Optimal Sensor and Actuator Choices for Discrete Event Systems *

Stanley D. Young †
Vijay K. Garg
Department of Electrical and Computer Engineering
University of Texas at Austin
Austin, Texas 78712
stanley@pine.ece.utexas.edu

January 6, 1994

Abstract

We present algorithms to optimally choose sensors and actuators to control a given discrete event system so that the closed loop behavior satisfies a specified constraint.

The main results of this paper are algorithms which demonstrate: the polynomial solution to the choice of actuators, the polynomial solution to the choice of sensors for certain supervisory control problems, and the solution to the general choice of sensors [1].

Keywords: Discrete Event Systems, Observability, Controllability

1 Introduction

This work addresses the question of how to choose sensors and actuators to control a discrete event system to attain a specified behavior. Supervision

*Supported in part by NSF CCR-911065 and TRW Faculty Assistantship Award.
†Supported in part by a DuPont Fellowship.
[1] A preliminary version of this paper appeared as [18].

of such systems is achieved by observing events and disabling those which are not desired or lead to undesirable behavior. A supervisor uses sensors and actuators to observe and control the system behavior.

The motivation for an optimal choice of sensors and actuators is that many systems do not require that all events be controllable or observable to attain a specified behavior. For these systems, a technique which provides a method for choosing which events to control and observe is needed. This work addresses the design question of what components a supervisor requires to be able to control a plant to attain a specified behavior.

This work is related to other work in the optimal control of discrete event systems. In [11], optimality is defined in terms of set inclusion of the closed loop behavior of the plant and supervisor relative to the desired behavior. We require that the closed loop behavior equal the desired behavior and define optimality in terms of sensor and actuator cost. In [3], optimality is defined in terms of a cost for the sequence of events which appear in the closed loop system. In [13, 8], optimality is defined in terms of a cost for the sequence of events which appear in the closed loop system and a cost for controlling events. We assume that costs may be allocated as an initial expense and that there is no cost during system operation. We also include the costs incurred for observing events in addition to controlling them.

This previous work in the design of supervisors for discrete event systems has been of the form: given a plant, desired behavior, and a set of actuators and sensors, what is the best closed loop behavior obtainable? This work is of the form: given a plant and a desired behavior, what is the least cost set of sensors and actuators which allow the closed loop system to obtain the desired behavior?

There are two aspects to this problem: choosing the events required for control and choosing the events to observe. The specific choice of events to control and observe specify the configuration of the supervisor. In order to determine an optimal configuration, a cost function is introduced for each event controlled or observed.

For the problem considered in this paper, which requires attaining the specified behavior, the choice of which events to control is straightforward. The algorithm is based on the idea that any event which causes a behavior outside of the specified behavior must be controlled. The choice of which events to observe is not as straightforward. The algorithm proposed for the general case is exponential in the cardinality of a subset of the events

which can occur in the desired behavior and polynomial in the state space of the plant and desired behavior. There are classes of this problem which do admit a polynomial solution. Two cases in particular are considered: 1) the plant and desired behavior are represented by completely specified state machines, and 2) a "minimal" set of observed events is chosen.

In this work, the state machine formalism is used to describe the plant and constraint behaviors. The previous work relating state machines to the control concepts needed for this work is reviewed in Section 2. Section 3.1 introduces the problem of sensor and actuator choice and some notation. Section 3.2 describes a technique for choosing the events required for control. Section 3.3 describes a technique for choosing a potential set of events for observation. Section 3.4 describes the special case of the problem when the plant and desired behavior are represented by completely specified state machines. Section 3.5 describes the special case of the problem where the observed event set is "minimal". This approach leads to a "greedy" style of algorithm. Section 4 provides some examples which demonstrate some of the concepts discussed in the paper. Section 5 summarizes the results and indicates direction for future work.

2 Machines, Languages, Observability and Controllability

Languages and machines provide a means for representing the behavior of discrete event systems. We use this modeling technique to describe systems and their specification.

2.1 Machines and Languages

A state machine is represented by a four tuple. Specifically, if M is a state machine, then one writes $M = (Q, A, \delta, q_0)$ where

$$\begin{array}{lcl} Q & = & \text{a set of states,} \\ A & = & \text{a finite set of transition labels or events,} \\ \delta & = & \text{the transition function, } \delta : Q \times A \rightarrow Q, \text{ and} \\ q_0 & = & \text{the initial state, } q_0 \in Q. \end{array}$$

The transition function δ is in general a partial function; consequently, $\delta(q,\sigma)!$ denotes that the transition event σ is defined from state q. This definition of the transition function, $\delta : Q \times A \rightarrow Q$, specifies a deterministic system.

A state machine, $P = (Q, A, \delta, q_0)$, can also be represented as a directed graph $M = (Q, T)$ where Q and T are the sets of nodes and arcs, respectively, or states and transitions in this instance. Q is the set of states in the machine and $T \subseteq Q \times A \times Q$ is the set of transitions. If $q_1, q_2 \in Q$ and $\delta(q_1, \sigma) = q_2$, then one denotes the transition by the three tuple (q_1, σ, q_2). With a machine represented by a directed graph, all the standard directed graph theoretic concepts and results can be used [1, 5].

A formal language is a set of strings which are sequences of symbols. The symbols are chosen from a finite set, or alphabet, A. A^* is used to denote the set of all finite sequences of symbols from the alphabet A. If $u \in A^*$, then $pr(u)$ denotes the set of all strings which are prefixes of u, i.e. for $u \in A^*$

$$pr(u) = \{s \in A^* | s = u(1) \ldots u(k), 0 < k \leq |u|\}.$$

The notation $s \leq u$ is used to denote that s is a prefix of u. Note that the empty string, ε, is the prefix of all strings. The ***prefix closure*** of a language L is defined by

$$\overline{L} = \{w \in pr(u) | u \in L\}.$$

For this work, we restrict our attention to the class of regular languages. A basic result relates regular languages and finite state machines: a language $L \subseteq A^*$ is regular if and only if it is generated by a finite state machine [6]. A language B is the language ***generated*** by machine M when

$$w \in B \Leftrightarrow \delta(q_0, w)!.$$

$L(M)$ denotes the language generated by machine M. It can be shown that regular languages are closed under the set operations of finite union and intersection [6].

