Analyzing Cyber-Physical Systems with Learning Enabled Components using Hybrid Predicate Transition Nets

Xudong He
Knight Foundation School of Computing and Information Sciences
Florida International University
Miami, USA
hex@cs.fiu.edu

***Abstract* — Cyber-physical systems (CPSs) are ubiquitous and are becoming increasingly important in the functioning of our society. CPSs have complex discrete and continuous behaviors. In recent years, learning enabled components (LECs) built using machine learning approaches are increasingly used in CPSs to perform autonomous tasks to deal with uncertain and unfamiliar environments. CPSs with LECs are even more difficult to develop. We have developed a methodology for formally modeling and analyzing CPSs with LECs. Hybrid predicate transition nets (HPrTNs) are used as the underlying formal method to model CPSs with LECs and their training through their simulation capability. In this paper, we present our new analysis methodology for CPSs with LECs consisting of three complementary techniques, including a testing technique based on HPrTN simulation capability, a simulation guided barrier certificate technique, and a SMT based bounded model checking technique. The above analysis methodology is partially supported by a tool chain and is demonstrated through an example.**

Keywords — cyber-physical systems; learning enabled components; formal methods; hybrid predicate transition nets; barrier certificate; bounded model checking

I. INTRODUCTION

Cyber-physical systems (CPSs) are ubiquitous and are becoming increasingly important in the functioning of our society. CPSs are hybrid systems that contain physical devices having continuous dynamics and computational control processes with discrete behaviors. These systems are extremely difficult to build and error-prone. In recent years, CPSs have started to use learning enabled components (LECs) as part of the control loop for performing various perception-based autonomy tasks. These data-driven components are trained using machine learning (ML) approaches such as deep learning – deep neural nets (DNNs) and reinforcement learning (RL) [17]. These approaches have provided CPSs the capability to continuously learn and work in uncertain and unfamiliar environments. Although many ML techniques have been developed in the past few decades and tremendous progresses have been made in the last decade, there is very little understanding of the properties of these data-driven models built using ML. Research on the formal analysis of these data-driven models has just emerged in recent years. LECs have added additional dimensions of difficulties to those of CPSs.

We have developed a methodology for modeling and analyzing CPSs with LECs, which contains the following new results: (1) A method for modeling deep neural nets (DNNs) using hybrid predicate transition nets (HPrTNs), (2) An reinforcement learning (RL) technique to train DNNs with an environment (plant) using HPrTNs, (3) A Simplex architecture to integrate advanced controller (a trained DNN) with a baseline controller defined using ordinary differential equations such that the overall system has a closed loop dynamics, (4) A simulation analysis method based on the dynamic semantics of HPrTNs and supported in tool PIPE+, (5) A barrier certificate analysis technique based on inductive invariant reasoning supported in tool Pyomo with linear program solver Gurobi and SMT solver Z3, (6) A bounded model checking analysis approach supported by tool dReach and backend solver dReal. We have presented our detailed modeling method that covers results (1) to (3) in [7]. In this paper, we will provide a brief overview of the modeling method while focus on the analysis techniques covering results (5) to (6). In the following sections, we provide some background information on the modeling method and the details of the analysis techniques.

II. HYBRID PREDICATE TRANSITION NETS

In this section, a formal definition of HPrTNs [6] is provided.

An HPrTN is a tuple $N = (P, T, F, \alpha, \beta, \gamma, \mu, \lambda, M_0)$, where

