Optimal Supervisory Control of Discrete Event Dynamical Systems [1,2]

Ratnesh Kumar
Department of Electrical Engineering
University of Kentucky
Lexington, KY 40506-0046
Email: kumar@engr.uky.edu

Vijay K. Garg
Department of Electrical and Computer Engineering
University of Texas at Austin
Austin, TX 78712-1084
Email: vijay@pine.ece.utexas.edu

[1]This research was supported in part by the Center for Robotics and Manufacturing, University of Kentucky, in part by the National Science Foundation under Grant NSF-CCR-9110605, and in part by a TRW faculty assistantship award.

[2]A preliminary version of this paper appeared in [10].

Abstract

We formalize the notion of optimal supervisory control of discrete event dynamical systems (DEDS's) in the framework of Ramadge and Wonham. A DEDS is modeled as a state machine, and is controlled by disabling some of its transitions. We define two types of cost functions: a cost of control function corresponding to disabling transitions in the state machine, and a penalty of control function corresponding to reaching some undesired states, or not reaching some desired states in the controlled system. The control objective is to design an *optimal* control mechanism, if it exists, so that the *net* cost is minimized. Since a DEDS is represented as a state machine—a directed graph—network flow techniques are naturally applied for designing optimal supervisors. We also show that our techniques can be used for solving supervisory control problems under complete as well as partial observation. In particular, we obtain, for the first time, techniques for computing the supremal controllable and normal sublanguage, and the infimal controllable and normal/observable superlanguage without having to perform alternate computations of controllable and normal/observable languages.

Keywords: Discrete Event Dynamical Systems, Supervisory Control, Automata Theory, Optimal Control, Max-flow Min-cut.

AMS (MOS) Subject Classification: 93

1 Introduction

Research on supervisory control of discrete event dynamical system was pioneered by Ramadge and Wonham [23]. In [23], a DEDS, also called plant, is modeled as a state machine (SM), and the behavior of a DEDS is described by the language accepted by the corresponding SM. A controller or a supervisor, based on its observation of the past behavior of the plant, determines the transitions to be disabled in the plant, so that some desired qualitative control objective is achieved. Usually, the control objective is to restrict the plant behavior such that it remains confined within a specific range [24]. In some other cases, the control objective is to design a supervisor so that the closed loop behavior *eventually* remains confined to a prescribed range [3, 18, 12]. Recently, there has also been some work in which the control objective is to restrict the plant behavior so that a certain cost function defined along the trajectory of the system is optimized [20, 19, 2, 26, 10].

In [20, 19], a cost function is defined on the set of transitions, and the control objective is to restrict the plant behavior in such a way that after starting from a *given* initial state the plant reaches one of the accepting states along a trajectory of optimal cost. Authors provide an efficient heuristic search based algorithm for solving the problem. In [2] also, a cost function is defined on the set of transitions and the control objective is to restrict the plant behavior so that after starting from *any* state the plant reaches one of the accepting states along a trajectory of optimal cost. In [26], two types of costs, a control cost and a path cost, are defined on the graph representing a plant. The control objective is to determine that subgraph of the plant graph for which the maximum of the total cost along all its trajectories is minimum. The notions of cost of control and penalty of control considered in this paper are some what similar to that of control cost and path cost in [26]. However, our control objective is to determine a *state-feedback* supervisor so that the net cost of disabling transitions, that of reaching undesired states, and that of not reaching desired states is minimized. Thus our control objective is different from that considered in [26].

In this paper, we consider two types of cost functions: (i) A positive *cost of control* function is defined on the set of transitions corresponding to the cost of disabling a transition. If a certain transition such as arrival of a customer in a queue is disabled by a supervisor at a certain point, then its cost of control is added to the net cost, otherwise—if the transition is not disabled—no cost is added to the net cost; (ii) A *penalty of control* function is defined on the set of states corresponding to their reachability in the controlled system. The penalty of control takes a negative or a positive value depending on whether the state is desired or undesired. If a state is desired—e.g. working or idle state of a machine, then a negative penalty is associated with it. If such a state remains unreachable in the controlled plant, then a positive cost equal in magnitude to its penalty of control is added to the net cost, otherwise—if the state is reachable—no cost is added to the net cost. On the other hand, if a state is undesired—e.g. over/underflow of a buffer, then a positive penalty is associated with it. If such a state can be reached in the controlled plant, then a cost equal to its penalty of control is added to the net cost, otherwise—if it remains unreachable—no cost is added to the net cost. The optimal control problem is to determine a state-feedback supervisor for which

the net cost is minimized. State-feedback supervisors [8, 21, 13, 6], exercise control based on the state of the plant rather than the sequence of events executed by it. However, this does not result in any loss of generality, since a "string-feedback" supervision is equivalent to a state-feedback supervision on a suitably refined [7] model of a plant.

Example 1 Consider for example a machine P shown in Figure 1. (For clarity, the state labels have been omitted.) Initially P is in the "first idle" state (i.e. idle and unused state).

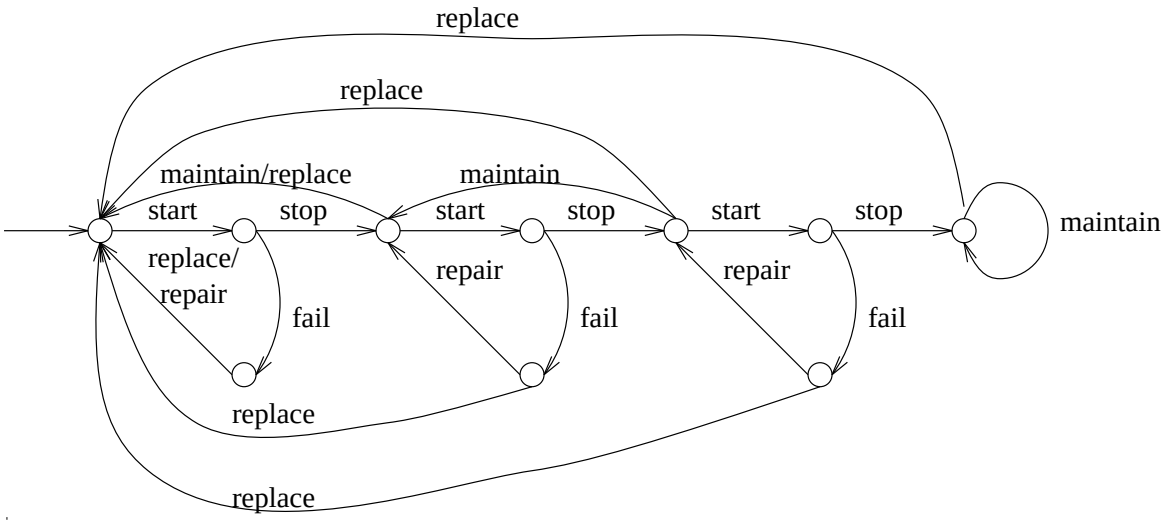


Figure 1: Diagram for Machine P of Example 1 with $N = 4$

When event "start" is executed, it goes to the "first working" state. In the "kth working" state, where $1 \leq k < N$, the machine may either "fail", in which case it goes to the "kth broken" state; or it may complete its operation, execute "stop", and go to the "$(k+1)$th idle" state. In the "kth broken" state, machine is either "repaired", in which case it goes back to the "kth idle" state; or it is "replaced", in which case it goes to the "first idle" state. In the "kth idle" state, either the event "start" is executed which sends the machine to the "kth working" state; or "replace" is executed which sends the machine to the "first idle" state; or "maintain" is executed which sends the machine to the "$(\max\{1, 2k-N\})$th idle" state. Thus, in the "Nth idle" state, execution of the "maintain" event does not result in a change in state. Note that the function $\max\{1, 2k-N\}$ is chosen only for an illustration, in general it is an increasing function of k taking values smaller than k except at $k = 1$ and $k = N$, where it equals k. An optimal control policy is needed to decide (i) whether to repair or replace the machine in a broken state, and (ii) whether to operate or maintain or replace the machine in an idle state. An optimal control policy evidently depends on the cost of replacement of machine, cost of repair in the kth broken state, cost of maintenance in the kth idle state, payoff of operating the machine in the kth idle state, penalty of being in a broken state, payoff of being in an idle or a working state etc.

Our setting is similar to that considered in [27], in which, appropriate cost and penalty

functions were defined on a suitably refined model of a given plant, and the dynamic programming algorithm was used to determine the *existence* of a supervisor for a given control problem. The computational complexity of the dynamic programming algorithm was used to determine the computational complexity of the supervisory control problem thus solved. However, no technique for the *synthesis* of a supervisor, if it exists, was given in that reference. Our approach to optimal supervisory control differs from that considered in [27] in two ways. Firstly, we show that an optimal supervisory control problem of the type described above can be solved using network flow algorithms. Thus a more general algorithm such as dynamic programming can also be used, although this will result in an *increase* in computational complexity. Secondly, we show that our techniques are equally applicable for the synthesis of supervisors, whenever they exist.

The motivation of formulating an optimal supervisory control problem is twofold: firstly, to introduce a formal framework for optimal supervisory control in which the control objective is to optimize a suitably defined cost and penalty of exercising controls; secondly, to present a unified technique for supervisory synthesis under complete as well as partial observation. In particular we obtain techniques for computing the supremal controllable and normal sublanguage [22, 16, 1, 11, 4], the infimal controllable and normal superlanguage [14], the infimal controllable and observable superlanguage [16, 5, 25, 9], *without* having to perform alternate computations of controllable and normal/observable languages as is done in [4] (also refer to Remark 3). Our techniques also illustrate that a supervisory control problem under complete or partial observation can be solved using a state-feedback type of control on a suitably refined state machine representation of the plant. We provide techniques to obtain the the appropriate refinements.

