Intrusion Detection in Controlled Discrete Event Systems

David Thorsley and Demosthenis Teneketzis

***Abstract*— The constituent controllers in a supervisory control system may sometimes fail as a result of an intruder interfering with the feedback performance of the system. The intrusion may allow the system to execute traces that the supervisor wishes to prevent from occurring. We derive conditions under which a supervisor can detect the presence of an intruder in time to prevent the execution of an illegal trace. In situations where it is not possible to block all illegal strings, we use a language measure method to assess the damage caused by a particular set of controllers failing. We also use the language measure technique to determine the optimal behavior of the controlled system in the presence of an intrusion.**

I. Introduction

Discrete event systems (DES) are dynamic systems that evolve in response to abrupt changes in the system called events. DES have natural applications in systems such as telecommunications networks [1] and computer systems that are fundamentally discrete in nature, but through abstraction DES models have also been applied to systems such as heating, ventilation, and air conditioning systems [2] and traffic systems [3].

Supervisory control theory [4] has been developed to address many questions related as to how to ensure that a DES will operate in a desired manner. A schematic diagram of a basic supervisory control system in DES is shown in Figure 1. A DES G is partially observed by a supervisor S_P. Based on its knowledge of its past observations and the specification that it is trying to implement, the supervisor chooses a set of events to enable in the DES G. This set of events is enabled by the supervisor and all other events are disabled.

Where our model differs from that of the basic centralized supervisory control problem is in the presence of interference affecting the output that the supervisor feeds back into the DES. There can be multiple interpretations of the cause of this interference: the communication channel between the supervisor and the system may by noisy; some of the controllers may be faulty and unable to execute the supervisor's commands to disable particular events; or a malicious agent may have infiltrated the channel in order to sabotage the system's performance. The approach of this paper is to consider the worst-case damage that can be caused by this interference; for that reason we prefer the interpretation of the interference as being caused by an intruder attempting to damage the system as much as possible. Such an interpretation is particularly useful with respect to communication networks and computer systems, where an unknown number of potentially malicious users have some access to the system.

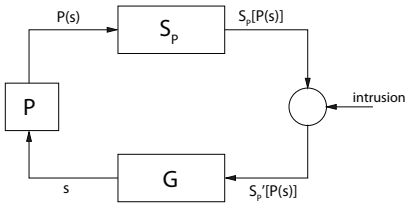

Fig. 1. A supervisory control system with an intruder affecting control actions.

The intrusion detection problem we consider relates somewhat to the problem of diagnosis of DES (for an overview, see [5]), particularly with the idea of safe diagnosability [6]. In safe diagnosability, the objective is to detect the occurrence of failure events before a "catastrophic" event takes place that can damage the system. In one of the problems we consider in this paper, we have a similar goal: to know if the system can detect the presence of interference and prevent an intruder from reaching an undesirable state where damage can be done to the system. The results in this paper were developed from a DES control perspective; this perspective has evolved independently of that of the intrusion detection and intrusion response communities [7]. The distinction between detection and response is not as clear in the DES framework proposed in this paper as it is in those communities.

If the system under consideration is part of a larger communication or computer network (e.g., the Internet), designing the system with the knowledge that intruders may interfere with its performance is essential. Allowing the system to execute tasks of little reward may result in large penalties; for example, on the Internet, allowing attachments to e-mails to run automatically saves a small amount of time at best, but allows viruses to cause a lot of damage. Determining what set of tasks should be allowed by the supervisor is a problem of optimization that we approach using a similar method to that of [8]. Like [9] and [10], we use dynamic programming as a tool in our optimization of DES.

In this paper we consider three main problems. The first problem is to determine conditions necessary and sufficient

This research is supported in part by NSF Grants ECS-0080406 and CCR-0325571 and ONR Grant N00014-03-1-0232.

D. Thorsley is with the Department of Electrical Engineering, University of Washington, Seattle, WA, 98195. D. Teneketzis is with the Department of EECS, University of Michigan, Ann Arbor, MI, 48109. This work was performed when D. Thorsley was with the Department of EECS, University of Michigan. E-mail: thorsley@u.washington.edu,teneket@eecs.umich.edu.

for a supervisor to block the execution of any bad strings while achieving a given specification and is discussed in Section III. The second problem, discussed in Section IV, is to develop a method to assess the damage that an intruder can achieve in the case where the supervisor cannot block all bad strings. The third problem is to determine an optimal specification for a supervisor to achieve in the case of an intrusion and is discussed in Section V. The results of these sections are illustrated with an example in Section VI. Discussion of the results and possibilities for future research are found in Section VII.

II. Modelling Formalism

The model under consideration is a discrete-event system modelled by a finite state automaton $G = (X, \Sigma, \delta, x_0, X_m)$, where

- X is a finite set of states
- Σ is a finite alphabet of events
- $\delta : X \times \Sigma \rightarrow X$ is a (partial) transition function
- $x_0 \in X$ is an initial state
- $X_m \subseteq X$ is a set of marked states

We denote the language generated by G as $\mathcal{L}(G) = L$ and the language marked by G as $\mathcal{L}_m(G) = L_m$. We denote the empty string containing no events by ϵ.

