Systems & Control Letters 119 (2018) 101–107

Contents lists available at ScienceDirect

Systems & Control Letters

journal homepage: www.elsevier.com/locate/sysconle

Trajectory detectability of discrete-event systems

Xiang Yin [a,b,*], Zhaojian Li [c], Weilin Wang [d]

[a] Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China
[b] Key Laboratory of System Control and Information Processing, Ministry of Education of China, Shanghai 200240, China
[c] Department of Mechanical Engineering, Michigan State University, East Lansing, MI 48824, USA
[d] Faculty of Engineering, Monash University, Clayton, VIC 3800, Australia

ARTICLE INFO

Article history:
Received 19 June 2017
Received in revised form 14 July 2018
Accepted 16 July 2018

Keywords:
Discrete-event systems
Trajectory estimation
Detectability

ABSTRACT

We investigate the trajectory estimation problem for partially-observed discrete event systems. In some applications, only knowing the current state of the system may be insufficient, and knowing which trajectory the system takes to reach the current state could be important. This requires more precise knowledge about the system. In this paper, a language-based framework is proposed in order to tackle this problem. Two new notions of detectability, called trajectory detectability and periodic trajectory detectability, are proposed to capture different requirements in the aforementioned trajectory estimation problem. Effective verification algorithms are also provided. Our results extend the theory on detectability of discrete event systems from state estimation problem to trajectory estimation problem.

1. Introduction

System estimation is one of the central problems in systems and control theory. In many applications, we do not have full access to the system's internal state and need to perform state estimation. The state estimation problem becomes particularly important when one wants to make decisions based on the limited system information. In this paper, we investigate the state estimation problem for partially-observed Discrete-Event Systems (DES).

The problem of state estimation has drawn considerable attentions in the DES literature due to its importance; see, e.g., [1–6]. This problem was initiated in [1,2], where the notion of observability was defined.[1] Recently, the state estimation problem has been studied more systematically in the framework of detectability; see, e.g., [4,5,8–14]. Particularly, in [4], the authors define four types of detectability in order to capture different requirements in practice. These notions of detectability have been further generalized by [5,8]. For example, [8] defines a generalized detectability based on the state disambiguation problem [3,6]. Detectability has also been studied in the framework of stochastic DES by [9,11]. When the original system is not detectable, several approaches have been proposed in order to enforce detectability, e.g., by sensor activations [15,16] and by supervisory control [17,18]. State estimation problem has also been investigated in the context of colored graph [19].

All of the aforementioned works on detectability are *state-based*. Namely, one wants to estimate the current state of the system based on a given model. However, in some applications, knowing the current state of the system is not sufficient. For example, in the application of location-based services (LBS) [20], a DES is usually used to represent the connective of a region and each state in it corresponds to a location. Sometimes, however, simply knowing the current location may not be sufficient, for instance, if we want to know by which path this location is reached. Therefore, instead of estimating the state of the system, one may also be interested in estimating the *trajectory* of the system.

In this paper, we systematically study the trajectory estimation problem in the context of partially-observed DES. Specifically, this paper has the following contributions. First, we define the notions of trajectory detectability and periodic trajectory detectability. These two notions provide the conditions for determining *a priori* if the trajectory of a given system can be determined after a bounded delay or be determined periodically. Second, for regular languages, i.e., languages that can be marked by finite-state automata, we provide effective algorithms to verify these two conditions. In particular, the verification algorithm for trajectory detectability requires polynomial-times using a twin-machine-like construction. On the other hand, the algorithm for verifying periodic trajectory detectability requires exponential complexity.

Note that, although the study of trajectory detectability is motivated by state detectability [4], their verifications are quite different. In general, we can always refine an automaton by expanding

* Corresponding author at: Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China.
E-mail address: yinxiang@sjtu.edu.cn (X. Yin).

[1] The notion of observability was also used in [7] as a condition under which a partial-observation supervisor can correctly make control decisions.

https://doi.org/10.1016/j.sysconle.2018.07.008
0167-6911/© 2018 Elsevier B.V. All rights reserved.

its state space such that each state carries more information. However, the domain of language is infinite and one may not always be able to use finite states to precisely capture the (infinite) trajectory information. Our framework is fully language-based and it does not depend on the automaton realizing the language. Moreover, we show that infinite language information can still be effectively verified by using its underlying automaton. In particular, we only exploit the standard twin-machine construction [21,22], which is different from the detector construction proposed in [8]. We show that the phenomenon of *information merge* plays an important role in the trajectory estimation problem. This issue also does not exist in the state-based framework for detectability.

We also would like to remark that our paper is not the first one investigating the trajectory estimation problem in DES. In [23], the authors investigate a similar problem and the notion of *invertibility* is proposed. Our work is different from [23] due to the following reasons. First, we systematically investigate both trajectory detectability and periodic trajectory detectability; both of these two notions are different from invertibility. Specially, invertibility only requires to recover the last n events but detectability requires to recover the precise trajectory executed by the system. Second, we provide a language-based framework for studying this problem, while the result in [23] is state dependent. Finally, invertibility is defined only for prefix-closed languages, while trajectory detectability is defined for non-prefix-closed languages.

2. Preliminaries

Let Σ be a finite set of events. A string $s = \sigma_1 \dots \sigma_n$ is finite sequence of events and we denote by $|s|$ the length of s. We use ϵ to denote the empty string with $|\epsilon| = 0$. We denote by Σ^* the set of all strings including ϵ. A language $L \subseteq \Sigma^*$ is a set of strings. The prefix-closure of language L is defined as $\overline{L} := \{w \in \Sigma^* : \exists v \in \Sigma^* \text{ s.t. } wv \in L\}$. We say that L is prefix-closed if $\overline{L} = L$. We denote by L/s the post-language of L after string s, i.e., $L/s := \{t \in \Sigma^* : st \in L\}$. We denote by $Card[L]$ the cardinality of L, which is the number of strings in L. For two strings $s, t \in \Sigma^*$, we write $s \leq t$ if $s \in \overline{\{t\}}$.

A deterministic finite-state automaton (DFA) is a 5-tuple $G = (Q, \Sigma, \delta, q_0, Q_m)$, where Q is the finite set of states, Σ is the finite set of events, q_0 is the initial state, Q_m is the set of marked states and $\delta : Q \times \Sigma \to Q$ is the partial transition function, where $\delta(q, \sigma) = q'$ means that there exists a transition labeled with event σ from state q to state q'. The transition function is also extended to $Q \times \Sigma^*$ in the usual manner; see, e.g., [24]. We denote by $\mathcal{L}(G)$ the language *generated* by G, i.e., $\mathcal{L}(G) = \{s \in \Sigma^* : \delta(q_0, s)!\}$, where ! means "is defined". We denoted by $\mathcal{L}_m(G)$ the language *marked* by G, i.e., $\mathcal{L}(G) = \{s \in \Sigma^* : \delta(q_0, s) \in Q_m\}$. We say that a language $L \subseteq \Sigma^*$ is *regular* if there exists a DFA G such that $\mathcal{L}_m(G) = L$.

