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

Data-Informativity for Data-Driven Supervisory Control of Discrete-Event Systems[1]

Tomofumi Ohtsuka[2], Kai Cai[3], *Senior Member, IEEE* and Kenji Kashima[2], *Senior Member, IEEE*

***Abstract*— In this paper we develop a data-driven approach for supervisory control of discrete-event systems (DES). We consider a setup in which models of DES to be controlled are unknown, but a set of data concerning the behaviors of DES is available. We propose a new concept of *data-informativity*, which captures the notion that the available data set contains sufficient information such that a valid supervisor may be constructed for a family of DES models that all can generate the data set. We then characterize data-informativity with a necessary and sufficient condition, based on which we design an algorithm for its verification.**

I. Introduction

Supervisory control of discrete-event systems (DES) is a relatively new area of control science and engineering [1]. A DES is a dynamical system that is discrete in time and usually in state space, and its dynamics is driven by instantaneous occurrences of events. To enforce a desired specification on a given DES, supervisory control theory aims to construct a feedback controller called a *supervisor*, whose control mechanism is of disabling or enabling occurrences of certain (controllable) events. Supervisory control is a *model-based* approach: a DES to be controlled is first modeled as a *finite-state automaton*, and its behaviors represented by *regular languages*; then supervisory control design is carried out based on these models. Recently, DES has been combined with continuous dynamical systems in areas called hybrid or cyber-physical systems [2], [3].

In the past few years, various data-driven techniques have been successfully utilized to analyze and synthesize controllers for continuous dynamical systems. Representative approaches include identification of dynamical models from data [4], learning control laws directly from observations [5] or through reinforcement learning [6]. This research thrust has been driven by the motivation to tackle challenging control problems involving unknown system dynamics, high nonlinearity, huge dimensionality, and on the other hand the increasing availability of large quantity of observation data.

The same thrust has so far been, however, obscure in supervisory control of DES. Although attempts to apply data-driven techniques exist [7], [8], [9], such works are scarce and not yet systematic. On the other hand, DES control problems are facing similar challenges like unknown system dynamics and/or high dimensionality, as well as the opportunity of exploding amount of observation data thanks to fast advancing data collection capabilities. Hence in this paper, we aim to initiate a systematic development of a *data-driven* approach for supervisory control of DES.

Specifically, we consider a setup in which automaton models of DES to be controlled are unknown, but a set of data concerning the behaviors of DES is available. We ask the question: *Under what conditions of the available data set can a valid supervisor be designed for the unknown DES to satisfy a given specification?*

Intuitively, the key to answering this question is the 'quality' of the data set for the purpose of supervisory control. For this, we identify and formalize a novel concept called *data-informativity*. Data-informativity characterizes a condition that the given data set contains sufficient information such that a valid supervisor may be constructed for *a family of DES models that all can generate the data set*. Thus rather than trying to first identify a model for the unknown DES, our approach based on data-informativity aims to directly construct from the data set a supervisor valid for all possible models undistinguishable from the unknown DES. If such a supervisor can be constructed, it is also valid for the unknown DES.

This idea of data-informativity has been introduced for data-driven control of linear systems [5], and recently attracted much attention e.g., [10], [11]. Although conceptually similar, due to the discrete, event-driven dynamics of DES and distinct supervisory control mechanisms, the data-informativity based approach we develop uses different settings, mathematical tools, and synthesis methods from those for continuous time-driven systems. A particular feature unique to DES is the phenomenon that *it is not always true that more data is better for supervisory control*. In other words, a large quantity of data without needed quality is not useful for data-driven supervisory control of DES. This feature may first seem counter-intuitive, but will become reasonable as explained in Section III below.

The main content and contributions are summarized below.

- First, we propose a new concept for data-driven supervisory control: *data-informativity*. The given data set about the behaviors of the unknown plant consists of two types. The first type is *observation data* D, which is a collection of observed behaviors (strings of events) from the unknown DES. The second type is *prior knowledge data* D^- about the impossible behaviors of the DES. Then we define data-informativity of the pair (D, D^-)

[1] This work was supported in part by JSPS KAKENHI Grant nos. 21H04875, 22KK0155.

[2]T. Ohtsuka and K. Kashima are with the Graduate School of Informatics, Kyoto University, Kyoto, Japan
otsuka.tomofumi.78z@st.kyoto-u.ac.jp;
kk@i.kyoto-u.ac.jp

[3]K. Cai is with the Department of Core Informatics, Osaka Metropolitan University, Osaka, Japan cai@omu.ac.jp

in terms of (language) controllability essential for the existence of valid supervisors.
- Second, we characterize the concept of data-informativity by establishing a necessary and sufficient condition for it. Based on this condition, we present an algorithm for the verification of data-informativity. The novelty of this verification algorithm is the construction of a special *data-driven automaton*. If the data set (D, D^-) is informative (for a given specification), we construct a valid supervisor for the unknown plant to satisfy the specification.

