「確率統計でもっと楽しくなるシステム制御」特集号

解　説

Data-Driven Supervisory Control of Discrete-Event Systems

Kai Cai*

1. Introduction

Discrete-event systems (DES) is a relatively new area of control science and engineering[2], which has taken its place in the mainstream of control research. A DES is a dynamical system that is *discrete*, in time and usually in state space; is *asynchronous* or *event driven*, i.e. driven by events or instantaneous happenings in time; and is *uncertain*, i.e. embodies internal chance or other unmodeled mechanisms of choice governing its state transitions. Applications of DES have been undertaken in areas such as database management, software engineering, flexible manufacturing, intelligent transportation, and logistic services. Recently, DES has been combined with continuous dynamical systems in areas called hybrid or cyber-physical systems[3,12].

In the past few years, various machine learning techniques have been successfully utilized to analyze and synthesize controllers for continuous dynamical systems. Representative approaches include identification of dynamic models from data[5], learning control laws directly from observations[13] or through reinforcement learning[7]. This research thrust has been driven by the motivation to tackle challenging control problems involving unknown system dynamics, high nonlinearity, huge dimensionality, and on the other hand the increasing availability of large quantity of observation data.

The same thrust is, at this time, obscure in the DES research. Although attempts to apply machine learning techniques exist[4,6], such works are scarce and not at all systematic. In this sense, the DES literature is still basically pre-bigdata or pre-machine learning. On the other hand, DES problems are facing similar challenges like unknown system dynamics and/or high dimensionality, as well as the opportunity of exploding amount of observation data thanks to fast advancing data collection capabilities. Hence machine learning methods may potentially make important impacts on certain challenging DES problems. With this in mind, this article aims to introduce the opportunity of developing data-driven approaches for the modeling and control of DES.

Specifically, this article will focus on introducing a model identification based approach for one of the mainstream DES control methods, the *supervisory control theory* (SCT). SCT was first proposed by Ramadge and Wonham in the 1980s[9,19,10], with the aim to formalizing general (high-level) control principles for a wide range of application domains involving man-made, computerized systems. For a comprehensive account of SCT, the reader is referred to[17]; also see[18] for a historical overview of the theory. In SCT, a DES is first modeled as a *finite-state automaton*, and its behaviors represented by *regular languages*. Then supervisory control designs are carried out based on these models; namely, a *model-based* approach of control design. If the automaton model of DES is unknown, then the model must be somehow identified or estimated before the existing supervisory control methods may be applied.

Identifying an automaton model of a system based on the observation data of the system has been investigated in the field of automata learning (an intersection of automaton theory and machine learning)[8,1,14]. Given a set of observed strings (i.e. behaviors) that must be accepted by the automaton (meaning that the behaviors are desired, positive ones), and a set of strings that must not be accepted (i.e. undesired, negative behaviors), a *prefix tree acceptor* **H** (a loopless automaton) is first constructed in which states reached by accepted strings are labeled "A" (these are the accepting states), states corresponding to non-accepted strings are labeled "N", and all other "don't care" states are not labeled. Then one tries to reduce the number of states of **H** by merging state pairs that are *consistent*: i.e. it is possible to merge a state labeled "A" (resp., "N") with either a state "A" (resp., "N") or a "don't care" state; it is also possible to merge two "don't care" states. By this state merging, a reduced automaton **G** is obtained that accepts all positive behaviors while rejects all negative behaviors. This reduced automaton **G** is the identified model from the observation data containing both positive and negative samples. That is, the dataset is labeled; thus automata learning is a *supervised learning* approach. More advanced topics of automata learning can be found in[11,15,16].

It is natural to leverage automata learning to develop a data-driven approach for supervisory control of DES. The approach consists of first applying au-

* This work was supported in part by JSPS KAKENHI Grant no. 21H0487 and the 2021 Osaka City University Strategic Research Grant to Encourage Applicants for Upper Level KAKENHI. K. Cai is with Department of Core Informatics, Osaka Metropolitan University. Email: cai@omu.ac.jp

Key Words: discrete-event systems, automata, data-driven methods, supervisory control.

