The Computation of Sensor Activation Decisions in Discrete-Event Systems

Technical Report 2013-603

David Sears* and Karen Rudie

March 9, 2013

Abstract

This paper considers partially-observed discrete-event systems where sensors are associated with events observable to an agent monitoring the system. The agent is capable of turning the sensors for events on and off dynamically, depending on the trajectory of the system. Reading data from the sensors may be costly so it is imperative that their use be reduced for reasons such as energy, bandwidth or security. When a sensor for an event is on / active any occurrence of the event is detected by the agent and is not detected otherwise. The agent may employ different sensor activation policies, depending on the task at hand. Sensor activation policies are defined over the transitions of a state-transition representation of the system. From sensor activation policies a map from observed event sequences to sensor activation decisions can be computed which the agent can use to determine which sensors to turn on / off and when. In this paper, we consider three subclasses of sensor activation policies of increasing generality. For each subclass, we demonstrate ways to compute maps from observed event sequences to sensor activation decisions in polynomial time. However, for the last subclass considered, we demonstrate that verifying if an arbitrary sensor activation policy belongs to this class is PSPACE-complete.

1 Introduction

The behaviour of many critical dynamical systems require monitoring by external agents. The behaviour of such systems is monitored so that critical failures in the system may be diagnosed or anticipated prior to their occurrence. Control of such systems can be conducted in order to prevent failures from occurring in cases when failures can be anticipated in advance and actuators for the system are available.

*E-Mail: sears@cs.queensu.ca; Address: School of Computing, Queen's University, Kingston, Ontario K7L 3N6, Canada

E-Mail: karen.rudie@queensu.ca; Address: Department of Electrical & Computer Engineering, Queen's University, Kingston, Ontario K7L 3N6, Canada

The behaviour of these systems is reported to agents via sensors embedded in the system. It may be the case that reading information from the sensors or keeping them active for a prolonged duration is costly. Using sensors may be costly in cases where each sensor only has a limited energy supply, or because limited bandwidth is available for transmission of readings from sensors to the agent or for security purposes in situations where the agent wishes to be undetected while taking measurements of the system. In all these cases it is imperative to minimize reading information from sensors or minimize the duration over which the sensors are turned on / active if the agent can turn sensors on / off.

The sensor activation problem can be modeled in many different ways, depending on the model of the system, the particular problem at hand (e.g., control, failure diagnosis, failure prognosis, opacity), the nature of the information reported by sensors, and the cost function to be minimized. In this paper we consider that the system is specifically a discrete-event system (DES): a system with a discrete state-space and event-driven dynamics. In such systems events occurs sequentially and asynchronously. Sensor activation problems for discrete-event systems have been considered in [1, 2, 5, 9, 11, 12, 13, 15, 16].

Specifically, we consider discrete-event systems which can be modeled by finite state automata. For such systems the state-space is finite and the set of events which cause the system to transition from one discrete state to another is finite. As these systems can be modeled by finite state automata, state-transition representations of such systems exist where events label the transitions of the system model. The individual sensors used by an agent are associated with individual events in the system. Turning an event sensor on / off depends on the particular trajectory of the system. Specifically, given an automaton state-transition representation of the system, we consider that turning an event sensor on / off depends on the particular transition of the system representation: a sensor for an event labeling a particular transition of the system is always on whenever the transition is encountered or is always off whenever the transition is encountered. When the turning of event sensors on / off depends on the transitions of the system representation we say that the sensor activation decisions by the agent are dictated by a sensor activation policy (introduced in [16]) defined over the transitions of the system. Depending on the problem at hand (e.g., control, failure diagnosis, failure prognosis, opacity), one sensor activation policy may be preferable over another or two sensor activation policies may be incomparable, depending on the cost function for active sensors or the relative cost of sensor activation policies.

In this paper, we do not consider the minimization of sensor activations. This topic has been explored in previous papers [1, 5, 9, 11, 12, 13, 15, 16]. Instead, we consider the computation of maps from sequences of event observations to sensor activation decisions corresponding to given sensor activation policies. For a given sensor activation policy defined over the transitions of the system automaton model, such a computation typically involves the determinization of nondeterministic finite automata. It is well known that the determinization of a nondeterministic finite automaton is, in the worst-case, exponential in the state-space of the nondeterministic finite automaton. In this paper, we consider specific classes of sensor activation policies and demonstrate procedures for computing these maps which are polynomial in the size of the representation of the system. The procedures we provide do not involve determinization

of nondeterministic finite automata. We do not investigate determinization of nondeterministic finite automata defined by the systems and sensor activation policies considered. This is an open topic.

Specifically, we consider sensor activation policies which satisfy various notions of "feasibility". Informally, a sensor activation policy is feasible if any two states of the system which are indistinguishable under the sensor activation policy are followed by the same sensor activation decisions for certain subsets of observable events. Feasibility was originally introduced in [7] in the context of communication policies between two agents in discrete-event systems. The most general notion of feasibility for sensor activation policies is introduced in [16]. The notions of feasibility that we consider are strictly stronger than the notion considered in [16]. We consider notions of feasibility of increasing generality.

The organization of the paper is as follows. A preliminaries section introducing the requirements for understanding the remainder of the paper is provided in Section 2. Afterward, in Section 3 we consider sensor activation policies which satisfy a very strong notion of feasibility and, for such sensor activation policies, demonstrate how the automaton representation of the system itself can be used for determining the set of sensors to be activated following a sequence of observable events. This notion of feasibility is relaxed in Section 4. There we demonstrate that a very coarse estimate of the true state of the system may be used in the computation of a map from observed event sequences to sensor activation decisions and, furthermore, the computation is polynomial in the state-space and event set cardinalities of the system. Following this, we consider a further generalization of sensor activation policies in Section 5. For this more general class, it may not be the case that the coarse estimate of the true state of the system introduced in Section 4 can be used for computing maps from observed event sequences to sensor activation decisions. In fact, we demonstrate that determining if the coarse estimate can be used is PSPACE-complete. When it can be used, we demonstrate a procedure polynomial in the state-space and event set cardinalities of the system for computing maps from observed event sequences to sensor activation decisions. Finally, we summarize the results of the paper in Section 6.

2 Preliminaries

Here some basics on formal languages, automata and the original sensor activation model of [16] are recalled. For a more detailed background, refer to [16]. Also, familiarity with asymptotic time complexity characterizations of algorithms and their associated notation (e.g., Big O) is required.

Given a set, S, we denote the cardinality of S by $|S|$. An *alphabet* is a finite set of distinct symbols (also called *events*, for our purposes). Let Σ represent an alphabet. Let Σ^+ denote the set of all finite sequences of symbols in Σ of the form $\sigma_1\sigma_2\ldots\sigma_k$ where $k \geq 1$ is arbitrary and $\sigma_i \in \Sigma$ for all $i \in \{1,\ldots,k\}$. We refer to a sequence of length zero (i.e., consisting of no symbols) as the empty sequence, denoted by $\varepsilon \notin \Sigma$. The *Kleene-closure* of Σ, denoted by Σ^*, is defined as $\Sigma^* = \{\varepsilon\} \cup \Sigma^+$. An element of Σ^* is a *string* over the alphabet Σ. We also refer to ε as the *empty string*.

For $s \in \Sigma^*$ we say $t \in \Sigma^*$ is a *prefix* of s, denoted by $t \leq s$, if $s = tu$ for some $u \in \Sigma^*$. Thus $\varepsilon \leq s$ and $s \leq s$ for all $s \in \Sigma^*$.

A *nondeterministic finite automaton* (NFA), A, is defined as a tuple $A = (Q, \Sigma, \delta, q_0)$ where Q is the nonempty set of states, Σ is the alphabet, q_0 is the initial state, and $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to 2^Q$ is the (state-)transition function, a partial function which determines the transitions from state to state on occurrence of symbols in $\Sigma \cup \{\varepsilon\}$. We extend δ to a function $\delta : Q \times \Sigma^* \to 2^Q$ by induction on length of strings. Automaton A is said to be a *deterministic finite automaton* (DFA) if δ is more specifically of the form $\delta : Q \times \Sigma \to Q$. We use $\delta(q, \sigma)!$ for $q \in Q$ and $\sigma \in \Sigma$ to denote that transition $\delta(q, \sigma)$ is defined. Similarly, we use $\delta(q, s)!$ for string $s \in \mathcal{L}(A)$ to denote that $\delta(q, s)$ is defined. The language generated by A is $\mathcal{L}(A) = \{s \in \Sigma^* \mid \delta(q_0, s)!\}$. Given A, a state-transition graph representation can be constructed [3]. Given a state $q \in Q$, the ε-reach of q is defined as $\{q' \in Q \mid q' \in \delta(q, \varepsilon)\}$.

When A is an NFA an algorithm known as the *subset construction* can be used for constructing a DFA $DET(A)$ where $\mathcal{L}(DET(A)) = \mathcal{L}(A)$ [3]. The state-space of $DET(A)$ is a subset of the power set of the state-space of A. For space considerations, we do not outline the operation of the subset construction and instead refer the reader to [3].

We consider untimed DES modeled by DFA $G = (X, \Sigma, \xi, x_0)$. The set of transitions of G is defined as $TR(G) := \{(x, e) \in X \times \Sigma \mid \xi(x, e)!\}$. From the perspective of an agent which observes events generated by G, the set of events Σ can be partitioned into Σ_o, the set of events whose occurrences can be observed by the agent, and Σ_{uo}, the set of events whose occurrences cannot be observed by the agent. Associated with each observable event is a sensor that can be used to detect occurrences of the event. When the sensor is activated (i.e., the event's sensor is on) any occurrence of the event is detected by the agent. Otherwise, when the sensor is deactivated (i.e., the event's sensor is off) any occurrence of the event is not detected by the agent.

Next the sensor activation model of [16] which is used in this paper is recalled.

When to activate event sensors is described by a *sensor activation map* $\omega : \mathcal{L}(G) \to 2^{\Sigma_o}$. Specifically, for a string $s \in \mathcal{L}(G)$, $\omega(s)$ is the subset of observable events corresponding to the sensors that are active after s. Given sensor activation map ω, we use induction to define the corresponding *information map* $\theta^\omega : \mathcal{L}(G) \to \Sigma_o^*$ as follows. For the empty string ε, $\theta^\omega(\varepsilon) = \varepsilon$, and for all $s, se \in \mathcal{L}(G)$ with $e \in \Sigma$

$$\theta^\omega(se) = \begin{cases} \theta^\omega(s)e & \text{if } e \in \omega(s) \\ \theta^\omega(s) & \text{otherwise} \end{cases}$$

In words, after the occurrence of s, the next event e is *seen* or *observed* by the agent when it occurs after s if and only if the sensor for e is active for the agent after the occurrence of s. The information map θ^ω plays a comparable role to the projection map P of standard partially observed discrete-event systems. That is, $\theta^\omega(s)$ indicates which events in the string s are observed. The difference is that an event e in s is either always observed under P or never observed, whereas with dynamic sensor activations, whether e is observed in $\theta^\omega(s)$ will depend on where in s it lies.

It is important to note that not all arbitrary sensor activation maps ω will be "feasible" based on the information available to the agent. To guarantee feasibility, it is required that any two strings of events that are confusable / indistinguishable to the agent must be followed by the same activation decision

for every event. Namely, ω must be "compatible" with the information map θ^ω that is built from it. Formally, ω is said to be *feasible* if

$$(\forall e \in \Sigma)(\forall se, s'e \in \mathcal{L}(G))\ \theta^\omega(s) = \theta^\omega(s') \Rightarrow [e \in \omega(s) \Leftrightarrow e \in \omega(s')]. \tag{1}$$

The above definitions of ω and θ^ω are language-based. In this paper we do not consider arbitary language-based sensor activation maps. Instead, we consider sensor activation maps that are defined over the transitions of a given automaton G. For such maps, any two strings leading to the same state of G are followed by the same sensor activation decision for every event. In [16] such sensor activation maps are called *implementable*. In [16] implementable sensor activation maps were considered in order to restrict the solution space of the problem they considered. By refining the state-transition structure of G, finer solutions to the problem they considered can be computed.

