Received 8 April 2023, accepted 26 April 2023, date of publication 3 May 2023, date of current version 10 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3272672

RESEARCH ARTICLE

stohMCharts: A Modeling Framework for Quantitative Performance Evaluation of Cyber-Physical-Social Systems

DONGDONG AN[1,2], ZONGXU PAN[1,2], XIN GAO[1,2], SHUANG LI[1,2], LING YIN[3], AND TENGFEI LI[4,5]

[1]Shanghai Engineering Research Center of Intelligent Education and Bigdata, Shanghai Normal University, Shanghai 200234, China
[2]College of Information, Mechanical and Electrical Engineering, Shanghai Normal University, Shanghai 200234, China
[3]College of Electronic and Electrical Engineering, Shanghai University of Engineering Science, Shanghai 201620, China
[4]Casco Signal Ltd., Shanghai 200071, China
[5]School of Software Engineering, East China Normal University, Shanghai 200050, China

Corresponding author: Xin Gao (sizegaoxin@126.com)

This work was supported in part by the Shanghai Sailing Program under Grant 21YF1432900, in part by the National Natural Science Foundation of China under Grant 61802251, and in part by the Shanghai Post-Doctoral Excellence Program under Grant 2021146.

ABSTRACT Cyber-physical-social systems (CPSS) have recently gained attention from researchers due to their combination of cyber, physical, and social spaces. Modeling and Analysis of Real-Time and Embedded systems (MARTE) is a Unified Modeling Language (UML) extension profile that supports the specification, design, and verification of Real-Time Embedded Systems (RTES). While MARTE Statecharts can assist in describing CPS, it does not model the uncertainty within a CPSS environment. To enhance the accuracy of CPSS analysis, we propose the stohMCharts (stochastic hybrid MARTE statecharts) modelling framework as an extension of MARTE statecharts for modelling and analyzing stochastic hybrid systems. stohMCharts can model CPSS in a unified manner. Additionally, based on the mapping rules and algorithms, we have developed a tool to convert models built in stohMChart language into Networks Stochastic Hybrid Automata (NSHA) which can be verified by statistical model checker UPPAAL-SMC. We demonstrate the efficiency and accuracy of the framework by applying it to one autonomous driving scenarios.

INDEX TERMS Cyber-physical-social systems (CPSS), modeling and analysis of real-time and embedded systems (MARTE), network of stochastic hybrid automata (NSHA), quantitative evaluation, automatic vehicles.

I. INTRODUCTION

The field of cyber-physical systems (CPS) and the Internet of Things (IoT) have focused on the interaction between physical objects, but not on human activity [1]. A new paradigm, cyber-physical-social systems (CPSS), has emerged to revolutionize the relationship between humans, computers, and the physical environment [2]. Due to the increasing interaction with the external uncertain physical environment, the complexity of CPSS design is rapidly developing. Modeling human reactions and system behaviors in an uncertain environment and guaranteeing critical functional, real-time, and performance specifications have proven to be significant challenges in CPSS design.

The UML profile for Modeling and Analysis of Real-Time and Embedded Systems (MARTE) has been published as a standard modeling language for Real-Time Embedded Systems (RTES) [3]. To represent the continuous dynamic behavior of CPS [4], a set of differential equations has been introduced to MARTE statecharts. hMChart(Hybrid MARTE statecharts) [4], a MARTE statecharts extension based on hybrid timed automata, has been adopted for the design and analysis of safety-critical systems. hMChart can support continuous modeling beyond stochastic process modeling.

The associate editor coordinating the review of this manuscript and approving it for publication was Taehong Kim.

Several model checking-based approaches have been proposed to enhance the accuracy and performance of hMChart [5], [6]. However, most approaches focus on safety issues with only "true" or "false" outputs based on given properties. Limited models have the ability to model the stochastic behaviors of hMChart under human action variations and uncertainties in the physical environment. For example, a key issue for hMChart designers is determining "the probability of triggering a specified scenario within time t." Due to the nondeterministic execution and accumulated time variation, the bottleneck lies in the lack of hMChart-supported stochastic modelling and effective quantitative analysis methods. In order to bridge the gap between the stochastic modeling and quantitative analysis of CPSS, we propose an approach that describes and formally verifies the stochastic behavior of CPSS, denoted as the *Stochastic Hybrid MARTE Statechart (stohMChart)*. stohMChart is a novel framework based on *Statistical Model Checking (SMC)* [7] techniques that rely on the monitoring of random system simulation runs. The simulation results are analyzed using sequential hypothesis testing or Monte Carlo simulations in order to verify the satisfaction probability of specified properties. Unlike traditional formal verification methods that explore the whole state space, SMC techniques only inspect a limited number of simulation runs [8]. SMC is suitable for the approximate functional validation of complex CPSS designs. Thus, we employ the statistical model-checker UPPAAL-SMC [9], [10] as the engine for our approach. In order to overcome the limitations of the current techniques used to model CPSS, the major contributions are as follows:

- We propose a novel formal visual language, called ***stohMCharts***(Stochastic Hybrid MARTE Statecharts), to support modeling and analysis stochastic behaviors in uncertain environments of CPSS.
- We propose a set of mapping rules and a construction algorithm that can automatically transform the stohMChart into NSHA models. The tool is available at ***https://beiyanpiki.github.io/stohMCharts/.*** It supports hierarchical modeling, allowing for the decomposition of complex systems into smaller, more manageable subsystems.
- Our formal framework, which supports the quantitative performance analysis of *stohMChart*, is integrated with the statistical model checker UPPAAL-SMC.

The rest of this paper is organized as follows. We present the preliminaries such as the probability and measure theory, stochastic hybrid automata and probabilistic computation temporal logic in Section II. After introducing the syntax and semantics of stohMChart in Section III, Section IV presents the mapping rules used to transform stohMCharts to NSHA models. Based on a case study of two autonomous driving scenarios, Section V demonstrates that our approach can be effectively applied to the quantitative analysis of stohMChart designs. The rest of the work present the related work and conclude the paper.

II. PRELIMINARIES

A. NETWORKS OF STOCHASTIC HYBRID AUTOMATA

Our approach adopts the network of stochastic hybrid automata (NSHA) [11] to model the stochastic behaviors of CPSS. Compared to traditional timed automata (TA) [12], the clocks in SHAs vary in different locations. NSHA consists of a set of SHAs that can communicate with each other through shared variables and broadcast channels. The syntax of SHA is defined as a tuple $SHA = \{L, l_0, V, C, A, I, D(l), E_p\}$, where

- L is a finite set of locations,
- l_0 is the initial location,
- V is a finite set of continuous variables,
- C is a finite set of clocks,
- A is the set of actions,
- I is a set of invariants,
- $D(l)$ is a time delay function for each location $l \in L$,
- E_p is a finite set of transitions with probability $p \in [0, 1]$. $E_p \subseteq L \times A \times 2^C \times \psi \times L$ where ψ is a binary relationship on R^V

Networks of Stochastic Hybrid Automata We define NSHA as $SHA^j = \{L^j, l_0^j, V^j, C^j, A^j, I^j, D(l)^j, E_p^j\}$(j =1…n), they are composed into a closed network iff they have the same action set $A = A^j = A^k$ for all j, k.

B. PROBABILISTIC COMPUTATION TEMPORAL LOGIC

In recent years, many works use probabilistic temporal logic such as Computation Temporal Logic (CTL) and Linear Temporal Logic (LTL) to express abstractions and properties of CPS [13], [14]. While one downside of specifying properties in CTL or LTL [15] is that the properties of the system and environment have to be expressed deterministically. Probabilistic Computation Temporal Logic (PCTL) based on CTL, which is an expressive language that closes this gap by using probabilistic atomic predicates parameterized with a time-varying random variable drawn from a given distribution [16]. Our framework adopt PCTL to express safety constraints in uncertainty environment.

The State and path formulae of PCTL are

$$\begin{aligned} \Phi &::= true|a|\Phi \wedge \Phi|\neg\Phi|P_{\bowtie p}[\Psi] \\ \Psi &::= X\Phi|\Phi U^{\leq n}\Phi|\Phi U \Phi \end{aligned}$$

respectively, where a is an atomic proposition, $p \in [0, 1]$ is a probability bound, $\bowtie \in \{<, >, \leq, \geq\}$ and $n \in N$.