In the following development, the concept of combining two finite state machines to create a single machine, the product machine, is useful. If $M_1 = (Q_1, A, \delta_1, q_{1,0})$ and $M_2 = (Q_2, A, \delta_2, q_{2,0})$ are two machines with the same event set, A, then the product of the two machines is denoted

$$M_1 \| M_2 = (Z, A, \delta_{\|}, z_0)$$

where,

$$Z = Q_1 \times Q_2 \text{ and } z_0 = (q_{1,0}, q_{2,0}) \in Q_1 \times Q_2,$$

$$\delta_{||}((q_1, q_2), \sigma)! \Leftrightarrow \delta_1(q_1, \sigma)! \wedge \delta_2(q_2, \sigma)!,$$

$$\delta_{||}((q_1, q_2), \sigma) = \begin{cases} (\delta_1(q_1, \sigma), \delta_2(q_2, \sigma)) & \text{if defined} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The language generated by the product machine has a specific relation to the languages of the machines from which it is composed. If $M_{||} = M_1 \| M_2$ then

$$L(M_{||}) = L(M_1) \cap L(M_2).$$

These relations follow directly from the definition of the transition function of the product machine.

A more complete description of machines and languages can be found in [6].

2.2 Controllability and Observability for Languages

The event set, A, can often be partitioned into two sets: A_c and A_{uc} representing controllable and uncontrollable events, respectively. With the concept of controllable and uncontrollable events, one defines a supervisor for a plant which affects the controllable events which a plant executes based on some specified constraints. A supervisor for a plant P where $L = L(P)$ is a map

$$f : L \to 2^A$$

which specifies a set of enabled inputs. These enabled inputs are those which the supervisor allows based on the string of previously generated events in L. Note that it is equivalent to specify which events are to be disabled as function of the string of previously generated events. This specification of events to be enabled and disabled is called a ***control pattern.*** Given an ordering of the events, this control pattern can be thought of as a bit string with a 1 in the place of an event which is enabled by the supervisor and a 0 in the place of an event which is disabled. The closed loop system consisting of a supervisor, f, and plant, P, has the finite closed loop behavior denoted by L_f, and defined as follows:

1. $\varepsilon \in L_f$,

2. $w\sigma \in L_f$ if and only if $w \in L_f, \sigma \in f(w),$ and $w\sigma \in \overline{L}$.

Note that the definition of L_f implies that $L_f \subseteq \overline{L}$ and that $\overline{L_f} = L_f$ [12].

A language K is said to be ***controllable*** with respect to a language L if the following condition holds:

$$\forall(s \in K) \wedge (\sigma \in A_{uc}) : \\ ((s \in K) \wedge (s\sigma \in L)) \Rightarrow (s\sigma \in K).$$

I.e. the desired behavior K is controllable with respect to the plant behavior L if any string s in K, which has a legal suffix σ such that $s\sigma$ is in L and σ is uncontrollable, is such that the string with the suffix, $s\sigma$, is an element of K. A procedure given in [2] gives a polynomial algorithm for determining the controllability of a language with respect to another language.

The event set, A, can also often be partitioned into another two sets: A_o and A_{uo} representing observable and unobservable events, respectively. In general, an observer detects events occurring in the system through a mask; this mask affects how an observer interprets events. Let $M : A \rightarrow A_o$ denote a mask [2]. This mask can be considered as a projection from the event set of the system to the observed event set.

In this work, an observed event σ is not affected by the mask M, $M(\sigma) = \sigma$; whereas, an unobserved event is projected onto the empty string by the mask, $M(\sigma) = \varepsilon$. This observation mask can be considered to be the natural projection of the event space onto the observation event space.

A language K is said to be ***observable*** with respect to a language L and mask M if the following condition holds:

$$\forall(s, s' \in K) \wedge (\sigma \in A) : \\ ((M(s) = M(s')) \wedge (s\sigma \in K) \wedge (s'\sigma \in L)) \Rightarrow (s'\sigma \in K).$$

I.e. the desired behavior K is observable with respect to the plant behavior L and mask M if any string s in K, which has a string s' in K to which it is equivalent under the mask and both these strings have a suffix σ in the plant language L and the first has the suffix in K also, is such that the second string, with the suffix, $s'\sigma$, is an element of K. A procedure given

[2]Note that in the general case a mask might have memory, i.e. $M : A^* \times A \rightarrow A_o$ in a manner similar to the block codes of symbolic dynamics [15].

in [14] gives a polynomial algorithm for determining the observability of a language with respect to another language and a mask.

The main result which relates observability, controllability, and supervisors is contained in Theorem 2.1, which is a restatement of previous results [4, 9].

Theorem 2.1 Let $K \subseteq L$ be closed and nonempty, and M a mask. Then there exists a supervisor f such that $L_f = K$ if and only if K is controllable with respect to L and observable with respect to L and mask M.

A more complete review of the observability and controllability of discrete event systems modeled with machines can be found in [12].

3 Choosing Sensors and Actuators

3.1 Notation, Assumptions, and an Initial Solution

We assume that a plant and the constraint behavior are given by the languages L and K, respectively. As described in Section 2.1, with each language L and K, we associate a machine, M_L and M_K, specified by

$$M_L = (Q_L, A, \delta_L, q_{L,0}) \text{ and } M_K = (Q_K, A, \delta_K, q_{K,0}).$$

The machine which results from the product of M_L and M_K is used extensively. We denote this machine by $M_{||} = (Q_{||}, A, \delta_{||}, q_{||,0})$.

