Supervisor Synthesis to Thwart Cyber Attack with Bounded Sensor Reading Alterations

Rong Su [a]

[a] *School of Electrical & Electronic Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798.*

Abstract

One of the major challenges about cyber physical systems is how to protect system integrity from cyber attacks. There has been a large number of different types of attacks discussed in the literature. In this paper we aim to investigate one special type of attacks in the discrete-event system framework, where an attacker can arbitrarily alter sensor readings after intercepting them from a target system, aiming to trick a given supervisor to issue improper control commands, which can drive the system to an undesirable state. We first consider the cyber attack problem from an attacker's point of view, and formulate an attack-with-bounded-sensor-reading-alterations (ABSRA) problem. We then show that the supremal (or least restrictive) ABSRA exists and can be computed, as long as the plant model and the supervisor model are regular, i.e., representable by finite-state automata. Upon the synthesis of the supremal ABSRA, we present a synthesis algorithm, which computes a supervisor that is ABSRA-robust in the sense that any ABSRA will either be detectable or inflict no damage to the system.

Key words: discrete-event systems, supervisory control, cyber security, attack under bounded sensor reading alterations, partial observation, controllability

1 Introduction

A cyber-physical system (CPS) is a mechanism controlled or monitored by computer-based algorithms. Examples of CPS include smart grid, autonomous automobile systems, medical monitoring, process control systems, distributed robotics, and automatic pilot avionics, etc. The connection between the cyber part and the physical part heavily relies on sensor and communication networks, which has been raising a major security concern, as different types of cyber attacks can tamper the data collection processes and interfere safety critical decision making processes, which may cause irreparable damadges to the physical systems being controlled and to people who depend on those systems [19] [20].

There has been a growing number of publications addressing the cyber security issues from both the computer science community, which focuses on the computer computation related issues, and the systems control community, which focuses on issues related to the system dynamics affected by cyber attacks. Recently, more and more efforts have been made in classifying different types of malicious attacks, assuming that the attackers are sufficiently intelligent [13] [14] [21] [22], instead of merely just generating random failures, which is well studied in the fields of reliability and fault tolerant control. Typically, an intelligent attacker requires *system knowledge*, and abilities for *resource disclosure* and *resource disruption* in order to carry out a successful attack, which is covert to a system user until the attacker's goal of causing a damage to the system is achieved. So *covertness* and *damage infliction* are two major characteristics of a successful attack. By analyzing different intelligent cyber attacks, proper countermeasures may be developed to prevent a target system from being harmed by a specific type of attacks.

In this paper we study a special type of data deception attacks in the discrete-event system framework, where an attacker can intercept sensor measurements (or observations) modeled by observable events and alter them arbitrarily but with an upper bound imposed on the length of each altered observation sequence. By sending those altered observation sequences to a given supervisor, whose function is known to the attacker in advance, the attacker can deliberately and covertly guide the system to move into some undesirable states without making any change to the supervisor. The key challenge is how to "fool" the supervisor to make it believe that the system is operating correctly, while using the supervisor's own control functions to carry out the attack, i.e., to lead the system move into a bad state. To this end, we first propose a novel concept of *attackability* and the concept of *attack under bounded sensor reading alterations* (ABSRA), which can be modelled as a finite-state automaton, possessing the properties of covertness, damadge infliction and control feasibility under partial observations. Then we show that the supremal (or least restrictive) ABSRA exists and is computable via a specific synthesis algorithm, as long as both the plant model G and the given supervisor S are finitely representable. Upon this novel ABSRA synthesis algorithm, we present a supervisor synthesis algorithm, which can ensure that a nonempty synthesized supervisor will be "robust" to any ABSRA, in the sense that such an attack will either reveal itself to the supervisor owing to abnormal event

⋆ Corresponding author: Rong Su. Tel. +65 6790-6042. Fax: +65 6793-3318. The support from Singapore Ministry of Education Tier 1 Academic Research Grant M4011221.040 RG84/13 (2013-T1-002-177) and from Singapore National Research Foundation Delta-NTU Corporate Lab Program (SMA-RP2) are gratefully acknowledged.

Email address: rsu@ntu.edu.sg (Rong Su).

executions (so that contingent actions can be taken by the supervisor, which is outside the scope of this paper) or will not be able to bring the system to a bad state (i.e., no damadge will be inflicted).

Our construction of an ABSRA model as a finite-state automaton is inspired by recent works on opacity analysis and enforcement [23] [29] [24] [15], which aim to analyze and/or enforce (via observable event insertions) the capability of a system to prevent a potential attacker from correctly determining the actual state of the system. A comprehensive survey on this subject can be found in [25]. Owing to different objectives of two frameworks, the modeling details and synthesis algorithms are different. There are some works on cyber attack detection and prevention in the discrete-event community [16] [17] [18], mainly from an adaptive fault tolerant control point of view, which heavily rely on real-time fault diagnosis to identify the existence of an attack and then take necessary supervisory control actions. In [26] the authors present a supervisory control approach for a dynamic cyber-security problem that captures progressive attacks to a computer network, which aims to compute an optimal policy in a game theoretical setup. In those works the intelligence of an attacker is not explicitly modeled, and an attack is treated as a fault or an (unintelligent) opponent. As a contrast, we do not rely on real time attack detection, but on prior knowledge of attack models, which assume that an attacker is intelligent to deliver attacks covertly and effectively, as captured by the concept of attackability, and simply build attack-robustness features into a supervisor to ensure that the supervisor will not be affected by any ABSRA unnoticeably. It is this robust control nature distinguishes our works from existing DES-based cyber attack detection and prevention approaches, which fall in the adaptive control domain. Our ABSRA-robust supervisor synthesis bears a slight conceptual similarity to the problem of supervisory control with intermittent sensor failures [30] [31], although the problem setups and solutions are completely different.

The remainder of the paper is organized as follows. In Section II we review the basic concepts and operations of discrete event systems. Then we formulate an ABSRA synthesis problem in Section III, where we show that the supremal ABSRA exists and computable. In Section IV we present an algorithm to synthesize an ABSRA-robust supervisor, and use a toy example to show that the supremal ABSRA-robust supervisor usually does not exist. A simple yet realistic example runs through the entire paper to illustrate all relevant concepts and algorithms. Conclusions are drawn in Section V.

2 An ABSRA problem

In this section we first recall basic concepts used in the Ramadge-Wonham supervisory control paradigm. Then we introduce the concept of ABSRA, followed by a concrete ABSRA synthesis algorithm, which reveals that the supremal ABSRA is computable, as long as both the plant model and the given supervisor are regular.

2.1 Preliminaries on supervisory control

Given a finite alphabet Σ, let Σ^* be the free monoid over Σ with the empty string ϵ being the unit element and the string concatenation being the monoid operation. Given two strings $s, t \in \Sigma^*$, we say s is a *prefix substring* of t, written as $s \leq t$, if there exists $u \in \Sigma^*$ such that $su = t$, where su denotes the concatenation of s and u. Any subset $L \subseteq \Sigma^*$ is called a *language*. The *prefix closure* of L is defined as $\overline{L} = \{s \in \Sigma^* | (\exists t \in L)\, s \leq t\} \subseteq \Sigma^*$. Given two languages $L, L' \subseteq \Sigma^*$, let $LL' := \{ss' \in \Sigma^* | s \in L \wedge s' \in L'\}$ denote the concatenation of two sets. Let $\Sigma' \subseteq \Sigma$. A mapping $P : \Sigma^* \to \Sigma'^*$ is called the *natural projection* with respect to (Σ, Σ'), if

(1) $P(\epsilon) = \epsilon$,

(2) $(\forall \sigma \in \Sigma)\, P(\sigma) := \begin{cases} \sigma & \text{if } \sigma \in \Sigma', \\ \epsilon & \text{otherwise,} \end{cases}$

(3) $(\forall s\sigma \in \Sigma^*)\, P(s\sigma) = P(s)P(\sigma)$.

Given a language $L \subseteq \Sigma^*$, $P(L) := \{P(s) \in \Sigma'^* | s \in L\}$. The inverse image mapping of P is

$$P^{-1} : 2^{\Sigma'^*} \to 2^{\Sigma^*} : L \mapsto P^{-1}(L) := \{s \in \Sigma^* | P(s) \in L\}.$$

A given target plant is modelled as a *deterministic finite-state automaton*, $G = (X, \Sigma, \xi, x_0, X_m)$, where X stands for the state set, Σ for the alphabet, $\xi : X \times \Sigma \to X$ for the (partial) transition function, x_0 for the initial state and $X_m \subseteq X$ for the marker state set. We follow the notation in [11], and use $\xi(x, \sigma)!$ to denote that the transition $\xi(x, \sigma)$ is defined. For each state $x \in X$, let $En_G(x) := \{\sigma \in \Sigma | \xi(x, \sigma)!\}$ be the set of events enabled at x in G. The domain of ξ can be extended to $X \times \Sigma^*$, where $\xi(x, \epsilon) = x$ for all $x \in X$, and $\xi(x, s\sigma) := \xi(\xi(x, s), \sigma)$. The *closed* behavior of G is defined as $L(G) := \{s \in \Sigma^* | \xi(x_0, s)!\}$, and the *marked* behavior of G is $L_m(G) := \{s \in L(G) | \xi(x_0, s) \in X_m\}$. G is *nonblocking* if $\overline{L_m(G)} = L(G)$. We will use $\mathbb{N}$ to denote natural numbers, $|G|$ for the size of its state set, and $|\Sigma|$ for the size of Σ. Given two finite-state automata $G_i = (X_i, \Sigma, \xi_i, x_{i,0}, X_{i,m})$ $(i = 1, 2)$, the *meet* of G_1 and G_2, denoted as $G_1 \wedge G_2$, is a (reachable) finite-state automaton whose alphabet is Σ such that $L(G_1 \wedge G_2) = L(G_1) \cap L(G_2)$ and $L_m(G_1 \wedge G_2) = L_m(G_1) \cap L_m(G_2)$.

