Temporal-Logic-Based Semantic Fault Diagnosis With Time-Series Data From Industrial Internet of Things

Gang Chen, *Member, IEEE*, Mei Liu, and Zhaodan Kong, *Member, IEEE*

***Abstract*—The maturity of sensor network technologies has facilitated the emergence of an industrial Internet of Things (IIoT), which has collected an increasing volume of data. Converting these data into actionable intelligence for fault diagnosis is key to reducing unscheduled downtime and performance degradation, among other examples. This article formalizes a problem called semantic fault diagnosis— to construct the formal specifications of faults directly from data collected from IIoT-enabled systems. The specifications are written as signal temporal logic formulas, which can be easily interpreted by humans. To tackle the issue of the combinatorial explosion that arises, we propose an algorithm that combines ideas from agenda-based searching and imitation learning to train a policy that searches formulas in a strategic order. Specifically, we formulate the problem as a Markov decision process, which is further solved with a reinforcement learning algorithm. Our algorithm is applied to time-series data collected from an IIoT-enabled iron-making factory. The results show empirically that our proposed algorithm is both scalable to the size of the data set and interpretable, therefore allowing human users to take actions, for example, predictive maintenance.**

***Index Terms*—Fault diagnosis, industrial Internet of Things, reinforcement learning, signal temporal logic, time-series classification.**

I. Introduction

THE industrial Internet of Things (IIoT) is the application of the Internet of Things (IoT) to the realm of various industries [1]. The connection of devices equipped with sensing, communication, and actuation capabilities has dramatically transformed the operation of many existing and new industrial systems, and achieved numerous successes in terms of, e.g., rapid manufacturing of new products and real-time optimization of industrial production. However, the complex, connected, and dynamic nature of many IIoT-enabled systems also brings the issues of safety and security to a scale and level far beyond that of traditional industrial systems. A small performance degradation or security risk in one sensor/processor/actuator may lead to catastrophic consequences to the entire system. Time-series data has been demonstrated to have rich information about these performance degradation or security risk. As a result, recently, there has been a surge of interest in the development of new time-series-based fault diagnosis theories, techniques, and practices for IIoT-enabled systems to ensure their safe and secure operations [2].

Existing methods on fault diagnosis can be roughly divided into two classes: data-driven and knowledge-driven. Data-driven methods do not depend extensively on expert-crafted knowledge or the construction of a reasoning mechanism. They mainly rely on the application of machine learning techniques to extract fault features from IIoT data directly [3]–[5]. For example, the fault detection method in [6] used reinforcement learning to find the optimal fault vector (feature) for fault detection. Data-driven methods are powerful tools for systems that are ill or difficult to specify or understand [7]. One important issue with these generalized fault diagnosis methods, however, is their lack of interpretability. Data-driven methods have obtained excellent performance in fault diagnosis, but humans find it hard to understand why they can reach good results. Humans still play an indispensable role in many, if not all, IIoT applications, particularly when the system involved is safety-critical [8]. The fault features extracted by machine learning techniques generally reside in some high-dimensional feature space and their meanings can be hard for humans to grasp, in the context of, e.g., identifying the causal chain that may lead to a fault and generalizing the learned knowledge to other scenarios.

Knowledge-driven methods, on the other hand, rely heavily on mechanical principles and the empirical knowledge of human experts [9]. These methods put the semantic interpretation of the IIoT-enabled systems at their core, leading to what is called "semantic IIoT systems" [10]. However, currently, most of these knowledge-driven methods are developed for processing static data and require domain experts to manually construct the rules that can be used, e.g., for ontology-based reasoning [11]. The construction process itself is time-consuming. Moreover, the

Manuscript received October 30, 2019; revised February 25, 2020 and February 26, 2020; accepted March 21, 2020. Date of publication April 7, 2020; date of current version January 27, 2021. *(Corresponding author: Zhaodan Kong.)*

Gang Chen and Zhaodan Kong are with the Department of Mechanical and Aerospace, University of California, Davis, CA 95616 USA (e-mail: ggchen@ucdavis.edu; zdkong@ucdavis.edu).

Mei Liu is with the Department of Automation School of Electrical and Information Engineering, Tianjin University, Tianjin 300072, China (e-mail: liumeimei@tju.edu.cn).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2020.2984976

predefined rules can be incomplete and/or brittle. As more and more IIoT-enabled systems are becoming increasingly complex, it is hard to scale these methods up, at least not in a straightforward manner, and ensure the robustness of the predefined rules [12].

In this article, we combine the data-driven and knowledge-driven methodologies together into solving a problem called semantic fault diagnosis. The method begins with data collected from an IIoT-enabled system—thus is data driven. But different from existing data-driven methods, our framework attempts to infer/learn a set of formulas written in signal temporal logic (STL) [13]. These formulas are, first of all, easily understandable by humans and, second, provide a formal description of the fault(s) exhibited by the system, therefore enabling further knowledge-based reasoning [14].

In recent years, temporal logic has proved to be a powerful and natural framework to describe complex temporal properties of industrial systems [15]. Since temporal logic formulas describe temporal patterns between events in a form that is close to our way of thinking, they can be intelligible and easily acceptable by humans. The availability of software and hardware for real-time verification makes this approach very attractive in monitoring engineered systems. Many temporal logic-based formal languages have been applied to monitoring tasks, such as first-order temporal logic [16], metric temporal logic [17], and linear temporal logic [18]. But these languages are defined over the discrete event, which indicates that they are not suitable for a continuous system. To address this issue, STL has been defined over continuous signals and is perfect for time-series monitoring.

Techniques exist for learning STL formulas from time-series data, under the name requirement mining [19]–[22]. Most of these methods assume that the output of the requirement mining problem is a temporal logic formula φ_θ with a fixed structure but an unknown parameter θ. Under this assumption, with the help of the concept of "robustness degree" [13], the requirement mining problem can be transformed into an optimization problem, the goal of which is to find a parameter θ^*, such that φ_{θ^*} (the fixed structure together with the optimal parameter θ^*) can distinguish time-series data labeled as "positive" from those labeled as "negative" and, in the meantime, maximize the "distance" (defined with respect to the robustness degree of φ_{θ^*}) between the two data sets. For example, the binary search algorithm is used to find the optimal parameters for logic expression in [20].

A less restricted (but more general) scenario relaxes the fixed-structure assumption and attempts to infer a discriminative temporal logic formula without specifying its structure first [22]–[24]. The investigation of this scenario is still in its infancy. One major flaw shared by existing methods is they suffer from the issue of combinatorial explosion or curse of dimensionality. To use the English language as an analogy, they begin their search from formulas of length one (one-letter words) alphabetically and then proceed to formulas of length two (two-letter words); they will continue to search all syntactically correct formulas (English sentences) according to such a fixed order until an appropriate formula (sentence) has been found. For example, the method in [22] tries all the combinations of basic formulas according to a predefined order and then selects the best structure. Some methods try to deal with the problem, such as [24] starts by devising a data-driven statistical abstraction of the system. Then, it proposes general optimization strategies for selecting formulae with high satisfaction probability. Bombara *et al.* [25] use a decision tree to explore the structure of the formula, which grows the formula by adding nodes to the decision tree. These methods do not fully solve the scalability problem or are under some assumptions.

