Towards Supervisory Control Theory in Tactical Environments: A Stackelberg Game Approach

Bohan Cui, Alessandro Giua and Xiang Yin*†

May, 2025

Abstract

In this paper, we propose a new framework for supervisory control of discrete-event systems in tactical environments. In contrast to the standard supervisory control theory, where the environments are considered fully adversarial, we consider the possibility of the presence of attackers who have their own objectives that may not necessarily be in opposition to the specification of the supervisor. We formulate this scenario as a Stakelberg game in the leader-follower setting, where the designer proposes a supervisor, and the attacker takes a best response to the supervisor. We characterize the solution to the Stakelberg supervisory control problem as having both cooperative and antagonistic solutions. Moreover, we provide an effective algorithm for synthesizing a cooperative supervisor that enables both players to achieve their objectives. Our work makes an initial step forward from the traditional zero-sum setting of supervisory control theory to the non-zero-sum setting. Examples are provided to illustrate our results.

Published as:
B. Cui, A. Giua, and X. Yin, "Towards Supervisory Control Theory in Tactical Environments: A Stackelberg Game Approach." *62nd IEEE Conference on Decision and Control (CDC)*, pp. 7937-7943, December, 2023.
DOI: 10.1109/CDC49753.2023.10384094

*This work was supported by the National Natural Science Foundation of China (62061136004, 62173226, 61833012).

†Bohan Cui and Xiang Yin are with Department of Automation and Key Laboratory of System Control and Information Processing, Shanghai Jiao Tong University, Shanghai 200240, China. {bohan_cui, yinxiang}@sjtu.edu.cn. Alessandro Giua is with the Department of Electrical and Electronic Engineering, University of Cagliari, Cagliari 09123, Italy. giua@unica.it.

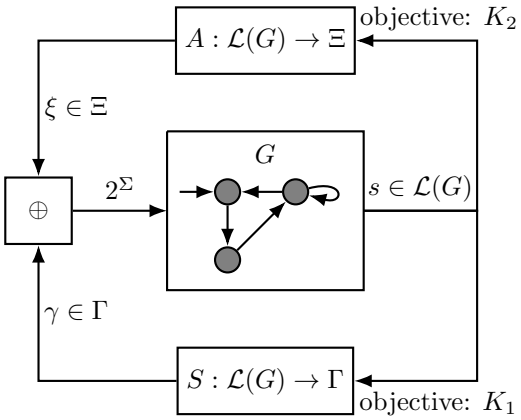


Figure 1: Conceptual illustration of the supervisory control theory under tactical environments.

1 Introduction

Discrete-Event Systems (DES) are an important class of systems characterized by their discrete state spaces and event-triggered dynamics [6]. They play a crucial role in modeling, analyzing, and controlling the high-level logic behaviors of complex systems such as intelligent manufacturing systems, embedded software, and autonomous robots. In the context of DES, one of the most important problem is to enforce the closed-loop properties of the system. Supervisory Control Theory (SCT), initiated by Ramadge and Wonham [17], is one of the most widely used formal frameworks for controller synthesis of DES and has been extensively studied in the past few decades; see, e.g., the textbooks [22] and some recent developments [1, 8–11, 18, 26].

In the context of SCT, it is typically assumed that the system has a language for expressing its desired behavior, known as a *specification*. Additionally, to account for the uncertainty of the environment, some events are considered to be *uncontrollable*, meaning that their occurrences cannot be prevented by the supervisor. The supervisor is then designed using a worst-case analysis, which ensures that the specification can be achieved regardless of the environment's actions. This standard SCT setting can be viewed as a *zero-sum game*, where the environment player aims to violate the supervisor's specification, while the supervisor player attempts to counter this by preventing such violations.

The emergence of networked control architecture nowadays has led to an increased demand for information transmission in the implementation of supervisory control. However, this also makes sensors and actuators vulnerable to *attacks*. As a result, there has been a growing interest in the development of *resilient supervisory control* strategies that can withstand such attacks. Recent research efforts have been devoted to the modeling and synthesis of resilient supervisors in the presence of attackers, as demonstrated by studies such as [2, 5, 12, 14, 15, 19–21, 23, 24]. In this context, the design process requires the consideration of an additional player, namely the *attacker player*, in addition to the supervisor and environment players in the standard SCT. As a result, the design of the supervisor may become more conservative since it must consider the potential damage that an attacker can inflict.

The majority of existing works on supervisory control under attack still adopt the zero-sum approach of the standard SCT framework. In this approach, the supervisor is designed to ensure that the system satisfies its specifications regardless of the behavior of the attacker. While this is reasonable for robustness analysis, it fails to account for the differences between attackers and passive environments. Unlike passive disturbances, attackers usually have their own objectives and motivations for launching an attack, and simply failing the supervisor may not be their only goal. Therefore, it is more appropriate to consider a *non-zero sum* setting, in which the attacker is assumed to be rational and seeking to maximize its own utility. This approach takes into account the potential gains that an attacker may derive from their actions and enables the supervisor to make more informed decisions to achieve its objectives while mitigating the effects of the attacker's actions.

Motivated by the above observations, in this paper, we propose a novel framework for supervisory control of DES under the presence of attackers. We adopt a leader-follower setting, specifically the *Stackelberg game*, where the supervisor is designed first. Once the supervisor is synthesized, its implementation becomes public information, and the attacker seeks to maximize its own utility by taking a best response. Our approach models the objectives of both the supervisor and the attacker using two different prefixed-closed specification languages. In addition to satisfying their respective specifications, both entities also strive to ensure the liveness of the system and maintain a high degree of permissiveness. We refer to this problem formulation as the Stackelberg supervisory control problem (SCCP), as illustrated in Figure 1. We then consider a specific type of attack known as the disablement attack (DA), and demonstrate that SCCP-DA may have two incomparable solutions: cooperative or antagonistic. We propose a sound and complete algorithm for synthesizing a cooperative solution that enables both the supervisor and the attacker to achieve their respective objectives.