We now recall the concept of supervisors. Let $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc} = \Sigma_o \dot{\cup} \Sigma_{uo}$, where Σ_c (Σ_o) and Σ_{uc} (Σ_{uo}) are disjoint, denoting respectively the sets of *controllable* (*observable*) and *uncontrollable* (*unobservable*) events. A *(feasible) supervisory control map of G under partial observation* $P_o : \Sigma^* \to \Sigma_o^*$ is defined as $V : L(G) \to 2^\Sigma$, where

- $(\forall s \in L(G))(\forall \sigma \in \Sigma_{uc})\, s\sigma \in L(G) \Rightarrow \sigma \in V(s)$,
- $(\forall s, s' \in L(G))\, P_o(s) = P_o(s') \Rightarrow V(s) = V(s')$.

For each $s \in L(G)$, $V(s)$ is interpreted as the set of events allowed to be fired after s. Thus, a supervisory control map will not disable any uncontrollable events, and will impose the same control pattern after strings, which cannot be distinguished based on observations. Let V/G denote the closed-loop system of G under supervision of V, i.e.,

- $\epsilon \in L(V/G)$,
- $(\forall s \in L(V/G))(\forall \sigma \in \Sigma) s\sigma \in L(V/G) \iff s\sigma \in L(G) \wedge \sigma \in V(s)$,
- $L_m(V/G) := L_m(G) \cap L(V/G)$.
- $L(V/G) = \overline{L_m(V/G)}$.

The control map V is *finitely representable* if V/G can be described by a finite-state automaton, say $S = (Z, \Sigma, \delta, z_o, Z_m = Z)$, such that

- $L(S \wedge G) = L(V/G)$ and $L_m(S \wedge G) = L_m(V/G)$,
- $(\forall s \in L(S))\, E_S(s) := \{\sigma \in \Sigma | s\sigma \in L(S)\} = V(s)$,

- $(\forall s, s' \in L(S))\, P_o(s) = P_o(s') \Rightarrow \delta(z_0, s) = \delta(z_0, s')$.

The last condition indicates that $V(s) = E_S(s) = E_S(s') = V(s')$ if $P_o(s) = P_o(s')$. Such a supervisor S can be computed by existing synthesis tools such as TCT [27] or SuSyNA [28]. It has been shown that [2], as long as a closed-loop language $K \subseteq L_m(G)$ is *controllable* [4] and *observable* [2], there always exists a finitely-representable supervisory control map V such that $L_m(V/G) = K$ and $L(V/G) = \overline{K}$. From now on we assume that V/G is finitely representable by S, which is called a *supervisor*. For any supervisor S, we can check that $S \wedge G$ is also a supervisor. Thus, without loss of generality, we assume that $L(S) \subseteq L(G)$, upon which we have $L(S \wedge G) = L(S) \cap L(G) = L(S)$.

2.2 A sensor attack model

We assume that *an attacker can intersept each observable event generated by the plant G, and replace it by a sequence of observable events from Σ_o* in order to "fool" the given supervisor S, whose function is known to the attacker. Considering that in practice any event occurrence takes an unnegligible amount of time, it is impossible for an attacker to insert an arbitrarily long observable sequence to replace a received observable event. For this reason, we assume that there exists a known natural number $n \in \mathbb{N}$ such that the length of any observable sequence that the attacker can insert is no more than n. Let $\Delta_n := \{s \in \Sigma_o^* || s| \leq n\}$ be the set of all such bounded observable sequences, where $|s|$ denotes the length of s, and by convention, $|\epsilon| = 0$. We model a sensor attack as a finite state automaton $A = (Y, \Sigma \times \Delta_n, \eta, y_0, Y_m)$, where Y is the state set, Σ the input alphabet, Δ_n the output alphabets, y_0 the initial state, Y_m the marker state set, which is specifically set as $Y_m = Y$, and $\eta : Y \times \Sigma \times \Delta_n \to Y$ the (partial) transition map, where for all $\sigma \in \Sigma_{uo}$ and $y \in Y$, $\eta(y, \sigma, \epsilon) = y$, i.e., at each state y all unobservable events are self-looped with ϵ as the output. This is natural because an attacker can only intercept observable events, thus, will not make any move upon unobservable events. We still keep unobservable events here to make it easy for us for subsequent technical development. Clearly, $L(A) = L_m(A) \subseteq (\Sigma \times \Delta_n)^*$. Let $\psi : (\Sigma \times \Delta_n)^* \to \Sigma^*$ and $\theta : (\Sigma \times \Delta_n)^* \to \Delta_n^*$ be the *input* and *output* maps, respectively, where for each $\mu = (\sigma_1, u_1)(\sigma_2, u_2) \cdots (\sigma_l, u_l) \in (\Sigma \times \Delta_n)^*$, $\psi(\mu) = \sigma_1 \sigma_2 \cdots \sigma_l$ and $\theta(\mu) = u_1 u_2 \cdots u_l$.

The basic procedure of an attack is to intercept every single observable event $\sigma \in \Sigma_o$ generated by the plant G, replace it with some observable string $u \in \Delta_n$, and send u to the supervisor S, in order to trick S to issue a control command $\gamma \in \Gamma(S) := \{E_S(s) | s \in L(S)\}$ that may lead to an illegal behaviour $s \in L(G) - L(S)$. This attack procedure is depicted in Figure 1. The sequential composition of the attack A and the supervisor S essentially forms a new supervisor, denoted as $S \circ A$, which receives an observable output $\sigma \in \Sigma_o$ and generates a control command $\gamma \in \Gamma(S)$. The exact definition of this new supervisor reveals the nature of the attack, which is given below. The *sequential composition* of A and S is a deterministic finite state automaton $S \circ A = (Z \times Y \cup \{d\}, \Sigma \times \Delta_n, \zeta, (z_0, y_0), Z \times Y_m \cup \{d\})$, where d is a dump state, and for each $w \in Z \times Y \cup \{d\}$ and $(\sigma, \nu) \in \Sigma \times \Delta_n$,

$$\zeta(w, \sigma, \nu) := \begin{cases} (z', y') & \text{if } w = (z, y),\ \eta(y, \sigma, \nu) = y',\ \delta(z, \nu) = z', \\ d & \text{if } w = (z, y),\ \eta(y, \sigma, \nu) = y',\ \delta(z, \sigma)!, \\ & \text{and } \delta(z, \nu) \text{ undefined}, \\ \text{undefined} & \text{otherwise.} \end{cases}$$

In the definition of ζ all transitions going to the dump state d in $S \circ A$ may potentially reveal the attack, which, for an intelligent attack, should be avoided.

The closed-loop behaviour of (G, S, A) is defined by the following composition

$$G \times (S \circ A) = (X \times (Z \times Y \cup \{d\}), \Sigma \times \Delta_n, \kappa, (x_0, z_0, y_0), X_m \times Z \times Y_m),$$

where for each $(x, w) \in X \times (Z \times Y \cup \{d\})$ and each $(\sigma, \nu) \in \Sigma \times \Delta_n$,

$$\kappa(x, w, \sigma, \nu) := \begin{cases} (x', w') & \text{if } \xi(x, \sigma) = x',\ w' = \zeta(w, \sigma, \nu), \\ \text{undefined} & \text{otherwise.} \end{cases}$$

Note that $\psi(L_m(G \times (S \circ A))) \not\subseteq L_m(S)$, namely the attacker may change the closed-loop behaviour by altering the received observable sequences fed to S.

To illustrate the aforementioned concepts, let us go through a simple single-tank example depicted in Figure 2, which consists of a water supply whose supply rate is q_i, a tank, and a control valve at the bottom of the tank controlling the outgoing flow rate q_o, whose value depends on the valve opening and the water level h. We assume that the valve can only be fully open or fully closed. The water level h can be measured, whose value can trigger some predefined events, denoting the water levels: *low* (h=L), *high* (h=H), and *extremely high* (h=EH). We construct a simple discrete-event model of the system depicted in Figure 3, where the alphabet Σ

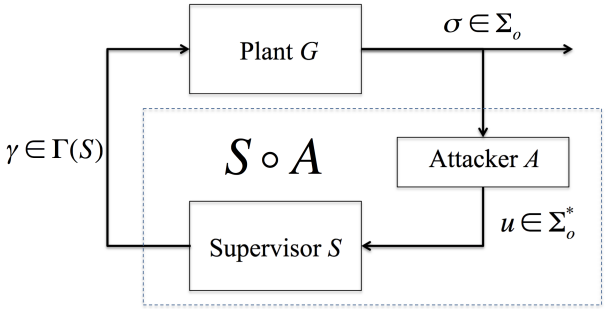

Fig. 1. The block diagram of a plant under attack

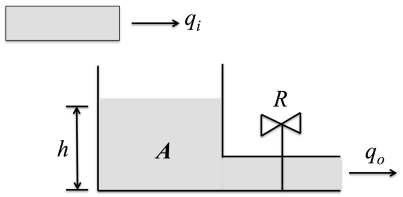

Fig. 2. A single tank system

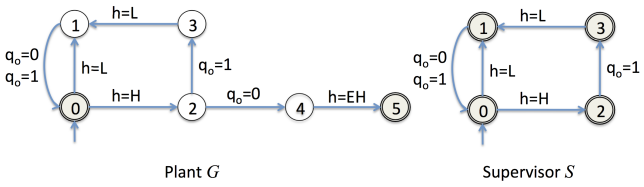

Fig. 3. Models of the plant G and the supervisor S

contains all events shown in the figure. All events are observable, i.e., $\Sigma_o = \Sigma$. Only the actions of opening the valve ($q_o = 1$) and closing the valve ($q_o = 0$) are controllable, and all water level events are uncontrollable. In the model we use a shaded double-edge oval to denote a marker state, i.e., state 0 and state 5 in Figure 3. Assume that we do not want the water level to be extremely high, i.e., the event h=EH should not occur. A supervisor S, derivable by using the standard Ramadge-Wonham supervisor synthesis technique, is also depicted in Figure 3. It is clear that the supervisor S opens the valve when the water level is high, i.e., it disables the event $q_o = 0$ at state 2 when the event h=H occurs. Our intuition tells us that if an attack always change events h=H and h=EH to the event h=L, then the supervisor will not prevent the water level from reaching the extreme high level, i.e., the event h=EH will happen. For this reason, we conjecture an attack model A shown in Figure 4, where water levels will be altered to h=L from originally received observations, whereas all other events will remain unchanged. In the picture we

