Decentralized Observation of Discrete-Event Systems: At Least One Can Tell

Stavros Tripakis, and Karen Rudie *Fellow, IEEE*

***Abstract*— In a truly decentralized system, agents must be able to make decisions based on their observations, without reliance on a centralized coordinator or fusion rule. A condition, called *at least one can tell*, is developed that characterizes when decentralized agents can determine whether behavior generated by a system is good (i.e., legal) or bad (i.e., illegal). It is shown that the *at least one can tell* condition is decidable and when the condition holds, finite-state decentralized observers can be constructed. Links between the *at least one can tell* condition and decentralized discrete-event systems are also made.**

***Index Terms*— Joint observability, discrete-event systems; supervisory control; decentralized observation**

I. Introduction

The move towards multi-agent, autonomous systems in daily living, including decentralized micro-grids for power, autonomous robots, self-driving cars, and smart small appliances and electronics, requires truly decentralized decision-making, obviating the need for a centralized decision point or fusion rule or coordinator. In this work we explore agents who make decentralized observations and we examine under what conditions the agents' decisions suffice to determine if some behavior is legal or not.

In particular, we introduce a new decentralized observation condition which we call *at least one can tell* and which attempts to capture the idea that for any possible behavior that a system can generate, at least one decentralized observation agent can tell whether that behavior was "good" or "bad", for given formal specifications of "good" and "bad". Specifically, we place ourselves in a *discrete-event system* (DES) setting [1], [5], [14] where systems, "good" behaviors, and "bad" behaviors, are all modeled as regular languages over finite alphabets. Decentralized observation is modeled by projection of words from the system alphabet onto subalphabets capturing the events observable by each decentralized agent.

We provide several equivalent formulations of the *at least one can tell* condition, and we relate it to (and show that it is different from) previously introduced *joint observability* [9], [10]. In fact, contrary to joint observability which is undecidable [9], [10], we show that the *at least one can tell* condition is decidable. We also show that when the condition holds, finite-state decentralized observers exist.

Some proofs are omitted due to lack of space. Omitted proofs are available in an extended version of this paper [12].

This work has been supported by NSF SaTC award CNS-1801546, by NSERC Discovery Grant RGPIN-2020-04279, and by the European Union's Horizon 2020 research and innovation program under grant agreement No 956123.

Stavros Tripakis is with the Khoury College of Computer Sciences, Northeastern University.

Karen Rudie is with the Department of Electrical and Computer Engineering and Ingenuity Labs Research Institute, Queen's University.

II. Background

The setting of this work is that of systems whose behavior can be thought of as sequences of actions or events, also referred to as *discrete-event systems* (DESs) and modelled as automata or languages. For more details on automata theory and languages, see the classic book by Hopcroft and Ullman [4] and for details on discrete-event systems the book by Wonham and Cai [14].

A. Preliminaries

A finite set of *letters* Σ is called an *alphabet*. The set of all finite sequences over Σ, also called *words*, is denoted by Σ^*. The empty word (i.e., the sequence of length 0) is denoted ϵ. Given a subalphabet $\Sigma_1 \subseteq \Sigma$, the *projection function from* Σ *onto* Σ_1 is the function $P_1 : \Sigma^* \to \Sigma_1^*$ that removes from words in Σ all letters except those in Σ_1. For example, if $\Sigma = \{a, b\}$ and $\Sigma_1 = \{a\}$, then $P_1(abbab) = aa$ and $P_1(bb) = \epsilon$. A *language* L over Σ is a set $L \subseteq \Sigma^*$. L will be used to denote the behavior of some system or process, where elements of Σ are events. In that context, the projection function will be used to capture observations of the behavior of that system or process. $pr(L)$ denotes the set of all *prefixes* of all words in L, e.g., $pr(\{ab, ba\}) = \{\epsilon, a, b, ab, ba\}$.

A *finite automaton over* Σ is a tuple $A = (\Sigma, Q, Q_0, F, \Delta)$, where Q is a finite set of *states*; $Q_0 \subseteq Q$ is a set of *initial* states; $F \subseteq Q$ is a set of *final* or *accepting* states; and $\Delta \subseteq Q \times \Sigma \times Q$ is the *transition relation*. When the set Q_0 is a singleton and the relation Δ can be represented as a function $\Delta : Q \times \Sigma \to Q$, we say that A is a *deterministic finite automaton* (DFA) and otherwise it is a *nondeterministic finite automaton* (NFA).