We assume that the primary cost associated with a sensor or an actuator is the initial capital investment and that the cost of operation is negligible compared to the initial cost. Hence, the cost functions used for evaluating optimality concern only the event to be observed, or controlled and not the number of times an event is observed or controlled [3]. The actuator and sensor cost functions are denoted by

$$c_c : A \to \Re^+ \text{ and } c_o : A \to \Re^+.$$

In order to specify that a specific event has been chosen for observation or control, "choice" functions are used. The "choice" functions used in this work are denoted by

$$i_c : A \to \{0, 1\} \text{ and } i_o : A \to \{0, 1\}$$

[3] Note that in the general case a state dependence for the costs may be required.

for the control and observation "choice" functions respectively. For these functions, $i_c(\sigma) = 0$ ($i_o(\sigma) = 0$) means that event σ has not been chosen for control (observation), and $i_c(\sigma) = 1$ ($i_o(\sigma) = 1$) means that event σ has been chosen for control (observation).
Now the optimization problem can be specified.

Problem 3.1 Given a plant behavior specified by the language L, a constraint behavior specified by the language K, with $K \subseteq L$, and cost functions c_c and c_o, choose i_c and i_o to minimize the cost C such that $L_f = K$, where

$$C = \sum_{\sigma \in A} (c_c(\sigma) \cdot i_c(\sigma) + c_o(\sigma) \cdot i_o(\sigma)).$$

In general, functions i_o and i_c can always be found such that $L_f = K$. The solution in which everything is chosen to be controlled and observed i.e. $\forall \sigma \in A : i_c(\sigma) = i_o(\sigma) = 1$, provides that $L_f = K$. However, this solution is most likely not the minimum cost solution.

An obvious choice to determine the optimal i_c and i_o functions is to determine the cost for each possibility and pick one which has the minimal cost and satisfies $L_f = K$. It is clear that the procedure of trying each possible function for i_o and i_c can be used to find the minimum. By [14], this procedure has a complexity of $O(2^{|A|} \cdot 2^{|A|} \cdot |\tilde{Q}|^3 \cdot |A|)$, where $\tilde{Q} = \max(Q_L, Q_K)$.

3.2 A Calculation for the Control "Choice" Function

The problem with the approach given above is the many exponentials which appear in the complexity measure. We now consider ways to reduce some of this complexity.

First, we consider ways to calculate i_c, the control "choice" function. In order to calculate the required function, consider the product $M_{||}$ of M_L and M_K. Each state $q_{||} \in Q_{||}$ in this product machine has corresponding states $q_L \in Q_L$ and $q_K \in Q_K$.

Definition 3.1 $E_c : Q_{||} \rightarrow 2^A$ is the function which gives the set of events $\sigma \in A$ which are defined in M_L at state q_L and not defined in M_K at q_K, i.e.

$$E_c(q) = \{\sigma \in A : \delta_L(q_L, \sigma)! \wedge \neg \delta_K(q_K, \sigma)!\},$$

where $q = (q_L, q_K)$.

Next we extend this definition to the entire state set.

Definition 3.2

$$\hat{E}_c = \bigcup_{\{q \in R_{\|}(q_{\|,0})\}} E_c(q),$$

where $R_{\|}(q_{\|,0}) = \{q \in Q_{\|} : (\exists w \in A^* : \delta_{\|}(q_{\|,0}, w) = q)\}$. In other words, the controllable events are those which must be disabled at states which are reachable from the initial state.

Proposition 3.1 $\hat{E}_c$ is the smallest set of controllable events such that $L_f = K$.

Proof:
By the definition of the product machine, any string which reaches a state in the product machine from the initial state is a string in both of the languages of the machines which were used to form the product. Consequently, for $L_f = K$ to hold, we require that all states reachable from the initial state in the product machine be reachable in the closed loop behavior. Hence, if $\sigma \in \hat{E}_c$, then $i_c(\sigma) = 1$, and if $\sigma \notin \hat{E}_c$, then $i_c(\sigma) = 0$. If i_c were defined some other way, there would be strings which either could not be generated but are in K or strings not in K which could be generated.

□

We can drop the distinction between $\hat{E}_c$ and A_c and let A_c denote this minimal $\hat{E}_c$.

Since calculating the product machine has complexity $O(|Q_L| \cdot |Q_K| \cdot |A|)$, we can replace one of the exponential factors in the complexity of the procedure specified in Section 3.1 by $|\tilde{Q}|^2 \cdot |A|$. This reduction results from replacing the tests for i_c with the calculation of the product machine and construction of $E_c(q)$ for each q. Also, note that this calculation must be done only once. Hence, it is an additive term and is not counted as part of the observation function calculation. The resulting complexity with this modification is $O(2^{|A|} \cdot |\tilde{Q}|^3 \cdot |A|)$.

An illustration of the controlled event set calculation is given in Example 3.1.

Example 3.1 Consider the plant and constraint behaviors as given in Figure 1. For these machines, the part of the product machine which is reach-