Our work is related to non-zero-sum graph games that have been studied in the context of reactive synthesis [3]. For instance, in [7], the co-synthesis problem was investigated, and the notion of assume-guarantee synthesis was proposed, where two processes compete conditionally. This idea has also been explored in the context of decentralized synthesis problems [13]. Additionally, in [4], the authors studied Stackelberg games played on graphs from a new perspective where the followers have several goals, and the best response is defined by a Pareto-optimal payoff. A similar idea has been employed in solving the minimum violation problem on stochastic systems [16], where the globally optimal solution with numeric utilities was investigated. However, to the best of our knowledge, the context of Stackelberg games has not yet been considered in the context of SCT of DES.

2 Preliminary

2.1 System Model

Let Σ be a finite set of events. A string is a finite sequence of events and Σ^* denotes the set of all strings over Σ including the empty string ϵ. For any string $s \in \Sigma^*$, $|s|$ denotes the length of s with $|\epsilon| = 0$. A language $L \subseteq \Sigma^*$ is a set of strings and we denote by $\bar{L}$ the prefix-closure of language L, i.e., $\bar{L} = \{s \in \Sigma^* : \exists w \in \Sigma^* \text{ s.t. } sw \in L\}$. We say that a language is prefix-closed if $L = \bar{L}$. Given language L and a string $s \in L$, we denote the active event set at s in L by $\Delta_L(s) = \{\sigma \in \Sigma : s\sigma \in L\}$. We say language L is live if $\forall s \in L : \Delta_L(s) \neq \emptyset$.

We consider a DES modeled by a deterministic finite-state automaton (DFA)

$$G = (X, \Sigma, \delta, x_0),$$

where X is the finite set of states, Σ is the finite set of events, $\delta : X \times \Sigma \to X$ is the partial transition function such that $\delta(x, \sigma) = x'$ denotes the transition labeled by σ from state x to state x', and $x_0 \in X$ is the unique initial state. The transition function is also extended to $\delta : X \times \Sigma^* \to X$ recursively by: (i) for any $x \in X$, $\delta(x, \epsilon) = x$; and (ii) for any $x \in X, s \in \Sigma^*, \sigma \in \Sigma$, we have $\delta(x, s\sigma) = \delta(\delta(x, s), \sigma)$. The set of all strings generated by G starting from state $x \in X$ is defined as $\mathcal{L}(G, x) = \{s \in \Sigma^* : \delta(x, s)!\}$, where "!" means "is defined". The language generated by G is defined as $\mathcal{L}(G) := \mathcal{L}(G, x_0)$. For simplicity, we write $\delta(x_0, s)$ as $\delta(s)$ for any $s \in \mathcal{L}(G)$.

2.2 Standard Supervisory Control Theory

As we mentioned above, in the standard framework of supervisory control theory, the event set is partitioned as

$$\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc},$$

where Σ_c is the set of controllable events and Σ_{uc} is the set of uncontrollable events. A supervisor is a mechanism that dynamically disables controllable events in order to enforce some specifications. We denote $\Gamma = \{\gamma \in 2^\Sigma : \Sigma_{uc} \subseteq \gamma\}$ as the set of admissible control decisions. Then a supervisor is a function $S : \mathcal{L}(G) \to \Gamma$. We use notation S/G to represent the controlled system and the language generated by S/G, denoted by $\mathcal{L}(S/G)$ is defined recursively by:

(i) $\epsilon \in \mathcal{L}(S/G)$;

(ii) $[s \in \mathcal{L}(S/G) \wedge s\sigma \in \mathcal{L}(G) \wedge \sigma \in S(s)] \Leftrightarrow [s\sigma \in \mathcal{L}(S/G)]$.

The specification of the system is a prefix-closed sub-language $\overline{K_1} = K_1 \subseteq \mathcal{L}(G)$. The standard safety control problem is to design a supervisor S such that $\mathcal{L}(S/G) \subseteq K_1$ and as permissive as possible.

3 Supervisory Control under Tactical Environments

In this section, we first present the model of attackers, then we formulate the supervisory control problem under tactical environments and provide examples to illustrate our problem setting and the underlying challenges.

3.1 General Model of Attackers

Similar to the supervisor, we model an attacker as a mechanism that dynamically makes attack decisions to achieve its own goals. Let Ξ be a set of symbols representing attack actions and ψ be a new symbol presenting that "no attack" is launched. An *attacker* is a function

$$A : \mathcal{L}(G) \to \Xi \cup \{\psi\}$$

that makes attack decisions based on the history. Then at each instant, the actual set of events enabled in the system is determined together by the decisions of the supervisor and the attacker under certain fusion mechanism. We use function

$$\oplus : \Gamma \times (\Xi \cup \{\psi\}) \to 2^{\Sigma}$$

to represent the *fusion rule* of the supervisor and the attacker, where for any $\gamma \in \Gamma$, we have $\oplus(\gamma, \psi) = \gamma$. We say an attacker A is *non-interfering* if $\forall s \in \mathcal{L}(G) : A(s) = \psi$ and denote this attacker strategy by A_ψ.

