Mutual Opacity between Multiple Adversaries

C K Sharpe, S L Ricker, Hervé Marchand

Mutual Opacity between Multiple Adversaries

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

* *Mount Allison University, Sackville, NB CANADA (e-mail: cksharpe,lricker@mta.ca)*
** *Univ Rennes, Inria, CNRS, IRISA, Rennes, France (e-mail: herve.marchand@inria.fr)*

Abstract: We investigate opacity, an information-flow privacy property, in a setting where there are two competing agents or adversaries whose objective is to hide their secrets and expose the secrets of the other agent. Each agent has only partial information about the state of the system. The agents can achieve their objective by enabling or disabling events from their set of controllable events. We examine two different scenarios. In the first problem, the agents are passive with no control capabilities, and we seek a global controller to enforce their mutual opacity. In the second problem, the formerly passive agents are autonomous and have control capabilities. We seek the plausibility of two controllers, one for each agent, to see if we can synthesize a winning control strategy so that one adversary can always discover the secrets of the other without revealing its own.

1. INTRODUCTION

Security is one of the most critical and challenging aspects in designing services deployed on large open networks such as the Internet, mobile phones, and e-voting systems. Such services are regularly subject to malicious attacks. Thus, developing methods to ensure their security is crucial. Over the past ten years, there has been significant research devoted to developing formal methods for designing secure systems, leading to a growing interest in the formal verification of security properties.

There are well-established notions of security, but we are interested in those that focus on characterizing the *information flow* from the system to an intruder (Riccardo Focardi (1994); Goguen and Meseguer (1982); McCullough (1988); Wittbold and Johnson (1990)). The notion of *opacity*, introduced in Mazare (2004), formalizes the absence of information flow in a system. More precisely, opacity refers to the impossibility that an intruder can infer the truth about some confidential activity of the system. For example, such activity could be the occurrence of some particular sequence of actions in the system or the fact that the system is in a specific configuration.

Opacity was initially modeled using trace-based systems (Bryans et al., 2008), but we are interested in examining opacity through the lens of supervisory control theory (Ramadge and Wonham, 1987). We model system behavior and, subsequently, its security properties using finite state automata. The work of Dubreil et al. (2008, 2010) introduced the use of supervisory control for enforcing opacity with a single malicious intruder pursuing secrets. Since then, this scenario with a single attacker has been extensively examined; refer to Jacob et al. (2016) for a general survey of the use of supervisory control theory and security properties like opacity.

We are interested in examining strategies where multiple agents want to acquire secret information without compromising their secrets. The first work to examine opacity with multiple agents appeared in Badouel et al. (2007) and, more recently, in Helouet et al. (2018). In particular, we examine a variation on the concurrent secrets problem (Badouel et al. (2007)) where the goal for each agent is to uncover the secret of the other agent.

The paper is organized as follows. Section 2 establishes the notation and background on using supervisory control to enforce opacity. In Section 3 we present the two multi-agent problems that we are interested in examining. We first consider a setting where a global control strategy ensures secrets are not directly or inadvertently leaked among members of a group of agents. We address this problem in Section 4. Secondly, we propose a setting where local control is the focus since the goal of each agent is to force the disclosure of the secrets of other agents while preventing its secrets from leaking. This problem is addressed in Section 5. We discuss extensions of this work in Section 6.

2. PRELIMINARIES

In this section, we provide a brief summary of the notation used in the paper, along with a relevant background for the opacity problem.

2.1 Languages and Automata

Let Σ be a finite alphabet (set of events) while Σ^* denotes the set of all finite-length sequences or *words* of events of Σ, including the empty word ε. The length of a word $w \in \Sigma^*$ is denoted by $|w|$, where $|\varepsilon| = 0$. A language $\mathcal{L}$ is a subset of words in Σ^*, i.e., $\mathcal{L} \subseteq \Sigma^*$. Given words $t, u \in \Sigma^*$, the word $t.u$ denotes the concatenation of t and u, i.e., the sequence of events of t followed by the sequence of events of u. Given two words $w, w' \in \Sigma^*$, we say that w' is a *prefix* of w whenever there exists $w'' \in \Sigma^*$ such that

$w = w'.w''$. The set of prefixes of a language $\mathcal{L} \subseteq \Sigma^*$ is given by $pref(\mathcal{L}) = \{w \in \Sigma^* : \exists w' \in \Sigma^*, w.w' \in \mathcal{L}\}$. $\mathcal{L}$ is said to be *prefix-closed* if $pref(\mathcal{L}) = \mathcal{L}$.

Given $\Sigma' \subseteq \Sigma$, we define the *projection* operator, $\pi_{\Sigma'} : \Sigma^* \to \Sigma'^*$, on finite words that removes, from a word of Σ^*, all the events that do not belong to Σ'. Formally, $\pi_{\Sigma'}$ is recursively defined as follows: $\pi_{\Sigma'}(\varepsilon) = \varepsilon$; for $w \in \Sigma^*, \sigma \in \Sigma$, $\pi_{\Sigma'}(w.\sigma) = \pi_{\Sigma'}(w).\sigma$, if $\sigma \in \Sigma'$; $\pi_{\Sigma'}(w)$, otherwise. The definition of projection for words extends to languages. Let $\mathcal{L} \subseteq \Sigma^*$. Then $\pi_{\Sigma'}(\mathcal{L}) = \{\pi_{\Sigma'}(w) \mid w \in \mathcal{L}\}$.

Alternatively, let $\mathcal{L}' \subseteq \Sigma'^*$. The *inverse projection* of $\mathcal{L}'$ is $\pi^{-1}_{\Sigma'}(\mathcal{L}') = \{w \in \Sigma^* \mid \pi_{\Sigma'}(w) \in \mathcal{L}'\}$. We will use the notation $[\![w]\!]_{\Sigma'}$ whenever we refer to $\pi^{-1}_{\Sigma'}\pi_{\Sigma'}(w)$.

A deterministic finite automaton (DFA) is commonly used to represent the behavior of systems at a very abstract level. It is composed of a finite number of states and transitions between those states, labeled by events from its associated alphabet. Formally, a DFA is defined as $G = (Q, \Sigma, \delta, q_0)$, where Q is a finite set of states, Σ is the alphabet, $\delta : Q \times \Sigma \to Q$ is the (possibly partially-defined) state transition function, and $q_0 \in Q$ is the initial state. Note that we can extend δ over Σ^* in the usual way. We say that a transition $\delta(q, w)$ *is defined*, denoted $\delta(q, w)!$, if for $q \in Q$, $w \in \Sigma^*$, $\exists q' \in Q$ such that $\delta(q, w) = q'$.

For a given state $q \in Q$, we define the *active event set* at q as follows: $\Gamma(q) = \{\sigma \in \Sigma \mid \delta(q, \sigma)!\}$. We can extend this to be defined over a set of states $Q' \subseteq Q$ such that $\Gamma(Q') = \bigcup_{q \in Q'} \Gamma(q)$.