The remainder of this paper is organized as follows. Section II provides preliminaries on model-based supervisory control of DES. Section III introduces data-informativity, presenting a necessary and sufficient condition and a verification algorithm for the property. Section IV states our concludings.

II. Preliminaries on model-based supervisory control theory

In supervisory control of DES, the plant to be controlled is modeled by a finite-state automaton[1]

$$G = (Q, \Sigma, \delta, q_0). \tag{1}$$

Here Q is the finite state set, Σ the finite event set, $\delta : Q \times \Sigma \to Q$ the (partial) state transition function, $q_0 \in Q$ the initial state. Write $\delta(q, \sigma)!$ to mean $\sigma \in \Sigma$ is defined at state $q \in Q$, and write $\neg\delta(q, \sigma)!$ to mean $\sigma \in \Sigma$ is not defined at state $q \in Q$. We say that the automaton G is *deterministic* if

$$(\forall q \in Q, \forall \sigma \in \Sigma) \quad \delta(q, \sigma)! \implies |\delta(q, \sigma)| = 1.$$

Namely, the destination state of every state transition is unique. We shall focus exclusively on deterministic automata unless otherwise stated. A string on Σ is a sequence of events from Σ. Write Σ^* for the set of all finite-length strings on Σ, including the empty string ϵ (containing no event). Then, the state transition function may be inductively defined, and we write $\delta(q, s)!$ to mean that string $s \in \Sigma^*$ is defined at state $q \in Q$. Any subset $K \subseteq \Sigma^*$ is called a language. Write $\overline{K} := \{s \in \Sigma^* \mid (\exists s' \in \Sigma^*) ss' \in K\}$ for the set of all *prefix* strings of those in K. We call $\overline{K}$ the *prefix closure* of K, and in general $K \subseteq \overline{K}$ holds. The closed behavior $L(G)$ of G is the language defined as the set of all strings of Σ^* which G can generate starting from the initial state q_0:

$$L(G) := \{s \in \Sigma^* \mid \delta(q_0, s)!\}. \tag{2}$$

By definition, we have $\overline{L(G)} = L(G)$.

Generally, not all strings in the closed behavior $L(G)$ of the plant G are desired. Thus we represent a desired behavior to be enforced on G as a control specification $K \subseteq L(G)$. Note that a more general control specification $E \subseteq \Sigma^*$ may be considered. In this case, set $K := E \cap L(G)$ and we again have a specification $K \subseteq L(G)$ to be enforced on G.

For a mechanism to enforce a specification $K \subseteq L(G)$ on the plant G, we assume that a subset of events $\Sigma_c \subseteq \Sigma$, called the controllable events, are capable of being enabled or disabled by an external controller. On the contrary, $\Sigma_u := \Sigma \setminus \Sigma_c$ is the set of uncontrollable events, which cannot be externally disabled and must be considered permanently enabled.

Under the above mechanism, define a supervisor to be a function $V : L(G) \to Pwr(\Sigma_c)$. Here $Pwr(\Sigma_c)$ denotes the set of all subsets of controllable events. Thus a supervisor assigns to each string $s \in L(G)$ generated by the plant G a subset of controllable events $V(s) \subseteq \Sigma_c$ to be disabled. Write V/G for the closed-loop system: "G is under the control of V". The language of closed-loop system $L(V/G)$ is defined as follows:

(i) $\epsilon \in L(V/G)$;
(ii) if $s \in L(V/G)$ and $\sigma \in \Sigma \setminus V(s)$ and $s\sigma \in L(G)$, then $s\sigma \in L(V/G)$;
(iii) no other strings belong to $L(V/G)$.

Thus $L(V/G)$ contains those strings in $L(G)$ that are not disabled by the supervisor V. By definition $L(V/G) \subseteq L(G)$ and $\overline{L(V/G)} = L(V/G)$.

Definition 1 (controllability)**.** *Given a plant G, a control specification $K \subseteq L(G)$ is said to be controllable with respect to G provided*

$$(\forall s \in \overline{K},\ \forall \sigma \in \Sigma_u) \quad s\sigma \in L(G) \implies s\sigma \in \overline{K}. \tag{3}$$

In words, a specification K is controllable wrt. G if and only if any string in the prefix closure $\overline{K}$ cannot exit $\overline{K}$ on a continuation by an uncontrollable event. Namely, the prefix closure of K is invariant under uncontrollable flows. Equivalently (3) can be written compactly as $\overline{K}\Sigma_u \cap L(G) \subseteq \overline{K}$. It is known that the specification language $K(\neq \emptyset)$ being controllable is necessary and sufficient for the existence of a supervisor V such that $L(V/G) = \overline{K}$ [9].

