFor` STA can be utilized to construct the Uppaal-SMC models of East-adl timing requirements R3 – R7 in Sect. 3.

Given two clocks $c1$ and $c2$, their `infimum` (resp. `supremum`) is informally defined as the slowest (resp. fastest) clock faster (resp. slower) than both $c1$ and $c2$. `infimum` and `supremum` are useful in order to group events occurring at the same time and decide which one occurs first and which one occurs last. The representative STAs for both *expressions* are utilized for the translation of East-adl `Synchronization` timing constraint (R5 in Sect. 3) into the Uppaal-SMC model.

Infimum creates a new clock c, which is the slowest clock faster than $c1$ and $c2$. The STA of `infimum` is illustrated in Fig. 4(d). When $c1$ ($c2$) ticks via $c1$? ($c2$?), the STA transits to the *s1* (*s2*) location and compares the history of the two clocks ($h1$ and $h2$) to check whether the current ticking clock $c1$ ($c2$) is faster than $c2$ ($c1$). If so, i.e., the condition "$h1 \geqslant h2$ ($h2 \geqslant h1$)" holds, the STA takes a transition to the *tick* location and activates the tick of c ($c!$). After updating the history (hc++), it returns to the *init* location and repeats the calculation.

Supremum builds a new clock c, which is the fastest clock slower than $c1$ and $c2$. It states that if $c1$ ticks at the current step and $c1$ is slower than $c2$, then c ticks. The STA of `supremum` is shown in Fig. 4(e). When $c1$ ($c2$) ticks via $c1$? ($c2$?), the STA transits to the *s1* (*s2*) location and compares the history of the two clocks and decides whether $c1$ ($c2$) is slower than $c2$ ($c1$). If $c1$ ($c2$) ticks

slower than $c2$ ($c1$), i.e., $h1 < h2$ ($h2 < h1$), or $c1$ and $c2$ tick at the same rate, i.e., "$h1 == h2$ && $t_2 == 1$ ($h1 == h2$ && $t_1 == 1$)" holds, the tick of c is triggered. The STA then updates the history of c and goes back to *init* and repeats the process.

5.2 Representation of PrCCSL in UPPAAL-SMC

In this section, the translation of EAST-ADL timing constraints specified in PrCCSL into STA and *Hypothesis Testing* query (refer to Sect. 2) is provided from the view point of the analysis engine UPPAAL-SMC.

Recall the definition of PrCCSL in Sect. 4. The probability of a *relation* being satisfied is interpreted as a ratio of runs that satisfies the *relation* among all runs. It is specified as *Hypothesis Testing* queries in UPPAAL-SMC, H_0: $\frac{m}{k} \geqslant P$ against H_1: $\frac{m}{k} < P$, where m is the number of runs satisfying the given *relation* out of all k runs. k is decided by strength parameters α (the probability of false positives, i.e., accepting H_1 when H_0 holds) and β (probability of false negatives, i.e., accepting H_0 when H_1 holds), respectively [10].

Based on the mapping patterns of tick and history in Sect. 5.1, the probabilistic extension of `exclusion`, `causality` and `precedence` relations are expressed as *Hypothesis Testing* queries straightforwardly.

Probabilistic Exclusion is employed to specify EAST-ADL `Exclusion` timing constraint, $turnLeft \ \#_p \ rightOn$ (Spec. R8 in Fig. 1). It states that the two events, *turnLeft* and *rightOn* (the vehicle is turning left and right), must be exclusive. The ticks of $turnLeft$ and $rightOn$ events are modeled using the STA in Fig. 3(b). Based on the definition of `probabilistic exclusion` (Sect. 4), R8 is expressed in *Hypothesis Testing* query: $Pr[bound]\ ([\,]((t_{turnLeft} \implies \neg\, t_{rightOn}) \wedge (t_{rightOn} \implies \neg\, t_{turnLeft}))) \geqslant P$, where $t_{turnLeft}$ and $t_{rightOn}$ indicate the ticks of $turnLeft$ and $rightOn$, respectively. *bound* is the time bound of simulation, in our setting $bound = 3000$.