For implementable sensor activation maps, we can associate the activation of sensors with the transitions in G: the event associated with each transition in $TR(G)$ is either sensed (activated) by the agent, or not. Given an implementable sensor activation map ω, the set of transitions sensed by the agent through sensor activations using ω can be defined. This set is denoted by $\Omega \subseteq TR(G)$. Here $(x, e) \in \Omega$ means that $\forall s \in \mathcal{L}(G)\ \xi(x_0, s) = x \Rightarrow e \in \omega(s)$. We call Ω a *(sensor activation) policy*.

In particular, ω can be obtained from Ω as follows:

$$\omega(s) = \{e \in \Sigma_o \mid (\xi(x_0, s), e) \in \Omega\}. \tag{2}$$

It is not difficult to see that, when ω is obtained from Ω by (2), the following holds:

$$\forall s \in \mathcal{L}(G), \forall e \in \Sigma_o, e \in \omega(s) \Rightarrow se \in \mathcal{L}(G) \tag{3}$$

Information map θ^Ω is used to denote the information map θ^ω when such ω is derived from Ω by (2).

Policy Ω is feasible if the corresponding ω is feasible. It is not difficult to see that Ω is *feasible* if

$$(\forall e \in \Sigma)(\forall se, s'e \in \mathcal{L}(G))\ \theta^\Omega(s) = \theta^\Omega(s') \Rightarrow [(\xi(x_0, s), e) \in \Omega \Leftrightarrow (\xi(x_0, s'), e) \in \Omega]. \tag{4}$$

For a given sensor activation policy Ω, we define the *unobserved reach* of state $x \in X$ under Ω as the set of states that can be reached from x via "unobserved" transitions (i.e., transitions that do not exist in Ω).

Given G and Ω, we can construct the observer automaton DET_Ω which maps sequences of observed events to sensor activation decisions by replacing those transitions in G not in Ω by ε and applying the subset construction resulting in a DFA $DET_\Omega(G)$. Given an observed event sequence s, $DET_\Omega(G)$ can then be used to compute the sensor activation decisions following s by taking the union of all events labelling transitions in $DET_\Omega(G)$ from the state in $DET_\Omega(G)$ reached by s.

Next we introduce the equivalence class of a state in G under a policy Ω. Given DFA G and policy Ω, we say that states $x, x' \in X$ are indistinguishable if there exists $s, s' \in \mathcal{L}(G)$ such that $\xi(x_0, s) = x$, $\xi(x_0, s') = x'$ and $\theta^\Omega(s) = \theta^\Omega(s')$.

Given G and Ω, we can compute the set of pairs of indistinguishable states of X, denoted by T_Ω, using the CLUSTER-TABLE algorithm of [14]. We can interpret $T_\Omega \subseteq X \times X$ as a binary relation. It is not difficult to see that T_Ω is reflexive and symmetric but not necessarily transitive. Such a relation is known as a tolerance relation [4]. Given a set $X' \subseteq X$ and state $x \in X$, when we say that $X' T_\Omega x$ (respectively, $x T_\Omega X'$) we mean that $\forall x' \in X', x' T_\Omega x$ (resp., $\forall x' \in X', x T_\Omega x'$).

The transitive-closure of T_Ω, denoted by T_Ω^*, is an equivalence relation on X. We consider the equivalence class of a state $x \in X$, denoted by $[x]$, to be defined using this equivalence relation: $[x] = \{z \in X \mid x T_\Omega^* z\}$. We will find the following characterization of $z \in [x]$ using T_Ω rather than T_Ω^* useful in the proof details of results in the sequel. For state $z \in X, z \in [x]$ if and only if

$$\begin{aligned} &\exists n \geq 0, \exists x^0, x^1, x^2, \ldots, x^n \in X, x^0 = x, \\ &\qquad x^1 T_\Omega x^0 \wedge x^2 T_\Omega x^1 \wedge \ldots \wedge z T_\Omega x^n. \end{aligned} \tag{5}$$

If $z \in [x]$ then $x \in [z]$ by (5). It follows that $[x] = [z]$.

Though the topic of computing policies for purposes of state disambiguation is considered in [16], it is not indicated how the computed policies can be used to construct a map from observed event sequences to sensor activation decisions. Explicitly, for policy Ω which satisfies (4), this is done by constructing observer automaton DET_Ω from G and Ω then taking the union of all events labelling transitions in $DET_\Omega(G)$ from the state in $DET_\Omega(G)$ reached by the observed event sequence. However, determinizing NFA is well known to be, in the worst-case, exponential in $|X|$.

In this paper we consider policies Ω where there exist procedures for computing a map from observed event sequences to sensor activation decisions that are polynomial in the size of G. In order to achieve this we consider policies that satisfy notions of feasibility that are stronger than (4), the feasibility condition considered in [16].

3 Computing sensor activation decisions for a small class of policies

In this section we consider policies that satisfy a stronger notion of feasibility than (4). The notion that we consider is the following.

$$\begin{aligned} (\forall e \in \Sigma_o)(\forall s, s' \in \mathcal{L}(G))\, \theta^\Omega(s) = \theta^\Omega(s') \\ \Rightarrow [(\xi(x_0, s), e) \in \Omega \Leftrightarrow (\xi(x_0, s'), e) \in \Omega] \end{aligned} \tag{6}$$

That (6) is satisfied by a policy implies that if two states are indistinguishable under the policy then an event sensor is activated following one state if and only if it is activated following the other state. For policies satisfying (4) this is only required of the intersection of events that follow both states, not their union as (6) requires.

We use the following example to illustrate the differences between policies satisfying (6) and those satisfying (4). Consider the plant in Fig. 1. We consider Ω to consist of those transitions whose event label is boxed. Specifically, $\Omega = \{(x_1, e_1), (x_2, e_2), (x_4, e_1)\}$. One can verify that this policy satisfies (4).

However, Ω does not satisfy (6). One can verify that $\theta^{\Omega}(e_1) = \theta^{\Omega}(e_2e_1) = e_1$, resulting in states x_2 and x_5 being indistinguishable. An observer which observes string e_1 using policy Ω knows that if the plant is in state x_2 then the sensor for e_2 needs to be activated. However, if the plant is in state x_5 then the sensor for e_2 does not necessarily have to be activated. This is permitted for satisfying (4). However, it is not permitted if the policy is to satisfy (6). In order to satisfy (6) transition (x_2, e_2) could be removed from Ω.

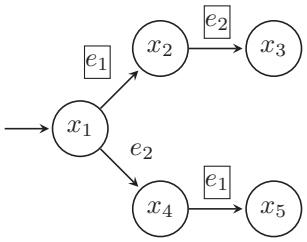

Figure 1: Plant and policy which satisfies (4) but does not satisfy (6).

It is not difficult to see that if policy Ω satisfies (6) then map ω derived from Ω by (2) satisfies the following notion of feasibility for language-based sensor activation maps which is stronger than (1).

$$(\forall e \in \Sigma_o)(\forall s, s' \in \mathcal{L}(G))\ \theta^{\omega}(s) = \theta^{\omega}(s') \Rightarrow [e \in \omega(s) \Leftrightarrow e \in \omega(s')]. \tag{7}$$

For policies satisfying (6), in this section we effectively demonstrate that if $s \in \mathcal{L}(G)$ then $\theta^{\Omega}(s) \in \mathcal{L}(G)$[1] and, furthermore, an event sensor is activated following the state reached by s if and only if it is activated following the state reached by $\theta^{\Omega}(s)$. This allows one to compute sensor activation decisions following a sequence of observed events (which are themselves determined by previous sensor activations) easily: simply transition from the initial state of G to the state reached by the observed event sequence then activate a sensor for an event from the reached state if and only if it is in Ω. In this way, G is used directly for mapping observations to sensor activation decisions rather than having to apply the subset construction. The remainder of this section is dedicated to proving the aforementioned result. In the following, the formal details focus on using a state's equivalence class for determining sensor activation decisions. However, we ultimately prove that for all $s \in \mathcal{L}(G)$, $[\xi(x_0, s)] = [\xi(x_0, \theta^{\Omega}(s))]$. This allows for sensor activation decisions to be based on the state in G reached by $\theta^{\Omega}(s)$.

First we prove some results regarding state equivalence classes for policies satisfying (6). We have the following immediate result on sensor activation decisions following states in the same equivalence class for policies satisfying (6).

Lemma 1. *Given $G = (X, \Sigma, \xi, x_0)$ and policy $\Omega \subseteq TR(G)$ which satisfies (6), $\forall x, z \in X$, $\forall e \in \Sigma_o$, if $[x] = [z]$ then $(x, e) \in \Omega \Leftrightarrow (z, e) \in \Omega$.*

[1] In constrast, in standard partially observed discrete-event systems, if a string s is in $\mathcal{L}(G)$, this does not imply that $P(S)$ is in $\mathcal{L}(G)$

Proof. Follows directly by definition of state equivalence class in (5) and that Ω satisfies (6). □

From this result we have the following Lemma which demonstrates that it suffices for an agent to keep track of a state's equivalence class for determining sensor activation decisions.

Lemma 2. *Given* $G = (X, \Sigma, \xi, x_0)$ *and policy* $\Omega \subseteq TR(G)$ *which satisfies* (6), $\forall x, z \in X$, $\forall e \in \Sigma_o$, *if* $[x] = [z]$ *and* $(x, e) \in \Omega$ *then* $[\xi(x, e)] = [\xi(z, e)]$.

Proof. The following holds by $[x] = [z]$ and definition of $[x]$:

$$\exists n \geq 0, \exists x^0, x^1, x^2, \ldots, x^n \in X, x^0 = x, \quad x^1 T_\Omega x^0 \wedge x^2 T_\Omega x^1 \wedge \ldots \wedge z T_\Omega x^n. \tag{8}$$

$\forall i \in \{1, \ldots, n\}$, (x^i, e) is defined and $(x^i, e) \in \Omega$ by **Lemma 1**, $e \in \Sigma_o$, $[x] = [x^i]$ and $(x, e) \in \Omega$. Similarly, $(z, e) \in \Omega$. The fact that $x^i T_\Omega x^{i-1}$ and definition of T_Ω implies that there exists $s, s' \in \mathcal{L}(G)$ where $\xi(x_0, s) = x^{i-1}$, $\xi(x_0, s') = x^i$ and $\theta^\Omega(s) = \theta^\Omega(s')$. Since $(x^{i-1}, e), (x^i, e) \in \Omega$, $\theta^\Omega(se) = \theta^\Omega(s)e$ and $\theta^\Omega(s'e) = \theta^\Omega(s')e$. Then $\theta^\Omega(se) = \theta^\Omega(s'e)$ by previous and $\theta^\Omega(s) = \theta^\Omega(s')$. This fact and $\xi(x_0, s) = x^{i-1}$, $\xi(x_0, s') = x^i$ implies that $\xi(x^i, e) T_\Omega \xi(x^{i-1}, e)$. Similarly, $\xi(z, e) T_\Omega \xi(x^n, e)$. By these results and (8) the following holds:

$$\xi(x^1, e) T_\Omega \xi(x^0, e) \wedge \xi(x^2, e) T_\Omega \xi(x^1, e) \wedge \ldots \wedge \xi(z, e) T_\Omega \xi(x^n, e).$$

Then $\xi(z, e) \in [\xi(x, e)]$ by this fact, $x = x^0$ and (5). It follows that $[\xi(x, e)] = [\xi(z, e)]$. □

We use this result to prove the main result of this section. The following theorem states that to determine an agent's sensor activation decision following string $s \in \mathcal{L}(G)$ it suffices to compute the sensor activation decision of the state reached by $\theta^\Omega(s)$ from x_0 in G. An observer automaton constructed from G and Ω where sensor activation decisions are defined from the states reached *need not be constructed*.

Theorem 1. *Given* $G = (X, \Sigma, \xi, x_0)$ *and policy* $\Omega \subseteq TR(G)$ *which satisfies* (6), $\forall s \in \mathcal{L}(G), [\xi(x_0, s)] = [\xi(x_0, \theta^\Omega(s))]$.

Proof. This proof follows by induction on the length of $\theta^\Omega(s)$.

When $|\theta^\Omega(s)| = 0$, $\theta^\Omega(s) = \varepsilon$. The fact that $\theta^\Omega(\varepsilon) = \varepsilon$ and previous imply $\theta^\Omega(\theta^\Omega(s)) = \varepsilon$. Then $\theta^\Omega(s) = \theta^\Omega(\theta^\Omega(s))$. It follows that $\xi(x_0, s) T_\Omega \xi(x_0, \theta^\Omega(s))$ by previous and definition of T_Ω. So $[\xi(x_0, s)] = [\xi(x_0, \theta^\Omega(s))]$ by this fact and (5).

