Teze disertace
k získání vědeckého titulu “doktor věd”
ve skupině věd fyzikálně-matematických

Complexity of Verification and Control of Modular Discrete Event Systems

Komise pro obhajoby doktorských disertací v oboru
informatika a kybernetika

Jméno uchazeče: RNDr. Tomáš Masopust, Ph.D.

Pracoviště uchazeče: Matematický ústav AV ČR, v.v.i.

Místo a datum: Brno, 02.05.2018

Abstract

Discrete event systems (DESs) are dynamical systems evolving by occurrences of events that appear independently in time. The evolution of a DES cannot be captured by classical modeling formalisms, such as differential equations, but it can be captured by tools like automata or Petri nets. Supervisory control is a formal approach for control of DESs solving problems such as safety, nonblockingness, and fault diagnosis encountered in systems with a high degree of automation. Supervisory control of monolithic systems is a well-understood problem. However, large-scale systems are modular, consisting of many small concurrent parts, where the state-space explosion problem is the main issue when making the system monolithic. The modular setting results in global specifications for which we suggest a new approach to synthesize local controllers. We further investigate the verification of nonblockingness for modular systems, the computation of a coordinator for nonblockingness, the effect of projections on the size of automata, the computation of a safe subset of specifications, and the complexity of verifying some properties of modular DESs. In decentralized supervisory control, several local supervisors cooperate to accomplish a common goal. We suggest a new approach to construct a controllable and coobservable subspecification by extending observable events of local supervisors via communication and applying a fully decentralized computation of local supervisors. We also discuss the possibility to extend the framework to non-regular systems.

Acknowledgements

I am very grateful to Dr. Jan Komenda and Prof. Jan H. van Schuppen who introduced me to the topic and with whom I had the opportunity to work, and to all my co-authors who have worked with me on the topic.

Special thanks go to the Institute of Mathematics of the Czech Academy of Sciences and to the Theoretical Computer Science department of the Faculty of Informatics of the Technical University Dresden that maintained a pleasant and flexible environment for my research.

During the time, my research was partially supported by several projects: by the Czech Science Foundation project P202/11/P028 (*Decentralized and coordination supervisory control*), by the Czech Science Foundation projects GA15-02532S (*Modular and Decentralized Control of Discrete-Event and Hybrid Systems with Communication*), by the Ministry of Education, Youth, and Sport of the Czech Republic under the Kontakt II research project LH13012 (*Multi-level supervisory control (MUSIC)*), by the German Research Foundation (DFG) in Emmy Noether grant MA 4938/2-1 (*Querschnitte: XML und formale Sprachen – Theorie und Praxis*), and by the German Research Foundation (DFG) in Emmy Noether grant KR 4381/1-1 (*Datenintegration und -abfrage durch die Zusammenführung von Ontologien und Datenbanken*).

My greatest thanks go to my wife Iveta for her infinite patience and support.

Introduction

As the complexity of man-made systems grows, the risk of a human operator error increases and a correct behavior of complex systems can only be ensured by a supervisory control system. Supervisory control is a formal approach to control a system solving problems such as safety (avoidance of forbidden states), nonblockingness (avoidance of deadlocks and live-locks), fault diagnosis (identification of erroneous behavior), opacity (hiding a secret information in the system), et cetera.

Discrete event systems (DESs) introduced by Ramadge and Wonham [6] are dynamical systems evolving so that, in each state of the system, a number of different transitions may occur. A supervisor provides a possibility for control actions that at any instance of time may prohibit certain transitions from occurring. The supervisory control problem is to design a supervisor that satisfies given specifications. Specifications considered in the literature often require that the supervisor prohibits certain undesirable sequences of events while, at the same time, allows desirable sequences to occur. The supervisory control problem changes with respect to different assumptions made on the information available to the supervisor: the supervisor may have full knowledge of the state of the system (perfect information) or it may have access only to some partial information (imperfect information).

In our work, a DES G is a deterministic finite automaton

$$G = (Q, \Sigma, \delta, q_0, Q_m)$$

where Q is a finite set of states, $q_0 \in Q$ is the initial state of the system, $Q_m \subseteq Q$ is the set of marked states, Σ is a finite set of events used to label transitions between states, and δ is a partial transition function from $Q \times \Sigma$ into Q describing the dynamics of the system. The interpretation of δ is that if $\delta(q,a)$ is defined for state $q \in Q$ and event $a \in \Sigma$, then a transition labeled with a may take place in state q and the next state of the system is state $\delta(q,a)$. We define $L(G)$, the language generated by G, as the set of all strings s such that $\delta(q_0,s)$ is defined, and $L_m(G)$, the language marked by G, as the set of all strings s such that $\delta(q_0,s)$ is a marked state.

To provide a possibility for control, the event set Σ is partitioned into two disjoint subsets Σ_u and Σ_c, where Σ_c is interpreted as the set of events that can be controlled (disabled). We define a supervisor for G as a function $\gamma\colon \Sigma^* \to 2^\Sigma$ such that $\Sigma_u \subseteq \gamma(s)$, for all $s \in \Sigma^*$; the set $\gamma(s)$ is the set of events that are allowed by the supervisor to occur after observing the string s.

A DES G together with a supervisor γ are called a supervised system. Given a supervised system (G,γ), the language $L(G,\gamma)$ is the set of all strings in Σ^* generated by the system G under the supervision of γ, that is, $\varepsilon \in L(G,\gamma)$, and if $s \in L(G,\gamma)$ and $sa \in L(G)$ such that $a \in \gamma(s)$, then $sa \in L(G,\gamma)$.

A supervisor need not have access to the entire history of events; it is with partial information as opposed to perfect information. To model partial information, let $P\colon \Sigma \to \Sigma_o \cup \{\varepsilon\}$ be a function where we interpret $P(a)$ as the information provided to the supervisor on the value of a. The possibility that $P(a)$ is the empty string means that the supervisor does not learn that a transition has occurred. We extend P to a mapping from Σ^* into Σ_o^* by letting $P(a_1 \cdots a_n)$ be the concatenation $P(a_1)\cdots P(a_n)$. A supervisor γ is under partial observation if there is a function $\gamma_P\colon \Sigma^* \to 2^\Sigma$ such that $\gamma(s) = \gamma_P(P(s))$ for all $s \in \Sigma^*$. The most common model of partial information is a natural projection where for $\Sigma_o \subseteq \Sigma$, $P(a) = a$ for $a \in \Sigma_o$ and $P(a) = \varepsilon$ otherwise.

A class of supervisors of interest is the set of finite-state feedback supervisors. A supervisor γ belongs to this class if there exists a DES $S = (Q', \Sigma, q_0', \delta', Q_m')$ and a function $\gamma_S\colon Q' \to 2^\Sigma$ such that $\gamma(s) =$

$\gamma_S(\delta'(q_0', s))$. Any such S, together with the mapping γ_S, is called a finite state realization of γ. We often identify the finite-state feedback supervisor γ_S with its DES S and use the notation $L(S/G)$ to denote the language $L(G, \gamma_S)$. The marked language of the supervised system depends on a specification K and is defined as $L_m(S/G) = L(S/G) \cap K$, that is, the supervisor may mark according to the marking in the specification. If the supervised system is nonblocking, that is,

$$\overline{L_m(S/G)} = L(S/G)$$

where $\overline{L}$ denotes the prefix-closure of a language L, then the supervisor S is called *nonblocking*.

In the monolithic case, the supervisory control problem is the question, give a specification K and a plant G, to synthesize a maximal (optimal) nonblocking finite-state feedback supervisor S such that

$$L_m(S/G) \subseteq K\,.$$

This problem can be solved in polynomial time with respect to the size of the DES representations of K and G [37, 53].

The Problem

Although the approaches for the monolithic supervisory control problem are well mastered, the main problem arises from the limited computational resources. A plant is often modeled as a composition of a large number of small systems (it is modular or concurrent). Even if each component consists only of a few states, the state complexity of their composition grows exponentially with respect to the number of components (known as the state-space explosion problem).

Let the plant be given as a set $\{G_1, G_2, \ldots, G_n\}$ of systems. Then the monolithic system is

$$G = G_1 \parallel G_2 \parallel \cdots \parallel G_n$$

and the state space of the monolithic plant G grows exponentially in the number of components. Therefore, the complexity of the monolithic supervisory control design grows very fast in the size of the model [38].

Let K be a specification. There are two possibilities:

1. *K can* be decomposed to $K = K_1 \parallel \cdots \parallel K_n$ in such a way that K_i is a specification for G_i (often, this is assumed in the literature [54, 15]), or

2. *K cannot* be decomposed to $K = K_1 \parallel \cdots \parallel K_n$ in such a way that K_i is a specification for G_i.