In Section 2, we introduce our notation and formally describe the optimal supervisory control problem under complete as well as partial state observation. In Section 3, we show how the network flow algorithms can be used to solve the optimal supervisory control problems. In Section 4, we show that by appropriately defining the cost and penalty functions, our techniques can be used for solving supervisory control problem under complete as well as partial observation.

2 Notation and Problem Formulation

A discrete event dynamical system to be controlled, called plant, is modeled as a state machine [7], denoted as a four tuple $G := (X, \Sigma, \delta, x_0)$, where X denotes the set of states, Σ denotes the finite set of *events*, $\delta : X \times \Sigma \to X$ denotes the partial deterministic state transition function, and $x_0 \in X$ denotes the initial state. A triple $(x_1, \sigma, x_2) \in X \times \Sigma \times X$, such that $\delta(x_1, \sigma) = x_2$ is called a *transition* in G. The behavior of G is described by the language $L(G)$:

$$L(G) := \{s \in \Sigma^\star \mid \delta(x_0, s) \text{ is defined}\},$$

where $\Sigma^\star$ denotes the set of finite sequences of events belonging to Σ, including the zero length sequence ϵ; the transition function is extended in a natural way to $\delta : X \times \Sigma^\star \to X$.

In general, a supervisor or a controller determines the set of events to be disabled after each transition, based on the record of observed states and events. We consider a supervisor, denoted S, to be a map $S : X \to 2^{\Sigma}$ that determines the set of events $S(x) \subseteq \Sigma$ to be disabled at each state $x \in X$. Events not belonging to the set $S(x)$ remain enabled at x. A supervisor as defined above is called a state-feedback [21, 13, 12], as it exercises control based on the state of G (and not based on the record of observed states and events). As is shown below, this does not result in any loss of generality, as a more general supervisor, which exercises control based on the observed sequence of events, can equivalently be viewed as a state-feedback supervisor on a suitably refined model of the plant. Readers are referred to [24] for a more general definition of a supervisor. The controlled plant, denoted G_S, is another state machine given as the 4-tuple $G_S := (X, \Sigma, \delta_S, x_0)$, where X, Σ, x_0 are as defined above, and δ_S denotes the state transition function of the controlled plant G_S:

$$\forall x \in X, \sigma \in \Sigma : \delta_S(x, \sigma) := \begin{cases} \delta(x, \sigma) & \text{if } \sigma \notin S(x) \\ \text{undefined} & \text{otherwise} \end{cases}$$

The behavior of the closed-loop system is described by the language $L(G_S)$ generated by the controlled plant. It is clear that $L(G_S) \subseteq L(G)$.

Next we formally describe the problem of optimal supervisory control. Let $c : X \times \Sigma \to \mathcal{R}^+$ denote a cost of control function, where $\mathcal{R}^+$ denotes the set of strictly positive reals, including infinity. The cost $c(x, \sigma)$ represents the cost of disabling the event $\sigma \in \Sigma$ at the state $x \in X$.

We assume for simplicity that the cost of control is a "one-time" cost. A justification for this simplifying assumption is the following: In G_S, a certain state may be visited either only once (if it is not a node on any cycle in the graph of G_S), or an unbounded number of times (otherwise). If a state is visited only once, then the corresponding transitions are controlled only once. Hence, in this case, the cost of control ought to be one-time. On the other hand, if a state is visited an unbounded number of times, then the corresponding transitions are controlled each time the state is visited. However, due to the state-feedback nature of supervision, the *same* control is exercised on each occasion. Hence, in this case, the cost of controlling such transitions can be associated with the first time the control is exercised. The one-time cost of control assumption can also be interpreted as follows: A transition once disabled/enabled at the corresponding state remains disabled/enabled in that state so that on subsequent visits to that state no cost is incurred in disabling/enabling that transition. Alternatively, the cost of control is primarily of setting up a control mechanism—a switch for example—and the cost of engaging or disengaging the switch is small compared to its one time set up cost. In each case, the assumption of state-feedback supervision is crucial.

Next, a penalty of control function $p : X \to \mathcal{R}$, is defined on the state set, where $\mathcal{R}$ denotes the set of reals, including positive and negative infinity. It corresponds to the penalty associated with reachability of a certain state in the controlled plant. The penalty function may take a positive or a negative value depending on whether the corresponding state is undesired or desired. Given a state $x \in X$, if $p(x) < 0$, then x is a desired state, and it should remain reachable in the controlled plant. In case x is unreachable in the controlled

plant, a positive cost equal to $-p(x)$ is added to the net cost as a penalty, else no cost is added to the net cost. Similarly, if $p(x) > 0$ for some $x \in X$, then x is an undesired state, and it should remain unreachable in the controlled plant. In case x is reachable in the controlled plant, a cost equal to $p(x)$ is added to the net cost as a penalty, else no cost is added to the net cost.

We assume for simplicity, as above for the cost of control, that the penalty of control is a one-time penalty, i.e. the penalty of reaching an undesired state does not depend on the number of times that state is visited. The justification for this simplified assumption is similar to that given above for the one-time cost of control assumption.

Remark 1 If $c(\cdot, \sigma) = \infty$ for an event $\sigma \in \Sigma$, then σ should not be disabled by any supervisor at any state of G. Thus an infinite cost of control of an event captures the notion of *uncontrollable* events [23, 24]. The notion of desired or *target* behavior can be captured by defining the penalty function to be infinity for those states that are reachable by the strings in the undesired behavior, and any fixed negative real for those states that are reachable by strings in the desired behavior. However, this requires that the state machine G be refined [7, 11] with respect to the given target behavior, so that the states corresponding to the desired and undesired behavior can be uniquely identified, and the penalty function is unambiguously defined. This is explained formally in Section 4.

With the above definitions of the cost and penalty of control functions we can define the optimal supervisory control problem.

Definition 1 For any supervisor $S : X \to 2^{\Sigma}$, the *net cost* of using S, denoted $C(S)$, is defined to be:

$$C(S) := \sum_{x \in Re(G_S)} \left[\sum_{\sigma \in S(x)} c(x, \sigma) \right] + \sum_{\substack{x \in Re(G_S), \\ p(x) > 0}} p(x) + \sum_{\substack{x \notin Re(G_S), \\ p(x) < 0}} -p(x),$$

where $Re(G_S)$ is the set of reachable states[1] in G_S.

Thus the net cost of control of using S consists of sum of three terms: (i) The first term corresponds to the cost of disabling the events by S. $\sum_{\sigma \in S(x)} c(x, \sigma)$ is the total cost of disabling events at state $x \in X$. Thus $\sum_{x \in Re(G_S)} \sum_{\sigma \in S(x)} c(x, \sigma)$ is the total cost of disabling the events by S. (ii) The second term, $\sum_{x \in Re(G_S), p(x) > 0} p(x)$, denotes the penalty of reaching undesired states. (iii) Finally, the third term $\sum_{x \notin Re(G_S), p(x) < 0} -p(x)$, denotes the penalty of not reaching the desired states.

Optimal Supervisory Control Problem 1 (OSCP1): Let a plant $G := (X, \Sigma, \delta, x_0)$, a cost of control function $c : X \times \Sigma \to \mathcal{R}^+$, and a penalty of control function $p : X \to \mathcal{R}$ be given. Design a supervisor $S : X \to 2^{\Sigma}$ such that the net cost is minimized, i.e. determine

$$\arg \left\{ \min_S C(S) \right\}.$$

[1] Given a state machine $V := (Q, \Sigma, \rho, q_0)$, the set of reachable states $Re(V)$ is recursively defined as: (i) $q_0 \in Re(V)$, and (ii) $q \in Re(V), \exists \sigma \in \Sigma : \rho(q, \sigma)$ is defined $\Rightarrow \rho(q, \sigma) \in Re(V)$.

2.1 Partial State Observation

In OSCP1, a supervisor while deciding its control actions assumes that a complete state information of G is available. We pose another optimal supervisory control problem in which a complete state information is not available, and there exists a mask $\Psi : X \to Y$ defined from the state space X to an observation space Y such that for each $x \in X$, $\Psi(x) \in Y$ is the state value observed by a supervisor. A supervisor in this case is given by a map $S' : Y \to 2^{\Sigma}$. Since a supervisor takes a control action based on observing a state $y \in Y$, given any $y \in Y$, the same control action is taken at all states in the set $\Psi^{-1}(y) := \{x \in X \mid \Psi(x) = y\}$. Thus corresponding to a supervisor $S' : Y \to 2^{\Sigma}$, we can equivalently define a supervisor $S : X \to 2^{\Sigma}$ with the constraint C1:

C1: $\forall x_1, x_2 \in X : \Psi(x_1) = \Psi(x_2) \Rightarrow S(x_1) = S(x_2)$.

Optimal Supervisory Control Problem 2 (OSCP2): Let a plant $G := (X, \Sigma, \delta, x_0)$, a cost of control function $c : X \times \Sigma \to \mathcal{R}^+$, a penalty of control function $p : X \to \mathcal{R}$, and a mask $\Psi : X \to Y$ be given. Design a supervisor $S' : Y \to 2^{\Sigma}$ (equivalently a supervisor $S : X \to 2^{\Sigma}$ satisfying C1) such that the net cost is minimized, i.e. determine

$$\arg \left\{ \min_{S:\ \text{C1 holds}} C(S) \right\}.$$

Remark 2 The difference between OSCP1 and OSCP2 is that, in OSCP2, the minimization is performed over all supervisors that also satisfy the constraint C1, whereas no such constraint exists in OSCP1. It is easily seen that given an instance of OSCP1, it can be reduced to an instance of OSCP2 by setting the mask function to be the identity function. We show in the next section that given an instance of OSCP2, it can be reduced to an instance of OSCP1 by suitably modifying the the cost of control function and the graph representing the plant. Thus the two formulations are reducible to each other.