C. STATISTICAL MODEL CHECKING

Statistical model checking techniques have been widely investigated to evaluate uncertainty-aware designs. For example, Du et al. [17] use UPPAAL-SMC tool to evaluate energy-aware buildings with time uncertainty. Chen et al. [18] present a way to evaluate the task allocation and scheduling strategies with time and power variation information. Gu et al. [19] analyze quantitative timing of UML activity diagrams based on statistical model checking.

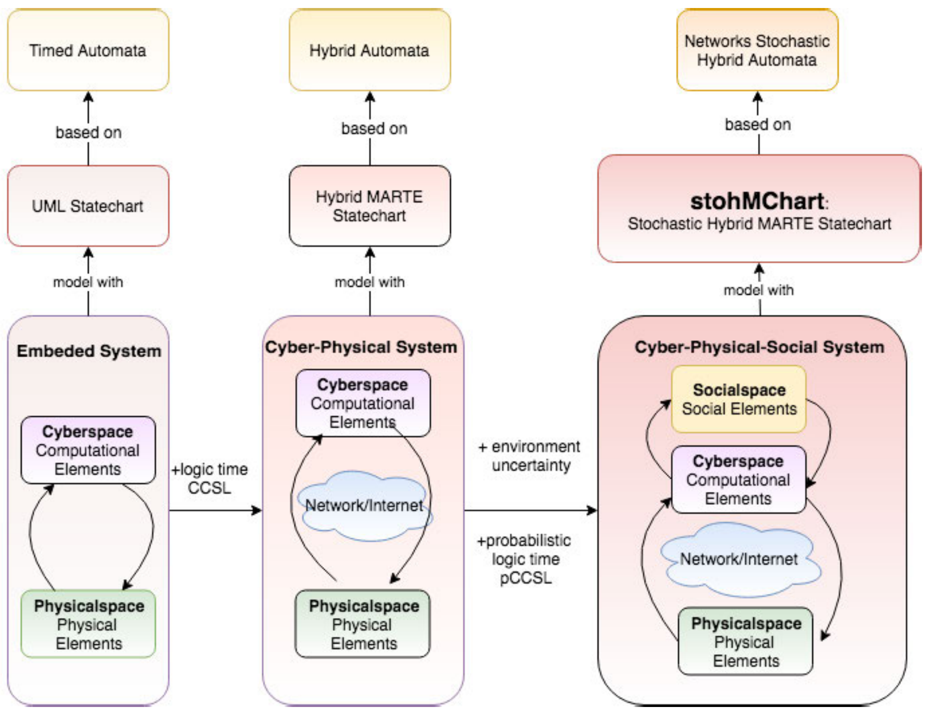

Timed Automata
based on
UML Statechart
model with
Embeded System
Cyberspace Computational Elements
Physicalspace Physical Elements
+logic time CCSL
Hybrid Automata
based on
Hybrid MARTE Statechart
model with
Cyber-Physical System
Cyberspace Computational Elements
Network/Internet
Physicalspace Physical Elements
+ environment uncertainty
+probabilistic logic time pCCSL
Networks Stochastic Hybrid Automata
based on
stohMChart: Stochastic Hybrid MARTE Statechart
model with
Cyber-Physical-Social System
Socialspace Social Elements
Cyberspace Computational Elements
Network/Internet
Physicalspace Physical Elements

FIGURE 1. The comparison of ES, CPS and CPSS.

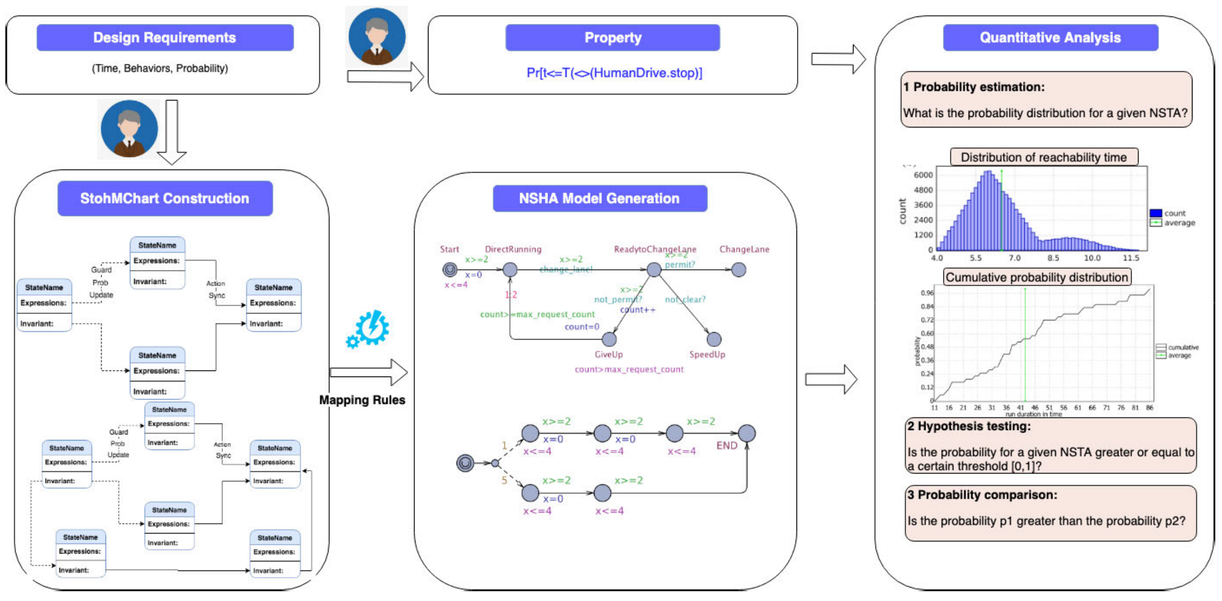

Design Requirements
(Time, Behaviors, Probability)
Property
Pr[t<=T(<>(HumanDrive.stop)]
Quantitative Analysis
1 Probability estimation:
What is the probability distribution for a given NSTA?
Distribution of reachability time
Cumulative probability distribution
2 Hypothesis testing:
Is the probability for a given NSTA greater or equal to a certain threshold [0,1]?
3 Probability comparison:
Is the probability p1 greater than the probability p2?
StohMChart Construction
StateName
Expressions:
Invariant:
Guard
Prob
Update
Action
Sync
Mapping Rules
NSHA Model Generation
Start
DirectRunning
ReadytoChangeLane
ChangeLane
GiveUp
SpeedUp
END

FIGURE 2. Workflow of proposed framework.

Through using the build-in function *random()*, we can define a large set of commonly used distributions instead of the uniform and exponential distributions which UPPAAL-SMC supports. During the checking of NSHA models, SMC simulates random runs which are bounded by either time, cost or some discrete steps. Upon a decision of an NSHA during the simulation, the transition with the shortest delay triggered, then all the continuous variables updated [7].

III. WORKFLOW OF PROPOSED FRAMEWORK

Fig.1 presents the basic concepts and relationship between the embedded systems, CPS and CPSS. The embedded systems include cyberspace and physical space and are modelled via timed automata [20], [21]. CPS extends embedded systems using logic time Clock Constraint Specification Language (CCSL) [22] defined in MARTE. CPSS extends CPS by considering social space, with human actions, uncertain environments and probabilistic logic time, probabilistic clock constraint specification language (pCCSL) [23] integrated into the system. The proposed Stochastic Hybrid MARTE Statechart (stohMChart) for CPSS analysis is based on Networks Stochastic Hybrid Automata [11].

Fig.2 presents the workflow of our proposed approach and the overview of our framework. Firstly, we model the CPSS using stohMChart, which extends hMChart by considering uncertainty information (e.g., human variation measurements and, action execution time). Following this, we transform the stohMChart model to NSHA based on the mapping rules demonstrated in Fig.3. In order to allow for the quantitative analysis of the stohMChart model via performance queries, we design specification properties as PCTL formulas. Once the NSHA models and performance query-based properties are ready, the framework employs the statistical model checker UPPAAL-SMC to quantitatively analyze CPSS stochastic behaviors.

A. THE SYNTAX OF stohMChart

1) VARIABLES AND EXPRESSIONS IN stohMChart

To describe the variables with the stochastic property, we enrich standard MARTE datatypes with probability variables as well as probability functions. Table 1 presents five types of variables in stohMChart.

TABLE 1. Variables in stohMChart.