The automaton G is partially observable; that is, the event set Σ is partitioned into $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$, where Σ_o denotes the set of observable events and Σ_{uo} the set of unobservable events. We define a projection function $P : \Sigma^* \rightarrow \Sigma_o^*$ and its inverse onto L, $P_L^{-1} : \Sigma_o^* \rightarrow 2^{\Sigma^*}$, in the usual manner [11].

G is also partially controllable; Σ is also partitioned as $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc}$, where Σ_c denotes the set of controllable events and Σ_{uc} the set of uncontrollable events. We make no assumption as to the relationship between Σ_o and Σ_c.

With access only to the given observable and controllable events, we can define a supervisor S_P. This supervisor is often defined as a function $S_P : P(L) \rightarrow 2^{\Sigma}$ which indicates which events the supervisor enables based on what sequence of events it has observed.

The supervisor S_P is able to achieve a given specification $K \subseteq \mathcal{L}_m(G)$. The specification consists of all those strings in $\mathcal{L}_m(G)$ that reach desirable marked states. We will denote this set of desirable marked states by X_m^+ and analogously denote the set of strings that terminate in desirable marked states as $L_m^+ = K$.

The supervisor also prevents the system from reaching marked states not included in X_m^+. We will denote this set of undesirable marked states by X_m^-. Thus the set of marked states is partitioned as $X_m = X_m^+ \dot{\cup} X_m^-$. The set of strings that terminate in undesirable marked states is denoted by L_m^-.

In the absence of an intrusion, the supervisory control system works as desired: all states in X_m^+ are reachable under control and no states in X_m^- are. Now suppose that an intruder has the ability to override the supervisor's decision to disable a particular event for a certain set of unreliable controllers $\Sigma_f \subseteq \Sigma_c$. Thus, an event is enabled if either the supervisor or the intruder chooses to enable it. We will denote the set of events that are controllable under intrusion as $\Sigma_{c,f} = \Sigma_c \backslash \Sigma_f$ and the set of events that are uncontrollable under intrusion as $\Sigma_{uc,f} = \Sigma_{uc} \cup \Sigma_f$.

In general, if the intruder chooses to interfere with the control actions chosen by the supervisor, the supervisor may not be able to achieve all the desirable states without allowing the intruder to execute traces that reach undesirable states. We desire conditions under which the supervisor can prevent the execution of illegal strings.

III. Conditions for Detecting Intruders

Whether or not the supervisor is capable of disabling strings that reach undesirable states depends on the specification that the supervisor is trying to achieve. Given a specification K, the first question we must consider is whether there exists a supervisor that achieves K, and, if so, how exactly to define that supervisor. The conditions under which K can be achieved are states in the controllability and observability theorem [12]. These conditions are stated using the two definitions that follow.

Definition 1: (From [13]) A specification K is *controllable* with respect to L and Σ_{uc} if

$$\overline{K}\Sigma_{uc} \cap L \subseteq \overline{K} \tag{1}$$

Definition 2: (From [12]) A specification K is *observable* with respect to L, P, and Σ_c if for all $s \in \overline{K}$ and $\sigma \in \Sigma_c$,

$$(s\sigma \notin \overline{K}) \text{ and } s\sigma \in L \Rightarrow P_L^{-1}[P(s)]\sigma \cap \overline{K} = \emptyset \tag{2}$$

Using these definitions, the controllability and observability theorem is stated as:

Theorem 1: There exists a supervisor S_P for G that achieves $K = L_m^+$ if and only if

1) K is controllable with respect to L and Σ_{uc}
2) K is observable with respect to L, P, and Σ_c

The proof of Theorem 1 given in [11] is constructive in that it defines a supervisor S_P that achieves the specification K as:

$$S_P(t) = \Sigma_{uc} \cup \{\sigma \in \Sigma_c : \exists s'\sigma \in \overline{K}(P(s') = t)\} \tag{3}$$

Having defined a supervisor that achieves the specification K, we can now discuss which strings in L_m^- can be disabled by this supervisor in the presence of an intrusion.

An arbitrary string $s^- \in L_m^-$ contains any number of controllable events, some of which are rendered uncontrollable in fact by the intruder. In order for the supervisor to prevent the system from executing s^-, it must choose to disable one of the events in $\Sigma_{c,f}$ that occurs along the trajectory of s^-.

The supervisor will disable an event in $\Sigma_{c,f}$ for one of two reasons. Either the supervisor has detected that the system is not moving towards any desirable state, in which case it should attempt to disable all possible events, or it has not detected any problem in the system and the event is disabled incidentally in order to achieve the specification K.

For the supervisor S_P that achieves the specification K as given in Theorem 1, we define the set of strings disabled by S_P as:

$$L_{disable} = \{s\sigma \in \Sigma\Sigma_{c,f} : P_L^{-1}[P(s)]\sigma \cap \overline{K} = \emptyset\} \quad (4)$$

Having defined $L_{disable}$, we can now state a condition under which we can ensure that a supervisor prevents an intruder with certain capabilities from steering the system to undesirable marked states.

Definition 3: A language L_m^- is *disarmable* with respect to a language $\mathcal{L}(G)$, a specification $K = L_m^+$, a projection P, Σ_c, and a set of unreliable controllers Σ_f if

$$L_m^- \subseteq L_{disable}\Sigma^* \quad (5)$$

The interpretation of this definition is L_m^- is disarmable if every string $s^- \in L_m^-$ contains a controllable event that will be disabled by the supervisor, even in the presence of an intruder.

