Verifying Nonblockingness and Deadlock-Freeness in Isomorphic Module Systems

Adéla Laštovičková and Tomáš Masopust

***Abstract*— Isomorphic module systems constitute a distributed discrete-event modeling paradigm for large-scale multi-agent and networked architectures. They are composed of modules, instantiated from a given template, coordinated through global events. A key challenge is that even if the template is nonblocking or deadlock-free, the parallel composition of the modules may fail to satisfy these properties for some number of modules. The fundamental problem is to determine the maximum number of modules that can be safely composed while preserving nonblockingness or deadlock-freeness. We establish the computational complexity of the associated decision problems, proving that both problems are PSPACE-complete, and show that the minimal number m of modules that causes the system to become blocking or deadlocked can be found in time $O(m2^m)$, which is bounded by $O(n2^n)$, where n is the number of states of the template, improving thus the known complexity of these problems.**

I. Introduction

Discrete-event systems (DES) provide a theoretical framework for modeling, verification, and supervisory control synthesis of complex engineering processes, including communication networks and large-scale multi-agent systems [1].

We investigate a class of distributed systems, referred to as *isomorphic module systems*, consisting of a large number of system modules that exhibit isomorphic local behavior. In our framework, we are given a template automaton G, represented as an accessible deterministic finite automaton with the event alphabet partitioned into global and private events. Individual modules are instantiated from the template by localizing private events, while global (shared) events are used for coordination among modules via the standard parallel composition operator.

Such an isomorphic modular structure naturally arises in various practical contexts, including symmetric sensor networks, replicated communication protocols, and coordinated groups of autonomous agents. For further discussion of motivation and applications, we refer the reader to the literature [2], [3], [4] and the references therein.

The theoretical foundations for analyzing isomorphic module systems were established by Rohloff and Lafortune [2]. Recognizing the limitations of standard parallel composition, the authors exploited structural symmetry to develop a scalable methodology addressing both verification and control problems. Specifically, they introduced the construction of a *quotient automaton*, a structure derived from the global system whose state-space size is independent of the number of similar modules. This approach enabled the verification of key properties such as nonblockingness and reachability with significantly reduced complexity. Moreover, they addressed the modeling challenge by presenting a polynomial-time decomposition algorithm that reconstructs component subsystems from a global monolithic model, thereby overcoming the typically exponential complexity of decomposition. They further established necessary and sufficient conditions for the existence of similar local supervisors that collectively enforce specified local and global control objectives—a fundamental problem in decentralized control.

However, the precise limits of parameterized systems remain an open challenge. In particular, even if the template G is nonblocking (deadlock-free), the parallel composition of multiple instances of G may exhibit blocking (deadlocked) behavior. The fundamental question, therefore, is how many modules can be composed while ensuring that their parallel composition remains nonblocking (deadlock-free).

While the quotient automaton proposed by Rohloff and Lafortune can be used to determine the maximum number of nonblocking (deadlock-free) n-state modules in $O(n2^n \log n)$ time—by performing a binary search over the ordered sequence $1, \ldots, n$ combined with the quotient automaton construction in $O(n2^n)$ time—the overall complexity of this problem has not been fully investigated.

Wang et al. [3] also addressed this problem and proposed two algorithms: one to identify the minimum number of blocking modules, running in $O(2^{4n})$ time, where n is the number of states in the template, and another to find the minimum number of modules that lead to a deadlock, running in $O(2^{2n})$ time (although Theorem 4 of [3] claims a complexity of $O(n2^n)$, the proof only shows it to be $O(2^{2n})$).

In this paper, we address the problem of determining the precise limits of the safe composition for isomorphic module systems by focusing on two key properties: nonblockingness and deadlock-freeness. We establish the exact complexity of the associated decision problems, proving that deciding whether the parallel composition of a given number of isomorphic modules is nonblocking or deadlock-free is PSPACE-complete. Consequently, the corresponding optimization problems determining the minimum number of modules that cause blocking, $\min_b(G)$, or deadlock, $\min_d(G)$, are PSPACE-hard. Finally, we show that by utilizing the quotient automaton of Rohloff and Lafortune [2], these minimal numbers, m, can be computed in time $O(m2^m)$, which is bounded by $O(n2^n)$, where n is the number of states in the template, improving thus the complexity of these problems.

A. Laštovičková and T. Masopust are with the Faculty of Science, Palacky University Olomouc, Czechia. tomas.masopust@upol.cz, adela.lastovickova01@upol.cz.

The research of A. Laštovičková was supported by the Palacky University under the grants IGA PrF 2025 018 and IGA PrF 2026 016.

II. Preliminaries

An *alphabet* Σ is a finite nonempty set of *events*. A *string* over Σ is a finite sequence of events. Let Σ^* denote the set of all strings over Σ, where the *empty string* is denoted by ε. A language over Σ is a subset of Σ^*. For a language L over Σ, we denote by $\overline{L}$ the prefix closure of L, i.e., $\overline{L} = \{u \in \Sigma^* \mid \text{there is } v \in \Sigma^* \text{ such that } uv \in L\}$. The language L is *prefix-closed* if $L = \overline{L}$.

