Deciding Coobservability is PSPACE-complete

Kurt Rohloff, Tae-Sic Yoo and Stéphane Lafortune

Abstract— **In this paper we reduce the deterministic finite-state automata intersection problem to the problem of deciding coobservability of regular languages using a polynomial-time many-one mapping. This demonstrates that the problem of deciding coobservability for languages marked by deterministic finite-state automata is PSPACE-complete. We use a similar reduction to reduce the deterministic finite-state automata intersection problem to deciding other versions of coobservability introduced in [17]. These results imply that the coobservability of regular languages most likely cannot be decided in polynomial time unless we make further restrictions on the languages. These results also show that deciding decentralized supervisor existence is PSPACE-complete and therefore probably intractable.**

I. Introduction

There has been a lot of interest lately in concepts related to the computational complexity of various system- and control-theoretic properties in the field of discrete-event systems. See for example [5, 10, 12, 16]. Coobservability is an important property related to the existence of supervisors for decentralized discrete-event systems [2, 13]. Rudie and Willems [12] showed that we can decide coobservability for two-supervisor systems in polynomial time. This result can be scaled to the problem of deciding coobservability for a system where the number of supervisors is bounded (in other words, there is an upper limit on the number of supervisors), but no results are available in the literature for the more general case where the number of supervisors is unbounded. Building on the results in [12] and [13], we show that deciding coobservability is PSPACE-complete, which means that deciding coobservability is at least as difficult as all NP-complete problems. This paper also extends some work in [17] to show that deciding alternate versions of coobservability are similarly PSPACE-complete. We note that in [15] it is shown that a property similar to coobservability called "joint-observability" is undecidable even for simple cases.

In Section 2, we review the definition of coobservability and its relation to decentralized control. In Section 3, we present a brief review of concepts related to computational complexity. We present our main results in Section 4 where we show that deciding coobservability is PSPACE-complete. In Section 5 we extend our results to other properties and discuss decentralized supervisor existence. We conclude the paper with a brief discussion of the implications of our results.

II. The Property of Coobservability

The concept of coobservability, defined below, is central to the existence of decentralized supervisors for discrete-event systems (DES) [2, 13]. Coobservability captures the notion for decentralized control systems that if an illegal event is about to occur there is always a supervisor that knows to disable the illegal event and can disable the illegal event. Coobservability is part of the set of necessary and sufficient conditions for the existence of decentralized supervisors in all known results in the literature for various decentralized architectures. Different architectures have different versions of coobservability; the version of coobservability in Definition 1 is for the "conjunctive" architecture discussed in [13]. Later in the paper, in Section 5, we discuss other architectures and other versions of coobservability. We assume the reader is familiar with supervisory control theory. Please see Chapter 3 of [1] for background information.

Let Σ be the set of system events. Let $\Sigma_{ci} \subseteq \Sigma$ and $\Sigma_{oi} \subseteq \Sigma$ be the locally controllable and observable events, respectively, for the local supervisors $S_i, i \in \{1, ..., n\}$ for the system under consideration. Let $P_i : \Sigma^* \rightarrow \Sigma_{oi}^*$ be the corresponding natural projections that "erase" locally unobservable events (see [1] for a formal definition of the projection operation). Let $M = \overline{M} \subseteq \Sigma^*$ be a prefix-closed language representing the system behavior.

Definition 1: A language $K \subseteq \Sigma^*$ is *coobservable* with respect to M, P_i, and Σ_{ci}, $i = 1, ..., n$, if for all $t \in \overline{K}$ and for all $\sigma \in \Sigma_c = \cup_{i=1}^{n} \Sigma_{ci}$,

$\left(t\sigma \notin \overline{K}\right)$ and $(t\sigma \in M) \Rightarrow$

$\exists i \in \{1, ..., n\}$ such that $P_i^{-1}\left[P_i(t)\right]\sigma \cap \overline{K} = \emptyset$ and $\sigma \in \Sigma_{ci}$.

Coobservability is the decentralized generalization of "observability" [9] for control under partial observation. It is well known that observability can be decided in polynomial time [16]. Be aware that coobservability is not non-observability. In this paper we use the terminology of control theory even though it may be counter to naming conventions currently used in computer science theory.

In [12], there is a construction called the $\mathcal{M}$-machine for deciding the coobservability of languages specified by finite automata in the two supervisor case. For the sake of completeness we show this construction in the appendix. The $\mathcal{M}$-machine construction can be easily extended to the case of n supervisors. We use the n-supervisor construction later in this paper.