Now for the inductive step. Consider when $|\theta^\Omega(s)| = n + 1$. Let $s = s_1 e s_2$ where $s_1, s_2 \in \Sigma^*$, $e \in \Sigma_o$ and $s_1 e$ is the shortest prefix of s where $\theta^\Omega(s_1 e) = \theta^\Omega(s_1)e = \theta^\Omega(s)$. Since $|\theta^\Omega(s_1)| = n$ and by the inductive hypothesis it follows that $[\xi(x_0, s_1)] = [\xi(x_0, \theta^\Omega(s_1))]$. Since $\theta^\Omega(s_1 e) = \theta^\Omega(s_1)e$ it must be that $(\xi(x_0, s_1), e) \in \Omega$. Then $[\xi(x_0, \theta^\Omega(s_1)e)] = [\xi(x_0, s_1 e)]$ by this fact, $e \in \Sigma_o$, $[\xi(x_0, s_1)] = [\xi(x_0, \theta^\Omega(s_1))]$ and **Lemma 2**. Since $\theta^\Omega(s_1 e) = \theta^\Omega(s_1)e$ it follows that $[\xi(x_0, \theta^\Omega(s_1 e))] = [\xi(x_0, s_1 e)]$. The fact that $\theta^\Omega(s) = \theta^\Omega(s_1 e s_2) = \theta^\Omega(s_1)e$ and $\theta^\Omega(s_1 e) = \theta^\Omega(s_1)e$ imply $\theta^\Omega(s_1 e s_2) = \theta^\Omega(s_1 e)$. It follows that $\xi(x_0, s_1 e s_2) T_\Omega \xi(x_0, s_1 e)$ by previous and definition of T_Ω. So $[\xi(x_0, s_1 e s_2)] =$

$[\xi(x_0, s_1 e)]$. Also, since $\theta^\Omega(s_1 e s_2) = \theta^\Omega(s_1 e)$ it follows that $[\xi(x_0, \theta^\Omega(s_1 e s_2))] = [\xi(x_0, \theta^\Omega(s_1 e))]$. Thus $[\xi(x_0, s_1 e s_2)] = [\xi(x_0, \theta^\Omega(s_1 e s_2))]$ by the previous two facts and $[\xi(x_0, \theta^\Omega(s_1 e))] = [\xi(x_0, s_1 e)]$. □

To reaffirm, after computing Ω which satisfies (6) and which is used for accomplishing some task (e.g., state disambiguation in G) it suffices to use G directly for computing sensor activation decisions given the agent's observation. From x_0, when an agent receives observation of an event $e_1 \in \Sigma_o$ whose sensor is active at x_0 it merely transitions to $\xi(x_0, e_1)$, which must be defined due to (3). Its sensor activation decision is then defined based on the transitions whose sensors are to be activated from state $\xi(x_0, e_1)$. Similarly, from $\xi(x_0, e_1)$ when the agent observes event $e_2 \in \Sigma_o$ the next sensor activation decisions are to be based on the transitions from state $\xi(x_0, e_1 e_2)$. The updating of the agent's sensor activation decision continues in this manner following events that it observes according to its previous sensor activation decisions.

For instance, consider the plant and policy of Fig. 2. We consider Ω to consist of those transitions whose event label is boxed. Specifically, $\Omega = \{(x_1, e_1), (x_2, e_2), (x_4, e_1), (x_5, e_2)\}$. One can verify that this policy satisfies (6). Consider string $e_2 e_1$. This string appears as $\theta^\Omega(e_2 e_1) = e_1$ to an observer using policy Ω. The event sensors to be activated following $e_2 e_1$ are $\{e_2\}$. That is, $\omega(e_2 e_1) = \{e_2\}$. Note also that $\theta^\Omega(e_2 e_1) = e_1$ is a string in $\mathcal{L}(G)$. Furthermore, $\omega(\theta^\Omega(e_2 e_1)) = \{e_2\}$, the same set of sensor activations as $\omega(e_2 e_1)$. By **Theorem 1** and the fact that Ω satisfies (6), for any string $s \in \mathcal{L}(G)$, $\theta^\Omega(s) \in \mathcal{L}(G)$ and $\omega(\theta^\Omega(s)) = \omega(s)$.

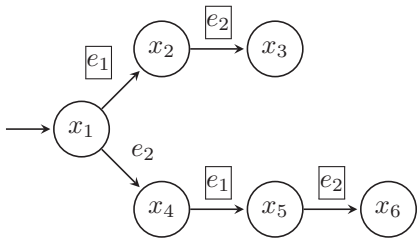

Figure 2: Plant and policy which satisfies (6) used to demonstrate how sensor activation decisions are computed.

We note that this approach does not allow one to compute a state estimate of G. To do this, computation of the observer automaton from G and Ω is required. However, maintaining a state estimate is not required so long as agent decisions following indistinguishable pairs of strings / states are the same. For instance, consider problems of centralized control when the controllable, observable specification automaton K is a subautomaton of G. When this is the case the enable / disable event control actions can be defined over the transitions of G. Observability [6] can be characterized as a state disambiguation condition defined over the states of G [14]: no two states where the control action from both states is different should be indistinguishable. For Ω satisfying (6) under which the state disambiguation condition is satisfied, it suffices to compute sensor activation and control decisions from $\xi(x_0, \theta^\Omega(s))$ instead of from the state estimate following s computed using an observer automaton constructed from G and Ω.

In the next section we consider a generalization of (6). For policies that satisfy this more general condition, it is not necessarily the case that $\theta^{\Omega}(s) \in \mathcal{L}(G)$ when $s \in \mathcal{L}(G)$, as was the case for policies satisfying (6) considered in this section. For such policies we prove that it suffices to base sensor activation decisions on the state equivalence class $[\xi(x_0, s)]$, once again avoiding the need to apply the subset construction. In Section 5 we continue to investigate conditions under which sensor activation decisions can be based on state equivalence classes.

4 Computing sensor activation decisions for a more general class of policies

In Section 3 we considered policies which satisfy a strong notion of feasibility: if two states of G are indistinguishable under a policy then (i) they must be followed by exactly the same sensor activation decisions and (ii) if the sensor for an event is activated then the event must label outgoing transitions from both states. In this section we consider policies which satisfy a weaker notion of feasibility: (i) must be satisfied and, furthermore, if the sensor for an event is activated then the event must label transitions following some unobserved sequences of transitions from both states. Formally, for the sensor activation maps considered in Section 3, if $e \in \omega(s)$ then $se \in \mathcal{L}(G)$. In this section we consider a generalization where instead if $e \in \omega(s)$ then $\exists u \in \Sigma^*$ such that $sue \in \mathcal{L}(G)$ and $\theta^{\omega}(s) = \theta^{\omega}(su)$.

However, it must be noted that the notion of feasibility that we consider in this section is still stronger than (4).

As before, when ω is implementable it can be implemented by a policy Ω. However, in this section we consider that ω is defined from Ω differently than usual (i.e., differently than by (2)). For any string $s \in \mathcal{L}(G)$, the set of event sensors activated following s (denoted by $\omega^2_{\Omega}(s)$) correspond to those events which are observed following unobserved extensions of s. Formally, this is defined as follows:

$$\omega^2_{\Omega}(s) = \{e \in \Sigma_o \mid \exists n \geq 0, \exists \sigma_1, \ldots, \sigma_n \in \Sigma \text{ such that } (\xi(x_0, s), \sigma_1) \notin \Omega \wedge \ldots \wedge (\xi(x_0, s\sigma_1 \ldots \sigma_{n-1}), \sigma_n) \notin \Omega \wedge (\xi(x_0, s\sigma_1 \ldots \sigma_n), e) \in \Omega\}. \quad (9)$$

To distinguish this sensor activation map from ω derived by (2), we use ω^1_{Ω} to denote the sensor activation map defined in (2) from Ω.

The sensor activation maps defined by (9) that we consider in this section satisfy (7). We consider what this means in terms of policies that implement these maps. By definition of (7) we know that ω^2_{Ω} is feasible if

$$(\forall e \in \Sigma_o)(\forall s, s' \in \mathcal{L}(G))\ \theta^{\omega^2_{\Omega}}(s) = \theta^{\omega^2_{\Omega}}(s') \Rightarrow [e \in \omega^2_{\Omega}(s) \Leftrightarrow e \in \omega^2_{\Omega}(s')].$$

This holds if and only if the following holds by definition of ω^2_Ω

$$
\begin{aligned}
&(\forall e \in \Sigma_o)(\forall s, s' \in \mathcal{L}(G))\ \theta^{\omega^2_\Omega}(s) = \theta^{\omega^2_\Omega}(s') \Rightarrow \\
&[(\exists \sigma_1, \ldots, \sigma_n \in \Sigma \text{ such that } (\xi(x_0, s), \sigma_1) \notin \Omega \wedge \ldots \wedge \\
&(\xi(x_0, s\sigma_1 \ldots \sigma_{n-1}), \sigma_n) \notin \Omega \wedge (\xi(x_0, s\sigma_1 \ldots \sigma_n), e) \in \Omega) \\
&\Leftrightarrow \\
&(\exists \sigma'_1, \ldots, \sigma'_m \in \Sigma \text{ such that } (\xi(x_0, s'), \sigma'_1) \notin \Omega \wedge \ldots \wedge \\
&(\xi(x_0, s'\sigma'_1 \ldots \sigma'_{m-1}), \sigma'_m) \notin \Omega \wedge (\xi(x_0, s'\sigma'_1 \ldots \sigma'_m), e) \in \Omega)].
\end{aligned}
$$

For brevity we can express the above equivalently by the following by definition of $\theta^{\omega^1_\Omega}$ and by selecting $u = \sigma_1 \ldots \sigma_n$ and $u' = \sigma'_1 \ldots \sigma'_m$ appropriately for each $e \in \Sigma_o$ and pair of strings $s, s' \in \mathcal{L}(G)$ where $\theta^{\omega^2_\Omega}(s) = \theta^{\omega^2_\Omega}(s')$:

$$
\begin{aligned}
&(\forall e \in \Sigma_o)(\forall s, s' \in \mathcal{L}(G))\ \theta^{\omega^2_\Omega}(s) = \theta^{\omega^2_\Omega}(s') \Rightarrow \qquad (10)\\
&[(\exists u \in \Sigma^* \text{ such that } sue \in \mathcal{L}(G) \wedge \theta^{\omega^1_\Omega}(su) = \theta^{\omega^1_\Omega}(s) \wedge (\xi(x_0, su), e) \in \Omega) \\
&\Leftrightarrow \\
&(\exists u' \in \Sigma^* \text{ such that } s'u'e \in \mathcal{L}(G) \wedge \theta^{\omega^1_\Omega}(s'u') = \theta^{\omega^1_\Omega}(s') \wedge (\xi(x_0, s'u'), e) \in \Omega)].
\end{aligned}
$$

As the information maps $\theta^{\omega^1_\Omega}, \theta^{\omega^2_\Omega}$ are derived from $\omega^1_\Omega, \omega^2_\Omega$ which are, in turn, derived from Ω, we consider (10) to be a feasibility condition satisfiable by Ω.

In the remainder of this section, when we say that Ω satisfies (4) (resp., (6)) we intend that the θ^Ω used in the definition of (4) (resp., (6)) be equal to $\theta^{\omega^1_\Omega}$.

It can be proven that if Ω satisfies (6) then it satisfies (10). More specifically, it is not difficult to see that if a given policy Ω satisfies (6) then $\omega^1_\Omega = \omega^2_\Omega$. However, if Ω does not satisfy (6) then it is not necessarily the case that $\omega^1_\Omega = \omega^2_\Omega$. Under such circumstances Ω may still satisfy (10). Thus the set of policies satisfying (10) contains the set of policies satisfying (6) (i.e., (6) is stronger than (10)).