A *nondeterministic finite automaton* (NFA) over Σ is a structure $\mathcal{A} = (Q, \Sigma, \delta, I, F)$, where Q is a finite nonempty set of states, $I \subseteq Q$ is a nonempty set of initial states, $F \subseteq Q$ is a set of accepting states, and $\delta\colon Q \times \Sigma \to 2^Q$ is a transition function that can be extended to the domain $2^Q \times \Sigma^*$ by induction. Equivalently, the transition function is a relation $\delta \subseteq Q \times \Sigma \times Q$. The *language generated* by $\mathcal{A}$ is the set $L(\mathcal{A}) = \{w \in \Sigma^* \mid \delta(I, w) \neq \emptyset\}$ and the *language accepted* by $\mathcal{A}$ is the set $L_m(\mathcal{A}) = \{w \in \Sigma^* \mid \delta(I, w) \cap F \neq \emptyset\}$. Obviously, $L_m(\mathcal{A}) \subseteq L(\mathcal{A})$ and $L(\mathcal{A})$ is prefix-closed.

An NFA $\mathcal{A}$ is *deterministic* (DFA) if it has a unique initial state, i.e., $|I| = 1$, and no nondeterministic transition, i.e., $|\delta(q, a)| \leq 1$ for every state $q \in Q$ and event $a \in \Sigma$. For DFAs, we identify singletons with their elements. Specifically, we simply write $\delta(q, a) = p$ instead of $\delta(q, a) = \{p\}$.

An NFA is *accessible* if all its states are reachable from an initial state, it is *nonblocking* if $\overline{L_m(G)} = L(G)$, and it is *deadlock free* if, for every string $w \in L(G)$, there is an event a such that $wa \in L(G)$.

In this paper, we assume that all modules are DFAs. Notice that, by definition, a state of a DFA is blocked if no accepting state can be reached from it, and that it is deadlocked if there is no transition defined in it. A DFA is nonblocking (deadlock-free) if no its states are blocked (deadlocked).

For alphabets Σ and Γ, a map $P\colon \Sigma^* \to \Gamma^*$ is a *morphism* for concatenation if $P(xy) = P(x)P(y)$ for every $x, y \in \Sigma^*$. An *isomorphism* is a bijective morphism. The *inverse image* of P, denoted by P^{-1}, is defined as the language $P^{-1}(s) = \{w \in \Sigma^* \mid P(w) = s\}$. If $\Gamma \subseteq \Sigma$ and the morphism P satisfies $P(a) = \varepsilon$ for $a \in \Sigma \setminus \Gamma$, and $P(a) = a$ for $a \in \Gamma$, then P is a *projection*. The action of the projection P on a string $w \in \Sigma^*$ is to erase all events from w that do not belong to Γ. The definitions can be readily extended to languages.

Let L_i be a language over Σ_i, for $i = 1, \ldots, k$. The *parallel composition* of languages $L_1, \ldots, L_k$ is defined by $\|_{i=1}^k L_i = \cap_{i=1}^k P_i^{-1}(L_i)$, where P_i is a projection from $(\cup_{i=1}^k \Sigma_i)^*$ to Σ_i^*. For automata, the parallel composition is defined as follows. For $i = 1, 2$, let $\mathcal{A}_i = (Q_i, \Sigma_i, \delta_i, I_i, F_i)$ be automata. The *parallel composition* of $\mathcal{A}_1$ and $\mathcal{A}_2$ is the accessible part of the automaton $(Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \delta, I_1 \times I_2, F_1 \times F_2)$, where for $(q_1, q_2) \in Q_1 \times Q_2$ and $a \in \Sigma_1 \cup \Sigma_2$,

$$\delta((q_1, q_2), a) = \begin{cases} \delta_1(q_1, a) \times \delta_2(q_2, a) & \text{if } a \in \Sigma_1 \cap \Sigma_2, \\ \delta_1(q_1, a) \times \{q_2\} & \text{if } a \in \Sigma_1 \setminus \Sigma_2, \\ \{q_1\} \times \delta_2(q_2, a) & \text{if } a \in \Sigma_2 \setminus \Sigma_1. \end{cases}$$

The parallel composition is commutative and associative, and hence $\mathcal{A}_1 \parallel \mathcal{A}_2 \parallel \cdots \parallel \mathcal{A}_k = (\cdots(\mathcal{A}_1 \parallel \mathcal{A}_2) \parallel \cdots \parallel \mathcal{A}_k)$ for $k > 2$. The composition of DFAs is a DFA [5].

A *decision problem* is a yes-no question formulated for instances of the problem. A problem is *decidable* if there is an algorithm that for every instance decides whether it is positive or negative. Decidable problems are classified into classes based on the time or space required by an algorithm to solve them. In particular, the class PSPACE consists of problems solvable in polynomial space. A problem is PSPACE-complete if it belongs to PSPACE (*membership*) and every problem from PSPACE can be reduced to it in polynomial time (*hardness*). It is believed that no polynomial-time algorithms exist for PSPACE-complete problems.

A. Isomorphic Module Systems

Let Σ be an alphabet partitioned into the sets of global events Σ_g and private events Σ_p. A *template* G over Σ is an accessible DFA over Σ.

The template is instantiated into modules via an isomorphism. We index the generated modules by consecutive non-negative integers. Given an integer $i \geq 1$, we define the isomorphism $\kappa_i\colon \Sigma^* \to (\Sigma_g \cup (\Sigma_p \times \{i\}))^*$ as

$$\kappa_i(a) = \begin{cases} a & \text{if } a \in \Sigma_g \\ (a, i) & \text{if } a \in \Sigma_p\,, \end{cases}$$

that is, we make a private copy of all private events of Σ while we keep the global events shared by all modules. We naturally extend the definition from events to sets of events by defining $\kappa_i(\Sigma) = \{\kappa_i(a) \mid a \in \Sigma\}$.

For an integer $i \geq 1$, we denote the ith *instance* of the template $G = (Q, \Sigma, \delta, I, F)$ by

$$G_i = (Q, \kappa_i(\Sigma), \delta_i, I, F)$$

where $\delta_i = \{(p, \kappa_i(a), q) \mid (p, a, q) \in \delta\}$. Since the template G is a DFA, every instance G_i is also a DFA.