(1) $P = P_d \cup P_c$ is a non-empty finite set of discrete places P_d and continuous places P_c (graphically represented by circles and double circles respectively);
(2) T is a non-empty finite set of discrete transitions (graphically represented by bars or boxes), which disjoins P, i.e. $P \cap T = \emptyset$;
(3) $F \subseteq P \times T \cup T \times P$ is a flow relation (the arcs of N);
(4) $\alpha: P \rightarrow Type$ associates each place $p \in P$ with a type in $Type$. $Type$ defines the structure of the data the places can hold. The basic types include *String*, *Integer*, and *Real*; and the composite types are defined using Cartesian product and power set;
(5) $\beta: T \rightarrow Constraint$ associates each transition $t \in T$ with a constraint. Each constraint is a disjunction $\bigvee_i d_i$ for $i \geq 1$, where each disjunct d_i has a canonical form $pre_i \wedge post_i$ that defines the precondition (enabling condition) and post-condition (output result) of a case of t respectively. The precondition contains only variables appearing in the labels of incoming arcs and the post-condition contains variables appearing in the labels of outgoing arcs;
(6) $\gamma: F \rightarrow Label$ associates each arc $f \in F$ with a label in the form of a simple variable x or a set element $\{x\}$;

DOI reference number: 10.18293/SEKE2022-010

(7) $\mu: P_c \rightarrow (\Re \times \Re)^n$ associates each continuous component of a continuous place with a pair of lower and upper bounds, where n is the number of continuous components;

(8) $\lambda: P_c \rightarrow ODE^n$ associates each continuous component of a continuous place an ordinary differential equation that defines its evolution;

(9) $M_0: P \rightarrow Token$ is an initial marking and associates each place $p \in P$ with some valid tokens (respecting the type of p and the bounds for continuous components). Each continuous place can only hold at most one token.

The dynamic semantics of HPrTNs are defined based on the markings (states) $M: P \rightarrow Token$. A transition $t \in T$ is *enabled* in marking M if one of its precondition is true, Formally: $\forall p \in P. (\theta(\bar{\gamma}(p,t)) \subseteq M(p) \wedge \exists i. (\theta(\beta(t).pre_i)))$, where θ is a substitution that instantiates all the variables in relevant arcs and constraint expression.

An enabled transition $t \in T$ in marking M with substitution θ can *fire*. The firing of transition t results in a new marking M' defined by $\forall p \in P. (M'(p) = M(p) \cup \theta(\bar{\gamma}(t,p)) - \theta(\bar{\gamma}(p,t)))$, which is denoted as: $M \xrightarrow{t/\theta} M'$. The firing of a transition is instant and does not consume time. Two enabled transitions are in conflict if the firing of one of them disables the other. Non-conflict enabled transitions can fire concurrently.

Tokens in continuous places are continuously evolving according to the differential equations governing the change rates as long as their bounds are not violated. Given a marking M, we use $[M]$ to denote the state space covering all possible continuous token evolution with the same token distribution.

Let T_i be a set of concurrently enabled non-conflict transitions with corresponding substitutions θ_i in marking $[M_i]$, and $[M_{i+1}]$ be the resulting new marking after firing T_i with θ_i. The behavior of the net N consists of the set of all firing sequences $[M_0] \xrightarrow{T_0/\theta_0} [M_1] \cdots [M_i] \xrightarrow{T_i/\theta_i} [M_{i+1}] \cdots$. The set of all reachable markings is denoted as $[[M_0] >$.

III. MODELING CPS WITH LECS USING HPRTNS

Our modeling methodology based on HPrTNs consists of three steps: (1) modeling LECs using DNNs, (2) training LECs though modeling environment and system dynamics using reinforcement learning, and (3) modeling and integrating LECs with other conventional system components within the HPrTN paradigm. We briefly discuss our modeling methods in steps (1) and (2) below.

DNNs have become a dominant deep learning approach in recent years. A DNN has an architecture, which consists of an input layer, multiple hidden layers and an output layer. Each layer contains multiple neurons (each is represented by a circle) that contain numerical values. The value of a neuron in layer l is calculated through an activation function on the weighed input from neurons in layer $l-1$. Different types of DNN architecture can be obtained based on how the adjacent layers are connected, including feedforward (fully connected), convolution, and recurrent. DNNs are trained using the output results. A cost function defined on the output is used to calculate the final error rate. By calculating and propagating the error rates layer by layer backwards starting from the final error rate, we can adjust the weights and biases based on the error rates.