Though a policy Ω may satisfy (10), this does not necessarily imply that if a sensor for an event e is active at a state $x \in X$ then all occurrences of e following unobserved transition sequences from x shall be observed. For instance, consider the plant and policy of Fig. 3. We consider Ω to consist of those transitions whose event label is boxed. Specifically, $\Omega = \{(x_2, e_2), (x_4, e_2)\}$. One can verify that this policy satisfies (10). The set of sensors activated following $\varepsilon \in \mathcal{L}(G)$ is $\omega^2_\Omega(\varepsilon) = \{e_2\}$. So if e_2 follows ε then it should be observed and the transition it labels should be in Ω. However, (x_1, e_2) is not in Ω where $\xi(x_1, \varepsilon) = x_1$.

So an additional requirement on policy Ω is required to ensure that when a sensor for observable event e is active at state x (i.e., $(x, e) \in \Omega$) any subsequent occurrence of the event is observed following an unobserved sequence of events occurring after e has been activated at x. This requirement is exactly (4), the feasibility condition on policies introduced in [16].

So in the remainder of this section we require that policy Ω satisfies both (4) and (10). The following result allows us to simplify the definition of (10) when Ω satisfies (4).

Lemma 3. *Given $G = (X, \Sigma, \xi, x_0)$ and policy $\Omega \subseteq TR(G)$ which satisfies* (4)*, $\forall e \in \Sigma, \forall se \in \mathcal{L}(G)$, $e \in \omega^1_\Omega(s) \Leftrightarrow e \in \omega^2_\Omega(s)$.*

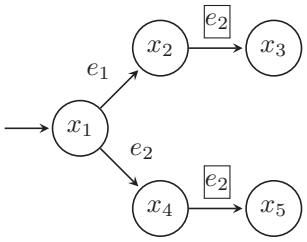

Figure 3: Plant and policy which satisfies (10) where sensor for e_2 is activated at x_1 but (x_1, e_2) is not observed.

Proof. Consider any $e \in \Sigma$, $se \in \mathcal{L}(G)$.

(i) Suppose $e \in \omega^1_\Omega(s)$. Since $e \in \omega^1_\Omega(s)$ and by definition of ω^1_Ω ((2)) it follows that $(\xi(x_0, s), e) \in \Omega$. Then $e \in \omega^2_\Omega(s)$ by definition of ω^2_Ω.

(ii) Suppose $e \notin \omega^1_\Omega(s)$. Then $(\xi(x_0, s), e) \notin \Omega$. Assume $e \in \omega^2_\Omega(s)$. Then $\exists \sigma_1 \dots \sigma_n \in \Sigma$, $(\xi(x_0, s), \sigma_1) \notin \Omega \wedge \dots \wedge (\xi(x_0, s\sigma_1 \dots \sigma_{n-1}), \sigma_n) \notin \Omega \wedge (\xi(x_0, s\sigma_1 \dots \sigma_n), e) \in \Omega$. It follows that $\theta^{\omega^1_\Omega}(s) = \theta^{\omega^1_\Omega}(s\sigma_1 \dots \sigma_n)$ by definition of $\theta^{\omega^1_\Omega}$. Since $se \in \mathcal{L}(G)$ it follows that $(\xi(x_0, s), e) \in TR(G)$. That $(\xi(x_0, s), e) \in TR(G)$, $(\xi(x_0, s), e) \notin \Omega$, $(\xi(x_0, s\sigma_1 \dots \sigma_n), e) \in \Omega$ and $\theta^{\omega^1_\Omega}(s) = \theta^{\omega^1_\Omega}(s\sigma_1 \dots \sigma_n)$ holds implies that Ω does not satisfy (4). A contradiction is reached. Thus, it must be that $e \notin \omega^2_\Omega(s)$. □

A corollary of the previous result is the following which can be used to simplify (10).

Corollary 1. *If* $\Omega \subseteq TR(G)$ *satisfies* (4) *then* $\forall s \in \mathcal{L}(G)$, $\theta^{\omega^1_\Omega}(s) = \theta^{\omega^2_\Omega}(s)$.

Proof. Follows by **Lemma 3** and by definition of $\theta^{\omega^1_\Omega}$, $\theta^{\omega^2_\Omega}$. □

By the previous result, when Ω satisfies (4), we can reason about the appearance of a string using $\theta^{\omega^1_\Omega}$ rather than $\theta^{\omega^2_\Omega}$ when the sensor activation map under consideration, ω^2_Ω, is derived from Ω by (9). In light of this, (10) can be simplified to the following where $\theta^{\omega^2_\Omega}$ has been replaced by $\theta^{\omega^1_\Omega}$.

$$
\begin{aligned}
&(\forall e \in \Sigma_o)(\forall s, s' \in \mathcal{L}(G))\ \theta^{\omega^1_\Omega}(s) = \theta^{\omega^1_\Omega}(s') \Rightarrow \qquad (11)\\
&[(\exists u \in \Sigma^* \text{ such that } sue \in \mathcal{L}(G) \wedge \theta^{\omega^1_\Omega}(su) = \theta^{\omega^1_\Omega}(s) \wedge (\xi(x_0, su), e) \in \Omega)\\
&\Leftrightarrow\\
&(\exists u' \in \Sigma^* \text{ such that } s'u'e \in \mathcal{L}(G) \wedge \theta^{\omega^1_\Omega}(s'u') = \theta^{\omega^1_\Omega}(s') \wedge (\xi(x_0, s'u'), e) \in \Omega)].
\end{aligned}
$$

An example is provided demonstrating a policy which satisfies (4) and (11) (equivalently, (10)) but not (6), the feasibility condition considered in Section 3. Consider the plant in Fig. 4. We consider Ω to consist of those transitions whose event label is boxed. Condition (6) is not satisfied by Ω as $\theta^{\omega^1_\Omega}(e_1) = \theta^{\omega^1_\Omega}(e_2e_1)$, $(\xi(x_0, e_1), e_2) \in \Omega$ and $(\xi(x_0, e_2e_1), e_2) \notin \Omega$ (in fact this transition is not defined). One can easily verify that (4) and (11) are satisfied. For any two

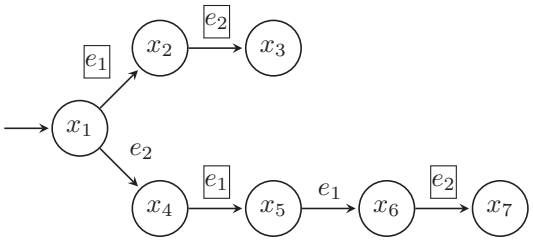

Figure 4: Plant and policy defined which satisfies (4) and (11) but not (6).

states which can be indistinguishable with each other, if an observable event e labels transitions from both states then the same sensor activation decision is made for e from both states. Also, if an event e is observed after some sequence of unobserved transitions from a given state x then it is observed after sequences of unobserved transitions from any state x' which can be indistinguishable with x.

In the remainder of this section we prove that, for policies satisfying (4) and (11), sensor activation decisions can be made based on the sensor activation decisions of an arbitrary state in the state equivalence class of the true state of G. Furthermore, for any observed event sequence, each state the plant could be in (i.e., each state of G in the state estimate of G reached by the observed event sequence in the observer automaton constructed from G and Ω) belong to the same state equivalence class. This allows one to construct an automaton denoted by $[G]$ (with some slight abuse of notation) whose states are the state equivalence classes of X and which maps an observed event sequence to a state equivalence class which contains states of G which are followed by the same sensor activation decisions as those following the true state of G. Furthermore, we provide an algorithm for the construction of this automaton which is in polynomial in $\|X$ and $|\Sigma|$ where the automaton's state-space cardinality is at most $|X|$. The subset construction need not be applied for computing a map from observed event sequences to sensor activation decisions. However, the complexity of determinizing G when transitions in Ω are replaced by ε and Ω satisfies (4) and (11) is open.

Before we proceed, some notation is introduced for convenience of presentation. For given Ω and $x \in X$, we denote the set of sequences of events which label transitions not in Ω from state x by $S_x^{\not\Omega}$. It is formally defined below:

$$S_x^{\not\Omega} = \{s \in \Sigma^* \mid \exists n \geq 0, s = \sigma_1 \dots \sigma_n \wedge (x, \sigma_1) \notin \Omega \wedge \dots \\ \wedge (\xi(x, \sigma_1 \dots \sigma_{n-1}), \sigma_n) \notin \Omega\}$$

The remainder of the section proceeds in the same manner as from **Lemma 1** to the end of Section 3. We first provide a result analogous to **Lemma 1**:

Lemma 4. *Given $G = (X, \Sigma, \xi, x_0)$, policy $\Omega \subseteq TR(G)$ which satisfies (4) and (11), $\forall x, z \in X, \forall e \in \Sigma_o$, if $[x] = [z]$ then $\exists s \in S_x^{\not\Omega}, (\xi(x, s), e) \in \Omega \Leftrightarrow \exists s' \in S_z^{\not\Omega}, (\xi(z, s'), e) \in \Omega$*

Proof. Since $z \in [x]$ and by definition of $[x]$ in (5) we have the following:

$$\exists n \geq 0, \exists x^0, x^1, x^2, \ldots, x^n \in X, x^0 = x,$$
$$x^1 T_\Omega x^0 \wedge x^2 T_\Omega x^1 \wedge \ldots \wedge z T_\Omega x^n.$$

Since $x^{i+1} T_\Omega x^i$ it follows that $\exists t, t' \in \mathcal{L}(G)$ such that $\xi(x_0, t) = x^{i+1}$, $\xi(x_0, t') = x^i$ and $\theta^{\omega^1_\Omega}(t) = \theta^{\omega^1_\Omega}(t')$. Then, since Ω satisfies (11), it follows that

$(\forall e \in \Sigma_o)$

$[(\exists u \in \Sigma^*$ such that $tue \in \mathcal{L}(G) \wedge \theta^{\omega^1_\Omega}(tu) = \theta^{\omega^1_\Omega}(t) \wedge (\xi(x^{i+1}, u), e) \in \Omega)$

$\Leftrightarrow$

$(\exists u' \in \Sigma^*$ such that $t'u'e \in \mathcal{L}(G) \wedge \theta^{\omega^1_\Omega}(t'u') = \theta^{\omega^1_\Omega}(t') \wedge (\xi(x^i, u'), e) \in \Omega)]$.

Let $u = u_1 \ldots u_m$. Since $\theta^{\omega^1_\Omega}(tu) = \theta^{\omega^1_\Omega}(t)$ it follows that $u_{j+1} \notin \omega^1_\Omega(u_j)$. By definition of ω^1_Ω, it follows that $(\xi(x^{i+1}, u_1 \ldots u_j), u_{j+1}) \notin \Omega$. Then $\exists v \in S^{\mathcal{A}}_{x^{i+1}}$, $(\xi(x^{i+1}, v), e) \in \Omega$ by above. Symmetrically, $\exists v' \in S^{\mathcal{A}}_{x^i}$, $(\xi(x^i, v'), e) \in \Omega$. Thus by these results and above the following holds

$$\exists v \in S^{\mathcal{A}}_{x^{i+1}}, (\xi(x^{i+1}, v), e) \in \Omega \Leftrightarrow \exists v' \in S^{\mathcal{A}}_{x^i}, (\xi(x^i, v'), e) \in \Omega.$$

From iterative application of the above fact the Lemma statement holds. □

From this result we have the following:

Lemma 5. *Given G, policy Ω defined over the transitions of G which satisfies (4) and (11), $\forall x, z \in X$, $\forall e \in \Sigma_o$, $\forall s \in S^{\mathcal{A}}_x$, $\forall s' \in S^{\mathcal{A}}_z$, if $[x] = [z]$, $(\xi(x, s), e) \in \Omega$ and $(\xi(z, s'), e) \in TR(G)$ then $[\xi(\xi(x, s), e)] = [\xi(\xi(z, s'), e)]$.*

Proof. The following holds by $[x] = [z]$ and definition of $[x]$:

$$\exists n \geq 0, \exists x^0, x^1, x^2, \ldots, x^n \in X, x^0 = x, \tag{12}$$
$$x^1 T_\Omega x^0 \wedge x^2 T_\Omega x^1 \wedge \ldots \wedge z T_\Omega x^n.$$

$\forall i \in \{1, \ldots, n\}$, $\exists s_{x^i} \in S^{\mathcal{A}}_{x^i}$, $(\xi(x^i, s_{x^i}), e)$ is defined and $(\xi(x^i, s_{x^i}), e) \in \Omega$ by **Lemma 4**, $e \in \Sigma_o$, $[x] = [x^i]$ and $(\xi(x, s), e) \in \Omega$. Similarly, $\exists s_z \in S^{\mathcal{A}}_z$, $(\xi(z, s_z), e)$ is defined and $(\xi(z, s_z), e) \in \Omega$. As $s_z, s' \in S^{\mathcal{A}}_z$, it follows that $\exists w, w' \in \mathcal{L}(G)$, $\xi(x_0, w) = \xi(z, s_z)$, $\xi(x_0, w') = \xi(z, s')$ and $\theta^{\omega^1_\Omega}(w) = \theta^{\omega^1_\Omega}(w')$. It follows that $(\xi(z, s'), e) \in \Omega$ by this fact, $we \in \mathcal{L}(G)$ (follows from $\xi(x_0, w) = \xi(z, s_z)$ and $(\xi(z, s_z), e) \in \Omega$), $w'e \in \mathcal{L}(G)$ (follows from $\xi(x_0, w') = \xi(z, s')$ and $(\xi(z, s'), e) \in TR(G)$), $(\xi(z, s_z), e) \in \Omega$ and Ω satisfies (4).