The semantic fault diagnosis problem solved in this article fundamentally belongs to the less restricted case of requirement mining. But it pertains to the IIoT domain. More importantly, to overcome the issue of combinatorial explosion, instead of following a predefined order, this article proposes a new agenda-based algorithm where syntactically correct formulas are ranked in an agenda (priority queue) according to a scoring function; at each step, the formula with the highest score (priority) is popped from the agenda and evaluated against the collected data set; the evaluation result, measured in terms of the formula's ability to distinguish positive and negative examples, is used to update the scoring function and subsequently the agenda. Essentially, our idea is inspired by imitation learning [26], [27]; we utilized the IIoT data to guide the search of discriminative formulas (with the English language analogy, this is similar to using English language patterns, e.g., the relative frequencies of words, appearing in the documents under investigation). We cast the agenda-based semantic fault diagnosis process as a Markov decision process (MDP), and solve the problem with reinforcement learning. Since the semantic fault diagnosis problem is a delayed reward (reward can only be obtained after all the actions are completed) and sequential action (a sequence of actions should be taken) process, it is suitable to use reinforcement learning to solve the problem. Specifically, the major contributions are as follows.

1) *Algorithm:* This article presents a semantic fault diagnosis problem and proposes an agenda-based, learning-enabled algorithm to tackle the issue of combinatorial explosion. To the best of our knowledge, our algorithm is the first instance where reinforcement learning has been used to solve requirement mining problems in the less restricted setting. The algorithm can easily be generalized to other domains where requirement mining is critical.
2) *Application:* The proposed algorithm is used to solve a semantic fault diagnosis problem with real data collected from 176 sensing devices on a blast furnace in an IIoT-enabled iron-making factory. The results demonstrate the scalability, diagnosis accuracy, and interpretability of our algorithm in the IIoT domain.

This rest of this article is organized as follows. Section II formulates the semantic fault diagnosis problem. Section III presents an agenda-based, learning-enabled algorithm to solve the problem. Section IV applies the algorithm on data collected from an iron-making factory. Finally, Section V concludes this article.

II. Semantic Fault Diagnosis Problem

In this section, we will first introduce STL, the logic used in this article. Then, we will provide an illustrative example to show the expressiveness of STL in IIoT fault diagnosis. Next, we will propose an attribute grammar, called STL attribute grammar, which can be used to construct syntactically correct STL formulas. Finally, we will present the formal definition of the semantic fault diagnosis problem.

A. Signal Temporal Logic

Given two sets A and B, $\mathcal{F}(A,B)$ denotes the set of all functions from A to B. Given a time domain $\mathbb{R}^+ := [0,\infty)$, a *continuous-time, continuous-valued signal* is a function $x \in \mathcal{F}(\mathbb{R}^+,\mathbb{R}^n)$. In this article, we use $x(t)$ to denote the value of signal x at time t.

Definition 1: STL is a temporal logic defined over signals [13]. Its syntax is defined recursively as

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

where τ_1 and τ_2 are nonnegative finite real numbers, and μ is a predicate over a signal, which can be defined as $l(x(t)) \sim c$ with $l \in \mathcal{F}(\mathbb{R}^n,\mathbb{R})$ being a function, $\sim \in \{\le,<,\ge,>\}$, and $c \in \mathbb{R}$ being 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* defined as follows.

Definition 2: STL is equipped with a quantitative semantics called *robustness degree* $\rho : \mathcal{F}(\mathbb{R}^+,\mathbb{R}^n) \times \Psi \to \mathbb{R}$ which maps an STL formula $\varphi \in \Psi$ and a signal $x \in \mathcal{F}(\mathbb{R}^+,\mathbb{R}^n)$ to a real value. $\rho(x,\varphi)$ indicates how far a signal x is away from satisfying STL formula φ and is defined as [13]

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

B. Illustrative Example

Example 1: STL formulas are quite expressive and interpretable. With knowledge on the syntax and semantics of STL, a human user can easily parse STL formulas. For instance, we can use an STL formula $\varphi_n = \Diamond_{[0,38.9]}(\Diamond_{[0,2.22]}(w_p < 0.88) \wedge \Box_{[2.22,11.4]}(w_p > 0.9))$ to distinguish normal and abnormal cold blast pressure signals shown in Fig. 1 (w_p in the formula is the cold blast pressure). φ_n can be dissected as follows: $\varphi_n := \Diamond_{[0,38.9]}\varphi'$, $\varphi' := \varphi_1 \wedge \varphi_2$, $\varphi_1 := \Diamond_{[0,2.22]}(w_p < 0.88)$, and $\varphi_2 := \Box_{[2.22,11.4]}(w_p > 0.9)$. φ_n reads as "the property φ' must eventually be satisfied at a time t between 0 and 38.9 hours"; φ' reads as "properties φ_1 and φ_2 must both be true at t to satisfy φ' at t"; φ_1 reads as "w_p must be smaller than 0.88 at least once within the next 2.22 hours"; and φ_2 reads as "w_p must always be larger than 0.9 within the next 2.22 and 11.4 hours". In summary, φ_n can be translated into an English

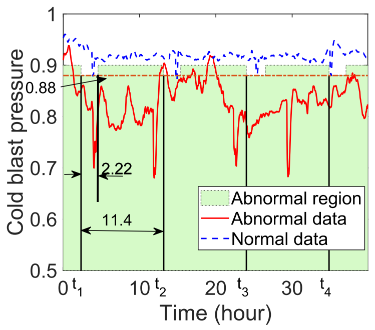

Fig. 1. Cold blast pressure time-series data collected from a blast furnace. $t_i, i = 1,\ldots,4$, are the time instances that φ' (see text) is satisfied.

TABLE I
PRODUCTION RULES OF $\mathcal{G}_{\mathcal{STL}}$

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

sentence "the following property must be satisfied at a time t between 0 and 38.9 hours: the blast pressure w_p must be smaller than 0.88 at least once within the next 2.22 hours (i.e., during the time interval $[t,t+2.22]$) and always be larger than 0.9 within the next 2.22 and 11.4 hours (i.e., during the time interval $[t+2.22,t+11.4]$)."

C. STL Attribute Grammar

To enable the learning process for semantic fault diagnosis, we define the following STL attribute grammar.

Definition 3: The *STL attribute grammar* $\mathcal{G}_{\mathcal{STL}}$ is an attribute grammar $\langle V_N, V_T, P, S, g\rangle$ with the following components [28]: $V_N = \{A,B\}$, 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 (1) and $\mu,\Diamond,\Box$ represent predicates and temporal operators with different attributes; $P = \{P_1,\ldots,P_7\}$ is the set of production rules, where the production rules are shown in Table I (there are five categories of rules, namely *Instance*, *Eventually*, *Always*, *Or*, and *And*); S is the start variable (or start symbol), used to represent the whole sentence (here $V_N = S$); 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 include *signal name*, *comparison operator*, and *constant*. To give an example, for a terminal node $\mu : (w_p < 0.88)$, its list of *time* attributes is $\mu.time = []$, which is empty, and its list

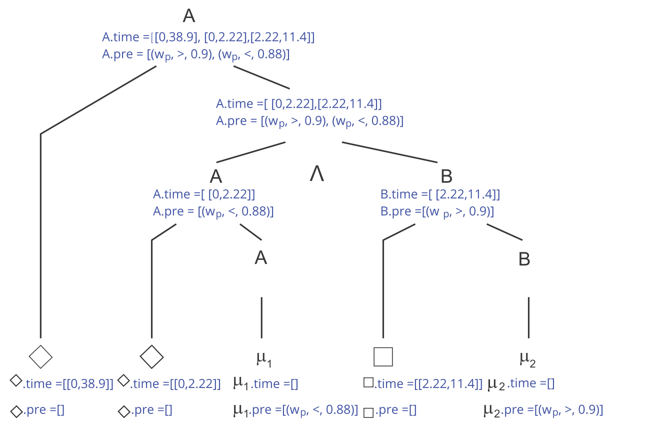

Fig. 2. The parsing tree of $\varphi_n = \Diamond_{[0,38.9]}(\Diamond_{[0,2.22]}(w_p < 0.88) \wedge \Box_{[2.22,11.4]}(w_p > 0.9))$. The *time* and *predicate* attributes of a node are shown immediately underneath the corresponding node.

of *predicate* attributes is $\mu.pre = [(w_p, <, 0.88)]$ (we use the notations $.pre$ and $.time$ throughout the article). Both types of attributes are synthesized [29]. For instance, production rule $P_6 : A \rightarrow A \vee B$ implies that $A.time = A.time \oplus B.time$ and $A.pre = A.pre \oplus B.pre$, where $\oplus$ denotes the concatenation operator of two lists, e.g., $[(w_p, <, 0.88)] \oplus [(w_p, >, 0.9)] = [(w_p, <, 0.88), (w_p, >, 0.9)]$.

