2026 American Control Conference (ACC)
May 27-29, 2026. New Orleans, Louisiana, USA

Modular Control for Critical Observability and Weak Opacity in Discrete Event Systems

Shaowen Miao, Jan Komenda, Yiding Ji, and Xiang Yin

***Abstract*— Large-scale systems are ubiquitous in various technological scenarios and often consist of interacting modules, where ensuring security is critical. In discrete event systems, normality ensures that, based on an system observations, one can unambiguously determine whether the corresponding plant behavior is safe. In this paper, we first investigate the computation of an earlier form of normality, known as (L, P)-normality, in modular discrete-event systems. We show that, under the condition that all shared events are observable, the supremal (L, P)-normal sublanguages are computable in a modular fashion. We then apply these results to study the enforcement of two related notions that share similar physical interpretations with (L, P)-normality, namely, critical observability and weak opacity. By integrating our contributions with existing results on controllability and normality, we develop a modular supervisory control approach to enforce these properties. Examples are also provided to illustrate the process.**

***Index Terms*— Discrete event system, modular supervisory control, normality, critical observability, weak opacity.**

I. Introduction

Modern technological systems evolve towards higher complexity, expand in scale, and face increasingly stringent safety-critical requirements. Modular control [1]–[5] facilitates the management of large-scale discrete-event systems (DES) and mitigates the complexity by synthesizing local supervisors. A key advantage of modular design is that only a subset of local supervisors needs to be recomputed when a new control requirement is introduced.

In supervisory control of DES, normality [6] is a fundamental property used to determine whether a given string belongs to a safety specification based on observations. It can be interpreted as the ability to distinguish between pairs of strings, one of which is safe and the other unsafe, by their observations. Since its introduction, normality has been extensively studied in the literature [7], [8]. Unlike observability, normality ensures the existence of a supremal normal sublanguage, making it a more robust and desirable property in supervisory control.

(L, P)-normality, which differs from normality in that the prefix-closure of the specification is not required, was investigated in some early studies [9]–[11]. Similar to normality, (L, P)-normality possesses several useful properties, for instance, it is closed under arbitrary unions. Moreover, (L, P)-normality is also closed under arbitrary intersections, which ensures the existence of an infimal (L, P)-normal superlanguage. This property enables the derivation of Lemma 1 and its applications in Section IV, which are not generally available for normality.

Recently, a property called critical observability (CO) has been widely studied [12], [13]. Originally introduced by De Santis et al. [14] for linear switching systems, CO is a fundamental property of cyber-physical systems that describes the ability to distinguish between legal and illegal states based on observations. In safety-critical applications, critical states represent system conditions that may be unsafe or require special attention. The formulation of CO naturally suggests a connection to (L, P)-normality—both involve the ability to distinguish between two classes of behavior in the system: critical and non-critical.

Pola et al. [12] studied CO in networks of automata and Cong et al. [15] extended the investigation to bounded labeled time Petri nets. More recently, several variants of CO have been widely explored; see [16], [17]. Nevertheless, all the aforementioned works focused primarily on verification. The computational complexity of verifying CO was thoroughly analyzed by Masopust [13], however, the problem of CO enforcement still remains largely unaddressed.

Cong et al. [18] first investigated the enforcement of CO, specifically for bounded labeled Petri nets, using NP-complete integer linear programming. However, their approach suffers from high computational complexity. In addition, their control algorithm operates online, making it unsuitable for computing maximally permissive sublanguages within that framework. In contrast, this paper develops an (L, P)-normality-based framework for the enforcement of CO. The approach exploits the modular structure of the plant and ensures that maximal permissiveness is realizable. Specifically, for a system composed of n subsystems, each with m states, the modular approach achieves a computational complexity of $\mathcal{O}(n \cdot 2^m)$ [7]. The work in [19] proposed a direct approach for enforcing CO in modular systems. In contrast, this work builds upon (L, P)-normality, which usually yields a larger controlled behavior.

This work is supported by National Natural Science Foundation of China grants 62303389 and 62373289; Guangdong Province Basic and Applied Basic Research Funding grant 2024A1515012586; Guangdong Province Scientific Research Platform and Project Scheme grant 2024KTSCX039; Youth Science and Technology Talent Support Program of Guangdong Provincial Association for Science and Technology grant SKXRC2025463, Guangdong Provincial Key Lab of Integrated Communication, Sensing and Computation for Ubiquitous Internet of Things (No.2023B1212010007) and Czech Academy of Sciences RVO 67985840. *(Corresponding author: Y. Ji.)*

Shaowen Miao and Yiding Ji are with the Robotics and Autonomous Systems Thrust, Systems Hub, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, 511453, China. (email: smiao585@connect.hkust-gz.edu.cn, jiyiding@hkust-gz.edu.cn). Jan Komenda is with the Institute of Mathematics, Czech Academy of Sciences, 115 67 Prague, Czechia. (email: komenda@ipm.cz). Xiang Yin is with the School of Automation and Intelligent Sensing, Shanghai Jiao Tong University, Shanghai, 200240, China. (email: yinxiang@sjtu.edu.cn)

In addition to CO, the characteristics of normality naturally lead us to revisit another extensively studied property in recent decades—opacity. Opacity is a security property of DES that concerns whether secret behaviors can be concealed from an external intruder [20]. Its enforcement has been widely explored through various approaches in the literature [21]–[24]. Recently, enforcement methods based on compositional or modular design have garnered increasing attention; see [25]–[27]. Interestingly, we observe that (L, P)-normality is equivalent to the violation of weak opacity (WO) [28]. This implies that the proposed (L, P)-normality-based framework can also be applied to enforce non-WO, particularly from an attacker's perspective. Related studies on the synthesis of attackers can be found in [29], [30].