Probabilistic Causality is used to specify EAST-ADL `Synchronization` timing constraint, $sup \preceq_p \{inf$ `delayFor` 40 `on` $ms\}$ (Spec. R5 in Fig. 1), where *sup* (inf) is the fastest (slowest) event slower (faster) than five input events, *speed*, *signType*, *direct*, *gear* and *torque*. Let `SUP` and `INF` denote the `supremum` and `infimum` operator, i.e., `SUP`($c1$, $c2$) (resp. `INF`($c1$, $c2$)) returns the `supremum` (resp. `infimum`) of clock $c1$ and $c2$. *sup* and *inf* can now be expressed with the nested operators (where $\triangleq$ means "is defined as"):

$$sup \triangleq \texttt{SUP}(speed, \texttt{SUP}(\texttt{SUP}(signType, direct), \texttt{SUP}(gear, torque)))$$

$$inf \triangleq \texttt{INF}(speed, \texttt{INF}(\texttt{INF}(signType, direct), \texttt{INF}(gear, torque)))$$

For the translation of *sup* (inf) into UPPAAL-SMC model, we employ the STA of `supremum` (resp. `infimum`) (Fig. 4(d) and (e)) for each `SUP` (`INF`) operator. A new clock *dinf* is generated by delaying *inf* for 40 ticks of *ms*: $dinf \triangleq \{inf$ `delayFor` 40 `on` $ms\}$. The UPPAAL-SMC model of *dinf* is achieved by adapting the spawnable *DelayFor* STA (Fig. 4). Based on the `probabilistic`

`causality` definition, R5 is interpreted as: $Pr[\leqslant bound]([\,]\, h_{sup} \geqslant h_{dinf}) \geqslant P$, where h_{sup} and h_{dinf} are the history of *sup* and *dinf* respectively. Similarly, `Execution` (R3) and `Comparison` (R7) timing constraints specified in `probabilistic causality` using `delayFor` can be translated into *Hypothesis Testing* queries. For further details, refer to the technical report [20].

Probabilistic Precedence is utilized to specify EAST-ADL `End-to-End` timing constraint (R4). It states that the time duration between the *source* event *signIn* (input signal on the *signType* port of `Controller`) and the *target* event *spOut* (output signal on the *speed* port of `VehicleDynamic`) must be within a time bound of [150, 250], and that is specified as UPPAAL-SMC queries (1) and (2):

$$\{signIn \texttt{ delayFor } 150 \texttt{ on } ms\} \prec_p spOut \tag{1}$$

$$spOut \prec_p \{signIn \texttt{ delayFor } 250 \texttt{ on } ms\} \tag{2}$$

Two clocks, *lower* and *upper*, are defined by delaying *signIn* for 150 and 250 ticks of *ms* respectively: $lower \triangleq \{signIn \texttt{ delayFor } 150 \texttt{ on } ms\}$, and $upper \triangleq \{signIn \texttt{ delayFor } 250 \texttt{ on } ms\}$. The corresponding UPPAAL-SMC models of *lower* and *upper* are constructed based on the `delayFor` STA (shown in Fig. 4). Finally, R4 specified in PrCCSL is expressed as UPPAAL-SMC queries (3) and (4), where $h_{\texttt{lower}}$, $h_{\texttt{upper}}$ and $h_{\texttt{spOut}}$ are the history of *lower*, *upper* and *spOut*. $t_{\texttt{spOut}}$ and $t_{\texttt{upper}}$ represent the tick of *upper* and *spOut* respectively:

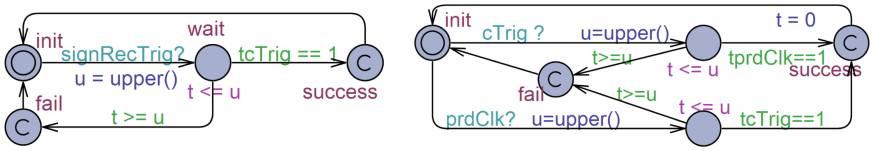
$$Pr[\leqslant bound]([\,]h_{\texttt{lower}} \geqslant h_{\texttt{spOut}} \wedge ((h_{\texttt{lower}} == h_{\texttt{spOut}}) \implies t_{\texttt{spOut}} == 0)) \geqslant P \tag{3}$$

$$Pr[\leqslant bound]([\,]h_{\texttt{spOut}} \geqslant h_{\texttt{upper}} \wedge ((h_{\texttt{spOut}} == h_{\texttt{upper}}) \implies t_{\texttt{upper}} == 0)) \geqslant P \tag{4}$$

Similarly, EAST-ADL `Sporadic` timing constraint (R6) specified in `probabilistic precedence` can be translated into *Hypothesis Testing* query [20].

