2019 IEEE 58th Conference on Decision and Control (CDC)
Palais des Congrès et des Expositions Nice Acropolis
Nice, France, December 11-13, 2019

Semantic Inference for Cyber-Physical Systems with Signal Temporal Logic

Gang Chen, Mei Liu, and Zhaodan Kong

Abstract— **Formal specification plays crucial roles in the rigorous verification and design of cyber-physical systems (CPS). The challenge of getting high-quality formal specifications is well documented. This challenge is further exacerbated in CPS with artificial-intelligence- or machine-learning-based components. This paper presents a problem called 'semantic inference', the goal of which is to automatically translate the behavior of a CPS to a formal specification written in signal temporal logic (STL). To reduce the potential combinatorial explosion inherent to the problem, this paper adopts a search strategy called agenda-based computation, which is inspired by natural language processing. Based on such a strategy, the semantic inference problem can be formulated as a Markov decision process (MDP) and then solved using reinforcement learning (RL). The obtained formal specification can be viewed as an interpretable classifier, which, on the one hand, can classify desirable and undesirable behaviors, and, on the other hand, is expressed in a human-understandable form. The performance of the proposed method is demonstrated with a case study.**

Index Terms— **Cyber-physical systems, reinforcement learning, semantic inference, signal temporal logic.**

I. Introduction

The development and deployment of cyber-physical systems (CPS) is revolutionizing our economy and society in domains, such as agriculture, health care, transportation, manufacturing, etc. Formal specification, a mathematical or logical statement of what the CPS is supposed to do, plays crucial roles in the rigorous verification and design of CPS, particularly those for safety-critical applications. However, coming up with high-quality formal specifications for complex CPS is a challenge task, even for domain experts [2]. The emergence of CPS equipped with artificial intelligence (AI) and machine learning (ML) components further exacerbates the challenge. For instance, it is not very clear how to construct a set of proper specifications for the vision system used by many autonomous vehicles [3].

We believe that finding formal specifications for complex CPS relies on solving an inverse problem first. Our rationale is that in order for a designer to specify *what a system is supposed to do*, the designer needs to first understand *what the system can do*, particularly when the system is complex. We will call the latter process as 'semantic inference', the specific goal of which is to automatically translate the behavior of a CPS into a formal specification. The specification will be 'formal' in the sense that it will be written in some formal language, ideally in temporal logic statement, such as signal temporal logic (STL) [4], which has been used extensively in the specification of CPS. We envision that semantic inference will enable an interactive process in which a human can inquire a CPS regarding its capability and eventually the human and the CPS can work together to come up with an intuitive and at the same time rigorous specification for the CPS. Specifically, semantic inference tries to find an STL formula φ, which serves as a classifier that distinguishes the desirable behaviors from the undesirable ones. Traditional classifiers, such as Support Vector Machine (SVM), usually require a human expert to interpret the classifier in the context of possibly complex and non-intuitive features. On the contrary, our classier φ is written in STL [4], which can be easily understood by even a lay user (with some minimal training). For instance, one such φ can be $\square_{[0,2]}(x < 60)$, which reads "between times 0 and 2, the value of x is always smaller than 60", where $\square$ is the temporal operator for "always".

a) Related work: Our work is closely related to the problem of requirement mining, which was first proposed in the field of software engineering for legacy code understanding, software maintenance, etc. [5]. In recent years, there has been a surge of interest in requirement mining for CPS [6], [1], [7], [8], [9], [10], particularly those with ML- or AI-based components [3]. Existing works address the requirement mining problem in two ways. The first way assumes that the output of the requirement mining problem is a formula φ_θ with a fixed structure but unknown parameter θ. With such an assumption, the requirement mining problem can be transformed into an optimization problem with the goal of finding a parameter θ^* such that φ_{θ^*} optimizes certain cost function, which is usually defined with the concept of *robustness degree* [4]. Obviously, the fixed structure assumption is not that realistic in the sense there needs to be a human domain expert, who ideally should also be knowledgeable of formal language, to prescribe the structure. To make the techniques of requirement mining or semantic inference more useful and widely adopted, the second way drops the assumption and finds a method to automatically construct a formula with the proper structure and parameter, such as the methods in [8], [9]. But such a relaxation makes the problem much harder, since the number of candidate structures that need to be searched grows exponentially with the length of the formula (this is essentially a combinatorial problem). Algorithms proposed in [8], [9] provide some strategies to conduct such a search. But the strategies work well only under some conditions. For instance, the algorithm presented in [8] organizes all the candidate structures in a lattice and then searches the lattice starting from the most restrictive

