March 1987 LIDS-P-1661

ON THE CONTROL OF DISCRETE-EVENT DYNAMICAL SYSTEMS

John N. Tsitsiklis[2]

ABSTRACT

We study a class of problems related to the supervisory control of a discrete–event system (DES), as formulated by Wonham and we focus on the computational effort required for their solution. While the problem of supervisory control of a perefetcly observed DES may be easily solved by dynamic programming, it is shown that the problem becomes intractable (in the sense of complexity theory) when a supervisor with a minimal number of states is sought. It is also shown that supervisory control is an intractable problem, in general, when imperfectly observed systems are considered.

1. Research supported by the Army Research Office (grant DAAL03–86–K–0171) and by a NSF PYI award, with matching funds from Bellcore Inc.

2. Room 35–214, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139.

MIT Document Services

Room 14-0551
77 Massachusetts Avenue
Cambridge, MA 02139
ph: 617/253-5668 | fx: 617/253-1690
email: docs@mit.edu
http://libraries.mit.edu/docs

DISCLAIMER OF QUALITY

Due to the condition of the original material, there are unavoidable flaws in this reproduction. We have made every effort to provide you with the best copy available. If you are dissatisfied with this product and find it unusable, please contact Document Services as soon as possible.

Thank you.

pgs. 6, 9, 11, 13 have text deletions.
This is the ~~of~~ Lids report provided.

I. INTRODUCTION.

Discrete Event Systems (DES) have been introduced by Ramadge and Wonham [RW1]. Roughly speaking a DES is a discrete time dynamical system such that, for each state, there is a number of different transitions that may occur. Furthermore, it is assumed that there is a possibility for control action through a supervisor who, at any given point in time, may prohibit certain transitions from occuring. It is then natural to consider the problem of designing such a supervisor satisfying certain specifications. Loosely speaking, the specifications that have been considered in the literature amount to a requirement that the supervisor prohibits from occuring certain (undesirable) sequences of events, while at the same time it allows some other (desirable) sequences of events to occur. Naturally, the supervisor design problem becomes different when different assumptions are made concerning the information available to the supervisor; for example the supervisor may have full knowledge of the state of the DES (perfect information), or it may have access only to some partial information on the state of the DES. Decentralized supervision by a set of noncommunicating supervisors, each one possessing partial state information, leads to another class of design problems.

A DES is very similar to a discrete time Markov chain, except that there are no assumptions on the probabilities of the different transitions out of a given state. For this reason, the supervisor design problem looks a little different from the traditional problems of Markovian decision theory, for which dynamic programming provides a solution [B]. On the other hand Markovian decision problems and the supervisor design problem for a DES are not completely unrelated. Consider the supervisor design problem under a constraint that certain states must be avoided. We may assign an infinite cost to the states to be avoided, zero cost to the remaining states, and assign arbitrarily a positive probability to each possible transition out of given state, thus defining a Markov decision problem. These two problems are closely related because any supervisor for the DES satisfying the specifications corresponds to a finite cost policy for the Markovian decision problem. Other types of specifications for the supervisor of the DES may be easily incorporated into the cost function of a corresponding Markovian decision problem; see Section III for a more detailed exposition.

Given the above remarks, it is natural to suspect that the types of problems which can be solved realistically within the DES framework (from a computational point of view) correspond to easily solvable problems in Markovian decision theory. Thus, in the light of available results [PT], it should be expected that problems with partial information are algorithmically intractable. One of the aims of this paper is to justify and give a precise content to the above statement.

The paper is organized as follows. In Section II we introduce the definitions, notation and terminology to be employed. In Section III we provide a brief background for the case of perfect information. In Section IV we consider the question of the implementation of a desired supervisor by a finite state machine with a minimal number of states. An exponential–time algorithm for this problem is available [VW] and we show that the problem is NP–hard, which implies that it is highly

unlikely that a faster algorithm may be found. In Section V we consider a variety of supervisor design problems when only partial information is available. While a special case of this problem, studied in [CDFW], is shown to be tractable, a number of negative results are derived for several interesting problems.