In the case of properties specified in either `probabilistic subclock` or `probabilistic coincidence`, such properties cannot be directly expressed as UPPAAL-SMC queries. Therefore, we construct observer STA that capture the semantics of standard `subclock` and `coincidence` *relations*. The observer STA are composed to the system STA (namely a network STA, NSTA) in parallel. Then, the probabilistic analysis is performed over the NSTA which enables us to verify the EAST-ADL timing constraints specified in `probabilistic subclock` and `probabilistic coincidence` of the entire system using UPPAAL-SMC. Further details are given below.

Probabilistic Subclock is employed to specify EAST-ADL `Periodic` timing constraint, given as $signRecTrig \subseteq_p cTrig$ (Spec. R2 in Fig. 1). The standard `subclock` *relation* states that superclock must tick at the same step where subclock ticks. Its corresponding STA is shown in Fig. 5(a). When *signRevTrig* ticks (*signRecTrig*?), the STA transits to the *wait* location and detects the occurrence of *cTrig* until the time point of the subsequent step (u). If *cTrig* occurs prior to the next step ($tcTrig == 1$), the STA moves to the *success* location,



(a) Subclock (b) Coincidence

Fig. 5. Observer STA of `Subclock` and `Coincidence`

i.e., the `subclock` *relation* is satisfied at the current step. Otherwise, it transits to the $fail$ location. R2 specified in `probabilistic subclock` is expressed as: $Pr[bound]([\,]\neg\ Subclock.fail) \geqslant P$. UPPAAL-SMC analyzes if the $fail$ location is never reachable from the system NSTA, and whether the probability of R2 being satisfied is greater than or equal to P.

Probabilistic Coincidence is adapted to specify EAST-ADL `Periodic` timing constraint, given as $cTrig \equiv_p$ {`periodicOn` ms `period` 50} (Spec. R1 in Fig. 1). To express R1 in UPPAAL-SMC, first, a periodic clock $prdClk$ ticking every 50^{th} tick of ms is defined: $prdClk \triangleq$ `periodicOn` ms `period` 50. The corresponding UPPAAL-SMC model of $prdClk$ is generated based on the `periodicOn` STA shown in Fig. 4(a) by setting q as 50. Then, we check if $cTrig$ and $prdClk$ are coincident by employing the `coincidence` STA shown in Fig. 5(b). When $cTrig$ ($prdClk$) ticks via $cTrig?$ ($prdClk?$), the STA checks if the other clock, $prdClk$ ($cTrig$), ticks prior to the next step, i.e., whether $tprdClk == 1$ ($tcTrig == 1$) holds or not when $t \leqslant u$. The STA then transits to either the $success$ or $fail$ location based on the judgement. R1 specified in `probabilistic coincidence` is expressed as: $Pr[bound]([\,]\neg\ Coincidence.fail) \geqslant P$. UPPAAL-SMC analyzes if the probability of R1 being satisfied is greater than or equal to P.

6 Experiments: Verification and Validation

We have formally analyzed over 30 properties (associated with timing constraints) of the system including *deadlock freedom* [20]. A list of selected properties (Sect. 3) are verified using UPPAAL-SMC and the results are listed in Table.1. Five types of UPPAAL-SMC queries are employed to specify R1–R8, *Hypothesis Testing* (HT), *Probability Estimation* (PE), *Probability Comparison* (PC), *Expected Value* (EV) and *Simulations* (SI).

