2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Decentralized State Estimation via Breadth-First Search through Partially Ordered Observation Sequences

Dajiang Sun, Christoforos N. Hadjicostis, and Zhiwu Li

Abstract— **We investigate the state estimation problem under a decentralized observation architecture. More specifically, we consider a discrete event system, modeled by a nondeterministic finite automaton, whose behavior is partially observed and recorded at a set of observation sites with distinct capabilities. When prompted, these observation sites send their sequences of observations to a coordinator that fuses and analyzes this information to estimate the specific system states of interest (current- and initial-states). The notion of S-builder is introduced to systematically infer possible (totally ordered) sequences of observations and an algorithm is proposed for constructing a synchronizer in a breadth-first search manner to efficiently perform current-state estimation. With slight extensions, the synchronizer construction algorithm can be also applied towards initial-state estimation.**

I. INTRODUCTION

The state estimation problem is important in many applications. For a discrete event system (DES), the state estimation problem was first introduced in [1], where the notion of observability was formally defined in the presence of unobservable events. In a partially observed DES, verification of properties, such as diagnosability [2]–[5], detectability [6], [7], and opacity [8], [9], as well as supervisory control strategies [10], [11], have been systematically studied. Determining the exact state or estimating a set of possible states [12], [13] (which necessarily includes the true state of the system) is fundamental for achieving these objectives.

Depending on the time instant at which the set of system states is needed, different notions have been proposed and studied, such as current-state estimation, delayed-state estimation and initial-state estimation [14]. Dealing with these estimation tasks depends on the observed system behavior. In decentralized observation settings, the system is observed by a set of local sites, each of which includes an observation site and a computational unit. Local sites occasionally send local information or decisions to a coordinator; such a procedure is called *synchronization* [14].

The decentralized information processing we adopt in this work is as follows. During the system evolution, each local site, based on its own observations, decides whether or not its preserved observation sequence should be sent to the coordinator. If a local site decides to send its observation sequence, it signals its intention to the coordinator, which immediately initiates synchronization globally by requesting information from all local sites. Then, the coordinator uses all received information to estimate the state of the system.

Each synchronization is assumed to happen instantaneously following the request from the coordinator, i.e., there are no occurrences of events between signaling and synchronization. We refer to the local observation sequences that the local sites send, as well as the rule that the coordinator adopts to process the local information it receives at each synchronization step, as the protocol for decentralized observation-based information processing (DO-based protocol). In the remainder of this work, we will refer to observation sites (OSs) instead of local sites, since local sites simply report their sequences of observations.

The main contributions of this paper are as follows. First, compared with the earlier work in [16], we relax the requirement that the sets of local observable events are disjoint (i.e., we allow the sets of local observable events to be incomparable). Second, after characterizing the DO-based protocol formally, we propose the notion of *S-builder* whose marked language contains all possible system behavior that matches the reported (partially ordered) observation sequences. The S-builder can encode all possible totally ordered sequences in a systematic way, without having to explicitly enumerate them, in order to estimate system states. Then, based on the S-builder, an algorithm to construct a *synchronizer* via a breadth-first search is presented to estimate the states of the system. Finally, with a slight modification of the *synchronizer*, we argue that initial-state estimation can also be addressed.

II. PRELIMINARIES

Let Σ be a finite set of events. A string over Σ is a sequence of n symbols, i.e., $s = \alpha_1\alpha_2\ldots\alpha_n$, $\alpha_i \in \Sigma$, $i \in \{1, 2, \ldots, n\}$. The length of s is the number of symbols in the sequence and is denoted by $|s|$. We denote by Σ^* the set of all finite-length strings over Σ, including the empty string ϵ with $|\epsilon| = 0$. A language $L \subseteq \Sigma^*$ is a set of strings [14], [17]. Given two strings s, $t \in \Sigma^*$, the string $s \cdot t$ (or simply st) denotes the concatenation of s and t. We also write $t \in s$ if string t is a sub-string of s, i.e., $s = utv$ for some strings $u, v \in \Sigma^*$. Also, for any $\sigma \in \Sigma$, $s \in \Sigma^*$, we use $\sigma \in s$ to denote that σ occurs in s, i.e., $s = u\sigma v$ for some strings $u, v \in \Sigma^*$. We let $\bar{s}$ be the prefix-closure of s,

This work was supported in part by the China Scholarship Council, and the National Key R&D Program of China under Grant No. 2018YFB1700104.

D. J. Sun is with the School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China. E-mail: djsun@stu.xidian.edu.cn

C. N. Hadjicostis is with the Department of ECE, University of Cyprus, Nicosia, Cyprus. E-mail: hadjicostis.christoforos@ucy.ac.cy

Z. W. Li is with the Institute of Systems Engineering, Macau University of Science and Technology, Taipa, Macao SAR, China, and also with the School of Electro-Mechanical Engineering, Xidian University, Xi'an 710071, China. E-mail: zhwli@xidian.edu.cn

i.e., $\bar{s} = \{t \in \Sigma^* | \exists t' \in \Sigma^*, tt' = s\}$, and use s/t to denote the symbol sequence after t in s, i.e., for $t \in \bar{s}$, we have $t \cdot (s/t) = s$ (note that $t/t = \epsilon$).

