A New Verification Algorithm for Inferencing in Supervisory Control of Discrete-Event Systems

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

A New Verification Algorithm for Inferencing in Supervisory Control of Discrete-Event Systems

C.K. Sharpe * **S.H. Yoon** * **S.L. Ricker** * and **H. Marchand** **

* *Mount Allison University, Sackville NB, CANADA(e-mail: {cksharpe,syoon2,lricker}@mta.ca).*
** *University of Rennes, Inria, CNRS and IRISA, Rennes, FRANCE (e-mail: herve.marchand@inria.fr).*

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 contains no *inferencing cycles*. When there are no cycles, the algorithm computes the smallest upper bound for an inferencing solution.

1. INTRODUCTION

Decentralized decision-making in supervisory control can involve direct knowledge of the correct control decision (Rudie and Wonham, 1992), a one-level inference about another agent's direct knowledge (Yoo and Lafortune, 2004) or multi-level inferencing involving multiple rounds of inferencing among the agents (Takai and Kumar, 2008). 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 Yoo and Lafortune (2004) noted, these architectures are incomparable. However, a general decision-making framework that combined the two in the context of diagnosis was presented in Takai and Kumar (2017).

The inferencing definition introduced by Takai and Kumar (2008) assumes the existence of two non-increasing sequences of regular languages defined over a set of controllable events. The first consists of words after which a disablement decision must occur for a given controllable event. The other has words after which an enablement decision must occur for the same controllable event. To have an N-level inferencing solution, at least one of these sequences of languages must eventually be empty in $N+1$ steps. Implicit in this definition is the assumption that multi-agent inferencing is defined only over finite languages up to self-loops.

A verification algorithm for the inferencing definition introduced by Takai and Kumar (2008) does not appear in the literature; however, Takai and Kumar (2017, 2018) introduce an algorithm to assert whether a pair of languages is N-inference *diagnosable*,[1] where the user provides N. These algorithms hinge on the observation that establishing the non-emptiness after N rounds of inferencing of both of the sets of failure sequences and the non-failure sequences can be characterized as the existence of $2k+1$ traces in the language, where $1 \leq k \leq N+1$, which have specific relationships to each other. The algorithm is formulated for only two agents, while the authors claim that extending it to more than two agents is a straightforward exercise. In particular, the algorithm's computational complexity depends largely on N and not the number of agents.

We are targeting the verification of inferencing in the control domain. Our state-based algorithm first asserts that an inferencing strategy exists, i.e., the existence of an upper bound on the number of rounds required to ensure that we take all the correct control decisions. The current version of our algorithm determines the feasibility of constructing a control solution by computing such a bound if it exists. Our algorithm can be further extended to calculate a set of offline state-based local and global control decisions by adapting the language-based strategy of Takai and Kumar (2008), which we do not present here.

Our algorithm uses a finite-state structure for verifying properties of decentralized architectures (Ricker and Marchand, 2013). Subsequently, we derive a bipartite graph from the states of this structure to examine the relationships between the bipartition of states where decentralized decision-makers must make disablement decisions and indistinguishable states where enablement decisions occur. In particular, when specific cycles are present in the bipartite graph, this corresponds to an inferencing cycle in which no agent can make the correct control decision or, equivalently, the lack of an inferencing bound.

The paper is organized as follows. First, Section 2 establishes the notation and background on inferencing in supervisory control. Then, in Section 3, we present the intuition behind our algorithm and review the finite structure (Ricker and Marchand, 2013) we use to determine if a system has an inferencing solution. Next, we present our algorithm in Section 4 and demonstrate it on an example that does not have an inferencing solution. Finally, we conclude the paper and discuss future extensions in Section 5.

2. NOTATION

Let Σ be a finite alphabet of *events*, and Σ^* be the set of words formed from elements of Σ, where ϵ is the empty word. 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 deterministic finite automaton is a 5-tuple $\mathcal{M}_L = (Q, \Sigma, T_L, q^l, F)$, where Q is a finite set of states, Σ is the alphabet, $q^l \in Q$ is the initial state, $F \subseteq Q$ is a set of final states, and $T_L \subseteq Q \times \Sigma^* \times Q$ is a deterministic transition relation closed under concatenation.

[1] 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.

Hence, $\forall q_i, q_j, q_k \in Q$ 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 $(q^\iota, w, q_k) \in T_L$ and $q_k \in F$. The regular language L recognized by $\mathcal{M}_L$ is the set of words accepted by $\mathcal{M}_L$. An automaton $\mathcal{M}_K = (Q_K, \Sigma, T_K, q^\iota_K, F_K)$ that recognizes regular language K is a sub-automaton of $\mathcal{M}_L$ and we write $\mathcal{M}_K \sqsubseteq \mathcal{M}_L$ if $Q_K \subseteq Q \wedge T_K \subseteq T_L \wedge F_K \subseteq F \wedge q^\iota_K = q^\iota$. Subsequently, $K \subseteq L$.

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 $u \in L$. Throughout this work, we will be considering only prefix-closed languages. Thus, all states are final, 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, without loss of generality, that $\mathcal{M}_K \sqsubseteq \mathcal{M}_L$. Each agent issues local control decisions 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 *observable* events Σ_o and *unobservable* events Σ_{uo}. We further define observable sub-alphabets $\Sigma_{o,i}$ (not necessarily disjoint) for each agent $i \in I$, where $\Sigma_{o,i} \subseteq \Sigma_o$, and $\Sigma_o := \cup_{i \in I} \Sigma_{o,i}$. 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 $\Sigma \setminus \Sigma_{o,i}$ from a word in Σ^*. Specifically, $\pi_i(\varepsilon) := \varepsilon$ and for $s \in \Sigma^*$, $\pi_i(s\sigma) := \begin{cases} \pi_i(s)\sigma & \text{when } \sigma \in \Sigma_{o,i} \\ \pi_i(s) & \text{otherwise} \end{cases}$. The inverse projection is defined as follows: $(\forall w \in \Sigma^*_{o,i})\ \pi_i^{-1}(w) := \{v \in \Sigma^* | \pi_i(v) = w\}$. We use $[\![w]\!]_i$ to denote $\pi_i^{-1}\pi_i(w) \cap L$, 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 $I_c(\sigma) = \{i \in I | \sigma \in \Sigma_{c,i}\}$.