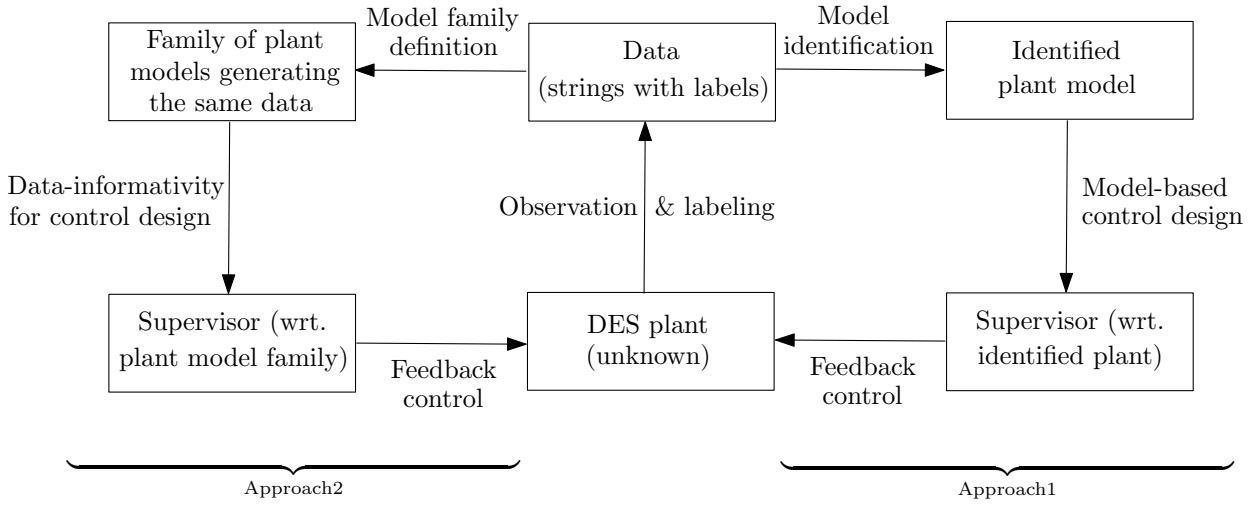


Fig. 1 Two data-driven approaches to supervisory control of discrete-event systems

tomata learning to identify an automaton model of DES from labeled observation data, and then applying standard SCT to design supervisory controllers based on the identified DES model. This is outlined as "Approach 1" in Fig. 1. This approach is the focus of this article, which will be introduced in Sections **2**–**4**.

A second data-driven approach that this article will briefly touch upon is to learn control laws directly from observation data, without going through a model identification step. This is outlined as "Approach 2" in Fig. 1, and briefly introduced in Section **5**. This approach is inspired by a recently developed method based on a new concept called *data-informativity*, which requires that observation data contain sufficient information such that a valid supervisor may be designed for a family of plant models that all can generate the same observed data.

2. Supervisory Control in a Nutshell

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

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

Here Q is the finite *state set*, Σ the finite *event set*, $\delta : Q \times \Sigma \to Q$ the (partial) *state transition function*, $q_0 \in Q$ the *initial state*, and $Q_m \subseteq Q$ the subset of *marker states* (or target states). A sequence of events in Σ is called a *string*, and write Σ^* for the set of all finite-length strings of Σ, including the empty string ϵ (containing no event). Then the state transition function δ may be inductively defined such that $\delta : Q \times \Sigma^* \to Q$. Write $\delta(q,s)!$ to mean string s is defined at state q. We say that the automaton $\mathbf{G}$ is *deterministic* if

$$(\forall q \in Q)(\forall s \in \Sigma^*)\delta(q,s)! \Rightarrow |\delta(q,s)| = 1.$$

Namely the destination state of every state transition is unique. We shall focus exclusively on deterministic automata unless otherwise stated.

