Data-Driven Supervisory Control of Discrete-Event Systems With Forcible Events

Chao Gu, *Member, IEEE*, Chao Gao, *Member, IEEE*, and Kai Cai, *Senior Member, IEEE*

***Abstract*— This paper studies the problem of data-driven analysis and control design for model-unknown discrete-event systems with *forcible events*. Using data, i.e., a subset of possible event sequences generated by the model-unknown plant and a subset of impossible behaviors of the system based on pre-established knowledge, we leverage the *event-forcing* mechanism to verify an extended concept of controllability, namely *forcible-controllability*, and develop data-driven control strategies to enforce a given specification. In particular, a new property is proposed for the given data, called *forcible-informativity*, which represents a criterion for determining forcible-controllability of the specification based on the data. If this property holds, data-driven forcing supervisory control is designed accordingly. Moreover, for data that fail to satisfy forcible-informativity, we introduce the novel notion of *forcible-informatizability*, which evaluates the potential of the data in identifying forcible-controllability of a smaller (but non-empty) specification. For the verification of forcible-informatizability, a new concept called *barrier language* is introduced. Based on barrier languages, a necessary and sufficient condition that characterizes forcible-informatizability is provided, and a corresponding verification algorithm is developed.**

***Index Terms*— Discrete-event system, supervisory control, data-driven control, forcible event.**

I. Introduction

DISCRETE-event systems (DES) are event-driven systems with discrete state space. The traditional approach to analyzing and controlling DES relies on model-based methods [1], [2]. While these methods have been successful in various applications, they are ineffective if DES models are unknown. Recently, *data-driven* methods have emerged as promising alternatives to potentially address model-unknown DES [3]–[5].

Data-driven analysis and control [6] involves utilizing data sets obtained from system observations or event logs to gain insights into system behavior, identify patterns, and develop effective control strategies. The concept of *data informativity*, proposed in [7] for data-driven analysis and control of continuous-time systems, extends and complements previous work in *system identification* [8], [9]. In data-driven analysis, informativity is essential to infer system properties from data. For data-driven control, meeting informativity criteria enables controller design using data alone. The authors in [10] develop a counterpart of this concept in the DES setting.

The study in [10] introduces a method for verifying the controllability of a given specification imposed on a model-unknown DES. This method is data-driven, as it only assumes two sets of data available. The first set D is a subset of possible behaviors generated by the model-unknown plant. The second set D^- is a subset of impossible behaviors of the system (either in contradiction to physical principles or based on preliminary knowledge). For instance, in a manufacturing system, an operation cannot begin before it has been initiated; a final product cannot be synthesized without first using its essential raw materials; a robot cannot take an item before it reaches an item supply location. Furthermore, the concept of *data informativity* is introduced from a DES perspective to evaluate whether the data sets D and D^- are sufficient to verify the controllability of the given specification. The data provided may not always be sufficiently informative for the given specification. To address this, the concept of *data informatizability* is introduced in [11] as a key property of the data sets D and D^- in determining the controllability of a smaller but non-empty specification.

Still, data informativity in [10], [11] can impose strict requirements on the quality of the data sets D and D^-, especially when dealing with systems with many uncontrollable events; roughly speaking, much information about uncontrollable events needs to be included in either D or D^-. A concrete example to illustrate this point is provided in Section III below. If neither informativity nor informatizability is satisfied, it is infeasible [10], [11] to design a valid data-driven supervisor whose control actions are (the conventional) enablement and disablement of controllable events. In view of this, we are motivated to ask the following question: *Can we augment new control actions/mechanisms to the supervisor, so that without changing the available data sets a valid data-driven supervisor can be designed?* More specifically, in this paper we consider augmenting the conventional supervisor with *event-forcing* actions (in additional to enabling/disabling actions), and thereby investigate new properties of the data sets under which a valid data-driven supervisor can be feasibly designed.

Forcible events and event-forcing mechanism have been well studied in the literature of model-based supervisory control. The event-forcing mechanism allows certain prescribed

This work was supported in part by China Scholarship Council Grant no. 202306960070; and in part by the Fundamental Research Funds for the Central Universities Grant no. XJSJ24025.

Chao Gu is with School of Electro-Mechanical Engineering, Xidian University, Xi'an, China (e-mail: guchao@xidian.edu.cn).

Chao Gao is with Laboratoire GREAH, University Le Havre Normandie, Le Havre, France (e-mail: chao.gao@univ-lehavre.fr).

Kai Cai is with Department of Core Informatics, Osaka Metropolitan University, Osaka, Japan (e-mail: cai@omu.ac.jp).

forcible events to *preempt* the occurrence of other events; namely these events can be forced to occur before other events. Consider, for example, controlling a robot to cross a road while a car is approaching (the latter being uncontrollable from the perspective of the robot). In this situation, a forcible event can be performed to quickly maneuver the robot to move either forward or backward in order to avoid collision with the approaching car (i.e., outpacing the car and thereby achieving safety). Event-forcing mechanism has typically been used in the supervisory control of timed DES [12]–[17] and is typically used to preempt the *tick event*. Event-forcing mechanism has also been introduced and studied in supervisory control of untimed DES [2], [18]–[23]. A recent work [24] introduces a key property called *forcible-controllability*, which characterizes the existence of a valid supervisor to enforce an imposed specification. This concept extends traditional controllability by emphasizing the relationship between the controllability of a supervisor and uncontrollable events in the event-forcing setting. In this paper we aim to paradigm-shift this forcible-controllability to the data-driven (model-unknown) supervisory control, and thereby develop effective new properties and methods for data-driven supervisory control design by harnessing the power of the event-forcing mechanism. The contributions of this paper are stated below.

First, for the given data sets D, D^- and the specification language E, by integrating the event-forcing mechanism, we introduce a new property of *forcible-informativity*, which is more general than informativity in [10]. Forcible-informativity provides a criterion of the data quality for determining the forcible-controllability of the specification E with respect to all plants *consistent* with the data, i.e., plants that can generate all behaviors in D but cannot generate any behaviors in D^-. Therefore whenever the provided data pair (D, D^-) is forcibly-informative for the specification E, a data-driven supervisor can be designed to enforce the specification.

Second, we propose a new necessary and sufficient condition that characterizes forcible-informativity. This condition also extends the counterpart condition in [10] to the event-forcing setting. We further develop an algorithm using a structure, called *data-driven automaton*, to algorithmically verify forcible-informativity. In addition, a data-driven supervisory control strategy with event-forcing actions is designed whenever this property is verified to hold.

Third, if the data pair (D, D^-) is not forcibly-informative, we introduce the new notion of *forcible-informatizability* to describe the ability to determine forcible-controllability for some smaller (but non-empty) sublanguage of the specification (with respect to all plants consistent with the data pair). This concept of forcible-informatizability generalizes informatizability in [11] to the event-forcing setting.

Finally, to characterize forcible-informatizability, it turns out to be technically challenging and there is no existing method in the literature (model-based or model-unknown) for this purpose. To overcome the challenge, we introduce a novel concept called *barrier language*. Roughly speaking, the barrier language defines a "safe region" that can be kept forcibly informative by using appropriate forcible events. Thus, the "boundary of this region" is as if having *barriers* so that all system evolutions inside the region will remain inside. Based on the barrier language, a necessary and sufficient condition for characterizing forcible-informatizability is presented, along with an algorithm for verification of the condition.

A preliminary result on forcible-informativity was reported in the conference precursor [25]. This paper substantially extends [25] by introducing new concepts of forcible-informatizability, barrier languages, as well as presenting necessary and sufficient conditions that characterize forcible-informatizability (which is both conceptually and technically novel). Based on these conditions, moreover, a verification algorithm is developed to determine forcible-informatizability. In addition, compared to [25], we have substantially enhanced the presentation by including motivating examples, graphical illustrations, and new examples.

The rest of this paper is organized as follows. Section II reviews the preliminaries used throughout the paper. Section III introduces the data-driven setting of DES, distinguishing it from the model-based setting, and presents the motivation of the paper. Section IV defines forcible-informativity, while Section V introduces forcible-informatizability. Necessary and sufficient conditions are given, as well as algorithms for verifying these two properties. Conclusions are drawn in Section VI.

II. Preliminaries

A. Basics of supervisory control theory

A finite-state automaton is a five-tuple $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m)$, where Q is the finite state set, Σ is the finite event set, $\delta : Q \times \Sigma \to Q$ is the partial state transition function[1], $q_0 \in Q$ is the initial state, and $Q_m \subseteq Q$ is the marker state set. A string s on Σ is a finite sequence of events from Σ. Let ϵ be the empty string and Σ^* be the set of all strings on Σ along with ϵ. Extend the transition function as $\delta : Q \times \Sigma^* \to Q$ and write $\delta(q, s)!$ to mean that $s \in \Sigma^*$ is defined at $q \in Q$. $L(\mathbf{G}) = \{s \in \Sigma^* \mid \delta(q_0, s)!\}$ is defined as the language generated by $\mathbf{G}$, while $L_m(\mathbf{G}) = \{s \in L(\mathbf{G}) \mid \delta(q_0, s) \in Q_m\}$ is the marked language of $\mathbf{G}$. For simplicity, in this paper we assume that in $\mathbf{G}$ all states are marked, i.e., $L(\mathbf{G}) = L_m(\mathbf{G})$ holds. Given two languages $L_1, L_2 \in \Sigma^*$, the language concatenation of L_1 and L_2, is denoted and defined as

$$L_1 \cdot L_2 = \{s \in \Sigma^* \mid s = s_1 \cdot s_2, s_1 \in L_1, s_2 \in L_2\}.$$

String s' is a prefix of string s, written $s' \in \overline{s}$, if there exists $s'' \in \Sigma^*$ such that $s's'' = s$. The prefix closure of a language L is denoted and defined as $\overline{L} = \{s \in \Sigma^* \mid (\exists s' \in \Sigma^*)ss' \in L\}$. The *length* $|s|$ of a string $s \in \Sigma^*$ is defined according to $|\epsilon| = 0; |s| = k$, if $s = \sigma_1 \cdots \sigma_k$. Define $\text{max\_len}(L) = \{|s| \mid s \in L \wedge [(\nexists s' \in L)|s'| > |s|]\}$ as the maximal length of strings in L.

For control, the event set Σ of $\mathbf{G}$ is partitioned as $\Sigma = \Sigma_c \dot{\cup} \Sigma_u$, where Σ_c and Σ_u are the sets of controllable and uncontrollable events, respectively. A *supervisor* (i.e., a control agent) can only enable/disable controllable events in Σ_c.

[1]Alternatively, δ can also be characterized as set $\delta = \{(q_1, \sigma) \to q_2 \mid (\sigma \in \Sigma) \wedge (q_1, q_2 \in Q)\}$.

A supervisory control V for $\mathbf{G}$ is a mapping $V : L(\mathbf{G}) \to \Gamma$, where $\Gamma = \{\gamma \mid \Sigma_u \subseteq \gamma \subseteq \Gamma\}$ is the set of all control patterns. That is to say, for any string $s \in L(\mathbf{G})$, V associates s with a control pattern $V(s) \in \Gamma$. Physically, all (controllable) events not in $V(s)$ will be disabled by the supervisory control V. Write $V/\mathbf{G}$ for the closed-loop system, representing that $\mathbf{G}$ is under the control of V. The language of the closed-loop system $L(V/\mathbf{G})$ is defined as follows:

- $\epsilon \in L(V/\mathbf{G})$;
- if $s \in L(V/\mathbf{G}), \sigma \in V(s)$, and $s\sigma \in L(\mathbf{G})$, then $s\sigma \in L(V/\mathbf{G})$;
- no other strings belong to $L(V/\mathbf{G})$.

From the above, $L(V/\mathbf{G}) \subseteq L(\mathbf{G})$ holds, which includes all strings in $L(\mathbf{G})$ that are not disabled under supervisory control V. A control specification $K \subseteq L(\mathbf{G})$ is a language (normally non-empty) that we intend to impose on $\mathbf{G}$ in order to achieve a specific control goal.

Definition 1 [controllability] *Let $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m)$ be a plant and $K \subseteq L(\mathbf{G})$ be a specification language. K is controllable if $\overline{K}\Sigma_{uc} \cap L(\mathbf{G}) \subseteq \overline{K}$, i.e., $(\forall s \in \Sigma^*)(\forall \sigma \in \Sigma) s \in \overline{K}, \sigma \in \Sigma_u, s\sigma \in L(\mathbf{G}) \Rightarrow s\sigma \in \overline{K}$.* □

As a basic concept, the controllability of a given specification characterizes the existence of a supervisory control synthesizing the specification, as shown in the following proposition.

Proposition 1 *There exists a supervisory control V for enforcing specification $K \subseteq L(\mathbf{G})$ on plant $\mathbf{G}$, i.e., $L(V/\mathbf{G}) = \overline{K}$, if and only if K is controllable with respect to $\mathbf{G}$.* □

For a controllable specification $K \subseteq L(\mathbf{G})$, the supervisory control $V : L(\mathbf{G}) \to 2^{\Sigma}$ (here 2^{Σ} denotes the power set of Σ) such that $L(V/\mathbf{G}) = \overline{K}$ can be defined as follows:

$$V(s) := \begin{cases} \Sigma_u \cup \{\sigma \in \Sigma_c \mid s\sigma \in \overline{K}\}, & \text{if } s \in \overline{K}; \\ \Sigma_u, & \text{if } s \in L(\mathbf{G}) \setminus \overline{K}. \end{cases}$$

Write $\mathcal{C}(K)$ for the family of all controllable sublanguages of the specification $K \subseteq L(\mathbf{G})$, i.e.,

$$\mathcal{C}(K) = \{K' \subseteq K \mid \overline{K'}\Sigma_u \cap L(\mathbf{G}) \subseteq \overline{K'}\}.$$

$\mathcal{C}(K)$ is closed under set union, so it contains a unique supremal element

$$\sup\mathcal{C}(K) = \bigcup_{K' \in \mathcal{C}(K)} K'.$$

If $\sup\mathcal{C}(K) \neq \emptyset$, there exists a non-empty maximally permissive supervisor to enforce $\sup\mathcal{C}(K)$, which can be synthesized as detailed in [2, section 3].

B. Event-forcing mechanism

Given a set of forcible events $\Sigma_{for} \subseteq \Sigma$, a forcible event $f \in \Sigma_{for}$, if it occurs, preempts the occurrence of other events. The concept of *forcible-controllability* is introduced in [24], which extends controllability, showing as follows.

