Data-Informativity for Data-Driven Supervisory Control of Discrete-Event Systems

Tomofumi Ohtsuka[1], Kai Cai[2], *Senior Member, IEEE* and Kenji Kashima[1], *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. Moreover, if the data set fails to be informative, we propose two related new concepts of *restricted data-informativity* and *informatizability*. Their characterization conditions and verification algorithms are also presented. Finally if the data set is informatizable, we develop an algorithm to compute the largest subset of control specification for which the data set is *least restricted informative*.

I. Introduction

Supervisory control of discrete-event systems (DES) is a relatively new area of control science and engineering [1], [2], [3], [4], [5]. 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

[1]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

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

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 [6], [7].

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 [8], learning control laws directly from observations [9] or through reinforcement learning [10]. 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 [11], [12], [13], 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 [9], and recently attracted much attention e.g., [14], [15], [16]. 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 Sections III, IV, and V 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* , which is a collection of observed behaviors (strings of events) from the unknown DES. The second type is *prior knowledge data* about the impossible behaviors of the DES. Then we define data-informativity of the pair - 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 (Theorem 1). 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 - is informative (for a given specification), we construct a valid supervisor for the unknown plant to satisfy the specification.

Third, in the case that the given data set - fails to be informative for a given specification, we propose another new concept called *restricted data-informativity*. restricted data-informativity means that for a given smaller subset of the specification, - is informative for the subset. Thus when restricted data-informativity holds, a valid supervisor can again be constructed for the unknown plant to satisfy the smaller specification. We characterize the existence of a nonempty subset of the specification for which - is informative as *informatizability*, and develop an algorithm to effectively verify this property. Finally, whenever - is verified to be informatizable, we show that there exists the *largest* (nonempty) subset of the specification for which - is informative. With respect to this largest subset, - is the *least restricted informative*. An algorithm is developed to find this largest subset of specification, and the corresponding supervisor that enforces this largest subset for the unknown plant is *optimal* in the sense of permitting maximal behaviors based on the data set -.

Overall, this work initiates and establishes the first systematic framework in the field of DES on data-driven supervisory control, which elucidates an intriguing interplay between data quality and control performance. Although in this first framework our focus is placed on the fundamental concept of controllability, this framework is naturally generalizable for many other DES concepts in the new data-driven setting.

This paper differs from its conference precursor [17] by developing concepts and algorithms for informatizability and least restricted informativity (i.e. Sections IV and V are new).

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 and Section IV introduces restricted data-informativity. Both sections present a necessary and sufficient condition and a verification algorithm for the respective property. Section V presents least restricted informativity and the corresponding synthesis algorithm. Section VI states our conclusions.

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]

$$= {}^{1}\&- \ -X-@_0{}^{o}\bullet \tag{1}$$

Here $\&$ is the finite state set, the finite event set, $X : \&$ $!$ $\&$ the (partial) state transition function,[2] $@_0 \ 2 \ \&$ the initial state. Write $X{}^{1}@- f{}^{o}!$ to mean $f \ 2$ is defined at state $@ \ 2 \ \&$, and write $:X{}^{1}@- f{}^{o}!$ to mean $f \ 2$ is not defined at state $@ \ 2 \ \&$. We say that the automaton is *deterministic* if

$${}^{1}8@ \ 2 \ \&-8f \ 2 \ {}^{o} \ X{}^{1}@- f{}^{o}! \ =) \ jX{}^{1}@- f{}^{o}j = 1\bullet$$

Namely, the destination state of every state transition is unique. We shall focus exclusively on deterministic automata unless otherwise stated. A string on is a sequence of events from . Write for the set of all finite-length strings on , including the empty string n (containing no event). Then, the state transition function may be inductively defined, and we write $X{}^{1}@-B{}^{o}!$ to mean that string $B \ 2$ is defined at state $@ \ 2 \ \&$. Any subset is called a language.

[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.

[2]It is sometimes convenient to treat X as a set $X = f{}^{1}@- f{}^{o} \ ! \ @^{0} \ j \ @-@^{0} \ 2 \ \&- f \ 2 \ g$. We shall use X either as a function or as a set as long as no confusion arises.

Write $\overline{\ } := \{B \in \ | (\exists B' \in \ ) BB' \in \ \}$ for the set of all *prefix* strings of those in . We call $\overline{\ }$ the *prefix closure* of , and in general $\ \subseteq \overline{\ }$ holds. The closed behavior $L(\ )$ of is the language defined as the set of all strings of which can generate starting from the initial state q_0:

$$L(\ ) := \{B \in \ | \delta(q_0, B)!\} \tag{2}$$

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

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

For a mechanism to enforce a specification $K \subseteq L(\ )$ on the plant , 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, $\Sigma_{uc} := \Sigma - \Sigma_c$ 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 $V : L(\ ) \to 2^{\Sigma_c}$. Here 2^{Σ_c} denotes the set of all subsets of controllable events. Thus a supervisor assigns to each string $B \in L(\ )$ generated by the plant a subset of controllable events $V(B) \subseteq \Sigma_c$ to be disabled. Write $V/$ for the closed-loop system: " is under the control of V". The language of closed-loop system $L(V/\ )$ is defined as follows:

(i) $\varepsilon \in L(V/\ )$;

(ii) if $B \in L(V/\ )$ and $\sigma \in \Sigma - V(B)$ and $B\sigma \in L(\ )$, then $B\sigma \in L(V/\ )$;

(iii) no other strings belong to $L(V/\ )$.

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

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

$$(\forall B \in \overline{K})(\forall \sigma \in \Sigma_{uc}) B\sigma \in L(\ ) \Rightarrow B\sigma \in \overline{K}. \tag{3}$$

In words, a specification is controllable wrt. 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 is invariant under uncontrollable flows. Equivalently (3) can be written compactly as $\overline{K}\Sigma_{uc} \cap L(\ ) \subseteq \overline{K}$

$\overline{\ }$. It is known that the specification language $\langle\ ;\rangle$ being controllable is necessary and sufficient for the existence of a supervisor $\ $ such that $\ \|\ \ =\ \overline{\ }$ [13].

Suppose that $\ \|\ \ $ is controllable. Then the supervisor $\ :\ \|\ \ \to\ _2$ such that $\ \|\ \ =\ \overline{\ }$ is constructed as follows:

$$\ (s)=\begin{cases}\{\sigma\in\ _2 \mid s\sigma\in\ \ \forall\ \in\ \}, & \text{if } s\in\ \\ \ , & \text{if } s\in\ \|\ \ \setminus\ \end{cases} \tag{4}$$

Whether or not $\ $ is controllable, we can write $\ \ $ for the family of all controllable sublanguages of $\ $:

$$\ \ :=\{\ \ \subseteq\ \mid\ \ \overline{\ }\cap\ \|\ \ \subseteq\ \overline{\ }\}. \tag{5}$$

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

$$\sup\ \ :=\bigcup\{\ \ \mid\ \ \in\ \ \}. \tag{6}$$

Since $\sup\ \ $ is controllable, as long as $\sup\ \ <\ $, there exists a supervisor $\ _{\sup}$ such that $\ \|_{\sup}\ \ =\ \overline{\sup\ \ }$. In this sense $\ _{\sup}$ is optimal (maximally permissive), allowing the generation by $\ $ of the largest possible set of 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 possible disturbance from the environment. An automaton modeling this described scenario is displayed in Fig. 1, and we consider this automaton (say $\ _1$) 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 $\ $ (dashed arrow from state 2 to 3) represents an uncontrollable transition, and other events are all controllable: i.e. $\ _2 = \{0, 1, 2, 4, 5\}$ and $\ _D = \{3\}$. State 5 represents a danger state, and

Figure 1 robot navigation: plant $_1$

Figure 2 robot navigation: speci cation $_1{}^1$! ^{1} $_1{}^{oo}$

the control speci cation is to avoid this danger state. This (safety) speci cation may be written as a sublanguage of ^{1} $_1{}^o$ as follows:

$$_1 = f012\,5-013\,15-021\,5-023\,1\,5$$

This speci cation $_1$ (indeed its pre x closure $\overline{_1}$) may be represented by the automaton shown in Fig. 2. Compared with the plant in Fig. 1, the speci cation automaton removes the (controllable) transition from state 4 to the danger state 5.