II. PRELIMINARIES.

A DES G can be defined [RW1] as a quadruple $G = (Q, \Sigma, \delta, q_0)$, where Q is a finite set (state space), q_0 is an element of Q (initial state), Σ is a finite alphabet (used to label possible transitions between states, also called events) and δ is a partial function (i.e. which is defined only on a subset of its domain) from $Q \times \Sigma$ into Q, which provides us with the dynamics of the system.[†] The interpretation of δ is the following: if $\delta(q, \sigma)$ is defined, for some given $q \in Q$, $\sigma \in \Sigma$, then it is possible that, starting from q, a transition carrying the label σ takes place and, in that case, the next state is equal to $\delta(q, \sigma)$.

Occasionally, we will find the following notation a little more convenient: for any $q \in Q$, we are given a set $\Sigma(q) \subset \Sigma$, the set of possible transition labels out of state q; in that case δ is a function (total, rather than partial) defined on the set $\cup_{q \in Q}(\{q\} \times \Sigma(q))$. In traditional systems–theoretic terminology, we are dealing with a dynamical system with state q, subject to uncertain disturbances σ; the system obeys the dynamical equation

$$q(t+1) = \delta(q(t), \sigma(t)); \qquad q(0) = q_0,^{\dagger\dagger} \tag{1}$$

and the disturbances $\sigma(t)$ are constrained to satisfy

$$\sigma(t) \in \Sigma(q(t)), \quad \forall t. \tag{2}$$

A *string* is the concatentaion of a finite (possibly empty) sequence $(\sigma(0), ..., sigma(t))$ of elements of Σ. Let ϵ denote the empty string and let Σ^* denote the set of all strings. We extend the function δ to a partial function from $Q \times \Sigma^*$ into Σ by means of the following recursive definition: $\delta(q, \epsilon) = q$ and $\delta(q, s\sigma) = \delta(\delta(q, s), \sigma)$, if $\delta(q, s)$ is defined and $\sigma \in \Sigma(\delta(q, s))$. In particular, $\delta(q, s)$ is equal to the current state, if the initial state is equal to q and the sequence of transitions represented by the string s has occured, assuming that this sequence of transitions is allowed by (2).

[†] Actually the definition usually given is somewhat more involved because it includes a special set Q_m of marked states ("accepting states", in the language of automata theory). We chose to omit them from the definition in order to simplify notation. Let us just mention here that the computational complexity of the problems considered in this paper is unaffected by the exclusion of marked states from the DES model.

[††] Time here is just a discrete variable used to index events and need not be related to "real time".

Any subset of Σ^* is called a *language*. We define $L(G)$, the language generated by G, as the set of all strings s such that $\delta(q_0, s)$ is defined. Notice that $L(G)$ always contains the empty string.

We now provide for the possibility of controlling a DES. We assume that the set Σ is partitioned into two disjoint subsets Σ_u and Σ_c. The set Σ_c is interpreted as the set of events which a supervisor may disable. We define a *supervisor* for G as a function $\gamma : \Sigma^* \mapsto 2^\Sigma$, such that $\gamma(s) \supset \Sigma_u, \forall s \in \Sigma^*$. The set $\gamma(s)$ is the set of events that are allowed by the supervisor to occur, as a function of the string s of past events. Accordingly, in the presence of a supervisor γ, we obtain a new dynamical system whose state again satisfies (1), but the constraint (2) now becomes

$$\sigma(t) \in \gamma(\sigma(0)...\sigma(t-1)) \cap \Sigma(\delta(q_0, \sigma(0)...\sigma(t-1)) = \gamma(\sigma(0)...\sigma(t-1)) \cap \Sigma(q(t)). \qquad (3)$$

A DES G together with a supervisor γ, are called a *supervised system*. Given a supervised system (G, γ), we define the language $L(G, \gamma)$ as the set of all strings in Σ^* that can be generated by that system. More formally, $L(G, \gamma)$ is the set of all strings $\sigma(0)...\sigma(T) \in L(G)$, which also satisfy (3), for each $t \leq T$, the empty string being included.†

In general, a supervisor need not have access to the entire string of past events; this may place a restriction on the set of supervisors under consideration. Consider a function $M : \Sigma \mapsto \Pi \cup \{\epsilon\}$, where Π is another finite alphabet and where ϵ denotes the empty string. We call such a function a *mask*. We interpret $M(\sigma(t))$ as the information provided to the supervisor on the value of $\sigma(t)$. However, the possibility that $M(\sigma(t))$ equals the empty string allows a situation where the supervisor does not learn that a transition has occured. We extend M to a mapping from Σ^* into Π^* by letting $M(\sigma(0)...\sigma(t))$ be the concatenation of $M(\sigma(0)), ..., M(\sigma(t))$. A supervisor γ is called an *M-supervisor* if there exists some function $\gamma_M : \Pi^* \mapsto 2^\Sigma$ such that $\gamma(s) = \gamma_M(M(s)), \forall s \in \Sigma^*$. Whenever we are given a mask M as above and we are allowed to choose γ only among the set of M-supervisors, we say that *partial information* prevails; if no such constraint is imposed, we say that *perfect information* prevails.