1. *Hypothesis Testing:* All properties are established as valid with 95% level of confidence; 2. *Probability Estimation:* The probability of each property being satisfied is computed and its approximate interval is given as [0.902, 1]; 3. *Expected Value:* The expected values of time durations of timing constraints (R1 – R7) are evaluated. For example, during the analysis of R1, the time interval between two consecutive triggerings of the `Camera` is evaluated as 50 and that validates R1. Furthermore, UPPAAL-SMC evaluates the expected maximum duration bound of `End-to-End` timing constraint by checking R4 and generates the frequency

Table 1. Verification results in UPPAAL-SMC

R	Q	Expression	Result	Time	Mem	CPU
R1	HT	$\Pr[\leqslant 3000]([\,]\ \neg Coin.fail) \geqslant 0.95$	Valid	48.7	32.7	31.3
	PE	$\Pr[\leqslant 3000]([\,]\ \neg Coin.fail)$	[0.902, 1]	12.6	35.6	29.8
	EV	$\mathrm{E}[\leqslant 3000;\ 500]([\,]\ max : cam.t)$	50 ± 0	83.3	33.3	31.7
	SI	simulate 500 $[\leqslant 3000](camtrig,\ p1trig)$	Valid	80.9	32.9	32.5
R2	HT	$\Pr[\leqslant 3000]([\,]\ \neg Sub.fail) \geqslant 0.95$	Valid	48.9	32.9	29.3
	PE	$\Pr[\leqslant 3000]([\,]\ \neg Sub.fail)$	[0.902, 1]	12.3	35.5	30.4
	EV	$\mathrm{E}[\leqslant 3000;\ 500]([\,]\ max : sf.t)$	200 ± 0	80.6	32.5	32.2
	SI	simulate 500 $[\leqslant 3000](strig,\ p2trig)$	Valid	85.5	33.1	32.3
R3	HT	$\Pr[\leqslant 3000]([\,]\ h_{\mathtt{SU}} \leqslant h_{\mathtt{S}}) \geqslant 0.95$	Valid	76.5	40.4	32.3
	PE	$\Pr[\leqslant 3000]([\,]\ h_{\mathtt{SU}} \leqslant h_{\mathtt{S}})$	[0.902, 1]	18.1	40.3	30.8
	HT	$\Pr[\leqslant 3000]([\,]\ h_{\mathtt{S}} \leqslant h_{\mathtt{SL}}) \geqslant 0.95$	Valid	77.6	37.7	31.7
	PE	$\Pr[\leqslant 3000]([\,]\ h_{\mathtt{S}} \leqslant h_{\mathtt{SL}})$	[0.902, 1]	16.5	40.0	31.5
	PC	$\Pr[\leqslant 3000]\ ([\,]\ SR.exec \implies (SR.t \geqslant 100 \wedge SR.t \leqslant 125)) \geqslant \Pr[\leqslant 3000]\ ([\,]\ SR.exec \implies (SR.t \geqslant 125 \wedge SR.t \leqslant 150))$	$\geqslant 1.1$	8.3	31.7	32.3
	EV	$\mathrm{E}[\leqslant 3000;\ 500]([\,]\ max : checkexe.t)$	147.2 ± 0.7	82.8	32.6	30.4
	SI	simulate 500 $[\leqslant 3000](h_{\mathtt{SU}}, h_{\mathtt{S}}, h_{\mathtt{SL}})$	Valid	86.9	33.2	33.4
R4	HT	$\Pr[\leqslant 3000]([\,]h_{\mathtt{lower}} \geqslant h_{\mathtt{spOut}} \wedge ((h_{\mathtt{lower}} == h_{\mathtt{spOut}}) \implies t_{\mathtt{spOut}} == 0)) \geqslant 0.95$	Valid	54.2	32.9	31.4
	PE	$\Pr[\leqslant 3000]([\,]h_{\mathtt{lower}} \geqslant h_{\mathtt{spOut}} \wedge ((h_{\mathtt{lower}} == h_{\mathtt{spOut}}) \implies \neg t_{\mathtt{spOut}}))$	[0.902, 1]	13.1	35.3	29.4
	HT	$\Pr[\leqslant 3000]([\,]h_{\mathtt{spOut}} \geqslant h_{\mathtt{upper}} \wedge ((h_{\mathtt{spOut}} == h_{\mathtt{upper}}) \implies t_{\mathtt{upper}} == 0)) \geqslant 0.95$	Valid	1.3 h	32.2	32.6