A *run* of A is a finite sequence of states and transitions $q_0 \xrightarrow{a_1} q_1 \cdots \xrightarrow{a_n} q_n$, for $n \geq 0$, such that $q_0 \in Q_0$ and $(q_i, a_{i+1}, q_{i+1}) \in \Delta$ for all $i = 0, ..., n-1$. The run is called *accepting* if $q_n \in F$. The run is said to *generate* the word $a_1 \cdots a_n$. A word $\rho \in \Sigma^*$ is said to be *accepted* or *recognized* by A if there is an accepting run of A generating ρ. The language accepted or recognized by A is the set of all words in Σ^* accepted by A. $L(A)$ denotes the language generated by A and $L_m(A)$ the language accepted by A.

Since projection can be used to capture the observations of a system, it is natural to ask whether the projection of a language recognized by a finite automaton A can also be recognized by a finite automaton. In fact, the answer is "yes". Informally the process is as follows. Let A be an automaton over Σ and let $\Sigma_1 \subseteq \Sigma$. We wish to find an automaton A' over Σ_1 that recognizes $P_1(L_m(A))$, where $P_1 : \Sigma^* \to \Sigma_1^*$, and for a language $L \subseteq \Sigma^*$, $P_1(L) = \{P_1(\rho) \mid \rho \in L\}$. We start by replacing all alphabet labels in A by ε, which results in an NFA A' with ε-transitions that recognizes $P_1(L_m(A))$. If A' is required to be deterministic, we can use the standard subset construction of [4] to convert an NFA with ε-transitions into a DFA.

The *inverse projection* $P_1^{-1} : \Sigma_1^* \to 2^{\Sigma^*}$ of a word returns all words whose projection could have yielded the original word. Inverse projection can be extended to languages:

$P_1^{-1}(L) = \{\rho \mid P_1(\rho) \in L\}$. Inverse projection allows us to speak about the words an agent thinks could have been produced based on the agent's observations. Inverse projection of a regular language can also be recognized by a finite automaton. For $\Sigma_1 \subseteq \Sigma$, let A be an automaton over Σ_1. To find an automaton that recognizes $P_1^{-1}(L_m(A))$, add self-loops of all events in $\Sigma \setminus \Sigma_1$ at all states of A.

B. Previous work: joint observability (JO)

Previous work [9], [10] introduced the following definition of *joint observability*:

Definition 1 (Joint Observability): Given alphabet Σ and subalphabets $\Sigma_i \subseteq \Sigma$, for $i = 1, ..., n$, and given regular languages $K \subseteq L \subseteq \Sigma^*$, K is called *jointly observable* if there exists a total function

$$f : \Sigma_1^* \times \cdots \times \Sigma_n^* \to \{0, 1\}$$

such that

$$\forall \rho \in L : \rho \in K \iff f(P_1(\rho), ..., P_n(\rho)) = 1$$

where $P_i : \Sigma^* \to \Sigma_i^*$ is the projection function onto Σ_i, for $i = 1, ..., n$.

In the definition above L models the plant, and K models the good behaviors of the plant. We want to know whether the behavior of the plant was good or bad. But we can't observe the behavior of the plant directly, so we have to rely on the decentralized projections. Joint observability says that whether a behavior is good can be completely determined based only on the decentralized observations of agents.

The following is a necessary and sufficient condition for joint observability:

Theorem 1 ([9], [10]): K is jointly observable iff

$$\nexists \rho, \rho' \in L : \rho \in K \wedge \rho' \in L - K \wedge \forall i \in \{1, ..., n\} : P_i(\rho) = P_i(\rho') \quad (1)$$

Informally, Condition (1) says that whether strings are good or not cannot possibly be determined if there are two strings—one good and one bad—that look the same to all agents.

We call Condition (1) the JO condition, or simply JO. It turns out that JO is undecidable [9], [10].

C. Previous work: co-observability

Previous work in DES [8] introduced the definition of *co-observability* to answer the following question: under what conditions do decentralized agents exist to guarantee that only the words in K are produced by some plant G.