Definition 2 [forcible-controllability] *Consider a plant $\mathbf{G}$ with $\Sigma = \Sigma_c \dot{\cup} \Sigma_u$ and $\Sigma_{for} \subseteq \Sigma$. A control specification $K \subseteq L(\mathbf{G})$ is forcibly-controllable with respect to $\mathbf{G}$ if:*

$$(\forall s \in \overline{K}, \forall \sigma \in \Sigma_u)\ s\sigma \in L(\mathbf{G}) \Rightarrow [s\sigma \in \overline{K}] \vee$$

$$[((\exists f \in \Sigma_{for}) sf \in \overline{K}) \wedge ((\forall \sigma' \in \Sigma \setminus \Sigma_{for}) s\sigma' \notin \overline{K})].$$ □

By Definition 2, forcible-controllability implies that either the specification K is controllable or there exists a forcible event $f \in \Sigma_{for}$ that preempts the occurrence of all non-forcible events (which violates controllability) in order to keep K invariant. If a specification is controllable, it is also forcibly-controllable. Controllability involves only the ability to disable events, whereas forcible-controllability includes both disabling and forcing. In other words, forcible-controllability allows potential violations of controllability to be preempted by the use of appropriate forcible events.

Proposition 2 *There exists a supervisory control V_{for} for enforcing specification $K \subseteq L(\mathbf{G})$ on plant $\mathbf{G}$, i.e., $L(V_{for}/\mathbf{G}) = \overline{K}$ if and only if K is forcibly-controllable.* □

Proposition 2 shows that the forcible-controllability of K ensures the existence of a supervisory control for enforcing K. Meanwhile, the work in [24] provides a detailed characterization of the supervisory control $V_{for} : L(\mathbf{G}) \to 2^{\Sigma}$, which is shown below:

$$V_{for}(s) := \begin{cases} \Sigma_u \cup \{\sigma \in \Sigma_c \mid s\sigma \in \overline{K}\}, \\ \quad \text{if } (\forall \sigma \in \Sigma_u)\ s\sigma \in L(\mathbf{G}) \Rightarrow [s\sigma \in \overline{K}]; \\ \{f \in \Sigma_{for} \mid sf \in \overline{K}\}, \\ \quad \text{if } [(\exists f \in \Sigma_{for}) sf \in \overline{K}] \wedge \\ \quad\ [(\exists \sigma' \in \Sigma_u)\ s\sigma' \in L(\mathbf{G}) \wedge s\sigma' \notin \overline{K}]. \end{cases}$$

In particular, if controllability is already satisfied, no event-forcing is necessary. However, if controllability is violated, certain forcible events are used to preempt events that violate controllability.

Further, the set of *forcibly-controllable sublanguages* for the specification K is denoted as $\mathcal{F}(K) = \{K' \subseteq K \mid K' \text{ is forcibly-controllable with respect to } \mathbf{G}\}$. In [24], forcible-controllability is proved to be closed under union and thus the set $\mathcal{F}(K)$ contains a unique supremal element

$$\sup\mathcal{F}(K) = \bigcup_{K' \in \mathcal{F}(K)} K'.$$

If $\sup\mathcal{F}(K) \neq \emptyset$, there exists a non-empty maximally permissive supervisor to enforce $\sup\mathcal{F}(K)$, which can be synthesized as detailed in [24, Algorithm 1].

III. Data-driven DES: Framework and Motivation

This section introduces the data-driven framework for DES, reviews related work, and presents our problem statement for data-driven DES with event-forcing mechanism. First, we present the following assumption.

Assumption 1 *Consider a plant $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m)$ with $\Sigma = \Sigma_c \dot{\cup} \Sigma_u$ and a specification (regular) language $E \subseteq \Sigma^*$.*

Assume that the dynamic structure of $\mathbf{G}$ is unknown, but: 1) part of $\mathbf{G}$'s behavior is known, which is denoted as a finite data set $D \subseteq L(\mathbf{G})$; 2) another (possibly infinite and regular) data set $D^- \subseteq \Sigma^*$ is also known, consisting of strings that can never happen in $\mathbf{G}$, i.e., $D^- \subseteq \Sigma^* \setminus L(\mathbf{G})$; 3) $\Sigma = \Sigma_c \dot{\cup} \Sigma_u$ is known. □

The data set D can be collected through logging or observing the unknown plant, while D^- can be identified in practical scenarios based on prior knowledge, such as sequences that contradict physical laws or specified requirements. Both data sets are considered fixed. Note that from $D \subseteq L(\mathbf{G})$ and $D^- \subseteq \Sigma^* \setminus L(\mathbf{G})$, we have that $D \cap D^- = \emptyset$. Compared with the data-driven setting in [10], Assumption 1 is the same except in 2) D^- is allowed to be infinite, whereas only finite D^- is considered in [10].

The data sets D, D^- may correspond to multiple plant models that can generate the behaviors in D and no behavior in D^-. On this basis, the concept of *consistency* is proposed in [10], i.e., a plant $\mathbf{G}$ is consistent with the data pair (D, D^-) if $\overline{D} \subseteq L(\mathbf{G})$ and $D^- \cap L(\mathbf{G}) = \emptyset$. For a given pair of data (D, D^-), there can be infinitely many plant models consistent with the data pair; the true (but unknown) plant $\mathbf{G}$ is one among them. In addition, the specification with regard to the data is denoted as

$$D_E = \overline{D} \cap E. \tag{1}$$

In [10], under Assumption 1 (but only for finite D^-), a key property called *informativity* is introduced. Specifically, a finite data pair (D, D^-) is informative for a specification E if D_E in (1) is non-empty and controllable with respect to all plants consistent with (D, D^-). Data informativity determines whether a supervisor (with only disabling/enabling mechanism) exists to enforce D_E for all plants consistent with the data, including the unknown true plant. However, ensuring informativity by only disabling/enabling controllable events can impose stringent requirements on the quality of the data pair (D, D^-), as illustrated by the example below.

Example 1 *Consider a model-unknown plant $\mathbf{G}$ that represents a robot navigation scenario. The task of the robot is to explore an unknown environment and collect data of interest. Let $\Sigma = \{a, b, c, d, e, f\}$ where each event represents a movement of the robot. Consider $\Sigma_c = \{a, b, d, e, f\}$ and $\Sigma_u = \{c\}$. Here controllable events a, b, d, e, f are intended movements of the robot, while uncontrollable event c is an unintended movements caused possibly by disturbances in the unknown environment. Suppose we observe a string $dfcd$, which corresponds to a trajectory of movement for the robot from the initial state to a target state. Thus, we have $D = \{dfcd\}$. Additionally, suppose that we have prior knowledge of the plant that: 1) the robot cannot initiate movement a from the initial state; and 2) it is impossible for the robot to initiate movements b or c immediately after completing a sequence that leads to a target state. Accordingly, we define the set $D^- = \{a, dfcdb, dfcdc\}$. Note that there are infinitely many plants consistent with the data (D, D^-); two of these plants, denoted as $\mathbf{G_1}$ and $\mathbf{G_2}$, are shown in Figs. 1 and 2.*

Suppose the specification $E = \{dfcd, dfd\}$, defining two desired navigation routes to target states. We have from (1) that $D_E = \overline{D} \cap E = \{dfcd\}$. Next, we verify the informativity of the data (D, D^-) for E. Consider a string $\epsilon \in \overline{D_E}$ and an uncontrollable event $c \in \Sigma_u$. Since $\epsilon \cdot c = c \notin \overline{D_E}$ and it is uncertain whether $c \in L(\mathbf{G})$ (as $c \notin D^-$), there is no guarantee, by Definition 1, that D_E is controllable for any plant consistent with (D, D^-). For instance, D_E is not controllable for plant $\mathbf{G}_2$ in Fig. 2, as $c \in L(\mathbf{G}_2)$. Therefore, (D, D^-) is not informative for the specification E. The same conclusion can be drawn by analyzing the strings $d, dfc \in \overline{D_E}$.

This lack of informativity is caused by insufficient data (either D or D^-) about the possible behavior caused by the uncontrollable event c. Worse, if there exist many uncontrollable events, the requirement on the data D and D^- become more stringent, causing the property of informativity more difficult to be satisfied. □

Since data informativity is not satisfied in the above example, it is impossible to synthesize a supervisory control—based on the conventional enablement and disablement of controllable events—to enforce the non-empty specification D_E for the model-unknown plant using only the provided data. If the available data (D, D^-) cannot be changed, one may consider introducing additional control mechanisms to possibly alleviates the stringent requirement on data quality. One promising such candidate is the event-forcing mechanism, which allows the supervisor to preempt (any number of) uncontrollable events by forcing appropriate forcible events. In view of this, information about possibility or impossibility of occurrences of these uncontrollable events is not needed, thereby alleviating demands on data.

Example 2 *[Example 1 ext.] Consider again the robot navigation scenario in Example 1. Now assume that event-forcing mechanism is available and the set of forcible events is $\Sigma_{for} = \{d, f\}$. Consider again the string $\epsilon \in \overline{D_E}$. Since $\epsilon \cdot d = d \in \overline{D_E}$ and for any event $\sigma \in \Sigma \setminus \Sigma_{for}$, $\epsilon \cdot \sigma = \sigma \notin \overline{D_E}$, the forcible event d can be used to preempt the (potential)*

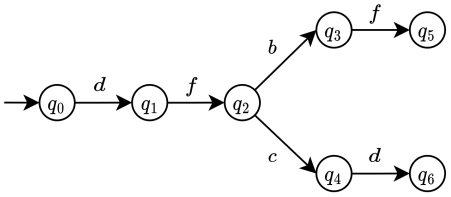


Fig. 1: A plant $\mathbf{G_1}$ consistent the data (D, D^-).

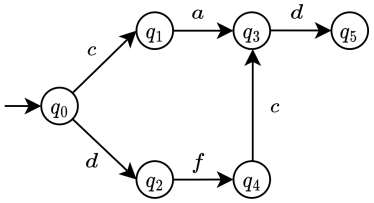


Fig. 2: A plant $\mathbf{G_2}$ consistent the data (D, D^-).

occurrence of any such σ (including the uncontrollable event c) in order to keep $\overline{D_E}$ invariant (as seen in plants $\mathbf{G_1}$ and $\mathbf{G_2}$). A similar argument applies to the strings d and dfc. However, for the strings df and $dfcd$, since $dfc \in \overline{D_E}$ and $dfcdc \in D^-$ (implying $dfcdc \notin L(\mathbf{G})$), controllability is not violated. Consequently, by Definition 2, we can infer that D_E is forcible-controllable for any plant consistent with (D, D^-), thanks to the introduced event-forcing mechanism. □

As shown in Example 2, even if a data pair is not informative [10], forcible events can be employed to ensure forcible-controllability (Definition 2) of the specification for any plant consistent with the data. This in turn can enable (data-driven) supervisor synthesis with event-forcing mechanism. Motivated by the above described potential usefulness of forcible events, we now present the following problem formulation.

Problem 1 *Consider a model-unknown plant $\mathbf{G}$ with a set of forcible events $\Sigma_{for} \subseteq \Sigma$. Assume that $D_E = \overline{D} \cap E$ in (1) is nonempty and Assumption 1 holds. Synthesize (if possible) a (data-driven) supervisory control $V_{D,for} : \overline{D} \to 2^{\Sigma}$ such that $L(V_{D,for}/\mathbf{G}) = \overline{D_E}$.* □

Problem 1 involves a subset of forcible events $\Sigma_{for} \subseteq \Sigma$, and focuses on the challenge of determining and constructing (if possible) a data-driven supervisory control using event-forcing mechanism. Unlike the assumptions in [10], [11], which require both data sets D and D^- to be finite, Assumption 1 in Problem 1 allows D^- to be infinite based on prior knowledge of the unknown plant model. This setting is practically relevant. For instance, in the robot navigation scenario from Example 1, suppose we know that the robot cannot initiate sequences consisting solely of consecutive f movements due to energy constraints. As a result, strings like $ff, fff, ffff \ldots$ (denoted as f^n for $n \in \{2, 3, 4 \ldots\}$) should be included in D^-, making D^- infinite. We will find a solution to Problem 1 in Section IV by introducing a new notion, namely *forcible-informativity*.

IV. Forcible-informativity and Data-driven Supervisory Control

To solve Problem 1, we propose a new concept of forcible-informativity, which refers to the ability to synthesize a forcing supervisory control that enforces the specification using only the available data. We then propose a structure, namely a data-driven automaton, which is built based on the data and is used to verify the forcible-informativity. If this property is satisfied, a data-driven supervisory control is designed to enforce the specification.

A. Forcible-informativity

In this part, we introduce forcible-informativity and give a necessary and sufficient condition for its characterization.

Definition 3 *Consider an event set $\Sigma = \Sigma_c \dot{\cup} \Sigma_u$ with $\Sigma_{for} \subseteq \Sigma$ and a specification language $E \subseteq \Sigma^*$. Given data sets $D, D^- \subseteq \Sigma^*$, the pair (D, D^-) is said to be forcibly-informative for E if the non-empty specification D_E is forcibly-controllable with respect to all plants $\mathbf{G}$ consistent with (D, D^-), i.e., there exists a supervisory control for $\mathbf{G}$ to enforce D_E.* □

Since forcible-controllability (see Definition 2) is a weaker property than controllability, forcible-informativity is also weaker than informativity [10]. Specifically, while informativity implies forcible-informativity, the reverse does not hold. The forcible-informativity of a data pair (D, D^-) for the specification E is closely linked to the forcible-controllability of the specification D_E. Forcible-informativity indicates that a supervisory control can be synthesized using an event-forcing mechanism to enforce the non-empty specification D_E for any plant consistent with (D, D^-), relying solely on the available data. A characterization condition for this property is provided below.

Proposition 3 *Consider an event set $\Sigma = \Sigma_c \dot{\cup} \Sigma_u$ with $\Sigma_{for} \subseteq \Sigma$ and a specification language $E \subseteq \Sigma^*$. Given $D, D^- \subseteq \Sigma^*$ with $D_E = \overline{D} \cap E$, the pair (D, D^-) is forcibly-informative for specification E if and only if the following holds:*

$$(\forall s \in \overline{D_E}, \forall \sigma \in \Sigma_u)$$
$$[s\sigma \in \overline{D_E} \cup D^-] \vee [((\exists f \in \Sigma_{for}) sf \in \overline{D_E}) \wedge$$
$$((\forall \sigma' \in \Sigma \setminus \Sigma_{for}) s\sigma' \notin \overline{D_E})].$$

Proof: (if) Consider an arbitrary plant $\mathbf{G}$ consistent with (D, D^-). Given an arbitrary string $s \in \overline{D_E}$, for any uncontrollable event $\sigma \in \Sigma_u$, if $s\sigma \in \overline{D_E}$, then the following holds:

$$s\sigma \in L(\mathbf{G}) \Rightarrow s\sigma \in \overline{E}.$$

If $s\sigma \in D^-$, $s\sigma \notin L(\mathbf{G})$ can be derived. Otherwise, if $s\sigma \notin \overline{D_E} \cup D^-$, there are two cases:

1) $s\sigma \in L(\mathbf{G})$: there exists a forcible event $f \in \Sigma_{for}$ such that $sf \in \overline{D_E}$ holds (hence $sf \in L(\mathbf{G}) \cap \overline{D_E}$ holds); also, for all events $\sigma' \in \Sigma \setminus \Sigma_{for}$, $s\sigma' \notin \overline{D_E}$ holds (hence $s\sigma' \notin L(\mathbf{G}) \cap \overline{D_E}$ holds);
2) $s\sigma \notin L(\mathbf{G})$: same situation as $s\sigma \in D^-$.