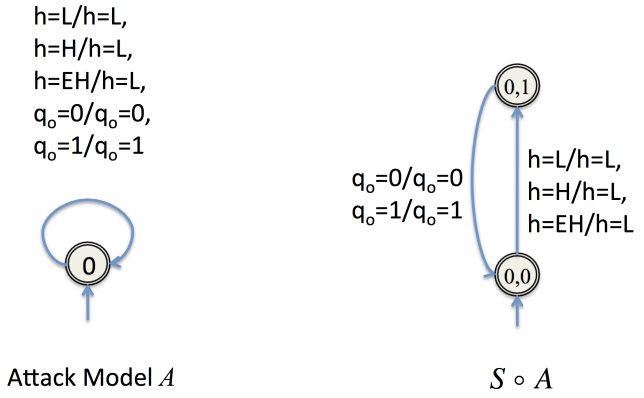

Fig. 4. Automaton models of an attack A (Left) and the sequential composition $S \circ A$ (Right)

use the notation a/b to denote that the concerned event is $(\sigma, u) = (a, b) \in \Sigma \times \Delta_n$, where a is an event actually received from the plant and b is the altered observation, after event a is intercepted, which will be passed to the supervisor S. The sequential composition $S \circ A$ indicates that, no matter which water level is reached, the attack A always sends h=L to the supervisor S, which tricks it to believe that it is safe to allow the valve to be either closed or opened. The impact of A on the closed-loop system (G, S) is depicted in Figure 5, which indicates that, under the influence of attack A, the closed-loop system can produce an illegal behaviour, where the water level can be extremely high, i.e., the event h=EH occurs.

Proposition 1 Given two attacks A_1 and A_2 with the same input and output alphabets Σ and Δ_n, assume that $L(A_1) \subseteq L(A_2)$. Then $L(S \circ A_1) \subseteq L(S \circ A_2)$.

Proof: By the above definition of sequential composition, the proposition follows. ■

Given two attacks A_1 and A_2 with the same input alphabet Σ and output alphabet Δ_n, let $A_1 \cup A_2$ be a deterministic finite-state automaton, denoting the *union* of A_1 and A_2, i.e., $L(A_1 \cup A_2) = L(A_1) \cup L(A_2)$.

Proposition 2 $L(S \circ (\cup_{i \in I} A_i)) = \cup_{i \in I} L(S \circ A_i)$, where I is an index set possibly infinite.

Proof: Since $L(A_i) \subseteq L(\cup_{j \in I} A_j)$ for all $i \in I$, by Prop. 1 we have $L(S \circ (\cup_{i \in I} A_i)) \supseteq \cup_{i \in I} L(S \circ A_i)$. To show the other direction, for each string $\mu = (\sigma_1, u_1) \cdots (\sigma_n, u_n) \in L(S \circ (\cup_{i \in I} A_i))$, by the definition of the sequential composition, we know that $\mu \in L(\cup_{i \in I} A_i)$. Thus, $\mu \in L(A_j)$ for some $j \in I$, which means $\mu \in L(S \circ A_j)$. Thus, $\mu \in \cup_{i \in I} L(S \circ A_i)$, which completes the proof. ■

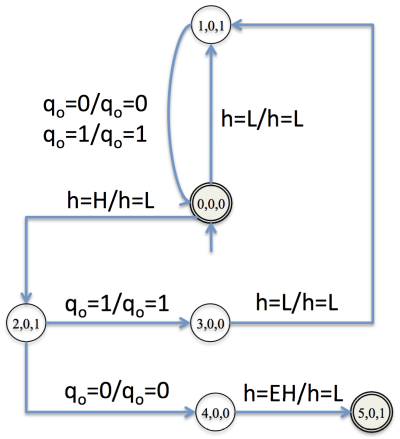

Fig. 5. Automaton model of $G \times (S \circ A)$

Proposition 3 Given two attacks A_1 and A_2 with the same input and output alphabets Σ and Δ_n, we have $L(G \times (S \circ (A_1 \cup A_2))) = L(G \times (S \circ A_1)) \cup L(G \times (S \circ A_2))$.

Proof: By the above definition of composition, the proposition follows. ■

So far we have introduced a simple sensor attack model, and explained how this attack affects the closed-loop system. But we have not described what kind of sensor attacks can be considered intelligent. Next, we will introduce the concept of ABSRA.

2.3 An ABSRA model

Definition 1 A given closed loop system (G, S) is *attackable* if there exists a non-empty attack model A such that the following properties hold:

(1) [**Covertness**] Event insertions by A must be covert to S, i.e.,

$$\theta(L(A)) \subseteq L(S), \tag{1}$$

namely the supervisor will not see any unexpected observable sequences from A.

(2) [**Control Existence**] A cannot stop uncontrollable events allowed by the supervisor S, i.e.,

$$(\forall \mu \in L(S \circ A))\, E_S(\theta(\mu)) \cap \Sigma_{uc} \subseteq \psi(E_{S \circ A}(\mu)). \tag{2}$$

(3) [**Strong damage infliction**] A may always cause "damages" to G, i.e.,

$$(\forall s \in L(G \times (S \circ A)))(\exists t \in L(G \times (S \circ A))) \\ s \leq t \wedge \psi(t) \in L(G) - L(S), \tag{3}$$

namely A will cause G to generate behaviours unspecified by S eventually.

(4) [**Control Feasibility**] The closed-loop language is normal [2] with respect to (G, P_o), i.e.,

$$P_o^{-1}(P_o(\psi(L(G \times (S \circ A))))) \cap L(G) \\ = \psi(L(G \times (S \circ A))). \tag{4}$$

A above is called an *Attack with Bounded Sensor Reading Alterations* (ABSRA) of (G, S). We say A is *non-redundant* w.r.t. (G, S) if $L(G \times (S \circ A)) = L(A)$. □

The last property of Control Feasibility is essentially about observability of the closed-loop behaviour under the control of $S \circ A$. By imposing normality to achieve observability, we can ensure the existence of the largest ABSRA model, which will be introduced shortly. In addition, imposing normality can bring some computational advantages, as almost all supervisor synthesis tools, e.g., TCT and SuSYNA, can compute a normal language. We can check that the attack A shown in Figure 4 does not satisfy Property (1) because S cannot fire $q_o = 0$ before h=L, but A can. Nevertheless, the sequential composition $S \circ A$ induces an ABSRA.

Theorem 1 Given a plant G and a supervisor S, let $\{A_i | i \in I\}$ be a (possibly infinite) collection of ABSRA models with respect to (G, S). Then $\cup_{i\in I} A_i$ satisfies properties (1)-(4).

Proof: By Prop. 2, we know that $L(S \circ (\cup_{i\in I} A_i)) = \cup_{i\in I} L(S \circ A_i)$. We verify properties (1)-(4).
1) For the covertness, since for each $i \in I$, A_i is an ABSRA model, we have that $\theta(L(A_i)) \subseteq L(S)$. Thus, we have that

$$\theta(L(\cup_{i\in I} A_i)) = \theta(\cup_{i\in I} L(A_i)) = \cup_{i\in I} \theta(L(A_i)) \subseteq L(S).$$

To show property (2), for all $\mu \in L(S \circ (\cup_{i\in I} A_i))$, let $J := \{i \in I | \mu \in L(S \circ A_j)\}$, and we have $\psi(E_{S\circ(\cup_{i\in I} A_i)}(\mu)) = \cup_{j\in J} \psi(E_{S\circ A_j}(\mu))$. Because each A_j ($j \in J$) is an ABSRA model, we have $E_S(\theta(\mu)) \cap \Sigma_{uc} \subseteq \psi(E_{S\circ A_j}(\mu))$. Thus, we have $E_S(\theta(\mu)) \cap \Sigma_{uc} \subseteq \cup_{j\in J} \psi(E_{S\circ A_j}(\mu)) = \psi(E_{S\circ(\cup_{i\in I} A_i)}(\mu))$.
2) For the strong damage infliction, for all $s \in L(G \times (S \circ (\cup_{i\in I} A_i))) = L(G \times (\cup_{i\in I} (S \circ A_i)))$, there must exist $j \in I$ such that $s \in L(G \times (S \circ A_j))$. Since A_j is an ABSRA model, there must exist $t \in L(G \times (S \circ A_j)) \subseteq L(G \times (S \circ (\cup_{i\in I} A_i)))$ such that $s \leq t$ and $\psi(t) \in L(G) - L(S)$. The last property of Control Feasibility can be shown straightforwardly, owing to the fact that $L(G \times (S \circ (\cup_{i\in I} A_i))) = \cup_{i\in I} L(G \times (S \circ A_i))$. This completes the proof of the theorem. ■

Let $\mathcal{A}(G, S)$ be the set of all non-redundant ABSRA models with respect to (G, S). Let $\preceq$ be a binary relation over $\mathcal{A}$ such that for any $A_1, A_2 \in \mathcal{A}$, $A_1 \preceq A_2$ iff $L(A_1) \subseteq L(A_2)$. Clearly, $\preceq$ is a partial order. Since for all $A \in \mathcal{A}$ we have $L(A) \subseteq L(G \times (S \circ A_0))$, where $L(A_0) = (\Sigma \times \Delta_n)^*$, Theorem 1 and Prop. 1 imply that the least restrictive (or supremal) non-redundant attack language exists. But it is not clear whether this supremal language is regular, i.e., whether it can be recognized by a finite-state automaton. Therefore, at this moment the existance of the supremal non-redundant ABSRA is still unknown. In the next section we will show that the supremal non-redundant attack language is regular, i.e., the supremal non-redundant ABSRA exists, and is computable.

