



**HAL**  
open science

# Verification of Inference Observability in Supervisory Control of Discrete-Event Systems

C K Sharpe, Laurie S. L. Ricker, Hervé Marchand

► **To cite this version:**

C K Sharpe, Laurie S. L. Ricker, Hervé Marchand. Verification of Inference Observability in Supervisory Control of Discrete-Event Systems. 2023. hal-04054414

**HAL Id: hal-04054414**

**<https://inria.hal.science/hal-04054414>**

Preprint submitted on 31 Mar 2023

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Verification of Inference Observability in Supervisory Control of Discrete-Event Systems

C.K. Sharpe, S.L. Ricker and H. Marchand

**Abstract**—We present a new algorithm to verify inference observability in supervisory control of decentralized discrete-event systems. The algorithm’s success relies on the following idea. An inferencing solution exists only when a language does not contain any inferencing cycles.

## I. INTRODUCTION

Decentralized decision-making in supervisory control can involve direct knowledge of the correct control decision [1] or a one-level inference about another agent’s direct knowledge [2] or multi-level inferencing involving multiple rounds of inferencing among the agents [3]. There are two decision-making architectures: *enable by default* (EBD), where the focus is on agents who can make the correct “disable” decision, and *disable by default* (DBD), which is concerned with agents who can make the correct “enable” decision. As noted in [2], these architectures are incomparable. However, a general decision-making framework that combined the two in the context of diagnosis was presented in [4].

The inferencing definition introduced in [3] assumes the existence of two non-increasing sequences of languages defined over the set of controllable events. The first consists of words of a regular language, after which a disablement decision must occur for a given controllable event. And the other has words after which an enablement decision must take place for the same given controllable event. To have an  $N$ -level inferencing solution, at least one of these sequences of languages must eventually be non-empty in  $N + 1$  steps. When applied to the diagnosis domain, the definition refers to two language sequences: failure sequences and non-failure sequences containing elements of a minimal length.

The verification algorithm of [4] asserts whether a pair of languages is  $N$ -inference diagnosable, where  $N$  is given. The most significant difference between the decentralized control and diagnosis domains is that we must verify the decentralized properties of interest over words of a specified minimum length for diagnosis. The computational complexity of the verification strategy of [4] is not affected by this length requirement but does depend on the number of agents.

We are interested in the verification of inferencing in the control domain. In contrast, our algorithm asserts whether a system is inference observable and, if it is, provides the value  $N$ . We rely on a characteristic of inferencing that is present when a system is not inference observable. Namely,

there are cycles of inferencing that occur. We revisit a finite structure on which we have verified decentralized properties to identify relevant states involved in control decisions and determine whether any give rise to inferencing cycles.

The paper is organized as follows. First, Section II establishes the notation and background on inferencing in supervisory control. Then in Section III, we present the intuition behind our algorithm and review the finite structure we use to determine if a system has an inferencing solution. Next, we present our algorithm in Section IV and demonstrate it on two examples, one that does not have an inferencing solution and one that does. Finally, a brief discussion concludes the paper in Section V.

## II. NOTATION

Let  $\Sigma$  be a finite alphabet, and  $\Sigma^*$  be the set of words formed from elements of  $\Sigma$ , where  $\epsilon$  is the empty word. The length of word  $w \in \Sigma^*$  is denoted by  $|w|$ . A language  $L \subseteq \Sigma^*$  is an arbitrary set of words over alphabet  $\Sigma$ .

Automata recognize words of a language, and an automaton is finite if it has a finite number of states. A finite deterministic automaton is a 5-tuple  $\mathcal{M}_L = (Q, \Sigma, T_L, q_0, F)$ , where  $Q$  is a finite set of states,  $\Sigma$  is the alphabet,  $q_0 \in Q$  is the initial state, and  $F \subseteq Q$  is a set of final states, and  $T_L \subseteq Q \times \Sigma^* \times Q$  is a transition relation closed under concatenation. Further, since  $T$  is deterministic,  $\forall q_i, q_j, q_k \in Q$  and  $\forall \sigma \in \Sigma$ , if  $(q_i, \sigma, q_j) \in T_L$  and  $(q_i, \sigma, q_k) \in T_L$  then  $q_j = q_k$ . We say that a word  $w$  is accepted by  $\mathcal{M}_L$  if  $(q_0, w, q_k) \in T_L$  and  $q_k \in F$ . The regular language recognized by  $\mathcal{M}_L$  is the set of words accepted by  $\mathcal{M}_L$  and is denoted by  $L$ . An automaton  $\mathcal{M}_K = (Q_K, \Sigma, T_K, q_{0,K}, F_K)$  that recognizes regular language  $K$  is a sub-automaton of  $\mathcal{M}_L$  and we write  $\mathcal{M}_K \sqsubseteq \mathcal{M}_L$  if  $Q_K \subseteq Q$ , and  $T_K \subseteq T_L$ , and  $F_K \subseteq F$ , and  $q_{0,K} = q_0$ . Subsequently,  $K \subseteq L$ .

Throughout this work, we will be considering only prefix-closed languages. A language  $L \subseteq \Sigma^*$  is *prefix closed* if for all words  $w \in L$  and any words  $u, v$  satisfying  $w = uv$ , we have that  $u \in L$ . Thus, all states are final states, and we will omit further reference to  $F$  in our automaton notation.

In the context of decentralized discrete-event control, which is our focus here, there is a set of  $n$  agents  $I = \{1, \dots, n\}$  that cooperate to ensure that an uncontrolled system  $\mathcal{M}_L$  adheres to a given specification  $\mathcal{M}_K$ . We further assume, wlog, that  $\mathcal{M}_K \sqsubseteq \mathcal{M}_L$ . Each agent issues local control decisions that are combined with those of other agents to produce a global decision to enable or disable events.

C.K. Sharpe and S.L. Ricker are with the Department of Mathematics and Computer Science, Mount Allison University, Sackville NB, CANADA {cksharpe, lricker}@mta.ca

H. Marchand is with the University of Rennes, Inria, CNRS and IRISA, Rennes, FRANCE herve.marchand@inria.fr

To facilitate partial observation, we partition the alphabet  $\Sigma$  into disjoint sets of unobservable events  $\Sigma_{uo}$  and observable events  $\Sigma_o$ . We further define observable sub-alphabets  $\Sigma_{o,i}$  (not necessarily disjoint) for each agent  $i \in I$ , where  $\Sigma_{o,i} \subseteq \Sigma_o$ . Later, we will refer to agent 0 and expand the agent index set:  $I_0 = I \cup \{0\}$ , where  $\Sigma_{o,0} = \Sigma_o$ . The *natural projection*  $\pi_i : \Sigma^* \rightarrow \Sigma_{o,i}^*$  defines the observations of an agent  $i \in I_0$  by removing all occurrences of events in  $\Sigma \setminus \Sigma_{o,i}$  from a sequence in  $\Sigma^*$ . Specifically,  $\pi_i(\varepsilon) = \varepsilon$  and for  $s \in \Sigma^*$ ,  $\pi_i(s\sigma) = \pi_i(s)\sigma$  when  $\sigma \in \Sigma_{o,i}$  and  $\pi_i(s)$  otherwise. The inverse projection is defined as follows:  $(\forall w \in \Sigma_{o,i}^*) \pi_i^{-1}(w) = \{v \in \Sigma^* \mid \pi_i(v)=w\}$ . We will use the notation  $\llbracket w \rrbracket_i$  whenever we refer to  $\pi_i^{-1}\pi_i(w) \cap L$ , for  $i \in I_0$ .

Similarly, for control purposes,  $\Sigma$  is partitioned into controllable events  $\Sigma_c$  and uncontrollable events  $\Sigma_{uc}$ . We define controllable sub-alphabets  $\Sigma_{c,i}$  for each agent  $i \in I$ , where  $\Sigma_{c,i} \subseteq \Sigma_c$ . To identify the agents that control an event  $\sigma \in \Sigma_c$ , we use the notation  $I_c(\sigma) = \{i \in I \mid \sigma \in \Sigma_{c,i}\}$ .

### A. Inference observability

The approach to inferencing, as proposed by [3], requires the construction of two non-increasing sequences of prefixed-closed languages, with respect to the subset relation [5], corresponding to words after which a controllable event must be disabled<sup>1</sup> and words after which a controllable event must be enabled. Thus, for all  $\sigma \in \Sigma_c$ :

$$D_0(\sigma) = \{w \in K : w\sigma \notin K\}; \quad (1)$$

$$E_0(\sigma) = \{w \in K : w\sigma \in K\}. \quad (2)$$

Intuitively,  $D_0(\sigma)$  is the set of all words in  $K$  after which  $\sigma$  must be disabled to keep the system within  $K$ , and  $E_0(\sigma)$  is the set of all words in  $K$  after which  $\sigma$  must/can be enabled.