We have developed a novel HPrTN template to model a DNN with backpropagation, where the architecture of the DNN is modeled as follows:

(1) Each layer l in DNN is modeled by a discrete place p_l of type $\Re \times \ldots \times \Re$, where the cardinality determines the number of neurons within the layer;

(2) Modeling neurons – each neuron is modeled by a token (or a token component) of a real type, and the neurons within the same layer is modeled by a structured token;

(3) Let l and $l+1$ be two layers with cardinality m and n respectively, a discrete place w_l of type $(\Re^m)^n$ is used to model the weight matrix between these two layers and a discrete place b_l of type $\Re^n$ is used to represent the bias vector;

(4) A transition t_l with input places p_l, w_l, b_l, and output place p_{l+1} is used to model activation function between these two layers, the transition constraint $\bigwedge_{i=1}^{n} z_i = \sigma(w^i x^T + b_i)$ defines the algebraic relationships between the activations (the neurons) in these two layers, where each z_i is a weighted input to neuron i in layer $l+1$;

(5) A transition $cost$ is added with the constraint defining initial error estimation. This transition has the place modeling the final output layer as an input, and an output place for error propagation;

(6) A place $doutput$ is added as an input to the $cost$ transition, a token specifying the desirable output y resides in this place;

(7) A place e_l abstracting backward error propagation is added between layers l and $l+1$;

(8) A transition g_l is added between layers l and $l+1$; this transition produces the backward error and updates the weights and biases of layer $l+1$.

Fig. 1 shows an HPrTN of two adjacent layers of a DNN with backpropagation:

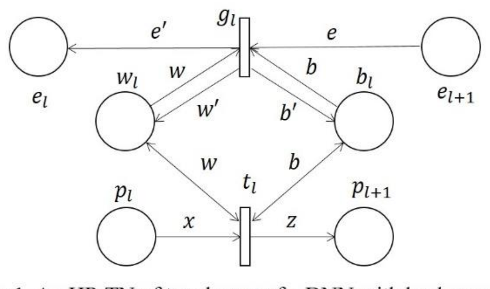


Fig.1. An HPrTN of two layers of a DNN with backpropagation

Reinforcement learning (RL) is a major machine learning approach, which learns how to attain a complex objective (goal) or how to maximize along a particular dimension over many steps. An agent (controller) continuously interacts with an environment (plant). The agent selects some action a according to a policy π defined using a value function on an input state s and reward r or defined using a Q-value function on an input pair of state s and action a. The environment generates a new state s' and reward r' according to the given action a. The goal is to maximize cumulative rewards when a final state is reached.

Our method is based on neural fitted Q-learning process [10], which builds an HPrTN model for a CPS with LECs and uses the simulation capability of HPrTNs to train LECs modeled as a DNN where a baseline controller or plant is used as the learning environment. We have developed several HPrTN templates to capture temporal difference methods in RL, which support a variety of RL learning settings, including on / off line, model based / model free, stationary / non-stationary, and discrete / continuous.

To demonstrate the applicability of the method to model and train a LEC, we have used the following car system adapted from [1]: a car needs to move along a circular track as closely as possible. The sensors (simulated) of the car can detect the center of the track. The car's position is defined by its coordinates (x, y). The car has a direction θ and a speed v. The car has three modes straight, left, and right and the dynamics in each mode is as follows:

- Right: $\dot{x} = (vcos\theta)/2, \dot{y} = (vsin\theta)/2, \dot{\theta} = -\pi, d \geq e$;
- Straight: $\dot{x} = vcos\theta, \dot{y} = vsin\theta, \dot{\theta} = 0, -e \leq d \leq e$;
- Left: $\dot{x} = (vcos\theta)/2, \dot{y} = (vsin\theta)/2, \dot{\theta} = \pi, d \leq -e$.

A parameter e is used to define the error margin [-e, e], and the distance d between the car's current position and the center of the track is calculated dynamically to control the switching between modes. A baseline controller mimicking the environment of the car and several advanced controllers (LECs) using different DNN architectures and activation functions have been tried. Fig. 2 shows a trained advanced controller (AC) together with a baseline controller (BC) modeled using an HPrTN developed in PIPE+ [14].