The *closed behavior* $L(\mathbf{G})$ of $\mathbf{G}$ is defined as the set of all strings of Σ^* which $\mathbf{G}$ can generate starting from the initial state q_0:

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

A central subset of $L(\mathbf{G})$, called the *marked behavior* of $\mathbf{G}$, is the collection of strings that can reach a marker state:

$$L_m(\mathbf{G}) := \{s \in L(\mathbf{G}) \mid \delta(q_0, s) \in Q_m\}.$$

To formulate a control problem for $\mathbf{G}$, we first adjoin a control mechanism by which $\mathbf{G}$ may be actuated to affect its behavior; that is, determine the strings it is permitted to generate. To this end we assume that a subset of events $\Sigma_c \subseteq \Sigma$, called the *controllable events*, are capable of being enabled or disabled by an external controller. The complementary event subset $\Sigma_u := \Sigma \setminus \Sigma_c$ is *uncontrollable*; events in Σ_u cannot be externally disabled and must be considered permanently enabled.

Consider a control specification given as a language $E \subseteq \Sigma^*$, describing desired behaviors to be enforced. Write $\overline{E} := \{s \in \Sigma^* \mid (\exists s' \in \Sigma^*) ss' \in E\}$ for the set of all *prefix strings* of E. A fundamental concept of supervisory control is the following: The language E is *controllable* (with respect to $\mathbf{G}$) provided

$$\begin{aligned} &\text{for all } s \in \overline{E} \text{ and for all } \sigma \in \Sigma_u, \\ &\quad \text{if } s\sigma \in L(\mathbf{G}) \text{ then } s\sigma \in \overline{E}. \end{aligned} \tag{2}$$

The basic result of supervisory control then is: The specification language E being controllable is necessary and sufficient for the existence of a *supervisor* $\mathbf{S}$ that disables only controllable events in Σ_c such that

$$L_m(\mathbf{S}) = L_m(\mathbf{G}) \cap E.$$

The supervisor $\mathbf{S}$ is also an automaton, which controllably restricts the plant $\mathbf{G}$'s behavior to realize the imposed specification E.

Even if E fails to be controllable, we can still de-

Observed strings in $L(\mathbf{G})$	Membership in $L_m(\mathbf{G})$
$s_1 = \alpha\beta\beta$	+
$s_2 = \alpha\beta\alpha\alpha$	+
$s_3 = \alpha$	−
$s_4 = \beta\alpha\beta$	+
$s_5 = \beta\beta$	−

Table 1 Running example: observed data with labels

sign a *maximally permissive* supervisor $\mathbf{S}^*$ that optimally approximates E. Let $K := L_m(\mathbf{G}) \cap E$ and define the family of controllable sublanguages of K:

$$\mathcal{C}(K) := \{K' \subseteq K \mid K' \text{ is controllable (wrt. } \mathbf{G})\}.$$

This $\mathcal{C}(K)$ contains a unique supremal element, written $\sup\mathcal{C}(K)$ (owing to the fact that the language controllability is closed under arbitrary set unions), which is the largest controllable sublanguage of K. Hence, whenever $\sup\mathcal{C}(K) \neq \emptyset$ there exists an optimal supervisor $\mathbf{S}^*$ such that

$$L_m(\mathbf{S}^*) = \sup\mathcal{C}(K) \subseteq L_m(\mathbf{G}) \cap E.$$

The above is all the basic supervisory control theory that is needed in this article. The theory is *model-based*. In the following sections, by contrast, we consider the alternative where the plant model is unknown but certain relevant *data* observed from the plant are available.

3. Automata Learning

Consider a plant whose automaton model $\mathbf{G}$ is unknown (i.e. internal working unknown), but whose behaviors can be externally observed. As a running example, consider five strings generated by $\mathbf{G}$ are observed and recorded in Table 1.

Denote by D the set of all observed strings, and call this set the *observation dataset*. Since each string in D is observed from $\mathbf{G}$, the dataset is a subset of the closed behavior of $\mathbf{G}$: i.e. $D \subseteq L(\mathbf{G})$.

Moreover, suppose that each string in the dataset D is labeled according to whether or not it belongs to the marked behavior $L_m(\mathbf{G})$. Thus this is a setting of *supervised learning*, where an expert exists for correctly labeling the 'raw data'. For the example in Table 1, the + label means that the observed string is in $L_m(\mathbf{G})$, while the − label means the opposite.