Since no uncontrollable event can exit $\overline{_1}$, the speci cation language $_1$ is controllable. Thus we can construct the following supervisor $: ! ^1$ $_1{}^o$! %FA $_2{}^o$ such that! $^1{}_1 \bullet$ $_1{}^o = \overline{_1}$:

$$^1{}_1B^o = \begin{cases} f4g & \text{if } B2\ f012-021-013g1 \\ ; & \text{if } B2\ ! ^1 \ _1{}^o \cap f012-021-013g1 \end{cases}$$

This supervisor disables at state 4 in Fig. 1.

In the above example, the supervisor $_1$ is designed based on the assumption that the plant model $_1$ is known. Now we pose this question: if $_1$ is unknown (e.g. the robot navigates in an unknown environment), 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 is unknown except for the event set $\Sigma = \Sigma_c \dot{\cup} \Sigma_o$. Even under this circumstance, there are often situations where strings generated by the plant may be observed, and thus certain amount of such 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 is turned on. In view of this, we assume that we can obtain a pair of finite data sets $(\mathcal{L}^+, \mathcal{L}^-)$, where $\mathcal{L}^+$ is the observed behavior from the plant and $\mathcal{L}^-$ is prior knowledge of impossible behavior of the plant. Since each string in $\mathcal{L}^+$ is observed from G, $\overline{\mathcal{L}^+}$ is a subset of the closed behavior of G, i.e. $\overline{\mathcal{L}^+} \subseteq \mathcal{L}(G)$. On the contrary since each string in $\mathcal{L}^-$ is known to be impossible to be generated by G, $\mathcal{L}^-$ and the closed behavior of G do not have any common elements: i.e. $\mathcal{L}^- \cap \mathcal{L}(G) = \emptyset$. As a result, $\overline{\mathcal{L}^+} \cap \mathcal{L}^- = \emptyset$. Given a control specification K, our goal is to design a supervisor (whenever it exists) to enforce K for the unknown plant based on the data pair $(\mathcal{L}^+, \mathcal{L}^-)$.

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 $\Sigma = \{0, 1, 2, 3, 4, 5\}$, and certain observations and prior knowledge of the plant behavior are available. For example, we have observed a string 01315 from the plant, i.e. the robot moves from the initial state to the goal state following the path 01315. Hence we have $\mathcal{L}^+ = \{\varepsilon, 0, 01, 013, 0131, 01315\}$. In addition, we have prior knowledge that the plant cannot generate string 2, namely the robot can never start its navigation from location 1 and makes a first move to location 3. Thus $\mathcal{L}^- = \{2\}$. For this pair $(\mathcal{L}^+, \mathcal{L}^-)$, there may exist infinitely many automata that can generate $\mathcal{L}^+$ and cannot generate $\mathcal{L}^-$. In other words, our unknown plant G_1 cannot be uniquely identified based on the pair $(\mathcal{L}^+, \mathcal{L}^-)$. 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 $(\mathcal{L}^+, \mathcal{L}^-)$, 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

Figure 3 G_2 in Example 2

Figure 4 G_3 in Example 2

real one. Say if we observe an additional string $02\,1\,5$ (so that $L = \{02\,1\,5, 01\,3\,1\,5\}$), 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 (L^+, L^-). This is defined below as a consistency property.

Definition 2 (consistency) Suppose that an event set Σ is given. Then, for finite sets L^+, L^- satisfying $\overline{L^+} \cap L^- = \emptyset$, an automaton $G = (X, \Sigma, \delta, x_0)$ is said to be consistent with (L^+, L^-) if $\overline{L} \supseteq L^+$ and $L \cap L^- = \emptyset$.

In words, an automaton G is consistent with a pair (L^+, L^-) if and only if all strings in L^+ can be generated by G, whereas no strings in L^- can be generated by G. Thus in Example 2, G_1, G_2 and G_3 are consistent with (L^+, L^-) where $L^+ = \{01315\}$ and $L^- = \{2\}$. If we observe an additional string $02\,1\,5$ ($L = \{02\,1\,5, 01\,3\,1\,5\}$), G_1 and G_3 are still consistent with (L^+, L^-), 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 [18] that for every regular language L, there exists an automaton G such that $L(G) = \overline{L}$. Thus for any finite sets L^+, L^- (which are regular) satisfying $\overline{L^+} \cap L^- = \emptyset$, one can always find an automaton such that $L(G) = \overline{L^+}$, and hence $L(G) \cap L^- = \emptyset$. This means that there exists at least one automaton consistent with (L^+, L^-).

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

Proposition 1. There exists a consistent plant with $\mathcal{L}(G) = L(G)$ if and only if $\overline{K} \cap L(G) = K$. If P is consistent with $\mathcal{L}(G) = L(G)$, then P is also consistent with $\mathcal{L}_m(G) = L_m(G)$. Moreover it holds that

$$\bigcup \{ L_m(P) \mid P \text{ is consistent with } \mathcal{L}(G) = L(G) \} = \overline{K} \tag{7}$$

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

Now we formulate our data-driven supervisory control problem. The last assertion (7) of Proposition 1 motivates us to investigate the supervisory control over $\overline{K}$. We denote a control specification based on K by

$$E := \overline{K} \cap A \quad \text{(regular language)} \tag{8}$$

Problem 1. Suppose that we are given an event set $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc}$, a control specification E, and finite data sets A such that E in (8) is nonempty and $\overline{K} \cap E = K$. Construct (if possible) a supervisor $V : \overline{E} \to 2^{\Sigma_c}$ such that $L_m(V/G) = K$ for every plant P consistent with $\mathcal{L}(G) = L(G)$.

Since the real plant is consistent with $\mathcal{L}(G) = L(G)$, 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 $K = L_m(G) \cap E$ (whenever K is controllable). In our data-driven setup, E represents the maximally accessible subset of $L_m(G)$ based on our observation of the plant; hence constructing a supervisor to enforce $K = \overline{E} \cap E$ is the most that can be done based on the data available. If the behavior represented by E is too small/restrictive, one can consider enlarging E by observing more behaviors of the plant. The closer E approximates $L_m(G)$, the closer the data-driven enforcible behavior approximates the original model-based behavior K.

B. Data-informativity and its criterion

As we mentioned in Section II, the existence of a supervisor such that $L_m(V/G) = K$ for all plants P consistent with $\mathcal{L}(G) = L(G)$ is equivalent to the controllability of specification language K with $L_m(G)$. This implies that whether the available data has sufficient information can be characterized in terms of controllability.

De nition 3 (informativity). We say that $(L - K)^{\circ}$ is informative for a given control speci cation if there exists a supervisor satisfying the required condition in Problem 1, or equivalently if in (8) is nonempty and controllable with respect to all plants consistent with $(L - K)^{\circ}$.

Recall that, for a known plant, if an uncontrollable event $\sigma \in \Sigma_{uc}$ can happen after $s \in \overline{K}$ in the plant, then sf needs to remain in $\overline{K}$ for the controllability of $\overline{K}$ 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 $\sigma \in \Sigma_{uc}$ can happen after $s \in \overline{K}$ unless $sf \notin L$ (known to be impossible). This observation leads to the following:

Theorem 1 (Criterion for informativity). Suppose that an event set $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc}$ and a control speci cation are given. $(L - K)^{\circ}$ is informative for if and only if

$$\forall s \in \overline{K} - \forall f \in \Sigma_{uc} \quad sf \in \overline{L} \quad sf \in \overline{K} \qquad (9)$$

holds with in (8).

Proof. (If) Suppose that (9) holds. Then it is easy to verify that

$$\forall s \in \overline{K} - \forall f \in \Sigma_{uc} \quad sf \in \overline{L} \Rightarrow sf \in \overline{K} \qquad (10)$$

holds for every plant consistent with $(L - K)^{\circ}$. Thus $(L - K)^{\circ}$ is informative for .

(Only if) Suppose that $(L - K)^{\circ}$ is informative for . This means by De nition 3 that (10) holds for every plant consistent with $(L - K)^{\circ}$. Consider a special such plant ^{0} such that $L(^0) = \overline{K}$. This ^{0} is consistent with $(L - K)^{\circ}$, and any string not in $\overline{L}(^0)$ belongs to . Let $s \in \overline{K}$ and $f \in \Sigma_{uc}$. Consider two cases. Case 1: $sf \in \overline{L}(^0)$. Since (10) holds for ^{0}, we have $sf \in \overline{K}$. Case 2: $sf \notin \overline{L}(^0)$. In this case, we have $sf \in$. Thus (9) is satis ed.