A well known result related to coobservability is that for a given DES modeled as a finite-state automaton $G = (X^G, x_o^G, \Sigma, \delta^G)$ and a specification also modeled as a finite-state automaton $H = (X^H, x_o^H, \Sigma, \delta^H)$ such that $\emptyset \neq \mathcal{L}(H) \subseteq \mathcal{L}(G)$, there exists a set of partial observation supervisors for G that can be realized as finite-state automata $S_i, i \in 1, ..., n$ and result in

$\mathcal{L}((S_1 \| ... \| S_n)/G) = \mathcal{L}(H)$

if and only if the following two conditions hold:

1. $\mathcal{L}(H)\Sigma_{uc} \cap \mathcal{L}(G) \subseteq \mathcal{L}(H)$ (controllability).
2. $\mathcal{L}(H)$ is coobservable with respect to $\mathcal{L}(G)$, P_1,...,P_n and Σ_{c1},...,Σ_{cn}.

This is a simplified version of the Controllability and Coobservability Theorem [13] for generated-language specifications. This result says that a set of supervisors exists that achieve a given non-trivial specification $\mathcal{L}(H) \subseteq \mathcal{L}(G)$ if and only if the system is controllable and coobservable. Therefore, deciding decentralized supervisor existence is at least as difficult as deciding coobservability. Note that controllability and language inclusion can be tested in polynomial time for deterministic machines using standard automata algorithms.

III. A Review of Computational Complexity

We present in this section a brief review of needed concepts from the theory of computation. For a more thorough exposition of these topics, the reader is encouraged to consult one of the standard texts in the field such as [3], [4] or [6].

Problems are said to be in class P if they can be solved in polynomial time by a deterministic computation device such as a deterministic Turing machine or a deterministic RAM machine. The exact type of computation device does not matter as long as it is a "reasonable" computation device. Similarly, problems are said to be in class NP if they can be solved in polynomial time by a nondeterministic computation device. The class PSPACE includes all problems that can be solved in a

This research was supported in part by NSF grant CCR-0082784.

Department of Electrical Engineering and Computer Science, The University of Michigan, 1301 Beal Ave., Ann Arbor, MI 48109-2122, USA {krohloff,tyoo,stephane}@eecs.umich.edu; www.eecs.umich.edu/umdes

polynomial amount of space by a deterministic computation device and the class NPSPACE includes all problems that can be solved in a polynomial amount of space by a nondeterministic computation device.

By Savitch's theorem [14] we know PSPACE=NPSPACE, but a similar result is not known for time-bounded computation. It is known that P $\subseteq$ NP $\subseteq$ PSPACE, but both of these inclusions are thought to be proper. Proving or disproving P$\neq$NP and NP $\neq$ PSPACE are major open problems in computer science.

Suppose we have a problem C such that $C \in$ PSPACE. We then know that $C' \in$ PSPACE where C' is the "complement" problem of C. Showing a system is not coobservable (non-coobservable) is the complement problem of showing that the system is coobservable.

We use the concept of a "polynomial-time many-one reduction" to denote that one problem is "more difficult" than another. For two problems $C \subseteq \Sigma_c^*$ and $D \subseteq \Sigma_d^*$, we say that there is a polynomial-time many-one reduction from C to D (denoted $C \leq_m^p D$) if there exists a polynomial-time computable function $f : \Sigma_c^* \rightarrow \Sigma_d^*$ such that for each $x \in \Sigma_c^*$, $x \in C$ if and only if $f(x) \in D$ [3]. Intuitively, it can be thought that if a polynomial-time many-one reduction exists, we can use the more difficult problem D to solve the easier problem C.

A different kind of polynomial time reduction based on Oracle Turing Machines (OTM's) is the polynomial time Turing reduction (denoted $\leq_T^p$) which is similar to the polynomial time many-one reduction described in the preceding paragraph. Although the distinction between many-one reductions and Turing reductions is beyond the scope of this paper, it suffices for us to state that a many-one reduction implies a Turing reduction (i.e., $A \leq_m^p B \Rightarrow A \leq_T^p B$). Readers interested in more information concerning details of Turing reductions should reference the texts mentioned in the beginning of this section.

Problem D is called PSPACE-complete if it is in PSPACE and all problems in PSPACE can be reduced to D using polynomial-time many-one reductions. PSPACE-complete problems are problems that are considered to be the "most difficult" of the problems in PSPACE and are at least as hard as all NP-complete problems. Showing a problem to be NP-complete or PSPACE-complete is generally considered good evidence that the problem is intractable. Given a PSPACE-complete problem A, if we can show for another problem B that $A \leq_T^p B$, then B is known to be PSPACE-hard by definition. Similar definitions also hold for NP-complete and NP-hard problem classes.