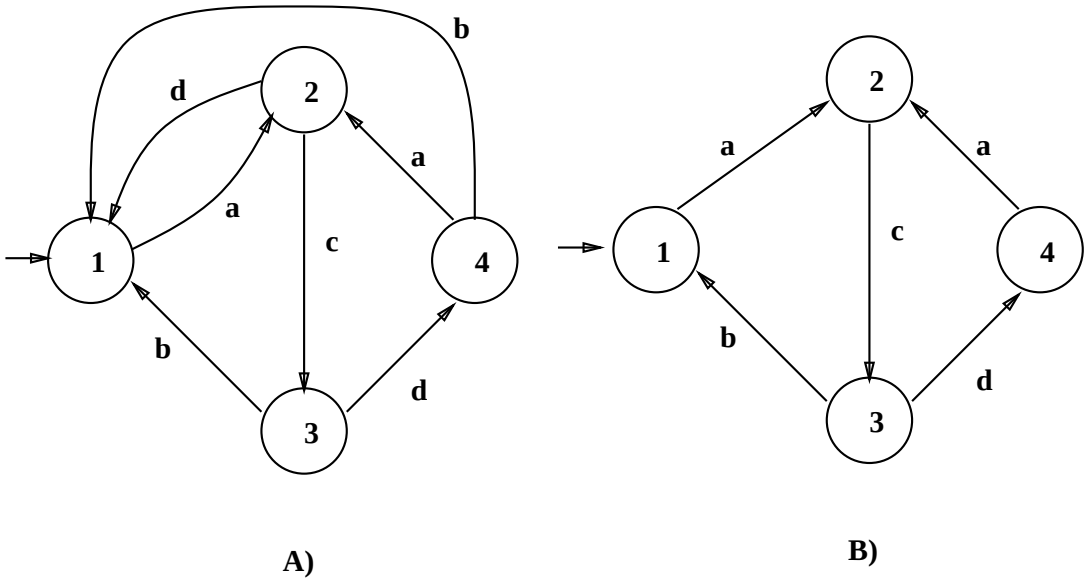

Figure 1: A) Plant and B) Specified Behavior.

able from the initial state is isomorphic to the constraint behavior. By comparing the states in these machines, it is clear that only at states 4 and 2 in the constraint machine are there any events which are defined in the plant and not in the constraint behavior. Hence, we have that $E_c(4) = \{b\}$, $E_c(2) = \{d\}$, and for all other states $E_c(q) = \emptyset$. For this system, the events which must be controllable are b and d, i.e. $E_c = \{b, d\}$.

The final result in Proposition 3.2 shows that the method prescribed has optimal computational complexity.

Proposition 3.2 The complexity of calculating the events which must be controlled to obtain the specified behavior is $\Theta(|\tilde{Q}|^2 \cdot |A|)$.

The proof of this proposition is obvious by considering that calculating the controllability of a language with respect to another one is a special case of the problem considered here. This controllability calculation is known to be $\Theta(|\tilde{Q}|^2 \cdot |A|)$ [14].

3.3 Calculation of the Observation "Choice" Function

Now we consider ways to calculate i_o, the observation "choice" function.

To obtain the correct closed loop behavior, $L_f = K$, a supervisor needs to identify which control pattern to use for the current state. Thus, any event σ labeling a transition between states [4] q_1 and q_2, which require sufficiently different control patterns, needs to be observed. Now we must investigate what conditions constitute a sufficient difference.

Let γ_q be the control pattern for state q and $E_c(q)$ be the events which must be disabled at state q. Let the function $E_d : Q \rightarrow 2^A$ give the transitions which are defined from state q, i.e.

$$E_d(q) = \{\sigma \in A : \delta_{||}(q, \sigma)!\}.$$

There are certain types of transitions which require a change in control patterns and hence must be observed. Consider a transition from state q_1 to q_2 by event σ, i.e. $\delta_{||}(q_1, \sigma) = q_2$. The transition for which $E_c(q_1) \cap E_d(q_2) \neq \emptyset$ must be identified. In this case, an event which must be disabled at q_1 is defined for q_2. Also, the transition for which $E_d(q_1) \cap E_c(q_2) \neq \emptyset$ must also be identified. In this case, an event which is defined at q_1 must be disabled at q_2.

Definition 3.3 A *Type O* transition is defined by

$$(E_c(q_1) \cap E_d(q_2) \neq \emptyset) \vee (E_d(q_1) \cap E_c(q_2) \neq \emptyset).$$

Other transitions might or might not require observation, depending on what control patterns the supervisor chooses for the states associated with the transition, as shown in Example 3.2.

Example 3.2 If none of the sets describing the controlled events and the defined events intersect, then the supervisor can assign the control pattern for each of the states to be the same. I.e. if $E_c(q_1) \cap E_c(q_2) = \emptyset$, $E_c(q_1) \cap E_d(q_2) = \emptyset$, $E_d(q_1) \cap E_c(q_2) = \emptyset$, and $E_d(q_1) \cap E_d(q_2) = \emptyset$, then the supervisor can set $\gamma_{q_1} = \gamma_{q_2} = E_c(q_1) \cup E_c(q_2)$ and need not identify this transition for

[4] Notice that in the remaining discussion most of the references to states and machines concern the product machine.

local control purposes. A specific illustration of these circumstances can be seen in states $(1,1)$ and $(2,2)$ of the product of the machines given in Figure 1.

However, the supervisor might need to observe some *non-Type O* transitions to correctly identify some *Type O* transitions, as shown in Example 3.3.

Example 3.3 Figure 2 gives the plant and desired behavior in which the supervisor must consider possible future behavior to decide which events to observe. Specifically, event b must be observed to determine which branch of the desired behavior is occurring, even though no change in control pattern is required after the first transition.

Proposition 3.3 Given machines M_K and M_L, to obtain $L_f = K$, we have the following condition on the observed events:

if

$$\exists q_1, q_2 \in Q_{||} : (q_1 \in R_{||}(q_{||,0})),$$
$$\sigma \text{ is } \textit{Type O}, \text{ and}$$
$$\delta_{||}(q_1, \sigma) = q_2,$$

then

$$\sigma \in A_o,$$

where $R_{||}(q_{||,0})$ is as defined in Section 3.2.

The proof of this proposition is obvious and follows from the requirement that $L_f = K$ and that q_1 is reachable from the initial state.

