Probabilistic Analysis of Weakly-Hard Real-Time Systems

Eun-Young Kang\textsuperscript{1,2} and Dongrui Mu\textsuperscript{2} and Li Huang\textsuperscript{2}

\textsuperscript{1}University of Namur, Belgium
\textsuperscript{2}School of Data & Computer Science, Sun Yat-Sen University, China

eykang@fundp.ac.be
{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
## List of Figures

| Figure | Description                                                                 | Page |
|--------|-----------------------------------------------------------------------------|------|
| 3.1    | AV in EAST-ADL augmented with TADL2 constraints (R. IDs) specified in PrCCSL (Spec. R. IDs) | 6    |
| 4.1    | Example of a Run                                                            | 10   |
| 6.1    | UPPAAL-SMC model of clock tick and history                                  | 21   |
| 6.2    | STA of CCSL expressions                                                     | 22   |
| 6.3    | Observer STA of Subclock and Coincidence                                    | 26   |
| 7.1    | STAs of $f_p$s in Environment $f_t$                                         | 29   |
| 7.2    | Modeling system behaviors in UPPAAL-SMC                                     | 30   |
| 7.3    | Periodically triggered ports speed and obstacle                             | 31   |
| 7.4    | Internal behaviours of Controller in UPPAAL-SMC                             | 31   |
| 7.5    | Representation of substates of Normal state in UPPAAL-SMC                   | 32   |
| 7.6    | STAs utilized to verify R3                                                  | 33   |
| 8.1    | Simulation Result of R13                                                    | 36   |
| 8.2    | Frequency Histogram of End-to-End timing constraint (R17)                   | 37   |
| 8.3    | Performance analysis of verifying R5 with Expected Value and Simulation. The number of runs ranges from 100 to 500 with increment as 100. | 37   |
Chapter 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 features. To ensure the reliability of safety critical hybrid dynamic systems, statistical model checking (SMC) techniques have been proposed [11,12,24]. 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) [8,28]. 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
function prototypes \((f_p)\), and the \(f_p\)s 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 [31]. 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 [29]. 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 [26].

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 [22]. 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: Chapter 2 presents an overview of CCSL and UPPAAL-SMC. The AV is introduced as a running example in Chapter 3. Chapter 4 presents the formal definition of PrCCSL. The timing constraints that are applied on top of AV are specified using CCSL in Chapter 5. Chapter 6 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 behaviours of AV system and the stochastic behaviours of the environments are represented as a network of Stochastic Timed Automata presented in Chapter 7. The applicability of our method is demonstrated by performing verification on the AV case study in Chapter 8. Chapter 9 and Chapter 10 present related work and the conclusion.
Chapter 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) \citep{6,26} is a UML profile for modeling and analysis of real-time systems (MARTE) \citep{7,25}. 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*: 

\[ ms = \text{idealClock} \text{ discretizedBy} 0.001. \]

*ms* representing a periodic clock that ticks every 1 millisecond in this paper. A step is a tick of the universal clock. Hence the length of one step is 1 millisecond.

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, \( c_1, c_2 \in C \), *coincidence* relation \( (c_1 \equiv c_2) \) specifies that two clocks must tick simultaneously. *Precedence* relation \( (c_1 \prec c_2) \) delimits that \( c_1 \) runs faster than \( c_2 \), i.e., \( \forall k \in \mathbb{N}^+ \), where \( \mathbb{N}^+ \) is the set of positive natural numbers, the \( k^{th} \) tick of \( c_1 \) must occur prior to the \( k^{th} \) tick of \( c_2 \). *Causality* relation \( (c_1 \preceq c_2) \) represents a relaxed version of *precedence*, allowing the two clocks to tick at the same time. *Subclock* \( (c_1 \subseteq c_2) \) indicates the relation between two clocks, *superclock* \( (c_1) \) and *subclock* \( (c_2) \), s.t. each tick of the subclock must correspond to a tick of its superclock at the same step. *Exclusion* \( (c_1 \not\equiv c_2) \) 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 clock 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 $c_1$ and $c_2$. **Supremum**, denoted $\text{sup}$, is defined as the fastest clock that is slower than $c_1$ and $c_2$.

**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 $\text{bound}$ 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 $\phi$ being satisfied for a given STA model within the time bound: $\Pr[\text{bound}] \phi$.

2. **Hypothesis Testing** checks if the probability of $\phi$ being satisfied is larger than or equal to a certain probability $P_0$: $\Pr[\text{bound}] \phi \geq P_0$.

3. **Probability Comparison** compares the probabilities of two properties being satisfied in certain time bounds: $\Pr[\text{bound}_1] \phi_1 \geq \Pr[\text{bound}_2] \phi_2$.

4. **Expected Value** evaluates the minimal or maximal value of a clock or an integer value while UPPAAL-SMC checks the STA model: $E[\text{bound}; N](\min : \phi)$ or $E[\text{bound}; N](\max : \phi)$.

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 $\text{bound}$: simulate $N \leq \text{bound}\{\phi_1, ..., \phi_k\}$. 
Chapter 3

Running Example: Traffic Sign Recognition Vehicle

An autonomous vehicle (AV) [20,21] 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 \( f_p \)s in Fig. 3.1: System function type contains four \( f_p \)s, i.e., the Camera captures sign images and relays the images to SignRecognition periodically. SignRecognition 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 \( f_p \)s, i.e., the information of traffic signs, random obstacles, and speed changes caused by environmental influence described in TrafficSign, Obstacle, and Speed \( f_p \)s 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. 3.1. Furthermore, we extend EAST-ADL/TADL2 with an Exclusion timing constraint (R27 – R31) that integrates relevant concepts from the CCSL constraint, i.e., two events cannot occur simultaneously.

R1. The camera must capture an image every 50ms. In other words, a Periodic acquisition of Camera must be carried out every 50ms.
R2. The captured image must be recognized by an AV every 200ms, which can be interpreted as a Periodic constraint on SignRecognition \( f_p \).
R3. The obstacle will be detected by vehicle every 40ms, i.e., a Periodic timing constraint should be applied on the obstacle input port of Controller.
R4. The speed of the vehicle should be updated periodically with the period as 30ms, i.e., a Periodic timing constraint should be applied on the speed input port of Controller.
R5. The detected image should be computed within $[100, 150] \text{ms}$ in order to generate the desired sign type, the \texttt{SignRecognition} must complete its execution within $[100, 150] \text{ms}$.

R6. After the \texttt{Camera} is triggered, the captured image should be sent out from \texttt{Camera} within 20 – 30ms, i.e., the execution time of \texttt{Camera} should be between 20 and 30ms.

R7. After an obstacle is detected, the \texttt{Controller} should send out a request to brake the vehicle within 100 – 150ms, i.e., the execution time for \texttt{Controller} should be in the range $[100, 150] \text{ms}$.

R8. After the command/request from controller is arrived at \texttt{VehicleDynamic}, the speed should be updated within 50 – 100ms. That is, the \texttt{Execution} timing constraint applied on \texttt{VehicleDynamic} is 50 – 100ms.

R9. If the mode of AV switches to “emergency stop” due to the certain obstacle, it should not revert back to “automatic running” mode within a specific time period. That is interpreted as a \texttt{Sporadic} constraint, i.e., the mode of AV is changed to “stop” because of the encounter of obstacle, it should not revert back to “run” mode within 500ms.

R10. If the mode of AV switches to “emergency stop” due to the certain obstacle, it should not revert back to “accelerate” mode within a specific time period. That is interpreted as a \texttt{Sporadic} constraint, i.e., the mode of AV is changed to “stop”
because of the encounter of obstacle, it should not revert back to “accelerate” mode within 500ms.
R11. If the mode of AV switches to “emergency stop” due to the certain obstacle, it should not revert back to “turn left” mode within a specific time period. That is interpreted as a Sporadic constraint, i.e., the mode of AV is changed to “stop” because of the encounter of obstacle, it should not revert back to “turn left” mode within 500ms.
R12. If the mode of AV switches to “emergency stop” due to the certain obstacle, it should not revert back to “turn right” mode within a specific time period. That is interpreted as a Sporadic constraint, i.e., the mode of AV is changed to “stop” because of the encounter of obstacle, it should not revert back to “turn right” mode within 500ms.
R13. The required environmental information should arrive to the controller within 40ms. That is 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 40ms.
R14. After the execution of Controller is finished, all the requests of controller should be updated within 30ms. That is output signals (on reqTorq, reqDirect, reqGear, reqBrake ports) must be sent within a given time window, i.e., the tolerated maximum constraint is 30ms.
R15. The requests from the controller should be arrived to VehicleDynamic within 30ms. That is input signals (reqTorq, reqDirect, reqGear, reqBrake) must be detected by VehicleDynamic within a given time window, i.e., the tolerated maximum constraint is 30ms.
R16. After execution of VehicleDynamic is finished, the information of vehicle should be updated within 40ms, i.e., the Synchronization applied on the output ports (speed, direct, gear, torque) is 40ms.
R17. 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 measured from the input arrival of Controller to the instant at which the corresponding output is sent out from VehicleDynamic must be within [150, 250]ms.
R18. After the camera is triggered to capture the image, the computation of the traffic sign should be finished within [120, 180]ms, i.e., the End-to-End timing constraint applied on Camera and SignRecognition should be between 120ms and 180ms.
R19. The time interval measured from the instant at which the camera captures an image of traffic sign, to the instant at which the status of AV (i.e., speed, direction) is updated, should be within [270, 430]ms. That is, End-to-End timing constraint applied on Camera and VehicleDynamic should be between 270 and 430ms.
R20. When a left turn sign is recognized, the vehicle should turn towards left within 500ms, which can be interpreted as an End-to-End timing constraint applied on the event DetectLeftSign and StartTurnLeft.

R21. When a right turn sign is recognized, the vehicle should turn towards right within 500ms, which can be interpreted as an End-to-End timing constraint applied on the event DetectRightSign and StartTurnRight.