3 Synthesis of an ABSRA

We first recall the concepts of controllability [4], and normality [2]. Let Λ be an alphabet, which can be either Σ or $(\Sigma \cup \{\epsilon\}) \times \Delta_n$, depending on a specific application context. Let $\Lambda_{uc} \subseteq \Lambda$ and $\Lambda_o \subseteq \Lambda$ be the uncontrollable and observable alphabets, respectively, where if $\Lambda = (\Sigma \cup \{\epsilon\}) \times \Delta_n$ then $\Lambda_o := (\Sigma_o \cup \{\epsilon\}) \times \Delta_n$. Let $\hat{P}_o : \Lambda^* \to \Lambda_o^*$ be the natural projection, where

- $\hat{P}_o(\epsilon) := \epsilon$,
- $(\forall \mu \in \Lambda)\, \hat{P}_o(\mu) := \begin{cases} \mu & \text{if } \mu \in \Lambda_o \\ \epsilon & \text{if } \mu \in \Sigma - \Sigma_o \\ (\epsilon, \theta(\mu)) & \text{if } \mu \in (\Sigma - \Sigma_o) \times \Delta_n \end{cases}$
- For all $s\mu \in \Lambda^*$, $\hat{P}_o(s\mu) = \hat{P}_o(s)\hat{P}_o(\mu)$.

Let $\hat{P}_o^{-1} : 2^{\Lambda_o^*} \to 2^{\Lambda^*}$ be the inverse image map of $\hat{P}_o$, where for all $L \subseteq \Lambda_o^*$, $\hat{P}_o^{-1}(L) := \{s \in \Lambda^* | \hat{P}_o(s) \in L\}$.

Definition 2 Given a finite-state automaton G whose alphabet is Λ, a sublanguage $K \subseteq L_m(G)$ is *controllable* with respect to G and Λ_{uc}, if $\overline{K}\Lambda_{uc} \cap L(G) \subseteq \overline{K}$. □

Definition 3 Given a finite-state automaton G whose alphabet is Λ, a sublanguage $K \subseteq L_m(G)$ is *normal* with respect to G and Λ_o, if $\hat{P}_o^{-1}(\hat{P}_o(K)) \cap L(G) = K$.□

Given a requirement $\mathcal{E} \subseteq \Lambda^*$, let

$$\mathcal{CN}(G, \mathcal{E}) := \{K \subseteq L_m(G) \cap \mathcal{E} | K \text{ is controllable w.r.t. } G \text{ and } \Lambda_{uc}, \text{ and } \overline{K} \text{ is normal w.r.t. } G \text{ and } \Lambda_o\}.$$

By [4], we know that the supremal controllable and normal sublanguage of $L_m(G)$ exists, denoted as $\sup\mathcal{CN}(G, \mathcal{E})$, such that for all $K \in \mathcal{CN}(G, \mathcal{E})$, we have $K \subseteq \sup\mathcal{CN}(G, \mathcal{E}) \in \mathcal{CN}(G, \mathcal{E})$.

In our setup an attacker is able to arbitrarily alter an observable event. Thus, each event $(\sigma, \nu) \in \Sigma_o \times \Delta_n$ is considered controllable, as the attacker can choose not to use this alteration. Under this consideration, the uncontrollable alphabet Λ_{uc} is actually empty. Thus, in the following attack model synthesis, we do not explicitly require controllability. This may sound a bit unusual because we do have an uncontrollable alphabet Σ_{uc} for the plant G - how those uncontrollable events affect the attack model synthesis? If we carefully check the properties of an ABSRA, we can see that Property (2) actually implicitly enforces controllability with respect to Σ_{uc} because it requires the attacker not to change the event enablement of the supervisor S on uncontrollable events at the current state.

Let $\Sigma_{o,p} \subseteq \Sigma_o$ denote a set of *protected* observable events that cannot be altered by an ABSRA, i.e., given an attack model $A = (Y, \Sigma \times \Delta_n, \eta, y_0, Y)$,

$$(\forall y \in Y)(\forall (\sigma, \nu) \in \Sigma_{o,p} \times \Delta_n)\, \eta(y, \sigma, \nu)! \Rightarrow \nu = \sigma. \quad (5)$$

We now state our first synthesis problem.

Problem 1 *Given a closed-loop system (G, S) and $\Sigma_{o,p}$, synthesize an ABSRA model $A \in \mathcal{A}(G, S)$ with respect to $\Sigma_{o,p}$ such that for any other ABSRA model $A' \in \mathcal{A}(G, S)$ with respect to $\Sigma_{o,p}$, we have $L(A') \subseteq L(A)$.* □

To solve this problem, we present the following ABSRA synthesis procedure.
Procedure 1: (Supremal ABSRA Synthesis)

(1) Input: a plant $G = (X, \Sigma, \xi, x_0, X_m)$, a supervisor $S = (Z, \Sigma, \delta, z_0, Z)$ and $\Sigma_{o,p}$.
(2) Construct a single-state automaton $A_0 = (Y, \Sigma \times \Delta_n, \eta, y_0, Y)$, where $Y = \{y_0\}$ and the transition map η encodes transitions labeled by events in $((\Sigma_o \setminus \Sigma_{o,p}) \times \Delta_n) \cup (\Sigma_{o,p} \times \Sigma_{o,p}) \cup (\Sigma_{uo} \times \{\epsilon\})$,

denoting all observable event alterations that the attacker can consider.

(3) Let $\mathcal{E} = (Q, \Sigma, \kappa, q_0, Q_m)$ be an automaton such that $L_m(\mathcal{E}) = \sup\mathcal{CN}(G, L(G) - L(S))$.

(4) Let $S \circ A_0 = (Z \times Y \cup \{d\}, \Sigma \times \Delta_n, \zeta, (z_0, y_0), Z \times Y \cup \{d\})$ be sequential composition.

(5) Let $B := \mathcal{E}\hat{\times}(S \circ A_0) = (U = Q \times (Z \times Y \cup \{d\}) \cup Q, \Sigma \times \Delta_n, \pi, u_0 = (q_0, z_0, y_0), U_m = Q_m \times Z \times Y)$ be an automaton, where for each $u \in U$ and $(\sigma, \nu) \in \Sigma \times \Delta_n$,

$$\pi(u, \sigma, \nu) :=$$

$$\begin{cases} (q', w') & \text{if } u = (q, w) \in U,\ \kappa(q, \sigma) = q' \text{ and} \\ & \zeta(w, \sigma, \nu) = w', \\ q' & \text{if } u = (q, w) \in U,\ \kappa(q, \sigma) = q' \text{ and} \\ & \zeta(w, \sigma, \nu) \text{ undefined}, \\ q' & \text{if } u \in Q \text{ and } \kappa(u, \sigma) = q' \\ \text{undefined} & \text{otherwise;} \end{cases}$$

(6) Compute $A_1 = (Y_1, \Sigma \times \Delta_n, \eta_1, y_{0,1}, Y_{m,1})$ via the subset construction on B, where

- $Y_1 := \{(u, < s >) \in U \times 2^U | u \in < s >\}$, where

$$u' \in < s > \iff (\exists t \in \hat{P}_o^{-1}(\hat{P}_o(s)))\, \pi(u_0, t) = u'.$$

- $Y_{m,1} := \{(u, < s >) \in Y_1 | u \in U_m\}$ and $y_{0,1} = (u_0, < \epsilon >)$.
- $\eta_1 : Y_1 \times \Sigma \times \Delta_n \to Y_1$, where for all $(u, < s >), (u', < s' >) \in Y_1$ and $\mu \in \Sigma \times \Delta_n$,

$$\eta_1(u, < s >, \mu) = (u', < s' >) \iff \pi(u, \mu) = u' \wedge < s' > = < s\mu > .$$

(7) Undertake the following iteration on $k = 1, 2, \cdots$:

(a) $\hat{\Omega}_k := \{(u, < s >) \in Y_k | (\exists u' \in < s >)\ (\forall t \in (\Sigma \times \Delta_n)^*)\, \eta_k(u', < s >, t)! \Rightarrow \eta_k(u', < s >, t) \notin Y_{m,k}\} \cup \{(q, z, v, < s >) \in Y_k | E_S(z) \cap \Sigma_{uc} \not\subseteq E_{A_k}(q, z, v, < s >)\}$.

(b) If $\hat{\Omega}_k = \varnothing$, go to Step (8). Otherwise, continue.

(c) $\Omega_k := \{(u, < s >) \in Y_k | (\exists u' \in < s >)\, (u', < s >) \in \hat{\Omega}_k\}$.

(d) Let $A_{k+1} := (Y_{k+1}, \Sigma \times \Delta_n, \eta_{k+1}, y_{0,k+1}, Y_{m,k+1})$, where

- $Y_{k+1} := Y_k - \Omega_k$, $Y_{m,k+1} := Y_{m,k} - \Omega_k$, $y_{0,k+1} := y_{0,k}$.
- $(\forall y, y' \in Y_{k+1})(\forall \mu \in \Sigma \times \Delta_n)$ $\eta_{k+1}(y, \mu) = y' \iff \eta_k(y, \mu) = y'$.

(e) Set $k := k + 1$, and go to Step (7.a).

(8) Output A_k. □

In Procedure 1, Step (3) is to determine the "illegal" sublanguage, which is not allowed by the supervisor S. An ABSRA attacker aims to bring the closed-loop system's behaviour into $L_m(\mathcal{E})$. Step (4) is to determine all possible observation alterations on S. Step (5) describes the closed-loop language under the influence of A_0. Steps (6)-(7) are used to ensure the closed-loop language to be nonblocking, normal and in addition, the outgoing uncontrollable event set of each state in S will not be affected by an attacker, as required by Property (2) in Definition 1. More explicitly, Step (6) is used to construct one automaton A_1, which not only is language equivalent to the automaton B, but also reveals for each state, say $(u, < s >) \in Y_1$ with $\pi(u_0, s) = u$, all observation equivalent states $\{(u', < s >) \in Y_1 | (\exists t \in \hat{P}_o^{-1}(\hat{P}_o(s)))\, \pi(u_0, t) = u'\}$. Later in Step (7), by enforcing the property that $(u, < s >)$ is "valid" iff all its observation equivalent states are "valid", the normality property can be ensured. The set $\hat{\Omega}_k$ in Step (7.a) is used to identify states in A_k, which may violate either Property (2) (denoted by $\{(q, z, v, < s >) \in Y_k | E_S(z) \cap \Sigma_{uc} \not\subseteq E_{A_k}(q, z, v, < s >)\}$), or the normality or nonblocking property, (denoted by $\{(u, < s >) \in Y_k | (\exists y' \in < s >)(\forall t \in (\Sigma \times \Delta_n)^*)\, \eta_k(y', t)! \Rightarrow \eta_k(y', t) \notin Y_{m,k}\}$). Clearly, each step in Procedure 1 terminates finitely, so does Procedure 1. Next, we show the prefix closure of A_k in Step (8), if not empty, is the supremal ABSRA model.