In the formulation of OSCP2, a state based mask function is used. This is in contrast to the setting of supervisory control, where usually an event based mask is used. However, an event based mask can be used to obtain a state based mask by first constructing a state estimator, as in [17], and next identifying all the states with identical state estimates to have equal mask value. Thus it is possible to reduce an optimal control problem under partial observation of events to one under partial observation of states.

3 Solution using Network Flow Algorithm

In this section we provide a solution to the optimal supervisory control problems introduced in section 2. The problem is to determine for each transition in the state machine G whether to disable or enable it, so that the net cost is minimized. We show that this problem is equivalent to determining an optimal partition of the state space X into the set of states that remain reachable in the controlled plant, and the set of remaining unreachable states. The desired optimal partition is determined using the max-flow min-cut theorem [15], a technique for optimal partitioning of directed graphs.

3.1 Max-Flow Min-Cut Theorem

Interested readers are referred to [15] for a formal and elaborate description of the max-flow min-cut theorem. Informally described, in its simplest form, a *flow network* is represented as a weighted directed graph having a single *source* node, from where the flow starts, and a single *terminal* node where the flow terminates. The weights on the directed edges of the graph represent the maximum flow capacities of the corresponding edges (the minimum capacity is zero unless specified). Formally,

Definition 2 A *flow network* N is a weighted directed graph described by a triple $N = (V, E, u)$, where V denotes the set of vertices or nodes of N; $E \subseteq V^2$—subset of *ordered* pairs of V^2—denotes the set of directed edges or links of N; and $u : E \rightarrow \mathcal{R}^+$ denotes the maximum capacity function of links. V contains two special nodes s and t, the source node and the terminal node.

The basic flow optimization problem is to determine for a given flow network a flow of maximum value between its source and terminal nodes subject to the edge capacity constraints, where a flow and its value is defined as follows:

Definition 3 A *flow* for a network N is a map $f : E \rightarrow \mathcal{R}^+$ such that

1. $\forall e \in E : f(e) \leq u(e)$, and
2. $\sum_{v \in V:(s,v) \in E} f((s,v)) = \sum_{v' \in V:(v',t) \in E} f((v',t))$, and
3. $\forall v \in V, v \neq s, v \neq t : \sum_{v' \in V:(v,v') \in E} f((v,v')) = \sum_{v'' \in V:(v'',v) \in E} f((v'',v))$.

$\sum_{v \in V:(s,v) \in E} f((s,v)) = \sum_{v' \in V:(v',t) \in E} f((v',t))$ is called the *value* of f. A *max-flow* is a flow of maximum (flow) value.

Thus a flow is an assignment of a positive number to each edge in the flow network which corresponds to the amount of flow on that edge, satisfying three constraints. Firstly, amount of flow through each edge is no greater than its capacity; secondly, the net flow out of source node equals the net flow into the terminal node; and finally, the net flow out of intermediate nodes is zero. The value of a flow equals the net flow out of the source node (equivalently, the net flow into the terminal node).

Definition 4 A *cut* of a network N is a partition of V such that s and t are in different partitions. Let $V_s \subseteq V$ and $V_t := V - V_s$ denote the partitions of a cut such that $s \in V_s$ and $t \in V_t$. Then the *capacity* of this cut is defined as:

$$\sum_{(i,j) \in (V_s \times V_t) \cap E} u((i,j))$$

A *min-cut* is a cut of minimum capacity.

Thus any partition of the nodes in N such that the source node and the terminal node belong to different partitions is called a cut. The capacity of a cut equals the sum of capacity of those edges that emerge out of a node contained in the partition containing the source node, and terminate at a node contained in the partition containing the terminal node.

Theorem 1 (Max-flow Min-cut) [15]: The value of a max-flow of a flow network equals the capacity of a min-cut of that network.

Algorithms for computing a max-flow can be found in [15]. In this paper, we are interested in computing a cost minimizing supervisor. This is shown to be equivalent to determining a min-cut for a suitably defined flow network, which in view of Theorem 1 can be computed using any of the max-flow computations.

3.2 Solution of OSCP1

In this subsection we provide a solution for the OSCP1 using the network flow technique discussed in the previous subsection. It is clear that each supervisor $S : X \rightarrow 2^{\Sigma}$ partitions the state space X into $Re(G_S) \cup (X - Re(G_S))$, the sets of reachable and unreachable states in the controlled plant G_S. We define a supervisor to be parsimonious if and only if it disables those transitions that are defined from a state in $Re(G_S)$ to a state in $X - Re(G_S)$. Formally,

Definition 5 A supervisor $S : X \rightarrow 2^{\Sigma}$ is said to be *parsimonious* if and only if for each state $x \in X$ and event $\sigma \in \Sigma$:

$$\sigma \in S(x) \Leftrightarrow [x \in Re(G_S), \delta(x, \sigma) \notin Re(G_S)].$$

We prove that parsimonicity is a necessary condition for optimality of a supervisor.

Lemma 1 If S is an optimal supervisor, then S is parsimonious.

Proof: Assume for contradiction that S is optimal but not parsimonious. Then there exist an event $\sigma \in \Sigma$, states $x_1, x_2 \in Re(G_S)$ such that $\delta(x_1, \sigma) = x_2$ and $\sigma \in S(x_1)$, i.e. σ is disabled at x_1. Consider a supervisor S' that exercises same control action as S does, except that at state x_1 it does not disable the event σ, i.e. $\sigma \notin S'(x_1)$. Then $C(S') = C(S) - c(x_1, \sigma) < C(S)$, for $c(x_1, \sigma) > 0$. Thus we obtain a contradiction to the optimality of S. ■

The following is an immediate Corollary of Lemma 1.

Corollary 1 OSCP1 is equivalent to determining

$$\arg \left\{ \min_{S:S \text{ parsimonious}} C(S) \right\}.$$

Proof: Follows from the fact that parsimonicity is a necessary condition for optimality, and the definition of OSCP1. ■

In the next Theorem we provide a technique for solving the OSCP1 using the max-flow min-cut theorem. First we define a flow network N_G corresponding to the state machine G, the cost of control function c, and the penalty of control function p as follows:

Definition 6 Given a SM $G := (X, \Sigma, \delta, x_0)$ with cost of control function $c : X \times \Sigma \to \mathcal{R}^+$, and penalty of control function $p : X \to \mathcal{R}$, a flow network, denoted N_G, is defined to be $N_G := (V_G, E_G, u_G)$, where

1. $V_G := X \cup \{s, t\}$ with $s, t \notin X$
2. $E_G := \{(x_1, x_2) \in X \times X \mid \exists \sigma \in \Sigma \text{ s.t. } \delta(x_1, \sigma) = x_2\}$
   $\cup \{(x, t) \in X \times \{t\} \mid p(x) > 0\}$
   $\cup \{(s, x) \in \{s\} \times X \mid p(x) < 0\}$
3. $\forall (x_1, x_2) \in E_G \cap (X \times X) : u_G((x_1, x_2)) := \sum_{\sigma \in \Sigma : \delta(x_1, \sigma) = x_2} c(x_1, \sigma)$
   $\forall x \in X \text{ s.t. } p(x) > 0 : u_G((x, t)) := p(x)$
   $\forall x \in X \text{ s.t. } p(x) < 0 : u_G((s, x)) := -p(x)$

Thus the node set of N_G is obtained by adding, to the state set X of G, two extra nodes s and t—the source and the terminal node, respectively. The edge set of N_G consists of: (i) An edge from $x_1 \in X$ to $x_2 \in X$ if there exists a transition from x_1 to x_2 in G. The capacity of such an edge equals the sum of cost of disabling each transition from x_1 to x_2. (ii) An edge from each $x \in X$ for which $p(x) > 0$ to the terminal node t, with capacity $p(x)$. (iii) An edge from the source node s to each $x \in X$ for which $p(x) < 0$, with capacity $-p(x)$.

Example 2 Consider the plant G shown in Figure 2, with the state set $X = \{1, 2\}$, the event set $\Sigma = \{a, b\}$, the initial state $x_0 = 1$, and the transition function $\delta(1, a) = 2, \delta(2, a) = \delta(2, b) = 2$. Then $L(G) = \overline{a(a + b)^\star}$. Let the cost of control function be defined as $c(a) = 10$

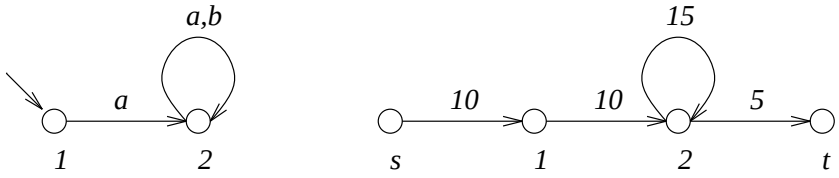


Figure 2: Diagram illustrating Construction of N_G

and $c(b) = 5$, and the penalty of control function be defined as $p(1) = -10$ and $p(2) = 5$. Then the flow network N_G, corresponding to the plant G, obtained using Definition 6 is shown in Figure 2.

Definition 7 Given a supervisor $S : X \to 2^\Sigma$, the cut of flow network N_G induced by S is defined to be: $[Re(G_S) \cup \{s\}] \cup [(X - Re(G_S)) \cup \{t\}]$. Given a cut $V_s \cup (V_G - V_s)$ of N_G, where $V_s \subseteq V_G$, $s \in V_s$ and $t \notin V_s$, the parsimonious supervisor $S : X \to 2^\Sigma$ induced by the cut is defined to be: for each $x \in X$ and $\sigma \in \Sigma$, $\sigma \in S(x)$ if and only if $x \in V_s$ and $\delta(x, \sigma) \notin V_s$.

Theorem 2 (Solution of OSCP1) The supervisor induced by a min-cut of N_G is a solution of OSCP1.

We prove a Lemma before proving Theorem 2.