Suppose we are given a set of deterministic finite-state automata $\{A_1, A_2, ..., A_n\}$ with a common alphabet Σ^A such that for $i \in \{1, ..., n\}$, $A_i = (X^{A_i}, x_o^{A_i}, \Sigma^A, \delta^{A_i}, X_m^{A_i})$. Suppose also that $A_1 \| ... \| A_n$ represents the parallel composition of the automata $A_1, ..., A_n$ and that $\mathcal{L}_m(A_i)$ represents the language marked by the automaton A_i. Readers unfamiliar with this notation should consult the text [1]. Kozen [7] demonstrates that the problem of deciding if $\mathcal{L}_m(A_1 \| A_2 \| ... \| A_n) = \emptyset$ is PSPACE-complete. This problem is called the deterministic finite-state automata intersection problem and is referred to as "DFA-Int". This problem has also been discussed in [4], [8] and [10].

IV. The Complexity of Deciding Coobservability

As was mentioned earlier, it has been demonstrated in [12] that coobservability can be verified in polynomial-time if the number of supervisors is bounded. However, the deterministic algorithm demonstrated in [12] takes time exponential in the number of supervisors and no space analysis is given. We can show that verifying coobservability is PSPACE-complete by reducing the DFA-Int problem to a problem instance of deciding coobservability using a polynomial time many-one reduction.

Theorem 1: The DFA-Int problem can be reduced by a polynomial-time many-one reduction to the problem of verifying that $\mathcal{L}(H)$ is coobservable with respect to $\mathcal{L}(G)$, P_i, and Σ_{ci}, $i \in \{1, ..., n\}$ where G and H are deterministic finite-state automata.

Proof: Consider an instance of DFA-Int as presented earlier where we are given a set of deterministic automata $\{A_1, A_2, ..., A_n\}$ with a common alphabet Σ^A such that for $i \in \{1, ..., n\}$, $A_i = (X^{A_i}, x_o^{A_i}, \Sigma^A, \delta^{A_i}, X_m^{A_i})$.

We show how to construct automata G and H, projection P_i, and the set of controllable events Σ_{ci}, $i \in \{1, ..., n\}$ in polynomial-time from $\{A_1, A_2, ..., A_n\}$ such that $\mathcal{L}(H)$ is coobservable with respect to $\mathcal{L}(G)$, P_i, and Σ_{ci}, $i \in \{1, ..., n\}$ if and only if $\mathcal{L}_m(A_1 \| A_2 \| ... \| A_n) = \emptyset$.

Construct the automaton $A' = (\{x_o^{A'}\}, x_o^{A'}, \Sigma^A, \delta^{A'}, \{x_o^{A'}\})$ seen in Figure 1 where $\delta^{A'}(x, \sigma) = x_o^{A'}$ for all $\sigma \in \Sigma^A$. A' marks all strings in $(\Sigma^A)^*$.

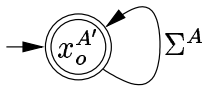

Fig. 1. The automaton A'.

Let us represent automaton A_i in a conceptual manner as seen in Figure 2 where only the initial state $x_o^{A_i}$ and marked states are shown. We do not show any more transitions or states in the automaton because this portrayal of A_i is used to present a simple graphical representation of the automaton's initial state and marked states. We assume that the marked states are reachable from the initial state, but we make no further assumptions on the structure of the automaton. The DFA-Int problem remains PSPACE-complete under this assumption.

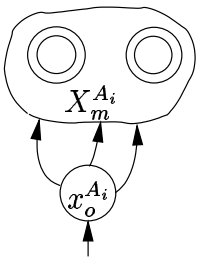

Fig. 2. A conceptual representation of A_i.

We use the automata $\{A_1, A_2, ..., A_n, A'\}$ to construct the automaton G as seen in Figure 3. For this purpose, let us define three new states x_o^G, *legal* and *illegal*. We define $G = (X^G, x_o^G, \Sigma, \delta^G)$ where $X^G = (X^{A_1} \cup ... \cup X^{A_n} \cup X^{A'} \cup \{x_o^G, legal, illegal\})$.