Let G_i be an automaton over Σ_i, $i = 1, \ldots, n$. Then the question leads to the problem whether there are languages K_i over Σ_i such that $K = K_1 \parallel \cdots \parallel K_n$, which is known as *separability* in the literature [54] and is equivalent to the question whether

$$K = \|_{i=1}^{n} P_i(K)\,,$$

where P_i is a projection from $\cup_{i=1}^{n}\Sigma_i$ to Σ_i. This problem is PSPACE-complete [16].

If the specification is separable, then we can take $K_i = P_i(K)$ and use the monolithic supervisory control synthesis for G_i and K_i to synthesize an optimal supervisor S_i such that

$$L_m(S_i/G_i) \subseteq K_i\,,$$

which ensures by construction that the parallel composition of supervised systems satisfies the specification, that is,

$$L_m(S_1/G_1) \parallel L_m(S_2/G_2) \parallel \cdots \parallel L_m(S_n/G_n) \subseteq K\,.$$

However, the question we often face is that K is not separable. It could be natural to take a maximal separable sublanguage of K (with respect to inclusion), but to find out whether there is such a nonempty sublanguage is an undecidable problem [28].

Our Approach

Our suggestion to overcome this issue is to use conditional decomposability [18], which abstracts the necessary information among the components to ensure separability.

A language K is *conditionally decomposable* with respect to alphabets $(\Sigma_i)_{i=1}^n$ and an alphabet Σ' if Σ' contains all shared events and K is separable with respect to alphabets $(\Sigma_i \cup \Sigma')_{i=1}^n$.

We have shown two advantages of conditional decomposability:

1. Every language can be made conditionally decomposable by finding a convenient alphabet Σ'. This helps overcome the undecidable issue of finding a maximal nonempty separable sublanguage.

2. The complexity of checking conditional decomposability is lower than that of separability—conditional decomposability can be verified in polynomial time, compared to PSPACE-completeness of deciding separability.

Theorem 1. *Deciding whether a language K is conditionally decomposable can be solved in polynomial time with respect to the size of the DES representation of K and the number of components. The algorithmic complexity is cubic.*

The way how to abstract the most appropriate information is still under investigation. We have shown that to compute a minimal such information with respect to set cardinality is NP-hard [19] and that such a minimal information is not always the best choice [23]. Therefore, we designed a polynomial algorithm to find some information making the language conditionally decomposable [18].

We proceed as follows. Consider (for simplicity only two) automata G_1 and G_2 over Σ_1 and Σ_2, respectively, and let K be a specification. We construct an alphabet Σ' and a coordinator for safety G' as follows:

1. We set $\Sigma' = \Sigma_1 \cap \Sigma_2$ to be the set of all pairwise shared events;
2. We use our polynomial algorithm to extend Σ' so that K and $\overline{K}$ are conditional decomposable;
3. We compute the coordinator for safety $G' = P_{\Sigma'}(G_1) \parallel P_{\Sigma'}(G_2)$, where $P_{\Sigma'}$ is a projection to Σ'.

This choice of the coordinator G' does not modify the behavior of the system in the sense that $G_1 \| G_2 \| G' = G_1 \| G_2$.

Since the projection of an automaton may result in an automaton of exponential size (see the next section for more details), we may need to further extend the alphabet Σ' so that the projection $P_{\Sigma'}$ is an $L(G_i)$-observer, for $i = 1,2$, which can be done in polynomial (cubic) time [36, 4]. This property ensures that the projected automaton is not bigger than the original automaton (it is often much smaller) and that the projection can be computed in polynomial time [55]. The coordinator for safety is then often much smaller than the monolithic plant.

Remark 2. For systems with many components, the abstracted information often implies too much communication among the supervisors although it is not needed. Therefore, rather than a single abstracted set of information, we suggested (and constructed an algorithm) to search for several local extensions that still ensure conditional decomposability [16]. The polynomial-time complexity is preserved. In the rest of this thesis, we simplify the discussion only to a single abstracted set of information.

Partial Observation and Projections

As seen above, projections play an important role in our approach. It is well known that the projection of an automaton may result in an automaton of exponential size. In our approach, we eliminated this state explosion by the requirement that the projection is an observer, which is a known property that bounds the size of the projected automaton to at most the size of the input automaton [55]. However, the question what is the worst-case complexity of projected automata is of its own interest.

For a regular language L, we denote by $\|L\|$ the smallest number of states in any DES accepting the language L.

Theorem 3 ([55, 35, 14])**.** *Let $n \geq 2$ and L be a regular language over Σ with $\|L\| = n$. Let $\Sigma_o \subseteq \Sigma$ and P be the projection of Σ^* onto Σ_o^*. The tight upper bound on the size of the minimal DES for the projected language $P(L)$ is $3 \cdot 2^{n-2} - 1$.*

We further improved the upper bound by considering the structure of the automata; namely, we consider the number of states incident with unobservable transitions.

Theorem 4. *Let $m, n \geq 2$, $\Sigma_o \subseteq \Sigma$, and P be the projection from Σ^* onto Σ_o^*. Let L be a regular language over the alphabet Σ with $\|L\| = n$, and $(Q, \Sigma, \delta, s, F)$ be the minimal DES recognizing the language L, in which $|\{p, q \in Q \mid p \neq q \text{ and } q \in \delta(p, \Sigma \setminus \Sigma_o)\}| = m$. Then $\|P(L)\| \leq 2^{n-1} + 2^{n-m} - 1$ and the bound is tight.*

Notice that Theorem 3 is a direct consequence of our Theorem 4 since $\|P(L)\|$ is maximal if $m = 2$.

The situation is significantly different for projections of regular languages with one-letter co-domains.

Theorem 5. *Let a be a symbol in an alphabet Σ and P be the projection of strings in Σ^* to strings in a^*. Let L be a regular language over Σ with $\|L\| = n$. Then $\|P(L)\| \leq e^{(1+o(1))\sqrt{n \ln n}}$.*

The following theorem discusses a special case that gives an idea how to treat the cases with more and more unobservable transitions.

Theorem 6. *Let $m, n \geq 2$ and $\Sigma_o \subseteq \Sigma$. Let P be the projection of strings in Σ^* to strings in Σ_o^*. Let L be a regular language over the alphabet Σ with $\|L\| = n$, and $(Q, \Sigma, \delta, s, F)$ be the minimal DES recognizing the language L, in which $|\{p, q \in Q \mid p \neq q \text{ and } q \in \delta(p, \Sigma \setminus \Sigma_o)\}| = m$. If at least m transitions in the DES are unobservable for the projection, then $\|P(L)\| \leq 2^{n-2} + 2^{n-3} + 2^{n-m} - 1$ and the bound is tight.*

We also considered the case of finite languages.

Proposition 7. *Let a be a symbol in an alphabet Σ and let P be the projection of Σ^* onto a^*. If L is a finite regular language over Σ, then $\|P(L)\| \leq \|L\|$. The bound is tight for any alphabet.*

Theorem 8. *Let $k, n \geq 2$. There exist alphabets Σ and Σ_o with $\Sigma_o \subseteq \Sigma$ and $|\Sigma_o| = k$, and a finite language L over Σ with $\|L\| = n$ such that*

$$\|P(L)\| = \frac{k^{\lfloor n/(\log k+1) \rfloor + 1} - 1}{k-1} - 1$$

where P is the projection of strings in Σ^* onto strings in Σ_o^*. In addition, for any finite language L' over Σ,

$$\|P(L')\| \leq \frac{k^{\lceil (n/(\log k+1)) \rceil + 1} - 1}{k-1} - 1.$$

To consider a more practical point of view, there is a study that, based on some empirical studies, conjectures that the average complexity of projections of DESs is quasi-polynomial [8].

Application in Modular Supervisory Control

We now formulate the modular supervisory control problem of our interest. For simplicity, we formulate it only for the case of two systems, but it should be clear how to generalize it to any number of components.

Problem 9. Let G_1 and G_2 be generators over the alphabets Σ_1 and Σ_2, respectively. Assume that a specification $K \subseteq L_m(G_1 \| G_2)$ and its prefix-closure $\overline{K}$ are conditionally decomposable with respect to Σ_1, Σ_2, and Σ', for an alphabet Σ' such that $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma' \subseteq \Sigma_1 \cup \Sigma_2$. Let $G' = P_{\Sigma'}(G_1) \parallel P_{\Sigma'}(G_2)$ be a coordinator that ensures the necessary communication between the systems G_1 and G_2. The aim is to determine nonblocking supervisors S_1 and S_2 such that

$$L_m(S_1/[G_1 \| G']) \parallel L_m(S_2/[G_2 \| G']) = K$$

and

$$L(S_1/[G_1 \| G']) \parallel L(S_2/[G_2 \| G']) = \overline{K},$$

i.e., the supervisors fulfill the specification and are nonconflicting.