Suppose that $K \subseteq L(G)$ is controllable. Then the supervisor $V : L(G) \to Pwr(\Sigma_c)$ such that $L(V/G) = \overline{K}$ is constructed as follows:

$$V(s) = \begin{cases} \{\sigma \in \Sigma_c \mid s\sigma \notin \overline{K}\} & \text{if } s \in \overline{K}, \\ \emptyset & \text{if } s \in L(G) \setminus \overline{K}. \end{cases}$$

Whether or not K is controllable, we can write $C(K)$ for the family of all controllable sublanguages of K:

$$C(K) := \{K' \subseteq K \mid \overline{K'}\Sigma_u \cap L(G) \subseteq \overline{K'}\}. \tag{4}$$

It is known that the union of two controllable sublanguages of K is still a controllable sublanguage of K. This means that $C(K)$ is closed under set union, so $C(K)$ contains a unique supremal element

$$\sup C(K) := \cup\{K' \mid K' \in C(K)\}. \tag{5}$$

Since $\sup C(K)$ is controllable, as long as $\sup C(K) \neq \emptyset$, there exists a supervisor $V_{\sup}$ such that $L(V_{\sup}/G) =$

[1]As a first step towards developing data-driven supervisory control in this paper, we assume for simplicity that every state is marked and thus the marker state set is omitted.

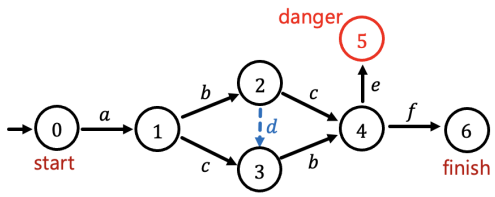

Fig. 1 robot navigation: plant G_1

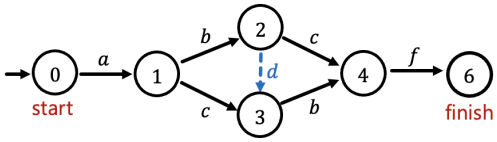

Fig. 2 robot navigation: specification $K_1 (\subseteq L(G_1))$

$\overline{\sup C(K)}$. In this sense $V_{\sup}$ is optimal (maximally permissive), allowing the generation by G of the largest possible set of marked strings that satisfies a given specification.

Example 1. *For illustration, we provide a running example of robot navigation. Consider a robot that moves from a starting point to a finishing point. There exist some paths into a dangerous zone along the route and the robot must avoid them while heading for the goal. Also, the robot may move uncontrollably at some location due to disturbance from the environment. An automaton modeling this described scenario is displayed in Fig. 1, and we consider this automaton (say G_1) as the plant to be controlled.*

In this plant, each state written in numbers represents a location of the environment where the robot navigates. Here state 0 is the starting point and state 6 is the finishing point. Each event written in alphabet represents the transition of the robot. We suppose that event d (dashed arrow from state 2 to 3) represents an uncontrollable transition, and other events are all controllable: i.e. $\Sigma_c = \{a, b, c, e, f\}$ and $\Sigma_u = \{d\}$. State 5 represents a danger state, and the control specification is to avoid this danger state. This (safety) specification may be written as a sublanguage of $L(G_1)$ as follows:

$$K_1 = \{abcf, abdbf, acbf\}.$$

This specification K_1 (indeed its prefix closure $\overline{K_1}$) may be represented by the automaton shown in Fig. 2. Compared with the plant in Fig. 1, the specification automaton removes the (controllable) transition from state 4 to the danger state 5.

Since no uncontrollable event can exit $\overline{K_1}$, the specification language K_1 is controllable. Thus we can construct the following supervisor $V_1 : L(G_1) \to Pwr(\Sigma_c)$ such that $L(V_1/G_1) = \overline{K_1}$:

$$V_1(s) = \begin{cases} \{e\} & \text{if } s \in \{abc, acb, abdb\}, \\ \emptyset & \text{if } s \in L(G_1) \setminus \{abc, acb, abdb\}. \end{cases}$$

This supervisor disables e at state 4 in Fig. 1.

In the above example, the supervisor V_1 is designed based on the assumption that the plant model G_1 is known. Now we pose this question: if G_1 is unknown, under what conditions can we still design a supervisor? This question motivates us to study a data-driven approach to supervisory control.