2.1 Inference observability

The approach to inferencing, as proposed by Kumar and Takai (2007), requires the construction of two non-increasing sequences of prefix-closed languages with respect to (w.r.t.) the *subset relation* (Rabin and Scott, 1959), corresponding to words after which a controllable event must be disabled [2] 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 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 [3] as follows:

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

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

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. (Takai and Kumar (2008)). *Let $K \subseteq L$ be a specification language defined w.r.t. system language L, and $\Sigma_{o,i}, \Sigma_{c,i}$ be the sets of observable and controllable events defined for agents $i \in I$. If there exists an upper bound $N \in \mathbb{N}_0$, 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** w.r.t. $L, \Sigma_{o,i}, \Sigma_{c,i}$, for $i \in I$.*

Thus, when K is not N-inference observable, then $\exists \sigma \in \Sigma_c$ such that $D_{N+1}(\sigma) \neq \emptyset$ and $E_{N+1}(\sigma) \neq \emptyset$. For the remainder of this paper, we say K is *inference observable* w.r.t. L if $\exists N \in \mathbb{N}_0$ satisfying Def. 1.

Example 2. Consider the example in Fig. 1, noting that the solid line transitions indicate transitions in T_K, whereas the dashed line transition is in $T_L \setminus T_K$.

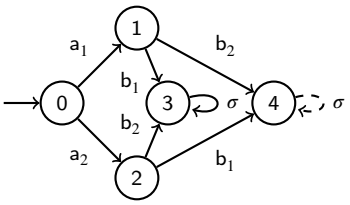

Fig. 1. Joint $\mathcal{M}_L$ (all transitions) and $\mathcal{M}_K$ (only solid-line transitions) for Ex. 2 where $n = 2, \Sigma_{o,1} = \{a_1, a_2\}, \Sigma_{o,2} = \{b_1, b_2\}$, and $I_c(\sigma) = \{1, 2\}$.

By Eqs (1)-(4), this system is not inference observable:

$$D_0(\sigma) = \{a_1b_2, a_2b_1\}, \quad E_0(\sigma) = \{a_1b_1, a_2b_2\};$$
$$D_1(\sigma) = D_0(\sigma), \quad E_1(\sigma) = E_0(\sigma). \quad \blacklozenge$$

Finally, we introduce notation that will be used in the sequel. Let R be a finite set $\{1, 2, \ldots r\}$ and $\rho(R)$ be a permutation of the set R of length $|R|$. We refer to the j^{th} element of $\rho(R)$ as $[\rho(R)]_j$, for $j \in I$. Given two permutations of R, namely $\rho_1(R)$ and $\rho_2(R)$, we denote their concatenation by $\rho_1(R)\rho_2(R)$.

3. A FINITE-STATE VERIFICATION STRUCTURE

We use the finite-state verification structure for decentralized architectures described in Ricker and Marchand (2013). Its construction uses a special product called *synchronized composition*, denoted by $\times_{\mathcal{S}}$. Specifically, we take $n+1$ copies of $\mathcal{M}^\varepsilon_L$, which is simply $\mathcal{M}_L$ augmented with self-loops of ε at each state:

$$\mathcal{U} = \overbrace{\mathcal{M}^\varepsilon_L \times_{\mathcal{S}} \mathcal{M}^\varepsilon_L \times_{\mathcal{S}} \ldots \times_{\mathcal{S}} \mathcal{M}^\varepsilon_L}^{(n+1)\ times} = (Q_{\mathcal{S}}, \Delta_{\mathcal{S}}, T_{\mathcal{S}}, q^\iota_{\mathcal{S}}),$$

where $Q_{\mathcal{S}} \subseteq Q^{n+1}$ is a set of state vectors of the form $\boldsymbol{q} = (q_0, q_1 \ldots, q_n)$; $\Delta_{\mathcal{S}} \subseteq \Sigma^{n+1}$ is a set of vector labels (Arnold, 1994) of the form $\boldsymbol{\sigma} = \langle \sigma_0, \sigma_1 \ldots, \sigma_i, \ldots, \sigma_n \rangle$; the transition relation $T_{\mathcal{S}} \subseteq Q_{\mathcal{S}} \times \Delta_{\mathcal{S}} \times Q_{\mathcal{S}}$ is defined as follows: $(\boldsymbol{q}, \boldsymbol{\sigma}, \hat{\boldsymbol{q}}) \in T_{\mathcal{S}}$, iff $\forall i \in I_0$, $(q_i, \sigma_i, \hat{q}_i) \in T^\varepsilon_L$, where T^ε_L is the transition

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

[3] In Kumar and Takai (2007), the more general mask is used instead of natural projection.

relation for $\mathcal{M}_L^{\varepsilon}$; and, the initial state $\boldsymbol{q}_{\mathcal{S}}^{I} = (q^I, q^I, \ldots, q^I)$. The j^{th} element of $\boldsymbol{q}$ and $\boldsymbol{\sigma}$ are referred to as $\boldsymbol{q}(j)$ or $\boldsymbol{q}_j$ and $\boldsymbol{\sigma}(j)$ or $\boldsymbol{\sigma}_j$, respectively, for $j \in \{0, 1, \ldots, n\}$. We can extend $T_{\mathcal{S}}$ to be defined over $\Delta_{\mathcal{S}}^*$. As a result, we have a word $\boldsymbol{v} = \boldsymbol{\sigma}_0\boldsymbol{\sigma}_1 \ldots \boldsymbol{\sigma}_{|\boldsymbol{v}|-1}$ in the language generated by $\mathcal{U}$ that consists of a vector of words in L: $\boldsymbol{v}(0) = \boldsymbol{\sigma}_0(0)\boldsymbol{\sigma}_1(0) \ldots \boldsymbol{\sigma}_{|\boldsymbol{v}|-1}(0), \boldsymbol{v}(1) = \boldsymbol{\sigma}_0(1)\boldsymbol{\sigma}_1(1) \ldots \boldsymbol{\sigma}_{|\boldsymbol{v}|-1}(1), \ldots, \boldsymbol{v}(n) = \boldsymbol{\sigma}_0(n)\boldsymbol{\sigma}_1(n) \ldots \boldsymbol{\sigma}_{|\boldsymbol{v}|-1}(n)$ where, by construction, $\pi_i(\boldsymbol{v}(0)) = \pi_i(\boldsymbol{v}(i))$, for all $i \in I$.