The *isomorphic module system* is the set of $k \geq 2$ instances $\{G_1, \ldots, G_k\}$ of a template G with the global behavior defined by $G^k = G_1 \parallel G_2 \parallel \cdots \parallel G_k$.

B. Quotient Automaton

In this section, we briefly recall the construction of the quotient automaton of Rohloff and Lafortune [2]. Given a template $G = (Q, \Sigma, \delta, q_0, F)$, the state set of the parallel composition $G^k = G_1 \parallel \cdots \parallel G_k$ of k modules is Q^k, and we denote the transition function of the DFA G^k by δ^k.

We further define the function $\mathrm{Comp}_k\colon Q^k \to 2^Q$ as follows. For every k-tuple $(q_1, \ldots, q_k) \in Q^k$,

$$\mathrm{Comp}_k(q_1, \ldots, q_k) = \{q_1, \ldots, q_k\}\,.$$

For instance, $\mathrm{Comp}_3(1, 2, 2) = \mathrm{Comp}_3(1, 1, 2) = \{1, 2\}$.

Using the function Comp_k, we may define the *quotient automaton* $\tilde{G}^k$ of Rohloff and Lafortune [2] as the accessible part of the automaton $(2^Q, \Sigma, \tilde{\delta}, \tilde{q}_0, 2^F)$, where

- the initial state $\tilde{q}_0 = \{q_0\}$ contains the initial state of the template, and
- for every event $a_i \in \kappa_i(\Sigma)$, $i \in \{1, \ldots, k\}$, we define the transition $(\tilde{p}, \kappa_i^{-1}(a_i), \tilde{q}) \in \tilde{\delta}$ in $\tilde{G}^k$ if and only if there are $p \in \mathrm{Comp}_k^{-1}(\tilde{p})$ and $q \in \mathrm{Comp}_k^{-1}(\tilde{q})$ such that $\delta^k(p, a_i) = q$ is a transition in G^k.

It is worth noting that the quotient automaton $\tilde{G}^k$ may be nondeterministic even if the template G is deterministic.

Technically, for every $k \geq 2$, the quotient automaton $\tilde{G}^k$ is constructed from the template $G = (Q, \Sigma, \delta, q_0, F)$ as the accessible part of the nondeterministic automaton

$$\big(\{\,\tilde{q} \subseteq Q \mid |\tilde{q}| \leq k\,\},\ \Sigma,\ \tilde{\delta},\ \tilde{q}_0,\ \{\,\tilde{q}_m \subseteq F \mid |\tilde{q}_m| \leq k\,\}\big),$$

where, for each state $\tilde{q} \subseteq Q$ and event $a \in \Sigma$, the transition function is defined by $\tilde{\delta}(\tilde{q}, a) = S_1 \cup S_2 \cup S_3$ with

$$\begin{aligned} S_1 &= \big\{\,\{\delta(q,a) \mid q \in \tilde{q}\} \;\big|\; a \in \Sigma_g \wedge (\forall q \in \tilde{q})\,\delta(q,a)!\big\}, \\ S_2 &= \big\{\,\tilde{q} \cup \{\delta(q,a)\} \;\big|\; |\tilde{q}| < k \wedge q \in \tilde{q} \wedge P\big\}, \\ S_3 &= \big\{\,(\tilde{q} \setminus \{q\}) \cup \{\delta(q,a)\} \;\big|\; |\tilde{q}| > 1 \wedge q \in \tilde{q} \wedge P\big\}, \end{aligned} \tag{1}$$

where $P \equiv a \in \Sigma_p \wedge \delta(q,a)!$; here, $\delta(q,a)!$ states that $\delta(q,a)$ is defined. For more details on the construction and its explanation, we refer to Rohloff and Lafortune [2].

Notice that the state space of the quotient automaton is independent of the number of modules as soon as the number of modules is larger than the number of states in the template.

Since the quotient automaton is nondeterministic, we need to define what we mean by a nonblocking (deadlock-free) state of the quotient automaton. We say that a state of the quotient automaton is nonblocking (deadlock-free) if the automaton is not blocked (deadlocked) in that state, that is, no accepting state can be reached from it (no transition is defined in it).

We now recall several basic results from the literature. The first result says that the DFA G^k is nonblocking if and only if all states of the quotient automaton $\tilde{G}^k$ are nonblocking.

Lemma 1 ([2, Theorem 5]): Let G be a template, and let $\{G_1, \ldots, G_k\}$ be an isomorphic module system instantiated from G. A state q is blocking in the DFA G^k iff the state $\mathrm{Comp}_k(q)$ is blocking in the quotient automaton $\tilde{G}^k$. ∎

The next result says that G^k is deadlock-free if and only if all states of the quotient automaton $\tilde{G}^k$ are deadlock-free.

Lemma 2 ([2, Theorem 3]): Let G be a template, and let $\{G_1, \ldots, G_k\}$ be an isomorphic module system instantiated from G. A state q deadlocks in the DFA G^k iff the state $\mathrm{Comp}_k(q)$ deadlocks in the quotient automaton $\tilde{G}^k$. ∎

The quotient automaton preserves reachability.

Lemma 3 ([2, Theorem 4]): Let G be a template, and let $\{G_1, \ldots, G_k\}$ be an isomorphic module system instantiated from G. A state q is reachable in the DFA G^k iff the state $\mathrm{Comp}_k(q)$ is reachable in the quotient automaton $\tilde{G}^k$. ∎

In the sequel, we need the following improvement of the previous result considering the cardinality of states of the quotient automaton.