Proposition 4 Let A_k be obtained in Procedure 1. Then $L(S \circ A_k) = L(A_k)$.

Proof: By the definition of sequential composition, we have $L(S \circ A_k) \subseteq L(A_k)$. So we only need to show that $L(S \circ A_k) \supseteq L(A_k)$. For each string $s \in L(A_k)$, assume that $s = (\sigma_1, \nu_1) \cdots (\sigma_l, \nu_l)$ for some $l \in \mathbb{N}$. By the definition of A_k, there must exist $y_1 = (u_1 = (q_1, z_1, v_1), < s_1 >), \cdots, y_l = (u_l = (q_l, z_l, v_l), < s_l >) \in Y_k$ such that $\eta(y_{i-1}, \sigma_i, \nu_i)) = y_i$ for $i = 1, 2, \cdots, l$. By the definition of η_k, we know that $\delta(z_{i-1}, \sigma_i) = z_i$ for $i = 1, 2, \cdots l$. Thus, by the definition of sequential composition, we have $s \in L(S \circ A_k)$. ■

Theorem 2 If the output finite-state automaton A_k in Procedure 1 is not empty, then its prefix closure is the supremal non-redundant ABSRA of (G, S) with respect to $\Sigma_{o,p}$. Otherwise, there is no ABSRA of (G, S) with respect to $\Sigma_{o,p}$.

Proof: (a) We first show that the prefix closure A_* of A_k is an ABSRA, i.e., A_* satisfies properties (1)-(4). By the definition of B in Step (5) of Procedure 1, we know that in each marker state $u = (q, w) \in U_m$ we have $w \neq d$. In addition, by the definiton of $S \circ A_0$ we know that the dump state d is a deadlock state. Thus, in Steps (6)-(7), which computing a normal nonblocking sublanguage [11], we know that no state in A_k (thus, in A_*) will contain d as a component, namely $\theta(L(A_*)) = \theta(L(A_k)) \subseteq L(S)$. To show that property (3) holds, we only need to show that for all $s \in L(G \times (S \circ A_*)) = L(G \times (S \circ A_k))$, there exists $t \in L(G \times (S \circ A_*)) = L(G \times (S \circ A_k))$ such that $s \leq t$ and $\psi(t) \in \psi(L_m(A_k))$, because $\psi(L_m(A_k)) \subseteq \psi(L_m(A_1)) = \psi(L_m(B)) \subseteq L_m(E) = L(G) - L(S)$. By the definition of the closed-loop system's behaviour, we know that $s \in L(S \circ A_k)$. By Prop. 4 we have $L(S \circ A_k) = L(A_k) = L(A_*)$, we know that $s \in L(A_*) = L(A_k) = \overline{L_m(A_k)}$ because A_k is nonblocking owing to Step (7), whic removes all blocking states from A_1. Thus, there must exist $t \in L_m(A_k)$ such that $s \leq t$. Clearly, $\psi(t) \in \psi(L_m(A_k))$, which completes the proof of property (3). To show property (2), for each string $\mu \in L(S \circ A_*) = L(S \circ A_k)$, by Prop. 4 we know that $E_{S \circ A_k}(\mu) = E_{A_k}(\mu)$. If $E_S(\theta(\mu)) \cap \Sigma_{uc} \not\subseteq \psi(E_{A_k}(\mu))$, then because $\mu \in L(S \circ A_k) = L(A_k)$, we know that there exists $(q, z, v, < \mu >) \in Y_k$ such that $\eta_k(u_0, < \epsilon >, \mu) = (q, z, v, < \mu >)$. Clearly, we have $E_S(z) \cap \Sigma_{uc} \not\subseteq E_{A_k}(q, z, v, < \mu >)$, meaning that $(q, z, v, < \mu >) \in \Omega_k$, namely $\mu \notin L(A_k)$,which contradicts our assumption that $\mu \in L(S \circ A_k) = L(A_k) = L(S \circ A_*)$. Thus, we have $E_S(\theta(\mu)) \cap \Sigma_{uc} \subseteq \psi(E_{S \circ A_*}(\mu)) = \psi(E_{A_k}(\mu))$, namely property (2) must hold. To show that A_* satisfies property (4), since $L(G \times (S \circ A_*)) = L(G \times (S \circ A_k))$, we need to show that

$$P_o^{-1}(P_o(\psi(L(G \times (S \circ A_k))))) \cap L(G) = \psi(L(G \times (S \circ A_k))).$$

It is true that

$$P_o^{-1}(P_o(\psi(L(G\times(S\circ A_k)))))\cap L(G) \supseteq \psi(L(G\times(S\circ A_k))).$$

Thus, we only need to show that

$$P_o^{-1}(P_o(\psi(L(G\times(S\circ A_k)))))\cap L(G) \subseteq \psi(L(G\times(S\circ A_k))).$$

Assume that this is not true. Then there must exist $s \in L(G) - \psi(L(G\times(S\circ A_k)))$ and $s' \in \psi(L(G\times(S\circ A_k)))$ such that $P_o(s) = P_o(s')$. Clearly, $s' \in \psi(L(S\circ A_k)) \subseteq L(E)$. Since $L_m(\mathcal{E}) = \sup\mathcal{CN}(G, L(G)-L(S))$, we know that $P_o^{-1}(\{P_o(s')\}) \cap L(G) \subseteq L(\mathcal{E})$. Thus, $s \in L(\mathcal{E})$. Since $s \notin \psi(L(G\times(S\circ A_k)))$, we know that $s \notin \psi(L(S\circ A_k))$. Thus, in Step (5) of Procedure 1, we know that there exists $\mu \in (\Sigma\times\Delta_n)^*$ such that $\psi(\mu) = s$ and $\pi(u_0,\mu) \in Q$ in B. Since $s' \in \psi(L(S\circ A_k))$, we know that there exists $\mu' \in (\Sigma\times\Delta_n)^*$ such that $\psi(\mu') = s'$ and $\pi(u_o,\mu') \in Q\times Z\times Y$. By the definition of A_1, since $P_o(s) = P_o(s')$, which means $\hat{P}_o(\mu) = \hat{P}_o(\mu')$, we know that $\pi(u_0,\mu') \in <\mu>=<\mu'>$. But since $\pi(u_0,\mu) \in Q \notin U_m$, by the definition of B, we know that for all $t \in (\Sigma\times\Delta_n)^*$, if $\pi(\pi(u_0,\mu),t)!$, then $\pi(\pi(u_0,\mu),t) \notin U_m$. By the definition of A_k, we know that $(\pi(u_0,\mu'), <\mu'>) \in \hat{\Omega}_k$ because $\pi(u_0,\mu) \in <\mu>=<\mu'>$ and for all $t \in (\Sigma\times\Delta_n)^*$, if $\eta_k(\pi(u_0,\mu), <\mu'>, t)!$ then $\eta_k(\pi(u_0,\mu), <\mu'>, t) \notin Y_{m,k} \subseteq A_{1,m} \subseteq U_m \times 2^U$. Thus, Procedure 1 cannot terminate at k, which contracts the fact that Procedure 1 terminates and outputs A_k. Thus, Property (4) of Definition 1 must hold, which completes our proof that A_* is an ABSRA model.

To show that A_* is non-redundant w.r.t. (G,S), we need to show that $L(G\times(S\circ A_*)) = L(A_*)$. It is clear that $L(G\times(S\circ A_*)) \subseteq L(A_*)$. So we only need to show that $L(G\times(S\circ A_*)) \supseteq L(A_*)$, which is true because $L(A_*) = L(A_k) \subseteq L(\mathcal{E}\times(S\circ A_k)) \subseteq L(G\times(S\circ A_k))$.

Because A_* is a non-redundant ABSRA model, to show that A_* is the supremal non-redundant ABSRA model, we only need to show that for all non-redundant ABSRA model $\hat{A}$, we have $L(\hat{A}) = L(G\times(S\circ\hat{A})) \subseteq L(G\times(S\circ A_*)) = L(A_*)$. Clearly, $L(\hat{A}) \subseteq L(A_0)$. By Prop. 1, we know that $L(S\circ\hat{A}) \subseteq L(S\circ A_0)$. Owing to properties (2)-(4) (i.e., controllability, strong damage infliction and normality) in Definition of 1, we know that $\psi(L(G\times(S\circ\hat{A}))) \subseteq L(\mathcal{E})$. Thus, $L(G\times(S\circ\hat{A})) \subseteq \psi(L(B))$. Since $L(A_1) = L(B)$, we know that $L(G\times(S\circ\hat{A})) \subseteq L(A_1)$. Owing to properties (2)-(4) in Definition 1, we know that $L(G\times(S\circ\hat{A})) \subseteq L(A_i)$ $(i = 2,3,\cdots)$. Thus, $L(G\times(S\circ\hat{A})) \subseteq L(A_k) = L(G\times(S\circ A_*))$. This means, if A_k is not empty, then its prefix closure A_* is the supremal non-redundant ABRSA model w.r.t. (G,S).

The proof also indicates that, if A_k is empty, then there does not exist ABSRA model, because by Theorem 1 and Prop. 1 we know that, if there exists one ABSRA model, then the supremal one must exist, and by the above proof, we know that this supremal one must be the prefix closure of A_k, which means it is not empty, contradicting to our assumption that A_k is empty. ■

Theorem 2 indicates that the existence of the supremal ABSRA model can be decided by running Procedure 1 and check whether the output A_k is empty. In case that A_k is not empty, its prefix closure is the supremal one. The computational complexity of Procedure 1 can be roughly estimated as follows. The number of states $|B|$ is upper bounded by $|E|(|S||A_0|+1) \leq |G||S|(|S|+1) \approx |G||S|^2$ because the number of states $|E|$ is no more than $|G||S|$. The subset construction of A_1 in Step (6) leads to $|A_1| \leq 2^{|B|} \leq 2^{|G||S|^2}$. The total complexity of constructing A_k is $O(|A_k|^2|\Sigma||\Delta_n|)$. In the worst case, Step (7) may iterate at most $2^{|B|}$ times, where at each iteration step $k \geq 2$ one state is removed from A_{k-1}. Considering that the complexity of Procedure 1 is determined by Steps (6)-(7), and $|\Delta_n| = \frac{|\Sigma|^{n+1}-1}{|\Sigma|-1}$, we have the overall complexity is

$$\sum_{k=1}^{|A_1|} O(|A_k|^2|\Sigma||\Delta_n|) \leq \sum_{k=1}^{2^{|G||S|^2}} O((|A_1|+1-k)^2|\Sigma||\Delta_n|)$$
$$\leq O\Big(2^{3|G||S|^2}|\Sigma|\frac{|\Sigma|^{n+1}-1}{|\Sigma|-1}\Big),$$

which is exponential in time with respect to $|G|$ and $|S|$ when Σ and n are given.