Syntax	Type	Domian	Continuous behavior
bool	boolean var	{true, false}	d(x)=0
int	integers	Z	d(x)=0
real	static real var	R	d(x)=0
clock	clocks	R_0^+	d(x)=1
var	continuous var	R	according to invariants

Now we enrich MARTE expressions, as we can see from Table 1, stohMChart expressions are assigned by variables we introduced in Table 1. Different kinds of stohMChart variables assigned to expressions can generate different kinds of expressions. The expressions in stohMChart are classified into eight subsets based on their types and their potential to contain references to a variable's first derivative or subexpressions with nondeterministic values. These subsets are summarized in Table 2, which also illustrates the relationship between variables and expressions in stohMChart. For example, in Tabl. 2, the logical expression Lxp depends on the variables 'Bool Variables', 'Event Variables', and 'Clock Variables', so these variables are marked with a check mark '$\surd$' in their corresponding columns, while the other variables are marked with '$\times$'. In addition, the action expression Acxp depends on the variables Nact, Pact, Sact, Cact, silent action, error action, and break action, and this relationship is separately indicated in the lower right part of the table. The expressions in stohMChart can be represented by the following notation: $Axp \uplus Bxp \uplus uBxp \uplus Sxp \uplus Lxp \uplus Dxp \uplus Cxp \uplus Acxp$ where

1) Axp: assignment expressions such as $x + 3.2$, which evaluate to R and do not contain derivatives, nondeterminism and sampling.
2) Bxp: boolean expression usually used to express guard conditions, such as $a == 1$.
3) $uBxp$: uncertain boolean expressions, may be nondeterministic or contain references to derivatives, for example $d(x) <= 3$ represents the first derivative of x can not exceed 3, $Bxp \subseteq uBxp$.
4) Sxp: expressions that do not contain references to derivatives, but may be nondeterministic and use sampling, for example $y = x + Uniform(1, 0.2)$ where $Uniform(a, b)$ denotes sampling from the uniform distribution.
5) Lxp: logical expressions including logic operators like &&, ||, $\neg$ etc.
6) Dxp:differential expressions. The first derivative of a continuous variable v, denote as $d(v)$. Note that the $d(v)$ is only valid for continuous variables and cannot be used with clock variables. We also define a measure expression, $\mathbb{M} = Normal(v, \delta)$ The actual value measured by the controller is sampled according to a normal distribution with the actual value as mean v and a standard deviation of δ.
7) Cxp: clock constraint expression is introduced in [23]. Let c, d be two clocks, the set of constraints can be defined as follows: $Clk ::= true| \ c \geq n| \ n + c \geq d + m| \ \neg c| \ c \cong d$ where $c, d \in \mathcal{C}$ and $m, n \in \mathbb{N}$ $c \cong d ::= c \prec_p d \mid c \preceq_p d \mid c \sim d \mid c \bowtie d \mid c \sharp d$.
8) $Acxp$:Action expressions. There are several categories of actions in stohMChart. $A ::= Nact \uplus Pact \uplus Sact \uplus Cact \uplus \{\perp, bk, \tau\}$

    a) $Nact$: normal actions which occur without nondeterminism.
    b) $Pact$: probabilistic actions that occur based on probability $p, p \in [0, 1]$.
    c) $Sact$: stochastic action which follow the distribution $act \in Sact, \mathbb{M}_{act} : act \rightarrow Dist_{act}$ is a mapping function that specifies the distributions of the execution time of actions.
    d) $Cact$: cycle actions which occur based on a time cycle $act \in Cact, act(every \ \ t)$
    e) $\perp, bk, \tau$ represents is the error, break and silent action respectively.

2) DEFINITION OF stohMChart

A stohMChart is a tuple
$stohMChart = \{S, s_0, T, Cmd, A, X, Inv, \mathfrak{D}\}$

1) $S = \{s_0, s_1, \ldots, s_m\}$ is a set of states. A state s is a tuple(l, v, exp, h) where l denotes the location, v denotes the value of a variable, exp denotes the expressions, h denotes the hierarchy.
2) $s_0 \in L$ is the initial state.
3) $T \subseteq s \times Cmds \times \Sigma \times 2^{X \cup V} \times s'$

TABLE 2. Expressions and variables in stohMChart.

Category	Type	Derivatives	Nondeterministic	Sampling	Variables				
					Discrete Variables	Bool Variables	Event Variables	Continuous Variables	Clock Variables
Axp	R	×	×	×	√	×	×	×	×
Bxp	bool	×	×	×	√	×	×	×	×
uBxp	bool	√	×	√	×	√	×	×	×
Sxp	R	×	√	√	×	×	×	√	×
Lxp	bool	×	×	×	×	√	√	×	√
Dxp	R	√	×	√	×	×	×	√	×
Cxp	clocks	×	√	√	×	×	×	×	√
Acxp	bool	×	√	√	actions: Nact,Pact,Sact,Cact,silent action,error action,break action				

4) $Cmd = \{g_0, g_1, \ldots, g_p\}$ is a set of probabilistic guard commands of the form $g \rightarrow p_1 : u_1 + \ldots + p_m : u_m$ where
   - $g \subseteq L \times R^k$ is a guard, $k \in N^+$ is the dimension of the stohMChart, i.e. there are k variables(including clock variables) in the model.
   - For all $1 \leq i \leq m$, we get $p_i \geq 0$ and $\Sigma_{i=1}^m p_i = 1$.
   - The update function is defined as $\Lambda : (L \times R^k) \rightarrow 2^{L \times R^k}$ for $1 \leq i \leq m$.
5) $A = \{act_0, act_1, act_2, \ldots, act_n\}$ is a set of actions. We define a single action τ representing the passing of time.
6) X is a finite set of clocks constraints.
7) $Inv = \{i_0, i_1, \ldots, i_n\}$ is a set of invariants. $Inv : L \rightarrow Exp$ assigns a set of invariants L to each location.
8) $\mathfrak{D}$ is the delay function. $\mathfrak{D} : (l, v, e) \rightarrow \text{Normal}(\mu, \delta)$ | Exp(rate) |Uniform(a,b).

B. THE SEMANTICS OF stohMChart

The semantics of stohMChart can be interpreted by the stochastic transition system [24]. A stochastic transition system is a tuple $\langle \mathcal{S}, s_0, \mathbf{D}, \rightarrow, \rightarrow_{\mathbf{D}}, \rightarrow_{\Gamma} \rangle$, where

- $S \times \mathbf{D}$ is the set of states and s_0 is the initial state.
- $\mathbf{D}$ is the delay density function, which will perform the stochastic output with $\int \mathbf{D}(t)dt = 1$.
- $\rightarrow: \mathcal{S} \times \Sigma \times \mathcal{S}'$ is the normal transition function between two states of the stohMChart,where Σ represents a finite collection of input symbols.
- $\rightarrow_{\mathbf{D}}$: $\mathcal{S} \times \Sigma \times \mathbf{D} \times \mathcal{S}'$ is a *delay* transition, where $\mathbf{D}$ is the *delay* function in the transitions.
- $\rightarrow_{\Gamma}$: $\mathcal{S} \times \Sigma \times \Gamma \times \mathcal{S}'$ is a set of *output probability* transition with $\sum \Gamma(t)dt = 1$, where $\Gamma(t)$ is the probability in each transition.

The semantics of stohMCharts is defined as $[[S, s_0, T, Cmd, A, X, Inv, \mathfrak{D}]] = \langle \mathcal{S}, s_0, \mathbf{D}, \rightarrow, \rightarrow_{\mathfrak{D}}, \rightarrow_{\Gamma} \rangle$ where

- $S : L \times \mathbb{R}^{X \cup \mathfrak{C}} \times \mathfrak{D}$ with $\mathbb{R}^{X \cup \mathfrak{C}} \models Inv$ is the set of states.
- $s_0 : (l_0, v)$ with $l_0 \in L$, $v \in X \cup \mathfrak{C}$ and $v \models Inv_0$ is the initial state.
- $\mathfrak{D}$ is the delay density function, which will perform the stochastic output with $\int \mathfrak{D}(t)dt = 1$.
- $\Sigma : Cmds \times A \times \Lambda$ is a set of labeling function, where *Cmds* is the set of commands, A the set of synchronous actions between different stohMChart and Λ is the set of update functions.
- $T :\rightarrow \cup \rightarrow_{\mathbf{D}} \cup \rightarrow_{\Gamma}$ denotes the *normal* transitions, *delay* transitions and *probability* transitions in each transition.