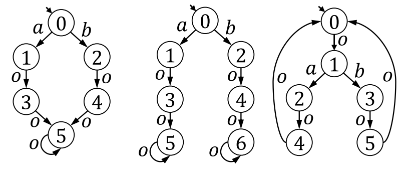
In many cases, the event generated by the system cannot be observed perfectly. Therefore, we assume that the event set Σ is partitioned into two disjoint sets $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$, where Σ_o is the set of observable events and Σ_{uo} denotes the set of unobservable events. Then $P : \Sigma^* \to \Sigma_o^*$ denotes the natural projection that erases event in Σ_{uo} from a string; this can be defined by

$$P(\epsilon) = \epsilon \text{ and } P(s\sigma) = \begin{cases} P(s)\sigma & \text{if } \sigma \in \Sigma_o \\ P(s) & \text{if } \sigma \in \Sigma_{uo} \end{cases} \tag{1}$$

The natural projection is also extended to 2^{Σ^*} by $P(L) = \{s \in \Sigma_o^* : \exists t \in L \text{ s.t. } P(t) = s\}$. We denote by P^{-1} the inverse projection.

Given a DFA G and a set of states $I \subseteq Q$, we denote by $\text{Acc}_G(I)$ the set of states accessible from some state in I, i.e.,

$$\text{Acc}_G(I) = \{q \in Q : \exists q' \in I, \exists s \in \Sigma^* \text{ s.t. } \delta(q', s) = q\}. \tag{2}$$



(a) System G_1 (b) System G_2 (c) System G_3

Fig. 1. For both G_1 and G_2: $\Sigma_o = \{o\}$ and $\Sigma_{uo} = \{a, b\}$.

Let $q \in Q$ be a state in G. We denote by $\text{In}_G(q)$ the set of events entering q, i.e.,

$$\text{In}_G(q) = \{\sigma \in \Sigma : \exists q' \in Q \text{ s.t. } \delta(q', \sigma) = q\}. \tag{3}$$

Finally, let $s \in P(\mathcal{L}(G))$, we denote by $R_G(s)$ the set of states that can be reached by observing s, i.e.,

$$R_G(s) = \{q \in Q : \exists t \in \Sigma^* \text{ s.t. } \delta(q_0, t) = q \wedge P(t) = s\}. \tag{4}$$

3. State-Based detectability and trajectory-based detectability

Due to measurement uncertainty, one may not always have a perfect knowledge about the current status of the system. In [4], the notion of (strongly) detectability was introduced in order to capture whether or not we can eventually have a perfect knowledge about the system after finite delay. In this paper, we refer to detectability defined in [4] as *state detectability*. First, we recall its definition.

Definition 1. A DFA $G = (Q, \Sigma, \delta, q_0, Q_m)$ is said to be *state detectable* w.r.t. Σ_o if

$$(\exists n \in \mathbb{N})(\forall s \in L)[|P(s)| \geq n \Rightarrow |R_G(P(s))| = 1] \tag{5}$$

Example 1. Let us consider system G_1 shown in Fig. 1(a), where $\Sigma_o = \{o\}$. Clearly, this system is state detectable since $R_G(o^n) = \{5\}$ for any $n \geq 2$. However, system G_2 shown in Fig. 1(b) is not state detectable. To see this, for any $n \geq 2$, we can find o^n such that $R_G(o^n) = \{5, 6\}$, i.e., we can never determine the current state of the system precisely.

Intuitively, state detectability says that, after a finite delay, we will know exactly the current state of the system and maintain this ability in the future. However, in some applications, this requirement may be too strong. Therefore, in [4], the notion of periodic state detectability was also proposed, which only requires that we can detect the state of the system periodically.

Definition 2. A DFA $G = (Q, \Sigma, \delta, q_0, Q_m)$ is said to be *periodically state detectable* w.r.t. Σ_o if

$$(\exists n \in \mathbb{N})(\forall s \in L)(\forall t \in L/s : |P(t)| \geq n)$$
$$(\exists t' \leq t)[|R_G(P(st'))| = 1] \tag{6}$$

Example 2. Let us consider system G_3 shown in Fig. 1(c) with $\Sigma_o = \{o\}$. This system is not state detectable, since we cannot distinguish states 4 and 5 after observing $oo(ooo)^n$ for any $n \geq 0$. However, it is periodically state detectable, since we always know for sure that the current state is 0 after observing $(ooo)^n$ for any $n \geq 0$.

Remark 1. Note that the system automaton G considered in [4] has multiple initial states; say $Q_0 \subseteq Q$. In the above definitions, we only consider the case where the initial state is unique. However, this setting is just for the sake of simplicity and is without loss of generality, since we can add a new single initial state and add unobservable transitions from this new initial state to each state in Q_0, which essentially simulates the case of multiple initial states.

In Example 1, we see that, although G_1 is state detectable but G_2 is not, they generate the same language, i.e., $\mathcal{L}(G_1) = \mathcal{L}(G_2)$. In other words, a system, which is originally state detectable, may become non-detectable after refining its state-space and vice versa. In fact, a state in an automaton only tells the system's status about potential future behaviors. However, some information about the history is lost. In some cases, this history information is also very important. For example, in Fig. 1(a), even though we know for sure that the system is at state 5 after observing oo, we still cannot determine *how* state 5 is reached, since both aoo and boo are potential trajectories. Motivated by the above discussion, we define the notion of trajectory detectability.

Definition 3. A language L is said to be *trajectory detectable* w.r.t. Σ_o if

$$(\exists n \in \mathbb{N})(\forall s \in L)[|P(s)| \geq n \Rightarrow Card[P^{-1}P(s) \cap L] = 1] \tag{7}$$

Definition 4. A language L is said to be *periodically trajectory detectable* w.r.t. Σ_o if

$$(\exists n \in \mathbb{N})(\forall s \in L)(\forall t \in L/s : |P(t)| \geq n)$$
$$(\exists t' \leq t)[Card[P^{-1}P(st') \cap L] = 1] \tag{8}$$

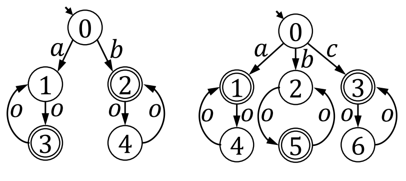
Intuitively, trajectory detectability requires that, after a finite delay, we are able to precisely recover the trajectory that leads to the current state and maintain this ability in the future. Similarly, periodic trajectory detectability requires that, we are able to precisely recover the trajectory of the system periodically.

Note that language L in the definition does not need to be prefix-closed. In fact, we consider non-prefix-closed language since it is more general than the prefix-closed case and using non-prefix-closed languages provides us more flexibility to describe the system's behavior of interest. This point is illustrated by the following example.

Example 3. Let us consider the DFA shown in Fig. 2(a) with $\Sigma_o = \{o\}$. Note that G_4 is neither state detectable nor $\mathcal{L}(G_4)$ is trajectory detectable. However, non-prefix-closed language $\mathcal{L}_m(G_4)$ is trajectory detectable. Clearly, by observing $o^{2k+1}, k \leq 0$, we know that the only possible string in $\mathcal{L}_m(G)$ is ao^{2k+1}. Similarly, by observing $o^{2k}, k \leq 1$, we know that the only possible string in $\mathcal{L}_m(G_4)$ is bo^{2k+1}. This non-prefix-closed language can be used to model the case, where we are only interested in strings in $\mathcal{L}_m(G)$ and do not care whether or not a wrong estimation is made for strings in $\mathcal{L}(G) \setminus \mathcal{L}_m(G)$. For instance, here we are only interested in distinguishing strings leading to state 2 and strings leading to state 3. Similarly, for G_5 shown in Fig. 2(b), we know that $\mathcal{L}_m(G_5)$ is periodically trajectory detectable but $\mathcal{L}(G_5)$ is not. These examples justify our early assertion that using non-prefix-closed languages provides us more flexibility to describe the system's behavior of interest.