R22. When a stop sign is recognized, the vehicle should start to brake within 200ms, which can be interpreted as an End-to-End timing constraint applied on the event DetectStopSign and StartBrake.

R23. When a stop sign is recognized, the vehicle should be stop completely within 3000ms, which can be interpreted as an End-to-End timing constraint applied on the event DetectStopSign and Stop.

R24. The execution time interval from Controller to VehicleDynamic should be less than or equal to the sum of the worst case execution time interval of each $f_p$.

R25. The execution time interval from Camera to SignRecognition should be less than or equal to the sum of the worst case execution time interval of each $f_p$.

R26. The execution time interval from Camera to VehicleDynamic should be less than or equal to the sum of the worst case execution time interval of each $f_p$.

R27. 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.

R28. While AV is braking, the “accelerate” mode should not be activated. The events of braking and accelerating are considered as exclusive and specified as an Exclusion constraint.

R29. When AV is in the emergency mode because of the obstacle occurrence, “turn left” mode must not be activated, i.e., the events of handling emergency and turning left are exclusive and specified as a Exclusion constraint.

R30. When AV is in the emergency mode because of the encounter of an obstacle, “turn right” mode must not be activated, i.e., the events of handling emergency and turning right are exclusive and specified as an Exclusion constraint.

R31. When AV is in the emergency mode because of the encounter of an obstacle, “accelerate” mode must not be activated, i.e., the events of handling emergency and accelerating are 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 (R5 – R8) or End-to-End constraint (R17 – R23). Synchronization constraint (R13 – R16) 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. Periodic constraint states that the period of successive
occurrences of a single event must have a time interval (R1 – R4). **Sporadic** constraint states that *events* can arrive at arbitrary points in time, but with defined minimum inter-arrival times between two consecutive occurrences (R9 – R12). **Comparison** constraint delimits that two consecutive occurrences of an event should have a minimum inter-arrival time (R24 – R26). **Exclusion** constraint refers that two events must not occur at the same time (R27 – R31). Those timing constraints are formally specified (see as R. IDs in Fig. 3.1) using the subset of clock *relations* and *expressions* (see Chapter 2) in the context of WH. The timing constraints are then verified utilizing UPPAAL-SMC and are described further in the following chapters.
Chapter 4

Probabilistic Extension of Relation in CCSL

To perform the formal specification and probabilistic verification of EAST-ADL timing constraints (R1 – R31 in Sec 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. 3.1, EAST-ADL/TADL2 timing constraints (R. IDs in Fig. 3.1) 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 $i$, $0 \leq i \leq n$, $R(i) \in R$, where $n$ is the number of steps of $R$.

Fig. 4.1 presents a run $R$ consisting of 10 steps and three clocks $c_1$, $c_2$ and $c_3$. The ticks of the three clocks along with steps are shown as “cross” symbols (x). For instance, $c_1$, $c_2$ and $c_3$ tick at the first step, hence $R(1) = \{c_1, c_2, c_3\}$.

![Figure 4.1: Example of a Run](image)

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:
\( H^c_R : \mathbb{N} \to \mathbb{N} \). For all instances of step \( i, i \in \mathbb{N} \), \( H^c_R(i) \) 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:

\[
H^c_R(i) = \begin{cases} 
0, & i = 0 \\
H^c_R(i - 1), & c \notin R(i) \land i > 0 \\
H^c_R(i - 1) + 1, & c \in R(i) \land i > 0
\end{cases}
\]

**Definition 3 (PrCCSL)** Let \( c_1, c_2 \) and \( R \) be two logical clocks and a run. The probabilistic extension of relation constraints, denoted \( c_1 \sim_p c_2 \), is satisfied if the following condition holds:

\[
R \models c_1 \sim_p c_2 \iff Pr(c_1 \sim c_2) \geq p
\]

where \( \sim \in \{\subseteq, \equiv, \prec, \preceq, \#\} \), \( Pr(c_1 \sim c_2) \) is the probability of the relation \( c_1 \sim c_2 \) 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 \( c_1, c_2 \) and \( M \) be two logical clocks and a system model. Given \( k \) runs \( = \{R_1, \ldots, R_k\} \), the probabilistic extension of subclock relation between \( c_1 \) and \( c_2 \), denoted \( c_1 \subseteq_p c_2 \), is satisfied if the following condition holds:

\[
M \models c_1 \subseteq_p c_2 \iff Pr[c_1 \subseteq c_2] \geq p
\]

where \( Pr[c_1 \subseteq c_2] = \frac{1}{k} \sum_{j=1}^{k} \{R_j \models c_1 \subseteq c_2\} \), \( R_j \in \{R_1, \ldots, R_k\} \), i.e., the ratio of runs that satisfies the subclock relation out of \( k \) runs.

A run \( R_j \) satisfies the subclock relation between \( c_1 \) and \( c_2 \) “if \( c_1 \) ticks, \( c_2 \) must tick” holds at every step \( i \) in \( R_j \), s.t., \( (R_j \models c_1 \subseteq c_2) \iff (\forall i \ 0 \leq i \leq n, \ c_1 \in R(i) \implies c_2 \in R(i)) \). “\( R_j \models c_1 \subseteq c_2 \)” returns 1 if \( R_j \) satisfies \( c_1 \subseteq c_2 \), otherwise it returns 0.

Coincidence relation delimits that two clocks must always tick at the same step, i.e, if \( c_1 \) and \( c_2 \) are coincident, then \( c_1 \) and \( c_2 \) are subclocks of each other.

**Definition 5 (Probabilistic Coincidence)** The probabilistic coincidence relation between \( c_1 \) and \( c_2 \), denoted \( c_1 \equiv_p c_2 \), is satisfied over \( M \) if the following condition holds:

\[
M \models c_1 \equiv_p c_2 \iff Pr[c_1 \equiv c_2] \geq p
\]

where \( Pr[c_1 \equiv c_2] = \frac{1}{k} \sum_{j=1}^{k} \{R_j \models c_1 \equiv c_2\} \) is determined by the number of runs satisfying the coincidence relation out of \( k \) runs.
A run, $R_j$, satisfies the coincidence relation on $c_1$ and $c_2$ if the assertion holds: \( \forall i, 0 \leq i \leq n, (c_1 \in R(i) \implies c_2 \in R(i)) \land (c_2 \in R(i) \implies c_1 \in R(i)) \). In other words, the satisfaction of coincidence relation is established when the two conditions “if $c_1$ ticks, $c_2$ must tick” and “if $c_2$ ticks, $c_1$ 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 $M$, the probabilistic exclusion relation between $c_1$ and $c_2$, denoted $c_1 \#_p c_2$, is satisfied if the following condition holds:

\[
M \models c_1 \#_p c_2 \iff \Pr[c_1 \# 2] \geq p
\]

where $\Pr[c_1 \# c_2] = \frac{1}{k} \sum_{j=1}^{k} \{R_j \models c_1 \# c_2\}$ is the ratio of the runs satisfying the exclusion relation out of $k$ runs.

A run, $R_j$, satisfies the exclusion relation on $c_1$ and $c_2$ if $\forall i, 0 \leq i \leq n, (c_1 \in R(i) \implies c_2 \notin R(i)) \land (c_2 \in R(i) \implies c_1 \notin R(i))$, i.e., for every step, if $c_1$ ticks, $c_2$ 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 $c_1$ and $c_2$ ($c_1$ is the cause and $c_2$ is the effect), denoted $c_1 \preceq_p c_2$, is satisfied if the following condition holds:

\[
M \models c_1 \preceq_p c_2 \iff \Pr[c_1 \preceq c_2] \geq p
\]

where $\Pr[c_1 \preceq c_2] = \frac{1}{k} \sum_{j=1}^{k} \{R_j \models c_1 \preceq c_2\}$, 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 $c_1$ and $c_2$ if the condition holds: $\forall i, 0 \leq i \leq n, H^{c_1}_R(i) \geq H^{c_2}_R(i)$. A tick of $c_1$ satisfies causality relation if $c_2$ does not occur prior to $c_1$, i.e., the history of $c_2$ is less than or equal to the history of $c_1$ 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 $c_1$ and $c_2$, denoted $c_1 \prec_p c_2$, is satisfied if the following condition holds:

\[
M \models c_1 \prec_p c_2 \iff \Pr[c_1 \prec c_2] \geq p
\]
where \( Pr[c_1 \prec c_2] = \frac{1}{k} \sum_{j=1}^{k} \{ R_j \models c_1 \prec c_2 \} \) 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) \( \land \) (2)) holds: \( \forall i, 0 \leq i \leq n, \)

\[
\underbrace{(H_{R}^{c_1}(i) \geq H_{R}^{c_2}(i))}_{(1)} \land \underbrace{(H_{R}^{c_2}(i) = H_{R}^{c_1}(i))}_{(2)} \implies (c_2 \notin R(i))
\]

(1) The history of \( c_1 \) is greater than or equal to the history of \( c_2 \); (2) \( c_1 \) and \( c_2 \) must not be coincident, i.e., when the history of \( c_1 \) and \( c_2 \) are equal, \( c_2 \) must not tick.
Chapter 5

Specification of Timing Constraints in PrCCSL

To describe the property that a timing constraint is satisfied with the probability greater than or equal to a given threshold, CCSL and its extension PrCCSL are employed to capture the semantics of probabilistic timing constraints in the context of WH. Below, we show the CCSL/PrCCSL specification of EAST-ADL timing constraints, including Execution, Periodic, End-to-End, Sporadic, Synchronization, Exclusion and Comparison timing constraints. In the system, events are represented as clocks with identical names. The ticks of clocks correspond to the occurrences of the events.