Each subsequent round of reasoning (for  $k \geq 0$ ) is inductively defined<sup>2</sup>

$$D_{k+1}(\sigma) = D_k(\sigma) \cap \left( \bigcap_{i \in I_c(\sigma)} \llbracket E_k(\sigma) \rrbracket_i \right); \quad (3)$$

$$E_{k+1}(\sigma) = E_k(\sigma) \cap \left( \bigcap_{i \in I_c(\sigma)} \llbracket D_k(\sigma) \rrbracket_i \right). \quad (4)$$

Remember that  $\llbracket E_k(\sigma) \rrbracket_i$  is the set of words that are indistinguishable from  $E_k(\sigma)$  w.r.t.  $\Sigma_{o,i}$  for agents  $i \in I_c(\sigma)$  during round  $k$ . By taking the intersection of these sets across all  $i \in I_c(\sigma)$  with  $D_k(\sigma)$ , we have the list of disablement decisions w.r.t.  $\sigma$  for which no agent has yet inferred the correct disablement decision. And equivalently for  $\llbracket D_k(\sigma) \rrbracket_i$ .

*Definition 1 ([3]):* Let  $K$  be a specification language defined with respect to system language  $L$ , and let  $\Sigma_{o,i}, \Sigma_{c,i}$  be the sets of observable and controllable events defined for agents  $i \in I$ . If there exists  $N \in \mathbb{N}$ , for all  $\sigma \in \Sigma_c$ , such that  $D_{N+1}(\sigma) = \emptyset$  or  $E_{N+1}(\sigma) = \emptyset$ , then  $K$  is ***N-inference observable*** with respect to  $L, \Sigma_{o,i}, \Sigma_{c,i}$  for  $i \in I$ .

The round when an agent answers ‘‘yes’’ to the question of whether to enable or disable event  $\sigma \in \Sigma_c$  is called the agent’s ambiguity level [3]. An ambiguity level of  $k$  means that if no other agent has issued a decision by round  $k - 1$ ,

<sup>1</sup>In the sequel, we use the color red to distinguish sequences  $w \in K$  such that  $\exists \sigma \in \Sigma_c$  and  $w\sigma \in L \setminus K$ .

<sup>2</sup>In [3], the more general mask is used instead of natural projection.

the agent(s) with ambiguity level  $k$  can issue a correct local decision at round  $k$ . The ambiguity levels w.r.t. disablement and enablement rounds for agent  $i \in I_c(\sigma)$  for word  $w \in L$  are calculated as:

$$\mathbf{a}_i^D(\pi_i(w), \sigma) := \min\{k \in \mathbb{N} : \pi_i(w) \notin \pi_i(E_k(\sigma))\}, \quad (5)$$

$$\mathbf{a}_i^E(\pi_i(w), \sigma) := \min\{k \in \mathbb{N} : \pi_i(w) \notin \pi_i(D_k(\sigma))\}. \quad (6)$$

So  $\mathbf{a}_i^D(\pi_i(w), \sigma)$  is the smallest integer  $k$  for which agent  $i$  does not confuse  $w$  with a word in  $E_k(\sigma)$  (after which  $\sigma$  is enabled). Similarly,  $\mathbf{a}_i^E(\pi_i(w), \sigma)$  is the smallest integer  $k$  for which agent  $i$  does not confuse  $w$  with a word in  $D_k(\sigma)$  (after which  $\sigma$  must be disabled).

Further, for each agent  $i \in I_c(\sigma)$ , to facilitate the overall local control decision, the ambiguity level is computed as the smaller of the two ambiguity levels for  $w \in K$  and  $\sigma \in \Sigma_c$ :

$$\mathbf{a}_i(\pi_i(w), \sigma) = \min\{\mathbf{a}_i^D(\pi_i(w), \sigma), \mathbf{a}_i^E(\pi_i(w), \sigma)\}. \quad (7)$$

## III. VERIFYING INFERENCE OBSERVABILITY

Inferencing has been considered based on properties of families of languages [3] and particular words of languages [2]. Examining inference observability through the lens of [2] opens up a characteristic of the words that remain when Eqs. (3) and (4) lead to a situation where neither sequence of prefix-closed languages converges to  $\emptyset$ . Ultimately, we borrow from both approaches to characterize when a language is not inference observable. Finally, we recast our analysis to a state-based framework in Section III-B. Subsequently, we revisit a finite-state structure for proving state-based decentralized properties [6] and use it here as part of an algorithm to verify state-based inference observability.

### A. Inferencing cycles

The definition of inference observability that follows is a generalization of the one introduced in [2], which considered only one round of inferencing. The first half of the definition, Eq. (8), is concerned with the *EBD* architecture. In contrast, the latter half, Eq. (9), refers to the *DBD* architecture.

*Definition 2:* A language  $K \subseteq L$  is ***inference observable*** w.r.t.  $L, \pi_i$  and  $\Sigma_{c,i}$  (for  $i \in I$ ) if either

$$(\forall w_0 \in K)(\forall \sigma \in \Sigma_c) w_0\sigma \in L \setminus K \Rightarrow (\exists m \in \mathbb{N}) \quad (8)$$

$$(\forall i_0 \in I_c(\sigma))(\forall w_1 \sigma \in \llbracket w_0 \rrbracket_{i_0} \sigma \cap K)$$

$$\llbracket w_1 \rrbracket_{i_0} \sigma \cap L \setminus K \neq \emptyset \Rightarrow$$

$$(\exists i'_0 \in I_c(\sigma))(\forall w_1 \sigma \in \llbracket w_0 \rrbracket_{i'_0} \sigma \cap K)$$

$$\llbracket w_1 \rrbracket_{i'_0} \sigma \cap L \setminus K \neq \emptyset \Rightarrow$$

$$(\forall i_1 \in I_c(\sigma))(\forall w_2 \sigma \in \llbracket w_1 \rrbracket_{i_1} \sigma \cap L \setminus K)$$

$$\llbracket w_2 \rrbracket_{i_1} \sigma \cap K \neq \emptyset \Rightarrow$$

$$(\exists i'_1 \in I_c(\sigma))(\forall w_2 \sigma \in \llbracket w_1 \rrbracket_{i'_1} \sigma \cap L \setminus K)$$

$$\llbracket w_2 \rrbracket_{i'_1} \sigma \cap K \neq \emptyset \Rightarrow$$

$$(\forall i_2 \in I_c(\sigma))(\forall w_3 \sigma \in \llbracket w_2 \rrbracket_{i_2} \sigma \cap K)$$

$$\llbracket w_3 \rrbracket_{i_2} \sigma \cap L \setminus K \neq \emptyset \Rightarrow$$

...

$$\begin{aligned}
& (\forall i_{m-1} \in I_c(\sigma)) (\forall w_m \sigma \in \llbracket w_{m-1} \rrbracket_{i'_{m-2}} \sigma \cap L \setminus K) \\
& \quad \llbracket w_m \rrbracket_{i_{m-1}} \sigma \cap K \neq \emptyset \Rightarrow \\
& \quad (\exists i'_{m-1} \in I_c(\sigma)) (\forall w_m \sigma \in \llbracket w_{m-1} \rrbracket_{i'_{m-1}} \sigma \cap L \setminus K) \\
& \quad \quad \llbracket w_m \rrbracket_{i'_{m-1}} \sigma \cap K \neq \emptyset \Rightarrow \\
& \quad \quad (\exists i_m \in I_c(\sigma)) (\forall w_{m+1} \sigma \in \llbracket w_m \rrbracket_{i_m} \sigma \cap K) \\
& \quad \quad \quad \llbracket w_{m+1} \rrbracket_{i_m} \sigma \cap L \setminus K = \emptyset
\end{aligned}$$

or

$$\begin{aligned}
& (\forall w_0 \in K) (\forall \sigma \in \Sigma_c) w_0 \sigma \in K \Rightarrow (\exists m \in \mathbb{N}) \quad (9) \\
& (\forall i_0 \in I_c(\sigma)) (\forall w_1 \sigma \in \llbracket w_0 \rrbracket_{i_0} \sigma \cap L \setminus K) \\
& \quad \llbracket w_1 \rrbracket_{i_0} \sigma \cap K \neq \emptyset \Rightarrow \\
& \quad (\exists i'_0 \in I_c(\sigma)) (\forall w_1 \sigma \in \llbracket w_0 \rrbracket_{i'_0} \sigma \cap L \setminus K) \\
& \quad \quad \llbracket w_1 \rrbracket_{i'_0} \sigma \cap K \neq \emptyset \Rightarrow \\
& (\forall i_1 \in I_c(\sigma)) (\forall w_2 \sigma \in \llbracket w_1 \rrbracket_{i_1} \sigma \cap K) \\
& \quad \llbracket w_2 \rrbracket_{i_1} \sigma \cap L \setminus K \neq \emptyset \Rightarrow \\
& \quad (\exists i'_1 \in I_c(\sigma)) (\forall w_2 \sigma \in \llbracket w_1 \rrbracket_{i'_1} \sigma \cap K) \\
& \quad \quad \llbracket w_2 \rrbracket_{i'_1} \sigma \cap L \setminus K \neq \emptyset \Rightarrow \\
& (\forall i_2 \in I_c(\sigma)) (\forall w_3 \sigma \in \llbracket w_2 \rrbracket_{i_2} \sigma \cap L \setminus K) \\
& \quad \llbracket w_3 \rrbracket_{i_2} \sigma \cap K \neq \emptyset \Rightarrow \\
& \quad \dots \\
& (\forall i_{m-1} \in I_c(\sigma)) (\forall w_m \sigma \in \llbracket w_{m-1} \rrbracket_{i_{m-1}} \sigma \cap K) \\
& \quad \llbracket w_m \rrbracket_{i_{m-1}} \sigma \cap L \setminus K \neq \emptyset \Rightarrow \\
& \quad (\exists i'_{m-1} \in I_c(\sigma)) (\forall w_m \sigma \in \llbracket w_{m-1} \rrbracket_{i'_{m-1}} \sigma \cap K) \\
& \quad \quad \llbracket w_m \rrbracket_{i'_{m-1}} \sigma \cap L \setminus K \neq \emptyset \Rightarrow \\
& \quad \quad (\exists i_m \in I_c(\sigma)) (\forall w_{m+1} \sigma \in \llbracket w_m \rrbracket_{i_m} \sigma \cap L \setminus K) \\
& \quad \quad \quad \llbracket w_{m+1} \rrbracket_{i_m} \sigma \cap K = \emptyset.
\end{aligned}$$