Write D^+ (resp. D^-) for the set of observed strings with the + label (resp. with the − label). Precisely

$$D^+ = \{s \in D \mid s \in L_m(\mathbf{G})\}$$
$$D^- = \{s \in D \mid s \notin L_m(\mathbf{G})\}.$$

Then $D = D^+ \cup D^-$ and

$$D^+ \subseteq L_m(\mathbf{G}), \quad D^- \cap L_m(\mathbf{G}) = \emptyset.$$

The problem of automata learning is to construct an estimate automaton $\hat{\mathbf{G}}$ based on the labeled dataset

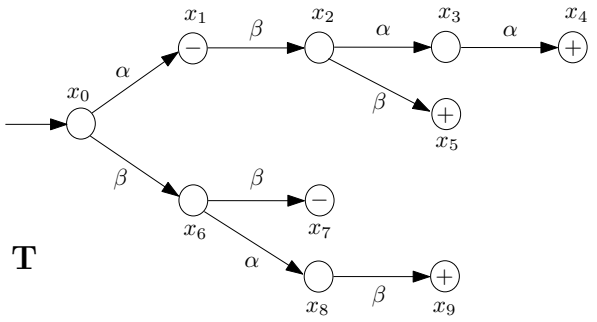


Fig. 2 Running example: prefix tree automaton

D, such that

$$D \subseteq L(\hat{\mathbf{G}}), \quad D^+ \subseteq L_m(\hat{\mathbf{G}}), \quad D^- \cap L_m(\hat{\mathbf{G}}) = \emptyset. \quad (3)$$

In words, the constructed $\hat{\mathbf{G}}$ should (i) be capable of generating all observed strings, (ii) contain all the 'positive data' (strings belong to $L_m(\mathbf{G})$) in its marked behavior, and (iii) contain no 'negative data' in its marked behavior. In this sense, $\hat{\mathbf{G}}$ is *consistent* with the true but unknown $\mathbf{G}$.

In general there exist many $\hat{\mathbf{G}}$ consistent with $\mathbf{G}$. Of particular interest is $\hat{\mathbf{G}}$ with the minimum number of states. This is the minimum-state automata learning problem, which (unfortunately) turns out to be NP-hard. To derive sub-optimal (wrt. state number) automaton $\hat{\mathbf{G}}$, many approaches have been proposed[8,1,14]. In the following we illustrate a widely used approach based on *state merging*.

The approach consists of two steps. The first step is construction of a *prefix tree automaton* $\mathbf{T}$. This $\mathbf{T}$ is constructed by going through each string in the dataset D, and adding states to $\mathbf{T}$ for generating all the prefix strings. Then label each state of $\mathbf{T}$ reached by a string in D^+ with the + symbol, whereas each state reached by a string in D^- with the − symbol, and all the other states unlabled. For the running example whose dataset is in Table 1, the constructed prefix tree automaton $\mathbf{T}$ is displayed in Fig. 2.

Two comments are immediate. First, the prefix tree automaton $\mathbf{T}$ contains no loops. Second, $\mathbf{T}$ itself is already a solution to the automaton learning problem (i.e. conditions in (3) are satisfied). However, $\mathbf{T}$ is a trivial solution in that it simply enumerates states, the number of which is unnecessarily large (especially for large datasets). Hence the second step of the approach is to suitably merge the states of $\mathbf{T}$ to achieve as much state reduction as possible.

When merging states, two levels of consistency need to be respected. The first is label consistency. It is possible to merge a state labeled + with another state labeled either + or not labeled. Similarly, it is also possible to merge a state labeled − with another state labeled either − or not labeled. In other words, it is (only) not possible to merge a + state with a − state. For example in Fig. 2, states x_7 and x_9 cannot be merged.