A special class of supervisors is the class of *state feedback supervisors*. A supervisor γ belongs to this class if there exists a function $\gamma_F : Q \mapsto 2^\Sigma$ such that $\gamma(s) = \gamma_F(\delta(q_0, s)), \forall s \in L(G)$.

Another interesting class of supervisors is the set of *finite state* supervisors. A supervisor γ belongs to this class if there exists a DES $\hat{G} = (\hat{Q}, \hat{q}_0, \Sigma, \hat{\delta})$ and a function $\gamma_R : \hat{Q} \mapsto 2^\Sigma$ such that a) $\hat{Q}$ is a finite set; b) $\hat{\delta}$ is a total function; c) $\gamma(s) = \gamma_R(\hat{\delta}(\hat{q}_0, s))$. Any such $\hat{G}$, together with the mapping γ_R is called a *finite state realization* of γ.

Let us point out that if Q is finite then any state feedback supervisor is also a finite state supervisor. The corresponding DES $\hat{G}$ is just a duplicate of the supervised DES G; it keeps track of the state $q(t)$ of G and at each time instance it chooses its supervisory action appropriately.

† Let us point out that γ could be a partial function defined only on the subset of Σ^* consisting of those strings whose occurence is possible, that is on $L(G, \gamma)$. However, we assume that γ is total to simplify notation and the discussion.

We shall use certain concepts from complexity theory which we mention briefly. We only consider "decision problems" that is problems in which a yes/no question is posed. As usual, P (respectively, NP, PSPACE) stands for the class of such problems solvable by a polynomial time (resp. non–deterministic polynomial, polynomial memory) algorithm. A problem is NP–complete (resp. PSPACE–complete) if it belongs to NP (resp., PSPACE) and any problem in NP (resp. PSPACE) may be reduced to via a polynomial time transformation. A problem is $NP--hard$, PSPACE–hard) if some NP–complete (resp. PSPACE–complete) may be reduced to it by a polynomial time transformation. We have $P \subset NP \subset PSPACE$ and it is widely conjectured that both inclusions are proper. If this conjecture is true, then there do not exist any polynomial time algorithms for NP–complete, NP–hard or PSPACE–complete problems. The reader is refered to [PS] for a more detailed and precise exposition of these concepts.

III. SUPERVISOR DESIGN: PERFECT INFORMATION.

A representative supervisor design problem introduced in [RW1] is the following: given three DES's G, G_1, G_2, employing the same alphabet Σ, we are asked to determine whether there exists a supervisor γ such that

$$L(G_1) \subset L(G,\gamma) \subset L(G_2).$$

We outline a solution to this problem.

Any DES may be modified so that the corresponding transition function is total. In particular, given a DES $G = (Q, q_0, \Sigma, \delta)$, we define a new DES $G' = (Q \cup \{*\}, q_0, \Sigma, \delta')$, where $*$ is a new (trap) state. We let $\delta'(q,\sigma) = \delta(q,\sigma)$, whenever $\delta(q,\sigma)$ is defined and $\delta'(q,\sigma) = *$, otherwise. Notice that $\delta'(q,s) = *$ if and only if $s \notin L(G)$. We assume that all three DES's introduced above (G, G_1, G_2) have been so augmented.