The question is whether it is possible to distribute the monolithic supervisor with help of a coordinator in such a way that there are nonblocking and nonconflicting supervisors S_1 and S_2 such that the supervised system satisfies safety and nonblockingness.

We have shown that such supervisors S_1 and S_2 exist if and only if $L_m(S_1/(G_1 \| G')) \parallel P_{\Sigma'}(S_2) = P_{1+\Sigma'}(K)$, $L_m(S_2/(G_2 \| G')) \parallel P_{\Sigma'}(S_1) = P_{2+\Sigma'}(K)$, and S_1 and S_2 are nonblocking and nonconflicting supervisors for $G_1 \| G'$ and $G_2 \| G'$, respectively. This gives the equations with

variable languages X_1 and X_2:

$$\begin{aligned} P_{1+\Sigma'}(K) \subseteq X_1 \subseteq G_1 \| G', \quad X_1 \| P_{\Sigma'}(X_2) = P_{1+\Sigma'}(K) \\ P_{2+\Sigma'}(K) \subseteq X_2 \subseteq G_2 \| G', \quad X_2 \| P_{\Sigma'}(X_1) = P_{2+\Sigma'}(K) \,. \end{aligned} \tag{3.1}$$

Let $\inf \mathrm{C}(K, L)$ denote the infimal prefix-closed controllable superlanguage of K with respect to L [6]. Then we immediately have the following.

Proposition 10. *If the specification K is prefix-closed, then there is a solution of Problem 9 if and only if the languages*

1. $T_1 = \inf \mathrm{C}(P_{1+\Sigma'}(K), G_1 \| G')$ *and*
2. $T_2 = \inf \mathrm{C}(P_{2+\Sigma'}(K), G_2 \| G')$

satisfy equations (3.1).

The result does not hold for general *non-prefix-closed* languages, because there is no infimal non-prefix-closed controllable superlanguage [6]. However, we have shown the following.

Theorem 11. *There is a solution for specification K if and only if there is a solution for its prefix-closure $\overline{K}$.*

Hence, it is sufficient to check whether there is a solution for the prefix-closure $\overline{K}$ of the specification K. Theorem 11 then gives a solution for K, which can actually be constructed from the solution for $\overline{K}$.

Infimal Observable Superlanguages

To generalize our approach to partial observation, we need to generalize the computation of infimal controllable superlanguages to infimal controllable and observable superlanguages.

Lafortune and Chen [26] have shown that the infimal prefix-closed and controllable superlanguage can be computed from a deterministic automaton in linear time. Kumar and Shayman [25] further showed that when considering the computation of the infimal prefix-closed

and observable superlanguage of a language K over Σ with respect to $L(G)$, it is sufficient to consider the computation with respect to the language Σ^*. The existence of a polynomial (time or space) algorithm for this computation was an open problem.

For a language K of state complexity n, we showed that the upper-bound on the state complexity of the infimal prefix-closed and observable superlanguage of K with respect to the language Σ^* is $2^n + 1$. We further prove that this bound is asymptotically tight by showing that the worst-case lower-bound state complexity is at least $\frac{3}{4} \cdot 2^n - 1 = \Omega(2^n)$. Since the state complexity is exponential, so is the time complexity of any algorithm computing the corresponding minimal DES.

Theorem 12. *Let K over Σ be a language with state complexity n, and let P be a projection. Then the worst-case state complexity of the infimal prefix-closed and observable superlanguage of K is $\Theta(2^n)$.*

Our construction shows that a DES representation of the infimal prefix-closed and observable superlanguage of K can be computed in time $O(2^n)$. This improves the complexity that can be derived from the previously known formulae of Rudie and Wonham [42] and Kumar and Shayman [25].

If Problem 9 has no Solution

If there is no solution of Problem 9, we focus on a slightly modified problem with help of a coordinator in order to obtain an acceptable sublanguage of the specification for which a solution exists.

Problem 13. Consider generators G_1 and G_2 over alphabets Σ_1 and Σ_2, respectively. Let Σ' be an alphabet such that $\Sigma_1 \cap \Sigma_2 \subseteq \Sigma' \subseteq \Sigma_1 \cup \Sigma_2$. Let G' over Σ' be a coordinator for safety. Assume that a specification $K \subseteq L_m(G_1 \| G_2 \| G')$ and its prefix-closure $\overline{K}$ are conditionally decomposable with respect to Σ_1, Σ_2, and Σ'. The aim is to determine nonblocking supervisors S_1 and S_2 such that

$$L_m(S_i/[G_i \| G']) \subseteq P_{i+\Sigma'}(K),$$

$i = 1,2$, and the supervised system satisfies

$$L_m(S_1/[G_1\|G']) \parallel L_m(S_2/[G_2\|G']) = K\,.$$

In this way, we basically reduced the modular supervisory control problem to several monolithic problems for small(er) systems. We then showed that there exist such supervisors if and only if the specification K is conditionally controllable [19, 23].

Theorem 14. *There exist nonblocking supervisors S_1 and S_2 such that the supervised system satisfies $L_m(S_1/[G_1\|G']) \parallel L_m(S_2/[G_2\|G']) = K$ if and only if the specification K is conditionally controllable.*

Similarly as in monolithic supervisory control, if the specification fails to be conditionally controllable, the supremal conditionally controllable sublanguage is computed.

Theorem 15. *The supremal conditionally controllable sublanguage of a language K always exists and is equal to the union of all conditionally controllable sublanguages of K.*

We define languages

$$\begin{aligned} \sup \mathrm{C}_{1+\Sigma'} &= \sup \mathrm{C}(P_{1+\Sigma'}(K), L(G_1\|G')) \\ \sup \mathrm{C}_{2+\Sigma'} &= \sup \mathrm{C}(P_{2+\Sigma'}(K), L(G_2\|G')) \end{aligned} \tag{3.2}$$

where $\sup \mathrm{C}(K,L)$ denotes the supremal controllable sublanguage of K with respect to L.

Theorem 16. *Assume that the languages* $\sup \mathrm{C}_{1+\Sigma'}$ *and* $\sup \mathrm{C}_{2+\Sigma'}$ *are synchronously nonconflicting (e.g., prefix-closed). Then we have that* $\sup \mathrm{C}_{1+\Sigma'} \parallel \sup \mathrm{C}_{2+\Sigma'}$ *is controllable. If, in addition, the intersection* $P_{\Sigma'}(\sup \mathrm{C}_{1+\Sigma'}) \cap P_{\Sigma'}(\sup \mathrm{C}_{2+\Sigma'})$ *is controllable, then* $\sup \mathrm{C}_{1+\Sigma'} \parallel \sup \mathrm{C}_{2+\Sigma'}$ *is conditionally controllable.*

The previous theorem suggest the following comparison between the solutions of Problems 9 and 13.

Theorem 17. *The optimal solution of Problem 13 (the supremal conditionally controllable sublanguage) is included in the solution of Problem 9 (the distributed solution* $\sup \mathrm{C}_{1+\Sigma'} \parallel \sup \mathrm{C}_{2+\Sigma'}$*).*

We now discuss conditions that ensure optimality. To this end, we make use of the existing notions of output control consistency (OCC) [58] or local control consistency (LCC) [46, 45].

Theorem 18. *If the projection* $P_{i+\Sigma'}$ *is an* $L(G_1 \| G_2)$*-observer and OCC (LCC) for* $L(G_1 \| G_2)$*,* $i = 1,2$*, then the composition of supervisors contains the optimal solution, that is,* $\sup \mathrm{C}(K, L(G_1 \| G_2)) \subseteq \sup \mathrm{C}_{1+\Sigma'} \parallel \sup \mathrm{C}_{2+\Sigma'}$.

However, our aim is not to compute the composition of supervisors that we computed in parallel to obtain a single huge supervisor, but rather to distribute the supervision to local plants. Notice that we already have that

$$L_m(S_1/G_1 \| G') \parallel L_m(S_2/G_2 \| G') = L_m(S_1 \| S_2/G_1 \| G_2)\,.$$

But if the supervisors S_1 and S_2 are conflicting, we only have that

$$L(S_1/G_1 \| G') \parallel L(S_2/G_2 \| G') \supsetneq \overline{L_m(S_1 \| S_2/G_1 \| G_2)}$$

i.e., the overall supervised system is blocking. To solve nonblockingness, we construct the language $L_C =$

$$\sup \mathrm{C}(P_0(\sup \mathrm{C}_{1+\Sigma'}) \| P_0(\sup \mathrm{C}_{2+\Sigma'}), \overline{P_0(\sup \mathrm{C}_{1+\Sigma'})} \| \overline{P_0(\sup \mathrm{C}_{2+\Sigma'})}) \tag{3.3}$$

which serves as a coordinator for nonconflictingness, where the projection P_0 is a $\sup \mathrm{C}_{i+\Sigma'}$-observer, $i = 1,2$.