A DES is modeled as a nondeterministic finite automaton (NFA) $G = (X, \Sigma, \delta, X_0)$, where X is the finite set of states, Σ is the finite set of events, $\delta : X \times \Sigma \to 2^X$ is the nondeterministic state transition function, and X_0 is the set of possible initial states. For a set $Q \subseteq X$ and $\sigma \in \Sigma$, we define $\delta(Q, \sigma) = \bigcup_{q \in Q} \delta(q, \sigma)$; with this notation at hand, the transition function δ can also be extended recursively to δ^* (denoted by δ for the sake of brevity) whose domain is $X \times \Sigma^*$ instead of $X \times \Sigma$: $\delta(x, \sigma s) = \delta(\delta(x, \sigma), s)$ for any $x \in X$, $\sigma \in \Sigma$, and $s \in \Sigma^*$. The system execution (or behavior) of G starting from state x is captured by $L(G, x) = \{s \in \Sigma^* | \delta(x, s) \neq \emptyset\}$. Then, the system behavior generated by G is $L(G) = \bigcup_{x_0 \in X_0} L(G, x_0)$. If there exists a set of marked states X_m, $X_m \subseteq X$, the marked behavior of G is $L_m(G) = \{s \in \Sigma^* | \exists x_0 \in X_0, \delta(x_0, s) \cap X_m \neq \emptyset\}$.

A state mapping $m^p \in 2^{X^2}$ is a subset of X^2. The composition operator $\circ : 2^{X^2} \times 2^{X^2} \to 2^{X^2}$ is defined as follows: for any $m_1^p, m_2^p \in 2^{X^2}$, $m_1^p \circ m_2^p = \{(x_1, x_3) | \exists x_2 \in X, \text{s.t.}, (x_1, x_2) \in m_1^p, (x_2, x_3) \in m_2^p\}$. We denote the set of starting (ending) states of m^p as $m^p[0] = \{x_s | (x_s, x_e) \in m^p\}$ ($m^p[1] = \{x_e | (x_s, x_e) \in m^p\}$).

III. Decentralized Observation-Based Information Processing

We assume that there are m observation sites O_i and let $\mathcal{I} = \{1, 2, \ldots, m\}$ denote the index set where $i \in \mathcal{I}$. We denote by Σ_{o_i} and Σ_{uo_i} the set of observable events and the set of unobservable events for OS O_i, respectively, such that $\Sigma_{o_i} \cap \Sigma_{uo_i} = \emptyset$ and $\Sigma_{o_i} \cup \Sigma_{uo_i} = \Sigma$. For any O_i, $i \in \mathcal{I}$, the natural projection function $P_{\Sigma_{o_i}} : \Sigma^* \to \Sigma_{o_i}^*$ is defined recursively as

$$P_{\Sigma_{o_i}}(\epsilon) = \epsilon \quad \text{and} \quad P_{\Sigma_{o_i}}(s\sigma) = \begin{cases} P_{\Sigma_{o_i}}(s)\sigma & \text{if} \quad \sigma \in \Sigma_{o_i} \\ P_{\Sigma_{o_i}}(s) & \text{if} \quad \sigma \notin \Sigma_{o_i}. \end{cases}$$

For the sake of notational simplicity, $P_{\Sigma_{o_i}}$ and Σ_{o_i} will be denoted by P_i and Σ_i, respectively. The domain of P_i is also extended to 2^{Σ^*}, i.e., $P_i(L) = \{\omega \in \Sigma_i^* | \exists s \in L, P_i(s) = \omega\}$.

For each event $\sigma \in \Sigma$, we use $I(\sigma) = \{i \in \mathcal{I} | \sigma \in \Sigma_i\}$ to denote the index set of local sites that can observe σ. For an event $\sigma \in \bigcup_{i \in \mathcal{I}} \Sigma_i$, we define for $i \in I(\sigma)$, $\Omega_i(\sigma) = \{\sigma\omega \in \Sigma_i^* | \exists x \in X, \sigma\omega \in P_i(L(G, x))\}$ to be the set of sequences of observations that can be observed by O_i and start with event σ. For any sequence of events t, $t \neq \epsilon$, we use t_f to denote the last symbol of t.

We assume all OSs have no knowledge of the system model. The coordinator knows the system model and is aware of Σ_i, $i \in \mathcal{I}$. We use the notation $P_{\mathcal{I}}$ to denote the natural projection function with respect to (w.r.t.) the set of observable events $\Sigma_{\mathcal{I}} = \bigcup_{i \in \mathcal{I}} \Sigma_i$. Next, we introduce the DO-based protocol which contains three ingredients:

1) The kind of information O_i sends to the coordinator: Each O_i uses its local projection function $P_i : L(G, X) \to \Sigma_i^*$ to record the sequence of observations. The stored observation sequence can be forgotten after each synchronization.

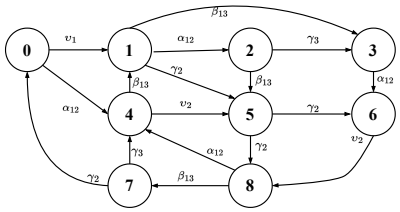

Fig. 1. An NFA model G. The subscripts of events in $\Sigma_{\mathcal{I}}$ match the indices of OSs that observe the specific events.

2) Synchronization strategy: Each O_i has its own decision-making policy as to whether or not to initiate a synchronization. In this paper, we do not specify a particular strategy since we focus on the state estimation task using the sequences of observations (which is independent of the specific synchronization strategy). The only thing that is needed is the subset of OSs that initiate a synchronization at a particular synchronization step (termed as critical index set and denoted by $\mathcal{I}^c$), since this information can be used to improve the accuracy of the state estimation process.

3) The coordinator utilizes the information received from OSs to obtain possible (totally ordered) sequences of observations. This task can be viewed as a synchronization function:

$$S : \Sigma_1^* \times \cdots \times \Sigma_m^* \to 2^{\Sigma_{\mathcal{I}}^*}.$$

We refer to the observation sequences provided by OSs at each synchronization as partially ordered sequences of observations (PO-sequences) [14]. The coordinator can reconstruct the possible system observation sequences $\omega_{\mathcal{I}} \in \Sigma_{\mathcal{I}}^*$ based on PO-sequences, $\tau^{(i)} \in P_i(L(G, X))$. In particular, we would like the following property for the synchronization function,

$$\omega_{\mathcal{I}} \in S(\tau^{(1)}, \cdots, \tau^{(m)})$$

if and only if $P_i(\omega_{\mathcal{I}}) = \tau^{(i)}$, for $i \in \mathcal{I}$.