Definition 2 (Co-observability): Given a plant G (where G is a DFA), a legal language K that is $L_m(G)$-closed (i.e., $pr(K) \cap L_m(G) = K$) and sets of events $\Sigma_{i,c}$ and $\Sigma_{i,o}$ representing the events that agent i ($i = 1, \ldots, n$) may control (through disablement) and observe, respectively, K is called *co-observable w.r.t. G* if

$$\forall s, s_1, s_2, \ldots, s_n \in \Sigma^* : \Big(\bigwedge_{i=1,\ldots,n} P_i(s) = P_i(s_i) \Big) \implies \Big(\forall \sigma \in \Sigma_c : s \in pr(K) \wedge s\sigma \in L(G) \bigwedge_{i \in N_\sigma} s_i\sigma \in pr(K) \implies s\sigma \in pr(K) \Big) \quad (2)$$

where $\Sigma_c = \bigcup_{i=1,\ldots,n} \Sigma_{i,c}$, and $N_\sigma = \{i \mid \sigma \in \Sigma_{i,c}\}$ is the set of the indices of all agents that can control σ.

If we look at the contrapositive of (2), then co-observability says that for every controllable event that would lead to an illegal word (i.e., $s\sigma \notin K$), an agent that *can* control that event ($i \in N_\sigma$) can determine from its observations that the event should be disabled (since at least one $s_i\sigma \notin K$). It turns out that decentralized agents exist if and only if K is controllable and co-observable w.r.t. G, where controllability captures the requirement that an event that cannot be controlled by any agent cannot lead to an illegal word.

III. Decentralized Observability: *At Least One Can Tell*

Even though JO has been used in [9], [10] as a stepping stone to showing the undecidability of decentralized supervisory control problems, JO itself is not really decentralized, because it requires a centralized decision point f. In this paper, we investigate "truly decentralized" observation conditions. We begin with a definition that tries to capture the *at least one can tell* property: namely, that there is no centralized decision point, but for every behavior of the plant, at least one of the decentralized observers can tell whether this behavior is good or bad, i.e., whether it belongs in K or not. We call this condition *OCT*.

We also present some equivalent formulations of OCT that are similar to conditions in the DES literature.

A. The OCT condition

Definition 3 (At Least One Can Tell): Given alphabet Σ and subalphabets $\Sigma_i \subseteq \Sigma$, for $i = 1, ..., n$, and given regular languages $K \subseteq L \subseteq \Sigma^*$, the *at least one can tell* condition (or simply OCT) is defined as follows:

$$\forall \rho \in L : \exists i \in \{1, ..., n\} : CanTell(i, \rho) \quad (3)$$

where $CanTell$ is defined as follows:

$$CanTell(i, \rho) = \rho \in K \implies \nexists \rho' \in L - K : P_i(\rho) = P_i(\rho') \quad (4)$$
$$\wedge$$
$$\rho \in L - K \implies \nexists \rho' \in K : P_i(\rho) = P_i(\rho') \quad (5)$$

Informally, the *at least one can tell* states that for any behavior ρ that can be generated by the system (i.e., $\rho \in L$), there exists at least one decentralized agent i that "can tell" whether ρ was a "good" behavior (i.e., $\rho \in K$) or a "bad" one (i.e., $\rho \in L - K$). Agent i "can tell" by looking at its local observation, i.e., at the projection of ρ, $P_i(\rho)$. In order for agent i to be sure, there should be no other system behavior ρ' which has exactly the same projection as ρ, yet does not belong to the same "good" or "bad" class as ρ.

In view of what follows, it is useful to state explicitly the negation of the OCT condition:

Lemma 1: The negation of OCT is equivalent to:

$$\Big(\exists \rho \in K : \forall i \in \{1, ..., n\} : \exists \rho_i \in L - K : P_i(\rho) = P_i(\rho_i)\Big) \quad (6)$$

$$\vee$$

$$\Big(\exists \rho \in L - K : \forall i \in \{1, ..., n\} : \exists \rho_i \in K : P_i(\rho) = P_i(\rho_i)\Big) \quad (7)$$

B. DES reformulations of OCT

This section explores the relationship between OCT and work in the supervisory control of discrete-event systems. We highlight the resemblance between our work on OCT which can be situated entirely within theoretical computing (i.e., automata theory and formal languages) without reference to control-theoretic properties and existing work in the supervisory control of DES community.

There are decentralized control conditions within the discrete-event systems literature that bear some resemblance to OCT but are not the same. Consider the following definition, which on first blush looks similar to a combination of *co-observability* [8] and *D&A co-observability* [15]. As we will see in Theorem 2,

it is actually decomposability [7] (which, informally, says that whether a string is legal can be determined from the observations of legal strings) plus a counterpart to decomposability that roughly says that whether a string is illegal can be determined from the observations of illegal strings.

