Complex Event Recognition From Discrete Sensor Data With a Discrete Event System Framework ⋆

Yu Liu * **, Shaolong Shu** * **, Feng Lin** **

* *School of Electronics and Information Engineering, Tongji University, Shanghai, China (e-mail: yuliu199711@tongji.edu.cn, shushaolong@tongji.edu.cn).*
** *Department of Electrical and Computer Engineering, Wayne State University, Detroit, MI 48202, USA (e-mail: flin@wayne.edu)*

Abstract: Recognizing complex events revealed by sensor readings is an increasingly crucial task that serves as the foundation for system monitoring and decision-making. In this paper, we investigate the recognition problem for one class of complex events that can be represented by a sequence of discrete sensor outputs. We call the outputs of discrete sensors as sensor events. We use an automaton to describe all the sequences of sensor events that can be generated in the given system. The complex events to be recognized is a set of sequences of events and can be represented by the marked language of automaton. For a given sensor event sequence, we introduce the notation "partition" to stand for a possible complex event sequence. Then the problem is translated to find all the possible partitions for the given sensor event sequence. By constructing an augmented automaton that includes all the possible partitions, we find sufficient and necessary conditions for the existence of solutions. We then find an algorithm to verify the conditions. Finally, an online complex event recognition procedure is proposed to determine the occurred complex events when the complex event problem is solvable.

Keywords: Discrete event systems, Complex event recognition, Discrete sensor data, Augmented automaton, State pair.

1. INTRODUCTION

In recent years, complex event recognition has received increasing attention from researchers in different fields. A successful and accurate complex event recognition approach is important for practical applications including health monitoring (Wu et al. (2016)), social networks (Shi et al. (2019)), power grid (Wiot (2004)), etc. Recognizing and comprehending the complex events revealed by sensor readings is a crucial task that serves as the foundation for system monitoring and decision-making.

Techniques for complex event recognition are classified into two primary categories depending on the type of complex events: data-based approaches and model-based approaches. Researchers favor data-based approaches when they do not have a priori knowledge of the complex events to be recognized or the generation mechanisms of complex events are not evident. This type of complex event detection is also known as anomaly detection (Chandola et al. (2008)), which aims at identifying unspecific complex events from raw data. Due to the inherent lack of labeled data, anomaly detection is typically treated as an unsupervised machine-learning task. Li et al. (2019) propose an unsupervised multivariate anomaly detection method, using the LSTM-RNN as the base model in the GAN framework to capture the temporal correlation of time series. Zong et al. (2018) employ Auto-Encoder (AE) as a deep learning model for anomaly detection by inspecting its reconstruction errors. Dai and Gao (2013) mentioned the K-Nearest Neighbor algorithm and derived anomaly scores based on average distance. The data-based method provides a feasible recognition method for complex event recognition of unknown patterns, but its high resource consumption and extended detection time limit its application in many scenarios.

Model-based approaches can well address the complex event recognition problem when the generation pattern of the complex events to be identified is well characterized. Although data-based approaches can also accomplish the complex event recognition problem when the pattern is explicit, model-based approaches have the advantages of low resource consumption, high interpretability and accuracy, and better recognition speed, and are widely used in many resource-constrained scenarios, such as complex event recognition in the sensor domain. The detection of complex events that feature evident patterns can be translated into the detection of sequences of events. Brenna et al. (2009) gives an evaluation of alternative approaches for distributing an event processing system that is based on NFAs. Ozer et al. (2011) demonstrate how utilizing Petri net can assist us to model and recognize complex events in large 3D scientific data sets. Sadoghi and Jacobsen (2011) proposed a tree-based approach that achieves superiority in comparison with previous index structures designed for

⋆ The authors of this paper are supported by the National Natural Science Foundation of China under Grants 62073242 and 61773287.

matching expressions, and graph-based alternatives have been examined intensively as well. (Chakravarthy and Mishra (1994), Gruber et al. (1999)).

In this study, we focus on complex events which can be revealed by discrete sensor readings. We call the outputs of discrete sensors as sensor events. A complex event is represented by a sequence of sensor events. The occurrence of sequences of sensor events is constrained by the dynamics of practical systems. Hence we use an automaton to describe all the sequences of sensor events that can be generated in the given system. The complex events to be recognized is a set of sequence of events and can be represented by the marked language of automaton. The problem to be investigated becomes to determine the occurred complex event sequence from the observed sensor event sequence. For a given sensor event sequence, there may be multiple possible complex event sequences. We introduce the notation "partition" to stand for a possible complex event sequence. Then the problem is translated to find all the possible partitions for the given sensor event sequence. By constructing an augmented automaton that includes all the possible partitions, we find the sufficient and necessary conditions under which the complex event recognition problem has solutions. We then construct an automaton in which each state is a state pair of the augmented automaton and successfully find an algorithm to verify the conditions. Finally, we propose an online complex event recognition procedure to determine the occurred complex events when the complex event problem is solvable.

In discrete event systems, related work is summarized as follows. Genc and Lafortune (2006), and Jeron et al. (2006) considered the pattern diagnosis problem. They modeled fault patterns as sequences of events. The goal is to determine the occurrence of faults or predict their occurrence in advance. They propose algorithms to verify the solvability of the pattern diagnosis problem. Ye and Dague (2012) extended the results of Genc and Lafortune (2006) and Jeron et al. (2006) into distributed discrete event systems. Jin et al. (2008) investigated the recognition problem of complex event generated in the RFID data stream and the main contribution is to find an algorithm to determine the occurrence of complex events online from the sequence of sensor events. Saives et al. (2015) adopted a similar algorithm as Jin et al. (2008) to determine the occurrence of activities of inhabitants in the smart home which is modeled as sequences of sensor events online. Viard et al. (2020) considered the probability of occurrence of events and solved the same problem in a probabilistic automaton framework.

Compared with the aforementioned related work, the work of this paper is different and novel in the following aspects. 1. Our problem is to find the unique occurred complex event sequence for any observed sensor events while the problem introduced in Genc and Lafortune (2006), Ye and Dague (2012), and Jeron et al. (2006) is to determine whether patterns (similar to complex events) occur or not. 2. We propose an algorithm to verify the existence of solutions for our problem. Our algorithm is implemented by constructing an augmented automaton G_c which is proposed for the first time. 3. The online complex event recognition procedure is more effective than these algorithms in Jin et al. (2008), Saives et al. (2015) and Viard et al. (2020) by adopting the property that the occurred complex event is unique.