Our primary contributions are summarized as follows: We first establish the modular computation of the supremal (L, P)-normal sublanguages while preserving maximal permissiveness as a foundation. Leveraging this result, we further extend the modular supervisor synthesis framework to enforce CO and non-WO.

The remainder of the paper is organized as follows. Section II reviews the necessary background on automata theory and supervisory control. Section III characterizes the properties of (L, P)-normality and introduces sufficient conditions for its modular computation. Section IV develops a language-based framework and applies it to the enforcement of CO and WO. Finally, Section V concludes the paper.

II. Preliminaries

A. Automata Theory

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, where the * operation denotes the Kleene star (closure). The concatenation of strings s_1 and s_2 is another string denoted by $s_1 s_2$. 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 prefix-closed.

A *projection* $R\colon \Sigma^* \to \Sigma'^*$, for $\Sigma' \subseteq \Sigma$, is a morphism for concatenation defined by $R(\sigma) = \varepsilon$ for $\sigma \in \Sigma \setminus \Sigma'$ and $R(\sigma) = \sigma$ for $\sigma \in \Sigma'$. The action of R on a string $\sigma_1 \cdots \sigma_n \in \Sigma^*$ is to remove events from $\Sigma \setminus \Sigma'$, and $R(\sigma_1 \cdots \sigma_n) = R(\sigma_1) \cdots R(\sigma_n)$. 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) [6], [11] is the tuple $G = (Q, \Sigma, \delta, q_0, Q_m)$, 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.

The *parallel composition* of languages $L_i \subseteq \Sigma_i^*$ 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 not in Σ_i. In particular, if $G_1, \ldots, G_n$ are automata, then the generated and marked languages of their parallel composition satisfy $L(\|_{i=1}^n G_i) = \|_{i=1}^n L(G_i)$ and $L_m(\|_{i=1}^n G_i) = \|_{i=1}^n L_m(G_i)$.

B. Supervisory Control

A system G over Σ is partially observed and partially controlled [6], if 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 w.r.t. 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 $s\sigma \in L(S/G)$ [6], [11]. 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 system is $L_m(S/G) = L(S/G) \cap K$.

A language $K \subseteq L(G)$ is *controllable* w.r.t. the language $L(G)$ and uncontrollable events Σ_{uc} if $\overline{K}\Sigma_{uc} \cap L(G) \subseteq \overline{K}$. The language K is *observable* w.r.t. $L(G)$, a projection P, and controllable events Σ_c provided that for every string $s \in \overline{K}$ and every controllable event $\sigma \in \Sigma_c$, if $s\sigma \notin \overline{K}$ and $s\sigma \in L(G)$, then $P^{-1}[P(s)]\sigma \cap \overline{K} = \emptyset$. The language K is *normal* w.r.t. $L(G)$ and P if $\overline{K} = P^{-1}[P(\overline{K})] \cap L(G)$. We also let $\mathrm{supCN}(K, L, \Sigma_{uc}, P)$ denote the supremal controllable and normal sublanguage of K w.r.t. L, Σ_{uc}, and P.

III. Modular Synthesis of (L, P)-Normal Supervisors

A. (L, P)-Normality

(L, P)-normality differs from conventional normality in [6] since it does not require the prefix-closure of the specification. Nevertheless, this concept provides a useful foundation for enforcing certain related system properties in Section IV, which are not generally applicable under normality. We begin by recalling the definition of (L, P)-normality.

Definition 1 ((L, P)-Normality): Given a language $L \subseteq \Sigma^*$ and a projection $P\colon \Sigma^* \to \Sigma_o^*$, a language $K \subseteq L$ is said to be (L, P)-*normal* if $K = P^{-1}[P(K)] \cap L$.

Remark 1: As pointed out in [11], (L, P)-normality is symmetric w.r.t. K and $L \setminus K$. Namely, K is (L, P)-normal if and only if $L \setminus K$ is (L, P)-normal.

From the definitions of normality and (L, P)-normality, it is clear that the key distinction lies in the inclusion of the prefix-closure: normality applies to the prefix-closure $\overline{K}$, whereas (L, P)-normality applies to K directly. Consequently, a language $K \subseteq L$ is normal w.r.t. L and P iff $\overline{K}$ is (L, P)-normal. Since normality is defined on the prefix-closure of a language, K is normal iff $\overline{K}$ is normal [6]. However, normality does not imply (L, P)-normality, and vice versa, as illustrated by the following example.

Example 1: Let $\Sigma = \{a, \tau\}$, $\Sigma_o = \{a\}$, $K = \{\tau a, a\tau\}$, and $L(G) = \overline{K}$. Then K is normal, but it is not (L, P)-normal, since $P(a) = P(a\tau)$ for $a\tau \in K$ and $a \in L(G) \setminus K$.

Consider another instance. Let $\Sigma = \{a, \tau\}$, $\Sigma_o = \{a\}$, $L(G) = \overline{\{\tau, a\tau\}}$, and $K = \{a, a\tau\}$. Then $P^{-1}[P(K)] \cap L(G) = K$, and hence K is (L, P)-normal. However, K is not normal, as $P(\varepsilon) = P(\tau)$ for $\varepsilon \in \overline{K}$ and $\tau \in L \setminus \overline{K}$. □

On the other hand, (L, P)-normality does not imply observability either, which is shown by the example below.