Lemma 2 If S is a parsimonious supervisor, then $C(S)$ equals the capacity of the cut of N_G induced by S.

Proof: Consider the cut $[Re(G_S) \cup \{s\}] \cup [(X - Re(G_S)) \cup \{t\})$ induced by S. Then capacity of this cut is given by the sum of capacities of all the edges that originate from a node in the set $Re(G_S) \cup \{s\}$, and terminate at a node in the set $(X - Re(G_S)) \cup \{t\}$. Let the set of such edges be denoted as E_S, i.e.,

$$E_S := \{(x_1, x_2) \in E_G \mid x_1 \in Re(G_S) \cup \{s\} \text{ and } x_2 \in (X - Re(G_S)) \cup \{t\}\}$$

Then the capacity of the cut of N_G induced by S equals $\sum_{e \in E_S} u_G(e)$. We note that

$$\begin{aligned} E_S \;=\; & \{(x_1, x_2) \in Re(G_S) \times (X - Re(G_S)) \mid \exists \sigma \in \Sigma \text{ s.t. } \delta(x_1, \sigma) = x_2\} \\ & \cup \{(x, t) \mid x \in Re(G_S), p(x) > 0\} \\ & \cup \{(s, x) \mid x \notin Re(G_S), p(x) < 0\} \end{aligned}$$

Hence

$$\sum_{e \in E_S} u_G(e) = \sum_{x \in Re(G_S)} \left[\sum_{\sigma \in \Sigma : \delta(x,\sigma) \in X - Re(G_S)} c(x, \sigma) \right] + \sum_{x \in Re(G_S) : p(x) > 0} p(x) + \sum_{x \notin Re(G_S) : p(x) < 0} -p(x).$$

Since S is parsimonious, for each $x \in Re(G_S)$, the set $\{\sigma \in \Sigma \mid \delta(x, \sigma) \in X - Re(G_S)\} = \{\sigma \in \Sigma \mid \sigma \in S(x)\}$. Hence $\sum_{e \in E_S} u_G(e) = C(S)$. ■

Proof (of Theorem 2): Consider the parsimonious supervisor S induced by a min-cut of N_G. Then, from the result of Lemma 2, we obtain that $C(S)$ equals capacity of min-cut of N_G. In view of Corollary 1, in order to prove the optimality of S it suffices to show that if S' is any other parsimonious supervisor, then $C(S') \geq C(S)$. Since S' is also parsimonious, it follows from Lemma 2 that $C(S')$ equals the capacity of cut of N_G induced by S' which is greater than or equal to the capacity of the min-cut (by definition of min-cut). Since $C(S)$ equals the capacity of the min-cut (by construction of S), we obtain the desired inequality: $C(S') \geq C(S)$. ■

3.3 Solution of OSCP2

In this subsection we provide a solution for the OSCP2. In this setting, given a plant G, a cost of control function c, a penalty of control function p, and an observation mask Ψ, the control objective is to determine an optimal supervisor $S : X \to 2^{\Sigma}$ satisfying the constraint C1. The constraint C1 can be satisfied by making a few modifications in G, and in the cost of control function c as described below. Firstly, we modify the state machine G; the modified state machine is denoted as G'.

Definition 8 Given $G = (X, \Sigma, \delta, x_0)$, the modified state machine G' corresponding to the constraint C1 is the quadruple $G' = (X, \Sigma', \delta', x_0)$, where

1. $\Sigma' := \Sigma \cup \{\theta\}$ with $\theta \notin \Sigma$
2. $\forall x \in X, \forall \sigma' \in \Sigma' : \delta'(x, \sigma') := \delta(x, \sigma')$ if $\sigma' \neq \theta$
3. $\forall \sigma \in \Sigma, x_1, x_2 \in X$ s.t. $\Psi(x_1) = \Psi(x_2), \delta(x_1, \sigma) \neq \delta(x_2, \sigma)$:
   $\delta'(\delta(x_1, \sigma), \theta) := \delta(x_2, \sigma)$ and $\delta'(\delta(x_2, \sigma), \theta) := \delta(x_1, \sigma)$.

Thus G' is obtained by adding in G, an oppositely directed pair of transitions labeled θ, between the pair of states reached by executing a common event from a pair of states that look alike under Ψ. No such transition is added if the *same* state is reached after executing a common event from a pair of states that look alike under Ψ.

Next, the cost of control function $c : X \times \Sigma \to \mathcal{R}^+$ is extended to $c' : X \times \Sigma' \to \mathcal{R}^+$ as:

$$\forall x \in X, \sigma' \in \Sigma' : c'(x, \sigma') := \begin{cases} c(x, \sigma') & \text{if } \sigma' \in \Sigma \\ \infty & \text{if } \sigma' = \theta \end{cases}$$

Thus the cost of control function c' is the extension of c obtained by assigning the cost of disabling the event θ to be infinity. For simplicity of notation, Let OSCP1 with respect to G', with cost of control c', and penalty of control p be denoted as OSCP1′.

Theorem 3 (Solution of OSCP2) OSCP2 is equivalent to OSCP1′.

We prove a few Lemmas before proving Theorem 3.

Lemma 3 If S is parsimonious, then S disables a transitions leading from states in $Re(G_S)$ into a state $x \in X$ if and only if it disables all transitions from states in $Re(G_S)$ leading into x.

Proof: If S disables some transitions, but not all transitions from states in $Re(G_S)$ leading into a state $x \in X$, then x remains reachable in G_S, i.e. $x \in Re(G_S)$, which contradicts that S is parsimonious. ■

Lemma 4 There exists a solution $S : X \to 2^{\Sigma'}$ of OSCP1′ such that

1. S never disables the event θ.
2. S satisfies constraint C1.

Proof: 1. Let $S : X \to 2^{\Sigma'}$ be a solution of OSCP1′. If $C(S) < \infty$, then it is clear that S never disables the event θ. Next consider the case when $C(S) = \infty$. If S ever disables the event θ, then consider a supervisor S' which takes the same control action as S does, except that it never disables the event θ. Then by optimality of S, $C(S') \geq C(S) = \infty$, which implies that $C(S') = \infty$. Hence S' is optimal, and it never disables the event θ.

2. Let $S : X \to 2^{\Sigma'}$ be a solution of OSCP1′. We show that if $x_1, x_2 \in Re(G_S)$ are such that $\Psi(x_1) = \Psi(x_2)$, then $S(x_1) = S(x_2)$. In other words, if an event $\sigma \in \Sigma$ is defined at such a pair of states $x_1, x_2 \in Re(G_S)$, then either σ is disabled at both x_1 and x_2, or it is disabled at neither of x_1 and x_2. Consider such a pair of states $x_1, x_2 \in Re(G_S)$ and an event $\sigma \in \Sigma$. Then either $\delta(x_1, \sigma) = \delta(x_2, \sigma)$, or $\delta(x_1, \sigma) \neq \delta(x_2, \sigma)$. If $\delta(x_1, \sigma) = \delta(x_2, \sigma)$, then S disables the event σ at both the states, or at neither of the states. This follows from the fact that any optimal supervisor is also parsimonious (Lemma 1), and any parsimonious supervisor disables either all transition leading into a state, or none of them (Lemma 3). Thus constraint C1 is satisfied in this case. If $\delta(x_1, \sigma) \neq \delta(x_2, \sigma)$, then according to the construction of G', these states are connected by a pair of oppositely directed transitions labeled θ. Since S never disables the event θ (from part 1 above), the states $\delta(x_1, \sigma)$ and $\delta(x_2, \sigma)$ do not belong to separate partitions induced by S. Suppose that they both are in $Re(G_S)$, then due to parsimonicity of S, σ is enabled at both x_1 and x_2. On the other hand, if they both are in $X - Re(G_S)$, then clearly, σ is disabled at x_1 and x_2. ■

Proof (of Theorem 3): We first show that if $S : X \to 2^{\Sigma'}$ is a solution of OSCP1′, then it is also a solution for OSCP2. In view of Lemma 4, it can be assumed, without loss of generality, that S never disables the event θ and satisfies C1. Since S never disables the event θ and satisfies C1, it can be viewed as a map $S : X \to 2^{\Sigma}$ satisfying C1, and hence it can also be used as a supervisor under partial state observation. Assume for contradiction that S is not a solution of OSCP2. Let $S' : X \to 2^{\Sigma}$ with $S' \neq S$ be a solution of OSCP2, then we must have $C(S') < C(S)$, where S, S' are treated as *feasible* solutions of OSCP2. Let $C'(S), C'(S')$ denote the net costs of using S and S' respectively, when S, S' are treated as *feasible* solutions of OSCP1′. Since (i) S and S' do not disable the event θ, and (ii) for each $x \in X$ and $\sigma \in \Sigma$, $c'(x, \sigma) = c(x, \sigma)$, we have $C'(S) = C(S)$ and $C'(S') = C(S')$. Since $C(S') < C(S)$, we obtain $C'(S') < C'(S)$. This contradicts the optimality of S (treated as a solution of OSCP1′).

Next we show that if $S : X \to 2^{\Sigma}$ is a solution of OSCP2, then it is also a solution of OSCP1′. It is clear that S can also be viewed as a map $S : X \to 2^{\Sigma'}$. Assume for contradiction that S is not a solution for OSCP1′. Let $S' : X \to 2^{\Sigma'}$ with $S' \neq S$ be a solution of OSCP1′, then we must have $C'(S') < C'(S)$. Also, from Lemma 4, S' never disables the event θ and satisfies C1. Thus S' can be used as a supervisor under partial state observation. As above, $C'(S) = C(S)$ and $C'(S') = C(S')$. However, since $C'(S') < C'(S)$, we obtain $C(S') < C(S)$. This is a contradiction to the optimality of S (treated as a solution of OSCP2). ■

4 Applications to Supervisory Control

It is clear from Theorem 3 that an instance of OSCP2 can be reduced to an instance of OSCP1 by suitably modifying the graph of the plant and the cost of control function. We show in this section that supervisory control problems under complete as well as partial observation can also be reduced to instances of OSCP1. We begin with the problem of com-

puting the supervisors under complete observation, which requires computation of supremal controllable sublanguage, and infimal controllable superlanguage.