The rest of the paper is organized as follows. Section II introduces the discrete event system and some necessary notations. Section III formally states the complex event recognition problem. Section IV finds an algorithm to check the existence of solutions, and Section V proposes an online complex event recognition procedure to determine the occurred complex events. Finally, we conclude the paper in Section VI.

2. DISCRETE EVENT SYSTEMS

A discrete event system is modeled by a deterministic automaton as

$$G = (Q, \Sigma, \delta, q_0, Q_m)$$

where Q is the finite state set, Σ is the finite event set, $\delta : Q \times \Sigma \to Q$ is the state transition function that defines the dynamics of the automaton. The transition function δ is extended to $\delta : Q \times \Sigma^* \to Q$ in the usual way, where Σ^* denotes the Kleene closure of Σ. $\Gamma(q)$ denotes the set of events generated at state $q \in Q$. The initial state is q_0. Q_m is the marked state set. If we do not care for the marked states, we can re-write G as $G = (Q, \Sigma, \delta, q_0)$.

We use $\delta(q, s)!$ to denote that $\delta(q, s)$ is defined. The language generated by G is defined as:

$$L(G) = \{s \in \Sigma^* : \delta(q_0, s)!\}$$

The language marked by G is

$$L_m(G) = \{s \in L(G) : \delta(q_0, s) \in Q_m\}$$

As usual, we extend an deterministic automaton into a nondeterministic automaton as

$$G_{nd} = (Q, \Sigma, \delta_{nd}, q_0, Q_m)$$

where the state transition function is defined as $\delta_{nd} : Q \times \Sigma \to 2^Q$. The language generated by G_{nd} is defined as:

$$L(G_{nd}) = \{s \in \Sigma^* : \delta(q_0, s)!\}$$

The language marked by G_{nd} is

$$L_m(G_{nd}) = \{s \in L(G_{nd}) : \delta(q_0, s) \cap Q_m \neq \emptyset\}$$

For a string s, s' and t in Σ^*, if $s't = s$, we say s' is a prefix of s and t is a suffix of s. The set of all prefixes of s is denoted as

$$Pre(s) = \{s' \in \Sigma^* : (\exists t \in \Sigma^*)s't = s\}$$

Especially, we define the set $Pre^+(s)$ as

$$Pre^+(s) = Pre(s) - \{\varepsilon\}$$

The set of all suffixes of s is denoted as

$$Suff(s) = \{s' \in \Sigma^* : (\exists t \in \Sigma^*)s't = s\}$$

Especially, we define the set $Suff^+(s)$ as

$$Suff^+(s) = Suff(s) - \{\varepsilon\}$$

For a string $s = \sigma_1\sigma_2\sigma_3\cdots\sigma_m$, we use s^k to denote the prefix consisting of the first k events, that is, $s^k = \sigma_1\sigma_2\cdots\sigma_k$. We use s^{-k} to denote the suffix composed of the last k events, that is, $s^{-k} = \sigma_{m-k+1}\sigma_{m-k+2}\cdots\sigma_m$.

For a string $s = \sigma_1\sigma_2\sigma_3\cdots\sigma_m$ and a prefix $s' = \sigma_1\sigma_2\cdots\sigma_k$, we use s/s' to denote the suffix obtained by removing the prefix s', that is, $s/s' = \sigma_{k+1}\cdots\sigma_m$.

With a slight abuse of notations, for a string like s, we use $|s|$ to denote its length. For a set like Q, we use $|Q|$ to denote its cardinality.

3. PROBLEM STATEMENT

3.1 Sensor events

Assume that there are k discrete sensors. The set of sensors is denoted as

$$SN = \{sn_1, ..., sn_k\}$$

A discrete sensor may have multiple discrete outputs. Let us take the photoelectric switch as an example. It has two discrete outputs. One is for the case when the switch is turned on and the other is for the case when the switch is turned off. Without loss of generality, we use a label to represent a discrete output and call it a sensor event. Hence for a sensor sn_i, we can define the set of all its sensor events as

$$\Sigma_{sn_i} = \{\sigma_1, \sigma_2, ..., \sigma_j\}$$

The set of all sensor events is then denoted as

$$\Sigma_{SN} = \bigcup_{sn_i \in SN} \Sigma_{sn_i}$$

The occurrence of sensor events is constrained by the dynamics of the system. It means not all the strings in Σ_{SN}^* can occur, but a subset will be generated. We assume the subset can be generated by an automaton G_{SN} as

$$G_{SN} = (Q_{SN}, \Sigma_{SN}, \delta_{SN}, q_{0,SN})$$

where Q_{SN} is the finite state set; Σ_{SN} is the finite sensor event set; $\delta_{SN} : Q_{SN} \times \Sigma_{SN} \to Q_{SN}$ is the state transition function that defines the dynamics of the automaton. The transition function δ is extended to $\delta_{SN} : Q_{SN} \times \Sigma_{SN}^* \to Q_{SN}$ in the usual way. The initial state is $q_{0,SN}$.

The language generated by the automaton is defined as

$$L(G_{SN}) = \{s \in \Sigma_{SN}^* : \delta_{SN}(q_{0,SN}, s)!\}$$

where $\delta_{SN}(q_{0,SN}, s)!$ means that $\delta_{SN}(q_{0,SN}, s)$ is defined.

3.2 Complex events

In this paper, we consider a class of complex events of which each is represented by a sequence of sensor events.

We assume there are l complex events to be recognized and denote the set of complex events as

$$E = \{e_1, e_2, ..., e_l\}$$

For each complex event e_i, it is a sequence of sensor events and is denoted as

$$e_i = \sigma_1\sigma_2...\sigma_m$$

Note that E is a set of complex events and a language on Σ.

Here, we assume that complex events in E satisfy

$$(\forall e_i, e_j \in E)e_i \notin Pre(e_j)$$

which ensures that for any two complex events, they can be distinguished by observing the occurrence of their sensor events.

3.3 Complex event recognition

In practice, as the given system runs, we will observe the occurrence of sensor events one by one. For the observed sensor event sequence, we wish to determine which complex event sequence has occurred.

In order to simplify the complex event recognition problem, we make the following two assumptions.

Assumption 1. The occurrence of complex events can not be overlapped. That is, at any time, at most one complex event can occur.

Assumption 2. When a complex event e_i occurs, only the corresponding sensor events in the sequence $\sigma_1\sigma_2...\sigma_m (= e_i)$ can be observed in the same order.