Theorem 1 provides a necessary and su cient condition for data-informativity. Compared (9) with the standard controllability condition of , absence of $\overline{L}(^0)$ and the part of mark the distinctions in the data-driven framework. Below we remark on the key role played by in a ecting the quality of data set.

Remark 2. The set contains strings that cannot be generated the unknown plant (i.e. prior knowledge of impossible behavior of the plant). If we have little such prior knowledge, i.e., $\overline{L}(^0)$, in order for (9) to be satis ed, $\overline{K}$ has to include (almost) all one-step uncontrollable

continuations of strings belonginig to itself. This in turn requires the observation data set to be rather exhaustive with respect to the occurrence of uncontrollable events, which may be challenging to obtain in practice (therefore (9) is di cult to be satis ed). Hence in general, a larger prior knowledge e ectively helps relieve the requirement on obtaining observation data . Note, however, that not all strings in are equally useful; according to (9), the useful strings in are precisely those of one-step uncontrollable continuation of strings in . In view of the above, it is not the sheer quantity of or , but the quality in terms of the `matchness' between and speci ed by (9) that matters for informativity.

Whenever $^1 - ^o$ is informative for , the following proposition provides the way to construct a supervisor to realize control speci cation (i.e. a solution to Problem 1).

Proposition 2. Suppose that we are given a nite set of strings $- ^o$ which is informative for a given speci cation . Then a supervisor $= : \overline{} ! \ \%FA \ _2{}^o$ such that $! \ ^1 + \bullet \ ^o = \overline{\overline{\ }}$ is constructed as follows:

$$+ \ ^1B^o = \begin{cases} \{ \ 2 \ _2 \mid Bf \ 8 \ \overline{\overline{\ }} \ g & \text{if } B2 \ \overline{\overline{\ }} - \\ ; & \text{if } B2 \ \overline{\ } \ n \ \overline{\overline{\ }} \bullet \end{cases} \tag{11}$$

The above proposition can be readily derived by employing the supervisor construction method used in the model-known case (as in (4)).

Example 3. Let us illustrate the concept of data-informativity using the robot navigation example. Again we suppose that the plant $_1$ in Fig. 1 is unknown except for the event set $= \ _2 [\ _D$, where $_2 = f0 - 1 - 2 - 4g$ and $_D = f3g$. Consider two pairs of nite data sets $^1 \ _1 - \ _1{}^o$ and $^1 \ _2 - \ _2{}^o$, where

$$_1 = f01315 - 012g$$

$$_1 = f3 - 03 - 0133 - 01313 - 0131g$$

$$_2 = f0125 - 0131g$$

$$_2 = f3 - 03 - 012g$$

Let $= f0125 - 01315 - 02115 - 0135g$ then the control speci cations in (8) are respectively

$$_1 = \overline{\ _1} \setminus \ = f01315g -$$

$$_2 = \overline{\ _2} \setminus \ = f0125g \bullet$$

First consider $(L_1 - L_1)$. Note that L_1 in Fig. 1 and L_2 in Fig. 3 are consistent with $(L_1 - L_1)$. From the control specification $K_1 = \{0131, 5\}$, we have $\overline{K_1} = \{\varepsilon, 0, 01, 013, 0131, 01315\}$ and $\overline{K_1}D = \{3, 03, 013, 0133, 01313, 013135\}$. Since only $013 \in \overline{K_1}$ and other strings in $\overline{K_1}D$ belong to L_1, the condition (9) holds and $(L_1 - L_1)$ is informative for K. Indeed, we can confirm the controllability of K_1 with respect to the consistent plants L_1 and L_2 (if they were available). Correspondingly a supervisor $S_1 : \overline{L_1} \to \{F, A\}^*$ such that $L(S_1/L_1) = \overline{K_1}$ is constructed for every plant consistent with $(L_1 - L_1)$ as follows:

$$S_1(s) = \begin{cases} \{1, 2, 4, 5\} & \text{if } s = \varepsilon \\ \{0, 2, 4, 5\} & \text{if } s = 0 \\ \{0, 1, 2, 4, 5\} & \text{if } s = 01 \\ \{0, 2, 4, 5\} & \text{if } s = 013 \\ \{0, 1, 2, 4\} & \text{if } s = 0131 \\ \{0, 1, 2, 4, 5\} & \text{if } s = 01315 \\ \vdots & \\ \text{if } s \in \overline{K_1} \cap \overline{L_1}^* \end{cases}$$

Next, we consider $(L_1 - L_2)$. Note that L_1 in Fig. 1 and L_3 in Fig. 4 are consistent with $(L_1 - L_2)$. From the control specification $K_2 = \{0125\}$, we have $\overline{K_2} = \{\varepsilon, 0, 01, 012, 0125\}$ and $\overline{K_2}D = \{3, 03, 013, 0123, 01253\}$. Since $013, 012533 \in \overline{K_2}[L_2, (L_1 - L_2)$ is not informative for K. The string 013 is in $\overline{K_2} \cap \overline{L_2}$, so 013 is the string which is outside of the specification and generatable by every consistent plant. On the contrary, we do not have the information of the string 01253, so it is generatable by some consistent plant and not generatable by others. For example, L_1 in Fig. 1 cannot generate 01253 but L_3 in Fig. 4 can generate it.

C. Verification of data-informativity

Based on the condition (9) 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 (as will be clear from the definition below, in the data-driven automaton a state and a string are uniquely corresponded).

Figure 5 Data-driven automaton $\hat{A}_1$ corresponding to $\{L_1^+, L_1^-\}$

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

$$\hat{A}(L^+, L^-) = (\hat{X}, \Sigma, \hat{\delta}, \hat{x}_0) \tag{12}$$

where $\hat{X} := \{x_B \mid B \subseteq \overline{L^+}\}$ is the state set, $\hat{\delta} := \{\delta(x_B, \sigma) = x_{B_f} \mid B \subseteq \overline{L^+}, \sigma \in \Sigma, B_f \subseteq \overline{L^+}\}$ is the (partial) state transition function, and x_0 is the initial state. In addition, given a control specification $K = \overline{K} \cap L_m(G)$ (where $L_m(G)$ is a regular language), we define $\hat{X}_K := \{x_B \in \hat{X} \mid B \subseteq \overline{K}\}$ and $\hat{X}_{\overline{K}} := \{x_B \in \hat{X} \mid B \subseteq L_m(G)\}$.

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

Example 4. Here we provide examples of data-driven automata $\hat{A}_1$ (in Fig. 5) and $\hat{A}_2$ (in Fig. 6) corresponding to the data sets $\{L_1^+, L_1^-\}$ and $\{L_2^+, L_2^-\}$ in Example 3. For clear display, we have omitted x in the figure and only the subscript B is written inside each state. State subsets $\hat{X}_K$ and $\hat{X}_{\overline{K}}$ are represented in orange and blue in the figures, respectively. The states without

Figure 6 Data-driven automaton $\hat{}_2$ corresponding to $_1$ $_2$– $_2$ ^{o}

colors correspond to strings in the observation data set $\bar{}$, but not in the specification (thus not in $\bar{}$).

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

Algorithm 1 checking informativity

Input: event set = $_2$ [$_D$, finite sets – ^{1} ^{o}, control specification = $\bar{}$ \

Ensure: informative or not informative

1: construct a data-driven automaton $\hat{}^1$ – – ^{o} = ^{1} & – – $\hat{}$ @ ^{o} and & – & (as in Definition 4)
2: for all @ 2 & do
3: for all f 2 $_D$ do
4: if $\hat{}^1$ @ – f 2 & $\hat{}$ n ^{1} & [& ^{o} or : $\hat{}^1$ @ – f ! then
5: return not informative
6: break
7: end if
8: end for
9: end for
10: return informative

In Algorithm 1, informativity of ^{1} – ^{o} for is determined by examining in the data-driven automaton every uncontrollable event at each state in & . If an uncontrollable event f can occur at state @ 2 & and the corresponding transition enters $\hat{}$ n ^{1} & [& ^{o}, then the transition is contained in ! ^{1} ^{o} (for all plants consistent with ^{1} – ^{o}) but not contained in $\bar{}$, which means that there exists a string in $\bar{}$ that exits $\bar{}$ by some uncontrollable event. Thus $\bar{}$ is

uncontrollable with respect to every plant consistent with $\mathcal{L}(G, \varphi)$, and consequently $\mathcal{L}(G, \varphi)$ is not informative. If an uncontrollable event cannot occur at state $q \in Q$, 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. As a result, $\mathcal{L}(G, \varphi)$ is not informative. The correctness of Algorithm 1 is asserted by the following proposition.