For instance, if  $K$  is EBD inference observable, i.e., Eq.(8), then for each  $w_0 \in K$  and  $\sigma \in \Sigma_c$  where  $w_0 \sigma \in L \setminus K$ , there is some finite set of  $m$  words  $w_0, w_1, w_2, \dots, w_{m-1}, w_m$  and associated ordering of agents  $i_0, i_1, \dots, i_{m-1}, i_m$  (where  $i_j \neq i_{j+1}$ , for  $j \in \{0, \dots, m-1\}$ ) in  $I_c(\sigma)$ , where eventually, after  $m$  rounds of inferencing, agent  $i'_0$  finds an agent  $i_m$  whose partial observation of the system, namely  $\llbracket w_m \rrbracket_{i_m} \sigma$ , allows  $i_m$  to take a definitive disablement decision on  $\sigma$  (with an ambiguity level of 0), while agent  $i'_0$  takes a disablement decision on the words in  $\llbracket w_0 \rrbracket_{i'_0}$  with ambiguity level  $m$ .

*Remark 1: Def. 2 relies on distinguishing, at each step  $j$ , the existence of a specific agent  $i_j \in I_c(\sigma)$  so that reaching the conclusion of inferencing the correct control decision is associated with a finite series of agents that contribute to the inferencing process. Ultimately, we are only interested in the existence of  $m \in \mathbb{N}$  and the corresponding final agent  $i_m$ . However, we will find discussing the absence of such a finite sequence of agents helpful when identifying that a system is not inference observable.*

We demonstrate this in Fig 1, with a possible disablement

inferencing scenario. Let the inferencing begin with a word  $d_0 \in K$  such that  $\sigma \in \Sigma_c$  and  $d_0 \sigma \in L \setminus K$ . (Note that we can equivalently begin with a word  $e \sigma \in K$  for an enablement decision). Let  $e_1 \in K$  be a word such that  $e_1 \sigma \in \llbracket d_0 \rrbracket_{i_0} \sigma \cap K$ . Agent  $i_0$  delegates to all  $i_1 \in I_c(\sigma) \setminus \{i_0\}$  to see if  $\llbracket e_1 \rrbracket_{i_1} \sigma \cap L \setminus K = \emptyset$ . If such an agent exists at this point, then the decision should be taken in the DBD architecture where the focus is on decisions about enablements. This pattern of delegation between the inability to distinguish words in  $L \setminus K$  and  $K$  continues until eventually,  $m$  steps later, we find agent  $i_m \in I_c(\sigma)$  such that  $\llbracket d_m \rrbracket_{i_m} \sigma \cap K = \emptyset$ , ie. agent  $i_m$  can take the definitive disablement decision about  $\sigma$  with an ambiguity level of 0 for the words in  $\llbracket d_m \rrbracket_{i_m} \sigma$ .

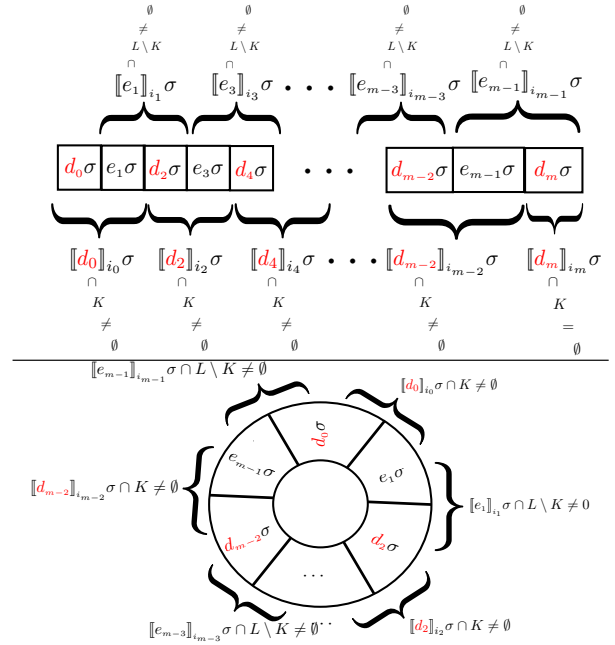


Fig. 1. (Top) Inferencing rounds that result in a successful disablement control decision for  $\sigma$  by agent  $i_m$  after it observes  $\llbracket d_m \rrbracket_{i_m}$ . (Bottom) An inferencing cycle for  $d_0 \sigma$ , for  $i_0, \dots, i_{m-1} \in I_c(\sigma)$ .

When there is no EBD or DBD inferencing solution, there exist  $\sigma \in \Sigma_c$ , words  $w, w' \in K$ , where  $w \sigma \in L \setminus K$  and  $w' \sigma \in K$ , for which a successful conclusion to inferencing cannot happen in a finite number of rounds. Indeed, by taking the negation of Def. 2, when a system is not inference observable, there are disablement (and enablement) decisions that cannot correctly be taken, no matter how many rounds of inferencing occur. In other words, whenever two infinite sequences exist such that Eq. (8) and Eq. (9) do not terminate, the system is not inference observable. (This is consistent with the situation when neither Eq. (3) nor Eq. (4) converges to  $\emptyset$ ). In each step  $i_j$  (for  $j \in \{0, \dots, m-1\}$ ), none of the agents can make a correct EBD decision, and in the  $i_{j+1}$  step, none of the agents can take the correct DBD decision. To characterize such infinite sequences of disablement (and enablement) decisions, we introduce the notion of an inference cycle.

*Definition 3: A word  $w \in K \wedge w \sigma \in L \setminus K$  gives rise to*

an EBD inferencing cycle if:

$$\begin{aligned}
& (\exists w_0 \in K)(\exists \sigma \in \Sigma_c) w_0 \sigma \in L \setminus K \Rightarrow (\exists m \in \mathbb{N}) \\
& (\forall i_0 \in I_c(\sigma))(\forall w_1 \sigma \in \llbracket w_0 \rrbracket_{i_0} \sigma \cap K) \\
& \quad \llbracket w_1 \rrbracket_{i_0} \sigma \cap L \setminus K \neq \emptyset \Rightarrow \\
& \quad (\exists i'_0 \in I_c(\sigma))(\forall w_1 \sigma \in \llbracket w_0 \rrbracket_{i'_0} \sigma \cap K) \\
& \quad \quad \llbracket w_1 \rrbracket_{i'_0} \sigma \cap L \setminus K \neq \emptyset \Rightarrow \\
& (\forall i_1 \in I_c(\sigma) \setminus \{i'_0\})(\forall w_2 \sigma \in \llbracket w_1 \rrbracket_{i_1} \sigma \cap L \setminus K) \\
& \quad \llbracket w_2 \rrbracket_{i_1} \sigma \cap K \neq \emptyset \Rightarrow \\
& \quad (\exists i'_1 \in I_c(\sigma) \setminus \{i'_0\})(\forall w_2 \sigma \in \llbracket w_1 \rrbracket_{i'_1} \sigma \cap L \setminus K) \\
& \quad \quad \llbracket w_2 \rrbracket_{i'_1} \sigma \cap K \neq \emptyset \Rightarrow \\
& (\forall i_2 \in I_c(\sigma) \setminus \{i'_1\})(\forall w_3 \sigma \in \llbracket w_2 \rrbracket_{i_2} \sigma \cap K) \\
& \quad \llbracket w_3 \rrbracket_{i_2} \sigma \cap L \setminus K \neq \emptyset \Rightarrow \\
& \quad (\exists i'_2 \in I_c(\sigma) \setminus \{i'_1\})(\forall w_3 \sigma \in \llbracket w_2 \rrbracket_{i'_2} \sigma \cap K) \\
& \quad \quad \llbracket w_3 \rrbracket_{i'_2} \sigma \cap L \setminus K \neq \emptyset \Rightarrow \\
& \quad \quad \dots \\
& (\exists i'_{m-1} \in I_c(\sigma) \setminus \{i'_{m-2}\})(\forall w_m \sigma \in \llbracket w_{m-1} \rrbracket_{i'_{m-1}} \sigma \cap L \setminus K) \\
& \quad \llbracket w_m \rrbracket_{i'_{m-1}} \sigma \cap K \neq \emptyset \Rightarrow \\
& (\forall i_m \in I_c(\sigma) \setminus \{i'_{m-1}\})(\forall w_{m+1} \sigma \in \llbracket w_m \rrbracket_{i_m} \sigma \cap K) \\
& \quad \llbracket w_{m+1} \rrbracket_{i_m} \sigma \cap L \setminus K \neq \emptyset \Rightarrow \\
& \quad (\exists j \in \{0, 1, \dots, m-1\}) \\
& \quad \quad \llbracket w_m \rrbracket_{i_m} \sigma = \llbracket w_j \rrbracket_{i_j} \sigma
\end{aligned}$$

Equivalently, a DBD inferencing cycle can be defined for  $w_0 \sigma \in K$  by changing the  $L \setminus K$  for  $K$  and vice versa.

