Enforcement of Critical Observability in Modular Discrete-Event Systems

Shaowen Miao, Jan Komenda, Tomáš Masopust, and Aiwen Lai

***Abstract*— In safety-critical applications, the ability to distinguish between critical and noncritical states based on observations, known as *critical observability* (CO), is essential for ensuring reliability and security. We address the enforcement of CO in discrete-event systems (DES) through supervisory control. While a supremal CO sublanguage does not exist, we overcome this challenge by leveraging the concept of normality, proposing an algorithm to compute the least restrictive closed-loop system that is CO. We extend this approach to modular DES, showing how CO can be enforced locally to achieve global enforcement, thereby addressing scalability challenges. Finally, we integrate our framework with safety specifications, enabling the synthesis of supervisors that concurrently ensure both safety and CO.**

***Index Terms*— Discrete-event systems, modular control, deterministic finite automata, critical observability.**

I. Introduction

Ensuring reliability and safety in complex systems is essential, especially given their increasing scale and safety-critical requirements. Discrete-event systems (DES) provide a powerful framework for modeling such systems. To effectively manage the inherent complexity of large-scale DES, modular control [1], [2] has emerged as a crucial paradigm, enabling complexity mitigation through the synthesis of local supervisors.

Within this context, *critical observability* (CO) is a fundamental property of cyber-physical systems, defining the ability to distinguish between designated critical and noncritical states based on observed events. In safety-critical applications, where certain operations may be unsafe or demand heightened attention, CO is essential for effective monitoring, diagnosis, and control to guarantee system reliability and security.

The concept of CO was introduced by Pola et al. [3] for finite automata and extended to Petri nets by Masopust [4]. Verifying CO is computationally challenging, being NL-complete for DES, PSPACE-complete for modular DES, and undecidable for labeled Petri nets (unless the set of critical markings is finite or co-finite) [4]. The inherent complexity underscores the need for effective methods to enforce this property.

While Pola et al. [3] primarily focused on verifying CO in networks of nondeterministic automata with full observation, employing bisimilarity-based reductions to manage complexity, our work takes a different direction. We focus on the *enforcement* of CO. Specifically, we leverage deterministic automata under partial observation and introduce a novel modular approach for CO enforcement through supervisory control, directly capitalizing on the plant's modular structure.

The enforcement of CO for bounded labeled Petri nets has been explored by Cong et al. [5], who utilized basis markings and integer linear programming (ILP). However, their monolithic ILP-based approach inherently faces significant computational challenges for large systems. In contrast, our work proposes a modular approach that enforces CO locally. This design paradigm ensures that the complexity remains exponential only in the size of individual local subsystems, rendering it highly advantageous for large-scale systems composed of numerous smaller components. Quantitatively, for n subsystems each with m states, our modular method achieves a complexity of $O(nm2^m)$, a substantial improvement over the monolithic complexity of $O(m^n 2^{m^n})$, which our method avoids. Furthermore, Cong et al. [5] did not address the non-existence of the supremal CO sublanguage, and their online control algorithm is not directly suitable for computing maximally permissive sublanguages. The integration with safety specifications was not considered in their framework.

Within the context of supervisory control theory [6], the synthesis of supervisors under partial observation necessitates satisfying both controllability and observability. A key challenge is that, unlike controllability, observability is not closed under language union, which typically leads to multiple incomparable maximal observable supervisors rather than a unique supremal supervisor. To overcome this limitation and ensure the existence of a unique supremal supervisor, concepts such as normality [7] and relative observability (RO) [8] were introduced, both of which are closed under language union. It is known that normality implies RO, and RO, in turn, implies observability. While enforcing RO can yield a more permissive supervisor in monolithic systems compared to normality, it introduces additional computational demands and constraints in modular scenarios [9]. Consequently, for the synthesis of supervisors under partial observation, we focus on exploiting the properties of normality rather than RO.

Normality [7] is a well-established property that determines

Supported by Fujian Provincial Natural Science Foundation of China under Grant 2025J01054; by the Natural Science Foundation of Xiamen, China under Grant 3502Z202573035; and by RVO 67985840. *(Corresponding author: Aiwen Lai.)*

S. Miao and A. Lai are with the Department of Automation, Xiamen University, Xiamen 361102, China. *miaosw0706@stu.xmu.edu.cn, aiwenlai@xmu.edu.cn*

J. Komenda is with the Institute of Mathematics of the Czech Academy of Sciences, 115 67 Prague, Czechia. *komenda@ipm.cz*

T. Masopust is with the Faculty of Science, Palacky University Olomouc, Czechia. *tomas.masopust@upol.cz*

whether a given string adheres to a safety specification by considering observable events. Conceptually, it signifies the ability to unambiguously distinguish between pairs of strings where one is legal (safe) and the other is not. Since its introduction, normality has been widely investigated in the literature [10]–[15]. Unlike observability, normality guarantees the existence of a unique supremal normal sublanguage, rendering it a particularly robust and desirable property in supervisory control synthesis. For a more comprehensive discussion, we direct the reader to dedicated works on normality [10], [12], [16], [17].

Our primary contributions are threefold: First, we develop a novel approach for enforcing CO of a given plant using supervisory control. We show that, despite the general nonexistence of a supremal CO sublanguage, the problem can be effectively resolved by incorporating normality. Our approach enables the unique synthesis of a supervisor that achieves the least restrictive CO closed-loop system, for which we propose an algorithm. Second, we extend the enforcement framework to modular DES. Building upon established results for modular controllability and normality, we show that a supervisor enforcing CO can be synthesized in a modular fashion, thereby offering significant scalability advantages for large systems. Third, we integrate the modular CO enforcement framework with external safety specifications. This extension allows for the concurrent synthesis of supervisors that ensure both desired safety properties and CO of the system.