As an illustration, we apply Procedure 1 to the plant G and the supervisor S shown in Figure 3. Assume that $\Sigma_{o,p} = \{q_0 = 0, q_0 = 1\}$. The finite-state automaton A_0 in Procedure 1 is depicted in Figure 6. The finite-state automaton E is depicted in Figure 7,

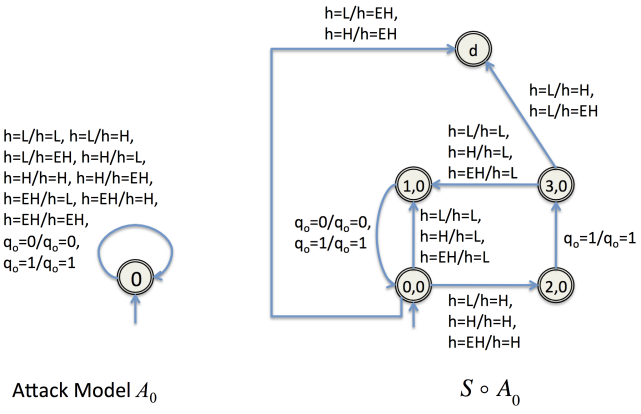

Fig. 6. The finite-state automata A_0 and $S\circ A_0$

where the marked behaviour consists of only strings of $L(G)-L(S)$. The finite-state automaton $B = \mathcal{E}\hat{\times}(S\circ A_0)$

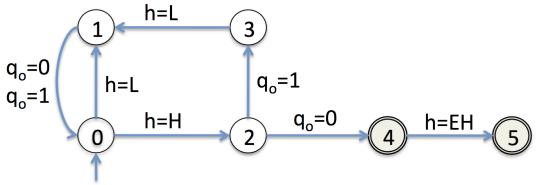

Fig. 7. The finite-state automaton $\mathcal{E}$

is depicted in Figure 8, where $\underline{\sigma}$ denotes $\{\sigma\}\times\Delta_n$, e.g., $\underline{q_0}$ denotes $\{q_0\}\times\Delta_n$. Because all events are observable, the finite-state automaton A_1 generated via the subset construction in Step (6) is the same as the automaton B depicted in Figure 8. We can easily check that $\hat{\Omega}_1 = \{(1,d),(2,d),0,1,2,3,4,5\}$. The resulting finite-state automaton A_2 is depicted in Figure 9. We can check that $\hat{\Omega}_2 = \varnothing$. Thus, the output of Procedure 1 is A_2. By Theorem 2, the supremal ABSRA model is the prefix closure of A_2.

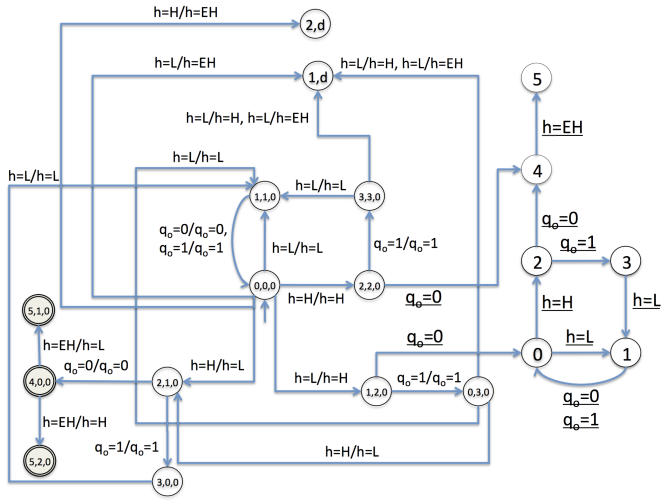

Fig. 8. The finite-state automaton $B = \mathcal{E}\hat{\times}(S \circ A_0)$

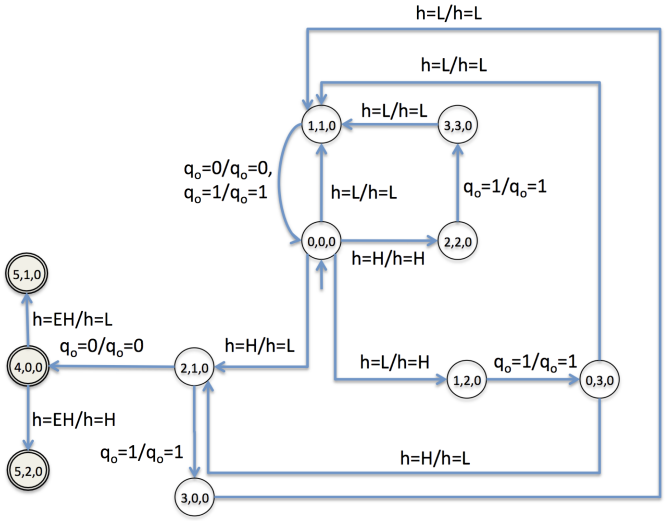

Fig. 9. The finite-state automaton A_2

4 Synthesis of an ABSRA-robust supervisor

In the previous section we discuss how to design an ABSRA model to interrupt a given system's operations from an attacker's point of view. In this section we present a synthesis approach to design a supervisor S, which is "robust" to any ABSRA in the sense that either the attack is not covert or incurs no damage to the system. More precisely, we introduce the following concept of *robustness*.

Definition 4 Given a closed-loop system (G, S), whose alphabet is Σ, let $\Sigma_{o,p} \subseteq \Sigma_o$ be a protected observable alphabet. Then S is *ABSRA-robust* with respect to $\Sigma_{o,p}$ if there does not exist an ABSRA model $A = (Y, \Sigma \times \Delta_n, \eta, y_0, Y)$ of (G, S), where

$$(\forall y \in Y)(\forall (\sigma, \nu) \in \Sigma_{o,p} \times \Delta_n)\eta(y, \sigma, \nu)! \Rightarrow \nu = \sigma.$$

□

In general, given two ABSRA-robust supervisors S_1 and S_2, their union need not be ABSRA-robust. A simple example is shown in Figure 10, where $\Sigma = \{a, b, c, d\}$, $\Sigma_o = \Sigma$, $\Sigma_{uc} = \varnothing$, and $\Sigma_{o,p} = \{c, d\}$. We can check that both S_1 and S_2 are ABSRA-robust with respect to $\Sigma_{o,p}$, because there is no ABSRA model satisfies Property (1)

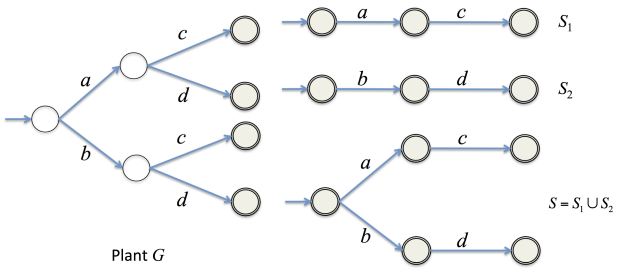

Fig. 10. Non-closure of ABSRA-robustness under union

of Definition 1. But $S = S_1 \cup S_2$ is not ABSRA-robust because an ABSRA attacker can replace a with b, and replace b with a. This example shows that the supremal ABSRA-robust supervisor does not exist. With the concept of ABSRA-robustness we introduce another synthesis problem shown below.

Problem 2 Given a plant G, a requirement $\mathcal{E}$, and a protected observable alphabet $\Sigma_{o,p} \subseteq \Sigma_o$, synthesize a supervisor S, which is ABSRA-robust with respect to $\Sigma_{o,p}$. □

With the same notations used in the previous section, let $\mathcal{CN}(G, \mathcal{E})$ be the collection of all controllable and normal supervisors [11]. Let $S_0 = \sup\mathcal{CN}(G, \mathcal{E})$, which always exists and computable, as long as $\mathcal{E} \subseteq \Sigma^*$ is regular. Our goal is to design a supervisor $S \in \mathcal{CN}(G, \mathcal{E})$ such that Procedure 1 returns an empty ABSRA A_k with respect to the given protected observable alphabet $\Sigma_{o,p}$. To this end, we present the following procedure:

Procedure 2: (Synthesis for ABSRA-Robustness)

(1) Input: a plant G, a requirement $\mathcal{E}$ and a protected observable alphabet $\Sigma_{o,p}$.
(2) Compute $\hat{K} = \sup\mathcal{CN}(G, \mathcal{E})$. If $\hat{K} = \varnothing$, set $K = \varnothing$ and go to Step (5). Otherwise, assume $\hat{K}$ is recognized by a finite-state automaton $\hat{S}$, and continue.
(3) Compute A by using Procedure 1 on the inputs $(G, \hat{S})$ and $\Sigma_{o,p}$.
(4) Compute $K := \sup\mathcal{CN}(G, L(\hat{S}) - \theta(L_m(A))\Sigma^*)$.
(5) Output: a recognizer S of $\overline{K}$. □

Theorem 3 Given a plant G, a requirement $E \subseteq \Sigma^*$, and a protected observable alphabet $\Sigma_{o,p}$, let S be computed above. If $L(S) \neq \varnothing$, then S is ABSRA-robust with respect to $\Sigma_{o,p}$.