Definition 4 (Discrete-Event Systems OCT): Given alphabet Σ and subalphabets $\Sigma_1, \Sigma_2, \ldots, \Sigma_n \subseteq \Sigma$, and given regular languages $K \subseteq L \subseteq \Sigma^*$, we say that *the discrete-event systems OCT (DESOCT) condition* holds if

$$\begin{aligned} &\forall s, s_1, s_2, \ldots, s_n \in \Sigma^* : \Big(\bigwedge_{i=1,\ldots,n} P_i(s) = P_i(s_i) \Big) \implies \\ &\Big(\big((\bigwedge_{i=1,\ldots,n} s_i \in K \wedge s \in L) \implies s \in K \big) \\ &\qquad \wedge \\ &\big((\bigwedge_{i=1,\ldots,n} s_i \in L - K \wedge s \in L) \implies s \notin K \big) \Big) \end{aligned} \quad (8)$$

where $P_i : \Sigma^* \to \Sigma_i^*$ is the projection function onto Σ_i, for $i = 1, ..., n$.

It can be shown, via logical transformations, that Definition 4 is equivalent to OCT.

Lemma 2: The DESOCT condition is equivalent to the OCT condition.

Definition 4 can be rewritten as a combination of language inclusions, which will make it immediately apparent that DESOCT is decidable for regular language inputs.

Definition 5 (Language OCT): Given alphabet Σ and subalphabets $\Sigma_1, \Sigma_2, \ldots, \Sigma_n \subseteq \Sigma$, and given regular languages $K \subseteq L \subseteq \Sigma^*$, we say that *the language OCT (LANGOCT) condition* holds if

$$\bigcap_{i=1,\ldots,n} P_i^{-1}(P_i(K)) \cap L \subseteq K \quad (9)$$

$$\wedge$$

$$\bigcap_{i=1,\ldots,n} P_i^{-1}(P_i(L - K)) \cap L \subseteq L - K \quad (10)$$

Theorem 2: The DESOCT condition is equivalent to the LANGOCT condition.

IV. Decidability, Computational Complexity, Finite Implementation

We now show that OCT is decidable for regular languages and we provide an asymptotic computational complexity analysis. We also show that, when the OCT condition is met, there exists a finite implementation.

A. OCT decidability and computational complexity

Using the formulation of OCT given by LANGOCT ((9) and (10)), we can demonstrate decidability of OCT. In addition, we will show that OCT can be decided in time $O(p^{n+2} \cdot m^{n+1})$ where n is the number of agents, and p and m are the number of states of the automata recognizing the languages L and K, respectively.

Theorem 3: If K and L are regular languages, OCT is decidable. Moreover, if K and L are recognized by DFAs whose state sets have cardinality m and p, respectively, then for n agents, OCT is decidable in time $O(p^{n+2} \cdot m^{n+1})$.

Proof: Decidability follows from Lemma 2, Theorem 2, and the fact that the operations of projection, inverse projection, intersection, and checking set containment are decidable for regular languages. Let us now examine the computational complexity. Let K be recognized by a DFA A_K with m states and let L be recognized by a DFA A_L with p states. We can compute a DFA A_{L-K} recognizing $L - K$ by noting that $L - K = L \cap \overline{K}$, where $\overline{K}$ denotes the complement of set K (unlike in the DES literature where the overbar notation is used to denote prefix-closure). Language $\overline{K}$ is recognized by a DFA $A_{\overline{K}}$ with exactly the same states as A_K, as it suffices to turn the accepting states of A_K into rejecting states and vice-versa. Then, A_{L-K} can be built as the product of A_L and $A_{\overline{K}}$ [4]. The number of states of A_{L-K} is $p \cdot m$.

Consider first condition (9). The language $P_i(K)$ is recognized by the same automaton that recognizes K but with all events not in Σ_i replaced by ε. This process can be completed in $O(m)$ time and the number of states of the resulting automaton is still m. The inverse projection $P_i^{-1}(P_i(K))$ is achieved by adding in self-loops of events not in Σ_i to the resulting automaton. This can be done in $O(m)$ time and the number of states of the resulting automaton is still m. The intersection of the n terms $P_i^{-1}(P_i(K))$, for $i = 1, \ldots n$, together with L can be done in $O(m^n \cdot p)$ time, since intersection of automata can be represented with an automaton whose state space is the Cartesian product of the constituent automata. The result is an NFA M_1 with $m^n \cdot p$

states, such that $L_m(M_1) = \bigcap_{i=1,\ldots,n} P_i^{-1}(P_i(K)) \cap L$. For condition (9) we need to check whether $L_m(M_1) \subseteq K$. This is equivalent to checking $L_m(M_1) \cap \overline{K} = \emptyset$, which in turn amounts to computing the product of M_1 with $A_{\overline{K}}$.[1] Therefore, the overall time complexity of checking condition (9) is $O(m^n \cdot p \cdot m) = O(m^{n+1} \cdot p)$.