Given supervisor S, we define $S_{A,\oplus}$ as the resulting overall supervisor under attacker A with fusion rule $\oplus$. Specifically, this overall supervisor is defined by:

$$\forall s \in \mathcal{L}(G) : S_{A,\oplus}(s) = \oplus(S(s), A(s))$$

and we denote by $\mathcal{L}(S_{A,\oplus}/G)$ the language generated by the system under attack. Clearly, for non-interfering attacker A, we always have $\mathcal{L}(S_{A,\oplus}/G) = \mathcal{L}(S/G)$.

3.2 Stackelberg Game Formulation

To formalize the *tactical behaviors* of the attacker, we consider the following *leader-follower* setting:

- The supervisor is designed first and the implementation of supervisor $S : \mathcal{L}(G) \to \Gamma$ is a public information;
- The attacker then determines its strategy $A : \mathcal{L}(G) \to \Xi \cup \{\psi\}$ based on the existing supervisor S, or chooses to not interfere at all.

In the context of game theory, the above leader-follower setting is referred to as a two-player *Stackelberg game*. In this game, since the supervisor is fixed first, a rational attacker can only seek its best response to achieve its own objective. Similarly, when designing the supervisor to achieve its own objective, it is assumed that the attacker will not deviate from its best response. This assumption enables the supervisor to anticipate the attacker's behavior and design a strategy that maximizes its own objective while taking into account the attacker's potential moves.

Formally, we assume the attacker has its own objective that may not necessarily be opposed to the objective of the supervisor. Specifically, we denote by $\overline{K_2} = K_2 \subseteq \mathcal{L}(G)$ the prefix-closed specification language of the attacker. When the supervisor S is given, the objective of the attacker is to interfere with the supervisor in such a way that the overall system behavior remains within K_2, while ensuring that the system's liveness is not affected.

Definition 1 (Successful Attacks) *Given system G and supervisor S, we say $A : \mathcal{L}(G) \to \Xi \cup \{\psi\}$ is a* successful attack strategy *w.r.t. S, $\oplus$ and K_2 if it satisfies the following conditions:*

C1 $\mathcal{L}(S_{A,\oplus}/G) \subseteq K_2$*; and*

C2 $\mathcal{L}(S_{A,\oplus}/G)$ *is live; and*

C3 For any A' satisfying C1 and C2, we have $\mathcal{L}(S_{A,\oplus}/G) \not\subset \mathcal{L}(S_{A',\oplus}/G)$.

It is important to note that in some cases, successful attackers may not be possible. In such situations, a rational attacker would choose not to interfere since it would not help achieve its objective. This concept of rational behavior leads to the definition of best responses.

Definition 2 (Best Responses) *Given system G and supervisor S, we say $A : \mathcal{L}(G) \to \Xi \cup \{\psi\}$ is a* best response *of the attacker against supervisor S if it is a successful attack strategy. If no successful attack is possible, then the best response is the non-interfering attack strategy A_ψ. We denote by $\mathbf{BR}(S)$ the set of best responses against supervisor S.*

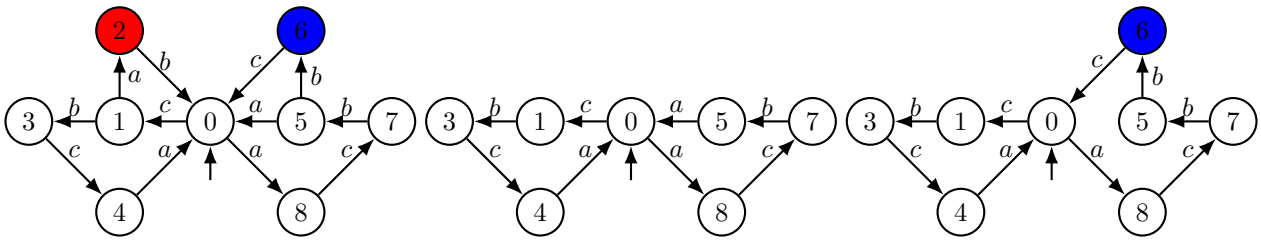
Since the supervisor assume that the attacker will take a best response, the supervisor needs to propose a strategy such that its own objective K_1 as well as liveness can always be achieved under any best responses. This leads to the following Stackelberg Supervisory Control Problem (SSCP).

Problem 1 (SSCP) *Given system G, specifications $K_1, K_2 \subseteq \mathcal{L}(G)$, synthesize a supervisor S, such that:*

C1 For any $A \in \mathbf{BR}(S)$, we have $\mathcal{L}(S_{A,\oplus}/G) \subseteq K_1$;

C2 For any $A \in \mathbf{BR}(S)$, we have $\mathcal{L}(S_{A,\oplus}/G)$ is live;

C3 "as permissive as possible" when satisfying C1 and C2.



(a) System G. (b) The controlled system for cooperating. (c) The controlled system for antagonizing.

Figure 2: An example of Stakelberg supervisory control problem, where $\Sigma_c = \{a\}$ and $\Sigma_a = \{b\}$.

Intuitively, C1 and C2 require that the proposed supervisor should be safe w.r.t. K_1 and live whenever the attacker reacts rationally. Regarding "as permissive as possible" in C3, we do not provide a mathematical definition here since it may not always be well-defined. In the next section, we will provide a concrete definition of permissiveness for a specific instance of the general problem formulation.

4 Case of Disablement Attackers

The previous section formulates a very general Stackelberg supervisory control problem. In particular, the attack decision set Ξ and the fusion rule $\oplus$ are generic without any physical meaning. Depending on the specific setting of the system, they can be further concretized.