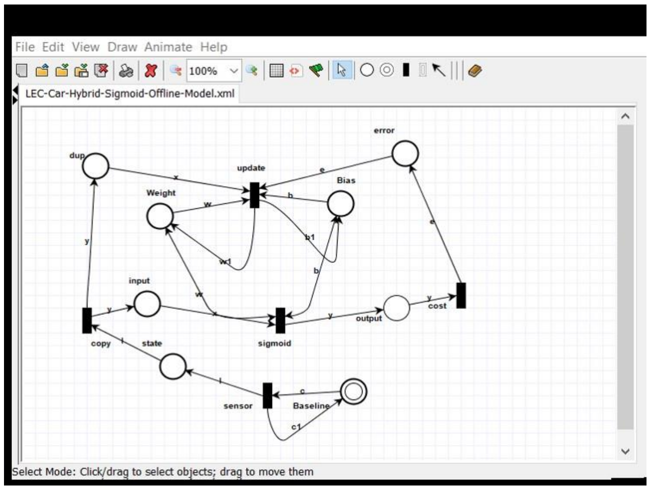

File Edit View Draw Animate Help
100%
LEC-Car-Hybrid-Sigmoid-Offline-Model.xml
error
dup
update
Bias
Weight
x
e
w
b
y
b1
w1
b
input
w
e
y
x
y
output
y
cost
copy
state
sigmoid
l
l
sensor
c
Baseline
c1
Select Mode: Click/drag to select objects; drag to move them

Fig.2. An HPrTN representing a CPS with a LEC

Training is done using HPrTN's simulation capability. For example, we have run six batches with randomly generated with position (x, y), and direction θ. Each batch contains 1000 execution steps. The overall training involves firing 100,000 transitions and takes 31673 milliseconds on a PC with Intel(R) Core(TM) i7-4770S CPU @ 3.10 GHz and 8 GB RAM running Windows 10 OS.

IV. Analyzing CPSs with LECs

Three techniques for analyzing CPSs with LECs are explored, including simulation, barrier certificate, and SMT based bounded model checking. Simulation is supported by the operational semantics of HPrTNs, which can be used to train LECs as well as test CPSs with LECs by selecting targeted or random initial markings. Simulation results also provide the basis for barrier certificate analysis. Simulation is easy to use, scalable, and fully automatic. Simulation is supported by our tool environment PIPE+. In the following sections, we describe the barrier certificate and the bounded model checking techniques.

A. *Barrier Certificate Technique*

Barrier certificate technique is based on symbolic simulations for finding inductive invariants to prove the safety requirements of a dynamic system. Since a CPS with LECs modeled in an HPrTN is executable and produces simulation traces, we can apply barrier certificate technique to analyze the dynamics of the whole closed loop system.

A barrier certificate is a differentiable function B from the set of states of the dynamical system to the set of real numbers satisfying the following conditions:

(1) $\forall x \in X_0$: $B(x) \leq 0$, where X_0 is the set of possible initial states,

(2) $\forall x \in U$: $B(x) > 0$, where U is the set of unsafe states,

(3) $\forall x$: $B(x) = 0 \Rightarrow (\nabla B)^T \bullet f(x) < 0$, where $(\nabla B)^T$ is the transpose of gradient $\nabla B = (\frac{\partial B}{\partial x_1}, \ldots, \frac{\partial B}{\partial x_n})$ and $f(x)$ defines the system dynamics.

Condition (3) ensures future system states are safe by ensuring the separation the set of unsafe states from the set of reachable states from the given initial states X_0. Thus a barrier certificate provides an unbounded-time safety certificate of the system.

The key idea is to find a symbolic representation of a barrier certificate from sample simulation traces. This analysis technique was first used to analyze hybrid systems in [15], and more recently applied to study CPSs with LECs [18]. We have adapted the process in [18] to find candidate barrier certificates using optimization system Pyomo [9] with linear solver Gurobi and to validate a barrier certificate using SMT solver Z3.