G.C. and Z.K. are with the Department of Mechanical and Aerospace Engineering, University of California, Davis (email: zdkong@ucdavis.edu) and M.L. is with the Department of Mechanical Engineering, University of Hong Kong.

structure; if a satisfactory parameter cannot be found for such a structure, its children (less restrictive structures) will then be searched. But such a method is not very efficient. Moreover, the search order is pre-defined and knowledge gained during the search, e.g., more promising structures vs. less promising ones, is not utilized. This paper will explore a more strategic method by borrowing techniques from natural language processing (NLP) and machine learning (ML).

b) Contributions: The main contributions of this paper is that we propose a way of formulating the requirement mining problem (under the challenging condition that the formula structure is not provided a priori) as a semantic inference problem (Section III). Such a formulation subsequently enables us to *strategically* search formula structures by using techniques from NLP and ML (Section IV) and improve the searching efficiency significantly (as demonstrated theoretically in Section IV-D and empirically in Section V).

II. Preliminaries

A. Signal Temporal Logic

Signal Temporal Logic (STL) is a temporal logic defined over signals [4]. It is basically a predicate logic with interval-based temporal semantics. The syntax of STL used in this paper is defined as

$$\varphi ::= \mu | \varphi_1 \wedge \varphi_2 | \varphi_1 \vee \varphi_2 | \Diamond_{[\tau_1,\tau_2)} \varphi | \Box_{[\tau_1,\tau_2)} \varphi, \quad (1)$$

where τ_1 and τ_2 are non-negative finite real numbers, and μ is a predicate over a signal, which can be defined as $l(x[t]) \sim c$, where $x \in \mathcal{E}(\mathbb{R}^n, \mathbb{R}^+)$ is a signal, and $x[t]$ is the valuation of signal x at time t, $l \in \mathcal{F}(\mathbb{R}^n, \mathbb{R})$ is a function, $\sim \in \{\leq, <, \geq, >\}$, and $c \in \mathbb{R}$ is a constant. The Boolean operators $\vee$ and $\wedge$ are disjunction ("or") and conjunction ("and"), respectively. The temporal operators $\Diamond$ and $\Box$ stand for "eventually" and "always", respectively. STL is equipped with a quantitative semantics called *robustness degree* $\rho : \Psi \times \mathcal{E}(\mathbb{R}^n, \mathbb{R}^+) \to \mathbb{R}$ which maps an STL formula $\varphi \in \Psi$ and a signal x to a real value. $\rho(x, \varphi)$ indicates how far a signal x is away from satisfying STL formula φ, which is defined nested as follows [4]:

$$\begin{aligned}
\rho(x, (l(x[t]) < c)) &= c - l(x[t]) \\
\rho(x, (l(x[t]) \geq c)) &= l(x[t]) - c \\
\rho(x, \varphi_1 \wedge \varphi_2) &= \min\big(\rho(x, \varphi_1), \rho(x, \varphi_2)\big) \\
\rho(x, \varphi_1 \vee \varphi_2) &= \max\big(\rho(x, \varphi_1), \rho(x, \varphi_2)\big) \\
\rho(x, \Box_{[a,b)} \varphi) &= \min_{t' \in [t+a, t+b)} \rho(x, \varphi) \\
\rho(x, \Diamond_{[a,b)} \varphi) &= \max_{t' \in [t+a, t+b)} \rho(x, \varphi).
\end{aligned}$$

B. Attribute Grammar

The attribute grammar used in this paper is defined by a 4-tuple [11]

$$G =< V_N, V_T, P, g >, \quad (2)$$

where V_N is the set of non-terminal nodes, V_T is the set of terminal nodes, P is the set of production rules, and g is a relation mapping each node to its attributes. We use capital letters to denote non-terminal nodes, e.g., $A \in V_N$, and use lowercase letters, including Greek ones, to denote terminal nodes, e.g. $a \in V_T$. The attributes of a node is specified by the relation g. For instance, the list of attributes of a non-terminal node A is $g(A)$. In this paper, we will use those attributes that are synthesized [11], meaning that if $A_0 \to A_1 A_2$ is a production rule in P (in the computation tree, A_1 and A_2 are the children of A_0), then $g(A_0)$ is the collection of $g(A_1), g(A_2)$, namely $g(A_0) = g(A_1) \cup g(A_2)$ (more details will be provided in Section III-A).

C. Markov Decision Process

Definition 1: A *Markov decision process (MDP)* is defined as a 5-tuple $\mathcal{M} =< S, A, \triangle(\cdot|\cdot,\cdot), r(\cdot|\cdot,\cdot), \gamma >$, where S is a set of states, A is a set of actions, $\triangle(s'|s,a)$ is the probability that action a in state s will lead to state s', $r(s'|s,a)$ is the immediate reward received for taking action a at state s, and $\gamma \in [0,1]$ is the discount factor.