Labels in $\Delta_{\mathcal{S}}$ are partitioned into sets of observable labels Δ_o and unobservable labels Δ_{uo}. For $\sigma \in \Sigma_o$, the corresponding label in Δ_o has the form $\boldsymbol{\sigma} := \langle \pi_0(\sigma), \pi_1(\sigma), \ldots, \pi_n(\sigma) \rangle$. Unobservable labels are created for each $i \in I_0$. If $\sigma \in \Sigma \setminus \Sigma_{o,i}$, the unobservable label in $\Delta_{uo,i}$ is assembled as follows: $\boldsymbol{\sigma}(i) = \sigma$, while for all $j \in I_0 \setminus \{i\}$ we have $\boldsymbol{\sigma}(j) = \varepsilon$. Then $\Delta_{uo} = \bigcup_{i \in I_0} \Delta_{uo,i} = \Delta_{\mathcal{S}} \setminus \Delta_o$. We also define the empty vector label $\boldsymbol{\varepsilon} = \langle \varepsilon, \ldots, \varepsilon \rangle$, where $\forall i \in I_0$, $\boldsymbol{\varepsilon}(i) = \varepsilon$. Finally, although we omit a formal definition here, we can partition $\Delta_{\mathcal{S}}$ into controllable and uncontrollable labels, denoted by Δ_c and Δ_{uc}, based on membership in Σ_c and Σ_{uc}.

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 3. *(a) The set* $\mathsf{D}_\sigma(\mathcal{U}) \subseteq Q_{\mathcal{S}}$ *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.*

$$\mathsf{D}_\sigma(\mathcal{U}) = \{q \in Q_{\mathcal{S}} | (\exists \hat{q} \in Q_{\mathcal{S}}) \text{ s.t. } (q_0, \sigma, \hat{q}_0) \in T_L \setminus T_K \wedge (\forall i \in I_c(\sigma))(q_i, \sigma, \hat{q}_i) \in T_L\}.$$

(b) The set $\mathsf{E}_\sigma(\mathcal{U}) \subseteq Q_{\mathcal{S}}$ *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.*

$$\mathsf{E}_\sigma(\mathcal{U}) = \{q \in Q_{\mathcal{S}} | (\exists \hat{q} \in Q_{\mathcal{S}}) \text{ s.t. } (q_0, \sigma, \hat{q}_0) \in T_K \wedge (\forall i \in I_c(\sigma))(q_i, \sigma, \hat{q}_i) \in T_L\}.$$

Note that for $(\boldsymbol{q}_{\mathcal{S}}^{I}, \boldsymbol{v}, \boldsymbol{q}) \in T_{\mathcal{S}}$, where $\boldsymbol{q} \in \mathsf{D}_\sigma(\mathcal{U})$ and $\sigma \in \Sigma_c$, $\boldsymbol{v}(0)$ corresponds to a word $w \in L$ such that $w \in D_0(\sigma)$, according to Eq. (1). As a result, for each word in Eq. (1), we identify state-based equivalents in Def. 3*(a)*. Similarly, we have state-based equivalents for the words in Eq. (2).

Instead of inferencing over words or families of languages, we want to inference over the finite set of control configurations. A state $\mathsf{d} \in \mathsf{D}_\sigma(\mathcal{U})$ requires an overall control decision of *disable*, and we want to be able to discuss the states in $\mathsf{E}_\sigma(\mathcal{U})$ that each agent $i \in I_c(\sigma)$ 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 $\mathcal{U}$.

Recall that 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 $\mathcal{O}_{\Delta_i}$, and use this structure to formulate state-based control decisions. Initially, we first replace all the transitions in $\mathcal{U}$ that have a label $\boldsymbol{\sigma} \in \Delta_{uo,i}$ with $\boldsymbol{\varepsilon}$. Subsequently, we apply the subset relation. More formally, we define the $\varepsilon-\mathsf{reach}$ of a state $\boldsymbol{q}_{\mathcal{S}} \in Q_{\mathcal{S}}$, w.r.t. alphabet $\Delta_i \subseteq \Delta_{\mathcal{S}}$ as $\varepsilon-\mathsf{reach}(\boldsymbol{q}_{\mathcal{S}}, \Delta_i) = \{\hat{\boldsymbol{q}}_{\mathcal{S}} \in Q_{\mathcal{S}} | (\exists u \in (\Delta_{\mathcal{S}} \setminus \Delta_i)^*)(\boldsymbol{q}_{\mathcal{S}}, u, \hat{\boldsymbol{q}}_{\mathcal{S}}) \in T_{\mathcal{S}}\}$. Then we have

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

where $X_i \subseteq 2^{Q_{\mathcal{S}}}$; $x_{i,0} = \varepsilon-\mathsf{reach}(\boldsymbol{q}_{\mathcal{S}}^{I}, \Delta_i)$; and $(x_i, \boldsymbol{\sigma}, x_i') \in T_{\Delta_i}$, where $x_i' = \bigcup_{\boldsymbol{q}_{\mathcal{S}} \in x} \varepsilon-\mathsf{reach}(\hat{\boldsymbol{q}}_{\mathcal{S}}, \Delta_i)$ s.t. $(\boldsymbol{q}_{\mathcal{S}}, \boldsymbol{\sigma}, \hat{\boldsymbol{q}}_{\mathcal{S}}) \in T_{\mathcal{S}}$.

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, and 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. 3. To determine the complete set of states for $\mathsf{D}_\sigma(\mathcal{U})$ and $\mathsf{E}_\sigma(\mathcal{U})$, we compute $\mathcal{O}_{\Delta_0}$, where $\Delta_0 = \Delta_o$. If a state $\boldsymbol{q}_{\mathcal{S}}$ of $\mathcal{U}$ appears in more than one state of $\mathcal{O}_{\Delta_0}$, we create a copy (or copies) of $\boldsymbol{q}_{\mathcal{S}}$ with a label $\boldsymbol{q}_{\mathcal{S}_j}$, 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 appropriately to either $\mathsf{E}_\sigma(\mathcal{U})$ or $\mathsf{D}_\sigma(\mathcal{U})$. The procedure is demonstrated below in the continuation of our example.