Consider a new DES consisting of the augmented versions of G, G_1, G_2, running simultaneously, under the influence of the same input sequence $(\sigma(0),\sigma(1),...)$ and starting from their respective initial states. Let $q'(t)$, $q_1'(t)$, $q_2'(t)$ denote their respective states at time t. We now interpret our supervisor specifications as state constraints. The inclusion $L(G_1) \subset L(G)$ requires that if $(\sigma(0),...,\sigma(t-1)) \in L(G_1)$ (that is, if $q_1'(t) \neq *$), then $(\sigma(0),...,\sigma(t-1)) \in L(G)$ (that is, $q'(t) \neq *$). Therefore, this constraint is captured by assigning infinite cost to any state (q', q_1', q_2') of the composite DES such that $q_1' \neq *$ and $q' = *$. Similarly, the constraint $L(G) \subset L(G_2)$ is equivalent to assigning infinite cost to any state (q', q_1', q_2') such that $q' \neq *$ and $q_2' = *$. Clearly, the original supervisor design problem has a solution if and only if there exists a control law for the above defined composite system under which the cost (starting from the appropriate initial state) is finite, for any possible sequence of events. This would be a standard dynamic programming problem: the only difference is that we are dealing with a worst case (minimax) criterion instead of an expected cost criterion. However, it is well known that the dynamic programming algorithm is equally applicable to such minimax problems and has polynomial computational requirements [B]. As this is a well–

known algorithm, we omit its detailed description. In fact the structure of this problem is so simple that the dynamic programming algorithm simplifies to a connectivity test; still, it is important to realize that the computational requirements of this problem are polynomial because it is a special case of a control problem solvable by dynamic programming.

An alternative design criterion that has been proposed is as follows: the objective is now to find a supervisor γ such that $L(G,\gamma)$ is maximal, subject to the constraint $L(G,\gamma) \subset L(G_2)$. This problem can be also formulated as aproblem amenable to dynamic programming. We do not provide the details which are rather trivial, but the key idea is the following: we express the requirement $L(G,\gamma) \subset L(G_2)$ as a state constraint (similarly with the previous problem) and we enforce maximality of the supervisor by introducing a penalty term which increases with the number of disabled transitions at each stage.

Notice that we are not suggesting that a reformulation to a traditional control problem be used in order to construct an algorithm for supervisor design problems such as the above. The value of the above arguments is that they prove with minimal effort that these problems are polynomially solvable.

IV. SUPERVISOR REDUCTION.

Let there be given a DES G and suppose that by means of some design procedure we have chosen a supervisor γ. Suppose, furthermore, that γ is a finite state supervisor. Since there are an infinite number of alternative finite state realizations of such a supervisor, we may be interested in finding a realization with a minimal number of states. This is the problem of optimal supervisor reduction and has been studied in [VW]. This reference provides an algorithm for constructing such a minimal supervisor. However, this algorithm requires, in general, a computational effort which is exponential in the number of states of G. While a polynomial algorithm would be desirable, it is shown below that this is very unlikely, because the problem under consideration is NP–complete.

In fact, we prove NP–completeness for a special case of the supervisor reduction problem: that is, we restrict to the case where the supervisor γ to be reduced is a state feedback supervisor. We can now formulate precisely the problem of interest:

Supervisor Reduction: We are given an integer L, a finite state DES G and a state feedback supervisor γ (in the form of a state feedback function $\gamma_F : Q \mapsto 2^{\Sigma}$). Does there exist a finite state supervisor γ_1 which has a finite state realization with no more than L states and such that $L(G,\gamma) = L(G,\gamma_1)$?

Theorem 3.1: "Supervisor Reduction" is NP–complete.

Proof: Notice that if L is larger than the cardinality $|Q|$ of the state space of G then the answer is always "yes". Thus, we assume, without loss of generality that $L \leq |Q|$.

We first show that "Supervisor Reduction" belongs to NP. If we are dealing with a "yes" instance a certificate testifying to this effect is a provided by a realization of a finite state supervisor γ_1 having

the desired properties. We only need to show that these properties may be tested in polynomial time. What is needed is a polynomial time test for deciding whether $L(G,\gamma) = L(G,\gamma_1)$. Since we are given a feedback map γ_F which determines γ, we have available a DES generating $L(G,\gamma)$. Similarly, having selected a finite state realization of γ_1, we have available a DES generating $L(G,\gamma_1)$. Furthermore, each one of the above two DES's has at most $|Q|^2$ states. We now use the fact that equality of the languages generated by two DES's may be tested in time polynomial in the cardinality of their state spaces. (A direct demonstration of this fact may be based on the argument provided in Section III, where we have shown that dynamic programming may be used for deciding whether one language contains another.)