Example 2: Let $\Sigma = \{a, b, \tau\}$, $\Sigma_o = \Sigma_c = \{a, b\}$, $L(G) = \overline{\{ab\tau, \tau a, \tau b\}}$, and $K = \{ab, ab\tau, \tau b\} = L(G) \setminus \{\varepsilon, a, \tau, \tau a\}$. Then, K is (L, P)-normal, but it is not observable. This is because for $\varepsilon \in \overline{K}$ and $\tau \in \overline{K}$, we have $\varepsilon a = a \in \overline{K}$ and $\tau a \in L(G)$, with $P(\varepsilon) = P(\tau) = \varepsilon$, yet $\tau a \notin \overline{K}$. □

Since the language in Example 1 is normal—and thus observable—this shows that (L, P)-normality and observability are also incomparable. As shown in [11, Proposition 7], (L, P)-normality is closed under arbitrary unions and intersections. Consequently, the supremal and infimal (L, P)-normal sublanguages always exist. We denote the supremal (L, P)-normal sublanguage of $K \subseteq L$ by $\sup\mathcal{N}(K, L, P)$, and the infimal (L, P)-normal superlanguage of K by $\inf\mathcal{N}(K, L, P)$. As established in [11, Proposition 12], it is given by

$$\sup\mathcal{N}(K, L, P) = K \setminus P^{-1}[P(L \setminus K)]. \tag{1}$$

For the languages that are (L, P)-normal and "closest" to K, namely $\sup\mathcal{N}(K, L, P)$ and $\inf\mathcal{N}(K, L, P)$, we have:

Lemma 1: $\sup\mathcal{N}(L \setminus K, L, P) = L \setminus \inf\mathcal{N}(K, L, P)$.

Proof: "$\supseteq$": Obviously, $L \setminus \inf\mathcal{N}(K, L, P) \subseteq L \setminus K$. Since $\inf\mathcal{N}(K, L, P)$ is (L, P)-normal, its complement $L \setminus \inf\mathcal{N}(K, L, P)$ is also (L, P)-normal by Remark 1.

"$\subseteq$": It suffices to show that $\inf\mathcal{N}(K, L, P) \subseteq L \setminus \sup\mathcal{N}(L \setminus K, L, P)$. Clearly, $K \subseteq L \setminus \sup\mathcal{N}(L \setminus K, L, P)$, so it remains to show that $L \setminus \sup\mathcal{N}(L \setminus K, L, P)$ is (L, P)-normal. Since $\sup\mathcal{N}(L \setminus K, L, P)$ is (L, P)-normal, its complement is also (L, P)-normal by Remark 1. ■

Remark 2: Notice the dual symmetry expressed in Lemma 1. Not only can the specification K and its complement $L \setminus K$ be interchanged, but the infimal (L, P)-normal superlanguage ($\inf\mathcal{N}(\cdot)$) and the supremal (L, P)-normal sublanguage ($\sup\mathcal{N}(\cdot)$) can also be swapped. Specifically, $\inf\mathcal{N}(\cdot)$ can be computed using Eq. (1) for $\sup\mathcal{N}(\cdot)$.

B. Modular Computation

We now extend the previous monolithic computation results in Eq. (1) to the modular setting. A modular system consists of $n \geq 2$ DFAs G_i, each defined over Σ_i, for $i = 1, \ldots, n$. The overall behavior of the modular system is given by the parallel composition $G = \|_{i=1}^n G_i$, where $G_i = (Q_i, \Sigma_i, \delta_i, q_{0,i}, Q_{m,i})$. The set of shared events is defined as $\Sigma_s = \bigcup_{i \neq j}(\Sigma_i \cap \Sigma_j)$. For every event a shared between two distinct automata G_i and G_j, we assume that if a is controllable or observable in G_i then it is also controllable or observable in G_j. We denote the set of locally observable events by $\Sigma_{i,o} = \Sigma_i \cap \Sigma_o$. Similar notation is used for other alphabet intersections. The projections used in this paper are defined as in Fig. 1.

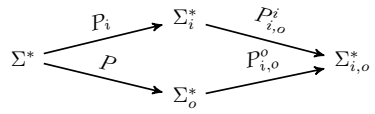

Fig. 1. Projection notations in this letter.

Let $L = L(G)$ and $L_i = L(G_i)$. In modular control, each local plant L_i is associated with a local safety specification $K_i \subseteq L_i$. Specifically, we have $L = \|_{i=1}^n L_i$ and $K = \|_{i=1}^n K_i$. The first step is to identify conditions under which $\sup\mathcal{N}(K, L, P)$ can be computed locally; that is, conditions ensuring $\sup\mathcal{N}(K, L, P) = \|_{i=1}^n \sup\mathcal{N}(K_i, L_i, P_{i,o}^i)$. To this end, the modified observation consistency (MOC) condition plays a crucial role.

Definition 2 (MOC [7]): A prefix-closed language L over Σ is modified observation consistent (MOC) w.r.t. P_i, P, and $P_{i,o}^i$ if, for every string $s \in L$ and $t' \in P_i(L)$ such that $P_{i,o}^i\big(P_i(s)\big) = P_{i,o}^i(t')$, there is a string $s' \in L$ such that $P(s) = P(s')$ and $P_i(s') = t'$.

Then Lemma 2 is presented and leveraged to derived the main result of Theorem 1 of this subsection.