That $x^i T_\Omega x^{i-1}$ and definition of T_Ω implies that there exists $w, w' \in \mathcal{L}(G)$ where $\xi(x_0, w) = x^{i-1}$, $\xi(x_0, w') = x^i$ and $\theta^{\omega^1_\Omega}(w) = \theta^{\omega^1_\Omega}(w')$.
Since $(\xi(x^{i-1}, s_{x^{i-1}}), e)$, $(\xi(x^i, s_{x^i}), e) \in \Omega$, $\theta^{\omega^1_\Omega}(ws_{x^{i-1}}e) = \theta^{\omega^1_\Omega}(ws_{x^{i-1}})e$ and $\theta^{\omega^1_\Omega}(w's_{x^i}e) = \theta^{\omega^1_\Omega}(w's_{x^i})e$. Furthermore, $\theta^{\omega^1_\Omega}(ws_{x^{i-1}}) = \theta^{\omega^1_\Omega}(w)$ since $s_{x^{i-1}} \in S^{\mathcal{A}}_{x^{i-1}}$ and $\theta^{\omega^1_\Omega}(w's_{x^i}) = \theta^{\omega^1_\Omega}(w')$ since $s_{x^i} \in S^{\mathcal{A}}_{x^i}$. By these five facts it follows that $\theta^{\omega^1_\Omega}(ws_{x^{i-1}}e) = \theta^{\omega^1_\Omega}(w's_{x^i}e)$ and definition of T_Ω. It follows that $\xi(\xi(x^i, s_{x^i}), e) T_\Omega \xi(\xi(x^{i-1}, s_{x^{i-1}}), e)$ by this fact, $\xi(x_0, w) = x^{i-1}$ and $\xi(x_0, w') =$

x^i. Similarly, $\xi(\xi(z,s_z),e)T_\Omega\xi(\xi(x^n,s_n),e)$. It follows that $\xi(\xi(z,s'),e)T_\Omega\xi(\xi(z,s_z),e)$ since $s_z,s' \in S_z^{\not\emptyset}$ implies $\exists w,w' \in \mathcal{L}(G)$, $\xi(x_0,w) = \xi(z,s_z)$, $\xi(x_0,w') = \xi(z,s')$ and $\theta^{\omega_\Omega^1}(w) = \theta^{\omega_\Omega^1}(w')$, and $(\xi(z,s_z),e)$, $(\xi(z,s'),e) \in \Omega$ furthermore implies $\theta^{\omega_\Omega^1}(we) = \theta^{\omega_\Omega^1}(w'e)$. By these results and (12) the following holds:

$$\xi(\xi(x^1,s_{x^1}),e)T_\Omega\xi(\xi(x^0,s),e) \wedge \xi(\xi(x^2,s_{x^2}),e)T_\Omega\xi(\xi(x^1,s_{x^1}),e)$$
$$\wedge \ldots \wedge \xi(\xi(z,s_z),e)T_\Omega\xi(\xi(x^n,s_{x^n}),e) \wedge \xi(\xi(z,s'),e)T_\Omega\xi(\xi(z,s_z),e).$$

Then $\xi(\xi(z,s'),e) \in [\xi(\xi(x,s),e)]$ by this fact, $x = x^0$ and (5). It follows that $[\xi(\xi(x,s),e)] = [\xi(\xi(z,s'),e)]$. □

Before we present the main result of this section we require the following. Given $s \in \mathcal{L}(G)$, we denote the set of state equivalence classes (with some slight abuse of notation) the agent believes the system could be in on observation of $\theta^{\omega_\Omega^1}(s) = \sigma_1\sigma_2\ldots\sigma_n$ when only state equivalence classes are kept track of by $[\theta^{\omega_\Omega^1}(s)]$. This set of state equivalence classes is defined below:

$$[\theta^{\omega_\Omega^1}(s)] = \{\ [x] \mid \exists q_0,q_1,q_2,\ldots,q_n \in X, x_0 \in [q_0], [x] = [q_n], \quad \forall i \in \{0,\ldots,n-1\}, \xi(q_i,\sigma_{i+1}) \in [q_{i+1}]\ \} \tag{13}$$

We aim to prove that for any $s \in \mathcal{L}(G)$, $[\theta^{\omega_\Omega^1}(s)]$ is a singleton set in the context of the sensor activation maps considered in this section where if the sensor for an event is active then an occurrence of the event is encountered after some unobserved sequence of events. For sensor activation maps where this does not hold, $[\theta^{\omega_\Omega^1}(s)]$ is not necessarily a singleton set. This is explored further in the next section.

Theorem 2. *Given $G = (X,\Sigma,\xi,x_0)$, policy $\Omega \subseteq TR(G)$ which satisfies* (4) *and* (11), $\forall s \in \mathcal{L}(G), \forall [x] \in [\theta^{\omega_\Omega^1}(s)], [x] = [\xi(x_0,s)]$.

Proof. This proof follows by induction on the length of $\theta^{\omega_\Omega^1}(s)$.

When $|\theta^{\omega_\Omega^1}(s)| = 0$, $\theta^{\omega_\Omega^1}(s) = \varepsilon$. It holds that $[\varepsilon] = \{\ [x] \mid \exists q_0 \in X, x_0 \in [q_0], [x] = [q_0]\}$ by (13). So, $\forall [x] \in [\varepsilon]$, $x_0 \in [x]$. It follows that $[\varepsilon] = \{[x_0]\}$ and so $[\theta^{\omega_\Omega^1}(s)] = \{[x_0]\}$. The fact that $\theta^{\omega_\Omega^1}(\varepsilon) = \varepsilon$ and $\theta^{\omega_\Omega^1}(s) = \varepsilon$ implies $\theta^{\omega_\Omega^1}(\varepsilon) = \theta^{\omega_\Omega^1}(s)$. Then $\xi(x_0,s)T_\Omega x_0$ by this fact, $\xi(x_0,\varepsilon) = x_0$ and definition of T_Ω. Then $\xi(x_0,s) \in [x_0]$ by this fact and (5). Thus, $\forall [x] \in [\theta^{\omega_\Omega^1}(s)], [x] = [\xi(x_0,s)]$.

Now for the inductive step. Consider when $|\theta^{\omega_\Omega^1}(s)| = n+1$. Let $s = s_1es_2$ where $s_1,s_2 \in \Sigma^*$, $e \in \Sigma_o$, and s_1e is the shortest prefix of s where $\theta^{\omega_\Omega^1}(s_1e) = \theta^{\omega_\Omega^1}(s_1)e = \theta^{\omega_\Omega^1}(s)$. Since $|\theta^{\omega_\Omega^1}(s_1)| = n$ and by the inductive hypothesis it follows that $\forall [x] \in [\theta^{\omega_\Omega^1}(s_1)]$, $[x] = [\xi(x_0,s_1)]$. Since $\theta^{\omega_\Omega^1}(s_1e) = \theta^{\omega_\Omega^1}(s_1)e$ it must be that $(\xi(x_0,s_1),e) \in \Omega$. It follows that $\forall [x] \in [\theta^{\omega_\Omega^1}(s_1)]$, $\forall q \in [x]$, $\forall s_q \in S_q^{\not\emptyset}$, if $(\xi(q,s_q),e) \in TR(G)$ then $[\xi(\xi(q,s_q),e)] = [\xi(\xi(x_0,s_1),e)]$ by this fact, Ω satisfies (4) and (11), $e \in \Sigma_o$, $[q] = [x] = [\xi(x_0,s_1)]$, and **Lemma 5**. Then $\forall [x] \in [\theta^{\omega_\Omega^1}(s_1e)]$, $[x] = [\xi(x_0,s_1e)]$.

It holds that $\xi(x_0,s_1es_2)T_\Omega\xi(x_0,s_1e)$ since $\theta^{\omega_\Omega^1}(s_1es_2) = \theta^{\omega_\Omega^1}(s_1e)$ and by definition of T_Ω. So $[\xi(x_0,s_1es_2)] = [\xi(x_0,s_1e)]$. Then $\forall [x] \in [\theta^{\omega_\Omega^1}(s_1es_2)]$, $[x] = [\xi(x_0,s_1es_2)]$ by this fact, $\theta^{\omega_\Omega^1}(s_1es_2) = \theta^{\omega_\Omega^1}(s_1e)$ and the previously established fact that $\forall [x] \in [\theta^{\omega_\Omega^1}(s_1e)]$, $[x] = [\xi(x_0,s_1e)]$. □

For the policies of Section 3 it sufficed to base sensor activation decisions on the state reached by the sequence of events observed by the agent from the initial state of G. However, for the policies considered in this section, it is not difficult to see that the sequence of events observed may not be a string in $\mathcal{L}(G)$ (e.g., consider plant and policy of Figure 3 where instead an unobserved event e_{uo} labels the transition from x_1 to x_4 instead of e_2). Instead, according to **Theorem 2**, it suffices for an agent to keep track of the equivalence class of the true state of G for determining sensor activation decisions. Tracking the state equivalence class of the true state of G can be implemented effectively. It will be demonstrated that the set of state equivalence classes of X, denoted by $[X]$ (with a slight abuse of notation), and a map, $\xi_{[G]} : [X] \times \Sigma_o \to [X]$ from a state equivalence class and observed event to the state equivalence class encountered on observation of the event can be computed polynomially in $|X|$ and $|\Sigma|$. This allows the following procedure for computing sensor activation decisions. The initial sensor activation decisions are based on any state in $[x_0]$. From $[x_0]$, if an event e_1 whose sensor is active is observed then the active state equivalence class where sensor activation decisions are made is set to $\xi_{[G]}([x], e_1)$. Sensors are updated according to any state in $\xi_{[G]}([x], e_1)$. Afterwards, if an event e_2 whose sensor is active is observed then the active state equivalence class where sensor activation decisions are made is set to $\xi_{[G]}([x_0], e_1e_2)$. The active state equivalence class from which sensor activation decisions are made is updated in this manner as further event occurrences are observed.

In the remainder of this section we describe a procedure for computing the transition function, $\xi_{[G]}$, of $[G]$. First we describe a procedure for computing $[X]$. From X and T_Ω we construct an undirected graph (X, T_Ω). Associated with each vertex in X is a flag indicating whether or not the vertex has been visited. Initially, for all $x \in X$, the flag of x is assigned the value 0. For each $x \in X$ whose flag is equal to 0 we conduct a depth-first search. When a vertex is visited we assign its flag to the value 1. It is not difficult to see that any vertex visited in the search belongs to $[x]$ and, furthermore, when the search terminates the set of vertices traversed is equal to $[x]$. It is not difficult to see that this procedure is in $\Theta(|X| + |T_\Omega|)$ due to the well-known tight asymptotic bounds on depth-first search. In the worst-case, T_Ω contains an edge between any two vertices in X (i.e., every state in X is indistinguishable with every other state in X). In this case $|T_\Omega| = \frac{|X| \cdot |X+1|}{2}$ and so the procedure is in $\Theta(|X|^2)$ for this case.

