IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 1, 2019, accepted November 14, 2019, date of publication November 20, 2019, date of current version December 4, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2954557

A New Supervisor Synthesis Approach for Discrete-Event Systems With Infinite Behavior

WEI TANG[1], TING JIAO[2], AND RENYUAN ZHANG[1]
[1]School of Automation, Northwestern Polytechnical University, Xi'an 710072, China
[2]Department of Automation, Shanxi University, Taiyuan 030006, China

Corresponding author: Ting Jiao (tjiao@sxu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61573289, in part by the Fundamental Research Funds for the Central Universities of China under Grant 3102019ZDHKY11, in part by the Fund Program for the Scientific Activity of Selected Returned Overseas Professionals in Shaanxi Province under Grant 2018024, and in part by the Scientific and Technological Innovation Programs of Higher Education Institutions in Shanxi under Grant 2019L0051.

ABSTRACT $\omega-$*closure* (together with $\omega-$*controllability*) plays an important role in the supervisor synthesis for liveness specifications imposed on discrete-event systems (DESs) with infinite behavior. However, it is not closed under arbitrary unions, and thus there does not exist the supremal $\omega-$closed (and $\omega-$controllable) sublanguage. In this paper, we first develop an algorithm to compute a subautomaton of the automaton representing an arbitrarily given ω-language by deleting all *bad loops* (which lead to un-closedness of the given language) in the automaton, and show that the obtained subautomaton represents an $\omega-$closed sublanguage and is maximal from the perspective of an automaton transition graph. With this algorithm, we propose a new approach to construct a supervisor synthesizing an $\omega-$language even if it is not $\omega-$closed. Furthermore, we prove that, compared with the supervisor computed by the Thistle's supervisor synthesis approach, the supervisor synthesized by our approach is often more permissive. Examples are presented for illustration.

INDEX TERMS Discrete-event systems, infinite behavior, $\omega-$closure, $\omega-$automata.

I. INTRODUCTION

Generally, there are two kinds of specifications in discrete-event systems (DESs) [1]–[3]. One is the safety specifications requiring that some conditions should not occur; the other is the liveness specifications requiring that some conditions must be met eventually [4]. For specifications restricting the set of finite trajectories, corresponding to safety specifications, they can be formalized as $*-$languages. For liveness specifications, $\omega-$languages are employed to model the infinite behavior.

Ramadge pioneered the research to model DESs by Büchi automata and extended the concept of $*-$controllability of $*-$languages first introduced in [2] to the case of $\omega-$languages [5]. Thistle and Wonham continued Ramadge's work to study the supervisory control of infinite behavior of DESs. In [6]–[8], the plant was represented by a deterministic Büchi automaton and the specification of legal behavior by a deterministic Rabin automaton. The definition of $\omega-$controllablility was proposed and the existence of the supremal $\omega-$controllable sublanguage was shown, which facilitated the solvability of infinitary supervisory control problem.

However, the $\omega-$controllable and $\omega-$closed languages proposed in [6]–[8] have different closure properties under union and intersection. In particular, $\omega-$controllability is preserved under arbitrary unions but not intersections, and $\omega-$closure is preserved under arbitrary intersections but not arbitrary unions. Thus, there does not exist the supremal $\omega-$controllable and $\omega-$closed sublanguage of a given language. We note that if a given specification language is $\omega-$closed, then we only need to tackle its $\omega-$controllability, which will facilitate the supervisor synthesis. We also note that for a given $\omega-$language K, the transitions leading to unclosedness of K are the loops defined in the transition graph of the automaton representing K, but not occurred infinitely; we call these loops as *bad loops* (formally defined in Section III). More importantly, such bad loops can be detected by the known depth-first-search (DFS) mechanism [24] on graphs. Thus in this paper, we propose an algorithm based on the DFS mechanism to construct a subautomaton by deleting all the bad loops; the resultant automaton

The associate editor coordinating the review of this manuscript and approving it for publication was Shouguang Wang.

obviously represents an $\omega-$closed language. Then, we show that the obtained automaton is maximal from the perspective of the automaton transition graph. Combining this algorithm with the one computing the supremal $\omega-$controllable sublanguage (proposed by Thistle and Wonham in [6], [8]), we may construct a supervisor synthesizing a sublanguage of K even if it is not $\omega-$closed. Furthermore, we prove that compared with the supervisor computed by the Thistle's supervisor synthesis approach, the synthesized supervisor by our approach is often more permissive.

The main contributions of this paper are two-fold. First, we propose an algorithm to construct a subautomaton of the automaton representing an $\omega-$language, and show that the obtained subautomaton represents an $\omega-$closed sublanguage of the given language and is maximal from the perspective of the automaton transition graph. Second, we propose a new approach to synthesize a supervisor for a given liveness specification, which is often more permissive than that synthesized by the Thistle's supervisor synthesis approach.

The control problem of DESs with infinite behavior also has been studied intensively by other researchers. Park and Cho proposed a state feedback supervisory control for real-time DESs with infinite time of untimed states [9]. The resultant supervisor restricted the behavior of the controlled timed DESs within the legal behavior. Oliveira, Cury and Kaestner presented a supervisor synthesis method for parameterized and infinity non-regular DESs [10], in which the plant incorporated a finite state transition system equipped with a data collection. Cury and Krogh designed robust supervisors for nominal plants to satisfy specifications with infinite behavior [11]. Kumar and Garg presented a finitely terminating algorithm for maximally permissive supervision of state avoidance control for infinite state systems in assignment program framework [12]. Chédor *et al*. studied the diagnosis and opacity problems for infinite state systems [13]. Garg and Kumar proposed a state-variable approach for the supervisory control of DESs with infinite states [14]. Gohari and Wonham employed first-in-first-out queues to achieve bounded fairness, which was often described as $\omega-$languages [15]. In [5], [16], deterministic Büchi automata were used to model DESs. Zhang and Cai designed a localization procedure to achieve supervisor localization of DESs with infinite behavior [17].

In addition to modeling with $\omega-$automata for DESs with infinite behavior, some other studies investigate the supervisory control problem based on Petri Nets (PNs), which are also capable of modeling DESs with infinite behavior. Lu *et al.* proposed complex reachability trees to solve the deadlock detection problem of unbounded PNs, in which the set of reachable markings were infinite [18]. A new reachability tree was constructed in [19] to characterize precisely their infinite reachability sets for unbounded PNs with semilinear reachability sets. A lean reachability tree for unbounded PNs was proposed in [20]. Behavior consistency was calculated in [21] for unbounded PNs with infinite branching process. A three-stage iterative deadlock prevention policy for systems of simple sequential processes with resources (S^3PR) with $\omega-$siphons was proposed in [22]. Reference [23] proposed a sufficient and necessary condition for a resource subset to generate a strict minimal siphon in systems of sequential systems with shared resources (S^4PR).

The paper is organized as follows. Section II provides basic preliminaries on supervisory control of DESs with infinite behavior. Section III develops an algorithm to obtain the maximal subautomaton representing an $\omega-$closed sublanguage of a given $\omega-$language. Section IV proposes a new approach to solve the supervisory control problem of DESs with infinite behavior. Section V illustrates our results with a case study on a small factory. Section VI concludes this paper.

II. PRELIMINARIES

Let Σ be a finite alphabet. Let Σ^* denote the set of all finite strings over Σ and Σ^ω denote the set of all infinite strings over Σ. Let $\Sigma^\infty := \Sigma^* \dot{\cup} \Sigma^\omega$.

For any $k \in \Sigma^*$, $v \in \Sigma^\infty$, write $k \leq v$ if k is a *prefix* of v. Define the map pre : $2^{\Sigma^\infty} \to 2^{\Sigma^*}$ by

$$\text{pre} : V \mapsto \{k \in \Sigma^* | (\exists v \in V) k \leq v\}.$$

The *limit* of a $*-$language is given by

$$\lim(K) := \text{pre}^{-1}(K) \cap \Sigma^\omega,$$

where $\text{pre}^{-1} : 2^{\Sigma^*} \to 2^{\Sigma^\infty}$ is the inverse of pre : $2^{\Sigma^\infty} \to 2^{\Sigma^*}$.

Define operator clo : $2^{\Sigma^\infty} \to 2^{\Sigma^\infty}$ as

$$\text{clo} : R \mapsto \lim(\text{pre}(R)) = \text{pre}^{-1}(\text{pre}(R)) \cap \Sigma^\omega.$$

clo(R) is called the $\omega-$*closure* of R. R is $\omega-$*closed* if $R = \text{clo}(R)$. R is $\omega-$*closed with respect to* S if $R = \text{clo}(R) \cap S$, where $S \subseteq \Sigma^\omega$.

A DES with infinite behavior (plant to be controlled) is modeled as a (deterministic) *Büchi automaton*

$$\mathbf{G} := (Q, \Sigma, \delta, q_0, \mathcal{B}_Q),$$

where Q is the finite state set, q_0 is the initial state, Σ is the finite event set, $\delta : Q \times \Sigma \to Q$ is the (partial) state transition function, and $\mathcal{B}_Q \subseteq Q$ is the Büchi acceptance criterion.

The notation $\delta(q, \sigma)!$ means that $\delta(q, \sigma)$ is defined. Moreover, δ is extended to a partial function $\delta : Q \times \Sigma^* \to Q$ by the rules

$$\delta(q, \epsilon) = q$$
$$\delta(q, s\sigma) = \delta(\delta(q, s), \sigma)$$

provided $q' := \delta(q, s)!$ and $\delta(q', \sigma)!$. In this paper, we also write $q' := \delta(q, s)!$ as $q \xrightarrow{s} q'$.