Hence, by Definition 2, D_E is forcibly-controllable with respect to $\mathbf{G}$; therefore (D, D^-) is forcibly-informative.

(only if) Since (D, D^-) is forcibly-informative, the specification D_E is forcibly-controllable with respect to any plant consistent with (D, D^-). Consider a plant $\mathbf{G}$ that is consistent with (D, D^-), where $\overline{D} \subseteq L(\mathbf{G})$ and $D^- = \Sigma^* \setminus L(\mathbf{G})$. Therefore, the following holds:

$$(\forall s \in \overline{D_E}, \forall \sigma \in \Sigma_u) s\sigma \in L(\mathbf{G}) \Rightarrow [s\sigma \in \overline{D_E}] \vee [((\exists f \in \Sigma_{for}) sf \in \overline{D_E}) \wedge ((\forall \sigma' \in \Sigma \setminus \Sigma_{for}) s\sigma' \notin \overline{D_E})].$$

If the string $s\sigma \notin L(\mathbf{G})$, then $s\sigma \in D^-$ holds; otherwise, the following applies:

$$[s\sigma \in \overline{D_E}] \vee [((\exists f \in \Sigma_{for}) sf \in \overline{D_E}) \wedge$$
$$((\forall \sigma' \in \Sigma \setminus \Sigma_{for}) s\sigma' \notin \overline{D_E})].$$

Hence, the necessary part holds, which concludes the proof. ■

Proposition 3 suggests that forcible-informativity can be verified by examining all strings s in the specification $\overline{D_E}$ and all uncontrollable events $\sigma \in \Sigma_u$. In this context, the

data set D^- is essential for resolving ambiguity when $s\sigma \notin \overline{D_E}$. Specifically, if $s\sigma \in D^-$, it indicates that $s\sigma$ cannot be generated by any plant $\mathbf{G}$ (including the true, unknown model) consistent with (D, D^-), since $D^- \cap L(\mathbf{G}) = \emptyset$. Comparing Proposition 3 with [10, Theorem 1] (the criterion for informativity), with the event-forcing mechanism, forcible-informativity allows more flexibility in using the data to devise a data-driven supervisory control that enforces the specification. Even if informativity does not hold, the specification can still be enforced through appropriate preemption of uncontrollable events. An illustrative example is provided below.

Example 3 *Consider again the model-unknown robot navigation in Example 1 with a subset of forcible events $\Sigma_{for} = \{d, f\}$. Consider a pair of data (D, D^-) where $D = \{e, dfbf, dfcd\}$ and $D^- = \{a, dcc, dfbfb, dfbfc, dfcdb, dfcdc, f^m\}$ $(m \in \{2, 3, 4, \dots\})$. Let $E = \{dca, dfd, dfcd, dfbf^n\}$ where $n \in \{1, 2, \cdots\}$. We have $D_E = \overline{D} \cap E = \{dfbf, dfcd\}$. According to the criterion for informativity [10, Theorem 1], the data pair (D, D^-) is not informative for the specification E because for the strings $\epsilon, d, dfb, dfc \in \overline{D_E}$ and the uncontrollable event $c \in \Sigma_u$, it holds that $\epsilon \cdot c \notin \overline{D_E} \cup D^-, dc \notin \overline{D_E} \cup D^-, dfbc \notin \overline{D_E} \cup D^-$, and $dfcc \notin \overline{D_E} \cup D^-$, respectively. With event-forcing, on the other hand, for string $\epsilon \in \overline{D_E}$, the following holds:*

$$(\exists d \in \Sigma_{for})[\epsilon \cdot d \in \overline{D_E}] \wedge [(\forall \sigma' \in \Sigma \setminus \Sigma_{for})\epsilon \cdot \sigma' \notin \overline{D_E}].$$

This means that forcible event d is available to "drive" the empty string ϵ within $\overline{D_E}$, thereby "rescuing" the lack of information $\epsilon \cdot c \notin \overline{D_E} \cup D^-$. This is similar for strings $d, dfb, dfc \in \overline{D_E}$ with suitable forcible events. For the strings $df, dfbf$, and $dfcd$, it holds that $dfc \in \overline{D_E}$, $dfbfc \in D^-$, and $dfcdc \in D^-$, respectively. By Proposition 3, (D, D^-) is verified to be forcibly-informative with respect to E; hence, D_E is forcibly-controllable with respect to all plants consistent with (D, D^-). □

As demonstrated in Example 3, compared to the control mechanism that simply enables and disables events, event-forcing has the advantage of reducing data quality requirements. While this approach requires the supervisor to be equipped with the forcing mechanism, the ability to force specific events allows it to bypass uncertainties from uncontrollable events (as seen with the strings ϵ, d, dfb, dfc in Example 3), thereby ensuring enforcement of the desired specifications.

B. Verification of forcible-informativity

This subsection presents an algorithm for verifying forcible-informativity by defining a *data-driven automaton*. We first construct a finite-state automaton to represent $D^- \subseteq \Sigma^*$ (this is always possible since by Assumption 1 D^- is a regular language) [2]. Specifically, the finite-state automaton representing D^- is defined as $\mathbf{G_{D^-}} = (Q_{D^-}, \Sigma, \delta_{D^-}, q^0_{D^-}, Q^m_{D^-})$, where

- Q_{D^-} is the finite state set and Σ is the finite event set,
- $\delta_{D^-} : Q_{D^-} \times \Sigma \to Q_{D^-}$ is the partial state transition function, which can be extended to $\delta_{D^-} : Q_{D^-} \times \Sigma^* \to Q_{D^-}$,
- $q^0_{D^-} \in Q_{D^-}$ is the initial state,
- $Q^m_{D^-} = \{\delta_{D^-}(q^0_{D^-}, s) \mid s \in D^-\} \subseteq Q_{D^-}$ is the marker state set.

Note that $L(\mathbf{G_{D^-}}) = \overline{D^-}$ and $L_m(\mathbf{G_{D^-}}) = D^-$. Building on $\mathbf{G_{D^-}}$, the definition of a data-driven automaton is presented below.

Definition 4 *Consider an event set Σ, a specification $E \subseteq \Sigma^*$, and two data sets $D, D^- \subseteq \Sigma^*$ with an automaton $\mathbf{G_{D^-}}$ representing D^-. A data-driven automaton is defined as $\hat{\mathbf{G}}(\Sigma, E, D, D^-) = (\hat{Q}, \Sigma, \hat{\delta}, \hat{q}_0, \hat{Q}_m)$, where*

- *$\hat{Q} = Q_{D^-} \cup \{\hat{q}_s \mid s \in \overline{D}\}$ is the state set where $\hat{q}_\epsilon = q^0_{D^-}$,*
- *$\hat{\delta} = \hat{\delta}_{D^-} \cup \{(\hat{q}_s, \sigma) \to \hat{q}_{s\sigma} \mid (\sigma \in \Sigma) \wedge (s, s\sigma \in \overline{D})\}$ is the partial state transition function,*
- *$\hat{q}_0 = q^0_{D^-}$ is the initial state,*
- *$\hat{Q}_m = Q^m_{D^-}$ is the marker state set.*

□

In short, a data-driven automaton $\hat{\mathbf{G}}(\Sigma, E, D, D^-)$ is a finite-state automaton where $L(\hat{\mathbf{G}}(\Sigma, E, D, D^-)) = \overline{D \cup D^-}$ and $L_m(\hat{\mathbf{G}}(\Sigma, E, D, D^-)) = D^-$. For the transition function $\hat{\delta}$: 1) all existing transitions from δ_{D^-} are included; 2) for all strings $s \in \overline{D}$ and events $\sigma \in \Sigma$, if $s\sigma \in \overline{D}$ and $\delta_{D^-}(\hat{q}_s, \sigma)$ is not defined, a transition is added from state $\hat{q}_s$ to $\hat{q}_{s\sigma}$ via event σ; 3) $\hat{\delta}$ can be extended to $\hat{\delta} = \{(\hat{q}_1, s) \to \hat{q}_2 \mid (s \in \Sigma^*) \wedge (\hat{q}_1, \hat{q}_2 \in \hat{Q})\}$. The concept of data-driven automaton was first proposed in [10] but only for finite D^-. We generalize this concept to accommodate possibly infinite D^-. A detailed comparison is given in Remark 1 below.

For the data-driven automaton $\hat{\mathbf{G}}(\Sigma, E, D, D^-)$, we define the set $\hat{Q}_{D_E} = \{\hat{q} \in \hat{Q} \mid (s \in \overline{D_E}) \wedge (\hat{q}_0, s) \to \hat{q}\}$ as the subset of states in $\hat{Q}$ that are associated with strings in $\overline{D_E}$. The specification D_E can be computed following a finite-state automaton generated by the specification E, as E is a regular language by Assumption 1. Furthermore, since it is assumed that $\overline{D} \cap D^- = \emptyset$ (hence $\overline{D_E} \cap D^- = \emptyset$), we also have $\hat{Q}_{D_E} \cap \hat{Q}_m = \emptyset$.

Remark 1 *The data-driven automaton defined in Definition 4 differs from and is more general than the one proposed in [10]. Specifically, the automaton in Definition 4 is built based on the finite-state automaton $\mathbf{G_{D^-}}$, derived from the potentially infinite set D^-. For any string in $\overline{D}$ that is not in $\overline{D^-}$, a corresponding additional state is introduced in $\mathbf{G_{D^-}}$, along with a new transition relation. In contrast, the data-driven automaton in [10] represents a prefix-tree automaton, created by unfolding the strings in $\overline{D \cup D^-}$, where both D and D^- are required to be finite. As a result, the data-driven automaton in [10] is alwasy loop-free, whereas in our case it may contain loops. The following example demonstrates this construction for an infinite D^-.* □

Example 4 *Consider again Example 3. We construct the data-driven automaton $\hat{G}(\Sigma, E, D, D^-)$ and compute the set $\hat{Q}_{D_E}$. First, based on the specification language E, we conclude that $D_E = \{dfbf, dfcd\}$. Based on the infinite (but regular) data D^-, a finite-state automaton $\mathbf{G_{D^-}} = (Q_{D^-}, \Sigma, \delta_{D^-}, q^0_{D^-}, Q^m_{D^-})$ is constructed, as shown in Fig. 3, where $L_m(\mathbf{G_{D^-}}) = D^-$. Next, a data-driven automaton*

$\hat{\mathbf{G}}(\Sigma, E, D, D^-)$ is constructed based on $\mathbf{G_{D^-}}$ and is shown in Fig. 4, which has 17 states. The state set $\hat{Q}_{D_E}$ is also derived based on the language D_E; the states in $\hat{Q}_{D_E}$ are color-coded in light blue (states $\hat{q}_0 \dots \hat{q}_6$). For clarity of display, only the states in $\hat{Q}_{D_E}$ are numbered in the data-driven automaton throughout the remainder of the paper. □

Based on the constructed data-driven automaton, we now present an algorithm (Algorithm 1 below) to verify the forcible-informativity of a given data pair (D, D^-). Specifically, all states in $\hat{Q}_{D_E}$ and all uncontrollable events $\sigma \in \Sigma_u$ are required to be examined. Consider a state $q \in \hat{Q}_{D_E}$, corresponding to a string $s \in \overline{D_E}$, and an uncontrollable event $\sigma \in \Sigma_u$. In line 4, if $\hat{\delta}(q, \sigma) \notin \hat{Q}_{D_E} \cup \hat{Q}_m$ or $\neg\hat{\delta}(q, \sigma)!$ holds, it implies that $s\sigma \notin \overline{D_E} \cup D^-$, indicating that informativity [10] does not hold.

Further, in line 5, if no forcible event $f \in \Sigma_{for}$ satisfies $\hat{\delta}(q, f) \in \hat{Q}_{D_E}$, then no forcible event can preempt the uncontrollable event σ at string s, resulting in the violation of forcible-informativity. Conversely, if a corresponding forcible event exists, but a non-forcible event σ' also occurs such that $\hat{\delta}(q, \sigma') \in \hat{Q}_{D_E}$, the event σ' will still be preempted. This leads to the omission of the string $s\sigma' \in \overline{D_E}$, thereby also failing to satisfy forcible-informativity. At this point, the algorithm terminates; otherwise, it continues the examining process. The termination of Algorithm 1 is guaranteed, since the sets $\hat{Q}_{D_E}$ and Σ_u are finite.

Comparing with [10, Algorithm 1] (which checks informativity), there are three main differences: 1) it requires a subset of forcible events $\Sigma_{for} \subseteq \Sigma$ as input; 2) the construction of the data-driven automaton can address possibly infinite D^-; and 3) an additional criterion (from Proposition 3) must be checked at line 5 if informativity does not hold after the check at line 4. The correctness of Algorithm 1 is established below.

Proposition 4 *A pair (D, D^-) is forcibly-informative for the given specification E if and only if Algorithm 1 returns "Yes".*

Proof: (if) If Algorithm 1 outputs "Yes", then for all $q \in \hat{Q}_{D_E}$ and $\sigma \in \Sigma_u$, the following holds:

$$[\hat{\delta}(q, \sigma) \in \hat{Q}_{D_E} \cup \hat{Q}_m] \vee [((\exists f \in \Sigma_{for})\hat{\delta}(q, f) \in \hat{Q}_{D_E}) \wedge ((\forall \sigma' \in \Sigma \setminus \Sigma_{for})\hat{\delta}(q, \sigma') \notin \hat{Q}_{D_E})],$$

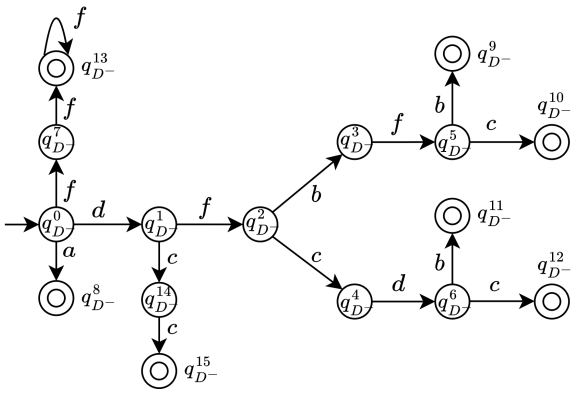


Fig. 3: An automaton $\mathbf{G_{D^-}} = (Q_{D^-}, \Sigma, \delta_{D^-}, q^0_{D^-}, Q^m_{D^-})$.