Lemma 4: Let G be a template with n states, and let $2 \leq k \leq n$. If a state $\tilde{q}$ is reachable in the quotient automaton $\tilde{G}^k$ and $|\tilde{q}| < k$, then the state $\tilde{q}$ is reachable in the quotient automaton $\tilde{G}^{k-1}$.

Proof: Let $\tilde{q} = \{q_1, \ldots, q_\ell\}$, for $\ell < k$, be a state reachable in $\tilde{G}^k$. By Lemma 3, $q^k = (q_1, \ldots, q_\ell, (q_\ell)^{k-\ell}) \in \mathrm{Comp}_k^{-1}(\tilde{q})$ is reachable in the DFA G^k. Consider a string that reaches q^k in G^k. By erasing all private events of the kth component, we obtain a string that reaches the state $q^{k-1} = (q_1, \ldots, q_\ell, (q_\ell)^{k-1-\ell})$ in the DFA G^{k-1}. Lemma 3 then implies that $\mathrm{Comp}_{k-1}(q^{k-1}) = \tilde{q}$ is reachable in the quotient automaton $\tilde{G}^{k-1}$, which was to be shown. ∎

We now discuss the complexity of verifying nonblockingness and deadlock-freeness for isomorphic module systems.

III. Nonblockingness

The problem of nonblockingness for isomorphic module systems asks, given a template G, whether the parallel composition of an arbitrary number of instances of the template is nonblocking. Formally, we say that a *template G is nonblocking* if, for every $k \geq 1$, the parallel composition

$$G^k = G_1 \parallel \cdots \parallel G_k$$

is nonblocking, where G_i denotes the ith instance of the template G, for $i = 1, \ldots, k$. We have the following problem.

NAME: Template Nonblockingness (TNB)
INPUT: A template G over Σ.
QUESTION: Is the template G nonblocking?

It is not hard to see that if G^k is blocking, then so is G^{k+1}. In particular, if the template G is blocking, then there is a minimal k such that G^k is blocking. Rohloff and Lafortune [2] have shown that if such a k exists, then it is bounded by the number of states of the template G. This observation gives rise to a natural question: How to determine the minimal value of k for which the template G becomes blocking? To discuss this question, we define the function

$$\min_b(G) = \begin{cases} \min\{k \mid G^k \text{ is blocking}\} & \text{if } k \text{ exists} \\ \infty & \text{otherwise} \end{cases}$$

and use it to define the following problem.

NAME: Max Nonblocking Modules (Max-NBM)
INPUT: A template G over Σ.
OUTPUT: $\min_b(G) - 1$.

To discuss the complexity of this problem, it is worth defining its corresponding decision version.

NAME: Nonblocking Modules (NBM)
INPUT: A template G over Σ and an integer $k \geq 1$.
QUESTION: Is G^k nonblocking, i.e., is $k < \min_b(G)$?

We show that NBM is PSpace-complete. Consequently, Max-NBM is PSpace-hard, and TNB is PSpace-complete; indeed, TNB is equivalent to NBM for k being the number of states of the template.

Theorem 5: NBM is PSpace-complete.

Proof: Let G be a template, and let $k \geq 1$. Since PSpace is closed under complement and nondeterminism, we can guess, step by step, a trajectory that violates nonblockingness and show that the guess is correct in polynomial space by remembering only the current state of the quotient automaton $\tilde{G}^k$ and computing the next state based on the guess of the violating trajectory. Since each state of $\tilde{G}^k$ is a subset of states of G, which is of polynomial size, we can find a blocking state of $\tilde{G}^k$ in polynomial space. Hence NBM is in PSpace.

To prove PSpace-hardness, we reduce Modular Nonblockingness [6], [7]. The problem asks, given a set of DFAs, whether their parallel composition is nonblocking. To this end, let $\mathcal{A}_1, \ldots, \mathcal{A}_\ell$ be an instance of Modular

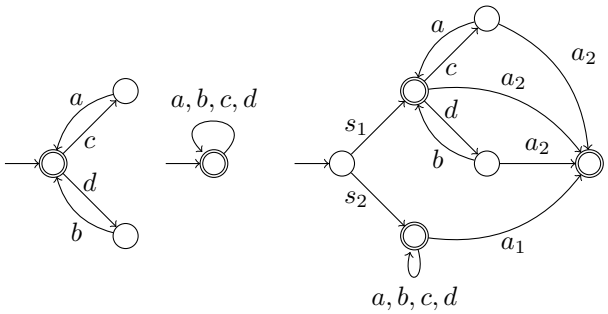

Fig. 1: An illustration of the reduction from Theorem 5 for two DFAs $\mathcal{A}_1$ and $\mathcal{A}_2$ with the resulting template G (right).

NONBLOCKINGNESS. Without loss of generality, we may assume that the DFAs are over the common alphabet Γ, see [7]. We set $k = \ell$ and create a DFA $G = (Q, \Sigma, \delta, q_0, F)$, the template, as a disjoint union of the automata $\mathcal{A}_1, \ldots, \mathcal{A}_\ell$, where each $\mathcal{A}_i = (Q_i, \Gamma, \delta_i, q_{0_i}, F_i)$.