Reasoning similarly, we can show that the overall complexity for checking condition (10) is $O\big((p \cdot m)^n \cdot p \cdot (p \cdot m)\big) = O(p^{n+2} \cdot m^{n+1})$. The details can be found in [12].

We can see that the complexity of condition (10) is higher than that of condition (9), therefore, the overall complexity of OCT is $O(p^{n+2} \cdot m^{n+1})$. ■

B. Functional characterization of OCT

Definition 3 captures our intuition about the *at least one can tell* property, namely, that at least one of the decentralized agents can be sure whether the behavior of the plant was good or bad. However, Definition 3 does not make explicit the existence of such decentralized observation agents. We rectify this by giving the definition that follows, which we then prove equivalent to OCT.

Definition 6 (Alternative OCT): We say that the *alternative OCT* (ALTOCT) condition holds iff there exist total functions $f_i : \Sigma_i^* \to \{Y, N, U\}$, such that

$$\begin{aligned}
&\big(\forall \rho \in L : \exists i \in \{1, ..., n\} : \\
&\qquad \rho \in K \implies f_i(P_i(\rho)) = Y \\
&\qquad\qquad \wedge \\
&\qquad \rho \in L - K \implies f_i(P_i(\rho)) = N\big) \\
&\qquad\qquad \wedge \qquad (11) \\
&\big(\forall \rho \in L : \forall i \in \{1, ..., n\} : \\
&\qquad f_i(P_i(\rho)) = Y \implies \rho \in K \\
&\qquad\qquad \wedge \\
&\qquad f_i(P_i(\rho)) = N \implies \rho \in L - K\big)
\end{aligned}$$

The value Y means that f_i knows that ρ was in K, N means that f_i knows that ρ was not in K, and U means f_i doesn't know. The bottom, $\forall \rho ... \forall i ...$ part says that no observer can "lie", namely, if it says Y then it's really the case that $\rho \in K$, and if it says N then it's really the case that $\rho \in L - K$. The top, $\forall \rho ... \exists i ...$ part says that at least one observer can tell.

Theorem 4: The ALTOCT condition is equivalent to the OCT condition.

Theorem 4 justifies the definition of OCT as a truly decentralized observation condition. Indeed, OCT holds if and only if decentralized functions satisfying condition (11) exist. In fact, we could equivalently have defined OCT to be the existence of functions satisfying condition (11). Then, Definition 3 could be seen as a necessary and sufficient condition for *at least one can tell* observability.

[1] It is well-known that checking subset inclusion for languages L_1 and L_2 can done in polynomial time when L_1 is represented by an NFA and L_2 by a DFA. A table in [3] provides a nice summary of the computational complexity of checking subset inclusion for various different automata representations of L_1 and L_2.

C. Finite-state OCT observers

The functions f_i in Theorem 4 can be seen as decentralized observers, each outputting Y, N, U depending on whether they can tell whether the original behavior ρ was in K, not in K, or unknown. Each of these functions takes as input the entire projection $P_i(\rho)$ which is generally unbounded in length. So a brute-force implementation of these functions requires unbounded memory. In this section, we show that we can also implement these functions using finite memory.

A *finite-state observer* is a DFA O_i over subalphabet Σ_i, such that the states of O_i are labeled by one of Y, N, U, corresponding to the three observation outcomes of function f_i. The requirement is that for every $\sigma \in \Sigma_i^*$, the label of the state that O_i ends up in after reading σ is exactly $f_i(\sigma)$. Note that O_i is deterministic, so for a given σ there is a unique state that O_i ends up in after reading σ.

Theorem 5: If OCT holds, then finite-state observers as above exist satisfying condition (11).

Proof: We build O_i as follows:

- Let $P_i(K)$ be the projection of regular language K onto Σ_i. $P_i(K)$ is a regular language. Let A_1 be a DFA recognizing $P_i(K)$. Without loss of generality we can assume that A_1 is complete, meaning there is a transition from every state of A_1 for every input letter.
- Similarly, let A_2 be a DFA recognizing $P_i(L - K)$.
- Both A_1, A_2 are DFA over the same alphabet Σ_i. Let A be the synchronous product of A_1 and A_2. Synchronous product means that each transition of A corresponds to a pair of transitions for each of A_1, A_2, labeled with the same letter.
- A state q of A is a pair of states (q_1, q_2), where q_1 is a state of A_1 and q_2 is a state of A_2. We label q as follows:
  - q is labeled with Y if q_1 is accepting and q_2 is rejecting.
  - q is labeled with N if q_1 is rejecting and q_2 is accepting.
  - q is labeled with U otherwise.