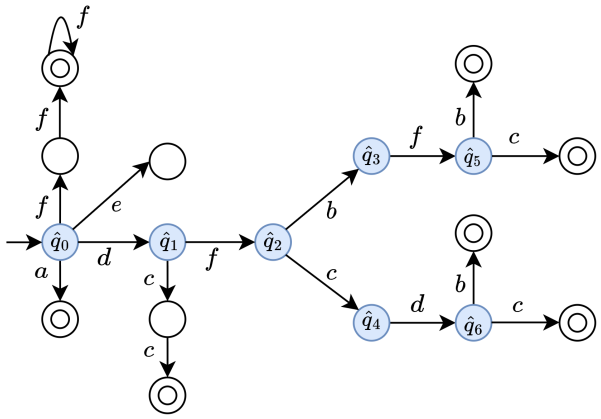


Fig. 4: A data-driven automaton $\hat{\mathbf{G}}(\Sigma, E, D, D^-)$.

Algorithm 1: Verification of forcible-informativity

Input: $\Sigma = \Sigma_c \dot{\cup} \Sigma_u, \Sigma_{for} \subseteq \Sigma$, the specification $E \subseteq \Sigma^*$, $D \subseteq \Sigma^*$ and $D^- \subseteq \Sigma^*$

Output: Yes ((D, D^-) is forcibly-informative for E) or No ((D, D^-) is not forcibly-informative for E)

1 Construct a data-driven automaton $\hat{\mathbf{G}}(\Sigma, E, D, D^-) = (\hat{Q}, \Sigma, \hat{\delta}, \hat{q}_0, \hat{Q}_m)$ with $\hat{Q}_{D_E} \subseteq \hat{Q}$;
2 **for** $q \in \hat{Q}_{D_E}$, **do**
3 **for** $\sigma \in \Sigma_u$, **do**
4 **if** $[\hat{\delta}(q, \sigma) \notin \hat{Q}_{D_E} \cup \hat{Q}_m] \vee [\neg\hat{\delta}(q, \sigma)!]$, **then**
5 **if** $[(\nexists f \in \Sigma_{for})\ \hat{\delta}(q, f) \in \hat{Q}_{D_E}] \vee [(\exists \sigma' \in \Sigma \setminus \Sigma_{for})\ \hat{\delta}(q, \sigma') \in \hat{Q}_{D_E}]$, **then**
6 **return** "No";
7 **return** "Yes";

which implies the forcible-informativity of (D, D^-) by its criteria given in Proposition 3.

(only if) The forcible-informativity of (D, D^-) with respect to E implies the following:

$$(\forall s \in \overline{D_E}, \forall \sigma \in \Sigma_u)$$
$$[s\sigma \in \overline{D_E} \cup D^-] \vee [((\exists f \in \Sigma_{for})\ sf \in \overline{D_E}) \wedge ((\forall \sigma' \in \Sigma \setminus \Sigma_{for}) s\sigma' \notin \overline{D_E})].$$

Based on the definition of data-driven automaton $\hat{\mathbf{G}}(\Sigma, E, D, D^-)$, for all $q \in \hat{Q}_{D_E}$ and for all $\sigma \in \Sigma_u$, the two conditions shown in lines 4 and 5 will never be satisfied, so the algorithm will never break in line 6. Since set D is finite, set D_E is also finite; hence $\hat{Q}_{D_E}$ is finite, suggesting that Algorithm 1 will terminate and eventually return "Yes". ■

Remark 2 *The complexity of Algorithm 1 is analyzed as follows. Line 1 builds a data-driven automaton representing the language $\overline{D \cup D^-}$, with the worst-case complexity $O(|D| \cdot max\_len(D) + |Q_{D^-}|)$. If D^- is finite, this simplifies to $O(|D \cup D^-| \cdot max\_len(D \cup D^-))$. For lines 2–11, the worst-case complexity is:*

- *iterating over all elements in the state set $\hat{Q}_{D_E}$ and the event set Σ_u, yielding complexity $O(|\overline{D}| \cdot |\Sigma|)$;*
- *in line 5, each forcible event in Σ_{for} and each non-forcible event in $\Sigma \setminus \Sigma_{for}$ are checked, which has complexity $O(|\Sigma|)$.*

Thus, the overall complexity of Algorithm 1 is $O(|D| \cdot max\_len(D) + |Q_{D^-}| + |\overline{D}| \cdot |\Sigma|^2)$. When D^- is finite, this complexity becomes $O(|D \cup D^-| \cdot max\_len(D \cup D^-) + |\overline{D}| \cdot |\Sigma|^2)$. □

Example 5 *Consider Example 3 again. We verify the forcible-informativity for the given (regular) specification language E. Following Example 4, $D_E = \{dfbf, dfcd\}$, and the corresponding data-driven automaton $\hat{\mathbf{G}}(\Sigma, E, D, D^-)$ is shown in Fig. 4. Consider a state $\hat{q}_0 \in \hat{Q}_{D_E}$ in the data-driven automaton and the only uncontrollable event $c \in \Sigma_u$. Since $\hat{\delta}(\hat{q}_0, c) \notin \hat{Q}_{D_E} \cup \hat{Q}_m$, the data (D, D^-) can be verified to be not informative [10]. However, since there exists a forcible event $d \in \Sigma_{for}$ such that $\hat{\delta}(q, d) \in \hat{Q}_{D_E}$ and for any non-forcible event $\sigma' \in \Sigma \setminus \Sigma_{for}$, $\hat{\delta}(\hat{q}_0, \sigma') \notin \hat{Q}_{D_E}$ holds. The forcible event d can preempt the potential occurrence of the uncontrollable event c, maintaining the specification D_E. Similarly, this argument applies to states $\hat{q}_1, \hat{q}_3, \hat{q}_4$. For states $\hat{q}_2, \hat{q}_5$, and $\hat{q}_6$, with the uncontrollable event $c \in \Sigma u$, we have $\hat{\delta}(\hat{q}_2, c) \in \hat{Q}_m$, $\hat{\delta}(\hat{q}_5, c) \in \hat{Q}_m$, and $\hat{\delta}(\hat{q}_6, c) \in \hat{Q}_m$. Consequently, based on Algorithm 1, we conclude that (D, D^-) is forcibly-informative for the specification E.* □

C. Data-driven supervisory control with forcible events

By Definition 3, if the pair (D, D^-) is forcibly-informative for E, the specification D_E is confirmed to be forcibly-controllable for all plants consistent with (D, D^-). This enables the synthesis of a supervisor to enforce the specification D_E. As a solution to Problem 1, for any plant $\mathbf{G}$ (including the model-unknown true plant) consistent with (D, D^-), a data-driven supervisory control $V_{D,for} : \overline{D} \to 2^{\Sigma}$ can be designed such that $L(V_{D,for}/\mathbf{G}) = \overline{D_E}$, according to

$$V_{D,for}(s) := \begin{cases} \Sigma_u \dot{\cup} \{t \in \Sigma_c \mid st \in \overline{D_E}\}, \\ \quad \text{if } [s \in \overline{D_E}] \wedge [(\forall \sigma \in \Sigma_u)\ s\sigma \in \overline{D_E} \cup D^-]; \\ \{f \in \Sigma_{for} \mid sf \in \overline{D_E}\}, \\ \quad \text{if } [s \in \overline{D_E}] \wedge [(\exists \sigma' \in \Sigma_u)\ s\sigma' \notin \overline{D_E} \cup D^-]; \\ \Sigma, \text{ if } s \in \overline{D} \setminus \overline{D_E}. \end{cases}$$

The correctness of this data-driven supervisory control follows from Proposition 3 and the model-based supervisor in Proposition 2. When (D, D^-) is forcibly-informative, there are three supervisory control scenarios for any given string $s \in \overline{D}$, as listed below.

- If the string $s \in \overline{D_E}$ and the occurrence of uncontrollable events does not violate the controllability of D_E, the supervisor will only disable all controllable events t where $st \notin \overline{D_E}$;
- If the string $s \in \overline{D_E}$ and the occurrence of an uncontrollable event violates the controllability of D_E, the supervisor will employ event-forcing, using forcible events f where $sf \in \overline{D_E}$. This preemption will not fail to enforce any sublanguage $s\sigma'$ where $\sigma' \in \Sigma \setminus \Sigma_{for}$, as $s\sigma' \notin \overline{D_E}$;
- In other scenarios ($s \in \overline{D} \setminus \overline{D_E}$), the supervisor will not disable any controllable events or force any forcible events.

Example 6 *Consider again Examples 3 and 5. For the specification E, the given pair (D, D^-) has been verified to be forcibly-informative. Consequently, for any plant $\mathbf{G}$ (include the model-unknown true plant) consistent with (D, D^-), the supervisory control $V_{D,for} : \overline{D} \to 2^{\Sigma}$ such that $L(V_{D,for}/\mathbf{G}) = \overline{D_E}$ is detailed in the following:*

$$\begin{cases} V_{D,for}(\epsilon) = \{d\}; \\ V_{D,for}(d) = \{f\}; \\ V_{D,for}(df) = \{b, c\}; \\ V_{D,for}(dfb) = \{f\}; \\ V_{D,for}(dfc) = \{d\}; \\ V_{D,for}(dfbf) = \{c\}; \\ V_{D,for}(dfcd) = \{c\}; \\ V_{D,for}(s) = \Sigma, \text{where } s \in \overline{D} \setminus \overline{D_E}. \end{cases}$$

□

V. Forcible-informatizability

What if a given data pair (D, D^-) fails to be forcibly-informative for a specification E? According to Definition 3, synthesizing a data-driven supervisor to enforce the non-empty specification D_E is impossible. In this case, we wish to explore the possibility of enforcing a smaller non-empty specification, which leads us to a new concept called *forcible-informatizability*. To verify this concept, we introduce a novel *barrier language*, which consists of a set of suffixes of a given string restricted by the data, providing local forcible-informativity. Using the barrier language, we establish a necessary and sufficient condition for forcible-informatizability and propose a data-driven automaton based algorithm for its verification.

A. Motivation

Consider a model-unknown plant $\mathbf{G}$ that satisfies Assumption 1 with $\Sigma_{for} \subseteq \Sigma$, two data sets D and D^-, and a specification language E. Suppose that $D_E = \overline{D} \cap E$ is non-empty. Fig. 5 shows a Venn diagram illustrating the case where (D, D^-) is neither informative nor forcibly-informative for E. The green-filled area represents $\overline{D_E}$ while the blue-filled area represents D^-. Within the set $\overline{D_E} \cdot \Sigma_u$ (outlined in red), the red-filled regions indicate strings in $\overline{D_E} \cdot \Sigma_u$ that do not belong to either $\overline{D_E}$ or D^-. These regions of strings prevent the pair (D, D^-) from achieving informativity [10]. For example, consider strings $s, s' \in \overline{D_E}$ as shown in Fig. 5. Suppose there exist uncontrollable events $\sigma_u, \sigma'_u \in \Sigma_u$ such that $s\sigma_u$ and $s'\sigma'_u$ are not contained in $\overline{D_E} \cup D^-$. By [10, Theorem 1] (the criterion for informativity), informativity fails.

Regarding forcible-informativity, for the string s', no forcible event is available to preempt the uncontrollable event σ'_u. Suppose there is only one forcible event $f \in \Sigma_{for}$ and

an event $\sigma \in \Sigma \setminus \Sigma_{for}$ such that $sf, s\sigma \in \overline{D_E}$. Although this forcible event f could preempt σ_u at string s, it may also inadvertently preempt the legitimate behavior $s\sigma$ due to the forcing mechanism [24]. By Proposition 3, the data pair (D, D^-) also fails to be forcibly-informative for E.

In this situation, one may consider a smaller specification language than $\overline{D_E}$, and test if the data pair (D, D^-) can be forcibly-informative for the smaller specification. This amounts to shrinking the red-filled regions $\overline{D_E} \cdot \Sigma_u \setminus (\overline{D_E} \cup D^-)$, thereby reducing associated uncertainties.

Without event-forcing, the concept of *informatizability* was introduced in [11]. Specifically, a data pair (D, D^-) is informatizable if there exists a non-empty sublanguage $K \subseteq \overline{D_E}$ such that K is controllable with respect to any plant consistent with (D, D^-). Informatizability means that even if informativity fails, there exists a supervisor that can enforce a smaller, non-empty specification K for any plant consistent with (D, D^-), including the true, unknown plant. However, informatizability can impose strict requirement on data in a similar way to informativity, as shown in the example below.

Example 7 *Consider Example 3 with the modified dataset* $D' = \{e, dfbf, dfcd, dca\}$ *(meaning a new string* dca *is observed). We find* $D'_E = \overline{D'} \cap E = \{dca, dfbf, dfcd\}$. *The corresponding data-driven automaton* $\hat{\mathbf{G}}(\Sigma, E, D', D^-)$ *is shown in Fig. 6, where all states in* $\hat{Q}_{D'_E}$ *are color-coded in blue. Following Algorithm 1, the presence of state* $\hat{q}_8$ *(corresponding to the string* $dca \in \overline{D'_E}$*) indicates that* (D', D^-) *is not forcibly-informative (thus not informative). According to the criterion for informatizability in [11, Theorem 2],* (D', D^-) *is not informatizable for* E*; namely there does not exist any non-empty sublanguage* $K \subseteq \overline{D'_E}$ *such that* (D', D^-) *is informative for* K. *With event-forcing mechanism, on the other hand, consider a sublanguage* $K' = \{dc\} \subseteq \overline{D'_E}$. *By Proposition 3,* (D', D^-) *is verified to be forcibly-informative for* K'. □

As shown in Example 7, similar to informativity, informatizability [11] often cannot be achieved due to stringent data quality requirements. However, incorporating forcible events allows us to enforce a non-empty sublanguage specification where forcible-informativity holds. This motivates us to investigate the concept of *forcible-informatizability*.