Example 1: Let us consider system G shown in Fig. 1. Suppose that there are three OSs, i.e., $\mathcal{I} = \{1, 2, 3\}$, where $\Sigma_1 = \{\alpha_{12}, \beta_{13}\}$, $\Sigma_2 = \{\alpha_{12}, \gamma_2\}$, and $\Sigma_3 = \{\gamma_3, \beta_{13}\}$. Suppose that the system starts from state 0 and reaches state 8 via the sequence of events $t = \upsilon_1\alpha_{12}\beta_{13}\gamma_2$. Then, OS O_1 records the sequence $\tau^{(1)} = P_1(t) = \alpha_{12}\beta_{13}$, OS O_2 records the sequence $\tau^{(2)} = P_2(t) = \alpha_{12}\gamma_2$, and OS O_3 records the sequence $\tau^{(3)} = P_3(t) = \beta_{13}$. If, at this time, O_2 initiates a synchronization, then we have $\mathcal{I}^c = \{2\}$.

IV. Current State Estimation under Partially Ordered Observation Sequences

A. DO-based Current-State Estimation

Given a system $G = (X, \Sigma, \delta, X_0)$, suppose that a sequence of events t is executed by the system such that OS O_i preserves the sequence of observations $P_i(t)$. The synchronization information state (SI-state) is defined based on the observation sequences recorded by the m OSs: $SI(t) = (P_1(t), \ldots, P_m(t))$. The SI-state describes the PO-sequences recorded by OSs upon the system behavior between two consecutive synchronizations. Note that the SI-state becomes $(\epsilon, \ldots, \epsilon)$ after each synchronization.

Definition 1: (DO-based Current-State Estimation (DO-CSE)) Given $G = (X, \Sigma, \delta, X_0)$, the coordinator knows G

is initially in the set of states Q. Following a string $t \in \Sigma^*$, which occurs in the system and results in a synchronization, the current-state estimate after the coordinator receives the SI-state $SI(t) = (P_1(t), \ldots, P_m(t))$, is defined as

$$\mathcal{E}^c(SI(t), Q) = \{x \in X | \exists q \in Q, \exists u \in L(G, q), \\ (\forall i \in \mathcal{I})[P_i(u) = P_i(t)] \wedge x \in \delta(q, u)\}.$$

B. Implementation of DO-based Current-State Estimation

In order to simplify the notation, we use T to denote the set of SI-states and $\tau \in T$ to denote $\tau = (\tau^{(1)}, \ldots, \tau^{(m)}) \in \Sigma_1^* \times \cdots \times \Sigma_m^*$ (since the SI-state comprises combinations of sequences of observations).

Definition 2: An SI-state $\tau = (\tau^{(1)}, \ldots, \tau^{(m)})$ is said to be a critical SI-state (CSI-state) if there exists a critical index set $\mathcal{I}_\tau^c \subseteq \mathcal{I}$, such that $|\mathcal{I}_\tau^c| > 0$.

Definition 2 provides the natural state of m OSs at one synchronization step where OSs indexed by $\mathcal{I}_\tau^c$ initiate a synchronization. Therefore, given a CSI-state τ, for the purpose of obtaining a better estimate of states, we have the following properties: A1) $\forall i, j \in \mathcal{I}_\tau^c, \tau_f^{(i)} = \tau_f^{(j)}$; A2) $\forall t \in S(\tau^{(1)}, \ldots, \tau^{(m)}), \forall i \in \mathcal{I}_\tau^c, t_f = \tau_f^{(i)}$.

The first property indicates that sequences of observations of the OSs which initiate a synchronization share the same last symbol. We use $[\mathcal{I}_\tau^c]_f$ to denote this last symbol. The second property indicates that the matching sequences of system observation should end with $[\mathcal{I}_\tau^c]_f$.

Definition 3: (TO-Sequences) Given a system G, a sequence of events t occurs in G such that the SI-state is a CSI-state $\tau = (\tau^{(1)}, \ldots, \tau^{(m)})$. The totally ordered sequences (TO-sequences) upon τ with $\mathcal{I}_\tau^c$, denoted by $\mathcal{TO}(\tau, \mathcal{I}_\tau^c)$, are defined as a subset of sequences of events in $\Sigma_\mathcal{I}^*$, i.e., $\mathcal{TO}(\tau, \mathcal{I}_\tau^c) = \{\omega \in \Sigma_\mathcal{I}^* | \forall i \in \mathcal{I}, P_i(\omega) = \tau^{(i)} \wedge \omega_f = [\mathcal{I}_\tau^c]_f]\}$.

Intuitively, $\mathcal{TO}(\tau, \mathcal{I}_\tau^c)$ summarizes each OS's knowledge about the order of its own observable events within the (unknown) system behavior t. It is obvious that $\mathcal{TO}(\tau, \mathcal{I}_\tau^c)$ is non-empty since it necessarily contains the sequence $P_\mathcal{I}(t)$. Under these circumstances, we have the following result.