Let us introduce some notations. For given two strings $s_1, s_2 \in \Sigma^*$, we say s_1 is a substring of s_2 and denote it as $s_1 \subseteq s_2$ if

$$s_2 = s's_1s''$$

where $s', s'' \in \Sigma^*$.

For a given sensor event sequence $s \in L(G)$, if it can be re-written as

$$s = s_1e_{k_1}s_2e_{k_2}...s_ne_{k_n}s_{n+1}$$

where

$$s_j \in \Sigma^* \wedge (\forall e_i \in E)e_i \not\subseteq s_j \qquad j = 1, 2, ..., n+1$$

We say $(s_1, e_{k_1}, s_2, e_{k_2}, ..., s_n, e_{k_n}, s_{n+1})$ is a partition pt of s. We use a set $\Pi(s)$ to denote all partitions as

$$\begin{aligned}\Pi(s) = &\{(s_1, e_{k_1}, s_2, e_{k_2}, ..., s_n, e_{k_n}, s_{n+1}) : \\ &s = s_1e_{k_1}s_2e_{k_2}...s_ne_{k_n}s_{n+1} \\ &\wedge (\forall j \in \{1, \cdots, n+1\})s_j \in \Sigma^* \\ &\wedge (\forall e_i \in E)e_i \not\subseteq s_j\}\end{aligned}$$

For a partition $pt = (s_1, e_{k_1}, s_2, e_{k_2}, ..., s_n, e_{k_n}, s_{n+1})$ of s, the corresponding complex event is

$$CE(pt) = e_{k_1}...e_{k_n}$$

With the above knowledge, we can formally state the complex event recognition problem as

Complex Event Recognition Problem of Discrete Event Systems *(CERP-DES)*: Given an automaton G_{SN} driven by sensor events and a set of complex events E, for any observed sensor event sequence $s \in L(G_{SN})$, we want to uniquely determine the occurred complex event sequence, that is, to uniquely determine the partition pt of s.

Note that for any string $s \in L(G_{SN})$, partitions always exist. if the problem is solvable, we have $|\Pi(s)| = 1$.

Let us use an example to illustrate these results.

Example 1. For a given system, the sensor event sequences which may be observed are described by automaton $G_{SN} = (Q_{SN}, \Sigma_{SN}, \delta_{SN}, q_{0,SN})$, which is shown in Fig. 1.

The set of complex events is $E = \{e_1, e_2\}$, where $e_1 = \sigma_1\sigma_2$ and $e_2 = \sigma_2\sigma_3$. E can be re-written as $E = \{\sigma_1\sigma_2, \sigma_2\sigma_3\}$.

Suppose that the observed sensor event sequence is $s = \sigma_1\sigma_2\sigma_3\sigma_4\sigma_1\sigma_3\sigma_2\sigma_2\sigma_3$, with complex event set E, s could be re-written as $s = e_1\sigma_3\sigma_4\sigma_1\sigma_3\sigma_2e_2$ and $s =$

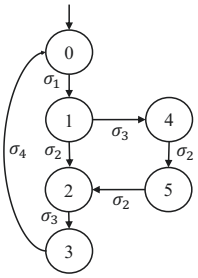

Fig. 1. Sensor event automaton G_{SN}

$\sigma_1 e_2 \sigma_4 \sigma_1 \sigma_3 \sigma_2 e_2$. Hence we can find two partitions for string s, that is

$$pt_1 = (\varepsilon, e_1, \sigma_3\sigma_4\sigma_1\sigma_3\sigma_2, e_2, \varepsilon)$$
$$pt_2 = (\sigma_1, e_2, \sigma_4\sigma_1\sigma_3\sigma_2, e_2, \varepsilon)$$

It means the complex event recognition problem is not solvable.

In fact, we can find two possible complex event sequences $CE(pt_1) = e_1e_2$ and $CE(pt_2) = e_2e_2$ from the partitions. We have no ways to determine which complex event sequence is the one that really occurred.

4. EXISTENCE OF SOLUTIONS

This section considers the existence of solutions. That is, we want to check the following statement

$$(\forall s \in L(G_{SN}))|\Pi(s)| = 1$$

holds or not.

Let us first introduce the following lemma.

Lemma 1. For given string $s \in L(G_{SN})$, if there exists a complex event e_i such that

$$(\exists s', s'' \in \Sigma^*)s = s'e_is'',$$

we then can find a partition for s

$$(s_1, e_{k_1}, s_2, e_{k_2}, \cdots, e_{k_{m-1}}, s_m, e_i, s_{m+1}, e_{k_m}, \cdots, s_{n+1})$$

such that

$$(s_1, e_{k_1}, s_2, e_{k_2}, \cdots, e_{k_{m-1}}, s_m) \in \Pi(s')$$
$$(s_{m+1}, e_{k_m}, \cdots, s_{n+1}) \in \Pi(s'')$$

Proof. Surely, we can find at least a partition for strings s' and s'', respectively. We write one partition for s' as

$$(s_1, e_{k_1}, s_2, e_{k_2}, \cdots, e_{k_{m-1}}, s_m)$$

and one partition for s'' as

$$(s_{m+1}, e_{k_m}, \cdots, s_{n+1})$$

Now we want to show

$$(s_1, e_{k_1}, s_2, e_{k_2}, \cdots, e_{k_{m-1}}, s_m, e_i, s_{m+1}, e_{k_m}, \cdots, s_{n+1})$$

is a partition of s.

Indeed, we have

$$(\forall s_i \in \{s_1, \cdots, s_{n+1}\})(\forall e_i \in E)e_i \not\subseteq s_j$$
$$\Rightarrow (s_1, e_{k_1}, \cdots, e_{k_{m-1}}, s_m, e_i, s_{m+1}, e_{k_m}, \cdots, s_{n+1}) \in \Pi(s)$$

Let us use an automaton G_E

$$G_E = (Y, \Sigma_{SN}, \delta_y, y_0, Y_m)$$

to represent complex event set E, such that

$$L_m(G_E) = E$$

Our idea is to combine G_E and G into an automaton that includes all the information on complex events and the dynamics of sensors.

We denote the combination as

$$G_c = (X, \Sigma_{SN}, \delta_c, x_0, X_m)$$
$$= CoAc(Q_{SN} \times Y, \Sigma_{SN}, \delta_c, (q_0, y_0), Q_{SN} \times Y_m)$$

which is constructed as follows.