*Proposition 1:* Given  $w, w' \in K$  and  $\sigma \in \Sigma_c$  such that  $w \sigma \in L \setminus K$  and  $w' \sigma \in K$ . If  $w$  and  $w'$  give rise to a DBD inferencing cycle and an EBD inferencing cycle, respectively, then  $K$  is not inference-observable w.r.t.  $L$ .

*Proof sketch:* In the presence of such EBD and DBD inferencing cycles, then starting from the word  $w$ , there exists a sequence of all agents in  $I_c(\sigma)$ , say  $I_j = \{i_0, \dots, i_j, \dots, i_m\}$  such that  $\llbracket w_m \rrbracket_{i_m} = \llbracket w_j \rrbracket_{i_j}$  for the EBD inferencing decision-making process such that  $\llbracket w_j \rrbracket_{i_j} \sigma \cap K \neq \emptyset$  (and similarly for the EBD inferencing decision-making process). Thus, by iterating these cycles of agent choices  $(i_j, \dots, i_{m-1}, i_j)$ , we can build infinite sequences of decisions for each round  $(i_0, \dots, i_{j-1}).(i_j, \dots, i_{m-1})^k$ , for  $k \geq 1$  (for EBD, and similarly for DBD) that, by construction, never converge to  $\emptyset$ . Thus,  $K$  is not inference-observable w.r.t.  $L$ . ■

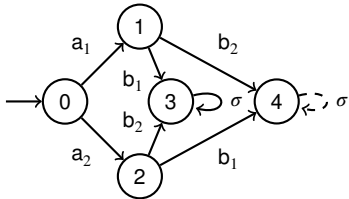


Fig. 2. Joint  $\mathcal{M}_L$  (all transitions) and  $\mathcal{M}_K$  (only solid-line transitions) for Ex. 1 where  $n = 2$ ,  $\Sigma_{\alpha,1} = \{a_1, a_2\}$ ,  $\Sigma_{\alpha,2} = \{b_1, b_2\}$ , and  $I_c(\sigma) = \{1, 2\}$ .

*Example 1:* Consider the example in Fig. 2, noting that the solid line transitions indicate transitions in  $T_K$ , whereas the dashed line transition is in  $T_L \setminus T_K$ . By Eqs (1), (3) and (2), (4), this system is not inference observable:

$$\begin{aligned}
D_0(\sigma) &= \{a_1 b_2, a_2 b_1\}, & E_0(\sigma) &= \{a_1 b_1, a_2 b_2\}; \\
D_1(\sigma) &= D_0(\sigma), & E_1(\sigma) &= E_0(\sigma).
\end{aligned}$$

Consequently, by Def. 3, we can find at least one EBD and DBD inferencing cycle. For the EBD cycle, let  $w_0 \sigma = a_1 b_2 \sigma$  and  $i'_0 = 1$ . We have an inferencing cycle for  $m = 4$  as follows.

$$\begin{aligned}
\llbracket a_1 b_2 \rrbracket_1 \sigma \cap K &\Rightarrow w_1 = a_1 b_1, i'_1 = 2, \\
\llbracket a_1 b_1 \rrbracket_2 \sigma \cap L \setminus K &\Rightarrow w_2 = a_2 b_1, i'_2 = 1, \\
\llbracket a_2 b_1 \rrbracket_1 \sigma \cap K &\Rightarrow w_3 = a_2 b_2, i'_3 = 2, \\
\llbracket a_2 b_2 \rrbracket_2 \sigma \cap L \setminus K &\Rightarrow w_4 = a_1 b_2, i_4 = 1, \\
\llbracket w_4 \rrbracket_{i_4} \sigma &= \llbracket w_0 \rrbracket_{i'_0} \sigma, i'_0 = i_4 \Rightarrow m = 4, j = 0
\end{aligned}$$

Equivalently, we have a DBD cycle by starting with  $w_0 \sigma = a_1 b_1 \sigma$  and  $i'_0 = 2$ , where  $m = 4$  and  $j = 0$ . ◆

We will evaluate the absence of inferencing cycles to determine whether a system is inference-observable. Further, we revisit a finite-state structure we have used previously for verifying decentralized properties. First, we construct a bipartite graph using particular states in this finite state space. Then we leverage the detection of inferencing cycles using the bipartite graph.

### B. A finite-state verification structure

We use a variation of the finite-state verification structure that was introduced in [7] and is constructed using a special product that we call *synchronized composition*, denoted by  $\times_S$ . We take  $n + 1$  copies of  $\mathcal{M}_L^\varepsilon$ , which is simply  $\mathcal{M}_L$  augmented with self-loops of  $\varepsilon$  at each state:

$$\begin{aligned}
\mathcal{U} &= \overbrace{\mathcal{M}_L^\varepsilon \times_S \mathcal{M}_L^\varepsilon \times_S \dots \times_S \mathcal{M}_L^\varepsilon}^{(n+1) \text{ times}} \\
&= (Q_S, \Delta_S, T_S, q_{0_S}),
\end{aligned}$$

where  $Q_S \subseteq Q^{n+1}$  is a set of state vectors of the form  $q_S = (q, q_1, \dots, q_n)$  and it will be convenient to refer to the  $j^{\text{th}}$  component of  $q_S$  as  $q_S(j)$ , for  $j \in \{0, 1, \dots, n\}$ ;  $\Delta_S \subseteq \Sigma^{n+1}$  is set of vector labels [8]; the transition relation  $T_S \subseteq Q_S \times \Delta_S \times Q_S$ , is defined as follows:  $(q_S, \ell, q'_S) \in T_S$ , where  $q_S = (q, q_1, \dots, q_n) \in Q_S$ ,  $q'_S = (q', q'_1, \dots, q'_n) \in Q_S$  and  $\ell = \langle \sigma, \sigma_1, \dots, \sigma_i, \dots, \sigma_n \rangle \in \Delta_S$  iff  $(q, \sigma, q') \in T_L^\varepsilon$ , and for all  $i \in I$ ,  $(q_i, \sigma_i, q'_i) \in T_L^\varepsilon$ , where  $T_L^\varepsilon$  is the transition relation for  $\mathcal{M}_L^\varepsilon$ ; and, the initial state  $q_{0_S} = (q_0, q_0, \dots, q_0)$ . Labels in  $\Delta_S$  are partitioned into sets of observable labels  $\Delta_i = \{\ell \in \Delta_S \mid \ell(0) = \ell(i) \in \Sigma_{\sigma_i}\}$  and unobservable labels  $\Delta_{u\sigma, i} = \Delta_S \setminus \Delta_i$ , for  $i \in I_0$ . We also define the empty vector label  $\ell^\varepsilon = \langle \varepsilon, \dots, \varepsilon \rangle$ , where for all  $i \in I_0$ ,  $\ell^\varepsilon(i) = \varepsilon$ . A word  $v = \ell_0 \ell_1 \dots \ell_{|v|-1}$  in the language generated by  $\mathcal{U}$  consists of a vector of sequences  $v(0) = \ell_0(0) \ell_1(0) \dots \ell_{|v|}(0)$ ,  $v(1) = \ell_0(1) \ell_1(1) \dots \ell_{|v|}(1) \dots v(n) = \ell_0(n) \ell_1(n) \dots \ell_{|v|}(n)$  such that  $\pi_i(v(0)) = \pi_i(v(i))$ , for all  $i \in I$ .