Lemma 1: Consider a system G that is known to be in a set of possible states Q. Also, consider that a sequence of events t occurs in the system such that the CSI-state is $\tau = (\tau^{(1)}, \ldots, \tau^{(m)})$ and the set of critical indices is $\mathcal{I}_\tau^c$. The DO-CSE of the coordinator after receiving τ is given by

$$\mathcal{E}^c(\tau, \mathcal{I}_\tau^c, Q) = \{x \in X | \exists q \in Q, \exists u \in L(G, q), \\ P_\mathcal{I}(u) \in \mathcal{TO}(\tau, \mathcal{I}_\tau^c) \wedge x \in \delta(q, u)\}.$$

At first sight, we need to enumerate all possible sequences of observations that match the given PO-sequences.[1] This could be done by re-ordering the events occurring in the CSI-state into one sequence. Thus, the key to dealing with this question is how to arrange the events that could be observed by multiple OSs. In a DO-based protocol, each synchronization could be viewed as the computation of TO-sequences and DO-CSE. Next, we define the notion of sequence-builder (*S-builder*) which provides (but does not enumerate) all possible TO-sequences given an SI-state τ.

Definition 4: (S-builder) Given a CSI-state τ_0 with $\mathcal{I}_{\tau_0}^c$ being the set of critical indices, an S-builder is a 5-tuple transition system $B = (T, \Sigma_s, T_0, T_e, h)$ where

- $T \subseteq \Sigma_1^* \times \cdots \times \Sigma_m^*$ is the set of SI-states;
- $\Sigma_s = \{\sigma | \exists i \in \mathcal{I}, \sigma \in \tau_0^{(i)}\} \subseteq \Sigma_\mathcal{I}$ is the set of events appearing in the initial state T_0;
- $T_0 \in T$ is the initial state where $T_0 = \tau_0$;
- $T_e \in T$ is the ending state where $T_e = (\underbrace{\epsilon, \ldots, \epsilon}_{m \text{ times}})$;
- $h : T \times \Sigma_s \to T$ is the event release transition function, which is defined as follows: for any $\tau = (\tau^{(1)}, \ldots, \tau^{(m)}) \in T$, $\tau' = (\tau'^{(1)}, \ldots, \tau'^{(m)}) \in T$, and $\sigma \in \Sigma_s$, it holds

$$h(\tau, \sigma) = \tau' \Rightarrow (\forall i \in I(\sigma))(\forall j \in \mathcal{I}/I(\sigma)) \\ [\tau^{(i)} \in \Omega_i(\sigma) \wedge \tau'^{(i)} = \tau^{(i)}/\sigma \wedge \tau'^{(j)} = \tau^{(j)} \\ \wedge (I(\sigma) = \mathcal{I}_{\tau_0}^c \Rightarrow (\tau'^{(i)} = \epsilon \Rightarrow \tau^{(j)} = \epsilon))]$$

Any path from the initial state to the ending state in the S-builder describes a potential TO-sequence which may form the SI-state τ_0. In order to infer the total orders of events occurring in SI-state τ_0, transition function h builds the state space of the S-builder recursively by releasing the events in τ_0. This could be viewed as the reverse process of OSs capturing and preserving their observations. Therefore, from the initial state τ_0, the construction of a transition σ from τ to τ' is called the procedure of releasing σ, i.e., $h(\tau, \sigma) = \tau'$. The event σ can be released only if σ appears in the leftmost of all components provided by the OSs whose sets of observable events contain σ (described by the proposition $(\forall i \in I(\sigma))[\tau^{(i)} \in \Omega_i(\sigma)]$), and τ' will be the remaining part of τ after removing the σ in these leftmost positions (described by the proposition $(\forall i \in I(\sigma))(\forall j \in \mathcal{I}/I(\sigma))[\tau'^{(i)} = \tau^{(i)}/\sigma \wedge \tau'^{(j)} = \tau^{(j)}]$). We use the proposition $(I(\sigma) = \mathcal{I}_{\tau_0}^c \Rightarrow (\tau'^{(i)} = \epsilon \Rightarrow \tau^{(j)} = \epsilon))$ to describe the scenario where OSs indexed by $\mathcal{I}_{\tau_0}^c$ will release their last event only if other sites have already released all their events. One can omit this proposition if the critical index set is not available in some scenario.

Theorem 1: Given a CSI-state $\tau = (\tau^{(1)}, \ldots, \tau^{(m)})$ with critical index set $\mathcal{I}_\tau^c$ and its corresponding S-builder $B_\tau = (T, \Sigma_s, T_0, T_e, h)$ with the marked state T_e, it holds $\mathcal{TO}(\tau, \mathcal{I}_\tau^c) = L_m(B_\tau)$.

Example 2: Consider again the system in Fig. 1. Suppose that since the last time a synchronization occurred, sequence of events $t = \upsilon_1\alpha_{12}\gamma_3\alpha_{12}\upsilon_2\beta_{13}\gamma_2$ occurs and assume that a synchronization is initiated by O_2 such that the coordinator will receive CSI-state $\tau_0 = (P_1(t), P_2(t), P_3(t)) = (\alpha_{12}\ \alpha_{12}\beta_{13}, \alpha_{12}\alpha_{12}\gamma_2, \gamma_3\beta_{13})$ with $\mathcal{I}_{\tau_0}^c = \{2\}$. The S-builder B_{τ_0} of τ_0 is shown in Fig. 2. At initial state τ_0, α_{12} appears in the leftmost of $\tau_0^{(1)}$ and $\tau_0^{(2)}$ provided by O_1 and O_2 which are indexed by $I(\alpha_{12})$ such that α_{12} could be released in τ_0 and $h(\tau_0, \alpha_{12}) = \tau_1$. The same procedure can be easily

[1] Note that, if each OS initiates a synchronization every time it observes a new event, the resulting state estimates will align with that of centralized observation, in which the global observer views $\Sigma_\mathcal{I}$. For example, if event $\sigma \in \Sigma_\mathcal{I}$ occurs, the CSI-state is $\tau = (P_1(\sigma), \ldots, P_m(\sigma))$. Then, it is obvious that $\mathcal{E}^c(\tau, \mathcal{I}_\tau^c, Q) = \{x \in X | \exists q \in Q,\ \exists u \in L(G, q), P_\mathcal{I}(u) = \sigma \wedge x \in \delta(q, u)\}$, which is the result of standard centralized state estimation following each observation.