After computing $[X]$ we can construct DFA $[G] = ([X], \Sigma_o, \xi_{[G]}, [x_0])$. Transition function $\xi_{[G]}$ is defined as follows:

$$\forall [x] \in [X], \forall e \in \Sigma_o, \xi_{[G]}([x], e) = [\xi(x, e)] \text{ if } \exists x \in [x] \text{ where } (x, e) \in \Omega.$$

Constructing $\xi_{[G]}$ also requires a map from each state $x \in X$ to its corresponding state class. Given $[X]$ this can be computed in $O(|X|)$. Then computing $\xi_{[G]}$ is in $O(|[X]| \cdot |\Sigma_o|) \subseteq O(|X| \cdot |\Sigma_o|)$ as any two states in the same state equivalence class are followed by the same sensor activation decisions.

To reaffirm, given G and policy $\Omega \subseteq TR(G)$ which satisfies (4) and (11) and where sensor activation decisions are determined by ω^2_Ω derived from Ω by (9), if event sequence s is generated by G then s is observed as $\theta^{\omega^2_\Omega}(s) = \theta^{\omega^1_\Omega}(s) \in \Sigma_o^*$ and the sensor activation decisions following s are $\omega^2_\Omega(s) = \omega^1_\Omega(s)$ which is equal to the set of events labeling transitions in $[G]$ from state $\xi_{[G]}([x_0], \theta^{\omega^1_\Omega}(s))$.

5 Using state equivalence classes to determine sensor activation decisions

In this section we continue investigation of a state's equivalence class defined in (5) and the true state's equivalence class estimate defined in (13) (given s generated by G and the observation of s denoted by $\theta^{\omega_\Omega^1}(s)$, $[\theta^{\omega_\Omega^1}(s)]$ returns a set of state equivalence classes which contains $[\xi(x_0, s)]$, the true state's equivalence class). In Subsection 5.1 we illustrate that, in general, verifying whether or not sensor activation decisions can be made based on state equivalence classes rather than state or string estimates is PSPACE-complete. When sensor activation decisions can be made based on state equivalence classes, in Subsection 5.2 we demonstrate a condition that, when satisfied, allows sensor activation decisions to be computed using an arbitrary state equivalence class in the estimate of the true state's equivalence class.

Note that when we state that sensor activation decisions are to be computed using a state equivalence class, we mean that an event sensor is activated if it is activated following some state in the state equivalence class and is deactivated otherwise (i.e., the set of sensors to be activated are the union of all sensor activations following all states in the state equivalence class).

Below we mention some differences between this section and Section 4 with respect to the policies considered.

In Section 4 we considered that policies satisfy (4) and (11). That these two conditions are satisfied implies that any two states in a state equivalence class must have exactly the same sensor activation decisions as we saw in Section 4. It is not required that condition (11) be satisfied by the policies considered in this section. Any policy where there is no ambiguity in sensor activation decisions when state / string estimates are maintained must satisfy condition (4). So the policies that we consider must still satisfy (4). Even so, it is not necessarily the case that any two states in a state equivalence class have exactly the same sensor activations decisions and so it may be the case that sensor activation decisions can not be based on state equivalence classes. We make further remarks on this topic below.

In Section 4 we considered that if a sensor for an event e is active then there exists some unobserved trace following activation of e's sensor after which e occurs. This resulted in the true state equivalence class estimate ($[\theta^{\omega_\Omega^1}(s)]$ where s is the string generated by G) being a singleton set. In this section we relax this restriction: though an event e's sensor is active, there may not exist any unobserved trace following its activation after which e occurs. Instead, for instance, the occurrence of some other event whose sensor is active may occur before e is generated by G. For these types of sensor activation maps, the true state equivalence class estimate may not be a singleton set. When this occurs it is not necessarily the case that an agent can base its sensor activation decisions on state equivalence class estimates rather than the state estimate provided by its observer automaton.

There are three scenarios that may be encountered which prevent basing sensor activation decisions on state equivalence class estimates.

In the first scenario, sensor activation conflicts may exist within a state equivalence class. That is, one of the state equivalence classes $[x] \in [X]$ contains two states $x_1, x_2 \in [x]$ such that for some event $e \in \Sigma_o$, $(x_1, e) \in TR(G)$,

$(x_2, e) \in TR(G)$, $(x_1, e) \in \Omega$ but $(x_2, e) \notin \Omega$ where Ω is the policy used. Verification of whether or not this scenario occurs is easy. For each state equivalence class, compute the set of events whose sensors are active following any state in the equivalence class and the set of events which occur after a state in the equivalence class but whose sensors are deactivated after those states. For a given state equivalence class, if the intersection of these two sets is nonempty then a sensor activation conflict exists for that state equivalence class. It is easy to see that this procedure is in $O(|[X]| \cdot |X| \cdot |\Sigma| + |[X]| \cdot (|\Sigma| \cdot \log(|\Sigma|) + |\Sigma|)) \subseteq O(|X|^2 \cdot |\Sigma| + |X| \cdot |\Sigma| \cdot \log(|\Sigma|))$ if a total order is imposed on Σ.

When the first scenario does not occur, sensor activation conflicts do not exist in a state equivalence class. However, it is possible that the set of sensor activation decisions computed from a given state equivalence class might result in sensors being activated that would not otherwise be activated if sensor activation decisions were computed from a state estimate. This is the second scenario. We explore this issue in Subsection 5.1. There we prove that determining if using a state equivalence class results in more sensors being activated than using a state estimate is PSPACE-complete.

When neither the first nor second scenarios occur it may still be the case that a third scenario occurs: sensor activation conflicts may exist between state equivalence classes. That is, for policy Ω, it may be the case that, for some string $s \in \mathcal{L}(G)$, $[\theta^{\omega_\Omega^1}(s)]$ contains two state equivalence classes $[x_1], [x_2]$ where $[x_1]$ contains a state x_1 such that for some event $e \in \Sigma_o$, $(x_1, e) \in \Omega$ but there does not exist a state $x_2 \in [x_2]$ where $(x_2, e) \in \Omega$. In Subsection 5.2 we provide a procedure for verifying whether or not the third scenario occurs. If none of the three scenarios occurs then sensor activation decisions can be computed using an arbitrary state equivalence class in the true state's equivalence class estimate.

Let us reflect on the policies considered in Section 4. In the policies considered there the first, second and third scenario do not occur. There it was proven that if the policies considered satisfy (4) and (11) then no sensor activation conflicts exist between any two states in a state equivalence class (see proof of **Lemma 5**). Also, for such policies, the second scenario does not occur by definition of (11). That these facts hold and the fact that there is only ever at most one state equivalence class in the true state equivalence class' estimate (see **Theorem 2**) implies that the third scenario does not occur for policies satisfying (4) and (11).

For simplicity, in this section we consider that the sensor activations following a string $s \in \mathcal{L}(G)$ are defined by (2), not by (9). We could consider that decisions are defined by (9) instead but when computing the sensor activation decisions of a state equivalence class, which is the union of all sensor activations following all states in the state equivalence class, the same set of activations will be computed as if decisions are defined by (2). The reason for this is that any state x' in the unobserved reach of a state x belongs to $[x]$ and so, in addition to sensors active at state x being included in the set of sensor activations of $[x]$, sensors active at state x' will also be included in the set of sensor activations of $[x]$.

5.1 Determining when sensor activation decisions can be based on a state equivalence class

In Section 4 we proved that we can use a state's equivalence class to determine sensor activation decisions when policies satisfy (4) and (11). For such policies, the sensor activation decisions following any two states in a state equivalence class are equal. Furthermore, if an event sensor is active at a given state then the event occurs following some unobserved transition sequence from the state. In this subsection we consider policies where activating an event's sensor does not necessarily imply that the event will occur following some unobserved transition sequence. For such policies it may be the case that activating sensors using a state's equivalence class results in more sensors being active than is necessary. This is the second scenario described in the introduction to this section. Note that in this subsection we presume that, for the policies considered, the first scenario does not occur.

For instance, suppose that, for a given $G = (X, \Sigma_o \cup \Sigma_{uo}, \xi, x_0)$ and policy $\Omega \subseteq TR(G)$, there exists a state $x \in X$ where event $e \in \Sigma_o$ does not occur following any string of unobserved transitions from x. Also, suppose that x is indistinguishable with some set of states Q where $QT_\Omega x$ (which, by definition, are also in x's state equivalence class) where, following any state in Q, the sensor for e is active. That x is indistinguishable with states in Q does not imply that, following any string leading to x, there exists an observationally-equivalent string leading to a state in Q. That is, following some strings leading to x, x might be distinguished from any state in Q. Then it may not be the case that the sensor for e needs to be activated at x. That is, the state estimate returned by the subset construction may not contain any states where the sensor for e is active. In such a situation the sensor for e does not need to be activated. However, the sensor for e is active if sensor activation decisions are computed from the state equivalence class.

So, given plant G and policy Ω, we would like to be able to verify if such a situation occurs. Formally, we consider the following problem:

Problem 1. *Given a DFA $G = (X, \Sigma, \xi, x_0)$, policy $\Omega \subseteq TR(G)$, state $x \in X$ and a set of states Q where $QT_\Omega x$, determine if there exists a string $s' \in \mathcal{L}(G)$ such that $x = \xi(x_0, s')$ and $\forall s \in \mathcal{L}(G)$, $\xi(x_0, s) \in Q \Rightarrow \theta^{\omega_\Omega^1}(s) \neq \theta^{\omega_\Omega^1}(s')$.*

By definition, the situation described previously occurs when an algorithm for solving **Problem 1** returns true for the given G, Ω, x and where Q is taken to be the largest set in X where $QT_\Omega x$ and for all $q \in Q$ the sensor for event e is activated following q.

However, verifying if the situation occurs is not trivial. We find that deciding **Problem 1** is *PSPACE-complete*. The proof of this follows in the remainder of this subsection.

In order to prove PSPACE-completeness we provide an algorithm in PSPACE for deciding **Problem 1** and show that a related problem, **Problem 2**, which we prove to be PSPACE-complete, is polynomial-time reducible to **Problem 1**. **Problem 2** is defined below.

Problem 2. *Given an NFA $N = (X, \Sigma, \xi, x_0)$, state $x \in X$ and a set of states $Q \subseteq X \setminus \{x\}$, where for any $q \in Q$ there exists a string $s \in \mathcal{L}(N)$ such that $\{x, q\} \subseteq \xi(x_0, s)$, determine if there exists a string $s' \in \mathcal{L}(N)$ such that $x \in \xi(x_0, s')$ and $Q \cap \xi(x_0, s') = \varnothing$.*

Next we provide an algorithm in PSPACE for solving **Problem 2** and provide a polynomial-time reduction to **Problem 2** from the problem of determining whether two NFA recognize the same language which is known to be PSPACE-complete [10].

Lemma 6. ***Problem 2*** *is PSPACE-complete.*

Proof. **Problem 2** is in PSPACE: We provide **Algorithm 1** which solves **Problem 2** in linear space. Specifically, this algorithm solves a generalization of **Problem 2** where we do not require that for any $q \in Q$ there exists a string $s \in \mathcal{L}(N)$ such that $\{x, q\} \in \xi(x_0, s)$.

We describe informally the operation of the algorithm. First we start with the ε-reach of the initial state, x_0. Denote this by X'. We determine if $x \in X'$ and if $Q \cap X' = \varnothing$. If so then there exists a string leading to x that does not lead to any state in Q and the algorithm terminates with "accept". Otherwise, we nondeterministically choose a letter $e \in \Sigma \setminus \{\varepsilon\}$ labelling a transition from a state in X', compute the set X'' of states reached by e followed ε transitions from a state in X', set X' to X'', then repeat the test mentioned. This is done until a certain bound is reached at which point the algorithm terminates with reject. We argue in the following why this bound is used.

When determinizing an NFA $N = (X, \Sigma, \xi, x_0)$ using the subset construction in the worst-case the DFA constructed, denoted by D, will have $2^{|X|} - 1$ states. Consider the shortest string s leading to a state in D containing x and not containing any state in Q. In the worst-case, for each nonempty subset $\hat{X}$ of X that does not contain x or contains x and a state of Q, there exists a prefix of s that leads to $\hat{X}$. Then one can establish a bound on the length of s as follows. The number of nonempty subsets of X containing x (resp., not containing x) is $2^{(|X|-1)}$ (resp., $2^{(|X|-1)} - 1$). The number of nonempty subsets of X containing x and no element of Q is $2^{|X \setminus (Q \cup \{x\})|} = 2^{(|X \setminus Q|-1)}$. Thus the number of subsets of X containing x and an element of Q is $2^{(|X|-1)} - 2^{(|X \setminus Q|-1)}$. Thus a bound on the length of s is $\left([2^{(|X|-1)} - 1] + [2^{(|X|-1)} - 2^{(|X \setminus Q|-1)}] - 1\right) + 1 = 2^{|X|} - (2^{(|X \setminus Q|-1)} + 1)$.