The *finite behavior* of $\mathbf{G}$ is the $*-$language $L(\mathbf{G}) \subseteq \Sigma^*$ accepted by the automaton (Q, Σ, δ, q_0), i.e. $L(\mathbf{G}) := \{s \in \Sigma^* | \delta(q, s)!\}$; and the *infinite behavior* of $\mathbf{G}$ is the $\omega-$language $S(\mathbf{G})$ accepted by $\mathbf{G}$ with the Büchi acceptance criterion $\mathcal{B}_Q$, i.e.

$$S(\mathbf{G}) := \{s \in \Sigma^\omega | \Omega(s) \cap \mathcal{B}_Q \neq \emptyset\},$$

where $\Omega(s)$ is set of states that s visits infinitely often. For the supervisory control of $\mathbf{G}$, event set Σ is partitioned into the subset of controllable events Σ_c and the subset of uncontrollable events Σ_u. A supervisory control for $\mathbf{G}$ is any map $f : L(G) \to \Gamma$, where $\Gamma := \{\gamma \subseteq \Sigma | \gamma \supseteq \Sigma_u\}$. Then the finite and infinite closed-loop behaviors of the controlled DES $\mathbf{G}^f$, representing the action of supervisor f on $\mathbf{G}$, are respectively given by

1) $L(\mathbf{G}^f)$, the $*-$language synthesized by f, defined by the following recursion:
   a) $\epsilon \in L(\mathbf{G}^f)$,
   b) $(\forall s \in \Sigma^*, \forall \sigma \in \Sigma) s\sigma \in L(\mathbf{G}^f) \Leftrightarrow s \in L(\mathbf{G}^f)$ & $s\sigma \in L(\mathbf{G})$ & $\sigma \in f(s)$,
   c) no other strings belong to $L(\mathbf{G}^f)$;
2) $S(\mathbf{G}^f)$, the $\omega-$language synthesized by f, given by

$$S(\mathbf{G}^f) := \lim[L(\mathbf{G}^f)] \cap S(\mathbf{G}).$$

An $\omega-$language $K \subseteq \Sigma^\omega$ is $*-$*controllable* with respect to $\mathbf{G}$ if

$$\text{pre}(K)\Sigma_u \cap \text{pre}(L(\mathbf{G})) \subseteq \text{pre}(K).$$

For an $\omega-$language $T \subseteq \Sigma^\omega$, its *controllability prefix* is defined as

$$\begin{aligned}\text{pre}_{\mathbf{G}}(T) := \{t \in \text{pre}(T) | (\exists T' \subseteq T/t)[T' \neq \emptyset \text{ is} \\ * -\text{controllable with respect to } L(\mathbf{G})/t \text{ and} \\ \omega - \text{closed with respect to } S(\mathbf{G})/t]\},\end{aligned}$$

where $T/t := \{s \in \Sigma^\infty | ts \in T\}$, and $L(\mathbf{G})/t$ and $S(\mathbf{G})/t$ are defined in the same fashion. T is $\omega-$*controllable* with respect to $\mathbf{G}$ if

1) T is $*-$controllable with respect to $\mathbf{G}$;
2) $\text{pre}(T) = \text{pre}_{\mathbf{G}}(T)$.

$\omega-$controllability is preserved under arbitrary unions but not intersections, and $\omega-$closure is preserved under arbitrary intersections but not unions. Thus, two language classes are defined separately as follows.

$$\begin{aligned}\mathcal{C}^\omega(E) := \{T \subseteq S(\mathbf{G}) | T \subseteq E \subseteq S(\mathbf{G}) \text{ and} \\ T \text{ is } \omega - \text{controllable with respect to } \mathbf{G}\}, \\ \mathcal{F}^\omega(A) := \{T \subseteq S(\mathbf{G}) | A \subseteq T \subseteq S(\mathbf{G}) \text{ and} \\ T \text{ is } \omega - \text{closed with respect to } S(\mathbf{G})\},\end{aligned}$$

where $E, A \subseteq \Sigma^\omega$ are $\omega-$languages representing respectively the *maximal legal* and *minimal acceptable* specifications imposed on $\mathbf{G}$. Due to the closure property of $\omega-$controllability and $\omega-$closure described above, there exist the unique supremal $\omega-$controllable sublanguage $\sup\mathcal{C}^\omega(E)$, given by

$$\sup\mathcal{C}^\omega(E) := \lim(\text{pre}_{\mathbf{G}}(E)) \cap E$$

and the unique infimal $\omega-$closed superlanguage $\inf\mathcal{F}^\omega(A)$, given by

$$\inf\mathcal{F}^\omega(A) := \text{clo}(A) \cap S(\mathbf{G}).$$

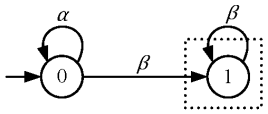

FIGURE 1. Büchi automaton of $\omega-$language S.

It is proved [[7], Theorem 5.3] that there exists an $\omega-$controllable and $\omega-$closed language T such that $A \subseteq T \subseteq E$ if and only if

$$\inf\mathcal{F}^\omega(A) \subseteq \sup\mathcal{C}^\omega(E).$$

A supervisor $f^\omega : L(\mathbf{G}) \to \Gamma$ synthesizing such T, i.e.

$$A \subset T = S(\mathbf{G}^{f^\omega}) \subseteq E$$

can be implemented by a $*-$automaton $\mathbf{SUP} := (X, \Sigma, \xi, x_0)$. $\mathbf{SUP}$ is an *implementation* of the supervisor f^ω with

$$\begin{aligned}L(\mathbf{G}) \cap L(\mathbf{SUP}) = L(\mathbf{G}^{f^\omega}), \\ S(\mathbf{G}) \cap \lim(L(\mathbf{SUP})) = S(\mathbf{G}^{f^\omega}).\end{aligned}$$

III. MAXIMAL SUBAUTOMATA REPRESENTING $\omega-$CLOSED SUBLANGUAGES

A given regular $\omega-$language is not always $\omega-$closed. For example, if $S = \alpha^*\beta^\omega$, then $\text{clo}(S) = \alpha^\omega \cup \alpha^*\beta^\omega$. Representing S by a Büchi automaton shown in Fig. 1, a direct explanation for S to be not closed is due to the existence of the selfloop at state 0 to represent α^*. Thus, to obtain a closed sublanguage of S, we need to delete the selfloop at state 0. The result is $S' = \beta^\omega$, which is obviously closed. By this inspection, we propose an algorithm in this section to compute the $\omega-$closed sublanguage of a given language, by deleting such bad loops (e.g. selfloop α^* in Fig. 1).

A. ALGORITHM FOR DETECTING LOOPS

We treat the structure of a Büchi automaton as a directed graph. A *direct graph* is an ordered pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consisting of:

1) $\mathcal{V}$ a set of vertices;
2) $\mathcal{E} \subseteq \{(q_1, q_2) \in \mathcal{V}^2\}$ a set of edges.

Let a Büchi automaton be denoted as

$$\mathbf{G} := (Q, \Sigma, \delta, q_0, \mathcal{B}_Q).$$

Then, we have $\mathcal{V} = Q, \mathcal{E} = \{(q_1, q_2) | q_2 = \delta(q_1, \sigma)!\}$ with $q_1, q_2 \in Q, \sigma \in \Sigma$. Namely, states of a Büchi automaton are regarded as vertices of a graph, and transitions of a Büchi automaton as edges of a graph. Our goal is to compute its $\omega-$closed sublanguage of the regular $\omega-$language.

First, we need to detect all loops in a graph. Let $(q_1, \cdots, q_n)^\circlearrowleft$ denote a *loop* in a graph with $q_1, \cdots, q_n \in Q, \delta(q_i, \sigma_i) = q_{i+1}, i \in \{1, \cdots, n\}, \sigma_i \in \Sigma, q_{n+1} = q_1$. We will employ the depth-first-search (DFS) mechanism [24] of a graph to realize this. The algorithm is realized by recursion and is described as Algorithm 1.

For example, for the Büchi automaton shown in Fig. 2, assume that the Büchi acceptance criterion is $\mathcal{B}_Q = \{3\}$.

Algorithm 1 DetectLoops

Input: x is the vertex number; num is the number of vertices; $v[num]$ is an array to memorize whether a vertex has been visited; $stack[num]$ is an array to implement the function of a stack and top is the subscript of the top element; $in[num]$ is an array to indicate whether a vertex is in the stack.

Output: The vertices of all loops

1: **procedure** DetectLoops(x, $v[num]$, $stack[num]$, top, $in[num]$)
2: $v[x] = true$; //Indicate that vertex x has been visited.
3: $stack[++top] = x$; //Push x to the stack.
4: $in[x] = true$; //Indicate that vertex x is in the stack.
5: **for** $i = 0$ to the number of vertices -1 **do**
6: **if** There is an edge from x to i; **then**
7: **if** $in[i] == false$; //Vertex i is not in the stack. **then**
8: DetectLoops(i, v, $stack$, top, in);
9: **else** Find the beginning vertex of a loop; //Find element i from the stack.
10: **end if**
11: Output loop $(i, \cdots, top)^{\circlearrowleft}$; //All vertices from element i to the top element of the stack.
12: **end if**
13: **end for**
14: $top--$; //Pop an element out from the stack because element x may be a vertex in another loop.
15: $in[x] = false$;
16: **end procedure**

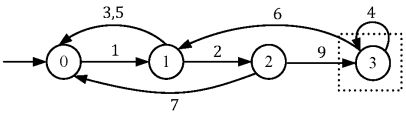

FIGURE 2. Transition graph of a Büchi automaton.