Let the events which satisfy the conditions in Proposition 3.3 form a set denoted by A_{req} (for the required events), i.e.

$$A_{req} = \left\{ \sigma \in A : \left(\exists q_1, q_2 \in Q_{||} : \begin{array}{l} q_1 \in R_{||}(q_{||,0}) \\ \wedge \\ \sigma \text{ is } \textit{Type O} \\ \wedge \\ \delta_{||}(q_1, \sigma) = q_2 \end{array} \right) \right\}.$$

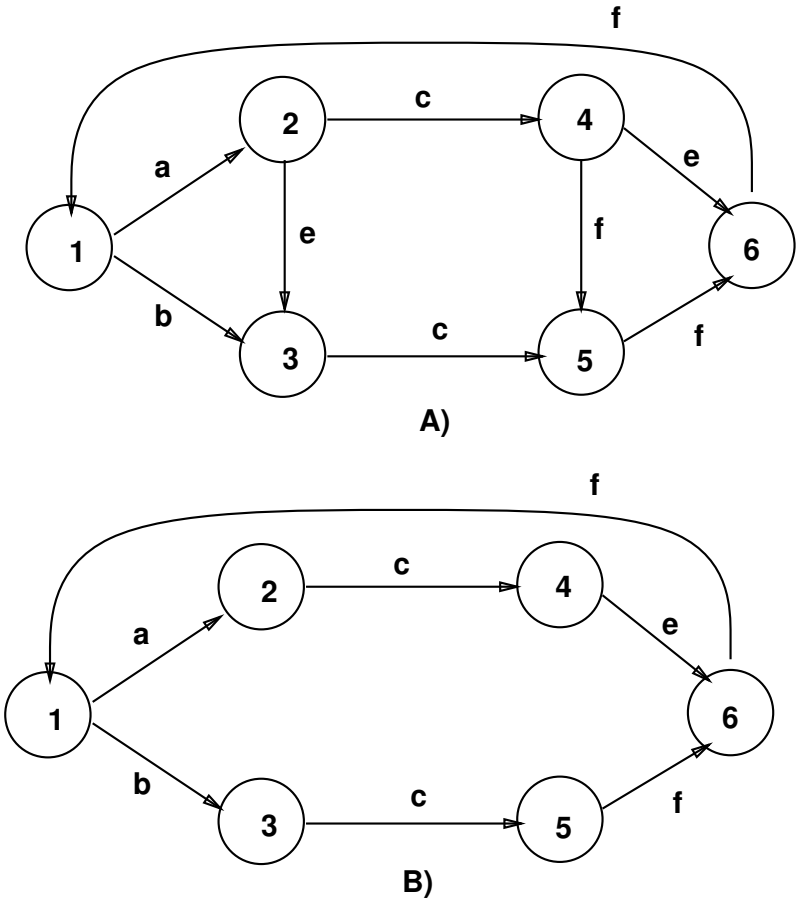

Figure 2: A) Plant and B) Specified Behavior.

The procedure for choosing sensors and actuators which results from the above considerations consists of the following steps: calculation of the events which must be controlled, $\hat{E}_c$, calculation of the events which must be observed, A_{req}, and then trying all choices for the observation "choice" function which are consistent with A_{req}. The reduction in complexity arises from the reduction in the average number of potential cases which must be considered for A_o. Since calculating $M_{||}$, A_c, and A_{req} have complexity $O(|\tilde{Q}|^2 \cdot |A|)$, and by [14] checking controllability and observability is $O(|\tilde{Q}|^3 \cdot |A|)$, we obtain a complexity of $O(2^{|A \setminus A_{req}|} \cdot |\tilde{Q}|^3 \cdot |A|)$ for this approach.

3.4 Fully Specified Behaviors

We consider some special cases as a means of further reducing the complexity of choosing the required sensors and actuators. The special case considered in this section is based on the polynomial complexity of finding the reduced machine for a fully specified Moore machine. The approach is to form the product of the plant and desired behavior and then to finish specifying the machine such that it is a fully specified Moore machine [7, 6], where the output at a state represents a bit pattern corresponding to the control pattern at the state. In general, most machines used to represent discrete event systems are not fully specified, i.e. there is usually not a transition specified for every event from every state. This characteristic arises from the fact that in the actual system it may not make any sense to define a given event from every state. In addition to an output control pattern for each state, a fully specified machine has a transition specified for every event from every state.

By [7], calculating the minimal equivalent state machine is polynomial in the number of states, i.e. $O(|\tilde{Q}|^2 \cdot |A|)$. Hence, if we can reduce the general problem of observed event set selection to the problem of determining the minimal equivalent machine, then we will have a polynomial technique for choosing the controlled and observed events. Recall that for an incompletely specified Moore machine determining the minimal equivalent machine has nondeterministic polynomial time complexity and is complete in this class [10]; consequently, we consider this approach only for completely specified machines. Note that in general this approach will not provide an optimal choice for the observed event set, due to possible suboptimal completions of the machine specification.

There are two aspects to completing the specification of the product machine: specifying the transitions and the output patterns.

The transitions may be specified in any manner consistent with the original product machine such that the resulting machine is deterministic.

From the incompletely specified product machine, one can calculate the events which must be controlled and those which can remain uncontrolled. This information is used to fully specify the output patterns. Recall that an uncontrollable event cannot be disabled by a supervisor. This fact provides a constraint on specifying the output patterns: any uncontrollable event must be enabled in every output pattern. There are two ways to specify the output pattern for the controlled events. Any controllable event which is not defined in the original product machine at a given state may be disabled in the output pattern for that state. Alternatively, the event may be enabled or disabled in an attempt to provide a lower cost solution. (This second alternative essentially requires the specifier to make the choices which place the incompletely specified problem in the nondeterministic polynomial complete complexity class.)

Observe that the first technique of choosing the output patterns has a benefit beyond providing a unique and deterministic technique for choosing the observed event set. The supervisor which results from this choice may be used with a wide collection of plants and still guarantee the correct closed loop behavior.

Example 3.4 Consider the original plant and specified behavior as given in Figure 3.

The product machine is isomorphic to the machine for the desired behavior. It is clear that $A_c = \{a, c\}$ and that $A_{uc} = \{b\}$. Let λ denote the output function for this machine. From the first alternative for selecting the output patterns, we obtain that $\lambda(1) = 110$, $\lambda(2) = 010$, and $\lambda(3) = 011$. This obviously is not optimal because to generate the correct output both a and b must be observed, but as shown in Section 4.2, it is sufficient to observe only a or b but not both. The benefit of this approach is apparent when one considers how many other plants, with the same state space and event set, can be controlled with the supervisor which results from this case. There are 4^4 possible plants which would give the same result. Hence, this approach provides a supervisor which can be applied in a wide variety of cases beyond the original plant.