For each state (q, y) in $Q_{SN} \times Y$ and sensor event σ in Σ_{SN}, we define its transition for the following two cases.

Case 1: $y = y_0 \vee y \in Y_m$**.** In this case, σ may be the first event of a complex event or is a disturbance. Here we define $\delta_c((q, y), \sigma)$ as (1) on the top of this page.

Case 2: $y \neq y_0 \wedge y \notin Y_m$**.** In this case, a complex event is occurring, σ should be a part of the complex event. Here we define $\delta_c((q, y), \sigma)$ as (2) on the top of this page.

After we define all the transitions, we remove those transitions which are not in the paths from the initial state x_0 to a marked state $x_m \in X_m$ by the operation $CoAc(\cdot)$. We then obtained G_c. We extend the transition function δ_c to $\delta_c : X \times \Sigma^* \to X$ in the usual way. Note that G_c is a non-deterministic automaton and has the same dynamics of G_{SN} as $L(G_c) = L(G_{SN})$.

Let us use an example to illustrate how to construct G_c.

Example 2. We continue to consider the sensor event automaton G_{SN} shown in Fig. 1. For the complex event set $E = \{\sigma_1\sigma_2, \sigma_2\sigma_3\}$, the corresponding automaton G_E is shown in Fig. 2 and the marked state set is $Y_m = \{y_2, y_4\}$.

We demonstrate the construction of the combination G_c as follows. The initial state of G_c is $(0, y_0)$. At initial state, event σ_1 can occur, which satisfies ***Case 1***. Hence we have $\delta_c((0, y_0), \sigma_1) = \{(1, y_0), (1, y_1)\}$. At state $(1, y_1)$, event σ_2 can occur, which satisfies ***Case 2***. Hence we have $\delta_c((1, y_1), \sigma_2) = (2, y_2)$.

By performing the same operation for each state, we can obtain the complete automaton. Note that after the operation $CoAc(\cdot)$, all the states that are not on the paths from the initial state to marked states will be deleted. The obtained automaton is shown as in Fig. 3

Because automaton G_c is non-deterministic, for each string $s = \sigma_1\sigma_2\cdots\sigma_l$ generated by it, s has multiple paths. Each path is denoted as

$$Path = x_0\sigma_1 x_{k_1}\sigma_2 x_{k_2}\cdots\sigma_l x_{k_l} \tag{3}$$

For any prefix $s' \in Pre(s)$, the state reachable in the path $Path$ is

$$Sta_{Path}(s') = x_{k_{|s'|}} \tag{4}$$

Note that $Sta_{Path}(\varepsilon) = x_0$. We further define the string generated by the path $Path$ as

$$Str(Path) = \sigma_1\sigma_2\cdots\sigma_l \tag{5}$$

Each state x in X is a doubleton. We use $SR_1(x)$ to denote its first element and $SR_2(x)$ to denote its second element. Automaton G_c has the following properties.

Proposition 1. For each path $Path$ and the string $s = Str(Path)$ generated by it, we can find a partition

$$Sta_{Path}(s) \in X_m \Rightarrow (\exists s' \in \Sigma^*)(\exists e_i \in E)s = s'e_i$$

$$\delta_c((q,y),\sigma) = \begin{cases} \{(\delta(q,\sigma),\delta_y(y,\sigma)),(\delta(q,\sigma),y_0)\} & \text{if } \delta(q,\sigma)! \wedge \delta_y(y,\sigma)! \\ \{(\delta(q,\sigma),y_0)\} & \text{if } \delta(q,\sigma)! \wedge \neg\delta_y(y,\sigma)! \\ \textit{undefined} & \text{if } \neg\delta(q,\sigma)! \wedge \neg\delta_y(y,\sigma)! \end{cases} \tag{1}$$

$$\delta_c((q,y),\sigma) = \begin{cases} \{(\delta(q,\sigma),\delta_y(y,\sigma))\} & \text{if } \delta(q,\sigma)! \wedge \delta_y(y,\sigma)! \\ \textit{undefined} & \text{if } \neg\delta(q,\sigma)! \vee \neg\delta_y(y,\sigma)! \end{cases} \tag{2}$$

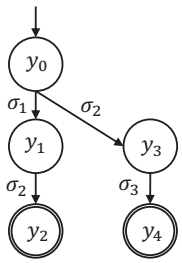

Fig. 2. Complex event automaton G_E

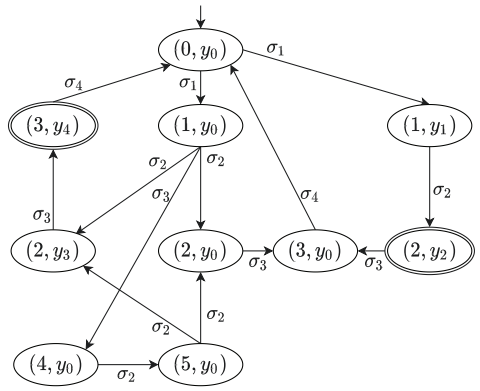

Fig. 3. Combination automaton G_c

Proof. By $Sta_{Path}(s) \in X_m$, s can be divided as $s = s''s'''$ such that

$Sta_{Path}(s'') = x_0 \wedge (\forall s'''' \in Pre^+(s'''))Sta_{Path}(s'''') \neq x_0$
$\Rightarrow (\forall s'''' \in Pre(s'''))\delta_y(y_0,s'''')! \wedge \delta_y(y_0,s''') \in Y_m$
(By the definition of δ_c)
$\Rightarrow s''' \in L_m(G_E)$
$\Rightarrow (\exists e_i \in E)s''' = e_i$
$\Rightarrow (\exists s' \in \Sigma^*)(\exists e_i \in E)s = s'e_i$
(Let $s' = s''$)

Proposition 2. For each string s and each partition pt of it as

$$pt = (s_1, e_{k_1}, s_2, e_{k_2}, ..., s_n, e_{k_n}, s_{n+1}),$$

we can find a path $Path(s = Str(Path))$ such that, for any s_i in pt,

$$(\forall s' \in Pre^+(s_i))Sta_{Path}(s_1e_{k_1}\cdots s') = (\delta_{SN}(q_0,s_1e_{k_1}\cdots s'),y_0)$$

and for any e_{k_i} in pt,

$$(\forall s' \in Pre^+(e_{k_i}))Sta_{Path}(s_1e_{k_1}\cdots s') = (\delta_{SN}(q_0,s_1e_{k_1}\cdots s'),\delta_y(y_0,s'))$$