A *run* of G is a sequence of transitions of the form $r = q_0 \xrightarrow{\sigma_0} q_1 \ldots \xrightarrow{\sigma_{|r|-1}} q_{|r|}$, where $\delta(q_j, \sigma_j)!$, for all $j \in \{0, \ldots, |r|-1\}$. Thus, the word w associated with the run r is the sequence of transition labels: $w = \sigma_0\sigma_1 \ldots \sigma_{|r|-1}$. We say that w labels the run r of G. Let $Run(G)$ denote the set of runs of G.

A DFA G generates a regular language, which is defined as $\mathcal{L}(G) := \{w \in \Sigma^* \mid \delta(q_0, w)!\}$. Equivalently, $\mathcal{L}(G)$ is the set of words that label the runs of G.

Given two DFAs $G_1 = (Q_1, \Sigma_1, \delta_1, q_{0,1})$ and $G_2 = (Q_2, \Sigma_2, \delta_2, q_{0,2})$, their parallel composition is defined as follows: $G_1 || G_2 = (Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \delta, (q_{0,1}, q_{0,2}))$, where $\delta((q_1, q_2), \sigma) = (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma))$, if $\delta_1(q_1, \sigma)!$ and $\delta_2(q_2, \sigma)!$; $(\delta_1(q_1, \sigma), q_2)$, if $\delta_1(q_1, \sigma)!$ and $\sigma \notin \Sigma_2$; $(q_1, \delta_2(q_2, \sigma))$, if $\delta_2(q_2, \sigma)!$ and $\sigma \notin \Sigma_1$; undefined, otherwise.

Due to the limitation of the sensors attached to the system or the distributed/concurrent nature of the system, not all events can be seen. Therefore, we decompose the alphabet of the system into *observable* and *unobservable* events and we denote the set of observable events by $\Sigma_o \subseteq \Sigma$. Based on this partition, one can define the *unobservable reach*, namely, the set of states that can be reached from one state by a sequence of unobservable events:

$$\varepsilon\text{-}\mathsf{reach}(q, \Sigma_o) = \{q\} \cup \{q' \in Q \mid \exists t \in (\Sigma \backslash \Sigma_o)^*,\ \delta(q, t) = q'\}.$$

We can now construct an *observer*, using subset construction (Rabin and Scott, 1959), namely, a DFA whose language is the set of words defined over Σ_o.

Definition 1. Consider a DFA $G = (Q, \Sigma, \delta, q_0)$. The ***observer automaton*** *for G, which is constructed with respect to $\Sigma_o \subseteq \Sigma$, is*

$$\mathcal{O}_{\Sigma_o} = (X, \Sigma_o, \delta_{\Sigma_o}, x_0),$$

where $X \subseteq 2^Q$, $\delta_{\Sigma_o} : X \times \Sigma_o \to X$ is the (partial) transition function such that

$$\delta_{\Sigma_o}(x, \sigma) = \bigcup_{q \in x \wedge \delta(q,\sigma)!} \varepsilon\text{-}\mathsf{reach}(\delta(q, \sigma), \Sigma_o),$$

and $x_0 = \varepsilon\text{-}\mathsf{reach}(q_0, \Sigma_o)$ is the initial state.

2.2 Opacity

We consider a setting where no runs of a system G should reveal any secret information. In this paper, we represent a secret as a subset of states $S \subseteq Q$. Elements of S can represent states in which a system is vulnerable to attacks or where confidential information is in use and is not yet declassified.

There exists an information flow whenever an intruder or *attacker*, denoted by $\mathcal{A}$, can deduce confidential information on the execution of a system G from the observation of a subset of events $\Sigma_\mathcal{A} \subseteq \Sigma$. In our model, we want to prevent an attacker from knowing that the system is in a secret state.

Given $w \in \mathcal{L}(G)$, the observation of the attacker $\mathcal{A}$ is given by the natural projection $\pi_{\Sigma_\mathcal{A}}(w)$. We consider that attackers collect information online with executions of the system and try to infer *with certainty* whether the system's current state belongs to S. Intuitively, opacity says that an attacker is never able to infer that a run is currently in a secret state based on its observation. An attacker observes the system and builds from its observation a *belief*, i.e., an estimation of a set of states in which the system could be.

Definition 2. Consider G with a set of secret states S, an attacker $\mathcal{A}$, who observes $\Sigma_\mathcal{A}$ and its observer automaton $\mathcal{O}_{\Sigma_\mathcal{A}}$. Then S is said to be ***opaque*** *w.r.t. G and $\Sigma_\mathcal{A}$ if $\forall w \in \mathcal{L}(\mathcal{O}_{\Sigma_\mathcal{A}}),\ \delta_{\Sigma_\mathcal{A}}(x_0, w) \not\subseteq S$.*

Alternatively, we could have considered formulating secrets from a language-based point of view, where a set of secrets $\widetilde{S} \subseteq \mathcal{L}(G)$ is provided as a non-prefixed-closed regular language (Dubreil et al., 2010). With a DFA $M_{\widetilde{S}}$ such that $\mathcal{L}(M_{\widetilde{S}}) = \widetilde{S}$, we can reduce this problem to a state-based setting by considering the product of G and $M_{\widetilde{S}}$ to identify the secret states.

2.3 Supervisory control

In this paper, we will consider opacity with supervisory control, which consists of restricting the system behavior by pairing it with an access control so that control can be used to protect a secret(s).

Given a prefix-closed behavior $K \subseteq \mathcal{L}(G) \subseteq \Sigma^*$ expected from the system G, the goal of supervisory control is to enforce this behavior K on G by pairing this system with a monitor $\mathcal{S}$ (also called *controller*). In the context of opacity, K is a language that contains words that do not pass through secret states. To facilitate opacity enforcement on G, the alphabet Σ is partitioned into a set

of controllable events $\Sigma_c \subseteq \Sigma_o$ and a set of uncontrollable events $\Sigma \setminus \Sigma_c$. After each of its observations, the controller $\mathcal{S}$ issues a command of *enable* or *disable* for elements in Σ_c, whereas the uncontrollable events in $\Sigma \setminus \Sigma_c$ are permanently enabled. Formally, the controller is given by a function $\mathcal{S} : \mathcal{L}(\mathcal{O}_{\Sigma_o}) \longrightarrow 2^{\Sigma_c}$. Alternatively, $\mathcal{S}$ can be seen as a (possibly infinite) DFA $\mathcal{S}_G = (Y, \Sigma_o, \delta_{\mathcal{S}_G}, y_o)$ such that for each $\mu \in \mathcal{L}(G_{\Sigma_o})$, $\delta_{\mathcal{O}_{\Sigma_o}}(x_o, \mu) = \mathcal{S}(\mu)$.

For implementability reasons, we put conditions on the admissible restrictions of G, namely controllability and normality.