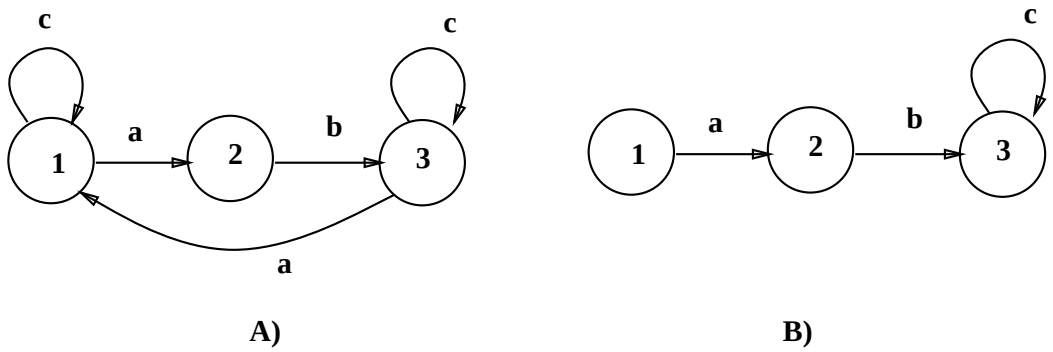

Figure 3: Plant and Specified Behavior for a Simple System.

The justification for considering events which label transitions in the minimal equivalent Moore machine is given in the Proposition 3.4.

Proposition 3.4 For a given completely specified Moore machine representation of the product of M_L and M_K, an event σ is in the observed event set, i.e. $\sigma \in A_o$ if and only if σ labels a transition between distinct states in the minimal equivalent Moore machine.

Proof: The concept of equivalence classes of states based on control patterns provides the essence of the proof that determining a reduced equivalent machine is sufficient for controlling a discrete event system. In a reduced machine, states remain differentiated only if an inconsistency arises from combining them. As applied to supervisor design, two states must remain differentiated if combining them would cause an incorrect control pattern to be generated at some other state. Consequently, any transition between distinct states in the reduced machine must be observed to apply the correct control pattern.

□

3.5 "Minimal" Choice of Observed Events

The special case considered in this section gives a minimal versus an optimal solution. The "minimal" choice of observed events is obtained by a greedy algorithm based on the observation cost function.

Definition 3.4 Let K and L denote the desired and plant behaviors, respectively. Let $P(B)$ denote the mask associated with the observed event set B. We say that B is *minimal* if there is no set X such that $X \subseteq B$ and $B \neq X$ with K observable with respect to L and $P(X)$.

In general, there will be several minimal sets of observed events.

This approach calculates the minimal set of observed events based on the order in which events are chosen to determine if an event is required for observability. Note that this choice is what provides the polynomial complexity of this approach.

The technique consists of choosing an event and determining if the desired behavior is observable with respect to the plant behavior and the proposed set of observed events. The proposed set of observed events is the current set of observed events minus the chosen event. If the behavior is observable, then remove this event from the set of observed events. If it is not, then pick another event to test. Continue until all events have been tested. The complexity of this procedure is $O(|A| \cdot |\tilde{Q}|^3)$. Note that this procedure cannot be used to guarantee an optimal solution for the general problem because the order in which events are considered does matter.

This approach can be implemented as a greedy type of algorithm by choosing the events for consideration in decreasing order of their cost. Example 3.5 demonstrates that this approach does not guarantee an optimal solution.

Example 3.5 Consider the plant and desired behaviors as specified in Figure 4 with the cost function defined as $c_o(a) = 6$, $c_o(b) = 5$, $c_o(c) = 4$, $c_o(d) = 3$, $c_o(e) = 2$, and $c_o(f) = 1$. For this cost assignment, a greedy algorithm provides an observed event set $A_o = \{c, d, e\}$ with an observation cost of 9 units. An optimal observed event set is $A_o = \{a, e\}$ with an observation cost of 8 units.

4 Applications

4.1 An Example Revisited

Figure 1 gives the machines for the plant and desired behaviors.