It can be shown that O_i as constructed above satisfies our requirements for a correct local observer. ■

We see in the construction in the proof of Theorem 5 that we produce DFA recognizing $P_i(K)$ and $P_i(L - K)$. While we have not proven that a polynomial-time algorithm for finite-state observers does not exist, we speculate that it does not since the first conjunct in the consequent of (8) is the same expression that appears in co-observability and producing finite-state supervisors for co-observable systems cannot be done in polynomial time [6]. The result from Theorem 3 and the proof from Theorem 5 are in keeping with results in the field of DES where, if the number of agents is fixed, verification of the necessary and sufficient conditions for supervisory control solutions to exist can be decided in polynomial time in the size of the state sets but where, even when the conditions are satisfied, the synthesis of corresponding supervisors cannot be done in polynomial time (cf., [13] for centralized control and [6] for decentralized control).

V. Relationship to Existing Work in Decentralized Observation and Control

In this section we highlight how OCT relates to and is different from existing concepts in decentralized observation (in particular, joint observability) and in decentralized control of DES (in particular, co-observability).

It is easy to show that OCT is a stronger condition than JO.

Theorem 6: OCT implies JO.

Proof: By contrapositive. Suppose JO doesn't hold. Then, by Theorem 1, there exist ρ, ρ' such that $\rho \in K$, $\rho' \in L - K$, and for all $i = 1, ..., n$, $P_i(\rho) = P_i(\rho')$. We will show that (6) holds. Indeed, this is done by setting ρ_i to ρ' for each $i = 1, ..., n$. By Lemma 1, (6) implies the negation of OCT. ■

The two conditions are *not* equivalent, however.

Theorem 7: JO does not imply OCT.

Proof: Consider the following example: $\Sigma = \{a, b\}$, $\Sigma_1 = \{a\}$, $\Sigma_2 = \{b\}$, $K = (ab)^*$ and $L = (ab)^*b^*$. We have two observers, and in this case JO holds: if the numbers of a's and b's are equal, we know that the word was in K, otherwise there are more b's than a's, and the word must have been in $L - K$. But the decentralized observers alone cannot tell: one observer sees a bunch of a's, the other a bunch of b's. There is no way of comparing the number of a's and b's (since there is no centralized decision point). So OCT shouldn't hold here. Indeed, take $\rho = abb$, $\rho_1 = ab$, and $\rho_2 = abab$. Note that $\rho \in L - K$, $\rho_1 \in K$ and $\rho_2 \in K$. We will show that (7) holds. Indeed, $P_1(\rho) = P_1(\rho_1) = a$, so observer 1 cannot tell. Similarly, $P_2(\rho) = P_2(\rho_2) = bb$, so observer 2 cannot tell. By Lemma 1, (7) implies that OCT does not hold. ■

The distinction between JO and OCT can also be seen by noticing that a counterexample to JO is a pair ρ, ρ', such that $P_i(\rho) = P_i(\rho')$ for all $i = 1, ..., n$ (Theorem 1), whereas as Lemma 1 indicates and as we saw in the proof of Theorem 7, a counterexample to OCT is a set of $n+1$ words, $\rho, \rho_1, ..., \rho_n$, such that for each $i = 1, ..., n$, we have $P_i(\rho) = P_i(\rho_i)$. Crucially, ρ' is the same word which must "match" ρ in every projection, whereas the words $\rho_1, ..., \rho_n$ need not be the same.

Joint observability and OCT are both conditions for whether decentralized agents—based on their observations—can determine if a word is in some prescribed language or not. Whereas JO indirectly requires a centralized decision point or fusion rule where information is combined, OCT does not. In this respect, we can see why OCT is stronger than JO: if even autonomously agents can determine whether a word is in some language, then *a fortiori* combining information (albeit unnecessary) would also allow them to determine whether a word is in the language.