Using the definition of disarmability and the results of Theorem 1, we can state conditions under which there exists a supervisor that can: 1) achieve K in the absence of an intrusion; and 2) disarm an intruder before executing a trace in L_m^-.

Theorem 2: There existing a supervisor S_P for G that achieves K and prevents the execution of L_m^- if and only if

1) K is controllable with respect to L and Σ_{uc}
2) K is observable with respect to L, P, and Σ_c
3) L_m^- is disarmable with respect to L,K,P,Σ_f, and Σ_c

Thus, in order for a supervisor to exists that achieves both stated goals, not only must the set of desirable strings be controllable and observable, but the set of undesirable strings must be disarmable with respect to the specification.

IV. Assessing Damage Caused by Intruders

It may not always be possible to completely prevent an intruder who has gained access to certain controllers from performing any undesirable task. In such a case, it would be useful to determine whether the tasks that an intruder can achieve are mere nuisances (e.g. accessing non-essential files) or more catastrophic (e.g. accessing financial or personal information). To address these questions, we use the concept of a measure for regular languages [14]. The language measure proposed there was chosen with the goal of automatically assessing controller performance. In this paper we briefly review the construction of such a measure; more details can be found in [14].

The measure is constructed by first assigning a terminal cost $\chi : X \to [-1, -1]$ to each state of G as follows:

$$\chi(x) \in \begin{cases} [-1, 0) & x \in X_m^- \\ \{0\} & x \notin X_m \\ (0, 1] & x \in X_m^+ \end{cases} \quad (6)$$

The terminal cost assures that strings reaching desirable marked states have positive measure and those reaching undesirable states have negative measure.

For each transition σ_k coming from a state x_j, we assign an arbitrary non-negative cost denoted by $\tilde{\pi}$ as follows:

$$\tilde{\pi}[\sigma_k, x_j] \equiv \tilde{\pi}_{jk} \in [0, 1); \sum_k \tilde{\pi}_{jk} < 1 \quad (7)$$

$$\tilde{\pi}[\sigma, x_j] = 0 \text{ if } \delta(x_j, \sigma) \text{ is undefined}; \tilde{\pi}[\epsilon, x_j] = 1 \quad (8)$$

The transition cost $\tilde{\pi}$ is extended to strings recursively by:

$$\tilde{\pi}[\sigma_k s, x_j] = \tilde{\pi}[\sigma_k, x_j]\tilde{\pi}[s, \delta(x_j, \sigma_k)] \quad (9)$$

The language measure is constructed by combining the transition cost $\tilde{\pi}[s, x_0]$ with the terminal cost $\chi(\delta(x_0, s))$. For a language consisting of single string s it is defined as:

$$\mu(\{s\}) = \tilde{\pi}[s, x_0]\chi(\delta(x, s)) \quad (10)$$

For any sublanguage $M \subseteq \mathcal{L}(G)$, the measure is defined by taking the sum of the measures of its constituent strings:

$$\mu(M) = \sum_{s \in M} \mu(s) \quad (11)$$

A language measure constructed in this manner is guaranteed to converge, i.e., $\mu(M)$ is guaranteed to be finite for all $M \subseteq \mathcal{L}(G)$ [14].

The language measure concept can be used to assess the damage done by an intruder interfering with the operation of a supervisory control system. For a given specification K and a given set of failed controllers Σ_f, some subset of the strings in L_m^- are reachable by the intruder as they are not disarmable. Define the language consisting of these strings as

$$L^-(K, \Sigma_f) = L_m^- \backslash L_{disable}\Sigma^* \quad (12)$$

From the above discussion, we can assign a measure to this language. Physically, this measure corresponds to the damage an intruder can achieve if it can enable events in Σ_f that the supervisor disables.

Thus, with respect to an intruder's capability Σ_f, we can assign a cost $V_{\Sigma_f} : 2_c^{\Sigma} \to \mathbb{R}$ as

$$V_{\Sigma_f}(K) = \mu[L^-(K, \Sigma_f)] \quad (13)$$

This cost is simply the measure of the set of undesirable strings that can be executed if the disabling of events in Σ_f can be overridden by the intruder. Since each string in $L^-(K, \Sigma_f)$ terminates in an element of X_m^-, $V_{\Sigma_f}(K)$ will be non-positive for all possible choices of Σ_f.

Although μ is a measure and thus additive, the cost function V is subadditive, that is,

$$V_{\Sigma_{f_1} \cup \Sigma_{f_2}}(K) \leq V_{\Sigma_{f_1}}(K) + V_{\Sigma_{f_2}}(K) \quad (14)$$

as there may be traces in L^- that are reachable only through the overriding of controllers in both Σ_{f_1} and Σ_{f_2} that cannot be reached if only one of those sets of controllers can be overridden.

A. Probabilistic Approach to Damage Assessment

In order to assess the damage caused by an intruder using the method described above, we need to know which controllers the intruder can override. Of course, in practical situations we would not have knowledge of the intruder's capabilities, as we would probably be unsure as to whether or not an intruder is actually present. However, a system designer can perceive how likely an intruder is to be able to access a particular set of controllers and assign a probability to that set.