Proof: To show that S is ABSRA-robust with respect to $\Sigma_{o,p}$, assume that it is not true. Then Procedure 1 returns a non-empty ABSRA model A with the inputs of (G, S) and $\Sigma_{o,p}$. Let $\mu \in L_m(A)$. By Prop. 5 we know that $\mu \in L(S \circ A) = L(A)$. Thus, $\theta(\mu) \in L(S)$. On the other hand, we know that $L(S) \subseteq L(\hat{S}) - \theta(L_m(A))\Sigma^*$, which means $\theta(\mu) \notin L(S)$, which leads to a contradiction. Thus, S is ABSRA-robust with respect to $\Sigma_{o,p}$. ■

We would like to emphasize again that, when Procedure 1 returns an empty A, and by Theorem 3 we conclude that there is no ABSRA A for the closed-loop system (G, S), it does not mean that a sensor reading alteration attack will not be carried out by an attacker. But such an attack will either not be able to inflict any damage to the system or reveal itself to the supervisor before it achieves its attack goal owing to abnormal sys-

tem executions, which can be formulated as an abnormality detection problem similar to fault diagnosis [33] and proper contingent actions such as system shutdown can be taken by the supervisor, which is nevertheless outside the scope of this paper.

To determine the complexity of Procedure 2, we know the complexity of Step (2) is $O(2^{2|G||\mathcal{E}|}|\Sigma|)$, where the state size of $\hat{S}$ is $|\hat{S}| \leq 2^{|G||\mathcal{E}|}$. Then the complexity of Step (3) is the same as that of Procedure 1 with respect to $|G|$ and $|\hat{S}|$, where the state size of A is $|A| \leq 2^{|G||\hat{S}|^2} \leq 2^{|G|2^{2|G||\mathcal{E}|}}$. The state size of $L(\hat{S}) - \theta(L_m(A))\Sigma^*$ in Step (4) is $|\hat{S}||A| \leq 2^{|G|(|\mathcal{E}|+2^{2|G||\mathcal{E}|})}$. The final complexity of computing S is $O(2^{2|G|2^{|G|(|\mathcal{E}|+2^{2|G||\mathcal{E}|})}}|\Sigma||\Delta_n|)$, which is triple exponential with respect to $|G|$ and $|\mathcal{E}|$ when Σ and n are given.

As an illustration of Procedure 2, we revisit the plant G depicted in Figure 3 with the same requirement that the water level should not be extremely high, i.e., the event h=EH should never occur. Then we can see that the finite-state automaton $\hat{S}$ is the same as the supervisor S shown in Figure 3. Assume that $\Sigma_{o,p} = \{q_0 = 0, q_0 = 1\}$. By the same description shown in the previous section, we can check that Procedure 1 terminates at $k = 2$, and the outcome A is depicted in Figure 9. The language $\theta(L_m(A))$ is depicted in Figure 11. The

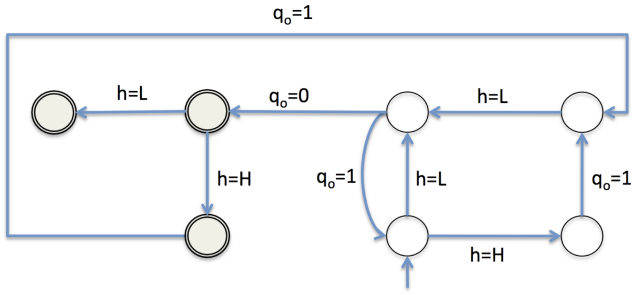

Fig. 11. A finite-state automaton recognising $\theta(L_m(A))$

language $L(\hat{S}) - \theta(L_m(A))\Sigma^*$ is depicted in Figure 12. The sublanguage $K = \sup\mathcal{CN}(G, L(\hat{S}) - \theta(L_m(A))\Sigma^*)$

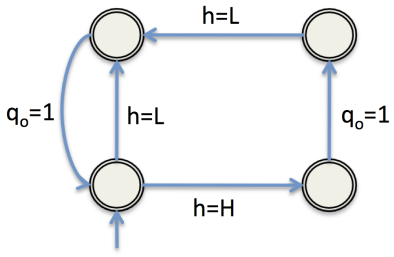

Fig. 12. An automaton recognizing $L(\hat{S}) - \theta(L_m(A))\Sigma^*$

and the final ABSRA supervisor S are depicted in Figure 13, which indicates that the event $q_0 = 0$ should not occur, namely the valve should not be closed. Otherwise, an ABSRA attacker can replace h=H with h=L and trick the supervisor S to issue the command $q_0 = 0$ improperly.

Recall that an ABSRA affects a target system (G, S) by altering the sequence of observable events, which tricks S to issue commands improperly. By protecting

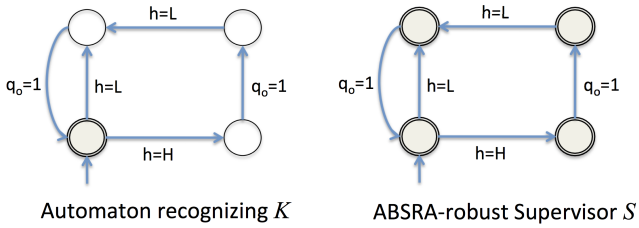

Fig. 13. Sublanguage K and ABSRA-robust supervisor S

observable events from being altered unnoticeably can in principle effectively deter an ABSRA. An observable event in this framework denotes a specific set of strongly associated measurements. For example, in the aforementioned single-tank system, the event h=H may either be associated with one simple water level measurement or possibly several sensor measurements such as the actual water level, and the corresponding pressure on the bottom of the tank - the more sensor measurements associated with the event, the harder for an attacker to alter the event without being detected. When applying suitable encryption techniques, it is even more complicated for an attacker to complete the job. Thus, it is indeed technically feasible to prevent observable events from being altered by either adopting new secure information transmission technologies or introducing more sensors to significantly increase the complication of altering the corresponding observable event without being detected. Nevertheless, there is always a financial consideration. An attractive solution to a potential industrial user is to identify only critical observable events, which, when being protected from external alterations, will lead to a supervisor robust to any ABSRA. Thus, we have another interesting problem, which is stated below.

Problem 3 Given a plant G and a requirement $\mathcal{E}$, compute a protected observable alphabet $\Sigma_{o,p} \subseteq \Sigma_o$ of the minimum size such that a nonempty ABSRA-robust supervisor S exists. □

It is clear that Problem 3 is solvable in the sense that it is decidable whether there exists such a $\Sigma_{o,p}$ with the minimum size, because we can simply apply Procedure 2 on each subset of Σ_o to compute the corresponding supervisor S. Since there is a finite number of such subsets, this brutal-force method will terminate, and provide a protected observable alphabet of the minimum size together with the corresponding supervisor, if it exists. The complexity of Procedure 3 is simply $2^{|\Sigma_o|}$ times of the complexity of Procedure 2, which is triple exponential with respect to $|G|$ and $|\mathcal{E}|$. Thus, to find a computationally viable algorithm that can solve Problem 3 becomes important, which will be addressed in our future works.

We now use that simple single-tank system to illustrate how to determine a minimum protected observable alphabet, which allows the existence of an ABSRA-robust supervisor. Let $\Sigma_{o,p} = \{q_0{=}0, q_0{=}1, h{=}H\}$. The model $\hat{S}$ is still the same as the supervisor S shown in Figure 3. The models of A_0 and $\hat{S} \circ A_0$ are shown in Figure 14. When we run Procedure 1, the finite-state automaton $\mathcal{E}$ is still the same as the one shown in Figure 7. The finite-state automaton $B = \mathcal{E}\hat{\times}(S \circ A_0)$ is depicted in Figure 15. We can see that the marked behaviour of

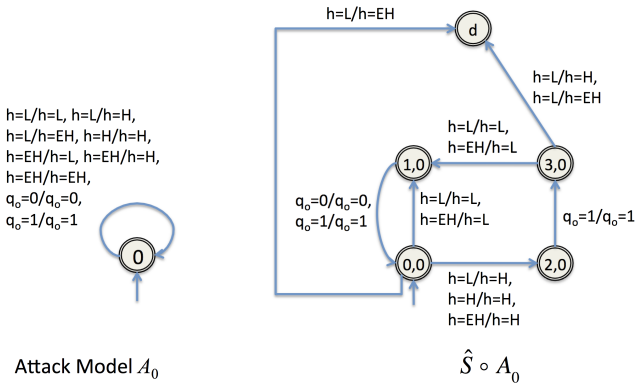

Fig. 14. Models of A_0 and $\hat{S}$

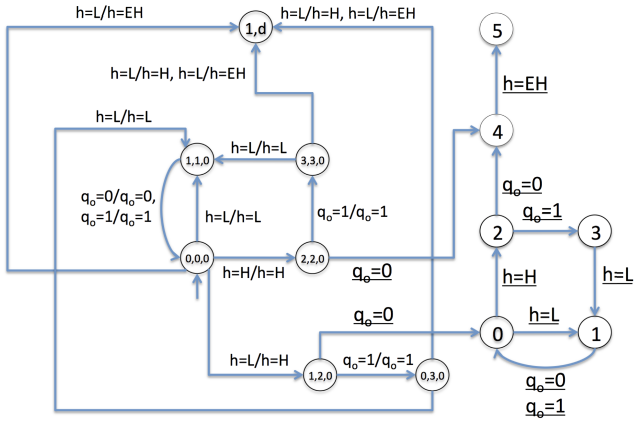

Fig. 15. The finite-state automaton $B = \mathcal{E}\hat{\times}(S \circ A_0)$

B is empty, which means the marked behaviour of A_1, which is the same as B, is also empty. Then in Step (7) the set $\hat{\Omega}_1$ contains all states of A_1, making A_2 an empty finite-state automaton. Thus, Procedure 1 terminates at $k = 2$, and output an empty A_2. By Theorem 2 we know that there does not exist an ABSRA model for the system. Thus, in Procedure 2 the resulting ABSRA-robust supervisor is the same as that S shown in Figure 3. Clearly, it is a solution to Problem 3 because we cannot find any other protected observable alphabet containing events q$_0$=0 and q$_0$=1 with a size smaller than 3, which can render an ABSRA-robust supervisor.