Decentralized observation problems similar to the ones we examine here are also studied in [11]. In particular, the so-called *local* observation problems defined in [11] are similar in spirit to the ALTOCT condition, but with a crucial difference. In ALTOCT, the local decision functions f_i can each return three possible values, Y, N, U, whereas in the local observation problems defined in [11], the local decision functions f_i are only allowed two possible return values, 0 or 1. Another difference is that the local observation problems defined in [11] include a global combination function B which can be any Boolean function, whereas in our setting of OCT and ALTOCT, the combination is implicitly disjunction: if at least one local observer says Y then this is enough to ensure that the behavior ρ of the plant was good, and if at least one local observer says N then we can be sure that ρ was bad. (By definition, it is impossible to have the case where some observer says Y and another says N.)

Decentralized control problems under partial observation are investigated in [2], [8], [15]. For decentralized discrete-event systems problems that require control, problem solutions require that the agents' observations together with their control capabilities are enough to effect the necessary control. Co-observability and other variations (such as D&A co-observability [15]) differ from joint observability and the one can tell condition in the following way. Co-observability roughly says "based on what sequence of events has occurred so far, can decentralized supervisors know enough about an upcoming event to know whether to prevent it from occurring". Together with controllability, co-observability ensures that decentralized supervisory control problems can be solved because for any event that could lead somewhere illegal, at least one agent that *can* stop that event from occuring knows enough from the agent's observations to do so. In contrast, joint observability and the at least one can tell condition are divorced from a particular control problem at hand. Rather, they speak to whether a string that has already occurred is or is not in K and whether or not decentralized agents are able to determine that.

We confirm formally that, in fact, joint observability and *at least one can tell* are different from co-observability. We start by showing that neither JO nor co-observability is a condition stronger than the other one.

Theorem 8: JO does not imply co-observability.

Proof: Consider the following example: $\Sigma = \{a, b\}$, $\Sigma_1 = \{a\}$, $\Sigma_2 = \{b\}$, $L = aa + ba + aba$ and $K = aba$.

Consider a plant G that recognizes L (and generates $pr(L)$) and assume that K is the legal language. To see that K is not co-observable w.r.t. G: suppose that $\Sigma_{1,c} = \Sigma_{2,c} = \{a, b\}$. Take $s = a$, $s' = ab$, and $s'' = \varepsilon$. Then we have $P_1(s) = P_1(s')$, $P_2(s) = P_2(s'')$, $s'\sigma \in pr(K)$, $s''\sigma \in pr(K)$, but $s\sigma \notin pr(K)$. That is, if string a occurs, agent 1 will not know if a or ab occurred and aba is legal (i.e., in $pr(K)$) but aa is not legal so agent 1 will *not* know whether or not to disable a. Similarly, agent 2 will not know if a occurred or no events occurred and again, since aa is illegal but a is legal (since it's in $pr(K)$), agent 2 will also not know whether or not to disable an upcoming a.

However, the system *is* JO: since L has only three words, one can easily compare all possible pairs of words and there is no pair ρ, ρ' where $P_1(\rho) = P_1(\rho')$ and $P_2(\rho) = P_2(\rho')$. Joint observability can also be seen this way: if agent 1 sees two a's *and* agent 2 sees one b then fusing that information leads one to know that aba *must* have occurred and hence is in K; any other combination of observations by the agents means a word is in $L - K$. Once again, JO requires a centralized decision point, in this case the comparison of the number of a's seen by agent 1 with the number of b's seen by agent 2.

■

Theorem 9: Co-observability does not imply JO.

Proof: Consider the following example: $\Sigma = \{a, b\}$, $\Sigma_1 = \{a\}$, $\Sigma_2 = \{b\}$, $L = b^*a^*b^*$ and $K = pr(ab)$

To see that K is not JO, take $\rho = ab$, $\rho' = ba$. Then $\rho \in K$, $\rho' \in L - K$, $P_1(\rho) = a = P_1(\rho')$ and $P_2(\rho) = b = P_2(\rho')$, which is a counterexample to joint observability.

Consider a plant G that recognizes L (and generates $pr(L)$) and a legal language K. Suppose that $\Sigma_{1,c} = \Sigma_{2,c} = \{a, b\}$. Since there are no uncontrollable events, K is trivially controllable. Moreover, it can easily be shown that K is co-observable w.r.t. G. Co-observability is equivalent to the existence of decentralized agents that guarantee that only words in $pr(K)$ are generated. The following agents work: after agent 1 sees an a, it disables a and after agent 2 sees a b, it disables b. ■

From Theorems 6 and 9, we have the following result.

Corollary 1: Co-observability does not imply OCT.

We know from Theorem 6 that OCT is stronger than JO so it is natural to ask whether it is also stronger than co-observability. As we see below, the answer is "no".