adapted to obtain the rest of the structure of the S-builder B_{τ_0}. According to B_{τ_0}, we know $\mathcal{TO}(\tau_0) = L_m(B_{\tau_0}) = \{\alpha_{12}\alpha_{12}\gamma_3\beta_{13}\gamma_2, \alpha_{12}\gamma_3\alpha_{12}\beta_{13}\gamma_2, \gamma_3\alpha_{12}\alpha_{12}\beta_{13}\gamma_2\}$. Note that the release of γ_2 in SI-states τ_3 and τ_5 is not possible (denoted by dotted transitions) since the synchronization is applied by O_2, indicating that the last symbol recorded by O_2 should be the last event released, i.e., $h(\tau_6, \gamma_2) = T_e$.

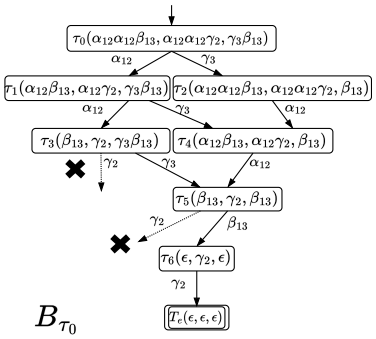

Fig. 2. The S-builder given CSI-state τ_0 with $\mathcal{I}^c_{\tau_0} = \{2\}$.

Remark 1: Consider an S-builder $B_\tau = (T, \Sigma_s, T_0, T_e, h)$ w.r.t. an SI-state τ, where T_e is marked. An intuitive method to perform state estimation is to compute the product of the S-builder and the observer of the original system w.r.t. observable event set $\Sigma_\mathcal{I}$ [17]. However, this approach needs to construct the S-builder and observer beforehand which suffers from high computational complexity. Next, we formally introduce a structure called *synchronizer*, which can be used to estimate the possible states of the system while building the S-builder.

We first introduce some notions which can be used to serve as the "state computation" part and properly prune states in the S-builder in order to reduce the computational complexity. Given system $G = (X, \Sigma, \delta, X_0)$, the state mapping induced by σ is defined as $M(\sigma) = \{(x_1, x_2) \in X \times X | \exists x_1, x_2 \in X, \exists t \in \Sigma^*, P_\mathcal{I}(t) = \sigma \wedge x_2 \in \delta(x_1, t)\}$. The non-standard state estimation function $\delta^N : 2^X \times \Sigma_\mathcal{I} \to 2^X$ is defined as follows: for any $Q \in 2^X$, $\sigma \in \Sigma_\mathcal{I}$, $\delta^N(Q, \sigma) = \{x \in X | \exists q \in Q \cap M(\sigma)[0], (q, x) \in M(\sigma)\}$. We are now ready to define the *synchronizer* by the construction in Algorithm 1, which essentially asynchronously infers the TO-sequences and simultaneously estimates the system states.

Consider a system G (known to be in a set of possible states $Q \subseteq X$). A synchronizer $\mathcal{S}$ w.r.t. a set of initial system states $Q \subseteq X$, and a CSI-state τ_0 with set of critical indices $\mathcal{I}^c_{\tau_0}$ is of the form $\mathcal{S}(\tau_0, \mathcal{I}^c_{\tau_0}, Q) = (T, \Sigma_s, T_0, T_e, h_s, c, C)$ with T_e being the marked state. $\mathcal{S}(\tau_0, \mathcal{I}^c_{\tau_0}, Q)$ describes a process of releasing the events in τ_0 and estimating the states of the system, where the component $(T, \Sigma_s, T_0, T_e, h_s)$ is a sub-structure capturing the S-builder; $c : T \to 2^X$ is the state-estimation function; C simply summarizes this mapping for each state of the S-builder.

The whole structure of the S-builder given τ_0 is not constructed beforehand. Therefore, Algorithm 1 builds the state space of the synchronizer recursively by adding the transition into the structure in a breadth-first search like process. From initial state τ_0, procedure *Release* concentrates on the states with tag "release" and applies the function h with the restriction "$\delta^N(c(\tau), e) \neq \emptyset$" which indicates that e is admissible to match the possible system behavior. After all events matching the restriction are released from τ, τ's tag will be removed and the next-state $h(\tau, e)$ will be assigned a tag. In line 6, we update the set in order to estimate the states and release the events in the next-depth level. In procedure *Estimation*, we use function c to estimate the states under the SI-state τ. It is noteworthy that for an SI-state τ, function c needs to collect all its preceding states and transitions and takes the union of their possible state-estimates. This algorithm will end at state T_e and the final state-estimate of the system is $c(T_e)$.

Algorithm 1 Construction of a synchronizer

Input: State set Q, CSI-state τ_0 with critical index set $\mathcal{I}^c_{\tau_0}$.
Output: $\mathcal{S}(\tau_0, \mathcal{I}^c_{\tau_0}, Q) = (T, \Sigma_s, T_0, T_e, h_s, c, C)$