Proof. By the definition of δ_c for Case 1, for $s_1 = \sigma_{k_1}\sigma_{k_2}...\sigma_{k_{|s_1|}}$, we can find a path for it as

$$(q_0,y_0) \xrightarrow{\sigma_{k_1}} (\delta_{SN}(q_0,\sigma_{k_1}),y_0) \xrightarrow{\sigma_{k_2}} (\delta_{SN}(q_0,\sigma_{k_1}\sigma_{k_2}),y_0)\cdots \xrightarrow{\sigma_{k_{|s_1|}}} (\delta_{SN}(q_0,s_1),y_0)$$

By the definition of δ_c for Case 2, for $e_{k_1} = \sigma_{k'_1}\sigma_{k'_2}...\sigma_{k'_{|e_{k_1}|}}$, we can find a path for it as

$$\delta_{SN}(q_0,s_1),y_0) \xrightarrow{\sigma_{k'_1}} (\delta_{SN}(q_0,s_1\sigma_{k'_1}),y_0) \xrightarrow{\sigma_{k'_2}} \cdots \xrightarrow{\sigma_{k'_{|e_{k_1}|}}} (\delta_{SN}(q_0,s_1e_{k_1}),y_0)$$

Consequently, we can find paths for $s_2, e_{k_2}, ...s_i, e_{k_i}....$ For $s_i = \sigma_{l_1}\sigma_{l_2}...\sigma_{l_{|s_i|}}$, the path is

$$\delta_{SN}(q_0,s_1e_{k_1}...e_{k_{i-1}}),y_0) \xrightarrow{\sigma_{l_1}} (\delta_{SN}(q_0,s_1e_{k_1}...\sigma_{l_1}),y_0) \xrightarrow{\sigma_{l_2}} \cdots \xrightarrow{\sigma_{l_{|s_i|}}} (\delta_{SN}(q_0,s_1e_{k_1}...s_i),y_0)$$

For $e_{k_i} = \sigma_{l_1}\sigma_{l'_2}...\sigma_{l'_{|e_{k_i}|}}$, the path is

$$\delta_{SN}(q_0,s_1e_{k_1}...s_i),y_0) \xrightarrow{\sigma_{l'_1}} (\delta_{SN}(q_0,s_1e_{k_1}...\sigma_{l'_1}),y_0) \xrightarrow{\sigma_{l'_2}} \cdots \xrightarrow{\sigma_{l'_{|e_{k_i}|}}} (\delta_{SN}(q_0,s_1e_{k_1}...s_ie_{k_i}),y_0)$$

The concatenation of these paths with the order of its corresponding string in pt is a path of s such that

$$(\forall s' \in Pre^+(s_i))Sta_{Path}(s_1e_{k_1}\cdots s') = (\delta_{SN}(q_0,s_1e_{k_1}\cdots s'),y_0)$$

and

$$(\forall s' \in Pre^+(e_{k_i}))Sta_{Path}(s_1e_{k_1}\cdots s') = (\delta_{SN}(q_0,s_1e_{k_1}\cdots s'),\delta_y(y_0,s'))$$

We then have the following theorem to show when *CERP-DES* is solvable.

Theorem 1. *CERP-DES* has solutions if and only if, there does not exist string $s = s's'' \in L(G_{SN})$ such that

$$(\exists y,y' \in Y - Y_m)(\exists y'',y''' \in Y_m)\{y,y''\} \in SR_2(\delta_c(x_0,s')) \wedge \{y',y'''\} \in SR_2(\delta_c(x_0,s's'')) \wedge (\exists e_i \in E)s'' \in Suff^+(e_i) \setminus \{e_i\} \wedge \delta_y(y,s'') = y''' \wedge \delta_y(y'',s'') = y' \tag{6}$$

Proof. Note that *CERP-DES* is not solvable if and only if there exists a string $s \in L(G_{SN})$ which has no less than two partitions. With it, let us prove Theorem 1.

(IF) When equation (6) holds, we have

$y'' \in Y_m \wedge y'' \in SR_2(\delta_c(x_0,s'))$
$\Rightarrow$there exists one path $Path$ such that $SR_2(Sta_{Path}(s')) = y''$
$\Rightarrow (\exists e_i \in E)s' = s'''e_i$
(By Proposition 1)
$\Rightarrow$we can find a partition for s' as $(s_1, e_k, ..., s_{n+1}, e_i)$
(By Lemma 1)

$\Rightarrow$we can find a partition pt_1 for s as

$$pt_1 = (s_1, e_{k_1}, ..., s_{n+1}, e_i, s'')$$

(Because $(\exists e_i \in E)s'' \in Suff^+(e_i) \setminus \{e_i\}$)

For equation (6), we further know that

$$y''' \in Y_m \wedge y''' \in SR_2(\delta_c(x_0, s))$$
$$\Rightarrow (\exists e_j \in E) s = s'''e_j$$

(By Proposition 1)

$\Rightarrow$we can find a partition pt_2 for s as

$$pt_2 = (s, e_{k_1}, ..., s_n, e_j)$$

(By Lemma 1)

Because $s'' \neq \varepsilon, pt_1 \neq pt_2$. It means that we can find at least two different partitions for s, hence *CERP-DES* is not solvable.

(ONLY IF) If *CERP-DES* is not solvable, we can find two different partitions pt_1 and pt_2 for string s as

$$pt_1 = (s_1, e_{k_1}, s_2, e_{k_2}, ..., e_{k_n}, s_{n+1}) \quad (7)$$
$$pt_2 = (s'_1, e_{m_1}, s'_2, e_{m_2}, ..., e_{m_{n'}}, s'_{n'+1}) \quad (8)$$

such that

$$(s_1, ..., e_{k_n}) \in Pre^+(s'_1, ..., e_{m_{n'}})$$
$$\wedge s'_1 ... e_{m_{n'}} / s_1 ... e_{k_n} \subset Suff^+(e_{m_{n'}})$$

Let

$$s' = s_1 ... e_{k_n},$$
$$s'' = s'_1 ... e_{m_{n'}} / s_1 ... e_{k_n},$$
$$s''' = e_{k_n} / (s^{|s|-|e_{m_{n'}}|} / s'^{|s'|-|e_{k_n}|}).$$