4.1 Computations related to Controllability of DEDS's

We first consider the computation of supremal controllable sublanguage: Given a desired prefix closed behavior $K \subseteq L(G)$, compute the *supremal* sublanguage $K^{\uparrow} \subseteq K$ such that it is *controllable* [23], i.e. $K^{\uparrow}\Sigma_u \cap L(G) \subseteq K^{\uparrow}$, where $\Sigma_u \subseteq \Sigma$ denotes the set of uncontrollable events. A closed form expression for $K^{\uparrow}$ is given in [1], and an optimal algorithm for computing $K^{\uparrow}$ is given in [11].

In order to reduce the problem of computing $K^{\uparrow}$ to an instance of OSCP1, we refine G with respect to K so that the states corresponding to strings in K are uniquely identified. This is done as follows. Let $V := (Q, \Sigma, \rho, q_0)$ be a trim deterministic state machine that generates K. The graph of V is made "complete" by adding a dump state d to its state set. If a certain event σ is not defined at some state $q \in Q$, then a transition labeled σ from the state q to the dump state d is added. Also, execution of any event in the dump state, leaves the system in that state. Formally, the *completion* of V is another state machine $V' := (Q', \Sigma, \rho', q_0)$, where $Q' := Q \cup \{d\}$ with $d \notin Q$, and

$$\forall q' \in Q', \sigma \in \Sigma : \rho'(q', \sigma) := \begin{cases} \rho(q', \sigma) & \text{if } q' \in Q \text{ and } \rho(q', \sigma) \text{ is defined} \\ d & \text{otherwise} \end{cases}$$

It is clear that $L(V') = \Sigma^{\star}$. Consider the synchronous composition [8, 11] of G and V': $G \square V' := (X \times Q', \Sigma, \alpha, (x_0, q_0))$, where

$$\forall x \in X, q' \in Q', \sigma \in \Sigma : \alpha((x, q'), \sigma) := \begin{cases} (\delta(x, \sigma), \rho'(q', \sigma)) & \text{if } \delta(x, \sigma), \rho'(q', \sigma) \text{ are defined} \\ \text{undefined} & \text{otherwise} \end{cases}$$

It is easily shown that $L(G \square V') = L(G) \cap L(V') = L(G) \cap \Sigma^{\star} = L(G)$. $G \square V'$ is called *refinement* of G with respect to K. Note that the the first coordinate of a state in $G \square V'$ corresponds to a state of G, and the second coordinate to a state of V'.

Example 3 Let G be the plant as in Example 2, and the target language K be given by $K = \overline{(ab)^{\star}} \subseteq L(G) = \overline{a(a+b)^{\star}}$. The generator V for the language K is shown in Figure 3, with the state set $Q = \{1', 2'\}$, the initial state $q_0 = 1'$, and the transition function $\rho(1', a) = 2', \rho(2', b) = a$. The state machine V' obtained by completing the graph of V, and the state machine $G \square V'$ obtained by synchronous composition of G and V' are both shown in Figure 3. Note that $L(V') = (a+b)^{\star} = \Sigma^{*}$, and $L(G \square V') = \overline{a(a+b)^{\star}} = L(G)$.

Lemma 5 Given a string $s \in \Sigma^{\star}$, $s \in L(G) - K$ if and only if the second coordinate of the state reached by executing s in $G \square V'$ is d.

Proof: Straightforward. ∎

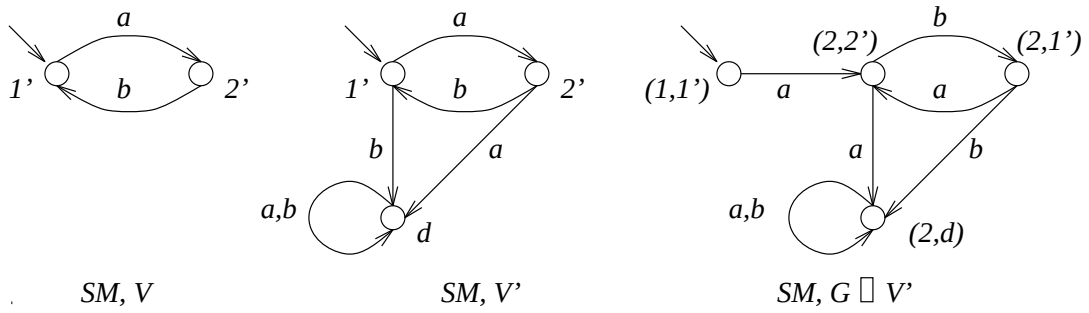


Figure 3: Diagram illustrating construction of $G\Box V'$

Example 4 Consider the state machine $G\Box V'$ of Example 3. Then the strings, execution of which take to the state $(2,d)$, belong to $L(G) - K = \overline{a(a+b)^*} - \overline{(ab)^*}$. Also, the strings, execution of which take to states $(1,1')$ or $(2,2')$ or $(2,1')$ belong to the language $K = \overline{(ab)^*}$.

The result of Lemma 5 can be used to unambiguously identify the desired and the undesired states in $G\Box V'$. The next Lemma proves that it is also possible to obtain the generator for $K^{\uparrow}$ by partitioning the graph of $G\Box V'$ into sets of reachable and unreachable states.

Lemma 6 [11, Proposition 3.6] Let $s, t \in K$ be such that $\alpha((x_0, q_0), s) = \alpha((x_0, q_), t)$, then $s \in K^{\uparrow}$ if and only if $t \in K^{\uparrow}$.

Based on the result of Lemma 5, we define a penalty of control function $p : X \times Q' \to \mathcal{R}$ for $G\Box V'$ as:

$$\forall (x, q') \in X \times Q' : p((x, q')) := \begin{cases} \infty & \text{if } q' = d \\ -p_0 & \text{otherwise,} \end{cases} \tag{1}$$

where $p_0 \in \mathcal{R}^+$ is any positive real. Since the penalty of control of a state in $G\Box V'$ with second coordinate d is infinity, it should remain unreachable in an optimally controlled plant, and since the penalty of control of all other states is $-p_0$, as many such states as possible should remain reachable in an optimally controlled plant.

Example 5 Consider the state machine $G\Box V'$ of Example 3. Then in order to compute the supremal controllable sublanguage of $K = \overline{(ab)^*}$ with respect to G of Example 2, we define the penalty of control function as: $p[(2,d)] = \infty, p[(1,1')] = p[(2,2')] = p[(2,1')] = -p_0$.

Next we define a cost of control function $c : X \times \Sigma \to \mathcal{R}^+$ as:

$$\forall x \in X, \sigma \in \Sigma : c(x, \sigma) := \begin{cases} \infty & \text{if } \sigma \in \Sigma_u \\ \frac{p_0}{|e|+1} & \text{otherwise,} \end{cases} \tag{2}$$

where $|e|$ denotes the total number of transitions in the graph of $G\Box V'$. Since the cost of control of an uncontrollable event is infinity, it should not be disabled by an optimal supervisor, and since the cost of control of a controllable event is $\frac{p_0}{|e|+1}$, as few such events as possible should be disabled. With the above definitions of cost and penalty of control functions, we prove in the following Theorem that the computation of $K^{\uparrow}$ can be posed as an instance of OSCP1.

Theorem 4 If $K^{\uparrow} \neq \emptyset$, then $K^{\uparrow}$ equals the language generated by $G \square V'$ under the control of a solution of OSCP1 with respect to $G \square V'$, with cost of control of function as in Equation 2, and penalty of control function as in Equation 1.

Proof: Let $S : X \times Q' \rightarrow 2^{\Sigma}$ be the parsimonious supervisor induced by the partition of $G \square V'$ into the set of states that correspond to the supremal controllable sublanguage, and the set of remaining states, i.e., S disables those transitions that are defined from states corresponding to the supremal controllable sublanguage to the set of remaining states. That such a partition exists follows from Lemma 6 and the fact that $K^{\uparrow} \neq \emptyset$. It is clear that S does not disable any uncontrollable events (otherwise the controlled system behavior is not a controllable language), and no state with second coordinate d remains reachable under the control of S (otherwise the controlled system behavior is not a sublanguage of K). Hence $C(S) < \infty$. Let $S' : X \times Q' \rightarrow 2^{\Sigma}$ be a solution of OSCP1. Then it follows from the optimality of S' that $C(S') \leq C(S) < \infty$. Thus S' does not disable any uncontrollable event and no state with second coordinate d remains reachable in the controlled system (otherwise $C(S') = \infty$). Since S' does not disable any of the uncontrollable events, the controlled system behavior under its control is controllable [11, Lemma 2.7]. Also, since no state with second coordinate d remains reachable under the control of S', the controlled system behavior under the control of S' is a sublanguage of K. Assume for contradiction that the controlled system behavior under the control of S' is not the supremal controllable sublanguage of K. So, there exists at least one state with second coordinate unequal to d such that it is reachable under the control of S, and it is unreachable under the control of S'. Hence $C(S') - C(S) \geq p_0 - n\frac{p_0}{|e|+1}$, where n is the difference between the number of controllable transitions disabled by S and the number of controllable transition disabled by S'. Since $n \leq |e|$, the total number of transitions in $G \square V'$, $n\frac{p_0}{|e|+1} < p_0$. In other words, $C(S') - C(S) > 0$. This is a contradiction to the optimality of S'. ■

Next we consider the problem of computing the infimal controllable superlanguage: Given a desired prefix closed behavior $K \subseteq L(G)$, compute the *infimal* superlanguage $K^{\downarrow} \supseteq K$ such that it is *controllable*. A closed form expression for $K^{\downarrow}$ and an algorithm for computing it is given in [14]. We use the same notations as above. The following modification is made to the penalty of control function.

$$\forall (x, q') \in X \times Q' : p((x, q')) := \begin{cases} p_0 & \text{if } q' = d \\ -\infty & \text{otherwise} \end{cases} \tag{3}$$

Since penalty of control of a state with second coordinate d is p_0, as few such states as possible should remain reachable in the controlled system, and since the penalty of control of a state with second coordinate unequal to d is $-\infty$, all such states should remain reachable in the controlled system.