	PE	$\Pr[\leqslant 3000]([\,]h_{\mathtt{spOut}} \geqslant h_{\mathtt{upper}} \wedge ((h_{\mathtt{spOut}} == h_{\mathtt{upper}}) \implies \neg t_{\mathtt{upper}}))$	[0.902, 1]	19.8	34.1	32.0
	EV	$\mathrm{E}[\leqslant 3000;\ 500]([\,]\ max : checke2e.t)$	229.7 ± 0.9	83.3	32.5	30.6
	SI	simulate 500 $[\leqslant 3000](h_{\mathtt{CU}}, h_{\mathtt{VD}}, h_{\mathtt{CL}}, t_{\mathtt{CU}}, t_{\mathtt{VD}})$	Valid	89.8	32.9	30.2
R5	HT	$\Pr[\leqslant 3000]([\,]\ h_{dinf} \geqslant h_{sup}) \geqslant 0.95$	Valid	53.9	32.7	31.9
	PE	$\Pr[\leqslant 3000]([\,]\ h_{dinf} \geqslant h_{sup})$	[0.902, 1]	13.7	35.5	30.4
	EV	$\mathrm{E}[\leqslant 3000;\ 500]([\,]\ max : checksync.t)$	30.6 ± 0.21	72.4	32.6	31.6
	SI	simulate 500 $[\leqslant 3000](h_{dinf}, h_{sup})$	Valid	86.8	32.6	32.0
R6	HT	$\Pr[\leqslant 3000]([\,]\ hv \leqslant ho \wedge ((hv == ho) \implies t_{\mathtt{va}} == 0)) \geqslant 0.95$	Valid	3h	33.1	30.0
	PE	$\Pr[\leqslant 3000]([\,]\ hv \leqslant ho \wedge ((hv == ho) \implies t_{\mathtt{va}} == 0))$	[0.902, 1]	45.4	33.1	29.4
	EV	$\mathrm{E}[\leqslant 3000;\ 500]([\,]\ max : obs.t)$	667 ± 79	80.8	29.7	31.7
	SI	simulate 500 $[\leqslant 3000](hv, ho, v))$	Valid	88.6	29.5	31.0
R7	HT	$\Pr[\leqslant 3000]([\,]\ (ex_{\mathtt{con}} == wcet_{\mathtt{con}}\ \wedge\ ex_{\mathtt{vd}} == wcet_{\mathtt{vd}}) \implies (h_{\mathtt{cu}} \geqslant h_{\mathtt{com}})) \geqslant 0.95$	Valid	57.4	36.7	28.4
	PE	$\Pr[\leqslant 3000]([\,]\ (ex_{\mathtt{con}} == wcet_{\mathtt{con}}\ \wedge\ ex_{\mathtt{vd}} == wcet_{\mathtt{vd}}) \implies (h_{\mathtt{cu}} \geqslant h_{\mathtt{com}}))$	[0.902, 1]	14.7	35.5	26.7
	EV	$\mathrm{E}[\leqslant 3000;\ 500]([\,]\ max : control.t)$	146.7 ± 0.28	74.9	29.4	32.7
	EV	$\mathrm{E}[\leqslant 3000;\ 500]([\,]\ max : vd.t)$	96.6 ± 0.27	74.2	29.4	31.4
	SI	simulate 500 $[\leqslant 3000](h_{\mathtt{cu}}, h_{\mathtt{com}}))$	Valid	86.6	29.5	32.5
R8	HT	$\Pr[\leqslant 3000]([\,]\ \neg(t_{\mathtt{Right}} == 1\ \wedge\ t_{\mathtt{Left}} == 1)) \geqslant 0.95$	Valid	57.4	36.7	28.4
	PE	$\Pr[\leqslant 3000]([\,]\ \neg(t_{\mathtt{Right}} == 1\ \wedge\ t_{\mathtt{Left}} == 1))$	[0.902, 1]	14.7	35.5	26.7
	SI	simulate 500 $[\leqslant 3000](t_{\mathtt{Right}}, t_{\mathtt{Left}})$	Valid	85.5	29.6	32.6

histogram of the expected bound (see Fig. 7). It illustrates that the expected bound is always less than 250 ms and 90% of the duration is within the range of [207, 249]; 4. *Probability Comparison:* is applied to confirm that the probability of `SignRecognition` f_p completing its execution within [100, 125] ms is greater than the probability of completion within [125, 150] ms (R3). The query results in a comparison probability ratio greater than or equal to 1.1, i.e., the execution time of `SignRecognition` f_p is most likely less than 125 ms. 5. *Simulation*: The