III. Data-Driven Supervisory Control and Data-Informativity

A. Problem formulation of data-driven supervisory control

Suppose that we have a plant whose automaton model G is unknown except for the event set $\Sigma (= \Sigma_c \cup \Sigma_u)$. Even under this circumstance, there are often situations where plant output sequences data are available. Also, from prior knowledge of the event set, it is often the case that there are certain output sequences that are obviously not generatable by the plant. For example, suppose we have a set of events: turn on a machine, press a switch on the machine, and the machine produces an output. Then, without knowing internal working mechanism of the machine, it is obvious that the machine cannot output anything before its power being turned on. In view of this, we assume that we can obtain a pair of finite data sets (D, D^-), where $D \subseteq \Sigma^*$ is the observed behavior from the plant and $D^- \subseteq \Sigma^*$ is prior knowledge of impossible behavior of the plant. Since each string in D is observed from G, $\overline{D}$ is a subset of the closed behavior of G: i.e. $\overline{D} \subseteq L(G)$. On the contrary since each string in D^- is known to be impossible to be generated by G, D^- and the closed behavior of G do not have any common elements: i.e. $D^- \cap L(G) = \emptyset$. As a result, $\overline{D} \cap D^- = \emptyset$. Given a control specification $E \subseteq \Sigma^*$, our goal is to design a supervisor to enforce E for the unknown plant based on the data pair (D, D^-).

Example 2. *Consider again the robot navigation example in Example 1 and the plant model G_1 in Fig. 1. Now we suppose that G_1 is unknown except for the event set $\Sigma = \{a, b, c, d, e, f\}$, and certain observations and prior knowledge of the plant behavior are available. For example, we have observed a string $abdbf$ from the plant, i.e. the robot moves from the initial state to the goal state following the path $abdbf$. Hence we have $D = \{abdbf\}$. In addition, we have prior knowledge that the plant cannot generate string c, namely the robot can never start its navigation from location 1 and makes a first move to location 3. Thus $D^- = \{c\}$. For this pair (D, D^-), there may exist infinitely many automata that can generate D and cannot generate D^-. In other words, our unknown plant G_1 cannot be uniquely identified based on the pair (D, D^-). For example, G_2 in Fig. 3 and G_3 in Fig. 4 cannot be distinguished from the real plant G_1. In order to design a supervisor for the real plant based only on (D, D^-),* we must construct a supervisor that is valid for all such possible plants. *Intuitively, more observations and prior knowledge can help reduce the number of plant models that cannot be distinguished from the real one. Say if we observe an additional string $acbf$*

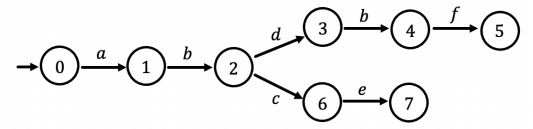

Fig. 3 G_2 in Example 2

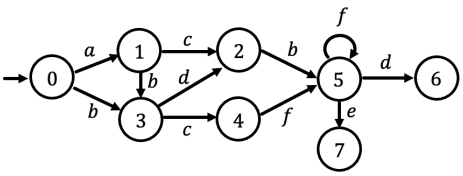

Fig. 4 G_3 in Example 2

(so that $D = \{acbf, abdbf\}$), then G_2 can be ruled out from the candidate while G_3 is still possible.

As in the example above, there are generally multiple possible plants compatible with the given data pair (D, D^-). This is defined below as a consistency property.

Definition 2 (consistency)**.** *Suppose that an event set Σ is given. Then, for finite sets $D, D^- \subseteq \Sigma^*$ satisfying $\overline{D} \cap D^- = \emptyset$, an automaton $G = (Q, \Sigma, \delta, q_0)$ is said to be consistent with (D, D^-) if $\overline{D} \subseteq L(G)$ and $D^- \cap L(G) = \emptyset$.*

In words, an automaton G is consistent with a pair (D, D^-) if and only if all strings in $\overline{D}$ can be generated by G, whereas no strings in D^- can be generated by G. Thus in Example 2, G_1, G_2 and G_3 are consistent with (D, D^-) where $D = \{abdbf\}$ and $D^- = \{c\}$. If we observe an additional string $acbf$ ($D = \{abdbf, acbf\}$), G_1 and G_3 are still consistent with (D, D^-), but G_2 becomes not consistent. It is easy to verify that if we observe some additional strings or we have more knowledge about the strings that the plant cannot generate, the number of consistent models decreases.

Remark 1. *It is well known [12] that for every regular language $L \subseteq \Sigma^*$, there exists an automaton G such that $L(G) = \overline{L}$. Thus for any finite sets $D, D^- \subseteq \Sigma^*$ satisfying $\overline{D} \cap D^- = \emptyset$, one can always find an automaton G such that $L(G) = \overline{D}$, and hence $L(G) \cap D^- = \emptyset$ which leads to that there exists at least one automaton G consistent with (D, D^-).*

Before we proceed, we summarize some basic properties of consistent plants.