III. Problem Formulation

In this section, we will first show that STL can be defined with a new formalism called *STL attribute grammar*. We will then present the semantic inference problem pertaining to this grammar.

A. STL Attribute Grammar

Definition 2: The *STL attribute grammar* $\mathcal{G}_{\mathcal{STL}}$ is an attribute grammar $< V_N, V_T, P, g >$ with the following specific components:

- $V_N = \{A, B\}$, where each element of V_N corresponds to an STL fragment (partial formula);
- $V_T = \{\mu, \Diamond, \Box, \vee, \wedge\}$, where the meanings of the symbols are the same as those in Eqn. (1);
- $P = \{P_1, \cdots, P_7\}$, where the specific production rules are shown in Table I (there are five categories of rules, namely *Instance*, *Eventually*, *Always*, *Or*, and *And*);
- g maps each node to two types of attributes: (a) *time* attributes that specify the time bounds of the temporal operators used in the node and (b) *predicate* attributes that specify the predicates used in the node. Specifically, the *predicate* attributes includes *signal name*, *comparison operator*, and *constant*. To give an example, for a terminal node $\mu : x_1 > 1$, its ordered list of *time* attributes is $\mu.time = \{\}$, which is empty, and its set of *predicate* attributes is $\mu.pre = \{x_1, >, 1\}$ (we will use the notations $.pre$ and $.time$ throughout the paper). Both types of attributes are synthesized. For instance, production rule $P_5 : A' \to A \vee B$ indicates that:

$$A'.time = A.time \cup B.time; \quad A'.pre = A.pre \cup B.pre.$$

Proposition 1: Any STL formula φ can be derived with the STL attribute grammar $\mathcal{G}_{\mathcal{STL}}$.

Example 1: This proposition can be best illustrated with an example as shown in Fig. 1. It can be easily seen that an STL formula $\varphi = \Box_{[0,3]}(\Diamond_{[0,2]}(x_1 > 1) \wedge \Box_{[0,1]}(x_2 < 2))$ can be derived by following a sequence of production rules $P_3 P_7 P_2 P_1 P_5 P_1$ applied to a set of properly attributed terminal nodes $V_T = \{\Box_{[0,3]}, \Diamond_{[0,2]}, \mu_1 := (x_1 > 1), \Box_{[0,1]}, \mu_2 := (x_2 < 2)\}$.

TABLE I: Production rules of $\mathcal{G}_{\mathcal{STL}}$.

Rule	Category	Notation	Attributes
P_1	Instance	$A\|B \rightarrow \mu$	$g(A\|B) = g(\mu)$
P_2	Eventually	$A \rightarrow \Diamond A$	$g(A) = g(\Diamond) \cup g(A)$
P_3	Always	$A \rightarrow \Box A$	$g(A) = g(\Box) \cup g(A)$
P_4	Eventually	$B \rightarrow \Diamond B$	$g(B) = g(\Diamond) \cup g(B)$
P_5	Always	$B \rightarrow \Box B$	$g(B) = g(\Box) \cup g(B)$
P_6	Or	$A\|B \rightarrow A \vee B$	$g(A\|B) = g(A) \cup g(B)$
P_7	And	$A\|B \rightarrow A \wedge B$	$g(A\|B) = g(A) \cup g(B)$

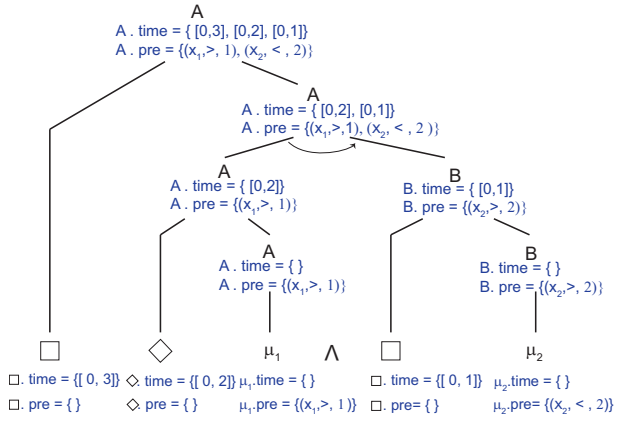


Fig. 1: The semantic inference tree of $\varphi = \Box_{[0,3]}(\Diamond_{[0,2]}(x_1 > 1) \wedge \Box_{[0,1]}(x_2 < 2))$. The arc with an arrow indicates the *And* operator. The *time* and *predicate* attributes of a node are shown immediately underneath the corresponding node.