By proposition 2, we have

$$\delta_y(y_0, s''') \in SR_2(\delta_c(x_0, s')) \wedge$$
$$\delta_y(y_0, e_{k_n}) \in SR_2(\delta_c(x_0, s')) \wedge$$
$$\delta_y(y_0, e_{m_{n'}}) \in SR_2(\delta_c(x_0, s's'')) \wedge$$
$$\delta_y(y_0, s'') \in SR_2(\delta_c(x_0, s's''))$$
$$\Rightarrow (\exists u.\, u' \in Y - Y_m)(\exists u''.\, u''' \in Y_m)$$

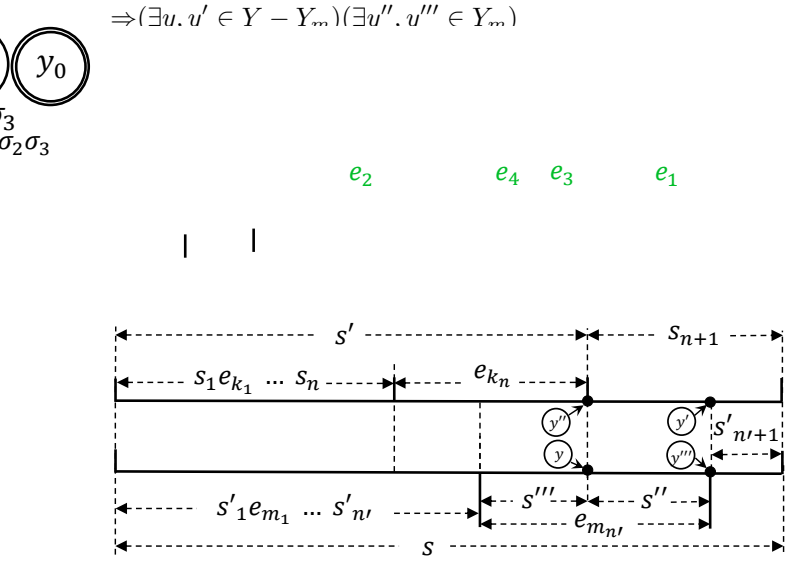

Fig. 4. Demonstration of proof of Theorem 1

Theorem 1 shows the state pairs in G_c should be considered. The following procedure is to construct an automaton G_A which includes all the state pairs we are interested in.

The automaton G_A is

$$G_A = (Z, \Sigma_{SN}, \delta_z, z_0, Z_m)$$
$$= Ac(X \times X, \Sigma_{SN}, \delta_z, (x_0, x_0), X_m \times X \cup X \times X_m)$$

For state (x, x'), and event $\sigma \in \Sigma$, we calculate their reachable states as

$$Reach(x, \sigma) = \delta_c(x, \sigma)$$
$$Reach(x', \sigma) = \delta_c(x', \sigma)$$

we then define the transition for state $z = (x, x')$ and event σ as (9) on the top of the next page.

With G_A, we have Algorithm 1 to check whether *CERP-DES* has solutions or not.

Algorithm 1: Checking Existence of Solutions

Input: Automaton

$$G_A = (Z, \Sigma, \delta_z, z_0, Z_m)$$

Output: YES or NO

1 Set $Ent = \emptyset$

2 For any two states $z_1, z_2 \in Z_m$ and any complex event $e_i \in E$, check whether G_A can reach z_2 from z_1 via a string $s \in Suff^+(e_i)/\{e_i\}$. If it is true, we set

$$Ent = Ent \cup \{(z_1, z_2, s)\}$$

3 **if** $Ent = \emptyset$ **then**

4 Go To Line 6

5 **OUTPUT** NO, END

6 **OUTPUT** YES, END

Algorithm 1 is correct as shown in the following theorem.

Theorem 2. If Algorithm 1 outputs YES, then *CERP-DES* is solvable. Else, Algorithm 1 outputs NO and *CERP-DES* has no solutions.

Proof. By Algorithm 1, we know

$$Ent \neq \emptyset$$
$$\Leftrightarrow (\exists z_1, z_2 \in Z_m)(\exists s \in \Sigma^*)(z_1, z_2, s) \in Ent$$
$$\Leftrightarrow (\exists z_1, z_2 \in Z_m)(\exists s \in \Sigma^*) \wedge s \in Suff^+(e_i)/\{e_i\}$$
$$\wedge \delta_z(z_1, s) = z_2$$
$$\Leftrightarrow (\exists ((q, y), (q'', y'')), ((q''', y'''), (q', y')) \in Z_m)(\exists s'' \in \Sigma^*)$$
$$\delta_z(((q, y), (q'', y'')), s'') = ((q''', y'''), (q', y'))$$
$$\wedge (\exists e_i \in E)s'' \in Suff^+(e_i)/\{e_i\}$$
$$\Leftrightarrow (\exists ((q, y), (q'', y'')), ((q''', y'''), (q', y')) \in Z_m)$$
$$(\exists s'' \in \Sigma^*)\delta_c((q, y), s) = (q''', y''')$$
$$\wedge \delta_c((q'', y''), s'') = (q', y')$$
$$\wedge (\exists e_i \in E)s'' \in Suff^+(e_i)/\{e_i\}$$
$$\Leftrightarrow [(\exists (q'', y''), (q''', y''') \in X_m)(\exists (q, y), (q', y') \in X - X_m)(\exists e_i \in E)s'' \in Suff^+(e_i)/\{e_i\} \wedge \delta_c((q, y), s'') = (q''', y''') \wedge \delta_c((q'', y''), s'') = (q', y')]$$
$$\vee [(\exists (q'', y''), (q''', y''') \in X_m)(\exists (q, y), (q', y') \in X - X_m)(\exists e_i \in E)s'' \in Suff^+(e_i)/\{e_i\} \wedge \delta_c((q, y), s'') = (q', y') \wedge \delta_c((q'', y''), s'') = (q''', y''')]$$
$$\text{(By } Z_m = X_m \times X \cup X \times X_m\text{)}$$
$$\Leftrightarrow (\exists (q'', y''), (q''', y''') \in X_m)(\exists (q, y), (q', y') \in X - X_m)(\exists e_i \in E)s'' \in Suff^+(e_i)/\{e_i\} \wedge \delta_c((q, y), s'') = (q''', y''') \wedge \delta_c((q'', y''), s'') = (q', y')$$

$$\delta_z((x,x'),\sigma) = \begin{cases} Reach(x,\sigma) \times Reach(x',\sigma) & \text{if } Reach(x,\sigma) \neq \emptyset \wedge Reach(x',\sigma) \neq \emptyset \\ \text{undefined} & \text{else} \end{cases} \tag{9}$$