Proposition 1. *There exists a consistent plant with (D, D^-) if and only if $\overline{D} \cap D^- = \emptyset$. If G is consistent with (D, D^-), then G is also consistent with $(\overline{D}, D^-)$. Moreover it holds that*

$$\bigcap \{L(G) \mid G \text{ is consistent with } (D, D^-)\} = \overline{D}. \quad (6)$$

The proof of Proposition 1 follows immediately from Definition 2 and Remark 1.

Now we formulate our data-driven supervisory control problem. The last claim (6) of Proposition 1 motivates us to investigate the supervisory control over $\overline{D}$. We denote a control specification based on D by

$$K_D := \overline{D} \cap E, \quad E \subseteq \Sigma^* \text{ (regular language)}. \quad (7)$$

Problem 1. *Suppose that we are given an event set $\Sigma = \Sigma_c \cup \Sigma_u$, a control specification $E \subseteq \Sigma^*$, and finite data sets $D, D^- \subseteq \Sigma^*$ such that K_D in (7) is nonempty and $\overline{D} \cap D^- = \emptyset$. Construct (if possible) a supervisor $V_D : \overline{D} \to Pwr(\Sigma_c)$ such that $L(V_D/G) = \overline{K_D}$ for every plant G consistent with (D, D^-).*

Since the real plant is consistent with (D, D^-), the supervisor satisfying the required condition in Problem 1 is valid for the real plant. If the real plant G was known, we would construct a supervisor to enforce $K = L(G) \cap E$ (whenever K is controllable). In our data-driven approach, $\overline{D}$ represents the maximally accessible subset of $L(G)$ based on our observation of the plant; hence constructing a supervisor to enforce $K_D = \overline{D} \cap E$ is the most that can be done based on the data available. If the behavior represented by K_D is too small/restrictive, one can consider enlarging $\overline{D}$ by observing more behaviors of the plant. The closer $\overline{D}$ approximates $L(G)$, the closer the data-driven enforcible behavior K_D approximates the original model-based behavior $L(G) \cap E$.

B. Data-informativity and its criterion

As we mentioned in Section II, the existence of a supervisor V_D such that $L(V_D/G) = \overline{K_D}$ for all plants G consistent with (D, D^-) is equivalent to the controllability of specification language $K_D (\neq \emptyset)$. This implies that whether the available data has sufficient information can be characterized in terms of controllability.

Definition 3 (informativity)**.** *We say that (D, D^-) is informative for a given control specification E if there exists a supervisor satisfying the required condition in Problem 1, or equivalently if K_D in (7) is nonempty and controllable with respect to all plants G consistent with (D, D^-).*

Recall that, for a known plant, if an uncontrollable event $\sigma \in \Sigma_u$ can happen after $s \in \overline{K_D}$ in the plant, then $s\sigma$ needs to remain in $\overline{K_D}$ for the controllability of $\overline{K_D}$ with respect to the plant; see (3). On the contrary, for the data-driven case (without knowledge of plant), *we need to assume any uncontrollable event $\sigma \in \Sigma_u$ can happen after $s \in \overline{K_D}$ unless $s\sigma \in D^-$*. This observation leads to the following:

Theorem 1 (Criterion for informativity)**.** *Suppose that an event set $\Sigma = \Sigma_c \cup \Sigma_u$ and a control specification $E \subseteq \Sigma^*$ are given. (D, D^-) is informative for E if and only if*

$$\left(\forall s \in \overline{K_D},\ \forall \sigma \in \Sigma_u\right) \quad s\sigma \in \overline{K_D} \cup D^- \quad (8)$$

holds with K_D in (7).

Proof: (If) Suppose that (8) holds. Then it is easy to verify that

$$\left(\forall s \in \overline{K_D}, \forall \sigma \in \Sigma_u\right) \quad s\sigma \in L(G) \implies s\sigma \in \overline{K_D} \quad (9)$$

holds for every plant G consistent with (D, D^-). Thus (D, D^-) is informative for E.