The second consistency that needs to be respected

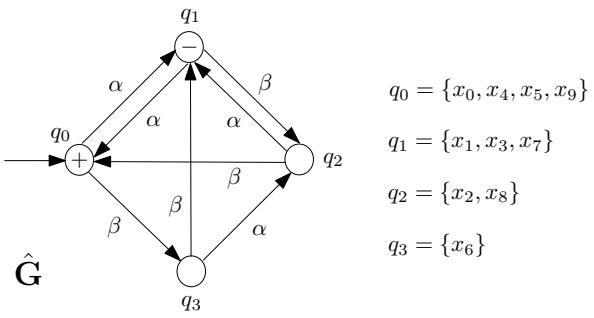


Fig. 3 Running example: state reduced automaton

is the transitional relation. Two label-consistent states may be merged if their common future states reached by the same strings can also be merged. That is, state merging must respect the dynamics of the automaton. This is to ensure that the resulting (state-reduced) automaton is deterministic. As an example, in Fig. 2 although states x_6 and x_8 are label consistent (both are not labeled), they have a pair of common future states x_7,x_9 (reached by β) which cannot be merged. As a result, states x_6 and x_8 cannot be merged. If they were merged, the resulting automaton is nondeterministic: from the merged state containing x_6,x_8, it is possible to reach two different states (one containing x_7, the other containing x_9) by β, i.e. the destination state is not unique.

Respecting the above mentioned two levels of consistency, a valid state merging scheme for the example in Fig. 2 is given by the partition $\mathcal{P} = \{C_1, C_2, C_3, C_4\}$:

$$C_1 = \{x_0, x_4, x_5, x_9\}, \quad C_2 = \{x_1, x_3, x_7\},$$
$$C_3 = \{x_2, x_8\}, \quad C_4 = \{x_6\}.$$

The resulting state-reduced automaton $\hat{\mathbf{G}}$ is displayed in Fig. 3. Observe that while $\mathbf{T}$ is loopless, $\hat{\mathbf{G}}$ contains loops and is deterministic. More importantly, it is easily verified that $\hat{\mathbf{G}}$ is a valid solution to the automata learning problem: all five strings are in $L(\hat{\mathbf{G}})$, $s_1, s_2, s_4 \in L_m(\hat{\mathbf{G}})$, and $s_2, s_5 \notin L_m(\hat{\mathbf{G}})$. For this example, in fact, the 4-state $\hat{\mathbf{G}}$ is the minimum-state solution that can be achieved.

4. Model Identification based Supervisory Control

With the method of automata learning, a straightforward approach to data-driven supervisory control is to first identify a plant model from observation data, and then design a supervisor based on the identified model.

We state the problem formally.

Problem: data-driven supervisory control (DDSC) Consider a plant whose model $\mathbf{G} = (Q, \Sigma, \delta, q_0, Q_m)$ is unknown, and an observation dataset $D \subseteq L(\mathbf{G})$ is available. Moreover D is labeled such that $D = D^+ \cup D^-$ where

$$D^+ = \{s \in D \mid s \in L_m(\mathbf{G})\}$$
$$D^- = \{s \in D \mid s \notin L_m(\mathbf{G})\}.$$

Given a control specification $E \subseteq \Sigma^*$, design a supervisor $\hat{\mathbf{S}}$ based on the labeled dataset D such that

$$\emptyset \neq L_m(\mathbf{G}) \cap L_m(\hat{\mathbf{S}}) \subseteq L_m(\mathbf{G}) \cap E.$$

The last line means that the supervisor $\hat{\mathbf{S}}$ designed based on the dataset D is nontrivial (i.e. the controlled behavior is nonempty) and satisfies the imposed specification.

To solve this problem, the following approach combines the methods introduced in Sections **2** and **3**.

Approach: model identification based supervisory control design (MISCD)
Step 1. Apply the method of automata learning to derive from the labeled dataset D an estimate plant model $\hat{\mathbf{G}} = (\hat{Q}, \hat{\Sigma}, \hat{\delta}, \hat{q}_0, \hat{Q}_m)$.
Step 2. Apply the method of supervisory control to derive a supervisor $\hat{\mathbf{S}}$, which enforces the specification E for the estimate plant $\hat{\mathbf{G}}$.