Lemma 2: Consider a modular system $G = \|_{i=1}^n G_i$ with $L = L(G)$. If L is MOC w.r.t. P_i, P, and $P_{i,o}^i$, then $K \subseteq L$ is (L, P)-normal implies $P_i(K)$ is $(P_i(L), P_{i,o}^i)$-normal, for $i = 1, \ldots, n$.

Proof: To prove that $(P_{i,o}^i)^{-1}[P_{i,o}^i(P_i(K))] \cap P_i(L) \subseteq P_i(K)$, let $t' \in (P_{i,o}^i)^{-1}[P_{i,o}^i(P_i(K))] \cap P_i(L)$. Then there exists $s \in K \subseteq L$ such that $t' \in (P_{i,o}^i)^{-1}[P_{i,o}^i(P_i(s))]$, i.e., $P_{i,o}^i(P_i(s)) = P_{i,o}^i(t')$. By MOC, there exists $s' \in L$ such that $P_i(s') = t'$ and $P(s) = P(s')$. Since $s \in K$ and K is (L, P)-normal, we have $s' \in P^{-1}[P(s)] \cap L \subseteq P^{-1}[P(K)] \cap L = K$. Thus, $t' = P_i(s') \in P_i(K)$, which completes the proof. ■

Theorem 1: For languages $K_i \subseteq L_i$ with prefix-closed L_i over Σ_i, for $i = 1, \ldots, n$ and $n \geq 2$, we define the languages $K = \|_{i=1}^n K_i$ and $L = \|_{i=1}^n L_i$. If $P_i(L) = L_i$ and L is MOC w.r.t. P_i, P, and $P_{i,o}^i$, then $\sup\mathcal{N}(K, L, P) = \|_{i=1}^n \sup\mathcal{N}(K_i, L_i, P_{i,o}^i)$.

Proof: To simplify the notation, we write $\sup\mathcal{N}_i$ for $\sup\mathcal{N}(K_i, L_i, P_{i,o}^i)$ and $\sup\mathcal{N}$ for $\sup\mathcal{N}(K, L, P)$.

To prove that $\|_{i=1}^n \sup\mathcal{N}_i$ is contained in $\sup\mathcal{N}$, we show that $\|_{i=1}^n \sup\mathcal{N}_i$ is (L, P)-normal:

$$\begin{aligned}
\|_{i=1}^n \sup\mathcal{N}_i &\subseteq P^{-1}[P(\|_{i=1}^n \sup\mathcal{N}_i)] \cap L \\
&\subseteq P^{-1}[\|_{i=1}^n P_{i,o}^i(\sup\mathcal{N}_i)] \cap L \\
&= \|_{i=1}^n (P_{i,o}^i)^{-1}[P_{i,o}^i(\sup\mathcal{N}_i)] \cap L \\
&= \|_{i=1}^n (P_{i,o}^i)^{-1}[P_{i,o}^i(\sup\mathcal{N}_i)] \ \cap \|_{i=1}^n L_i \\
&= \|_{i=1}^n [(P_{i,o}^i)^{-1}[P_{i,o}^i(\sup\mathcal{N}_i)] \cap L_i] \\
&= \|_{i=1}^n \sup\mathcal{N}_i\,.
\end{aligned}$$

Now, assume that $P_i(L) = L_i$, for $i = 1, \ldots, n$, and that L is MOC. Then, Lemma 2 implies that $P_i(\sup\mathcal{N})$ is $(L_i, P_{i,o}^i)$-normal. Consequently, we have $P_i(\sup\mathcal{N}) \subseteq \sup\mathcal{N}_i$ for $i = 1, \ldots, n$, and therefore $\sup\mathcal{N} \subseteq \|_{i=1}^n \sup\mathcal{N}_i$. ■

Checking the MOC condition is PSPACE-hard, and its decidability remains an open question [31]. To address this issue, [7, Lemma 5] shows that MOC holds if all shared events are observable, i.e., $\Sigma_s \subseteq \Sigma_o$. As a direct consequence, we can derive the following result.

Theorem 2: Let $G = \|_{i=1}^{n} G_i$ be a modular plant. Let $L_i = L(G_i)$ over Σ_i for $i = 1, \ldots, n$ and $n \geq 2$, and let $L = L(G)$. For $K = \|_{i=1}^{n} K_i$ with $K_i \subseteq L_i$, if $P_i(L) = L_i$ and all shared events are observable ($\Sigma_s \subseteq \Sigma_o$), then $\sup\mathcal{N}(K, L, P) = \|_{i=1}^{n} \sup\mathcal{N}(K_i, L_i, P_{i,o}^{i})$. ■

IV. Applications to Enforcement of Critical Observability and Weak Opacity

A. Critical Observability

We begin by characterizing the relationship between (L, P)-normality and CO, and then illustrating how to apply the results presented in Section III to the enforcement of CO. To this end, we first recall the definition of CO.

Definition 3 (Critical Observability [12]): A DFA $G = (Q, \Sigma, \delta, q_0)$ is *critically-observable* (CO) w.r.t. a projection $P\colon \Sigma^* \to \Sigma_o^*$ and the 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 $\delta(q_0, P^{-1}[P(s)]) \subseteq Q \setminus Q_c$, where $\delta(q_0, P^{-1}[P(s)]) = \bigcup_{t \in P^{-1}[P(s)]} \delta(q_0, t)$.