We now proceed to show that the problem is NP–complete by reducing to it the graph coloring problem, which is known to be NP–complete [PS]. The latter problem is as follows:
Graph Coloring: Given an undirected graph (V,E) and a nonnegative integer K, does there exist a function $f : V \mapsto \{1,...,K\}$ such that $f(u) \neq f(v)$ whenever $\{u,v\} \in E$?

Let there be given a graph (V,E) and an integer K, corresponding to an instance of the graph coloring problem. Without any loss of generality we assume that there exists at least one arc adjacent to each node. We now define a corresponding instance of "Supervisor Reduction". We first define the state space Q. There will be two states v_i, v_i', corresponding to the i-th node of the graph V. We let q_0, the initial state, be equal to v_1. The alphabet Σ is equal to $\{\sigma_1,...,\sigma_n\} \cup \{\sigma_e : e \in E\}$, where n is the cardinality of V. The transitions of the unsupervised DES are as follows: for any state $q \in Q$, we have $\delta(q,\sigma_i) = v_i$; also, if $e = \{v_i, v_j\}$, then $\delta(q,e)$ is defined if and only if $q = v_i$ or $q = v_j$, and $\delta(v_i,\sigma_e) = v_i'$, $\delta(v_j,\sigma_e) = v_j'$. The state feedback supervisor γ is described by the state feedback map $\gamma_F : Q \mapsto 2^\Sigma$, defined by $\gamma_F(v_i) = \Sigma, \forall i$, and $\gamma_F(v_i') = \Sigma - \{\sigma_i\}, \forall i$. In other words, whenever a transition σ_e occurs, with $e = \{v_i, v_j\}$, the supervisor must be able to tell apart the states v_i' and v_j' so as accordingly to disable σ_i or σ_j. We also let $L = K + |V|$.

Suppose that the graph (V,E) is K–colorable and that f is a coloring function. We construct a supervisor with $|V| + K$ states by partitioning Q into $|V| + K$ disjoint subsets. For each color k, we have one subset S_k containing all states v_i such that $f(v_i) = k$; also, for each i, we have one subset consisting of the single element v_i'. Each state of the supervisor corresponds to one of the above subsets in the partition of Q and the feedback law used by the supervisor is the appropriate one: it disables σ_i if and only if the state of the supervisor corresponds to the subset $\{v_i'\}$. It remains to be shown that the supervisor is always able to generate its next state appropriately. Whenever a transition σ_i occurs, we know that the next state is equal to v_i; accordingly, the supervisor moves to the state $S_{f(v_i)}$. If a transition of the form σ_e occurs, with $e = \{i,j\}$, then the supervisor infers that the previous state of the system was either v_i or v_j. Since $e \in E$, it follows that $f(v_i) \neq f(v_j)$ and therefore the previous state of the supervisor is sufficient for these two states to be distinguished. Therefore, there exists a supervisor with $K + |V|$ states.

Suppose now that there exists a supervisor with $K + |V|$ states. For each i there must exist at

least one supervisor state at which σ_i is disabled. Therefore, there exist at most K supervisor states at which no σ_i is disabled. We number these states from 1 to K', where $K' \leq K$. Suppose that the first transition of the DES (starting from the initial state) is σ_i. Then, the first state of the DES is equal to v_i. Since no σ_j should be disabled at v_i, it follows that the state visited by the supervisor is one of those states at which no σ_j is disabled. Accordingly, we define a function $f : V \mapsto \{1, ..., K\}$ by letting $f(v_i) = k$ if and only if $\hat{\delta}(\hat{q}_0, \sigma_i) = k$, where $\hat{\delta}$ and $\hat{q}_0$ are the state transition function and the initial state, respectively, of the supervisor. We will show that $f(v_i) \neq f(v_j)$, whenever $\{i, j\} \in E$. Suppose the contrary. Then, there exist two nodes v_i, v_j, such that $f(v_i) = f(v_j)$ and $\{i, j\} \in E$. Consider the two scenarios whereby the first two transitions of the DES are (σ_i, σ_e) and (σ_j, σ_e), respectively. Since $f(v_i) = f(v_j)$, the supervisor moves to the same state after the first transition; since the second transition carries the same symbol, the supervisor state is the same under either scenario after the second transition occurs. However, after the second transition, the state of the DES is v'_i for one s[illegible]io and v'_j for the other and the supervisor must disable different a transition in each ca[illegible] contradiction which shows that $f(v_i) \neq f(v_j)$, $\forall \{i, j\} \in E$ and shows that (V, E) is [illegible]