First, a candidate barrier certificate W (similar to find a Lyapunov candidate in stability analysis is found using a typical template (sum of squares polynomials): $W(x) = x^T Px$, where $P \in \Re^{m \times m}$ is symmetric. The key is to find the values of P using linear constraints: $W(x[t_i]) > 0$ and $W(x[t_i]) - W(x[t_{i+1}]) > 0$, where $x[t_i]$ $(0 \leq i \leq N)$ is a simulation trace of the closed loop (including both plant and DNN controller) system dynamics f. The above linear constraints correspond to the negations of conditions (1) and (3) in barrier certificate. W is a positive function and decreases along system trajectories. Then, a level l is found such that $B(x) = W(x) - l$, where l is a non-negative real number that separates X_0 from U. The overall process in [18] is shown in Fig. 3.

In the above process:

- Equation (3.1): $\forall x \in D. (x \notin X_0 \wedge (\nabla W)^T \bullet f(x) \geq -\gamma)$
- Equation (3.2): $\exists x \in X_0. (x \notin \{x | W(x) - l \leq 0\})$
- Equation (3.3): $\exists x \in \{x | W(x) - l \leq 0\}. (x \in U)$

Equations (3.2) and (3.3) define the opposite of separation, i.e. unsafe.

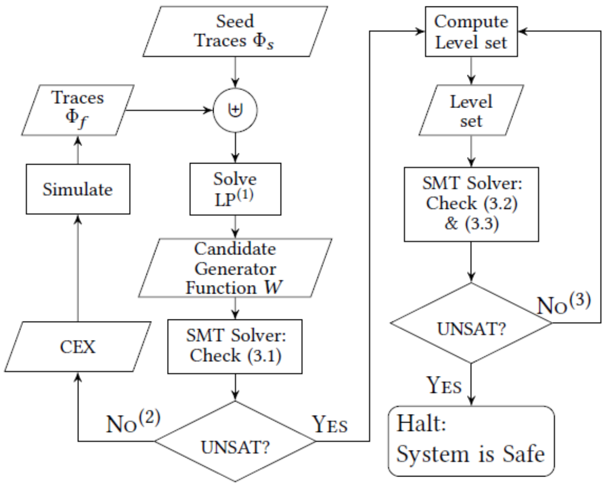


Fig.3. The process of finding barrier certificate

The barrier certificate technique is applied to the car system presented in a previous section. Eight simulation traces (200 steps) of the LEC automated vehicle are run and collected around the region x: [-1,1], y: [49, 51], and θ: [-5, 5]. Python is used to process the raw simulation data into 10 steps of data of system error dynamics (distance error: $\sqrt{x^2+y^2}$-50, where 50 is the radius of the circular track, and angular error: 0). The sum of squares polynomials template is used to fit the simulation data, and the resulting equations are solved using optimization system Pyomo [9] with solvers glpk and Gurobi. Among the 8 sets of equations, four are successfully solved while the other four have no solutions.

One of the candidate barrier certificate is $W(d,\theta) = 0.776 * d^2 + 0.2 * \theta * d + 0.013\theta^2$, where $d = \sqrt{x^2+y^2} - 50$ and $\nabla W = (\frac{\partial W}{\partial d}, \frac{\partial W}{\partial \theta}) = (1.552 * d + 0.2 * \theta,\ 0.2 * d + 0.026 * \theta)$. The error dynamics is $f = [\dot{d}, \dot{\theta}]$: $\dot{d} = (x * \dot{x} + y * \dot{y})/\sqrt{x^2+y^2} = (x * \sin(\theta) + y * \cos(\theta))/\sqrt{x^2+y^2}$, and Equation (3.1) is $\forall x \in D. (x \notin X_0 \wedge (\nabla W)^T \bullet f(x) \geq -\gamma)$, where $\gamma = 0.0001$, which is formulated as a constraint satisfaction problem using Pyomo and solved using Z3. Z3 confirms $W(d,\theta)$ as a valid barrier certificate candidate. Z3 is then applied to check Equation (3.2) in finding level $l = 60$, and thus the barrier certificate $B(x,y,\theta) = W(x,y,\theta) - 60$. Finally, Z3 is used to check Equation (3.3) to ensure the safety region, where unsafe region is defined by half spaces: $x \leq -5$, $x \geq 5, y \leq 48, x \geq 52$.