4.1 Disablement Attackers

Hereafter in this paper, we will focus a concrete type of attacker called *disablement attackers* [5]. In the setting, the attacker can further disable some events that are originally enabled by the supervisor. Specifically, we assume that

$$\Sigma_a \subseteq \Sigma$$

is the set of events that can be disabled by the attacker and we denote $\Sigma_{ua} = \Sigma \setminus \Sigma_a$. The the action space of the attacker for this setting can be concretized by

$$\Xi = \{\xi \in 2^{\Sigma} : \Sigma_{ua} \subseteq \xi\}.$$

The fusion rule $\oplus$ then boils down to the set intersection $\cap$, i.e., for any control decision $\gamma \in \Gamma$ and attack decision $\xi \in \Xi$, we have $\oplus(\gamma, \xi) = \gamma \cap \xi$. Furthermore, the "no attack" decision ψ can also be replaced by $\psi = \Sigma$, i.e., the attacker disables nothing. By understanding the concrete meaning of Ξ and $\oplus = \cap$, hereafter, the overall supervisor $S_{A,\cap}$ will be simplified as S_A.

4.2 Properties of Disablement Attackers

Note that, given a supervisor S, the set of its best responses can be of form

- $\mathbf{BR}(S) = \{A_\psi\}$; or
- $\mathbf{BR}(S) = \{A_1, \ldots, A_n\}$, where $A_i \neq A_\psi$.

In the first case, we say that the supervisor is *antagonistic* since it does not leave any opportunity for the attacker to achieve its objective K_2, and thus, the attacker gives up. In the second case, however, we say that the supervisor is *cooperative* since it allows the attacker to interfere with the system without harming the objective K_1 of the supervisor.

In general, when the supervisor is cooperative, its best response may not be unique. This is why "permissiveness" may not be well-defined in Problem 1. However, the following result shows that, for disablement attacks, the best response is always unique in terms of the generated language.

Proposition 1 *Given supervisor S, for any best responses $A, A' \in \mathbf{BR}(S)$, we have $\mathcal{L}(S_A/G) = \mathcal{L}(S_{A'}/G)$.*

Based on the above result, we know that the best response against supervisor S is always unique (in the sense of the generated language), which is either A_ψ for the antagonistic case, or an arbitrary strategy in $\mathbf{BR}(S)$ for the cooperative case. Therefore, hereafter, we denote by $S_{\mathbf{BR}} = S_A$ the overall supervisor under the unique best response A.

With the above properties for the case of disablement attacks (DA), the general formulation of SSCP in Problem 1 can be further concretized as SSCP-DA as follows.

Problem 2 (SSCP-DA) *Given system G, specifications $K_1, K_2 \subseteq \mathcal{L}(G)$, synthesize a supervisor S, such that:*

C1 $\mathcal{L}(S_{\mathbf{BR}}/G) \subseteq K_1$;

C2 $\mathcal{L}(S_{\mathbf{BR}}/G)$ *is live;*

C3 For any S' satisfying C1 and C2, we have $\mathcal{L}(S_{\mathbf{BR}}/G) \not\subset \mathcal{L}(S'_{\mathbf{BR}}/G)$.

Note that we can formulate "permissiveness" as C3, since the overall supervisor under rational attack is unique. Without Proposition 1, this definition may be problematic.

Without loss of generality, we assume that the specification language $K_i, i = 1, 2$ is generated by $H_i = (X_{H_i}, \Sigma, \delta_{H_i}, x_{H_i,0})$, which is a *sub-automaton* of G. Then for any $s \in \mathcal{L}(G)$, we have $s \in K_i$ iff it does not pass through any states in $X \setminus X_{H_i}$.

4.3 Illustrative Examples and Potential Challenges

Before we present the solution to SSCP-DA. We first present two examples to illustrate the problem formulation as well as its potential challenges.

Example 1 (Non-Uniqueness of the Solution) *Let us consider system G shown in Figure 2(a), where we have $\Sigma_c = \{a\}$ and $\Sigma_a = \{b\}$. For the supervisor, the specification K_1 is to avoid reaching red state 2, and for the attacker, the specification K_2 is to avoid reaching blue state 6.*

One can easily find a cooperative supervisor S_{cop} that only disables event a at state 1. Once this supervisor is posted, the best response of the attacker is to disable event b at state 5. Then the overall controlled system $S_{\mathbf{BR}}/G$ is shown in Figure 2(b), where both supervisor and attacker achieve their goals K_1 and K_2, respectively. This supervisor is indeed a solution to SSCP-DA. However, one can find an antagonistic supervisor S_{ant} that is incomparable with the above cooperative solution S_{cop}. Specifically, one can consider a supervisor S_{ant} that not only disables event a at state 1, but also disables event a at state 5. Once this supervisor is posted, the rational attacker will not choose to disable event b at state 5 anymore since this will block the entire system. Therefore, the best response to such a supervisor is the non-interfering attacker. In this case, the overall controlled system $S_{\mathbf{BR}}/G$ is shown in Figure 2(c), where only the supervisor achieve its objective K_1. Clearly, S_{cop} and S_{ant} are incomparable, although S_{cop} is already a solution to SSCP-DA.

In the above example, we have show that for a cooperative solution, there may exists an incomparable antagonistic solution. The following result further shows that a maximally permissive antagonistic solution may not even exist.