Let Σ^E be a set of symbols $\{e_1, ..., e_n, e', \gamma\}$ such that $\Sigma^E \cap \Sigma^A = \emptyset$. We define Σ to be $\Sigma^E \cup \Sigma^A$. Let $\Sigma_{oi} = \Sigma^G \backslash \{e_i, e'\}$ and define P_i to be the natural projection from Σ^G to Σ_{oi} for $i \in \{1, ..., n\}$. Also let $\Sigma_{c1} = \Sigma_{c2} = ... = \Sigma_{cn} = \{\gamma\}$.

The transition function $\delta^G : X^G \times \Sigma^G \rightarrow X^G$ is defined as

$$\delta^G(x,\sigma) = \begin{cases} x_o^{A_i} & \text{if } (x = x_o^G) \wedge (\sigma = e_i) \\ & \quad \text{for } i \in \{1,...,n\} \\ x_o^{A'} & \text{if } (x = x_o^G) \wedge (\sigma = e') \\ \delta^{A_i}(x,\sigma) & \text{if } (x \in X^{A_i}) \wedge (\sigma \in \Sigma^A) \\ & \quad \text{for } i \in \{1,...,n\} \\ \delta^{A'}(x,\sigma) & \text{if } (x \in X^{A'}) \wedge (\sigma \in \Sigma^A) \\ legal & \text{if } (x \in X_m^{A_i}) \wedge (\sigma = \gamma) \\ & \quad \text{for } i \in \{1,...,n\} \\ illegal & \text{if } (x \in X_m^{A'}) \wedge (\sigma = \gamma) \\ \text{undefined} & \text{otherwise} \end{cases}$$

The transition function of G is described as follows. From the initial state x_o^G, transitions lead to initial states $x_o^{A_i}$ or $x_o^{A'}$ on the occurrence of events e_i or e', respectively, for $i \in \{1,...,n\}$. Behavior inside the A_i and A' modules for $i \in \{1,...,n\}$ occur as normal except at the marked states. All marked states of the A_i modules have γ labelled transitions to the state *legal* for $i \in \{1,...,n\}$. The state of the A' module has a transition labelled γ to the state *illegal*.

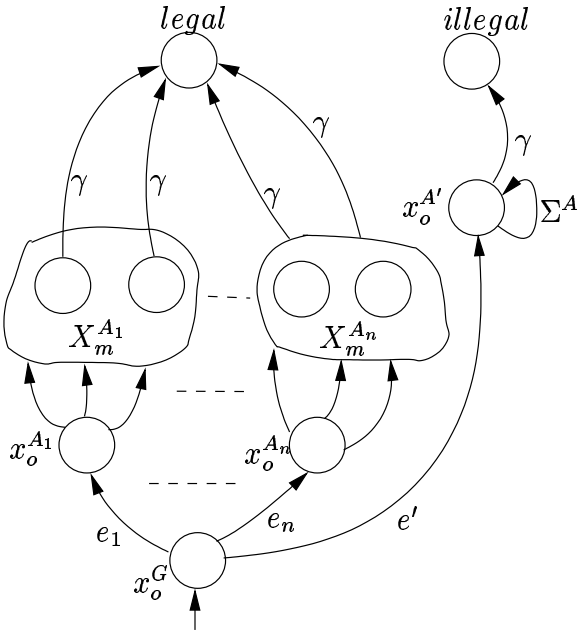

Fig. 3. A minimal representation of G.

We let H be the same as G except the state *illegal* is removed along with the transition leading to state *illegal*. It should be readily apparent that $\mathcal{L}(G)\backslash\mathcal{L}(H) = \{e'\}(\Sigma^A)^*\{\gamma\}$. We let all states of G and H be unmarked so we deal only with generated (i.e. prefix-closed) languages. This is consistent with the definition of coobservability.

As can be seen from Figure 3, event γ needs to be disabled before state *illegal* is entered. If there is always at least one supervisor that knows to disable γ before an illegal state is entered, then the system is coobservable. If at a given state γ leads to an illegal state, but no supervisor knows to disable it, the system is not coobservable. Suppose at a given state γ is legal, but no controller knows for sure if it should be enabled. Due to our definition of coobservability, the system would not become non-coobservable due to this case.