(Only if) Suppose that (D, D^-) is informative for E. This means by Definition 3 that (9) holds for every plant G consistent with (D, D^-). Consider a special such plant G' such that $L(G') = \Sigma^* \setminus D^-$. This G' is consistent with (D, D^-), and any string not in $L(G')$ belongs to D^-. Let $s \in \overline{K_D}$ and $\sigma \in \Sigma_u$. Consider two cases. Case 1: $s\sigma \in L(G')$. Since (9) holds for G', we have $s\sigma \in \overline{K_D}$. Case 2: $s\sigma \notin L(G')$. In this case, we have $s\sigma \in D^-$. Thus (8) is satisfied. □

Theorem 1 provides a necessary and sufficient condition for data-informativity. If (D, D^-) is informative for a given specification E, then a supervisor $V_D : \overline{D} \to Pwr(\Sigma_c)$ such that $L(V_D/G) = \overline{K_D}$ (i.e. a solution to Problem 1) is constructed as follows:

$$V_D(s) = \begin{cases} \{\sigma \in \Sigma_c \mid s\sigma \notin \overline{K_D}\} & \text{if } s \in \overline{K_D}, \\ \emptyset & \text{if } s \in \overline{D} \setminus \overline{K_D}. \end{cases}$$

Example 3. *Let us illustrate the concept of data-informativity using the robot navigation example. Again we suppose that the plant G_1 in Fig. 1 is unknown except for the event set $\Sigma = \Sigma_c \cup \Sigma_u$, where $\Sigma_c = \{a, b, c, e, f\}$ and $\Sigma_u = \{d\}$. Consider two pairs of finite data sets (D_1, D_1^-) and (D_2, D_2^-), where*

$$D_1 = \{abdbf, abce\},$$
$$D_1^- = \{d, ad, abdd, abdbd, abdbfd\};$$
$$D_2 = \{abcf, abdbe\},$$
$$D_2^- = \{d, ad, abcd\}.$$

Let $E = \{abcf, abdbf, acbf, abdf\}$. Then the control specifications in (7) are respectively

$$K_{D_1} = \overline{D_1} \cap E = \{abdbf\},$$
$$K_{D_2} = \overline{D_2} \cap E = \{abcf\}.$$

First consider (D_1, D_1^-). Note that G_1 in Fig. 1 and G_2 in Fig. 3 are consistent with (D_1, D_1^-). From the control specification $K_{D_1} = \{abdbf\}$, we have $\overline{K_{D_1}} = \{\epsilon, a, ab, abd, abdb, abdbf\}$ and $\overline{K_{D_1}}\Sigma_u = \{d, ad, abd, abdd, abdbd, abdbfd\}$. Since only abd belongs to $\overline{K_{D_1}}$ and other strings in $\overline{K_{D_1}}\Sigma_u$ belong to D_1^-, the condition (8) holds and (D_1, D_1^-) is informative for E. Indeed, we can confirm the controllability of K_{D_1} with respect to the consistent plants G_1 and G_2. Correspondingly a supervisor $V_{D_1} : \overline{D_1} \to Pwr(\Sigma_c)$ such that $L(V_{D_1}/G) = \overline{K_{D_1}}$ is constructed for every plant G consistent with (D_1, D_1^-) as follows:

$$V_{D_1}(s) = \begin{cases} \{b, c, e, f\} & \text{if } s = \epsilon, \\ \{a, c, e, f\} & \text{if } s = a, \\ \{a, b, c, e, f\} & \text{if } s = ab, \\ \{a, c, e, f\} & \text{if } s = abd, \\ \{a, b, c, e\} & \text{if } s = abdb, \\ \{a, b, c, e, f\} & \text{if } s = abdbf, \\ \emptyset & \text{if } s \in \overline{D_1} \setminus \overline{K_{D_1}}. \end{cases}$$

Next, we consider (D_2, D_2^-). Note that G_1 in Fig. 1 and G_3 in Fig. 4 are consistent with (D_2, D_2^-). From the control specification $K_{D_2} = \{abcf\}$, we have $\overline{K_{D_2}} = \{\epsilon, a, ab, abc, abcf\}$ and $\overline{K_{D_2}}\Sigma_u = \{d, ad, abd, abcd, abcfd\}$. Since $abd, abcfd \notin \overline{K_{D_2}} \cup D_2^-$, (D_2, D_2^-) is not informative for E. The string abd is in $\overline{D_2} \setminus \overline{K_{D_2}}$, so abd is the string which is outside of the specification and generatable by every consistent plant G. On the contrary, we do not have the information of the string $abcfd$, so it is generatable by some consistent plant G and not generatable by others. For example, G_1 in Fig. 1 cannot generate $abcfd$ but G_3 in Fig. 4 can generate it.

Based on the condition in Theorem 1, we next present an algorithm for checking data-informativity. For this purpose, we first define a *data-driven automaton*. We denote by q_s a state reached by a string s from the initial state of the automaton.