B. Semantic Inference Problem Formulation

Problem 1: (Semantic Inference) Given the STL attribute grammar $\mathcal{G}_{\mathcal{STL}} =< V_N, V_T, P, g >$, a positive integer T, and two labeled signal sets, X^+, the desirable behaviors of a CPS, and X^-, the undesirable behaviors of the CPS, find a formula d such that the robustness degree

$$\rho(X, d) = \min(\min_{x \in X^+}(\rho(x, d)), \min_{x \in X^-}(\rho(x, \neg d))) \quad (3)$$

is maximized, where (a) $X = X^+ \cup X^-$, (b) $\rho(x, d)$ and $\rho(x, \neg d)$ denote the robustness degrees of a signal x with respect to the formulas d and $\neg d$, respectively, and (c) $|d| \leq T$ with $|d|$ denoting the number of production rules used to generate d.

Remark 1: It is helpful to understand Eqn. (3) by visualizing it in terms of Support Vector Machine (SVM): formula d defines the boundary between desirable and undesirable behaviors; $\min_{x \in X^+}(\rho(x, d))$ and $\min_{x \in X^-}(\rho(x, \neg d))$ are the distances between the boundary and the desirable and undesirable behaviors, respectively; and finally maximizing $\rho(X, d)$ results in a boundary that maximally separates the desirable and undesirable behaviors. But compared with the boundaries in SVM, which are hyper-planes in some high dimensional feature spaces, which might be hard to interpret, our boundary is defined by an STL formula φ, which is easily understandable and obtained without any human intervention.

IV. Solution

To solve the semantic inference problem, several challenges present themselves. First, the space of possible formulas (formula structures) grows exponentially with respect to $|d|$, the number of production rules (or equivalently the length of the corresponding STL formula φ). We will need to handle this *combinatorial explosion*. Second, finding the sequence of production rules to derive d is a *sequential decision process*: the computation decisions made in the past may affect both the availability and goodness of future decisions.

We observe that the CPS semantic inference problem, i.e., Problem 1, is quite similar to the semantic parsing problem in question answering [12], the goal of which is, given a natural language sentence, to assign appropriate semantic roles to each component of the sentence. So our main idea is to borrow techniques from question answering, in conjunction with machine learning, to address the aforementioned challenges. Specifically, we will first use the idea of *agenda-based computation* to search possible formulas in a strategic order [12], thereby handling the combinatorial explosion (Section IV-A); we will then re-formulate the agenda-based computation problem as an MDP (Section IV-B), which enables the utilization of reinforcement learning to deal with issues due to sequential decision (Section IV-C).

A. Agenda-based Semantic Inference

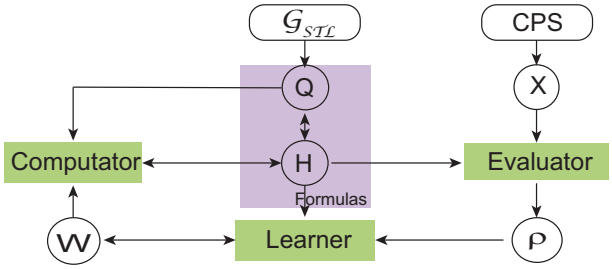


Fig. 2: Agenda-based semantic inference framework.

1) Framework: Fig. 2 shows our agenda-based framework for the semantic inference problem. The framework takes the STL attribute grammar $\mathcal{G}_{\mathcal{STL}}$ and a set of labeled signals X as the inputs and outputs an STL formula φ (or the equivalent formula d). At each computation step, the *Computator* chooses a formula from agenda Q and adds it to chart H (the roles of Q and H will be explained later) based on a value function with a parameter vector w. The *Evaluator* evaluates the performance of the chosen formulas in chart H with respect to X and the *Learner* updates w according to the performance. For readers who are familiar with reinforcement learning, it is helpful to think the *Evaluator* and the *Learner* work together as a *critic* and the *Computator* works as an *actor*. The *Computator* selects actions based on the value function defined by the *Evaluator* and the *Learner*, which criticize the actions made by the *Computator*.

2) Discretization: In this paper, instead of treating the time bounds of the temporal operators and the constants in the predicates as real numbers, we discretize them. We choose a sequence of time instances $\tau = (t_0, t_1, \cdots, t_n)$ with an equal interval, i.e., $t_i - t_{i-1} = \tau_0, \forall 1 \leq i \leq n$, where τ_0 is the time interval, as the *time* attributes. Combining these time bounds with temporal operators $\Box$ and $\Diamond$, we can get temporal operators with different time bounds. For example, for a time bound $[t_1, t_2)$, its two corresponding temporal operators are $\Box_{[t_1,t_2)}$ and $\Diamond_{[t_1,t_2)}$, called timed temporal operators. We treat the *predicate* attributes the same way. With discretization, agenda Q will be initialized with a set of simple formulas, each of which is constructed by combining a timed temporal operator and a predicate, i.e., $\phi_1 = \Diamond_{[0,2]}(x_1 > 1)$ and $\phi_2 = \Box_{[0,1]}(x_2 < 2)$; chart H will be initialized as an empty set.