Theorem 10: OCT does not imply co-observability.

Proof: Consider the following example: $\Sigma = \{a, b\}$, $\Sigma_1 = \{a\}$, $\Sigma_2 = \{b\}$, $L = ba + aba$ and $K = aba$.

It is easy to see that OCT holds since all strings in K have have two a's and all illegal strings have one a. So agent 1 can always tell whether a string is in K or not.

Consider a plant G that recognizes L (and generates $pr(L)$) and legal language K. Let $\Sigma_{1,c} = \{b\}$ and $\Sigma_{2,c} = \{a\}$. The system is controllable since there are no uncontrollable events. However, the system is not co-observable since after $s = ab$ occurs, agent 2—which is the only agent that can possibly disable a—will not know if a should be disabled since $P_2(ab) = P_2(b) = b$ and $aba \in pr(K)$ but $ba \notin pr(K)$. So agent 2 upon seeing a single b cannot know if ab or b occurred and hence cannot know whether to disable the upcoming a. ■

Co-observability is a condition for determining whether agents know enough about a system to make control decisions. So it is not surprising that whether or not a plant can be controlled to ensure that only words in K are generated will not necessarily affect whether, if that same plant were allowed to evolve without disruption, agents could determine if a string that plant generates is in K or not. Thus it is not surprising that Theorems 8, 9, and 10 and Corollary 1 confirm OCT/JO and co-observability are not comparable in terms of one being stronger than the other.

VI. Conclusions

We have shown that checking decentralized observability amounts to ensuring that at least one agent can tell if a word is legal or not. This condition is decidable for regular languages and if it is satisifed then finite-state observers can be constructed (albeit not likely in polynomial time).

VII. Acknowledgements

We thank Martijn Goorden who observed that in an example presented in a lecture by Tripakis, there doesn't seem to be a way for *at least one observer to tell*, yet joint observability holds. This motivated us to come up with the novel definition of OCT which is different from JO. We also thank Richard Ean for finding an error in an earlier version of the example that shows that joint observability does not imply co-observability.

References

[1] C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Springer, second edition, 2008.

[2] R. Cieslak, C. Desclaux, A.S. Fawaz, and P. Varaiya. Supervisory Control of Discrete-Event Processes with Partial Observation. *IEEE Transactions on Automatic Control*, 33(3):249–260, 1988.

[3] L. Clemente. On the complexity of the universality and inclusion problems for unambiguous context-free grammars (technical report). In *VPT/HCVS@ETAPS*, 2020.

[4] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[5] P. Ramadge and W. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, January 1989.

[6] K. Rudie and J.C. Willems. The Computational Complexity of Decentralized Discrete-Event Control Problems. *IEEE Transactions on Automatic Control*, 40(7):1313–1319, 1995.

[7] K. Rudie and W.M. Wonham. Supervisory Control of Communicating Processes. In L. Logrippo, R.L. Probert and H. Ural, editor, *Proceedings of the IFIP WG6.1 Tenth International Symposium on Protocol Specification, Testing and Verification X*, pages 243–257, 1990.

[8] K. Rudie and W.M. Wonham. Think Globally, Act Locally: Decentralized Supervisory Control. *IEEE Transactions on Automatic Control*, 37(11):1692–1708, 1992.

[9] S. Tripakis. Undecidable Problems of Decentralized Observation and Control. In *40th IEEE Conference on Decision and Control (CDC'01)*, pages 4104–4109. IEEE Computer Society, December 2001.

[10] S. Tripakis. Undecidable Problems of Decentralized Observation and Control on Regular Languages. *Information Processing Letters*, 90(1):21–28, April 2004.

[11] S. Tripakis. Decentralized Observation Problems. In *44th IEEE Conference on Decision and Control and 2005 European Control Conference (CDC-ECC'05)*, pages 6–11. IEEE Computer Society, December 2005.

[12] S. Tripakis and K. Rudie. Decentralized observation of discrete-event systems: At least one can tell. arXiv eprint 2108.04523, 2021.

[13] J.N. Tsitsiklis. On the Control of Discrete-Event Dynamical Systems. *Mathematics of Control, Signals, and Systems*, 2:95–107, 1989.

[14] W.M. Wonham and K. Cai. *Supervisory Control of Discrete-Event Systems*. Springer International Publishing, 2019.

[15] T.-S. Yoo and S. Lafortune. A Generalized Architecture for Decentralized Supervisory Control of Discrete-Event Systems. *Journal of Discrete Event Dynamic Systems*, 12(3):335–377, 2002.