Example 1: (Continued) This grammar can be best demonstrated with an example. Here, $V_N = \{A, B\}$, $V_T = \{\mu_1 := (w_p < 0.88), \mu_2 := (w_p > 0.9), \Box_{[2.22,11.4]}, \Diamond_{[0,38.9]}, \Diamond_{[0,2.22]}, \wedge, \vee\}$, $S = V_N$, P and g are defined in Table I, where P_1 has two forms as $A|B \rightarrow \mu_1, A|B \rightarrow \mu_2$. We want to generate formula φ_n in this illustrative example. It can be easily seen that the STL formula $\varphi_n = \Diamond_{[0,38.9]}(\Diamond_{[0,2.22]}(w_p < 0.88) \wedge \Box_{[2.22,11.4]}(w_p > 0.9))$ can be derived by following a sequence of production rules $A \rightarrow \Diamond A$, $A \rightarrow A \wedge B$, $A \rightarrow \Diamond A$, $A \rightarrow \mu_1$, $B \rightarrow \Box B$, $B \rightarrow \mu_2$. The parsing tree is shown in Fig. 2.

III. Agenda-Based, Learning-Enabled Algorithm

In this section, we will present our framework to solve the formula inference problem. The main challenge is the issue of combinatorial explosion as mentioned in Section I. The number of syntactically correct (thus candidate) formulas grows exponentially with respect to the length of the formula. Our algorithm adopts two main strategies to tackle the issue. First, inspired by agenda-based semantic parsing (the task of mapping natural language to semantic representations) [27], we develop an agenda-based framework to search for a satisfying discriminative formula (Section III-A). Second, we borrow ideas from imitation learning [26], [27], cast the agenda-based semantic fault diagnosis process as a MDP, and solve the problem with reinforcement learning.

A. Agenda-Based Strategy

Fig. 3 shows our agenda-based framework to solve the semantic fault diagnosis problem. The framework takes the STL attribute grammar $\mathcal{G}_{\mathcal{STL}}$ and a set of labeled signals X as the inputs and generates an optimal STL formula d^*. At step t, the *Agent* chooses a formula, d_t, from agenda Q and adds it to chart H (the roles of Q and H will be explained later) based on a policy $\pi(s_t)$ at current state s_t of the H and Q. The *Evaluator* evaluates the performance of the formula d_t with respect to X and the *Learner* updates the policy function according to the performance. Then the agenda Q is updated based on the formulas in H and d_t (the state is updated and will be described later). The *Agent* selects actions based on the current policy, which is related to the scoring function learned by the *Evaluator* and the *Learner*. Therefore, the efficiency and effectiveness of our agenda-based framework are completely determined by the quality of policy $\pi(\cdot)$. In Section III-C, we will focus on how to systematically learn $\pi(\cdot)$ from data. But first, let us elaborate on the rules to update agendas and charts.

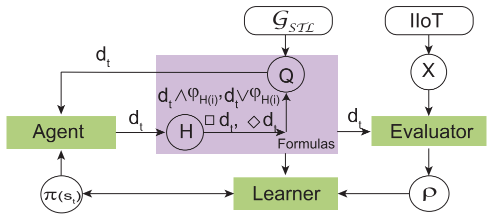

Fig. 3. Our agenda-based strategy to solve the semantic fault diagnosis problem.

TABLE II
AGENDA Q AND CHART H GENERATED IN THE FIRST THREE STEPS OF EXAMPLE 1

Step	Agenda Q	Chart H	Chosen d_t
0	φ_1, φ_2	-	φ_2
1	$\varphi_1, \Box\varphi_2, \Diamond\varphi_2$	φ_2	φ_1
2	$\Box\varphi_2, \Diamond\varphi_2, \Box\varphi_1,$ $\Diamond\varphi_1, \varphi_2 \vee \varphi_1, \varphi_2 \wedge \varphi_1$	φ_2, φ_1	$\varphi_1 \wedge \varphi_2$
3	$\Diamond(\varphi_1 \wedge \varphi_2), \cdots$	$\varphi_2, \varphi_1, \varphi_1 \wedge \varphi_2$	$\Diamond(\varphi_1 \wedge \varphi_2)$

1) Agenda and Chart Update:

Agenda Q and chart H are updated whenever the *Agent* chooses a formula from agenda Q. At step t, after the *Agent* has chosen a formula d_t from agenda Q, the agenda and chart are updated as follows:

$$\begin{aligned} &H \leftarrow H \cup sp(d_t), \\ &Q \leftarrow Q \setminus d_t \cup \{sp(\Box_{[a,b]}d_t) \cup, sp(\Diamond_{[a,b]}d_t)\} \\ &\qquad \{d_t \wedge \varphi_H(i), d_t \vee \varphi_H(i) \mid \forall \varphi_H(i) \in H\}. \end{aligned} \tag{2}$$

Specifically, $sp(\cdot)$ simplify the formula by combining the same temporal operator expansion, e.g., $\Box_{[a,b]}\Box_{[c,d]} \rightarrow \Box_{[a+c,b+d]}$, and formulas $\Box_{[a,b]}d_t, \Diamond_{[a,b]}d_t, d_t \wedge \varphi_H(i)$ and $d_t \vee \varphi_H(i), i = 1, \ldots, |H|$ are generated by production rules P_3, P_2, P_7, and P_6 and added to agenda Q. The temporal interval $[a, b]$ is generated randomly to avoid the syntactically and semantically equivalent cases.

Example 1: (Continued) Table II shows how agenda Q and chart H are updated in the first three steps of Example 1. Agenda Q is initialized with formulas φ_1 and φ_2 and chart H is initialized as an empty set. At step 0, the *Agent* chooses formula φ_2 from agenda Q and put it into chart H. The chosen formula is used to generate new formulas $\Box\varphi_2$ and $\Diamond\varphi_2$ with production rules P_3 and P_2 (as H is empty initially, production rules P_4 and P_5 are not applied). All these new formulas are added to agenda Q. At step 1, the *Agent* chooses formula φ_1, generating new formulas $\Box\varphi_1$, $\Diamond\varphi_1$, $\varphi_1 \wedge \varphi_2$, and $\varphi_1 \vee \varphi_2$ with production rules P_3, P_2, P_5, and P_4, respectively. The new formulas are added to Q and φ_1 is added to H. At step 2, the *Agent* chooses formula $\varphi_1 \wedge \varphi_2$, then formula φ_n in Example 1 can be generated with production rule P_2. The *Agent* can proceed with choosing a formula from agenda Q, putting it to chart H, generating new formulas, adding them to agenda Q, and continuing until the number of formulas in the chart, $|H|$, reaches the limitation T.