By applying Algorithm DetectLoops to the graph, vertices 0 and 1 will be pushed to the stack in sequence. At vertex 1, as there exists an edge $(1, 0)$ corresponding to transitions $1 \xrightarrow{3,5} 0$ (Line 6), and $in[0] = true$ (Line 7), the beginning vertex 0 of a loop is found from the stack (Line 9). Then loop $(0, 1)^{\circlearrowleft}$ is output (Line 11). Other loops will be found in the same way. After the termination of this algorithm, four loops will be output, i.e. loops $(0, 1)^{\circlearrowleft}$, $(0, 1, 2)^{\circlearrowleft}$, $(1, 2, 3)^{\circlearrowleft}$, $(3)^{\circlearrowleft}$.

As the time complexity of the DFS algorithm is $\mathcal{O}(n_V + n_E)$ [24], where n_V, n_E are the numbers of vertices and edges of a graph respectively, the time complexity of Algorithm DetectLoops is $\mathcal{O}(n_V + n_E)$.

B. ALGORITHM FOR COMPUTING MAXIMAL SUBAUTOMATA

We say that a loop $(q_1, \cdots, q_n)^{\circlearrowleft}$ is a *good loop* if

$$\{q_1, \cdots, q_n\} \cap \mathcal{B}_Q \neq \emptyset.$$

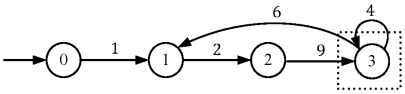

FIGURE 3. The obtained Büchi automaton generated from the Büchi automaton shown in Fig. 2.

Namely, the intersection of the vertex set of this loop and the Büchi acceptance criterion is not empty. Otherwise, we say that this loop is a *bad loop*. Obviously, the good loop corresponds to the infinite part of an $\omega-$language. For the graph shown in Fig. 2, loops $(1, 2, 3)^{\circlearrowleft}$ and $(3)^{\circlearrowleft}$ are good loops, but loops $(0, 1)^{\circlearrowleft}$ and $(0, 1, 2)^{\circlearrowleft}$ are bad loops.

Owing to the existence of bad loops, the $\omega-$language generated by the Büchi automaton is not $\omega-$closed. To obtain a closed sublanguage of a regular $\omega-$language, we need to prevent bad loops by deleting its back edges, where a *back edge* is the edge connecting the top vertex with the beginning vertex during the DFS process. For the loops $(0, 1)^{\circlearrowleft}$, $(0, 1, 2)^{\circlearrowleft}$, $(1, 2, 3)^{\circlearrowleft}$, $(3)^{\circlearrowleft}$ of the graph shown in Fig. 2, the back edges are $1 \xrightarrow{3,5} 0$, $2 \xrightarrow{7} 0$, $3 \xrightarrow{6} 1$, $3 \xrightarrow{4} 3$ respectively. Therefore, if we delete back edges $1 \xrightarrow{3,5} 0$ and $2 \xrightarrow{7} 0$ of bad loops $(0, 1)^{\circlearrowleft}$ and $(0, 1, 2)^{\circlearrowleft}$ respectively, and trim[1] the result, we have that the obtained Büchi automaton shown in Fig. 3 generates an $\omega-$closed sublanguage, with the Büchi acceptance criterion $\mathcal{B}_Q = \{3\}$.

However, required by the $\omega-$controllability, uncontrollable events cannot be prohibited directly and thus we require that only controllable edges be deleted at some reached states (of the plant). Thus we need to find the controllable edges of a bad loop nearest to the beginning vertex in a counterclockwise fashion and call them as the *nearest controllable back edges*. For example, in Fig. 2, for bad loop $(0, 1, 2)^{\circlearrowleft}$, the nearest controllable back edge is $2 \xrightarrow{7} 0$. If there doesn't exist any nearest controllable back edges in a bad loop, then this bad loop cannot be prevented and the result is empty.

Based on the descriptions above, we summarize the algorithm to obtain the subautomaton representing the $\omega-$closed sublanguage of a given $\omega-$language in Algorithm 2. By our aforementioned analysis, the time complexity of Algorithm DetectLoops is $\mathcal{O}(n_V + n_E)$. The worst case for step 3 in Algorithm Maxclo is to search all edges of a loop, which has the time complexity $\mathcal{O}(n_E)$. The worst case for step 5 in Algorithm Maxclo is to delete all edges and vertices, which has the time complexity $\mathcal{O}(n_V + n_E)$. Thus, the time complexity of Algorithm Maxclo is $\mathcal{O}(n_V + n_E)$.

In fact, the resultant Büchi automaton $\mathbf{B}'$ is a subautomaton of $\mathbf{B}$.

Proposition 1: The resultant Büchi automaton $\mathbf{B}'$ is a maximal subautomaton of $\mathbf{B}$, i.e. there doesn't exist a Büchi subautomaton of $\mathbf{B}$ having more transitions than $\mathbf{B}'$ and representing a $*-$controllable and $\omega-$closed sublanguage.

Proof: The proof is by contradiction. Assume that there exists Büchi automaton $\mathbf{B}''$, a subautomaton of $\mathbf{B}$, having

[1] An automaton is *trim* if it is both reachable and coreachable [1].

Algorithm 2 Maxclo

Input: A Büchi automaton $\mathbf{B}$

Output: A resultant Büchi automaton $\mathbf{B}'$

1: **procedure** Maxclo($\mathbf{B}$)
2: Call Algorithm DetectLoops to obtain all loops in $\mathbf{B}$.
3: For each loop detected, if it is a bad loop, say $(q_1, q_2, \cdots, q_n)^{\circlearrowleft}$, then find its nearest controllable back edges, i.e. $q_i \xrightarrow{\sigma_i} q_{i+1}, \sigma_i \in \Sigma_c$, such that for $j = i+1, \cdots, n$, we have $q_j \xrightarrow{\sigma_j} q_{j+1}, \sigma_j \in \Sigma_u$ with $q_{n+1} = q_1$.
4: **if** Such nearest controllable back edges exist **then**
5: Delete transitions $q_i \xrightarrow{\sigma_i} q_{i+1}$ and remove the uncoreachable transitions iteratively.
6: **else**There are no nearest controllable back edges available, then return an empty Büchi automaton directly.
7: **end if**
8: Return a trim resultant Büchi automaton $\mathbf{B}'$, which is controllable and does not contain bad loops.
9: **end procedure**

more states than $\mathbf{B}'$ and representing a $*-$controllable and $\omega-$closed sublanguage. As Algorithm Maxclo deletes all bad loops, the additional states in $\mathbf{B}''$ belong to good loops not incorporated in $\mathbf{B}'$. Hence, there exist strings corresponding to additional good loops, but these strings cannot be generated by $\mathbf{B}'$. However, according to the semantics of Algorithm Maxclo, only bad loops are deleted. Thus, these additional good loops are non-existent. Namely, the resultant Büchi automaton $\mathbf{B}'$ is a maximal subautomaton of $\mathbf{B}$. □

The reason we discuss from the perspective of the automaton transition graph is that there does not exist a supremal or maximal $\omega-$closed sublanguage, but there exists a maximal subautomaton representing an $\omega-$closed sublanguage. Moreover, it is convenient of modeling and computation from the perspective of automata. Thus, we use Algorithm Maxclo to obtain a maximal Büchi subautomaton representing a $*-$controllable and $\omega-$closed sublanguage.

Furthermore, the automaton returned by Algorithm Maxclo has the following properties, and these properties are important for the supervisor synthesis in Section IV.

Proposition 2: Assume that the DES to be controlled is modeled by Büchi automaton $\mathbf{G}$. *Let* B *and* B' *denote the* $\omega-$*closed sublanguages represented by Büchi automata* $\mathbf{B}$ *and* $\mathbf{B}'$ *defined in Algorithm Maxclo respectively. If* $B \subseteq S(\mathbf{G})$ *is* $\omega-$*controllable with respect to* $\mathbf{G}$, *then* B' *is* $\omega-$*controllable with respect to* $\mathbf{G}$ *and* $\omega-$*closed with respect to* $S(\mathbf{G})$.

Proof: To show that B' is $\omega-$controllable with respect to $\mathbf{G}$, we need to show that

1) B' is controllable with respect to $\mathbf{G}$;
2) $\mathrm{pre}(B') = \mathrm{pre}_{\mathbf{G}}(B')$.

For 1), we need to show that

$$\mathrm{pre}(B')\Sigma_u \cap \mathrm{pre}(L(\mathbf{G})) \subseteq \mathrm{pre}(B').$$

Let $s \in \mathrm{pre}(B'), \sigma \in \Sigma_u, s\sigma \in \mathrm{pre}(L(\mathbf{G}))$. String s should be part of a good loop; otherwise, $s \notin \mathrm{pre}(B')$. We have $s\sigma \in \mathrm{pre}(B)$ as B is $\omega-$controllable with respect to $\mathbf{G}$. Furthermore, $s\sigma$ is not part of a bad loop as Algorithm Maxclo deletes all bad loops of $\mathbf{B}$. Thus, $s\sigma \in \mathrm{pre}(B')$ as $\mathbf{B}'$ retains all good loops of $\mathbf{B}$.

For 2), we first show that B' is $\omega-$closed with respect to $S(\mathbf{G})$. By the semantics of Algorithm Maxclo, as all bad loops are deleted, we have that B' is $\omega-$closed, i.e. $B' = \mathrm{clo}(B')$. Thus, $\mathrm{clo}(B') \cap S(\mathbf{G}) = B' \cap S(\mathbf{G}) = B'$ as $B' \subseteq B \subseteq S(\mathbf{G})$.