B. Forcible-informatizability

Definition 5 [forcible-informatizability] *Consider an event set* $\Sigma = \Sigma_c \dot{\cup} \Sigma_u$ *with* $\Sigma_{for} \subseteq \Sigma$. *Given a pair* (D, D^-) *and a specification language* $E \subseteq \Sigma^*$ *with* $D_E = \overline{D} \cap E$, *the pair* (D, D^-) *is said to be forcibly-informatizable for* E *if there exists a non-empty language* $K \subseteq \overline{D_E}$ *such that* K *is forcibly-controllable with respect to every plant* $\mathbf{G}$ *consistent with* (D, D^-), *i.e., there exists a supervisory control for* $\mathbf{G}$ *to enforce* K. □

The concept of forcible-informatizability shows that even when the data pair (D, D^-) is not forcibly-informative for E, a data-driven forcing supervisor can still be designed to enforce a smaller non-empty sublanguage $K \subseteq \overline{D_E}$. Compared to informatizability [11], both concepts indicate the possibility of enforcing a smaller specification sublanguage $K \subseteq \overline{D_E}$. However, forcible-informatizability specifically involves the existence of a data-driven forcing supervisor with the event-forcing mechanism. Just as forcible-informativity is weaker than informativity, forcible-informatizability is also weaker than informatizability.

Example 8 *Consider again Example 7. Since there exists a non-empty sublanguage* $K' = \{dc\} \subseteq \overline{D'_E}$ *such that* (D', D^-) *is forcibly-informative for* K', *we conclude that* (D', D^-) *is forcibly-informatizable for* E. □

From Example 8, we have the following corollary.

Corollary 1 *A data pair* (D, D^-) *is forcibly-informatizable for a specification* E *if and only if there exists a non-empty specification language* $K \subseteq \overline{D_E}$ *such that* (D, D^-) *is forcibly-informative for* K.

Proof: (if) Since $K \subseteq \overline{D_E} = \overline{\overline{D} \cap E} \subseteq \overline{D} \cap \overline{E}$, it follows that $\overline{D} \cap K = K$. Therefore, if (D, D^-) is forcibly-informative for K, then the non-empty specification $\overline{D} \cap K = K$ is forcibly-controllable for any plant consistent with (D, D^-). By Definition 5, (D, D^-) is forcibly-informatizable.

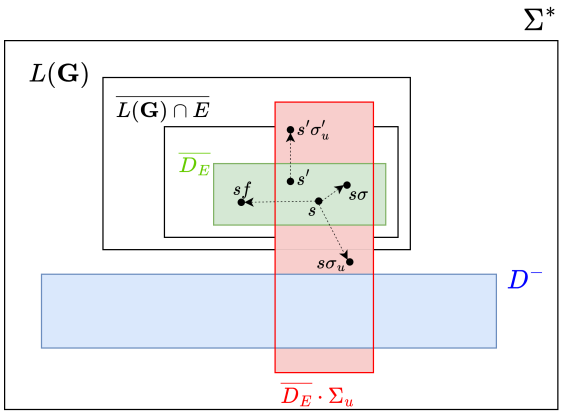


Fig. 5: A Venn diagram.

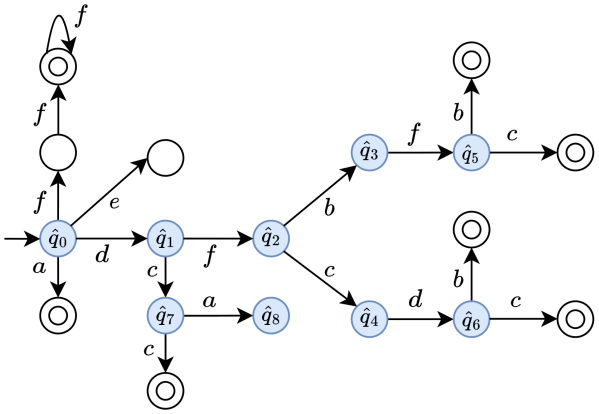


Fig. 6: A data-driven automaton $\hat{\mathbf{G}}(\Sigma, E, D', D^-)$.

(only if) If (D, D^-) is forcibly-informatizable for E, then there exists a non-empty language $K \subseteq \overline{D_E}$ such that K is forcibly-controllable for any plant consistent with (D, D^-). This implies the forcible-informativity of (D, D^-) for K, since $\overline{D} \cap K = K$. ■

Corollary 1 illustrates the relationship between forcible-informatizability and forcible-informativity with respect to the specification sublanguage $K \subseteq \overline{D_E}$. This result will be applied in the following sections to verify forcible-informatizability.

C. Barrier language

To verify forcible-informatizability, this subsection introduces a novel concept of barrier language. The barrier language defines a "safe region" that remains forcibly-informative through appropriate forcible events. The boundary of this region acts as a barrier, ensuring that system evolutions within it will stay within. Based on this barrier language, we propose a necessary and sufficient condition for verifying forcible-informatizability in the next subsection. We begin by introducing a preliminary concept.

Definition 6 *Given an event set Σ, a language $L \subseteq \Sigma^*$ and a string $s \in \Sigma^*$, we define $\Sigma_L(s) = \{s' \in \Sigma^* \mid ss' \in L\}$.*

Definition 6 indicates that the set $\Sigma_L(s)$ contains all suffixes of the string s that complete s to a member of the language L. Based on this, the concept of a barrier language is defined as follows.

Definition 7 *Consider a data pair (D, D^-) and a specification $E \subseteq \Sigma^*$. Given a string $s \in \overline{D_E}$, we define*

$$\mathcal{B}(s) = \{B \subseteq \Sigma_{\overline{D_E}}(s) \mid B \neq \emptyset \wedge (\Sigma_{\overline{D_E}}(s), \Sigma_{D^-}(s)) \text{ is forcibly-informative for } B\} \quad (2)$$

as the set of barrier languages for s. The language B in $\mathcal{B}(s)$ is called a barrier language for s.

Note that $\Sigma_{D^-}(s) = \{s' \in \Sigma^* \mid ss' \in D^-\}$ and $\Sigma_{\overline{D_E}}(s) = \{s'' \in \Sigma^* \mid ss'' \in \overline{D_E}\}$. Two Venn diagrams in Fig. 7 illustrate the concept of a barrier language $B \in \mathcal{B}(s)$. In plain terms, for a string $s \in \overline{D_E}$, shown in the upper Venn diagram, its barrier language B is a non-empty sublanguage containing a set of *suffix strings* of s with respect to $\overline{D_E}$ such that $(\Sigma_{\overline{D_E}}(s), \Sigma_{D^-}(s))$ is forcibly-informative for B (as shown in the lower Venn diagram). This means that the language B is forcibly-controllable with respect to all plants consistent with $(\Sigma_{\overline{D_E}}(s), \Sigma_{D^-}(s))$. For a string $s \in \overline{D_E}$, its barrier language $B \in \mathcal{B}(s)$ can be seen as a "safe area", maintaining its own invariance in terms of forcible-controllability. See the example below for illustration.

Example 9 *Consider again Example 7 with a corresponding data-driven automaton $\hat{\mathbf{G}}(\Sigma, E, D', D^-)$ shown in Fig. 8. Take a string $d \in \overline{D'_E}$ for which we want to determine a barrier language. First, we calculate $\Sigma_{\overline{D'_E}}(d) = \{\epsilon, c, ca, f, fb, fc, fbf, fcd\}$ and $\Sigma_{D^-}(d) = \{cc, fbfb, fbfc, fcdb, fcdc\}$. Now, consider the sublanguage $B = \{c\} \subseteq \Sigma_{\overline{D'_E}}(d)$. We find that $\overline{\Sigma_{\overline{D'_E}}(d)} \cap B = \{c\}$. For $\epsilon, c \in \overline{\{c\}}$ and considering the uncontrollable event $c \in \Sigma_u$, we have $\epsilon \cdot c = c \in \overline{\{c\}}$, and $cc \in \Sigma_{D^-}(d)$ also holds.*

Applying Proposition 3, we conclude that the pair $(\Sigma_{\overline{D'_E}}(d), \Sigma_{D^-}(d))$ is forcibly-informative for B. Therefore, $B = \{c\}$ is a barrier language for the string d, as illustrated by the red dotted box in Fig. 8. □

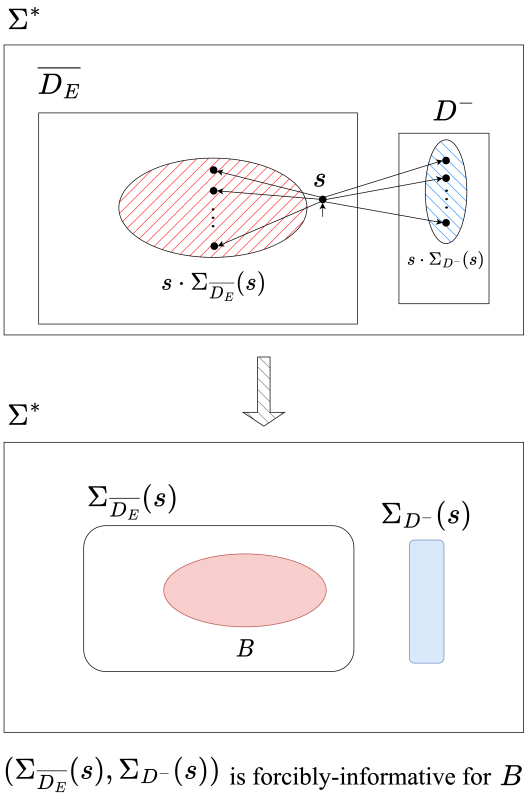


Fig. 7: Venn diagram for a barrier language $B \in \mathcal{B}(s)$.

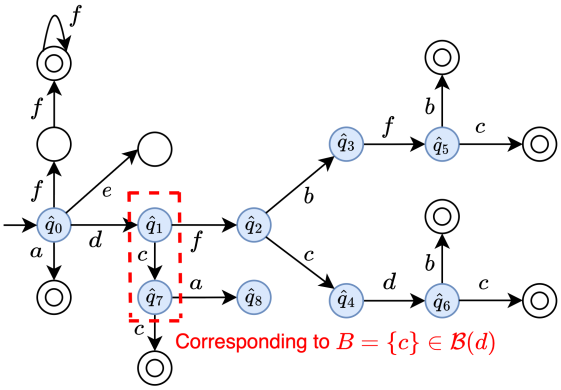


Fig. 8: A data-driven automaton with illustration of a barrier language $B = \{c\} \in \mathcal{B}(d)$.

For a string $s \in \overline{D_E}$, there generally exist multiple barrier languages for s, so the set of barrier languages $\mathcal{B}(s)$ can have multiple elements. Based on this, a property of barrier languages is shown below.

Proposition 5 *Consider a string $s \in \overline{D_E}$ with two barrier languages $B_1 \in \mathcal{B}(s)$ and $B_2 \in \mathcal{B}(s)$. Then it holds that $B_1 \cup B_2 \in \mathcal{B}(s)$, i.e., $B_1 \cup B_2$ is also a barrier language for the string s.*

Proof: For string s, we have $(\Sigma_{\overline{D_E}}(s), \Sigma_{D^-}(s))$ is forcibly-informative for B_1 and B_2 where $B_1, B_2 \subseteq \Sigma_{\overline{D_E}}(s)$. Therefore, the following holds:

$$(\forall s \in \overline{B_1}, \forall \sigma \in \Sigma_u)[s\sigma \in \overline{B_1} \cup D^-] \vee$$

$$[(\exists f \in \Sigma_{for})\ sf \in \overline{B_1}) \wedge ((\forall \sigma' \in \Sigma \setminus \Sigma_{for}) s\sigma' \notin \overline{B_1}].$$

A similar result can be obtained by replacing B_1 with B_2 in the equation above. Consequently, the following holds:

$$(\forall t \in \overline{B_1 \cup B_2}, \forall \sigma' \in \Sigma_u)[t\sigma' \in \overline{B_1 \cup B_2} \cup D^-] \vee$$

$$[(\exists f' \in \Sigma_{for})\ tf' \in \overline{B_1 \cup B_2} \wedge$$

$$((\forall \sigma'' \in \Sigma \setminus \Sigma_{for}) t\sigma'' \notin \overline{B_1 \cup B_2}]],$$

which implies that $(\Sigma_{\overline{D_E}}(s), \Sigma_{D^-}(s))$ is also forcibly-informative for $B_1 \cup B_2$. ■

Proposition 5 implies that barrier languages are closed under set unions. Therefore, for a string $s \in \overline{D_E}$, if $\mathcal{B}(s) \neq \emptyset$, there exists a non-empty *supremal barrier language* defined as $\sup\mathcal{B}(s) = \bigcup_{B \in \mathcal{B}(s)} B$. An example is provided below for illustration.

Example 10 *Revisit Example 7 with the data-driven automaton $\hat{\mathbf{G}}(\Sigma, E, D', D^-)$ depicted in Fig. 9. For the string $d \in \overline{D'_E}$, as shown in Example 9, the sublanguage $B = \{c\} \subseteq \Sigma_{\overline{D_E}}(d)$ is a barrier language for d, highlighted by the red dotted box in Fig. 9. Similarly, we can identify that the sublanguage $B' = \{fbf, fcd\} \in \mathcal{B}(d)$, marked with a blue dotted line in the same figure, is also a barrier language for d. According to Proposition 5, the union $B'' = B \cup B' = \{c, fbf, fcd\}$ also belongs to $\mathcal{B}(d)$, meaning that B'' is a barrier language for d. By inspection, in this example, the supremal barrier language, $\sup\mathcal{B}(d)$, is indeed B''.* □

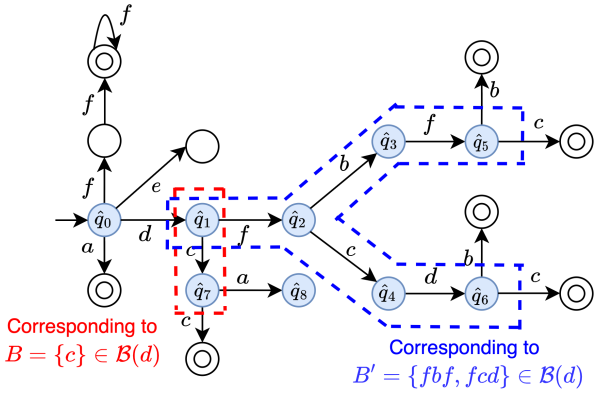


Fig. 9: A data-driven automaton with illustration of barrier languages $B, B' \in \mathcal{B}(d)$.

D. Verification of forcible-informatizability

This subsection studies the verification of forcible-informatizability using barrier languages. To start, Theorem 1 provides a necessary and sufficient condition for this verification.