B. Bounded Model Checking Technique

Bounded model checking was first developed to analyze safety properties of discrete systems [3]. In bounded model checking, the following logic formula ϕ_k is constructed from a given system model and property: $\phi_k = I(s_0) \wedge \bigwedge_{i=0}^{k-1} T(s_i, s_{i+1}) \wedge \bigvee_{i=0}^{k} f(s_i)$, where $I(s_0)$ is the characteristic function of the initial state s_0, $T(s_i, s_{i+1})$ is the characteristic function of the transition relation, and $f(s_i)$ represents the negated safety property in unrolled state s_i $(0 \leq i \leq k)$. If ϕ_k is satisfiable, there is a transition sequence or a trace from the initial state s_0 to a state s_i that satisfies f, thus violates the safety property. An SMT solver is used to check ϕ_k. Bounded model checking can be used to find violation of a safety property through a counter example, and can ensure a safety property up to k steps.

Bounded model checking techniques have been generalized to analyze hybrid systems with limited successes in the past decade ([2], [4], [5], [13]) and thus can be applied to analyze CPSs. However formal analysis techniques for LECs modeled using DNNs barely exist, a few existing works ([8], [12], [16]) can only handle simple activation functions such as ReLU.

A recent work [11] shows promise to formally analyze CPSs with LECs, in which the LEC modeled in DNN is transformed into a hybrid automaton, and then the overall system is analyzed by composing the LEC generated hybrid automaton with the hybrid automaton modeling the rest of the system. The overall closed loop system model in [11] is shown in Fig. 4.

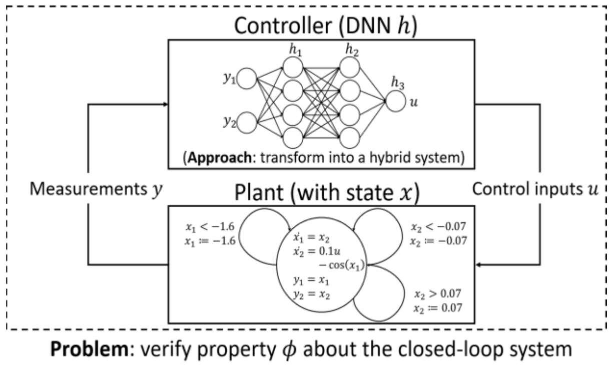


Fig.4. The closed loop system of a controller with plant

The plant dynamics is defined by $\dot{x} = f_p(x,u)$ and $y = g(x)$, where $x \in \mathfrak{R}^n$ is system state, and $u \in \mathfrak{R}^m$ is the input, f_p is a locally Lipschitz-continuous vector field, $g: \mathfrak{R}^n \rightarrow \mathfrak{R}^q$. The DNN controller is defined by $u = h(y)$, where $h: \mathfrak{R}^q \rightarrow \mathfrak{R}^m$. The closed loop system dynamics: $\dot{x} = f_p(x, h(g(x)))$.