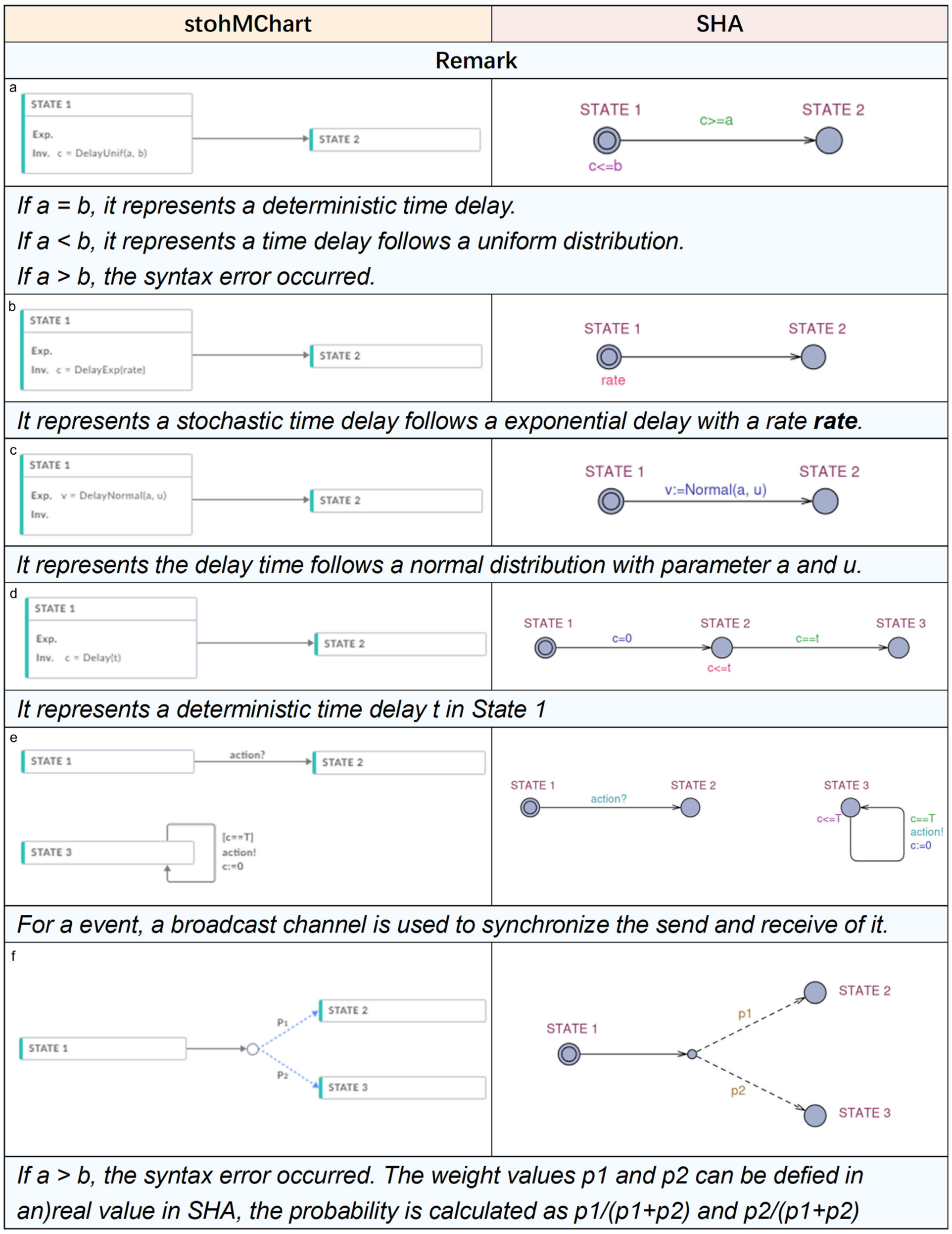
IV. TRANSFORMING stohMChart TO STOCHASTIC HYBRID AUTOMATA

A. MAPPING RULES

In Fig.3 it shows the mapping rules of the stohMChart and SHA in UPPAAL-SMC.

1) In Fig.3.a, it shows the **DelayUnif(a,b)** which means the time stay in State1 follows the uniform distribution with parameter a and b. We model it in UPPAAL-SMC with a clock c, set the invariant $c <= b$ in State1 and a transition guard $c >= a$.
2) In Fig.3.b, it shows the **DelayExp(rate)** which means the time stay in State1 is nondeterminism which follows the exponential distribution with parameter *rate*. In UPPAAL-SMC it just already encoded in the state with *Rate of Exponential*.
3) In Fig.3.c, it shows the **v**∼ **DelayNormal(a, u)** which means the continuous variable in State1 follows the normal distribution with two parameters: mean a and variance$u$. The UPPAAL-SMC hasn't supported the normal distribution yet, so we define in the function based on *random*() function that UPPAAL-SMC provides.
4) In Fig.3.d, it shows the **Delay(t)** which means the time stay in State1 is determinate with time t.
5) In Fig.3.e, it shows the action in the state of stohMChart. In UPPAAL-SMC, it is modeled as broadcast channel to synchronize.
6) In Fig.3.f, it shows the probabilistic transition with p_1 and p_2, in UPPAAL-SMC, the probability is calculated as $\frac{p_1}{p_1+p_2}$ and $\frac{p_2}{p_1+p_2}$ respectively.

The algorithm 1 presents the mapping mechanism from stoMChart to NSHA. Given a stochastic model $\mathcal{C}$ and the initial state s_0, the algorithm generates an NSHA model $\mathcal{U}$. The initial state s_0 is removed from the set of states S. For each expression in a state, the delay uniform distribution and the delay expression, a new state in NSHA is created. A new transition is created associated to the state. The guard, action,



If a = b, it represents a deterministic time delay.
If a < b, it represents a time delay follows a uniform distribution.
If a > b, the syntax error occurred.

*It represents a stochastic time delay follows a exponential delay with a rate **rate**.*

It represents the delay time follows a normal distribution with parameter a and u.

It represents a deterministic time delay t in State 1

For a event, a broadcast channel is used to synchronize the send and receive of it.

If a > b, the syntax error occurred. The weight values p1 and p2 can be defied in an)real value in SHA, the probability is calculated as p1/(p1+p2) and p2/(p1+p2)

FIGURE 3. The mapping rules of the stohMChart and SHA.

and probability weight are transformed to the corresponding part in the edges of NSHA. More importantly, the hierarchy of the states in stoMChart is translated to another template in NSHA, and the synchronous actions are inserted.

V. CASE STUDY

To illustrate our approach, we present a case study based on the interaction between an autonomous vehicle and a human-driven motorcycle (represented by the blue and red cars in Fig. 4, respectively) on a shared road. In the upper part of Fig. 4, Scene 1 demonstrates the case where the human-driven motorcycle slows down to make room for the autonomous vehicle to pass and change lanes, which ultimately succeeds. In the lower part of Fig. 4, Scene 2 illustrates the case where the human-driven motorcycle refuses to allow the autonomous vehicle to change lanes, and as a result, the autonomous vehicle either abandons the lane change or performs an emergency braking maneuver. We define this problem as a simple CPSS, which can be modeled using the stohMChart framework.