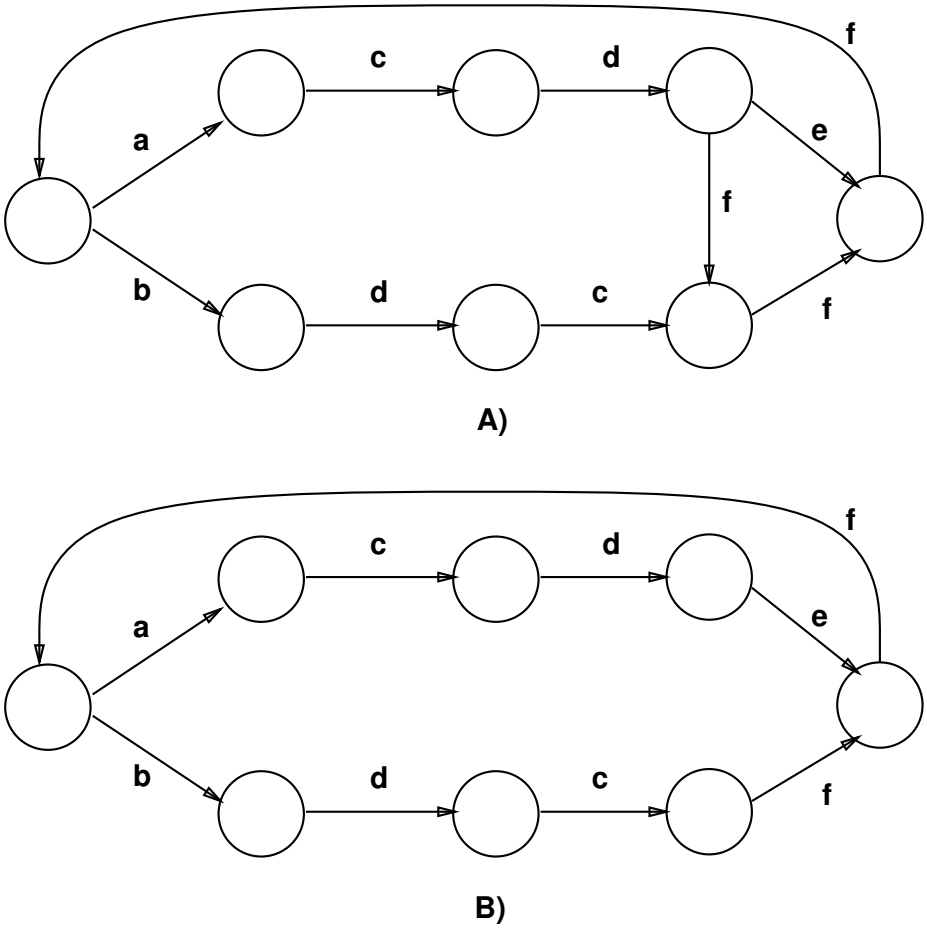

Figure 4: A) Plant and B) Specified Behavior.

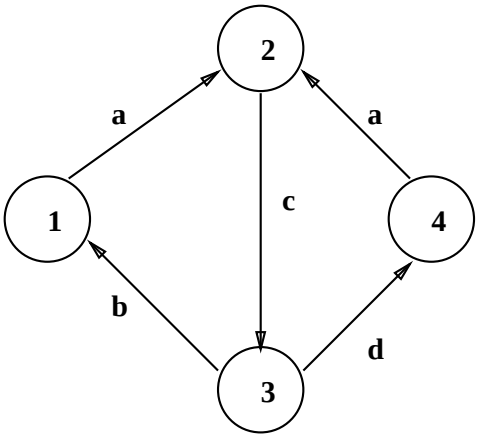

Figure 5: Product of Plant and Specified Behavior.

From the reduced product machine, $M_{||}$ in Figure 5, we observed in Example 3.1 that $E_c(4) = \{b\}$, $E_c(2) = \{d\}$, $E_c(1) = \emptyset$, and $E_c(3) = \emptyset$. We can also observe that the events which are defined at each state are given by $E_d((c,1)) = \{a\}$, $E_d((c,2)) = \{c\}$, $E_d((c,3)) = \{b,d\}$, and $E_d((c,4)) = \{a\}$.

From these observations, we can conclude which transitions are ***Type O***: $(2,c,3)$ is ***Type O*** and $(3,d,4)$ is ***Type O***. Hence, we obtain that $A_c = \{b,d\}$ and that $A_{req} = \{c,d\}$. These specifications provide that $E^{req} = \{\{c,d\},\{b,c,d\},\{a,c,d\},\{a,b,c,d\}\}$.

If we choose $A_o = \{c,d\}$, then the system is not observable. If we choose $A_o = \{b,c,d\}$, then the system is observable. If we choose $A_o = \{a,c,d\}$, then the system is observable. If we choose $A_o = \{a,b,c,d\}$, then the system is observable.

4.2 A Simple Example

Figure 6 gives a simple plant and desired behavior which demonstrate the choices required in choosing the observed event set.

For this example, $E_c(1) = \{c\}$, $E_c(2) = \emptyset$, and $E_c(3) = \{a\}$. We observe that $E_d((c,1)) = \{a\}$, $E_d((c,2)) = \{b\}$, and that $E_d((c,3)) = \{c\}$.

From these observations, we conclude that no transitions are ***Type O***. Hence, we obtain that $A_c = \{a,c\}$ and that $A_{req} = \emptyset$.

The system is observable if we choose $A_o = \{a\}$ or $A_o = \{b\}$, but not

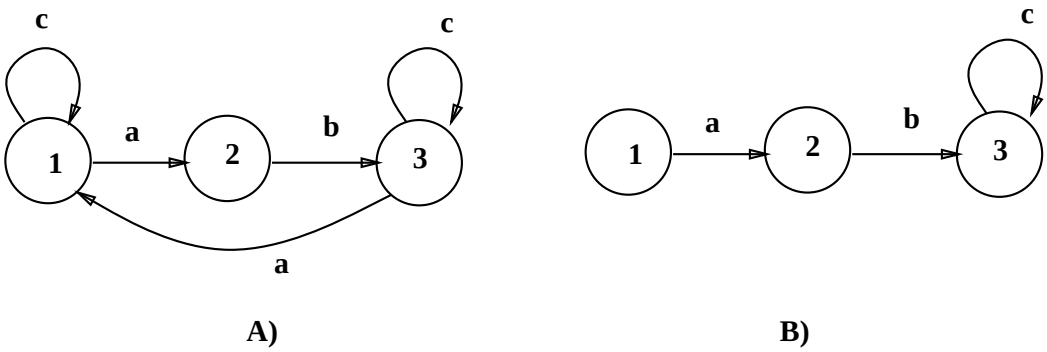

Figure 6: Plant and Specified Behavior for a Simple System.

observable if we choose $A_o = \{c\}$.

4.3 Desired Behavior Not a Subautomaton of Plant Behavior

Figure 7 gives a plant and desired behavior which demonstrate the choices required in choosing the observed event set when the desired behavior is not a subautomaton of the plant behavior.

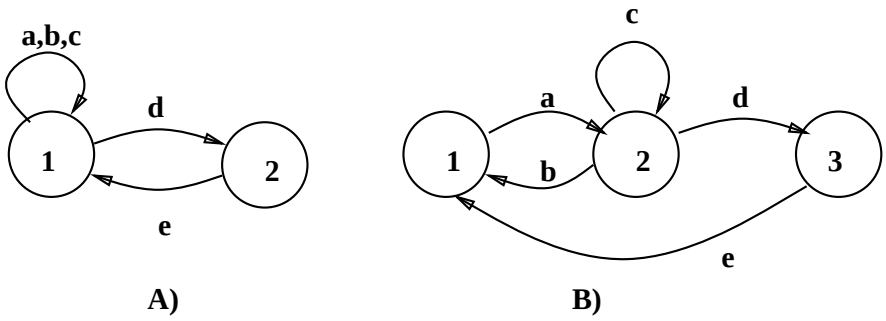

Figure 7: Plant and Specified Behavior for a Desired Behavior Not a Subautomaton of Plant Behavior.