**Periodic** timing constraints (R1 – R4) can be specified using `periodicOn` expression and probabilistic coincident relation. R1 states that the camera must be triggered periodically with a period 50ms. We first construct a periodic clock `prd_50` which ticks after every 50 ticks of `ms` (the universal clock). Then the property that the periodic timing constraint is satisfied with probability no less than the threshold $p$ can be interpreted as the probabilistic coincidence relation between `cmrTrig` (the event that Camera $f_p$ being triggered) and `prd_50`. The corresponding specification is given below, where $\triangleq$ means “is defined as”:

\[
prd_50 \triangleq \text{periodicOn ms period 50} \tag{5.1}
\]

\[
cmrTrig \equiv_p prd_50 \tag{5.2}
\]

By combining (1) and (2), we can obtain the the specification of R1:

\[
cmrTrig \equiv_p \{\text{periodicOn ms period 50}\} \tag{5.3}
\]

In similar, the CCSL/PrCCSL specification of R2 – R4 can be derived:

\[
\text{R2 : signTrig} \equiv_p \{\text{periodicOn ms period 200}\} \tag{5.4}
\]

\[
\text{R3 : obsDetect} \equiv_p \{\text{periodicOn ms period 40}\} \tag{5.5}
\]

\[
\text{R4 : spUpdate} \equiv_p \{\text{periodicOn ms period 30}\} \tag{5.6}
\]
where \textit{signTrig} is the event/clock that \textit{SignRecognition} \( f_p \) is triggered, \textit{obsDetect} represents the event that the object detection is activated by the vehicle and \textit{spUpdate} denotes the event that the speed is updated (i.e., received by \textit{Controller}) from the environment.

Since the \textit{period} attribute of the \textit{Periodic} timing constraint \( R_2 \) is 200ms, which is an integral multiple of the \textit{period} of \( R_1 \), \( R_2 \) can be interpreted as a \textit{subclock} relation, i.e., the event \textit{signTrig} should be a \textit{subclock} of \textit{cmrTrig}. The specification is given below:

\[
\text{signTrig} \subseteq_p \text{cmrTrig}
\]  

\textbf{Execution} timing constraints (\( R_5 \) – \( R_8 \)) can be specified using \textit{delayFor expression} and \textit{probabilistic causality relation}. To specify \( R_5 \), which states that the \textit{SignRecognition} \( f_p \) must finish execution within \([100, 150]|ms\), i.e., the interval measured from the input event of the \( f_p \) (i.e., the event that the image is received by the \( f_p \), denoted \textit{imIn}) to the output event of the \( f_p \) (denoted \textit{imIn}) must have a minimum value 100 and a maximum value 150. We divide this property into two subproperties: \( R_5(1) \) The time duration between \textit{imIn} and \textit{signOut} should be greater than 100ms. \( R_5(2) \) The time duration between \textit{imIn} and \textit{signOut} should be less than 150ms. To specify property \( R_5(1) \), we first construct a new clock \textit{imIn\_dly100} by delaying \textit{imIn} for 100 ticks on \textit{ms}. Afterwards, the property that \( R_5(2) \) is satisfied with a probability greater than or equal to \( p \) relies on whether the \textit{probabilistic causality} relation between \textit{imIn\_dly100} and \textit{signOut} is valid. The specification of \( R_5(1) \) is given below:

\[
\text{imIn\_dly100} \triangleq \text{imIn} \text{ delayFor 100 on ms}
\]  

\[
\text{imIn\_dly100} \preceq_p \text{signOut}
\]  

By combining (7) and (8), we can obtain the the specification of \( R_5(1) \):

\[
\{\text{imIn delayFor 100 on ms}\} \preceq_p \text{signOut}
\]  

Similarly, to specify property \( R_5(2) \), a new clock \textit{imIn\_dly150} is generated by delaying \textit{imIn} for 150 ticks on \textit{ms}. Afterwards, the property that \( R_5(2) \) is satisfied with a probability greater than or equal to \( p \) relies on whether the \textit{probabilistic causality relation} is satisfied. The specification is illustrated as follows:

\[
\text{imIn\_dly150} \triangleq \text{imIn} \text{ delayFor 150 on ms}
\]  

\[
\text{signOut} \preceq_p \text{mIn\_dly150}
\]  

By combining (10) and (11), we can obtain the the specification of \( R_5(2) \):

\[
\text{signOut} \preceq_p \{\text{imIn delayFor 150 on ms}\}
\]
Analogously, the CCSL/PrCCSL specification of R6 – R8 can be derived:

\[
\begin{align*}
R6 &: \{\text{cmrTrig delayFor 20 on ms}\} \preceq_p \text{cmrOut} \\
    &\quad \text{cmrOut} \preceq_p \{\text{cmrTrig delayFor 30 on ms}\} \\
R7 &: \{\text{ctrlIn delayFor 100 on ms}\} \preceq_p \text{ctrlOut} \\
    &\quad \text{ctrlOut} \preceq_p \{\text{ctrlIn delayFor 150 on ms}\} \\
R8 &: \{\text{vdIn delayFor 50 on ms}\} \preceq_p \text{vdOut} \\
    &\quad \text{vdOut} \preceq_p \{\text{vdIn delayFor 100 on ms}\}
\end{align*}
\]

where \text{cmrTrig} is the event that the \text{Camera} \_p being triggered, \text{cmrOut} represents the event that the captured image is sent out. \text{ctrlIn} (\text{ctrlOut}) represents the input (resp. output) event of \text{Controller} \_p. \text{vdIn} (\text{vdOut}) represents the input (resp. output) event of \text{VehicleDynamic} \_p.

\textbf{Sporadic} timing constraints (R9 – R12) can be specified using \text{delayFor expression} and \text{probabilistic precedence relation}. R9 states that there should be a minimum delay between the event \text{veRun} (the event that the vehicle is in the “run” mode) and the event \text{obstc} (the event that the vehicle detects an obstacle), which is specified as 500ms. To specify R9, we first build a new clock \text{obstc\_dly500} by delaying \text{obstc} for 500 ticks of ms. We then check the \text{probabilistic precedence} relation between \text{obstc\_dly500} and \text{veRun}:

\[
\begin{align*}
\text{obstc\_dly500} &\triangleq \text{obstc delayFor 500 on ms} \\
\text{obstc\_dly500} &\preceq_p \text{veRun}
\end{align*}
\]

By combining (16) and (17), we can obtain the specification of R9:

\[
\{\text{obstc delayFor 500 on ms}\} \preceq_p \text{veRun}
\]

Analogously, the CCSL/PrCCSL specification of R10 – R12 can be derived:

\[
\begin{align*}
R10 &: \{\text{obstc delayFor 500 on ms}\} \preceq_p \text{veAcc} \\
R11 &: \{\text{obstc delayFor 500 on ms}\} \preceq_p \text{tLeft} \\
R12 &: \{\text{obstc delayFor 500 on ms}\} \preceq_p \text{tRight}
\end{align*}
\]

where \text{veAcc} is the event/clock that the vehicle is accelerating. \text{tLeft} and \text{tRight} represent the event that the vehicle transits from the “emergency stop” mode to “turn left” and “turn right” mode respectively.

\textbf{Synchronization} timing constraints (R13 – R16) can be specified using \text{infimum} and \text{supremum expression}, together with \text{probabilistic precedence relation}. R13 states that the five input events must be detected by \text{Controller} within the maximum tolerated time, given as 40ms. The synchronization timing constraint
can be interpreted as: the time interval between the earliest/fastest and the latest/slowest event among the five input events, i.e., speed, signType, direct, gear and torque, must not exceed 40ms. To specify the constraints, infimum is utilized to express the fastest event (denoted \(inf_{ctrlIn}\)) while supremum is utilized to specify the slowest event \(sup_{ctrlIn}\). \(sup_{ctrlIn}\) and \(inf_{ctrlIn}\) are defined as:

\[
\begin{align*}
sup_{ctrl} & \triangleq \text{Sup}(\text{Sup}(\text{speed, signType}), \text{Sup}(\text{Sup}(\text{direct, gear}, \text{torque}))) \\
inf_{ctrl} & \triangleq \text{Inf}(\text{Inf}(\text{speed, signType}), \text{Inf}(\text{Inf}(\text{direct, gear}, \text{torque})))
\end{align*}
\]  

(5.23)  

(5.24)

where \(\text{Inf}(c_1, c_2)\) (resp. \(\text{Sup}(c_1, c_2)\)) is the infimum (resp. supremum) operator returns the slowest clock faster than \(c_1\) and \(c_2\). Afterwards, we construct a new clock \(inf_{ctrlIn\_dly40}\) that is the \(inf_{ctrlIn}\) delayed for 40 ticks of \(ms\), which is defined as:

\[
inf_{ctrlIn\_dly40} \triangleq \text{inf}_{ctrl} \text{ delayFor} 40 \text{ on ms}
\]  

(5.25)

Therefore, the synchronization constraint R13 can be represented as the probabilistic causality relation between \(sup_{ctrlIn}\) and \(inf_{ctrlIn\_dly40}\), given as the CCSL/PrCCSL expression below:

\[
\begin{align*}
sup_{ctrlIn} & \leq_p \text{inf}_{ctrlIn\_dly40}
\end{align*}
\]  

(5.26)

In similar, the CCSL/PrCCSL specification of R14 – R16 can be derived. For R14, we first construct the clocks that represent the fastest and slowest output event/clock among the four output events of Controller \(f_p\), i.e., reqTorq, reqDirect, reqGear and reqBrake. Then the property that the synchronization constraint is satisfied with a probability greater than or equal to \(p\) can be interpreted as a probabilistic causality relation:

\[
\begin{align*}
\text{R14 : } sup_{ctrlOut} & \triangleq \text{Sup}(\text{Sup}(\text{reqTorq, reqDirect}), \text{Sup}(\text{reqGear, reqBrake})) \\
inf_{ctrlOut} & \triangleq \text{Inf}(\text{Inf}(\text{reqTorq, reqDirect}), \text{Inf}(\text{reqGear, reqBrake})) \\
sup_{ctrlOut} & \leq_p \{\text{inf}_{ctrlOut} \text{ delayFor} 30 \text{ on ms}\}
\end{align*}
\]  

(5.28)