B. Problem Definition

Problem 1: (Semantic Fault Diagnosis): Given the STL attribute grammar $\mathcal{G}_{STL} =< V_N, V_T, P, g >$, a positive integer T, and two labeled sets of signals, X^+, positive signals of an IIoT-enabled system, and X^-, negative signals of the system, find an optimal policy $\pi(\cdot)$, such that after T steps, the *Agent* chooses a formula d_T, such that

$$\rho(X, d_T) = \min\big(\min_{\mathbf{x}\in X^+}(\rho(\mathbf{x}, d_T)), \min_{\mathbf{x}\in X^-}\left(\rho(\mathbf{x}, \neg d_T)\right)\big) \quad (3)$$

is expected to be maximized, where $X = X^+ \cup X^-$, $\rho(\mathbf{x}, d_T)$, and $\rho(\mathbf{x}, \neg d_T)$ denote the robustness degrees of $\mathbf{x}$ with respect to d_T and its negation, respectively. T denotes the number of steps used to generate d_T, which is equal to the number of formulas in H and the value is set based on experimental result, and $\neg d_T$ is the negation of formula d_T.

Here, positive signals can correspond to normal signals, while negative signals are those that are abnormal. However, the signals need not always be classified as such (remember we may wish to diagnose different types of faults; more details are provided in Section IV-B).

C. Reinforcement Learning for Solution

1) Agenda-Based Semantic Fault Diagnosis as MDP: The agenda-based semantic fault diagnosis process is essentially a sequential decision process. At each step, the *Agent* needs to decide on which formula to choose based on the current formulas in agenda Q; a sequence of decisions made by the *Agent* will finally lead to a formula d (or φ). Therefore, we can conveniently cast the agenda-based fault diagnosis process as a traditional MDP $\mathcal{M} = \langle S, A, \triangle(\cdot|\cdot,\cdot), r(\cdot|\cdot,\cdot), \gamma\rangle$ as follows (here we continue to use Example 1): the set of states $S = \langle Q, H\rangle$ 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 $A(s)$, are the formulas in the corresponding agenda Q; the transition probability $\triangle(s'|s, d)$ is either 1 or 0. It is 1 if formula d is chosen to transit state s, i.e., agenda Q and chart H, to state s', i.e., agenda Q' and chart H' (from one row of Table II to the next row); otherwise, it is 0; the immediate reward $r(s'|s, d)$ is the robustness degree over all the signals in X with respect to formula d, as defined in (3); $\gamma \in [0, 1]$ is a constant.

Here we would like to point out that we only use the MDP for the construction process of the agenda-based formulas, thereby enabling us to utilize reinforcement learning to solve the problem later. As in the mainstream reinforcement learning setting [30], the MDP model will not be explicitly constructed.

2) Reinforcement Learning of Optimal MDP Policy: In the following, we will present a reinforcement-learning-based algorithm to solve the semantic fault diagnosis problem in the context of the MDP formalism. We will first introduce features used to characterize formulas and then elaborate on two components, *Agent* and *Learner* (see Fig. 3).

Features: In this article, the set of features, which capture the essential characteristics of an STL formula, 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, conjunction, and disjunction operators, and 6) the logarithm of the number of "always" and "eventually" temporal operators. These features are selected based on the features in [27] and empirical trails, which have been demonstrated to have good performance.

Agent: The *Agent* takes the role of the actor, which, at step t, observes the current state s_t (i.e., the current chart H_t and agenda Q_t) and chooses an action (formula) $d_t \in A(s_t)$. A policy π induces a distribution over trajectories ε of the MDP (i.e., sequences of formulas constructed during the fault diagnosis process) as follows:

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

The *Agent* attempts to find an optimal policy π that maximizes the expected accumulative robustness degree over the distribution (4) as follows:

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

where $\rho(X, d_t)$ is the robustness degree for choosing d_t [as defined in (3)]. In our setting, the discounting factor γ is set to 0.2, which is quite small. Since the formula in the chart might not be on the same production branch, we use a small discount factor, such that we focus more on the current reward. We assume that the formulas in the chart with an optimal policy are on the same production branch, which is a reasonable assumption since we need a complex formula to reach better performance in terms of that simple formula cannot obtain good performance. Here we can approximate the reward with a linear dynamic system, due to the fact that we can find a policy, which can increase the performance of the formula step by step. When the formula is simple, the reward function can be monotonic, thus it can be approximated with a linear system.

The optimal policy π^* (5) is a time-varying one, i.e., $\pi(d_t|s_t)$. We approximate the policy $\pi(d_t|s_t)$ by a time-varying parametric function $\pi(d_t|s_t, w_t)$, where $w_t \in \mathbb{R}^F$ $(t = 0, \ldots, T-1)$ are the parameter vectors with F being the dimension of the feature space, and $W = \{w_0, w_1, \ldots, w_{T-1}\}$ denotes the parametric matrix [30]. The policy $\pi(d_t|s_t)$ defines a conditional probability density function of an 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 *Agent* to choose action d_t at state s_t. During the learning process, the agent will randomly choose a formula in Q at the beginning, and then a formula d_t will be chosen with probability $\pi(d_t|s_t)$. The preference function is a linear function, $h(d) = f(d)^T w$, where $f(d) \in \mathbb{R}^F$ is the feature vector, and $w \in \mathbb{R}^F$ is the parameter vector to be obtained. With such a parameterization method (6), (5) can be transformed into the following equation:

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

Now the problem of searching for an optimal policy π^* (with a search space of infinite dimensions) has been converted into the problem of searching for an optimal parameter w^* (with a search space of F dimensions).

Learner: The role of the *Learner* is to find an optimal value for w (7) by using reinforcement learning [30]. It can be observed from (6) that the state s only provides the support for the distribution, and the policy $\pi_w(d_t|s_t)$ depends only on the feature $f(d_t)$ and the parameter vector w. In this article, the *Learner* applies the Monte–Carlo policy gradient method to learn the optimal policy [30]. 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 by

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

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

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

Overall Algorithm: The overall algorithm to solve the semantic fault diagnosis problem is shown in Algorithm 1. Line 1 initializes the state s_0. During the initialization process, we choose a sequence of time instances $\tau = (t_0, t_1, \ldots, t_n)$ randomly as the *time* attributes. Combining these time bounds with temporal operators $\Box$ and $\Diamond$, we can get temporal operators with different time bounds. We treat the *predicate* attributes the same way. With the initialization process, the 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; chart H will be initialized as an empty set. During the training process, the size of s_0 can be small, while after the optimal policy has been obtained, a larger s_0 will lead to a better formula with a little computational cost. Line 6 updates the chart and the agenda. During the formula generation process, the agenda-based approach can make sure all the generated formulas are syntactically correct and avoid pathological cases. Moreover, the horizon limitation of the MDP limits the length of the formula.