Example 2 (Non-Existence of Antagonistic Solution) *We still consider the system in Example 1. In fact, we can make S_{ant} in the above example more permissive as follow. We define a supervisor S^k_{ant} such that it enables event a for the first k times visiting state 5 and choose to disable event a at state 5 when this state is visited for more that k times. We note that, for any k, $\mathbf{BR}(S^k_{ant}) = \{A_\psi\}$ since if the attacker tries to avoid reaching state 6 by disabling event b at state 5 after the kth visit of state 5, then the system will be blocked. Furthermore, S^{k+1}_{ant} is always strictly more permissive than S^k_{ant}. Therefore, a maximally permissive antagonistic solution does not even exist.*

Based on the above discussions, hereafter, we will focus on finding a cooperative solution to SSCP-DA. How to find a reasonably permissive antagonistic solution is left for our future work.

5 Synthesis of Cooperative Solutions

In this section, we discuss how to synthesis a supervisor in a tactical environment. Specifically, the synthesized supervisor is cooperative in the sense that its best response attacker is interfering and can also achieve its own objective.

5.1 Attack-Decision Structures

The SSCP involves three key players: the supervisor, the attacker, and the environment. Specifically, the *supervisor* is the first to propose a decision, which is then followed by the *attacker*'s best response. Finally, the *environment* selects an actual event that is permitted by the combination of the first two players' decisions. To capture this three-stage process, the plant is unfolded by incorporating the decisions of the supervisor and the attacker into it.

Definition 3 (Attack-Decision Structure) *An* attack decision structure *(ADS) w.r.t. G is defined by*

$$T = (X^T, \hat{X}^T, h^T_{DE}, h^T_{ED}, \Sigma, \Gamma, \Xi, x^T_0),$$

where

- $X^T \subseteq X$ *is the set of* decision states *(D-states), which is a subset of plant states;*
- $\hat{X}^T \subseteq X \times \Gamma \times \Xi$ *is the set of* environment states *(E-states), which is a subset of plant states augmented with control and attack decisions;*
- $h^T_{DE} : X \times (\Gamma \times \Xi) \to \hat{X}$ *is the partial transition function from D-states to E-states such that: for any $x \in X^T, \gamma \in \Gamma, \xi \in \Xi$ and $\hat{x} = (x', \gamma', \xi') \in \hat{X}^T$, we have*

$$h^T_{DE}(x, (\gamma, \xi)) = \hat{X} \Rightarrow (x', \gamma', \xi') = (x, \gamma, \xi)$$

- $h^T_{ED} : \hat{X} \times \Sigma \to X^T$ *is the partial transition function from E-states to D-states such that: for any $\hat{x} = (x, \gamma, \xi) \in \hat{X}^T, \sigma \in \Sigma$ and $x' \in X^T$, we have*

$$h^T_{ED}(\hat{x}, \sigma) = x' \Rightarrow [\sigma \in \oplus(\gamma, \xi)] \wedge [x' = \delta(x, \sigma)]$$

- Σ *is the set of events of G, Γ is the set of admissible control decisions of the supervisor and Ξ is the set of attack decisions;*
- $x^T_0 = x_0 \in X^T$ *is the initial D-state.*

Intuitively, a D-state $x \in X^T$ is just a plant state at which both the supervisor and the attack make decisions $(\gamma, \xi) \in \Gamma \times \Xi$. Then the system moves to E-state $\hat{x} = (x, \gamma, \xi)$ by simply remembering the decisions. Then at this E-state, only events allowed by both players can occur, and once $\sigma \in \oplus(\gamma, \xi)$ occurs, the system moves to D-state $x' = \delta(x, \sigma)$ following the dynamic of the plant.

Given an ADS T, we define the set of decisions at D-state $x \in X^T$ as $D_T(x) := \{(\gamma, \xi) \in \Gamma \times \Xi : h^T_{DE}(x, (\gamma, \xi))!\}$. Then for the purpose of control, we need to require the supervisor and the attacker can always react to any events that occurred. Formally, given an ADS T, we say

- a D-state $x \in X^T$ is *complete* in T if $D_T(x) \neq \emptyset$;
- an E-state $\hat{x} = (x, \gamma, \xi) \in \hat{X}^T$ is *complete* in T if for any $\sigma \in \oplus(\gamma, \xi)$, we have $\delta(x, \sigma)! \Rightarrow h^T_{ED}(\hat{x}, \sigma)!$.

We say ADS T is complete if all states in it are complete.

Note that the ADS contains multiple control and attack decisions in general. Given a supervisor S and an attacker A, the corresponding D- and E-state reached upon the string $s \in \mathcal{L}(S_A/G)$ can be computed recursively by applying the corresponding control and attack decisions along string s, which we denote as $X^T_{S_A}(s)$ and $\hat{X}^T_{S_A}(s)$, respectively.

Definition 4 (Included Supervisors) *Given a supervisor S and an attacker A, we say an overall supervisor S_A is* included *in ADS T if for any $s \in \mathcal{L}(S_A/G)$, we have $(S(s), A(s)) \in D_T(X^T_{S_A}(s))$. Also, we say a supervisor S is included in ADS T if there exists an attacker A such that the overall supervisor S_A is included in ADS T.*

5.2 Characterizing Attacker's Best Response

As we mentioned above, in this work, we seek for finding a cooperative solution. Therefore, we already assume that the attacker can achieve its objective K_2, which means that we can restrict the state space of T to X_{H_2} Furthermore, in addition to the satisfication of K_2, the attacker also concerns about the liveness of the controlled system.

Definition 5 (Live States) *Given an ADS T, an E-state $\hat{x} = (x, \gamma, \xi) \in \hat{X}^T$ is said to be* live *if there exists an event $\sigma \in \oplus(\gamma, \xi)$ such that $\delta(x, \sigma)!$.*

The following result connects the liveness of the overall controlled system and live states.