Next we characterize the space complexity of **Algorithm 1**. Note that the bounds we provide are not necessarily tight.

First, we evaluate the storage requirements for variables used in the algorithm. Each of x', X', X'' requires $O(|X|)$ tape cells for storage. Variable i requires $\lceil \log(2^{|X|} - (2^{(|X \setminus Q|-1)} + 1)) \rceil \leq \lceil \log(2^{|X|}) \rceil = c \cdot |X| \in O(|X|)$ tape cells for storage where $c > 0$. Variable e requires $O(|\Sigma|)$ tape cells for storage.

Second, we evaluate the storage requirements for operations involving variables used in the algorithm. Computing X' in line 1 requires $O(|X|)$ tape cells since computing the ε-reach of a state in X requires $O(|X|)$ tape cells. The assignment to i on line 2 requires $O(1)$ tape cells. The comparisons in the first clause of the loop invariant on line 3 requires $O(|X|)$ tape cells. The comparison in the second clause of the loop invariant on line 3 requires $O(|X|)$ tape cells. The assignment to e on line 4 requires $O(|X| + |\Sigma|)$ tape cells. Iterating over X' on line 6 requires $O(|X|)$ tape cells. Evaluating the condition of line 7 requires $O(1)$ tape cells. The assignment on line 8 requires $O(|X|)$ tape cells. The assignment on line 11 requires $O(|X|)$ tape cells. The incrementing of i on line 12 can be done in place on i if i is represented appropriately and hence requires $O(1)$ tape cells. The comparisons on line 14 require $O(|X|)$ tape cells.

All of the storage requirements mentioned are further upper bounded by the size of the input to **Algorithm 1**. Thus, **Problem 2** is in NPSPACE. By

Algorithm 1 A decider for **Problem 2**

Require: $N = (X, \Sigma, \xi, x_0)$
Require: $x \in X$
Require: $Q \subseteq X$
1: $X' \leftarrow \varepsilon\text{-reach}_\xi(x_0)$
2: $i \leftarrow 0$
3: **while** $(x \notin X' \vee Q \cap X' \neq \varnothing) \wedge i < 2^{|X|} - (2^{(|X \setminus Q|-1)} + 1)$ **do**
4: Nondeterministically select e from $\Sigma \setminus \{\varepsilon\}$ where $\exists x' \in X', \xi(x', e)!$.
5: $X'' \leftarrow \varnothing$
6: **for all** $x' \in X'$ **do**
7: **if** $\xi(x', e)!$ **then**
8: $X'' \leftarrow X'' \cup \varepsilon\text{-reach}_\xi(\xi(x', e))$
9: **end if**
10: **end for**
11: $X' \leftarrow X''$
12: $i \leftarrow i + 1$
13: **end while**
14: **if** $x \in X' \wedge Q \cap X' == \varnothing$ **then**
15: *accept*
16: **else**
17: *reject*
18: **end if**

Savitch's Theorem [8], **Problem 2** is also in PSPACE.

Every problem in PSPACE is polynomial-time reducible to **Problem 2**:
Consider two NFA $N_1 = (X^{N_1}, \Sigma^{N_1}, x_0^{N_1}, \xi^{N_1}, F^{N_1})$ and
$N_2 = (X^{N_2}, \Sigma^{N_2}, x_0^{N_2}, \xi^{N_2}, F^{N_2})$.

From N_1 construct NFA $N_1' = (X^{N_1} \cup \{x_F^{N_1'}\}, \Sigma^{N_1}, x_0^{N_1}, \xi^{N_1'}, x_F^{N_1'})$ where $x_F^{N_1'}$ is not in X^{N_1}. Transition function $\xi^{N_1'}$ is defined the same as ξ^{N_1} but where $\xi^{N_1'}(x_f, \varepsilon) = \xi^{N_1}(x_f, \varepsilon) \cup \{x_F^{N_1'}\}$ for each $x_f \in F^{N_1}$. Symmetrically, from N_2 we construct NFA $N_2' = (X^{N_2} \cup \{x_F^{N_2'}\}, \Sigma^{N_2}, x_0^{N_2}, \xi^{N_2'}, x_F^{N_2'})$. It is easy to see that $\mathcal{L}(N_1') = \mathcal{L}(N_1)$ (resp., $\mathcal{L}(N_2') = \mathcal{L}(N_2)$).

From N_1' and N_2' we construct NFA $N = (X^{N_1} \cup \{x_F^{N_1'}\} \cup X^{N_2} \cup \{x_F^{N_2'}\} \cup \{x_0^N\}, \Sigma^{N_1} \cup \Sigma^{N_2} \cup \{\alpha\}, x_0^N, \xi^N)$ which recognizes $\mathcal{L}(N_1') \cup \mathcal{L}(N_2') \cup \{\alpha\}$. Without loss of generality, suppose the state sets whose union defines the state-space of N are pairwise disjoint. Also, suppose $\alpha \notin \Sigma^{N_1} \cup \Sigma^{N_2}$. Transition function ξ^N is defined in the following cases and undefined otherwise:

$$\forall e \in \Sigma_{N_1'} \cup \{\varepsilon\}, \forall q \in X^{N_1} \cup \{x_F^{N_1'}\}, \xi^N(q, e) = \xi^{N_1'}(q, e)$$
$$\forall e \in \Sigma_{N_2'} \cup \{\varepsilon\}, \forall q \in X^{N_2} \cup \{x_F^{N_2'}\}, \xi^N(q, e) = \xi^{N_2'}(q, e)$$
$$\xi^N(x_0^N, \varepsilon) = \xi^{N_1'}(x_0^{N_1}, \varepsilon) \cup \xi^{N_2'}(x_0^{N_2}, \varepsilon)$$
$$\xi^N(x_0^N, \alpha) = \{x_F^{N_1'}, x_F^{N_2'}\}$$

Let $x = x_F^{N_1'}$ and $Q = \{x_F^{N_2'}\}$. Determine if $s' \in \mathcal{L}(N)$ of **Problem 2** exists for the constructed NFA N, x and Q. If so, it is easy to see that $\mathcal{L}(N_1) \neq \mathcal{L}(N_2)$.

Otherwise, let $x = x_F^{N_2'}$ and $Q = \{x_F^{N_1'}\}$. Determine if $s' \in \mathcal{L}(N)$ of **Problem 2** exists for the constructed NFA N, x and Q. If so, it is easy to see that $\mathcal{L}(N_1) \neq \mathcal{L}(N_2)$. Otherwise, it is easy to see that $\mathcal{L}(N_1) = \mathcal{L}(N_2)$.

Constructing NFA N from N_1 and N_2 requires polynomial time as we are only adding 3 states and $|F^{N_1}| + |F^{N_2}| + 3$ transitions and determining where each transition is to be added requires constant time (i.e., F^{N_1}, F^{N_2} and x_0^N, the states from which transitions are to be added, are provided directly in the definition of N_1, N_2 or are introduced in the construction). □

From the previous result we achieve the following.

Theorem 3. ***Problem 1*** *is PSPACE-complete.*

Proof. **Problem 1** is in PSPACE: **Algorithm 1** can be adapted to solve **Problem 1** in linear space with few modifications. The modifications are described:

1. In place of ε-reach$_\xi$ we use the unobserved reach using policy Ω, which is denoted as $UR_{\not\emptyset}$-reach and defined as follows:

   $$UR_{\not\emptyset}\text{-reach}(\hat{x}) = \{\hat{x}\} \cup \{\breve{x} \in UR_{\not\emptyset}\text{-reach}(\overline{x}) \mid \exists e \in \Sigma, \overline{x} = \xi(\hat{x}, e) \wedge (\hat{x}, e) \notin \Omega\}$$

2. In line 4, instead of nondeterministically selecting e from $\Sigma \setminus \{\varepsilon\}$ where $\exists x' \in X', \xi(x', e)!$, we nondeterministically select e from Σ where $\exists x' \in X', \xi(x', e)! \wedge (x', e) \in \Omega$.

3. In line 7, instead of using $\xi(x', e)!$ as the condition we use $\xi(x', e)! \wedge (x', e) \in \Omega$

Note that the bounds we provide next are not necessarily tight. A depth-first search algorithm can be used to compute $UR_{\not\emptyset}$-reach(x) for a given $x \in X$ using $O(|X|)$ tape cells. The truth of the modified conditions of lines 4 and 7 can be tested using $O(|X| \cdot |\Sigma|)$ tape cells.

The storage requirements of this algorithm (including those of **Algorithm 1**) are further bounded above by the size of the input to **Problem 1**. Thus, **Problem 1** is in NPSPACE. By Savitch's Theorem [8], **Problem 1** is also in PSPACE.

Every problem in PSPACE is polynomial-time reducible to **Problem 1**:
Consider NFA $N = (X_N, \Sigma, \xi_N, x_0)$, $x \in X$ and $Q \subseteq X_N \setminus \{x\}$ where for any $q \in Q$ there exists a string $s \in \mathcal{L}(N)$ such that $\{x, q\} \subseteq \xi_N(x_0, s)$.

From N we construct NFA $N' = (X_N \cup X', \Sigma \cup \{\varepsilon\}, \xi', x_0)$. The NFA N' is to be defined such that it utilizes the state-space of N in addition to some auxiliary states and ϵ-transitions to these states which are defined in such a way that $\mathcal{L}(N') = \mathcal{L}(N)$ and nondeterminism from a state in N' only occurs on ϵ-transitions.

Specifically, transition function ξ' is defined in the same way as ξ_N except for the following cases. For every $e \in \Sigma \setminus \{\varepsilon\}$, for every state $x_N^1 \in X_N$, for every state $x_N^2 \in X_N$, if $x_N^2 \in \xi_N(x_N^1, e)$ and $|\xi_N(x_N^1, e)| > 1$ then a new state $x_{x_1 \to^e x_2}$ is included in X', x_N^2 is not included in $\xi'(x_N^1, e)$, $x_{x_1 \to^e x_2}$ is included in $\xi'(x_N^1, \varepsilon)$ and x_N^2 is included in $\xi'(x_{x_1 \to^e x_2}, e)$.

It is easy to see that $\mathcal{L}(N') = \mathcal{L}(N)$.

From N' we construct DFA $G = (X_N \cup X', \Sigma \cup \Sigma', \xi^G, x_0)$. The DFA G is defined such that any transition labelled by ε in N' is relabelled with a unique symbol $\alpha \in \Sigma'$ where $\alpha \notin \Sigma \cup \{\varepsilon\}$. Furthermore, no two ϵ-transitions from N' are labelled with the same symbol in Σ'. As a result, it is easy to see that G is deterministic.

Specifically, for every $x_1 \in X_N \cup X'$, $\xi^G(x_1, \varepsilon) = \{x_1\}$. Also, let Σ' be defined such that $\Sigma' \cap \Sigma = \varnothing$ and $|\Sigma'| = \sum_{x' \in X_N \cup X'} |\xi'(x', \varepsilon) \setminus \{x'\}|$. For every $x_1 \in X_N \cup X'$, for every $x_2 \in X_N \cup X'$, if $x_2 \in \xi'(x_1, \varepsilon) \setminus \{x_1\}$ then $x_2 \in \xi^G(x_1, \alpha)$ where $\alpha \in \Sigma'$. Furthermore, for every $\alpha \in \Sigma'$, for every $x_1 \in X_N \cup X'$, for every $x_2 \in X_N \cup X'$, $\xi^G(x_1, \alpha)! \cap \xi^G(x_2, \alpha)! \Rightarrow x_1 == x_2$.

Take Σ' to be a set of observable events. Let $TR(G)$ denote the transitions of G. That is, for any event $e \in \Sigma \cup \Sigma'$, for any state $x' \in X_N \cup X'$, if $\xi^G(x', e)!$ then $(x', e) \in TR(G)$. Let policy $\Omega = TR(G) \setminus \{(x', e) \mid x' \in X_N \cup X' \wedge e \in \Sigma'\}$.