Several facts follow directly from the results of supervisory control (Section **2**). First, the derived supervisor $\hat{\mathbf{S}}$ from MISCD satisfies

$$L_m(\hat{\mathbf{S}}) = L_m(\hat{\mathbf{G}}) \cap E$$

if and only if E is controllable wrt. $\hat{\mathbf{G}}$. Second, whenever $\sup\mathcal{C}(L_m(\hat{\mathbf{G}}) \cap E) \neq \emptyset$, there exists an optimal supervisor $\hat{\mathbf{S}}^*$ such that

$$L_m(\hat{\mathbf{S}}^*) = \sup\mathcal{C}(L_m(\hat{\mathbf{G}}) \cap E) \subseteq L_m(\hat{\mathbf{G}}) \cap E.$$

The above assertions mean that if we treat the identified plant model $\hat{\mathbf{G}}$ as the true plant and consider that the specification E is imposed on $\hat{\mathbf{G}}$, then the derived supervisor $\hat{\mathbf{S}}$ (or the optimal $\hat{\mathbf{S}}^*$) is valid for $\hat{\mathbf{G}}$ as a direct consequence of supervisory control.

The identified $\hat{\mathbf{G}}$, however, is not the true plant $\mathbf{G}$. There is a gap between $\hat{\mathbf{G}}$ and $\mathbf{G}$, and how closely $\hat{\mathbf{G}}$ approximates $\mathbf{G}$ depends on the quality and/or richness of the labeled dataset D. Since the formulated problem DDSC seeks a supervisor to impose the specification E on the true plant $\mathbf{G}$, a relation between $\hat{\mathbf{G}}$ and $\mathbf{G}$ is needed.

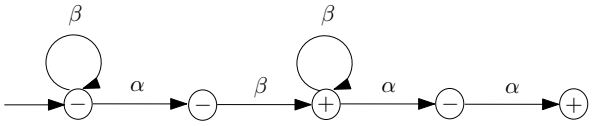
We know from the result of automata learning that the learned plant model ensures that $\hat{\mathbf{G}}$ satisfies (3). Based on this knowledge, the following condition is key:

$$D^+ \cap E \text{ is nonempty and controllable wrt. } \mathbf{G} \text{ and } \hat{\mathbf{G}} \tag{4}$$

If (4) holds and we know $D^+ \subseteq L_m(\hat{\mathbf{G}})$, it follows that the family $\mathcal{C}(L_m(\hat{\mathbf{G}}) \cap E)$ contains at least one nonempty element which is controllable wrt. $\mathbf{G}$ (i.e. $D^+ \cap E$). Hence the supemal element satisfies

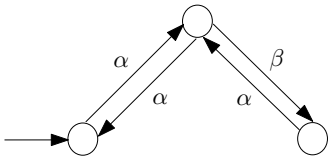
$$\sup\mathcal{C}(L_m(\hat{\mathbf{G}}) \cap E) \supseteq D^+ \cap E \neq \emptyset.$$

Accordingly there exists an optimal supervisor $\hat{\mathbf{S}}^*$ such that $L_m(\hat{\mathbf{S}}^*) = \sup\mathcal{C}(L_m(\hat{\mathbf{G}}) \cap E)$; hence



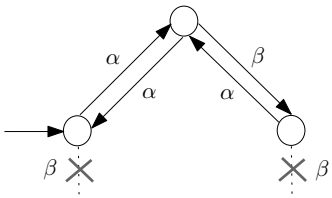
G

Fig. 4 Running example: true plant model



E

Fig. 5 Running example: control specification



$\hat{\mathbf{S}}$

Fig. 6 Running example: estimated supervisor

$$L_m(\mathbf{G})\cap L_m(\hat{\mathbf{S}}^*) = L_m(\mathbf{G})\cap \sup\mathcal{C}(L_m(\hat{\mathbf{G}})\cap E) \supseteq D^+\cap E \neq \emptyset.$$

On the other hand, it follows from

$$L_m(\mathbf{G})\cap L_m(\hat{\mathbf{S}}^*) = L_m(\mathbf{G})\cap \sup\mathcal{C}(L_m(\hat{\mathbf{G}})\cap E) \subseteq E$$
$$L_m(\mathbf{G})\cap L_m(\hat{\mathbf{S}}^*) \subseteq L_m(\mathbf{G})$$

that

$$L_m(\mathbf{G})\cap L_m(\hat{\mathbf{S}}^*) \subseteq L_m(\mathbf{G})\cap E.$$

In conclusion, if the key condition (4) holds, then the MISCD approach solves the DDSC problem.