We now demonstrate that if there exists a string t marked by all automata $\{A_1, A_2, ..., A_n\}$ then $\mathcal{L}(H)$ is not coobservable with respect to $\mathcal{L}(G)$, P_i, and Σ_{ci}, $i \in \{1,...,n\}$.
$t \in \mathcal{L}_m(A_1\|...\|A_n)$
$\Rightarrow (t \in \mathcal{L}_m(A_1)) \wedge ... \wedge (t \in \mathcal{L}_m(A_n)) \wedge (t \in \mathcal{L}_m(A'))$
$\Rightarrow (e_1 t\gamma \in \mathcal{L}(H)) \wedge ... \wedge (e_n t\gamma \in \mathcal{L}(H))$
$\Rightarrow \left[P_1^{-1}(P_1(e_1 t))\gamma \cap \mathcal{L}(H) \neq \emptyset\right] \wedge$
$\quad ... \wedge \left[P_n^{-1}(P_n(e_n t))\gamma \cap \mathcal{L}(H) \neq \emptyset\right]$
$\quad \wedge \left[P_1^{-1}(P_1(e't)) = P_1^{-1}(P_1(e_1 t))\right] \wedge$
$\quad ... \wedge \left[P_n^{-1}(P_n(e't)) = P_n^{-1}(P_n(e_n t))\right]$
$\quad$ (because $\Sigma_{uoi} = \{e_i, e'\}$)
$\Rightarrow \left[P_1^{-1}(P_1(e't))\gamma \cap \mathcal{L}(H) \neq \emptyset\right] \wedge$
$\quad ... \wedge \left[P_n^{-1}(P_n(e't))\gamma \cap \mathcal{L}(H) \neq \emptyset\right]$
$\Rightarrow (\nexists i \in \{1,...,n\}) \left[\left(P_i^{-1}(P_i(e't))\gamma \cap \mathcal{L}(H) = \emptyset\right) \wedge (\gamma \in \Sigma_{ci})\right]$.

Therefore $\mathcal{L}_m(A_1\|...\|A_n) \neq \emptyset$ implies $\mathcal{L}(H)$ is not coobservable with respect to $\mathcal{L}(G)$, P_i, and Σ_{ci}, $i \in \{1,...,n\}$ since $(e't \in \mathcal{L}(H))$, $(e't\gamma \notin \mathcal{L}(H))$ and $(e't\gamma \in \mathcal{L}(G))$.

To prove the other direction, we assume that $\mathcal{L}(H)$ is not coobservable with respect to $\mathcal{L}(G)$, P_i, and Σ_{ci}, $i \in \{1,...,n\}$ and show that this implies $\mathcal{L}_m(A_1\|...\|A_n) \neq \emptyset$.

Suppose that string s is a string that violates the coobservability of $\mathcal{L}(H)$ with respect to $\mathcal{L}(G)$, P_i, and Σ_{ci}, $i \in \{1,...,n\}$. It must be that $(s \in \mathcal{L}(H))$, $(s\gamma \in \mathcal{L}(G))$ and $(s\gamma \notin \mathcal{L}(H))$ since $\Sigma_c = \{\gamma\}$. It should be apparent due to the construction of G and H that s must be of the form $e't$. From the violation of coobservability, we obtain that
$(\nexists i \in \{1,...,n\}) \left[\left(P_i^{-1}(P_i(e't))\gamma \cap \mathcal{L}(H) = \emptyset\right) \wedge (\gamma \in \Sigma_{ci})\right]$
$\Rightarrow \left[P_1^{-1}(P_1(e't))\gamma \cap \mathcal{L}(H) \neq \emptyset\right] \wedge$
$\quad ... \wedge \left[P_n^{-1}(P_n(e't))\gamma \cap \mathcal{L}(H) \neq \emptyset\right]$
We also know that:
$\left[P_1^{-1}(P_1(e't)) = P_1^{-1}(P_1(e_1 t))\right] \wedge$
$\quad ... \wedge \left[P_n^{-1}(P_n(e't)) = P_n^{-1}(P_n(e_n t))\right]$
$\Rightarrow [\{e_1 t\gamma\} \cap \mathcal{L}(H) \neq \emptyset] \wedge ... \wedge [\{e_n t\gamma\} \cap \mathcal{L}(H) \neq \emptyset]$
$\quad$ (from the structure of H)
$\Rightarrow (e_1 t\gamma \in \mathcal{L}(H)) \wedge ... \wedge (e_n t\gamma \in \mathcal{L}(H))$
$\Rightarrow (t \in \mathcal{L}_m(A_1)) \wedge ... \wedge (t \in \mathcal{L}_m(A_n))$
$\quad$ (from the structure of H)
$\Rightarrow t \in \mathcal{L}_m(A_1\|...\|A_n)$.