5 Conclusions

In this paper we have first introduced the concept of attackability and the concept of ABSRA, upon which we have shown that the supremal non-redundant ABSRA model exists and computable by Procedure 1, as long as the plant model G and the supervisor S are finitely representable, i.e., their languages are regular. After that, we have formulated the problem of synthesizing an ABSRA-robust supervisor, and provided a concrete algorithm of Procedure 2 to solve it. We have also shown that it is possible to find a minimum protected observable sub-alphabet, which may render an ABSRA-robust supervisor.

It is worth to mention that, if we use the standard concept of observability [2] in the definition of attackability, instead of normality, which is a strong property to ensure observability, the supremal non-redundant ABSRA may not exist any more. Nevertheless, the existence of an ABSRA is still decidable with possibly a higher computational complexity, as this ABSRA synthesis problem is close to the problem of supervisor synthesis under partial observation, which has been shown solvable [9] [32]. Fortunately, the normality property can be easily satisfied in reality, as it requires that only events that are both observable and controllable can be disabled - in real applications, it is typical that all control commands are observable. For this reason, the supervisor synthesis approach proposed in this paper aiming to defy ABSRA is practically feasible.

It has been shown in Section III that the complexity of Procedure 1 is exponential in time, making the ABSRA-robust supervisor synthesis problem NP-hard, as it is more complex than synthesizing the supremal controllable and normal supervisor, which is NP-hard [34]. How to synthesize an ABSRA-robust supervisor, not necessarily the supremal one, with a tractable computational complexity will be an interesting topic for future research. For example, if the natural projection P_o happens to be a natural observer [10], then the time complexity of Procedure 2 will be polynomial with respect to $|G||\mathcal{E}|$ (and a fixed Σ and n), whose degree may yet still be high for real applications. In addition, the strong damage infliction property shown in Definition 1 can be weakened so that an attacker may only want to break the integrity of the system in the sense that there is a chance that the closed-loop system may go wrong, but not always necessary. For example, we can specifiy the damage infliction property in the following way:

$$(\forall s \in L(G \times (S \circ A)))(\exists t \in L(G \times (S \circ A)))$$
$$s \leq t \wedge P_o^{-1}(P_o(\{\psi(t)\})) \cap (L(G) - L(S)) \neq \varnothing, \qquad (6)$$

which means any string $s \in L(G \times (S \circ A))$ can be extended into a string $t \in L(G \times (S \circ A))$ such that there exists $t' \in L(G \times (S \circ A))$ with $P_o(t) = P_o(t')$ and $\psi(t') \in L(G) - L(S)$. This new property can be called Weak Damage Infliction, and the corresponding supremal ABSRA synthesis problem can be solved by an algorithm almost the same as Procedure 1, except that $Y_{m,1}$ in Step (6) of Procedure 1 is now defined as $Y_{m,1} = \{(u, < s >) \in Y_1 | < s > \cap U_m \neq \varnothing\}$. A result similar to Theorem 2 can be proved with this weak damage infliction property, which is skipped in this paper, as our purpose here is to illustrate that new types of sensor attacks can be defined by changing some property in the concept of attackability.

Acknowledgement

The idea of this paper was originated from a discussion between the author and Prof Stephane Lafortune on opacity enforcement. For this reason, the author would like to thank Prof Lafortune for his contribution.

References

[1] Cassandra, C., & Lafortune, S. (2008). *Introduction to Discrete Event Systems* (2nd Ed.), Springer.

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

[3] Papadimitriou, C. H. (1994). *Computational Complexity*. Addison Wesley.

[4] Ramadge, P. J., & Wonham, W. M. (1987). Supervisory control of a class of discrete event systems. *SIAM J. Control and Optimization*, 25(1), 206–230.

[5] Su, R. (2013). Discrete-event modeling of multi-agent systems with broadcasting-based parallel composition. *Automatica*, 49(11), 3502-3506.

[6] Su, R., Van Schuppen, J. H., & Rooda, J. E. (2010). Aggregative synthesis of distributed supervisors based on automaton abstraction. *IEEE Trans. Automatic Control*, 55(7), 1627-1640.

[7] Su, R., Van Schuppen, J. H., & Rooda, J. E. (2012). Maximally permissive coordinated distributed supervisory control of nondeterministic discrete-event systems. *Automatica*, 48(7), 1237-1247.

[8] Su, R., & Wonham, W. M. (2004). Supervisor reduction for discrete-event systems. *Journal of Discrete Event Dynamic Systems*, 14(1), 31-53.

[9] Yoo, T. S., & Lafortune, S. (2006). Solvability of centralized supervisory control under partial observation.. *Discrete Event Dynamic Systems: Theory and Applications*, 16(4), 527–553.

[10] Wong, K. C. (1998). On the complexity of projections of discrete-event systems. *Proceedings of the 4th international workshop on discrete event systems (WOODES98)*, (pp. 201-206).

[11] Wonham, W. M. (2014). *Supervisory Control of Discrete-Event Systems*. Systems Control Group, Dept. of ECE, University of Toronto. URL: www.control.utoronto.ca/DES.

[12] Wonham, W. M., & Ramadge, P. J. (1987). On the supremal controllable sublanguage of a given language. *SIAM J. Control and Optimization*, 25(3), 637-659.

[13] Cardenas, A. A., Amin, S., & Sastry, S. (2008). Secure control: towards survivable cyber-physical systems. *Proceedings of the 28th International Conference on Distributed Computing Systems Workshops*, (pp. 495-500).

[14] Teixeira, A., Perez, D., Sandberg, H., & Johansson, K. H. (2012). Attack models and scenarios for networked control systems. *Proceedings of the 1st International Conference on High Confidence Networked Systems*, (pp. 55-64).

[15] Wu, Y. C., & Lafortune, S. (2014). Synthesis of insertion functions for enforcement of opacity security properties. *Automatica*, 50(5), 1336-1348.

[16] Paoli, A., Sartini, M., & Lafortune, S. (2011). Active fault tolerant control of discrete event systems using online diagnostics. *Automatica*, 47(4), 639-649.

[17] Thorsley, D., & Teneketzis, D. (2006). Intrusion detection in controlled discrete event systems. *Proceedings of the 45th IEEE Conference on Decision and Control*, (pp. 6047-6054).

[18] Carvalho, L. K., Wu, Y., Kwong, R., & Lafortune, S. (2016). Detection and prevention of actuator enablement attacks in supervisory control systems. *Proceedings of the 13th International Workshop on Discrete Event Systems*, (pp. 298-305).

[19] Cherdantseva, Y., Burnap, P., Blyth, A., Eden, P., Jones, K., Soulsby, H., & Stoddart, K. (2016). A review of cyber security risk assessment methods for SCADA systems. *Computers & Security*, 56, 1-27.

[20] Evancich, N., & Li, J. (2016). Attacks on industrial control systems. *Cyber-security of SCADA and Other Industrial Control Systems*, 66, 95-110.

[21] Fawzi, H., Tabuada, P., & Diggavi, S. (2014). Secure estimation and control for cyber-physical systems under adversarial attacks. *IEEE Trans. Automatic Control*, 59(6), 1454-1467.

[22] Knowles, W., Prince, D., Hutchison, D., Disso, J. F. P., & Jones, K. (2015). A survey of cyber security management in industrial control systems. *International Journal of Critical Infrastructure Protection*, 9, 52-80.

[23] Saboori, A., & Hadjicostis, C. N. (2007). Notions of security and opacity in discrete event systems. *Proceedings of the 46th IEEE Conference on Decision and Control*, (pp. 5056-5061).

[24] Saboori, A., & Hadjicostis, C. N. (2013). Verification of initial-state opacity in security applications of discrete event systems. *Information Sciences*, 246, 115-132.

[25] Jacob, R., Lesage, J., & Faure, J. (2016). Overview of discrete event systems opacity: models, validation, and quantification. *Annual Reviews in Control*, 41, 135-146.

[26] Rasouli, M., Miehling, E., & Teneketzis, D. (2014). A supervisory control approach to dynamic cyber-security. *Proceedings of 2014 International Conference on Decision and Game Theory for Security*, (pp. 99-117).

[27] TCT:
A Computation Tool for Supervisory Control Synthesis. http://www.control.utoronto.ca/DES/Research.html.

[28] SuSyNA:
Supervisor Synthesis for Nondeterministic Automata. http://www.ntu.edu.sg/home/rsu/Downloads.htm.

[29] Dubreil, J., Darondeau, P., & Marchand, H. (2010). Supervisory control for opacity. *IEEE Trans. Automatic Control*, 55(5), 1089-1100.

[30] Alves, M., Basilio, J. C., da Cunha, A., Carvalho, L. K., & Moreira, M. V. (2014). Robust supervisory control against intermittent loss of observations. *Proceedings of the 12th Int. Workshop on Discrete Event Syst.*, (pp. 294-299).

[31] Rohloff, K. (2012). Bounded sensor failure tolerant supervisory control. *Proceedings of the 11th Int. Workshop on Discrete Event Syst.*, (pp. 272-277).

[32] Yin, X., & Lafortune, S. (2016). Synthesis of maximally permissive supervisors for partially observed discrete event systems. *IEEE Trans. Automatic Control*, 61(5), 1239-1254.

[33] Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., & Teneketzis, D. (1995). Diagnosability of discrete-event systems. *IEEE Transactions on automatic control*, 40(9), 1555-1575.

[34] Tsitsiklis, J. N. (1989). On the control of discrete event dynamic systems. *Mathematics of Control Signals and Systems*, 2(2), 95-107.

Dr Rong Su received the BE degree in Automatic Control from the University of Science and Technology of China in 1997, and the MASc and PhD degrees both in electrical engineering from the University of Toronto in 2000 and 2004, respectively. Since then, he was affiliated with University of Waterloo and Technical University of Eindhoven before he joined Nanyang Technological University in 2010. His research interests include discrete event systems, supervisory control, model-based fault diagnosis, multi-agent systems, optimization and scheduling with applications in green buildings, flexible manufacturing, power management and intelligent transportation systems. In the aforementioned areas, he has over 140 publications in journals, book chapters, and conference proceedings, and two patents. Dr Su is a senior member of IEEE, an Associate Editor for Automatica, Journal of Discrete Event Dynamic Systems: Theory and Applications, Journal of Control and Decision, and Transactions of the Institute of Measurement and Control, and the Chair of IEEE Control Systems Society Technical Committee on Smart Cities.