(By $(\forall e_i, e_j \in E) e_i \not\subseteq e_j$)
$\Leftrightarrow (\exists y'', y''' \in Y_m)(\exists y, y' \in Y - Y_m)(\exists s', s'' \in \Sigma^*)$
$\{y, y''\} \in SR_2(\delta_c(x_0, s'))$
$\wedge \{y', y'''\} \in SR_2(\delta_c(x_0, s's''))$
$\wedge \delta_y(y'', s'') = y' \wedge \delta_y(y, s'') = y'''$
$\wedge (\exists e_i \in E) s'' \in Suff^+(e_i)/\{e_i\}$
$\Leftrightarrow$*CERP-DES* has no solutions
(By Theorem 1)

Example 3. We continue to consider the system and complex events shown in Fig. 1 and Fig. 2. The set of complex events is $E = \{e_1, e_2\}$, where $e_1 = \sigma_1\sigma_2$ and $e_2 = \sigma_2\sigma_3$. Automaton G_A is constructed as follows. The initial state is $z_0 = ((0, y_0), (0, y_0))$. From the initial state, for event σ_1, we have

$$\delta_z(z_0, \sigma_1) = \{((1, y_0), (1, y_0)), ((1, y_0), (1, y_1)), ((1, y_1), (1, y_0)), ((1, y_1), (1, y_1))\}$$

At state $z_1 = ((1, y_0), (1, y_1))$, for event σ_2, we have

$$\delta_z(z_1, \sigma_2) = \{((2, y_3), (2, y_2)), ((2, y_0), (2, y_2))\}$$

We demonstrate part of G_A as in Fig. 5.

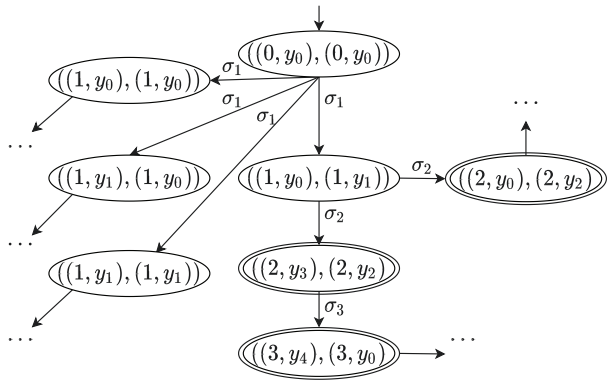

Fig. 5. Part of automaton G_A

We now check if there exists a solution by Algorithm 1. Initially, $Ent = \emptyset$. For marked states $z_1 = ((2, y_3), (2, y_2))$ and $z_2 = ((3, y_4), (3, y_0))$, we have $\delta_z(z_1, \sigma_3) = z_2$. Since $\sigma_3 \in Suff^+(e_2)/\{e_2\}$, the set Ent is updated as

$$Ent = \{(((2, y_3), (2, y_2)), ((3, y_4), (3, y_0)), \sigma_3)\}$$

In Line 3 of Algorithm 1, $Ent \neq \emptyset$ and the output of Algorithm 1 is NO. Thus, *CERP-DES* is not solvable.

5. ONLINE COMPLEX EVENT RECOGNITION

Theorem 2 tells us when *CERP-DES* is solvable. We know if *CERP-DES* is solvable, for any string generated by $L(G_{SN})$, there is only one partition and we can determine the complex event sequence uniquely. This section will develop one effective algorithm to calculate the unique complex event sequence for each observed sensor event sequence, which is performed online. Note that our algorithm works when *CERP-DES* is solvable.

For a complex event, before the sensor event sequence is observed, we should store its prefix which has occurred. Hence we have the following procedure to calculate the complex event sequence.

Initially, the observed sequence is ε and the prefix set is an empty set as $PreSet = \emptyset$. The output is nothing which means no complex event has occurred.

Assume a sensor event sequence s has occurred and the prefix set is $PreSet = \{s_1, s_2, \cdots, s_n\}$.

When a new sensor event σ_i occurs, we revise the prefix set as

$$TempSet = \{s_1\sigma_i, s_2\sigma_i, \cdots, s_n\sigma_i, \sigma_i\}$$

Note that σ_i may be the first sensor event of a complex event.

Check if there is a string in $TempSet$ which is a complex event. If it is true, we know a complex event has occurred. The output is the occurred complex event. The prefix set should be updated as $\emptyset$. Else, we check if these strings are prefixes of complex events. If a string in $TempSet$ is a prefix, we then put it in $PreSet$ as

$$PreSet = \{s_i \in TempSet : (\exists e_j \in E) s_i \in Pre^+(e_j)\}$$

and output nothing.

Example 4. We continue to consider the system shown in Fig.1, the complex event set is $E = \{e_1, e_2\}$ as $e_1 = \sigma_1\sigma_2\sigma_3$ and $e_2 = \sigma_2\sigma_2\sigma_3$, automaton G_E is shown in Fig. 6. We can verify *CERP-DES* is solvable with the approach proposed in the previous section.

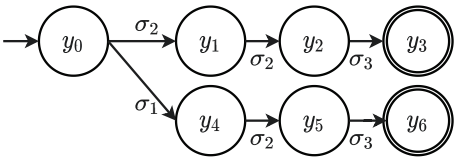

Fig. 6. Complex event automaton G_E

Now let us consider how to determine the occurred complex event sequence online. Assuming that the observed sensor event sequence is $\sigma_1\sigma_2\sigma_3\sigma_4\sigma_1\sigma_3\sigma_2\sigma_2\sigma_3$.

Initially, no sensor event occurs and the prefix set is $PreSet = \{\varepsilon\}$.

After sensor event σ_1 is observed, we have $\sigma_1 \in Pre^+(e_1)$. We then update the prefix set as $PreSet = \{\sigma_1\}$ and output nothing.

The next observed sensor event is σ_2, we have $TempSet = \{\sigma_1\sigma_2, \sigma_2\}$. Because $\sigma_1\sigma_2 \in Pre^+(e_1)$ and $\sigma_2 \in Pre^+(e_2)$, $PreSet$ is updated as $PreSet = \{\sigma_1\sigma_2, \sigma_2\}$ and we still output nothing.

When σ_3 is observed, $TempSet = \{\sigma_1\sigma_2\sigma_3, \sigma_2\sigma_3\}$. Since $\sigma_1\sigma_2\sigma_3 = e_1$, complex event e_1 is recognized and the output is e_1. $PreSet$ is then updated as empty set.