For the running example in Section **3**, suppose that the true (unknown) plant model is the automaton displayed in Fig. 4 and the imposed specification $E = L_m(\mathbf{E})$ where the automaton $\mathbf{E}$ is in Fig. 5. Also let α be an uncontrollable event while β a controllable event. First compute

$$D^+\cap E = \{\alpha\beta\alpha\alpha\}.$$

It is verified that $D^+\cap E$ is controllable wrt. both the true plant $\mathbf{G}$ (Fig. 4) and the estimated plant $\hat{\mathbf{G}}$ (Fig. 3). Therefore condition (4) holds. Hence by the MISCD approach, we derive the estimated supervisor $\hat{\mathbf{S}}$ as displayed in Fig. 6. Observe that in $\hat{\mathbf{S}}$ the controllable event β is disabled at two states so that the specification E is enforced on $\hat{\mathbf{G}}$. Since

$$L_m(\mathbf{G})\cap L_m(\hat{\mathbf{S}}) = D^+\cap E = \{\alpha\beta\alpha\alpha\}$$

we conclude that this estimated supervisor $\hat{\mathbf{S}}$ constructed from the observed dataset D satisfies the requirement of the DDSC problem.

5. Data Informativity

In this section we briefly outline another data-driven approach that directly designs control laws from observation data. In contrast with the approach in Section **4**, this one circumvents the step of (explicitly) identifying automaton models.

Consider a plant whose model $\mathbf{G} = (Q,\Sigma,\delta,q_0,Q_m)$ is unknown, and an observation dataset $D\subseteq L(\mathbf{G})$ is available. For simplicity assume that $Q_m = Q$, i.e. all states are marked; hence $L_m(\mathbf{G}) = L(\mathbf{G})$. Given a control specification $E\subseteq\Sigma^*$, the goal is to design based on D a supervisor that satisfies E.

To this end, define the family of all automata that can generate the dataset D:

$$\mathcal{M}(D) = \{\mathbf{G}' \mid D\subseteq L(\mathbf{G}')\}.$$

It is evident that the true $\mathbf{G}\in\mathcal{M}(D)$. Call $\mathcal{M}(D)$ the *model family* wrt. D. The idea is: if a supervisor can be designed to make *every* automaton in the model family $\mathcal{M}(D)$ satisfy the imposed specification E, then this supervisor is also valid for the true, unknown plant model $\mathbf{G}$.

For this to be possible, consider the following condition:

$$(\forall \mathbf{G}'\in\mathcal{M}(D))\ \ D\cap E \text{ is nonempty and controllable wrt. } \mathbf{G}'. \tag{5}$$

This means that the dataset D needs to be 'rich' enough such that the language $D\cap E$ is nontrivial and controllable for the whole model family. We say that D satisfying this condition (5) is *informative*.

Suppose that (5) holds. Then for an arbitrary automaton $\mathbf{G}'\in\mathcal{M}(D)$, the supemal controllable sublanguage

$$\emptyset \neq D\cap E\subseteq \sup\mathcal{C}(L(\mathbf{G}')\cap E)\subseteq E.$$

Accordingly there exists an optimal supervisor $\mathbf{S}^*$ such that

$$L(\mathbf{G}')\cap L(\mathbf{S}^*) = \sup\mathcal{C}(L(\mathbf{G}')\cap E).$$

Therefore this supervisor $\mathbf{S}^*$ is valid for the model family, and in particular, for the true plant model $\mathbf{G}$.

In conclusion, if the data-informativity condition (5) holds, then a valid supervisor can be designed based on dataset D for the unknown plant G to satisfy an imposed specification E (without explicitly identifying the model of G).

6. Concluding Remarks

In this article, a model identification based data-driven supervisory control approach has been introduced. The model identification part leverages automata learning methods to construct a model from a

labeled observation dataset; then the identified model is used to synthesize a supervisor to enforce an imposed specification for the original unknown plant. A second data-driven supervisory control approach that does not explicitly identify automaton models has also been briefly outlined.

The key conditions (4) and (5) can impose strong requirements on the observation datasets; whether or not such datasets are practically available or collectable is case dependent. Hence an important direction along this line of research is to find weaker (or practically more reasonable) conditions for enforcing control specifications (that may need to be suitably downgraded). An alternative direction is to design data collection strategies such that datasets satisfying the imposed requirements can be made available.