Definition 3. A prefix-closed language $K \subseteq \mathcal{L}(G)$ is

- ***controllable** w.r.t. $\mathcal{L}(G)$ and Σ_c if $K.(\Sigma \setminus \Sigma_c) \cap \mathcal{L}(G) \subseteq K$;*
- ***normal** w.r.t. $\mathcal{L}(G)$, Σ_o if $K = [\![K]\!]_{\Sigma_o} \cap \mathcal{L}(G)$.*

Under the assumption that $\Sigma_c \subseteq \Sigma_o$, there exists a supremal controllable and normal prefix-closed sub-language $K^\uparrow$ of K corresponding to the largest language included in K that can be enforced by control and this language is regular (Wonham and Ramadge (1987); Kumar et al. (1991)). If $K^\uparrow$ is not empty, there exists a *maximal* controller $\mathcal{S}$ such that $\mathcal{L}(G/\mathcal{S}) = K^\uparrow$, where $G/\mathcal{S} = G \| \mathcal{S}_G$. To ensure controllability under partial observation, given G, Σ_o and Σ_c, the valid control policies for $\mathcal{S}_G$ at state $y \in Y$ are given by $\mathcal{C}(y) = \{\gamma \in 2^{\Sigma_c} \mid \gamma \cap \Sigma_c = \Gamma(y) \cap \Sigma_c\}$, where $\Gamma(y)$ is the active event set at y.

3. OPACITY IN A MULTI-AGENT SETTING

We are interested in the problem of two adversarial agents $\mathcal{A}_1$ and $\mathcal{A}_2$, each of which independently observes the system G. Each agent has its own set of observable events $\Sigma_{o,i} \subseteq \Sigma_o$ and its own set of secret states $S_i \subseteq Q$, for $i \in \{1, 2\}$.

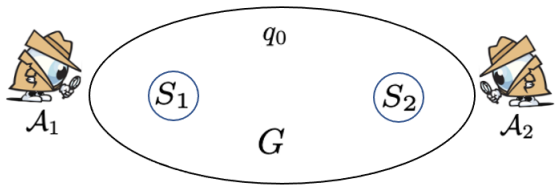

Fig. 1. A multi-agent secret-keeping scenario.

The problem here is that each agent has the aim of finding the secret of the other agent. In that sense, we shall call them *adversaries* rather than attackers in the sequel. We will consider the following problems under this scenario:

(1) **Static concurrent secrets:** Does there exist a maximal controller that can avoid an information leak from one adversary to the other, neither of which controls the system?
(2) **Dynamic concurrent secrets**: Does there exist a local controller for $\mathcal{A}_1$ to guess the secret(s) of $\mathcal{A}_2$, who is simultaneously trying to control the system to guess the secret(s) of $\mathcal{A}_1$?

Let $G = (Q, \Sigma, \delta, q_0)$ be a DFA representing the system. We assume the existence of two adversaries, each of which observes a (sub)set of observable and (possibly) controllable events, denoted by $\Sigma_{o,i}$ and $\Sigma_{c,i}$ respectively, for $i \in \{1, 2\}$. A possible controller acting globally upon G is observing $\Sigma_o \subseteq \Sigma$ and controlling $\Sigma_c \subseteq \Sigma_o$. Let $\mathcal{O}_i = (X_i, \Sigma_{o,i}, \delta_i, x_{0,i})$ be the observer automaton defined over $\Sigma_{o,i}$ (as per Definition 1, where for readability we use δ_i in lieu of $\delta_{\Sigma_{o,i}}$) for $\mathcal{A}_i$ (for $i \in \{1, 2\}$), where each observer state is a set of estimates of which states G is currently in, based on $\mathcal{A}_i$'s partial observation of G.

We will use a running example, Ex. 1, to demonstrate the concepts of the paper, beginning with the observers of G.

Example 1. Consider the system G shown on the left side of Fig. 2. Let $\Sigma_{o,1} = \{\mathsf{a}, \mathsf{c}\}$ and $\Sigma_{o,2} = \{\mathsf{b}, \mathsf{c}\}$ be the observable alphabets for $\mathcal{A}_1$ and $\mathcal{A}_2$. The unique observer automata $\mathcal{O}_1$ and $\mathcal{O}_2$ are shown in the middle and right of Fig. 2.

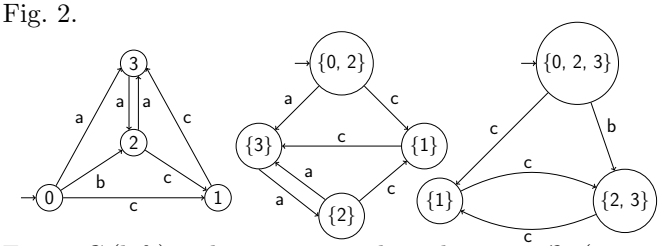

Fig. 2. G (left) and its corresponding observers $\mathcal{O}_1$ (center) and $\mathcal{O}_2$ (right)

◇

The approach for solving each problem will depend on the power and observation of the controllers as well as the powers of control and observation of the adversaries. For the first problem, we must synthesize a global controller that avoids leaking secrets from one adversary to the other, as the adversaries are passive. For the second problem, there is a concurrent game between the adversaries as they try to discover the secret(s) of the other. In this setting, the system is passive.

4. STATIC CONCURRENT SECRETS

In this section, we suppose that the adversaries cannot control the system (i.e., $\Sigma_{c,i} = \emptyset$), but must rely on an external controller to keep their secrets. The aim of the system, then, is to exert control so that no secrets are leaked between adversaries.

Problem 1. Given a system G and two agents $\mathcal{A}_1$, $\mathcal{A}_2$ with respective secrets $S_1, S_2 \subseteq Q$, does there exist a supervisor $\mathcal{S}$ such that when G is controlled by $\mathcal{S}$, $\mathcal{A}_1$ is not able to discover the secret of $\mathcal{A}_2$ and vice-versa.

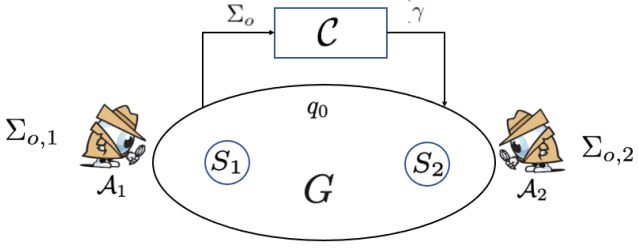

Fig. 3. Static concurrent secrets with a global controller.

Based on results from (Dubreil et al. (2008, 2010)),

$$\mathcal{F}_i = \mathcal{L}_{2^{S_j}}(\mathcal{O}_{\Sigma_{o,i}})$$

represents the set of observations leading to an information flow from $\mathcal{A}_j$ to $\mathcal{A}_i$, i.e., $w \in \Sigma^*_{o,i}$ such that $\delta_i(w, x_{0,i}) \subseteq 2^{S_j}$.

From $\mathcal{F}_i$, we can build the automata $\mathcal{G}'_i$ such that $\mathcal{F}_i = \mathcal{L}_{2^{S_j}}(\mathcal{G}'_i)$ with $S'_i = 2^{S_j}$ as the secret states. Further, it is sufficient to build $G \| \mathcal{G}'_1 \| \mathcal{G}'_2$ and consider the set of states

$$Bad = (Q \times S'_2 \times X_{\mathcal{G}'_2}) \cup (Q \times X'_{\mathcal{G}_1} \times S'_1).$$

Assuming $\mathcal{S}$ observes Σ_o and controls $\Sigma_c \subseteq \Sigma_o$, one can compute the supremal controllable and normal controller $\mathcal{S}$ that avoids the set of states Bad to solve our problem (Wonham and Ramadge, 1987; Kumar et al., 1991).