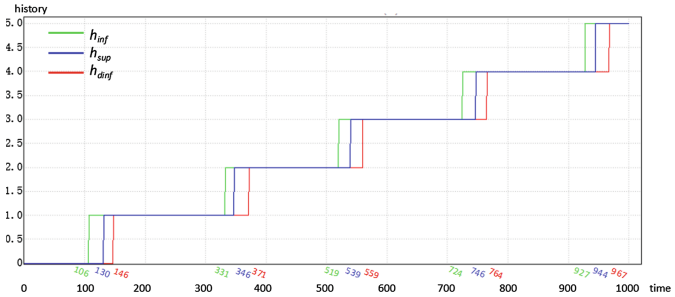


Fig. 6. Simulation result of R6. (Color figure online)

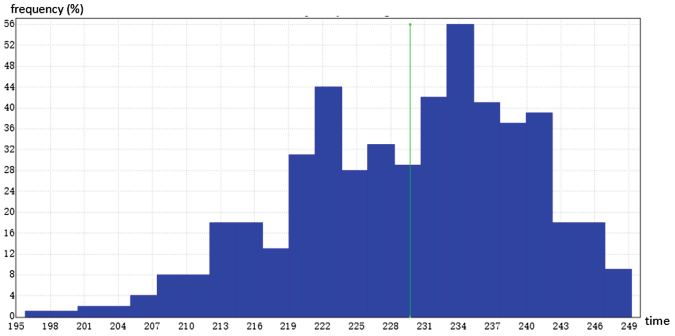


Fig. 7. Frequency histogram of `End-to-End` timing constraint (R4)

simulation result of `Synchronization` timing constraint (R5) is demonstrated in Fig. 6. h_{inf}, h_{sup} and h_{dinf} are history of inf, sup and $dinf$ respectively. Recall Spec. R5 (see Fig. 1), the `causality` *relation* between $dinf$ and sup is satisfied. As the simulation of R6 shows (Fig. 6), the rising edge of h_{sup} (in blue) always occurs prior to h_{dinf} (in red). It indicates that sup always runs faster than $dinf$, thus the `causality` *relation* is validated.

7 Related Work

In the context of EAST-ADL, efforts on the integration of EAST-ADL and formal techniques based on timing constraints were investigated in several works [15,17,24,31], which are however, limited to the executional aspects of system functions without addressing stochastic behaviors. Kang [23] and Suryadevara [33,34] defined the execution semantics of both the controller and the environment of industrial systems in CCSL which are also given as mapping to UPPAAL models amenable to model checking. In contrast to our current work, those approaches lack precise stochastic annotations specifying continuous dynamics in particular regarding different clock rates during execution. Ling [35] transformed a subset of CCSL constraints to PROMELA models to perform formal

verification using SPIN. Zhang [36] transformed CCSL into first order logics that are verifiable using SMT solver. However, their works are limited to functional properties, and no timing constraints are addressed. Though, Kang et al. [16,19] and Marinescu et al. [28] present both simulation and model checking approaches of SIMULINK and UPPAAL-SMC on EAST-ADL models, neither formal specification nor verification of extended EAST-ADL timing constraints with probability were conducted. Our approach is a first application on the integration of EAST-ADL and formal V&V techniques based on probabilistic extension of EAST-ADL/TADL2 constraints using PrCCSL and UPPAAL-SMC. An earlier study [18,21,22] defined a probabilistic extension of EAST-ADL timing constraints and presented model checking approaches on EAST-ADL models, which inspires our current work. Specifically, the techniques provided in this paper define new operators of CCSL with stochastic extensions (PrCCSL) and verify the extended EAST-ADL timing constraints of CPS (specified in PrCCSL) with statistical model checking. Du et al. [13] proposed the use of CCSL with probabilistic logical clocks to enable stochastic analysis of hybrid systems by limiting the possible solutions of clock ticks. Whereas, our work is based on the probabilistic extension of EAST-ADL timing constraints with a focus on probabilistic verification of the extended constraints, particularly, in the context of WH.