Let

$$\begin{aligned}\mathrm{pre}_{\mathbf{G}}(B') := \{t \in \mathrm{pre}(B') | (\exists T' \subseteq B'/t)[T' \neq \emptyset \text{ is} \\ * -\text{controllable with respect to } L(\mathbf{G})/t \text{ and} \\ \omega - \text{closed with respect to } S(\mathbf{G})/t]\}.\end{aligned}$$

As B' is $*-$controllable with respect to $\mathbf{G}$ and $\omega-$closed with respect to $S(\mathbf{G})$, we have that $\mathrm{pre}(B') = \mathrm{pre}_{\mathbf{G}}(B')$ by the definition of $\mathrm{pre}(B')$. □

Remark 1: Proposition 2 tells us that if B *is* $\omega-$*controllable with respect to* $\mathbf{G}$, *the resultant* B' *is still* $\omega-$*controllable with respect to* $\mathbf{G}$. *Thus, Algorithm Maxclo does not change the* $\omega-$*controllability of input* B.

Remark 2: Algorithm Maxclo could also be applied to a Rabin automaton, because the main function of Algorithm Maxclo is to delete all bad loops by preventing the occurrence of the nearest controllable back edges of bad loops. Correspondingly, Algorithm Maxclo returns a Rabin automaton representing an $\omega-$*closed sublanguage.*

Lemma 1: Let B *and* B' *denote the* $\omega-$*closed sublanguages represented by Rabin automata* $\mathbf{B}$ *and* $\mathbf{B}'$ *used in Algorithm Maxclo respectively. If* $B \subseteq S(\mathbf{G})$ *is* $\omega-$*controllable with respect to* $\mathbf{G}$, *then* B' *is* $\omega-$*controllable with respect to* $\mathbf{G}$ *and* $\omega-$*closed with respect to* $S(\mathbf{G})$.

Proof: As B and B' denote the $\omega-$closed sublanguages represented by Rabin automata $\mathbf{B}$ and $\mathbf{B}'$ respectively, we don't need to consider the Rabin acceptance condition after we have obtained B and B'. With this foundation, we can show that B' is $\omega-$controllable with respect to $\mathbf{G}$ and $\omega-$closed with respect to $S(\mathbf{G})$ in the same fashion of Proposition 2. □

IV. SUPERVISORY CONTROL OF $\omega-$CLOSED LANGUAGES

Construct a deterministic Rabin-Büchi automaton (RBA)

$$\mathbf{RBA} = (Q, \Sigma, \delta, q_0, \{(R_p, I_p) : p \in P\}, \mathcal{B}_Q).$$

In $\mathbf{RBA}$, the $*-$automaton (Q, Σ, δ, q_0) accepts the $*-$behavior of $\mathbf{G}$, the Büchi automaton $\mathbf{RBA}_B := (Q, \Sigma, \delta, q_0, \mathcal{B}_Q)$ accepts the $\omega-$behavior $S(\mathbf{G}) \subseteq \Sigma^\omega$ of $\mathbf{G}$, and the Rabin automaton $\mathbf{RBA}_R := (Q, \Sigma, \delta, q_0, \{(R_p, I_p) : p \in P\})$ accepts $E \subseteq \Sigma^\omega$. In this section, we will propose a new approach to obtain an $\omega-$controllable and $\omega-$closed language T such that $A \subseteq T \subseteq E$.

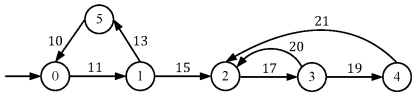

FIGURE 4. Transition graph of RBA RBA.

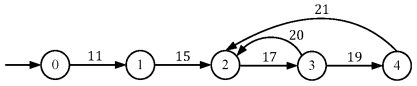

FIGURE 5. Transition graph of Rabin automaton $\mathbf{RBA}'_R$.

A. THE MINIMAL ACCEPTABLE SPECIFICATION $A = \emptyset$

If $A = \emptyset$, the minimal acceptable specification is satisfied automatically. We could apply Algorithm Maxclo to $\mathbf{RBA}_R$ and denote the result as $\mathbf{RBA}'_R$. Let R and R' denote the $\omega-$languages represented by $\mathbf{RBA}_R$ and $\mathbf{RBA}'_R$ respectively. If R is $\omega-$controllable with respect to $\mathbf{G}$, by Lemma 1 we have that R' is $\omega-$controllable with respect to $\mathbf{G}$ and $\omega-$closed with respect to $S(\mathbf{G})$. Thus, R' can act as a supervisor by Theorem 5.3 in [7]. If R is not $\omega-$controllable with respect to $\mathbf{G}$, we should apply the Thistle's controllability computation approach to R' to obtain an $\omega-$controllable and $\omega-$closed language T such that $A \subseteq T \subseteq E$.

Take the RBA $\mathbf{RBA}$ shown in Fig. 4 for example.

The Büchi acceptance criterion is $\mathcal{B}_Q = \{0, 1, 2, 3, 4, 5\}$, and the recurrence family of Rabin automaton $\mathbf{RBA}_R$ is $(R_p = \{3\}, I_p = \{2, 3, 4\})$. We could verify that $\mathbf{RBA}_R$ is $\omega-$controllable, but not $\omega-$closed, owing to the existence of bad loop $(0, 1, 5)^{\circlearrowleft}$.

By applying Algorithm Maxclo to $\mathbf{RBA}_R$, we have that loop $(0, 1, 5)^{\circlearrowleft}$ is a bad loop. Thus, the nearest controllable back edge $1 \xrightarrow{13} 5$ of this loop is deleted, and the resultant Rabin automaton $\mathbf{RBA}'_R$ is shown in Fig. 5. As the $\omega-$language represented by $\mathbf{RBA}'_R$ is $\omega-$controllable and $\omega-$closed, it is an implementation of a supervisor.

If we use the Thistle's supervisor synthesis approach to compute the supervisor, it will return a state feedback map $\phi^{\mathcal{A}} : C^{\mathcal{A}} \to C$ shown in Table 1, where

$$C^{\mathcal{A}} = \bigcup_{i,j,l=1}^{|X|} \bigcup_{k=0,1} C^{\mathcal{A}}_{i,j,k,l}$$

is the controllability subset. For each of the subsets $C^{\mathcal{A}}_{i,j,k,l}$, a total feedback map $\phi^{\mathcal{A}}_{i,j,k,l} : C^{\mathcal{A}}_{i,j,k,l} \to C$ is defined as detailed in [6]. Finally, the total map

$$\begin{aligned} \phi^{\mathcal{A}} : C^{\mathcal{A}} &\to C \\ x &\mapsto \phi^{\mathcal{A}}_{i,j,k,l}(x) \end{aligned}$$

is defined, where (i, j, k, l) is the least $4-$tuple in the lexicographic ordering such that $x \in C^{\mathcal{A}}_{i,j,k,l}$.

Readers are referred to [6] for the details to compute $C^{\mathcal{A}}$, with the illustration shown in Table 2 in the appendix. As at state 3, event 19 is disabled by $\phi^{\mathcal{A}}$, state 4 and transitions $3 \xrightarrow{19} 4, 4 \xrightarrow{21} 2$ will be removed, thereby returning the result shown in Fig. 6.

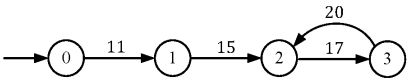

FIGURE 6. Transition graph of the Rabin automaton returned by the Thistle's supervisor synthesis approach.

TABLE 1. Map $\phi^{\mathcal{A}}$ computed by the Thistle's controllability computation approach.

$C^{\mathcal{A}}$	$\phi_{\mathcal{A}}$
0	{11}
1	{13}
2	{17}
3	{20}
4	{21}
5	{13}

Obviously, the result of the Thistle's supervisor synthesis approach is more restrictive as at state 3, event 19 is disabled. While at state 3 in $\mathbf{RBA}'_R$, event 19 is enabled. The reason for this phenomenon is to be explained in the next section.

B. THE MINIMAL ACCEPTABLE SPECIFICATION $A \neq \emptyset$

Let Büchi automaton $\mathbf{A}$ represent the minimal acceptable language A. If the minimal acceptable specification $A \neq \emptyset$, we need to incorporate $\mathbf{A}$ into $\mathbf{RBA}$. For this, we first need to modify the transition function of $\mathbf{A} := (Q_A, \Sigma_A, \delta_A, q_{0A}, \mathcal{B}_{QA})$ to a total function. Namely, for any $q \in Q_A$, if event $\sigma \in \Sigma_A$ is not defined at state q, then add transition $\delta_A(q, \sigma) = q_d$, where q_d is a newly added dump state, and all events have selfloops at this state. Denote the resultant Büchi automaton as $\mathbf{A}'$. Clearly, the transition function of $\mathbf{A}'$ is a total function. For brevity, we assume that $\mathcal{B}_{QA} = Q_A$ in this paper.

Next we compute an $\omega-$controllable and $\omega-$closed sublanguage by the following steps.

Step 1: Compute the synchronous product of $\mathbf{RBA}$ and $\mathbf{A}'$, and denote the result as $\mathbf{NRBA}$.

Step 2: Apply the Thistle's controllability computation approach to $\mathbf{NRBA}$ and denote the result as $\mathbf{SUPW}$.

Step 3: Compute Maxclo($\mathbf{SUPW}$) to obtain a RBA representing its $\omega-$closed sublanguage. Denote the resultant RBA as $\mathbf{T}$.

Lemma 2: Let $\mathbf{NRBA}$ *denote the synchronous product of* $\mathbf{RBA}$ *and* $\mathbf{A}'$. *We have* $S(\mathbf{NRBA}) = S(\mathbf{RBA})$.