Formally, G is defined as follows. The set of states $Q = \{q_0, q_a\} \cup \bigcup_{i=1}^{\ell} Q_i$, where q_0 is a fresh initial state and q_a is a fresh accepting state. Without loss of generality, we assume that $Q_i \cap Q_j = \emptyset$ whenever $i \neq j$; otherwise, we rename the states of $\mathcal{A}_i$ and adjust the transition function. The alphabet $\Sigma = \Gamma \cup \{s_1, \ldots, s_\ell\} \cup \{a_1, \ldots, a_\ell\}$, where the events $s_1, \ldots, s_\ell$ are the only private events. The transition function $\delta = \bigcup_{i=1}^{\ell} \delta_i$, extended with the following transitions. For every $i = 1, \ldots, \ell$, we add to δ the transitions

1) (q_0, s_i, q_{0_i}), where q_{0_i} is the initial state of $\mathcal{A}_i$,
2) and (q, a_j, q_a) for every $q \in Q_i$ and $j \neq i$.

The set of accepting states $F = \{q_a\} \cup \bigcup_{i=1}^{\ell} F_i$. An illustration of the construction is shown in Fig. 1.

We show that the DFA G^k is nonblocking iff the composition $\mathcal{A}_1 \parallel \cdots \parallel \mathcal{A}_k$ is nonblocking; recall that $k = \ell$, and G^k is a composition of k instances of the template G.

First, assume that G^k is nonblocking. For a contradiction, let w be a string that leads $\mathcal{A}_1 \parallel \cdots \parallel \mathcal{A}_k$ to a state $(q_1, \ldots, q_k)$ from which no accepting state is reachable. Then, G^k reaches state $(q_1, \ldots, q_k)$ under the string $\kappa_1(s_1) \cdots \kappa_k(s_k)w$; note that w consists solely of global events. None of the events a_j, $1 \leq j \leq k$, can be executed in $(q_1, \ldots, q_k)$ of G^k, since a_j is undefined in q_j. Hence, no accepting state is reachable from $(q_1, \ldots, q_k)$ in G^k, and thus G^k is blocking—a contradiction.

Now, assume that $\mathcal{A}_1 \parallel \cdots \parallel \mathcal{A}_k$ is nonblocking. To show that G^k is nonblocking, let $q \in Q_{i_1} \times \cdots \times Q_{i_k}$ be a state of G^k. We consider two cases. If $\{i_1, \ldots, i_k\} \neq \{1, \ldots, k\}$, then, for $j \notin \{i_1, \ldots, i_k\}$, there is an a_j-transition from state q to the accepting state $(q_a)^k$. Hence, q is nonblocking in G^k. If $\{i_1, \ldots, i_k\} = \{1, \ldots, k\}$, then q is a state of the composition $\mathcal{A}_{i_1} \parallel \cdots \parallel \mathcal{A}_{i_k}$. Since $\mathcal{A}_{i_1} \parallel \cdots \parallel \mathcal{A}_{i_k}$ is nonblocking iff $\mathcal{A}_1 \parallel \cdots \parallel \mathcal{A}_k$ is nonblocking, there is a string that leads $\mathcal{A}_{i_1} \parallel \cdots \parallel \mathcal{A}_{i_k}$ from state q to an accepting state. Because the same computation is possible in G^k, by construction, state q is nonblocking in G^k. Thus, in both cases, G^k is nonblocking. ∎

We now improve the algorithmic complexity of finding the minimal k for which a given template is blocking. To this end, we make use of several auxiliary results.

Lemma 6: Let G be a template with n states, and let $k \geq 1$. If there is a blocking state $\tilde{q}$ with $|\tilde{q}| = k$ in the quotient automaton $\tilde{G}^n$, then the DFA G^k is blocking.

Proof: If $\tilde{q} = \{q_1, \ldots, q_k\}$ is blocking in $\tilde{G}^n$ with $k < n$, then $\tilde{q}$ is reachable in $\tilde{G}^k$ by Lemma 4. By Lemma 1, every state of $\mathrm{Comp}_n^{-1}(\tilde{q})$ is blocking in the DFA G^n. In particular, $q^n = (q_1, \ldots, q_k, (q_k)^{n-k}) \in \mathrm{Comp}_n^{-1}(\tilde{q})$ is blocking in G^n. Since $\tilde{q}$ is reachable in $\tilde{G}^k$, state $q^k = (q_1, \ldots, q_k) \in \mathrm{Comp}_k^{-1}(\tilde{q})$ is reachable in the DFA G^k by Lemma 3. If an accepting state $(f_1, \ldots, f_k)$ were reachable from q^k in G^k, then the accepting state $(f_1, \ldots, f_k, f_k, \ldots, f_k)$ would be reachable from q^n in G^n, which is a contradiction because q^n is blocking. ∎

Lemma 7: Given a template G with n states, and a number $1 \leq k \leq n$, if all states $\tilde{q}$ with $|\tilde{q}| \leq k$ are nonblocking in $\tilde{G}^n$, then G^k is nonblocking.

Proof: By contraposition, if G^k is blocking, then there is a blocking state q^k reachable in G^k. Let $q^k = (q_1, \ldots, q_k)$. Then, state $q^n = (q_1, \ldots, q_k, (q_k)^{n-k})$ is reachable in G^n. If an accepting state $(f_1, \ldots, f_n)$ were reachable from q^n in G^n, then the accepting state $(f_1, \ldots, f_k)$ would be reachable from q^k in G^k, which is a contradiction. Therefore, q^n is blocking in G^n. By Lemma 1, state $\mathrm{Comp}_n(q^n) = \{q_1, \ldots, q_\ell\}$, for $\ell \leq k$, is blocking in $\tilde{G}^n$, which was to be shown. ∎

Now, we show that the minimal cardinality of a blocking state in $\tilde{G}^n$ specifies the blocking number of the template.

Theorem 8: If G is a template with n states, then $\min_b(G)$ is the minimal cardinality of a reachable blocking state of the quotient automaton $\tilde{G}^n$, or infinity if all states of $\tilde{G}^n$ are nonblocking.

