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

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

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.

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

II. Notation

Let Σ be a finite alphabet, and Σ^* be the set of words formed from elements of Σ, where ϵ 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 Σ.

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, Σ 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 $I = \{1, \ldots, n\}$ 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.

To facilitate partial observation, we partition the alphabet Σ into disjoint sets of unobservable events Σ_{uo} and observable events Σ_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* $\pi_i : \Sigma^* \to \Sigma_{o,i}^*$ defines the observations of an agent $i \in I_0$ by removing all occurrences of events in $\Sigma \backslash \Sigma_{o,i}$ from a sequence in Σ^*. 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 $[\![w]\!]_i$ whenever we refer to $\pi_i^{-1}\pi_i(w) \cap L$, for $i \in I_0$.

Similarly, for control purposes, Σ is partitioned into controllable events Σ_c and uncontrollable events Σ_{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 $I_c(\sigma) = \{i \in I | \sigma \in \Sigma_{c,i}\}$.

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[1] and words after which a controllable event must be enabled. Thus, for all $\sigma \in \Sigma_c$:

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

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

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

Each subsequent round of reasoning (for $k \geq 0$) is inductively defined[2]

$$D_{k+1}(\sigma) = D_k(\sigma) \cap \big(\cap_{i \in I_c(\sigma)} [\![E_k(\sigma)]\!]_i \big); \tag{3}$$

$$E_{k+1}(\sigma) = E_k(\sigma) \cap \big(\cap_{i \in I_c(\sigma)} [\![D_k(\sigma)]\!]_i \big). \tag{4}$$

Remember that $[\![E_k(\sigma)]\!]_i$ 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. σ for which no agent has yet inferred the correct disablement decision. And equivalently for $[\![D_k(\sigma)]\!]_i$.

*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 $k-1$, 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:

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

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

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

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

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, π_i and $\Sigma_{c,i}$ (for $i \in I$) if either*

$$\begin{aligned}
&(\forall w_0 \in K)(\forall \sigma \in \Sigma_c)\ w_0\sigma \in L \setminus K \Rightarrow (\exists m \in \mathbb{N}) \\
&\quad (\forall i_0 \in I_c(\sigma))(\forall w_1\sigma \in [\![w_0]\!]_{i_0}\sigma \cap K) \\
&\qquad [\![w_1]\!]_{i_0}\sigma \cap L \setminus K \neq \emptyset \Rightarrow \\
&\qquad\quad (\exists i_0' \in I_c(\sigma))(\forall w_1\sigma \in [\![w_0]\!]_{i_0'}\sigma \cap K) \\
&\qquad\qquad [\![w_1]\!]_{i_0'}\sigma \cap L \setminus K \neq \emptyset \Rightarrow \\
&\qquad\quad (\forall i_1 \in I_c(\sigma))(\forall w_2\sigma \in [\![w_1]\!]_{i_0'}\sigma \cap L \setminus K) \\
&\qquad\qquad [\![w_2]\!]_{i_1}\sigma \cap K \neq \emptyset \Rightarrow \\
&\qquad\qquad\quad (\exists i_1' \in I_c(\sigma))(\forall w_2\sigma \in [\![w_1]\!]_{i_1'}\sigma \cap L \setminus K) \\
&\qquad\qquad\qquad [\![w_2]\!]_{i_1'}\sigma \cap K \neq \emptyset \Rightarrow \\
&\qquad\qquad (\forall i_2 \in I_c(\sigma))(\forall w_3\sigma \in [\![w_2]\!]_{i_1'}\sigma \cap K) \\
&\qquad\qquad [\![w_3]\!]_{i_2}\sigma \cap L \setminus K \neq \emptyset \Rightarrow \\
&\qquad\qquad\qquad \cdots
\end{aligned} \tag{8}$$

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

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

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

or

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

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 $w_0, w_1, w_2, \ldots, w_{m-1}, w_m$ and associated ordering of agents $i_0, i_1, \ldots, i_{m-1}, i_m$ (where $i_j \neq i_{j+1}$, for $j \in \{0, \ldots, m-1\}$) 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 $[\![w_m]\!]_{i_m}\sigma$, allows i_m to take a definitive disablement decision on σ (with an ambiguity level of 0), while agent i'_0 takes a disablement decision on the words in $[\![w_0]\!]_{i'_0}$ 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 $e_1\sigma \in [\![d_0]\!]_{i_0}\sigma \cap K$. Agent i_0 delegates to all $i_1 \in I_c(\sigma) \setminus \{i_0\}$ to see if $[\![e_1]\!]_{i_1}\sigma \cap L \setminus K = \emptyset$. 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 $[\![d_m]\!]_{i_m}\sigma \cap K = \emptyset$, ie. agent i_m can take the definitive disablement decision about σ with an ambiguity level of 0 for the words in $[\![d_m]\!]_{i_m}\sigma$.