The system designer can thus define a probability measure on Σ_c as $Q : 2^{\Sigma_c} \to [0,1]$. The potential damage that an intruder can achieve is thus

$$V(K) = \sum_{\Sigma_f \in 2^{\Sigma_c}} \mu[L^-(K, \Sigma_f)] Q(\Sigma_f) \tag{15}$$

The cost of a language is thus a weighted sum of the capabilities of different intruders weighted on the probability than an intruder will have that particular set of capabilities.

V. Finding an Optimal Control Specification

In the previous two sections, the specification K we have considered has been fixed and thus so has the supervisor S_P that achieves K. The specification and supervisor are two ways of expressing the control law that is applied to the system G.

This approach is naïve in the sense that if we design our system with the expectation that it may be attacked by an intruder, we may decide that some of the tasks in L_m^+ have a poor risk/reward ratio; that is, an intruder may be able to cause a lot of damage by executing an illegal trace that looks similar to a desired task that is not very valuable.

It would then be ideal to find a smaller control specification that can be achieved in the absence of an intruder and is optimal in the sense that the value of the good tasks in L_m^+ that it achieves minus the value of the bad tasks in L_m^- that an intruder can achieve is maximized.

To do this, we modify the language measure from the previous section by considering the benefit of achieving certain behaviors in K. Thus, with respect to a specification K, we can assign a cost $V : \Sigma_c \to \mathbb{R}$ as

$$V_{\Sigma_f}(K) = \mu[L^-(K, \Sigma_f)] + \mu[K] \tag{16}$$

We only wish to optimize over the set of specifications that are actually achievable by some supervisor S_P that we can construct with the given controllable and observable events. In order for there to exist a supervisor that achieves a given specification, that specification must be controllable and observable. Denote the set of all controllable and observable sublanguages of $\overline{K}$ by $\mathcal{CO}_{in}(\overline{K})$.

The optimization problem is therefore stated as follows: Find a specification K^* such that for all $K' \in \mathcal{CO}_{in}(\overline{K})$,

$$V_{\Sigma_f}(K^*) \leq V_{\Sigma_f}(K') \tag{17}$$

A. Probabilistic Optimization

As in Subsection IV-A, the system designer can define a probability measure on Σ_c as $Q : 2^{\Sigma_c} \to [0,1]$. The cost of a particular specification is therefore

$$J(K) = \sum_{\Sigma_f \in 2^{\Sigma_c}} \{\mu[L^-(K, \Sigma_f)] + \mu[K]\} Q(\Sigma_f) \tag{18}$$

The probabilistic optimization problem is therefore: Find a specification K^* such that for all $K' \in \mathcal{CO}_{in}(\overline{K})$,

$$J(K^*) \leq J(K') \tag{19}$$

To simplify the development of the methods used in this paper, we will restrict our attention to the case where $\overline{K}$ is acyclic.

Instead of optimizing for K^* in the set $\mathcal{CO}_{in}(\overline{K})$, we solve these problems indirectly by finding an optimal supervisor S_P^*, and then calculating K^* as the specification achieved by S_P^* acting on the system G.

In order to find an optimal specification, it is necessary to develop an appropriate information structure on which to optimize.

B. Information Structure

In order to systematically describe the information needed to find an optimal supervisor, we use an approach based on the intrinsic model for stochastic control. This approach was proposed in [15], [16] for general informationally distributed systems and was further used in [8], [17], [18].

The information available to the supervisor is based on both the observations that the supervisor has made of the system and the actions that the supervisor has applied to the system. We express this idea with the following definition:

Definition 4: An *information state* π is the set of strings in $\mathcal{L}(G)$ consistent with the observations made and control actions performed by the supervisor S_P.

This definition is consistent with the requirements for an information state given in [19]. Those two requirements are

1) Given the initial information state $\pi_0 = \epsilon$, a sequence of observed events $s_o = \sigma_1\sigma_2\ldots\sigma_n$, and the sequence of control actions $S_P(\epsilon), S_P(\sigma_1), S_P(\sigma_1\sigma_2), \ldots, S_P(\sigma_1\sigma_2\ldots\sigma_{n-1})$, we can determine the current information state π_n as a function of these given quantities.
2) Given the current information state π_n, an observation σ_{n+1}, and a control action $S_P(\pi_n)$, we can determine the next information state π_{n+1} as a function of these three elements.

The information state is thus a summary of the record of observations made and control actions performed by the supervisor.

When optimizing the supervisor, the goal is to determine the control actions necessary for every possible information state that can be reached by any control policy we may end up using. We need to consider how information can evolve when the observation actions are fixed but the control actions are to be determined.

Suppose that two strings contain an identical sequence of observable events. In order to have an admissible supervisor, we must choose the same control action after both of these strings occur. This control action may enable or disable any number of unobservable or observable controllable events, but we cannot update our control action until a new observation is made.

Furthermore, suppose that between successive observable events, these two strings contain identical sets of unobservable controllable events. If we choose to disable one of these strings by disabling an unobservable event, we must also disable the second string as that string also contains any event that we can feasibly disable. Therefore, we must disable both these traces or we must disable neither.

We formalize this notion of traces that must be enabled or disabled jointly using the idea of the *control projection*. We extend the standard projection operation by introducing symbolize to indicate which set of unobservable controllable events occurs between each pair of observable events in the projection.