1: $T_0 = \tau_0$, $T = \{T_0\}$, $\Sigma_s = \{\sigma | \exists i \in \mathcal{I}, \sigma \in \tau_0^{(i)}\}$, $c(\tau_0) = Q$, $C = \{c(\tau_0)\}$, $T_e = (\epsilon, \ldots, \epsilon)$, and $c(T_e) = \emptyset$;
2: assign tag "release" to τ_0;
3: **while** $c(T_e) == \emptyset$ **do**
4: T_{tag} is the set of states with tag "release";
5: *Release*(T_{tag});
6: update the set T_{tag};
7: *Estimation*(T_{tag});
8: **end while**
9: **procedure** *Release*(T_1)
10: **for** $\tau \in T_1$ **do**
11: **for** $e \in \{\sigma | \forall i \in \mathcal{I}, \tau^{(i)} \in \Omega_i(\sigma)\}$ **do**
12: **if** $h(\tau, e)! \wedge \delta^N(c(\tau), e) \neq \emptyset$ **then**
13: add transition $(\tau, e, h(\tau, e))$ and $h(\tau, e)$ to h_s and T, respectively;
14: assign tag "release" to $h(\tau, e)$;
15: **end if**
16: **end for**
17: remove tag "release" from τ;
18: **end for**
19: **procedure** *Estimation*(T_1)
20: **for** $\tau \in T_1$ **do**
21: $c(\tau) = \bigcup_{(\tau_1, e, \tau) \in h_s} \delta^N(c(\tau_1), e)$;
22: $C = C \cup \{c(\tau)\}$;
23: **end for**

Theorem 2: Consider a system G that is known to be in a set of possible states Q. Suppose that a sequence of events t occurs in the system such that CSI-state $\tau = (P_1(t), \ldots, P_m(t))$ with critical index set $\mathcal{I}^c_\tau$ is reached, and $\mathcal{S}(\tau, \mathcal{I}^c_\tau, Q) = (T, \Sigma_s, T_0, T_e, h_s, c, C)$ with the marked state T_e is the corresponding synchronizer. We have that $\mathcal{E}^c(\tau, \mathcal{I}^c_\tau, Q) = c(T_e)$.

Example 3: Consider the S-builder presented in Example 2. If the coordinator knows that the system is initially in the set of states $\{0, 2\}$, according to Algorithm 1, the synchronizer $\mathcal{S}(\tau_0, \{2\}, \{0, 2\})$ is shown in Fig. 3 surrounded

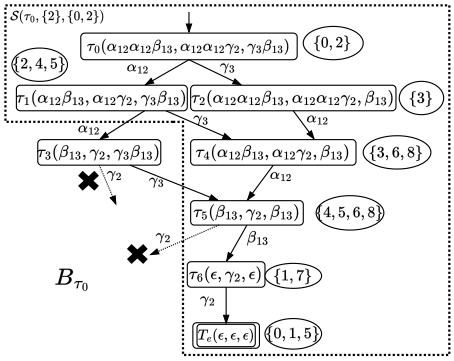

Fig. 3. Synchronizer $\mathcal{S}(\tau_0,\{2\},\{0,2\})$.

by dotted lines. The oval states stand for the state estimation part related to the corresponding SI-states. Note that $c(\tau_4) = \delta^N(c(\tau_1),\gamma_3) \cup \delta^N(c(\tau_2),\alpha_{12})$ indicates that τ_1 and τ_2 are constructed before τ_4. SI-state τ_3 is excluded from the synchronizer since $\delta^N(c(\tau_1),\alpha_{12}) = \emptyset$ in this system. Finally, the coordinator concludes that DO-CSE is $c(T_e) = \{0,1,5\}$.

C. Analysis of Computational Complexity for DO-based Current-state Estimation

While there exist various alternative strategies to address the DO-CSE problem, the method proposed in this paper can reduce the corresponding computational complexity. Given a synchronizer $\mathcal{S}(\tau,\mathcal{I}^c_\tau,Q)$, we use κ_i to denote the length of the observation sequence recorded by O_i, i.e., $|\tau^{(i)}| = \kappa_i$. In the worst scenario, $\mathcal{S}$ contains $\prod_{i=1}^m(\kappa_i+1)$ states and its number of transitions is bounded by $\prod_{i=1}^m(\kappa_i+1)\cdot m$ (a $\tau^{(i)}$ has at most κ_i symbols to be released and an SI-state can release at most m symbols at a time). The complexity of the state estimation part at an SI-state is $O(|X|^2)$. Therefore, the entire complexity for DO-CSE is $O(|X|^2\cdot\prod_{i=1}^m(\kappa_i+1)\cdot m)$.

With the above analysis, we can deduce that performing the current-state estimation at the coordinator has polynomial complexity of $\prod_{i=1}^m(\kappa_i+1)$. However, this bound actually treats the situation where the set of local observable events are disjoint. Next, we use an example to illustrate that this bound could be significantly reduced when there exist events observed by more than one OS (termed as shared-observable events).

Example 4: Consider an NFA observed by two OSs, where $\mathcal{I} = \{1,2\}$. Suppose that the sequence of events t occurs in the system such that $\tau = (P_1(t),P_2(t))$. We consider the following two cases:

(1) $\Sigma_1 \cap \Sigma_2 = \emptyset$ where we assume $\Sigma_1 = \{\alpha,\gamma\}$, $\Sigma_2 = \{\beta,\overline{\gamma}\}$, and $\tau = (\alpha\alpha\gamma\alpha,\beta\overline{\gamma}\beta\beta)$. The corresponding S-builder is shown in Fig. 4(a). There are $(|\alpha\alpha\gamma\alpha|+1)\times(|\beta\overline{\gamma}\beta\beta|+1) = 5\times5 = 25$ SI-states in this case. The release procedure at each SI-state only removes one event since this event is unobservable by the other OS.

(2) $\Sigma_1 \cap \Sigma_2 = \{\gamma\}$ where we assume $\Sigma_1 = \{\alpha,\gamma\}$, $\Sigma_2 = \{\beta,\gamma\}$, and $\tau = (\alpha\alpha\gamma\alpha,\beta\gamma\beta\beta)$. The corresponding S-builder is shown in Fig. 4(c). The event γ divides the release procedures of τ into two parts: the part before γ which is $(\alpha\alpha,\beta)$ and the part after γ which is $(\alpha,\beta\beta)$; the number of corresponding SI-states is $(|\alpha\alpha|+1)\times(|\beta|+1)+(|\alpha|+1)\times(|\beta\beta|+1) = 3\times2+2\times3 = 12$.