Algorithm 1: Reinforcement-Learning-Enabled Fault Diagnosis.

```
Input: A set of labeled signals $X = X^+ \cup X^-$, STL
  attribute grammar $\mathcal{G}_{\mathcal{STL}}$, episode horizon $T$, learning
  rate $\alpha$, number of training episodes $M$
Output: The optimal parameter vector $w$.
1:  Initialize state $s_0$ by randomly generating 100 atomic
    STL formulas, having the form of $\Diamond_{[a,b]}(x_i \sim c)$ or
    $\Box_{[a,b]}(x_i \sim c)$, where $a, b, c \in \mathbb{R}, 0 \leq a < b$, are
    random number and $\sim \in \{\leq, \geq <, >\}$.
2:  Initialize each parametric vector $w_t$
    $(t = 0, \ldots, T-1)$, to vector $\mathbf{0} \in \mathbb{R}^F$.
3:  for $m = 1$ to $M$ do
4:    for $t = 0$ to $T - 1$ do
5:      Choose a formula from current agenda $Q$, with $d_t$
        having probability $\pi(d_t|s_t, w_t)$ been chosen.
6:      Update $(H, Q)$ to get state $s_{t+1}$ based on (2) and
        reward $\rho_t$ based on (3).
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

Finding an STL formula is a structure inference problem, known to be NP-complete. With our agenda-based paradigm, the semantic fault diagnosis problem can be transformed into an MDP with finite states and finite actions. Let us assume the time and predicate domains have been divided into U and V intervals, respectively. Moreover, let us fix the episode horizon as T. Then, the total number 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 $\mathcal{O}(|S||A|)$ [31]. Therefore, our framework has a complexity that is exponential with respect to the dimension of the signals $|X|$, which is generally small, and the horizon length T.

The complexity of the proposed algorithm is different from the decision tree approach in [25]. Denote the complexity of the algorithm in [25] as $C(N)$, which is defined as

$$C(N) = \Theta\left(N \cdot \left(1 + \int_1^x \frac{g(u)}{u^2} du\right)\right) \tag{10}$$

where N is the number of signals and $g(\cdot)$ is the complexity of the local optimization algorithm, and x is the local number of signals. $C(N)$ is a function of the N, since the algorithm partitions the signals with the decision tree and terminates when all the signals are partitioned. Obviously, the algorithm assumes all the signals can be partitioned with a decision tree, while there exist scenarios that the signals cannot be classified with a temporal logic-based decision tree due to the existence of noise. On the contrary, the algorithm in this article tries to learn a policy that maximizes the expected robustness degree, which can deal with noise signals by statistical testing mentioned in the following section. Moreover, with an optimal policy, a good

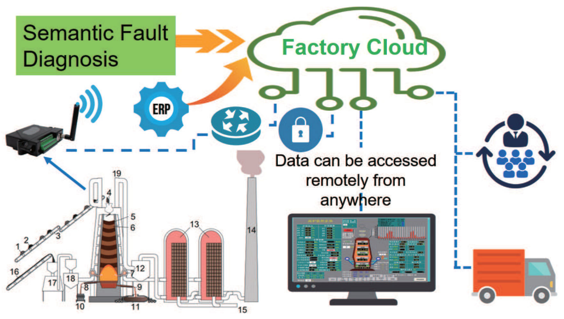

Fig. 4. Illustrative diagram of the IIoT-enabled blast furnace.

formula can be obtained by increasing the size of initial state s_0 with little computation cost.

IV. Case Study: Semantic Fault Diagnosis for an IIoT-Enabled Blast Furnace

During an iron-making process, raw iron-containing materials, e.g., sinter, pellet, and lump ore, and coke are charged into the top of a furnace with a charging system. These raw materials are pre-heated, creating a large amount of heat and producing gas consisting of carbon monoxide and hydrogen. The combustion process inside the furnace is complex, making it quite difficult to monitor the furnace and diagnose faults if they happen. The IIoT-enabled blast furnace illustration diagram is shown in Fig. 4, where the signals come from the wireless sensors, and the semantic fault diagnosis process is carried out in the cloud. In this section, based on data collected from an IIoT-enabled iron-making factory (Section IV-A), we will show how our proposed methodology can effectively and efficiently facilitate the fault diagnosis process (Section IV-B).

A. Collected Time-Series Data

Data covering three years of operation were collected from a blast furnace inside an iron-making factory. The furnace was equipped with an array of sensors. A total of 176 variables were measured. Only a fraction of these variables are used in the case study. They include blast kinetic energy (x_9) in kJ, theoretical flame temperature (x_{12}) in °C, total pressure drop (x_{19}) in kPa, total temperature drop (x_{30}) in °C, bosh gas volume (x_{10}) in m^3, enriched oxygen pressure (x_{17}) in MPa, resistance index (x_{28}), and blasting humidity (x_{29}). We choose these variables based on experts' recommendations and their performance in practice.

B. Semantic Fault Diagnosis for Blast Furnace

In this subsection, we will conduct three experiments on the collected data to illustrate both the effectiveness (in terms of interpretability and expressiveness) and scalability of our proposed semantic fault diagnosis algorithm.

1) Effectiveness of Our Algorithm: The blast furnace under investigation exhibited five types of conditions, a normal one and four abnormal ones, including *Low Stock Line*, *Cooling*, *Heating*, and *Chimney*. In the first experiment, we study whether it is possible to use Algorithm 1 to learn four separate STL formulas, φ_L, φ_C, φ_H, and φ_{Ch}, that can specify the four abnormal conditions, i.e., distinguishing one abnormal conditions from other abnormal ones as well as the normal one. We hand-pick four variables, x_9, x_{12}, x_{19}, and x_{30}, and use the time-series data related to these variables to learn the formulas. Specifically, the dataset X is of dimension $5 \times 4 \times 1440 \times 20$, where 5 is the number of conditions, 4 is the number of variables, 1440 is the number of time instances (the length of a signal), and 20 is the number of signals/repetitions/trials for each condition. Before the application of our algorithm, the data points corresponding to the same variable are normalized. Moreover, 80% of the signals (i.e., 16 out of 20) are used for training while the remaining 20% (i.e., 4 out of 20) are left for testing. *All the performance results presented in this subsection are evaluated with respect to the testing data.*

To further illustrate how to apply our algorithm, let us take the learning of φ_L, the STL formula for the *Low Stock Line* condition, as an example. First, we set X^+ as all the data corresponding to the *Low Stock Line* condition (a total number of $4 \times 1440 \times 16$ data points), and X^- as the data belonging to other conditions (a total number of $4 \times 4 \times 1440 \times 16$ data points). Then, we apply Algorithm 1 (the horizon T is set to 3 and the iteration limitation M is set to 5) to the data set $X = X^+ \cup X^-$. The algorithm will be terminated when either a satisfactory formula φ_L has been found or the iteration limitation has been reached. Here a satisfactory formula φ_L means that any signal in X^+ has a nonnegative robustness degree with respect to φ_L, while any signal in X^- has a negative robustness degree with respect to φ_L. Or equivalently, the robustness degree defined as (3) is nonnegative. We would like to point out that if such a satisfactory formula has been found, it has a 100% classification accuracy.