We denote the set of symbols denoting the sets of unobservable events as C_A. Each symbol in this alphabet will be of the form 1_A, where A is a set of unobservable controllable events. For example, if our system has three unobservable controllable events $\{\alpha, \beta, \gamma\}$, the associated alphabet is $C_A = \{1_\emptyset, 1_{\{\alpha\}}, 1_{\{\beta\}}, 1_{\{\gamma\}}, 1_{\{\alpha,\beta\}}, 1_{\{\alpha,\gamma\}}, 1_{\{\beta,\gamma\}}, 1_{\{\alpha,\beta,\gamma\}}\}$.

The control projection is a string whose events alternate between the symbols indicating controllable unobservable events and observable events. Formally it is defined for events as:

$$CP(\sigma) = \begin{cases} \sigma & \text{if } \sigma \in \Sigma_o \\ 1_{\{\sigma\}} & \text{if } \sigma \in \Sigma_c \cap \Sigma_{uo} \\ \epsilon & \text{otherwise} \end{cases} \tag{20}$$

Each symbol in C_A indicates the unobservable events that can be disabled before the next observation is made.

When the control projection is extended to traces, we must ensure the alternation of symbols from C_A and symbols from Σ_o. In the case where two observable events may end up adjacent, we simply insert the symbol $1_\emptyset$ between them, as there as no unobservable events, either controllable or uncontrollable, between those events. When two symbols in C_A are adjacent, we must merge the two symbols using the function $\vee : C_A \times C_A \to C_A$:

$$1_A \vee 1_B = 1_{A \cup B} \tag{21}$$

The merge function is extended to strings in $(C_A\Sigma_o) * C_A$ by maintaining all symbols in the string except for the last, which is merged with the symbol to be concatenated to the string.

$$(t1_A) \vee 1_B = t1_{A \cup B} \tag{22}$$

Formally, the function is extended to traces as $CP : \Sigma^* \to \overline{(C_A\Sigma_o)*}$ such that:

$$CP(s\sigma) = \begin{cases} CP(s)1_\emptyset CP(\sigma) & \text{if } s_f \in \Sigma_o,\ \sigma \in \Sigma_o \\ CP(s) \vee 1_{\{\sigma\}} & \text{if } s_f \in C_A,\ \sigma \in \Sigma_c \cap \Sigma_{uo} \\ CP(s)CP(\sigma) & \text{otherwise} \end{cases} \tag{23}$$

where s_f denotes the final event in the string $CP(s)$.

For each string of unobservable events, the control projection records the set of controllable events that occur along that string. If two strings contain the same sequence of observable events and the same sets of unobservable controllable events between pairs of observable events, it is not possible to choose a policy that distinguishes between these two traces.

Using the control projection, we can sequentially decompose the set of possible information states so as to facilitate optimization. To do so, we first partition the projection of $\overline{K}$ according to how many more observations are possible following a given string.

For $n = 0 \ldots \max_{s \in K} \|s\|$, define the set X_n as:

$$X_n = \{s_o \in P(\overline{K}) : \max_{t \in P(\overline{K})/s_o} \|t\| = n\} \tag{24}$$

To generate the sets of possible information states, we take each element s_o of each X_n and generate the set of all possible information states by partitioning s_o into a set of elements where each element has its own distinct control projection. We then take the field generated by this partition and define a set of information states Y_n for each n as follows:

$$Y_n = \bigcup_{s_o \in X_n} 2^{CP_L^{-1}[CP(P_L^{-1}(s_o))]} \tag{25}$$

By defining the information space in this manner, we have accomplished two goals. Firstly, every information state in $\overline{K}$ reachable under any control policy we may choose to implement is a member of one of the sets Y_n. Secondly, the information space is decomposed into the sequence $(Y_0, Y_1, \ldots, Y_{max})$, allowing us to determine an optimal control policy using the backwards induction approach of dynamic programming.

C. Dynamic Programming Approach

Having defined an information structure for the largest possible specification K, we can use this structure to find an optimal supervisor S_P^* using dynamic programming.

We must first consider information states that lie outside the sequence $\{Y_n\}$. If an information state is not an element of any Y_n, then the observations made by the supervisor indicate that the behavior of the system is outside of the largest possible specification K. In this case, as no positive reward can be reached, the optimal action is to disable all events that can be disabled. The cost of such an information state is therefore given by:

$$V(\pi) = \mu[\pi\Sigma_{uc,f}^*] \text{ for } \pi : P(\pi) \notin P(\overline{K}) \tag{26}$$

The cost of π in this case is therefore the measure of π and its uncontrollable extension.

We must ensure that the action implemented at each information state is admissible, i.e., we cannot choose a control action that would disable an uncontrollable event. If we allow our supervisor to be blocking, we define the set of admissible actions as $A = \{g(\pi) \in 2^{\Sigma} : \Sigma_{uc} \subseteq g(\pi)\}$. The dynamic programming equation to find an optimal action at π is given by:

$$V(\pi) = \max_{g(\pi)\in A} \{\mu[\pi(\Sigma_{uo} \cap (\Sigma_f \cup g(\pi)))^*] + \Sigma_{\sigma\in\Sigma_o} V(\pi(\Sigma_{uo} \cap (\Sigma_f \cup g(\pi)))^*\sigma)\} \quad (27)$$

To solve for the optimal cost sequentially, we first solve this equation for information states in Y_0, then for those in Y_1, and so on. Proof of the optimality of the result given by this procedure can be found in Chapter 6 of [19].