If we assume $\gamma \equiv \overline{\gamma}$, 13 SI-states (more than half of the size of the S-builder in Case 1) are reduced, as shown in Fig. 4(b) surrounded by the dotted rectangles; this is due to the existence of shared-observable event γ. From the above example, we see that the bound $\prod_{i=1}^m(\kappa_i+1)$ does not specifically describe the cases when there are shared-observed events, which always break the bound of multiplications of lengths of several strings into the sum of multiplications of lengths of several sub-strings. Therefore, the bound on the size of an S-builder can be reduced depending on the number of shared-observable events and the time instants at which these events occur in an SI-state. This constitutes the primary distinction between the approach presented here and the work in [16]. Actually, in a real system, these situations could be more complex depending on how many events are shared and among which subsets of observation sites.

V. Initial-State Estimation under Partially Ordered Observation Sequences

Definition 5: (DO-based Initial-State Estimation (DO-ISE) Consider $G = (X,\Sigma,\delta,X_0)$. Following a string $t\in\Sigma^*$ which occurs in the system and results in a synchronization step, the initial-state estimate after the coordinator receives the SI-state $\tau = (P_1(t),\ldots,P_m(t))$ is defined as

$$\mathcal{E}^\infty(\tau,X_0) = \{x_0\in X_0 | \exists u\in L(G,x_0), (\forall i\in\mathcal{I})[P_i(u) = P_i(t)]\}.$$

Similar to DO-CSE, DO-ISE depends on the inference of TO-sequences associated with the corresponding CSI-state. Thus, we have the following result.

Lemma 2: Consider a system G and a sequence of events t that occurs in G such that the CSI-state is $\tau = (\tau^{(1)},\ldots,\tau^{(m)})$ and the set of critical indices is $\mathcal{I}^c_\tau$. The DO-ISE of the coordinator after receiving τ is given by

$$\mathcal{E}^\infty(\tau,\mathcal{I}^c_\tau,X_0) = \{x_0\in X_0 | \exists u\in L(G,x_0), P_\mathcal{I}(u)\in\mathcal{TO}(\tau,\mathcal{I}^c_\tau)\}.$$

Next, we review and redefine the notions that will be used to realize DO-ISE. Given system $G = (X,\Sigma,\delta,X_0)$ and $\sigma\in\Sigma_\mathcal{I}$, the DO-ISE transition function $\delta^\infty_{obs}: 2^{X^2}\times\Sigma_\mathcal{I}\to 2^{X^2}$ is defined as: for $m^p\in 2^{X^2}$, $\sigma\in\Sigma_\mathcal{I}$, $\delta^\infty_{obs}(m^p,\sigma) = m^p\circ M(\sigma)$. With these notions, we only need to slightly modify Algorithm 1 to perform DO-ISE.

Algorithm 2 Computation of DO-ISE based on Algorithm 1

Input: Initial-state mappings $X_0^2 = \{(q,q)|q\in X_0\}$, a CSI-state τ_0, and the corresponding critical index set $\mathcal{I}^c_{\tau_0}$

Output: $\mathcal{S}(\tau_0,\mathcal{I}^c_{\tau_0},X_0^2) = (T,\Sigma_s,T_0,T_e,h_s,c^\infty,C)$

1: Run Line 1 to Line 23 of Algorithm 1 where the state transition function $\delta^N(\cdot,\cdot)$ is changed into DO-ISE transition function $\delta^\infty_{obs}(\cdot,\cdot)$ and notation of function $c(\cdot)$ is changed into $c^\infty(\cdot)$ globally;

Proposition 1: Consider a system $G = (X,\Sigma,\delta,X_0)$ and a sequence of events t that occurs in the system such that

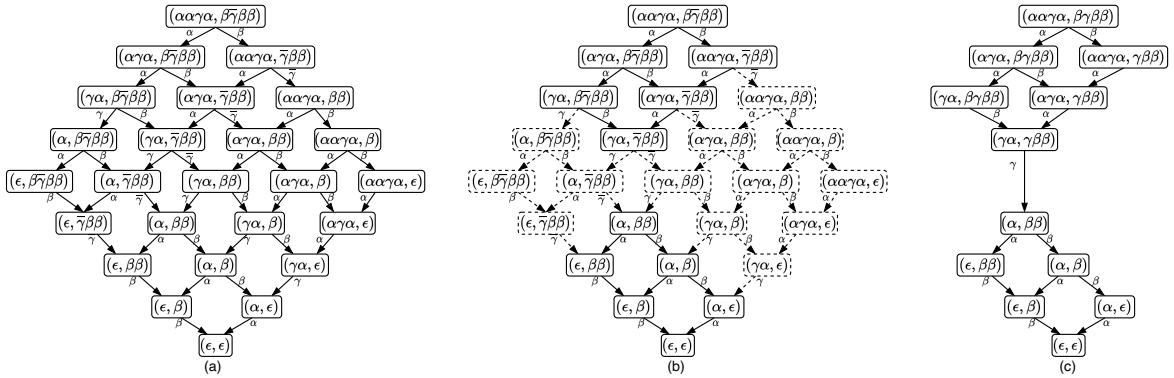

Fig. 4. (a) is the S-builder for Case 1, (c) is the S-builder for Case 2, and (b) shows difference between the S-builders for (a) and (c) when $\gamma \equiv \overline{\gamma}$.