The remainder of the paper is organized as follows. Section II reviews fundamental concepts from automata theory and supervisory control. Section III-A details the proposed algorithm for deriving a CO closed-loop system in a monolithic setting. The modular computation of supervisors enforcing CO is presented in Section III-B, building on the supremal controllable and normal sublanguages of CO subautomata. Section III-C develops the framework for incorporating external safety specifications. Section IV concludes the paper.

II. Preliminaries and Definitions

We first review the basic concepts of automata theory and supervisory control [6], [18] essential for subsequent discussion.

The cardinality of a set A is denoted by $|A|$. An alphabet, Σ, is a finite nonempty set of events. By Σ^* we denote the set of strings over Σ of finite length, including the empty string ε. The concatenation of strings s_1 and s_2 is the string $s_1 s_2$.

A language over Σ is a subset of Σ^*. The concatenation of languages L_1 and L_2 is the language $L_1 L_2 = \{st \mid s \in L_1 \text{ and } t \in L_2\}$. The *prefix-closure* of a language L over Σ is the set $\overline{L} = \{s \in \Sigma^* \mid \exists t \in \Sigma^* \text{ such that } st \in L\}$ of its prefixes. If $L = \overline{L}$, then L is *prefix-closed*.

A *projection* $R\colon \Sigma^* \to \Gamma^*$, for $\Gamma \subseteq \Sigma$, is a morphism for concatenation defined by $R(\sigma) = \varepsilon$ for $\sigma \in \Sigma \setminus \Gamma$ and $R(\sigma) = \sigma$ for $\sigma \in \Gamma$. The action of R on a string $\sigma_1 \cdots \sigma_n \in \Sigma^*$ is to remove events from $\Sigma \setminus \Gamma$, i.e., $R(\sigma_1 \cdots \sigma_n) = R(\sigma_1) \cdots R(\sigma_n)$. The inverse of R is defined by $R^{-1}(t) = \{s \in \Sigma^* \mid R(s) = t\}$. The definitions can be extended to languages in a usual way.

A *deterministic finite automaton* (DFA) is the quintuple $G = (Q, \Sigma, \delta, q_0, Q_m)$, where Q is a finite set of states, Σ is an alphabet, $\delta\colon Q \times \Sigma \to Q$ is a partial transition function that can be extended to the domain $Q \times \Sigma^*$ by induction, $q_0 \in Q$ is the initial state, and $Q_m \subseteq Q$ is the set of marked states. The *language generated* by G is $L(G) = \{s \in \Sigma^* \mid \delta(q_0, s) \in Q\}$ and the *language marked* by G is $L_m(G) = \{s \in \Sigma^* \mid \delta(q_0, s) \in Q_m\}$. By definition, $L_m(G) \subseteq L(G)$ and $L(G)$ is prefix-closed. If $\overline{L_m(G)} = L(G)$, then G is *nonblocking*. For a language K over Σ, we define the set of states reachable in G under strings of K by $\delta(q_0, K) = \{\delta(q_0, t) \mid t \in K \cap L(G)\}$.

A *subautomaton* of G is a DFA obtained from G by removing some transitions and states (together with their transitions).

The *observer* of G with respect to a projection P is denoted by $\mathrm{Obs}(G)$ and defined as the accessible part of the DFA obtained by the standard subset construction applied to a nondeterministic automaton computed from G by replacing every event a by $P(a)$; see [18] for more details.

An automaton G is a *state-partition automaton* (SPA) with respect to P if every two states of its observer $\mathrm{Obs}(G)$ are either identical or their intersection is empty.

A. Supervisory Control

A system G over Σ is partially observed and partially controlled if the alphabet Σ is partitioned into *observable events* Σ_o and *unobservable events* $\Sigma_{uo} = \Sigma \setminus \Sigma_o$, and into *controllable events* Σ_c and *uncontrollable events* $\Sigma_{uc} = \Sigma \setminus \Sigma_c$.

Let $\Gamma = \{\gamma \subseteq \Sigma \mid \Sigma_{uc} \subseteq \gamma\}$ be the set of control patterns, and let P be the projection from Σ^* to Σ_o^*. The *supervisor* of G with respect to Γ is a map $S\colon P(L(G)) \to \Gamma$ that defines the behavior of the *closed-loop system* S/G as follows: $\varepsilon \in L(S/G)$; if $s \in L(S/G)$, $s\sigma \in L(G)$, and $\sigma \in S(P(s))$, then $s\sigma \in L(S/G)$. Intuitively, based on the observation $P(s)$, the supervisor disables events from $\Sigma_c \setminus S(P(s))$. Given a specification K, the language marked by the closed loop is $L_m(S/G) = L(S/G) \cap K$. The supervisor is nonblocking if the closed loop is nonblocking, i.e., $\overline{L_m(S/G)} = L(S/G)$. In particular, we consider only supervisors S that can be represented by an automaton, G_S. Then $S/G = G_S \parallel G$, and we can consider S/G to be an automaton.

A language $K \subseteq L(G)$ is *controllable* with respect to $L(G)$ and the set of uncontrollable events Σ_{uc} if $\overline{K}\Sigma_{uc} \cap L(G) \subseteq \overline{K}$. A language $K \subseteq L(G)$ is *normal* with respect to $L(G)$ and a projection P if $\overline{K} = P^{-1}[P(\overline{K})] \cap L(G)$.