Proposition 3. Algorithm 1 returns informative if and only if $\mathcal{L}(G, \varphi)$ is informative for L.

Proof. (If) Suppose that $\mathcal{L}(G, \varphi)$ is informative for L. Then from Theorem 1, every string $s \in \overline{L}$ and every uncontrollable event $\sigma \in \Sigma_{uc}$ satisfy (9). From the definition of Q_d and δ_d, $\delta_d(q, \sigma)!$ is defined at all $q \in Q_d$ and $\delta_d(q, \sigma) = x_0 \Rightarrow \delta(q, \sigma) \in Q_d$ holds. This means that Algorithm 1 returns informative G_d.

(Only if) Suppose that Algorithm 1 returns informative G_d. Then $\delta_d(q, \sigma) = x_0 \Rightarrow \delta(q, \sigma) \in Q_d$ holds for all $q \in Q_d$ and for all $\sigma \in \Sigma_{uc}$. If $\delta_d(q, \sigma)!$ then $s\sigma \in \overline{L}$ holds, and if $\delta_d(q, \sigma)!$ then $s\sigma \in L$ holds. Therefore, (9) holds and $\mathcal{L}(G, \varphi)$ is informative for L.

Remark 3. We analyze the complexity of Algorithm 1. Let $n = |Q|$ be the total number of strings in Q, and m be the length of the longest string in Q. Then the complexity of line 1 is $O(nm \frac{m(m+1)}{2}) = O(nm^2)$ (a string of length m can generate at most $\frac{m(m+1)}{2}$ states in the data-driven automaton, and all strings share at least the initial state which needs to be counted only once). Note that although the determination of Σ_{uc} (also in line 1) requires the automaton representing the specification, the complexity does not depend on the state size of that automaton but only on the lengths of strings in Q. Lines 2–9 consist of two for-loops, whose complexity is $O(|Q||\Sigma_{uc}|)$ (the complexity of line 4 can be made constant by augmenting each state $q \in Q_d$ with a binary-valued attribute: (say) 0 for $q \in Q_d \setminus Q$, and 1 for $q \in Q$; this state attribute can be determined during the construction of Q_d in line 1). Since $|Q| \leq |Q_d|$ and $O(nm) = O(nm^2)$, the total complexity of Algorithm 1 is $O(nm^2|\Sigma_{uc}|)$. If we treat $|\Sigma_{uc}|$ as a constant, Algorithm 1's complexity is related to the amount and length of the data sets—n in terms of $O(nm^2)$.

Example 5. Consider the data-driven automaton G_{d1} in Fig. 5. For state $q_{d1} \in Q$ (orange) and the (only) uncontrollable event β, it is satisfied that $x_0 \Rightarrow \delta(q_{d1}, \beta)! \in Q$ (orange). For every other state $q_B \in Q$ (orange) and the uncontrollable event β, it is satisfied that $x_0 \Rightarrow \delta(q_B, \beta)! \notin Q$ (blue). Thus Algorithm 1 returns informative G_d, which corresponds to the result in Example 3.

Next consider the data-driven automaton $\hat{G}_2$ in Fig. 6. For state $q_1 \in Q_d$ (orange) and the uncontrollable event β, we see that $\hat{\delta}(q_1, \beta)!$ and $\hat{\delta}(q_1, \beta) \notin Q_d$ (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 $q_{125} \in Q_d$ (orange); here $\hat{\delta}(q_{125}, \beta)!$, so Algorithm 1 returns *not informative*.

We end this section with a note on the quantity versus the quality of the data set-pair $(\mathcal{L}, \mathcal{K})$. On one hand, enlarging the set (by making more observations) can reduce the number of models that are indistinguishable from the real plant, as well as allow more behaviors to be enforced. On the other hand, by Theorem 1 (and also Algorithm 1), a larger $\mathcal{L}$ means that more strings need to be checked against the condition (9), and thus data-informativity is more difficult to hold (unless the prior knowledge data $\mathcal{K}$ can also be enlarged accordingly). Hence data-informativity is concerned not just with the sheer quantity of the data, but with the matching quality between the observation and the prior knowledge (in the sense of satisfying (9) as we pointed out in Remark 2). In case that $(\mathcal{L}, \mathcal{K})$ fails to be informative for a specification K and the prior knowledge $\mathcal{K}$ cannot be enlarged, then rather than considering making more observations for $\mathcal{L}$, one should look for a smaller subset $\bar{K} \subseteq K$ such that $(\mathcal{L}, \mathcal{K})$ may be informative. This problem is studied in the next section.

IV. K-restricted Data-informativity

A. K-informativity

Given an event set $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc}$, finite data sets $\mathcal{L}, \mathcal{K} \subseteq \Sigma^*$, and a control specification $K \subseteq \Sigma^*$ in (8), if $(\mathcal{L}, \mathcal{K})$ is verified to be *not informative* for K, then there exists no supervisor to solve Problem 1. However, it is still possible that $(\mathcal{L}, \mathcal{K})$ is informative for a smaller subset $\bar{K} \subseteq K$. If this is the case, a valid supervisor may be constructed to enforce the smaller specification $\bar{K}$ for all the plants consistent with $(\mathcal{L}, \mathcal{K})$ (including the real plant). In this section we formulate the notion of *restricted data-informativity* which aims to establish informativity by constraining the specification to a smaller subset. Whether or not this restricted informativity holds hinges on the particular subset $\bar{K}$ in question, so we simply term it $\bar{K}$-informativity and define it as follows.

Definition 5 ($\bar{K}$-informativity). Consider a specification K in (8) and let $\bar{K} \subseteq K$. We say $(\mathcal{L}, \mathcal{K})$ is $\bar{K}$-informative if there exists a supervisor $V : \bar{K} \to \Gamma_{FA}(\Sigma_c)$ such that $\mathcal{L}(\mathcal{L}, \mathcal{K}) \cap K = \bar{K}$

for every plant consistent with ¹ – º, or equivalently if is controllable with respect to every plant consistent with ¹ – º.

By this de nition, if ¹ – º is already informative for the speci cation , then ¹ – º is -informative (= ¯ \ ). On the other hand, if = ; , since ; is trivially controllable, ¹ – º is always ; -informative. However, enforcing (i.e. empty behavior) is of little practical use, so we will henceforth only consider nonempty .

If ¹ – º is -informative for a given , then the supervisor to realize can be constructed for every plant consistent with ¹ – º in the same way as Proposition 2.

For the veri cation of -informativity, a straightforward modi cation of Algorithm 1 su ces. This is asserted below, as a corollary of Proposition 3.

Corollary 1. Suppose that we are given an event set = $_2$ [$_D$, nite data sets – , a control speci cation and a subset ¯ with in (8). Then Algorithm 1 with & rede ned as & := f X̂ ¹ @ – B j B 2 ¯ g returns informative if and only if ¹ – º is -informative.

We note that in general -informativity of ¹ – º does not imply ^{0}-informativity for ¯ ^{0} ¯ (similar to the fact that a sublanguage of a controllable language need not be controllable). This is illustrated by an example.

Example 6. Consider again the data pair ¹ $_1$ – $_1$ º in Example 3. Since ¹ $_1$ – $_1$ º is informative for , ¹ $_1$ – $_1$ º is $_1$-informative. However, let $^0_1$:= f 0 1 g (so ¯ $^0_1$ (¯ $_1$ º. Applying the modi ed Algorithm 1 as in Corollary 1 returns not informative . This is because @ $_1$ 2 & but X̂ ¹ @ $_1$ – 3º = @ $_{13}$ 8 & [& . Consequently, ¹ $_1$ – $_1$ º is not $^0_1$-informative.

B. Informatizability and its criterion

While in the preceding subsection -informativity is de ned and checked for a given subset ¯ , we investigate in this section whether or not such a nonempty subset exists. This problem is formulated below.

Problem 2. Suppose that we are given an event set = $_2$ [$_D$, a control speci cation , and nite data sets – such that < ; and ¯ \ = ; . Determine whether or not there exists a nonempty sublanguage ¯ such that ¹ – º is -informative.

We term the solvability of Problem 2 as a property of the data pair (P, P_D).

Definition 6 (informatizability). We say that (P, P_D) is informatizable for a given control specification K if there exists a nonempty sublanguage $H \subseteq K$ such that (P, P_D) is H-informative.