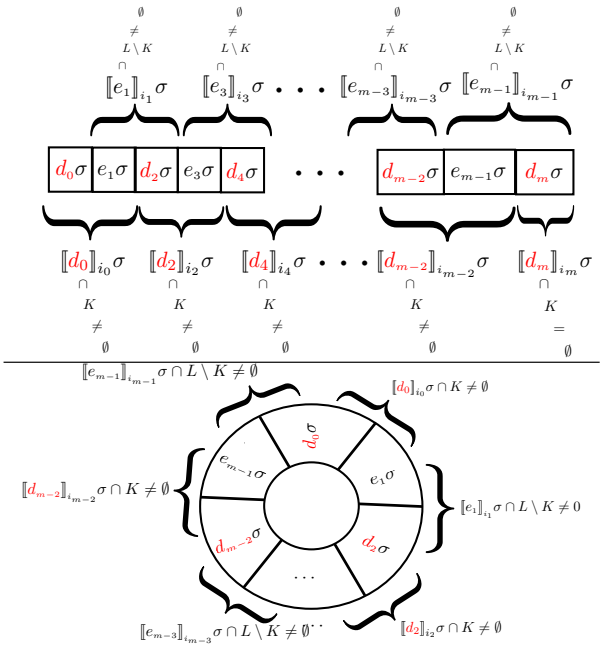

Fig. 1. (Top) Inferencing rounds that result in a successful disablement control decision for σ by agent i_m after it observes $[\![d_m]\!]_{i_m}$. (Bottom) An inferencing cycle for $d_0\sigma$, for $i_0, \ldots, i_{m-1} \in I_c(\sigma)$.

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 $j \in \{0, \ldots, m-1\}$), 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:

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

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 $I_J = \{i_0, \ldots, i_j, \ldots, i_m\}$ such that $[\![w_m]\!]_{i_m} = [\![w_j]\!]_{i_j}$ for the EBD inferencing decision-making process such that $[\![w_j]\!]_{i_j}\sigma \cap K \neq \emptyset$ (and similarly for the EBD inferencing decision-making process). Thus, by iterating these cycles of agent choices $(i_j, \ldots, i_{m-1}, i_j)$, we can build infinite sequences of decisions for each round $(i_0, \ldots i_{j-1}).(i_j, \ldots, i_{m-1})^k$, 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. ■

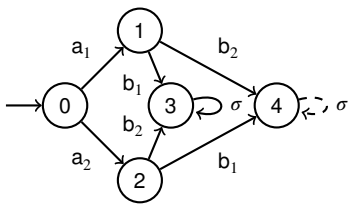

Fig. 2. Joint $\mathcal{M}_L$ (all transitions) and $\mathcal{M}_K$ (only solid-line transitions) for Ex. 1 where $n = 2$, $\Sigma_{o,1} = \{a_1, a_2\}$, $\Sigma_{o,2} = \{b_1, b_2\}$, and $I_c(\sigma) = \{1, 2\}$.

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:

$$
\begin{aligned}
D_0(\sigma) &= \{a_1b_2, a_2b_1\}, \quad E_0(\sigma) = \{a_1b_1, a_2b_2\};\\
D_1(\sigma) &= D_0(\sigma), \qquad\qquad E_1(\sigma) = E_0(\sigma).
\end{aligned}
$$

Consequently, by Def. 3, we can find at least one EBD and DBD inferencing cycle. For the EBD cycle, let $w_0\sigma = a_1b_2\sigma$ and $i'_0 = 1$. We have an inferencing cycle for $m = 4$ as follows.

$$
\begin{aligned}
&[\![a_1b_2]\!]_1\sigma \cap K \Rightarrow w_1 = a_1b_1, i'_1 = 2,\\
&\quad [\![a_1b_1]\!]_2\sigma \cap L \setminus K \Rightarrow w_2 = a_2b_1, i'_2 = 1,\\
&\qquad [\![a_2b_1]\!]_1\sigma \cap K \Rightarrow w_3 = a_2b_2, i'_3 = 2,\\
&\qquad\quad [\![a_2b_2]\!]_2\sigma \cap L \setminus K \Rightarrow w_4 = a_1b_2, i_4 = 1,\\
&\qquad\qquad [\![w_4]\!]_{i_4}\sigma = [\![w_0]\!]_{i'_0}\sigma, i'_0 = i_4 \Rightarrow m = 4, j = 0
\end{aligned}
$$