A. DEFINE STOCHASTIC BEHAVIORS AND EXPRESSIONS

1) ABSTRACT DRIVING STYLE

It clears that not all humans behave the same way. In [25], it provides a survey on driving style characterization and recognition revising a variety of algorithms, with particular emphasis on machine learning approaches based on current

Algorithm 1 Mapping stohMChart to NSHA

```
Require:
    The stohMChart model $\mathcal{C}$ with the initial state $s_0$;
Ensure:
    The model of NSHA, $\mathcal{U}$;
    The initial state of $\mathcal{U}$, $u_0$;
1:  WHILE ($S$ is not empty)
2:  $S \leftarrow S \backslash s_0$;
3:  $t_s$ the transition links to the state $s$, $s \in S$;
4:  create the initial state of $\mathcal{U}$, $u_0$;
5:  create new transition links to the state $u_0$;
6:  SWITCH(Type of Delay Expression in State)
7:  Case: stohMChart.State1.Inv.DelayUnif(a,b); then
        NSHA.$State1.invariant[c \le b]$;
        NSHA.$Edge_{State1 \to State2}.guard[c \ge a]$;
8:  Case: stohMChart.State1.Inv.DelayExp(rate); then
        NSHA.$Edge_{State1 \to State2}.guard[c \ge a]$;
        NSHA.$State1.rate\ of\ exponential[rate]$;
9:  Case: stohMChart.State1.Exp.v=Normal(a,u)
      NSHA.$Edge_{State1 \to State2}.update.[v = Norm(a, u)]$;
10: Case: stohMChart.State1.Inv.Delay(t); then
        Generate two new states $State2$ and $State3$;
        Generate two new edges $Edge_{State1 \to State2}$ and
            $Edge_{State1 \to State2}$
        NSHA.$Edge_{State1 \to State2}.update\ [c = 0]$;
        NSHA.$State2.inv[c \le t]$;
        NSHA.$Edge_{State2 \to State3}.guard\ [c == t]$;
11: Case: stohMChart.Transition.Action[action?]
        stohMChart.Transition.Action[action!]; then
      NSHA.$Edge_{State1 \to State2}.sync\ [action?]$;
      NSHA.$Edge_{State3 \to State3}.sync\ [action!]$;
    END SWITCH
12: SWITCH(Type of Transition)
13: Case: stohMChart.Transition.prob=p1
        stohMChart.Transition.prob=p2; then
      NSHA.$Edge_{State1 \to State2}.probabilityweight\ [p1]$;
      NSHA.$Edge_{State1 \to State3}.probabilityweight\ [p2]$;
14: construct new state with new transition with new parameters;
15: construct new NSHA.
    END SWITCH
    END WHILE
```

and future trends. To identify different driving style, three driving style are defined in [26] and [27],

- **Aggressive**: aggressive drivers drive with sharp and abrupt acceleration and deceleration, aiming at dynamic vehicle performance, and increased the likelihood of accidents [28].
- **Conservative**: Conservative drivers often exhibit mild operational behaviors with small amplitudes and low-frequency actions on the steering wheel, accelerator, and brake pedal [25].
- **Moderate**: Moderate drivers are positioned between the above two. They would like to balance multiple performances, such as dynamic vehicle performance, ride comfort, and energy efficiency [28].

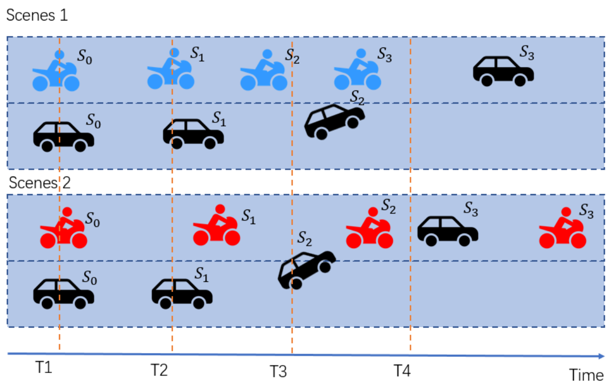


FIGURE 4. A scenario of autonomous car and human-driven motorcycle.

2) DEFINE EXPRESSIONS

The model presented in this study represents a car. In Fig.5, we used our developed tool to model its behavior on the road (which can be accessed through our public GitHub repository:https://github.com/beiyanpiki/stohMCharts). The pink border represents three sub-models, STOP, STRAIGHT, and CHANGELANE, which respectively represent the car's braking and stopping behavior, straight driving behavior, and lane-changing behavior.

Specifically, in the STRAIGHT sub-model, the car can choose to accelerate at a more aggressive speed or decelerate at a more conservative speed while driving at a constant speed. When encountering dangerous situations (represented by risk_1 and risk_3, which represent different risk levels), the car will choose to enter either the CHANGE_LANE state or the STOP state.

In the CHANGE_LANE sub-model, the car will choose to either continue driving straight in the STRAIGHT state or abort the lane-changing operation based on the driving styles of surrounding cars and the driver. Specifically, before choosing to change lanes, if the driver has an aggressive driving style, the car will choose to accelerate until the lane change is successful. If the driver has a conservative driving style, the car will decelerate and abandon the lane change, transitioning into the STRAIGHT or EMERGENCYBREAK state.

B. MAPPING TO UPPAAL-SMC MODEL

After building the stohMCharts, we use the mapping rules in Fig.3 to transform the model to NSHA in UPPAAL-SMC. The constructed NSHA model consists of four sub-templates, namely the composite template **Composite**, the change lane template **ChangeLane**, the Straight travel template **Straight-Driving**,and the environment risk template **EnvRisk**. Therefore, the NSHA model of UPPAAL-SMC can be expressed as:

$$\begin{aligned} DrivingModel &= Composite \cup ChangeLane \\ &\cup \quad StraightDriving \cup EnvRisk \end{aligned}$$

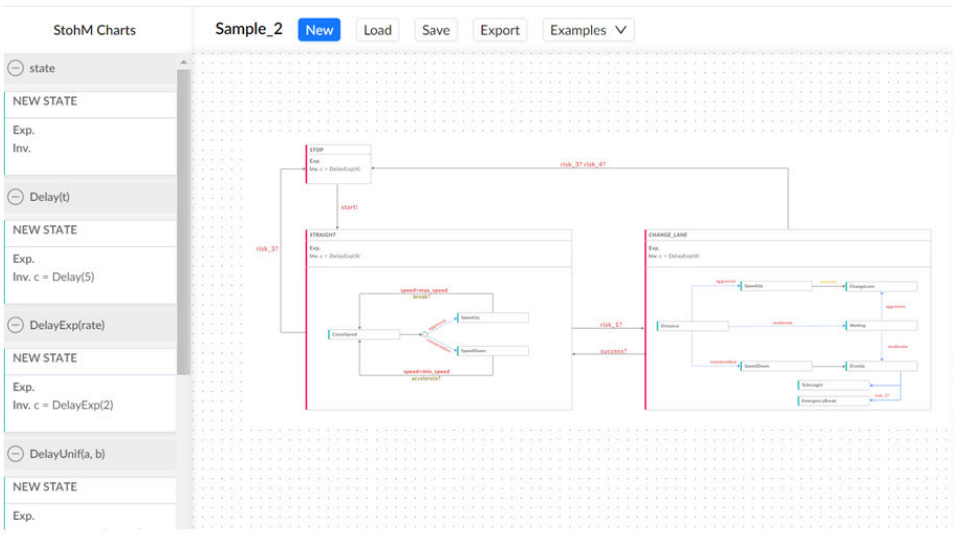


FIGURE 5. The stohMchart model of autonomous vehicle.

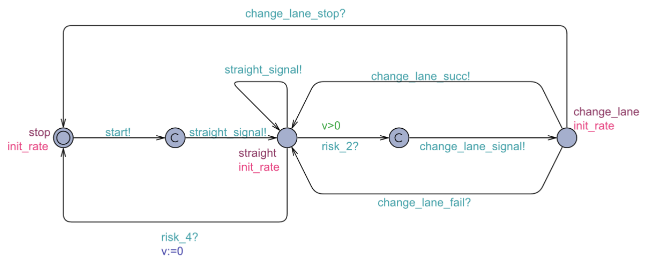


FIGURE 6. The UPPAAL-SMC template: the composite layer.

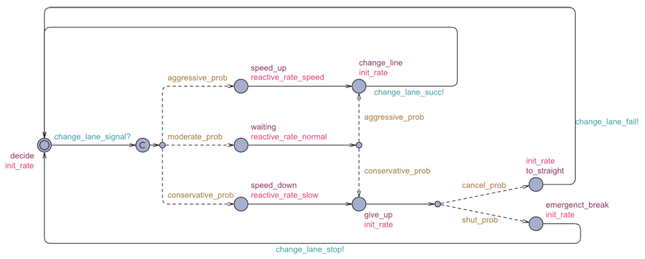


FIGURE 8. The UPPAAL-SMC template: the scenario of straight driving.

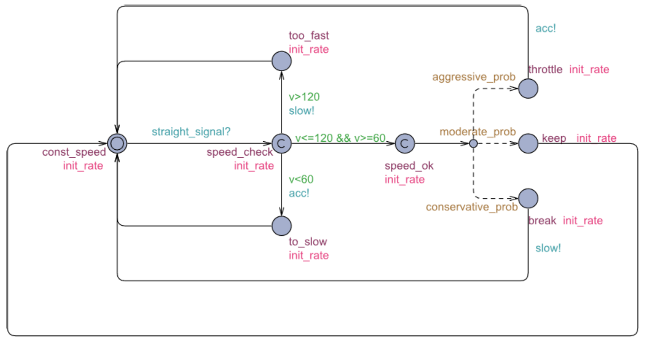


FIGURE 7. The UPPAAL-SMC template: the scenario of change lane.