Now we characterize this property of informatizability. Consider that (P, P_D) is not informative for a given specification K. Then by Theorem 1, there exists some string $s \in \overline{H}$ that can uncontrollably exit $\overline{K}$, i.e., $sE_{uc} \cap L(P) \not\subseteq \overline{K}$. If for every such 'bad' string, a controllable event exists in its prefixes, then this controllable event may be disabled to prevent the bad string from exiting $\overline{K}$ uncontrollably. Putting it in another way, as long as we can make sure that all the purely uncontrollable strings (i.e. containing only uncontrollable events) in $\overline{H}$ will not uncontrollably exit $\overline{K}$, then a nonempty sublanguage H of K can be found by blocking all bad strings through properly disabling controllable events. This reasoning leads to the following theorem.

Theorem 2 (Criterion for informatizability). Suppose that an event set $E = E_o \dot{\cup} E_{uo}$ and a control specification K are given. (P, P_D) is informatizable for K if and only if

$$\forall s \in \overline{K} \cap L_D : \forall \sigma \in E_{uc} : s\sigma \in L_D \Rightarrow s\sigma \in \overline{K} \quad (13)$$

Proof. (If) Suppose that (13) holds. Consider a sublanguage $H \subseteq K$ such that

$$\overline{H} = \{s \in \overline{K} \cap L_D \mid \forall \sigma \in E_{uc} : s\sigma \in L_D \Rightarrow s\sigma \in \overline{K}\} \cup \{\varepsilon\}$$

Note that H is nonempty since $\overline{H} \neq \emptyset$. The latter is because under (13) we have $\varepsilon \in \overline{H}$. Moreover, since every string in $\overline{H}$ remains in $\overline{H}$ following an uncontrollable event, the sublanguage H is controllable for all plants consistent with (P, P_D). This by Definition 5 means that (P, P_D) is H-informative. Therefore (P, P_D) is informatizable for K.

(Only if) Suppose that (13) does not hold. Then we have

$$\neg(\forall s \in \overline{K} \cap L_D : \forall \sigma \in E_{uc} : s\sigma \in L_D \Rightarrow s\sigma \in \overline{K}) \quad (14)$$

This means that even for the empty string ε will exit $\overline{K}$ following the string $s\sigma$ in (14). Thus there does not exist any nonempty sublanguage of K which is controllable. As a result, (P, P_D) is not informatizable for K.

Example 7. Consider again the data pair $(P_1, P_{1,D})$ in Example 3, which is not informative for the specification K. We check if $(P_1, P_{1,D})$ is informatizable for K by checking the condition (13)

in Theorem 2. Since $\overline{L_2} \cap L_D = \{ng$ and $3 \in L_2$, (13) holds and $L_1 - L_2$ is informatizable for .

Based on the condition (13) in Theorem 2, we next present an algorithm for the verification of informatizability. This algorithm is again based on data-driven automaton (as in Definition 4).

Algorithm 2 checking informatizability

Input: event set $= \Sigma_2 \cup \Sigma_D$, finite sets $L_1 \quad L_2$, control specification $= \overline{\quad} \cap$

Ensure: informatizable or not informatizable

1: construct a data-driven automaton $\hat{1} \quad \quad = \hat{1}\&\quad \quad \hat{X}\quad @$ and $\& \quad \&$ (as in Definition 4)
2: initiate $\&_D := f @g$, $\&_{tmp} := ;$
3: while $\&_D < ;$ do
4: for all $@2 \&_D$ do
5: for all $f \in \Sigma_D$ do
6: if $\hat{X}^1 @ f 2 \&$ then
7: $\&_{tmp} = \&_{tmp} [f \hat{X}^1 @ fg$
8: else if $\hat{X}^1 @ f 2 \hat{\&} n \&$ or $: \hat{X}^1 @ f!$ then
9: return not informatizable
10: break
11: end if
12: end for
13: end for
14: update $\&_D = \&_{tmp}$, and reset $\&_{tmp} = ;$
15: end while
16: return informatizable

In Algorithm 2, informatizability of $L_1 - L_2$ for is checked by examining in the data-driven automaton every uncontrollable event at every state in that is reached by a sequence of uncontrollable events. If an uncontrollable event can occur at such an uncontrollably reachable state $@2 \&$ and the corresponding transition enters (line 6), then the new state is also uncontrollably reachable which needs to be further examined by storing it in (line 7). On

the other hand, if an uncontrollable event t can occur at an uncontrollably reachable state $q \in Q$ and the corresponding transition exists $\in [\ ]$ (line 8), then informatizability cannot hold by Theorem 2. Another case where informatizability cannot hold is when an uncontrollable event is not defined at an uncontrollable reachable state (line 8); this means that the available data do not contain enough information to determine informativity for any nonempty sublanguage $\overline{\ }$. The correctness of Algorithm 2 is asserted below.

Proposition 4. Algorithm 2 returns informatizable if and only if $(\ , \ )$ is informatizable for $\ $.

Proof. (If) Suppose that Algorithm 2 returns informatizable. Then every string $s \in \ _D$ and every uncontrollable event $t \in \ _D$ satisfy $\delta(q, s) \in Q$ and $\delta^{-1}(\delta(q, s), t) \in [\ ]$. From the definition of Q and $[\ ]$, (13) holds and $(\ , \ )$ is informatizable for $\ $.

(Only if) Suppose that Algorithm 2 returns not informatizable. Then there exists a state $\delta(q, s) \in Q$ with $s \in \ _D$ and an uncontrollable event $t \in \ _D$ such that either of the following two conditions holds: (i) $\delta^{-1}(\delta(q, s), t) \neq \emptyset$ and $\delta^{-1}(\delta(q, s), t) \notin [\ ]$, or (ii) $\delta^{-1}(\delta(q, s), t) = \emptyset$. Thus from the definition of Q and $[\ ]$, we have

$$(\exists s \in \overline{\ }) \ s \in \ _D \wedge (\exists t \in \ _D) \ st \in \ _D \wedge (\forall f \in \overline{\ }) \ st \notin f \ . \quad \bullet$$

This is exactly (14), and by the same proof for Theorem 2 we conclude that $(\ , \ )$ is not informatizable for $\ $.

Remark 4. We analyze the complexity of Algorithm 2. As in Remark 3, let $n = |Q|$ be the total number of strings in $[\ ]$, and m be the length of the longest string in $[\ ]$. Then the complexity of line 1 of constructing the data-driven automaton $\mathcal{A}$ together with Q and $[\ ]$ is $O(nm^2)$ (as analyzed in Remark 3). Since $\mathcal{A}$ has a tree structure (loopless) and the way $Q \to Q_D$ are updated respectively in lines 7 and 14, the while-loop (from line 3) and the first for-loop (from line 4) together can be executed no more than $|Q|$ times (in other words, no state in Q is checked more than once). Hence the complexity of lines 3–15 is $O(|Q||\Sigma||Q_D|)$ (the complexity of line 6/8 can be made constant as noted above in Remark 3). Since $|\Sigma||Q| = O(nm^2)$, the total complexity of Algorithm 2 is $O(nm^2|Q_D|)$ (the same as that of Algorithm 1). If we treat $|Q_D|$ as a constant, Algorithm 2's complexity is related again to the amount and length of the data sets – in terms of $O(nm^2)$.

Example 8. Consider the data-driven automaton $\hat{\ }_2$ in Fig. 6. For the initial state $@_H$ (orange) and the (only) uncontrollable event 3, it is satis ed that $\hat{X}^1 @_H - 3^o 2 \&$ (blue). Thus Algorithm 2 terminates and returns informatizable , which corresponds to the result in Example 7.

We remark that if a data pair $^1 -\ ^o$ is veri ed to be informatizable for a given speci cation , then a valid supervisor can be designed to enforce a nonempty sublanguage of $\overline{\ }$ for all plants consistent with $^1 -\ ^o$. On the other hand, if $^1 -\ ^o$ fails to be informatizable, then no nonempty supervisor can exist. The latter case indicates that the quality of the currently available $^1 -\ ^o$ must be improved, by possibly making more observations to enlarge and matching it up with more prior knowledge in .

V. Least Restricted Informativity

In Section IV, if informatizability of the data pair $^1 -\ ^o$ is veri ed to hold, then the existence of a nonempty subset $\overline{\ }$ is assured such that $^1 -\ ^o$ is -informative. Namely $^1 -\ ^o$ satis es limited informativity wrt. . In this section, we further investigate how to systematically nd such a nonempty . Of particular interest is to nd (if possible) the largest $_{sup}$ $\overline{\ }$, so that $^1 -\ ^o$ is least restricted informative wrt. $_{sup}$. Then the corresponding supervisor that enforces $_{sup}$ is the maximally permissive one in the sense of allowing the largest set of behaviors as possible.