Theorem 19. *The language*

$$\overline{\sup \mathrm{C}_{1+\Sigma'} \parallel \sup \mathrm{C}_{2+\Sigma'} \parallel L_C} = \overline{\sup \mathrm{C}_{1+\Sigma'}} \parallel \overline{\sup \mathrm{C}_{2+\Sigma'}} \parallel \overline{L_C}$$

is nonconflicting (nonblocking) and controllable with respect to the plant $G_1 \| G_2$.

We can now summarize the method as an algorithm.

Theorem 20 (Solving Problem 9)**.**

1. *Verify whether there is a solution of Problem 9 using Theorem 11. If so, stop; otherwise, continue.*
2. *Compute* $\sup \mathrm{C}_{1+\Sigma'}$ *and* $\sup \mathrm{C}_{2+\Sigma'}$ *as defined in (3.2).*
3. *Let* $\Sigma_0 := \Sigma'$ *and* $P_0 := P_{\Sigma'}$.
4. *Extend the alphabet* Σ_0 *so that the projection* P_0 *is both a* $\sup \mathrm{C}_{1+\Sigma'}$*- and a* $\sup \mathrm{C}_{2+\Sigma'}$*-observer.*
5. *Define the coordinator C as the minimal nonblocking generator such that* $L_m(C) = L_C$ *from (3.3).*

We developed our approach for systems with perfect information [22, 19] as well as for systems with partial information [17] and implemented it in the software library libFAUDES [34].

Complexity of Verifying Nonblockingness

Nonblockingness is an important property of discrete event systems ensuring that every task can be completed. An automaton is *nonblocking* if every sequence of events generated by the automaton can be extended to a marked sequence. The property is easy to verify for deterministic automata.

Theorem 21. *Given a DFA* $\mathscr{A}$*, the problem whether* $\mathscr{A}$ *is nonblocking is* NL*-complete.*

If the automaton is nondeterministic, the verification becomes computationally more demanding.

Theorem 22. *Given an NFA* $\mathscr{A}$*, the problem whether* $\mathscr{A}$ *is nonblocking is* PSPACE*-complete.*

Given a set of nonblocking automata, the *modular nonblockingness problem* asks whether the parallel composition of all the automata of the set results in a nonblocking automaton.

The simplest case of the problem we consider is that there is no interaction among the subsystems.

Theorem 23. *Let J be a finite set, and let $\mathscr{A}_j$ be a nonblocking nondeterministic automata over Σ_j, for $j \in J$. If the alphabets are pairwise disjoint, then the parallel composition $\|_{j\in J}\mathscr{A}_j$ is nonblocking.*

In many complex systems, however, there are events that are shared among (some of) the subsystems, and then checking nonblockingness is in general PSPACE-complete [39]. A more fine-grained complexity can be distinguished based on the following criteria. Let $(\mathscr{A}_i)_{i=1}^n$ be deterministic automata:

1. The number of automata is not restricted;
2. The number of automata is restricted by a function $g(m)$, that is, $n \leq g(m)$, where m is the length of the encoding of the DFAs $\mathscr{A}_1, \mathscr{A}_2, \ldots, \mathscr{A}_n$;
3. The number of automata is restricted by a constant k, i.e., $n \leq k$.

We have shown the following result.

Theorem 24. *Given nonblocking DESs $(\mathscr{A}_i)_{i=1}^n$ with $\mathscr{A}_i$ over Σ_i, $2 \leq n \leq g(m)$, where m is the length of an encoding of the sequence of automata $\mathscr{A}_1, \mathscr{A}_2, \ldots, \mathscr{A}_n$. The problem whether $\|_{i=1}^n \mathscr{A}_i$ is nonblocking is* NSPACE$(g(m)\log m)$*-complete.*

Immediate consequences are, for instance, if k is a constant and $g(m) \leq k$ for every m, then the problem is NL-complete, or if $g(m) \leq \log^k m$, then the problem is NSPACE$(\log^{k+1} m)$-complete.

Despite the worst-case complexity, explicit model checking algorithms without any special data structures work well on standard computers for several practical systems with 100 million states [29].

We further studied the case where exactly one event is shared. An application of this case is, for instance, in the Brandin and Wonham [2] modular framework for timed discrete event systems, where only one event simulating the tick of a global clock is shared and all the other events are local [44]. Unless NP = PSPACE, nonblockingness is computationally easier in this case.

Theorem 25. *Given $n \geq 2$ nonblocking DESs $(\mathscr{A}_i)_{i=1}^n$ with $\mathscr{A}_i$ over Σ_i such that $|\cup_{i\neq j}(\Sigma_i \cap \Sigma_j)| = 1$. Deciding whether $\|_{i=1}^n \mathscr{A}_i$ is nonblocking is* NP*-complete.*

MRI Scanner Example

To demonstrate our approach, we consider the model and specification of an MRI scanner presented by Theunissen [51]. The plant consists of four parts

$$\text{VAxis} \parallel \text{HAxis} \parallel \text{HVNormal} \parallel \text{UI}.$$

The specification consists of the parts

$$\text{VReq} \parallel \text{HReq} \parallel \text{HVReq} \parallel \text{UIReq}$$

and is not conditionally decomposable with respect to the four alphabets of the four parts of the system.

VReq is a specification that concerns the plant VAxis. VReq has 12 states and 44 transitions, VAxis has 15 states and 50 transitions. A monolithic approach was used to compute a supervisor with 15 states and 36 transitions.

HReq is a specification that concerns the plant HAxis. HReq has 112 states and 736 transitions, HAxis has 128 states and 1002 transitions. A monolithic approach results in a supervisor with 80 states and 320 transitions.

The specification HVReq (7 states and 35 transitions) concerns the plant VAxis $\parallel$ HAxis $\parallel$ HVNormal. We computed a coordinator for safety consisting of 160 states and 1287 transitions and three supervisors with 516, 1132 and 283 states and 3395, 10298 and 1692 transitions, respectively. In comparison, the monolithic plant VAxis $\parallel$ HAxis $\parallel$ HVNormal has 1920 states and 23350 transitions.

The specification UIReq (256 states and 2336 transitions) concerns the whole plant VAxis $\parallel$ HAxis $\parallel$ HVNormal $\parallel$ UI. We computed a coordinator with 4 states and 30 transitions, and four supervisors with 432, 768, 12, and 96 states and 3488, 6652, 74 and 808 transitions, respectively. In comparison, the monolithic plant VAxis $\parallel$ HAxis $\parallel$ HVNormal $\parallel$ UI has 3840 states and 75500 transitions.

Altogether, while the minimal monolithic supervisor for the whole system consists of 68672 states and 616000 transitions, we have computed 9 supervisors with altogether 3334 states and 26763 transitions.

Systems without a Modular Structure

The abstraction of timed automata into region automata as well as the discretization of hybrid systems do not preserve the modular structure. The original structure of a system is thus lost and we have to face the decentralized supervisory control problem instead of the modular supervisory control problem.

Decentralized supervisory control was developed by Rudie and Wonham [40] based on the idea to distribute the actuator and sensor capabilities among several local supervisors. Each supervisor issues a control decision according to its observation and the global control action is given by a fusion rule of local control actions.

There are several different control policies based on two elementary ones: conjunctive and permissive (C & P) and disjunctive and antipermissive (D & A). For any decentralized control architecture, a corresponding notion of coobservability was proposed, which together with controllability form the necessary and sufficient conditions to achieve a specification by the controlled system.

Almost all results available in the literature so far are only existential. We showed how to compute a controllable and coobservable sublanguage using additional communications among the supervisors. Our study is motivated by the relationship between decentralized and modular supervisory control and their key concepts – coobservability and separability. We improve an existing approach based on the concurrent (separable) over-approximation where both the system and the specification are replaced with their infimal separable superlanguages. However, in the likely case the specification fails to be separable, the

existing approach only computes a solution for this new specification, which often fails to be included in the specification, and hence safety is not fulfilled [15]. In our work, we overcome this issue by making the specification separable via communication (using conditional decomposability).