Ex. 2 cont. The $\mathcal{U}$ structure for Ex. 2 is shown in Fig. 2, and its $\mathcal{O}_{\Delta_0}$ is shown in Fig. 3. The control configurations (highlighted in red—$\mathsf{D}_\sigma(\mathcal{U})$—and green—$\mathsf{E}_\sigma(\mathcal{U})$) are listed as follows:

$$\mathsf{D}_\sigma(\mathcal{U}) = \{(4,3,3), (4,3,4), (4,4,3), (4,4,4)\}; \quad (5)$$

$$\mathsf{E}_\sigma(\mathcal{U}) = \{(3,3,3), (3,3,4), (3,4,3), (3,4,4)\}. \quad (6)$$

In Fig. 1, 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. 2, 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 $(3,3,4)$, 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}. Each state in $\mathsf{D}_\sigma(\mathcal{U})$ (Eq. (5)) and $\mathsf{E}_\sigma(\mathcal{U})$ (Eq. (6)) appears in two different states of $\mathcal{O}_{\Delta_0}$: x_{03} and x_{05} for $\mathsf{E}_\sigma(\mathcal{U})$, and x_{04} and x_{06} for $\mathsf{D}_\sigma(\mathcal{U})$. Here, we label the copies with a subscript of 'α' to distinguish them from the originals. The states of $\mathcal{O}_{\Delta_0}$ after relabeling the copies are presented in the leftmost column of Table 1.

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)$_\alpha$, (3,3,4)$_\alpha$,(3,4,3)$_\alpha$,(3,4,4)$_\alpha$, (4,2,3),(4,2,4),(4,3,3)$_\alpha$, (4,3,4)$_\alpha$,(4,4,3)$_\alpha$,(4,4,4)$_\alpha$}	x_{21}	{(1,3,3),(1,3,4),(3,3,3), (3,3,4),(3,4,3),(3,4,4), (4,2,3),(4,2,4),(4,3,3)$_\alpha$, (4,3,4)$_\alpha$,(4,4,3)$_\alpha$,(4,4,4)$_\alpha$}
x_{03}	{(3,1,3),(3,1,4),(3,3,3), (3,3,4),(3,4,3),(3,4,4)}				
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)$_\alpha$, (3,3,4)$_\alpha$,(3,4,3)$_\alpha$,(3,4,4)$_\alpha$}			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)$_\alpha$, (3,3,4)$_\alpha$,(3,4,3)$_\alpha$,(3,4,4)$_\alpha$}
x_{06}	{(4,2,3),(4,2,4),(4,3,3)$_\alpha$, (4,3,4)$_\alpha$,(4,4,3)$_\alpha$,(4,4,4)$_\alpha$}				

Table 1. State sets X_0 (left), X_1 (center) and X_2 (right) for Ex. 2. The control configurations in $\mathsf{E}_\sigma(\mathcal{U})$ are noted in green, while $\mathsf{D}_\sigma(\mathcal{U})$ are red.

After relabeling the duplicate states:

$$\mathsf{D}_\sigma(\mathcal{U}) = \{(4,3,3), (4,3,4), (4,4,3), (4,4,4), (4,3,3)_\alpha, (4,3,4)_\alpha, (4,4,3)_\alpha, (4,4,4)_\alpha\};$$

$$\mathsf{E}_\sigma(\mathcal{U}) = \{(3,3,3), (3,3,4), (3,4,3), (3,4,4), (3,3,3)_\alpha, (3,3,4)_\alpha, (3,4,3)_\alpha, (3,4,4)_\alpha\}. \quad \blacklozenge$$

Using $\mathcal{O}_{\Delta_i}$, we can determine the states of $\mathcal{U}$ that are indistinguishable for agent i. By construction, such states correspond

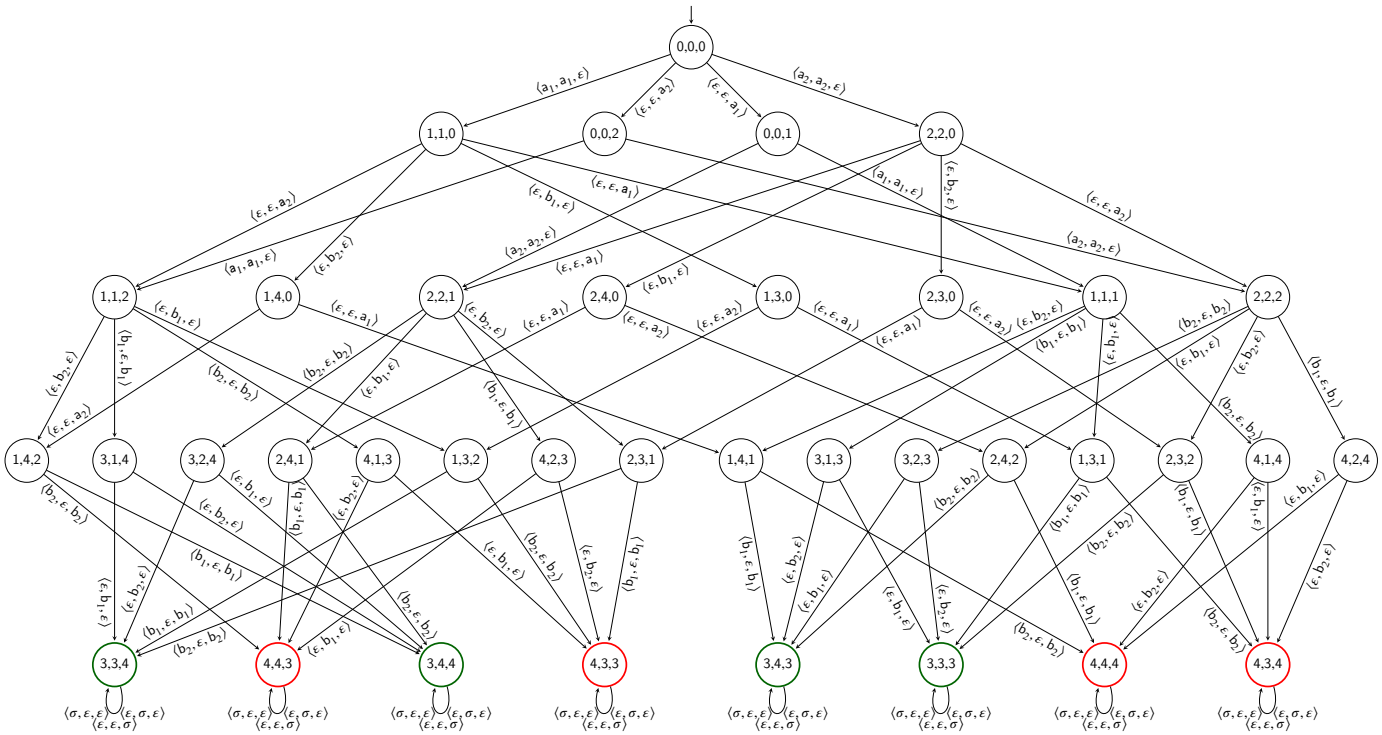

Fig. 2. The $\mathcal{U}$ structure for Ex. 2. Self-loops of ϵ at each state and labels involving σ at states that are not control configurations omitted for readability.