Example 6 Consider the state machine $G \square V'$ of Example 3. Then in order to compute the infimal controllable superlanguage of $K = \overline{(ab)^*}$ with respect to G of Example 2, we define the penalty of control function as follows: $p[(2, d)] = p_0, p[(1, 1')] = p[(2, 2')] = p[(2, 1')] = -\infty$.

With the above modification in the penalty function, we prove in the following Theorem that the computation of $K^{\downarrow}$ can be posed as an instance of OSCP1.

Theorem 5 $K^{\downarrow}$ equals the language generated by $G\Box V'$ under the control of a solution of OSCP1 with respect to $G\Box V'$, with cost of control function as in Equation 2, and penalty of control function as in Equation 3.

Proof: Similar to that of Theorem 4. ∎

4.2 Computations related to Observability of DEDS's

Suppose that the supervisor's observation of events is filtered through a mask of the type $M : \Sigma \to \Lambda \cup \{\epsilon\}$. Computation of a supervisor under such a partial observation requires computation of languages such as supremal normal sublanguage, infimal normal/observable superlanguage, supremal controllable and normal sublanguage, infimal controllable and normal/observable superlanguage etc. We show that each of these computations can be reduced to instances of OSCP1. Our techniques illustrate how supervisory control under partial observation can be solved using a state-feedback type control on a suitably refined state machine representation of the plant.

We first consider the problem of computing the supremal normal sublanguage: Given a desired prefix closed language $K \subseteq L(G)$, compute the *supremal* sublanguage $K^{\circ} \subseteq K$, such that it is *normal*, i.e. $M^{-1}(M(K^{\circ})) \cap L(G) \subseteq K^{\circ}$. The existence of K° is shown in [16], and a closed form expression for K° is given in [1, 11]. We first refine the state machine G with respect to V (the generator for K), and mask function M, so that the states corresponding to K° are uniquely identified. We begin by constructing the machine $G_1 := G\Box V'$. Recall that $L(G_1) = L(G)$. Using G_1 we construct a machine that generates the language $M^{-1}M(L(G_1))$ by employing the following algorithm:

Algorithm 1

1. Replace each transition $\sigma \in \Sigma$ in G_1 by the transition $M(\sigma) \in \Lambda \cup \{\epsilon\}$. Call this machine G_2; clearly, $L(G_2) = M(L(G_1))$.

2. Construct a deterministic machine G_3 which is language equivalent to G_2 [7]. Then the state space of G_3 is $2^{X \times Q'}$, and $L(G_3) = L(G_2) = M(L(G_1))$.

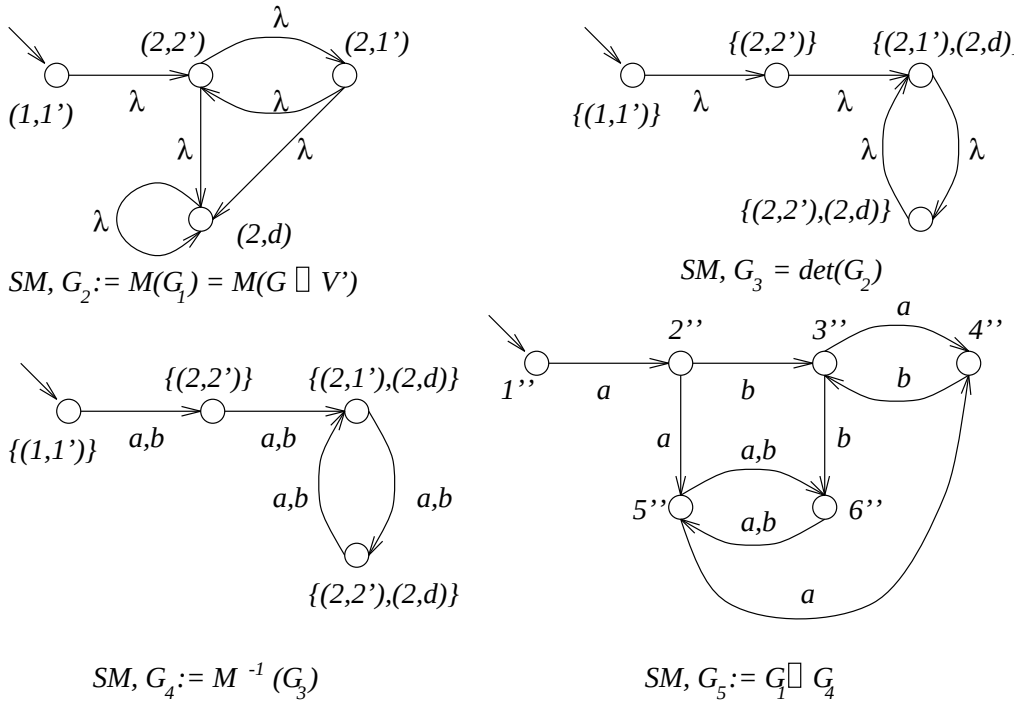
3. Replace each transition $\lambda \in \Lambda$ in machine G_3 by the events in the set $M^{-1}(\lambda) := \{\sigma \in \Sigma \mid M(\sigma) = \lambda\}$. Also, at each state in G_3, add self-loops corresponding to the events in the set $M^{-1}(\epsilon) = \{\sigma \in \Sigma \mid M(\sigma) = \epsilon\}$. Call this machine G_4; clearly, $L(G_4) = M^{-1}(L(G_3)) = M^{-1}(L(G_2)) = M^{-1}M(L(G_1))$.

G_4 has a nice property that if two strings $s, t \in L(G_4)$ are such that $M(s) = M(t)$, then the state reached by executing them are the same. This follows from the observations that (i) the state reached by executing s in G_4 is the same as that reached by executing $M(s)$ in G_3 (by construction), (ii) since $M(s) = M(t)$ and G_3 is deterministic, the same state is reached

by executing $M(t)$ in G_3; and (iii) by construction, this is the state reached by t in G_4. We exploit the above property of G_4 in identifying the states corresponding to the strings in the language K°. This, however, requires the construction of machine $G_5 := G_1 \Box G_4$, for which it is clear that $L(G_5) = L(G_1) \cap L(G_4) = L(G) \cap M^{-1}M(L(G)) = L(G)$, and the state space of G_5 equals $X \times Q' \times 2^{X \times Q'}$. Construction of state machines G_1 through G_5, their state spaces, and their languages are summarized in Table 1. Note that in Table 1 and in Figure

SM	Construction	State space	Language
G_1	$G \Box V'$	$X \times Q'$	$L(G)$
G_2	$M(G_1)$	$X \times Q'$	$M(L(G))$
G_3	$det(G_2)$	$2^{X \times Q'}$	$M(L(G))$
G_4	$M^{-1}(G_3)$	$2^{X \times Q'}$	$M^{-1}(M(L(G))$
G_5	$G_1 \Box G_4$	$X \times Q' \times 2^{X \times Q'}$	$L(G)$

Table 1: Various Machines used for computation of K°



1'' = (1,1'), {(1,1')}; 3'' = (2,1'), {(2,1'), (2,d)}; 5'' = (2,d), {(2,1'), (2,d)}

2'' = (2,2'), {(2,2')}; 4'' = (2,2'), {(2,2'), (2,d)}; 6'' = (2,d), {(2,2'), (2,d)}

Figure 4: Diagram illustrating construction of G_2, G_3, G_4, G_5

4, we have used the notations (i) $M(G_1)$ to represent that G_2 is obtained by "masking" the

transitions of G_1, (ii) $det(G_2)$ to represent that G_3 is obtained by "determinizing" G_2, and (iii) $M^{-1}(G_3)$ to represent that G_4 is obtained by "unmasking" the transitions of G_3.

Example 7 Consider the state machine $G_1 = G \Box V'$ of Example 3. Let the mask $M : \Sigma = \{a, b\} \rightarrow \{\lambda\}$ be defined as: $M(a) = M(b) = \lambda$. Then the state machine G_2, G_3, G_4 and G_5 obtained by using Algorithm 1 are shown in Figure 4.

We use $r = ((x, q'), \{(x_1, q'_1), (x_2, q'_2), \ldots, (x_r, q'_r)\}) \in X \times Q' \times 2^{X \times Q'}$ to denote a typical state of G_5, where $(x, q') \in X \times Q'$ and $\{(x_1, q'_1), (x_2, q'_2), \ldots, (x_r, q'_r)\} \in 2^{X \times Q'}$. We call $r' := (x, q')$ to be the G_1 part of r, and $R := \{(x_1, q'_1), (x_2, q'_2), \ldots, (x_r, q'_r)\}$ to be the G_4 part of r. We prove in the next Lemma that if r_1 and r_2 are two states of G_5 with identical G_4 part, then corresponding to each string, execution of which takes to state r_1, there exists another string, execution of which takes to state r_2, such that it looks like the former string. We call such a pair of states to be a matching pair. Formally,

Definition 9 Let $r_1 = (r'_1, R_1), r_2 = (r'_2, R_2) \in X \times Q' \times 2^{X \times Q'}$ be such that $R_1 = R_2$. Then the pair r_1 and r_2 of states is called a *matching* pair of states.