Critical observability requires that, given an observation of the system, one can unambiguously determine whether the system is in a critical state. This concept is closely related to (L, P)-normality. In particular, (L, P)-normality ensures that for every two generated strings, one safe and the other unsafe, the projection should distinguish one from the other.

For a DFA $G = (Q, \Sigma, \delta, q_0)$ with a set of critical states $Q_c \subseteq Q$. We define

$$K_{co} = \{w \in L(G) \mid \delta(q_0, w) \in Q_c\} \tag{2}$$

as the *critical language* of G. We now relate the notions of CO and (L, P)-normality.

Proposition 1: Let $G = (Q, \Sigma, \delta, q_0)$ be a DFA, and $Q_c \subseteq Q$ be a set of critical states. Then the critical language K_{co} is (L, P)-normal if and only if G is CO w.r.t. P and Q_c.

Proof: The language K_{co} is (L, P)-normal if and only if $P(s) \neq P(s')$ for every $s \in L(G) \setminus K_{co}$ and $s' \in K_{co}$, which is equivalent to $\delta(q_0, s) \in Q \setminus Q_c$ and $\delta(q_0, s') \in Q_c$, defining CO of G. Note that Definition 3 can be rewritten as $\forall s, s' \in L(G), \delta(q_0, s) \in Q \setminus Q_c \wedge \delta(q_0, s') \in Q_c \Rightarrow P(s) \neq P(s')$, which coincides with Eq. (2). ■

Corollary 1: Let $G = (Q, \Sigma, \delta, q_0)$ be a DFA with a set of critical states $Q_c \subseteq Q$. If the critical language K_{co} is prefix-closed, then it is normal w.r.t. $L(G)$ and P if and only if G is CO w.r.t. P and Q_c. ■

Proposition 1 and Corollary 1 does not hold for nondeterministic finite automata, as a single string may lead to both critical and non-critical states. We now consider the case where *it is possible to reclassify a state—changing it from critical to non-critical or vice versa*. In such scenarios, the methods developed for enforcing (L, P)-normality can be leveraged to enforce CO. Specifically, we proceed as follows:

1) We keep the original plant G unchanged.
2) We compute a language $M \subseteq L(G)$, which—according to the results in Section III—can be chosen as either $\sup\mathcal{N}(K_{co}, L, P)$ or $\inf\mathcal{N}(K_{co}, L, P)$.
3) Based on M, we can define a refined set of critical states $Q_c' \subseteq Q_c$ as:

$$Q_c' = \{q \in Q_c \mid \exists w \in M\colon q = \delta(q_0, w)\}. \tag{3}$$

It follows Proposition 1 that G is CO w.r.t. P and Q_c'.

Lemma 3: Let $G = (Q, \Sigma, \delta, q_0)$ be a DFA, and let $Q_c \subseteq Q$ be a set of critical states. Let M be a language of set $\{\sup\mathcal{N}(K_{co}, L, P), \inf\mathcal{N}(K_{co}, L, P)\}$. Define Q_c' as Eq.(3). Then G is CO w.r.t. P and Q_c'.

Proof: The proof follows directly from Proposition 1 by replacing K_{co} with M and Q_c with Q_c'. ■

Consider two DFAs $G_i = (Q_i, \Sigma, \delta_i, q_{0,i}, Q_{m,i})$ for $i = 1, 2$, where $\overline{L_m(G_1)} = L(G_1) \subseteq L(G_2)$. We say that G_1 is a *sub-automaton* of G_2, denoted by $G_1 \sqsubseteq G_2$, if $\delta_1(q_{0,1}, s) = \delta_2(q_{0,2}, s)$ for every $s \in L(G_1)$.

Some operations in the remainder of this paper may result in the critical state set Q_c containing states that do not belong to the plant state set Q. For simplicity, we directly remove such states from Q_c.

The following lemma states that CO is preserved under sub-automata, which follows directly from Definition 3.

Lemma 4: Let $G = (Q, \Sigma, \delta, q_0)$ be a DFA, let $Q_c \subseteq Q$ be a set of critical states, and let $P\colon \Sigma^* \to \Sigma_o^*$ be a projection. If G is CO w.r.t. P and Q_c, then any sub-automaton $G' \sqsubseteq G$ is also CO w.r.t. P and Q_c. ■

Lemmas 3 and 4 together solve CO enforcement via the computation of (L, P)-normality, and we proceed as:

1) We first compute M as in Lemma 3 to derive a refined set of critical states Q_c'.
2) Given a safety specification $K \subseteq L(G)$, we then compute $\mathrm{supCN}(K, L, \Sigma_{uc}, P)$ using classical methods (e.g., [6], [32]). By Lemma 4, $\mathrm{supCN}(K, L, \Sigma_{uc}, P)$ is CO w.r.t. P and the refined set Q_c'.

We illustrate the above described approach through the following example.

Example 3: Consider the system G depicted in Fig. 2, where the critical language is given by $K_{co} = \{aa\}$, as defined in Eq. (2). Applying Eq. (1), we compute

$$\sup\mathcal{N}(K_{co}, L, P) = K_{co} \setminus P^{-1}[P(L \setminus K_{co})] = \emptyset,$$

which implies that we cannot refine the critical state set Q_c' via $\sup\mathcal{N}(K_{co}, L, P)$ by *reducing critical states*. However, by Lemma 1, we can alternatively compute

$$\inf\mathcal{N}(K_{co}, L, P) = L \setminus \sup\mathcal{N}(L \setminus K_{co}, L, P) = \{aa, a\tau a\}.$$

Firstly, using Eq. (1), we obtain $\sup\mathcal{N}(L \setminus K_{co}, L, P) = (L \setminus K_{co}) \setminus P^{-1}[P(K_{co})] = \{\varepsilon, a, a\tau, a\tau a, ab, ab\tau\} \setminus \{a\tau a\} = \{\varepsilon, a, a\tau, ab, ab\tau\}$. Secondly, the refined set corresponding to $\inf\mathcal{N}(K_{co}, L, P)$ becomes $Q_c' = \{4, 5\}$, corresponding to $\inf\mathcal{N}(K_{co}, L, P)$ via Eq.(3). This implies that *the previously non-critical state* 4 *is now reclassified as critical*. As a result, G becomes CO w.r.t. P and Q_c', in accordance with Lemma 3.