CSI-state $\tau_0 = (P_1(t), \ldots, P_m(t))$ with critical index set $\mathcal{I}^c_{\tau_0}$ is reached. DO-ISE can be estimated based on Algorithm 2, i.e., $\mathcal{E}^\infty(\tau_0, \mathcal{I}^c_{\tau_0}, X_0) = c^\infty(T_e)[0]$.

The complexity analysis of DO-ISE is similar to that of DO-CSE since, given an IS-state, both procedures share the same S-builder. The difference relies on the state estimation part at each state of the S-builder. Each time an event is released in the synchronizer, the complexity of updating state mappings can be bounded by $O(|X|^2)$, i.e., the complexity of the computation of each $\delta^\infty(\cdot,\cdot)$. Therefore, the entire complexity for DO-ISE is $O(|X|^2 \cdot \prod_{i=1}^m (\kappa_i + 1) \cdot m)$.

Remark 2: Similar to the modification of the synchronizer used to estimate the initial states, K-delayed-state estimation could also be realized by using the corresponding K-delayed-state transition function [19] in the state estimation part at each state of the S-builder. We do not present this enhancement here in order to avoid dense notations. It is worth noting that "K" is not required to be smaller than the length of $P_{\mathcal{I}}(t)$ where t is the system behavior during one synchronization since, if we have sequential synchronizations, the delayed-state estimation obtained at the last synchronization will also be input at the next synchronization.

VI. Conclusions

In this paper, we have considered the problem of state estimation in a DES. The notion of S-builder was proposed to construct a synchronizer. Depending on different inputs and state estimation functions, current-state and initial-state estimation can be realized. The complexity of the algorithm relies on the shared-observable events and the lengths of observation sequences provided by OSs, which is related to the synchronization policy. In the future, we plan to incorporate communication delay and time synchronization abnormalities [20], [21] in the decentralized architecture, especially for the state estimation task using an appropriately modified S-builder.

References

[1] P. Ramadge, "Observability of discrete event systems," in *Proceedings of 25th IEEE Conference on Decision and Control (CDC)*, 1986, pp. 1108–1112.

[2] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis, "Diagnosability of discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 40, no. 9, pp. 1555–1575, 1995.

[3] T.-S. Yoo and S. Lafortune, "Polynomial-time verification of diagnosability of partially observed discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 47, no. 9, pp. 1491–1495, 2002.

[4] J. C. Basilio and S. Lafortune, "Robust codiagnosability of discrete event systems," in *Proceedings of American Control Conference (ACC)*, 2009, pp. 2202–2209.

[5] G. S. Viana, M. V. S. Alves and J. C. Basilio, "Codiagnosability of networked discrete event systems with timing structure," *IEEE Transactions on Automatic Control*, vol. 67, no. 8, pp. 3933–3948, 2022.

[6] S. Shu, F. Lin, and H. Ying, "Detectability of discrete event systems," *IEEE Transactions on Automatic Control*, vol. 52, no. 12, pp. 2356–2359, 2007.

[7] X. Yin, Z. Li, and W. Wang, "Trajectory detectability of discrete-event systems," *Systems & Control Letters*, vol. 119, pp. 101–107, 2018.

[8] A. Saboori and C. N. Hadjicostis, "Notions of security and opacity in discrete event systems," in *Proceedings of 46th IEEE Conference on Decision and Control (CDC)*, 2007, pp. 5056–5061.

[9] R. Jacob, J. Lesage, and J. Faure, "Overview of discrete event systems opacity: Models, validation, and quantification," *Annual Reviews in Control*, vol. 41, pp. 135–146, 2016.

[10] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM Journal on Control and Optimization*, vol. 25, no. 1, pp. 206–230, 1987.

[11] R. Cieslak, C. Desclaux, A. S. Fawaz and P. Varaiya, "Supervisory control of discrete-event processes with partial observations," *IEEE Transactions on Automatic Control*, vol. 33, no. 3, pp. 249–260, 1988.

[12] F. Lin, W. Wang, L. Han and B. Shen, "State estimation of multichannel networked discrete event systems," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 1, pp. 53–63, 2020.

[13] X. Han, J. Wang, Z. Li, X. Chen and Z. Chen, "Revisiting state estimation and weak detectability of discrete-event systems," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 1, pp. 662–674, 2023.

[14] C. N. Hadjicostis, *Estimation and Inference in Discrete Event Systems*. Springer, 2020.

[15] R. Debouk, S. Lafortune, and D. Teneketzis, "Coordinated decentralized protocols for failure diagnosis of discrete event systems," *Discrete Event Dynamic Systems*, vol. 10, no. 1, pp. 33–86, 2000.

[16] C. N. Hadjicostis and C. Seatzu, "Decentralized state estimation in discrete event systems under partially ordered observation sequences," in *Proceedings of 13th International Workshop on Discrete Event Systems (WODES)*, 2016, pp. 367–372.

[17] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Springer, 2008.

[18] A. Saboori and C. N. Hadjicostis, "Verification of initial-state opacity in security applications of DES," in *Proceedings of 9th International Workshop on Discrete Event Systems (WODES)*, 2008, pp. 328–333.

[19] A. Saboori and C. N. Hadjicostis, "Verification of K-step opacity and analysis of its complexity," in *Proceedings of 48th IEEE Conference on Decision and Control (CDC) held jointly with 28th Chinese Control Conference (CCC)*, 2009, pp. 205–210.

[20] F. Lin, "Control of networked discrete event systems: Dealing with communication delays and losses," *SIAM Journal on Control and Optimization*, vol. 52, no. 2, 2014, pp. 1276–1298.

[21] C. E. V Nunes, M. V. Moreira, M. V. S. Alves, L. K. Carvalho, and J. C. Basilio, "Codiagnosability of networked discrete event systems subject to communication delays and intermittent loss of observation," *Discrete Event Dynamic Systems*, vol. 28, 2018, pp. 215–246.