Theorem 1 [N&S condition] *A pair (D, D^-) is forcibly-informatizable for a specification E if and only if:*

$$(\forall s \in \overline{D_E} \cap \Sigma_u^*, \forall \sigma \in \Sigma_u)\ [s\sigma \in \overline{D_E} \cup D^-] \vee$$

$$[(\exists s' \in \overline{s}, \exists f \in \Sigma_{for}) s'f \in \overline{D_E} \wedge \mathcal{B}(s'f) \neq \emptyset.$$

□

The proof of Theorem 1 is given in Section V-F below. Here we make several remarks in order. Theorem 1 is different from the verification criterion for informatizability [11, Theorem 2]. In the latter, informatizability fails, if there exists a string $s \in \overline{D_E} \cap \Sigma_u^*$ and an uncontrollable event $\sigma \in \Sigma_u$ such that $s\sigma \notin \overline{D_E} \cup D^-$. Theorem 1, however, shows that even when informatizability fails for a string $s \in \overline{D_E} \cap \Sigma_u^*$ due to an uncontrollable event $\sigma \in \Sigma_u$, forcible-informatizability can still be achieved if the following two conditions are both met:

- **prefix condition**: there exists a prefix $s' \in \overline{s}$ and a forcible event $f \in \Sigma_{for}$ such that $s'f \in \overline{D_E}$, allowing f to prevent the occurrence of $s\sigma$.
- **suffix condition**: there is at least one barrier language $B \in \mathcal{B}(s'f)$ associated with the string $s'f$.

Together, these conditions make it possible to relax the data requirements necessary for forcible-informatizability. An example illustrating this result is provided below, followed by a corollary derived from Theorem 1.

Example 11 *Consider again Example 7. We verify the forcible-informatizability of the data (D', D^-) for the specification E. From the data-driven automaton in Fig. 6, for the empty string $\epsilon \in \overline{D'_E} \cap \Sigma_u^*$, there exists an uncontrollable event $c \in \Sigma_u$ such that $\epsilon \cdot c \notin \overline{D'_E} \cup D^-$. Nonetheless, there exist $\epsilon \in \overline{\epsilon}$ and $d \in \Sigma_{for}$ such that $\epsilon \cdot d \in \overline{D'_E}$. Next, we determine whether there exists a barrier language for $\epsilon \cdot d = d$, i.e., whether $\mathcal{B}(d) \neq \emptyset$. To this end, following Definition 7, we first compute $\Sigma_{\overline{D_E}}(d) = \{\epsilon, c, ca, f, fb, fc, fbf, fcd\}$ and $\Sigma_{D^-}(d) = \{cc, fbfb.fbfc, fcdb, fcdc\}$.*

By Example 10, $B = \{c\} \subseteq \Sigma_{\overline{D_E}}(d)$ and $B' = \{fbf, fcd\} \subseteq \Sigma_{\overline{D_E}}(d)$ are two barrier languages for the string d, as shown in Fig. 9. In other words, $\mathcal{B}(d) \neq \emptyset$, which confirms the forcible-informatizability of (D, D^-) for E. □

Corollary 2 [sufficient condition] *A pair (D, D^-) is forcibly-informatizable for a specification E if:*

$$(\forall s \in \overline{D_E} \cap \Sigma_u^*, \forall \sigma \in \Sigma_u)\ [s\sigma \in \overline{D_E} \cup D^-] \vee$$

$$[(\exists s' \in \overline{s}, \exists f \in \Sigma_{for}) s'f \in \overline{D_E} \wedge (\Sigma_{\overline{D_E}}(s'f), \Sigma_{D^-}(s'f))$$

is forcibly-informative for the specification $\Sigma_{\overline{D_E}}(s'f)$].

Proof: According to Definition 5, for the specification language $\Sigma_{D_E}(s'f)$, the forcibly-informativity of $(\Sigma_{\overline{D_E}}(s'f), \Sigma_{D^-}(s'f))$ implies that $\Sigma_{\overline{D_E}}(s'f) \in \mathcal{B}(s'f) \neq \emptyset$.

In other words, $\Sigma_{\overline{D_E}}(s'f)$ is a barrier language for the string $s'f$. Thus, by Theorem 1, this statement holds. ■

Corollary 2 provides a sufficient condition that requires checking forcible-informativity directly, rather than examining the barrier language for the string $s'f$. This approach can leverage the results from Section IV. To check the necessary and sufficient condition in Theorem 1 for verifying forcible-informatizability, we introduce an algorithm in the following subsection.

E. Verification algorithm for forcible-informatizability

Based on Theorem 1, we introduce an algorithm in this subsection to verify forcible-informatizability for a given data pair (D, D^-) with respect to a specification E. In Theorem 1, the main point is to check, when needed, whether a barrier language exists for specific strings. Below, we demonstrate how this can be achieved algorithmically using a data-driven automaton. By Definition 7, verifying that the set of barrier languages $\mathcal{B}(s) \neq \emptyset$ for a string $s \in \overline{D_E}$ is equivalent to checking whether there is a non-empty language $B \subseteq \Sigma_{\overline{D_E}}(s)$ such that $(\Sigma_{\overline{D_E}}(s), \Sigma_{D^-}(s))$ is forcibly-informative for B. Now, consider the specification language $\Sigma_{\overline{D_E}}(s)$. Since $B \subseteq \Sigma_{\overline{D_E}}(s) = \overline{\Sigma_{\overline{D_E}}(s)}$ (this can be derived by Definition 6), it follows from Corollary 1 that this verification amounts to checking the forcible-informatizability of the data pair $(\Sigma_{\overline{D_E}}(s), \Sigma_{D^-}(s))$ for the specification $\Sigma_{\overline{D_E}}(s)$.

From this perspective, verifying forcible-informatizability can be approached as a recursive *divide and conquer* problem. This process is illustrated with two Venn diagrams in Fig. 10. In the upper part of Fig. 10, consider a data pair (D, D^-) and a specification E. For a string $s \in \overline{D_E} \cap \Sigma_u^*$ and an uncontrollable event $\sigma \in \Sigma_u$, since $s\sigma \notin \overline{D_E} \cup D^-$, further checking is required, as indicated by Theorem 1.

Suppose that there exist a prefix $s' \in \overline{s}$ and a forcible event $f \in \Sigma_{for}$ such that $s'f \in \overline{D_E}$. According to Theorem 1, to verify the forcible-informatizability of (D, D^-) for E, the task shifts to verifying $\mathcal{B}(s'f) \neq \emptyset$. This can be reduced to a sub-problem: checking forcible-informatizability for the pair $(\Sigma_{\overline{D_E}}(s'f), \Sigma_{D^-}(s'f))$ with respect to $\Sigma_{\overline{D_E}}(s'f)$.

This checking process is illustrated in the lower part of the Venn diagram in Fig. 10. Again, we apply Theorem 1. For example, consider a string $t \in \Sigma_{\overline{D_E}}(s'f) \cap \Sigma_u^*$ and an uncontrollable event $\sigma' \in \Sigma_u$ such that $t\sigma' \notin \Sigma_{\overline{D_E}}(s'f) \cup \Sigma_{D^-}(s'f)$. By Theorem 1, further checking is needed. Suppose that there exist a prefix $t' \in \overline{t}$ and a forcible event $f' \in \Sigma_{for}$ such that $t'f' \in \Sigma_{\overline{D_E}}(s'f)$. To find a barrier language for the string $t'f'$, this can again be reduced to another sub-problem: checking forcible-informatizability for the pair $(\Sigma_{\Sigma_{\overline{D_E}}(s'f)}(t'f'), \Sigma_{\Sigma_{D^-}(s'f)}(t'f'))$ with respect to $\Sigma_{\Sigma_{\overline{D_E}}(s'f)}(t'f')$. By iterating through this process and checking all strings in $\overline{D_E} \cap \Sigma_u^*$, we can determine the forcible-informatizability of (D, D^-) for the specification E.

Based on the above discussion, we present Algorithm 2 for verifying forcible-informatizability. In Algorithm 2, function `Check_FI` (a subroutine used recursively) takes as input an automaton and a specified set of states within that automaton. The data-driven automaton $\hat{\mathbf{G}}(\Sigma, E, D, D^-)$ and the set $\hat{Q}_{D_E}$

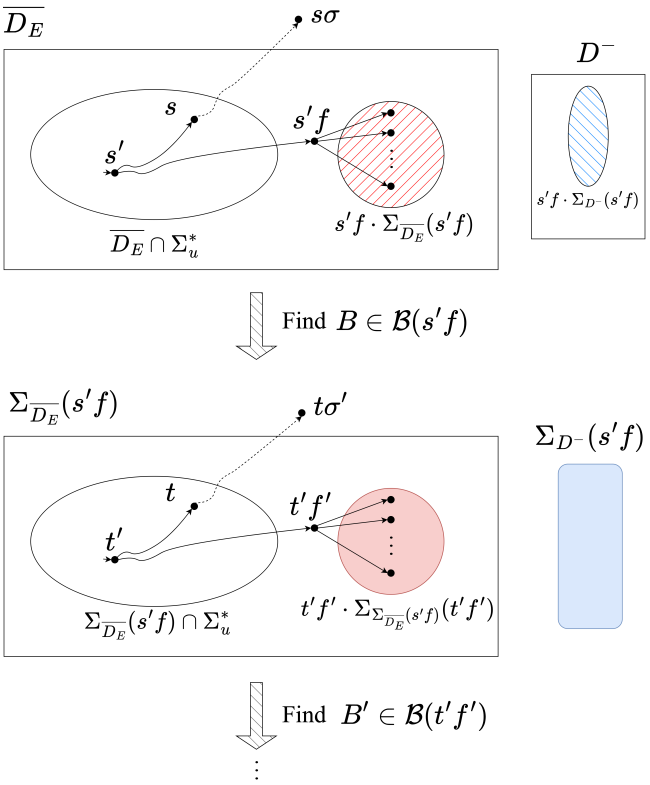


Fig. 10: A Venn diagram describing the recursive process.

(representing the states related to the specification D_E) are computed in line 1. In line 2, the function is called as `Check_FI`($\hat{\mathbf{G}}(\Sigma, E, D, D^-), \hat{Q}_{D_E}$), where the initial inputs are the data-driven automaton $\hat{\mathbf{G}}(\Sigma, E, D, D^-)$ and the set $\hat{Q}_{D_E}$. In line 3, three variables are initialized:

- Q_{new}: this variable stores the newly checked set of states within $\overline{D_E} \cap \Sigma_u^*$, focusing on states that have not yet been verified;
- Q_u: this set holds states associated with each event-wise exploration within $\overline{D_E} \cap \Sigma_u^*$, starting from the initial state. It guides the step-by-step verification of uncontrollable transitions;
- $Q_{u,all}$: this set accumulates all historical states related to $\overline{D_E} \cap \Sigma_u^*$, which is essential for maintaining a record of past states to allow comprehensive prefix testing whenever necessary.

For a string $s \in \overline{D_E} \cap \Sigma_u^*$ (corresponding to $q \in Q_u$ in line 5), if there exists an uncontrollable event $\sigma \in \Sigma_u$ such that $s\sigma \notin \overline{D_E} \cup D^-$ and no prefix $s' \in \overline{s}$ (corresponding to $q' \in Q_{u,all}$ in line 10) and forcible event $f \in \Sigma_{for}$ exist such that $s'f \in \overline{D_E}$, then, by Theorem 1, the pair (D, D^-) is not forcibly-informatizable. In this scenario, Algorithm 2 will return "No" at line 16.

Otherwise, it becomes necessary to check whether $\mathcal{B}(s'f) \neq \emptyset$, which corresponds to verifying the forcible-informatizability of the new pair $(\Sigma_{D_E}(s'f), \Sigma_{D^-}(s'f))$ with respect to the new specification $\Sigma_{D_E}(s'f)$. To perform this verification, Algorithm 2 will proceed to line 14, calling the function `Check_FI` recursively with updated inputs:

Algorithm 2: Verification of forcible-informatizability

Input: $\Sigma = \Sigma_c \dot\cup \Sigma_u, \Sigma_{for} \subseteq \Sigma$, the specification $E \subseteq \Sigma^*$, $D \subseteq \Sigma^*$ and $D^- \subseteq \Sigma^*$

Output: Yes ((D, D^-) is forcibly-informatizable)/No ((D, D^-) is not forcibly-informatizable)

1 Construct a data-driven automaton $\hat{\mathbf{G}}(\Sigma, E, D, D^-) = (\hat{Q}, \Sigma, \hat{\delta}, \hat{q}_0, \hat{Q}_m)$ with $\hat{Q}_{D_E} \subseteq \hat{Q}$;
2 **function** `Check_FI`($\hat{\mathbf{G}}(\Sigma, E, D, D^-), \hat{Q}_{D_E}$)
3 $Q_{u,all} := \{\hat{q}_0\}; Q_u := \{\hat{q}_0\}; Q_{new} := \emptyset$;
4 **while** $Q_u \neq \emptyset$ **do**
5 **for** $q \in Q_u$, **do**
6 **for** $\sigma \in \Sigma_u$ **do**
7 **if** $\hat{\delta}(q, \sigma) \in \hat{Q}_{D_E}$, **then**
8 $Q_{new} = Q_{new} \cup \{\hat{\delta}(q, \sigma)\}$;
9 **else if** $[\hat{\delta}(q, \sigma) \in \hat{Q} \setminus \hat{Q}_m] \vee [\neg\hat{\delta}(q, \sigma)!]$, **then**
10 **for** $q' \in Q_{u,all}$ **do**
11 **for** $f \in \Sigma_{for}$ **do**
12 **if** $\hat{\delta}(q', f) \in \hat{Q}_{D_E}$, **then**
13 Obtain $\hat{\mathbf{G}}' = (\hat{Q}', \Sigma', \hat{\delta}', \hat{q}'_0, \hat{Q}'_m) \sqsubseteq \hat{\mathbf{G}}(\Sigma, E, D, D^-)$ with $\hat{q}'_0 = \hat{\delta}(q', f)$;
14 **if** `Check_FI`($\hat{\mathbf{G}}', \hat{Q}' \cap \hat{Q}_{D_E}$) returns "Yes", **then**
15 **goto** line 5;
16 **return** "No";
17 **else**
18 **continue**;
19 $Q_{u,all} = Q_{u,all} \cup Q_{new}; Q_u = Q_{new}; Q_{new} = \emptyset$;
20 **return** "Yes";
21 **end function**

- automaton $\hat{\mathbf{G}}' = (\hat{Q}', \Sigma', \hat{\delta}', \hat{q}'_0, \hat{Q}'_m)$: this is a *subautomaton*[2] of the original data-driven automaton $\hat{\mathbf{G}}(\Sigma, E, D, D^-)$, where $\hat{q}'_0 = \hat{\delta}(q', f)$ is the updated initial state (corresponding to the state relates to the string $s'f$ in the original data-driven automaton), and $L(\hat{\mathbf{G}}') = \overline{\Sigma_{D_E}(s'f) \cup \Sigma_{D^-}(s'f)}$;
- state set $\hat{Q}' \cap \hat{Q}_{D_E}$: this is the subset of states within $\hat{Q}'$ that are relevant to the original specification D_E.