Proof: Let

$$\begin{aligned} \mathbf{RBA} &= (Q, \Sigma, \delta, q_0, \{(R_p, I_p) : p \in P\}, \mathcal{B}_Q), \\ \mathbf{NRBA} &= (Q_N, \Sigma_N, \delta_N, q_{0N}, \{(R_{pN}, I_{pN}) : p \in P\}, \mathcal{B}_{QN}), \\ \mathbf{A} &= (Q_A, \Sigma_A, \delta_A, q_{0A}, \mathcal{B}_{QA}), \\ \mathbf{A}' &= (Q'_A, \Sigma'_A, \delta'_A, q'_{0A}, \mathcal{B}'_{QA}). \end{aligned}$$

The transition function δ_N of $\mathbf{NRBA}$ is defined as

$$\begin{aligned} \delta_N(q_N, \sigma) &:= \delta_N((q, q'_A), \sigma) \\ &= (\delta(q, \sigma), \delta'_A(q'_A, \sigma)), \end{aligned}$$

where $q_N \in Q_N, q \in Q, q'_A \in Q'_A$. As the transition function of $\mathbf{A}'$ is a total function, we have that whenever $\delta(q,\sigma)$ is defined, $\delta_A(q'_A,\sigma)$ is defined.

In addition, $\mathcal{B}_{QN} = \{q_N := (q,q'_A) | q \in \mathcal{B}_Q\}$; for a given $p \in P$, $R_{pN} = \{q_N := (q,q'_A) | q \in R_p\}, I_{pN} = \{q_N := (q,q'_A) | q \in I_p\}$.

($\supseteq$) This direction is automatic by the definition of **NRBA**.

($\subseteq$) The proof is by induction. Clearly, the basis step holds when the string is ϵ.

(Inductive step:) Let $s \in S(\mathbf{NRBA}) \cap S(\mathbf{RBA}), s\sigma \in S(\mathbf{NRBA})$. We need to show that $s\sigma \in S(\mathbf{RBA})$.

Assume that $\delta_N(q_{0N},s) = q_N := (q,q_A), \delta_N(q_N,\sigma) = q'_N$. By the inductive assumption, we have that $\delta(q_0,s) = q$. By the definition of the synchronous product of **RBA** and $\mathbf{A}'$, we have $\delta(q,\sigma)$ is defined. Thus, $s\sigma \in S(\mathbf{RBA})$. □

Lemma 2 tells us that **NRBA** incorporates the behavior of $\mathbf{A}'$ without changing the behavior of **RBA**. This is a premise of this subsection.

Theorem 1: Denote the language represented by RBA $\mathbf{T}$ *as* T. *By following the aforementioned approach to obtain an* $\omega-$*controllable and* $\omega-$*closed language* T, *there exists a supervisor* $f^\omega : L(\mathbf{NRBA}) \to \Gamma$ *synthesizing* T, *i.e.*

$$A \subset T = S(\mathbf{NRBA}^{f^\omega}) \subseteq E.$$

Proof: By the definition of T, we have that $T = \sup\mathcal{C}^\omega(E)$. Moreover, T is $\omega-$closed as there is no bad loop in $\mathbf{T}$. Thus, $\inf\mathcal{F}^\omega(A') \subseteq T$. Hence, $\sup\mathcal{C}^\omega(E) \neq \emptyset$ and $\inf\mathcal{F}^\omega(A') \subseteq \sup\mathcal{C}^\omega(E)$. By Theorem 5.3 in [7], we have that the supervisory control problem for $\omega-$languages is solvable. Hence, there exists a supervisor $f^\omega : L(\mathbf{NRBA}) \to \Gamma$ synthesizing T. □

Theorem 1 tells us that $\mathbf{T}$ is an implementation of the supervisor f^ω.

Theorem 2: [*[6], Theorem 8.15*] *Let*

$$\mathcal{A} = (\Sigma, X, \delta, x_0, \{(R_p, I_p) : p \in P\}, \mathcal{B}_Q)$$

be a deterministic Rabin-Büchi automaton. There exists a map $\phi^{\mathcal{A}} : C^{\mathcal{A}} \to C$ *such that, for any* $x \in C^{\mathcal{A}}$ *for which* $E_x \subseteq S_x$, *the map defined by*

$$f_x : \Sigma^* \to C$$
$$l \mapsto \phi^{\mathcal{A}}(\delta(l,x))$$

is a complete, deadlock-free supervisor for (L_x, S_x) *such that* $(S_x)^{f_x} \subseteq E_x$.

Theorem 3: The state feedback f_x *defined in Theorem 2 is more restrictive than* f^ω *defined in Theorem 1.*

Proof: Let $T' := (S_x)^{f_x}$. To show that f_x defined in Theorem 2 is more restrictive than f^ω, we need to show $T' \subseteq T$.

Clearly, $T' \subseteq T$ if $s = \epsilon$.

Let $s \in T \cap T'$, and $s\sigma \in T'$; for each $\sigma \in \Sigma$, we need to show $s\sigma \in T$. Since $\Sigma = \Sigma_c \dot\cup \Sigma_{uc}$, we consider the following two cases.

(i) $\sigma \in \Sigma_{uc}$. In this case, by Proposition 2 we know that T is ω-controllable; thus $s\sigma \in T$ by the definition of $\omega-$controllability.

(ii) $\sigma \in \Sigma_c$. In this case, we prove the contraposition, i.e. if $s\sigma \notin T$, then $s\sigma \notin T'$.

Let $x = \delta(x_0,s)$. By the semantics of Algorithm Maxclo, since $s \in T$ and σ is removed by it, (1) state x must be coreachable to a state in R_p by a string $t \in \Sigma \setminus \{\sigma\}\Sigma^*$, and (2) σ and its downstream string will lead x to a bad loop. (1) means that there must exist event $\sigma' \in \Sigma$ and string $t \in \Sigma^*$ such that $\sigma' \neq \sigma, \delta(x,\sigma' t) \in R_p$. According to the definition of supervisor $\phi^{\mathcal{A}}_{i,j,k,l} : C^{\mathcal{A}}_{i,j,k,l} \to C$ (as described in the proof of Theorem 8.15 in [6]), when x is firstly added to $C^{\mathcal{A}}_{i,j,k,l}$ for some (i,j,k,l), σ' should be chosen by $\phi^{\mathcal{A}}_{i,j,k,l}(x)$ (due to (1)), but σ will not (due to (2)). Further, $\phi^{\mathcal{A}}(x) = \phi^{\mathcal{A}}_{i,j,k,l}(x)$ because i,j,k,l is the least 4-tuple in the lexicographic ordering. Hence, we conclude that $\sigma \notin \phi^{\mathcal{A}}(x)$, and thus $T' \subseteq T$. □

Remark 3: We should clarify that the control patterns in both of the two approaches (our approach and Thistle's supervisor synthesis approach in [7]) are closed under union. Considering the case that there are multiple events defined at a state, but leading to different good cycles (legal infinite behavior), there may exist multiple control patterns $\phi^{\mathcal{A}}_{i,j,k,l} : C^{\mathcal{A}}_{i,j,k,l} \to C$ *(as described in the proof of Theorem 8.15 in [6]). According to the supervisor construction rule in the Thistle's supervisor synthesis approach, only one control pattern is selected (see the definition of* $\phi^{\mathcal{A}} : C^{\mathcal{A}} \to C$ *in [6]). By contrast, in our approach, all events (in all the control patterns) will be retained, because the control patterns are closed under union.*

Remark 4: Regarding to the $\omega-$*controllability, we should clarify that none of the two approaches can derive the unique maximal* $\omega-$*controllable sublanguage if this language is not* $\omega-$*closed. The reason is as follows.*

It is proved in [Proposition 4.5, [7]] that both of $\omega-$*controllability and* $\omega-$*closure are necessary for the existence of a complete and deadlock-free supervisor. It is also true that every specification language contains a unique maximal* $\omega-$*controllable sublanguage because* $\omega-$*controllability is closed under set union. However, if the maximal* $\omega-$*controllable sublanguage is not* $\omega-$*closed, the supervisor does not exist, and thus none of the two approaches can construct a supervisor synthesizing it. In this case, our approach will first compute a maximal subautomaton which represents an* $\omega-$*closed sublanguage, and then construct a complete and deadlock-free supervisor. Moreover, in the case that the maximal* $\omega-$*controllable sublanguage is* $\omega-$*closed, our approach will directly construct a complete and deadlock-free supervisor synthesizing the maximal* $\omega-$*controllable sublanguage.*

The supervisor to synthesize state feedback f_x is implemented by the Thistle's supervisor synthesis approach. An intuitive explanation for the restrictiveness of the result obtained by this procedure is that, if there exist several events defined

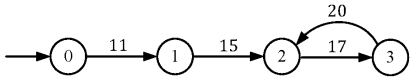

FIGURE 7. Transition graph of Büchi automaton A.

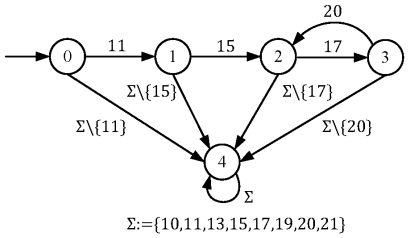

FIGURE 8. Transition graph of A′.

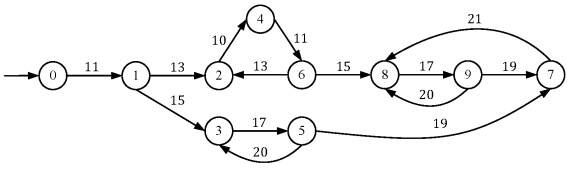

FIGURE 9. Transition graph of NRBA.