V. SUPERVISOR DESIGN: PARTIAL INFORMATION.

Let M be a mask, as defined in Section II. We consider here certain supervisor design problems similar to those considered in Section III, except for the additional requirement that the supervisor designed is an M–supervisor. The simplest such problem addresses the question whether there exists some M–supervisor γ such that $L(G, \gamma) = L(G_1)$, where G_1 is a known DES. This problem has been studied in [CFDV] where the following result is proved:

Proposition 5.1: Given two DES's G, G_1 such that $L(G_1) \subset L(G)$ and a mask $M : \Sigma \mapsto \Pi \cup \{*\}$ there exists an M–supervisor γ such that $L(G, \gamma) = L(G_1)$ if and only if the following two properties hold:

(a) There exists a supervisor γ such that $L(G, \gamma) = L(G_1)$;

(b) If $s, s' \in L(G_1)$, $\sigma \in \Sigma_e$, $s\sigma \in L(G_1)$, $s'\sigma \in L(G)$ and $M(s) = M(s')$, then $s'\sigma \in L(G_1)$.

We now show that the conditions in Proposition 5.1 may be tested in a computationally efficient way:

Proposition 5.2: There exists a polynomial time algorithm (polynomial in the cardinalities of the state spaces of G and G_1) for deciding whether the conditions in Proposition 5.1 are valid.

Proof: (a) Testing this condition is equivalent to solving the first problem of Section III, for the special case where $G_1 = G_2$, and can be therefore done in polynomial time. (b) Let $G = (Q, q_0, \Sigma, \delta)$, $G_1 = (Q_1, q_0^1, \Sigma, \delta_1)$ and let $\Sigma(q)$, $\Sigma_1(q^1)$ be the allowed transitions under G, G_1 respectively, when the current state is q, q^1, respectively. Consider the following game: we start as the "state" (q_0^1, q_0^1, q_0). In general, at each point in time our state is a triple $(q^1, q^2, q^3) \in Q_1 \times Q_1 \times Q$ and we have the following options: choose some $\sigma \in \Sigma_1(q^1)$, choose some $\sigma' \in \Sigma(q^3)$, or choose both a σ and a σ' as above. The rules of the game are as follows: if we have chosen some $\sigma \in \Sigma_1(q^1)$ such

that $M(\sigma) \neq \epsilon$, then we must simultaneously choose some $\sigma' \in \Sigma(q^3)$ such that $M(\sigma) = M(\sigma')$. Similarly, if we have chosen some $\sigma' \in \Sigma(q^3)$ such that $M(\sigma') \neq \epsilon$, then we must simultaneously choose some $\sigma \in \Sigma(q^1)$ such that $M(\sigma) = M(\sigma')$. After we make our choices, the states move as follows: q^1 does not change if no σ is chosen; otherwise it moves to $\delta_1(q^1, \sigma)$. The states q^2, q^3, do not change if no σ' is chosen; otherwise, q^3 moves to $\delta(q^3, \sigma')$ and q^2 moves to $\delta_1(q^2, \sigma')$, except if $\sigma' \notin \Sigma_1(q^2)$, in which case the game terminates. We win if and only if the game terminates and the σ' causing the termination is equal to the last σ chosen and belongs to Σ_c.

Suppose that there exists a winning strategy in this game. Let $\bar{s} = (\sigma(0), ..., \sigma(m))$ and $\bar{s}' = (\sigma'(0), ..., \sigma'(n))$ be the sequences of σ's and σ' 's used in our winning strategy. Let $s = (\sigma(0), ..., \sigma(m-1))$ and $s' = (\sigma'(0), ..., \sigma'(n-1))$. Since we win, we have $\sigma(m) = \sigma'(n) = \sigma$, for some $\sigma \in \Sigma_c$. Therefore $\bar{s} = s\sigma$ and $\bar{s}' = s'\sigma$. Because of the rules of the game, we have $M(s) = M(s')$. Furth[illegible] $\in L(G_1)$, because at each time we choose a σ belonging to $\Sigma_1(q^1)$. Similarly, since the [illegible]erminated before $\sigma'(n)$ was chosen, it follows that in each choice except for the last one we had [illegible] $\in \Sigma_1(q^2)$ and therefore $s' \in L(C_*)$. Si[illegible] also have $s' \in L(G)$ and since $\sigma'(n) \in \Sigma(q^3)$, it follows that $s'\sigma \in L(G)$. On [illegible] we have won the game, we must have $\sigma \notin \Sigma_1(q^2)$. Therefore, $s'\sigma \notin L(G_1)$ and condition (b) is violated.