3) Agenda and Chart Update: Agenda Q and chart H will be updated whenever the *Computator* chooses a formula from agenda Q. At each computation step, after the *Computator* chooses a formula from agenda Q, the chosen partial formula will be removed from agenda Q and added to chart H; moreover, new formulas will be added to agenda Q based on grammar $\mathcal{G}_{\mathcal{STL}}$. Specifically, let's assume the chosen formula is d_t and the current set of formulas in H is $\varphi_H(i) \in H(i = 1, \cdots, |H|)$, then formulas $\Box d_t, \Diamond d_t, d_t \wedge \varphi_H(i)$ and $d_t \vee \varphi_H(i)(i = 1, \cdots, |H|)$ can be generated with production rules $P_2/P_4, P_3/P_5, P_6$ and P_7 and then added to agenda Q.

TABLE II: Agenda Q and chart H generated in the first three steps for Example 1.

Step	**Agenda** Q	**Chart** H	**Chosen** d_t
0	ϕ_1, ϕ_2	-	ϕ_2
1	ϕ_1, $\Box\phi_2$, $\Diamond\phi_2$	ϕ_2	ϕ_1
2	$\Box\phi_2$, $\Diamond\phi_2$, $\Box\phi_1$, $\Diamond\phi_1$, $\phi_2 \vee \phi_1$, $\phi_2 \wedge \phi_1$	ϕ_2, ϕ_1	$\phi_1 \wedge \phi_2$
3	$\Box(\phi_1 \wedge \phi_2), \cdots$	ϕ_2, ϕ_1, $\phi_1 \wedge \phi_2$	$\Box(\phi_1 \wedge \phi_2)$

Example 1: (Cont.) Table II shows how agenda Q and chart H are updated in the first three steps of Example 1. In this case, agenda Q is initialized with formulas $\phi_1 = \Diamond_{[0,2]}(x_1 > 1)$ and $\phi_2 = \Box_{[0,1]}(x_2 < 2)$ and the H is initialized as an empty set. At step 0, the *Computator* chooses formula ϕ_2 from agenda Q (the policy to determine which formula to choose will be specified in Section IV-C.1) and put it into chart H. The chosen formula is used to generate new formulas $\Box\phi_2$ and $\Diamond\phi_2$ with production rules $P_2/P_4, P_3/P_5$ (As H is empty initially, production rules P_6 and P_7 are not applied). Then the new formulas are added to agenda Q. At step 1, the *Computator* chooses formula ϕ_1, which is used to generate new formulas $\Box\phi_1$, $\Diamond\phi_1$, $\phi_1 \wedge \phi_2$ and $\phi_1 \vee \phi_2$ with production rules P_2/P_4, P_3/P_5, P_6 and P_7, respectively. The new formulas are added to Q and ϕ_1 is added to H. At step 2, the *Computator* chooses formula $\phi_1 \wedge \phi_2$, then formula φ in Example 1 can be generated with production rule P_2/P_4. The *Computator* can proceed with choosing a formula from agenda Q, putting it to chart H, generating new formulas, adding them back to agenda Q, and continuing until the number of formulas in the chart, $|H|$, reaches the limitation T.

B. Agenda-based Semantic Inference as an MDP

The semantic inference process is essentially a *delayed reward* and *sequential decision process*. At each step, the *Computator* needs to make a decision on which formula to choose based on the current formulas in agenda Q; a sequence of decisions made by the *Computator* generates a formula d, the performance of which can only be evaluated once a complete formula, i.e., computation step reaches the limitation, has been derived. Thereby, we can conveniently model the computation process as a finite horizon MDP as follows (here we use Example 1 as an example):

- The space $S =< Q, H >$ is the set of all possible charts and agendas, e.g., a state $s \in S$ corresponds to the formulas in the second and the third columns of a row of Table II.
- The actions available at state s, denoted as $A(s)$, is the formulas in the corresponding agenda Q.
- The transition probability $\triangle(s'|s, d)$ is either 1 or 0. It is 1 if choosing a formula d updates state s, i.e., agent Q and chart H, to state s', i.e., agent Q' and chart H' (please consult Section IV-A.3 to see how agendas and charts are updated); otherwise, it is 0.
- The reward $r(s'|s, d)$ is the robustness degree over all the signals in X when the computation step reaches the limitation T, or the reward will be zero before the limitation.