Equivalently, we have a DBD cycle by starting with $w_0\sigma = a_1b_1\sigma$ 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 $\mathcal{M}_L^\epsilon$, which is simply $\mathcal{M}_L$ augmented with self-loops of ϵ at each state:

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

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

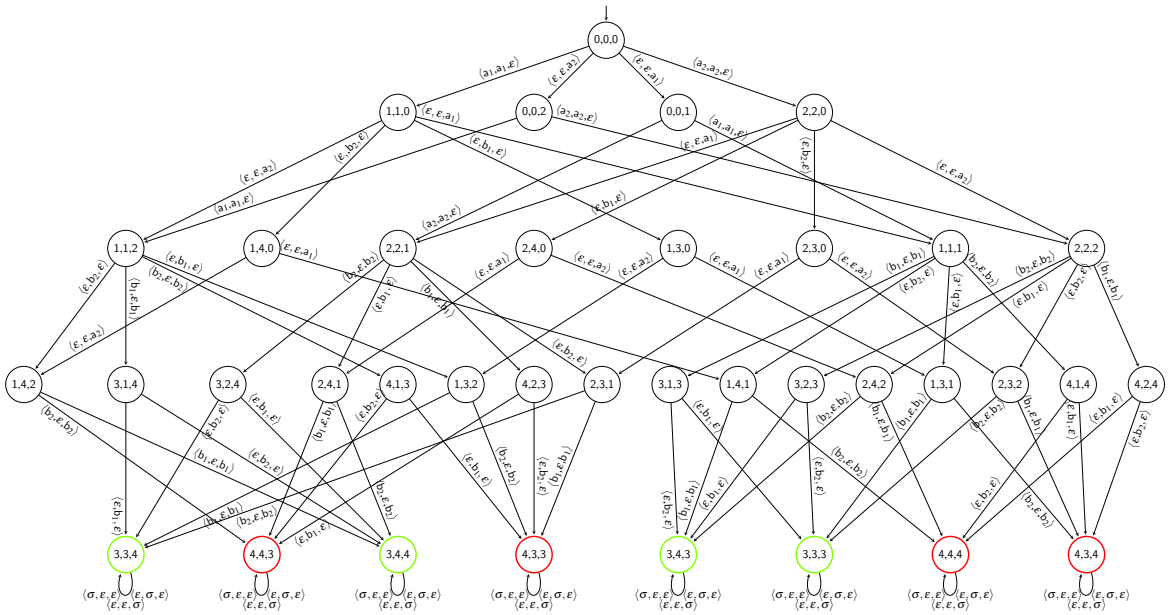

Fig. 3. The $\mathcal{U}$ structure for Ex. 1. Selfloops of ℓ^{ϵ} at each state and labels involving σ at non-configuration states omitted for readability.

Ex. 1 cont. The $\mathcal{U}$ structure for Ex. 1 is shown in Fig. 3.

We focus on states of $\mathcal{U}$, where a control decision for $\sigma \in \Sigma_c$ must be taken, which we call *control configurations*.

Definition 4: (a) The set of ***control configurations for σ w.r.t. disablement*** *for $\mathcal{U}$ is a set of states with outgoing transitions that have labels involving $\sigma \in \Sigma_c$ as follows.*

$$\mathsf{D}_\sigma = \{q_S \in Q_S | (\exists q'_S \in Q_S) \text{ s.t. } (q_S(0), \sigma, q'_S(0)) \in T_L \setminus T_K \wedge (\forall i \in I_c(\sigma))(q_S(i), \sigma, q'_S(i)) \in T_L\}.$$

(b) The set of ***control configurations for σ w.r.t. enablement*** *for $\mathcal{U}$ is a set of states with outgoing transitions that have labels involving $\sigma \in \Sigma_c$ as follows.*

$$\mathsf{E}_\sigma = \{q_S \in Q_S | (\exists q'_S \in Q_S) \text{ s.t. } (q_S(0), \sigma, q'_S(0)) \in T_K \wedge (\forall i \in I_c(\sigma))(q_S(i), \sigma, q'_S(i)) \in T_L\}.$$