The four STL formulas, one for each abnormal condition, learned from the training data are listed in Table III. In the table, "Time" means the computational time, i.e., the time it takes for the algorithm to terminate when applied to the training signals, and "Robust." means the robustness degree, (3), of the testing signals with respect to the formulas learned from the training data. All the learned formulas have positive robustness degrees, implying that each one of them can perfectly distinguish its corresponding abnormal condition from the three other abnormal conditions and the normal one. Note that the variables used in the formulas are determined by the algorithm, and there may be other formulas that lead to positive robustness.

These formulas can easily be understood by humans and the semantic fault diagnosis process can be conducted by checking whether the signal is satisfied by the formula, and interpreting the meaning of the formula explicitly. For instance, formula φ_L can be translated into an English sentence, "the following property must be true at a time instance t between 0 and 14 400 seconds: within the next 6208 seconds, the total pressure drop must always be larger than 0.444 (normalized value) and the total temperature drop must always be smaller than 0.0211 (normalized value)." With such explicit knowledge, a human user can either take actions to maintain a small pressure drop and a large temperature drop or further study the furnace to determine factors which may prevent a large pressure drop and a small temperature drop.

TABLE III
SEMANTIC FAULT DIAGNOSIS RESULTS FOR THE FOUR ABNORMAL CONDITIONS

Fault Mode	Time (sec.)	Robust.	Formula ($\Diamond_{[0,14400]}$ is added manually for each formula for time-invariant property)
Low stock line	157	0.0161	$\varphi_L = \Diamond_{[0,14400]}(\Box_{[0,6208]}(x_{19} \geq 0.4444) \vee \Box_{[0,6208]}(x_{30} < 0.0211))$
Cooling	258	0.0230	$\varphi_C = \Diamond_{[0,14400]}(\Diamond_{[5324,9051]}(x_{19} \leq 0.9) \wedge \Box_{[5324,9051]}(x_9 \geq 0.2))$
Heating	24.7	0.0101	$\varphi_H = \Diamond_{[0,14400]}(\Box_{[10,4354]}(x_{12} \geq 0.651) \wedge \Box_{[7000,10550]}(x_{19} \geq 0.664))$
Chimney	241	0.0133	$\varphi_{Ch} = \Diamond_{[0,14400]}(\Diamond_{[2151,5354]}(x_{19} \geq 0.352) \wedge \Diamond_{[9976,12354]}(x_{30} \geq 0.461))$

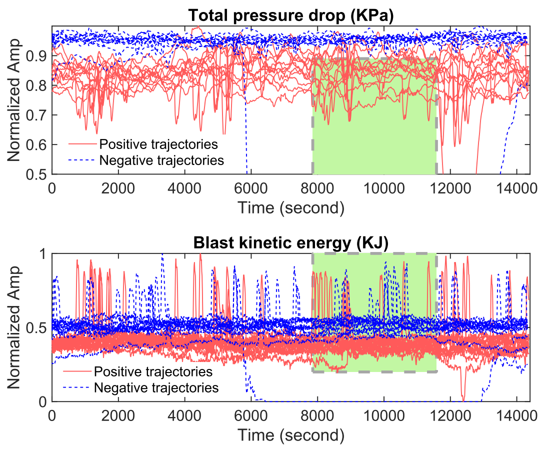

Fig. 5. Illustration of φ_C together with the relevant data.

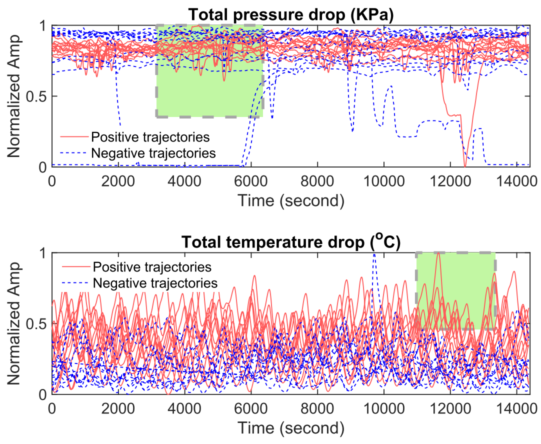

Fig. 6. Illustration of φ_{Ch} together with the relevant data. The green region in the top figure corresponds to fragment $\Diamond_{[2151,5354]}(x_{19} \geq 0.352)$, while the one in the bottom figure corresponds to fragment $\Diamond_{[9976,12354]}(x_{30} \geq 0.461)$.

Figs. 5 and 6 illustrate φ_C and φ_{Ch}, the STL formulas for the *Cooling* and *Chimney* conditions, respectively. We can use the same parsing method as explained in Section II-B to dissect φ_C and φ_{Ch}. For instance, $\varphi_C := \Diamond_{[0,14\,400]}(\Diamond_{[5324,9051]}(x_{19} \leq 0.9) \wedge \Box_{[5324,9051]}(x_9 \geq 0.2))$ can be parsed sequentially as follows: $\varphi_C := \Diamond_{[0,14\,400]}\varphi'$ (meaning "there is a time t between 0 and 14 400 s that property φ' must be true at t"), $\varphi' := \varphi_1 \wedge \varphi_2$ (meaning "properties φ_1 and φ_2 must be true at t at the same time"), $\varphi_1 := \Diamond_{[5324,9051]}(x_{19} \leq 0.9)$ (meaning that "at t, x_{19} must be less than or equal to 0.9 (the upper bound of the green region shown in the top plot of Fig. 5) for at least one time within the next 5324 and 9051 s"), and $\varphi_2 := \Box_{[5324,9051]}(x_9 \geq 0.2)$ (meaning that "at t, x_9 must be larger than than or equal to 0.9 (the lower bound of the green region shown in the bottom plot of Fig. 5) for at least one time within the next 5324 and 9051 s"). Actually, as shown in the top plot of Fig. 5, φ_1 is able to separate all the red trajectories (positive examples) from the blue ones (negative examples), except for one blue trajectory, which can subsequently be described by φ_2, illustrated by the bottom plot of Fig. 5. Figs. 5 and 6 also show the expressiveness of STL formulas for the purpose of semantic fault diagnosis. For instance, we cannot simply use thresholds to distinguish the positive (red) and negative (blue) signals shown in Fig. 5. However, when we combine thresholds with "always" $\Box$ and "eventually" $\Diamond$ operators, such as φ_C, the two sets of signals are now distinguishable.

2) Scalability of Our Algorithm: In the second experiment, we validate the scalability of our proposed algorithm by investigating the change in computational time as the number of variables increases from four to eight. Specifically, we solve the semantic fault diagnosis problem in a similar fashion as in the first experiment. The only difference is that the list of variables is expended to include x_9, x_{12}, x_{19}, x_{30}, x_{10}, x_{17}, x_{28}, and x_{29}. Their corresponding robustness degrees (evaluated against testing data) and computational times (evaluated against training data) are used as the evaluation metric. We would like to point out that the reason we chose x_9, x_{12}, x_{19}, and x_{30} in the first experiment is that these four variables have been shown to be good indicators based on the experience of blast furnace operators. These variables are currently used in conditioned monitoring in practice [32]. Interestingly, our results show that some other variables, such as x_{10} regarding the *Cooling* fault, have comparable capabilities in terms of specifying faults [quantified by (3)]. These variables may require further investigation.

The results are shown in Fig. 7, which shows that the computational time of our algorithm increases roughly linearly with respect to the number of variables (the size of the data set X is $5 \times 4 \times 1440 \times 16$ with four variables and $5 \times 8 \times 1440 \times 16$ with eight variables). While the robustness degrees are not linearly increased with respect to the number of variables. They are random values. There are two main reasons for this. First, the algorithm is terminated when a satisfactory formula has been found, and this formula may not be the optimal one. Second, the formula found in different settings contains different variables. As shown in Table IV, the formulas for *Heating* have different variables. These formulas will lead to an inconsistent robustness degrees pattern. Along the same line of thought, the

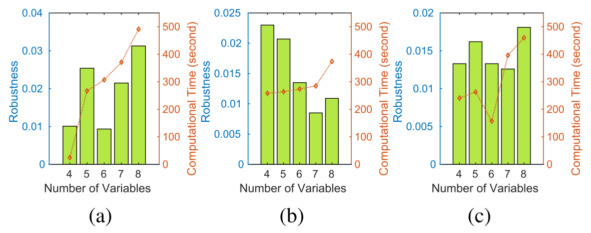

Fig. 7. Obtained robustness degrees and computational times with respect to the different numbers of variables used in the semantic diagnosis of three abnormal conditions. (a) Heating. (b) Cooling. (c) Chimney.

TABLE IV
FRACTION OF SEMANTIC FAULT DIAGNOSIS RESULTS WITH EIGHT VARIABLES

Fault-No.	Formula
$\varphi_H - 5$	$\Diamond_{[0,14400]}(\Diamond_{[3337,12776]}(x_{12} \geq 0.768))$
$\varphi_H - 6$	$\Diamond_{[0,14400]}((\Diamond_{[3337,12776]}((x_{17} < 0.0211))$ $\vee(\Box_{[3337,12776]}(x_{12} \geq 0.637))$
$\varphi_H - 7$	$\Diamond_{[0,14400]}(\Box_{[4590,9963]}(x_{10} \geq 0.325))$
$\varphi_H - 8$	$\Diamond_{[0,14400]}(\Box_{[7000,10550]}(x_{19} \geq 0.664))$