The following two results establish some relationships between state detectability and trajectory detectability. Their proofs are omitted, since the results follow directly from the definitions.

Proposition 1. *Let G be a DFA and Σ_o be a set of observable events. If G is not (periodically) state detectable, then $\mathcal{L}(G)$ is not (periodically) trajectory detectable.*



(a) System G_4 (b) System G_5

Fig. 2. $\mathcal{L}_m(G)$ is trajectory detectable but $\mathcal{L}(G)$ is not trajectory detectable, where $\Sigma_o = \{o\}$ and $\Sigma_{uo} = \{a, b, c\}$.

Remark 2. Note that the opposite direction of the above statement is not true in general; an example has been provided in Fig. 1(a). Moreover, when the language under consideration is non-prefix-closed, (periodic) trajectory detectability is no longer stronger than (periodic) state detectability in general; this issue has already been discussed in Example 3.

Given a DFA G, we say that G is a tree if $\forall s, t \in \mathcal{L}(G) : s \neq t \Rightarrow \delta(q_0, s) \neq \delta(q_0, t)$. The following result reveals that (periodic) state detectability and prefix-closed (periodic) trajectory detectability coincide when G is a tree.

Proposition 2. *Let G be a tree and Σ_o be a set of observable events. Then G is (periodically) state detectable if and only if $\mathcal{L}(G)$ is (periodically) trajectory detectable.*

Remark 3. Intuitively, if G is a tree, then states and trajectories carry the same information. Clearly, if a language is finite, or equivalently, G is acyclic, then we can always refine G as a tree and make sure that each state in G uniquely carries the corresponding trajectory information. However, if we consider the general case, the following difficulties arise. First, we need to deal with non-prefix-closed languages. As we discuss earlier, trajectory detectability for the non-prefix-closed case is incomparable with either state detectability or its prefix-closed case. More importantly, in general, the language under consideration is infinite, i.e., G is not acyclic. Consequently, refining G as a tree or another structure requires infinite memory and it is not clear whether or not this trajectory information can be effectively verified. Therefore, we need new methods to handle these difficulties.

Remark 4. The definition of trajectory detectability is related to several language-based properties in the literature, e.g., opacity [25], diagnosability [26] and normality [7]. In [25], a security property called opacity was defined. Particularly, given two languages $K, L \subseteq \mathcal{L}(G)$, we say that (L, K) is *weakly opaque* if $\exists s \in L, \exists t \in K : P(s) = P(t)$. This definition is similar to the negation of trajectory detectability. However, they have the following main differences. First, weak opacity is a indistinguishable property that should hold for *one specific string s*. However, non-trajectory-detectability is a indistinguishable property that should hold *eventually* for *all continuation strings* starting from a sufficiently long point. Moreover, for weak opacity, language K to be distinguished from is static in the sense that it does not depend on the string considered in L. However, for trajectory detectability, for each string $s \in \mathcal{L}_m(G)$, we need to distinguish it from strings in $\mathcal{L}_m(G) \setminus \{s\}$. Therefore, the language to be distinguished from is "dynamic" in the sense that it depends on the string considered in $\mathcal{L}_m(G)$. Trajectory detectability is also related to diagnosability [26] since both of them require some information ambiguity after a finite delay. Particular, diagnosability requires to distinguish fault strings from non-fault strings after some delays, where the language to be distinguished from is still "static". This situation is the same for normality [7], where we still need to distinguish two "static" languages.

4. Verification of trajectory detectability for regular languages

In this section, we investigate the verification of trajectory detectability for the case where the language L under consideration is *regular*. Therefore, we denote by $G = (Q, \Sigma, \delta, q_0, Q_m)$ the DFA accepting L, i.e., $\mathcal{L}_m(G) = L$. We also make the following assumptions:

A-1 $\mathcal{L}(G)$ is live; and
A-2 G does not contain an unobservable cycle; and
A-3 $\mathcal{L}(G) = \overline{\mathcal{L}_m(G)}$, i.e., G is non-blocking.

Assumptions A-1 and A-2 are standard assumptions in the analysis of partially-observed DES in order to avoid trivial result. We make assumption A-3 just to simplify the technical development and it is without of loss generality, since we can take the trim part of G, i.e., removing all states that cannot reach a mark state from G, if $\mathcal{L}(G) \neq \overline{\mathcal{L}_m(G)}$.

4.1. Twin-Machine and information merge

In order to verify trajectory detectability, we construct a new verification DFA $V = (X_V, \Sigma_V, f_V, x_{0,V}, X_{m,V})$, where

- $X_V \subseteq Q \times Q$ is the set of states;
- $\Sigma_v = (\Sigma_o \times \Sigma_o) \cup (\Sigma_{uo} \times \{\epsilon\}) \cup (\{\epsilon\} \times \Sigma_{uo})$ is the set of events;
- $x_{0,V} = (q_0, q_0)$ is the initial state;
- $X_{m,V} = Q_m \times Q_m$ is the set of marked states;
- $f_V : X_V \times \Sigma_v \to X_V$ is the partial transition function defined by: for any state $(x_1, x_2) \in X_V$ and an event $\sigma \in \Sigma$, we have
  (a) If $\sigma \in \Sigma_o$, then the following transition is defined:
  $$f_V((x_1, x_2), (\sigma, \sigma)) = (\delta(x_1, \sigma), \delta(x_2, \sigma)) \tag{9}$$
  (b) If $\sigma \in \Sigma_{uo}$, then the following transitions are defined:
  $$f_V((x_1, x_2), (\sigma, \epsilon)) = (\delta(x_1, \sigma), x_2) \tag{10}$$
  $$f_V((x_1, x_2), (\epsilon, \sigma)) = (x_1, \delta(x_2, \sigma)) \tag{11}$$

Hereafter, we only consider the reachable part of V.

Remark 5. The construction of DFA V essentially follows the idea of the well-known twin-machine (or verifier) construction used in the literature for the verification of diagnosability; see, e.g., [21,22]. Roughly speaking, V tracks all pairs of observational equivalent strings in G. Specifically, if $s_1, s_2 \in \mathcal{L}(G)$ are two strings in G such that $P(s_1) = P(s_2)$, then there exists a string $s \in \mathcal{L}(V)$ in V such that its first and second components are s_1 are s_2, respectively, i.e., $f_V(x_{0,V}, s) = (\delta(q_0, s_1), \delta(q_0, s_2))$. On the other hand, for any string $s = (s_1, s_2) \in \mathcal{L}(V)$ in V, we have that $P(s_1) = P(s_2)$.

Before we show how to use DFA V to verify trajectory detectability, let us consider the following scenario. Suppose that there exist two *different* strings $s_1, s_2 \in \mathcal{L}(G)$ such that $P(s_1) = P(s_2)$ and $\delta(q_0, s_1) = \delta(q_0, s_2)$. Since these two strings lead to the same state, we know that they have the same continuations. Moreover, since s_1 and s_2 are observational equivalent, we know that we cannot distinguish any continuations of these two strings from this state. We call such a phenomenon *information merge*. Clearly, if the information merge phenomenon occurs, then $\mathcal{L}_m(G)$ becomes non-detectable. Therefore, we need to check whether or not the information merge phenomenon occurs as the first step towards the verification of trajectory detectability. To the end, we first introduce the notion of merging state.