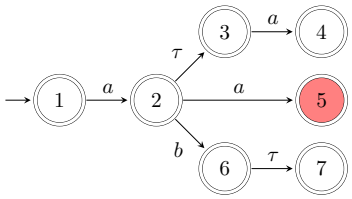

Fig. 2. DFA G with $\Sigma_o = \Sigma_c = \{a, b\}$, $\Sigma_{uo} = \Sigma_{uc} = \{\tau\}$, and $Q_c = \{5\}$.

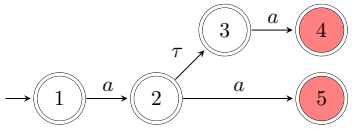

Fig. 3. $\mathrm{supCN}(K, L, \Sigma_{uc}, P)$.

Moreover, it is straightforward to verify that any sub-automaton $G' \sqsubseteq G$ is CO w.r.t. P and Q'_c, as stated in Lemma 4. For instance, $\mathrm{supCN}(K, L, \Sigma_{uc}, P)$, depicted in Fig. 3, is CO w.r.t. P and $Q'_c = \{4, 5\}$, where $K = \{\varepsilon, a, a\tau, a\tau a, aa, ab\} \subseteq L(G)$. In this case, the string ab is excluded due to controllability. □

We then consider extending the results to the modular case, where the modular plant is given by $G = \|_{i=1}^{n} G_i$ with each local component $G_i = (Q_i, \Sigma_i, \delta_i, q_{0,i}, Q_{m,i})$ modeled as a DFA having $L(G_i) = L_i$. Note that the definition of system properties, such as critical observability in Definition 3, remains the same for both monolithic and modular systems. Let $Q_{c,i} \subseteq Q_i$ denote the local sets of critical states, which defines the *local critical language* as $K_{co,i} = \{w \in L_i \mid \delta_i(q_{0,i}, w) \in Q_{c,i}\}$. Similar to the monolithic case, we define the local refined set $Q'_{c,i} = \{q \in Q_{c,i} \mid \exists w \in M_i \colon q = \delta_i(q_{0,i}, w)\}$ using

$$M_i \in \{\sup \mathcal{N}(K_{co,i}, L_i, P^i_{i,o}), \inf \mathcal{N}(K_{co,i}, L_i, P^i_{i,o})\}. \quad (4)$$

According to [12, Proposition 3.1], CO is closed under parallel composition. Specifically, if each G_i is CO w.r.t. $\mathcal{Q}_{c,i}$, then their parallel composition $G = \|_{i=1}^{n} G_i$ is CO w.r.t. the global set of critical states

$$\mathcal{Q}_c = \{(q_1, \ldots, q_n) \in \prod_{i=1}^{n} Q_i \mid \exists i \in \{1, \ldots, n\} \colon q_i \in \mathcal{Q}_{c,i}\}. \quad (5)$$

Then, given local safety specifications $K_i \subseteq L_i \subseteq \Sigma_i^*$, the computation procedure for modular supervisors enforcing CO is summarized in Algorithm 1. In particular, the output implies that supervisors enforcing CO can be synthesized modularly, whose correctness is guaranteed by [12, Proposition 3.1] together with Lemmas 3 and 4. Furthermore, to ensure maximal permissiveness, that is, to guarantee that $K = \|_{i=1}^{n} K_i$ and $\mathrm{supCN}(K, L, \Sigma_{uc}, P) = \|_{i=1}^{n} \mathrm{supCN}(K_i, L_i, \Sigma_{i,uc}, P^i_{i,o})$, the results follow from Theorem 1/2 and [7, Theorem 16], respectively, when the corresponding conditions are satisfied.

Example 4: Consider again the system G shown in Fig. 2, where we now append the subscript 1 to each of its associated parameters. Next, consider another system G_2, depicted in Fig. 4 (left), with $Q_{c,2} = \{1\}$ and $K_2 = \{\varepsilon, a\}$. Following the same procedure outlined in Example 3, we compute $\mathrm{supCN}(K_2, L_2, \Sigma_{2,uc}, P^2_{2,o})$, which is illustrated in Fig. 4 (right) and is CO w.r.t. $P^2_{2,o}$ and $Q_{c,2}$. It is then verified that:

$$\begin{aligned} &\mathrm{supCN}(K_1 \| K_2, L_1 \| L_2, \Sigma_{1,uc} \cup \Sigma_{2,uc}, P) \\ &= \mathrm{supCN}(K_1, L_1, \Sigma_{1,uc}, P^1_{1,o}) \| \mathrm{supCN}(K_2, L_2, \Sigma_{2,uc}, P^2_{2,o}) \\ &= \{\varepsilon, a, a\tau, a\tau a, aa\} \| \{\varepsilon, a\} = \{\varepsilon, a, a\tau\}, \end{aligned}$$