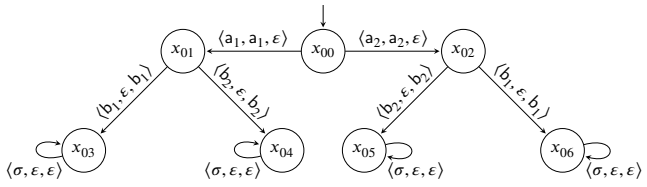

Fig. 3. $\mathcal{O}_{\Delta_0}$ for Ex. 2.

to words that are equal up to the projection over the observable alphabet of i.

Definition 4. *Given* $\boldsymbol{q_S}, \hat{\boldsymbol{q}}_{\boldsymbol{S}} \in Q_{\mathcal{S}}$, $\boldsymbol{q_S}$ *is* ***indistinguishable*** *from* $\hat{\boldsymbol{q}}_{\boldsymbol{S}}$ *for agent* $i \in I$, *denoted* $\boldsymbol{q_S} \sim_i \hat{\boldsymbol{q}}_{\boldsymbol{S}}$, *iff* $\exists x \in X_i$ *s.t.* $\boldsymbol{q_S}, \hat{\boldsymbol{q}}_{\boldsymbol{S}} \in x$.

We overload our notation and let $[\![\boldsymbol{q_S}]\!]_i$ represent the states in $Q_{\mathcal{S}}$ that agent i cannot distinguish from $\boldsymbol{q_S}$ under $\sim_i$.

Ex. 2 further cont.
The alphabets for each agent are: $\Delta_1 = \{\langle \mathsf{a}_1, \mathsf{a}_1, \varepsilon\rangle, \langle \mathsf{a}_2, \mathsf{a}_2, \varepsilon\rangle\}$ and $\Delta_2 = \{\langle \mathsf{b}_1, \varepsilon, \mathsf{b}_1\rangle, \langle \mathsf{b}_2, \varepsilon, \mathsf{b}_2\rangle\}$. The states of $\mathcal{O}_{\Delta_1}$ and $\mathcal{O}_{\Delta_2}$ are presented in the two rightmost columns of Table 1. Finally, by Table 1, we apply Def. 4 to control configurations w.r.t. σ, noting that $(3,3,3) \sim_1 (4,3,3)$ as both these control configurations w.r.t. σ are in $x_{11} \in X_1$, and $(3,3,3) \sim_2 (4,3,3)_\alpha$, as these are both in $x_{21} \in X_2$. ♦

Our focus from here on in is the indistinguishability between states in $\mathsf{D}_\sigma(\mathcal{U})$ and states in $\mathsf{E}_\sigma(\mathcal{U})$ for agents in $I_c(\sigma)$.

4. AN ALGORITHM FOR VERIFYING INFERENCE OBSERVABILITY

The inspiration for our algorithm came from using a *bipartite graph* to examine the relationships between sequences in Eqs. (3) and (4) when K is not inference observable w.r.t. L. A bipartite graph G consists of two disjoint sets of vertices V_1, V_2, and each edge of G connects a vertex $v_1 \in V_1$ and a vertex $v_2 \in V_2$. Initially, we constructed vertices based on the sequences in $D_{k+1}(\sigma)$ and $E_{k+1}(\sigma)$, for $\sigma \in \Sigma_c$ and $k \in \mathbb{N}_0$, and drew an edge with a label $i \in I_c(\sigma)$ between two vertices corresponding to $w_d \in D_{k+1}(\sigma)$ and $w_e \in E_{k+1}(\sigma)$ if $\pi_i(w_d) = \pi_i(w_e)$. The members of $D_1(\sigma)$ and $E_1(\sigma)$ from Ex. 2 are presented as a bipartite graph in Fig. 4, where orange lines and purple lines correspond to edge labels $i = 1$ and $i = 2$, respectively. There are cycles present in this graph, e.g., a vertex sequence of ($\mathsf{a}_1\mathsf{b}_2$, $\mathsf{a}_1\mathsf{b}_1$, $\mathsf{a}_2\mathsf{b}_1$, $\mathsf{a}_2\mathsf{b}_2$, $\mathsf{a}_1\mathsf{b}_2$) and an edge trail of $(1,2,1,2)$. But ultimately, we want a state-based version of an *inferencing cycle*.

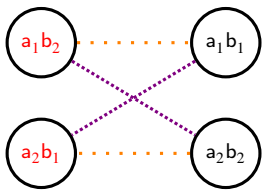

Fig. 4. Bipartite graph based on $D_1(\sigma)$ and $E_1(\sigma)$ from Ex. 2.

Definition 5. *Given* $\sigma \in \Sigma_c$, $\mathsf{D} = \{\mathsf{d}_0, \ldots, \mathsf{d}_{|\mathsf{D}|-1}\} \subseteq \mathsf{D}_\sigma(\mathcal{U})$, *and* $\mathsf{E} = \{\mathsf{e}_0, \ldots, \mathsf{e}_{|\mathsf{E}|-1}\} \subseteq \mathsf{E}_\sigma(\mathcal{U})$. *If* $\exists m, r \in \mathbb{N}_0$ *s.t.* $i_0, i_1, \ldots i_{m-1} = \rho_0(I_c(\sigma))\ \rho_1(I_c(\sigma))\ \ldots\ \rho_{r-1}(I_c(\sigma))$ *where* $[\rho_{j-1}(I_c(\sigma))]|_{I_c(\sigma)|-1} \neq [\rho_j(I_c(\sigma))]_0$ *(for* $j \in \{1, \ldots, r-1\}$*) and*

$$\mathsf{d}_0 \sim_{i_0} \mathsf{e}_0 \sim_{i_1} \ldots \mathsf{e}_{m-1} \sim_{i_{m-1}} \mathsf{d}_m,$$

then we have an ***EBD state-based inferencing cycle*** *if* $\exists t < m-1$ *where* $d_m = d_t$.