For R15, we first construct the fastest and slowest input event/clock among the four input events of VehicleDynamic, i.e., reqTorq, reqDirect, reqGear and reqBrake. Then the property that the synchronization constraint is satisfied with a probability greater than or equal to \(p\) can be interpreted as a probabilistic causality relation:

\[
\begin{align*}
\text{R15 : } sup_{vdIn} & \triangleq \text{Sup}(\text{Sup}(\text{reqTorq, reqDirect}), \text{Sup}(\text{reqGear, reqBrake})) \\
inf_{vdIn} & \triangleq \text{Inf}(\text{Inf}(\text{reqTorq, reqDirect}), \text{Inf}(\text{reqGear, reqBrake})) \\
sup_{vdIn} & \leq_p \{\text{inf}_{vdIn} \text{ delayFor} 40 \text{ on ms}\}
\end{align*}
\]  

(5.29)
For R16, we first construct the fastest and slowest output event/clock among the four output events of \textit{VehicleDynamic}, i.e., \textit{speed}, \textit{direct}, \textit{torque} and \textit{gear}. Then the property that the synchronization constraint is satisfied with a probability greater than or equal to \(p\) can be interpreted as a \textbf{probabilistic causality relation}:

\begin{align*}
\textbf{R16}: \quad \sup_{\text{vdOut}} & \triangleq \text{Sup}(\text{Sup}(\text{speed}, \text{direct}), \text{Sup}(\text{gear}, \text{torque})) \\
\inf_{\text{vdOut}} & \triangleq \text{Inf}(\text{Inf}(\text{speed}, \text{direct}), \text{Inf}(\text{gear}, \text{torque})) \\
\sup_{\text{vdOut}} & \preceq_p \{\inf_{\text{vdOut}} \text{ delayFor 40 on ms}\} 
\end{align*}

\textbf{End-to-End} timing constraints (R17 – R23) can be specified using \texttt{delayFor} expression and \textbf{probabilistic precedence relation}. To specify R17, which limits that the time duration measured from the instant of the occurrence of the event that \textit{Controller} \(_f\) \textit{receive} the traffic sign type information (denoted as \texttt{signIn}), to the occurrence of event that the \textit{speed} is sent out from the output port of \textit{VehicleDynamic} \(_f\) (denoted as \texttt{spOut}) should be between 150 and 250ms. We divide this property into two subproperties: R17(1). The time duration between \texttt{signIn} and \texttt{spOut} should be more than 150ms. R17(2). The time duration between \texttt{signIn} and \texttt{spOut} should be less than 250ms. To specify property R17(1), we first construct a new clock \texttt{signIn\_dly150} by delaying \texttt{signIn} for 150ticks on ms. To check whether R17(1) is satisfied within a probability threshold \(p\) is to verify whether the \textbf{probabilistic precedence} between \texttt{signIn\_dly150} and \texttt{spOut} is valid. The specification of R17(1) is given below:

\begin{align*}
\texttt{signIn\_dly150} & \triangleq \texttt{signIn \ delayFor 150 on ms} \\
\texttt{signIn\_dly150} & \prec_p \texttt{spOut} \\
\{\texttt{signIn \ delayFor 150 on ms}\} & \prec_p \texttt{spOut} 
\end{align*}

By combining (30) and (31), we can obtain the the specification of R17(1):

\begin{align*}
\{\texttt{signIn \ delayFor 150 on ms}\} & \prec_p \texttt{spOut} 
\end{align*}

Similarly, to specify property R17(2), a new clock \texttt{signIn\_dly250} is generated by delaying \texttt{signIn} for 250 ticks on ms. Afterwards, the property that R17(2) is satisfied with a probability greater than or equal to \(p\) relies on whether the \textbf{probabilistic precedence} relation is satisfied. The specification is illustrated as follows:

\begin{align*}
\texttt{signIn\_dly250} & \triangleq \texttt{signIn \ delayFor 250 on ms} \\
\texttt{spOut} & \prec_p \texttt{signIn\_dly250} \\
\texttt{spOut} & \prec_p \{\texttt{signIn \ delayFor 250 on ms}\} 
\end{align*}

By combining (33) and (34), we can obtain the the specification of R17(2):
In similar, the CCSL/PrCCSL specification of R18 – R23 can be derived:

\[
\begin{align*}
\text{R18} : \{\text{cmrTrig delayFor 120 on ms}\} & \prec_p \text{signOut} \\
\text{signOut} & \prec_p \{\text{cmrTrig delayFor 180 on ms}\} \\
\text{R19} : \{\text{cmrTrig delayFor 270 on ms}\} & \prec_p \text{spOut} \\
\text{spOut} & \prec_p \{\text{cmrTrig delayFor 430 on ms}\}
\end{align*}
\] (5.37)

\[
\begin{align*}
\text{R20} : \{\text{startTurnLeft} \prec_p \text{DetectLeftSign delayFor 500 on ms}\} \\
\text{R21} : \{\text{startTurnRight} \prec_p \text{DetectRightSign delayFor 500 on ms}\} \\
\text{R22} : \{\text{startBrake} \prec_p \text{DetectStopSign delayFor 500 on ms}\} \\
\text{R23} : \{\text{Stop} \prec_p \text{DetectStopSign delayFor 3000 on ms}\}
\end{align*}
\] (5.38)

**Comparison** timing constraints (R24 – R26) can be specified using delayFor expression and probabilistic causality relation. R24 states that the execution time interval from Controller to VehicleDynamic should be less than or equal to the sum of the worst case execution time of Controller and VehicleDynamic, denoted as \(W_{ctrl}\) and \(W_{vd}\) respectively. To specify comparison constraint, we first construct a new clock \(\text{signIn\_dly250}\) by delaying \(\text{signIn}\) for 250 ticks of ms. Afterwards, we generate another new clock \(\text{signIn\_dlysw}\) that is the \(\text{signIn}\) clock delayed for sum of the worst case execution time of the two \(f_p\)s. The specification is illustrated as follows:

\[
\begin{align*}
\text{signIn\_dly250} & \triangleq \text{signIn delayFor 250 on ms} \\
\text{signIn\_dlysw} & \triangleq \text{signIn delayFor } (W_{ctrl} + W_{vd}) \text{ on ms}
\end{align*}
\] (5.43)

(5.44)

Therefore, the property that the probability of comparison constraint is satisfied should be greater than or equal to the threshold \(p\) can be interpreted as a probabilistic causality relation between \(\text{signIn\_dly250}\) and \(\text{signIn\_dlysw}\):

\[
\text{signIn\_dly250} \preceq_p \text{signIn\_dlysw}
\] (5.45)

By combining (42), (43) and (44), we can obtain the the specification of R24:

\[
\{\text{signIn delayFor 250 on ms}\} \preceq_p \{\text{signIn delayFor } (W_{ctrl} + W_{vd}) \text{ on ms}\}
\] (5.46)

Analogously, the CCSL/PrCCSL specification of R25 and R26 can be derived:

\[
\begin{align*}
\text{R25} : \{\text{cmrTrig delayFor 180 on ms}\} & \preceq_p \{\text{cmrTrig delayFor } (W_{cmr} + W_{sr}) \text{ on ms}\} \\
\text{R26} : \{\text{cmrTrig delayFor 430 on ms}\} & \preceq_p \{\text{cmrTrig delayFor } (W_{cmr} + W_{sr} + W_{ctrl} + W_{vd}) \text{ on ms}\}
\end{align*}
\] (5.47)

(5.48)
where $W_{cmr}$ and $W_{vd}$ represent the worst case execution time of Camera and SignRecognition respectively.

**Exclusion** timing constraints (R27 – R31) can be specified using exclusion relation directly. R27 states that the two events turnLeft (the event that the vehicle is turning left) and rightOn (the event that the turn right mode is activated) should be exclusive, which can be expressed as:

$$\text{turnLeft} \#_p \text{rightOn} \quad (5.49)$$

Analogously, the **Exclusion** timing constraints R28 – R31 can be specified using exclusion relation:

- **R28**: $\text{veAcc} \#_p \text{veBrake}$
- **R29**: $\text{emgcy} \#_p \text{turnLeft}$
- **R30**: $\text{emgcy} \#_p \text{rightOn}$
- **R31**: $\text{emgcy} \#_p \text{veAcc}$

where emgcy is the event that the vehicle is in the emergency mode, veBrake and veAcc represent the event that the vehicle is braking or accelerating, respectively.
Chapter 6

Translating CCSL & PrCCSL into UPPAAL-SMC

To formally verify the EAST-ADL timing constraints given in Chapter 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.

6.1 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 [22]. The STA of TimeUnit (universal time defined as ms) consists of one location and one outgoing transition whereby the physical time and the duration of TimeUnit ms are represented by the clock variable $t$ in Fig. 6.1(a). clock resets every time a transition is taken. The duration of TimeUnit is expressed by the invariant $t \leq 1$, and guard $t \geq 1$, i.e., a single step of the discrete time progress (tick) of universal time.

![Figure 6.1: UPPAAL-SMC model of clock tick and history](image)
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 runs ($R$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. 6.1(b). $hc$ is the history of $c$, and $tc$ indicates whether $c$ ticks at the current step. A function $\text{upper()}$ 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. 6.1(c)), $\text{upper()}$ 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 $\text{periodicOn}$, $\text{delayFor}$, $\text{infimum}$ and $\text{supremum}$ expressions can be represented as UPPAAL-SMC models.

**PeriodicOn**: $c \triangleq \text{periodicOn } ms \text{ period } q$, where $\triangleq$ means “is defined as”. $\text{PeriodicOn}$ 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 $\text{periodicOn}$ is illustrated in Fig. 6.2(a). This STA initially stays in the $\text{loop}$ 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 $\text{loop}$ and repeats the calculation. This $\text{periodicOn}$ STA can be used for the translation of EAST-ADL $\text{Periodic}$ timing constraint (R1 in Fig. 3.1) into its UPPAAL-SMC model.