Example 2. We continue to work with G as in Fig. 2, but now we identify the secret states for each adversary: $S_1 = \{1\}$ and $S_2 = \{2\}$. The observers remain the same and we set the controllable event sets as follows $\Sigma_{c,1} = \{\mathsf{a}, \mathsf{c}\}$ and $\Sigma_{c,2} = \{\mathsf{b}, \mathsf{c}\}$. Note that there are now states in each $\mathcal{O}_i$ where the secrets are revealed. In particular, at state $\{2\}$ in $\mathcal{O}_1$, $\mathcal{A}_1$ can discover S_2. Similarly, at state $\{1\}$ in $\mathcal{O}_2$, $\mathcal{A}_2$ can discover S_1.

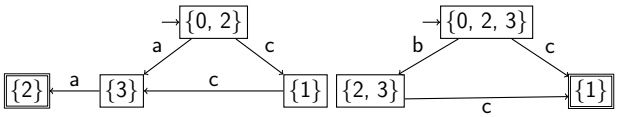

Fig. 4. $\mathcal{G}'_1$ (left) and $\mathcal{G}'_2$ (right)

Fig. 4 features the automata that generate the minimal languages that end in an adversary's secret state. A portion of $G||\mathcal{G}'_1||\mathcal{G}'_2$ is shown in Fig. 5, where states belonging to Bad are shaded. To ensure that $\mathcal{A}_2$ does not discover $\mathcal{A}_1$'s secret state, the controller disables the first occurrence of c, while it disables the second occurrence of a to prevent $\mathcal{A}_1$ from discovering $\mathcal{A}_2$'s secret state.

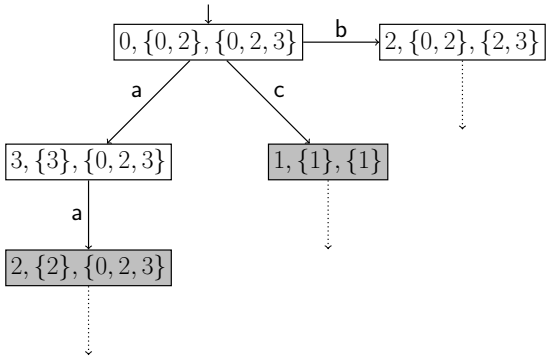

Fig. 5. A portion of $G||\mathcal{G}'_1||\mathcal{G}'_2$. States in Bad are shaded grey.

◇

5. DYNAMIC CONCURRENT SECRETS

In this section, we again assume that each adversary has a partial observation of the system. Thus, $\Sigma_{o,i} \subseteq \Sigma_o$ (resp., $\Sigma_{uo,i}$) is the set of observable (resp., unobservable) events of $\mathcal{A}_i$. Each adversary $\mathcal{A}_i$ (for $i \in \{1, 2\}$) has an associated observer automaton $\mathcal{O}_i$. To influence the discovery of their

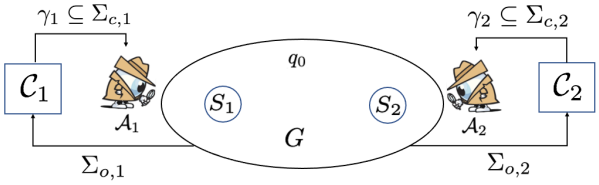

Fig. 6. Dynamic concurrent secrets with local controllers

opponent's secret state(s), both adversaries are able to issue local (i.e., their own) control decisions at each state $x_i \in X_i$. Each agent $\mathcal{A}_i$ can only control a subset of observable events $\Sigma_{c,i} \subseteq \Sigma_{o,i}$. A control decision consists of a set of events to be disabled: $\gamma_i \subseteq \Sigma_{c,i}$. A set of *valid control policies* $\mathcal{C}_i$ for $\mathcal{A}_i$ is, thus, a mapping between observer states and possible control decisions, i.e., $\mathcal{C}_i : X_i \to 2^{\Sigma_{c,i}}$, with the assumption that $\forall x_i \in X_i$, and $\gamma_i \in \mathcal{C}_i(x_i)$ then $\gamma_i \subseteq \Gamma(x_i) \cap \Sigma_{c,i}$, i.e., $\mathcal{A}_i$ can only disable controllable events that are admissible at x_i. In the sequel, we shall denote the *set* of valid control policies by $\mathcal{PC}_i = \{\mathcal{C}_i : X_i \to 2^{\Sigma_{c,i}}\}$, for $i \in \{1, 2\}$.

In terms of control in the system, an event σ may only occur in G if neither adversary has disabled σ. Thus, after each adversary has issued a control decision, σ may occur in G as long as $\sigma \in \Gamma(q) \setminus (\gamma_1 \cup \gamma_2)$. If all events in $\Gamma(q)$ have been disabled by at least one adversary, the system is stalled and no events can occur. Note that fully uncontrollable events, i.e., $\sigma \in \Sigma \setminus (\Sigma_{c,1} \cup \Sigma_{c,2})$, cannot be disabled, so G can only become stalled in states that have no transitions labelled with uncontrollable events.

Definition 4. Given $\mathcal{O}_i$, the observer of $\mathcal{A}_i$ for $i = \{1, 2\}$, a ***sequence of valid control policies of*** *$\mathcal{A}_i$ is denoted by $(\gamma^j \in \mathcal{C}_i^j)_{j \in \mathbb{N}}$ such that there exists a run in $\mathcal{O}_i$ of the form*

$$r = x_0 \xrightarrow{\sigma_0} x_1 \xrightarrow{\sigma_1} x_2 \ldots \xrightarrow{\sigma_{j-1}} x_j \xrightarrow{\sigma_j} x_{j+1} \ldots$$

with $\gamma^j \subseteq \Gamma(x_j) \cap \Sigma_{c,i}$ and $\sigma_j \in \Gamma(x_j) \setminus \gamma_j$.

Based on this definition, we can now identify the *Dynamic Concurrent Secrets* problem:

Problem 2. Given a system G and two agents $\mathcal{A}_1, \mathcal{A}_2$ with respective secrets $S_1, S_2 \subseteq Q$, does there exist a sequence of valid control policies $(\gamma_1^j \in \mathcal{C}_1^j)_{j \in \mathbb{N}}$ for agent $\mathcal{A}_1$, such that for all possible sequences of valid control policies of the form $(\gamma_2^j \in \mathcal{C}_2^j)_{j \in \mathbb{N}}$ for agent $\mathcal{A}_2$, $\mathcal{A}_1$ can discover S_2?[1]

◇

To solve this problem, we define an arena

$$\mathcal{H} = (V_{\mathsf{D}} \cup V_{\mathsf{S}}, E_{\mathsf{D}} \cup \Sigma, \delta_{\mathsf{DS}}, \delta_{\mathsf{SD}}, v_0, Bad_1, Bad_2),$$

which contains all the possible control policies for each agent, as well as the possible outcomes in the system G. Here we describe the arena in further detail.