Definition 5. A state $x = (x_1, x_2) \in X_V$ is said to be a *merging state* if
$$[x_1 \neq x_2] \wedge [\exists \sigma \in \Sigma_o : \delta(x_1, \sigma) = \delta(x_2, \sigma)] \tag{12}$$
or
$$\begin{aligned}&\left[In^V(x) \cap (\Sigma_o \times \Sigma_o) \neq \emptyset \vee x = x_{0,V}\right] \\ &\wedge \left[\exists e_1, e_2 \in \Sigma_{uo}^* : [\delta(x_1, e_1) = \delta(x_2, e_1)] \wedge [x_1 = x_2 \Rightarrow e_1 \neq e_2]\right]\end{aligned} \tag{13}$$
We denote by $X_{merge} \subseteq X_V$ the set of merging states.

Intuitively, a merging state (x_1, x_2) is either the initial state $x_{0,V}$ or a state reached immediately after a pair of observable events such that x_1 and x_2 can reach a same state unobservably or via a single common observable event. Moreover, the strings from x_1 and x_2 to the same state should be different if $x_1 \neq x_2$. The following result reveals that the information merge phenomenon will not happen in G if and only if V does not contain a merging state.

Theorem 1. *There exist two different strings $s_1, s_2 \in \mathcal{L}(G)$ such that $P(s_1) = P(s_2)$ and $\delta(q_0, s_1) = \delta(q_0, s_2)$ if and only if $X_{merge} \neq \emptyset$.*

Proof. ($\Leftarrow$) Suppose that $X_{merge} \neq \emptyset$. This means that there exists a state $(x_1, x_2) \in X_V$ such that either Eq. (12) holds or Eq. (13) holds. Let $s = (s_1, s_2)$ be a string such that $f_V(x_{0,V}, s) = (x_1, x_2)$, where we know that $P(s_1) = P(s_2)$ and $s_1, s_2 \in \mathcal{L}(G)$. We define two new strings s_1' and s_2' such that
$$s_i' = \begin{cases} s_i\sigma_0 & \text{if Eq. (12) holds} \\ s_i e_i & \text{if Eq. (13) holds}\end{cases} \tag{14}$$
where $i = 1{,}2$, σ_0 is an event satisfying Eq. (12) and $e_1, e_2 \in \Sigma_{uo}^*$ are strings satisfying Eq. (13). Clearly, if Eq. (12) holds, we know that $P(s_1') = P(s_1)\sigma_0 = P(s_2)\sigma_0 = P(s_2')$. Similarly, if Eq. (13) holds, we also have that $P(s_1') = P(s_1) = P(s_2) = P(s_2')$. Moreover, since either $x_1 \neq x_2$ or $e_1 \neq e_2$, we know that $s_1' \neq s_2'$, i.e., they are two different strings. Overall, we know that $P(s_1') = P(s_2')$ and $\delta(q_0, s_1') = \delta(q_0, s_2')$.

($\Rightarrow$) Suppose that there exist two different strings $s_1, s_2 \in \mathcal{L}(G)$ such that $P(s_1) = P(s_2)$ and $\delta(q_0, s_1) = \delta(q_0, s_2)$. We denote by $l_1 \leq s_1$ the shortest prefix of s_1 such that
$$\exists l' \leq s_2 : P(l_1) = P(l') \wedge l_1 \neq l' \wedge \delta(q_0, l_1) = \delta(q_0, l') \tag{15}$$
Note that l_1 is well-defined, since Eq. (15) always holds by taking $l_1 = s_1$ and $l' = s_2$. We denote by l_2 a string satisfying Eq. (15) for l_1. Also, we know that $l_1 \neq \epsilon$; otherwise, it implies that there exists an unobservable cycle at q_0 in G, which contradicts the assumption. Similarly, we know that $l_2 \neq \epsilon$. Therefore, we write $l_1 = \hat{l}_1\sigma_1$ and $l_2 = \hat{l}_2\sigma_2$, where σ_1 and σ_2 are the last events in l_1 and l_2, respectively. Then we consider the following four cases for σ_1 and σ_2:

Case 1: $\sigma_1, \sigma_2 \in \Sigma_o$.

In this case, first, we know that $\sigma_1 = \sigma_2$; otherwise, $P(l_1) \neq P(l_2)$. Second, we know that $\delta(q_0, \hat{l}_1) \neq \delta(q_0, \hat{l}_2)$; otherwise $\hat{l}_1$ will be the shortest prefix of s_1 satisfying Eq. (15). Therefore, since $P(\hat{l}_1) = P(\hat{l}_2)$, we know that state $x := (\delta(q_0, \hat{l}_1), \delta(q_0, \hat{l}_2))$ is reachable in V. Hence, state x and event $\sigma := \sigma_1 = \sigma_2$ satisfy Eq. (12), i.e., $x \in X_{merge} \neq \emptyset$.

Case 2: $\sigma_1 \in \Sigma_o$ and $\sigma_2 \in \Sigma_{uo}$.

In this case, we know that the last observable event in $\hat{l}_2$ must be σ_1; otherwise, $P(l_1) \neq P(l_2)$. Therefore, we can write l_2 by $l_2 = w_2\sigma_1\xi_2$, where $\xi_2 \in \Sigma_{uo}^*$. Also, we know that $\delta(q_0, \hat{l}_1\sigma_1) \neq \delta(q_0, w_2\sigma_1)$; otherwise, ξ_2 yields an unobservable cycle at $\delta(q_0, w_2\sigma_1)$. Since $P(\hat{l}_1\sigma_1) = P(w_2\sigma_1)$, we know that $x := (\delta(q_0, \hat{l}_1\sigma_1), \delta(q_0, w_2\sigma_1))$ is reachable in V. Moreover, (σ_1, σ_1)

$\in \mathrm{In}_V(x)$, i.e., $\mathrm{In}_V(x) \cap (\Sigma_o \times \Sigma_o) \neq \emptyset$. Hence, state x satisfies Eq. (13) by taking $e_1 = \epsilon$ and $e_2 = \xi_2$ and we know that $x \in X_{merge} \neq \emptyset$.

Case 3: $\sigma_1 \in \Sigma_{uo}$ and $\sigma_2 \in \Sigma_o$.

This case is analogous to Case 2 and we still can show that $X_{merge} = \emptyset$.

Case 4: $\sigma_1 \in \Sigma_{uo}$ and $\sigma_2 \in \Sigma_{uo}$.

First, suppose that $P(l_1) = P(l_2) = \epsilon$. Then we can take $(q_0, q_0) = x_{0,V}$ and we know that $\delta(q_0, l_1) = \delta(q_0, l_2)$ and $l_1 \neq l_2$, i.e., Eq. (13) is satisfied. Next, suppose that $P(l_1) = P(l_2) \neq \epsilon$. Let σ be the last observable event in l_1 and l_2 and we write $l_1 = w_1\sigma\xi_1$ and $l_2 = w_2\sigma\xi_2$, where $\xi_1, \xi_2 \in \Sigma_{uo}^*$. First, we know that $\delta(q_0, w_1\sigma) \neq \delta(q_0, w_2\sigma)$; otherwise $w_1\sigma$ will be the shortest prefix of s_1 satisfying Eq. (15). Since $P(w_1\sigma) = P(w_2\sigma)$, we know that state $x = (\delta(q_0, w_1\sigma), \delta(q_0, w_2\sigma))$ is reachable in V. Moreover, $(\sigma, \sigma) \in \mathrm{In}_V(x)$, i.e., $\mathrm{In}_V(x) \cap (\Sigma_o \times \Sigma_o) \neq \emptyset$. Therefore, by taking $x = (\delta(q_0, w_1\sigma), \delta(q_0, w_2\sigma))$, $e_1 = \xi_1$ and $e_2 = \xi_2$, we know that Eq. (13) is satisfied, i.e., $x \in X_{merge} \neq \emptyset$. □

4.2. Verification of trajectory detectability

Let $x \in X_V$ be a state in V. We say that x is in a cycle of V if there exists a non-empty string $s \in \Sigma_V^*$ such that $f_V(x, s) = x$. We denote by $\mathcal{C}_V \subseteq X_V$ the set of states in some cycle of V.