Here we would like to point out that we mainly use the MDP formulation to enable us to utilize reinforcement learning later. As in the mainstream reinforcement learning setting [13], the MDP model will not be explicitly constructed.

C. Reinforcement-learning-based Semantic Inference

In this subsection, we will present a reinforcement-learning-based algorithm to solve the agenda-based computation problem in the context of the MDP formalism. We will first discuss the three components, *Computator*, *Evaluator*, and *Learner* (see Fig. 2), and then the overall algorithm.

1) Computator: The *Computator* acts as an agent, which, at step t, observes the current state s_t (or equivalently the current chart H_t and agenda Q_t) and chooses an action (formula) $d_t \in A(s_t)$. The policy π induces a distribution over trajectories ε of the MDP (or equivalently the formulas d of the computation process):

$$p^{\pi}(\varepsilon) = p(s_0) \prod_{t=0}^{T-1} \triangle(s_{t+1}|s_t, d_t)\pi(d_t|s_t). \tag{4}$$

The agent tries to find an optimal computation policy π that maximizes the expected accumulate robustness degree over the distribution (Eqn. (4))

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, E_{p^{\pi}(\varepsilon)}[\sum_{t=0}^{T-1} \gamma^t \rho(X, d_t)], \tag{5}$$

where γ is the discount factor and $\rho(X, d_t)$ is the robustness degree for taking action d_t (as defined in Eqn. (3)).

The optimal policy π^* for Eqn. (5) is a time-varying one, i.e., $\pi(d_t|s_t)$. We further approximate the policy $\pi(d_t|s_t)$ by a set of softmax functions $\pi(d_t|s_t, w_t), t = 0, \cdots, T-1$ with $W = \{w_0, w_1, \cdots, w_{T-1}\}$ as the parameter vector [13]. The policy $\pi(d_t|s_t)$ defines a conditional probability density function of action (formula) d_t given current state s_t, denoted as

$$\pi_{w_t}(d_t|s_t) = \frac{exp\{f(d_t)^T w_t\}}{\sum_{d' \in A(s_t)} exp\{f(d')^T w_t\}}, \tag{6}$$

where the probability indicates the preference of the *Computator* to choose action d_t at state s_t. The preference function is a linear function, $h(d) = f(d)^T w$, where $f(d) \in \mathbb{R}^F$ is the feature vector with F being the dimension of the feature space (which will be elaborated later), and $w \in \mathbb{R}^F$ is the parameter vector to be obtained. With such a parameterization method (Eqn. (6)), the objective function of the semantic inference process, Eqn. (5), can be transformed into another one regarding parameters:

$$w^* = \underset{w}{\operatorname{argmax}}\, E_{p^{\pi_w}(\varepsilon)}[\sum_{t=0}^{T-1} \gamma^t \rho(X, d_t)]. \tag{7}$$

The set of features, that define the preference function $h(d)$ for a formula d, includes: 1) the start and end times of *time* attributes, 2) the maximum and minimum values of *predicate* attributes, 3) the logarithm of the number of time bounds, 4) the logarithm of the number of production rules, 5) the logarithm of the number of predicates, 6) the logarithm of the number of conjunction and disjunction operators, and 7) the logarithm of the number of temporal operators.

2) Evaluator: The *Evaluator* in Fig. 2 evaluates the performance of the formulas in chart H (or equivalently the actions of the MDP) based on the set of the labeled signals of the CPS, X, according to Eqn. (3).

3) Learner: The role of the *Learner* is to find an optimal value for w (Eqn. (7)) by using reinforcement learning [13]. It can be observed from Eqn. (6) that the state s only provides the support for the distribution, the policy $\pi_w(d_t|s_t)$ depends only on the feature $f(d_t)$ and the parameter vector w. In this paper, the *Learner* applies the Monte-Carlo policy gradient method to learn the optimal policy [13]. At each step t, the *Learner* samples full-sized trajectories ε based on the current policy π_w. Then the policy parameter w is updated with the following rule:

$$w_t \leftarrow w_t + \alpha \gamma^t R_t \nabla_{w_t} \ln \pi(d_t|s_t), \tag{8}$$

where α is the learning rate and $R_t = \sum_{k=t+1}^{T} \rho(X, d_k)$ is the rewards received after time step t. The gradient of each policy decision is defined as follows:

$$\nabla_{w_t} \ln \pi(d_t|s_t) = f(d_t) - \sum_{d' \in A(s_t)} \pi_{w_t}(d'|s_t) f(d'). \tag{9}$$

4) Overall Algorithm: The overall algorithm to solve the semantic inference problem is shown in Algorithm 1. Line 1 initializes the state s_0, where the chart is empty and the formulas in the agenda come from the combination of predicates and temporal operators. Line 6 updates the chart and the agenda.