We can similarly define a DBD state-based inferencing cycle when we instead begin with some $\mathsf{e}_0 \in \mathsf{E}_\sigma(\mathcal{U})$.

Our algorithm runs on a bipartite graph G_σ whose disjoint sets of vertices correspond to the sets of control configurations $\mathsf{D}_\sigma(\mathcal{U})$ and $\mathsf{E}_\sigma(\mathcal{U})$, denoted here by $\mathsf{V}_{\mathsf{D}_\sigma(\mathcal{U})}$ and $\mathsf{V}_{\mathsf{E}_\sigma(\mathcal{U})}$, respectively. There is an edge labelled i between a vertex $v_{\mathbf{d}} \in \mathsf{V}_{\mathsf{D}_\sigma(\mathcal{U})}$ and a vertex $v_{\mathbf{e}} \in \mathsf{V}_{\mathsf{E}_\sigma(\mathcal{U})}$ if $\mathbf{d} \sim_i \mathbf{e}$ in $\mathcal{O}_{\Delta_i}$, for $i \in I_c(\sigma)$.

The bipartite graph using the control configurations of $\mathcal{U}$ for Ex. 2 is shown in Fig. 5, using the same color scheme as in Fig. 4. Solid lines indicate a sample inferencing cycle: $(3,3,3) \sim_1 (4,3,4) \sim_2 (3,3,3)_\alpha \sim_1 (4,3,4)_\alpha \sim_2 (3,3,3)$.

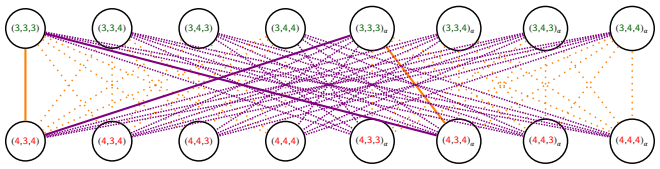

Fig. 5. G_σ for Ex. 2.

We now present our algorithm for verifying whether a system is inference observable based on whether it contains inferencing cycles. The basic outline of the verification strategy is as follows:

(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 $\mathsf{E}_\sigma(\mathcal{U})$ and $\mathsf{D}_\sigma(\mathcal{U})$, for all $\sigma \in \Sigma_c$, relabel and update these states in both $\mathcal{U}$ and $\mathcal{O}_{\Delta_0}$ (see Section 3).
(4) Construct $\mathcal{O}_{\Delta_i}$ and $\sim_i$ (for $i \in I$).
(5) Run Algorithm 1 consisting of the following milestones:
   (a) For some $\sigma \in \Sigma_c$, build G_σ using the (relabeled) states of $\mathsf{E}_\sigma(\mathcal{U})$ and $\mathsf{D}_\sigma(\mathcal{U})$ as the two vertex sets. An edge $(v_{\mathsf{e}}, v_{\mathsf{d}})$ with label $i \in I_c(\sigma)$ exists in G_σ between a vertex $v_{\mathsf{e}} \in \mathsf{V}_{\mathsf{E}_\sigma(\mathcal{U})}$ and a vertex $v_{\mathsf{d}} \in \mathsf{V}_{\mathsf{D}_\sigma(\mathcal{U})}$ if agent i cannot distinguish between the states in $\mathcal{U}$ represented by the two vertices, i.e., $\mathbf{e} \sim_i \mathbf{d}$.
   (b) Run a cycle detection on G_σ according to Def. 5. Since we work with a finite structure, there is no issue detecting arbitrarily long cycles. 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, compute the value for N and exit successfully.

4.1 The algorithm

Algorithm 1 begins by initializing N for $\sigma \in \Sigma_c$ (line 4) and the set of the adjacent ("neighboring") vertices of vertex v, denoted by $\mathsf{A}_{i,v}$ (line 8). At that point (lines 10 - 15), the adjacent 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 20 - 24) determines if any vertices can be distinguished at the outset. Such vertices are stored in $\mathsf{V}_{\mathsf{dist}}$ and N for this $\sigma \in \Sigma_c$ is updated to 0 since v can be distinguished immediately in inferencing round 0. Note that at this stage, if we can distinguish none of the vertices, then $\mathsf{V}_{\mathsf{dist}} = \emptyset$ at line 25, so $prev_{\mathsf{dist}}$ will likewise be empty, and the algorithm will return `False` at line 46, as we have detected an inferencing cycle involving *all* the vertices of G_σ.

Lines 27 - 45 are where the primary inferencing cycle detection occurs. This loop also represents the manipulation of the bipartite graph itself. If agent i can fully distinguish a vertex v, its set of adjacent vertices $\mathsf{A}_{i,v}$ 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, i.e., $\mathsf{A}_{i,v'}$, empty, then v' is added to the list of distinguished vertices for the next iteration and $N(\sigma)$ is updated with the value of the current inferencing round. These lines repeat until there are no more vertices to distinguish. If the condition at line 46 is not met, then $N(\sigma)$ is the smallest number of rounds required to distinguish all the vertices in G_σ. The algorithm repeats from line 2 with the next controllable event.

When the condition at line 46 is met, the algorithm returns `False`, signalling the presence of an inferencing cycle since the number of distinguished vertices is not equal to the number of control configurations w.r.t. $\sigma \in \Sigma_c$. Recall that a vertex v is deemed distinguished by agent $i \in I_c(\sigma)$ when $\mathsf{A}_{i,v} = \emptyset$. Thus, when the loop at lines 27 - 45 completes in this scenario, it is because none of the agents in $I_c(\sigma)$ can distinguish the remaining vertices $v \in (\mathsf{V}_{\mathsf{D}_\sigma(\mathcal{U})} \cup \mathsf{V}_{\mathsf{E}_\sigma(\mathcal{U})}) \setminus \mathsf{V}_{\mathsf{dist}}$, indicating the presence of an inferencing cycle.

The algorithm returns `True` at line 49 if all control configurations across Σ_c are distinguished. The final value of N is determined by taking the maximum value across the bounds computed for each $\sigma \in \Sigma_c$ at line 48.