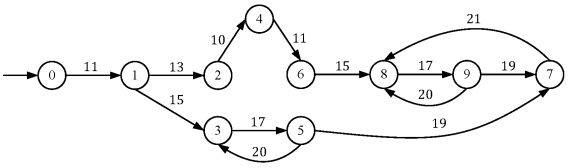

FIGURE 10. Transition graph of T.

at a given state to form several good loops, the synthesized supervisor may only enable one event at this state, as in the proof of Theorem 8.15 in [6], (i, j, k, l) of $\phi^{\mathcal{A}}_{i,j,k,l} : C^{\mathcal{A}}_{i,j,k,l} \to C$ defining $\phi^{\mathcal{A}} : C^{\mathcal{A}} \to C$ is the least 4−tuple in the lexicographic ordering such that $x \in C^{\mathcal{A}}_{i,j,k,l}$. However, RBA **T** retains all good loops. Therefore, RBA **T** is often more permissive than the result obtained by applying the Thistle's supervisor synthesis approach directly to **NRBA**.

We continue the example of Subsection IV-A. Let the minimal acceptable language A be represented by the Büchi automaton shown in Fig. 7, whose Büchi acceptance criterion is $\mathcal{B}_Q = \{0, 1, 2, 3\}$. We modify the transition function of **A** to a total function and denote the result as **A**′, which is shown in Fig. 8.

The product of **RBA** and **A**′ is denoted as **NRBA** shown in Fig. 9, in which $R_p = \{5, 9\}$, $I_p = \{3, 5, 7, 8, 9\}$. Apply the Thistle's controllability computation approach to **NRBA** and denote the result as **SUPW**. By applying Algorithm Maxclo to **SUPW**, bad loop $(2, 4, 6)^{\circlearrowleft}$ is prevented by deleting the nearest controllable back edge $6 \xrightarrow{13} 2$, and denote the result as **T** shown in Fig. 10. As **T** is ω−controllable and ω−closed, by Theorem 1 we have that **T** is an implementation of the supervisor f^{ω}.

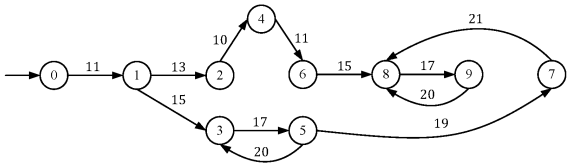

FIGURE 11. Transition graph of the RBA computed by the Thistle's supervisor synthesis approach.

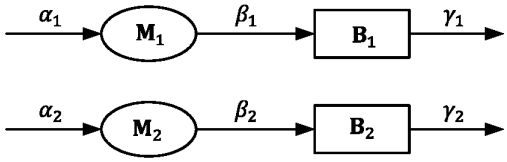

FIGURE 12. Layout of Small Factory.

If we use the Thistle's supervisor synthesis approach [6], [7] to compute the supervisor, we obtain the RBA shown in Fig. 11. The result of our approach is more permissive as event 19 at state 9 is enabled in our approach but disabled in the Thistle's supervisor synthesis approach.

V. CASE STUDY: SMALL FACTORY

A. MODEL DESCRIPTIONS: PLANT AND SPECIFICATIONS

We illustrate the above supervisor synthesis approach for DESs with infinite behavior by studying a Small Factory example, adapted from [6, Chapt. 3]. As displayed in Fig. 12, the plant to be controlled, denoted by **SF**, consists of two machines $\mathbf{M}_i$ $(i = 1, 2)$ coupled with two buffers $\mathbf{B}_i$ $(i = 1, 2)$. The alphabet of event symbols for **SF** is

$$\Sigma = \{\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2\},$$

in which events α_1 and α_2 are considered as controllable events, i.e. $\Sigma_c = \{\alpha_1, \alpha_2\}$.

The finite behavior of the plant is described as follows. There are two routines in the plant. At each routine i $(i = 1, 2)$, machine $\mathbf{M}_i$ processes one workpiece each time. When $\mathbf{M}_i$ begins a job, it acquires a workpiece from elsewhere in the factory (event α_i). Upon completion, $\mathbf{M}_i$ pushes the workpiece into buffer $\mathbf{B}_i$ (event β_i). Machines not shown in Fig. 12 remove workpieces from buffer $\mathbf{B}_i$ for further processing (event γ_i); we assume that certain control mechanism prevents such events from inducing buffer $\mathbf{B}_i$ to "underflow". For the sake of simplicity, we assume that each buffer has only one slot. These two machines and two buffers are modeled by the automata shown in Fig. 13.

The infinite behavior of the plant describes that removing workpieces from the buffers are in continual operation, so that every occurrence of β_i is *eventually* followed by an occurrence of γ_i. This behavior is captured by Büchi automata $\mathbf{F}_i$ $(i = 1, 2)$ shown in Fig. 14.

Now we have a complete model of the uncontrolled DES plant **SF**: the finite behavior is the intersection of the languages accepted by the four automata shown

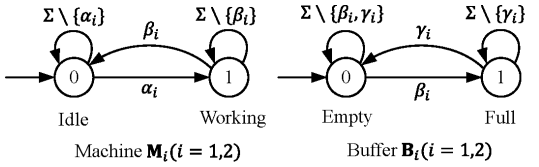

FIGURE 13. Automata representing finite behavior of machines $\mathbf{M}_i$, and buffers $\mathbf{B}_i$ $(i = 1, 2)$.

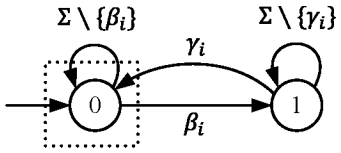

FIGURE 14. Büchi automata $\mathbf{F}_i$ $(i = 1, 2)$ representing infinite behavior of Small Factory.

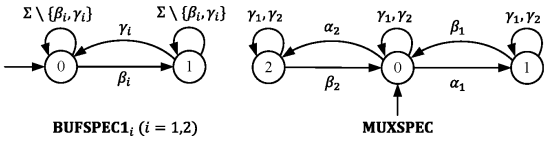

FIGURE 15. Safety specifications: prevention of buffers' overflow represented by automata $\mathbf{BUFSPEC}_i$ $(i = 1, 2)$ and mutual exclusion requirement represented by automaton **MUXSPEC**.

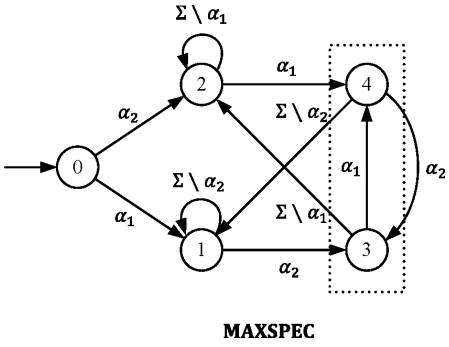

FIGURE 16. Maximal legal liveness specification represented by a Büchi automaton.

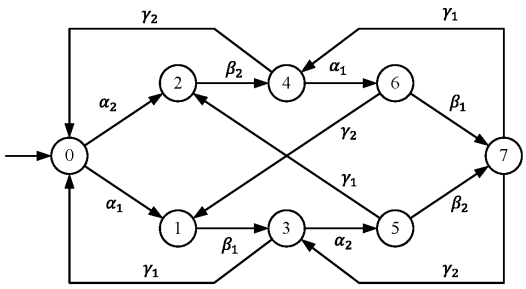

FIGURE 17. Transition graph of $\mathbf{SF}^{f^*}$ and $\mathbf{SUP}^*$.

in Fig. 13, i.e.

$$L(\mathbf{SF}) = L(\mathbf{M}_1) \cap L(\mathbf{M}_2) \cap L(\mathbf{B}_1) \cap L(\mathbf{B}_2);$$

the infinite behavior is the intersection of $\lim(L(\mathbf{SF}))$ with the $\omega-$languages accepted by the two Büchi automata shown in Fig. 14, i.e.

$$S(\mathbf{SF}) = \lim[L(\mathbf{SF})] \cap S(\mathbf{F}_1) \cap S(\mathbf{F}_2).$$

The plant under control must satisfy a number of specifications.

1) It should prevent buffer overflows. Namely, two occurrences of β_i should be separated by an occurrence of γ_i (S1).
2) Because $\mathbf{M}_i$ $(i = 1, 2)$ employ the same resources, they must not be allowed to operate simultaneously (S2). Namely, α_i should not occur between successive occurrences of α_j and β_j.
3) Because the "mutual exclusion" requirement (S2) raises the possibility that one machine may continually preempt the other, we add a liveness specification that each machine operates *infinitely often*. Namely, each α_i should occur infinitely often.

Specifications (S1) and (S2) are represented by automata $\mathbf{BUFSPEC}_i$ $(i = 1, 2)$ and **MUXSPEC** shown in Fig. 15. They describe finite behavioral requirements on the system, and thus are considered as safety specifications. Let E_s denote the overall safety specification, i.e.

$$E_s = L(\mathbf{BUFSPEC}_1) \cap L(\mathbf{BUFSPEC}_2) \cap L(\mathbf{MUXSPEC}).$$

(S3) is represented by Büchi automaton **MAXSPEC** shown in Fig. 16, and considered as the maximal legal liveness specification, i.e.

$$E_l = S(\mathbf{MAXSPEC}).$$

We assume that the minimal acceptable liveness specification is empty, i.e.

$$A = \emptyset.$$

B. SUPERVISOR SYNTHESIS

There are two types of specifications imposed on the system **SF**: safety specification E_s and liveness specifications E_l and A.