Decentralized Supervisory Control

A *controlled generator* over an alphabet A is a structure

$$(G, (A_{c,i})_{i=1}^{n}, (A_{o,i})_{i=1}^{n})$$

where G is a generator over A, $A_{c,i} \subseteq A$ are sets of locally controllable events, and $A_{o,i} \subseteq A$ are sets of locally observable events. Let $A_c = \bigcup_{i=1}^{n} A_{c,i}$ denote the set of controllable events, $A_o = \bigcup_{i=1}^{n} A_{o,i}$ the set of observable events, $A_{uc} = A \setminus A_c$ the set of uncontrollable events, and $A_{uo} = A \setminus A_o$ the set of unobservable events. Projections to locally observable events $A_{o,i}$ are denoted by $P_{o,i} \colon A^* \to A_{o,i}^*$.

Let $\Gamma_i = \{X \subseteq A \mid X \supseteq (A \setminus A_{c,i})\}$ be a set of local control patterns. A supervisor S_i is a mapping $S_i \colon P_{o,i}(L(G)) \to \Gamma_i$, where $S_i(s)$ is the set of locally enabled events if S_i observes $s \in A_{o,i}^*$. The global control law S is the conjunction of local supervisors S_i given by

$$S(w) = \bigcap_{i=1}^{n} S_i(P_{o,i}(w))$$

for $w \in A^*$.

The control law of local supervisors associated to the C & P architecture is called permissive, since the default action is to enable an event whenever a supervisor has an ambiguity what to do with it. Specifically, the control law of supervisor S_i on s is defined as

$$S_i(s) = (A \setminus A_{c,i}) \cup \{a \in A_{c,i} \mid \exists s' \in K \text{ s.t. } P_{o,i}(s') = P_{o,i}(s) \;\&\; s'a \in K\}.$$

With the permissive local policy, we always achieve all words in the specification. The concern is then safety, expressed by coobservability.

The idea of our approach is to compute local languages (supervisors) $\mathscr{R}_i$ over B_i such that their synchronous product $\mathscr{R} = \|_{i=1}^{n} \mathscr{R}_i$ is a sublanguage of K controllable and coobservable with respect to L. Although there are well-known conditions on local languages in modular supervisory control that ensure that their synchronous product is controllable, conditions on local languages that ensure coobservability of their synchronous product were not known. We identify two such sufficient conditions.

Theorem 26. *Let L be a prefixed-closed language over $B = \bigcup_{i=1}^{n} B_i$ and assume that $B_{o,i} \cap B_c \subseteq B_{c,i}$, for $i = 1, \ldots, n$. Let $M \subseteq L$ be a language such that $\overline{M} = \|_{i=1}^{n} \overline{M_i}$, where M_i is a language over B_i. If*

1. *either M_i is normal with respect to $P_i(L)$ and $P_{o,i}^{i}$, for all $i = 1, \ldots, n$,*
2. *or $B_c \subseteq B_o$ and M_i is observable with respect to $P_i(L)$ and $P_{o,i}^{i}$, for all $i = 1, \ldots, n$,*

then M is coobservable with respect to L and $(B_{o,i})_{i=1}^{n}$.

The way we compute the languages $\mathscr{R}_i$ is as follows. We decompose specification K in such a way that $K = \|_{i=1}^{n} K_i$, where K_i are languages over B_i, and over-approximate L by the synchronous product of its projections $P_i(L)$ on alphabets B_i. The condition required on K does not always hold and is equivalent to the notion of separability. The languages $\mathscr{R}_i$ are then computed locally as sublanguages or superlanguages of K_i that satisfy the sufficient conditions that make their synchronous product $\mathscr{R}$ controllable, coobservable and included in K.

Consider the settings of decentralized control, and let $(\Sigma_i)_{i=1}^{n}$ be extensions of local alphabets $(A_{o,i})_{i=1}^{n}$, such that the specification K is separable with respect to $(A_{o,i} \cup \Sigma_i)_{i=1}^{n}$. This can be computed using conditional decomposability.

Assumption 27. For $i = 1, 2, \ldots, n$, let $P_{i+\Sigma_i}$ denote the projection from A^* to $(A_{o,i} \cup \Sigma_i)^*$. Let $\mathscr{R}_i$ be languages that are controllable with respect to projection $P_{i+\Sigma_i}(L)$ of the plant language L to alphabet $A_{o,i} \cup \Sigma_i$ and locally uncontrollable events $(A_{o,i} \cup \Sigma_i)_{uc} = (A_{o,i} \cup \Sigma_i) \cap$

A_{uc} such that their synchronous product $\mathscr{R} = \|_{i=1}^{n} \mathscr{R}_i$ is included in K. Furthermore, we assume that $\mathscr{R}_i$ are

1. either normal with respect to $P_{i+\Sigma_i}(L)$ and $A_{o,i} \cup \Sigma_{o,i}$,
2. or observable with respect to $P_{i+\Sigma_i}(L)$ and $A_{o,i} \cup \Sigma_{o,i}$, and all controllable events are observable ($A_c \subseteq A_o$).

We now state our main result showing how to use our framework to compute a controllable and coobservable sublanguage. An important feature of our computation is that we automatically obtain a coobservable sublanguage.

Theorem 28. *Consider Assumption 27. If the languages $\mathscr{R}_i$ are synchronously nonconflicting (in particular, if they are prefix-closed), then $\mathscr{R} = \|_{i=1}^{n} \mathscr{R}_i$ is a sublanguage of K controllable with respect to L and coobservable with respect to L and $(A_{o,i} \cup \Sigma_{o,i})_{i=1}^{n}$.*

There are many ways how to compute the languages $\mathscr{R}_i$. In the case of full local observations, it is natural to define the language

$$\mathscr{R}_i = \sup \mathrm{C}_{i+\Sigma_i} = \sup \mathrm{C}(P_{i+\Sigma_i}(K), P_{i+\Sigma_i}(L)) \tag{4.1}$$

as the supremal controllable sublanguage of $P_{i+\Sigma_i}(K)$. In the case of partial observations, we may define the language

$$\mathscr{R}_i = \sup \mathrm{CN}(P_{i+\Sigma_i}(K), P_{i+\Sigma_i}(L), (A_{o,i} \cup \Sigma_i)_{uc})$$

as the supremal controllable and normal sublanguage of $P_{i+\Sigma_i}(K)$ [6, 3]. Similarly, if $A_c \subseteq A_o$, we can define $\mathscr{R}_i$ as the supremal controllable and relatively observable sublanguage [5, 20], or we can use any of the methods to compute a controllable and observable sublanguage discussed in the literature [9, 50, 56]. In these cases, we have that $\mathscr{R}_i \subseteq P_{i+\Sigma_i}(K)$, and separability of K then implies that the synchronous product $\|_{i=1}^{n} \mathscr{R}_i$ is included in K as required.

However, we do not restrict language $\mathscr{R}_i$ to be included in $P_{i+\Sigma_i}(K)$. This allows us to define $\mathscr{R}_i$ in many different ways. For instance, we can define $\mathscr{R}_i$ as the infimal controllable (and normal/observable)

superlanguage of $P_{i+\Sigma_i}(K)$ as discussed in the literature [6, 24, 26, 41]. We can further combine the approaches so that one of the $\mathscr{R}_i$ can be computed as a sublanguage and another one as a superlanguage, etc. In such a case, we do not get the assumption $\|_{i=1}^n \mathscr{R}_i \subseteq K$ by construction, but we need to check it, which is in general a PSPACE-complete problem. On the other hand, the advantage it brings is a potentially better (larger) solution.

Another problem with infimal controllable superlanguages is that they do not exist for general languages, but only for prefix-closed languages. This issue can be avoided by the following choice of $\mathscr{R}_i$ based on the computation of prefix-closed superlanguages.

Lemma 29. *Let T_i be the prefix-closed infimal controllable (and normal/observable) superlanguage of $P_{i+\Sigma_i}(K)$. Then $\mathscr{R}_i = P_{i+\Sigma_i}(K) \cup [T_i \setminus \overline{P_{i+\Sigma_i}(K)}]$ is controllable (and normal/observable).*

We further discussed conditions under which the solution is optimal in the sense of maximal permissiveness. Since no centralized optimal solution exists in the case of partial observations, we restrict our attention to the case of full observations.

Theorem 30. *Let $K \subseteq L$ be prefix-closed languages, and let K and L be separable with respect to $(A_{o,i} \cup \Sigma_i)_{i=1}^n$. Let $\sup \mathrm{C}_{i+\Sigma_i}$ be defined in (4.1). If $P_{i+\Sigma_i}(L)$ and $P_{j+\Sigma_j}(L)$ are mutually controllable [27], for $i, j = 1, 2, \ldots, n$, then $\|_{i=1}^n \sup \mathrm{C}_{i+\Sigma_i} = \sup \mathrm{C}(K, L)$ is coobservable with respect to L and $(A_{o,i} \cup \Sigma_i)_{i=1}^n$.*

Depending on the system, mutual controllability may be a strong condition. We now present a result that ensures optimality and is based on the notions of an L-observer and local control consistency (LCC).