Algorithm 1 Reinforcement-learning based semantic inference

Input: A set of labeled signals $X = X^+ \cup X^-$, STL attribute grammar $\mathcal{G}_{\mathcal{STL}}$, episode horizon T, learning rate α, number of training episodes M

Output: The optimal parameter vector w.

1: Initialize the state to s_0
2: Initialize the parameter vector to $W_0 \in \mathbb{R}^{F \times T}$ (e.g., to **0**)
3: **for** $m = 1$ to M **do**
4: **for** $t = 0$ to $T - 1$ **do**
5: Choose action d_t according to policy $\pi(d_t|s_t, w_t)$
6: Update (H, Q) to get state s_{t+1} and reward ρ_{t+1}
7: **for** $t = 0$ to $T - 1$ **do**
8: $R_t \leftarrow \sum_{k=t+1}^{T} \rho(X, d_k)$
9: $w_t \leftarrow w_t + \alpha \gamma^t R_t \bigtriangledown \ln \pi(d_t|s_t, w_t)$

D. Complexity Analysis

Semantic inference is a structure inference problem, known to be NP-complete [14]. With our agenda-based paradigm, the semantic inference problem can be transformed into an MDP with finite states and finite actions. Let's assume the time and predicate domains have been divided into U and V intervals, respectively. Moreover, let's fix the episode horizon as T. Then the total numbers of states (denoted as $|S|$) and actions (denoted as $|A|$) of the MDP are of the orders of $\mathcal{O}(V^{|X|}U^{2|X|(T-1)})$ and $\mathcal{O}(V^{|X|}U^{2|X|T})$ with $|X|$ being the dimension of the signals in set X (see Problem 1), respectively. Using reinforcement learning to solve MDP has a well established complexity of $\mathcal{O}(|S||A|)$ [15]. Therefore, our semantic inference framework has a complexity that is exponential with respect to the dimension of the signals $|X|$, which is generally small (e.g., 1 for the case study in Section V) and the horizon length T.

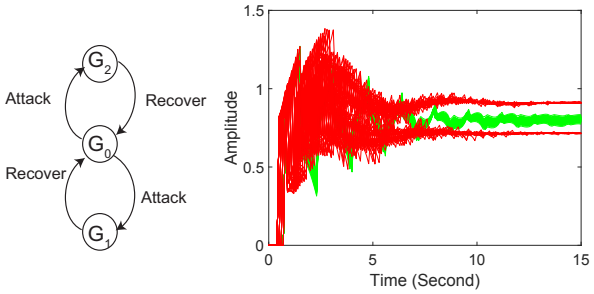
V. Case Study

In this section, we validate the performance of our proposed method with an academic case study.

Here we investigate a hybrid system with 3 modes as shown in Fig. 3 (a). Mode G_0 corresponds to the normal condition while modes G_1 and G_2 correspond to two attacked conditions. The attacker attacks the system randomly to generate the conditions. The transfer functions of the three modes are as follows:

$$\begin{aligned} G_0 &= e^{-0.8s}(0.8s^2 + s + 2 + 2\zeta)/(s^2 + s + 0.5), \\ G_1 &= e^{-(0.4+2|\zeta|)s}(0.8s^2 + s + 2)/(s^2 + s + 0.2), \\ G_2 &= e^{-(0.4+2|\zeta|)s}(0.8s^2 + s + 2)/(s^2 + s + 0.8), \end{aligned} \tag{10}$$

where $\zeta \sim \mathcal{N}(0, 0.1)$ is the uncertainty of the model. Fig. 3 (b) shows the set of step response signal, X, used in the case study, which are labeled by human agents.



(a) (b)

Fig. 3: a) The hybrid system model and b) X, the set of labeled signals used in the case study. The green signals are with modes G_0, i.e., those that are normal and belong to X^+, while the red ones are with mode G_1, G_2, i.e., those that are attacked and belong to X^-. The number of signals in X^- and X^+ are both one hundred.

We discretize the time domain $[0, 15]$, i.e., the range of all *time* attributes, with an interval of 1, and the scale domain $[0.5, 1.5]$, i.e., the range of all *predicate* attributes, with an interval of 0.2. With these time and scale bounds, we can define the proper STL attribute grammar $\mathcal{G}_{\mathcal{STL}}$. Then we apply Algorithm 1 (the horizon T is set to 3) to solve the semantic inference problem. The algorithm is terminated when either a satisfactory formula, i.e., a formula with a positive robustness degree $\rho(X, d)$, which indicates all the signals in X are correctly classified with the formula d, has been found or the limit M has been reached.