8 Conclusion

We present an approach to perform probabilistic verification on EAST-ADL timing constraints of automotive systems based on WH at the early design phase: 1. Probabilistic extension of CCSL, called PrCCSL, is defined and the EAST-ADL/TADL2 timing constraints with stochastic properties are specified in PrCCSL; 2. The semantics of the extended constraints in PrCCSL is translated into verifiable UPPAAL-SMC models for formal verification; 3. A set of mapping rules is proposed to facilitate guarantee of translation. Our approach is demonstrated on an autonomous traffic sign recognition vehicle (AV) case study. Although, we have shown that defining and translating a subset of CCSL with probabilistic extension into UPPAAL-SMC models is sufficient to verify EAST-ADL timing constraints, as ongoing work, advanced techniques covering a full set of CCSL constraints are further studied. Despite the fact that UPPAAL-SMC supports probabilistic analysis of the timing constraints of AV, the computational cost of verification in terms of time is rather expensive. Thus, we continue to investigate complexity-reducing design/mapping patterns for CPS to improve effectiveness and scalability of system design and verification.

Acknowledgment. This work is supported by the NSFC, EASY Project: 46000-41030005.

References

1. Automotive Open System Architecture. https://www.autosar.org/
2. UPPAAL-SMC. http://people.cs.aau.dk/~adavid/smc/
3. IEC 61508: Functional Safety of Electrical Electronic Programmable Electronic Safety Related Systems. International Organization for Standardization, Geneva (2010)
4. ISO 26262–6: Road Vehicles Functional Safety Part 6. Product Development at the Software Level. International Organization for Standardization, Geneva (2011)
5. MAENAD (2011). http://www.maenad.eu/
6. André, C.: Syntax and semantics of the clock constraint specification language (CCSL). Ph.D. thesis, INRIA (2009)
7. André, C., Mallet, F.: Clock constraints in UML/MARTE CCSL. HAL - INRIA (2008)
8. Bernat, G., Burns, A., Llamosi, A.: Weakly hard real-time systems. Trans. Comput. **50**(4), 308–321 (2001)
9. Blom, H., et al.: TIMMO-2-USE timing model, tools, algorithms, languages, methodology, use cases. Technical report, TIMMO-2-USE (2012)
10. Bulychev, P., et al.: UPPAAL-SMC: statistical model checking for priced timed automata. In: QAPL, pp. 1–16. EPTCS (2012)
11. David, A., et al.: Statistical model checking for stochastic hybrid systems. In: HSB, pp. 122–136. EPTCS (2012)
12. David, A., Larsen, K.G., Legay, A., Mikučionis, M., Poulsen, D.B.: UPPAAL-SMC tutorial. STTT **17**(4), 397–415 (2015)
13. Du, D., Huang, P., Jiang, K., Mallet, F., Yang, M.: MARTE/pCCSL: modeling and refining stochastic behaviors of CPSs with probabilistic logical clocks. In: Kouchnarenko, O., Khosravi, R. (eds.) FACS 2016. LNCS, vol. 10231, pp. 111–133. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57666-4_8
14. EAST-ADL Consortium: EAST-ADL domain model specification v2.1.9. Technical report, MAENAD European Project (2011)
15. Goknil, A., Suryadevara, J., Peraldi-Frati, M.-A., Mallet, F.: Analysis support for TADL2 timing constraints on EAST-ADL models. In: Drira, K. (ed.) ECSA 2013. LNCS, vol. 7957, pp. 89–105. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39031-9_8
16. Kang, E.Y., Chen, J., Ke, L., Chen, S.: Statistical analysis of energy-aware real-time automotive systems in EAST-ADL/Stateflow. In: ICIEA, pp. 1328–1333. IEEE (2016)
17. Kang, E.Y., Enoiu, E.P., Marinescu, R., Seceleanu, C., Schobbens, P.Y., Pettersson, P.: A methodology for formal analysis and verification of EAST-ADL models. Reliabil. Eng. Syst. Saf. **120**(12), 127–138 (2013)
18. Kang, E.Y., Huang, L., Mu, D.: Formal verification of energy and timed requirements for a cooperative automotive system. In: SAC, pp. 1492–1499. ACM (2018)
19. Kang, E.Y., Ke, L., Hua, M.Z., Wang, Y.X.: Verifying automotive systems in EAST-ADL/Stateflow using UPPAAL. In: APSEC, pp. 143–150. IEEE (2015)
20. Kang, E.Y., Mu, D., Huang, L.: Probabilistic analysis of weakly-hard real-time systems. Technical report, SYSU (2018). https://sites.google.com/site/kangeu/home/publications
21. Kang, E.Y., Mu, D., Huang, L., Lan, Q.: Model-based analysis of timing and energy constraints in an autonomous vehicle system. In: QRS, pp. 525–532. IEEE (2017)