For every $q \in Q$, there exists an $s \in \mathcal{L}(N)$ such that $\{x, q\} \subseteq \xi(x_0, s)$. Consider projection $P_\Sigma : (\Sigma \cup \Sigma')^* \to \Sigma^*$. The previous fact and definition of $\mathcal{L}(G)$ imply $P_\Sigma^{-1}(s) \cap \{s' \in \mathcal{L}(G) \mid \xi^G(x_0, s') = x\} \neq \varnothing$ and $P_\Sigma^{-1}(s) \cap \{s' \in \mathcal{L}(G) \mid \xi^G(x_0, s') = q\} \neq \varnothing$. Then, by definition of Ω and ω_Ω^1 from Ω we have that there exists $s', s'' \in \mathcal{L}(G), \xi^G(x_0, s') = x$, $\xi^G(x_0, s'') = q$ and $\theta^{\omega_\Omega^1}(s') = \theta^{\omega_\Omega^1}(s'')$. Then, by definition of T_Ω, $QT_\Omega x$.

Thus, $< G, \Omega, x, Q >$ is an instance of **Problem 1**.

Determine if $s' \in \mathcal{L}(G)$ of **Problem 1** exists for this instance. If s' exists then it is easy to see that $x \in \xi_N(x_0, P_\Sigma(s'))$ and $Q \cap \xi_N(x_0, P_\Sigma(s')) = \varnothing$. Otherwise, it is easy to see there does not exist a string $t \in \mathcal{L}(N)$ such that $x \in \xi_N(x_0, t)$ and $Q \cap \xi_N(x_0, t) = \varnothing$.

Constructing N' from N can be done in polynomial time in the size of X_N and Σ. A simple algorithm for computing N' would proceed by iterating through states in X_N. If an event $e \in \Sigma \setminus \{\varepsilon\}$ labels multiple transitions from the current state $x_N^1 \in X_N$ then we replace every transition to a state x_N^2 on e from x_N^1 by a transition from x_N^1 to a new state $x_{x_1 \to^e x_2}$ labelled by ε followed by a transition from $x_{x_1 \to^e x_2}$ to x_N^2 on e. This algorithm is in $O(|X_N|^2 \cdot |\Sigma|)$ which generates N' containing $O(|X_N|^2 \cdot |\Sigma|)$ states.

Constructing G and Ω from N' can be done in polynomial time in the size of $X_N \cup X'$ and Σ. A simple algorithm for computing G and Ω iterates over the transitions of N'. If a transition is encountered that is labeled with ε then it is relabeled with a new symbol α and the resulting transition is added to Ω. An obvious upper bound for the algorithm is $O(|X_N \cup X'|^2 \cdot |\Sigma \cup \{\varepsilon\}|)$.

As $|X_N \cup X'|$ is in $O(|X_N|^2 \cdot |\Sigma|)$, it follows that the reduction from **Problem 2** to **Problem 1** is in $O(|X_N|^2 \cdot |\Sigma| + (|X_N|^2 \cdot |\Sigma|)^2 \cdot |\Sigma \cup \{\varepsilon\}|) \subseteq O(|X_N|^2 \cdot |\Sigma| + |X_N|^4 \cdot |\Sigma|^2 \cdot (|\Sigma| + 1)) = O(|X_N|^2 \cdot |\Sigma| + |X_N|^4 \cdot |\Sigma|^3 + |X_N|^4 \cdot |\Sigma|) = O(|X_N|^4 \cdot |\Sigma|^3)$. Thus the reduction from **Problem 2** to **Problem 1** is in polynomial time. Since **Problem 2** is PSPACE-complete, every problem in PSPACE is polynomial-time reducible to **Problem 1**. □

5.2 Sensor activation decisions based on an arbitrary state equivalence class

In this subsection we consider that the first and second scenarios described in the introduction to this section do not occur. However, the third scenario may occur. That is, sensor activation conflicts may exist between state equiva-

lence classes. We consider a condition which, if satisfied, avoids such conflicts and allows sensor activation decisions to be computed using an arbitrary state equivalence class in the true state's equivalence class estimate. Also, when the condition is satisfied, maintaining an estimate of the true state's equivalence class is not required. This allows for the method of Section 4 to be applied for computing sensor activation decisions. That is, one only needs to remember a single state equivalence class in the state equivalence class estimate, compute sensor activation decisions from the state equivalence class, then transition to an arbitrary state equivalence class in the new state equivalence class estimate computed as a result of an observed event occurrence.

The condition considered is stated as follows: $\forall s \in \mathcal{L}(G)$, $[\theta^{\omega^1_\Omega}(s)]$ contains only state equivalence classes which are followed by exactly the same sensor activation decisions for any given observable event. Formally, this is expressed as the following:

$$\forall s \in \mathcal{L}(G), \forall \sigma \in \Sigma_o, \forall [q_1], [q_2] \in [\theta^{\omega^1_\Omega}(s)], \quad (14)$$
$$(\exists q_1 \in [q_1], (q_1, \sigma) \in \Omega \Leftrightarrow \exists q_2 \in [q_2], (q_2, \sigma) \in \Omega)$$

We consider the following procedure for verifying if (14) is satisfied for given $G = (X, \Sigma_o \cup \Sigma_{uo}, \xi, x_0)$ and policy Ω. Using the procedure described at the end of Section 4 we compute the set of state equivalence classes $[X]$ of X. After computing $[X]$ we can construct NFA $[G] = ([X], \Sigma_o, \xi_{[G]}, [x_0])$. Transition function $\xi_{[G]}$ is defined as follows:

$$\forall [x] \in [X], \forall e \in \Sigma_o, \xi_{[G]}([x], e) = \{[\xi(x, e)] \mid \exists x \in [x] \text{ where } (x, e) \in \Omega\}.$$

Constructing $\xi_{[G]}$ also requires a map from each state $x \in X$ to its corresponding state class. Given $[X]$ this can be computed in $O(|X|)$. Then computing $\xi_{[G]}$ is in $O(|X| \cdot |\Sigma_o|)$.

From $[G]$ the pairs of indistinguishable state equivalence classes in $[X]$, denoted by $T^{[G]}$, can be computed using the CLUSTER-TABLE algorithm of [14] in $O(|[X]|^2 \cdot |\Sigma_o|) \subseteq O(|X|^2 \cdot |\Sigma_o|)$. It is not difficult to see that a pair of state equivalence classes $([x_1], [x_2]) \in T^{[G]}$ if and only if $\exists s \in \mathcal{L}(G), [x_1], [x_2] \in [\theta^{\omega^1_\Omega}(s)]$. Then (14) is false if $\exists ([x_1], [x_2]) \in T^{[G]}$, $\exists e \in \Sigma_o$, ($\xi_{[G]}([x_1], e)$ is undefined $\wedge \xi_{[G]}([x_2], e)!$) $\vee$ ($\xi_{[G]}([x_1], e)! \wedge \xi_{[G]}([x_2], e)$ is undefined). Otherwise, (14) is true. This can be computed by iterating through pairs in $T^{[G]}$ and checking each pair. This computation is in $O(|[X]|^2 \cdot |\Sigma_o|) \subseteq O(|X|^2 \cdot |\Sigma_o|)$.

Given G and policy $\Omega \subseteq TR(G)$, when (14) is satisfied and where sensor activation decisions are determined by ω^1_Ω derived from Ω by (2), if event sequence s is generated by G then s is observed as $\theta^{\omega^1_\Omega}(s) \in \Sigma_o^*$ the sensor activation decisions are the set of events labeling transitions in $[G]$ from any of the states in $\xi_{[G]}([x_0], \theta^{\omega^1_\Omega}(s))$.

We note that, when $[\theta^{\omega^1_\Omega}(s)] = 1$, it must be the case that the agent's other decisions (e.g., control, communication) can be based on the single state equivalence class in $[\theta^{\omega^1_\Omega}(s)]$. However, for cases where $\exists s \in \mathcal{L}(G)$ such that $[\theta^{\omega^1_\Omega}(s)] > 1$, though (14) may hold it is not necessarily the case that an agent can base its other decisions on an arbitrary state equivalence class in $[\theta^{\omega^1_\Omega}(s)]$. Verification of whether or not other decisions can be based on any state equivalence class in $[\theta^{\omega^1_\Omega}(s)]$ for all $s \in \mathcal{L}(G)$ can be done by checking whether or

not the decisions following two state equivalence classes $[x_1], [x_2]$ are equivalent for all $([x_1], [x_2]) \in T$ where T is computed from $[G]$ as mentioned previously. This verification is in $O(|[X]|^2 \cdot |\Sigma_o|) \subseteq O(|X|^2 \cdot |\Sigma_o|)$.

6 Conclusions

In this paper we considered sensor activation policies which satisfy various notions of feasibility. In Section 3 we considered sensor activation policies which satisfy a very strong notion of feasibility. In Section 4 we considered sensor activation policies which are more general. Finally, we considered when the coarse estimate of the true state of the system can be used for computing a map from observed event sequences to sensor activation decisions in Section 5. For the classes of sensor activation policies considered in each section we demonstrated how a map from observed event sequences to sensor activation decisions can be computed in polynomial time. However, determining if an arbitrary sensor activation policy belongs to the policies considered in Section 5 is PSPACE-complete. For the sensor activation policies considered in this paper, the complexity of determinizing an automaton representation G when transitions not in a given sensor activation policy are replaced with the empty string remains open. Also, investigating other classes of sensor activation policies from which computation of a map from observed event sequences to sensor activation decisions can be done efficiently remains as future work.

References

[1] F. Cassez and S. Tripakis. Fault diagnosis with static and dynamic observers. *Fundamenta Informaticae*, 88(4):497–540, 2008.

[2] E. Dallal and S. Lafortune. Efficient computation of most permissive observers in dynamic sensor activation problems. In *Proceedings of the 2nd International Workshop on Logical Aspects of Fault-Tolerance (LAFT 2011)*, Toronto, ON, Canada, June 2011.

[3] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, Reading, Massachusetts, USA, 1979.

[4] K. Hryniewiecki. Relations of tolerance. *Journal of Formalized Mathematics*, 2, 1990.

[5] X. Juqin, J. Yan, and S. Shaolong. Minimal k-step event observation policy for on-line observability of discrete event systems. In *Proceedings of the 29th Chinese Control Conference*, pages 1476–1482, Beijing, China, July 2010.

[6] F. Lin and W. M. Wonham. On observability of discrete-event systems. *Information Sciences*, 44(3):173–198, 1988.

[7] K. Rudie, S. Lafortune, and F. Lin. Minimal communication in a distributed discrete-event system. *IEEE Transactions on Automatic Control*, 48(6):957–975, June 2003.

[8] W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970.

[9] S. Shu and F. Lin. Detectability of discrete event systems with dynamic event observation. *Systems & Control Letters*, 59(1):9–17, Jan 2010.

[10] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time: Preliminary report. In A. V. Aho, A. Borodin, R. L. Constable, R. W. Floyd, M. A. Harrison, R. M. Karp, and H. R. Strong, editors, *STOC*, pages 1–9. ACM, 1973.

[11] D. Thorsley and D. Teneketzis. Active acquisition of information for diagnosis and supervisory control of discrete event systems. *Discrete Event Dynamic Systems*, 17(4):531–583, December 2007.

[12] W. Wang, C. Gong, and A. R. Girard. Language-based minimization of sensor activation for event diagnosis. In *49th IEEE Conference on Decision and Control*, pages 6734–6739. IEEE, Dec. 2010.

[13] W. Wang, S. Lafortune, A. R. Girard, and F. Lin. Optimal sensor activation for diagnosing discrete event systems. *Automatica*, 46(7):1165–1175, 2010.

[14] W. Wang, S. Lafortune, and F. Lin. An algorithm for calculating indistinguishable states and clusters in finite-state automata with partially observable transitions. *Systems & Control Letters*, 59(9-10):656–661, Sep./Oct. 2007.

[15] W. Wang, S. Lafortune, F. Lin, and A. R. Girard. An online algorithm for minimal sensor activation in discrete event systems. In *Proceedings of the 48th IEEE Conference on Decision and Control, CDC/CCC 2009*, pages 2242–2247, December 2009.

[16] W. Wang, S. Lafortune, F. Lin, and A. R. Girard. Minimization of dynamic sensor activation in discrete event systems for the purpose of control. *IEEE Transactions on Automatic Control*, 55(11):2447–2461, Nov. 2010.