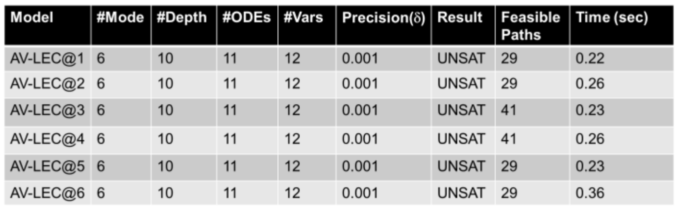
The DNN is transformed into a hybrid automaton as follows: $h(y) = h_L \circ h_{L-1} \circ \ldots \circ h_1(y)$, where each hidden layer i has an element-wise sigmoid activation function $h_i(y) = 1/(1 + e^{-(W_i y + b_i)})$, with the last layer being a linear function: $h_L(y) = W_L y + b_L$. The derivative of sigmoid function $\sigma(y) = 1/(1 + e^{-y})$ is $\frac{d\sigma}{dy} = \sigma(y)(1 - \sigma(y))$. The timed proxy function $\sigma(t,y) = 1/(1 + e^{-ty})$ is used to define evolution with derivative: $\dot{\sigma}(t,y) = \frac{\partial \sigma}{\partial t} = y\sigma(t,y)(1 - \sigma(t,y))$. The resulting hybrid automaton has one mode for each DNN layer. A set of continuous variables (each corresponding to a neuron) is introduced. The flow of continuous variable y_j in each mode i

is defined by proxy function $\sigma_{ij}(t, y)$ from $t = 0.5$ to 1 defined using differential equation $\dot{\sigma_{ij}}(t, y)$. Another set of continuous variables are used to keep track the linear functions (weighted sums) of each neuron and have constant change rates 0. Discrete jumps between modes happen when $t = 1$. The overall closed loop system model (composition) is $S = h\,||H_p$, where H_p is the plant automaton with dynamics $f_p(x, u)$. The reachability property on plant with initial state X_0 is defined by $\varphi(X_0) \Rightarrow f(x(t))$, for $t \geq 0$.

A bounded model checking technique has been developed for CPSs with LECs in this research. The approach in [11] is adapted to translate the component of an HPrTN representing a LEC into a hybrid automaton. The translation of the rest part of HPrTN to a hybrid automaton is straightforward based on the relationships between HPrTNs and hybrid automata given in [6]. Bounded model checker dReach [13] with dReal [5] for non-linear hybrid automata are used to check the resulting composed hybrid automaton.

The bounded model checking technique is applied to the car system presented in a previous section. The Plant (Car) H_p has 3 continuous variables – location x and y and direction angle θ, and three modes – forward, left, and right. The controller (DNN) h has three layers with one hidden layer containing sigmoid activation function. The overall closed loop system model $S = h\,||H_p$ contains 6 modes (after reduction), and twelve continuous variables (including time). The reachability property to check (the car off the center of circular track of radius 50 by 5) is $sqrt(x^2 + y^2) - 50 > 5 \vee sqrt(x^2 + y^2) - 50 < -5$. The model checking results are shown Table I below:

Table I Bounded model checking results



Model	#Mode	#Depth	#ODEs	#Vars	Precision(δ)	Result	Feasible Paths	Time (sec)
AV-LEC@1	6	10	11	12	0.001	UNSAT	29	0.22
AV-LEC@2	6	10	11	12	0.001	UNSAT	29	0.26
AV-LEC@3	6	10	11	12	0.001	UNSAT	41	0.23
AV-LEC@4	6	10	11	12	0.001	UNSAT	41	0.26
AV-LEC@5	6	10	11	12	0.001	UNSAT	29	0.23
AV-LEC@6	6	10	11	12	0.001	UNSAT	29	0.36

V. CONCLUDING REMARKS

HPrTNs are well suited for modeling CPSs with LECs due to (1) their capability to capture concurrency and hybrid behaviors in typical CPSs, (2) their graphical representation and executability to naturally fit the machine learning techniques based DNNs and RL, and (3) their distributed and concurrent data flow computational model to easily integrate different system components. This paper contributes a unique analysis methodology to analyze CPSs with LECs modeled using HPrTNs. Although both barrier certificate and bounded model checking techniques have been around for decades, their applications to deal with LECs only happened in recent years. Integrating both analysis techniques within the same framework with unique tool support is a new contribution of this work.