Note that the states in Q_S represent a state-based "snapshot" of system behavior and the response of each agent $i \in I$ to the observed behavior, i.e., the state agent i believes the system is in. Let $\mathsf{d} \in \mathsf{D}_\sigma$ and w be a word in the language of $\mathcal{U}$ such that $(q_{0_S}, w, \mathsf{d}) \in T_S$. Thus, $(q_{0_S}(0), w(0), \mathsf{d}(0)) \in T_K$ and $\exists q'_S \in Q_S$ such that $(q_{0_S}(0), w\sigma(0), q'_S(0)) \in T_L \setminus T_K$. By construction, $w(i) \in [\![w(0)]\!]_i$, for all $i \in I$. Thus, some agents in $I_c(\sigma)$ consider that the system will remain in K, i.e., $w(i)\sigma \in K$, while others consider that the system is about to leave K, i.e., $w(i)\sigma \in L \setminus K$. By construction, Q_S includes states of the form $q_S \in Q_S$ where $(q_{0_S}(0), w(0), q_S(0) = \mathsf{d}(0))$ and $(q_{0_S}(i), w(i), q_S(i))$ for all possible combinations of $w(i) \in [\![w(0)]\!]_i$, for all $i \in I$, although we are only interested in the states where for all $i \in I_c(\sigma)$ we have $q_S(i) \in \mathsf{D}_\sigma \cup \mathsf{E}_\sigma$. In lieu of inferencing on sequences, we want to inference on control configurations. Although state d requires an overall control decision of disable, based on the behavior of σ at $\mathsf{d}(0)$, we want to be able to discuss the states in E_σ that each $i \in I_c(\sigma)$ cannot distinguish from d. To capture these aspects, we shall use the natural projection over each alphabet of each agent to capture the states that are indistinguishable for agent i in $\mathcal{U}$.

The natural projection extends to automata via the determinization algorithm, i.e., the subset relation. To represent the partial observation of $\mathcal{U}$ w.r.t. agent $i \in I_0$, we construct its determinization, denoted by $\mathcal{O}_{\Delta_i}$, by first replacing all the transitions in $\mathcal{U}$ that have a label $\ell \in \Delta_{uo,i}$ with ℓ^{ϵ}. Subsequently, we apply the subset relation. More formally, we define the ϵ-reach of a state $q_S \in Q_S$, w.r.t. alphabet $\Delta_i \subseteq \Delta_S$ as $\epsilon\text{-reach}(q_S, \Delta_i) = \{q'_S \in Q_S | (\exists u \in (\Delta_S \setminus \Delta_i)^*)(q_S, u, q'_S) \in T_S)\}$. Then we have

$$\mathcal{O}_{\Delta_i} = (X_i, \Delta_i, T_{\Delta_i}, x_{i,0}),$$

where $X_i \subseteq 2^{Q_S}$; $x_{i,0} = \epsilon\text{-reach}(q_{0,S}, \Delta_i)$; and $(x_i, \ell, x'_i) \in T_{\Delta_i}$, where $x'_i = \bigcup_{q_S \in x} \epsilon\text{-reach}(q'_S, \Delta_i)$ s.t. $(q_S, \ell, q'_S) \in T_S$.

Based on these new automata $\mathcal{O}_{\Delta_i}$, we can determine the states of $\mathcal{U}$ that are indistinguishable for agent i. Such states will correspond to sequences that are equal up to the projection over the observable alphabet of i.

Definition 5: Given $q_S, q'_S \in Q_S$, q_S is ***indistinguishable*** *from q'_S for agent $i \in I$, denoted $q_S \sim_i q'_S$, iff $\exists x \in X_i$ s.t. $q_S, q'_S \in x$.*

We abuse notation and let $[\![q_S]\!]_i$ represent the set of states in Q_S that agent i cannot distinguish from q_S under $\sim_i$.

Note that we can cast inferencing and inferencing cycles into their state-based equivalents. Indeed, we can redefine Def. 3 in terms of states in $\mathcal{U}$ and $\mathcal{O}_{\Delta_i}$ as follows.

Definition 6: The control configuration $\mathsf{d} \in \mathsf{D}_\sigma$ gives rise to an ***EBD state-inferencing cycle*** *if:*