The above argument can be reversed: if there exist s, s', σ for which condition (b) is violated then we use them to define a winning strategy in the above game. Thus (b) holds if and only if there exists no winning strategy in our game.

It follows that it is sufficient to devise an algorithm which determines if there exists a winning strategy for the above game. This is just a deterministic optimal control problem on a finite state space, with state–dependent control constraints. Dynamic programming applies and provides a polynomial time algorithm for solving it, which concludes the proof. •

Proposition 5.2 is a positive result, especially given the fact that control problems with partial information are often intractable. Notice, however, that we have only found a way for deciding whether a M–supervisor exists, but we do not have yet an efficient method for constructing it. It is shown in [CDFV] that if there exists a M–supervisor γ such that $L(G, \gamma) = L(G_1)$ and if G, G_1 have finite state space, then the supervisor γ may be chosen to be a finite state supervisor. A reasonable choice for the state space of γ is to let it be equal to the power set of $Q \times Q_1$, where Q, Q_1 are the state spaces of G, G_1, respectively. With this choice, a state of the supervisor indicates the set of all states of G,G_1, which are possible, given the available information. However, such a state space has cardinality which is exponential in the size of the state space of G and therefore an exponential amount of computational resources is required to construct it. Given the positive result in Proposition 5.2, one might hope that a supervisor with a polynomial state space may be always found. The family of examples provided below shows that this is not so.

Example: Let us fix some positive integer n. The DES G to be supervised has an associated

alphabet $\Sigma = \{u_1, ..., u_n\} \cup \{d_1, ..., d_n\} \cup \{0, 1\} \cup \{\alpha_1, ..., \alpha_n\}$. The language $L(G)$ generated by G consists of all prefixes of strings of the form $(\sigma(0), ...\sigma(n+2))$ with the following properties: $\sigma(0) \in \{u_1, ..., u_n\} \cup \{d_1, ..., d_n\}$; $\sigma(i) \in \{0, 1\}$, for $i = 1, ..., n$ and $i = n+2$; $\sigma(n+1) \in \{\alpha_1, ..., \alpha_n\}$. Furthermore, if $\sigma(0) = u_k$, then $\sigma(k) = 1$ and $\sigma(n+1) = \alpha_k$; also, if $\sigma(0) = d_k$, then $\sigma(k) = 0$ and $\sigma(n+1) = \alpha_k$.

Notice that $L(G)$ is a finite language and therefore may be generated by a finite state DES. In fact, we may choose the state space of G to be as small as $O(n^2)$. This is done as follows: except for an initial state q_0, we let the other states be pairs (x, t) where t counts the number of transitions made so far and where x is equal to $\sigma(0)$. Figure 1 presents a state transition diagram for the case $n = 3$.

We observe G through a mask M defined as follows: $M(u_k) = M(d_k) = \epsilon$, $\forall k$, and $M(\sigma) = \sigma$ if $\sigma = \alpha_k$ or $\sigma \in \{0, 1\}$. Let our target language $L(G_1)$ be the same as $L(G)$ except that if $\sigma(0) = u_k$, we then require that $\sigma(n+2) = 1$ and if $\sigma(0) = d_k$, we require $\sigma(n+2) = 0$. Notice that G_1 is a finite state DES: it coincides with G except that we delete one of the two possible transitions out of any state that can be reached after exactly $n+2$ transitions. (The deleted transitions correspond to the heavy lines in Figure 1.)