Proof: Assume that $\tilde{G}^n$ is blocking, and let k_b be the minimal cardinality of a reachable blocking state in $\tilde{G}^n$. By Lemma 6, G^{k_b} is blocking, and hence $\min_b(G) \leq k_b$. By Lemma 7, $G^{k_b - 1}$ is nonblocking, and hence $k_b - 1 < \min_b(G)$. Altogether, $\min_b(G) = k_b$. ∎

Theorem 8 can be used to compute $\min_b(G)$ by constructing the quotient automaton, which is doable in $O(n2^n)$ time.

Theorem 9: Given a template G with n states, we can find $\min_b(G)$ in $O(n2^n)$ time. ∎

However, we can find $\min_b(G)$ in $O(\min_b(G) \cdot 2^{\min_b(G)})$ time. The idea is that, rather than to build the quotient automaton $\tilde{G}^n$ at once, we build it incrementally. We first build $\tilde{G}^2$, verify that it contains no blocking state, and if so, we build $\tilde{G}^3$ from $\tilde{G}^2$, etc.; see Algorithm 1.

Algorithm 1 The incremental computation of $\min_b(G)$.

Require: A template G.
Ensure: $\min_b(G)$.

```
if G is a blocking DFA then return 1
Compute G̃².
if G̃² contains a blocking state then return 2
for i = 3, ..., n do
    Compute G̃^i from G̃^{i-1}
    if a state of cardinality i is blocking in G̃^i then
        return i
return ∞
```

If the input DFA G is nonblocking, Algorithm 1 constructs $\tilde{G}^i$, using the algorithm in [2] if $i = 2$, and extending the automaton $\tilde{G}^{i-1}$ if $i \geq 3$. Then, it checks whether $\tilde{G}^i$ has a blocking state. If so, it terminates and returns i; otherwise, it repeats the **for**-loop. Since the loop is repeated at most $n-3$ times, the algorithm terminates. To analyze its complexity, we need to discus lines 5 and 6 in more detail.

The computation of $\tilde{G}^i$ from $\tilde{G}^{i-1}$ is outlined in Algorithm 2. The algorithm relies on the observation that $\tilde{G}^i$ can be constructed from $\tilde{G}^{i-1}$ by first considering the states of cardinality $i-1$ and applying the transition rules in (1) that increase the state's cardinality. The remaining transition rules are then applied only to the newly generated states, which have cardinality i.

Algorithm 2 Construction of $\tilde{G}^i$ from $\tilde{G}^{i-1}$

Require: $\tilde{G}^{i-1}$, for $3 \leq i \leq n$, with all states reachable.
Ensure: $\tilde{G}^i$ with all states reachable.

```
1: Set G̃^i ← G̃^{i-1}
2: Set S ← ∅
3: ▷ We use δ to denote the transition function of G, and
   δ̃^i to denote the transition function of G̃^i. ◁
4: for every state q̃ of G̃^i with |q̃| = i − 1 do
5:     for every q ∈ q̃ do
6:         for every a ∈ Σ_p do
7:             if δ(q, a)! then X ← q̃ ∪ {δ(q, a)}
8:                 add X to δ̃^i(q̃, a)
9:                 if |X| = i then insert X into S
10: ▷ From states of cardinality i − 1, the cycle above
    created states of cardinality i, which we further
    process. ◁
11: while S ≠ ∅ do
12:     Extract q̃ from S, and mark q̃ as "visited"
13:     for every a ∈ Σ_p do
14:         for every q ∈ q̃ do
15:             if δ(q, a)! then X ← (q̃ \ {q}) ∪ {δ(q, a)}
16:                 add X to δ̃^i(q̃, a)
17:                 if |X| = i & X not "visited" then
18:                     insert X into S
19:     for every a ∈ Σ_g do
20:         if δ(q, a)! for every q ∈ q̃ then
21:             X ← {δ(q, a) | q ∈ q̃}
22:             add X to δ̃^i(q̃, a)
23:             if |X| = i & X not "visited" then
24:                 insert X into S
```

The complexity of Algorithm 2 is $O(\binom{n}{i-1} \cdot (i-1) \cdot |\Sigma_p| + \binom{n}{i} \cdot (|\Sigma_p| \cdot i + |\Sigma_g| \cdot i))$, since every state is marked "visited" only once. Thus, disregarding the size of the alphabet in the analyzes, the complexity is $O(i\binom{n}{i})$.

Line 6 of Algorithm 1 can be implemented in the construction of line 5. If $\tilde{G}^{i-1}$ contains no blocking states, we mark all states of $\tilde{G}^{i-1}$ as "nonblocking". Then, in $\tilde{G}^i$, it suffices to determine whether the states of cardinality i are nonblocking. To achieve this, when constructing $\tilde{G}^i$ from $\tilde{G}^{i-1}$, each time a marked state is reached via a transition, we traverse backward along the transition edges through unmarked states, marking them "nonblocking" as we go. Note that marked states are not revisited, ensuring thus that every state of cardinality i is marked and traversed at most once. Consequently, the computational complexity of line 6 is $O(\binom{n}{i} + i \cdot \binom{n}{i})$; the second part expresses the number of added transitions.

The complexity of Algorithm 1 is based on line 4, which terminates after $\min_b(G) - 3$ iterations if $\min_b(G) \neq \infty$, and the cycle is dominated by line 5, which altogether gives $O\left(\sum_{i=3}^{\min_b(G)} i\binom{n}{i}\right) = O(\min_b(G) \cdot 2^{\min_b(G)})$.