If `Check_FI`($\hat{\mathbf{G}}', \hat{Q}' \cap \hat{Q}_{D_E}$) returns "Yes", then $\mathcal{B}(s'f) \neq \emptyset$ holds, and the algorithm proceeds to line 5 to continue checking the remaining string (if any) in the same manner. If, however, no prefix $s' \in \overline{s}$ and no forcible event $f \in \Sigma_{for}$ exist such that $\mathcal{B}(s'f) \neq \emptyset$, Algorithm 2 terminates and returns "No". The correctness of Algorithm 2 is asserted in Proposition 6.

[2]For a finite-state automaton $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m)$, a subautomaton $\mathbf{G}'$ is defined as a five-tuple $\mathbf{G}' = (Q', \Sigma', \delta', q'_0, Q'_m)$, where $Q' \subseteq Q, \Sigma' \subseteq \Sigma, \delta' = \delta|_{Q' \times \Sigma'}$ (meaning δ' is the restriction of δ to $Q' \times \Sigma'$), $q'_0 \in Q'$, and $Q'_m \subseteq Q' \cap Q_m$. Subautomaton is denoted by $\mathbf{G}' \sqsubseteq \mathbf{G}$.

Proposition 6 *Algorithm 2 returns "Yes" if and only if (D, D^-) is forcibly-informatizable.*

Proof: (only if) We analyze the scenario where Algorithm 2 returns "Yes". First, for all $q \in Q_u$ and $\sigma \in \Sigma_u$, if $\hat{\delta}(q, \sigma) \in \hat{Q}_{D_E} \cup \hat{Q}_m$ holds all the time, eventually any string in Σ_u^* will be tested with the following holds:

$$(\forall s \in \overline{D_E} \cap \Sigma_u^*, \forall \sigma \in \Sigma_u) s\sigma \in \overline{D_E} \cup D^-,$$

which implies the forcible-informatizability of (D, D^-). Second, if there exist $q \in Q_u$ and $\sigma \in \Sigma_u$ such that $\hat{\delta}(q, \sigma) \notin \hat{Q}_{D_E} \cup \hat{Q}_m$ holds, then $\hat{\delta}(q', f) \in \hat{Q}_{D_E}$ and the updated data-driven automaton with the initial state $\hat{\delta}(q', f)$ as the input for function `Check_FI` returns "Yes", which implies the following

$$(\exists s' \in \overline{s}, \exists f \in \Sigma_{for})[s'f \in \overline{D_E}] \wedge [\mathcal{B}(s'f) \neq \emptyset].$$

Therefore, by Theorem 1, (D, D^-) is verified to be forcibly-informatizable.

(if) By contrapositive, assume that Algorithm 2 returns "No". Therefore, first, there exist $q \in Q_u$ and $\sigma \in \Sigma_u$ such that $\hat{\delta}(q, \sigma) \notin \hat{Q}_{D_E} \cup \hat{Q}_m$ holds, which implies the following

$$(\exists s \in \overline{D_E} \cap \Sigma_u^*, \exists \sigma \in \Sigma_u) s\sigma \notin \overline{D_E} \cup D^-.$$

Then, for any $q' \in Q_{u,all}$ and $f \in \Sigma_{for}$, either $\hat{\delta}(q', f) \notin \hat{Q}_{D_E}$, or $\hat{\delta}(q', f) \in \hat{Q}_{D_E}$ but the updated data-driven automaton with the initial state $\hat{\delta}(q', f)$ as the input for function `Check_FI` returns "No". From the above, the following statement can be made:

$$[(\nexists s' \in \overline{s}, \nexists f \in \Sigma_{for}) s'f \in \overline{D_E}] \vee$$
$$[(\exists s' \in \overline{s}, \exists f \in \Sigma_{for}) s'f \in \overline{D_E}) \wedge \mathcal{B}(s'f) = \emptyset],$$

which implies that (D, D^-) is not forcibly-informatizable. ■

An example illustrating Algorithm 2 is given below.

Example 12 *Reconsider Example 7. We verify the forcible-informatizability of the data (D', D^-) for the specification E using Algorithm 2. The data-driven automaton $\hat{\mathbf{G}}(\Sigma, E, D', D^-)$ is shown in Fig. 6, where $\hat{Q}_{D'_E} = \{\hat{q}_0, \hat{q}_1, \hat{q}_2, \hat{q}_3, \hat{q}_4, \hat{q}_5, \hat{q}_6, \hat{q}_7, \hat{q}_8\}$. First, we call the function `Check_FI`($\hat{\mathbf{G}}(\Sigma, E, D', D^-), \hat{Q}_{D'_E}$). Let $Q_{u,all} = \{\hat{q}_0\}, Q_u = \{\hat{q}_0\}$, and $Q_{new} = \emptyset$. For the state $\hat{q}_0 \in Q_u$ (corresponding to the empty string ϵ), we examine the only uncontrollable event $c \in \Sigma_u$. Since $\hat{\delta}(\hat{q}_0, c) \notin \hat{Q}_{D'_E}$ and $\neg\hat{\delta}(\hat{q}_0, c)!$, we check the historical state set $Q_{u,all} = \{\hat{q}_0\}$. Because there exists a forcible event $d \in \Sigma_{for}$ such that $\hat{\delta}(\hat{q}_0, d) = \hat{q}_1 \in \hat{Q}_{D'_E}$, we obtain a subautomaton $\hat{\mathbf{G}}' = (\hat{Q}', \Sigma', \hat{\delta}', \hat{q}'_0, \hat{Q}'_m) \sqsubseteq \hat{\mathbf{G}}(\Sigma, E, D, D^-)$ with $\hat{q}'_0 = \hat{\delta}(\hat{q}_0, d) = \hat{q}_1$, as shown in Fig. 11.*

Next, we recursively call the function `Check_FI` with updated inputs $\hat{\mathbf{G}}'$ and $\hat{Q}' \cap \hat{Q}_{D'_E} = \{\hat{q}_1, \hat{q}_2, \hat{q}_3, \hat{q}_4, \hat{q}_5, \hat{q}_6, \hat{q}_7, \hat{q}_8\}$. Reinitialize the variables as $Q_{u,all} = \{\hat{q}_1\}, Q_u = \{\hat{q}_1\}$, and $Q_{new} = \emptyset$. For $\hat{q}_1 \in Q_u$ and $c \in \Sigma_u$, since $\hat{\delta}(\hat{q}_1, c) = \hat{q}_7 \in \hat{Q}' \cap \hat{Q}_{D'_E}$, we update $Q_{new} = \{\hat{q}_7\}$. In line 19, the variables are updated as follows: $Q_{u,all} = \{\hat{q}_0, \hat{q}_7\}$, $Q_u = \{\hat{q}_7\}$, and $Q_{new} = \emptyset$.

Back in the while loop in line 4. For $\hat{q}_7 \in Q_u$, and $c \in \Sigma_u$, since $\hat{\delta}'(\hat{q}_7, c) \in \hat{Q}'_m$, the function `Check_FI` will not be

called recursively. In line 19, the variables are re-updated as follows: $Q_{u,all} = \{\hat{q}_0, \hat{q}_7\}$, $Q_u = \emptyset$, *and* $Q_{new} = \emptyset$. *Since the while loop is terminated, the function* `Check_FI`$(\hat{\mathbf{G}}', \hat{Q}' \cap \hat{Q}_{D'_E})$ *returns "Yes". We then proceed to line 5 to complete the function* `Check_FI`$(\hat{\mathbf{G}}(\Sigma, E, D', D^-), \hat{Q}_{D'_E})$. *For function* `Check_FI`$(\hat{\mathbf{G}}(\Sigma, E, D', D^-), \hat{Q}_{D'_E})$, *since* $Q_u = \{\hat{q}_0\}$ *and state* $\hat{q}_0$ *has already been checked, the variables are updated to:* $Q_{u,all} = \{\hat{q}_0\}$, $Q_u = \emptyset$, *and* $Q_{new} = \emptyset$. *Finally, the function* `Check_FI`$(\hat{\mathbf{G}}(\Sigma, E, D', D^-), \hat{Q}_{D'_E})$ *returns "Yes", confirming that the data pair* (D, D^-) *is forcibly-informatizable for the specification* E. □

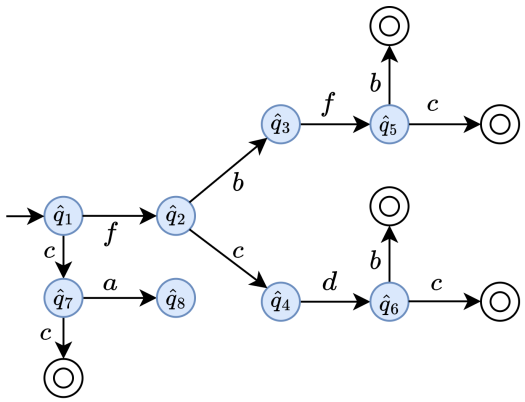


Fig. 11: A subautomaton $\hat{\mathbf{G}}' = (\hat{Q}', \Sigma', \hat{\delta}', \hat{q}'_0, \hat{Q}'_m)$ with $\hat{q}'_0 = \hat{q}_1$.

Remark 3 *The complexity of Algorithm 2 partly depends on the construction of the data-driven automaton* $\hat{\mathbf{G}}(\Sigma, E, D, D^-) = (\hat{Q}, \Sigma, \hat{\delta}, \hat{q}_0, \hat{Q}_m)$. *According to Remark 2, constructing this automaton has a complexity of* $O(|D| \cdot max\_len(D) + |Q_{D^-}|)$. *When* D^- *is finite, this complexity becomes* $O(|D \cup D^-| \cdot max\_len(D \cup D^-))$.

Now, we analyze the complexity for the function `Check_FI` *in detail. First, if line 14 does not require calling* `Check_FI` *recursively, the complexity is* $O(|\hat{Q}|^2 \cdot |\Sigma_u| \cdot |\Sigma_{for}|) = O(|\hat{Q}|^2 \cdot |\Sigma|^2)$, *where* $|\hat{Q}|$ *is the number of states of the data-driven automaton,* $|\Sigma|$ *is the size of the event set. Second, If the function* `Check_FI` *requires recursive calls at line 14, it can be called at most* $|\hat{Q}|$ *times, yielding a total complexity of:* $O(|\hat{Q}|^2 \cdot |\Sigma|^2 \cdot |\hat{Q}|) = O(|\hat{Q}|^3 \cdot |\Sigma|^2)$. *Given the complexity of constructing the data-driven automaton, the overall complexity is* $O((|D| \cdot (max\_len(D)) + |Q_{D^-}|)^3 \cdot |\Sigma|^2 + |D| \cdot max\_len(D) + |Q_{D^-}|)$. *When* D^- *is finite, this complexity is* $O(|D \cup D^-|^3 \cdot (max\_len(D \cup D^-))^3 \cdot |\Sigma|^2 + |D \cup D^-| \cdot max\_len(D \cup D^-))$. □

F. Proof of Theorem 1

Proof: (if) We prove that there exists $K \subseteq \overline{D_E}$ such that K is forcibly-controllable with respect to any plant $\mathbf{G}$ consistent with (D, D^-). First, assume the following holds:

$$(\forall s \in \overline{D_E} \cap \Sigma_u^*, \forall \sigma \in \Sigma_u)\ s\sigma \in \overline{D_E} \cup D^-.$$

By [11], it can be inferred that (D, D^-) is informatizable, which further implies (D, D^-) is also forcibly-informatizable. Second, if the statement mentioned above does not hold, we define a sublanguage $K_F = \{s \in \overline{D_E} \cap \Sigma_u^* \mid [(\exists f \in \Sigma_{for}) sf \in \overline{D_E}] \wedge \mathcal{B}(sf) \neq \emptyset]\}$.

The sublanguage K_F comprises all strings in $\overline{D_E} \cap \Sigma_u^*$ such that: 1) there exists a forcible event f satisfying $sf \in \overline{D_E}$; 2) for the string sf, $\mathcal{B}(sf) \neq \emptyset$ implies that there exists a barrier suffix language $B \subseteq \Sigma_{\overline{D_E}}(sf)$, i.e., $(\Sigma_{\overline{D_E}}(sf), \Sigma_{D^-}(sf))$ is forcibly-informative for B. Additionally, we define a corresponding sublanguage $K_{F,min} = \{s \in K_F \mid (\nexists s' \in K_F) s' \in \overline{s} \setminus \{s\}\}$, which represents the set of all minimal-length strings within K_F. Furthermore, based on the data set $K_{F,min}$, we define another sublanguage

$$K_{basic} = \{s \in \overline{D_E} \cap \Sigma_u^* \mid (\nexists s' \in K_{F,min}) s' \in \overline{s} \setminus \{s\}\}.$$

K_{basic} contains all strings in $\overline{D_E} \cap \Sigma_u^*$ that never exceed any string in $K_{F,min}$ in the sense of prefix inclusion. Finally, we define a set

$$K = K_{basic} \cup [\bigcup_{s \in K_{F,min}} \{sf\} \cdot B],$$

where $f \in \Sigma_{for}$ is an arbitrary forcible event with respect to $s \in K_{F,min}$ such that $sf \in \overline{D_E}$ and $B \in \mathcal{B}(sf)$ is an arbitrary barrier language for sf, i.e., $(\Sigma_{\overline{D_E}}(sf), \Sigma_{D^-}(sf))$ is forcibly-informative for B.