$$\begin{aligned}
& (\exists d_0 \in Q_S)(\exists \sigma \in \Sigma_c) d_0 \in D_\sigma \Rightarrow (\exists m \in \mathbb{N}) \\
& (\forall i_0 \in I_c(\sigma))(\forall e_1 \in \llbracket d_0 \rrbracket_{i_0} \cap E_\sigma) \\
& \quad \llbracket e_1 \rrbracket_{i_0} \cap D_\sigma \neq \emptyset \Rightarrow \\
& \quad (\exists i'_0 \in I_c(\sigma))(\forall e_1 \in \llbracket d_0 \rrbracket_{i'_0} \cap E_\sigma) \\
& \quad \quad \llbracket e_1 \rrbracket_{i'_0} \cap D_\sigma \neq \emptyset \Rightarrow \\
& (\forall i_1 \in I_c(\sigma) \setminus \{i'_0\})(\forall d_2 \in \llbracket e_1 \rrbracket_{i'_0} \cap D_\sigma) \\
& \quad \llbracket d_2 \rrbracket_{i_1} \cap E_\sigma \neq \emptyset \Rightarrow \\
& \quad (\exists i'_1 \in I_c(\sigma) \setminus \{i'_0\})(\forall d_2 \in \llbracket e_1 \rrbracket_{i'_0} \cap D_\sigma) \\
& \quad \quad \llbracket d_2 \rrbracket_{i'_1} \cap E_\sigma \neq \emptyset \Rightarrow \\
& \quad \quad \dots \\
& \quad (\exists i'_{m-1} \in I_c(\sigma) \setminus \{i'_{m-2}\})(\forall d_m \in \llbracket e_{m-1} \rrbracket_{i'_{m-1}} \cap D_\sigma) \\
& \quad \quad \llbracket d_m \rrbracket_{i'_{m-1}} \cap E_\sigma \neq \emptyset \Rightarrow \\
& (\forall i_m \in I_c(\sigma) \setminus \{i'_{m-1}\})(\forall e_{m+1} \in \llbracket d_m \rrbracket_{i'_m} \cap E_\sigma) \\
& \quad \llbracket e_{m+1} \rrbracket_{i_m} \cap D_\sigma \neq \emptyset \Rightarrow \\
& \quad (\exists j \in \{0, 1, \dots, m-1\}) \\
& \quad \quad \llbracket d_m \rrbracket_{i_m} = \llbracket d_j \rrbracket_{i'_j}.
\end{aligned}$$

Instead of writing  $d_0 \sim_{i_0} e_1 \wedge e_1 \sim_{i_1} d_2 \wedge \dots \wedge e_{m-1} \sim_{i_{m-1}} d_0$ , for  $i_0, \dots, i_m \in I_c(\sigma)$ , we will use the following notation to denote a specific inferencing cycle of  $d_0$ :  $d_0 \sim_{i_0} e_1 \sim_{i_1} d_2 \sim_{i_2} \dots \sim_{i_{m-1}} e_{m-1} \sim_{i_{m-1}} d_0$ .

We can equivalently define a DBD state-inferencing cycle for some control configuration  $e \in E_\sigma$ .

A consequence of using subset construction on  $\mathcal{U}$  is that states, including control configurations, may appear in more than one state of the determinized state space. This is because the subset relation is not an equivalence relation in that it may be the case that the determinization process allocates states into more than one deterministic state. Thus, we can refine the set of control configurations of Def. 4. Hence, to determine the complete set of states for  $D_\sigma$  and  $E_\sigma$ , we compute  $\mathcal{O}_{\Delta_0}$ , where  $\Delta_0 = \Delta_\sigma$ . If a state  $q_S$  of  $\mathcal{U}$  appears in more than one state of  $\mathcal{O}_{\Delta_0}$ , we create a copy of  $q_S$  with a label  $q_{S_j}$ , for  $j \in$  some finite index set  $J$ , in  $\mathcal{U}$  and we relabel these states in  $\mathcal{O}_{\Delta_0}$ . Further, each copy is then added to  $E_\sigma$  and  $D_\sigma$ . The procedure is depicted below in the continuation of our example.

*Proposition 2:* Given  $\mathcal{U}$ ,  $\mathcal{M}_L$ , and  $\mathcal{M}_K$ . If control configurations  $d \in D_\sigma$  and  $e \in E_\sigma$  give rise to EBD and DBD state-inferencing cycles, respectively, then  $K$  is not inference-observable w.r.t.  $L$ .

The proof follows equivalent reasoning to that of Prop. 1.

*Ex. 1 cont.* From Fig. 3, we identify the control configurations:

$$D_\sigma = \{(4,3,3), (4,3,4), (4,4,3), (4,4,4)\}; \quad (10)$$

$$E_\sigma = \{(3,3,3), (3,3,4), (3,4,3), (3,4,4)\}. \quad (11)$$

Note that in Fig. 2 some words have different projections w.r.t.  $\pi_i$  (for  $i = \{1, 2\}$ ), but happen to go to the same state. For example,  $a_1b_1$  and  $a_2b_2$  have different projections for

both agents, but both go to state 3. In  $\mathcal{U}$ , it is also possible to find two different words that go to the same state, but because they have different projections, that common state will appear in two different states of  $\mathcal{O}_{\Delta_0}$ . As can be seen in Fig. 3, the word  $w = \langle a_1, a_1, \epsilon \rangle \langle \epsilon, b_1, \epsilon \rangle \langle \epsilon, \epsilon, a_2 \rangle \langle b_1, b_1, \epsilon \rangle$  goes to control configuration state  $(3, 3, 4)$ . But so does the word  $w' = \langle a_2, a_2, \epsilon \rangle \langle \epsilon, \epsilon, a_1 \rangle \langle \epsilon, b_2, \epsilon \rangle \langle b_2, b_2, \epsilon \rangle$ . We then construct  $\mathcal{O}_{\Delta_0}$ , where  $\Delta_0 = \{\langle a_1, a_1, \epsilon \rangle, \langle a_2, a_2, \epsilon \rangle, \langle b_1, \epsilon, b_1 \rangle, \langle b_2, \epsilon, b_2 \rangle\}$ . Because  $\pi_0(w(0)) \neq \pi_0(w'(0))$ , the subset relation over  $\Delta_0$  results in the control configuration  $(3, 3, 4)$  appearing in two different states of  $\mathcal{O}_{\Delta_0}$ , namely  $x_{03}$  and  $x_{05}$ . In fact, each state in  $D_\sigma$  (Eq. (10)) and  $E_\sigma$  (Eq. (11)) appears in two different states of  $\mathcal{O}_{\Delta_0}$ ,  $x_{03}$  and  $x_{05}$  for  $E_\sigma$ , and  $x_{04}$  and  $x_{06}$  for  $D_\sigma$ . So we label the copies with a subscript of ‘A’ to distinguish them from the originals. The related states of  $\mathcal{O}_{\Delta_0}$  are presented in the leftmost column of Table I.

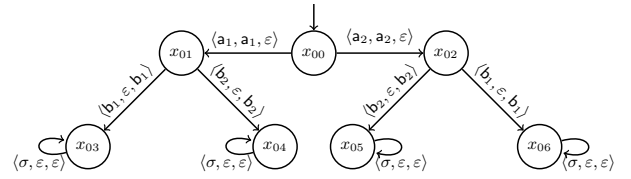


Fig. 4.  $\mathcal{O}_{\Delta_0}$  for Ex. 1.

TABLE I

STATE SETS  $X_0$  (LEFT),  $X_1$  (CENTER) AND  $X_2$  (RIGHT) FOR EX. 1. THE CONTROL CONFIGURATIONS  $E_\sigma$  IDENTIFIED IN GREEN AND  $D_\sigma$  IN RED.

States of $\mathcal{O}_{\Delta_0}$	States of $\mathcal{O}_{\Delta_1}$	States of $\mathcal{O}_{\Delta_2}$
$x_{00}$ $\{(0,0,0), (0,0,1), (0,0,2)\}$	$x_{10}$ $\{(0,0,0), (0,0,1), (0,0,2)\}$	$x_{20}$ $\{(0,0,0), (0,0,1), (0,0,2), (1,1,0), (1,1,1), (1,1,2), (1,3,0), (1,3,1), (1,3,2), (1,4,0), (1,4,1), (1,4,2)\}$
$x_{01}$ $\{(1,1,0), (1,1,1), (1,1,2), (1,3,0), (1,3,1), (1,3,2), (1,4,0), (1,4,1), (1,4,2)\}$	$x_{11}$ $\{(1,1,0), (1,1,1), (1,1,2), (1,3,0), (1,3,1), (1,3,2), (1,4,0), (1,4,1), (1,4,2), (3,1,3), (3,1,4), (3,3,3), (3,3,4), (3,4,3), (3,4,4), (4,1,3), (4,1,4), (4,3,3), (4,3,4), (4,4,3), (4,4,4)\}$	$x_{21}$ $\{(3,1,3), (3,1,4), (3,3,3), (3,3,4), (3,4,3), (3,4,4), (4,2,3), (4,2,4), (4,3,3), (4,3,4), (4,4,3), (4,4,4)\}$
$x_{02}$ $\{(2,2,0), (2,2,1), (2,2,2), (2,3,0), (2,3,1), (2,3,2), (2,4,0), (2,4,1), (2,4,2)\}$	$x_{12}$ $\{(2,2,0), (2,2,1), (2,2,2), (2,3,0), (2,3,1), (2,3,2), (2,4,0), (2,4,1), (2,4,2), (3,2,3), (3,2,4), (3,3,3), (3,3,4), (3,4,3), (3,4,4), (4,2,3), (4,2,4), (4,3,3), (4,3,4), (4,4,3), (4,4,4)\}$	$x_{22}$ $\{(4,1,3), (4,1,4), (4,3,3), (4,3,4), (4,4,3), (4,4,4)\}$
$x_{03}$ $\{(3,1,3), (3,1,4), (3,3,3), (3,3,4), (3,4,3), (3,4,4)\}$		$x_{21}$ $\{(3,1,3), (3,1,4), (3,3,3), (3,3,4), (3,4,3), (3,4,4), (4,2,3), (4,2,4), (4,3,3), (4,3,4), (4,4,3), (4,4,4)\}$
$x_{04}$ $\{(4,1,3), (4,1,4), (4,3,3), (4,3,4), (4,4,3), (4,4,4)\}$		$x_{22}$ $\{(4,1,3), (4,1,4), (4,3,3), (4,3,4), (4,4,3), (4,4,4), (3,2,3), (3,2,4), (3,3,3), (3,3,4), (3,4,3), (3,4,4)\}$
$x_{05}$ $\{(3,2,3), (3,2,4), (3,3,3), (3,3,4), (3,4,3), (3,4,4)\}$		
$x_{06}$ $\{(4,2,3), (4,2,4), (4,3,3), (4,3,4), (4,4,3), (4,4,4)\}$		

After relabeling,  $E_\sigma = \{(3,3,3), (3,3,4), (3,4,3), (3,4,4), (3,3,3)_A, (3,3,4)_A, (3,4,3)_A, (3,4,4)_A\}$  and  $D_\sigma = \{(4,3,3), (4,3,4), (4,4,3), (4,4,4), (4,3,3)_A, (4,3,4)_A, (4,4,3)_A, (4,4,4)_A\}$ .