In the case where we wish to ensure that the supervisor is non-blocking, we modify the set of admissible actions to ensure that the action we choose at any information state ensures that a desirable marked state will eventually be reached if the intruder does not intervene. For each information state π, the set of admissible actions that do not result in blocking is given by:

$$A_{NB}(\pi) = \{g(\pi) \in 2^{\Sigma} : (\Sigma_{uc} \subseteq g(\pi)) \wedge (\forall s \in \pi)(\Gamma[s(g(\pi) \cap \Sigma_{uo})^*] \subseteq g(\pi))\} \quad (28)$$

To search for a non-blocking supervisor, we simply modify Equation (27) by looking for the argument that maximizes $V(\pi)$ in the set $A_{NB}(\pi)$ as opposed to the set A.

D. Probabilistic Dynamic Programming Approach

The case of probabilistic knowledge of the intruder's capabilities can be handled in a very similar manner. Again, if the observed string indicates that the system's behavior is outside the specification, the optimal action is to disable all possible events and the cost of such an information state is given by:

$$V(\pi) = \sum_{\Sigma_f \in 2^{\Sigma_c}} Q(\Sigma_f)\mu[\pi\Sigma^*_{uc,f}] \text{ for } \pi : P(\pi) \notin P(\overline{K}) \quad (29)$$

For π in the sequence Y_n, optimal rewards and observation actions can be found using the equation:

$$V(\pi) = \max_{g(\pi)\in A} \{ \sum_{\Sigma_f \in 2^{\Sigma_c}} Q(\Sigma_f)\{\mu[\pi(\Sigma_{uo} \cap (\Sigma_f \cup g(\pi)))^*] + \Sigma_{\sigma\in\Sigma_o} V(\pi(\Sigma_{uo} \cap (\Sigma_f \cup g(\pi)))^*\sigma)\}\} \quad (30)$$

As before, the optimal cost can be found sequentially, solving the above equation first for information states in Y_0.

E. Discussion of Optimization Procedure

If we have knowledge of all previous control actions, the calculation of an optimal action becomes independent of the previous control law [20]. When performing optimization using dynamic programming, the control laws at previous stages are not yet defined; thus, it is necessary to use information states that summarize the effects of all possible previous control actions. The information sequence $\{Y_n\}$ has this desired property: every reachable information state in $\overline{K}$ is a member of the sequence. By determining an optimal control action for each of these information states, we can find an optimal policy without depending on the previous control law.

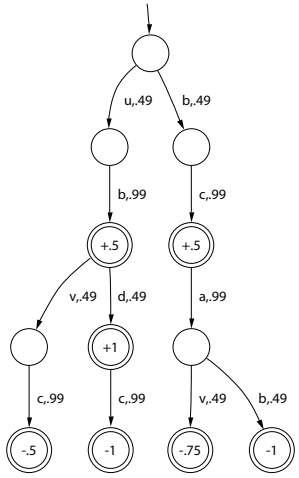

Fig. 2. A supervisory control system with an intruder affecting control actions.

However, in order to define a supervisor as either a function $S_P : P[\mathcal{L}(G)] \rightarrow 2^{\Sigma}$ (as in [11]) or as a function dependent on the state of an automaton (as in, e.g. [4]), it is necessary to know the entire control law because the actions implemented in these definitions are explicitly defined only in terms of an observed state of sequence, and thus implicitly dependent on the previous control laws. By using the information sequence $\{Y_n\}$ which is larger than $P[\mathcal{L}(\overline{K})]$, we can eliminate the dependence on the previous control law and use dynamic programming to find an optimal supervisor.

Once the optimization procedure is complete and the entire control law is known, we can then implement the chosen optimal supervisor as a function from $P[\mathcal{L}(G)]$ to 2^{Σ}, For each s_o in $P[\mathcal{L}(G)]$, only one information state with projection s_o will be reachable when the chosen optimal policy is implemented, and we can choose the action at that information state as the one that the supervisor should implement when it observes s_o.

VI. Example

To illustrate the concepts introduced in Sections III through V, consider the automaton shown in Figure 2. In this automaton, all events are controllable, i.e. $\Sigma_c = \Sigma$, and the set of unobservable events is $\Sigma_{uo} = \{u, v\}$. With these particular choices for Σ_c and Σ_{uo}, we can quickly confirm using Theorem 1 that the language

$$K = L_m^+ = \{bc, ub, ubd\} \quad (31)$$

is both controllable and observable. Thus there exists a supervisor S_P that achieves the specification K in the absence

of an intruder.

Now suppose that an intruder is present and interfering with the output of the supervisor, and that the intruder's capabilities are given by $\Sigma_f = \{a, v\}$, i.e., the intruder can enable the events a and v even if the supervisor has disabled them. The set of events still controllable by the supervisor with the intruder present is given by $\Sigma_{c,f} = \{b, c, d, u\}$.

The set of all undesirable strings is

$$L_m^- = \{bcab, bcav, ubdc, ubvc\}. \tag{32}$$

We first check to see if L_m^- is disarmable with respect to L, K, P, Σ_f, and Σ_c. The set of traces that will be disabled by the supervisor with the intruder present is:

$$L_{disable} = \{bcab, ubdc\} \tag{33}$$

L_m^- is therefore not disarmable because the strings $bcav$ and $ubvc$ are elements of L_m^- but not $L_{disable}\Sigma^*$.