Now, we are ready to show how to verify trajectory detectability by using DFA V.

Theorem 2. *Let $G = (Q, \Sigma, \delta, q_0, Q_m)$ be a DFA and V be the DFA constructed from G. Then $\mathcal{L}_m(G)$ is trajectory detectable w.r.t. Σ_o if and only if $X_{merge} = \emptyset$ and*

$$Acc_V(\mathcal{C}_V) \cap (X_{m,V} \setminus \{(q, q) : q \in Q_m\}) = \emptyset \tag{16}$$

That is, V does not contain a merging state and any cycle in V cannot reach a marked state in which the first and the second components are not identical.

Proof. ($\Rightarrow$) By contraposition. First, suppose that Eq. (16) does not hold. Let $x = (x_1, x_2)$ be a state in $\mathrm{Acc}_V(\mathcal{C}_V) \cap (X_{m,V} \setminus \{(q, q) : q \in Q_m\})$. Since $x \in \mathrm{Acc}_V(\mathcal{C}_V)$, we know that there exists a string

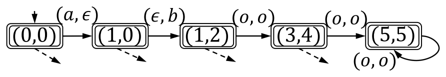
$$\underbrace{\sigma_0\sigma_1 \ldots \sigma_p}_{=:s} \underbrace{\sigma_{p+1} \ldots \sigma_{p+m}}_{=:w} \underbrace{\sigma_{p+m+1} \ldots \sigma_{p+m+k}}_{=:t} \in \mathcal{L}(V)$$

such that

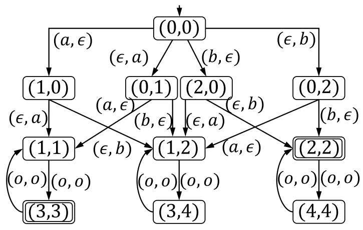
1. $f_V(x_{0,V}, sw^n t) = x$ for any $n \in \mathbb{N}$; and
2. $\exists i = 1, \ldots, m : \sigma_{p+i} \in \Sigma \times \Sigma$.

The first condition says that string w forms a cycle and the second condition says that w must contain an event in the form of (σ, σ), $\sigma \in \Sigma_o$. This comes from Assumption A-2, since we know that a state in $\mathcal{C}_V$ can never reach itself by a string in $((\Sigma_{uo} \times \{\epsilon\}) \cup (\{\epsilon\} \times \Sigma_{uo}))^*$; otherwise, it implies that there exists an unobservable cycle in G. For strings s, w and t, we denote by s_i, w_i and t_i, $i = 1,2$ their ith components, respectively, i.e., $swt = (s_1, s_2)(w_1, w_2)(t_1, t_2)$. We know that $P(s_1) = P(s_2)$, $P(w_1) = P(w_2)$ and $P(t_1) = P(t_2)$. Moreover, we know that $|P(w_1)| = |P(w_2)| \geq 1$. Since $x \in X_m \setminus \{(q, q) : q \in Q_m\}$, we know that $s_i w_i^n t_i \in \mathcal{L}_m(G), \forall i = 1, 2, \forall n \in \mathbb{N}$ and $s_1 w_1^n t_1 \neq s_2 w_2^n t_2, \forall n \in \mathbb{N}$. Therefore, for any $n \in \mathbb{N}$, we can find a string $s_1 w_1^n t_1 \in \mathcal{L}_m(G)$ such that $|P(s_1 w_1^n t_1)| \geq |P(w_1^n)| \geq 1$ and $P^{-1}P(s_1 w_1^n t_1) \supseteq \{s_1 w_1^n t_1, s_2 w_2^n t_2\}$, i.e., $\mathrm{Card}[P^{-1}P(s_1 w_1^n t_1) \cap \mathcal{L}_m(G)] > 1$. This implies that $\mathcal{L}_m(G)$ is not trajectory detectable.

Next, we suppose that $X_{merge} \neq \emptyset$. By Theorem 1, we know that there exist two different strings $s_1, s_2 \in \mathcal{L}(G)$ such that $P(s_1) = P(s_2)$ and $\delta(q_0, s_1) = \delta(q_0, s_2) =: q$. Therefore, for any $n \in \mathbb{N}$, we can choose a string w defined at q, such that $\delta(q, w) \in X_m$, $|w| \geq n$ and $|P(w)| \geq n$. Such a string w is well-defined since we assume that G is live and non-blocking and there is no unobservable cycle in G. For the above s_1, s_2 and w, we know that $s_1 w, s_2 w \in \mathcal{L}_m(G)$ and $P(s_1 w) = P(s_2 w)$, i.e., $\mathrm{Card}[P^{-1}P(s_1 w) \cap \mathcal{L}_m(G)] > 1$. This implies that $\mathcal{L}_m(G)$ is not trajectory detectable.



(a) Part of the DFA V for G_1 in Figure 1(a).



(b) The DFA V for G_4 in Figure 2(a).

Fig. 3. Examples of the verification DFA V.

($\Leftarrow$) Still by contraposition. Suppose that $\mathcal{L}_m(G)$ is not trajectory detectable. This implies that there exist two different arbitrarily long strings in $\mathcal{L}_m(G)$ having the same projection. Since the state-space of G is finite, any arbitrarily long string must be contributed by some cycle. Therefore, we can find strings $t_i, w_i, v_i \in \Sigma^*, i = 1, 2$ such that $P(t_1) = P(t_2), P(w_1) = P(w_2), P(v_1) = P(v_2)$ and $\delta(q_0, t_1 w_1^n v_1), \delta(q_0, t_2 w_2^n v_2) \in X_m$ for any $n \in \mathbb{N}$, i.e., $t_1 w_1^n v_1, t_2 w_2^n v_2 \in \mathcal{L}_m(G)$. Next, we consider two cases:

Case 1: $\delta(q_0, t_1 w_1 v_1) \neq \delta(q_0, t_2 w_2 v_2)$;
Case 2: $\delta(q_0, t_1 w_1 v_1) = \delta(q_0, t_2 w_2 v_2)$.

First, assume that Case 1 holds. Let us consider the state reached by $sw := (s_1, s_2)(w_1, w_2)$ in V, say x_w. Since $sw^n \in \mathcal{L}(V)$ for any $n \in \mathbb{N}$, we know that $x_w \in \mathcal{C}_V$. Since $x_v := (\delta(q_0, t_1 w_1 v_1), \delta(q_0, t_2 w_2 v_2)) \in X_{m,V}$ and $\delta(q_0, t_1 w_1 v_1) \neq \delta(q_0, t_2 w_2 v_2)$, we know that $x_v \in X_m \setminus \{(q, q) : q \in Q_m\}$. Moreover, since state x_w can reach state x_v via string $v = (v_1, v_2)$, we know that $x_v \in \mathrm{Acc}_V(\mathcal{C}_V) \cap (X_m \setminus \{(q, q) : q \in Q_m\}) \neq \emptyset$. Second, we assume that Case 2 holds. Then, since $t_1 w_1 v_1 \neq t_2 w_2 v_2$, $P(t_1 w_1 v_1) = P(t_2 w_2 v_2)$ and $\delta(q_0, t_1 w_1 v_1) = \delta(q_0, t_2 w_2 v_2)$, by Theorem 1, we know immediately that $X_{merge} \neq \emptyset$. This completes the contraposition proof. □

Let us illustrate how to use Theorem 2 to verify trajectory detectability by the following example.