We have  $\Delta_1 = \{\langle a_1, a_1, \epsilon \rangle, \langle a_2, a_2, \epsilon \rangle\}$  and  $\Delta_2 = \{\langle b_1, \epsilon, b_1 \rangle, \langle b_2, \epsilon, b_2 \rangle\}$ . The relabeled states in  $\mathcal{O}_{\Delta_1}$  and  $\mathcal{O}_{\Delta_2}$  are presented in the two rightmost columns of Table I. Finally, by Table I, we apply Def. 5 to control configuration states, noting that  $(3,3,3) \sim_1 (4,3,3)$  as both these states are in  $x_{11} \in X_1$ , whereas  $(3,3,3) \sim_2 (4,3,3)_A$  as these control configuration states are both in  $x_{21} \in X_2$ .  $\blacklozenge$

#### IV. AN ALGORITHM FOR VERIFYING INFERENCE OBSERVABILITY

Here we present our algorithm for verifying whether a system is inference observable. In particular, when there is an inferencing solution, Algorithm 1 records the minimum ambiguity levels consistent with those computed in Eq. (7). The basic outline of the verification strategy is as follows:

---

**Algorithm 1** Verifying state-based inference-observability and computing ambiguity levels
 

---

```

1: procedure ISINFOBS( $\Sigma_c, \{I_c(\sigma)\}_{\sigma \in \Sigma_c},$ 
    $\{(V_{D_\sigma}, V_{E_\sigma})\}_{\sigma \in \Sigma_c}, \{\sim_i\}_{i \in I_c(\sigma)}$ 
2:   for all  $\sigma \in \Sigma_c$  do
3:      $\triangleright$  Initialize the ambiguity levels and the set of
       neighboring vertices for all  $v \in V_{D_\sigma} \cup V_{E_\sigma}$ .
       The loop below runs in  $O(|I_c(\sigma)| \cdot |E_\sigma \cup D_\sigma|)$ 
       time.  $\triangleleft$ 
4:     for all  $i \in I_c(\sigma)$  do
5:       for all  $v \in V_{D_\sigma} \cup V_{E_\sigma}$  do
6:          $N_{i,v} \leftarrow \emptyset$ 
7:          $a_i^D(v, \sigma) \leftarrow 0$ 
8:          $a_i^E(v, \sigma) \leftarrow 0$ 
9:        $\triangleright$  Building edges of bipartite graph. This can
       be achieved in  $O(|E_\sigma| \cdot |D_\sigma| \cdot |I_c(\sigma)|)$  time.  $\triangleleft$ 
10:      for all  $d \in V_{D_\sigma}$  do
11:        for all  $e \in V_{E_\sigma}$  do
12:          for all  $i \in I_c(\sigma)$  do
13:            if  $d \sim_i e$  then
14:               $N_{i,d} \leftarrow N_{i,d} \cup \{e\}$ 
15:               $N_{i,e} \leftarrow N_{i,e} \cup \{d\}$ 
16:             $\triangleright R$  is the set of vertices in  $V_{D_\sigma} \cup V_{E_\sigma}$  that can
            be distinguished by at least one  $i \in I_c(\sigma)$   $\triangleleft$ 
17:             $R \leftarrow \emptyset$ 
18:             $\triangleright$  Initialize the ambiguity level to 0 and identify
            vertices that are immediately distinguishable before checking for cycles.
            This can be done in  $O((|E_\sigma| + |D_\sigma|) \cdot |I_c(\sigma)| + |E_\sigma| \cdot |D_\sigma| \cdot |I_c(\sigma)| + \text{time})$ .  $\triangleleft$ 
19:             $ambLevel \leftarrow 0$ 
20:            for all  $v \in V_{D_\sigma} \cup V_{E_\sigma}$  do
21:              for all  $i \in I_c(\sigma)$  do
22:                if  $N_{i,v} = \emptyset$  then
23:                   $R \leftarrow R \cup \{v\}$ 
24:                 $\triangleright$  Set appropriate ambiguity level
                for  $v$  to 0 w.r.t. agent  $i$   $\triangleleft$ 
25:                if  $v \in V_{D_\sigma}$  then
26:                   $a_i^D(v, \sigma) \leftarrow ambLevel$ 
27:                else
28:                   $a_i^E(v, \sigma) \leftarrow ambLevel$ 
29:                 $Resolved \leftarrow R$ 
30:                 $\triangleright$  Loop to detect cycles in  $G_\sigma$ . If  $G_\sigma$  has
                 $t_\sigma$  edges in the bipartite graph, then the
                complexity is  $O(\sum_{\sigma \in \Sigma_c} (|E_\sigma| + |D_\sigma| + t_\sigma))$ 
                because each node and edge can be removed
                at most once.  $\triangleleft$ 
31:                while  $R \neq \emptyset$  do
32:                   $R' \leftarrow \emptyset$ 
33:                   $ambLevel \leftarrow ambLevel + 1$ 
34:                  for all  $r \in R$  do
35:                    for all  $i \in I_c(\sigma)$  do
36:                      for all  $v' \in N_{i,r}$  do
37:                         $N_{i,v'} \leftarrow N_{i,v'} \setminus \{r\}$ 
38:                        if  $N_{i,v'} = \emptyset$  then
39:                           $\triangleright$  Only add  $v'$  if it's not
                          already resolved.  $\triangleleft$ 
40:                          if  $v' \notin Resolved$  then
41:                             $R' \leftarrow R' \cup \{v'\}$ 
42:                           $\triangleright$  Set ambiguity level for  $v'$ 
                          and agent  $i$   $\triangleleft$ 
43:                          if  $v' \in D_\sigma$  then
44:                             $a_i^D(v, \sigma) \leftarrow ambLevel$ 
45:                          else
46:                             $a_i^E(v, \sigma) \leftarrow ambLevel$ 
47:                           $R \leftarrow R'$ 
48:                           $Resolved \leftarrow Resolved \cup R$ 
49:                          if  $|Resolved| < |V_{D_\sigma}| + |V_{E_\sigma}|$  then return False
50:                        return True
  
```

---

- 1) Before running Algorithm 1, compute  $\mathcal{U}$  to identify the control configurations w.r.t. enablement and disablement.
- 2) Run subset construction on  $\mathcal{U}$  over the alphabet of observable labels  $\Delta_0$  to build  $\mathcal{O}_{\Delta_0}$ .
- 3) Check states of  $\mathcal{O}_{\Delta_0}$  for duplicates of  $E_\sigma$  and  $D_\sigma$ , relabel and update these states in both in  $\mathcal{U}$  and  $\mathcal{O}_{\Delta_0}$ .
- 4) Construct  $\mathcal{O}_{\Delta_i}$ , for  $i \in I, \sim_i$ .
- 5) Run Algorithm 1 that consists of the following milestones:
  - a) Build the bipartite graph using the (relabelled) states of  $E_\sigma$  and  $D_\sigma$  as vertices. An edge with label  $i \in I_c(\sigma)$  exists between a vertex in  $V_{E_\sigma}$  and a vertex in  $V_{D_\sigma}$  if agent  $i$  cannot distinguish between the states in  $\mathcal{U}$  represented by the two vertices.
  - b) Run an inferencing cycle detection on the bipartite graph according to Def. 6. If inferencing cycles exist, exit with no solution, otherwise, if more controllable

events are left to check, pick one at random and repeat from 5a. If all controllable events have been checked, exit with an inferencing solution.

#### A. The algorithm

Algorithm 1 begins by initializing the set of the neighboring vertices of vertex  $v$ , denoted by  $N_{i,v}$  (line 6) and the agents' ambiguity levels w.r.t.  $\sigma \in \Sigma_c$  for  $v$  (lines 7 - 8). At that point (lines 10 - 15), the neighboring vertices for each vertex are built from the  $\sim_i$  relation via subset construction of  $\mathcal{U}$  w.r.t.  $i \in I$ .

One further initialization pass (lines 17 - 29) determines if any vertices can be distinguished at the outset. Such vertices are stored in  $R$ , and their ambiguity levels are set to 0. Note that if none of the vertices can be distinguished at this stage, no vertices can be resolved, and the algorithm will return false at line 49.