Given a plant G over Σ and a language $K \subseteq \Sigma^*$, we use the notation $\mathrm{supCN}(K, L(G), \Sigma_{uc}, P)$ to denote the supremal sublanguage of $K \cap L(G)$ that is controllable with respect to $L(G)$ and Σ_{uc}, and normal with respect to $L(G)$ and P. Analogously, we denote the supremal normal sublanguage of $K \cap L(G)$ with respect to $L(G)$ and P by $\mathrm{supN}(K, L(G), P)$. Sometimes, we simplify the notation and write the automaton, H, instead of its language $L(H)$; for instance, if $K = L(H)$, we write $\mathrm{supCN}(H, G, \Sigma_{uc}, P)$ instead of $\mathrm{supCN}(K, L(G), \Sigma_{uc}, P)$. When Σ_{uc} and P are clear from the context, we drop them from the notation.

B. Modular Supervisory Control

The parallel composition of languages L_i over Σ_i, for $i = 1, \ldots, n$, $n \geq 2$, is the language $\|_{i=1}^n L_i = \bigcap_{i=1}^n P_i^{-1}(L_i)$, where each projection $P_i\colon (\bigcup_{i=1}^n \Sigma_i)^* \to \Sigma_i^*$ erases all events

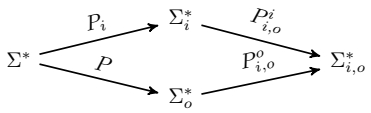

Fig. 1: Projection notations in this note.

that are not in Σ_i. For the definition of parallel composition for automata, we refer to the literature [18]. In particular, if $G_1, \ldots, G_n$ are automata, then $L(\|_{i=1}^n G_i) = \|_{i=1}^n L(G_i)$ and $L_m(\|_{i=1}^n G_i) = \|_{i=1}^n L_m(G_i)$. The languages L_i are *nonconflicting* if $\|_{i=1}^n L_i = \|_{i=1}^n \overline{L_i}$.

A *modular system* $\{G_i\}_{i=1}^n$ consists of $n \geq 2$ automata, also known as modules, G_i over Σ_i for $i = 1, \ldots, n$. The behavior of the modular system is the behavior of the *monolithic* system G defined as a parallel composition of the individual modules, that is, $G = \|_{i=1}^n G_i$. Shared events of the modules form the set $\Sigma_s = \bigcup_{i \neq j} (\Sigma_i \cap \Sigma_j)$. For every event a shared by two modules G_i and G_j, we assume that if a is controllable/observable in G_i then it is also controllable/observable in G_j.

The modular control problem involves a modular system $\{G_i\}_{i=1}^n$ with the global (monolithic) behavior $L = \|_{i=1}^n L(G_i)$, and a specification K given either as the parallel composition $K = \|_{i=1}^n K_i$ of local specifications $K_i \subseteq L(G_i)$, or as a global specification $K \subseteq \|_{i=1}^n L(G_i)$. The objective is to synthesize local supervisors S_i such that $\|_{i=1}^n L_m(S_i/G_i) = L_m(S/\|_{i=1}^n G_i)$, where S is a nonblocking and maximally permissive monolithic supervisor for specification K and global language L.

We denote by $\Sigma_{i,o} = \Sigma_i \cap \Sigma_o$ the set of local observable events. A similar notation applies to the intersection of other alphabets. The projections we use are summarized in Fig. 1.

III. Enforcement of Critical Observability

In this section, we present our methodology for enforcing CO of a given plant using supervisory control. We begin by recalling the definition of CO [3].

Definition 1 (Critical Observability): A DFA $G = (Q, \Sigma, \delta, q_0, Q_m)$ is *critically observable* (CO) with respect to a projection P and a set of critical states $Q_c \subseteq Q$ if, for every string $s \in L(G)$, either $\delta(q_0, P^{-1}[P(s)]) \subseteq Q_c$ or $\delta(q_0, P^{-1}[P(s)]) \subseteq Q \setminus Q_c$.

The concept of CO requires that, given an observation of the system, one can unambiguously determine whether the system is currently in a critical state. This property is symmetric with respect to critical and noncritical states; interchanging Q_c and $Q \setminus Q_c$ in the definition does not alter whether a DFA is critically observable.

A. Monolithic Computation

To enforce CO for a monolithic plant, we synthesize a supervisor that restricts the plant's behavior. This necessitates a formal definition of a CO sublanguage.

Definition 2 (Critically Observable Sublanguage): Let $G = (Q, \Sigma, \delta, q_0, Q_m)$ be a DFA, and let $Q_c \subseteq Q$ be a set of critical states. A language $K \subseteq L(G)$ is *critically observable* (CO) with respect to G, P, and Q_c if, for every $s, s' \in \overline{K}$ with $P(s) = P(s')$, $\delta(q_0, s) \in Q_c$ if and only if $\delta(q_0, s') \in Q_c$.

Any sublanguage of a critically observable plant is critically observable, as formalized by the following lemma.

Lemma 1: Let $G = (Q, \Sigma, \delta, q_0, Q_m)$ be a DFA that is CO with respect to a projection P and a set of critical states $Q_c \subseteq Q$. Then every language $K \subseteq L(G)$ is CO with respect to G, P, and Q_c.