STL formulas obtained with respect to different variables, taken together, may offer a more comprehensive picture of possible mechanisms that may lead to faults. Let us take the *Heating* fault as an example. Formulas in Table IV show that such a fault can be caused or indicated by a large bosh gas volume, formally specified by $\varphi_H - 7 := \Diamond_{[0,14\,400]}(\Box_{[4590,9963]}(x_{10} \geq 0.325))$, or a large total total pressure drop, formally specified by $\varphi_H - 8 := \Box_{[7000,10\,550]}(x_{19} \geq 0.664))$. Such data-generated knowledge can be utilized by human users to improve the safety and efficiency of the IIoT-enabled blast furnace, which provides a fine example of how data-driven and knowledge-driven procedures can be integrated holistically.

In order to demonstrate both the effectiveness and the efficiency of our approach over other state-of-the-art methods, we compared our method with that developed in [22] for the low stock line diagnosis. Both methods can find an STL formula for semantic fault diagnosis. The results are shown in Table V. An eight-core HP desktop was used. For these comparisons, we controlled the number of episodes in the agenda-based method, and the number of learning cycles for [22]. We used five-fold cross-validation to test the performance. Both methods were repeated 10 times. Table V shows the average time, average error rate, and standard deviation among the 50 results, denoted as T, $\bar{\varepsilon}$, and ε_σ, respectively. The results show that, with the former method, a satisfactory STL formula can be derived in 265 s, while such a formula cannot be obtained with the latter method within roughly 1000 s (the error is 6% and the robustness is still negative after 1033 s). One reason for this is due to the lack of scalability for the latter method.

For the purposes of optimal policy evaluation, a robust metric to use is the *success rate* (SR) for each environment. The agenda-based learning tasks are designed so that the successful rewards are positive, while negative rewards indicate a failure.

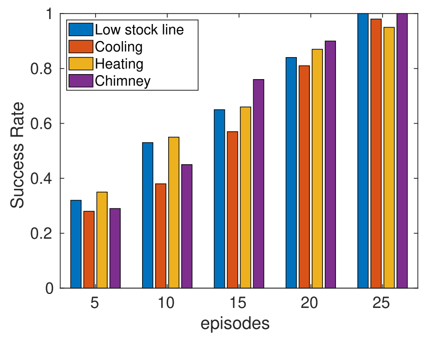

Fig. 8. Average success rates of different faults at chosen episodes.

Given a list of rewards R, the success rate is computed as $\sum \mathbf{1}[R \geq 0]/|R|$. The average success rate for optimal polices, trained within 300 s CPU time, is shown in Fig. 8 and the five-fold cross-validation is used to evaluate the performance. The average shows the algorithm can achieve a high success rate within 300 s (25 episodes).

Table V and Fig. 8 show the performance of the proposed method increases almost linearly with episodes, while the method in [22] cannot increase performance with time linearly. The method in [22] searches the structure of the formula first based on some predefined order, then finds the optimal parameters for the formula. When the length of the formula increases, the number of parameters that need to be optimized will increase. Within relatively the same CPU time, this method cannot reach the optimal parameters. Moreover, searching along a predefined order may lead to local optimal formula.

3) Statistical Testing: Machine learning algorithm usually is robust to noise. The learning algorithm in this article chooses formula based on the current policy, with d_t having probability $\pi(d_t|s_t, w_t)$ been chosen, but the agenda is initialized randomly. The probabilistic approach indeed may raise failure percentile. However, since the there is noise in the data, probabilistic approach has high probability to overcome the effect of noise, which can be illustrated with statistical testing.

Let us consider a t-test for a given STL formula: (a) the robustness for a baseline fault signal is a random variable having a normal distribution with unknown mean $\bar{\rho}$ and unknown standard deviation $\bar{\rho}_\sigma$; and (b) the robustness for a signal that must be diagnosed is also normally distributed with unknown mean $\hat{\rho}$ and unknown standard deviation $\hat{\rho}_\sigma$. The t-test compares means of values of two fragments of time series, and thereby a whole time-series fragment for a fault occurrence is tested. In other words, if the result of the test is that the null hypothesis is rejected, the current signal cannot be categorized as faults in the baseline samples.

In several cases in our study, the nature of the robustness degree does not imply a normal distribution, due to the operators in the given STL formula. In this case, the objective of the fault classification problem is to test the equality of value distributions in the robustness from the baseline and tested operating modes. For this purpose, the two-sample Kolmogorov–Smirnov test

TABLE V
COMPARISON RESULTS BETWEEN OUR PROPOSED METHOD AND THE METHOD IN [22]

	15 episodes/10 cycles			20 episodes/20 cycles			25 episodes/40 cycles		
	$T(s)$	$\bar{\varepsilon}(\%)$	ε_σ	$T(s)$	$\bar{\varepsilon}(\%)$	ε_σ	$T(s)$	$\bar{\varepsilon}(\%)$	ε_σ
Our method	151	35	0.063	216	16	0.035	**265**	**0**	**0**
Method in [22]	255	42	0.042	512	18	0.031	**1033**	**6**	**0.026**

TABLE VI
STATISTICAL TESTING RESULTS

Test Method	**t-test**	**K-S test**
Reject null hypothesis	18	17
Accept null hypothesis	2	3

(K-S test) is used. As a result, a chosen window of a sensor time series is considered an outlier if a null hypothesis of value distribution equality is rejected with a level of significance.

During the *t*-test and K-S test, the significance level is set to 0.05. To test the hypothesis, we segmented the signal that to be diagnosed using a window of length 10000 and used the signal in the window to calculate the robustness. Next, we moved this window forward 20 steps along the time axis and repeated the calculations. The resulting trace of robustness values was used to determine the distribution. To test φ_L, the other classes of conditions' signals are used as baseline samples and the 20 low stock line signals are tested. The results are shown in Table VI, which shows the formula is robust to noise.

V. CONCLUSION

This article solved a semantic fault diagnosis problem, where formal, but human-understandable specifications of faults can be learned from data collected from IIoT-enabled systems. The algorithm proposed in this article adopted two mechanisms to tackle the issue of combinatorial explosion: an agenda-based strategy and imitation learning. The scalability and effectiveness of the algorithm were demonstrated by a case study, where the algorithm was applied to sensor data collected from an IIoT-enabled blast furnace inside an iron-making factory.

REFERENCES

[1] S. Jeschke, C. Brecher, T. Meisen, D. Özdemir, and T. Eschert, "Industrial internet of things and cyber manufacturing systems," in *Industrial Internet of Things*. Berlin, Germany: Springer, 2017, pp. 3–19.

[2] S. Shao, S. McAleer, R. Yan, and P. Baldi, "Highly accurate machine fault diagnosis using deep transfer learning," *IEEE Trans. Ind. Informat.*, vol. 15, no. 4, pp. 2446–2455, Apr. 2019.

[3] R. Razavi-Far *et al.*, "Information fusion and semi-supervised deep learning scheme for diagnosing gear faults in induction machine systems," *IEEE Trans. Ind. Electron.*, vol. 66, no. 8, pp. 6331–6342, Aug. 2019.