Example 8 Consider the SM G_5 of Example 7. The states $3'' = (2, 1'), \{(2, 1'), (2, d)\}$ and $5'' = (2, d), \{(2, 1'), (2, d)\}$ constitute a matching pair of states. Similarly, the pair of states $4'' = (2, 2'), \{(2, 2'), (2, d)\}$ and $6'' = (2, d), \{(2, 2'), (2, d)\}$ is another matching pair of states.

Lemma 7 Let $r_1, r_2 \in X \times Q' \times 2^{X \times Q'}$ be a matching pair of states. Then given a string s, execution of which takes to r_1, there exists a string t, execution of which takes to r_2, such that $M(s) = M(t)$.

Proof: First note that $r = (r', R)$ is a reachable state of G_5 if and only if $r' \in R$, i.e. if and only if the G_1 part of r is an element of the G_4 part of r. This follows from (i) if $s \in L(G_5)$ is a string, execution of which takes to r, then execution of s takes to the state r' in G_1, and to the state R in G_4, and (ii) execution of s takes to the state $R = \{(x_1, q'_1), (x_2, q'_2), \ldots, (x_r, q'_r)\}$ in G_4 implies that there exists a state $(x_j, q'_j) \in R$ such that execution of s takes to the state (x_j, q'_j) in G_1.

Consider then the states $r_1, r_2 \in X \times Q' \times 2^{X \times Q'}$ such that $R_1 = R_2 := R$. Then $r_1 = (r'_1, R)$ and $r_2 = (r'_2, R)$. It follows from the discussion in the preceding paragraph that $r'_1 \in R$ and $r'_2 \in R$. Let $s \in L(G_5)$ be a string, execution of which takes to state r_1 in G_5, then execution of s takes to state r'_1 in G_1, and to state R in G_4. Since both the states $r'_1, r'_2 \in R$, there exists at least one string t, execution of which takes to state r'_2 in G_1, and to state R in G_4, such that $M(t) = M(s)$. This follows from: (i) states r'_1 and r'_2 of G_2 belong to the same state R of G_3 if and only if there exists a string in $L(G_3) = L(G_2) \subseteq \Lambda^\star$ execution of which takes to state R in G_3, and execution of it takes to both states r'_1, r'_2 in G_2 (note that G_2 is a nondeterministic machine in general), and (ii) a string, execution of which takes to states r'_1, r'_2 in G_2, corresponds to two different strings in $L(G_1)$ having the same mask value. ■

The result of Lemma 7 can be used for identifying the strings in K°. Note that a string $s \in K - K^\circ$ if and only if there exists a string $t \in L(G) - K$ such that $M(t) = M(s)$. A state $r = (r', R)$ in G_5 corresponds to strings in $L(G) - K$ if and only if $r' = (x, d)$ for some $x \in X$. Thus strings in K, and those in $L(G) - K$ can be easily identified in G_5. Let r_1 and r_2 be a matching pair of states in G_5, with $R_1 = R_2 := R$, such that the second coordinate of r_1' does not equal d, whereas the second coordinate of r_2' equals d. Then as discussed above, strings leading to r_1 belong to K, and those leading to r_2 are in $L(G) - K$. Moreover, strings leading to r_1 are in $K - K^\circ$. This follows from Lemma 7, which asserts that corresponding to each string that leads to r_1 (i.e. the string is in K), there exists a string leading to r_2 (i.e. this string is in $L(G) - K$) such that it looks like the former string. Thus states corresponding to $K - K^\circ$ can also be identified in G_5 by first identifying all those matching pair of states for which exactly one of the states in each pair has its second coordinate of the G_1 part equal to d, and then among these matching pair of states, determining those states for which the second coordinate of the G_1 part does not equal d. Hence it is possible to obtain the generator for K° by partitioning the graph of G_5 into the set of reachable and unreachable states. Finally, we pose the problem of computing the supremal normal sublanguage as an instance of OSCP1 with respect to the machine obtained by adding an equally directed pair of transitions labeled θ between each matching pair of states in G_5. We call the machine thus obtained to be G_5'.

Define the following cost of control function for G_5':

$$\forall r \in X \times Q' \times 2^{X \times Q'}, \sigma' \in \Sigma \cup \{\theta\} : c(r, \sigma') := \begin{cases} \infty & \text{if } \sigma' = \theta \\ \frac{p_0}{|e|+1} & \text{otherwise,} \end{cases} \tag{4}$$

where $|e|$ denotes the total number of transitions in G_5'. The cost of control function for the event θ is infinity. Such a cost of control function ensures that the matching pair of states remain in the same partition of states induced by an optimal supervisor. Define the following penalty of control function on G_5':

$$\forall r = (r', R) \in X \times Q' \times 2^{X \times Q'} : p(r) := \begin{cases} \infty & \text{if } r' \in X \times \{d\} \\ -p_0 & \text{otherwise} \end{cases} \tag{5}$$

Thus the penalty of control is positive infinity whenever a state corresponds to strings in $L(G) - K$. Such states are undesired and should remain unreachable in the controlled plant under an optimal supervision. Other states have a negative penalty of control, $-p_0$, implying that such states are desired, and as many of them as possible should remain reachable in the controlled plant.

Example 9 Consider the state machine G_5 of Example 7. Then in order to compute the supremal normal sublanguage of the language $K = \overline{(ab)^*}$ with respect to plant G of Example 2 and mask $M(a) = M(b) = \lambda$, G_5' is constructed by adding, in G_5, a pair of oppositely directed transitions labeled θ between both the pair of matching states, namely, between $3''$ and $5''$, and between $4''$ and $6''$. The cost of disabling θ is assigned to be infinity. Finally, the penalty of control function is defined to be infinity for the states $5''$ and $6''$, and $-p_0$ for the

remaining states, $1'', 2'', 3''$, and $4''$. Note that the states $5''$ and $6''$ are such that for them the second part of the G_1 part equals d.

Theorem 6 If $K^\circ \neq \emptyset$, then K° equals the controlled plant behavior of G_5' under the control of a solution of OSCP1 with respect to G_5', with cost of control function as in Equation 4, and penalty of control function as in Equation 5.

Proof: Similar to that of Theorem 4. The key to the proof is that each matching pair of states belong to the same partition induced by an optimal supervisor. ■

It can be shown that the cost of control function in Equation 4 can be slightly modified for computing the supremal controllable and normal sublanguage of K:

$$\forall r \in X \times Q' \times 2^{X \times Q'}, \sigma' \in \Sigma \cup \{\theta\} : c(r, \sigma') := \begin{cases} \infty & \text{if } \sigma' \in \Sigma_u \cup \{\theta\} \\ \frac{p_0}{|e|+1} & \text{otherwise} \end{cases} \tag{6}$$

Also, the penalty of control function in Equation 5 can be replaced by the following penalty of control function to compute the infimal controllable and normal superlanguage of K:

$$\forall r = (r', R) \in X \times Q' \times 2^{X \times Q'} : p(r) := \begin{cases} p_0 & \text{if } r' \in X \times \{d\} \\ -\infty & \text{otherwise} \end{cases} \tag{7}$$

Theorem 7 If the supremal controllable and normal sublanguage of K is nonempty, then it equals the controlled plant behavior of G_5' under the control of a solution of OSCP1 with respect to G_5', with cost of control function as in Equation 6, and penalty of control function as in Equation 5. Furthermore, if instead penalty of control function as in Equation 7 is used, then the controlled plant behavior equals the infimal controllable and normal superlanguage of K.

Proof: Similar to that of Theorem 4. ■

Finally we show that the computation of the *infimal observable* superlanguage of K can be posed as an instance of OSCP2. Refer to [16] for a detailed discussion of observable languages and their properties. It is shown in [16] that the infimal observable superlanguage of a prefix closed language exists, and a closed form expression for computing it is obtained in [25, 9]. Observability of a language K requires that whenever a pair of strings, belonging to K and having the same mask value, are extended by a common event, then either both the resulting strings belong to K, or both do not belong to K. This condition is needed so that the supervisor can take the same control action after execution of a pair of strings that have the same mask value. If K does not satisfy this property, then infimal observable superlanguage of K is computed, which satisfies such a property.

The notion of pairs of strings in K with the same mask value is captured by the matching pair of states having the second coordinate of the G_1 part unequal to d. Let $r_1, r_2 \in X \times Q' \times 2^{X \times Q'}$ be a matching pair of states so that the second coordinate of the G_1 part of both the states is unequal to d. Since r_1 and r_2 is a matching pair of states, Lemma 7 implies that corresponding to each string that leads to the state r_1, there exists a string having the

same mask value as of the former string, such that it leads to the state r_2; and vice versa. Since the supervisor must take the same control action after the execution of a pair of strings that have the same mask value, an event is enabled at state r_1 if and only if it is enabled at r_2. This constraint is similar to the constraint C1, and can be captured by defining a mask function Ψ on the state space of G_5 as follows:

$$\forall r_1 = (r_1', R_1), r_2 = (r_2', R_2) \in X \times Q' \times 2^{X \times Q'} : \Psi(r_1) = \Psi(r_2) \Leftrightarrow R_1 = R_2 \tag{8}$$

Thus two states in state space of G_5 have the same mask value if and only if they constitute a matching pair. Hence, according to constraint C1 of OSCP2, it is ensured that the same control action is taken at any matching pair of states. Next the cost of control function is defined as:

$$\forall r \in X \times Q' \times 2^{X \times Q'}, \sigma \in \Sigma : c(r, \sigma) := \frac{p_0}{|e| + 1} \tag{9}$$

where $|e|$ denotes the number of transitions in G_5.