Definition 4 (data-driven automaton)**.** *Suppose that the event set Σ and finite data sets $D, D^- \subseteq \Sigma^*$ (satisfying $\overline{D} \cap D^- = \emptyset$) are given. Then a* data-driven automaton *is defined as follows:*

$$\hat{G}(\Sigma, D, D^-) = (\hat{Q}, \Sigma, \hat{\delta}, \hat{q}_\epsilon), \tag{10}$$

where $\hat{Q} := \{\hat{q}_s | s \in \overline{D \cup D^-}\}$ is the state set, $\hat{\delta} := \{(\hat{q}_s, \sigma) \to \hat{q}_{s\sigma} | s \in \overline{D \cup D^-}, \sigma \in \Sigma, s\sigma \in \overline{D \cup D^-}\}$ is the (partial) state transition function. In addition, given a control specification $K_D = \overline{D} \cap E$ (where $E \subseteq \Sigma^*$ is a regular language), we define $Q_D := \{\hat{\delta}(\hat{q}_\epsilon, s) \mid s \in \overline{K_D}\}$ and $Q_- := \{\hat{\delta}(\hat{q}_\epsilon, s) \mid s \in D^-\}$.

A data-driven automaton $\hat{G}$ is a *prefix tree automaton* for $\overline{D \cup D^-}$: i.e. a loop-less automaton whose closed behavior is $L(\hat{G}) = \overline{D \cup D^-}$. The set Q_D contains those states reached by strings in $\overline{K_D}$. Note that since E may not be a finite language in general, in order to determine Q_D, we need to first construct an automaton for E (always possible since E is regular) and then check if each string in the finite $\overline{D}$ can occur in the automaton for E. On the other hand, the set Q_- contains those states reached by strings in D^-, so a transition to Q_- represents an impossible behavior of the (unknown) plant. Since $\overline{D} \cap D^- = \emptyset$, we have $Q_D \cap Q_- = \emptyset$. It should be remarked that in general $Q_D \cup Q_- \neq \hat{Q}$. Also note that $\hat{G}$ is by no means consistent with (D, D^-), since $\hat{G}$ generates the strings in D^- which is the set of impossible behavior: i.e. $D^- \subseteq L(\hat{G})$.

Example 4. *Here we provide examples of data-driven automata $\hat{G}_1$ (in Fig. 5) and $\hat{G}_2$ (in Fig. 6) corresponding to the data sets (D_1, D_1^-) and (D_2, D_2^-) in Example 3. For clear display, we have omitted $\hat{q}_s$ in the figure and only the subscript s is written inside each state. State sets Q_{D_i} $(i = 1, 2)$ and Q_- are represented in orange and blue in the figures, respectively.*

Now we are ready to present the algorithm for verifying informativity based on data-driven automaton.

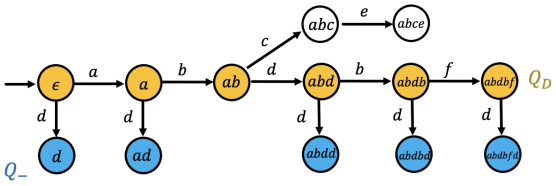

Fig. 5 Data-driven automaton $\hat{G}_1$ corresponding to (D_1, D_1^-)

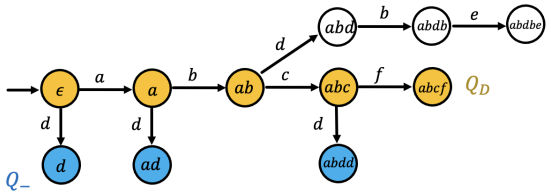

Fig. 6 Data-driven automaton $\hat{G}_2$ corresponding to (D_2, D_2^-)

Algorithm 1 checking informativity

Input: event set $\Sigma = \Sigma_c \cup \Sigma_u$, finite sets $D, D^-(\subseteq \Sigma^*)$, control specification $K_D = \overline{D} \cap E$

Ensure: "informative" or "not informative"

```
construct a data-driven automaton Ĝ(Σ, D, D⁻) = (Q̂, Σ, δ̂, q̂_ϵ) and Q_D, Q_− (as in Definition 4)
for all q̂ ∈ Q_D do
    for all σ ∈ Σ_u do
        if δ̂(q̂, σ) ∈ Q̂ \ (Q_D ∪ Q_−) or ¬δ̂(q̂, σ)! then
            return "not informative"
            break
        end if
    end for
end for
return "informative"
```

In Algorithm 1, informativity of (D, D^-) for K_D is determined by examining in the data-driven automaton every uncontrollable event at each state in Q_D. If an uncontrollable event σ can occur at state $\hat{q} \in Q_D$ and the corresponding transition enters $\hat{Q} \setminus (Q_D \cup Q_-)$, then the transition is contained in $L(G)$ (for all plants G consistent with (D, D^-)) but not contained in $\overline{K_D}$, which means that there exists a string in $\overline{K_D}$ that exits $\overline{K_D}$ by some uncontrollable event. Thus $\overline{K_D}$ is uncontrollable with respect to every plant G consistent with (D, D^-), and consequently (D, D^-) is not informative. If an uncontrollable event σ cannot occur at state $\hat{q} \in Q_D$, this means that we have no data or prior knowledge about the corresponding transition, and thus we cannot determine whether the transition is generatable by the unknown true plant G. As a result, (D, D^-) is not informative.