Proposition 6. Algorithm 1 correctly verifies a violation of inference observability in the presence of state-based inferencing cycles in $\mathcal{U}$.

Proof sketch: Given $\mathcal{U}$ and, for simplicity, assume that $\Sigma_c = \{\sigma\}$. Further, let the control configurations w.r.t. σ be $\mathsf{D}_\sigma(\mathcal{U}) = \{\mathbf{d}\}$ and $\mathsf{E}_\sigma(\mathcal{U}) = \{\mathbf{e}\}$, and assume $\mathcal{U}$ contains an EBD state-inferencing cycle starting at $\mathbf{d}$. That is, $\exists m, r \in \mathbb{N}_0$ such that $\rho_0(I_c(\sigma))\rho_1(I_c(\sigma)) \dots \rho_{r-1}(I_c(\sigma)) = i_0, i_1, \dots i_{m-1}$ such that $\mathbf{d} \sim_{i_0} \mathbf{e} \sim_{i_1} \dots \mathbf{d} \sim_{i_j} \mathbf{e} \sim_{i_{j+1}} \dots \sim_{i_{m-1}} \mathbf{d}$. In this case, $r = 1$ and $m = |I_c(\sigma)|$ since one permutation of $I_c(\sigma)$ results in an inferencing cycle for $\mathbf{d}$ and $\mathbf{e}$, regardless of the permutation. Additionally, $\mathcal{U}$ contains a DBD state-inferencing cycle starting at e. Since there is no $i \in I_c(\sigma)$ that can distinguish $\mathbf{d}$ from e, at line 25 of the algorithm, $\mathsf{A}_{i,\mathbf{d}} = \{\mathbf{e}\}$ and $\mathsf{A}_{i,\mathbf{e}} = \{\mathbf{d}\}$, for all $i \in I_c(\sigma)$. Since neither of these sets will allow the algorithm ever to reach line 23, it will never be the case that either of these states will be added to $prev_{\mathsf{dist}}$ in line 25. As a result, when reaching line 46, the algorithm will return `False` since neither $\mathbf{d}$ nor $\mathbf{e}$ was distinguished. ■

Ex. 2 concluded. We use Algorithm 1 to confirm that the example contains inference cycles and is thus not inference observable. After the initialization loop in lines 6 - 8, the edges of the bipartite graph are constructed in lines 10 - 15. In particular, as can be deduced from Table 1, when $v = (4,3,3)$, we have $\mathsf{A}_{1,v} = \{(3,3,3), (3,3,4), (3,4,3), (3,4,4)\}$ and $\mathsf{A}_{2,v} = \{(3,3,3)_\alpha, (3,3,4)_\alpha, (3,4,3)_\alpha, (3,4,4)_\alpha\}$. All the other vertices have identically sized (non-empty) sets of adjacent vertices. As a result, after $\mathsf{V}_{\mathsf{dist}}$ is initialized to $\emptyset$ in line 17, it never changes since the conditional at line 22 is `False` for every single vertex in $\mathsf{V}_{\mathsf{D}_\sigma(\mathcal{U})} \cup \mathsf{V}_{\mathsf{E}_\sigma(\mathcal{U})}$. Consequently, in line 25, $prev_{\mathsf{dist}} = \emptyset$, and the conditional at line 27 is immediately `False`, thus terminating the program at line 46 and returning `False`. ♦

Overall, the computational complexity for Algorithm 1 is $O(\sum_{\sigma \in \Sigma_c}(|I_c(\sigma)| \cdot |\mathsf{E}_\sigma(\mathcal{U})| \cdot |\mathsf{D}_\sigma(\mathcal{U})|))$. We have implemented

Algorithm 1 Verifying state-based inference-observability using a bipartite graph