$$
\begin{aligned}
&(\exists \mathsf{d}_0 \in Q_S)(\exists \sigma \in \Sigma_c)\ \mathsf{d}_0 \in \mathsf{D}_\sigma \Rightarrow (\exists m \in \mathbb{N})\\
&\quad (\forall i_0 \in I_c(\sigma))(\forall \mathsf{e}_1 \in [\![\mathsf{d}_0]\!]_{i_0} \cap \mathsf{E}_\sigma)\\
&\qquad [\![\mathsf{e}_1]\!]_{i_0} \cap \mathsf{D}_\sigma \neq \emptyset \Rightarrow\\
&\qquad\quad (\exists i'_0 \in I_c(\sigma))(\forall \mathsf{e}_1 \in [\![\mathsf{d}_0]\!]_{i'_0} \cap \mathsf{E}_\sigma)\\
&\qquad\qquad [\![\mathsf{e}_1]\!]_{i'_0} \cap \mathsf{D}_\sigma \neq \emptyset \Rightarrow\\
&\qquad (\forall i_1 \in I_c(\sigma) \setminus \{i'_0\})(\forall \mathsf{d}_2 \in [\![\mathsf{e}_1]\!]_{i'_0} \cap \mathsf{D}_\sigma)\\
&\qquad\quad [\![\mathsf{d}_2]\!]_{i_1} \cap \mathsf{E}_\sigma \neq \emptyset \Rightarrow\\
&\qquad\qquad (\exists i'_1 \in I_c(\sigma) \setminus \{i'_0\})(\forall \mathsf{d}_2 \in [\![\mathsf{e}_1]\!]_{i'_0} \cap \mathsf{D}_\sigma)\\
&\qquad\qquad\quad [\![\mathsf{d}_2]\!]_{i'_1} \cap \mathsf{E}_\sigma \neq \emptyset \Rightarrow\\
&\qquad\qquad\qquad \ldots\\
&\qquad\qquad\quad (\exists i'_{m-1} \in I_c(\sigma) \setminus \{i'_{m-2}\})(\forall \mathsf{d}_m \in [\![\mathsf{e}_{m-1}]\!]_{i'_{m-1}} \cap \mathsf{D}_\sigma)\\
&\qquad\qquad\qquad [\![\mathsf{d}_m]\!]_{i'_{m-1}} \cap \mathsf{E}_\sigma \neq \emptyset \Rightarrow\\
&\qquad\quad (\forall i_m \in I_c(\sigma) \setminus \{i'_{m-1}\})(\forall \mathsf{e}_{m+1} \in [\![\mathsf{d}_m]\!]_{i_m} \cap \mathsf{E}_\sigma)\\
&\qquad\qquad [\![\mathsf{e}_{m+1}]\!]_{i_m} \cap \mathsf{D}_\sigma \neq \emptyset \Rightarrow\\
&\qquad\qquad\quad (\exists j \in \{0, 1, \ldots, m-1\}\\
&\qquad\qquad\qquad [\![\mathsf{d}_m]\!]_{i_m} = [\![\mathsf{d}_j]\!]_{i'_j}.
\end{aligned}
$$

Instead of writing $\mathsf{d}_0 \sim_{i_0} \mathsf{e}_1 \wedge \mathsf{e}_1 \sim_{i_1} \mathsf{d}_2 \wedge \ldots \wedge \mathsf{e}_{m-1} \sim_{i_{m-1}} \mathsf{d}_0$, *for* $i_0, \ldots, i_m \in I_c(\sigma)$, *we will use the following notation to denote a specific inferencing cycle of* d_0: $\mathsf{d}_0 \sim_{i_0} \mathsf{e}_1 \sim_{i_1} \mathsf{d}_2 \sim_{i_2} \ldots \sim_{i_{m-1}} \mathsf{e}_{m-1} \sim_{i_{m-1}} \mathsf{d}_0$.

We can equivalently define a DBD state-inferencing cycle for some control configuration $\mathsf{e} \in \mathsf{E}_\sigma$.

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_σ and E_σ, we compute $\mathcal{O}_{\Delta_0}$, where $\Delta_0 = \Delta_o$. 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_σ and D_σ. 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 $\mathsf{d} \in \mathsf{D}_\sigma$ and $\mathsf{e} \in \mathsf{E}_\sigma$ 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:

$$\mathsf{D}_\sigma = \{(4,3,3), (4,3,4), (4,4,3), (4,4,4)\}; \quad (10)$$

$$\mathsf{E}_\sigma = \{(3,3,3), (3,3,4), (3,4,3), (3,4,4)\}. \quad (11)$$

Note that in Fig. 2 some words have different projections w.r.t. π_i (for $i = \{1, 2\}$), but happen to go to the same state. For example, $\mathsf{a}_1\mathsf{b}_1$ and $\mathsf{a}_2\mathsf{b}_2$ 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 $w = \langle \mathsf{a}_1, \mathsf{a}_1, \varepsilon\rangle\langle \varepsilon, \mathsf{b}_1, \varepsilon\rangle\ \langle \varepsilon, \varepsilon, \mathsf{a}_2\rangle\langle \mathsf{b}_1, \mathsf{b}_1, \varepsilon\rangle$ goes to control configuration state $(3, 3, 4)$. But so does the word $w' = \langle \mathsf{a}_2, \mathsf{a}_2, \varepsilon\rangle\langle \varepsilon, \varepsilon, \mathsf{a}_1\rangle\ \langle \varepsilon, \mathsf{b}_2, \varepsilon\rangle\langle \mathsf{b}_2, \mathsf{b}_2, \varepsilon\rangle$. We then construct $\mathcal{O}_{\Delta_0}$, where $\Delta_0 = \{\langle \mathsf{a}_1, \mathsf{a}_1, \varepsilon\rangle, \langle \mathsf{a}_2, \mathsf{a}_2, \varepsilon\rangle, \langle \mathsf{b}_1, \varepsilon, \mathsf{b}_1\rangle, \langle \mathsf{b}_2, \varepsilon, \mathsf{b}_2\rangle\}$. Because $\pi_0(w(0)) \neq \pi_0(w'(0))$, the subset relation over Δ_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_σ (Eq. (10)) and E_σ (Eq. (11)) appears in two different states of $\mathcal{O}_{\Delta_0}$, x_{03} and x_{05} for E_σ, and x_{04} and x_{06} for D_σ. So we label the copies with a subscript of 'A' to distinguish them from the originals. The relabeled states of $\mathcal{O}_{\Delta_0}$ are presented in the leftmost column of Table I.

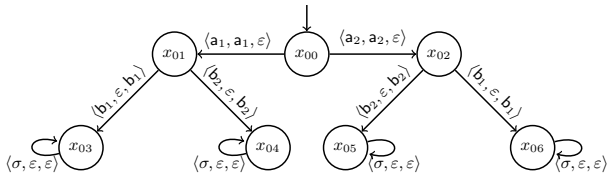

Fig. 4. $\mathcal{O}_{\Delta_0}$ for Ex. 1.

TABLE I

STATE SETS X_0 (LEFT), X_1 (CENTER) AND X_2 (RIGHT) FOR EX. 1. THE CONTROL CONFIGURATIONS E_σ IDENTIFIED IN GREEN AND D_σ 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), (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_{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_{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)$_A$, (3,3,4)$_A$,(3,4,3)$_A$,(3,4,4)$_A$, (4,2,3),(4,2,4),(4,3,3)$_A$, (4,3,4)$_A$,(4,4,3)$_A$, (4,4,4)$_A$}		
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)$_A$, (4,3,4)$_A$, (4,4,3)$_A$, (4,4,4)$_A$}
x_{04}	{(4,1,3),(4,1,4),(4,3,3), (4,3,4),(4,4,3),(4,4,4)}				
x_{05}	{(3,2,3),(3,2,4),(3,3,3)$_A$, (3,3,4)$_A$,(3,4,3)$_A$,(3,4,4)$_A$}			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)$_A$, (3,3,4)$_A$, (3,4,3)$_A$,(3,4,4)$_A$}
x_{06}	{(4,2,3),(4,2,4),(4,3,3)$_A$, (4,3,4)$_A$,(4,4,3)$_A$, (4,4,4)$_A$}				

After relabeling, $\mathsf{E}_\sigma = \{(3,3,3),(3,3,4),(3,4,3),(3,4,4), (3,3,3)_A,(3,3,4)_A,(3,4,3)_A,(3,4,4)_A\}$ and $\mathsf{D}_\sigma = \{(4,3,3), (4,3,4),(4,4,3),(4,4,4),(4,3,3)_A,(4,3,4)_A,(4,4,3)_A,(4,4,4)_A\}$.

We have $\Delta_1 = \{\langle \mathsf{a}_1, \mathsf{a}_1, \varepsilon\rangle, \langle \mathsf{a}_2, \mathsf{a}_2, \varepsilon\rangle\}$ and $\Delta_2 = \{\langle \mathsf{b}_1, \varepsilon, \mathsf{b}_1\rangle, \langle \mathsf{b}_2, \varepsilon, \mathsf{b}_2\rangle\}$. 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 $x_{21} \in X_2$. ♦