which is CO w.r.t. P and $Q_c = \{(1, 1)\}$. □

Algorithm 1 Modular CO supervisor computation

Input: Local safety specifications $K_i \subseteq L_i \subseteq \Sigma_i^*$ and local critical state sets $Q_{c,i} \subseteq Q_i$ for $i = 1, \ldots, n$.
Output: Modularly enforced CO subplant.

```
if K_i is controllable w.r.t. L_i and Σ_{i,uc}, and normal w.r.t. L_i and P^i_{i,o} then
    K_i ← K_i;
else
    K_i ← supCN(K_i, L_i, Σ_{i,uc}, P^i_{i,o});
if K_i is CO w.r.t. P^i_{i,o} and Q_{c,i} then
    Q_{c,i} ← Q_{c,i};
else
    Compute M_i as in Eq. (4) and define Q'_{c,i} accordingly.
    Q_{c,i} ← Q'_{c,i}.
return ||_{i=1}^{n} K_i, which is CO w.r.t. P and Q_c as in Eq. (5).
```

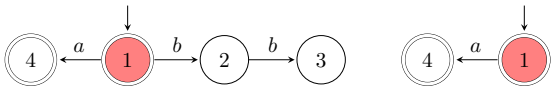

Fig. 4. DFA G_2 with $\Sigma_2 = \Sigma_{o,2} = \Sigma_{c,2} = \{a, b\}$ and $Q_{c,2} = \{1\}$, where $K_2 = \{\varepsilon, a\}$; $\mathrm{supCN}(K_2, L_2, \Sigma_{2,uc}, P^2_{2,o})$.

B. Weak Opacity

Given the concept of weak opacity from [28], Lemma 5 implies that the results obtained in this paper are applicable to enforce non-WO from an *attacker* perspective.

Definition 4 (Weak Opacity): A language $L_1 \subseteq \Sigma^*$ is *weakly opaque* w.r.t. a language $L_2 \subseteq \Sigma^*$ and a projection $P \colon \Sigma^* \to \Sigma_o^*$ if $L_1 \cap P^{-1}[P(L_2)] \neq \emptyset$.

Lemma 5: A language $K \subseteq L$ is (L, P)-normal if and only if K is *not* weakly opaque w.r.t. $L \setminus K$ and P.

Proof: By definition, the language K is (L, P)-normal if and only if $P(s) \neq P(s')$ whenever $s \in L \setminus K$ and $s' \in K$. In other words, $s' \notin P^{-1}[P(s)]$, which is equivalent to the fact that K is not weakly opaque w.r.t. $L \setminus K$ and P. ■

Lemma 5 states that, from an attacker's perspective, given a *secret language* K that is not (L, P)-normal, we can apply the results from Section III to compute a (L, P)-normal language $M \in \{\sup \mathcal{N}(K, L, P), \inf \mathcal{N}(K, L, P)\}$. This implies that the resulting secret language M is not weakly opaque w.r.t. $L \setminus M$ and P. We then show that WO is preserved under parallel composition, thus the previously established modular enforcement results remain applicable to WO.

Lemma 6: If languages $K_i \subseteq \Sigma_i^*$ are weakly opaque w.r.t. languages $L_i \subseteq \Sigma_i^*$ and projections $P_{i,o}^i$, respectively, then $\|_{i=1}^n K_i \subseteq \Sigma^*$ is weakly opaque w.r.t. languages $\|_{i=1}^n L_i \subseteq \Sigma^*$ and projection P.

Proof: Let $s \in \|_{i=1}^n K_i$ and $t \in \|_{i=1}^n L_i$. Then we have $P_i(s) \in P(\|_{i=1}^n K_i) \subseteq K_i$ and $P_i(t) \in P(\|_{i=1}^n L_i) \subseteq L_i$. By definition, we have $K_i \cap (P_{i,o}^i)^{-1}[P_{i,o}^i(L_i)] \neq \emptyset$ for $i = 1, \ldots, n$. Thus, we have $P_{i,o}^i(P_i(s)) = P_{i,o}^i(P_i(t))$. Then we have $P_{i,o}^o(P(s)) = P_{i,o}^o(P(t))$ for $i = 1, \ldots, n$, by Fig. 1.

Assume for contradiction that $P(s) \neq P(t)$. Let $\alpha \in \Sigma_o$ be the symbol at which they differ. Since $\alpha \in \Sigma_o = \bigcup_{i=1}^n \Sigma_{i,o}$, there exists some k such that $\alpha \in \Sigma_{k,o}$. That is, this symbol is retained by $P_{k,o}^o$. Therefore, when applying $P_{k,o}^o$, this difference must be revealed: $P_{k,o}^o(P(s)) \neq P_{k,o}^o(P(t))$, contradicting the earlier conclusion. ∎

Lemma 6 implies that, for a modular system $G = \|_{i=1}^n G_i$ with $L(G_i) = L_i$, once each local module realizes $M_i \in \{\sup \mathcal{N}(K_i, L_i, P_{i,o}^i), \inf \mathcal{N}(K_i, L_i, P_{i,o}^i)\}$, the composed global behavior $\|_{i=1}^n M_i$ is weakly opaque w.r.t. the languages $\|_{i=1}^n (L_i - M_i)$. We argue that, although the above methods may directly alter the system's secrets, such scenarios can still occur in practice. Consider a situation where a hacker knows certain system's secrets but does not want to reveal that he knows the secret. Instead of disclosing the secrets by himself, the hacker may attempt to launch attacks that cause the system to become non-weakly opaque to the public.