We start by de ning the following family of subsets of $\overline{\ }$ with respect to which the data pair $^1 -\ ^o$ is restricted informative:

$$I^1\ ^o := f\ \overline{\ }\ j\ ^1 -\ ^o \text{ is -informative} g\bullet \tag{15}$$

Thus if $^1 -\ ^o$ is informatizable, the family $I^1\ ^o$ contains a nonempty member. Note also that since $^1 = \overline{\ } \backslash\ ^o$ is nite, the number of members in $I^1\ ^o$ is nite.

The next result is key, which asserts that the family $I^1\ ^o$ is closed under set unions.

Proposition 5. Consider the family $I^1\ ^o$ in (15). If $_1 -\ _2\ 2\ I^1\ ^o$, then $_1 [\ _2\ 2\ I^1\ ^o$.

Proof. According to the de nition of limited informativity, letting $B2\ \overline{_1 [\ _2}$ and $f\ 2\ _D$, we will show that $Bf\ 2\ \overline{_1 [\ _2} [$. Since $B2\ \overline{_1 [\ _2} = \overline{_1} [\ \overline{_2}$, either $B2\ \overline{_1}$ or $B2\ \overline{_2}$. Consider the former case $B2\ \overline{_1}$ (the latter case $B2\ \overline{_2}$ is symmetric). Since $^1 -\ ^o$ is $_1$-informative, we have

$$Bf\ 2\ \overline{_1} [\ \ ^1\ \overline{_1} [\ \overline{_2}^o [\ \ = \overline{_1 [\ _2} [\ \ \bullet$$

This completes the proof.

In view of Proposition 5, the family $\mathcal{V}(L, K)$ contains a unique largest member $V^\uparrow_{\sup}$, which is the union of all the members in the family:

$$V^\uparrow_{\sup} := \bigcup \{V \mid V \in \mathcal{V}(L, K)\} \tag{16}$$

With respect to this $V^\uparrow_{\sup}$, the data pair (L, K) is least restricted informative.

The supervisor that enforces $V^\uparrow_{\sup}$ can be constructed for every plant consistent with (L, K) in the same way as Proposition 2; due to the largestness of $V^\uparrow_{\sup}$, this supervisor is maximally permissive.

Now that we have shown the existence and uniqueness of the largest subset $V^\uparrow_{\sup}$, we proceed to develop an algorithm to compute $V^\uparrow_{\sup}$.

A. Non-informative state

For computing $V^\uparrow_{\sup}$, we first introduce a useful concept of non-informative state.

Given an event set Σ, a control specification K, and finite data sets S^+, S^- (satisfying $S^+ \cap S^- = \emptyset$), construct the corresponding data-driven automaton $\mathcal{A}(S^+, S^-) = (Q, \Sigma, \delta, q_0)$ with state subsets Q_+ (corresponding to $S^+ = L \cap K$) and Q_- (corresponding to S^-) as in Definition 4. We identify those states in Q_+ that violate the condition (9) of informativity. These are exactly the states for which the condition in line 4 of Algorithm 1 holds. According to the condition, we state the following definition.

Definition 7 (non-informative state): Consider the data-driven automaton $\mathcal{A}(S^+, S^-) = (Q, \Sigma, \delta, q_0)$ with state subsets Q_+, Q_- (as in Definition 4). We say that $q \in Q_+$ is a non-informative state if

$$\exists \sigma \in \Sigma_c \cup \{\sigma_m\} \text{ such that } \delta(q, \sigma) \in Q_+ \text{ and } \delta(q, \sigma) \in Q_- \text{ or } \delta(q, \sigma)! \tag{17}$$

In addition, define the set of non-informative states as follows:

$$\#(Q_+, Q_-) := \{q \in Q_+ \mid q \text{ is a non-informative state}\} \tag{18}$$

The set $\#(Q_+, Q_-)$ in (18) provides alternative characterizations (to Propositions 3 and 4) for informativity and informatizability, as asserted below.

Proposition 6. Consider an event set $\Sigma = \Sigma_c \cup \Sigma_{uc}$, finite data sets S^+, S^- (satisfying $S^+ \cap S^- = \emptyset$), a control specification K, a subset $V^\uparrow$ with V in (8), a corresponding

data-driven automaton $\hat{}^{1}$ – – º = ¹&– –X-Q^{o}, and the non-informative state set #¹& º in (18). Then the following hold:

1) #¹& º = ; if and only if ¹ – º is informative for .

2) f X¹Q– Bj B2 ¯\ $_{D}$g \ #¹& º = ; if and only if ¹ – º is informatizable for .

Proof. 1) According to Algorithm 1, #¹& º = ; if and only if the condition in line 4 is never satis ed, which in turn means that Algorithm 1 turns informative . It then follows from Proposition 3 that Algorithm 1 turns informative if and only if ¹ – º is informative for ; hence the conclusion holds.

2) (If) According to Algorithm 2, f X¹Q– BjB2 ¯\ $_{D}$g \ #¹& º < ; if and only if the condition in line 8 is satis ed at least once. To see this, f X¹Q– BjB2 ¯\ $_{D}$g \ #¹& º < ; means that there exists a state Q2 #¹& º such that B2 $_{D}$. On one hand, it follows from B2 $_{D}$ that state Q is included in &$_{D}$ according to Algorithm 2 On the other hand, according to De nition 7, Q2 #¹& º means that

$$^{1}9f \ 2 \ _{D}{}^{o} \ X^{1}Q– f^{o} 2 \hat{} n \ ^{1}\& \ [\ \& \ ^{o} \ or \ : X^{1}Q– f^{o}!$$

This above implies that the condition in line 8 is satis ed for the state Q and the uncontrollable event f . Hence, f X¹Q– BjB2 ¯\ $_{D}$g \ #¹& º < ; if and only if Algorithm 2 returns not informatizable . It then follows from Proposition 4 that Algorithm 2 turns not informatizable if and only if ¹ – º is not informatizable for ; hence the conclusion holds.

The computation of the non-informative state set can be adapted from Algorithm 1: instead of returning not informative immediately after identifying the rst non-informative state, the new algorithm checks all states in Q against all uncontrollable events in $_{D}$, and stores all identi ed non-informative states. This new algorithm of computing #¹& º is presented in Algorithm 3 below.

Algorithm 3 non-informative state set

Input: event set = $_2[$ $_D$, nite sets $-$ ^{1} ^{o}, control speci cation = $\overline{\ }\backslash$

Ensure: non-informative state set $\#^1\&\ ^o$

1: construct a data-driven automaton $\hat{}^1$ $-$ $-$ $^o = {}^1\&\hat{}$ $-$ $-\hat{X}$ @ ^{o} and $\&$ $-\&$ (as in De nition 4)
2: $\#^1\&\ ^o = ;$
3: for all @ 2 $\&$ do
4: for all f 2 $_D$ do
5: if $\hat{X}^1$@$-$f 2 $\&\hat{}$ n $^1\&$ [$\&$ ^{o} or : $\hat{X}^1$@$-$f! then
6: $\#^1\&\ ^o = \#^1\&\ ^o$ [f @g
7: end if
8: end for
9: end for
10: return $\#^1\&\ ^o$

The correctness of Algorithm 3 is immediate from De nition 7, and its complexity is the same as that of Algorithm 1.

B. Algorithm for computing $_{sup}$

Having introduced and identi ed the set $\#^1\&\ ^o$ of non-informative states in the data-driven automaton $\hat{}$, it follows from its de nition (De nition 7) that any string in $\overline{\ }$ that reaches a non-informative state in $\#^1\&\ ^o$ must be excluded in order to achieve restricted imformativity. For achieving least restricted informativity, namely computing $_{sup}$ $\overline{\ }$ (as in (16)), such exclusion (of strings entering $\#^1\&\ ^o$) must be done in a minimally intrusive manner.

We propose to compute $_{sup}$ in (16) based on the structure of the data-driven automaton $\hat{}$. Thus intuitively, the abovementioned string exclusion amounts to `avoiding' the subset $\#^1\&\ ^o$ of states in the data-driven automaton $\hat{}$. In order to obtain the largest $_{sup}$ $\overline{\ }$, the `avoidance' of $\#^1\&\ ^o$ should be performed as `close' as possible to $\#^1\&\ ^o$ and by means of removing a controllable event. This strategy is similar to that of computing the supremal controllable sublanguage in the standard model-based supervisory control theory. In view of this, we craft a supervisory control problem based on a modi ed structure of the data-driven automaton with the

speci cation of avoiding # ¹& º, and solve this problem by the standard supervisory synthesis algorithm in order to obtain $_{sup}$ in (16).