Fig.6 shows the composite layer of the UPPAAL-SMC template, The straight line driving template is shown in Fig.7, The template of the lane change is shown in Fig.8. Due to the space limit, we only show three template here, for more template, please refer https://github.com/beiyanpiki/stohMCharts

C. DEFINE QUERIES

We considered four model queries to quantitatively analyze how scenarios are affected by uncertain factors, such as different driving styles.

$$
\begin{aligned}
&Pr[t <= 180](<> v >= 100)\\
&Pr[t <= 180](<> v >= 80)\\
&Pr[t <= 180](<> v >= 60)
\end{aligned} \tag{1}
$$

The query 1 means the probability that the speed of a car exceeds a certain value within 180 units of time. In our experiment, we set three speed values, 60 for slow speed, 80 for *normalspeed*, and 100 for *fasterspeed*.

$$
\begin{aligned}
&Pr[t <= 180](<> Straight.too\_fast)\\
&Pr[t <= 180](<> Straight.throttle)\\
&Pr[t <= 180](<> Straight.keep)\\
&Pr[t <= 180](<> Straight.break)
\end{aligned} \tag{2}
$$

The query 2 means the probability that the car will take certain actions within 180 units of time, among which *too_fast* indicates overspeed, while *throttle keep break* indicates throttle, constant speed, and brake, respectively.

$$
\begin{aligned}
&Pr[t <= 180](<> ChangeLane.emergency\_break)\\
&Pr[t <= 180](<> ChangeLane.change\_line)
\end{aligned} \tag{3}
$$

The query 3 means the probability that a car will overtake and enter different states within 180 time units.

$$Pr[t <= 1000](<> Composite.straight)$$
$$Pr[t <= 1000](<> Composite.change\_lane) \qquad (4)$$

The query 4 means the probability of a car in different states within 1000 time units, namely, straight line driving and lane changing.

D. EXPERIMENT RESULTS AND ANALYSIS

1) RESULTS OF QUERY 1

Query 1 describes the speed profile of the car during normal operation, where higher speeds indicate a more aggressive driving style of the car's driver. Column 2 of Table 3 displays the probability distribution of vehicle speed exceeding a certain value over 180 time units. The values of these three speeds are 100, 80, and 60, respectively. The third column of the table shows the time spent on each of the three properties in verification query 1. The fourth column shows the number of states visited and explored during the verification of each property. This indicates that within 180 units of time, the probability of the car's speed being between 60 and 80 is relatively high, while the probability of the speed being greater than 100 is relatively low.

We have plotted Fig.9 to illustrate the cumulative probability confidence intervals for a vehicle's speed exceeding 80 within 180 unit times, from an academic perspective. The horizontal axis represents time, while the vertical axis represents probability. The relevant parameters are listed below the graph.

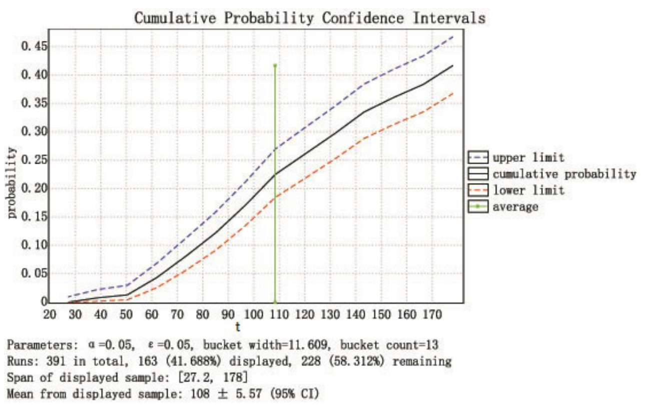
2) RESULTS OF QUERY 2

Query 2 demonstrates the operational behavior of a vehicle on a straight road within a unit time. The column *too_fast* indicates the vehicle's speed status, where a higher value indcates a higher likelihood of overspeeding. In addition, *throttle*, *keep*, and *break* represent the probabilities of the vehicle accelerating, maintaining its current speed, and braking, respectively. A higher probability for both "throttle" and "break" indicates a more complex road situation that requires frequent adjustments to the vehicle's speed.Column 2 of Table 4 shows the results of probability distribution of vehicle overspeed, throttle, constant speed driving, and braking actions within 180 unit times. The third and fourth columns of the table respectively describe the time taken for validating the properties and the number of states accessed and explored during the verification process.

We have plotted Fig.10, which illustrates the cumulative probability confidence intervals of a vehicle's throttle within 180 unit times while driving straight. The relevant parameters are listed below the graph.

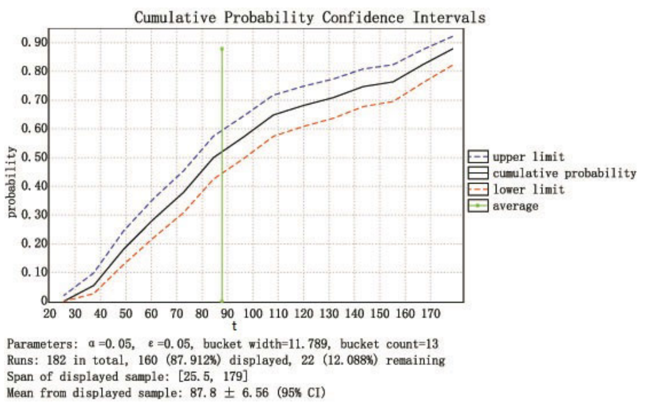
3) RESULTS OF QUERY 3

Query 3 describes the situations in which a car reaches different states when changing lanes. *emergency_break*



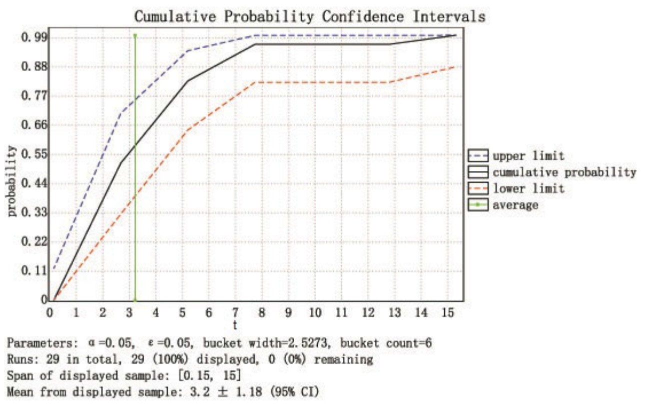
```
Parameters: α=0.05, ε=0.05, bucket width=11.609, bucket count=13
Runs: 391 in total, 163 (41.688%) displayed, 228 (58.312%) remaining
Span of displayed sample: [27.2, 178]
Mean from displayed sample: 108 ± 5.57 (95% CI)
```

FIGURE 9. The verification results of property Pr[t<=180](<> v>=80).



```
Parameters: α=0.05, ε=0.05, bucket width=11.789, bucket count=13
Runs: 182 in total, 160 (87.912%) displayed, 22 (12.088%) remaining
Span of displayed sample: [25.5, 179]
Mean from displayed sample: 87.8 ± 6.56 (95% CI)
```

FIGURE 10. The verification results of property Pr[t<=180](<> Straight.throttle).



```
Parameters: α=0.05, ε=0.05, bucket width=2.5273, bucket count=6
Runs: 29 in total, 29 (100%) displayed, 0 (0%) remaining
Span of displayed sample: [0.15, 15]
Mean from displayed sample: 3.2 ± 1.18 (95% CI)
```

FIGURE 11. The verification results of property Pr[t<=1000](<> Composite.straight).

represents a failed attempt to change lanes, where the car suddenly brakes, while *change_line* represents a successful lane change. The second column of Table 5 shows the probability distribution of a vehicle's emergency braking and successful lane change within 180 unit times. It can be observed that the values of these two probability distributions are equal. The third and fourth columns of the table respectively describe

TABLE 3. Verification results of vehicles at different speeds.

Properties	Result	Validation Time	States explored
Pr[t<=180] (<> v>=100)	[0.003307, 0.102528]	0.002 s	3064
Pr[t<=180] (<> v>=80)	[0.34802, 0.447984]	0.066 s	863
Pr[t<=180](<> v>=60)	[0.9000628, 1]	0.011 s	9

TABLE 4. Verification results of the vehicle under different actions.