Example 4. First, let us consider language $\mathcal{L}(G_1)$ generated by DFA G_1 shown in Fig. 1(a). Note that, since the definition of trajectory detectability considers marked languages, we need to mark all states in G_1 such that $\mathcal{L}_m(G_1) = \mathcal{L}(G_1)$. Therefore, all states in V are also marked states. Then part of the verification DFA V for G_1 is shown in Fig. 3(a). Note that the self-loop at state $(5, 5) \in X_{m,V}$ does not violate Eq. (16), since the first and the second components in $(5, 5)$ are the same. However, state $(3, 4)$ is a merging state, since (i) it is entered by $(o, o) \in \Sigma_o \times \Sigma_o$, i.e., $\mathrm{In}_V((3, 4)) \cap (\Sigma_o \times \Sigma_o) \neq \emptyset$; and (ii) state 3 and 4 are different states; and (iii) event $(o, o) \in \Sigma_o \times \Sigma_o$ is defined at $(3, 4)$ and it leads to state $(5, 5)$ whose two components are the same state. Therefore, we know that $\mathcal{L}(G_1)$ is not trajectory detectable. Note that, since trajectory detectability does not depend on the DFA generating the language, one can also construct the verification DFA V based on G_2 shown in Fig. 1(b), which will give the same result.

Next, let us consider language $\mathcal{L}(G_4)$ marked by DFA G_4 shown in Fig. 2(a). The complete verification DFA V for G_4 is shown in Fig. 3(b). First, we can easily check that there is no merging state in V, i.e., $X_{merge} = \emptyset$. Second, for states that can be reached from a cycle in V, i.e., states $(1,1),(3,3),(1,2),(3,4),(2,2)$ and $(4,4)$, they are either unmarked states or marked states but in the form of (q,q). Therefore, we know that $\mathcal{L}_m(G_4)$ is trajectory detectable. However, if we want to consider the language generated by G_4, i.e., $\mathcal{L}(G_4)$, then we need to mark all states in G_4. In this case, states $(1,2)$ and $(3,4)$ become marked states and they are still in a cycle. Therefore, we know that the generated language $\mathcal{L}(G_4)$ is not trajectory detectable. □

Remark 6. Let us explain the intuition behind Theorem 2. First, the condition in Eq. (16) requires that, for any marked state that can be reached from a cycle in V, it must be in the form of $\{(q,q): q \in Q_m\}$. Otherwise, suppose that a state $(q_1,q_2) \in Q_m \times Q_m, q_1 \neq q_2$ is reached from a cycle. Then we know that there exist two arbitrarily long strings in $\mathcal{L}_m(G)$ leading to q_1 and q_2, respectively, such that they have the same observation. Moreover, $s_1 \neq s_2$ since $q_1 \neq q_2$. However, only requiring this condition is not sufficient to recover the trajectory, since it is possible that all states in some cycle are in the form of $\{(q,q): q \in Q_m\}$ but two strings may merge to a same state before they enter some cycle. If such a scenario occurs, then we can identify the current state of the system restricting to Q_m, but we fail to identify the precise trajectory leading to this state. This is why we need that $X_{merge} \neq \emptyset$ in addition to Eq. (16). □

Remark 7. We discuss the complexity for verifying trajectory detectability by using Theorem 2. Let $G = (Q, \Sigma, \delta, q_0, Q_m)$ be the DFA for which we want to verify whether or not $\mathcal{L}_m(G)$ is trajectory detectable. First, we need to construct the verification DFA V, which takes $O(|\Sigma||Q|^2)$, since there are at most $|Q|^2$ states and $3|\Sigma||Q|^2$ transitions in V. Then for each state in V, it takes $|\Sigma||Q|$ to determine whether it is a merging state or not. Therefore, it takes $O(|\Sigma|^2|Q|^3)$ to check whether or not $X_{merge} = \emptyset$. Finally, we need to check whether or not there exists a state violating Eq. (16). To this end, we can first compute $\mathcal{C}_V$, which can be done in $O(|\Sigma||Q|^2)$ by using Kosaraju's algorithm; see, e.g., [27]. Computing the reachable set $\text{Acc}_V(\mathcal{C}_V)$ still requires $O(|\Sigma||Q|^2)$. Overall, the total complexity of the verification of trajectory detectability is $O(|\Sigma|^2|Q|^3)$, which is polynomial w.r.t. the size of G. □

4.3. Verification of periodic trajectory detectability

Now, let us investigate the verification of periodic trajectory detectability. In contrast to the verification of trajectory detectability, we cannot use the verification DFA V to verify periodic trajectory detectability. Instead, we use the observer structure [24].

Let $G = (Q, \Sigma, \delta, q_0, Q_m)$ be a DFA. The *observer* of G w.r.t. Σ_o is a DFA $Obs(G) = (X_{obs}, \Sigma_o, f_{obs}, x_{0,obs})$, where $X_{obs} \subseteq 2^Q$, $x_{0,obs} = R_G(\epsilon)$ and $f_{obs} : X_{obs} \times \Sigma_o \to X_{obs}$ such that, by extending it to $X_{obs} \times \Sigma_o^*$, we have $f_{obs}(x_{0,obs}, s) = R_G(s)$ for any $s \in P(\mathcal{L}(G))$.

Let $x \in X_{obs}$ be a state in the observer. We say that x is

- a *certain state* if $\exists q \in x : q \in Q_m \wedge x \setminus \{q\} \subseteq Q \setminus Q_m$;
- an *uncertain state* if $\exists q_1, q_2 \in x : q_1, q_2 \in Q_m \wedge q_1 \neq q_2$.

We denote by X_{obs}^{ct} and X_{obs}^{uc} the set of certain states and the set of uncertain states, respectively. Intuitively, a certain state only contains one marked state and a uncertain state contains some different marked states. Also, we say that a set of distinct states $\{x_1, \ldots, x_n\} \subseteq X_{obs}$ is a cycle in $Obs(G)$ if we can find a sequence of events $\sigma_1, \ldots, \sigma_n$ such that $f_{obs}(x_i, \sigma_i) = x_{i+1}, \forall i = 1, \ldots, n$, where $x_{n+1} := x_1$.

The following result reveals how to use the observer to verify periodic trajectory detectability.