(2022 年 3 月 24 日受付)

References

[1] M. Bugalho and A. Oliveira: Inference of regular languages using state merging algorithms with search. *Pattern Recognition*, 38(9):1457–1467, 2005.

[2] K. Cai and W. M. Wonham: *Supervisory control of discrete-event systems.* Encyclopedia of Systems and Control, 2nd ed., Springer, 2020.

[3] J. Cury, B. Krogh and T. Ninomi: Synthesis of supervisory controllers for hybrid systems based on approximating automata. *IEEE Trans. Autom. Control*, 43(4):564–568, 1998.

[4] A. Farooqui, F. Hagebring and M. Fabian: MIDES: A tool for supervisor synthesis via active learning. In *IEEE Int. Conf. on Automation Science and Engineering*, page 792–797, 2021.

[5] M. Gevers, A. S. Bazanella, X. Bombois and L. Miskovic: Identification and the information matrix: How to get just sufficiently rich? *IEEE Trans. Autom. Control*, 54(12):2828–2840, 2009.

[6] M. Konishi, T. Sasaki and K. Cai: On efficient safe control based on supervisory control theory and deep reinforcement learning. In *Japan Joint Automatic Control Conference*, page 388–393, 2021.

[7] S. Mukherjee, H. Bai and A. Chakrabortty: On model-free reinforcement learning of reduced-order optimal control for singularly perturbed systems. In *IEEE Conf. on Decision and Control*, page 5288–5293, 2018.

[8] J. Oncina and P. Garcia: Inferring regular languages in polynomial update time. *Pattern Recognition and Image Analysis, World Scientific*, 1:49–61, 1992.

[9] P. J. Ramadge and W. M. Wonham: Supervisory control of a class of discrete event processes. *SIAM J. Control and Optimization*, 25(1):206–230, 1987.

[10] P. J. Ramadge and W. M. Wonham: The control of discrete event systems. *Proc. IEEE*, 77(1):81–98, 1989.

[11] B. Steffen, F. Howar and M. Isberner: Active automata learning: from dfas to interface programs and beyond. In *Int. Conf. on Grammatical Inference*, page 195–209, 2012.

[12] P. Tabuada and G. J. Pappas: Linear time logic control of discrete-time linear systems. *IEEE Trans. Autom. Control*, 51(12):1862–1877, 2006.

[13] 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 Trans. Autom. Control*, 65(11):4753–4768, 2020.

[14] S. Verwer, M. de Weerdt and C. Witteveen: Identifying an automaton model for timed data. In *Proc. of the 15th Annual Machine Learning Conference of Belgium and the Netherlands*, pages 57–64, 2006.

[15] S. Verwer, M. de Weerdt and C. Witteveen: Efficiently identifying deterministic real-time automata from labeled data. *Machine learning*, 86(3):295–333, 2012.

[16] S. Verwer, R. Eyraud and C. de la Higuera: Pautomac: a probabilistic automata and hidden markov models learning competition. *Machine learning*, 96(1):129–154, 2014.

[17] W. M. Wonham and K. Cai: *Supervisor Control of Discrete-Event Systems.* Communications and Control Engineering, Springer, 2016.

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

[19] W. M. Wonham and P. J. Ramadge: On the supremal controllable sublanguage of a given language. *SIAM J. Control and Optimization*, 25(3):637–659, 1987.

Author

Kai CAI

Kai CAI 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 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 distributed control of discrete-event systems and cooperative control of multi-agent systems. He is the co-author (with W.M. Wonham) of Supervisory Control of Discrete-Event Systems (Springer 2019) and Supervisor Localization (Springer 2016). He is serving as the Chair for the IEEE CSS Technical Committee on Discrete Event Systems and an Associate Editor for the IEEE Transactions on Automatic Control. He was the recipient of the Pioneer Award of SICE in 2021, the Best Paper Award of SICE in 2013, the Best Student Paper Award of the IEEE Multi-Conference on Systems and Control, and the Young Author's Award of SICE in 2010.