Properties	Result	Validation Time	States explored
Pr[t<=180](<> Straight.too_fast)	[0, 0.099372]	0.016 s	3322
Pr[t<=180](<> Straight.throttle)	[0.818246, 0.918241]	0.003 s	9
Pr[t<=180](<> Straight.break)	[0.900628]	0.005 s	863

TABLE 5. The probability of a car making emergency braking and lane changes within 180 time units.

Properties	Result	Validation Time	States explored
Pr[t<=180](<>ChangeLane.emergency_break)	[0, 0.099372]	0.009 s	43
Pr[t<=180](<>ChangeLane.change_line)	[0, 0.099372]	0.015 s	33

TABLE 6. Verification results of vehicles in different driving states.

Properties	Result	Validation Time	States explored
Pr[t<=1000](<>Composite.straight)	[0.900628, 1]	0.023 s	5
Pr[t<=1000](<>Composite.change_lane)	[0, 0.099372]	0.011 s	20

the time taken for validating the properties and the number of states accessed and explored during the verification process.

4) RESULTS OF QUERY 4

Query 4 represents the probabilities of a vehicle being in the state of driving straight and changing lanes from a global perspective. The second column of Table 6 shows the probability distribution of the vehicle being in either a straight driving or a lane-changing state within 1000 unit time. The probability of the vehicle being in a straight driving state is significantly higher than that of being in a lane-changing state, indicating that the vehicle spends most of its time driving normally with few overtaking situations.

We have plotted Fig.11, which illustrates the cumulative probability confidence intervals of a vehicle's straight-line driving within 1000 unit times. The relevant parameters are listed below the graph.

VI. RELATED WORK

The modeling and analysis of CPSS is both multi-faceted and complex, and has been the subject of extensive research in the fields of model driven architecture and model checking. In the following, we briefly review several approaches employed to model & analyze CPSS. Statecharts, a visual language initially introduced by Harel in the late 1980s [29], has become a popular means of specifying the behavior of embedded systems. In [30], the authors present a formal semantics for UML statecharts via model transition systems. However, the authors fail to consider the stochastic property of the system. Moreover, [31] expand the semantics proposed in [32] with SCCharts [31] for specifying safety-critical reactive systems. In particular, SCCharts uses statechart notation and provides determinate concurrency based on a synchronous computation model. SCCharts can effectively model CPS [33], however, it does not consider without considering human behaviors. With respect to CPSS models, a domain ontology is proposed in [34], while [35], introduces a framework for command and control self-synchronization. However, both studies fail to provide a formal definition of key CPSS components.

The U-Test European Horizon 2020 project focuses on standardizing uncertainty modeling at OMG, playing a key role in current research on MARTE uncertainty models. More information about the initiative of Precise Semantics for Uncertainty Modeling (PSUM) is defined in [36]. Furthermore, [37] extends the Restricted Use Case Modeling (RUCM) methodology and its supporting tools to specify uncertainty as part of the system requirements.

Contributing to the progress in CPSS analysis and verification frameworks, Gu et al. [19] analyze the quantitative timing of UML activity diagrams via statistical models. Results provide a complete workflow for the stochastic modelling analysis of action executions.

Soudjani et al. [38] introduce a MATLAB-coded procedure to analyze stochastic modeling. In particular, the authors focus on modelling stochastic hybrid systems (SHS), and establish a framework to generate abstractions for uncountable-state discrete-time stochastic processes for single discrete mode and finite actions SHS models. Their algorithm also verifies reachability-like properties and corresponding policy synthesis, and consequently targets a class of SHS models that only depend on discrete time.

The work of [39] makes similar assumptions in order to improve the FAUST^2 tool by simplifying the input model description via sparse matrices for the manipulation of transition probabilities and by reducing the computational time required to generate abstractions. In [40], the authors present a method for the statistical verification of quantitative properties over a partially unknown system with actions by employing a parametric Markov decision process (pMDP) model [41]. The work presented in the current paper shares several common points with the aforementioned literature. For example, as in [19] and related studies, we also use statistical methods to analysis and verify the quantitative properties of CPSS.

VII. CONCLUSION

We proposed a formal visual language stohMChart, which support uncertain and hierarchical modeling for CPSS. Based on the stohMChart modeling language, mapping rules, and algorithms, we developed a tool to automatically convert stohMChart to NSHA and verified it by UPPAAL-SMC engine. We also present a case study based on the interaction between an autonomous vehicle and a human-driven motorcycle on a shared road to demonstrate our approach.

Our future work will extend our tool with a AI-based submodule to learn human uncertainty in CPSS. Furthermore, the development of an algorithm for the automatic generation of the UPPAAL-SMC model from stohMCharts could have enhanced the applicability of the proposed approach.

REFERENCES

[1] G.-G. Wang, X. Cai, Z. Cui, G. Min, and J. Chen, "High performance computing for cyber physical social systems by using evolutionary multi-objective optimization algorithm," *IEEE Trans. Emerg. Topics Comput.*, vol. 8, no. 1, pp. 20–30, Jan./Mar. 2017.

[2] M. Gharib, A. Ceccarelli, P. Lollini, and A. Bondavalli, "A cyber–physical–social approach for engineering functional safety requirements for automotive systems," *J. Syst. Softw.*, vol. 189, Jul. 2022, Art. no. 111310.

[3] S. Graf, S. Gérard, O. Haugen, I. Ober, and B. Selic, "Modeling and analysis of real-time and embedded systems," in *Proc. Int. Conf. Model Driven Eng. Lang. Syst.* Berlin, Germany: Springer, 2005, pp. 58–66.

[4] J. Liu, Z. Liu, J. He, F. Mallet, and Z. Ding, "Hybrid MARTE statecharts," *Frontiers Comput. Sci.*, vol. 7, no. 1, pp. 95–108, Feb. 2013.

[5] D. Drusinsky, *Modeling and Verification Using UML Statecharts: A Working Guide to Reactive System Design, Runtime Monitoring and Execution-based Model Checking*. Amsterdam, The Netherlands: Elsevier, 2011.

[6] V. A. D. S. Júnior and F. E. C. D. Silva, "From statecharts into model checking: A hierarchy-based translation and specification patterns properties to generate test cases," in *Proc. 2nd Brazilian Symp. Syst. Automated Softw. Test.*, Sep. 2017, pp. 1–10.

[7] A. David, K. G. Larsen, A. Legay, M. Mikučionis, and D. B. Poulsen, "Uppaal SMC tutorial," *Int. J. Softw. Tools Technol. Transf.*, vol. 17, no. 4, pp. 397–415, Aug. 2015.

[8] J. Wang, Z. Huang, Y. Zhu, and G. Shen, "Statistical model checking for stochastic and hybrid autonomous driving based on spatio-clock constraints," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 32, no. 4, pp. 553–582, Apr. 2022.

[9] E.-Y. Kang, D. Mu, and L. Huang, "UPPAAL-SMC," in *Proc. 14th Int. Conf. Integr. Formal Methods (IFM)*, vol. 11023. Maynooth, Ireland: Springer, Sep. 2018, p. 236.

[10] K. G. Larsen and A. Legay, "Statistical model checking the 2018 edition!" in *Proc. Int. Symp. Leveraging Appl. Formal Methods*. New York, NY, USA: Springer, 2018, pp. 261–270.

[11] A. David, K. G. Larsen, A. Legay, and D. B. Poulsen, "Statistical model checking of dynamic networks of stochastic hybrid automata," *Electron. Commun. EASST*, vol. 66, no. 9, pp. 122–136, 2014.

[12] J. Bendík, A. Sencan, E. A. Gol, and I. Černá, "Timed automata robustness analysis via model checking," 2021, *arXiv:2108.08018*.

[13] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Berlin, Germany: Springer, 2004, pp. 152–166.

[14] H. A. Bardh Hoxha and G. Fainekos, "Benchmarks for temporal logic requirements for automotive systems," *Proc. Appl. Verification Continuous Hybrid Syst.*, 2014, pp. 25–30.

[15] L. M. Tabajara and M. Y. Vardi, "Linear temporal logic—From infinite to finite horizon," in *Automated Technology for Verification and Analysis*, Z. Hou and V. Ganesh, Eds. Cham, Switzerland: Springer, 2021, pp. 3–12.