- The arena features two types of states:
  - $V_{\mathsf{D}} \subseteq Q \times X_1 \times X_2 \times \mathcal{PC}_1 \times \mathcal{PC}_2$ is the set of *decision* states where
    - $q \in Q$ is the current state of G;
    - $x_i \in X_i$ is the current belief state in $\mathcal{O}_i$ for $\mathcal{A}_i$;

[1] Equivalently, this problem can be solved with respect to $\mathcal{A}_2$ always being able to discover S_1.

  - $\mathcal{C}_i(x_i) \subseteq 2^{\Sigma_{c,i}}$, where $\mathcal{C}_i \in \mathcal{PC}_i$, is the set of valid control policies of $\mathcal{A}_i$ in x_i; and
  - $V_{\mathsf{S}} \subseteq Q \times X_1 \times X_2 \times \mathcal{PC}_1 \times \mathcal{PC}_2 \times 2^{\Sigma}$ is the set of *system* states, where the last element of the tuple is the set of events that can occur next in G.
- $E_{\mathsf{D}} \subseteq 2^{\Sigma_{c,1}} \times 2^{\Sigma_{c,2}}$ are events corresponding to the local control decisions of $\mathcal{A}_1$ and $\mathcal{A}_2$.
- $v_0 = (q_0, x_{0,1}, x_{0,2}, \mathcal{C}_1(x_{0,1}), \mathcal{C}_2(x_{0,2})) \in V_{\mathsf{D}}$ is the initial state in the arena, where $q_0, x_{0,1}$, and $x_{0,2}$ are the initial states of G, $\mathcal{O}_1$, $\mathcal{O}_2$ respectively, whereas $\mathcal{C}_1(x_{0,1})$ corresponds to the set of valid control policies for each adversary in the initial state $x_{0,1}$.
- $Bad_1 = (Q \times X_1 \times S_1 \times \mathcal{PC}_1 \times \mathcal{PC}_2) \cap V_{\mathsf{D}}$ are states where $\mathcal{A}_2$ has discovered $\mathcal{A}_1$'s secret.
- $Bad_2 = (Q \times S_2 \times X_2 \times \mathcal{PC}_1 \times \mathcal{PC}_2) \cap V_{\mathsf{D}}$ are states where $\mathcal{A}_1$ has discovered $\mathcal{A}_2$'s secret.
- $\delta_{\mathsf{DS}} : V_{\mathsf{D}} \times E_{\mathsf{D}} \to V_{\mathsf{S}}$ is the transition function mapping decision states to system states. Given state $v_{\mathsf{D}} = (q, x_1, x_2, \mathcal{C}_1(x_1), \mathcal{C}_2(x_2)) \in V_{\mathsf{D}}$, then, for all valid local control decisions $\gamma_i \in \mathcal{C}_i(x_i)$ of $\mathcal{A}_i$:
$$\delta_{\mathsf{DS}}(q, x_1, x_2, \mathcal{C}_1(x_1), \mathcal{C}_2(x_2), (\gamma_1, \gamma_2)) = (q, x_1, x_2, \{\gamma_1\}, \{\gamma_2\}, \Gamma(q) \setminus (\gamma_1 \cup \gamma_2))$$
- $\delta_{\mathsf{SD}} : V_{\mathsf{S}} \times \Sigma \to V_{\mathsf{D}}$ maps system states to decision states and corresponds to an event occurring in G. For a given $v_{\mathsf{S}} = (q, x_1, x_2, \{\gamma_1\}, \{\gamma_2\}, \sigma) \in V_{\mathsf{S}}$ and $\sigma \in \Gamma(q) \setminus (\gamma_1 \cup \gamma_2)$,
  - $q' = \delta(q, \sigma)$
  - $x_i' = \delta_i(x_i, \pi_i(\sigma))$

  and we define the transition function as follows:
$$\delta_{\mathsf{SD}}((q, x_1, x_2, \{\gamma_1\}, \{\gamma_2\}), \sigma) = \begin{cases} (q', x_1, x_2, \{\gamma_1\}, \{\gamma_2\}), & \text{if } \sigma \in \Sigma \setminus (\Sigma_{o,1} \cup \Sigma_{o,2}); \\ (q', x_1', x_2, \mathcal{C}_1(x_1'), \{\gamma_2\}), & \text{if } \sigma \in \Sigma_{o,1} \setminus \Sigma_{o,2}; \\ (q', x_1, x_2', \{\gamma_1\}, \mathcal{C}_2(x_2')), & \text{if } \sigma \in \Sigma_{o,2} \setminus \Sigma_{o,1}; \\ (q', x_1', x_2', \mathcal{C}_1(x_1'), \mathcal{C}_2(x_2')), & \text{if } \sigma \in \Sigma_{o,1} \cap \Sigma_{o,2}. \end{cases}$$

Intuitively, $\mathcal{H}$ represents all the possible valid control decisions of $\mathcal{A}_i$ in each state of G, as well as the consequences of their decisions.

We now explain, in more detail, the different components of the arena $\mathcal{H}$.

Given a state $v_{\mathsf{d}} = (q, x_1, x_2, \mathcal{C}_1(x_1), \mathcal{C}_2(x_2)) \in V_{\mathsf{D}}$, each adversary can choose any valid control policy $\gamma_i \in \mathcal{C}_i(x_i)$ according to its belief state x_i. By triggering the tuple (γ_1, γ_2), we enter into state $v_{\mathsf{s}} = (q, x_1, x_2, \mathcal{C}_1(x_1), \mathcal{C}_2(x_2), (\gamma_1, \gamma_2))$ that "memorizes" the chosen control policies.

Similarly, for a state $v_{\mathsf{s}} = (q, x_1, x_2, \mathcal{C}_1(x_1), \mathcal{C}_2(x_2), (\gamma_1, \gamma_2)) \in V_{\mathsf{S}}$, the system G can choose any event that is allowed by $\gamma_1 \cup \gamma_2$, i.e., $\sigma \in \Gamma(q) \setminus (\gamma_1 \cup \gamma_2)$. If σ is unobservable to both the adversaries, i.e., $\sigma \in \Sigma \setminus (\Sigma_{o,1} \cup \Sigma_{o,2})$, then the belief state of each adversary $\mathcal{A}_i$ does not change and we keep track of their previous decisions, i.e., $\mathcal{C}_i(x_i)$ is reduced to the singleton $\{\gamma_i\}$. If $\sigma \in \Sigma_{o,1} \setminus \Sigma_{o,2}$, then it can only be observed by $\mathcal{A}_1$. Thus, $\mathcal{A}_1$ is free to choose any valid control policy in $\mathcal{C}_1(x_1')$, with $x_1' = \delta_1(x_1, \sigma)$, while $\mathcal{A}_2$ does not observe anything and, thus, its control decision remains γ_2 (and vice-versa if σ is only observable by $\mathcal{A}_2$). Finally, when σ is observable by both adversaries, then both evolve to a new belief state $x_i' = \delta_i(x_i, \sigma)$ and both are free to choose any new control policy according to x_i' (for $i \in \{1, 2\}$).