Since L_m^- is not disarmable, we would like to assess the damage that the intruder can cause by allowing the execution of undesirable traces. The set of undesirable traces reachable by the intruder is:

$$L^-(K, \Sigma_f) = \{bcav, ubvc\} \tag{34}$$

We assess the damage of this set of failed controllers by calculating the language measure μ of this language. For each marked state x_m, the terminal reward χx_m is shown inside the state in Figure 2, and for each transition, the cost $\pi[x, s]$ is shown next to the event label. Using these language measure values, we can determine the cost V_K of Σ_f as:

$$V_K(\{a, u_2\}) = \mu[\{bcav, ubvc\}] \tag{35}$$

$$= (.49)^2(.99)^2(-.75) + (.49)^2(.99)^2(-.5) \tag{36}$$

$$= -.294 \tag{37}$$

Now consider the case where the capabilities of the intruder are modelled probabilistically. Suppose that the probabilities that the intruder can override the events a and v are independent of each other and are denoted by p_a and p_v respectively. Then the assessment of the damage that can be caused by the intruder is given by:

$$V(K) = p_a p_v \mu[bcav] + p_v \mu[ubvc] \tag{38}$$

In order for the string $bcav$ to be executed, the intruder must be capable of overriding the supervisor's attempts to disable both a and v, and can do so with probability $p_a p_v$. However, for the string $ubvc$ can be executed, the intruder need only be able to enable v, which it can do with probability p_v.

To determine an optimal control specification, we partition the language $P(\overline{K})$ into the sets X_0, X_1, and X_2 as follows:

$$X_0 = \{bc, bd\} \tag{39}$$

$$X_1 = \{b\} \tag{40}$$

$$X_2 = \{\epsilon\} \tag{41}$$

π	Enable	Disable	Expression	Reward
bc	—	b, a, v	$\mu(bc + bcav)$	.066
$ubvc$	—	—	$\mu(ubvc)$	-.118
$bc + ubvc$	—	b, a, v	$\mu(bc + bcav + ubvc)$	-.050
$\mathbf{ubd}$	—	c	$\mu(ubd)$	.238
b	c	—	$\mu(bc + bcav)$	.066
ub	d	c, v	$\mu(ub + ubd)$	.480
$\mathbf{b + ub}$	d	c, v	$\mu(ub + ubd)$	.480
ϵ	u, b	—	$\mu(ub + ubd)$	.480

TABLE I

INFORMATION STATES AND THEIR ASSOCIATED REWARDS FOR THE AUTOMATON IN FIGURE 2. THE INFORMATION STATES REACHABLE BY THE POLICY S^* ARE HIGHLIGHTED.

From these sets X_n, we generate the sets of possible information states using the control projections of each element:

$$Y_0 = \{bc, ubcv, bc + ubvc, ubd\} \tag{42}$$

$$Y_1 = \{b, ub, b + ub\} \tag{43}$$

$$Y_2 = \{\epsilon\} \tag{44}$$

In order to determine an optimal specification, we determine an optimal control action for each information state. We start by calculating optimal actions for the elements of Y_0, then for elements of Y_1, and lastly for Y_2.

For example, to calculate an optimal action for the information state $\pi = bc + ubvc$, we apply Equation (27). Since the only feasible observable event from this information state is a, we derive the equation:

$$V(bc + ubvc) = \max_{g(\pi) \in A} \{\mu[ubvc, bc] + V(bca)\} \tag{45}$$

To determine the value of $V(bca)$, we note that the information state bca lies outside of $\overline{K}$ and thus using Equation (26), we can calculate that:

$$V(bca) = \mu[\{bca, bcav\}] = -.176 \tag{46}$$

The reward for the information state $bc + ubvc$ is determined by choosing the policy where no events are enabled, and is calculated as:

$$V(bc + bcav + ubvc) = .243 - .118 - .176 = -.050 \tag{47}$$

Since this information state has a negative reward, an optimal policy will avoid reaching this information state if possible. As the complete results of the optimization algorithm in Table I show, this is indeed the case; the information states that can be reached through following the chosen optimal policy are highlighted.

To express the supervisor that generates the optimal controlled language as a function $S_P^* : P[\mathcal{L}(G)] \rightarrow 2^\Sigma$, we assign to each string in $P[\mathcal{L}(G)]$ the control action in 2^Σ that coincides with the information state that is reachable under the optimal policy. The optimal supervisor S_P^* can

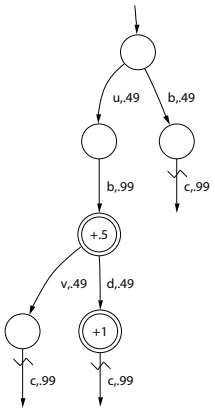

Fig. 3. An optimal specification under intrusion for the automaton in Figure 2.

thus be expressed more compactly as:

$$S_P^*(\epsilon) = \{u, b\} \tag{48}$$
$$S_P^*(b) = \{d\} \tag{49}$$
$$S_P^*(bd) = \emptyset \tag{50}$$
$$S_P^*(t) = \emptyset \text{ for all other } t \in P(\mathcal{L}(G)) \tag{51}$$

The action $S_P^*(b)$ is $\{d\}$ because when S^* is implemented at ϵ, the information state generated when b is observed in $b + ub$, and thus we choose the action associated with this information state when b is observed. However, in order to determine that the information state $b + ub$ was indeed reachable under S^*, we needed to know the control policy for all stages of the automaton preceding the supervisor's observation of b.