Proposition 2 *Given system G and supervisor S_A, the overall controlled system S_A/G is live if and only if for any $s \in \mathcal{L}(S_A/G)$, the E-state encountered $\hat{X}^T_{S_A}(s)$ is live.*

Now, we introduce the *all cooperative structure* (ACS) that contains all cooperative solutions.

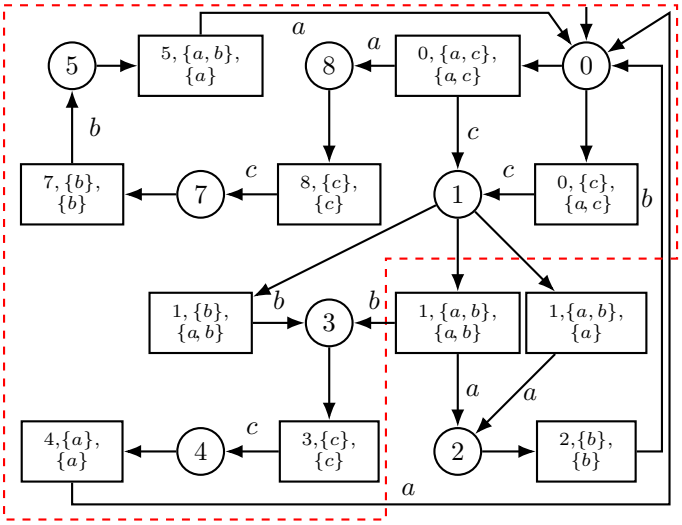


Figure 3: The ACS, where D-states and E-states are represented by circle and rectangular states, respectively.

Definition 6 (All Cooperative Structures) *Given system G and attack specification H_2, the all cooperative structure (ACS), denote by*

$$\mathsf{ACS} = (X^{ACS}, \hat{X}^{ACS}, h_{DE}^{ACS}, h_{ED}^{ACS}, \Sigma, \Gamma, \Xi, x_0^{ACS}),$$

is the largest complete ADS w.r.t. G such that

1) $X^{ACS} \subseteq X_{H_2}$; and

2) For any $\hat{x} \in \hat{X}^{ACS}$, E-state $\hat{x}$ is live.

By "largest", we mean that for any complete ADS T satisfying 1) and 2), we have $T \sqsubseteq \mathsf{ACS}$, where "$\sqsubseteq$" denotes the standard sub-graph inclusion.

Remark 5.1 *The ACS construction procedure is similar to the construction of the all inclusive controller in [25]. We refer the reader to [25] for details on the construction algorithm.*

We use the following example to illustrate the ACS.

Example 3 *Again we consider system G shown in Fig 2(a). An example of the ACS is as shown in Fig 3, where circle states represent D-states, rectangular states represent E-states. Take a part of this figure as an example, at D-state 5, the supervisor has two possible decisions $\{a, b\}$ and $\{b\}$, and the attacker can make decisions $\{a, b\}$ or $\{a\}$. Thus, in the largest ADS T_{total}, there should be four decision-pairs at state 5. However, for the cases $\{a, b\} \cap \{a, b\}$ or$\{b\} \cap \{a, b\}$, since event b is feasible, the system can go into D-state 6, which is in X_{H_2}. Therefore, E-states $(5, \{a, b\}, \{a, b\})$ and $(5, \{b\}, \{a, b\})$ should be removed when constructing the ACS. On the other hand, in case of $\{b\} \cap \{a\}$, the system will be blocked since no out-going event is feasible. Therefore, there is only one transition defined at state 5 in the ACS as shown in Figure 3.*

The following theorem shows that the ACS indeed and only contains all cooperative supervisors.

Theorem 1 *For any supervisor S and attacker A, the following two statements are equivalent:*

1) $\mathcal{L}(S_A/G) \subseteq K_2$ and $\mathcal{L}(S_A/G)$ is live;

2) S_A is included in ACS.

Given a cooperative supervisor S in the ACS, since the attacker also wants to be maximally permissive, at each state, once the supervisor makes a control decision, it will pick a locally maximal decision within the ACS. Formally, for any D-state $x \in X^{ACS}$ and control decision $\gamma \in \Gamma$, we define

$$\textsc{LocMax}(x, \gamma) = \left\{ \xi \in \Xi : \begin{array}{c} (\gamma, \xi) \in D_{ACS}(x) \wedge \\ [\forall (\gamma, \xi') \in D_{ACS}(x) : \xi \not\subset \xi'] \end{array} \right\}$$

as the set of locally maximal attacker decisions at x when the supervisor's decision is γ. In fact, $\textsc{LocMax}(x, \gamma)$ is a singleton in the sense of non-redundant decisions when we consider the disablement attackers.

Then the best response of the attacker is characterized by the following result.

Proposition 3 *Let S be a cooperative supervisor such that $\mathbf{BR}(S) = \{A\}$, where $A \neq A_\psi$. Then for any $s \in \mathcal{L}(S_A/G)$, we have $A(s) \in \textsc{LocMax}(X_{S_A}^{ACS}(s), S(s))$.*

5.3 Synthesis of Cooperative Supervisors

Moving on to the synthesis of cooperative supervisors, as discussed earlier, when given a cooperative supervisor, the attacker always seeks to pick a locally maximal attack decision based on the supervisor's decision. As a result, to ensure the satisfaction of its own specification K_1 (or X_{H_1} in terms of states), the supervisor needs to anticipate the "rational" behavior of the attacker. This motivates us to introduce the following definition.