IV. Deadlock-freeness

We say that a *template G is deadlock-free* if the DFA G^k is deadlock-free for every $k \geq 1$. If G^k is not deadlock-free, neither is G^{k+1}. If such a k exists, then it is bounded by the number of states of G, see Rohloff and Lafortune [2]. We formulate the problem as follows:

NAME: Template Deadlock-freeness (TDF)
INPUT: A template G over Σ.
QUESTION: Is the template G deadlock-free?

A closely related problem is to find the minimal k for which G^k is not deadlock-free. We define the function

$$\min_d(G) = \begin{cases} \min\{k \mid G^k \text{ is not deadlock-free}\} & \text{if } k \text{ exists} \\ \infty & \text{otherwise}. \end{cases}$$

The problem of our interest asks to find the maximal number of deadlock-free modules:

NAME: Max Deadlock-free Modules (Max-DFM)
INPUT: A template G.
OUTPUT: $\min_d(G) - 1$.

Its corresponding decision version can be defined as follows:

NAME: Deadlock-free Modules (DFM)
INPUT: A template G and a natural number $k \geq 1$.
QUESTION: Is G^k deadlock free, i.e., is $k < \min_d(G)$?

We prove that DFM is PSPACE-complete. To do this, we first show that the following problem is PSPACE-hard.

NAME: Modular Deadlock-Freeness (Modular-DF)
INPUT: DFAs $\mathcal{A}_1, \ldots, \mathcal{A}_k$.
QUESTION: Is $\mathcal{A}_1 \parallel \cdots \parallel \mathcal{A}_k$ deadlock-free?

Theorem 10: Modular-DF is PSPACE-hard.

Proof: We reduce the PSPACE-complete problem DFA-Intersection, which asks, given DFAs $\mathcal{A}_1, \ldots, \mathcal{A}_k$ over a common alphabet Σ, whether $L_m(\mathcal{A}_1) \cap \cdots \cap L_m(\mathcal{A}_k) = \emptyset$. Without lost of generality, we may assume that the DFAs are complete [8].

Let $d \notin \Sigma$ be a fresh event. From $\mathcal{A}_i$, we construct the DFA $\mathcal{B}_i$ by adding a new state, *dead*$_i$ and d-transitions from every accepting state to state *dead*$_i$. Note that $L_m(\mathcal{B}_i) = L_m(\mathcal{A}_i)$.

We show that $\mathcal{B} = \mathcal{B}_1 \parallel \cdots \parallel \mathcal{B}_k$ is deadlock-free iff $\mathcal{L} = L_m(\mathcal{A}_1) \cap \cdots \cap L_m(\mathcal{A}_k) = \emptyset$. To this end, note that $\mathcal{B}$ is deadlocked only in state *dead* $= (dead_1, \ldots, dead_k)$; in other reachable states, $\mathcal{B}$ can execute a transition over every event from Σ due to completeness of all $\mathcal{A}_i$'s.

If $\mathcal{L} = \emptyset$, then state *dead* is not reachable in $\mathcal{B}$; if a string wd reached state *dead*, then w would reach an accepting

state in $\mathcal{A}_i$ for every $i = 1, \ldots, k$, which would make $\mathcal{L}$ nonempty.

If $\mathcal{L} \neq \emptyset$, then there is a string $w \in \mathcal{L}$. In particular, for every i, w leads $\mathcal{A}_i$ to an accepting state. Then, wd leads $\mathcal{B}_i$ to state *dead*$_i$. Thus, wd leads $\mathcal{B}$ to state *dead*, and hence $\mathcal{B}$ is not deadlock-free. ∎

Theorem 11: DFM is PSPACE-complete.

Proof: Membership in PSPACE can be shown by nondeterministic search as in the proof of Theorem 5.

To show PSPACE-hardness, we reduce MODULAR-DF. Let $\mathcal{A}_1, \ldots, \mathcal{A}_\ell$ over Γ be an instance of MODULAR-DF. We set $k = \ell$ and create a DFA $G = (Q, \Sigma, \delta, q_0, F)$, the template, as a disjoint union of the automata $\mathcal{A}_1, \ldots, \mathcal{A}_\ell$, where each $\mathcal{A}_i = (Q_i, \Gamma, \delta_i, q_{0_i}, F_i)$. Formally, G is defined as follows. The set of states $Q = \{q_0, q_{df}\} \cup \bigcup_{i=1}^{\ell} Q_i$, where q_0 is a fresh initial state and q_{df} is a fresh deadlock-free state. Without loss of generality, we assume that $Q_i \cap Q_j = \emptyset$ whenever $i \neq j$; otherwise, we rename the states of $\mathcal{A}_i$ and adjust the transition function. The alphabet $\Sigma = \Gamma \cup \{s_1, \ldots, s_\ell\} \cup \{a_1, \ldots, a_\ell\} \cup \{e\}$, where the events $s_1, \ldots, s_\ell$ are the only private events. The transition function $\delta = \bigcup_{i=1}^{\ell} \delta_i$, extended with the following transitions. For every $i = 1, \ldots, \ell$, we add to δ the transitions

1) (q_0, s_i, q_{0_i}), where q_{0_i} is the initial state of $\mathcal{A}_i$,
2) (q, a_j, q_{df}) for every $q \in Q_i$ and $j \neq i$, and
3) (q_{df}, e, q_{df}).

The set of accepting states F is not relevant.

Similarly as in the proof of Theorem 5, we can show that $\mathcal{A}_1 \parallel \cdots \parallel \mathcal{A}_k$ is deadlock-free iff G^k is deadlock-free. ∎

Lemma 12: Let G be a template with n states, and let $k \geq 1$. If there is a deadlocked state $\tilde{q}$ with $|\tilde{q}| = k$ in $\tilde{G}^n$, then the DFA G^k is not deadlock-free.