Proof: For every $s, s' \in \overline{K} \subseteq L(G)$ with $P(s) = P(s')$, the definition of CO for G directly implies that $\delta(q_0, s) \in Q_c$ if and only if $\delta(q_0, s') \in Q_c$. ■

To enforce CO of a given DFA G, our aim is to synthesize a supervisor ensuring that the supervised automaton achieves this property. However, we can achieve only behaviors that are controllable and observable. Since observability is not closed under language union, we use normality instead. Therefore, our aim is to synthesize a supervisor that achieves the *supremal controllable, normal, and critically observable sublanguage of* $L(G)$ with respect to G, Σ_{uc}, P, and Q_c, which we denote by

$$\mathrm{supCNCO}(G, \Sigma_{uc}, P, Q_c)$$

or simply $\mathrm{supCNCO}(G)$ if the components are clear from the context. We now show that $\mathrm{supCNCO}(G)$ exists.

Theorem 1: Let G be a DFA, and let $M_i \subseteq L(G)$, for $i \in I$, be controllable, normal, and CO with respect to G, P, Σ_{uc}, and Q_c. Then $\bigcup_{i \in I} M_i$ is controllable, normal, and CO with respect to G, P, Σ_{uc}, and Q_c.

Proof: The union $\bigcup_{i \in I} M_i$ is controllable and normal with respect to G, Σ_{uc}, and P, since controllability and normality are closed under union. It remains to show that $\bigcup_{i \in I} M_i$ is CO with respect to G, P, and Q_c. For the sake of contradiction, assume that there are $m_i \in \overline{M_i}$ and $m_j \in \overline{M_j}$ such that $P(m_i) = P(m_j)$, $\delta_G(q_0, m_i) = p$, $\delta_G(q_0, m_j) = q$, and p is critical if and only if q is not. By normality of M_i, we have $m_j \in P^{-1}[P(m_i)] \cap L(G) \subseteq \overline{M_i}$. Consequently, M_i is not CO with respect to G, P, and Q_c, which is a contradiction. ■

Given a DFA G, the fundamental question is how to compute $\mathrm{supCNCO}(G)$. Since $L(G)$ is controllable and normal with respect to itself, we basically need to find a "suitable" CO sublanguage of $L(G)$.

Ideally, one would compute the supremal CO sublanguage. Unfortunatelly, this language does not exist—consider the SPA G with transitions $(1, a, 2)$, $(1, \tau, 3)$, and $(3, a, 4)$, where a is observable, τ is unobservable, and the initial state is 1. Then G is not CO with respect to $Q_c = \{2\}$. However, removing state 2 or state 4 yields two different subautomata, G_2 and G_4, both of which are CO with respect to Q_c.

Therefore, we identify a subautomaton G' of $\tilde{G} = G \parallel \mathrm{Obs}(G)$ as described in Algorithm 1, whose language is CO with respect to $\tilde{G}$. Then, we synthesize a supervisor that can achieve $L(G')$ by restricting the plant G. Since it may not always be possible to achieve $L(G')$ entirely, we compute the supremal sublanguage of $L(G')$ that can be enforced by a supervisor, specifically we compute $\mathrm{supCN}(G', G)$.

The automaton $\tilde{G} = G \parallel \mathrm{Obs}(G)$ is a state-partition automaton (SPA) [19] that plays a key role in our approach. Therefore, we focus on SPAs for a while. In Algorithm 2, we show that this is only a technical detail and not a restriction.

Algorithm 1 formally describes the construction of a "suitable" CO subautomaton of a given SPA G. It is based on a

modified parallel composition of two copies of G, denoted by $G \,|||\, G$. This composition models the indistinguishability of unobservable events. Specifically, let $G = (Q, \Sigma, \delta, q_0, F)$ be a DFA and $\Sigma_o \subseteq \Sigma$ be the set of observable events. We define $G \,|||\, G$ as the accessible part of the nondeterministic automaton $(Q \times Q, \Sigma, f, (q_0, q_0), F \times F)$, where

$$f((x,y),e) = \begin{cases} \{(\delta(x,e), \delta(y,e))\} & \text{if } e \in \Sigma_o \\ \{(\delta(x,e), y), (x, \delta(y,e))\} & \text{if } e \notin \Sigma_o \,. \end{cases}$$

The following result from [4] relates CO to reachability in the automaton $G \,|||\, G$.

Lemma 2: Let $G = (Q, \Sigma, \delta, q_0, F)$ be a DFA, $\Sigma_o \subseteq \Sigma$ be the set of observable events, and $Q_c \subseteq Q$ be a set of critical states. Then G is not critically observable with respect to Σ_o and Q_c if and only if there is a reachable state in $G \,|||\, G$ that belongs to the set $Q_c \times (Q \setminus Q_c)$. ∎

We can now formulate the algorithm for computing a "suitable" subautomaton of a given SPA.

Algorithm 1 Construction of a CO Subautomaton

Input: An SPA G, critical states Q_c, and a projection P.
Output: A subautomaton of G whose language is CO wrt G.
1: Set $G' \leftarrow G$
2: Compute the accessible part of $H \leftarrow G' \,|||\, G'$
3: **for** every reachable state $(p, q) \in Q_c \times (Q \setminus Q_c)$ of H **do**
4: Remove both state p and state q from G' together with the corresponding transitions
5: **return** the accessible part of G'

Let G' be the result of Algorithm 1 applied to an SPA G. We define a technical auxiliary language $N(G)$ as the supremal normal sublanguage of $L(G')$ with respect to G and P, i.e.,

$$N(G) = \mathrm{supN}(G', G)$$

that will be useful later. However, to enforce CO via supervisory control, we need not only normality, but also controllability. Therefore, we further define

$$\mathrm{Opt}(G) = \mathrm{supCN}\big(N(G), G\big)\,,$$

which is our desired language, as we show below.