Theorem 31. *If every projection $P_{i+\Sigma_i}$ is an L-observer and LCC for L, and the languages $\sup \mathrm{C}_{i+\Sigma_i}$ defined in (4.1) are synchronously nonconflicting (e.g., prefix-closed), then the language $\|_{i=1}^n \sup \mathrm{C}_{i+\Sigma_i} = \sup \mathrm{C}(K, L)$ is coobservable.*

A similar result to Theorem 31 can be obtained for supremal controllable and normal sublanguages [17].

The theorems require that the local languages $\mathscr{R}_i$ are synchronously nonconflicting. The remaining question is thus the case of conflicting local supervisors.

Theorem 32. *Let $L_C \subseteq \|_{i=1}^n P_{\Sigma'}(\mathscr{R}_i)$ be a language that is controllable and normal (observable) with respect to $P_{\Sigma'}(L)$, Σ'_{uc} and Σ'_o, where $\Sigma' \subseteq A$ contains all events shared by any pair of $\mathscr{R}_i$ and $\mathscr{R}_j$, for $i \neq j$, and $P_{\Sigma'} : A^* \to \Sigma'^*$ is an $\mathscr{R}_i$-observer, for $i = 1, \ldots, n$. If $(A_{o,i} \cup \Sigma_{o,i} \cup \Sigma'_o) \cap A_c \subseteq (A_{c,i} \cup \Sigma_{c,i} \cup \Sigma'_c)$, then $\|_{i=1}^n (\mathscr{R}_i \parallel L_C)$ is a sublanguage of K controllable (and normal/observable) that is coobservable, and whose components are synchronously nonconflicting.*

In case of full observations, we strengthen the previous result by computing the language L_C as a sublanguage of $\|_{i=1}^n P_{\Sigma'}(\mathscr{R}_i)$ controllable with respect to $\|_{i=1}^n \overline{P_{\Sigma'}(\mathscr{R}_i)}$ rather than to $P_{\Sigma'}(L)$, which may result in a larger language L_C because $\|_{i=1}^n \overline{P_{\Sigma'}(\mathscr{R}_i)} \subseteq P_{\Sigma'}(L)$.

Theorem 33. *Let L_C be the supremal sublanguage of $\|_{i=1}^n P_{\Sigma'}(\mathscr{R}_i)$ that is controllable with respect to $\|_{i=1}^n \overline{P_{\Sigma'}(\mathscr{R}_i)}$ and Σ'_{uc}, where $\Sigma' \subseteq A$ contains all events shared by any pair of $\mathscr{R}_i$ and $\mathscr{R}_j$, for $i \neq j$, and $P_{\Sigma'} : A^* \to \Sigma'^*$ is an $\mathscr{R}_i$-observer, for $i = 1, \ldots, n$. If every projection $P_{i+\Sigma_i}$ is an L-observer and LCC for L, and the projection $P_{\Sigma'}$ is LCC for $\|_{i=1}^n \overline{\mathscr{R}_i}$, then $\|_{i=1}^n (\mathscr{R}_i \parallel L_C) = \sup \mathrm{C}(K, L)$ is coobservable with respect to L and $(A_{o,i} \cup \Sigma_i \cup \Sigma')_{i=1}^n$, whose components are synchronously nonconflicting. It is again under the assumption that $(A_{o,i} \cup \Sigma_i \cup \Sigma') \cap A_c \subseteq (A_{c,i} \cup \Sigma_{c,i} \cup \Sigma'_c)$.*

Controllability of Context-Free Systems

Sreenivas [49] has shown that controllability is undecidable for systems with undecidable inclusion (if equivalence is also undecidable). However, there are systems, such as deterministic pushdown automata, for which inclusion is undecidable and equivalence is decidable. For such systems, it is undecidable whether $K \subseteq L$, but if we have some additional knowledge that the languages are in the inclusion $K \subseteq L$, the results of Sreenivas cannot be used to show undecidability of controllability for two deterministic context-free languages. This problem was open in the literature [11].

We solved this problem by showing that although equivalence is decidable for deterministic context-free languages, controllability is undecidable even if the containment of the specification language in the plant language is known.

Theorem 34. *Controllability is undecidable for two deterministic (linear) context-free languages K and L, where L is prefix-closed, even if $K \subseteq L$.*

Because of these undecidability results, the only possibilities which deserve consideration in supervisory control of discrete event systems are:

1. the specification language K is regular and the plant language L is (deterministic) context-free, or
2. the specification language K is deterministic context-free and the plant language L is regular.

The later case has recently been treated in the literature [10, 11, 12, 47] while the former approach still waits for its investigation.

By the closure properties of regular and context-free languages, the language $\overline{K}\Sigma_u$ is regular, and $\overline{K}\Sigma_u \cap L$ is context-free. Since the controllability property $\overline{K}\Sigma_u \cap L \subseteq \overline{K}$ is equivalent to the emptiness problem $(\overline{K}\Sigma_u \cap L) \cap (\Sigma^* \setminus \overline{K}) = \emptyset$, and the language $(\overline{K}\Sigma_u \cap L) \cap (\Sigma^* \setminus \overline{K})$ is context-free, decidability of controllability follows from decidability of the emptiness problem for context-free languages, and hence controllability is decidable for the former case. The complexity still needs to be investigated.

These approaches are of interest because they present an opportunity to treat the most interesting problem of the current supervisory control theory of discrete event systems, the state-space explosion problem, so that we can describe the plant language or the specification language by a more concise representation which is up to exponentially smaller when using deterministic pushdown automata instead of finite-state machines.

Properties of Modular DESs

Detectability, opacity and fault diagnosability are important system-theoretic properties of discrete event systems. Detectability arises in the state estimation of DES [48]. Opacity is a property related to the privacy and security analysis of DES [13]. Fault diagnosis is another important task in DES. Several different notions of diagnosability have been proposed in the literature [57]. For example, the notion of diagnosability of Sampath et al. [43] requires that an occurrence of a fault can always be detected within a finite delay. Thorsley and Teneketzis [52] proposed a weaker version of diagnosability, called A-diagnosability. Compared to diagnosability, where the fault has to be detected on every path within a finite delay, A-diagnosability requires that there always exists a possibility to detect the fault event, and hence the probability of detection goes to one as the length of the trajectory increase.

Deciding weak detectability, opacity and A-diagnosability are PSPACE-complete problems for monolithic systems. However, their complexity was open for modular systems. We resolved these questions.

Modular Detectability

Let Σ be an alphabet, $\Sigma_o \subseteq \Sigma$ be the set of observable events, and P be the projection from Σ to Σ_o. Let $\mathbb{N}$ denote the set of all natural numbers. The set of infinite sequences of events generated by a DES G

is denoted by $L^{\omega}(G)$. For $w \in L^{\omega}(G)$, we denote the set of its prefixes by $Pr(w)$.

A DES $G = (Q, \Sigma, \delta, I)$ is strongly detectable with respect to Σ_{uo} if we can determine, after a finite number of observations, the current and subsequent states of the system for all trajectories of the system, that is,

$$(\exists n \in \mathbb{N})(\forall s \in L^{\omega}(G))(\forall t \in Pr(s))\left[|P(t)| > n \Rightarrow |R_G(t)| = 1\right]$$

where $R_G(t) = \{x \in Q \mid \exists t' \in L(G) \text{ such that } P(t) = P(t') \text{ and } x \in \delta(I, t')\}$. It is strongly periodically detectable with respect to Σ_{uo} if we can periodically determine the current state of the system for all trajectories of the system, that is,

$$(\exists n \in \mathbb{N})(\forall s \in L^{\omega}(G))(\forall t \in Pr(s))(\exists t' \in \Sigma^*) \left[tt' \in Pr(s) \wedge |P(t')| < n \wedge |R_G(tt')| = 1\right].$$

The modular version of the problems is defined as follows.

Definition 35. Given a set of discrete event systems $\{G_1, G_2, \ldots, G_n\}$ and a set of unobservable events Σ_{uo}. The strong (periodic) modular detectability problem asks whether the DES $G_1 \| G_2 \| \cdots \| G_n$ is strongly (periodically) detectable with respect to Σ_{uo}.

Deciding strong modular detectability is a PSPACE-hard problem. Our result improves this lower bound by showing that polynomial space is sufficient to solve the problem.