procedure IsInfObs($\Sigma_c, \{I_c(\sigma)\}_{\sigma\in\Sigma_c}$, $\{(\mathsf{V}_{\mathsf{D}_\sigma(\mathcal{U})}, \mathsf{V}_{\mathsf{E}_\sigma(\mathcal{U})})\}_{\sigma\in\Sigma_c}, \{\sim_i\}_{i\in I_c(\sigma)}$)
  for all $\sigma \in \Sigma_c$ **do**
    ▷ *Initialize the value of N for each* $\sigma \in \Sigma_c$ ◁
    $N(\sigma) \leftarrow -1$
    ▷ *For each agent i, initialize the set of adjacent vertices for all* $v \in \mathsf{V}_{\mathsf{D}_\sigma(\mathcal{U})} \cup \mathsf{V}_{\mathsf{E}_\sigma(\mathcal{U})}$*:* $\mathsf{A}_{i,v}$ *The loop below runs in* $O((|\mathsf{E}_\sigma(\mathcal{U})|+|\mathsf{D}_\sigma(\mathcal{U})|)\cdot|I_c(\sigma)|)$ *time.* ◁
    for all $i \in I_c(\sigma)$ **do**
      for all $v \in \mathsf{V}_{\mathsf{D}_\sigma(\mathcal{U})} \cup \mathsf{V}_{\mathsf{E}_\sigma(\mathcal{U})}$ **do**
        $\mathsf{A}_{i,v} \leftarrow \emptyset$
    ▷ *Building edges of bipartite graph. This can be achieved in* $O(|\mathsf{E}_\sigma(\mathcal{U})|\cdot|\mathsf{D}_\sigma(\mathcal{U})|\cdot|I_c(\sigma)|)$ *time.* ◁
    for all $\mathsf{d} \in \mathsf{V}_{\mathsf{D}_\sigma(\mathcal{U})}$ **do**
      for all $\mathsf{e} \in \mathsf{V}_{\mathsf{E}_\sigma(\mathcal{U})}$ **do**
        for all $i \in I_c(\sigma)$ **do**
          if $\mathsf{d} \sim_i \mathsf{e}$ **then**
            $\mathsf{A}_{i,\mathsf{d}} \leftarrow \mathsf{A}_{i,\mathsf{d}} \cup \{\mathsf{e}\}$
            $\mathsf{A}_{i,\mathsf{e}} \leftarrow \mathsf{A}_{i,\mathsf{e}} \cup \{\mathsf{d}\}$
    ▷ $\mathsf{V}_{\mathsf{dist}}$ *is the set of vertices in* $\mathsf{V}_{\mathsf{D}_\sigma(\mathcal{U})}\cup\mathsf{V}_{\mathsf{E}_\sigma(\mathcal{U})}$ *that can be distinguished by at least one* $i \in I_c(\sigma)$ ◁
    $\mathsf{V}_{\mathsf{dist}} \leftarrow \emptyset$
    $infLevel \leftarrow 0$
    ▷ *Identify vertices that are immediately distinguishable before checking for cycles. This can be done in* $O((|\mathsf{E}_\sigma(\mathcal{U})| + |\mathsf{D}_\sigma(\mathcal{U})|)\cdot|I_c(\sigma)|)$ *time.* ◁
    for all $v \in \mathsf{V}_{\mathsf{D}_\sigma(\mathcal{U})} \cup \mathsf{V}_{\mathsf{E}_\sigma(\mathcal{U})}$ **do**
      for all $i \in I_c(\sigma)$ **do**
        if $\mathsf{A}_{i,v} = \emptyset \wedge v \notin \mathsf{V}_{\mathsf{dist}}$ **then**
          $\mathsf{V}_{\mathsf{dist}} \leftarrow \mathsf{V}_{\mathsf{dist}} \cup \{v\}$
          $N(\sigma) \leftarrow infLevel$
    $prev_{\mathsf{dist}} \leftarrow \mathsf{V}_{\mathsf{dist}}$
    ▷ *Loop to detect cycles in the bipartite graph* G_σ*, If* G_σ *has* t_σ *edges, then the complexity is* $O(\sum_{\sigma\in\Sigma_c}(|\mathsf{E}_\sigma(\mathcal{U})| + |\mathsf{D}_\sigma(\mathcal{U})| + \mathsf{t}_\sigma))$ *because each vertex and edge can be removed at most once.* ◁
    while $prev_{\mathsf{dist}} \neq \emptyset$ **do**
      $infLevel \leftarrow infLevel + 1$
      $curr_{\mathsf{dist}} \leftarrow \emptyset$
      for all $v \in prev_{\mathsf{dist}}$ **do**
        for all $i \in I_c(\sigma)$ **do**
          if $\mathsf{A}_{i,v} = \emptyset \wedge v \notin \mathsf{V}_{\mathsf{dist}}$ **then**
            $curr_{\mathsf{dist}} \leftarrow curr_{\mathsf{dist}} \cup \{v\}$
            if $infLevel > N(\sigma)$ ***then***
              $N(\sigma) \leftarrow infLevel$
          else
            for all $v' \in \mathsf{A}_{i,v}$ **do**
              ▷ *Removing* v *from* $\mathsf{A}_{i,v'}$ *effectively removes the edge with label* i *between* v *and* v' ◁
              $\mathsf{A}_{i,v'} \leftarrow \mathsf{A}_{i,v'} \setminus \{v\}$
              if $\mathsf{A}_{i,v'} = \emptyset \wedge v' \notin \mathsf{V}_{\mathsf{dist}}$ **then**
                $curr_{\mathsf{dist}} \leftarrow curr_{\mathsf{dist}} \cup \{v'\}$
                if $infLevel > N(\sigma)$ ***then***
                  $N(\sigma) \leftarrow infLevel$
      $\mathsf{V}_{\mathsf{dist}} \leftarrow \mathsf{V}_{\mathsf{dist}} \cup curr_{\mathsf{dist}}$
      $prev_{\mathsf{dist}} \leftarrow curr_{\mathsf{dist}}$
    if $\mathsf{V}_{\mathsf{dist}} \neq (\mathsf{V}_{\mathsf{D}_\sigma(\mathcal{U})} \cup \mathsf{V}_{\mathsf{E}_\sigma(\mathcal{U})})$ **then return** `False`
    ▷ *Find the overall upper bound across* Σ_c ◁
    $N \leftarrow \max_{\sigma\in\Sigma_c} N(\sigma)$
  return `True`

Algorithm 1 and have examined various examples, including examples with more than two inferencing agents. Our implementation, along with a variety of solved examples, can be found in Yoon and Stairs (2015 - 2024).

5. 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. We proposed an algorithm for verifying this interpretation of inferencing observability. Most notably, when an inferencing solution exists, our algorithm computes the upper bound N. Note that Takai and Kumar (2017, 2018) verify N but do not compute it, so our algorithmic results are incomparable. Future work includes extending Algorithm 1 to compute the local and global control decisions when the system is inference observable. Finally, our new understanding of inferencing cycles opens up new avenues we began to explore in decentralized architectures beyond inferencing (Ricker et al., 2017).

REFERENCES

Arnold, A. (1994). *Finite Transition Systems*. Masson.

Kumar, R. and Takai, S. (2007). Inference-based ambiguity management in decentralized decision-making: Decentralized control of discrete event systems. *IEEE Trans. Autom. Control*, 52(10), 1783–1794.

Rabin, M. and Scott, D. (1959). Finite automata and their decision problems. *IBM J. Res. Dev.*, 3(2), 114–125.

Ricker, S., Lidbetter, T., and Marchand, H. (2017). Inferencing and beyond: further adventures with parity-based architectures for decentralized discrete-event systems. *IFAC-PapersOnLine*, 50(1), 13447–13452.

Ricker, S. and Marchand, H. (2013). A parity-based architecture for decentralized discrete-event control. In *Am. Control Conf.*, 5678–5684.

Rudie, K. and Wonham, W.M. (1992). Think globally, act locally: Decentralized supervisory control. *IEEE Trans. Autom. Control*, 37(11), 1692–1708.

Takai, S. and Kumar, R. (2008). Synthesis of inference-based decentralized control for discrete event systems. *IEEE Trans. Autom. Control*, 53(2), 522–534.

Takai, S. and Kumar, R. (2017). A generalized framework for inference-based diagnosis of discrete event systems capturing both disjunctive and conjunctive decision-making. *IEEE Trans. Autom. Control*, 62(6), 2778–2793.

Takai, S. and Kumar, R. (2018). Implementation of inference-based diagnosis: computing delay bound and ambiguity levels. *Discrete Event Dyn. Syst.*, 28(2), 315–348.

Yoo, T.S. and Lafortune, S. (2004). Decentralized supervisory control with conditional decisions: supervisor existence. *IEEE Trans. Autom. Control*, 49(11), 1886–1904.

Yoon, S.H. and Stairs, M. (2015 - 2024). JDec. URL `https://github.com/Summer2023SHY/JDec`.