Lemma 3: $\mathrm{Opt}(G) = \mathrm{supCN}(G', G)$.

Proof: As $N(G) \subseteq L(G')$, we have $\mathrm{supCN}(N(G), G) \subseteq \mathrm{supCN}(G', G)$. On the other hand, $\mathrm{supCN}(G', G) \subseteq N(G)$, which implies $\mathrm{supCN}(G', G) = \mathrm{supCN}(\mathrm{supCN}(G', G), G) \subseteq \mathrm{supCN}(N(G), G)$. ∎

We are now ready to formulate Algorithm 2 that computes $\mathrm{supCNCO}(G)$ for a DFA G. To prove correctness of Algorithm 2, we provide several auxiliary results.

The first result is a simple observation that we implicitly use in the following, see, e.g., [18], [19].

Lemma 4: Let G be a DFA, then $\tilde{G} = G \parallel \mathrm{Obs}(G)$ is an SPA, and $L(G) = L(\tilde{G})$. ∎

The second result states that a language is CO with respect to G if and only if it is CO with respect to $\tilde{G}$.

Lemma 5: Let G be a DFA with the set of states Q, and let $\tilde{G} = G \parallel \mathrm{Obs}(G)$. Then, for every $K \subseteq L(G)$, K is CO with respect to G, P, and Q_c if and only if K is CO with respect to $\tilde{G}$, P, and $Q_c \times 2^Q$.

Proof: Let $x_0 \in Q \times 2^Q$ denote the initial state of $\tilde{G}$. For every $s \in L(G) = L(\tilde{G})$, we have $\delta_G(q_0, s) \in Q_c$ if and only if $\delta_{\tilde{G}}(x_0, s) \in Q_c \times 2^Q$, which proves the claim. ∎

Algorithm 2 Computation of $\mathrm{supCNCO}(G)$

Input: A DFA $G = (Q, \Sigma, \delta, q_0, F)$, critical states $Q_c \subseteq Q$, and a projection P.
Output: $\mathrm{supCNCO}(G, \Sigma_{uc}, P, Q_c)$.
1: Set $\tilde{G} \leftarrow G \parallel \mathrm{Obs}(G)$
2: Set the critical states of $\tilde{G}$ to be $Q_c \times 2^Q$
3: Compute $N(\tilde{G}) = \mathrm{supN}(\tilde{G}', \tilde{G})$, where $\tilde{G}'$ is obtained from $\tilde{G}$ by Algorithm 1
4: Compute $\mathrm{Opt}(\tilde{G}) = \mathrm{supCN}(N(\tilde{G}), \tilde{G})$
5: **return** $\mathrm{Opt}(\tilde{G})$

The following result is a simple observation.

Lemma 6: Consider the automata and languages of Algorithm 2, then $\mathrm{supCN}(N(\tilde{G}), \tilde{G}) = \mathrm{supCN}(N(\tilde{G}), G)$.

Proof: By Lemma 4, $L(\tilde{G}) = L(G)$, and therefore $\mathrm{supCN}(N(\tilde{G}), L(\tilde{G})) = \mathrm{supCN}(N(\tilde{G}), L(G))$. ∎

The next result is key to the correctness of Algorithm 2.

Lemma 7: Consider the automata and languages used in Algorithm 2. Then, $\mathrm{supN}(L', G) \subseteq N(\tilde{G})$ for every language $L' \subseteq L(G)$ that is CO with respect to G, P, and Q_c.

Proof: For the sake of contradiction, we assume that there is a string $w \in \mathrm{supN}(L', G)$ such that $w \notin N(\tilde{G})$. Without loss of generality, we assume that w is the shortest in the sense that every strict prefix of w belongs to both languages. Since $w \in L'$ and L' is CO with respect to G, we have

$$\delta_G(q_0, P^{-1}[P(w)]) \subseteq Q_c \text{ or } \delta_G(q_0, P^{-1}[P(w)]) \subseteq Q \setminus Q_c\,. \quad (1)$$

Let $x_0 \in Q \times 2^Q$ be the initial state of $\tilde{G}$. Since $w \notin N(\tilde{G})$, but every strict prefix of w does, the state $\delta_{\tilde{G}}(x_0, w) = (q, X)$ was removed from $\tilde{G}$ by Step 3 of Algorithm 2. Thus, there are $w', w'' \in L(G)$ such that $\delta_{\tilde{G}}(x_0, w') = (q, X)$, $\delta_{\tilde{G}}(x_0, w'') = (p, Y)$, and (q, X) is critical if and only if (p, Y) is not. Since $\tilde{G}$ is an SPA, $\delta_{\tilde{G}}(x_0, P^{-1}[P(w)]) = \delta_{\tilde{G}}(x_0, P^{-1}[P(w')])$. However, $\{(q, X), (p, Y)\} \subseteq \delta_{\tilde{G}}(x_0, P^{-1}[P(w')])$, and by Lemma 5, $\{p, q\} \subseteq \delta_G(q_0, P^{-1}[P(w)])$, which contradicts (1). ∎

We are now ready to prove the correctness of Algorithm 2.

Theorem 2: Algorithm 2 returns $\mathrm{supCNCO}(G)$.