V. Conclusion

This paper addressed the control synthesis problem of security-related properties in modular discrete event systems. We first studied the modular computation of (L, P)-normality. Then we extended the modular approach to the enforcement of critical observability and non-weak opacity, based on modular computation of supremal/infimal (L, P)-normal sub/superlanguages. By integrating our findings with existing results on controllability and normality, we demonstrated that supervisors ensuring these properties can also be synthesized in a modular fashion, thus enhancing scalability and practicality for complex dynamic systems.

References

[1] K. Schmidt, T. Moor, and S. Perk, "Nonblocking hierarchical control of decentralized discrete event systems," *IEEE Transactions on Automatic Control*, vol. 53, no. 10, pp. 2252–2265, 2008.

[2] M. Rosa, J. E. Cury, and F. L. Baldissera, "A modular synthesis approach for the coordination of multi-agent systems: the multiple team case," *Dis. Event Dyn. Sys.*, vol. 34, no. 1, pp. 163–198, 2024.

[3] J. Li and S. Takai, "Maximally permissive modular similarity control of composite nondeterministic discrete event systems," *IEEE Control Systems Letters*, vol. 6, pp. 2305–2310, 2022.

[4] R. Malik, S. Mohajerani, and M. Fabian, "A survey on compositional algorithms for verification and synthesis in supervisory control," *Discrete Event Dynamic Systems*, vol. 33, no. 3, pp. 279–340, 2023.

[5] J. Yang, K. Tan, L. Feng, and Z. Li, "A model-based deep reinforcement learning approach to the nonblocking coordination of modular supervisors of discrete event systems," *Information Sciences*, vol. 630, pp. 305–321, 2023.

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

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

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

[9] 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.

[10] 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.

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

[12] 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.

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

[14] E. De Santis, M. D. Di Benedetto, S. Di Gennaro, A. D'Innocenzo, and G. Pola, "Critical observability of a class of hybrid systems and application to air traffic management," in *Stochastic Hybrid Systems: Theory and Safety Critical Applications*, pp. 141–170, Springer, 2006.

[15] X. Cong, M. P. Fanti, A. M. Mangini, and Z. Li, "Critical observability of labeled time petri net systems," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 3, pp. 2063–2074, 2022.

[16] Y. Tong and Z. Ma, "Verification of k-step and definite critical observability in discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 68, no. 7, pp. 4305–4312, 2022.

[17] Y. Zhang, C. N. Hadjicostis, and Z. Li, "Generalized critical observability of labeled petri nets under nondeterministic observations," *IEEE Trans. on Automation Sci. and Eng.*, vol. 22, pp. 14633 – 14645, 2025.

[18] 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.

[19] S. Miao, J. Komenda, T. Masopust, and A. Lai, "Enforcement of critical observability in modular discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 71, no. 3, pp. 2114–2119, 2026.

[20] S. Miao, A. Lai, and J. Komenda, "Always guarding you: Strong initial-and-final-state opacity of discrete-event systems," *Automatica*, vol. 173, p. 112085, 2025.

[21] X. Yin and S. Lafortune, "A general approach for optimizing dynamic sensor activation for discrete event systems," *Automatica*, vol. 105, pp. 376–383, 2019.

[22] Y. Luo, S. Miao, W. Lan, and A. Lai, "Supervisory control for current-state e-opacity enforcement under encrypted observations," in *37th Chinese Control and Decision Conference*, pp. 5601–5607, 2025.

[23] Y. Ji, X. Yin, and S. Lafortune, "Enforcing opacity by insertion functions under multiple energy constraints," *Automatica*, vol. 108, p. 108476, 2019.

[24] Y. Ji, X. Yin, and S. Lafortune, "Opacity enforcement using nondeterministic publicly known edit functions," *IEEE Transactions on Automatic Control*, vol. 64, no. 10, pp. 4369–4376, 2019.

[25] S. Mohajerani, Y. Ji, and S. Lafortune, "Compositional and abstraction-based approach for synthesis of edit functions for opacity enforcement," *IEEE Trans. on Autom. Control*, vol. 65, no. 8, pp. 3349–3364, 2019.

[26] G. Zinck, L. Ricker, H. Marchand, and L. Hélouët, "Enforcing opacity in modular systems," in *21st IFAC World Cong.*, pp. 2157–2164, 2020.

[27] N. E. Souid, K. Klai, C. A. Abid, and S. B. Ahmed, "Enforcing the opacity of modular discrete event systems using supervisory control," in *9th International Conference on Control, Decision and Information Technologies*, pp. 1990–1995, 2023.

[28] F. Lin, "Opacity of discrete event systems and its applications," *Automatica*, vol. 47, no. 3, pp. 496–503, 2011.

[29] R. Meira-Góes, E. Kang, R. H. Kwong, and S. Lafortune, "Synthesis of sensor deception attacks at the supervisory layer of cyber–physical systems," *Automatica*, vol. 121, p. 109172, 2020.

[30] R. Tai, L. Lin, and R. Su, "Synthesis of optimal covert sensor–actuator attackers for discrete-event systems," *Automatica*, vol. 151, p. 110910, 2023.

[31] 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.

[32] T. Moor, C. Baier, T.-S. Yoo, F. Lin, and S. Lafortune, "On the computation of supremal sublanguages relevant to supervisory control," in *11th IFAC Workshop on Discrete Event Systems*, pp. 175–180, 2012.