Theorem 3. *Let $G = (Q, \Sigma, \delta, q_0, Q_m)$ be a DFA and $Obs(G)$ be the observer w.r.t. Σ_o. Then $\mathcal{L}(G)$ is periodically trajectory detectable w.r.t. Σ_o if and only if*

(i) *$X_{merge} = \emptyset$; and*
(ii) *for any cycle $C \subseteq X_{obs}$ in $Obs(G)$, if $C \cap X_{obs}^{ct} = \emptyset$, then any state in C cannot reach a state in X_{obs}^{uc} through states in $X_{obs} \setminus X_{obs}^{ct}$.*

Proof. ($\Rightarrow$) By contraposition. Suppose that $X_{merge} \neq \emptyset$. We know that there exist two strings $s_1, s_2 \in \mathcal{L}(G)$ such that $P(s_1) = P(s_2)$ and $\delta(q_0, s_1) = \delta(q_0, s_2) =: q$. Let $\hat{s}_1 \leq s_1$ be the longest prefix of s_1 such that $\hat{s}_1 \in \mathcal{L}_m(G)$ and we write $s_1 = \hat{s}_1 s_1'$. Let w be an arbitrarily long string from q to a marked state. Then for any $n \in \mathbb{N}$, there exists a string $\hat{s}_1 \in \mathcal{L}_m(G)$ and a continuation $s_1'w \in \mathcal{L}_m(G)/\hat{s}_1$ such that $|P(s_1'w)| \geq n$. Moreover, for any $w' \leq w : \hat{s}s_1'w \in \mathcal{L}_m(G)$, we have $\text{Card}[P^{-1}P(\hat{s}_1 s_1' w') \cap \mathcal{L}_m] > 1$, since $s_2 w' \neq \hat{s}_1 s_1' w'$ and $P(\hat{s}_1 s_1' w') = P(s_1 w') = P(s_2 w')$. This implies that $\mathcal{L}_m(G)$ is not periodically trajectory detectable.

Now, let us suppose that there exists a cycle C such that $C \cap X_{obs}^{ct} = \emptyset$ and a state in it can reach a state in X_{obs}^{uc} through states in $X_{obs} \setminus X_{obs}^{ct}$. Let $x \in C$ be such a state in the cycle. Let $s_o \in \Sigma_o^*$ be a string that reaches x from $x_{0,obs}$. Let $w_o \in \Sigma_o^*$ be a string such that $f_{obs}(x, w_o) = x$ and it only visits all states in C. Let $t_o \in \Sigma_o^*$ be a string that reaches a state in X_{obs}^{uc} from x through states in $X_{obs} \setminus X_{obs}^{ct}$. Formally, we have that $f_{obs}(x, t_o) \in X_{obs}^{uc}$ and $\forall t_o' \leq t_o : f_{obs}(x, t_o') \notin X_{obs}^{ct}$. By the property of the observer, we know that, for any $n \in \mathbb{N}$, there exists a string $sw^n t \in P^{-1}(s_o w_o^n t_o)$ such that $P(s) = s_o, P(w) = w_o, P(t) = t_o$ and $\delta(q_0, sw^n t) \in Q_m$. Still, we denote by $\hat{s} \leq s$ the longest prefix of s such that $\hat{s} \in \mathcal{L}_m(G)$ and write $s = \hat{s}s'$. Then for any $n \in \mathbb{N}$, there exists a string $\hat{s} \in \mathcal{L}_m(G)$ and a continuation $s'w^n t \in \mathcal{L}_m(G)/\hat{s}$ such that $|P(s'w^n t)| \geq n$. Moreover, for any $w' \leq w^n t : \hat{s}s'w' \in \mathcal{L}_m(G)$, we have $\text{Card}[P^{-1}P(\hat{s}s'w') \cap \mathcal{L}_m(G)] > 1$, since $f_{obs}(x_{0,obs}, P(\hat{s}_1 s' w'))$ is always not a certain state.

($\Leftarrow$) Still by contraposition. Suppose that $\mathcal{L}_m(G)$ is not periodically trajectory detectable. This implies that there exists a string $s \in \mathcal{L}_m(G)$ and an arbitrarily long continuation $u \in \mathcal{L}_m(G)/s$ such that $\forall u' \leq u : su' \in \mathcal{L}_m(G)$, we have that $\text{Card}[P^{-1}P(su') \cap \mathcal{L}_m(G)] > 1$. Since the state-spaces of G and $Obs(G)$ are both finite, any arbitrarily long string and its projection must be contributed by some cycle in G and some cycle in $Obs(G)$, respectively. Therefore, we can find string swt such that $\forall n \in \mathbb{N}$, $\delta(q_0, sw^n t) \in \mathcal{L}_m(G)$, $f_{obs}(x_{0,obs}, P(s)P(w)^n P(t))!$ and $(\forall w' \leq w^n t : sw' \in \mathcal{L}_m(G))[\text{Card}[P^{-1}P(sw') \cap \mathcal{L}_m(G)] > 1]$.

Next, we consider the following two cases. First, assume that

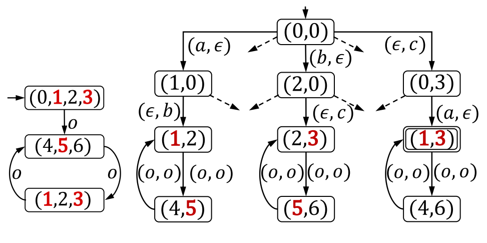
$$(\exists w' \leq wt : sw' \in \mathcal{L}_m(G))(\exists s_2 \in \mathcal{L}_m(G)) \tag{17}$$
$$[s_2 \neq sw' \wedge P(sw') = P(s_2) \wedge \delta(q_0, sw') = \delta(q_0, s_2)]$$

Then, by Theorem 1, we know that $X_{merge} \neq \emptyset$. Therefore, hereafter, we assume that

$$(\forall w' \leq wt : sw' \in \mathcal{L}_m(G))(\forall s_2 \in \mathcal{L}_m(G)) \text{ s.t.} \tag{18}$$
$$[s_2 \neq sw' \wedge P(sw') = P(s_2)] \Rightarrow [\delta(q_0, sw') \neq \delta(q_0, s_2)]$$

Let $x_{sw} := f_{obs}(x_{0,obs}, P(sw))$ and $x_{swt} := f_{obs}(x_{0,obs}, P(swt))$. First, by $\text{Card}[P^{-1}P(swt) \cap \mathcal{L}_m(G)] > 1$ and Eq. (18), we know that $x_{swt} \in X_{obs}^{uc}$. Moreover, for any $w' \leq wt$, we know that $f_{obs}(x_{0,obs}, P(sw')) \notin X_{obs}^{ct}$, since if it contains one state in Q_m, then it must contain a distinct state in Q_m. Therefore, we know that x_{sw} is in a cycle that does not contain a certain state and x_{sw} can reach uncertain state x_{swt} without reaching certain states. This completes the contrapositive proof. □

Example 5. Let us consider $\mathcal{L}_m(G_5)$ marked by DFA G_5 shown in Fig. 2(b) with $\Sigma_o = \{o\}$. First, we can show that $X_{merge} = \emptyset$. Then the observer $Obs(G_5)$ is shown in Fig. 4(a). We know that state $\{4,5,6\}$ is a certain state and state $\{1,2,3\}$ is an uncertain state. However, these two states form the only cycle in $Obs(G_5)$ and one of which is a certain state. Therefore, we know that $\mathcal{L}_m(G_5)$ is periodically trajectory detectable. Note that, if we consider $\mathcal{L}(G_5)$,



(a) $Obs(G_5)$ (b) The DFA V for G_5 in Figure 2(b).

Fig. 4. Examples of the verification DFA V.

i.e, we need to mark all states in G_5, then both $\{1, 2, 3\}$ and $\{4, 5, 6\}$ are uncertain states, which means that the generated language $\mathcal{L}(G_5)$ is not periodically trajectory detectable. □

Remark 8. Note that, in the worst case, the size of the observer is exponential in the size of G. To implement Theorem 3, first, we need to check whether or not $X_{merge} = \emptyset$. This can still be done by using the verification DFA V and it takes $O(|\Sigma|^2|Q|^3)$. Determining whether or not there exists a cycle that does not contain a certain state can be done in $O(|X_{obs}||\Sigma_o|)$. Checking whether or not there exists a state in such a cycle that can reach an uncertain state without visiting certain states can still be done in $O(|X_{obs}||\Sigma_o|)$ by a depth-first search. Therefore, the complexity for verifying periodic trajectory detectability is $O(|\Sigma|2^{|Q|} + |\Sigma|^2|Q|^3)$, which is exponential in the size of G.