[4] R. Razavi-Far, E. Hallaji, M. Farajzadeh-Zanjani, and M. Saif, "A semi-supervised diagnostic framework based on the surface estimation of faulty distributions," *IEEE Trans. Ind. Informat.*, vol. 15, no. 3, pp. 1277–1286, Mar. 2019.

[5] Y. Wang, Z. Wei, and J. Yang, "Feature trend extraction and adaptive density peaks search for intelligent fault diagnosis of machines," *IEEE Trans. Ind. Informat.*, vol. 15, no. 1, pp. 105–115, Jan. 2019.

[6] D. Zhang, Z. Lin, and Z. Gao, "A novel fault detection with minimizing the noise-signal ratio using reinforcement learning," *Sensors*, vol. 18, no. 9, p. 3087, 2018.

[7] P. Henriquez, J. B. Alonso, M. A. Ferrer, and C. M. Travieso, "Review of automatic fault diagnosis systems using audio and vibration signals," *IEEE Trans. Syst. Man Cybern.: Syst.*, vol. 44, no. 5, pp. 642–652, May 2014.

[8] U. Lindqvist and P. G. Neumann, "The future of the internet of things," *Commun. ACM*, vol. 60, no. 2, pp. 26–30, 2017.

[9] Z. Gao, C. Cecati, and S. Ding, "A survey of fault diagnosis and fault-tolerant techniques—Part II: Fault diagnosis with knowledge-based and hybrid/active-based approaches," *IEEE Trans. Ind. Electron*, vol. 62, pp. 3768–3774, Jun. 2015.

[10] Q. Zhou, P. Yan, and Y. Xin, "Research on a knowledge modelling methodology for fault diagnosis of machine tools based on formal semantics," *Adv. Eng. Informat.*, vol. 32, pp. 92–112, 2017.

[11] R. Agarwal *et al.*, "Unified IoT ontology to enable interoperability and federation of testbeds," in *Proc. IEEE 3rd World Forum Internet Things*, 2016, pp. 70–75.

[12] S. A. Seshia, S. Hu, W. Li, and Q. Zhu, "Design automation of cyber-physical systems: Challenges, advances, and opportunities," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 36, no. 9, pp. 1421–1434, Sep. 2017.

[13] A. Donzé, "On signal temporal logic," in *Proc. Int. Conf. Runtime Verification*, 2013, pp. 382–383.

[14] E. M. Clarke Jr., O. Grumberg, D. Kroening, D. Peled, and H. Veith, *Model Checking*. Cambridge, MA: MIT Press, 2018.

[15] S. Bufo, E. Bartocci, G. Sanguinetti, M. Borelli, U. Lucangelo, and L. Bortolussi, "Temporal logic based monitoring of assisted ventilation in intensive care patients," in *Proc. Int. Symp. Leveraging Appl. Formal Methods Verification Validation*, 2014, pp. 391–403.

[16] K. Havelund, D. Peled, and D. Ulus, "First order temporal logic monitoring with BDDs," in *Proc. IEEE Formal Methods Comput. Aided Des.*, 2017, pp. 116–123.

[17] D. Basin, B. N. Bhatt, and D. Traytel, "Almost event-rate independent monitoring of metric temporal logic," in *Proc. Int. Conf. Tools Algorithms Construction Anal. Syst.*, Springer, 2017, pp. 94–112.

[18] F. M. Maggi, M. Montali, M. Westergaard, and W. M. Van Der Aalst, "Monitoring business constraints with linear temporal logic: An approach based on colored automata," in *Proc. Int. Conf. Bus. Process Manage.*, Springer, 2011, pp. 132–147.

[19] L. Nenzi, S. Silvetti, E. Bartocci, and L. Bortolussi, "A robust genetic algorithm for learning temporal specifications from data," in *Proc. Int. Conf. Quantitative Eval. Syst.*, Springer, 2018, pp. 323–338.

[20] X. Jin, A. Donzé, J. V. Deshmukh, and S. A. Seshia, "Mining requirements from closed-loop control models," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 34, no. 11, pp. 1704–1717, Nov. 2015.

[21] M. Vazquez-Chanlatte, J. V. Deshmukh, X. Jin, and S. A. Seshia, "Logical clustering and learning for time-series data," in *Proc. Int. Conf. Comput. Aided Verification*, Springer, 2017, pp. 305–325.

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

[23] D. Neider and I. Gavran, "Learning linear temporal properties," in *Proc. IEEE Formal Methods Comput. Aided Des.*, 2018, pp. 1–10.

[24] E. Bartocci, L. Bortolussi, and G. Sanguinetti, "Data-driven statistical learning of temporal logic properties," in *Proc. Int. Conf. Formal Model. Anal. Timed Syst.*, Springer, 2014, pp. 23–37.

[25] G. Bombara, C.-I. Vasile, F. Penedo, H. Yasuoka, and C. Belta, "A decision tree approach to data classification using signal temporal logic," in *Proc. 19th Int. Conf. Hybrid Syst.: Computation Control*, 2016, pp. 1–10.

[26] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Comput. Surv.*, vol. 50, no. 2, pp. 21–56, 2017.

[27] J. Berant and P. Liang, "Imitation learning of agenda-based semantic parsers," *Trans. Assoc. Comput. Linguistics*, vol. 3, pp. 545–558, 2015.

[28] J. E. Hopcroft, R. Motwani, and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation: Pearson New International Edition*. Pearson Higher Ed., London, England, 2013.

[29] S. Park and S.-C. Zhu, "Attributed grammars for joint estimation of human attributes, part and pose," in *Proc. IEEE Int. Conf. Comput. Vision*, 2015, pp. 2372–2380.

[30] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 2018.

[31] S. Koenig and R. G. Simmons, "Complexity analysis of real-time reinforcement learning," in *Proc. Assoc. Advancement Artif. Intell.*, 1993, pp. 99–107.

[32] B. Zhou, H. Ye, H. Zhang, and M. Li, "Process monitoring of iron-making process in a blast furnace with PCA-based methods," *Control Eng. Pract.*, vol. 47, pp. 1–14, 2016.

Gang Chen (Member, IEEE) received the bachelor's and master's degrees in mechanical engineering from Shanghai Jiao Tong University, Shanghai, China, in 2012 and 2015, respectively, and the Ph.D. degree in mechanical and aerospace engineering from the University of California, Davis, CA, USA, in 2020.

His current research interests include machine learning, formal methods, control, signal processing, and fault diagnosis.

Mei Liu received the B.Sc. degree in mathematics from the China University of Mining and Technology, Xuzhou, China, in 2012, and the Ph.D. degree in control science and engineering from the University of Science and Technology of China, Hefei, China, in 2017.

From August 2017 to February 2019, she worked with The University of Hong Kong and The Hong Kong Polytechnic University as a Research Associate/Assistant. She is currently an Associate Professor with the Department of Automation, Tianjin University, Tianjin, China. Her current research interests include negative imaginary systems, positive real systems, state–space symmetric systems, and cyber–physical systems.

Zhaodan Kong (Member, IEEE) received the bachelor's and master's degrees in astronautics and mechanics from the Harbin Institute of Technology, Harbin, China, in 2004 and 2006, respectively, and the Ph.D. degree in aerospace engineering with a minor in cognitive science from the University of Minnesota, Twin Cities, MN, USA, in 2011.

Before joining the University of California (UC) at Davis, CA, USA, in 2015, he was a Postdoctoral Researcher with the Laboratory for Intelligent Mechatronic Systems and the Hybrid and Networked Systems Lab, Boston University, Boston, MA, USA. He is currently an Associate Professor in mechanical and aerospace engineering with the UC Davis. His current research interests include control theory, machine learning, formal methods, and their applications to human–machine systems, cyber-physical systems, and neural engineering.