Therefore, using the reduction described above, an instance of DFA-Int is non-empty if and only if $\mathcal{L}(H)$ is not coobservable with respect to $\mathcal{L}(G)$, P_i, and Σ_{ci}, $i \in \{1,...,n\}$. It should be readily apparent that the construction described can be built in polynomial-time with respect to the size of the encodings of $\{A_1, A_2, ..., A_n\}$, so DFA-Int $\leq_m^p$ coobservability for languages encoded by deterministic finite-state automata. ■

We now show that verifying coobservability is in PSPACE using a nondeterministic path argument. This puts an upper bound on the computational complexity of deciding coobservability.

Proposition 1: The problem of verifying that $\mathcal{L}(H)$ is coobservable with respect to $\mathcal{L}(G)$, P_i, and Σ_{ci}, $i \in \{1,...,n\}$, is in PSPACE where H and G are deterministic finite-state automata.

Proof: It is sufficient to show that deciding non-coobservability for a system is in NPSPACE because PSPACE=NPSPACE [14]. We show an algorithm that is essentially a nondeterministic search over the reachable states of the $\mathcal{M}$-machine presented in [12] and the appendix for a marked state. If the $\mathcal{M}$-machine constructed from $\Sigma_{c1}, \Sigma_{o1}, ..., \Sigma_{cn}, \Sigma_{on}, G$ and H contains a reachable marked state, the system is not coobservable.

We start our search over the reachable states of the $\mathcal{M}$-machine at the initial $\mathcal{M}$-machine state and nondeterministically choose a string of state transitions in $\mathcal{M}$. We follow the path of states that the nondeterministic string of transitions

induces and if a marked state is ever reached by any possible string of transitions then the system is not coobservable.

To run this nondeterministic search we need to store in memory only one state of the $\mathcal{M}$-machine at a time if we update the $\mathcal{M}$-machine state as new transitions are nondeterministically chosen. To store a state of the $\mathcal{M}$-machine we need only store a state of G and $n+1$ states of H, as seen in [12]. It should be apparent that this nondeterministic search can be performed using a polynomial amount of space with respect to the encoding of the problem instance. Therefore, deciding non-coobservability is in NPSPACE, which completes the proof. ■

The result in Proposition 1 also holds if we use the non-prefix-closed marked language $\mathcal{L}_m(H)$ instead of the prefix-closed generated language $\mathcal{L}(H)$.

There is a method in [14] to convert a nondeterministic polynomial space-bounded computation such as the one described in the preceding proof to a deterministic polynomial space-bounded computation, so we know of a deterministic algorithm using polynomial space to decide coobservability.

We have already shown in Theorem 1 that the DFA-Int problem can be reduced by a polynomial-time many-one reduction to the problem of verifying the coobservability of languages specified by finite-state automata. DFA-Int is a known PSPACE-complete problem [7]. Proposition 1 shows that the problem of deciding if languages specified by finite-state automata are coobservable is in PSPACE. These three statements are sufficient to prove Corollary 1, our main result.

Corollary 1: The problem of deciding coobservability for languages encoded by deterministic finite-state automata is PSPACE-complete.

V. Further Results

We discuss in this section other versions of coobservability for the different decentralized control architectures presented in [17]. The concept we refer to as coobservability in this paper up to now is called "C&P coobservability" in [17]. C&P coobservability is a property relevant for systems with decentralized supervisors and fusion by union of locally *disabled* events. This captures the notion that one supervisor always knows to disable an event when needed. The dual property D&A coobservability is relevant for systems with decentralized supervisors and fusion by union of locally *enabled* events.

Definition 2: A language K is *D&A coobservable* with respect to M, P_i, and Σ_{ci}, $i = 1,...,n$ if for all $t \in \overline{K}$ and for all $\sigma \in \Sigma_c = \cup_{i=1}^{n} \Sigma_{ci}$, $t\sigma \in \overline{K} \Rightarrow$

$$(\exists i \in I)\left[\left[\left(P_i^{-1}(P_i(t)) \cap \overline{K}\right)\sigma \cap M \subseteq \overline{K}\right] \wedge [\sigma \in \Sigma_{ci}]\right].$$

In [17], a different type of coobservability that we call here *general coobservability* is also introduced that combines C&P coobservability and D&A coobservability. This is Definition 4 in [17]:

Definition 3: A language K is general coobservable w.r.t. M, P_i, Σ_{cdi} and Σ_{cei}, $i = 1,...,n$ if

1. K is C&P coobservable w.r.t. M, P_i and Σ_{cei}, $i = 1,...,n$.
2. K is D&A coobservable w.r.t. M, P_i and Σ_{cdi}, $i = 1,...,n$.