Proof: If $\tilde{q} = \{q_1, \ldots, q_k\}$ deadlocks in $\tilde{G}^n$ with $k < n$, then $\tilde{q}$ is reachable in $\tilde{G}^k$ by Lemma 4. By Lemma 2, every state of $\mathrm{Comp}_n^{-1}(\tilde{q})$ deadlocks in the DFA G^n. In particular, $q^n = (q_1, \ldots, q_k, (q_k)^{n-k}) \in \mathrm{Comp}_n^{-1}(\tilde{q})$ deadlocks in G^n. Since $\tilde{q}$ is reachable in $\tilde{G}^k$, $q^k = (q_1, \ldots, q_k) \in \mathrm{Comp}_k^{-1}(\tilde{q})$ is reachable in the DFA G^k by Lemma 3. As q^n deadlocks in G^n, so does q^k in G^k. Hence, G^k is not deadlock-free. ∎

Lemma 13: Given a template G with n states, and a number $1 \leq k \leq n$, if all states $\tilde{q}$ with $|\tilde{q}| \leq k$ are deadlock-free in $\tilde{G}^n$, then G^k is deadlock-free.

Proof: By contraposition, if G^k is not deadlock-free, then there is a deadlocked state q^k reachable in G^k. Let $q^k = (q_1, \ldots, q_k)$. Then, $q^n = (q_1, \ldots, q_k, (q_k)^{n-k})$ is reachable in G^n. Since q^k deadlocks in G^k, so does q^n in G^n. By Lemma 2, state $\mathrm{Comp}_n(q^n)$ deadlocks in $\tilde{G}^n$. The observation that the cardinality of $\mathrm{Comp}_n(q^n)$ is at most k completes the proof. ∎

Now, we show that the minimal cardinality of a deadlocked state in $\tilde{G}^n$ specifies the deadlocking number of the template.

Theorem 14: If G is a template with n states, then $\min_d(G)$ is the minimal cardinality of a reachable deadlocked state of the quotient automaton $\tilde{G}^n$, or infinity if all states of $\tilde{G}^n$ are deadlock-free.

Proof: Let k_d be the minimal cardinality of a reachable deadlocked state in $\tilde{G}^n$. By Lemma 12, G^{k_d} is not deadlock-free, and hence $\min_d(G) \leq k_d$. By Lemma 13, G^{k_d-1} is deadlock-free, and hence $k_d - 1 < \min_d(G)$. Altogether, $\min_d(G) = k_d$. ∎

Theorem 14 can be used to compute $\min_d(G)$ by constructing the quotient automaton, which is doable in $O(n2^n)$ time.

Theorem 15: Given a template G with n states, we can find $\min_d(G)$ in $O(n2^n)$ time. ∎

Similarly as above, we can find $\min_d(G)$ in time $O(\min_d(G) \cdot 2^{\min_d(G)})$; see Algorithm 3. The complexity analysis of Algorithm 3 is analogous to the analyses of Algorithm 1.

Algorithm 3 The incremental computation of $\min_d(G)$.

Require: A template G.
Ensure: $\min_d(G)$.
1: **if** G is not a deadlock-free DFA **then return** 1
2: Compute $\tilde{G}^2$.
3: **if** $\tilde{G}^2$ contains a deadlocked state **then return** 2
4: **for** $i = 3, \ldots, n$ **do**
5: Compute $\tilde{G}^i$ from $\tilde{G}^{i-1}$
6: **if** a state of cardinality i is deadlocked in $\tilde{G}^i$ **then return** i
7: **return** ∞

V. CONCLUSION

We investigated the verification of nonblockingness and deadlock-freeness in isomorphic module systems, focusing on the fundamental question of finding the maximum number of modules that can be safely composed while maintaining these properties. We established the complexity of the related decision problems—both are PSPACE-complete—and demonstrated that the minimal number, m, of modules of a template G with n states that cause the system to become blocking ($\min_b(G)$) or deadlocked ($\min_d(G)$) can be found in $O(\min(m2^m, n2^n))$ time by incrementally analyzing the minimal cardinality of the reachable blocking/deadlocked states of the quotient automaton.

REFERENCES

[1] P. Ramadge and W. Wonham, "The control of discrete event systems," *Proceedings of IEEE*, vol. 77, pp. 81–98, 1989.

[2] K. Rohloff and S. Lafortune, "The verification and control of interacting similar discrete-event systems," *SIAM Journal on Control and Optimization*, vol. 45, no. 2, pp. 634–667, 2006.

[3] W. Wang, R. Su, L. Lin, and C. Gong, "Model checking in isomorphic module systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 2, pp. 728–735, 2019.

[4] R. Su and B. Lennartson, "Control protocol synthesis for multi-agent systems with similar actions instantiated from agent and requirement templates," *Automatica*, vol. 79, p. 244–255, 2017.

[5] C. Cassandras and S. Lafortune, *Introduction to discrete event systems*, 3rd ed. Springer, 2021.

[6] K. Rohloff and S. Lafortune, "PSPACE-completeness of modular supervisory control problems," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 15, pp. 145–167, 2005.

[7] T. Masopust, "Complexity of verifying nonblockingness in modular supervisory control," *IEEE Transactions on Automatic Control*, vol. 63, no. 2, pp. 602–607, 2018.

[8] D. Kozen, "Lower bounds for natural proof systems," in *Annual Symposium on Foundations of Computer Science*, 1977, pp. 254–266.