A run of $\mathcal{H}$, which begins at v_0, is of the form:

$$r = \begin{pmatrix} v_0 = (q_0, x_{0,1}, x_{0,2}, \mathcal{C}_1(x_{0,1}), \mathcal{C}_2(x_{0,2})) \\ \downarrow (\gamma_1, \gamma_2), \gamma_i \in \mathcal{C}_i(x_{0,i}) \\ (q_0, x_{0,1}, x_{0,2}, \{\gamma_1\}, \{\gamma_2\}) \\ \downarrow \sigma \in \Gamma(q) \setminus (\gamma_1 \cup \gamma_2) \\ (\delta(q_0, \sigma), \delta_1(x_{0,1}, \sigma), \delta_2(x_{0,2}, \sigma), \\ \quad \mathcal{C}_1(\delta_1(x_{0,1}, \sigma)), \mathcal{C}_2(\delta_2(x_{0,2}, \sigma))) \\ \vdots \end{pmatrix}$$

Note that if $\sigma \in \Sigma_{uo,i}$ for $\mathcal{A}_i$, then a transition of $\delta_i(x_i, \sigma)$ does not result in a state change for $\mathcal{A}_i$, and we assume that $\mathcal{C}_i(\delta_i(x_i, \sigma)) = \{\gamma_i\}$ from the preceding state of the arena.

We denote by $Run(\mathcal{H})$ the set of possible runs in $\mathcal{H}$. Then $Run_{B_2}(\mathcal{H}) \subseteq Run(\mathcal{H})$ is the set of runs that ends in a state of Bad_2. Intuitively, for any runs in Run_{B_2}, $\mathcal{A}_1$ will reach the secret of its adversary $\mathcal{A}_2$.

The language of $\mathcal{H}$ is given by
$$\mathcal{L}(\mathcal{H}) = ((\gamma_1^j, \gamma_2^j).\sigma_j)^*$$
such that there exists a run $r \in Run(\mathcal{H})$ of the form
$$(v_{\mathsf{d}_j} \xrightarrow{(\gamma_1^j, \gamma_2^j).\sigma_j} v_{\mathsf{d}_{j+1}})^*.$$

The language $\mathcal{L}_{B_2}(\mathcal{H}) \subseteq \Sigma^* \Sigma_{o,1}$ is defined as the sequences of $\mathcal{L}(\mathcal{H})$ that end in Bad_2 by triggering an event of $\Sigma_{o,1}$ (otherwise $\mathcal{A}_1$ will not be aware that it reaches Bad_2).

Lemma 5. Given a sequence $t = ((\gamma_1^j, \gamma_2^j).\sigma_j))_{j \in \mathbb{N}}^* \in \mathcal{L}_{B_2}(\mathcal{H})$, *then* $\delta_1(x_{0,1}, \pi_1^{\mathcal{H}}(t)) \subseteq S_2$, *where* $\pi_1^{\mathcal{H}}$ *is a function from* $(2^{\Sigma_{c,1}} \times 2^{\Sigma_{c,2}}) \times \Sigma \to \Sigma_{o,1}$ *such that* $\pi_1^{\mathcal{H}}((\gamma_1, \gamma_2).\sigma) = \pi_1(\sigma)$.

Proof. Assume that $\exists t \in \mathcal{L}_{B_2}(\mathcal{H})$, then there exists a run of the form
$$r_t = v_0 \xrightarrow{(\gamma_1^0, \gamma_2^0).\sigma_0} \ldots \xrightarrow{(\gamma_1^k, \gamma_2^k).\sigma_k} v_{\mathsf{D}_{k+1}} \in Bad_2,$$
with $\sigma_k \in \Sigma_{o,1}$ and $v_{\mathsf{D}_j} = (q_j, x_1^j, x_2^j, \mathcal{C}_1(x_1^j), \mathcal{C}_2(x_2^j))$, for $j \in \{0..k\}$. From r_t, one can extract a run of the form
$$r_1 = x_{0,1} \xrightarrow{\sigma_1'} x_1' \ldots \xrightarrow{\sigma_k} x_k \in S_2$$
where each event of the run is in $\Sigma_{o,1}$. By definition of $\mathcal{H}$, r_1 is a run of $\mathcal{O}_1$. Hence the result. ■

Based on the previous definitions, let us now define the notion of a state-based control strategy.

Definition 6. A **state-based control strategy** *for* $\mathcal{A}_1$ *is given by the function*
$$\Xi_{\mathcal{A}_1} : V_{\mathsf{D}} \to \Sigma_{c,1}.$$
That is, $\Xi_{\mathcal{A}_1}((q, x_1, x_2, \mathcal{C}_1(x_1), \mathcal{C}_2(x_2))) = \gamma_i \in \mathcal{C}_1(x_1)$. *Similarly, we define a state-based control strategy for* $\mathcal{A}_2$, *where* $\Xi_{\mathcal{A}_2} : V_{\mathsf{D}} \to \Sigma_{c,2}$. ◇

Given $(q, x_1, x_2, \mathcal{C}_1(x_1), \mathcal{C}_2(x_2))$, the goal of a state-based control strategy for $\mathcal{A}_1$ is to choose or fix, in all states $x_1 \in X_1$, one possible valid control among the ones of $\mathcal{C}_1(x_1)$ and then free the choice of $\mathcal{A}_2$ amongst all its possible valid control policies at $x_2 \in X_2$.

Given $\Xi_{\mathcal{A}_1}$, the runs of $\mathcal{H}$ are reduced to runs of the form

$$r_{\Xi_{\mathcal{A}_1}} = \begin{pmatrix} v_0 = (q_0, x_{0,1}, x_{0,2}, \Xi_{\mathcal{A}_1}(v_0), \mathcal{C}_2(x_{0,2})) \\ \downarrow (\Xi_{\mathcal{A}_1}(v_0), \gamma_2), \gamma_2 \in \mathcal{C}_2(x_{0,2}) \\ (q_0, x_{0,1}, x_{0,2}, \Xi_{\mathcal{A}_1}(v_0), \{\gamma_2\}) \\ \downarrow \sigma \in \Gamma(q) \setminus (\Xi_{\mathcal{A}_1}(v_0) \cup \gamma_2) \\ (\delta(q_0, \sigma), \delta_1(x_{0,1}, \sigma), \delta_2(x_{0,2}, \sigma), \\ \qquad \Xi_{\mathcal{A}_1}(v_1), \mathcal{C}_2(\delta_2(x_{0,2}, \sigma))) \\ \vdots \end{pmatrix}$$

with

$$v_1 = (\delta(q_0, \sigma), \delta_1(x_{0,1}, \sigma), \delta_2(x_{0,2}, \sigma), \\ \mathcal{C}_1(\delta_1(x_{0,1}, \sigma)), \mathcal{C}_2(\delta_2(x_{0,2}, \sigma))).$$

Similarly, if $\sigma \in \Sigma_{uo,1}$ for $\mathcal{A}_1$, then a transition of $\delta_1(x, \sigma)$ does not result in a state change for $\mathcal{A}_1$, and we assume that $\Xi_{\mathcal{A}_1}(v_0) = \Xi_{\mathcal{A}_1}(v_1)$ from the preceding state of the arena and we denote this set of runs by $Run^{\Xi_{\mathcal{A}_1}}(\mathcal{H}) \subseteq Run(\mathcal{H})$.