Definition 7 (Safe Control Decisions) *Given the ACS w.r.t. H_2, a control decision $\gamma \in \Gamma$ is said to be safe at D-state $x \in X^{ACS}$ w.r.t. H_1 if for any maximal attack decision $\xi \in \textsc{LocMax}(x, \gamma)$ and any feasible event $\sigma \in \oplus(\gamma, \xi)$*

$$h_{ED}^{ACS}(h_{DE}^{ACS}(x, (\gamma, \xi)), \sigma) \in X_{H_1}$$

Note that the above condition is equivalent to $\delta(x, \sigma) \in X_{H_1}$.

The following result shows that, from the supervisor's perspective, to enforce its own specification K_1, it always needs to make safe control decisions.

Proposition 4 *Let S be a cooperative supervisor such that the attacker can achieve K_2, then the supervisor achieves K_1, i.e., $\mathcal{L}(S_{\mathbf{BR}}/G) \subseteq K_1$ if and only if*

$$\forall s \in \mathcal{L}(S_{\mathbf{BR}}/G) : S(s) \text{ is safe at } \delta(s).$$

Intuitively, a control decision is safe if it cannot reach an unsafe state outside of X_{H_2} in one step when the attack responses in a maximally permissive manner. However, to ensure safety of S, the above result states that the control decision needs to be safe *globally*. Therefore, to decode cooperative supervisor from the ACS, we need to further find the largest complete sub-system of the ACS such that 1) all control decisions are safe. To find T^*, we need to first remove all transitions $h_{DE}^{ACS}(x, (\gamma, \xi))$ such that γ is not safe at x. However, this may introduce new incomplete states, and we still need to iteratively remove incomplete state until the ADS becomes complete. If the T^* is non-empty, we can decode the maximally permissive cooperative supervisor S^* from it by picking the locally maximal control and attack decisions for each reachable D-states x, i.e.,

$$\textsc{LocMax}(x) = \left\{ \gamma \in \Gamma : \begin{array}{c} ((\gamma, \cdot) \in D_{T^*}(x) \wedge \\ [\forall (\gamma', \cdot) \in D_{T^*}(x) : \gamma \not\subset \gamma'] \end{array} \right\}$$

and picking all feasible events for each reachable E-states. This will result an ADS T containing one supervisor S_T, which is our solution. The complete synthesis algorithm is formally described by Algorithm 1. Similar to the case of $\textsc{LocMax}(x, \gamma)$, for disablement attacks, we also have that $\textsc{LocMax}(x)$ a singleton in the sense of non-redundant decisions.

Algorithm 1: Cooperative Supervisor Synthesis

Input: system G and trim automatons H_1 and H_2
Output: S^*

construct ACS
construct the largest complete sub-system of ACS in which all control decisions are safe, denoted by T^*
if $T^* = \emptyset$ **then**
    return no solution exists
else
    $X^T = \{x_0\}$, $\hat{X}^T = \emptyset$, $h^T = \emptyset$
    Expand(T, T^*, x_0)
    return S_T

procedure Expand(T, T^*, x)
$\gamma \in \textsc{LocMax}(x)$, $\xi \in \textsc{LocMax}(x, \gamma)$
$h^T = h^T \cup \{(x, \gamma, \xi, \hat{x})\}$
if $\hat{x} \notin \hat{X}^T$ **then**
    $\hat{X}^T = \hat{X}^T \cup \{\hat{x}\}$
    for $\sigma \in \oplus(\gamma, \xi)$ **do**
        $x' = h_{ED}^{T^*}(\hat{x}, \sigma)$, $h^T = h^T \cup \{(\hat{x}, \sigma, x')\}$
        if $x' \notin X^T$ **then**
            $X^T = X^T \cup \{x'\}$
            Expand(T, T^*, x')

Now we show the correctness of our algorithm.

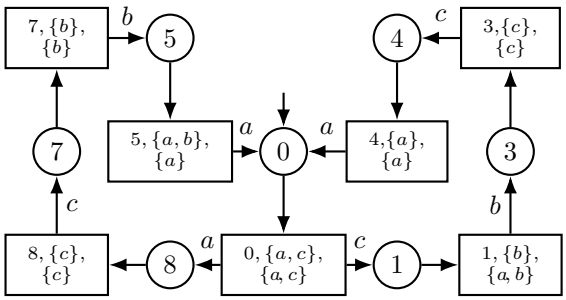


Figure 4: The maximally permissive cooperative supervisor S^*.

Theorem 2 *Algorithm 1 is both sound and complete*

We illustrate Algorithm 1 by the following example.

Example 4 *We still return to our running example. The ACS has been shown in Fig 3. We note that control decision* $\{a, b\}$ *is not safe at state* 1 *because there exists an event* $a \in \{a, b\} \cap \{a, b\}$ *leading to state* $2 \notin X_{H_1}$ *under the locally maximal attack decision* $= \{a, b\} \in \textsc{LocMax}(1, \{a, b\})$*. Therefore, we remove transitions* $h^{ACS}_{DE}(1, \{a, b\}, \{a, b\})$ *and* $h^{ACS}_{DE}(1, \{a, b\}, \{a\})$*. This removal does not introduce any new incomplete state and we obtain* T^* *as shown in the box marked by red lines. Now we synthesize a supervisor as follows. Starting from state* 0*, the algorithm chooses a maximally permissive control decision* $\{a, c\}$*, which will be responded by attack decision* $\{a, c\}$ *as shown in* T^**. By this choice, the system moves to E-state* $(0, \{a, c\}, \{a, c\})$*, and we need to consider all successor D-states* 1 *and* 8*, at which the locally maximal control decision are* $\{c\}$ *and* $\{b\}$*, respectively, and so forth. The result structure is shown in Figure 4, which essentially gives us a cooperative supervisor and its best response attacker. This is consistent with our earlier discussion as shown in Fig 2(b).*