TABLE III: Comparsion results between our proposed method and the method in [8]

Our method			Method in [8]		
Time (s)	**Error** %	**Robustness** $\rho(X, d)$	**Time** (s)	**Error** %	**Robustness** $\rho(X, d)$
191	30	- 0.3571	188	50	-0.5262
392	16	- 0.2432	396	24	-0.2533
815	0	0.0452	819	11	-0.1684

In order the demonstrate the effectiveness and efficiency of the agenda-based semantic inference method proposed in this paper over state of the art methods, here we compare the performances of the proposed method with the method developed in [8]. Both methods can solve the semantic inference problem without the formula structure being given. The comparison result is shown in Table III. An eight-core HP desktop was used. Moreover, during the comparison, we controlled the number of episodes in the agenda-based method, and the number of learning cycles in the method developed in [8]. It can be seen that, with the former method, a satisfactory STL formula can be derived in 815 seconds, while such a formula cannot be obtained with the latter method within roughly the same number of seconds (the error is 11% and the robustness degree $\rho(X, d)$ is still negative after 819 seconds). The formula obtained by the agenda-based method is

$$\varphi = \Diamond_{[0,15]}(\Box_{[11,15]}(x < 0.9) \wedge \Box_{[11,15]}(x > 0.7)),$$

which quantitatively specifies the stable behavior of X^+ around the amplitude of 0.8, between 0.7 and 0.9 specifically.

VI. Conclusions

This paper introduced the problem of semantic inference for CPS, which infers the formal specification of a CPS with a parser. To reduce the potential combinatorial explosion inherent to the problem, we first formulated the process as an agenda-based parsing process, then as a Markov decision process, and finally used reinforcement learning to solve the problem. The performance of our proposed method was demonstrated with an academic case study. Further research will learn specifications with positive examples only.

References

[1] G. Chen, Z. Sabato, and Z. Kong, "Semantic parsing of automobile steering systems," in *Proceedings of the 8th International Conference on the Internet of Things*. ACM, 2018, p. 41.

[2] A. Van Lamsweerde, *Requirements engineering: From system goals to UML models to software*. Chichester, UK: John Wiley & Sons, 2009, vol. 10.

[3] S. A. Seshia, D. Sadigh, and S. S. Sastry, "Towards verified artificial intelligence," *arXiv preprint arXiv:1606.08514*, 2016.

[4] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals," in *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 2010, pp. 92–106.

[5] N. Ali, Y.-G. Guéhéneuc, and G. Antoniol, "Trustrace: Mining software repositories to improve the accuracy of requirement traceability links," *IEEE Transactions on Software Engineering*, vol. 39, no. 5, pp. 725–741, 2013.

[6] X. Jin, A. Donzé, J. V. Deshmukh, and S. A. Seshia, "Mining requirements from closed-loop control models," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 11, pp. 1704–1717, 2015.

[7] S. Jha, A. Tiwari, S. A. Seshia, T. Sahai, and N. Shankar, "Telex: Passive STL learning using only positive examples," in *International Conference on Runtime Verification*. Springer, 2017, pp. 208–224.

[8] Z. Kong, A. Jones, and C. Belta, "Temporal logics for learning and detection of anomalous behavior," *IEEE Transactions on Automatic Control*, vol. 62, no. 3, pp. 1210–1222, 2017.

[9] A. Bakhirkin, T. Ferrère, and O. Maler, "Efficient parametric identification for STL," in *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (part of CPS Week)*. ACM, 2018, pp. 177–186.

[10] B. Hoxha, A. Dokhanchi, and G. Fainekos, "Mining parametric temporal logic properties in model-based design for cyber-physical systems," *International Journal on Software Tools for Technology Transfer*, vol. 20, no. 1, pp. 79–93, 2018.

[11] S. Park and S.-C. Zhu, "Attributed grammars for joint estimation of human attributes, part and pose," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2372–2380.

[12] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic parsing on freebase from question-answer pairs," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 1533–1544.

[13] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[14] L. J. Guibas, J. E. Hershberger, J. S. Mitchell, and J. S. Snoeyink, "Approximating polygons and subdivisions with minimum-link paths," *International Journal of Computational Geometry & Applications*, vol. 3, no. 04, pp. 383–415, 1993.

[15] S. Koenig and R. G. Simmons, "Complexity analysis of real-time reinforcement learning," in *AAAI*, 1993, pp. 99–107.