Theorem 36. *Deciding strong (periodic) modular detectability is a PSPACE-complete problem.*

Strong detectability requires that one can always determine the current state of the system unambiguously after a finite number of observations. A weaker version of detectability, weak detectability, requires that one can determine the current state of the system for some trajectory.

A DES $G = (Q, \Sigma, \delta, I)$ is weakly detectable with respect to Σ_{uo} if we can determine, after a finite number of observations, the current

and subsequent states of the system for some trajectories of the system, that is,

$$(\exists n \in \mathbb{N})(\exists s \in L^{\omega}(G))(\forall t \in Pr(s))\left[|P(t)| > n \Rightarrow |R_G(t)| = 1\right].$$

It is weakly periodically detectable with respect to Σ_{uo} if we can periodically determine the current state of the system for some trajectories of the system, that is,

$$(\exists n \in \mathbb{N})(\exists s \in L^{\omega}(G))(\forall t \in Pr(s))(\exists t' \in \Sigma^*) \\ [tt' \in Pr(s) \wedge |P(t')| < n \wedge |R_G(tt')| = 1].$$

Similarly as strong (periodic) modular detectability we define weak (periodic) modular detectability.

Definition 37. Given a set of discrete event systems $\{G_1, G_2, \ldots, G_n\}$ and a set of unobservable events Σ_{uo}. The weak (periodic) modular detectability problem asks whether the system $G_1 \| G_2 \| \cdots \| G_n$ is weakly (periodically) detectable with respect to Σ_{uo}.

Deciding weak modular detectability is a PSPACE-hard problem, however the complexity was left open. We showed that the problem requires exponential space.

Theorem 38. *Deciding weak (periodic) modular detectability is an EXPSPACE-complete problem.*

In some cases, the projection erases only private events. This, for instance, ensures that the projection commutes with parallel composition, that is, $P(L_m(G_1 \| G_2)) = P(L_m(G_1)) \parallel P(L_m(G_2))$. We showed that under this assumption, deciding weak modular detectability is a simpler problem.

Theorem 39. *Let $\{G_1, G_2, \ldots, G_n\}$ be a set of discrete event systems and $P\colon \Sigma \to \Sigma_o$ be a projection such that all shared events of any two systems are included in Σ_o. Then deciding weak (periodic) modular detectability is PSPACE-complete.*

Modular Opacity

Opacity is a property related to the privacy and security analysis of DES. The system has a secret modeled as a set of states and an intruder is modeled as a passive observer with limited observation. The system is opaque if the intruder never knows for sure that the system is in a secret state, i.e., the secret is not revealed.

A DES $G = (Q, \Sigma, \delta, I)$ is current-state opaque with respect to Σ_{uo} and a set of secret states $Q_S \subseteq Q$ if $(\forall s \in L(G))[R_G(s) \not\subseteq Q_S]$. The modular version of the problem is defined as follows.

Definition 40. Given a set of discrete event systems $\{G_1, G_2, \ldots, G_n\}$, a set of unobservable events Σ_{uo}, and a set of secret states Q_S. The modular opacity problem asks whether the system $G_1 \| G_2 \| \cdots \| G_n$ is opaque with respect to Σ_{uo} and Q_S.

Theorem 41. *Deciding modular opacity is an EXPSPACE-complete problem.*

Theorem 42. *Let $\{G_1, G_2, \ldots, G_n\}$ be a set of discrete event systems and $P\colon \Sigma \to \Sigma_o$ be a projection such that all shared events of any two systems are included in Σ_o. Then deciding modular opacity is PSPACE-complete.*

Modular A-Diagnosability

Let $\Sigma_F \subseteq \Sigma$ be a set of fault events, and let $L_F = \Sigma^* \Sigma_F \Sigma^*$ be the set of all trajectories that contain a fault event. The goal of fault diagnosis is to detect the occurrence of fault events. Several different notions of diagnosability have been proposed in the literature. For example, the diagnosability problem of Sampath et al. [43] requires that the occurrence of a fault can always be detected within a finite delay. This problem is polynomial-time decidable for monolithic systems and PSPACE-complete for modular systems.

Thorsley and Teneketzis [52] proposed a weaker version of diagnosability, A-diagnosability. Compared to diagnosability, where a fault has to be detected on every path after it happens within a finite

delay, A-diagnosability requires that for any path that contains a fault event, it always has an extension where a fault can be detected.

A DES $G = (Q, \Sigma, \delta, I)$ is A-diagnosable with respect to Σ_{uo} and Σ_F if for any fault trajectory, there exists an extension under which a fault event has occurred, that is,

$$(\forall s \in L(G) \cap L_F)(\exists t \in L(G)/s)[P^{-1}P(st) \cap L(G) \subseteq L_F],$$

where $L(G)/s = \{t \in \Sigma^* \mid st \in L(G)\}$. Intuitively, A-diagnosability says that after the occurrence of any fault, we can detect the occurrence of a fault with probability one.

The modular version of the problem is defined as follows.

Definition 43. Given a set of discrete event systems $\{G_1, G_2, \ldots, G_n\}$, a set of unobservable events Σ_{uo}, and a set of fault events Σ_F. The modular A-diagnosability problem asks whether the discrete event system $G_1 \| G_2 \| \cdots \| G_n$ is A-diagnosable with respect to Σ_{uo} and Σ_F.

Bertrand et al. [1] and Chen et al. [7] have shown that testing A-diagnosability is PSPACE-complete for monolithic systems. We showed that the problem is EXPSPACE-complete for modular systems.

Theorem 44. *Deciding modular A-diagnosability is EXPSPACE-complete.*

Theorem 45. *Let $\{G_1, G_2, \ldots, G_n\}$ be a set of discrete event systems and $P\colon \Sigma \to \Sigma_o$ be a projection such that all shared events of any two systems are included in Σ_o. Then deciding modular A-diagnosability is PSPACE-complete.*

Papers in the Collection

This thesis is a collection of papers listed below. It is common in the field to list the names of authors in the alphabetic order. The numbers correspond to the numbers in the bibliography.

[14] G. Jirásková and T. Masopust. On a structural property in the state complexity of projected regular languages. *Theoretical Computer Science*, 449:93–105, 2012.

[16] J. Komenda and T. Masopust. Computation of controllable and coobservable sublanguages in decentralized supervisory control via communication. *Discrete Event Dynamic Systems*, 27(4):585–608, 2017.

[17] J. Komenda, T. Masopust, and J. van Schuppen. Synthesis of controllable and normal sublanguages for discrete-event systems using a coordinator. *Systems & Control Letters*, 60(7):492–502, 2011.

[19] J. Komenda, T. Masopust, and J. van Schuppen. Coordination control of discrete-event systems revisited. *Discrete Event Dynamic Systems*, 25(1-2):65–94, 2015.

[21] J. Komenda, T. Masopust, and J. H. van Schuppen. On conditional decomposability. *Systems & Control Letters*, 61:1260–1268, 2012.

[22] J. Komenda, T. Masopust, and J. H. van Schuppen. Supervisory control synthesis of discrete-event systems using a coordination scheme. *Automatica*, 48(2):247–254, 2012.

[23] J. Komenda, T. Masopust, and J. H. van Schuppen. On a distributed computation of supervisors in modular supervisory control. In *Conference on Complex Systems Engineering (ICCSE)*, pages 1–6, 2015.

[30] T. Masopust. A note on controllability of deterministic context-free systems. *Automatica*, 48(8):1934–1937, 2012.

[31] T. Masopust. Complexity of infimal observable superlanguages. *IEEE Transactions on Automatic Control* 63(1), 249–254, 2018.

[32] T. Masopust. Complexity of verifying nonblockingness in modular supervisory control. *IEEE Transactions on Automatic Control* 63(2), 602–607, 2018.

[33] T. Masopust and X. Yin. Complexity of Detectability, Opacity and A-Diagnosability for Modular DES. 2017. Submitted manuscript, after revision.

Bibliography

[1] N. Bertrand, S. Haddad, and E. Lefaucheux. Foundation of diagnosis and predictability in probabilistic systems. In *FSTTCS*, volume 29 of *LIPIcs*, pages 417–429, 2014.

[2] B. Brandin and W. Wonham. Supervisory control of timed discrete-event systems. *IEEE Trans. Automat. Control*, 39:329–342, 1994.

[3] R. D. Brandt, V. Garg, R. Kumar, F. Lin, S. I. Marcus, and W. M. Wonham. Formulas for calculating supremal controllable and normal sublanguages. *Systems & Control Letters*, 15:111–117, 1990.

[4] H. J. Bravo, A. E. C. Da Cunha, P. Pena, R. Malik, and J. E. R. Cury. Generalised verification of the observer property in discrete event systems. In *WODES*, pages 337–342, 2012.