The methods used in the previous section to show deciding coobservability is PSPACE-complete can also be applied to these other decentralized observation properties.

To show D&A coobservability is in PSPACE, we can use a nondeterministic path argument similar to the one employed for C&P coobservability. To show deciding D&A coobservability is PSPACE-complete, we can use the same reduction as for C&P coobservability but swap the two states *legal* and *illegal* in the construction of G. The automaton H is still the same as G with state *illegal* removed. As the name implies, it should be readily apparent that general coobservability is more general than both C&P coobservability and D&A coobservability, but it is easy to show using previously discussed methods that deciding general coobservability is in PSPACE. Therefore, deciding general coobservability is also PSPACE-complete.

VI. Conclusion

Because deciding coobservability is PSPACE-complete and due to the Controllability and Coobservability Theorem mentioned above, deciding supervisor existence for a decentralized control system is also PSPACE-complete. This demonstrates that deciding decentralized supervisor existence is most likely intractable when the number of supervisors is not bounded. This result also holds for other decentralized control architectures that correspond to more general forms of coobservability.

Because of the controllability and coobservability theorem supervisor synthesis is at least as difficult computationally as deciding coobservability. Therefore, supervisor synthesis for decentralized systems is PSPACE-hard. Using similar reasoning we can also show that generating a maximal coobservable sublanguage is also PSPACE-hard.

The results shown here are particularly discouraging because if a problem is found to be PSPACE-complete or PSPACE-hard, this is generally considered good evidence that a polynomial time algorithm to solve this problem does not exist. A potential approach for future research in the area of decentralized control might be for online methods such as discussed in [11]. Another interesting area for future research might be to investigate special cases where we can make assumptions on the structure of the systems and specifications that might lead to an easing of the computational difficulty.

References

[1] C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems.* Kluwer Academic Publishers, Boston, MA, 1999.

[2] R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya. Supervisory control of discrete-event processes with partial observations. *IEEE Trans. Auto. Contr.*, 33(3):249–260, March 1988.

[3] D.Z. Du and K.I. Ko. *Theory of Computational Complexity.* John Wiley and Sons, Inc., 2000.

[4] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, 1979.

[5] P. Gohari and W.M. Wonham. On the complexity of supervisory control design in the RW framework. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 30(5):643–652, 2000.

[6] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison Wesley, Reading, MA, USA, 1979.

[7] D. Kozen. Lower bounds for natural proof systems. In *Proc. 18th Symp. on the Foundations of Computer Science*, pages 254–266, 1977.

[8] K.-J. Lange and P. Rossmanith. The emptiness problem for intersections of regular languages. In I. Havel, editor, *Proc. of the 17th Conf. on Mathematical Foundations of Computer Science, number 629 in LNCS*, pages 346–354. Springer-Verlag, 1992.

[9] F. Lin and W. M. Wonham. On observability of discrete-event systems. *Information Sciences*, 44:173–198, 1988.

[10] K. Rohloff and S. Lafortune. On the computational complexity of the verification of modular discrete-event systems. In *Proc. 41st IEEE Conf. on Decision and Control*, Las Vegas, Nevada, December 2002.

[11] K. Rohloff and S. Lafortune. n the synthesis of safe control policies in decentralized control of discrete event systems. *IEEE Trans. Auto. Contr.*, 48(6):1064–1068, 2003.

[12] K. Rudie and J.C. Willems. The computational complexity of decentralized discrete-event control problems. *IEEE Trans. Auto. Contr.*, 40(7):1313–1318, 1995.

[13] K. Rudie and W.M. Wonham. Think globally, act locally: Decentralized supervisory control. *IEEE Trans. Auto. Contr.*, 37(11):1692–1708, November 1992.

[14] W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal Comput. Sys. Sci.*, 4(2):177–192, 1970.

[15] S. Tripakis. Undecidable problems of decentralized observation and control. In *Proc. 40th IEEE Conf. on Decision and Control*, pages 4104–4109, 2001.

[16] J. Tsitsiklis. On the control of discrete-event dynamical systems. *Mathematics of Control, Signals and Systems*, 2:95–107, 1989.

[17] T.-S. Yoo and S. Lafortune. A general architecture for decentralized supervisory control of discrete-event systems. *Journal of Discrete Event Dynamical Systems: Theory and Applications*, 13(3):335–377, 2002.