6 Conclusion

In this paper, we introduced a novel framework for supervisory control in tactical environments, where attacks with their own objectives may not be fully opposed to the control objective. We formulated this scenario as a leader-follower Stackelberg game and proposed an algorithm for synthesizing a cooperative supervisor that enables both players to achieve their objectives, while the supervisor has the priority to maximize its objective. There are several promising directions for future research within this framework. Firstly, instead of cooperative solutions, we plan to investigate how to synthesize a reasonably permissive antagonistic solution. Secondly, we aim to extend our results to different types of attacks beyond disablement attacks. Lastly, we note that in this work, the objectives of both the supervisor and the attacker are captured by safety. However, in the future, we plan to consider non-prefix-closed specification languages for both players.

References

[1] M. VS Alves, L. Carvalho, and J. Basilio. Supervisory control of networked discrete event systems with timing structure. *IEEE Transactions on Automatic Control*, 66(5):2206–2218, 2020.

[2] J. Basilio, C.N. Hadjicostis, and R. Su. Analysis and control for resilience of discrete event systems: Fault diagnosis, opacity and cyber security. *Foundations and Trends® in Systems and Control*, 8(4):285–443, 2021.

[3] R. Brenguier, L. Clemente, P. Hunter, G. Pérez, M. Randour, J.-F. Raskin, O. Sankur, and Ma. Sassolas. Non-zero sum games for reactive synthesis. In *LATA*, pages 3–23, 2016.

[4] V. Bruyère, J.-F. Raskin, and C. Tamines. Stackelberg-pareto synthesis. In *CONCUR*, 2021.

[5] L. Carvalho, Y.-C. Wu, R. Kwong, and S. Lafortune. Detection and mitigation of classes of attacks in supervisory control systems. *Automatica*, 97:121–133, 2018.

[6] C. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Springer, 2008.

[7] K. Chatterjee and T.A. Henzinger. Assume-guarantee synthesis. In *TACAS*, pages 261–275, 2007.

[8] Y. Ji, X. Yin, and S. Lafortune. Optimal supervisory control with mean payoff objectives and under partial observation. *Automatica*, 123:109359, 2021.

[9] Jan Komenda and Tomáš Masopust. Hierarchical supervisory control under partial observation: Normality. *IEEE Transactions on Automatic Control*, 2023.

[10] Z. Liu, X. Yin, S. Shu, F. Lin, and S. Li. Online supervisory control of networked discrete event systems with control delays. *IEEE Transactions on Automatic Control*, 67(5):2314–2329, 2022.

[11] Z. Ma and K. Cai. Optimal secret protections in discrete-event systems. *IEEE Transactions on Automatic Control*, 67(6):2816–2828, 2021.

[12] Z. Ma and K. Cai. On resilient supervisory control against indefinite actuator attacks in discrete-event systems. *IEEE Control Systems Letters*, 6:2942–2947, 2022.

[13] R. Majumdar, K. Mallik, A.-K. Schmuck, and D. Zufferey. Assume–guarantee distributed synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):3215–3226, 2020.

[14] R. Meira-Góes, S. Lafortune, and H. Marchand. Synthesis of supervisors robust against sensor deception attacks. *IEEE Transactions on Automatic Control*, 66(10):4990–4997, 2021.

[15] R. Meira-Góes and S. Marchand, H.and Lafortune. Dealing with sensor and actuator deception attacks in supervisory control. *Automatica*, 147:110736, 2023.

[16] J. Niu, L.and Fu and A. Clark. Optimal minimum violation control synthesis of cyber-physical systems under attacks. *IEEE Transactions on Automatic Control*, 66(3):995–1008, 2020.

[17] P. Ramadge and W. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal on Control and Optimization*, 25(1):206–230, 1987.

[18] S. Shu and F. Lin. Predictive networked control of discrete event systems. *IEEE Transactions on Automatic Control*, 62(9):4698–4705, 2016.

[19] R. Tai, L. Lin, and R. Su. Synthesis of optimal covert sensor–actuator attackers for discrete-event systems. *Automatica*, 151:110910, 2023.

[20] M. Wakaiki, P. Tabuada, and J.P. Hespanha. Supervisory control of discrete-event systems under attacks. *Dynamic Games and Applications*, 9:965–983, 2019.

[21] Y. Wang and M. Pajic. Supervisory control of discrete event systems in the presence of sensor and actuator attacks. In *IEEE 58th Conference on Decision and Control*, pages 5350–5355, 2019.

[22] W. Wonham and K. Cai. Supervisory control of discrete-event systems, 2019.

[23] Y. Xie, X. Yin, and S. Li. Opacity enforcing supervisory control using nondeterministic supervisors. *IEEE Transactions on Automatic Control*, 67(12):6567–6582, 2022.

[24] J. Yao, X. Yin, and S. Li. On attack mitigation in supervisory control systems: A tolerant control approach. In *59th IEEE Conference on Decision and Control*, pages 4504–4510, 2020.

[25] X. Yin and S Lafortune. Synthesis of maximally permissive supervisors for partially-observed discrete-event systems. *IEEE Transactions on Automatic Control*, 61(5):1239–1254, 2016.

[26] X. Yin and S. Lafortune. A uniform approach for synthesizing property-enforcing supervisors for partially-observed discrete-event systems. *IEEE Transactions on Automatic Control*, 61(8):2140–2154, 2016.