**DelayFor**: $c \triangleq c1 \text{ delayFor } d \text{ on } c2$. $\text{DelayFor}$ 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 \( c_2 \), \( c \) ticks (each tick of \( c \) corresponds to a tick of \( c_1 \)). Kang et al. \cite{22} and Suryadevara et al. \cite{32} presented translation rules of \texttt{delayFor} into UPAAL models. However, their approaches are not applicable in the case after \( c_1 \) ticks, and \( c_1 \) ticks again before the \( d^{th} \) tick of \( c_2 \) occurs. For example (see Fig. 4.1), assume that \( d \) is 3. After the \( 1^{st} \) tick of \( c_1 \) (at step 0) happens, if \( c_1 \) ticks again (at step 2) before the \( 3^{rd} \) tick of \( c_2 \) occurs (at step 4), the \( 2^{nd} \) tick of \( c_1 \) is discarded in their approaches. To alleviate the restriction, we utilize spawnable STA \cite{12} as semantics denotation of \texttt{delayFor} expression and the STA of \texttt{delayFor} is shown in Fig. 6.2(c). As presented in Fig. 6.2(b), when the \( v^{th} \) tick of \( c_1 \) occurs (\( c_1[v] ? \)), its \texttt{delayFor} STA is spawned by \texttt{source} STA. The spawned STA stays in the \texttt{wait} location until \( c_2 \) ticks \( d \) times. When \( c_2 \) ticks \( d \) times (\( x == d \)), it transits to the \texttt{tick} location and triggers \( c \) (!). At the next step (\( ms ? \)), the STA increases \( hc \) by 1 and moves to \texttt{finish} location and then becomes inactive, i.e., calculation of the \( v^{th} \) tick of \( c \) is completed. This \texttt{delayFor} STA can be utilized to construct the UPAAL-SMC models of EAST-ADL timing requirements R5 – R26 in Chapter 3.

Given two clocks \( c_1 \) and \( c_2 \), their \texttt{infimum} (resp. \texttt{supremum}) is informally defined as the slowest (resp. fastest) clock faster (resp. slower) than both \( c_1 \) and \( c_2 \). \texttt{infimum} and \texttt{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 \texttt{expressions} are utilized for the translation of EAST-ADL \texttt{Synchronization} timing constraint (R13 in Chapter 3) into the UPAAL-SMC model.

\textbf{Infimum} creates a new clock \( c \), which is the slowest clock faster than \( c_1 \) and \( c_2 \). The STA of \texttt{infimum} is illustrated in Fig. 6.2(d). When \( c_1 \) (\( c_2 \)) ticks via \( c_1 ? \) (\( c_2 ? \)), the STA transits to the \texttt{s1} (\texttt{s2}) location and compares the history of the two clocks (\( h_1 \) and \( h_2 \)) to check whether the current ticking clock \( c_1 \) (\( c_2 \)) is faster than \( c_2 \) (\( c_1 \)). If so, i.e., the condition “\( h_1 \geq h_2 \) (\( h_2 \geq h_1 \))” holds, the STA takes a transition to the \texttt{tick} location and activates the tick of \( c \) (!). After updating the history (\( hc++ \)), it returns to the \texttt{init} location and repeats the calculation.

\textbf{Supremum} builds a new clock \( c \), which is the fastest clock slower than \( c_1 \) and \( c_2 \). It states that if \( c_1 \) ticks at the current step and \( c_1 \) is slower than \( c_2 \), then \( c \) ticks. The STA of \texttt{supremum} is shown in Fig. 6.2(e). When \( c_1 \) (\( c_2 \)) ticks via \( c_1 ? \) (\( c_2 ? \)), the STA transits to the \texttt{s1} (\texttt{s2}) location and compares the history of the two clocks and decides whether \( c_1 \) (\( c_2 \)) is slower than \( c_2 \) (\( c_1 \)). If \( c_1 \) (\( c_2 \)) ticks slower than \( c_2 \) (\( c_1 \)), i.e., \( h_1 < h_2 \) (\( h_2 < h_1 \)), or \( c_1 \) and \( c_2 \) tick at the same rate, i.e., “\( h_1 == h_2 \) &\& \( t_2 == 1 \) \( h_1 == h_2 \) &\& \( t_1 == 1 \)” holds, the tick of \( c \) is triggered. The STA then updates the history of \( c \) and goes back to \texttt{init} and repeats the process.
6.2 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 Chapter 2) is provided from the viewpoint of the analysis engine UPPAAL-SMC.

Recall the definition of PrCCSL in Chapter 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, \( H_0: \frac{m}{k} \geq P \) 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 \( \alpha \) (the probability of false positives, i.e., accepting \( H_1 \) when \( H_0 \) holds) and \( \beta \) (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 Chapter 6.1, 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, \( \text{turnLeft} \#_p \text{rightOn} \) (Spec. R27 in Fig. 3.1). It states that the two events, \( \text{turnLeft} \) and \( \text{rightOn} \) (the vehicle is turning left and right), must be exclusive. The ticks of \( \text{turnLeft} \) and \( \text{rightOn} \) events are modeled using the STA in Fig. 6.1(b). Based on the definition of probabilistic exclusion (Chapter 4), R8 is expressed in Hypothesis Testing query: \( \Pr[\text{bound}] \left[ (t_{\text{turnLeft}} \Rightarrow \neg t_{\text{rightOn}}) \land (t_{\text{rightOn}} \Rightarrow \neg t_{\text{turnLeft}}) \right] \geq P \), where \( t_{\text{turnLeft}} \) and \( t_{\text{rightOn}} \) indicate the ticks of \( \text{turnLeft} \) and \( \text{rightOn} \), respectively. \( \text{bound} \) is the time bound of simulation, in our setting \( \text{bound} = 3000 \).

**Probabilistic Causality** is used to specify EAST-ADL Synchronization timing constraint, \( \sup \preceq_p \{ \inf \text{delayFor 40 on ms} \} \) (Spec. R13 in Fig. 3.1), where \( \sup \) (resp. \( \inf \)) is the fastest (slowest) event slower (faster) than five input events, \( \text{speed}, \text{signType}, \text{direct}, \text{gear} \) and \( \text{torque} \). Let \( \sup \) and \( \inf \) denote the supremum and infimum operator, i.e., \( \sup(c_1, c_2) \) (resp. \( \inf(c_1, c_2) \)) returns the supremum (resp. infimum) of clock \( c_1 \) and \( c_2 \). \( \sup \) and \( \inf \) can now be expressed with the nested operators (where \( \triangleq \) means “is defined as”):

\[
\sup \triangleq \sup(\text{speed}, \sup(\sup(\text{signType}, \text{direct}), \sup(\text{gear}, \text{torque})))
\]

\[
\inf \triangleq \inf(\text{speed}, \inf(\inf(\text{signType}, \text{direct}), \inf(\text{gear}, \text{torque})))
\]

For the translation of \( \sup \) (resp. \( \inf \)) into UPPAAL-SMC model, we employ the STA of \( \sup \) (resp. \( \inf \)) (Fig. 6.2(d) and (e)) for each \( \sup \) (INF) operator. A new clock \( d\inf \) is generated by delaying \( \inf \) for 40 ticks of \( \text{ms} \): \( d\inf \triangleq \{ \inf \text{delayFor 40 on ms} \} \). The UPPAAL-SMC model of \( d\inf \) is achieved by adapting the spawnable \( \text{DelayFor} \) STA (Fig. 6.2). Based on the probabilistic causality definition, R13 is interpreted as: \( \Pr[\leq \text{bound}] \left[ h_{\sup} \geq h_{d\inf} \right] \geq P \), where \( h_{\sup} \) and \( h_{d\inf} \) are the history of \( \sup \) and \( d\inf \) respectively.
Similarly, Execution (R5) and Comparison (R25) timing constraints specified in probabilistic causality using delayFor can be translated into Hypothesis Testing queries. R5 (\{imIn delayFor 100 on ms\} \preceq_p signOut, signOut \preceq_p \{imIn delayFor 150 on ms\}) specifies that the execution time of SignRecognition \(f_p\) measured from input port imIn to output port signOut should be limited within [100, 150]ms. To translate Execution timing constraint into UPPAAL-SMC STA, two new clocks SL and SU are constructed by delaying imIn for 100 and 150 ticks of ms: \(SL \triangleq \{imIn delayFor 100 on ms\}\), \(SU \triangleq \{imIn delayFor 150 on ms\}\). According to the definition of probabilistic causality, R5 can be specified as: \(Pr[[\leq \text{bound}]](\sum h_{SL} \geq h_S) \geq P, Pr[[\leq \text{bound}]](\sum h_S \geq h_{SU}) \geq P\), where \(h_{SU}\) and \(h_{SL}\) represent the history of SU and SL, and \(h_S\) indicates the history of clock signOut.

Comparison constraint (R25) specified as \(\{signIn delayFor 250 on ms\} \preceq_p \{signIn delayFor \sum WCET on ms\}\) can be model using the DelayFor STA. Two new clocks CU, com are generated: \(CU \triangleq \{signIn delayFor 250 on ms\}\), \(com \triangleq \{signIn delayFor \sum WCET on ms\}\), where \(\sum WCET\) represents the sum of worst case execution time of Controller and VehicleDynamics \(f_p\). Therefore, R25 can be expressed as the query: \(Pr[[\leq \text{bound}]](\sum ex_{con} == wcet_{con} \land ex_{vd} == wcet_{vd}) \implies (h_{con} \geq h_{CU}) \geq P, \) where \(ex_{con} == wcet_{con} \land ex_{vd} == wcet_{vd}\) restricts that when the execution is the worst case (i.e., the execution time is the longest), the probabilistic causality relation between \(con\) and \(CU\) should be guaranteed.