The complete recognition process is shown in Table 1. We can see that for the observed sensor event sequence

Table 1. Recognize complex events online

Observed sensor events	Occurred sensor event sequence	Preset	Output	Occurred complex event sequence
ε	ε	$\{\varepsilon\}$	None	None
σ_1	σ_1	$\{\sigma_1\}$	None	None
σ_2	$\sigma_1\sigma_2$	$\{\sigma_1\sigma_2, \sigma_2\}$	None	None
σ_3	$\sigma_1\sigma_2\sigma_3$	$\emptyset$	e_1	e_1
σ_4	$\sigma_1\sigma_2\sigma_3\sigma_4$	$\emptyset$	None	e_1
σ_1	$\sigma_1\sigma_2\sigma_3\sigma_4\sigma_1$	$\{\sigma_1\}$	None	e_1
σ_3	$\sigma_1\sigma_2\sigma_3\sigma_4\sigma_1\sigma_3$	$\emptyset$	None	e_1
σ_2	$\sigma_1\sigma_2\sigma_3\sigma_4\sigma_1\sigma_3\sigma_2$	$\{\sigma_2\}$	None	e_1
σ_2	$\sigma_1\sigma_2\sigma_3\sigma_4\sigma_1\sigma_3\sigma_2\sigma_2$	$\{\sigma_2\sigma_2, \sigma_2\}$	None	e_1
σ_3	$\sigma_1\sigma_2\sigma_3\sigma_4\sigma_1\sigma_3\sigma_2\sigma_2\sigma_3$	$\emptyset$	e_2	e_1e_2

$\sigma_1\sigma_2\sigma_3\sigma_4\sigma_1\sigma_3\sigma_2\sigma_2\sigma_3$, the occurred complex event sequence is e_1e_2.

6. CONCLUSIONS

This paper investigates complex event recognition from discrete sensor data. The main contributions are summarized as follows: 1) We formally state the complex event recognition problem in a discrete event system framework. 2) We propose an algorithm to verify if there exists a solution for the complex event recognition problem. 3) An effective online recognition approach is designed to handle the sensor event stream when the complex event recognition problem is solvable. With these results, the complex event recognition problem is successfully solved.

REFERENCES

Brenna, L., Gehrke, J., Hong, M., and Johansen, D. (2009). Distributed event stream processing with nondeterministic finite automata. In *Proceedings of the Third ACM International Conference on Distributed Event-Based Systems*, DEBS '09. Association for Computing Machinery.

Chakravarthy, S. and Mishra, D. (1994). Snoop: An expressive event specification language for active databases. *Data Knowl. Eng.*, 14, 1–26.

Chandola, V., Mithal, V., and Kumar, V. (2008). Comparative evaluation of anomaly detection techniques for sequence data. In *2008 Eighth IEEE International Conference on Data Mining*, 743–748.

Dai, X. and Gao, Z. (2013). From model, signal to knowledge: A data-driven perspective of fault detection and diagnosis. *IEEE Transactions on Industrial Informatics*, 9(4), 2226–2238.

Genc, S. and Lafortune, S. (2006). Diagnosis of patterns in partially-observed discrete-event systems. In *Proceedings of the 45th IEEE Conference on Decision and Control*, 422–427. doi:10.1109/CDC.2006.377450.

Gruber, R., Krishnamurthy, B., and Panagos, E. (1999). The architecture of the ready event notification service. In *Proceedings. 19th IEEE International Conference on Distributed Computing Systems. Workshops on Electronic Commerce and Web-based Applications. Middleware*, 108–113.

Jeron, T., Marchand, H., Pinchinat, S., and Cordier, M.O. (2006). Supervision patterns in discrete event systems diagnosis. In *2006 8th International Workshop on Discrete Event Systems*, 262–268. doi: 10.1109/WODES.2006.1678440.

Jin, X., Lee, X., Kong, N., and Yan, B. (2008). Efficient complex event processing over rfid data stream. In *Seventh IEEE/ACIS International Conference on Computer and Information Science (icis 2008)*, 75–81. doi: 10.1109/ICIS.2008.60.

Li, D., Chen, D., Jin, B., Shi, L., Goh, J., and Ng, S.K. (2019). Mad-gan: Multivariate anomaly detection for time series data with generative adversarial networks. In I.V. Tetko, V. Kůrková, P. Karpov, and F. Theis (eds.), *Artificial Neural Networks and Machine Learning – ICANN 2019: Text and Time Series*, 703–716. Springer International Publishing, Cham.

Ozer, S., Silver, D., Bemis, K., Martin, P., and Takle, J. (2011). Activity detection for scientific visualization. In *2011 IEEE Symposium on Large Data Analysis and Visualization*, 117–118.

Sadoghi, M. and Jacobsen, H.A. (2011). Be-tree: An index structure to efficiently match boolean expressions over high-dimensional discrete space. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, 637–648.

Saives, J., Pianon, C., and Faraut, G. (2015). Activity discovery and detection of behavioral deviations of an inhabitant from binary sensors. *IEEE Transactions on Automation Science and Engineering*, 12(4), 1211–1224. doi:10.1109/TASE.2015.2471842.

Shi, L.L., Liu, L., Wu, Y., Jiang, L., Kazim, M., Ali, H., and Panneerselvam, J. (2019). Human-centric cyber social computing model for hot-event detection and propagation. *IEEE Transactions on Computational Social Systems*, 6(5), 1042–1050.

Viard, K., Fanti, M.P., Faraut, G., and Lesage, J.J. (2020). Human activity discovery and recognition using probabilistic finite-state automata. *IEEE Transactions on Automation Science and Engineering*, 17(4), 2085–2096. doi:10.1109/TASE.2020.2989226.

Wiot, D. (2004). A new adaptive transient monitoring scheme for detection of power system events. *IEEE Transactions on Power Delivery*, 19(1), 42–48.

Wu, J., Xia, M., and Shu, S. (2016). Location tracking of a single inhabitant in smart home: A discrete event system approach. In *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Comput-*

ing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 615–620.

Ye, L. and Dague, P. (2012). A general algorithm for pattern diagnosability of distributed discrete event systems. In *2012 IEEE 24th International Conference on Tools with Artificial Intelligence*, volume 1, 130–137. doi: 10.1109/ICTAI.2012.26.

Zong, B., Song, Q., Min, M.R., Cheng, W., Lumezanu, C., Cho, D., and Chen, H. (2018). Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International conference on learning representations*.