Once an optimal supervisor is calculated, we can determine its associated optimal specification. The optimal specification that is realized by the supervisor S^* is

$$K^* = \overline{\{b, ubd\}} \tag{52}$$

This specification is shown in Figure 3. In this example, K^* is blocking; in general, we can modify our dynamic program using Equation (28) to restrict our policy search to non-blocking supervisors.

VII. Discussion

This paper proposes a framework for modeling supervisory control systems in DES that are subject to interference as a result of failure, noise, or intrusion. The problems we consider in this framework are how to check if an intruder with certain capabilities can execute any undesirable strings, how to assess the damage that an intruder can cause to the system, and how to optimize the control specification so as to maximize the rewards from executing desirable strings and minimize the penalties from the intruder executing undesirable ones. The method of optimizing supervisors is described only for acyclic specifications in this paper; it can be extended to cyclic specifications in a manner similar to the one described for diagnosis in [21].

Extensions for future work on this problem involve increasing the complexity of the system architecture under consideration or altering the capabilities of the intruder. While in this paper we have modeled interference in the output of the supervisor, it is also conceivable that an intruder could interfere with the supervisor's input, thereby affecting the control actions the supervisor chooses. Furthermore, as the problem of intrusion detection is in part motivated by communications and computer systems modeled by DES, extending these results to a decentralized supervisory control framework could have many potential applications.

References

[1] Y. Pencolé, "Decentralized diagnoser approach: Application to telecommunication networks," in *Proc. 11th International Workshop on Principles of Diagnosis (DX'00)*, June 2000, pp. 185–192.

[2] K. Sinnamohideen, "Discrete-event diagnostics of heating, ventilation, and air-conditioning systems," in *Proc. 2001 American Control Conference*, June 2001.

[3] R. Sengupta, "Discrete-event diagnostics of automated vehicles and highways," in *Proc. 2001 American Control Conference*, June 2001.

[4] P. Ramadge and W. Wonham, "The control of discrete-event systems," *Proc. IEEE*, vol. 77, no. 1, pp. 81–98, 1989.

[5] S. Lafortune, D. Teneketzis, M. Sampath, R. Sengupta, and K.Sinnamohideen, "Failure diagnosis of dynamic systems: An approach based on discrete event systems," in *Proc. 2001 American Control Conference*, June 2001, pp. 2058–2071.

[6] A. Paoli and S. Lafortune, "Safe diagnosability of discrete event systems," in *Proc. 42st IEEE Conf. on Decision and Control*, Dec. 2003, pp. 2658–2664.

[7] P. Eng and M. Haug, "Automatic response to intrusion detection," masters Thesis in Information and Communication Technology, Agder University College.

[8] D. Thorsley and D. Teneketzis, "Active acquisition of information for diagnosis of discrete event systems," in *Proc. 42th Allerton Conference on Communication, Control, and Computing*, Sept. 2004.

[9] R. Sengupta and S. Lafortune, "An optimal control theory for discrete event systems," *SIAM J. Control and Optim.*, vol. 36, no. 2, 1998.

[10] J. Tsitsiklis, "On the control of discrete-event dynamical systems," *Maths. of Control, Signals and Systems*, vol. 2, pp. 95–107, 1989.

[11] C. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Boston, MA: Kluwer Academic Publishers, 1999.

[12] F. Lin and W. M. Wonham, "On observability of discrete-event systems," *Information Sciences*, vol. 44, pp. 199–224, 1988.

[13] P. Ramadge and W. Wonham, "Supervisory control of a class of discrete-event processes," *SIAM J. of Control Optimization*, vol. 25, no. 1, pp. 206–230, 1987.

[14] X. Wang and A. Ray, "Signed real measure of regular languages," in *Proc. 2002 American Control Conference*, 2002, pp. 3937–3942.

[15] H. Witsenhausen, "On information structures, feedback and causality," *SIAM J. of Control*, vol. 9, no. 2, pp. 149–160, May 1971.

[16] ——, "The intrinsic model for discrete stochastic control: Some open problems," in *Lecture Notes in Economics and Mathematical Systems, 107*. Berlin: Springer-Verlag, 1975, pp. 322–335.

[17] M. Andersland and D. Teneketzis, "Information structures, causality, and non-sequential stochastic control, I: design-independent properties," *SIAM J. of Control Optim.*, vol. 30, no. 6, pp. 1447–1475, Nov. 1992.

[18] ——, "Information structures, causality, and non-sequential stochastic control, II: design-dependent properties," *SIAM J. of Control Optim.*, vol. 32, no. 6, 1994.

[19] P. Kumar and P. Varaiya, *Stochastic Systems: Estimation, Identification, and Adaptive Control*. Englewood Cliffs, NJ: Prentice Hall, 1986.

[20] H. Witsenhausen, "Separation of estimation and control for discrete time systems," *Proc. of the IEEE*, vol. 59, no. 11, pp. 1557–1566, Nov. 1971.

[21] D. Thorsley and D. Teneketzis, "Diagnosis of cyclic discrete-event systems using active acquisition of information," in *Proc. 8th International Workshop on Discrete Event Systems (WODES '06)*, July 2006, pp. 248–255.