Proof: Let G be a DFA, and $M = \mathrm{supCNCO}(G)$, then

$$\begin{aligned} \mathrm{Opt}(\tilde{G}) &= \mathrm{supCN}(N(\tilde{G}),\, \tilde{G}) \\ &= \mathrm{supCN}(N(\tilde{G}),\, G) && \text{(Lemma 6)} \\ &\subseteq M = \mathrm{supCN}(M,\, G) \\ &\subseteq \mathrm{supCN}(N(\tilde{G}),\, G) && \text{(Lemma 7)} \\ &= \mathrm{supCN}(N(\tilde{G}),\, \tilde{G}) && \text{(Lemma 6)} \\ &= \mathrm{Opt}(\tilde{G}). \end{aligned}$$ ∎

We now illustrate the enforcement procedure.

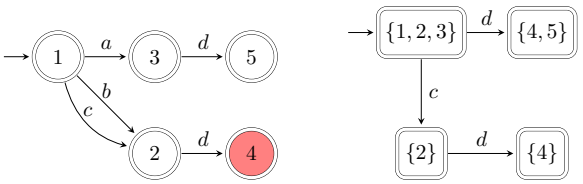

Fig. 2: A DFA G with $\Sigma_{uo} = \{a, b\}$, $\Sigma_{uc} = \{c\}$, and $Q_c = \{4\}$, and $\mathrm{Obs}(G)$.

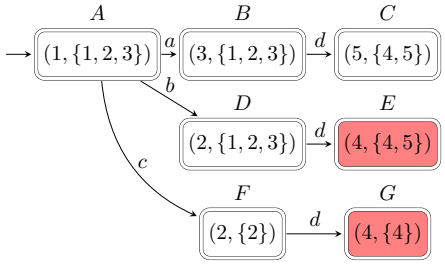

Fig. 3: SPA $\tilde{G} = G \parallel \mathrm{Obs}(G)$ with critical states $(4, \{4, 5\})$ and $(4, \{4\})$.

Example 1: We consider the DFA G depicted in Fig. 2, together with its observer. Algorithm 2 constructs $\tilde{G} = G \parallel \mathrm{Obs}(G)$ shown in Fig. 3. The subautomaton G' of $\tilde{G}$ is achieved by removing all states from $\tilde{Q}_c \times (\tilde{Q} \setminus \tilde{Q}_c)$ reachable in $\tilde{G} \,|||\, \tilde{G}$, see Fig. 4. The resulting language $\mathrm{Opt}(\tilde{G}) = N(\tilde{G}) = \mathrm{supN}(G', \tilde{G})$ is depicted in Fig. 4. ◇

B. Modular Computation

We now extend our framework to the supervisor enforcement of CO for modular systems $\{G_i\}_{i=1}^n$, where each module $G_i = (Q_i, \Sigma_i, \delta_i, q_{0,i}, Q_{m,i})$ is a DFA with the local set of critical states $Q_{c,i} \subseteq Q_i$.

The question is how to define the critical states Q_c of the monolithic system $G = \|_{i=1}^n G_i$ based on the critical states of the local modules G_i. There are many options, and therefore we formulate the definition of Q_c in general as follows:

$$Q_c = \{(q_1, \ldots, q_n) \in \prod_{i=1}^{n} Q_i \mid \varphi(q_1, \ldots, q_n)\}\,, \qquad (2)$$

where φ is a formula over the variables $q_1, \ldots, q_n$ and the local critical sets $Q_{c,1}, \ldots, Q_{c,n}$. For example,

$$\varphi(q_1, \ldots, q_n) \equiv \exists i \in \{1, \ldots, n\}.(q_i \in Q_{c,i})$$

results in the definition of Pola et al. [3], while

$$\varphi(q_1, \ldots, q_n) \equiv \forall i \in \{1, \ldots, n\}.(q_i \in Q_{c,i})$$

is the standard Cartesian product of $Q_{c,i}$ for $i = 1, \ldots, n$.

For the following result, we require that the parallel composition is closed under CO, that is, *if G_i is CO with respect to $Q_{c,i}$, for $i = 1, \ldots, n$, then $\|_{i=1}^n G_i$ is CO with respect to Q_c.* Note that, besides others, both the definition of Pola et al. [3] and the standard Cartesian product satisfy this property.

We now have the following result.

Theorem 3: Let $\{G_i\}_{i=1}^n$ be a modular plant, and let $G = \|_{i=1}^n G_i$ be its monolithic counterpart. Let Q_c be a set of critical states of G as defined in (2), such that if G_i is CO with respect to $Q_{c,i}$, for $i = 1, \ldots, n$, then G is CO with respect to Q_c. Then, $\|_{i=1}^n \mathrm{supCNCO}(G_i, \Sigma_{i,uc}, P_{i,o}^i, Q_{c,i})$ is controllable, normal, and CO with respect to G, Σ_{uc}, P, Q_c.

Proof: Since the languages $\mathrm{supCNCO}(G_i)$ are prefix closed, they are nonconflicting. Therefore, by [20, Proposition 4.6], [14, Theorem 16], and the assumption that Q_c preservers CO of the parallel composition, we obtain that $\|_{i=1}^n \mathrm{supCNCO}(G_i, \Sigma_{i,uc}, P_{i,o}^i, Q_{c,i})$ is controllable, normal, and CO with respect to G, $\Sigma_{uc} = \bigcup_{i=1}^n \Sigma_{i,uc}$, P from $\Sigma = \bigcup_{i=1}^n \Sigma_i$ to $\Sigma_o = \bigcup_{i=1}^n \Sigma_{i,o}$, and Q_c. ■

C. Safety Specifications

The enforced CO plant can be used to handle additional safety specifications in a classical supervisory control manner. Specifically, either the initial plant G is CO, or we have already synthesized a supervisor

$$S = \mathrm{supCNCO}(G)$$

such that the enforced plant S/G is CO. Given an arbitrary safety specification K, we proceed by computing the supremal controllable and normal sublanguage of K with respect to the CO plant S/G, i.e., we compute

$$\mathrm{supCN}(K, S/G)\,.$$

All existing results from the supervisory control theory apply to this problem, since we are computing the supremal controllable and normal sublanguage of K with respect to the plant S/G, see, e.g., [6], [18]. In particular, the modular approach of Komenda and Masopust [14] can be directly applied to handle both local and global safety specifications; see the reference for further details.