Remark 9. Let us explain why DFA V cannot be used to check periodic trajectory detectability. Let us consider again the system G_5 studied in Example 5 and we construct part of the verification DFA V shown in Fig. 4(b). However, we cannot determine the "certainty" of a trajectory based on V. For example, state $(4, 5)$, where 5 is a marked state, corresponds to certain state $\{4, 5, 6\}$ in $Obs(G_5)$. However, state $(1, 2)$, which also only contains a single marked state, corresponds to uncertain state $\{1, 2, 3\}$ in $Obs(G_5)$. This is because, although state $(1, 2)$ looks like "certain" in the leftmost cycle, the marked state in it can be confused with other marked state in a different cycle, e.g., the rightmost cycle in Fig. 4(b). Therefore, exponential complexity arises when this combination is considered. One interesting future direction is to identify the precise complexity class for the verification of periodic trajectory detectability.

5. Conclusion

We investigated the trajectory estimation problem for partially-observed DES. A language-based framework was proposed to study this problem. The notions of trajectory detectability and periodic trajectory detectability were proposed in order to capture different requirements in practice. We provided effective algorithms to verify these notions when the system language is regular.

There are several future directions for the proposed framework. First, in this paper, we only consider to verify a priori if the trajectory of a system can be detected. Investigating effective online detection mechanism is also an important future direction. Second, it is important to investigate the decidability of trajectory detectability for other classes of languages, e.g. Petri nets languages [28]. Finally, the observation mapping considered in this paper is static. Extending the proposed framework to the dynamic observation setting [29,30] or the non-deterministic observation setting [31–33] is also an interesting direction.

References

[1] P. Ramadge, Observability of discrete event systems, in: 25th IEEE Conf. Decision and Control, no. 25, 1986, pp. 1108–1112.
[2] C. Özveren, A. Willsky, Observability of discrete event dynamic systems, IEEE Trans. Automat. Control 35 (7) (1990) 797–806.
[3] W. Wang, S. Lafortune, F. Lin, An algorithm for calculating indistinguishable states and clusters in finite-state automata with partially observable transitions, Syst. Control Lett. 56 (9) (2007) 656–661.
[4] S. Shu, F. Lin, H. Ying, Detectability of discrete event systems, IEEE Trans. Automat. Control 52 (12) (2007) 2356–2359.
[5] S. Shu, F. Lin, Detectability of discrete event systems with dynamic event observation, Syst. Control Lett. 59 (1) (2010) 9–17.
[6] D. Sears, K. Rudie, On computing indistinguishable states of nondeterministic finite automata with partially observable transitions, in: 53rd IEEE Conf. Decision and Control, 2014, pp. 6731–6736.
[7] F. Lin, W. Wonham, On observability of discrete-event systems, Inform. Sci. 44 (3) (1988) 173–198.
[8] S. Shu, F. Lin, Generalized detectability for discrete event systems, Syst. Control Lett. 60 (5) (2011) 310–317.
[9] C. Keroglou, C. Hadjicostis, Detectability in stochastic discrete event systems, Syst. Control Lett. 84 (2015) 21–26.
[10] C. Hadjicostis, C. Seatzu, K-detectability in discrete event systems, in: 55th IEEE Conf. Decision and Control, 2016, pp. 420–425.
[11] X. Yin, Initial-state detectability of stochastic discrete-event systems with probabilistic sensor failures, Automatica 80 (2017) 127–134.
[12] K. Zhang, The problem of determining the weak (periodic) detectability of discrete event systems is pspace-complete, Automatica 81 (2017) 217–220.
[13] T. Masopust, Complexity of deciding detectability in discrete event systems, Automatica 93 (2018) 257–261.
[14] X. Yin, S. Lafortune, Verification complexity of a class of observational properties for modular discrete events systems, Automatica 83 (2017) 199–205.
[15] S. Shu, Z. Huang, F. Lin, Online sensor activation for detectability of discrete event systems, IEEE Trans. Autom. Sci. Eng. 10 (2) (2013) 457–461.
[16] X. Yin, S. Lafortune, A general approach for solving dynamic sensor activation problems for a class of properties, in: 54th IEEE Conf. Decision and Control, 2015, pp. 3610–3615.
[17] S. Shu, F. Lin, Enforcing detectability in controlled discrete event systems, IEEE Trans. Automat. Control 58 (8) (2013) 2125–2130.
[18] X. Yin, S. Lafortune, A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems, IEEE Trans. Automat. Control 61 (8) (2016) 2140–2154.
[19] R. Jungers, V. Blondel, Observable graphs, Discrete Appl. Math. 159 (10) (2011) 981–989.
[20] Y.-C. Wu, K. Sankararaman, S. Lafortune, Ensuring privacy in locationbased services: An approach based on opacity enforcement, in: Proc. 14th Int. Workshop of Discrete Event Systems, 2014, pp. 33–38.
[21] S. Jiang, Z. Huang, V. Chandra, R. Kumar, A polynomial algorithm for testing diagnosability of discrete-event systems, IEEE Trans. Automat. Control 46 (8) (2001) 1318–1321.
[22] T.-S. Yoo, S. Lafortune, Polynomial-time verification of diagnosability of partially observed discrete-event systems, IEEE Trans. Autom. Control 47 (9) (2002) 1491–1495.
[23] C. Özveren, A. Willsky, Invertibility of discrete-event dynamic systems, Math. Control Signals Systems 5 (4) (1992) 365–390.
[24] C. Cassandras, S. Lafortune, Introduction to Discrete Event Systems, second ed., Springer, 2008.
[25] F. Lin, Opacity of discrete event systems and its applications, Automatica 47 (3) (2011) 496–503.
[26] J. Zaytoon, S. Lafortune, Overview of fault diagnosis methods for discrete event systems, Annu. Rev. Control 37 (2) (2013) 308–320.
[27] R. Sedgewick, K. Wayne, Algorithms, fourth ed., Addison-Wesley Professional, 2011.
[28] X. Yin, S. Lafortune, On the decidability and complexity of diagnosability for labeled Petri nets, IEEE Trans. Automat. Control 62 (11) (2017) 5931–5938.
[29] X. Yin, S. Lafortune, Codiagnosability and coobservability under dynamic observations: Transformation and verification, Automatica 61 (2015) 241–252.
[30] D. Sears, K. Rudie, Minimal sensor activation and minimal communication in discrete-event systems, Discrete Event Dyn. Sys.: Theory & Appl. 26 (2) (2016) 295–349.
[31] S. Takai, T. Ushio, Verification of codiagnosability for discrete event systems modeled by mealy automata with nondeterministic output functions, IEEE Trans. Automat. Control 57 (3) (2012) 798–804.
[32] T. Ushio, S. Takai, Nonblocking supervisory control of discrete event systems modeled by mealy automata with nondeterministic output functions, IEEE Trans. Automat. Control 61 (3) (2016) 799–804.
[33] X. Yin, Supervisor synthesis for mealy automata with output functions: a model transformation approach, IEEE Trans. Automat. Control 62 (5) (2017) 2576–2581.