[16] J. V. Deshmukh, P. Kyriakis, and P. Bogdan, "Stochastic temporal logic abstractions: Challenges and opportunities," in *Formal Modeling and Analysis of Timed Systems*, D. N. Jansen and P. Prabhakar, Eds. Cham, Switzerland: Springer, 2018, pp. 3–16.

[17] D. Du, M. Chen, X. Liu, and Y. Yang, "A novel quantitative evaluation approach for software project schedules using statistical model checking," in *Proc. 36th Int. Conf. Softw. Eng.*, May 2014, pp. 476–479.

[18] M. Chen, D. Yue, X. Qin, X. Fu, and P. Mishra, "Variation-aware evaluation of MPSoC task allocation and scheduling strategies using statistical model checking," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, 2015, pp. 199–204.

[19] F. Gu, X. Zhang, M. Chen, D. Große, and R. Drechsler, "Quantitative timing analysis of UML activity diagrams using statistical model checking," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*. San Jose, CA, USA: EDA Consortium, 2016, pp. 780–785.

[20] Y. Yang, Y. Jiang, M. Gu, and J. Sun, "Verifying simulink stateflow model: Timed automata approach," in *Proc. 31st IEEE/ACM Int. Conf. Automated Softw. Eng.*, Aug. 2016, pp. 852–857.

[21] M. Z. Golonka, "Comparative analysis of methods and tools for formal modelling and verification for embedded systems. Probabilistic approach," in *Proc. 28th Int. Conf. Mixed Design Integr. Circuits Syst.*, Jun. 2021, pp. 265–273.

[22] M. Zhang, F. Song, F. Mallet, and X. Chen, "SMT-based bounded schedulability analysis of the clock constraint specification language," in *Fundamental Approaches to Software Engineering* (Lecture Notes in Computer Science), R. Hähnle and W. van der Aalst, Eds. Cham, Switzerland: Springer, 2019, pp. 61–78.

[23] D. Du, P. Huang, K. Jiang, and F. Mallet, "PCSSL: A stochastic extension to MARTE/CCSL for modeling uncertainty in cyber physical systems," *Sci. Comput. Program.*, vol. 166, pp. 71–88, Nov. 2018.

[24] L. de Alfaro, "Stochastic transition systems," in *CONCUR'98 Concurrency Theory*. Berlin, Germany: Springer, 1998, pp. 423–438.

[25] C. M. Martinez, M. Heucke, F.-Y. Wang, B. Gao, and D. Cao, "Driving style recognition for intelligent vehicle control and advanced driver assistance: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 3, pp. 666–676, Mar. 2018.

[26] C. Lv, X. Hu, A. Sangiovanni-Vincentelli, Y. Li, C. M. Martinez, and D. Cao, "Driving-style-based codesign optimization of an automated electric vehicle: A cyber-physical system approach," *IEEE Trans. Ind. Electron.*, vol. 66, no. 4, pp. 2965–2975, Apr. 2019.

[27] Z. Deng, D. Chu, C. Wu, Y. He, and J. Cui, "Curve safe speed model considering driving style based on driver behaviour questionnaire," *Transp. Res. F, Traffic Psychol. Behav.*, vol. 65, pp. 536–547, Aug. 2019.

[28] C. Lv, Y. Liu, X. Hu, H. Guo, D. Cao, and F.-Y. Wang, "Simultaneous observation of hybrid states for cyber-physical systems: A case study of electric vehicle powertrain," *IEEE Trans. Cybern.*, vol. 48, no. 8, pp. 2357–2367, Aug. 2018.

[29] D. Harel, "Statecharts: A visual formalism for complex systems," *Sci. Comput. Program.*, vol. 8, no. 3, pp. 231–274, 1987.

[30] D. Varró, "A formal semantics of UML statecharts by model transition systems," in *Proc. Int. Conf. Graph Transformation*. Berlin, Germany: Springer, 2002, pp. 378–392.

[31] L. Grimm, S. Smyth, A. Schulz-Rosengarten, R. V. Hanxleden, and M. Pouzet, "From Lustre to graphical models and SCCharts," *ACM Trans. Embedded Comput. Syst.*, vol. 21, pp. 1–8, Jul. 2022.

[32] C. André, "Semantics of SyncCharts," I3S Lab., Sophia-Antipolis, France, Tech. Rep. ISRN I3S/RR–2003–24–FR, 2003.

[33] J. C. Jensen, D. H. Chang, and E. A. Lee, "A model-based design methodology for cyber-physical systems," in *Proc. 7th Int. Wireless Commun. Mobile Comput. Conf.*, Jul. 2011, pp. 1666–1671.

[34] G. Xiong, F. Zhu, X. Liu, X. Dong, W. Huang, S. Chen, and K. Zhao, "Cyber-physical-social system in intelligent transportation," *IEEE/CAA J. Autom. Sinica*, vol. 2, no. 3, pp. 320–333, Jul. 2015.
[35] Z. Liu, D.-S. Yang, D. Wen, W.-M. Zhang, and W. Mao, "Cyber-physical-social systems for command and control," *IEEE Intell. Syst.*, vol. 26, no. 4, pp. 92–96, Jul./Aug. 2011.
[36] M. Zhang, S. Ali, T. Yue, and R. Norgre, "Uncertainty-wise evolution of test ready models," *Inf. Softw. Technol.*, vol. 87, pp. 140–159, Jul. 2017.
[37] M. Zhang, S. Ali, T. Yue, R. Norgren, and O. Okariz, "Uncertainty-wise cyber-physical system test modeling," *Softw. Syst. Model.*, vol. 18, no. 2, pp. 1379–1418, Apr. 2019.
[38] S. Soudjani, C. Gevaerts, and A. Abate, "Faust^2: Formal abstractions of uncountable-state stochastic processes," in *Proc. TACAS*, 2015, pp. 1–15.
[39] N. Cauchi, K. Degiorgio, and A. Abate, "StocHy: Automated verification and synthesis of stochastic processes," 2019, *arXiv:1901.10287*.
[40] E. Polgreen, V. B. Wijesuriya, S. Haesaert, and A. Abate, "Automated experiment design for data-efficient verification of parametric Markov decision processes," in *Proc. Int. Conf. Quant. Eval. Syst.*, 2017, vol. 3, no. 4, pp. 259–274.
[41] S. Junges, J.-P. Katoen, G. A. Pérez, and T. Winkler, "The complexity of reachability in parametric Markov decision processes," *J. Comput. Syst. Sci.*, vol. 119, pp. 183–210, Aug. 2021.

DONGDONG AN received the B.S. and Ph.D. degrees in software engineering from East China Normal University, Shanghai, China, in 2013 and 2020, respectively. She is currently a Lecturer with the Department of Computer Science and Technology, Shanghai Normal University, Shanghai. Her current research interests include model-driven architecture, machine learning, formal methods, and statistical model checking techniques.

ZONGXU PAN received the B.S. degree in computer science and technology from the Zhengzhou University of Light Industry, China, in 2022. He is currently pursuing the master's degree with the College of Information, Mechanical and Electrical Engineering, Shanghai Normal University, China. His current research interests include formal methods and machine learning.

XIN GAO received the bachelor's degree in computer science and technology from the Hangzhou College of Commerce, Zhejiang Gongshang University, Zhejiang, China, in 2021. He is currently pursuing the master's degree with the College of Information, Mechanical and Electrical Engineering, Shanghai Normal University, Shanghai, China. His main research interests include adversarial machine learning and neural network verification.

SHUANG LI received the Ph.D. degree from the School of Computer Engineering and Science, Shanghai University, Shanghai, in 2019. She is currently a Lecturer with the College of Information, Mechanical and Electrical Engineering, Shanghai Normal University, China. Her research interests include parallel computing, swarm intelligence systems, and ternary optical computer.

LING YIN received the B.S. degree in software engineering and the Ph.D. degree in computer technology from East China Normal University, China, in 2008 and 2016, respectively. She is currently a Lecturer with the School of Electronic and Electrical Engineering, Shanghai University of Engineering Science, China. Her research interests include deep learning, time series analysis, and software engineering with formal methods.

TENGFEI LI received the Ph.D. degree from the Software Engineering Institute, East China Normal University, in 2021. He is a Postdoctoral Researcher with East China Normal University and CASCO Signal Ltd. His research interests include formal verification, safety-critical cyber-physical systems, and spatio-temporal logics.