The following theorem is a direct consequence of Theorem 3 and the integration with safety.

Theorem 4: Let $\{G_i\}_{i=1}^n$ be a modular plant with G_i over Σ_i, and let $G = \|_{i=1}^n G_i$ be the monolithic counterpart over $\Sigma = \bigcup_{i=1}^n \Sigma_i$. For $K = \|_{i=1}^n K_i$ with $K_i \subseteq L(G_i)$, $S_i = \mathrm{supCNCO}(G_i, \Sigma_{i,uc}, P_{i,o}^i, Q_{c,i})$, and $S = \|_{i=1}^n S_i$, if the languages $\mathrm{supCN}(K_i, S_i/G_i)$ are nonconflicting, $P_i(S/G) = S_i/G_i$, and $\Sigma_s \subseteq \Sigma_o \cap \Sigma_c$, then $\mathrm{supCN}(K, S/G) = \|_{i=1}^n \mathrm{supCN}(K_i, S_i/G_i)$. ■

Example 2: Consider the SPA G_1 and the specification K_1 depicted in Fig. 5. Then, $S_1 = \mathrm{Opt}(G_1)$ is depicted in Fig. 6, and we have $\mathrm{supCN}(K_1, S_1/G_1) = \mathrm{Opt}(G_1)$. Let G_2 be the DFA depicted in Fig. 7. Then, $S_2 = \mathrm{Opt}(G_2) = G_2$, and $\mathrm{supCN}(K_2, S_2/G_2) = K_2$ is depicted in Fig. 7. Observable events are $\Sigma_o = \{a, b\}$, and shared events of G_1 and G_2 are $\Sigma_s = \{a, b\}$, which are also controllable. It can be verified that $P_i(S/G) = S_i/G_i$, and $\mathrm{supCN}(K_i, S_i/G_i)$ are prefix-closed, and hence nonconflicting. Therefore, Theorem 4 is applicable, and hence $\mathrm{supCN}(K_1 \| K_2, (S_1 \| S_2)/(G_1 \| G_2)) = \mathrm{supCN}(K_1, S_1/G_1) \parallel \mathrm{supCN}(K_2, S_2/G_2) = \{\varepsilon, a, a\tau, b, b\tau\} \parallel \{\varepsilon, a\} = \{\varepsilon, a, a\tau\}$. ◇

IV. Conclusion

We presented a novel approach to enforcing CO within the supervisory control framework for discrete-event systems. We first addressed a fundamental challenge by showing that while a supremal CO sublanguage does not exist in general, this

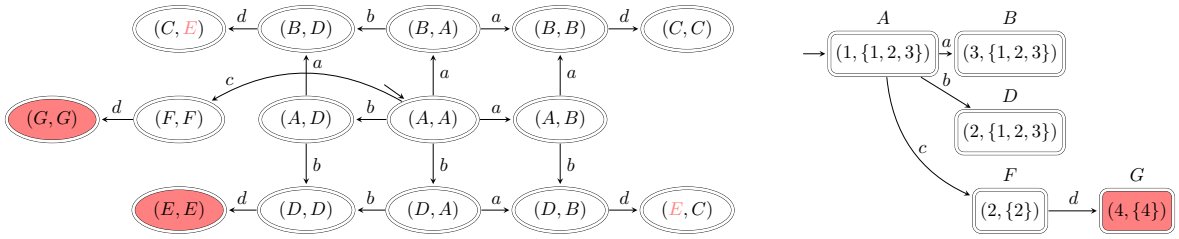

Fig. 4: The structure $\tilde{G} \,|||\, \tilde{G}$ and $\mathrm{Opt}(\tilde{G}) = N(\tilde{G}) = \mathrm{supN}(G', \tilde{G})$.

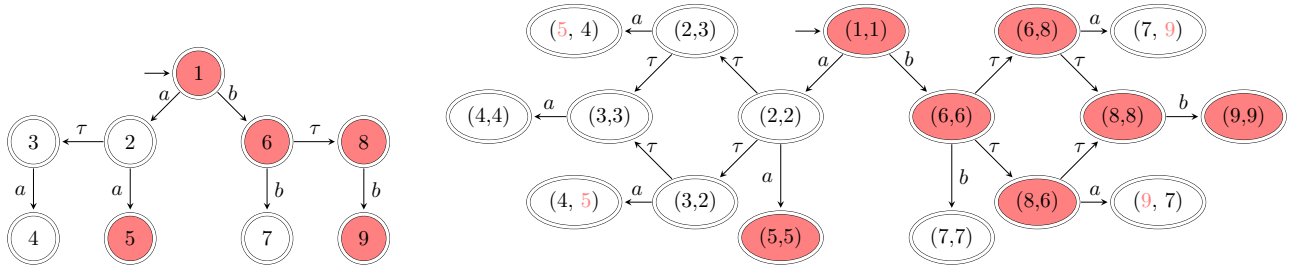

Fig. 5: SPA G_1 with $\Sigma_{o,1} = \Sigma_{c,1} = \{a, b\}$, $\Sigma_{uo,1} = \Sigma_{uc,1} = \{\tau\}$, and $Q_c = \{1, 5, 6, 8, 9\}$, where $K_1 = L(G_1)$, and $G_1 \,|||\, G_1$.