For the safety specification, by the algorithm presented in [5], [8], we compute supervisor $\mathbf{SUP}^* := (X^*, \Sigma, \xi^*, x_0^*)$ as displayed in Fig. 17, which has 8 states and 14 transitions. The controlled behavior of **SF** under the control of $\mathbf{SUP}^*$ is represented by Büchi automaton $\mathbf{SF}^{f^*}$, i.e.

$$L(\mathbf{SF}^{f^*}) = L(\mathbf{SF}) \cap L(\mathbf{SUP}^*),$$
$$S(\mathbf{SF}^{f^*}) = S(\mathbf{SF}) \cap \lim(L(\mathbf{SUP}^*)).$$

$\mathbf{SF}^{f^*}$ has the same transition graph with $\mathbf{SUP}^*$, and the Büchi acceptance criterion accepting language $S(\mathbf{SF}^{f^*})$ is $\mathcal{B}_{X^*} = \{0, 1, 2, 3, 4\}$.

It is easily verified that the safety specifications (S1) and (S2) are both satisfied, i.e.

$$L(\mathbf{SF}^{f^*}) = \sup \mathcal{C}^*(E_s \cap L(\mathbf{SF})) \subseteq E_s.$$

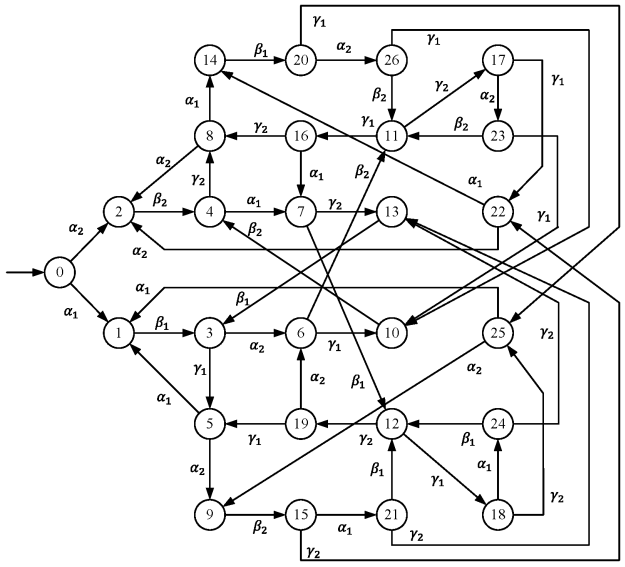

FIGURE 18. Transition graph of RBA.

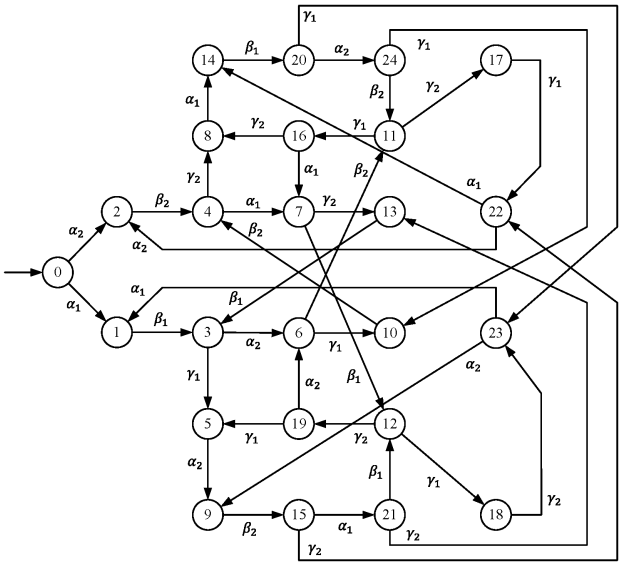

FIGURE 19. Transition graph of $\mathbf{SUP}_1^{\omega}$.

However, there may exist the case that one of machines, e.g. $\mathbf{M}_1$, may work recursively all the time. In other words, $\mathbf{M}_1$ may preempt the start of $\mathbf{M}_2$ infinitely, thereby violating the liveness specification (S3).

For maximal legal liveness specification E_l and minimal acceptable liveness specification A, we treat $\mathbf{SF}^{f^*}$ as the new plant to be controlled. For the supervisor synthesis, we first construct a Rabin-Büchi automaton $\mathbf{RBA} := (Q', \Sigma, \delta', q_0', \{(R_p', I_p') : p \in P'\})$ with 27 states and 48 transitions, as displayed in Fig. 18, where $Q' = \{0, 1, \cdots, 26\}$, the Büchi acceptance criterion is $\mathcal{B}_{Q'} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 14, 17, 18\}$, and the Rabin acceptance criterion is $\{(R_1' = \{6, 7, 9, 14\}, I_1' = Q')\}$.

By inspecting the transition graph of **RBA**, it is easily found that there are four bad loops $(1, 3, 5)^{\circlearrowleft}$, $(2, 4, 8)^{\circlearrowleft}$, $(12, 18, 24)^{\circlearrowleft}$, and $(11, 17, 23)^{\circlearrowleft}$. By applying Algorithm Maxclo to it, we remove four corresponding transitions $(5 \xrightarrow{\alpha_1} 1)$, $(8 \xrightarrow{\alpha_2} 2)$, $(18 \xrightarrow{\alpha_1} 24)$, and $(17 \xrightarrow{\alpha_2} 23)$, and thus obtain Büchi automaton $\mathbf{SUP}_1^{\omega}$ with the Büchi acceptance criterion $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 14\}$, as displayed in Fig. 19, which has 25 states and 40 transitions. It is easily verified that $S(\mathbf{SUP}_1^{\omega})$ is $\omega-$controllable and $\omega-$closed with respect to the new plant $\mathbf{SF}^{f^*}$. Thus, the corresponding $*-$automaton $\mathbf{SUP}_1$ implements a complete and deadlock-free supervisor $f_1^{\omega} : L(\mathbf{SF}^{f^*}) \to \Gamma$, i.e.

$$L([\mathbf{SF}^{f^*}]^{f_1^{\omega}}) = L(\mathbf{SF}^{f^*}) \cap L(\mathbf{SUP}_1),$$
$$S(\mathbf{SUP}_1^{\omega}) = S([\mathbf{SF}^{f^*}]^{f_1^{\omega}}) = S(\mathbf{SF}^{f^*}) \cap \lim(\mathbf{SUP}_1).$$

By the Thistle's supervisor synthesis approach, another Büchi automaton $\mathbf{SUP}_2^{\omega}$ with the Büchi acceptance criterion $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 14\}$ is obtained, as displayed in Fig. 20, which has 23 states and 32 transitions. Also, the corresponding $*-$automaton $\mathbf{SUP}_2$ implements another complete and deadlock-free supervisor $f_2^{\omega} : L(\mathbf{SF}^{f^*}) \to \Gamma$,

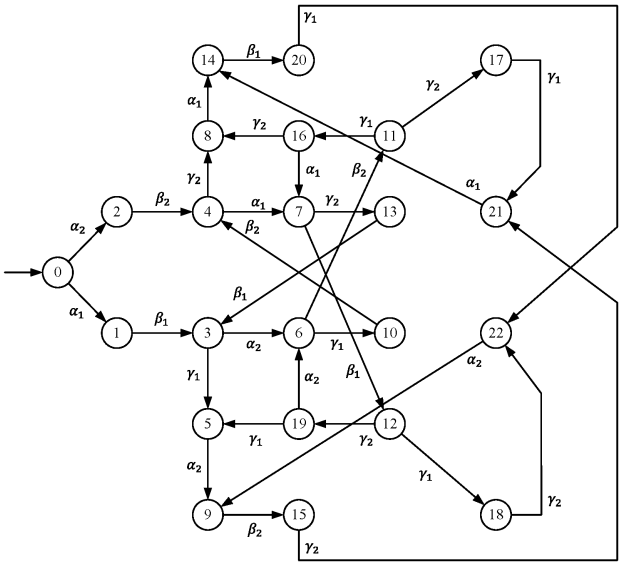

FIGURE 20. Transition graph of $\mathbf{SUP}_2^{\omega}$.

i.e.

$$L([\mathbf{SF}^{f^*}]^{f_2^{\omega}}) = L(\mathbf{SF}^{f^*}) \cap L(\mathbf{SUP}_2),$$
$$S(\mathbf{SUP}_2^{\omega}) = S([\mathbf{SF}^{f^*}]^{f_2^{\omega}}) = S(\mathbf{SF}^{f^*}) \cap \lim(\mathbf{SUP}_2).$$

It is easily verified that both of the controlled behavior satisfy the given liveness specifications, i.e.

$$A \subseteq S([\mathbf{SF}^{f^*}]^{f_1^{\omega}}) \subseteq E_l,$$
$$A \subseteq S([\mathbf{SF}^{f^*}]^{f_2^{\omega}}) \subseteq E_l.$$

It is also verified that $S(\mathbf{SUP}_2^{\omega}) \subseteq S(\mathbf{SUP}_1^{\omega})$, which means that the controlled plant behavior of the supervisor synthesized by our approach is more permissive than that of the Thistle's supervisor synthesis approach. The reason for this difference is as follows.