Theorem 8 The infimal observable superlanguage of K equals the controlled plant behavior of G_5 under the control of a solution of OSCP2 with respect to G_5, with cost of control function as in Equation 9, penalty of control function as in Equation 7, and state mask function Ψ as in Equation 8. Furthermore, if the cost of control function is modified so that $c(\cdot, \sigma) = \infty$ whenever $\sigma \in \Sigma_u$, then the controlled plant behavior equals the infimal controllable and observable superlanguage of K.

Proof: Similar to that of Theorem 4. ∎

Example 10 Consider the state machine G_5 of Example 7. Then in order to compute the infimal observable superlanguage of the language $K = \overline{(ab)^*}$ with respect to the plant G of Example 2 and mask $M(a) = M(b) = \lambda$, we define the mask Ψ on the state space G_5 such that $\Psi(3'') = \Psi(5'')$ and $\Psi(4'') = \Psi(6'')$. Next the penalty of control function is defined to be negative infinity for the states $1'', 2'', 3''$, and $4''$. The penalty of control for the states $5''$ and $6''$ is defined to be p_0.

Remark 3 An advantage of using the above techniques for computing controllable and normal/observable sublanguages/superlanguages is that they do not require alternate computations of controllable and normal/observable sublanguages/superlanguages as is done in [4]. Note that if M is a *projection* type mask, then the formula in [1] can be used to compute the supremal controllable and normal sublanguage without having to perform alternate computations of supremal controllable and supremal normal sublanguages. However, if the mask is non-projection type, then no such formula is known, and techniques developed above can be used.

In case M is a non-projection type mask, then the fact that a computation of supremal controllable sublanguage followed by a computation of supremal normal sublanguage does not necessarily yield the supremal controllable and normal sublanguage can be illustrated as follows. Suppose $\Sigma = \{a, b, c, u\}, \Sigma_u = \{u\}, M(b) = M(c) = M(u) \neq \epsilon$, $K = \{\epsilon, a, au\}$,

and $L(G) = \{\epsilon, a, au, ab, ac, acu\}$. Clearly K is controllable, i.e., $K^{\uparrow} = K$; and K is not normal, as $au \in K, ab \in L(G) - K$ and $M(au) = M(ab)$, i.e., $K^{\circ} \neq K$. It is easily seen that $K^{\circ} = \{\epsilon, a\}$. Then K° is not controllable, as $a \in K^{\circ}, u \in \Sigma_u$ and $au \in L(P) - K^{\circ}$. Thus $(K^{\uparrow})^{\circ} = K^{\circ}$ does not equal the supremal controllable and normal sublanguage of K. On the other hand, if we let $\hat{K} = \{\epsilon, a, au, ab, ac\}$, then $\hat{K}$ is normal, i.e., $\hat{K}^{\circ} = \hat{K}$; and $\hat{K}$ is not controllable, i.e., $\hat{K}^{\uparrow} \neq \hat{K}$, as $ac \in \hat{K}, u \in \Sigma_u$ and $acu \in L(G) - \hat{K}$. It is easily seen that $\hat{K}^{\uparrow} = \{\epsilon, a, au, ab\}$. Then $\hat{K}^{\uparrow}$ is not normal, as $ab \in \hat{K}^{\uparrow}, ac \in L(G) - \hat{K}^{\uparrow}$ and $M(ab) = M(ac)$. Thus $(\hat{K}^{\circ})^{\uparrow} = \hat{K}^{\uparrow}$ does not equal the supremal controllable and normal sublanguage of $\hat{K}$. Also, note that the formula for computing the supremal controllable and normal sublanguage given in [1, Theorem 4] is only applicable in a setting where, whenever a controllable and an uncontrollable event have "non-epsilon" mask values, then their mask values are different; so that the set of "masked uncontrollable" events is unambiguously identified. Since $u \in \Sigma_u$ and $b, c \in \Sigma - \Sigma_u$ are such that $M(b) = M(c) = M(u) \neq \epsilon$, the formula of [1, Theorem 4] is not applicable here.

5 Conclusion

We have introduced the problem of optimal supervisory control for DEDS by introducing the notions of cost and penalty of using a controller. Cost of control is incurred when an event is disabled by a controller, and penalty of control is incurred whenever undesired states remain reachable, or desired states remain unreachable in the controlled plant. The control objective is to optimize the net cost of control. This is formulated as OSCP1 for the case of complete state observation, and OSCP2 for the case of incomplete state observation. We show that a solution to OSCP1 can be obtained as a min-cut of an associated flow network, and a solution for OSCP2 is obtained by reducing an instance of OSCP2 to an instance of OSCP1. We show that supervisory control problems under complete as well as partial observations can be reduced to instances of OSCP1. In particular, we provide techniques for the computation of supremal controllable and normal sublanguage, infimal controllable and normal/observable superlanguage without having to perform alternate computations of controllable and normal/observable languages until a fixed point is reached. Thus above theory serves as a unified computational framework for supervisory control problems.

We did not comment on the computational complexity of any of the algorithms derived in this paper. However, since (i) all the algorithms, developed in this paper, are instances of OSCP1, and (ii) OSCP1 is solved using the max-flow min-cut computation; the computational complexity of any of the algorithms presented in this paper can be obtained from that of the max-flow min-cut computation, which is $O(|v| \cdot |e| \log(|v|^2/|e|))$, where $|v|$ denotes the number of vertices, and $|e|$ denotes the number of edges in the underlying flow network. Note that we are not suggesting that the computation of a supervisor under partial observation can be performed in a polynomial time; as in case of partial observation, OSCP1 is solved with respect to a state machine having its state space as the *power set* of the state space of the plant composed with the generator of the desired behavior.

References

[1] R. D. Brandt, V. K. Garg, R. Kumar, F. Lin, S. I. Marcus, and W. M. Wonham. Formulas for calculating supremal and normal sublanguages. *Systems and Control Letters*, 15(8):111–117, 1990.

[2] Y. Brave and M. Heymann. On optimal attraction in discrete event processes. Technical Report CIS 9010, Technion - Israel Institute of Technology, Hafia, Israel 32000, 1990.

[3] Y. Brave and M. Heymann. On stabilization of discrete event processes. *International Journal of Control*, 51(5):1101–1117, 1990.

[4] H. Cho and S. I. Marcus. On supremal languages of class of sublanguages that arise in supervisor synthesis problems with partial observations. *Mathematics of Control Signals and Systems*, 2:47–69, 1989.

[5] R. Cieslak, C. Desclaux, A. Fawaz, and P. Varaiya. Supervisory control of discrete event processes with partial observation. *IEEE Transactions on Automatic Control*, 33(3):249–260, 1988.

[6] V. K. Garg and R. Kumar. State-variable approach for controlling discrete event systems with infinite states. In *Proceedings of 1992 American Control Conference*, pages 2809–2813, Chicago, IL, July 1992.

[7] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, Reading, MA, 1979.

[8] R. Kumar. *Supervisory Synthesis Techniques for Discrete Event Dynamical Systems: Transition Model Based Approach.* PhD thesis, Department of Electrical and Computer Engineering, University of Texas at Austin, 1991.

[9] R. Kumar. Formulas for observability of discrete event dynamical systems. In *Proceedings of 1993 Conference on Information Sciences and Systems*, pages 581–586, Johns Hopkins University, Baltimore, MD, March 1993.

[10] R. Kumar and V. K. Garg. Optimal control of discrete event dynamical systems using network flow techniques. In *Proceedings of 1991 Annual Allerton Conference*, pages 705–714, Urbana, IL, October 1991.

[11] R. Kumar, V. K. Garg, and S. I. Marcus. On controllability and normality of discrete event dynamical systems. *Systems and Control Letters*, 17(3):157–168, 1991.

[12] R. Kumar, V. K. Garg, and S. I. Marcus. Language stability and stabilizability of discrete event dynamical systems. *SIAM Journal of Control and Optimization*, 31(5):1294–1320, September 1993.

[13] R. Kumar, V. K. Garg, and S. I. Marcus. Predicates and predicate transformers for supervisory control of discrete event systems. *IEEE Transactions on Automatic Control*, 38(2):232–247, February 1993.

[14] S. Lafortune and E. Chen. On the infimal closed and controllable superlanguage of a given language. *IEEE Transactions on Automatic Control*, 35(4):398–404, 1990.

[15] E. Lawler. *Combinatorial Optimization - Networks and Matroids*. Holt Rinehart and Winston, 1976.

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

[17] C. M. Ozveren and A. S. Willsky. Observability of discrete event dynamical systems. *IEEE Transactions on Automatic Control*, 35(7):797–806, 1990.

[18] C. M. Ozveren, A. S. Willsky, and P. J. Antsaklis. Stability and stabilizability of discrete event dynamical systems. *Journal of ACM*, 38(3):730–752, July 1991.

[19] K. M. Passino and P. J. Antsaklis. Near-optimal control of discrete event systems. In *Proceedings of 1989 Allerton Conference*, pages 915–924, Allerton, IL, September 1989.

[20] K. M. Passino and P. J. Antsaklis. On the optimal control of discrete event systems. In *Proceedings of 1989 IEEE Conference on Decision and Control*, pages 2713–2718, Tampa, FL, December 1989.

[21] P. J. Ramadge and W. M. Wonham. Modular feedback logic for discrete event systems. *SIAM Journal of Control and Optimization*, 25(5):1202–1218, 1987.

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

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

[24] P. J. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of IEEE: Special Issue on Discrete Event Systems*, 77:81–98, 1989.

[25] K. Rudie and W. M. Wonham. The infimal prefix closed and observable superlanguage of a given language. *Systems and Control Letters*, 15(5):361–371, 1990.

[26] R. Sengupta and S. Lafortune. A graph-theortic optimal control problem for terminating discrete event processes. *Discrete Event Dynamic Systems: Theory and Applications*, 2(2):139–172, 1992.

[27] J. N. Tsitsiklis. On the control of discrete event dynamical systems. *Mathematics of Control Signals and Systems*, 2(2):95–107, 1989.