Example 5. *Consider the data-driven automaton $\hat{G}_1$ in Fig. 5. For state $\hat{q}_{ab} \in Q_{D_1}$ (orange) and the (only) uncontrollable event d, it is satisfied that $\hat{\delta}(\hat{q}_{ab}, d) \in Q_{D_1}$ (orange). For every other state $q_s \in Q_{D_1}$ (orange) and the uncontrollable event d, it is satisfied that $\hat{\delta}(q_s, d) \in Q_-$ (blue). Thus Algorithm 1 returns "informative", which corresponds to the result in Example 3.*

Next consider the data-driven automaton $\hat{G}_2$ in Fig. 6. For state $\hat{q}_{ab} \in Q_{D_2}$ (orange) and the uncontrollable event d, we see that $\hat{\delta}(\hat{q}_{ab}, d)!$ and $\hat{\delta}(\hat{q}_{ab}, d) \notin Q_{D_2} \cup Q_-$ (since the transition enters a white state). As a result, Algorithm 1 returns "not informative", which again corresponds to the result in Example 3. In fact, the same conclusion can be drawn based on state $\hat{q}_{abcf} \in Q_{D_2}$ (orange); here $\neg\hat{\delta}(\hat{q}_{abcf}, d)!$, so Algorithm 1 returns "not informative".

IV. Conclusions

In this paper we have initiated a study on data-driven supervisory control of DES. A new concept of data-informativity has been proposed, and its characterization and verification algorithm presented.

As a first step in this data-driven direction, this work has left several interesting issues for future work. One is to consider the case that the given data set (D, D^-) fails to be informative for a given specification. Another problem is to consider marker states and thus to deal with data-informativity for nonblocking supervisors.

References

[1] W. M. Wonham and K. Cai, *Supervisor Control of Discrete-Event Systems*. Communications and Control Engineering, Springer, 2019.

[2] J. Cury, B. Krogh, and T. Niinomi, "Synthesis of supervisory controllers for hybrid systems based on approximating automata," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 564–568, 1998.

[3] P. Tabuada and G. J. Pappas, "Linear time logic control of discrete-time linear systems," *IEEE Transactions on Automatic Control*, vol. 51, no. 12, pp. 1862–1877, 2006.

[4] M. Gevers, A. S. Bazanella, X. Bombois, and L. Miskovic, "Identification and the information matrix: How to get just sufficiently rich?," *IEEE Transactions on Automatic Control*, vol. 54, no. 12, pp. 2828–2840, 2009.

[5] H. J. van Waarde, J. Eising, H. L. Trentelman, and M. K. Camlibel, "Data informativity: A new perspective on data-driven analysis and control," *IEEE Transactions on Automatic Control*, vol. 65, no. 11, pp. 4753–4768, 2020.

[6] S. Mukherjee, H. Bai, and A. Chakrabortty, "On model-free reinforcement learning of reduced-order optimal control for singularly perturbed systems," in *Proc. IEEE CDC*, pp. 5288–5293, 2018.

[7] A. Farooqui, F. Hagebring, and M. Fabian, "MIDES: A tool for supervisor synthesis via active learning," in *Proc. IEEE CASE*, pp. 792–797, 2021.

[8] M. Konishi, T. Sasaki, and K. Cai, "Efficient safe control via deep reinforcement learning and supervisory control – case study on multi-robot warehouse automation," in *Proc. IFAC WODES*, 2022.

[9] K. Cai, "Data-driven supervisory control of discrete-event systems," *Transactions of the Institute of Systems, Control and Information Engineers*, vol. 66, no. 9, pp. 359–364, 2022.

[10] P. M. Van den Hof and K. R. Ramaswamy, "Path-based data-informativity conditions for single module identification in dynamic networks," in *Proc. IEEE CDC*, pp. 4354–4359, 2020.

[11] T. R. V. Steentjes, M. Lazar, and P. M. J. Van den Hof, "On data-driven control: Informativity of noisy input-output data with cross-covariance bounds," *IEEE Control Systems Letters*, vol. 6, pp. 2192–2197, 2022.

[12] J. Hopcroft, R. Motwani, and J. Ullman, *Introduction to Automata Theory, Languages, and Computation*. Introduction to Automata Theory, Languages, and Computation, Pearson/Addison Wesley, 2007.