**Probabilistic Precedence** is utilized to specify EAST-ADL End-to-End timing constraint (R17). 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 quires (56) and (57):

\[
\{signIn delayFor 150 on ms\} \preceq_p spOut \quad (6.1)
\]

\[
spOut \preceq_p \{signIn delayFor 250 on ms\} \quad (6.2)
\]

Two clocks, lower and upper, are defined by delaying signIn for 150 and 250 ticks of ms respectively: \(lower \triangleq \{signIn delayFor 150 on ms\}\), and \(upper \triangleq \{signIn delayFor 250 on ms\}\). The corresponding UPPAAL-SMC models of lower and upper are constructed based on the delayFor STA (shown in Fig. [6.2]). Finally, the R17 specified in PrCCSL is expressed as UPPAAL-SMC quires (3) and (4), where \(h_{lower}, h_{upper}\) and \(h_{spOut}\) are the history of lower, upper and spOut. \(t_{spOut}\) and \(t_{upper}\) represent the tick of upper and spOut respectively:

\[
Pr[[\leq \text{bound}]](\sum h_{lower} \geq h_{spOut} \land (h_{lower} == h_{spOut}) \implies t_{spOut} == 0) \geq P \quad (6.3)
\]

\[
Pr[[\leq \text{bound}]](\sum h_{spOut} \geq h_{upper} \land (h_{spOut} == h_{upper}) \implies t_{upper} == 0) \geq P \quad (6.4)
\]
In similar, EAST-ADL Sporadic timing constraint (R9) specified in \textbf{probabilistic precedence} can be translated into \textit{Hypothesis Testing} query $Pr[\leq \text{bound}]([\text{ho} \geq hv \land ((hv == ho) \implies t_{va} == 0)) \geq P$, where \text{ho} represents the history of the clock/event that obstacle occurs, and \text{t}_{va} and \text{hv} indicates the ticks and history of the clock that the vehicle starts to move.

In the case of properties specified in either \textbf{probabilistic subclock} or \textbf{probabilistic coincidence}, such properties can not be directly expressed as UPPAAL-SMC queries. Therefore, we construct an observer STA that captures the semantics of standard \textit{subclock} and \textit{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 \textbf{probabilistic subclock} and \textbf{probabilistic coincidence} of the entire system using UPPAAL-SMC. Further details are given below.

\textbf{Probabilistic Subclock} is employed to specify EAST-ADL Periodic timing constraint, given as $\text{signRecTrig} \subseteq_p \text{cTrig}$ (Spec. R2 in Fig.1). The standard \textit{subclock relation} states that \textit{superclock} must tick at the same step where \textit{subclock} ticks. Its corresponding STA is shown in Fig. 6.3(a). When $\text{signRevTrig}$ ticks ($\text{signRecTrig}$?), the STA transits to the \textit{wait} location and detects the occurrence of $\text{cTrig}$ until the time point of the subsequent step ($u$). If $\text{cTrig}$ occurs prior to the next step ($\text{tcTrig} == 1$), the STA moves to the \textit{success} location, i.e., the \textit{subclock relation} is satisfied at the current step. Otherwise, it transits to the \textit{fail} location. R2 specified in \textbf{probabilistic subclock} is expressed as: $Pr[\text{bound}]([\neg \text{Subclock.fail}) \geq P$. UPPAAL-SMC analyzes if the \textit{fail} location is never reachable from the system NSTA, and whether the probability of R2 being satisfied is greater than or equal to $P$.

\textbf{Probabilistic Coincidence} is adapted to specify EAST-ADL Periodic timing constraint, given as $\text{cTrig} \equiv_p \{\text{periodicOn ms period 50}\}$ (Spec. R1 in Fig.1). To express R1 in UPPAAL-SMC, first, a periodic clock $\text{prdClk}$ ticking every 50th tick of ms is defined: $\text{prdClk} \triangleq \text{periodicOn ms period 50}$. The corresponding UPPAAL-SMC model of $\text{prdClk}$ is generated based on the periodicOn STA shown in Fig. 6.2(a) by setting $q$ as 50. Then, we check if $\text{cTrig}$ and $\text{prdClk}$ are coincident by employing the \textit{coincidence} STA shown in Fig. 6.3(b). When $\text{cTrig}$ ($\text{prdClk}$) ticks via $\text{cTrig}$? ($\text{prdClk}$?), the STA checks if the other

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig63.png}
\caption{Observer STA of Subclock and Coincidence}
\end{figure}
clock, prdClk ($cTrig$), ticks prior to the next step, i.e., whether $tprdClk == 1$ ($tcTrig == 1$) holds or not when $t \leq u$. The STA then transits to either the success or fail location based on the judgement. R1 specified in probabilistic coincidence is expressed as: $Pr[bound][\neg Coincidence.fail] \geq P$. UPPAAL-SMC analyzes if the probability of R1 being satisfied is greater than or equal to $P$. 
Chapter 7

Modeling the Behaviors of AV and its Environment in UPPAAL-SMC

To capture the behaviours of the AV system and the stochastic behaviours of its environments, e.g., random traffic signs, each $f_p$ in Fig. 3.1 is modeled as an STA in UPPAAL-SMC. The random traffic sign in the environment is recognised by AV. The speed of the AV is influenced by the condition of the road. Obstacles on the road occurs randomly. To model these stochastic behaviours, we model the three $f_p$s in the Environment $f_i$ into three STAs, which are presented in Fig. 7.1. In TrafficSign (Fig. 7.2(b)) STA, $sign\_num$ represents the random traffic sign type, which is generated every 4ms to 8ms. To represent the integration of the AV system and the environment, the speed of AV is equal to the speed of in the environment, the Speed (shown in Fig. 7.1(c)) STA updates the speed of the vehicle in the environment from the by activating the execution of $update()$ function periodically. Obstacle STA generates a signal randomly based on probability distribution to represent random obstacles.

The system model of AV is represented as the STAs shown in Fig. 7.2. Camera STA is triggered periodically (Fig. 7.2(a)). When the execution of camera is finished, i.e., the transition from $s4$ to $s5$ is taken, the $update()$ function is triggered and the value of $sign\_num$ is assigned to $signType$. Since the input ports ($speed$ and $obstacle$) of Controller are triggered periodically, the AV system obtains the speed of the vehicle and the road information by executing the $update()$ periodically (Fig. 7.3).

The internal behaviours of Controller $f_p$ is captured in Fig. 7.4. When the vehicle is in the “normal” mode (Fig. 7.4(c)) and it encounters an obstacle, the “emergency stop” mode will be activated (Fig. 7.4(b)) and the vehicle begins to stop. In “normal” mode, the vehicle adjusts its movement according to the traffic signs, e.g., when it detects a turn left sign, it will turn left (Fig. 7.5(d)). The Controller then sends out requests for VehicleDynamic to change the direction or the speed of the four wheels.

To verify R1 to R31, STAs of CCSL expressions periodicOn, infimum, supremum and delayFor and STAs of PrCCSL relations coincidence and subclock.
Figure 7.1: STAs of $f_p$ in Environment $f_t$
Figure 7.2: Modeling system behaviors in UPPAAL-SMC
Probabilistic Analysis of Weakly-Hard Real-Time Systems

Figure 7.3: Periodically triggered ports speed and obstacle

(a) speed

(b) obstacle

Figure 7.4: Internal behaviours of Controller in UPPAAL-SMC

(c) Normal
Probabilistic Analysis of Weakly-Hard Real-Time Systems

Figure 7.5: Representation of substates of Normal state in UPPAAL-SMC
are utilized. For example, to verify R3, a \texttt{periodicOn} STA generates a new clock \(c\) with period 40 (Fig. 7.6(a)). When \(c\) ticks, the \texttt{periodico} will be assigned to 1. The probabilistic coincidence relation between \(c\) and the triggering of the \texttt{obstacle} port should hold. When the input port is triggered, \texttt{obstrig} will become 1 in Fig. 7.3(b). Coincidence STA (Fig. 7.6(b)) is employed for checking the coincidence relation between \(c\) and \texttt{obstacle}.
Chapter 8

Experiments: Verification & Validation

We have formally analyzed over 30 properties (associated with timing constraints) of the system including deadlock freedom. A list of selected properties (Chapter 3) are verified using UPPAAL-SMC and the results are listed in Table 8.1. Five types of UPPAAL-SMC queries are employed to specify R1 – R31, Hypothesis Testing (HT), Probability Estimation (PE), Probability Comparison (PC), Expected Value (EV) and Simulations (SI).

1. **Deadlock Freedom**: Because of the insufficient memory caused by the periodically triggered STA ms, the Deadlock Freedom property cannot be checked successfully. 2. **Hypothesis Testing**: All properties are established as valid with 95% level of confidence; 3. **Probability Estimation**: The probability of each property being satisfied is computed and its approximate interval is given as [0.902, 1]; 4. **Expected Value**: The expected values of time durations of timing constraints (R1, R2, R5, R9, R13, R15, R17, R24 – R25) 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 R17 and generates the frequency histogram of the expected bound (see Fig. 8.2). It illustrates that the expected bound is always less than 250ms and 90% of the duration is within the range of [207, 249]; 5. **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 (R5). 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 125ms. Similarly, R6 – R8 can be analyzed. 6. **Simulation**: The simulation result of Synchronization timing constraint (R13) is demonstrated in Fig. 8.1. $h_{inf}$, $h_{sup}$ and $h_{dinf}$ are history of $inf$, $sup$ and $dinf$ respectively. Recall Spec. R13 (see Fig. 3.1), the causality relation between $dinf$ and $sup$ is satisfied. As the simulation of R13 shows (Fig. 8.1), 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.