Speci cally, we de ne the following relevant automata. Given an event set , a control speci cation , and nite data sets – , construct the corresponding data-driven automaton ^¹ – – º = ¹&^ – – X^ – @ₑº with state subsets & (corresponding to = ¯ \ ) and & (corresponding to) as in De nition 4. First, construct a subautomaton of the data-driven automaton ^ by removing & and the corresponding transitions:

$$:= ¹& \; – \; – X – @_ₑº \tag{19}$$

where & := &^ n & , and X := X^ n f¹@ – f ! @º j @ 2 & or @º 2 & g. This automaton will serve as the plant in our supervisory control problem. Note that ¹ º = ¯. Next, let &~ := & n & and construct a subautomaton (of by removing # ¹& º [&~ and the corresponding transitions:

$$(= ¹&_(– – X_(– @_ₑº \tag{20}$$

where &$_($ = & n ¹# ¹& º [&~º = & n # ¹& º and X$_($:= X n f¹@ – f ! @º j @ 2 # ¹& º [&~ or @º 2 # ¹& º [&~g. This automaton (will serve as the speci cation in our supervisory control problem. Note that since &$_($ = & n # ¹& º, every state in @ 2 &$_($ satis es the negation of (17):

$$¹8f \; 2 \; _Dº X^¹@ – f! \text{ and } X^¹@ – f 2 & \; [\; & • \tag{21}$$

The plant in (19) and the speci cation (in (20) constitute our de ned supervisory control problem. We apply the standard supervisory control synthesis algorithm to compute the maximally permissive supervisor, which is also an automaton (indeed in this case a subautomaton of ( )

$$\% = ¹&_\% – – X_\% – @_ₑº \tag{22}$$

where &$_\%$ &$_($ and X$_\%$ X$_($ such that

$$! ¹\% º = \sup \; ¹! ¹(ºº$$

where

$$¹! ¹(ºº = f \; ! ¹(º j ¯ \; _D \backslash \; ! ¹ \; º \; ¯ g• \tag{23}$$

We summarize the above procedure in the form of an algorithm below. Our main result (Theorem 3 below) is that the resulting $\mathcal{L}(\mathcal{S}^{\uparrow})$ is exactly the largest L_{sup} in (16) we are after.

Algorithm 4 L_{sup} for least restricted informativity

Input: event set $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc}$, finite sets E, M, control specification $K = \overline{K} \cap L_m(G)$

Ensure: $\mathcal{L}(\mathcal{S}^{\uparrow}) = L_{sup}$

1: construct a data-driven automaton $\hat{A} = (E \times M, \Sigma, \hat{\delta}, (\varepsilon, \varepsilon))$ and $L_m(\hat{A}) = L_m(G) \cap K$ (as in Definition 4)

2: construct a subautomaton A of $\hat{A}$ as in (19)

3: compute non-informative state set $Q_{ni} \subseteq Q$ by Algorithm 3

4: construct a subautomaton $\mathcal{S}$ of A as in (20)

5: compute the supervisor $\mathcal{S}^{\uparrow}$ by standard supervisory synthesis algorithm as in (22)

6: return $\mathcal{L}(\mathcal{S}^{\uparrow}) = L_{sup}$

Theorem 3. The language $\mathcal{L}(\mathcal{S}^{\uparrow}) = L_{sup}$ in (16).

Proof. We make the following key claim:

$$\mathcal{L}(\mathcal{S}^{\uparrow}) = \sup\{L \subseteq K \mid L \text{ is controllable and } L \in \mathcal{L}(\mathcal{S})\}. \quad (23)$$

Namely the family $\mathcal{L}(\mathcal{S}^{\uparrow})$ in (23) of all controllable sublanguages of $\mathcal{L}(\mathcal{S})$ is exactly the same as the family $\mathcal{L}(\mathcal{S}^{\uparrow})$ in (15) of all subsets of $\overline{K}$ for which $L \cap L_m(G)$ is restricted informative. Under this claim, we derive

$$\mathcal{L}(\mathcal{S}^{\uparrow}) = \sup\{L \mid L \in \mathcal{L}(\mathcal{S}^{\uparrow})\} = \sup\{L \mid L \in \mathcal{L}(\mathcal{S})\} = \sup\{L \mid L \in \mathcal{L}(\mathcal{S}^{\uparrow})\} = L_{sup}.$$

The last equality is from (16), and our conclusion is established.

Now we prove the claim. First let $L \in \mathcal{L}(\mathcal{S}^{\uparrow})$, which means that $L \in \mathcal{L}(\mathcal{S})$ and $L \subseteq K \cap L_m(G)$. By the definition of $\mathcal{S}$ in (20), $L \subseteq K$ and thus $L \in \mathcal{L}(\mathcal{S}) \cap \overline{K}$, which in turn implies that $L \in \overline{K}$. In order to show that $L \in \mathcal{L}(\mathcal{S}^{\uparrow})$, we must prove that $L \cap L_m(G)$ is $\mathcal{S}$-informative. For this, let $s \in \overline{L}$ and $\sigma \in \Sigma_{uc}$. If $s\sigma \in L \cap L_m(G)$, then it follows from $L \in \overline{K} \cap L_m(G)$ that $s\sigma \in \overline{K}$. On the other hand if $s\sigma \notin L \cap L_m(G)$, we derive $s\sigma \notin \overline{L}$. To see this, consider the state q_s which is in $\mathcal{S}(\varepsilon) = \mathcal{S}(\hat{\delta}(\varepsilon, s))$. This means that (21) holds for q_s. But $\hat{\delta}(q_s, \sigma) \notin \mathcal{S}$ cannot be in $\mathcal{S}$ since $s\sigma \notin L \cap L_m(G) = \overline{L} \cap K$. Thus it is only possible that $\hat{\delta}(q_s, \sigma) \in \mathcal{S}$,

which implies that $Bf \in$. To summarize, for an arbitrary $B \in \overline{\ }$ and an arbitrary $f \in D$, we have $Bf \in \overline{\ } [$, which means that $\ - \ o$ is -informative. Therefore $\in I^{1} \ o$, and $^{1}! ^{1}(\ oo \subseteq I^{1} \ o$.

It remains to prove that $I^{1} \ o \subseteq \ ^{1}! ^{1}(\ oo$. Let $\in I^{1} \ o$, which means that $\overline{\overline{\ }}$ and $^{1} - \ o$ is -informative. Consider an arbitrary string $B \in \overline{\overline{\ }}$. Since $^{1} - \ o$ is -informative, the state $@_B \in \ \& \ n\#^{1}\& \ o = \&_($. Hence $B \in \ ! ^{1}(\ o$, and in turn we have $\ ! ^{1}(\ o$. To show that $\in \ ^{1}! ^{1}(\ oo$, we need to establish that is controllable wrt. . To this end, let $B \in \overline{\ }$, $f \in D$, and assume that $Bf \in \ ! ^{1} \ o = \overline{\ }$. It follows again from -informativity of $^{1} - \ o$ that $Bf \in \overline{\ } [$. But we have assumed that $Bf \in \overline{\ }$, so it is impossible that $Bf \in$ (by $\overline{\ } \setminus \ = \ ;$). Hence $Bf \in \overline{\ }$, which means that is controllable wrt. . Therefore $\in \ ^{1}! ^{1}(\ oo$, and $I^{1} \ o \subseteq \ ^{1}! ^{1}(\ oo$.

In view of the above, we have established our claim that $\ o = \ ^{1}! ^{1}(\ oo$, and the proof is now completed.

By Theorem 3, Algorithm 4 correctly computes the largest subset of $\overline{\ }$ for which the data pair $^{1} - \ o$ is least restricted informative. If the output $^{1}\% \ o = \ _{\sup} < \ ;$, then $^{1} - \ o$ is $_{\sup}$-informative, so $^{1} - \ o$ is informatizable. On the other hand, if $^{1}\% \ o = \ _{\sup} = \ ;$, then no nonempty subset exists for which $^{1} - \ o$ is restricted informative, which means that $^{1} - \ o$ is not informatizable. Hence, the nonemptiness of Algorithm 4's output $^{1}\% \ o$ is equivalent to informatizability of $^{1} - \ o$. In this sense, Algorithm 4 additionally serves as an alternative test for informatizability to Algorithm 2. Nevertheless, since Algorithm 4 has higher complexity (which is provided in Remark 5 below) than Algorithm 2, if the purpose is solely to check informatizability, Algorithm 2 is more e cient and thus advantageous.