Now, we prove that K is forcibly-controllable with respect to any plant $\mathbf{G}$ consistent with (D, D^-). Note that since $\overline{K_{F,min}} \subseteq \overline{\bigcup_{s \in K_{F,min}} \{sf\}}$, we have $\overline{K_{F,min}} \subseteq \overline{K}$. First, regarding the sets K_{basic} and $K_{F,min}$, the following holds:

$$(\forall s \in \overline{K_{basic} \cup K_{F,min}}), \forall \sigma \in \Sigma_u)$$
$$[s\sigma \in \overline{K_{basic} \cup K_{F,min}} \cup D^- \subseteq \overline{K} \cup D^-]$$
$$\vee [((\exists f \in \Sigma_{for}) sf \in \overline{\bigcup_{s \in K_{F,min}} sf}) \wedge ((\nexists \sigma' \in \Sigma \backslash \Sigma_{for}) s\sigma' \in \overline{K})].$$

Next, for the set $\bigcup_{s \in K_{F,min}} \{sf\} \cdot B$ where $B \in \mathcal{B}(sf)$, by Proposition 3, the following holds:

$$(\forall s' \in \overline{B}, \forall \sigma' \in \Sigma_u)[s'\sigma'' \in \overline{B} \cup \Sigma_{D^-}(sf)] \vee$$
$$[((\exists f' \in \Sigma_{for}) s'f' \in \overline{B} \wedge ((\forall \sigma'' \in \Sigma \setminus \Sigma_{for}) s'\sigma'' \notin \overline{B})].$$

From the above, the following can be derived:

$$(\forall s'' \in \{sf\} \cdot \overline{B}, \forall \sigma'' \in \Sigma_u)[s''\sigma'' \in \{sf\} \cdot (\overline{B} \cup \Sigma_{D^-}(sf))] \vee$$
$$[((\exists f'' \in \Sigma_{for}) s''f'' \in \{sf\} \cdot \overline{B}) \wedge$$
$$((\forall \sigma''' \in \Sigma \setminus \Sigma_{for}) s''\sigma''' \notin \{sf\} \cdot \overline{B})],$$

where 1) $s''\sigma''' \notin \overline{sf}$ holds since $f \in \Sigma_c$ and $B \neq \emptyset$ by Definition 7; 2) $s''\sigma''' \notin D_{basic}$ holds since D_{basic} is defined over Σ_u^*. Analogously, we have

$$(\forall \hat{s} \in \bigcup_{s \in D_{F,min}} \{sf\} \cdot \overline{B}, \forall \hat{\sigma} \in \Sigma_u)$$
$$[\hat{s}\hat{\sigma} \in \bigcup_{s \in D_{F,min}} \{sf\} \cdot (\overline{B} \cup \Sigma_{D^-}(sf))] \vee$$
$$[((\exists \hat{f} \in \Sigma_{for}) \hat{s}\hat{f} \in \bigcup_{s \in D_{F,min}} \{sf\} \cdot \overline{B}) \wedge$$

$$((\forall \hat{\sigma}' \in \Sigma \setminus \Sigma_{for})\hat{s}\hat{\sigma}' \notin \bigcup_{s \in D_{F,min}} \{sf\} \cdot \overline{B})],$$

where $s''\sigma''' \notin \overline{s'f}$ and $\hat{s}\hat{\sigma}' \notin D_{basic}$ also holds. Hence, the following holds:

$$(\forall t \in \overline{K}, \forall \sigma_t \in \Sigma_u)[t\sigma_t \in \overline{K} \cup D^-] \vee$$

$$[(\exists f_t \in \Sigma_{for})(tf_t \in K) \wedge ((\forall \sigma_t' \in \Sigma \setminus \Sigma_{for})t\sigma_t' \notin K)],$$

which implies that (D, D^-) is forcibly-informative for K, where $K \subseteq \overline{D_E}$.

(only if) By contraposition, we prove that (D, D^-) is not forcibly-informatizable, i.e., there does not exist a sublanguage $K \subseteq \overline{D_E}$ such that (K, D^-) is forcibly-informative for K, if the following statement holds:

$$(\exists s \in \overline{D_E} \cap \Sigma_u^*, \exists \sigma \in \Sigma_u)\ [s\sigma \notin \overline{D_E} \cup D^-] \wedge$$

$$[(\forall s' \in \overline{s}, \forall f \in \Sigma_{for})s'f \notin \overline{D_E} \vee \mathcal{B}(s'f) = \emptyset].$$

For $s'f \in \overline{D_E}$, recall that $\mathcal{B}(s'f) = \{B \subseteq \Sigma_{\overline{D_E}}(s'f) \mid B \neq \emptyset \ \wedge \ (\Sigma_{\overline{D_E}}(s'f), \Sigma_{D^-}(s'f))$ is forcibly-informative for $B\}$ where $\Sigma_{\overline{D_E}}(s'f) = \Sigma_{\overline{D_E}}(s'f)\{s'' \in \Sigma^* \mid s'fs'' \in \overline{D_E}\}$ and $\Sigma_{D^-}(s'f) = \{s''' \in \Sigma^* \mid s'fs''' \in D^-\}$. Consider the string $s\sigma \notin \overline{D_E} \cup D^-$. Two cases are separately analyzed below. For the first case, consider

$$(\forall s' \in \overline{s})(\nexists f \in \Sigma_{for})s'f \in \overline{D_E}.$$

In this case, string $s\sigma$ will exceed $\overline{D_E} \cup D^-$ while no event-forcing mechanism can be exploited to achieve forcible-controllability (and forcible-informativity); hence there is no sublanguage $K \subseteq \overline{D}$ such that (K, D^-) is forcibly-informative for K.

For the second case, consider

$$(\exists s' \in \overline{s})(\exists f \in \Sigma_{for})s'f \in \overline{D_E} \wedge \mathcal{B}(s'f) = \emptyset.$$

By Proposition 3, for string $s'f$ and an arbitrary barrier language $B \in \mathcal{B}(s'f)$, the following holds:

$$(\exists t \in \overline{B}, \exists \sigma_u \in \Sigma_u)[t\sigma_u \notin \overline{B} \cup \Sigma_{D^-}(s'f)] \wedge$$

$$[(\forall t' \in \overline{t}, \forall f' \in \Sigma_{for})[t'f' \notin \overline{B}] \vee (\exists \sigma_u' \in \Sigma \setminus \Sigma_{for})t'\sigma_u' \in \overline{B}].$$

It follows that either the string $s'ft$ will eventually exceed $\overline{D_E} \cup D^-$, or the string $s'f't'\sigma_u' \in \overline{s'f \cdot B}$ will eventually be preempted by the forcible event f' thus forcible-informativity is not valid.

Therefore, no non-empty sublanguage of $\overline{D_E}$ can be proved forcibly-controllable, which implies that there does not exist a non-empty language $K \subseteq \overline{D_E}$ such that data pair (D, D^-) is forcibly-informative for K. Consequently, (D, D^-) is not informatizable for E. This concludes the proof. ■

VI. Conclusion

In this paper, we have studied data-driven analysis and supervisory control for DESs using the event-forcing mechanism and introduced two key concepts: forcible-informativity and forcible-informatizability. Forcible-informativity refers to the adequacy of the given data to determine the forcible-controllability of the specification, and is thus related to the existence of a data-driven forcing supervisory control. Moreover, forcible-informatizability allows for the enforcement of a smaller (but non-empty) specification for cases where forcible-informativity does not hold. The novel concept of barrier language is proposed for verifying forcible-informatizability. Necessary and sufficient conditions for both properties are given, together with verification algorithms of polynomial complexity with respect to the data D and D^-.

In future work, we aim to develop algorithms to synthesize the least restrictive data-driven event-forcing supervisor (if it exists) when the given data pair (D, D^-) is verified to be forcibly-informatizable. We also aim to extensively test the developed concepts and algorithms on benchmark examples. It is moreover of practical interest to consider a dynamic setting where data are collected not just one-shot but incrementally, and extend our current static solutions to adapt to newly collected data.

[1] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Springer Nature, 2021.

[2] W. M. Wonham and K. Cai, *Supervisory Control of Discrete-Event Systems*. Cham: Springer, 2019.

[3] Y. Wardi, C. G. Cassandras, and X.-R. Cao, "Perturbation analysis: A framework for data-driven control and optimization of discrete event and hybrid systems," *Annual Reviews in Control*, vol. 45, pp. 267–280, 2018.

[4] A. Farooqui, F. Hagebring, and M. Fabian, "Mides: A tool for supervisor synthesis via active learning," in *Proceedings of the 17th IEEE International Conference on Automation Science and Engineering*, pp. 792–797, IEEE, 2021.

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

[6] Z.-S. Hou and Z. Wang, "From model-based control to data-driven control: Survey, classification and perspective," *Information Sciences*, vol. 235, pp. 3–35, 2013.

[7] 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.

[8] L. Ljung, "System identification," in *Signal analysis and prediction*, pp. 163–173, Springer, 1998.

[9] 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.

[10] T. Ohtsuka, K. Cai, and K. Kashima, "Data-informativity for data-driven supervisory control of discrete-event systems," in *Proceedings of the 62nd IEEE Conference on Decisions and Control*, pp. 6917–6922, 2023.

[11] T. Ohtsuka, K. Cai, and K. Kashima, "Data-informativity for data-driven supervisory control of discrete-event systems." `https://www.control.eng.osaka-cu.ac.jp/publication/OhtsukaCaiKashima_2024.pdf`, Accessed online, 2024.

[12] B. Brandin and W. M. Wonham, "Supervisory control of timed discrete-event systems," *IEEE Transactions on Automatic control*, vol. 39, no. 2, pp. 329–342, 1994.

[13] R. Zhang, K. Cai, Y. Gan, Z. Wang, and W. M. Wonham, "Supervision localization of timed discrete-event systems," *Automatica*, vol. 49, no. 9, pp. 2786–2794, 2013.

[14] S. Miura and S. Takai, "Decentralized control of timed discrete event systems with conditional decisions for enforcement of forcible events," in *Proceedings of the 57th IEEE Conference on Decisions and Control*, pp. 3956–3961, IEEE, 2018.

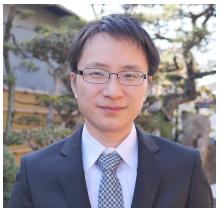
[15] A. Rashidinejad, P. van der Graaf, M. Reniers, and M. Fabian, "Non-blocking supervisory control of timed automata using forcible events," *IFAC-PapersOnLine*, vol. 53, no. 4, pp. 356–362, 2020.

[16] A. Rashidinejad, M. Reniers, and M. Fabian, "Supervisory control synthesis of timed automata using forcible events," *IEEE Transactions on Automatic Control*, 2023.

[17] B. Brandin, R. Su, and L. Lin, "Supervisory control of time-interval discrete event systems," *IFAC-PapersOnLine*, vol. 53, no. 4, pp. 217–222, 2020.
[18] C. Golaszewski and P. Ramadge, "Control of discrete event processes with forced events," in *26th IEEE Conference on Decision and Control*, vol. 26, pp. 247–251, IEEE, 1987.
[19] M. Heymann, "Concurrency and discrete event control," *IEEE Control Systems Magazine*, vol. 10, no. 4, pp. 103–112, 1990.
[20] D. Weidemann and R. Diekmann, "Optimal control design for des using supervisory control theory with enforceable events," in *2012 17th International Conference on Methods & Models in Automation & Robotics (MMAR)*, pp. 649–654, IEEE, 2012.
[21] Y. Li, F. Lin, and Z. Lin, "A generalized framework for supervisory control of discrete event systems," *International Journal of Intelligent Control and Systems*, vol. 2, no. 1, pp. 139–159, 1998.
[22] R. Diekmann and D. Weidemann, "Event enforcement in the context of the supervisory control theory," in *2013 18th International Conference on Methods & Models in Automation & Robotics (MMAR)*, pp. 783–788, IEEE, 2013.
[23] F. L. Baldissera, J. E. Cury, and J. Raisch, "A supervisory control theory approach to control gene regulatory networks," *IEEE Transactions on Automatic Control*, vol. 61, no. 1, pp. 18–33, 2015.
[24] M. Reniers and K. Cai, "Supervisory control theory with event forcing," *IEEE Transactions on Automatic Control*, pp. 1–8, doi: 10.1109/TAC.2024.3522024, 2024.
[25] C. Gu, C. Gao, and K. Cai, "Data-driven supervisory control of discrete-event systems with forcible events," in *Proceedings of the IFAC Workshop on Discrete Event Systems*, vol. 58, pp. 120–125, 2024.

Chao Gu (Member, IEEE) received the B.Eng. degree in Automation from North China Electric Power University, Baoding, China, in 2014, and the M.Eng. degree in Control Engineering from Xidian University, Xi'an, China, in 2017. In 2022 he got the Ph.D. degree in cotutorship between the School of Electro-Mechanical Engineering of Xidian University, Xi'an, China (in Control Science and Control Engineering), and the Department of Electrical and Electronic Engineering of University of Cagliari, Cagliari, Italy (in Electronics and Computer Engineering). He joined Xidian University in 2023. Previously, he was a Postdoctoral Fellow at Kyoto University, Kyoto, Japan from Oct. 2022 to Mar. 2023. His current research interests include discrete-event systems, Petri net theories, and data-driven control.

Chao Gao (Member, IEEE) received the B.Eng. degree in Communication Engineering from Science and Technology College, North China Electric Power University, Baoding, China, in 2014, the M.Eng. degree in Electronics and Communication Engineering from North China Electric Power University, Baoding, China, in 2017, and the Ph.D. degree in Electronics and Computer Engineering from the University of Cagliari, Cagliari, Italy, with the cotutorship between the School of Electro-Mechanical Engineering of Xidian University, Xi'an, China (in Control Science and Control Engineering), in 2023. She then joined the University of Le Havre Normandy as a postdoctoral researcher. Her current research interests include timed discrete event systems, timed automata, analysis of data-driven systems.



Kai Cai (Senior Member, IEEE) received the B.Eng. degree in Electrical Engineering from Zhejiang University, Hangzhou, China, in 2006; the M.A.Sc. degree in Electrical and Computer Engineering from the University of Toronto, Toronto, ON, Canada, in 2008; and the Ph.D. degree in Systems Science from the Tokyo Institute of Technology, Tokyo, Japan, in 2011. He is currently a Full Professor at Osaka Metropolitan University. Previously, he was an Associate Professor at Osaka City University (2014–2020), an Assistant Professor at the University of Tokyo (2013–2014), and a Postdoctoral Fellow at the University of Toronto (2011–2013).

Dr. Cai's research interests include cooperative control of multi-agent systems, discrete-event systems, and cyber-physical systems. He is the co-author with Z. Lin of "Directed Cooperation" (KDP 2023), and with W.M. Wonham of "Supervisory Control of Discrete-Event Systems" (Springer 2019) and "Supervisor Localization" (Springer 2016). He is serving as a Senior Editor for Nonlinear Analysis: Hybrid Systems. He was an Associate Editor for IEEE Transactions on Automatic Control (2017–2023), the Chair for IEEE CSS Technical Committee on Discrete Event Systems (2019–2022), and a member of IEEE CSS Conference Editorial Board (2017–2022). He received the Pioneer Award of SICE in 2021, the Best Paper Award of SICE in 2013, the Best Student Paper Award of IEEE Multi-Conference on Systems & Control in 2010, and the Young Author's Award of SICE in 2010.