[5] K. Cai, R. Zhang, and W. M. Wonham. Relative observability of discrete-event systems and its supremal sublanguages. *IEEE Trans. Automat. Control*, 60:659–670, 2015.

[6] C. Cassandras and S. Lafortune. *Introduction to discrete event systems*. Springer, second edition, 2008.

[7] J. Chen, C. Keroglou, C. N. Hadjicostis, and R. Kumar. Revised test for stochastic diagnosability of discrete-event systems. *IEEE Transactions on Automation Science and Engineering*, 15:404–408, 2018.

[8] L. B. Clavijo and J. C. Basilio. Empirical studies in the size of diagnosers and verifiers for diagnosability analysis. *Discrete Event Dynamic Systems*, 27:701–739, 2017.

[9] J. Fa, X. Yang, and Y. Zheng. Formulas for a class of controllable and observable sublanguages larger than the supremal controllable and normal sublanguage. *Systems & Control Letters*, 20:11–18, 1993.

[10] C. Griffin. A note on deciding controllability in pushdown systems. *IEEE Trans. Automat. Control*, 51:334–337, 2006.

[11] C. Griffin. *Decidability and optimality in pushdown control systems: a new approach to discrete event control*. PhD thesis, Penn State University, 2007.

[12] C. Griffin. On partial observability in discrete event control with pushdown systems. In *ACC*, pages 2619–2622, 2010.

[13] R. Jacob, J.-J. Lesage, and J.-M. Faure. Overview of discrete event systems opacity: Models, validation, and quantification. *Annual Reviews in Control*, 41:135–146, 2016.

[14] G. Jirásková and T. Masopust. On a structural property in the state complexity of projected regular languages. *Theoretical Computer Science*, 449:93–105, 2012.

[15] J. Komenda, H. Marchand, and S. Pinchinat. A constructive and modular approach to decentralized supervisory control problems. In *DESDes*, volume 39, pages 111–116, 2006.

[16] J. Komenda and T. Masopust. Computation of controllable and coobservable sublanguages in decentralized supervisory control via communication. *Discrete Event Dynamic Systems*, 27:585–608, 2017.

[17] J. Komenda, T. Masopust, and J. van Schuppen. Synthesis of controllable and normal sublanguages for discrete-event systems using a coordinator. *Systems & Control Letters*, 60:492–502, 2011.

[18] J. Komenda, T. Masopust, and J. van Schuppen. On conditional decomposability. *Systems & Control Letters*, 61:1260–1268, 2012.

[19] J. Komenda, T. Masopust, and J. van Schuppen. Coordination control of discrete-event systems revisited. *Discrete Event Dynamic Systems*, 25:65–94, 2015.

[20] J. Komenda, T. Masopust, and J. van Schuppen. Relative observability in coordination control. In *IEEE CASE*, pages 75–80, 2015.

[21] J. Komenda, T. Masopust, and J. H. van Schuppen. On conditional decomposability. *Systems & Control Letters*, 61:1260–1268, 2012.

[22] J. Komenda, T. Masopust, and J. H. van Schuppen. Supervisory control synthesis of discrete-event systems using a coordination scheme. *Automatica*, 48:247–254, 2012.

[23] J. Komenda, T. Masopust, and J. H. van Schuppen. On a distributed computation of supervisors in modular supervisory control. In *ICCSE*, pages 1–6, 2015.

[24] R. Kumar and V. K. Garg. Optimal supervisory control of discrete event dynamical systems. *SIAM Journal on Control and Optimization*, 33:419–439, 1995.

[25] R. Kumar and M. A. Shayman. Formulae relating controllability, observability, and co-observability. *Automatica*, 34:211–215, 1998.

[26] S. Lafortune and E. Chen. The infimal closed controllable superlanguage and its application in supervisory control. *IEEE Trans. Automat. Control*, 35:398–405, 1990.

[27] S.-H. Lee and K. Wong. Structural decentralized control of concurrent discrete-event systems. *European Journal of Control*, 8:477–491, 2002.

[28] L. Lin, A. Stefanescu, R. Su, W. Wang, and A. Shehabinia. Towards decentralized synthesis: Decomposable sublanguage and joint observability problems. In *ACC*, pages 2047–2052, 2014.

[29] R. Malik. Programming a fast explicit conflict checker. In *WODES*, pages 438–443, 2016.

[30] T. Masopust. A note on controllability of deterministic context-free systems. *Automatica*, 48:1934–1937, 2012.

[31] T. Masopust. Complexity of infimal observable superlanguages. *IEEE Trans. Automat. Control*, 63:249–254, 2018.

[32] T. Masopust. Complexity of verifying nonblockingness in modular supervisory control. *IEEE Trans. Automat. Control*, 63:602–607, 2018.

[33] T. Masopust and X. Yin. Complexity of detectability, opacity and a-diagnosability for modular discrete event systems. Manuscript, 2017.

[34] T. Moor et al. libFAUDES – a discrete event systems library, 2012. [Online]. Available at `http://www.rt.eei.uni-erlangen.de/FGdes/faudes/`.

[35] A. Payne. Matrix-based algorithms and an analysis of system structure for partially-observable discrete-event systems. Master's thesis, Queen's University, Kingston, Ontario, Ca., 1997.

[36] P. Pena, J. Cury, and S. Lafortune. Polynomial-time verification of the observer property in abstractions. In *ACC*, pages 465–470, 2008.

[37] P. Ramadge and W. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal on Control and Optimization*, 25:206–230, 1987.

[38] K. Rohloff and S. Lafortune. On the computational complexity of the verification of modular discrete-event systems. In *CDC*, volume 1, pages 16–21, 2002.

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

[40] K. Rudie and W. Wonham. Think globally, act locally: Decentralized supervisory control. *IEEE Trans. Automat. Control*, 37:1692–1708, 1992.

[41] K. Rudie and W. M. Wonham. The infimal prefix-closed and observable superlanguage of given language. *Systems & Control Letters*, 15:361–371, 1990.

[42] K. Rudie and W. M. Wonham. The infimal prefix-closed and observable superlanguange of a given language. *Systems & Control Letters*, 15:361–371, 1990.

[43] M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete-event systems. *IEEE Trans. Automat. Control*, 40:1555–1575, 1995.

[44] G. Schafaschek, M. Queiroz, and J. Cury. Local modular supervisory control of timed discrete-event systems. In *WODES*, pages 271–277, 2014.

[45] K. Schmidt and C. Breindl. On maximal permissiveness of hierarchical and modular supervisory control approaches for discrete event systems. In *WODES*, pages 462–467, 2008.

[46] K. Schmidt and C. Breindl. Maximally permissive hierarchical control of decentralized discrete event systems. *IEEE Trans. Automat. Control*, 56:723–737, 2011.

[47] A. Schmuck, S. Schneider, J. Raisch, and U. Nestmann. Supervisory control synthesis for deterministic context free specification languages - enforcing controllability least restrictively. *Discrete Event Dynamic Systems*, 26:5–32, 2016.

[48] S. Shu, F. Lin, and H. Ying. Detectability of discrete event systems. *IEEE Trans. Automat. Control*, 52:2356–2359, 2007.

[49] R. Sreenivas. On a weaker notion of controllability of a language K with respect to a language L. *IEEE Trans. Automat. Control*, 38:1446–1447, 1993.

[50] S. Takai and T. Ushio. Effective computation of an $L_m(G)$-closed, controllable, and observable sublanguage arising in supervisory control. *Systems & Control Letters*, 49:191–200, 2003.

[51] R. Theunissen. *Supervisory Control in Health Care Systems*. PhD thesis, TU Eindhoven, 2015.

[52] D. Thorsley and D. Teneketzis. Diagnosability of stochastic discrete-event systems. *IEEE Trans. Automat. Control*, 50:476–492, 2005.

[53] J. N. Tsitsiklis. On the control of discrete-event dynamical systems. *MCSS*, 2:95–107, 1989.

[54] Y. Willner and M. Heymann. Supervisory control of concurrent discrete-event systems. *International Journal of Control*, 54:1143–1169, 1991.

[55] K. Wong. On the complexity of projections of discrete-event systems. In *WODES*, pages 201–206, 1998.

[56] X. Yin and S. Lafortune. Synthesis of maximally permissive supervisors for partially-observed discrete-event systems. *IEEE Trans. Automat. Control*, 61:1239–1254, 2016.

[57] J. Zaytoon and S. Lafortune. Overview of fault diagnosis methods for discrete event systems. *Annual Reviews in Control*, 37:308–320, 2013.

[58] H. Zhong and W. M. Wonham. On the consistency of hierarchical supervision in discrete-event systems. *IEEE Trans. Automat. Control*, 35:1125–1134, 1990.