We estimate the performance (i.e., time, memory and CPU consumption)
### Table 8.1: Verification Results in UPPAAL-SMC

| Type       | R.ID | Q Expression | Result | Time  | Mem  | CPU  |
|------------|------|--------------|--------|-------|------|------|
| Periodic   | HT   | Pr(\[3000\]| ¬(Coin.fail) \geq 0.95) | valid | 48.7  | 32.7 | 31.3 |
|            | PE   | Pr(\[3000\]| ¬(Coin.fail)) \geq 0.902, 1 | 12.6  | 35.6  | 29.8 |
|            | EV   | E(\[3000\]; 500\]| max : cam.f) | 50±0 | 83.3  | 33.3 | 31.7 |
|            | SI   | simulate 500 \[\[3000\]| cam.trig, p1trigger) | valid | 80.9  | 32.9 | 32.5 |
|           | HT   | Pr(\[3000\]| ¬(Sub.fail) \geq 0.95) | valid | 48.9  | 32.9 | 29.3 |
|            | PE   | Pr(\[3000\]| ¬(Sub.fail)) \geq 0.902, 1 | 12.3  | 35.5  | 30.4 |
|            | EV   | E(\[3000\]; 500\]| max : sf.t) | 200±0 | 80.6  | 32.5 | 32.2 |
|            | SI   | simulate 500 \[\[3000\]| strig, p2trigger) | valid | 85.5  | 33.1 | 32.3 |
|           | HT   | Pr(\[3000\]| ¬(Coin_may fail) \geq 0.95) | valid | 57.6  | 40.5 | 34.6 |
|            | PE   | Pr(\[3000\]| ¬(Coin_may fail)) \geq 0.902, 1 | 13.8  | 40.4  | 31.1 |
|            | HT   | Pr(\[3000\]| ¬(Coin fail) \geq 0.95) | valid | 56.7  | 40.4 | 32.4 |
|            | PE   | Pr(\[3000\]| ¬(Coin fail)) \geq 0.902, 1 | 13.6  | 35.9  | 34.0 |
| Execution  | HT   | Pr(\[3000\]| h_{may} \leq h_{s}) \geq 0.95 | valid | 76.5  | 40.4 | 32.3 |
|            | PE   | Pr(\[3000\]| h_{may} \leq h_{s}) \geq 0.902, 1 | 18.1  | 40.3  | 30.8 |
|            | HT   | Pr(\[3000\]| h_{s} \leq h_{may}) \geq 0.95 | valid | 77.6  | 37.7 | 31.7 |
|            | PE   | Pr(\[3000\]| h_{s} \leq h_{may}) \geq 0.902, 1 | 16.5  | 40.0  | 31.5 |
|            | PC   | Pr(\[3000\]| (SR.exe \Rightarrow (SR.t \geq 100 \land SR.t \leq 125)) \geq 0.902, 1 | 8.3   | 31.7  | 32.3 |
|            | SI   | simulate 500 \[\[3000\]| h_{may}, h_{s}, h_{may}) | valid | 85.8  | 32.4 | 36.0 |
|            | HT   | Pr(\[3000\]| h_{may} \leq h_{con}) \geq 0.95 | valid | 46.1  | 38.2 | 33.7 |
|            | PE   | Pr(\[3000\]| h_{may} \leq h_{con}) \geq 0.902, 1 | 10.8  | 37.9  | 30.8 |
|            | HT   | Pr(\[3000\]| h_{con} \leq h_{may}) \geq 0.95 | valid | 38.1  | 34.4 | 32.3 |
|            | PE   | Pr(\[3000\]| h_{con} \leq h_{may}) \geq 0.902, 1 | 9.9   | 37.9  | 31.2 |
|            | PC   | Pr(\[3000\]| (cam.exe \Rightarrow (cam.t \geq 20 \land cam.t \leq 25)) \geq 0.902, 1 | 4s    | 34.0  | 30.9 |
|            | SI   | simulate 100 \[\[3000\]| h_{may}, h_{con}, h_{con}) | valid | 33.4  | 38.8 | 34.2 |
| Sporadic   | PE   | Pr(\[3000\]| h_{may} \leq h_{s}) \geq 0.95 | valid | 14.5  | 35.8 | 35.4 |
|            | PE   | Pr(\[3000\]| h_{may} \leq h_{s}) \geq 0.902, 1 | 15.4  | 35.9  | 33.1 |
|            | PC   | Pr(\[3000\]| (\| VD.exe \Rightarrow (VD.t \geq 50 \land VD.t \leq 75)) \geq 0.902, 1 | 10.1  | 34.1  | 31.5 |
|            | SI   | simulate 100 \[\[3000\]| h_{may}, h_{s}, h_{con}) | valid | 35.8  | 39.2 | 33.3 |
| Synchronization | HT   | Pr(\[3000\]| h_{fail} \leq h_{sup}) \geq 0.95 | valid | 3h    | 33.1 | 30.0 |
|            | PE   | Pr(\[3000\]| h_{fail} \leq h_{sup}) \geq 0.902, 1 | 45.4  | 33.1  | 29.4 |
|            | EV   | E(\[3000\]; 500\]| max : obs.t) | 667±79 | 80.8  | 29.7 | 31.7 |
|            | SI   | simulate 500 \[\[3000\]| h_{fail}, h_{sup}) | valid | 88.6  | 29.5 | 31.0 |
|            | HT   | Pr(\[3000\]| h_{tr} \leq h_{sup} \land (h_{tr} \leq h_{sup}) \Rightarrow t_{va} = 0) \geq 0.95 | valid | 52.9  | 44.1 | 31.0 |
|            | PE   | Pr(\[3000\]| h_{tr} \leq h_{sup} \land (h_{tr} \leq h_{sup}) \Rightarrow t_{va} = 0) \geq 0.902, 1 | 56.2  | 42.4  | 30.7 |
|            | SI   | simulate 100 \[\[3000\]| h_{tr}, h_{sup}) | valid | 56.7  | 41.7 | 29.8 |
|            | HT   | Pr(\[3000\]| h_{delay} \leq h_{sup}) \geq 0.95 | valid | 53.9  | 32.7 | 31.9 |
|            | PE   | Pr(\[3000\]| h_{delay} \leq h_{sup}) \geq 0.902, 1 | 13.7  | 35.5  | 30.4 |
|            | EV   | E(\[3000\]; 500\]| max : checksync.t) | 30.6±21 | 72.4  | 32.6 | 31.6 |
|            | SI   | simulate 500 \[\[3000\]| h_{delay} \leq h_{sup}) | valid | 86.8  | 32.6 | 32.0 |
|            | HT   | Pr(\[3000\]| h_{delay} \geq h_{sup}) \geq 0.95 | valid | 41.8  | 37.5 | 33.4 |
|            | PE   | Pr(\[3000\]| h_{delay} \geq h_{sup}) \geq 0.902, 1 | 14.3  | 40.5  | 35.1 |
|            | EV   | E(\[3000\]; 100\]| max : checksyncd.t) | 16.5±0.2 | 19.4  | 46.5 | 25.5 |
|            | HT   | Pr(\[3000\]| h_{delay} \geq h_{sup}) \geq 0.95 | valid | 55.2  | 45.3 | 32.1 |
|            | PE   | Pr(\[3000\]| h_{delay} \geq h_{sup}) \geq 0.902, 1 | 13.9  | 40.7  | 33.5 |
| Type     | R.ID | Q | Expression                                                                 | Result | Time | Mem | CPU |
|----------|------|---|----------------------------------------------------------------------------|--------|------|-----|-----|
| End-to-End | R17  | HT | Pr[<3000]([ | h_{lower} ≥ h_{appOut} ∧ (h_{lower} == h_{appOut}) ===> l_{appOut} == 0)] ≥ 0.95 | valid  | 54.2 | 32.9 | 31.4 |
|          |      | PE | Pr[<3000]([ | h_{lower} ≥ h_{appOut} ∧ (h_{lower} == h_{appOut}) ===> l_{appOut} == 0)] ≥ 0.95 | 0.902, 1 | 13.1 | 35.3 | 29.4 |
|          |      | HT | Pr[<3000]([ | h_{appOut} ≥ h_{upper} ∧ (h_{appOut} == h_{upper}) ===> l_{upper} == 0)] ≥ 0.95 | valid  | 1.3h | 32.2 | 32.6 |
|          |      | PE | Pr[<3000]([ | h_{appOut} ≥ h_{upper} ∧ (h_{appOut} == h_{upper}) ===> l_{upper} == 0)] ≥ 0.95 | 0.902, 1 | 19.8 | 34.1 | 32.9 |
|          |      | EV | E[<3000; 500][ | max : check2e.t] | 229.7±0.9 | 83.3 | 32.5 | 30.6 |
|          |      | SI | simulate 100 [%3000](h_{cam}, h_{camOut}, h_{signOut}), (h_{cam}, h_{camOut}) | valid  | 89.8 | 32.9 | 30.2 |
|          |      | R18 | HT | Pr[<3000]([ | h_{cam} ≥ h_{signOut} ∧ (h_{cam} == h_{signOut}) ===> t_{signOut} == 0)] ≥ 0.95 | valid  | 3.1h | 45.3 | 31.5 |
|          |      | HT | Pr[<3000]([ | h_{cam} ≤ h_{signOut} ∧ (h_{cam} == h_{signOut}) ===> t_{signOut} == 0)] ≥ 0.95 | valid  | 56.6 | 46.7 | 31.6 |
|          |      | SI | simulate 100 [%3000](h_{cam}, h_{signOut}, t_{camOut}) | valid  | 50.5 | 39.9 | 28.6 |
|          |      | PE | Pr[<3000]([ | h_{cam} ≥ h_{signOut} ∧ (h_{cam} == h_{signOut}) ===> t_{signOut} == 0)] | [0.902, 1] | 52.7 | 39.3 | 30.4 |
|          |      | PE | Pr[<3000]([ | h_{cam} ≤ h_{signOut} ∧ (h_{cam} == h_{signOut}) ===> t_{signOut} == 0)] | [0.902, 1] | 2.4h | 45.6 | 30.2 |
|          |      | R19 | HT | Pr[<3000]([ | h_{cam} ≥ t_{camOut} ∧ (h_{cam} == t_{camOut}) ===> t_{camOut} == 0)] | valid  | 1.9h | 40.8 | 29.8 |
|          |      | SI | simulate 100 [%3000](h_{cam}, h_{camOut}, t_{camOut}) | valid  | 151.3 | 41.9 | 29.1 |
|          |      | SI | simulate 100 [%3000](h_{cam}, t_{cam}) | valid  | 58.4 | 37.3 | 24.5 |
|          |      | R20 | HT | Pr[<3000]([ | h_{cam} ≥ t_{camOut} ∧ (h_{cam} == t_{camOut}) ===> t_{camOut} == 0)] | valid  | 75.9 | 46.8 | 31.3 |
|          |      | SI | simulate 100 [%3000](h_{cam}, h_{camOut}, t_{camOut}) | valid  | 64.8 | 41.8 | 32.0 |
|          |      | PE | Pr[<3000]([ | h_{cam} ≥ t_{camOut} ∧ (h_{cam} == t_{camOut}) ===> t_{camOut} == 0)] | [0.902, 1] | 18.5 | 41.9 | 27.3 |
|          |      | SI | simulate 100 [%3000](h_{cam}, h_{camOut}, t_{camOut}) | valid  | 57.5 | 36.9 | 35.5 |
|          |      | PE | Pr[<3000]([ | h_{camOut} ≥ t_{camOut} ∧ (h_{camOut} == t_{camOut}) ===> t_{camOut} == 0)] | [0.902, 1] | 26.8 | 42.3 | 27.8 |
|          |      | SI | simulate 100 [%3000](h_{camOut}, t_{camOut}, t_{cam}) | valid  | 73.6 | 42.4 | 27.9 |
| Comparison | R24  | HT | Pr[<3000]([ | ex_{con} == wcet_{cam} ∧ ex_{vd} == wcet_{vd}] ===> (h_{cu} ≥ h_{con})) | valid  | 57.4 | 36.7 | 28.4 |
|          |      | PE | Pr[<3000]([ | ex_{con} == wcet_{cam} ∧ ex_{vd} == wcet_{vd}] ===> (h_{cu} ≥ h_{con})) | [0.902, 1] | 14.7 | 35.5 | 26.1 |
|          |      | EV | E[<3000; 500][ | max : control.t] | 146.7±0.28 | 74.9 | 29.4 | 32.7 |
|          |      | EV | E[<3000; 500][ | max : vd.t] | 98.6±0.27 | 74.2 | 29.4 | 31.4 |
|          |      | SI | simulate 100 [%3000](h_{cu}, h_{con}) | valid  | 86.6 | 29.5 | 32.5 |
|          |      | R25 | EV | E[<3000; 100][ | max : camera.t] | 29.8±0.02 | 18.7 | 39.6 | 29.9 |
|          |      | EV | E[<3000; 100][ | max : signreg.t] | 143±0.7 | 16.5 | 33.2 | 28.7 |
|          |      | SI | simulate 100 [%3000](h_{sign}, h_{signOut}) | valid  | 12.6 | 36.6 | 29.8 |
|          |      | R26 | HT | Pr[<3000]([ | ex_{con} == wcet_{con} ∧ ex_{vd} == wcet_{vd} ∧ ex_{cam} == wcet_{cam} ∧ ex_{sign} == wcet_{sign}] ===> (h_{cu} ≥ h_{con})) | valid  | 2.1h | 42.5 | 30.1 |
|          |      | PE | Pr[<3000]([ | ex_{con} == wcet_{con} ∧ ex_{vd} == wcet_{vd} ∧ ex_{cam} == wcet_{cam} ∧ ex_{sign} == wcet_{sign}] ===> (h_{cu} ≥ h_{con})) | [0.902, 1] | 56.9 | 40.7 | 29.7 |
|          |      | HT | Pr[<3000]([ | -(highth == 1 ∧ t_{left} == 1)] ≥ 0.95 | valid  | 57.4 | 36.7 | 28.4 |
|          |      | PE | Pr[<3000]([ | -(highth == 1 ∧ t_{left} == 1)] ≥ 0.95 | [0.902, 1] | 14.7 | 35.5 | 26.7 |
|          |      | FC | Pr[<3000]([ | -(highth == 1 ∧ t_{left} == 1)] ≥ 0.95 | ≥11.0 | 10.9 | 34.2 | 31.3 |
|          |      | SI | simulate 100 [%3000](h_{sign}, h_{signOut}) | valid  | 85.5 | 29.6 | 32.6 |
|          |      | R27 | HT | Pr[<3000]([ | -(tone == 1 ∧ t_{acc} == 1)] ≥ 0.95 | valid  | 57.5 | 44.6 | 35.9 |
|          |      | PE | Pr[<3000]([ | -(tone == 1 ∧ t_{acc} == 1)] ≥ 0.95 | [0.902, 1] | 14.3 | 40.7 | 35.6 |
|          |      | R28 | HT | Pr[<3000]([ | -(tone == 1 ∧ t_{acc} == 1)] ≥ 0.95 | valid  | 62.6 | 40.6 | 33.6 |
|          |      | SI | simulate 100 [%3000](t_{acc}, t_{accOut}) | valid  | 46.5 | 36.4 | 34.2 |
|          |      | R29 | HT | Pr[<3000]([ | -(tone == 1 ∧ t_{acc} == 1)] ≥ 0.95 | valid  | 63.8 | 36.3 | 34.2 |
|          |      | SI | simulate 100 [%3000](t_{acc}, t_{accOut}) | valid  | 41.7 | 36.4 | 34.5 |
|          |      | PE | Pr[<3000]([ | -(tone == 1 ∧ t_{acc} == 1)] ≥ 0.95 | [0.902, 1] | 15.5 | 36.7 | 30.1 |