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(Σ_c, $\{I_c(\sigma)\}_{\sigma\in\Sigma_c}$, $\{(\mathsf{V}_{\mathsf{D}_\sigma}, \mathsf{V}_{\mathsf{E}_\sigma})\}_{\sigma\in\Sigma_c}$, $\{\sim_i\}_{i\in I_c(\sigma)}$)
2: **for all** $\sigma \in \Sigma_c$ **do**
3: ▷ *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.* ◁
4: **for all** $i \in I_c(\sigma)$ **do**
5: **for all** $v \in \mathsf{V}_{\mathsf{D}_\sigma} \cup \mathsf{V}_{\mathsf{E}_\sigma}$ **do**
6: $N_{i,v} \leftarrow \emptyset$
7: $\mathsf{a}_i^{\mathsf{D}}(v,\sigma) \leftarrow \infty$
8: $\mathsf{a}_i^{\mathsf{E}}(v,\sigma) \leftarrow \infty$
9: ▷ *Building edges of bipartite graph. This can be achieved in $O(|E_\sigma| \cdot |D_\sigma| \cdot |I_c(\sigma)|)$ time.* ◁
10: **for all** $\mathsf{d} \in \mathsf{V}_{\mathsf{D}_\sigma}$ **do**
11: **for all** $\mathsf{e} \in \mathsf{V}_{\mathsf{E}_\sigma}$ **do**
12: **for all** $i \in I_c(\sigma)$ **do**
13: **if** $\mathsf{d} \sim_i \mathsf{e}$ **then**
14: $N_{i,\mathsf{d}} \leftarrow N_{i,\mathsf{d}} \cup \{\mathsf{e}\}$
15: $N_{i,\mathsf{e}} \leftarrow N_{i,\mathsf{e}} \cup \{\mathsf{d}\}$
16: ▷ *R is the set of vertices in $V_{D_\sigma} \cup V_{E_\sigma}$ that can be distinguished by at least one $i \in I_c(\sigma)$* ◁
17: $R \leftarrow \emptyset$
18: ▷ *Initialize the ambiguity level to 0 and identify vertices that are immediately distinguishable before checking for cycles. This can be done in $O((|E_\sigma|+|D_\sigma|)\cdot|I_c(\sigma)|) + |E_\sigma| \cdot |D_\sigma|) \cdot |I_c(\sigma)|$ + time.* ◁
19: $ambLevel \leftarrow 0$
20: **for all** $v \in \mathsf{V}_{\mathsf{D}_\sigma} \cup \mathsf{V}_{\mathsf{E}_\sigma}$ **do**
21: **for all** $i \in I_c(\sigma)$ **do**
22: **if** $N_{i,v} = \emptyset$ **then**
23: $R \leftarrow R \cup \{v\}$
24: ▷ *Set appropriate ambiguity level for v to 0 w.r.t. agent i* ◁
25: **if** $v \in \mathsf{V}_{\mathsf{D}_\sigma}$ **then**
26: $\mathsf{a}_i^{\mathsf{D}}(v,\sigma) \leftarrow ambLevel$
27: **else**
28: $\mathsf{a}_i^{\mathsf{E}}(v,\sigma) \leftarrow ambLevel$
29: $Resolved \leftarrow R$
30: ▷ *Loop to detect cycles in G_σ, If G_σ has t_σ 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.* ◁
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: ▷ *Only add v' if it's not already resolved.* ◁
40: **if** $v' \notin Resolved$ **then**
41: $R' \leftarrow R' \cup \{v'\}$
42: ▷ *Set ambiguity level for v' and agent i* ◁
43: **if** $v' \in \mathsf{D}_\sigma$ **then**
44: $\mathsf{a}_i^{\mathsf{D}}(v,\sigma) \leftarrow ambLevel$
45: **else**
46: $\mathsf{a}_i^{\mathsf{E}}(v,\sigma) \leftarrow ambLevel$
47: $R \leftarrow R'$
48: $Resolved \leftarrow Resolved \cup R$
49: **if** $|Resolved| < |\mathsf{V}_{\mathsf{D}_\sigma}| + |\mathsf{V}_{\mathsf{E}_\sigma}|$ **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 Δ_0 to build $\mathcal{O}_{\Delta_0}$.
3) Check states of $\mathcal{O}_{\Delta_0}$ for duplicates of E_σ and D_σ, 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 (relabeled) states of E_σ and D_σ as vertices. An edge with label $i \in I_c(\sigma)$ exists between a vertex in $\mathsf{V}_{\mathsf{E}_\sigma}$ and a vertex in $\mathsf{V}_{\mathsf{D}_\sigma}$ 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. σ 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 Σ_c are resolvable.