For this example, $E_c(1,1) = \{b,c,d\}$, $E_c(1,2) = \{a\}$, and $E_c(2,3) = \emptyset$. We observe that $E_d((1,1)) = \{a\}$, $E_d((1,2)) = \{b,c,d\}$, and that $E_d((2,3)) = \{e\}$. (The states are specified as elements of $Q_L \times Q_K$.)

From these observations, we conclude which transitions are *Type O*: $((1,1),a,(1,2))$ is *Type O* and $((1,2),b,(1,1))$ is *Type O*. Hence, we obtain that $A_c = \{a,b,c,d\}$ and that $A_{req} = \{a,b\}$.

The system is observable if we choose $A_o = \{a,b,d\}$ or $A_o = \{a,b,e\}$.

5 Conclusions

This paper addresses the issue of how to choose sensors and actuators for a system which may be modeled using state machines. The goal is to restrict the system's behavior to a specified constraint behavior. In order to insure that the system adheres to the constraint behavior certain events in the system must be observed and controlled with sensors and actuators.

We have provided a technique for determining which events must be observed and controlled to achieve the specified constraint behavior. A cost function associated with the sensors and actuators is used to provide a metric for choosing between different controller configurations.

Using different types of equivalence classes, an algorithm is given for calculating the controller and observer parts of the supervisor which restricts the system's behavior to the specified constraint behavior. The algorithm for choosing the actuators has polynomial complexity in terms of the events and the states in the product machine. The approach presented in this work provides an algorithm for choosing sensors and actuators with exponential complexity in the number of events and polynomial complexity in the states. The special cases considered were a completely specified machine with its associated minimization algorithm and a minimal set of observed events with its greedy algorithm. Both special cases provide polynomial complexity in the number of events and states for choosing the sensors and actuators.

There are several areas for further work associated with this paper. One extension is to include a state dependence of the cost functions for the sensors and actuators. Observe that for the case in which we require that the closed loop behavior equal the specified constraint behavior the state dependence of the actuators is irrelevant. But the state dependence of the sensors can affect how the events to be observed are chosen.

Another extension is to allow the closed loop behavior to be a subset of the specified constraint behavior. In this case, a technique for measuring

how much of the specified behavior is not attained is needed. This measure would need to have an associated cost and be included in the minimization Problem 3.1. One approach assigns a cost to states and uses the reachability of the state in the closed loop behavior as a measure of how much of the specified constraint behavior is attained. This approach is similar to that specified in [8]. Another approach is to assign a measure to the strings in the specified constraint behavior which are not possible in the closed loop behavior and then to assign a cost to this measure and include it in the minimization problem. In either of these cases, the state dependence of the cost for sensors and actuators must be included in the general case analysis of the minimization problem.

Another extension considers the effects which faults in the plant have on controlling its behavior to achieve a specified constraint behavior. For this case, a necessary change would be to extend Definition 3.2 to those states which are not necessarily reachable from the initial state. Also, the two approaches of observing events or states can be considered. See [17] for more details concerning systems which can fail.

Another extension considers which sensors and actuators are required to be able to properly identify the system model from some set of possible models. See [16] for more details concerning the requirements for correctly identifying the correct model from a set of models.

References

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading, MA, 1974.

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

[3] Y. Brave and M. Heymann. "On Stabilization of Discrete Event Processes". *Internation Journal Control*, 51:1101–1117, 1990.

[4] R. Cieslak, C. Desclaux, A.S. Fawaz, and P. Varaiya. "Supervisory Control of Discrete-Event Processes with Partial Observations". *IEEE Transactions on Automatic Control*, 33(3):249–260, 1988.

[5] N. Deo. *Graph Theory with Applications to Engineering and Computer Sciences.* Prentice–Hall, Englewood Cliffs, N.J., 1974.

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

[7] Z. Kohavi. *Switching and Finite Automata Theory.* McGraw-Hill, New York, NY, 1978.

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

[9] F. Lin and W.M. Wonham. "On Observability of Discrete-Event Systems". *Information Sciences*, 44:173–198, 1988.

[10] C.P. Pfleeger. "State Reduction in incompletely specified finite state machines". *IEEE Transactions Computers*, 22(12):1099–1102, 1973.

[11] P.J.G. Ramadge and W.M. Wonham. "Supervisory control of a class of discrete event processes". *SIAM Journal Control Optimization*, 25:206–230, January 1987.

[12] P.J.G. Ramadge and W.M. Wonham. "The control of discrete event systems". *Proceedings of IEEE*, 77(1):81–98, January 1989.

[13] R. Sengupta and S. Lafortune. "Optimal control of a class of discrete events systems". In *Proceedings 1991 IFAC symposium on distributed intelligence systems*, pages 25–30, Arlington, VA, August 1991.

[14] J.N. Tsitsiklis. "On the control of discrete event dynamical systems". *Math. Contr., Signals, and Syst.*, 2(1):95–107, 1989.

[15] B. Weiss. "Subshifts of Finite Type and Sofic Systems". *Monatshefte fur Mathematik*, 77:462–474, 1973.

[16] S.D. Young and V.K. Garg. "Transition Uncertainty in Discrete Event Systems". In *Proceedings 6th IEEE International Symposium on Intelligent Control*, pages 245–250, Arlington, VA, August 1991.

[17] S.D. Young and V.K. Garg. "On Self-Stabilizing Systems: An Approach to the Specification and Design of Fault Tolerant Systems". In 32^{st} *IEEE Conference on Decision and Control*, San Antonio, TX, 1993.

[18] S.D. Young and V.K. Garg. "Optimal Sensor and Actuator Choices for Discrete Event Systems". In 31^{st} *Allerton Conference on Communication, Control, and Computing*, Allerton, IL, 1993.