Figure 8.1: Simulation Result of R13
Figure 8.2: Frequency Histogram of End-to-End timing constraint (R17)

Figure 8.3: Performance analysis of verifying R5 with Expected Value and Simulation. The number of runs ranges from 100 to 500 with increment as 100.

of verifying R5 by using Expected Value and Simulation queries with different numbers of runs assigned. As shown in Fig. 8.3, along with the increase of the number of runs, for both queries, the verification time grows proportionally, while the CPU and memory have no significant changes.
Chapter 9

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, 23, 30], which are however, limited to the executional aspects of system functions without addressing stochastic behaviors. Kang [22] and Suryadevara [32, 33] 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 [34] transformed a subset of CCSL constraints to PROMELA models to perform formal verification using SPIN. Zhang [35] 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. [27] presented 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, 20, 21] 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.
Chapter 10

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. \url{https://www.autosar.org/}

[2] UPPAAL-SMC. \url{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. \url{http://www.maenad.eu/} (2011)

[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. Transactions on Computers 50(4), 308 – 321 (2001)

[9] Blom, H., Feng, L., Lönn, H., Nordlander, J., Kuntz, S., Lisper, B., Quinton, S., Hanke, M., Peraldi-Frati, M.A., Goknil, A., Deantoni, J., Defo, G.B., Klobedanz, K., Özhan, M., Honcharova, O.: TIMMO-2-USE Timing Model, Tools, Algorithms, Languages, Methodology, Use Cases. Tech. rep., TIMMO-2-USE (2012)

[10] Bulychev, P., David, A., Larsen, K.G., Mikučionis, M., Poulsen, D.B., Legay, A., Wang, Z.: UPPAAL-SMC: Statistical model checking for priced timed automata. In: QAPL. pp. 1–16. EPTCS (2012)

[11] David, A., Du, D., Larsen, K.G., Legay, A., Mikučionis, M., Poulsen, D.B., Sedwards, S.: 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: FACS. pp. 111 – 133. Springer (2016)

[14] EAST-ADL Consortium: EAST-ADL domain model specification v2.1.9. Tech. rep., 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: ECSA. pp. 89 – 105. Springer (2013)

[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., Enouiu, E.P., Marinescu, R., Seceleanu, C., Schobbens, P.Y., Pettersson, P.: A methodology for formal analysis and verification of EAST-ADL models. Reliability Engineering & System Safety 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., Lan, Q.: Model-based analysis of timing and energy constraints in an autonomous vehicle system. In: QRS. pp. 525 – 532. IEEE (2017)

[21] 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)

[22] Kang, E.Y., Schobbens, P.Y.: Schedulability analysis support for automotive systems: from requirement to implementation. In: SAC. pp. 1080 – 1085. ACM (2014)

[23] Kang, E.Y., Schobbens, P.Y., Pettersson, P.: Verifying functional behaviors of automotive products in EAST-ADL2 using UPPAAL-PORT. In: SAFE-COMP. pp. 243 – 256. Springer (2011)

[24] Legay, A., Viswanathan, M.: Statistical model checking: challenges and perspectives. STTT 17(4), 369 – 376 (2015)
[25] Mallet, F., Peraldi-Frati, M.A., Andre, C.: MARTE CCSL to execute EAST-ADL timing requirements. In: ISORC. pp. 249 – 253. IEEE (2009)

[26] Mallet, F., De Simone, R.: Correctness issues on MARTE/CCSL constraints. Science of Computer Programming 106, 78 – 92 (2015)

[27] Marinescu, R., Kaijser, H., Mikučionis, M., Seceleanu, C., Lönn, H., David, A.: Analyzing industrial architectural models by simulation and model-checking. In: FTSCS. pp. 189 – 205. Springer (2014)

[28] Nicolau, G.B.: Specification and analysis of weakly hard real-time systems. Transactions on Computers pp. 308 – 321 (1988)

[29] Object Management Group: UML profile for MARTE: Modeling and analysis of real-time embedded systems (2015)

[30] 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)

[31] Simulink and Stateflow. https://www.mathworks.com/products.html

[32] Suryadevara, J.: Validating EAST-ADL timing constraints using UPPAAL. In: SEAA. pp. 268 – 275. IEEE (2013)

[33] Suryadevara, J., Seceleanu, C., Mallet, F., Pettersson, P.: Verifying MARTE/CCSL model behaviors using UPPAAL. In: SEFM. pp. 1 – 15. Springer (2013)

[34] Yin, L., Mallet, F., Liu, J.: Verification of MARTE/CCSL time requirements in PROMELA/SPIN. In: ICECCS. pp. 65 – 74. IEEE (2011)

[35] Zhang, M., Ying, Y.: Towards SMT-based LTL model checking of clock constraint specification language for real-time and embedded systems. ACM SIGPLAN Notices 52(4), 61 – 70 (2017)