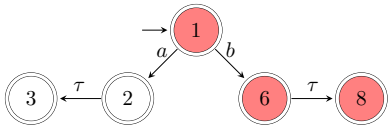

Fig. 6: $\mathrm{Opt}(G_1) = N(G_1) = \mathrm{supCN}(G_1', G_1)$.

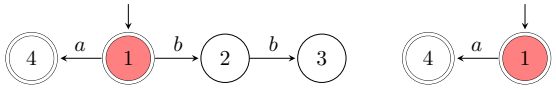

Fig. 7: DFA G_2 with $\Sigma_2 = \Sigma_{o,2} = \Sigma_{c,2} = \{a, b\}$ and $Q_{c,2} = \{1\}$, where $K_2 = \{\varepsilon, a\}$; $\mathrm{supCN}(K_2, S_2/G_2)$.

limitation can be effectively overcome by incorporating the concept of normality. Building on this insight, we proposed an algorithm to compute the least restrictive closed-loop system that inherently guarantees CO.

To tackle scalability, we developed a comprehensive modular enforcement framework. By leveraging existing results on controllability and normality, our framework enables the synthesis of local supervisors that collectively ensure CO in a decentralized manner.

Furthermore, we extended this modular framework to seamlessly integrate safety specifications. This extension allows for the synthesis of supervisors that not only enforce CO but also guarantee desired safety properties, thereby providing a robust and practical solution for real-world applications.

Acknowledgement

We gratefully acknowledge the suggestions and comments of anonymous referees.

References

[1] W. M. Wonham and P. J. Ramadge, "Modular supervisory control of discrete-event systems," *Mathematics of Control, Signals and Systems*, vol. 1, no. 1, pp. 13–30, 1988.

[2] M. H. De Queiroz and J. E. Cury, "Modular supervisory control of large scale discrete event systems," in *Discrete Event Systems: Analysis and Control*. Springer, 2000, pp. 103–110.

[3] G. Pola, E. De Santis, M. D. Di Benedetto, and D. Pezzuti, "Design of decentralized critical observers for networks of finite state machines: A formal method approach," *Automatica*, vol. 86, pp. 174–182, 2017.

[4] T. Masopust, "Critical observability for automata and Petri nets," *IEEE Transactions on Automatic Control*, vol. 65, no. 1, pp. 341–346, 2020.

[5] X. Cong, M. P. Fanti, A. M. Mangini, and Z. Li, "Critical observability verification and enforcement of labeled Petri nets by using basis markings," *IEEE Transactions on Automatic Control*, vol. 68, no. 12, pp. 8158–8164, 2023.

[6] W. M. Wonham and K. Cai, *Supervisory control of discrete-event systems*. Springer, 2019.

[7] F. Lin and W. M. Wonham, "On observability of discrete-event systems," *Information sciences*, vol. 44, no. 3, pp. 173–198, 1988.

[8] K. Cai, R. Zhang, and W. M. Wonham, "Relative observability of discrete-event systems and its supremal sublanguages," *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 659–670, 2015.

[9] J. Komenda and T. Masopust, "Conditions for hierarchical supervisory control under partial observation," *IFAC-PapersOnLine*, vol. 53, no. 4, pp. 303–308, 2020.

[10] H. Cho and S. I. Marcus, "On supremal languages of classes of sublanguages that arise in supervisor synthesis problems with partial observation," *Mathematics of Control, Signals and Systems*, vol. 2, no. 1, pp. 47–69, 1989.

[11] R. Brandt, V. Garg, R. Kumar, F. Lin, S. Marcus, and W. Wonham, "Formulas for calculating supremal controllable and normal sublanguages," *Systems & Control Letters*, vol. 15, no. 2, pp. 111–117, 1990.

[12] S. Takai and T. Ushio, "Effective computation of an $L_m(G)$-closed, controllable, and observable sublanguage arising in supervisory control," *Systems & Control Letters*, vol. 49, no. 3, pp. 191–200, 2003.

[13] J. Komenda and T. Masopust, "Hierarchical supervisory control under partial observation: Normality," *IEEE Transactions on Automatic Control*, vol. 68, no. 12, pp. 7286–7298, 2023.

[14] ——, "Supervisory control of modular discrete-event systems under partial observation: Normality," *IEEE Transactions on Automatic Control*, vol. 69, no. 6, pp. 3796–3807, 2024.

[15] S. Miao, J. Komenda, and F. Lin, "Hierarchical supervisory control of networked and cyber-attacked discrete-event systems," *Automatica*, vol. 183, p. 112578, 2026.

[16] J. G. Thistle and H. Lamouchi, "Effective control synthesis for partially observed discrete-event systems," *SIAM Journal on Control and Optimization*, vol. 48, no. 3, pp. 1858–1887, 2009.

[17] X. Yin and S. Lafortune, "A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 61, no. 8, pp. 2140–2154, 2016.

[18] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*. Springer, 2021.

[19] G. Jirásková and T. Masopust, "On properties and state complexity of deterministic state-partition automata," in *IFIP International Conference on Theoretical Computer Science*, 2012, pp. 164–178.

[20] L. Feng, "Computationally efficient supervisor design for discrete-event systems," Ph.D. dissertation, University of Toronto, 2007.