Appendix

Suppose we are given a specification automaton $H = (X^H, x_0^H, \Sigma, \delta^H)$, a system automaton $G = (X^G, x_0^G, \Sigma, \delta^G)$, observable event sets Σ_{o1} and Σ_{o2} and controllable event sets Σ_{c1} and Σ_{c2} such that we wish to test if $\mathcal{L}(H)$ is coobservable with respect to $\mathcal{L}(G)$, Σ_{o1}, Σ_{o2}, Σ_{c1} and Σ_{c2}. This can be done using the $\mathcal{M}$-machine construction shown in [12]. Let us define:

$$\mathcal{M} = (X^{\mathcal{M}}, x_0^{\mathcal{M}}, \Sigma, \delta^{\mathcal{M}}, X_m^{\mathcal{M}})$$

where

$$\begin{aligned} X^{\mathcal{M}} &:= X^H \times X^H \times X^H \times G^G \cup \{d\}, \\ x_0^{\mathcal{M}} &:= (x_0^H, x_0^H, x_0^H, x_0^G), \\ X_m^{\mathcal{M}} &:= \{d\}. \end{aligned}$$

Let us define the set of conditions that together imply a violation of coobservability. Note that these conditions are only defined for the controllable events. For $\sigma \in \Sigma_c$[1], we call the following set of conditions the $(*)$ conditions.

$$\left.\begin{array}{ll} \delta^H(x_1,\sigma) & \text{is defined if } \sigma \in \Sigma_{c1} \\ \delta^H(x_2,\sigma) & \text{is defined if } \sigma \in \Sigma_{c2} \\ \delta^H(x_3,\sigma) & \text{is not defined} \\ \delta^G(x_4,\sigma) & \text{is defined} \end{array}\right\} \quad (*)$$

The transition relation $\delta^{\mathcal{M}}$ is defined as follows.
For $\sigma \notin \Sigma_{o1}$ and $\sigma \notin \Sigma_{o2}$,

$$\delta^{\mathcal{M}}((x_1,x_2,x_3,x_4),\sigma) = \begin{cases} (\delta^H(x_1,\sigma),x_2,x_3,x_4) \\ (x_1,\delta^H(x_2,\sigma),x_3,x_4) \\ (x_1,x_2,\delta^H(x_3,\sigma),\delta^G(x_4,\sigma)) \\ (\delta^H(x_1,\sigma),\delta^H(x_2,\sigma),\delta^H(x_3,\sigma),\delta^G(x_4,\sigma)) \\ d \quad \text{if } (*) \end{cases}$$

For $\sigma \notin \Sigma_{o1}$ and $\sigma \in \Sigma_{o2}$,

$$\delta^{\mathcal{M}}((x_1,x_2,x_3,x_4),\sigma) = \begin{cases} (\delta^H(x_1,\sigma),x_2,x_3,x_4) \\ (x_1,\delta^H(x_2,\sigma),\delta^H(x_3,\sigma),\delta^G(x_4,\sigma)) \\ (\delta^H(x_1,\sigma),\delta^H(x_2,\sigma),\delta^H(x_3,\sigma),\delta^G(x_4,\sigma)) \\ d \quad \text{if } (*) \end{cases}$$

For $\sigma \in \Sigma_{o1}$ and $\sigma \notin \Sigma_{o2}$,

$$\delta^{\mathcal{M}}((x_1,x_2,x_3,x_4),\sigma) = \begin{cases} (x_1,\delta^H(x_2,\sigma),x_3,x_4) \\ (\delta^H(x_1,\sigma),x_2,\delta^H(x_3,\sigma),\delta^G(x_4,\sigma)) \\ (\delta^H(x_1,\sigma),\delta^H(x_2,\sigma),\delta^H(x_3,\sigma),\delta^G(x_4,\sigma)) \\ d \quad \text{if } (*) \end{cases}$$

For $\sigma \in \Sigma_{o1}$ and $\sigma \in \Sigma_{o2}$,

$$\delta^{\mathcal{M}}((x_1,x_2,x_3,x_4),\sigma) = \begin{cases} (\delta^H(x_1,\sigma),\delta^H(x_2,\sigma),\delta^H(x_3,\sigma),\delta^G(x_4,\sigma)) \\ d \quad \text{if } (*) \end{cases}$$

For $\sigma \in \Sigma$, $\delta^{\mathcal{M}}(d,\sigma)$ is undefined.

The state d is reachable from the initial state in $\mathcal{M}$ if and only if $\mathcal{L}(H)$ is coobservable with respect to $\mathcal{L}(G)$, Σ_{o1}, Σ_{o2}, Σ_{c1} and Σ_{c2}.

[1] This condition is not mentioned in [12].