More case studies will be carried out to show the effectiveness and scalability of this analysis methodology. Additional research will be done to develop new techniques to analyze the robustness of DNNs and the overall stability and safety of CPSs with LECs built using neural fitted RL.

ACKNOWLEDGMENT

This work was partially supported by AFRL under FA8750-15-2-0106 and FA9550-15-0001. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

REFERENCES

[1] R. Alur: "Principles of Cyber-Physical Systems", The MIT Press, 2015.

[2] K. Bae and S. Gao: "Modular SMT-based analysis of nonlinear hybrid systems," 2017 Formal Methods in Computer Aided Design (FMCAD), Vienna, 2017, pp. 180-187.

[3] E. Clarke, A. Biere, R. Raimi, Y. Zhu: "Bounded model checking using satisfiability solving". Formal Methods in System Design 19(1), 7–34 (2001).

[4] A. Cimatti, S. Mover, and S. Tonetta: "SMT-based verification of hybrid systems". In Proc. AAAI, 2012.

[5] S. Gao, S. Kong, and E. M. Clarke: "dReal: An SMT solver for nonlinear theories over the reals". In CADE, volume 7898 of LNCS, pages 208–214. Springer, 2013.

[6] X. He and D. Alam: "Hybrid Predicate Transition Nets - A Formal Method for Modeling and Analyzing Cyber-Physical Systems", Proc. of The 2019 IEEE International Conference on Software Quality, Reliability & Security (QRS'19), Sofia, Bulgaria, 2019, 216-227.

[7] X. He: "Modeling and Analyzing Cyber Physical Systems with Learning Enabled Components using Hybrid Predicate Transition Nets", Proc. of 2021 IEEE 21th International Conference on Software Quality, Reliability and Security Companion (QRS-C), Hainan, China, 2021.

[8] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu: "Safety verification of deep neural networks", In International Conference on Computer Aided Verification, 2017, Springer, 3–29.

[9] W. E. Hart, C. D. Laird, J. Watson, D. L. Woodruff, G. A. Hackebeil, B. L. Nicholson, J. D. Siirola: "Pyomo – Optimization Modeling in Python", Springer, 2017.

[10] R. Hafner, M. Riedmiller: "Reinforcement learning in feedback control – Challenges and benchmarks from technical process control", Mach Learn (2011) 84:137–169.

[11] R. Ivanov, J. Weimer, R. Alur, G. J Pappas, and I. Lee: "Verisig: verifying safety properties of hybrid systems with neural network controllers" In Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control (HSCC'19), 2019, 169–178.

[12] G. Katz, C. Barrett, D. L Dill, K. Julian, and M. J Kochenderfer: "Reluplex: An efficient SMT solver for verifying deep neural networks", In International Conference on Computer Aided Verification. Springer, 2017, 97–117.

[13] S. Kong, S. Gao, W. Chen, and E. M. Clarke: "dReach: □-reachability analysis for hybrid systems". In TACAS, volume 9035 of LNCS. Springer, 2015.

[14] S. Liu, R. Zeng, X. He: "PIPE+ - A Modeling Tool for High Level Petri Nets", Proc. of International Conference on Software Engineering and Knowledge Engineering (SEKE11), Miami, July 2011, 115 - 121.

[15] S. Prajna and A. Jadbabaie: "Safety Verification of Hybrid Systems Using Barrier Certificates". In In Hybrid Systems: Computation and Control. Springer, 477–492, 2004.

[16] L. Pulina and A. Tacchella: "An Abstraction-Refinement Approach to Verification of Artificial Neural Networks". In Proc. 22nd Int. Conf. on Computer Aided Verification (CAV), pages 243-257, 2010.

[17] R. S. Sutton, and A. G. Barto: "Reinforcement Learning: An Introduction", (2nd edition), Cambridge, MA: MIT Press, 2018.

[18] C. Tuncali, J. Kapinski, H. Ito, J. Deshmukh: "Reasoning about Safety of Learning-Enabled Components in Autonomous Cyber-physical Systems". Design Automation Conference (DAC) 2018.