Only part of the runs of $Run_{B_2}(\mathcal{H})$ solve Problem 2, as it should be true for any choices of $\mathcal{A}_2$ along the trajectories that lead to Bad_2 and, equivalently, for the language $\mathcal{L}_{B_2}(\mathcal{H}) \subseteq \Sigma^* \Sigma_{o,1}$.

Moreover, given $\Xi_{\mathcal{A}_1}$, there might exist a run r such that, once the choice of $\mathcal{A}_1$ is fixed for every state in V_D, there exists a choice of $\mathcal{A}_2$ that forms a loop in $\mathcal{H}$. If such a loop exists, then $\mathcal{A}_2$ can go against the decisions of $\mathcal{A}_1$ w.r.t. $\Xi_{\mathcal{A}_1}$ so that any runs cannot lead to Bad_2, whatever the decisions of $\Xi_{\mathcal{A}_1}$. If $\Xi_{\mathcal{A}_1}$ does not meet such a problem then it is called a *loop-free* strategy. We denote this set of loop-free strategies, w.r.t. $\mathcal{A}_2$, by $\mathcal{L}^{\ell f}_{B_2}(\mathcal{H})$.

Next, we define a winning control strategy of $\mathcal{A}_1$ that leads to a state in Bad_2 and is loop-free w.r.t. $\mathcal{A}_2$.

Definition 7. A ***winning*** *state-based control strategy is given by the function $\Xi^{B_2}_{\mathcal{A}_1}$ of Def 6 such that, for $v_i = (q, x_1, x_2, \mathcal{C}_1(x_1), \mathcal{C}_2(x_2))$*

$$(\Xi^{B_2}_{\mathcal{A}_1}(v_i))^* \subseteq \{(\gamma_1^i)^* \in \Sigma^*_{c,1} \mid \\ \forall \gamma_2^i \in \mathcal{C}_2(x_2), \exists \sigma_i \in \Gamma_{\mathcal{H}}(v_i) \setminus (\gamma_1^i \cup \gamma_2^i)),\ s.t. \\ ((\gamma_1^i, \gamma_2^i).\sigma_i))^* \in \mathcal{L}^{\ell f}_{B_2}(\mathcal{H})\}.$$

We denote the language that fulfills the definition of $\mathcal{L}(\mathcal{H})$, according the Definition 7, by $\mathcal{L}^{\Xi^{B_2}_{\mathcal{A}_1}}(\mathcal{H})$.

Theorem 1. Given $\Xi^{B_2}_{\mathcal{A}_1}$ as in Definition 7, if $\mathcal{L}^{\Xi^{B_2}_{\mathcal{A}_1}}(\mathcal{H}) \not\subseteq \emptyset$, then there exists a winning control strategy that solves Problem 2. ◇

Let us now explain how one can prune the arena $\mathcal{H}$ in order to see whether adversary $\mathcal{A}_1$ can win against $\mathcal{A}_2$. Let $B \subseteq V_\mathsf{D}$ be an arbitrary set of states in $\mathcal{H}$.

$$Pre_{\mathcal{A}_1}(B) = \{\ v_\mathsf{d} = (q, x_1, x_2, \mathcal{C}_1(x_1), \mathcal{C}_2(x_2)) \in V_\mathsf{D} \setminus B \mid \\ \exists \gamma_1 \in \mathcal{C}_1(x_1), \forall \gamma_2 \in \mathcal{C}_2(x_2), \\ \exists \sigma \in (\Gamma(q) \setminus (\gamma_1 \cup \gamma_2)) \cap \Sigma_{o,1} \\ \text{s. t. } \delta_{\mathsf{SD}}(\delta_{\mathsf{DS}}(v_d, (\gamma_1, \gamma_2)), \sigma) \in B\}$$

Then $Pre_{\mathcal{A}_1}(B)$ is the set of "previous" states of V_D from which $\mathcal{A}_1$ has a possible control decision γ_1 such that regardless of $\mathcal{A}_2$'s control decision γ_2, there is an enabled event σ that leads to a state in B and is observable to $\mathcal{A}_1$.

In other words, if we consider $B = Bad_2$, we compute the set of states of V_D for which there exists a valid control policy for $\mathcal{A}_1$ such that for any possible valid control policy of $\mathcal{A}_2$, the system can trigger an event enabled by all the adversaries that leads to Bad_2, while avoiding loops (as we only consider new states in $Pre_{\mathcal{A}_1}(B)$. Further, the set of states of V_D that lead to Bad_2 is given by the following fixed-point computation:

$$E_0 = Bad_2; \\ E_1 = Pre_{\mathcal{A}_1}(Bad_2); \\ E_{k+1} = E_k \cup Pre_{\mathcal{A}_1}(E_k), k > 1.$$

Let E be the result of this fixed-point computation.

As noted previously, E_1 is the set of states that allow us to reach Bad_2 in one step by choosing the set of possible valid control policies $\gamma_1 \in \mathcal{C}_1(x_1)$ w.r.t. x_1 so that whatever the choice of $\mathcal{A}_2$, the system G can reach Bad_2 by an observable event. Meanwhile, starting from any state of E, it is easy to see that there exists a sequence that reaches Bad_2.

Theorem 2. Given the following arena

$$\mathcal{H} = ((V_D \cup V_\mathsf{S}), E_\mathsf{D} \cup \Sigma, \delta_{\mathsf{DS}}, \delta_{\mathsf{SD}}, v_0, Bad_1, Bad_2).$$

Then consider

$$\mathcal{H}_E = (((V_\mathsf{D} \setminus E) \cup V_\mathsf{S}), E_\mathsf{D} \cup \Sigma, \delta_{\mathsf{DS}}, \delta_{\mathsf{SD}}, v_0, Bad_1, Bad_2).$$

If $\mathcal{H}_E$ is not reduced to the empty automaton, then there exists a winning control strategy that solves Problem 2. ◇

Example 3. We reuse G and its observers $\mathcal{O}_1$ and $\mathcal{O}_2$ from Ex. 2, but examine it within the setting of Problem 2. As a reminder, $\Sigma_{o,1} = \Sigma_{c,1} = \{\mathsf{a}, \mathsf{c}\}$ and $\Sigma_{o,2} = \Sigma_{c,2} = \{\mathsf{b}, \mathsf{c}\}$. A portion of the arena $\mathcal{H}$ is shown in Fig 7. As noted before, the system is not opaque for either adversary. Even without displaying the entire arena, there are, in fact, winning strategies for $\mathcal{A}_1$. For example, at the initial state, if $\mathcal{A}_1$ chooses $\gamma_1 = \{\mathsf{c}\}$, then G is forced to execute either the sequence aa or baa, leading to a winning configuration for $\mathcal{A}_1$. ◇

6. CONCLUSIONS

We presented two problems involving the use of control to prevent information leakage between two adversaries. In the first problem, we used an existing result from Dubreil et al. (2010) to determine whether or not there is a global control policy, where the agents are passive, to preserve the secret of each adversary. For the second problem, where the system is passive and each agent exerts local control to discover the secret of its adversary, we presented a sufficient condition for the existence of a winning strategy for $\mathcal{A}_1$ to always discover $\mathcal{A}_2$'s secret. Future extensions of this work include extending the results to multiple secrets and incorporating a quantitative component to the secret to present different incentives for the interactions of the adversaries.