Remark 5. We analyze the complexity of Algorithm 4. As in Remark 3, let $\ = \ j \ [\ j$ be the total number of strings in $[$, and $"$ be the length of the longest string in $[$. Then the complexity of line 1 of constructing the data-driven automaton $\hat{\ }$ together with $\&$ and $\&$ is $\$ ^{1}\#" \ ^{2}o$ (as analyzed in Remark 3). Lines 2 and 4 are creating subautomata of $\hat{\ }$ by removing certain states and transitions, so their repsective complexities are no more than $\$ ^{1}\#" \ ^{2}o$. Line 3 is the execution of Algorithm 3, whose complexity is the same as Algorithm 1, namely $\$ ^{1}\#" \ ^{2}o$. Finally, line 5 applies the standard supervisor synthesis algorithm [2], [5], whose complexity in this case where the speci cation is a subautomaton of the plant is $\$ ^{1}j\&_(j^{2}j \ j^{o}$. Since $j\&_(j \ \ j \ \&j$, the complexity of line 5 is $\$ ^{1}j\&\hat{j}^{2}j \ j^{o} = \$ ^{1}\#^{2}" \ ^{4}j \ j^{o}$. This

Figure 7 Data-driven automaton $\hat{}_3$ corresponding to $(_3, _3)$

complexity is also the complexity of Algorithm 4.

Finally we provide an illustrative example for Algorithm 4.

Example 9. Suppose that the event set and the specification is as same as Example 3. Consider a pair of finite data sets $(_3, _3)$, where

$$_3 = \{01315, 02115, 0124\}$$

$$_3 = \{3, 03, 01313, 023, 0213\}$$

Then the control specification in (8) is

$$_3 = \overline{_3} \setminus = \{01315, 02115\}$$

The data-driven automaton $\hat{}_3$ corresponding to the data set $(_3, _3)$ is shown in Fig. 7. By applying Algorithm 1 and Algorithm 2 to $\hat{}_3$, it is determined that $(_3, _3)$ is not informative but is informatizable for . This implies that there exists a nonempty sublanguage of $\overline{_3}$ for which $(_3, _3)$ is limited informative. Next we apply Algorithm 4 to compute the largest such sublanguage.

Line 1 in Algorithm 4 of constructing the data-driven automaton $\hat{}_3$ has been done, as displayed in Fig. 7. Line 2 computes the subautomaton $_3$ by removing the states in $\&$ from $\hat{}_3$ and the relevant transitions; the result is displayed in Fig. 8. Line 3 applies Algorithm 3 to derive the set of non-informative states: $\#_1\& = \{q_{013}, q_{01315}, q_{0215}\}$. The reason why these three states are non-informative is because the uncontrollable event is not defined at any of these states. Line 4 constructs the subautomaton $(_3, _3)$ by removing the three states in $\#_1\&$ and the two states in $\&$ (white color coded) from $_3$ including the relevant transitions; the

Figure 8 Subautomaton $_3$ by line 2 of Algorithm 4

Figure 9 Subautomaton ($_3$ by line 4 of Algorithm 4

result is displayed in Fig. 9. Finally line 5 compute the maximally permissive supervisor % $_3$ by treating $_3$ as the plant and ($_3$ as the speci caiton; the result is in Fig. 10. Comparing (to $_3$, not only the nonreachable state @$_{131}$ but also the reachable state @$_1$ are removed. The removal of @$_1$ is because the string 01 violates the controllability condition (as 0132 ! ^{1} $_3$o n ! ^{1}($_3$o and 3 is uncontrollable). Hence the nal result is

$$! ^1\% _3 o = f n – 0 – 02 – 021 g$$

which is the largest subset of $\overline{}_3$ for which the data pair ^{1} $_3$ – $_3$o is least restricted informative.

Based on $_{sup}$ = ! ^{1}% $_3$, we can construct the corresponding supervisor +$_{sup}$: $\overline{}_3$! %FA $_2$o such that ! ^{1}+$_{sup}$• o = $_{sup}$ for every plant consistent with ^{1} $_3$ – $_3$o as follows:

$$+_{sup}{}^1 B^o = \begin{cases} f 1 – 2 – 4 – 5 g & \text{if } B = n – \\ f 0 – 1 – 4 – 5 g & \text{if } B = 0 – \\ f 0 – 2 – 4 – 5 g & \text{if } B = 02 – \\ f 0 – 1 – 2 – 4 – 5 g & \text{if } B = 021 – \\ ; & \text{if } B 2 \overline{}_3 n \ _{sup} • \end{cases}$$

Figure 10 Supervisor V_3 by line 5 of Algorithm 4

VI. Concluding Remarks

In this paper we have initiated the first systematic study on data-driven supervisory control of DES. We have proposed new concepts of data-informativity, informatizability, and least restricted informativity, as well as developed the corresponding verification and synthesis algorithms based on a novel structure of data-driven automaton. The recipe of using these concepts/algorithms is summarized below: For a given data pair (L_1, L_2) and a specification K, first apply Algorithm 1 to verify if (L_1, L_2) is informative for K. If yes, we can build a supervisor to enforce $L_m = \overline{K} \cap L_m$. If no, we apply Algorithm 2 to verify if (L_1, L_2) is informatizable for K. If yes, we further apply Algorithm 4 (in which Algorithm 3 is used) to compute the largest subset $\overline{K}$ of K for which (L_1, L_2) is least restricted informative. If Algorithm 2 returns no, then there is no supervisor that can be built to enforce any subset $\overline{K}$ of K. In this case, one may consider to obtain more data L_1 and L_2.

As a first step into a data-driven approach for supervisory control, many interesting questions are left unanswered and await for future research. For example, if data on marking behavior of the unknown plant are available, can we strengthen the current theory by always guranteeing nonblocking supervisory control? If the means of control is not only disabling controllable events, but also preempting uncontrollable events by forcing, can we relax the current requirement on having to assume that all uncontrollable events can occur unless the occurrences lead to K (this second question is recently investigated in [19]). More broadly, can we extend informativity from the basic controllability to other important properties such as observability, diaganosability, and opacity [20]. Finally, we aim to apply the developed data-driven approach to real applications with large data sets, including benchmark scenarios of autonomous driving in uncertain urban environment, robotic exploration of unknown terrains, and human-machine interactions.

References

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

[2] W. M. Wonham and P. J. Ramadge, On the supremal controllable sublanguage of a given language, SIAM Journal on Control and Optimization, vol. 25, no. 3, pp. 637 659, 1987.

[3] W. M. Wonham, K. Cai, and K. Rudie, Supervisory control of discrete-event systems: A brief history, Annual Reviews in Control, vol. 45, pp. 250 256, 2018.

[4] K. Cai and W. M. Wonham, Supervisory control of discrete-event systems, Encyclopedia of Systems and Control, pp. 1 9, 2019.

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

[6] 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.

[7] 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.

[8] M. Gevers, A. S. Bazanella, X. Bombois, and L. Miskovic, Identi cation and the information matrix: How to get just su ciently rich?, IEEE Transactions on Automatic Control, vol. 54, no. 12, pp. 2828 2840, 2009.

[9] 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.

[10] 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.

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

[12] M. Konishi, T. Sasaki, and K. Cai, E cient safe control via deep reinforcement learning and supervisory control case study on multi-robot warehouse automation, Proc. IFAC WODES, 2022.

[13] 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.

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

[15] 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.

[16] H. J. Van Waarde, J. Eising, M. K. Camlibel, and H. L. Trentelman, The informativity approach: To data-driven analysis and control, IEEE Control Systems Magazine, vol. 43, no. 6, pp. 32 66, 2023.

[17] T. Ohtsuka, K. Cai, and K. Kashima, Data-informativity for data-driven supervisory control of discrete-event systems, in 2023 62nd IEEE Conference on Decision and Control (CDC), pp. 6923 6928, 2023.

[18] 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.

[19] C. Gu, C. Gao, and K. Cai, Data-driven supervisory control of discrete-event systems with forcible events, Proc. IFAC WODES, 2024.

[20] C. Hadjicostis, Estimation and Inference in Discrete-Event Systems, Communications and Control Engineering, Springer, 2020.