22. Kang, E.Y., Mu, D., Huang, L., Lan, Q.: Verification and validation of a cyber-physical system in the automotive domain. In: QRS, pp. 326–333. IEEE (2017)
23. Kang, E.Y., Schobbens, P.Y.: Schedulability analysis support for automotive systems: from requirement to implementation. In: SAC, pp. 1080–1085. ACM (2014)
24. Kang, E.-Y., Schobbens, P.-Y., Pettersson, P.: Verifying functional behaviors of automotive products in EAST-ADL2 using UPPAAL-PORT. In: Flammini, F., Bologna, S., Vittorini, V. (eds.) SAFECOMP 2011. LNCS, vol. 6894, pp. 243–256. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-24270-0_18
25. Legay, A., Viswanathan, M.: Statistical model checking: challenges and perspectives. STTT **17**(4), 369–376 (2015)
26. Mallet, F., Peraldi-Frati, M.A., Andre, C.: MARTE CCSL to execute EAST-ADL timing requirements. In: ISORC, pp. 249–253. IEEE (2009)
27. Mallet, F., De Simone, R.: Correctness issues on MARTE/CCSL constraints. Sci. Comput. Program. **106**, 78–92 (2015)
28. Marinescu, R., Kaijser, H., Mikučionis, M., Seceleanu, C., Lönn, H., David, A.: Analyzing industrial architectural models by simulation and model-checking. In: Artho, C., Ölveczky, P.C. (eds.) FTSCS 2014. CCIS, vol. 476, pp. 189–205. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17581-2_13
29. Nicolau, G.B.: Specification and analysis of weakly hard real-time systems. Trans. Comput. 308–321 (1988)
30. Object Management Group: UML profile for MARTE: Modeling and analysis of real-time embedded systems (2015)
31. Qureshi, T.N., Chen, D.J., Persson, M., Törngren, M.: Towards the integration of UPPAAL for formal verification of EAST-ADL timing constraint specification. In: TiMoBD Workshop (2011)
32. Simulink and Stateflow. https://www.mathworks.com/products.html
33. Suryadevara, J.: Validating EAST-ADL timing constraints using UPPAAL. In: SEAA, pp. 268–275. IEEE (2013)
34. Suryadevara, J., Seceleanu, C., Mallet, F., Pettersson, P.: Verifying MARTE/CCSL mode behaviors using UPPAAL. In: Hierons, R.M., Merayo, M.G., Bravetti, M. (eds.) SEFM 2013. LNCS, vol. 8137, pp. 1–15. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40561-7_1
35. Yin, L., Mallet, F., Liu, J.: Verification of MARTE/CCSL time requirements in PROMELA/SPIN. In: ICECCS, pp. 65–74. IEEE (2011)
36. Zhang, M., Ying, Y.: Towards SMT-based LTL model checking of clock constraint specification language for real-time and embedded systems. ACM SIGPLAN Not. **52**(4), 61–70 (2017)