TABLE 2. Calculation of $C^{\mathcal{A}}$.

i	j	k	l	$C_i^{\mathcal{A}}$	$C_{i,j}^{\mathcal{A}}$	$C_{i,j,k}^{\mathcal{A}}$	$C_{i,j,k,l}^{\mathcal{A}}$	$\phi_{i,j,k,l}^{\mathcal{A}}$
0				$\emptyset$				
1	0				$\emptyset$			
1	1	0	0				$\emptyset$	
1	1	0	1				$\emptyset$	
1	1	0				$\emptyset$		
1	1	1	0				$\emptyset$	
1	1	1	1				$\{2\}$	$\phi^{\mathcal{A}}(2) = \{17\}$
1	1	1	2				$\{2,3,4\}$	$\phi^{\mathcal{A}}(3) = \{20\}$;$\phi^{\mathcal{A}}(4) = \{19\}$
1	1	1				$\{2,3,4\}$		
1	1				$\{2,3,4\}$			
1				$\{2,3,4\}$				
2	0				$\emptyset$			
2	1	0	0				$\emptyset$	
2	1	0	1				$\{1,2,3,4\}$	$\phi^{\mathcal{A}}(1) = \{15\}$
1	1	0				$\{1,2,3,4\}$		
2	1	1	0				$\emptyset$	
2	1	1	1				$\{2,3,4\}$	
2	1	1				$\{2,3,4\}$		
2	1				$\{1,2,3,4\}$			
2				$\{1,2,3,4\}$				
3	0				$\emptyset$			
3	1	0	0				$\emptyset$	
3	1	0	1				$\{0,1,2,3,4\}$	$\phi^{\mathcal{A}}(0) = \{11\}$
3	1	0				$\{0,1,2,3,4\}$		
3	1	1	0				$\emptyset$	
3	1	1	1				$\{2,3,4\}$	
3	1	1				$\{2,3,4\}$		
3	1				$\{0,1,2,3,4\}$			
3				$\{0,1,2,3,4\}$				
4	0				$\emptyset$			
4	1	0	0				$\emptyset$	
4	1	0	1				$\{0,1,2,3,4,5\}$	$\phi^{\mathcal{A}}(5) = \{10\}$
4	1	0				$\{0,1,2,3,4,5\}$		
4	1	1	0				$\emptyset$	
4	1	1	1				$\{2,3,4\}$	
4	1	1				$\{2,3,4\}$		
4	1				$\{0,1,2,3,4,5\}$			
4				$\{0,1,2,3,4,5\}$				

By inspecting the transition graph of **RBA**, according to the definition of $\phi_{i,j,k,l}^{\mathcal{A}} : C_{i,j,k,l}^{\mathcal{A}} \rightarrow C$ (as described in the proof of Theorem 8.15 in [6]), at state 15 the enablement of event α_1 and that of event γ_2 belong to two different control patterns, say ϕ_1 and ϕ_2, respectively. According to the Thistle's supervisor synthesis approach, only one of the control patterns is selected. In this example, ϕ_2 is selected; thus only event γ_2 is enabled by supervisor $\mathbf{SUP}_2^{\omega}$. However, by our approach, both of events α_1 and γ_2 are enabled, because the control patterns are closed under union.

Physically, at state 15, a workpiece in buffer $\mathbf{B}_1$ has been taken out for further processing, and a workpiece from $\mathbf{M}_2$ has been put into buffer $\mathbf{B}_2$. At this state, it is legal (without violating liveness specification (S3)) to enable event α_1 (i.e. $\mathbf{M}_1$ starts a new work cycle) if the supervisor can disable event 11 after one more work cycle (i.e. at state 18 of $\mathbf{SUP}_1^{\omega}$). Certainly, it is legal to disable event α_1 as in $\mathbf{SUP}_2^{\omega}$ (by the Thistle's supervisor synthesis approach); however, in that case the plant behavior will be more restrictive.

VI. CONCLUSION

We have first proposed an algorithm to compute the maximal subautomaton by deleting all bad loops in the transition graph of the automaton representing a given $\omega-$language of a specification. Then, based on this algorithm we have proposed a new approach to construct a supervisor for a specification language even if it is not $\omega-$closed, and proven that the constructed supervisor is often more permissive than the one computed by the Thistle's supervisor synthesis approach. In future work, we shall extend the proposed algorithm to study the supervisor synthesis approach for the nonblocking and deadlock-free supervisory control of DESs with infinite behavior.

APPENDIX

See Table 2.

REFERENCES

[1] W. M. Wonham and K. Cai, *Supervisory Control of Discrete-Event Systems*. New York, NY, USA: Springer, 2019.

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

[3] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, 2nd ed. New York, NY, USA: Springer-Verlag, 2008.
[4] L. Lamport, "Proving the correctness of multiprocess programs," *ACM Trans. Softw. Eng.*, vol. SE-3, no. 2, pp. 125–143, Mar. 1977.
[5] P. J. G. Ramadge, "Some tractable supervisory control problems for discrete-event systems modeled by Buchi automata," *IEEE Trans. Autom. Control*, vol. 34, no. 1, pp. 10–19, Jan. 1989.
[6] J. G. Thistle, "Control of infinite behavior of discrete event systems," Ph.D. dissertation, Syst. Control Group, Dept. Elect. Eng., Univ. Toronto, Toronto, ON, Canada, 1991.
[7] J. G. Thistle and W. M. Wonham, "Supervision of infinite behavior of discrete-event systems," *SIAM J. Control Optim.*, vol. 32, no. 4, pp. 1098–1113, 1994.
[8] J. G. Thistle and W. M. Wonham, "Control of infinite behavior of finite automata," *SIAM J. Control Optim.*, vol. 32, no. 4, pp. 1075–1097, 1994.
[9] S.-J. Park and K.-H. Cho, "State feedback control of real-time discrete event systems with infinite states," *Int. J. Control*, vol. 88, no. 5, pp. 1078–1088, 2015.
[10] C. de Oliveira, J. E. R. Cury, and C. A. A. Kaestner, "Synthesis of supervisors for parameterized and infinity non-regular discrete event systems," *IFAC Proc. Volumes*, vol. 40, no. 6, 2007, pp. 181–186.
[11] J. E. R. Cury and B. H. Krogh, "Design of robust supervisors for discrete event systems with infinite behavior," in *Proc. 35th Conf. Decis. Control*, Kobe, Japan, vol. 2, Dec. 1996, pp. 2219–2224.
[12] R. Kumar and V. K. Garg, "On computation of state avoidance control for infinite state systems in assignment program framework," *IEEE Trans. Autom. Sci. Eng.*, vol. 2, no. 1, pp. 87–91, Jan. 2005.
[13] S. Chédor, C. Morvan, S. Pinchinat, and H. Marchand, "Diagnosis and opacity problems for infinite state systems modeled by recursive tile systems," *Discrete Event Dyn. Syst.*, vol. 25, nos. 1–2, pp. 271–294, Jun. 2015.
[14] V. K. Garg and R. Kumar, "A state-variable approach for controlling discrete event systems with infinite states," in *Proc. Amer. Control Conf.*, Chicago, IL, USA, Jun. 1992, pp. 2809–2813.
[15] P. Gohari and W. M. Wonham, "Efficient implementation of fairness in discrete-event systems using queues," *IEEE Trans. Autom. Control*, vol. 50, no. 11, pp. 1845–1849, Nov. 2005.
[16] S. Young, D. Spanjol, and V. Garg, "Control of discrete event systems modeled with deterministic Buchi automata," in *Proc. Amer. Control Conf.*, Chicago, IL, USA, Jun. 1992, pp. 2814–2818.
[17] R. Zhang and K. Cai, "Supervisor localization of discrete-event systems with infinite behavior," *IFAC-PapersOnLine*, vol. 51, no. 7, pp. 361–366, 2018.
[18] F. Lu, Q. Zeng, M. Zhou, Y. Bao, and H. Duan, "Complex reachability trees and their application to deadlock detection for unbounded Petri Nets," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 6, pp. 1164–1174, Jun. 2019.
[19] S. G. Wang, D. You, and M. C. Zhou, "New reachability trees for analyzing unbounded Petri nets with semilinear reachability sets," *Sci. China Inf. Sci.*, vol. 61, no. 12, Dec. 2018, Art. no. 129104.
[20] J. Li, X. Yu, M. Zhou, and X. Dai, "Lean reachability tree for unbounded Petri nets," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 2, pp. 299–308, Feb. 2018.
[21] M. Wang, G. Liu, P. Zhao, C. Yan, and C. Jiang, "Behavior consistency computation for workflow nets with unknown correspondence," *IEEE/CAA J. Autom. Sinica*, vol. 5, no. 1, pp. 281–291, Jan. 2018.
[22] X. Guo, S. Wang, D. You, Z. Li, and X. Jiang, "A siphon-based deadlock prevention strategy for S^3PR," *IEEE Access*, vol. 7, pp. 86863–86873, 2019.
[23] S. Wang, D. You, and M. Zhou, "A necessary and sufficient condition for a resource subset to generate a strict minimal siphon in S^4PR," *IEEE Trans. Autom. Control*, vol. 62, no. 8, pp. 4173–4179, Aug. 2017.
[24] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: Massachusetts Institute of Technology, 2009.

WEI TANG was born in Xiangyang, China, in 1977. He received the M.S. degree in electrical engineering and the Ph.D. degree in control theory and control engineering from Northwestern Polytechnical University (NPU), Xi'an, China, in 2003 and 2006, respectively. He is currently an Associate Professor with NPU. His research interests include control systems and signal processing, and related applications.

TING JIAO received the B.S. degree in automation from Central South University, Changsha, China, in 2010, and the M.S. degree in control science and engineering and the Ph.D. degree in electrical engineering from Xi'an Jiaotong University, Xi'an, China, in 2013 and 2017, respectively. He was an International Visiting Student with the University of Toronto, Toronto, ON, Canada, from 2014 to 2015, sponsored by the China Scholarship Council. He is currently a Lecturer with the Department of Automation, Shanxi University, Taiyuan, China. His current research interests include supervisory control and reconfiguration of discrete-event systems.

RENYUAN ZHANG received the B.Eng. and Ph.D. degrees in electrical engineering from Xi'an Jiaotong University, Xi'an, China, in 2007 and 2013, respectively. He is currently an Associate Professor with the School of Automation, Northwestern Polytechnical University, Xi'an. He was an International Visiting Student with the University of Toronto, from 2011 to 2012. His research interests include distributed control of discrete-event systems and hybrid control of multi-agent systems.