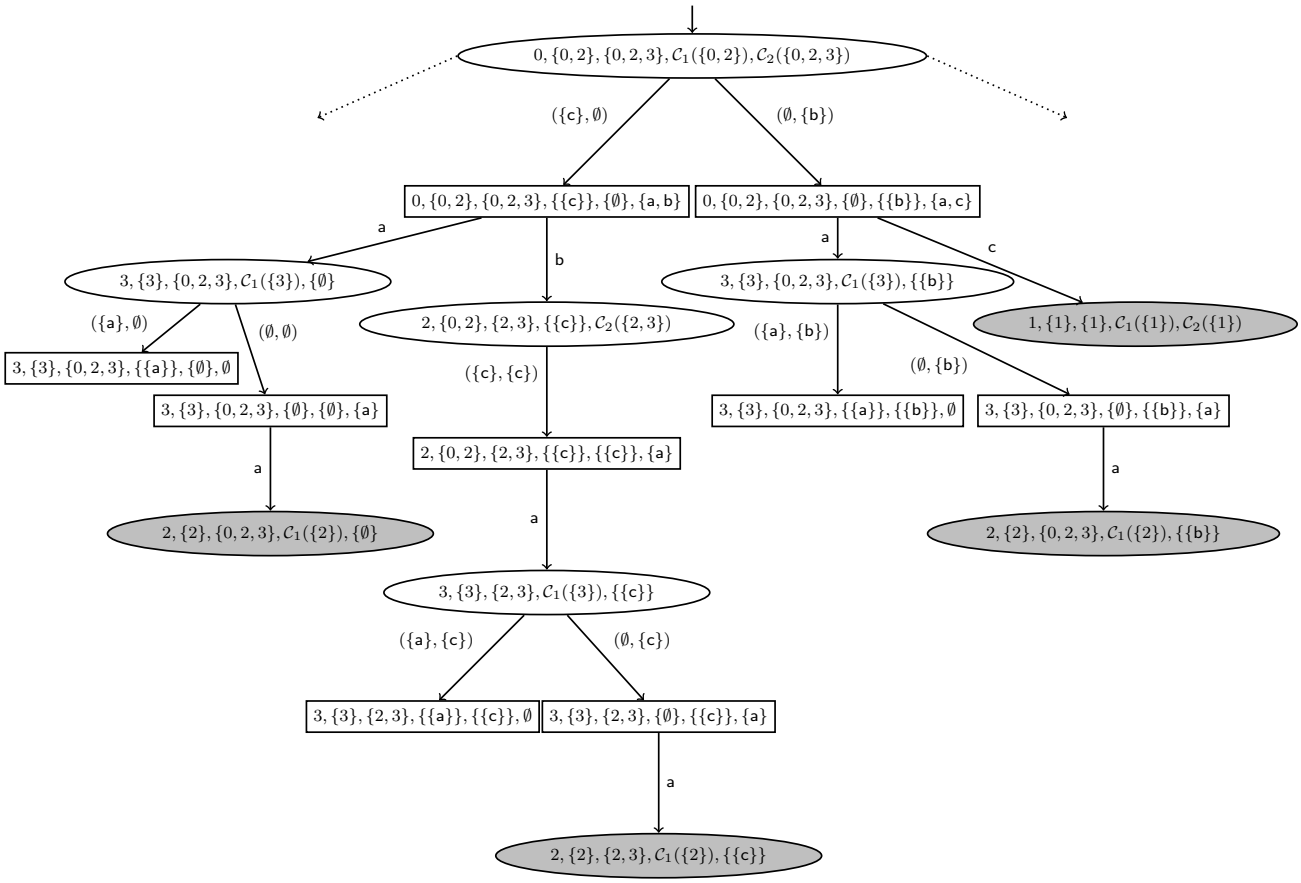

Fig. 7. A portion of the arena $\mathcal{H}$ for the dynamic concurrent secrets version of Ex. 1. States in V_{D} are denoted as ellipses, while states in V_{S} are denoted as rectangles. The states shown in grey are in $Bad_1 \cup Bad_2$.

REFERENCES

Badouel, E., Bednarczyk, M., Borzyszkowski, A., Caillaud, B., and Darondeau, P. (2007). Concurrent secrets. *Discrete Event Dyn Syst*, 17, 425–446.

Bryans, J., Koutny, M., Mazare, L., and Ryan, P. (2008). Opacity generalised to transition systems. *Int. J. Inf. Secur.*, 7(6), 421–435.

Dubreil, J., Darondeau, P., and Marchand, H. (2008). Opacity enforcing control synthesis. In *Proc. WODES*, 28–35.

Dubreil, J., Darondeau, P., and Marchand, H. (2010). Supervisory control for opacity. *IEEE Transactions on Automatic Control*, 55(5), 1089–1100.

Goguen, J.A. and Meseguer, J. (1982). Security policies and security models. In *1982 IEEE Symposium on Security and Privacy*, 11–11. IEEE.

Helouet, L., Mullins, J., and Marchand, H. (2018). Concurrent secrets with quantified suspicion. In *Proc. ACSD*, 75–84.

Jacob, R., Lesage, J.J., and Faure, J.M. (2016). Overview of Discrete Event Systems Opacity: models, validation, and quantification. *Annual Reviews in Control*.

Kumar, R., Garg, V., and Marcus, S.I. (1991). On controllability and normality of discrete event dynamical systems. *Systems & Control Letters*, 17, 157–169.

Mazare, L. (2004). Using unification for opacity properties. In *Proc. IFIP WG1.7 Workshop on Issues in the Theory of Security*, 165–176.

McCullough, D. (1988). Noninterference and the composability of security properties. In *Proceedings of the IEEE Symposium on Security and Privacy*, 177–186. IEEE Computer Society.

Rabin, M. and Scott, D. (1959). Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2), 114–125.

Ramadge, P. and Wonham, W. (1987). Supervisory control of a class of discrete event processes. *SIAM J. Contr. Optim.*, 25(1), 206–230.

Riccardo Focardi, R.G. (1994). A taxonomy of trace-based security properties for ccs. In *Proceedings The Computer Security Foundations Workshop VII*, 126–136.

Wittbold, J.T. and Johnson, D.M. (1990). Information flow in nondeterministic systems. In *IEEE Symposium on Security and Privacy*, volume 161.

Wonham, W. and Ramadge, P. (1987). On the supremal controllable sublanguage of a given language. *SIAM Journal on Control and Optimization*, 25(3), 637–659.