Lines 31 - 48 are where the inferencing cycle detection occurs. This loop also represents the manipulation of the



bipartite graph itself. If agent  $i$  can fully distinguish, i.e., resolve, a vertex  $v$ , then its set of neighboring vertices  $N_{i,v}$  is empty. Each iteration endeavors to remove edges from the bipartite graph, that is, edges with a label  $i$  between any vertices  $v'$  and  $v$ , since in a previous "round"  $i$  could distinguish  $v$ , no other agent needs to bother. If removing edges now renders the set of  $v'$ 's neighbors, specifically  $N_{i,v'}$ , empty, then  $v'$  is added to the list of resolved vertices for the next iteration. These lines repeat until there are no more vertices to resolve.

In line 49, the algorithm returns False if the number of distinguished vertices is less than the number of control configurations. But if all the control configurations w.r.t.  $\sigma$  are resolved (i.e., at least one agent knows the correct control decision to take with a finite level of ambiguity for each configuration), the procedure begins again at line 2 for the next controllable event. The algorithm returns True if all control configurations across  $\Sigma_c$  are resolvable.

In our algorithm, after initialization, the ambiguity level for an agent regarding vertex  $v \in (V_{D_\sigma} \cup V_{E_\sigma})$  is updated only when the edge set for  $v$  w.r.t. some  $i \in I_c(\sigma)$  is empty, i.e.,  $N_{i,v} = \emptyset$  (line 44 or 46). As a result, we define the ambiguity level for a vertex  $v \in V_{D_\sigma}$  or a vertex  $v \in V_{E_\sigma}$  as follows:

$$\mathbf{a}_i^D(v, \sigma) = \min\{k \in \mathbb{N} \mid N_{i,v} = \emptyset\}, \quad (12)$$

$$\mathbf{a}_i^E(v, \sigma) = \min\{k \in \mathbb{N} \mid N_{i,v} = \emptyset\}. \quad (13)$$

Overall, the computational complexity for Algorithm 1 is  $O(\sum_{\sigma \in \Sigma_c} (|I_c(\sigma)| \cdot |E_\sigma| \cdot |D_\sigma|))$ . Finally, we have implemented Algorithm 1 and have examined a wide variety of examples, including automata that generate regular languages with an infinite set of words, notably, languages that cannot be handled by Eq(1) – (4). We intend to share our implementation on GitHub in the near future.

*Ex. 1 concluded.* The bipartite graph  $G_\sigma$  is shown in Fig. 5. Elements of  $V_{E_\sigma}$  are on the top of the graph, while those in  $V_{D_\sigma}$  are on the bottom. Vertices indistinguishable to agent 1 are connected with the densely-dotted line, while those indistinguishable to agent 2 are connected with a loosely-dotted line. We use Algorithm 1 to confirm that the example is not inference observable. After the initialization loop in lines 4-8, the edges of the bipartite graph are constructed in lines 10 - 15. In particular, as can be deduced from both Table I and Fig. 5, when  $v = (4, 3, 3)$ , we have  $N_{1,v} = \{(4, 3, 3), (4, 3, 4), (4, 4, 3), (4, 4, 4), (3, 3, 3), (3, 3, 4), (3, 4, 3), (3, 4, 4)\}$  and  $N_{2,v} = \{(4, 3, 3), (4, 3, 4), (4, 4, 3), (4, 4, 4), (3, 3, 3)_A, (3, 3, 4)_A, (3, 4, 3)_A, (3, 4, 4)_A\}$ . All the other vertices have identically-sized (non-empty) sets of neighboring vertices. As a result, after  $R$  is initialized to  $\emptyset$  in line 17, it never changes, since line 23 is false for every single vertex in  $V_{D_\sigma} \cup V_{E_\sigma}$ . Consequently, at line 30, *Resolved* is empty, and line 31 is immediately False, thus terminating the program at line 49 and returning False.  $\blacklozenge$

*Proposition 3:* Algorithm 1 correctly verifies a violation of inference observability in the presence of state-based inferencing cycles.

*Proof sketch:* For simplicity, assume that  $\mathcal{U}$  has  $D_\sigma = \{\mathbf{d}\}$  and  $E_\sigma = \{\mathbf{e}\}$ , and contains an EBD state-inference cycle starting at  $\mathbf{d}$  (equivalently a DBD state-inference cycle starting at  $\mathbf{e}$ )  $\mathbf{d} \sim_{i_0} \mathbf{e} \sim_{i_1} \dots \mathbf{d} \sim_{i_j} \dots \sim_{i_m} \mathbf{d}$ , for  $i_k \in I_c(\sigma)$  and  $k \in \{0, \dots, m\}$ . Since there is no  $i \in I_c(\sigma)$  that can distinguish  $\mathbf{d}$  from  $\mathbf{e}$ , after Line 16 of the algorithm,  $N_{i,\mathbf{d}} = \{\mathbf{e}\}$  and  $N_{i,\mathbf{e}} = \{\mathbf{d}\}$ , for all  $i \in I_c(\sigma)$ . Since neither of these sets will ever reach Line 23, it will never be the case that either of these states will be added to *Resolved* in Line 29. As a result, when reaching Line 49, the algorithm will return False, since neither  $\mathbf{d}$  nor  $\mathbf{e}$  were resolved.

## B. Example

We conclude with a demonstration of our algorithm on a system that is inference-observable. Consider the system in Fig 6. By Eqns (1)– (4) this system is inference observable:

We first compute  $\mathcal{U}$ , which has 1881 states after the relabelling for duplicates in control configurations, including 90 states in  $D_\sigma$  and 45 in  $E_\sigma$ . There are 23 states in  $\mathcal{O}_{\Delta_0}$ , including 9 where control configurations of  $\mathcal{U}$  appear. As a result, we demonstrate Algorithm 1 on these 9 representative states. That is, one from each equivalence class of control configurations in  $\mathcal{O}_{\Delta_0}$ , with the understanding that the results for the other states in  $D_\sigma$  and  $E_\sigma$  will be identical.

Table II shows the nine representative states of  $\mathcal{U}$  on which we demonstrate the computation of ambiguity levels.

Recall that  $\sim_i$  is determined based on the states of  $\mathcal{O}_{\Delta_0}$ . The loop at line 2 will only be executed once since  $\Sigma_c = \{\sigma\}$ .

The loop at line 4 initializes the vertex sets to  $\emptyset$  and ambiguity levels for all agents and vertices to  $\infty$ . The bipartite graph construction occurs in the loop at line 10. For instance, for  $\mathbf{e} = (9, 9, 9, 9)_A \in V_{E_\sigma}$ ,  $N_{1,\mathbf{e}} = \{(10, 10, 10, 10)_A, (11, 11, 11, 11)\}$  because  $(9, 9, 9, 9)_A \sim_1 (10, 10, 10, 10)_A$  and  $(9, 9, 9, 9)_A \sim_1 (11, 11, 11, 11)$ . But  $N_{3,\mathbf{e}} = \emptyset$  because  $(9, 9, 9, 9)_A$  is not indistinguishable from any vertex in  $V_{D_\sigma}$ . As can be seen from Table II, this is the case for five other control configurations. Subsequently, the ambiguity levels for these states and the associated  $i \in I_c(\sigma)$  is set to 0, e.g.,  $\mathbf{a}_3^E((9, 9, 9, 9)_A, \sigma) = 0$ .

Going into the loop at line 31 for the first time, we have identified 6 vertices that are initially distinguishable by at least one  $i \in I_c(\sigma)$ . As a result, at line 30, *Resolved* contains the states indicated in the eighth line of Table II.

During Iteration 1, after removing the vertices in  $R$ , which is initialized to *Resolved* prior to the first iteration, vertex  $(11, 11, 11, 11)$  can be distinguished by agent 1 and vertex  $(10, 10, 10, 10)$  can be distinguished by agent 2, both with ambiguity level 1.

During Iteration 2, the remaining vertex  $(9, 9, 9, 9)$  is distinguished by all three agents with ambiguity level 2. And  $R = (9, 9, 9, 9)$  to begin the final iteration.

After the final iteration, once  $(9, 9, 9, 9)$  is removed from the remaining vertex sets, the last of the ambiguity levels are set to 3.

As can be seen from Table II, all the disablement decisions can be taken after one round of inferencing, i.e., four decisions are taken with an ambiguity level of 0 and the

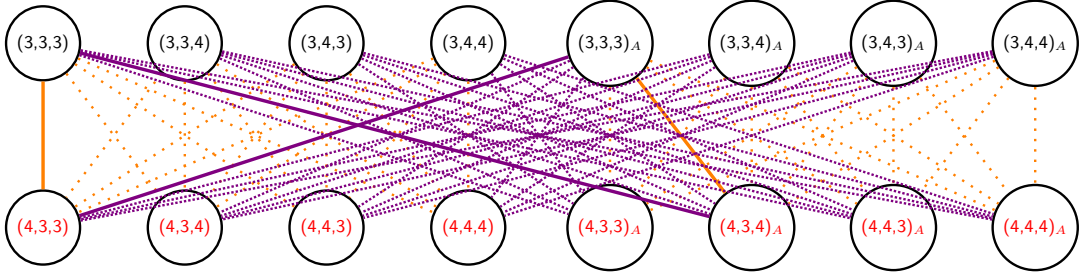


Fig. 5.  $G_\sigma$  for Ex. 1. Sample cycle indicated by solid lines:  $(3,3,3) \sim_1 (4,3,3) \sim_2 (3,3,3)_A \sim_1 (4,3,4)_A \sim_2 (3,3,3)$ .