In our algorithm, after initialization, the ambiguity level for an agent regarding vertex $v \in (\mathsf{V}_{\mathsf{D}_\sigma} \cup \mathsf{V}_{\mathsf{E}_\sigma})$ 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 $v \in \mathsf{V}_{\mathsf{D}_\sigma}$ or a vertex $v \in \mathsf{V}_{\mathsf{E}_\sigma}$ as follows:

$$\mathsf{a}_i^D(v,\sigma) = min\{k \in \mathbb{N} | N_{i,v} = \emptyset\}, \tag{12}$$

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

Overall, the computational complexity for Algorithm 1 is $O(\sum_{\sigma\in\Sigma_c}(|I_c(\sigma)|\cdot|\mathsf{E}_\sigma|\cdot|\mathsf{D}_\sigma|)$. 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_σ is shown in Fig. 5. Elements of $\mathsf{V}_{\mathsf{E}_\sigma}$ are on the top of the graph, while those in $\mathsf{V}_{\mathsf{D}_\sigma}$ 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 $\mathsf{V}_{\mathsf{D}_\sigma} \cup \mathsf{V}_{\mathsf{E}_\sigma}$. Consequently, at line 30, *Resolved* is empty, and line 31 is immediately False, thus terminating the program at line 49 and returning False. ♦

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 $\mathsf{D}_\sigma = \{\mathsf{d}\}$ and $\mathsf{E}_\sigma = \{\mathsf{e}\}$, and contains an EBD state-inference cycle starting at d (equivalently a DBD state-inference cycle starting at e) $d \sim_{i_0} \mathsf{e} \sim_{i_1} \ldots d \sim_{i_j} \ldots \sim_{i_m} d$, for $i_k \in I_c(\sigma)$ and $k \in \{0,\ldots,m\}$. Since there is no $i \in I_c(\sigma)$ that can distinguish d from e, after Line 16 of the algorithm, $N_{i,\mathsf{d}} = \{\mathsf{e}\}$ and $N_{i,\mathsf{e}} = \{\mathsf{d}\}$, 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 d nor 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_σ and 45 in E_σ. 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_σ and E_σ 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 ∞. The bipartite graph construction occurs in the loop at line 10. For instance, for $\mathsf{e} = (9,9,9,9)_A \in \mathsf{V}_\mathsf{E}$, $N_{1,\mathsf{e}} = \{$(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 $N_{3,\mathsf{e}} = \emptyset$ because $(9,9,9,9)_A$ is not indistinguishable from any vertex in $\mathsf{V}_{\mathsf{D}_\sigma}$. 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., $\mathsf{a}_3^E((9,9,9,9)_A,\sigma) = 0$.

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

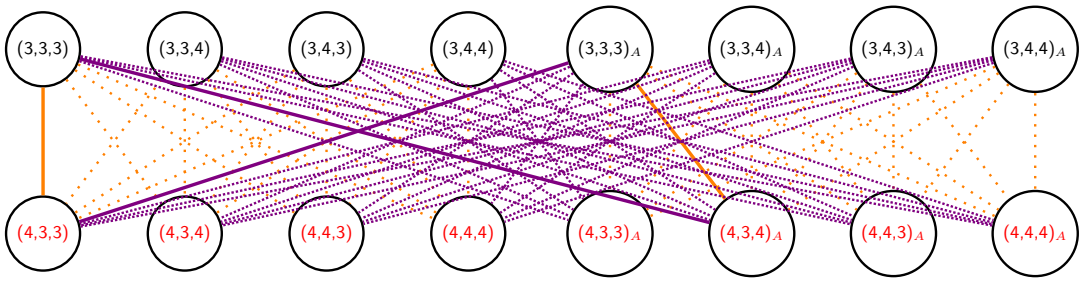

Fig. 5. G_σ for Ex. 1. Sample cycle indicated by solid lines: (3,3,3) $\sim_1$ (4,3,3) $\sim_2$ (3,3,3)$_A$ $\sim_1$ (4,3,4)$_A$ $\sim_2$ (3,3,3).

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.

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

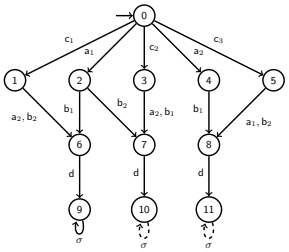

Fig. 6. Joint $\mathcal{M}_L$ (all transitions) and $\mathcal{M}_K$ (only solid-line transitions) where $n = 3$, $\Sigma_{o,1} = \{a_1, a_2, d\}$, $\Sigma_{o,2} = \{b_1, b_2, d\}$, $\Sigma_{o,3} = \{c_1, c_2, c_3, d\}$, and and $I_c(\sigma) = \{1, 2, 3\}$.

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 Eqs. (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. Lafortune, "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.