TABLE II

VALUES OF  $N_{i,e}$  (LEFT) AND  $N_{i,d}$  (RIGHT) AFTER INITIALIZATION (LINE 29 OF ALGORITHM 1) AND EACH ITERATION OF THE WHILE LOOP (LINE 30 OF ALGORITHM 1) TO ITS TERMINATION. BOXES INDICATE EARLIEST ROUND IN WHICH THE CORRECT CONTROL DECISION CAN BE TAKEN BY A GIVEN AGENT.

	$e \in V_{E_e}$			$d \in V_{D_d}$					
	$(9,9,9)$	$(9,9,9)_A$	$(9,9,9)_B$	$(10,10,10)$	$(10,10,10)_A$	$(10,10,10)_B$	$(11,11,11)$	$(11,11,11)_A$	$(11,11,11)_B$
Initialization	$N_{1,e}$	$\{(10,10,10), (11,11,11)_A\}$	$\{(11,11,11), (10,10,10)_B\}$	$N_{1,d}$	$\{(9,9,9)\}$	$\{(9,9,9)_A\}$	$\{(9,9,9)_B\}$	$\{(9,9,9)_A\}$	$\{(9,9,9)_B\}$
	$N_{2,e}$	$\{(10,10,10)_B, (11,11,11)\}$	$\{(11,11,11)_A, (10,10,10)_A\}$	$N_{2,d}$	$\{(9,9,9)_B\}$	$\{(9,9,9)_A\}$	$\{(9,9,9)\}$	$\{(9,9,9)\}$	$\{(9,9,9)_A\}$
	$N_{3,e}$	$\{(10,10,10), (11,11,11)\}$	$\emptyset$	$\emptyset$	$N_{3,d}$	$\{(9,9,9)\}$	$\emptyset$	$\emptyset$	$\{(9,9,9)\}$
	$a_1^E(e,\sigma)$	$\infty$	$\infty$	$a_1^D(d,\sigma)$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
	$a_2^E(e,\sigma)$	$\infty$	$\infty$	$a_2^D(d,\sigma)$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
	$a_3^E(e,\sigma)$	$\infty$	$\infty$	$a_3^D(d,\sigma)$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
	$Resolved = \{(9,9,9)_A, (9,9,9)_B, (10,10,10)_A, (10,10,10)_B, (11,11,11)_A, (11,11,11)_B\}$								
Iteration 1	$N_{1,e}$	$\{(10,10,10)\}$	$\{(11,11,11)\}$	$N_{1,d}$	$\{(9,9,9)\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\{(9,9,9)\}$
	$N_{2,e}$	$\{(11,11,11)\}$	$\emptyset$	$N_{2,d}$	$\emptyset$	$\emptyset$	$\{(9,9,9)\}$	$\{(9,9,9)\}$	$\emptyset$
	$N_{3,e}$	$\{(10,10,10), (11,11,11)\}$	$\emptyset$	$N_{3,d}$	$\{(9,9,9)\}$	$\emptyset$	$\emptyset$	$\{(9,9,9)\}$	$\emptyset$
	$a_1^E(e,\sigma)$	$\infty$	$\infty$	$a_1^D(d,\sigma)$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
	$a_2^E(e,\sigma)$	$\infty$	$\infty$	$a_2^D(d,\sigma)$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
	$a_3^E(e,\sigma)$	$\infty$	$\infty$	$a_3^D(d,\sigma)$	$\infty$	$\infty$	$\infty$	$\infty$	$\infty$
	$Resolved = \{(9,9,9)_A, (9,9,9)_B, (10,10,10)_A, (10,10,10)_B, (11,11,11)_A, (11,11,11)_B, (10,10,10)\}$								
Iteration 2	$N_{1,e}$	$\emptyset$	$\emptyset$	$N_{1,d}$	$\{(9,9,9)\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\{(9,9,9)\}$
	$N_{2,e}$	$\emptyset$	$\emptyset$	$N_{2,d}$	$\emptyset$	$\emptyset$	$\{(9,9,9)\}$	$\{(9,9,9)\}$	$\emptyset$
	$N_{3,e}$	$\emptyset$	$\emptyset$	$N_{3,d}$	$\{(9,9,9)\}$	$\emptyset$	$\emptyset$	$\{(9,9,9)\}$	$\emptyset$
	$a_1^E(e,\sigma)$	2	2	$a_1^D(d,\sigma)$	$\infty$	1	1	1	$\infty$
	$a_2^E(e,\sigma)$	2	1	$a_2^D(d,\sigma)$	1	1	$\infty$	$\infty$	1
	$a_3^E(e,\sigma)$	2	0	$a_3^D(d,\sigma)$	$\infty$	0	0	$\infty$	0
	$Resolved = \{(9,9,9)_A, (9,9,9)_B, (10,10,10)_A, (10,10,10)_B, (11,11,11)_A, (11,11,11)_B, (11,11,11), (10,10,10), (9,9,9)\}$								
Iteration 3	$N_{1,e}$	$\emptyset$	$\emptyset$	$N_{1,d}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
	$N_{2,e}$	$\emptyset$	$\emptyset$	$N_{2,d}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
	$N_{3,e}$	$\emptyset$	$\emptyset$	$N_{3,d}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
	$a_1^E(e,\sigma)$	2	2	$a_1^D(d,\sigma)$	3	1	1	1	3
	$a_2^E(e,\sigma)$	2	1	$a_2^D(d,\sigma)$	1	1	3	3	1
	$a_3^E(e,\sigma)$	2	0	$a_3^D(d,\sigma)$	3	0	0	3	0

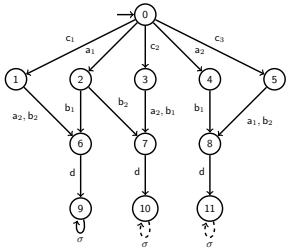


Fig. 6. Joint  $\mathcal{M}_L$  (all transitions) and  $\mathcal{M}_K$  (only solid-line transitions) where  $n = 3$ ,  $\Sigma_{o,1} = \{a_1, a_2, d\}$ ,  $\Sigma_{o,2} = \{b_1, b_2, d\}$ ,  $\Sigma_{o,3} = \{c_1, c_2, c_3, d\}$ , and  $I_c(\sigma) = \{1, 2, 3\}$ .

other two are taken with an ambiguity of level 1. This is consistent with the result from applying Eqns (1), (3) and (2), (4) to  $E_0(\sigma)$  and  $D_0(\sigma)$  for this example, and can be shown to be consistent with the ambiguity levels computed by Eqns. (5) and (6).

## V. CONCLUSION

We have presented a new way of thinking about inferencing in decentralized discrete-event systems. Specifically,

we explored the idea of inferencing cycles and their role in determining when an inferencing solution exists. Most notably, when an inferencing solution exists, our algorithm computes all the ambiguity values to express an overall solution in either the EBD or DBD architectures. Note that our algorithm asks and answers a different question than that of [4], and for that reason our algorithmic results are incomparable. Our new understanding of inferencing cycles opens up new avenues we began to explore in decentralized architectures beyond inferencing [9].

## REFERENCES

- [1] K. Rudie and W. M. Wonham, "Think globally, act locally: Decentralized supervisory control," *IEEE Trans. Autom. Control*, vol. 37, no. 11, pp. 1692–1708, 1992.
- [2] T.-S. Yoo and S. Lafontaine, "Decentralized supervisory control with conditional decisions: supervisor existence," *IEEE Trans. Autom. Control*, vol. 49, no. 11, pp. 1886–1904, 2004.
- [3] R. Kumar and S. Takai, "Inference-based ambiguity management in decentralized decision-making: Decentralized control of discrete event systems," *IEEE Trans. Autom. Control*, vol. 52, no. 10, pp. 1783–94, 2007.

- [4] S. Takai and R. Kumar, "A generalized framework for inference-based diagnosis of discrete event systems capturing both disjunctive and conjunctive decision-making," *IEEE Trans. Autom. Control*, vol. 62, no. 6, pp. 2778–2793, 2017.
- [5] M. Rabin and D. Scott, "Finite automata and their decision problems," *IBM Journal of Research and Development*, pp. 114–125, 1959.
- [6] S. Ricker and H. Marchand, "A parity-based architecture for decentralized discrete-event control," in *Proc. ACC*, 2013, pp. 5678–84.
- [7] S. L. Ricker and B. Caillaud, "Mind the gap: Expanding communication options in decentralized discrete-event control," *Automatica*, 2011.
- [8] A. Arnold, *Finite transition systems*. Prentice–Hall, 1994.
- [9] S. Ricker, T. Lidbetter, and H. Marchand, "Inferencing and beyond: further adventures with parity-based architectures for decentralized discrete-event systems," in *IFAC Proceedings Online*, 2017.