It is easy to see that there exists an M–supervisor such that $L(G, \gamma) = L(G_1)$: the supervisor remembers $\sigma(1), ..., \sigma(n)$. When $\sigma(n+1)$ occurs, the supervisor observes α_k, for some k, and retrieves the value of $\sigma(k)$. If $\sigma(k) = 1$, (respectively, 0) it decides that the unobserved transition $\sigma(0)$ was equal to u_k (respectively, d_k), and decides accordingly which transition to suppress. This supervisor uses $O(2^n)$ states, since at time $n+1$ it remembers n bits of information and intuition suggests that no reduction of its state space is possible. We prove this formally. For any string $s = (\sigma(1), ..., \sigma(n))$, let $g(s)$ denote the state of the supervisor before $\sigma(n+1)$ is observed. The transition which is not disabled after $\sigma(n+1)$ is observed is therefore a function of $\sigma(n+1)$ and $g(s)$. Therefore, there exists some function f such that:
(i) $f(g(s), \alpha_k) = 1$, if $\sigma(k) = 1$;
(ii) $f(g(s), \alpha_k) = 0$, if $\sigma(k) = 0$;
Let s, s', be such that $s \neq s'$ and $g(s) = g(s')$. Assume that s and s' differ in their k–th symbol. Then, we must have $f(g(s), \alpha_k) \neq f(g(s'), \alpha_k)$, which implies that $g(s) \neq g(s')$. This shows that g is a one–to–one mapping and therefore its range has cardinality 2^n. Hence, the state space of the supervisor must have cardinality at least 2^n. We have thus constructed a family of partially observed supervision problems (parameterized by n) for which a M–supervisor exists (for each n) the state space of the DES being supervised has cardinality polynomial in n, but the state space of any M–supervisor must have cardinality exponential in n. Furthermore, this happens even though there exists a supervisor (which is not a M–supervisor) with small (polynomial in n) state space
•

The supervisor design problem of Propostion 5.1 seems to be about the only partial informatio

problem for which something can be done in polynomial time. We justify this claim by studying three variants of the imperfect information problem, all of which are found to be algorithmically intractable.

Problem A: This problem is very similar to the one considered in Section III. Given three finite-state DES's G, G_1, G_2, and a mask M, does there a M-supervisor γ such that $L(G_1) \subset L(G,\gamma) \subset L(G_2)$?

Proposition 5.3: Unless P=NP, there is no polynomial time algorithm for Problem A. (In particular, the complement of

Proof: We reduce the complement of the 3SAT ("three-satisfiability") problem of propositional calculus to Problem A. In this problem we are given n literals (Boolean variables) $v_1,...,v_n$, and K clauses $C_1,...,C_K$, and we are asked whether there exists an assignment of truth values to the literals so that all clauses are true. We now assume that an instance of 3SAT is given and we construct an equivalent instance of Problem A. We start by describing the DES G. It has a starting state q_0, out of which K transitions may happen, associated with the symbols $u_1,...,u_K$, and leading to different states. Suppose that u_k was the first transition. Then, the next n transitions carry labels 0 or 1. (The label of the i-th such transition will be associated with a truth value of the literal v_i.) After these n transitions are completed, the next state is one of two possible states x_k or x'_k. In particular, it is equal to x_k if and only if the sequence of zeroes and ones associated with the previous n transitions are such that the k-th clause is true. Although we have not transition diagram for G, it is easy to see that there exists a DES with the above describ and with a number of states polynomial in K and n. Figure 2 displays such a realization, for an example in which $K = 1$, $n = 4$.

Let the mask M be such that $M(u_k) = \epsilon$, $\forall k$, and $M(0) = 0$, $M(1) = 1$. Although we have not completed the definition of G, we pause to point out a property of that part of G that we have constructed.

Lemma 5.4: If the instance of 3SAT is satisfiable, then there exist strings $s^1,...,s^K$ of length $n+1$ such that $\delta(q_0,s^k) = x_k$ and $M(s^k) = M(s^j)$, $\forall j,k$. The converse is also true.

Proof of Lemma 5.4: Suppose that the instance of 3SAT is satisfiable and let s be a string (of length n) of zeroes and ones (corresponding to an assignment of truth values to $v_1,...v_n$) such that all clauses are satisfied. Let $s^k = u_k s$. It is easy to see that these strings have the desired property. For the converse, suppose that such strings $s^1,...,s^k$ exist. Since $M(s^k) = M(s^j)$ it follows that s^k and s^j agree on all except for their first symbol. Furthermore, since $\delta(q_0,s^k) = x_k$, it follows that the first symbol of s^k is u_k. Now, since $\delta(q_0,s^k) = x_k$ (rather than x'_k), for each k, it follows that the last n symbols of any s^k correspond to an assignment of truth values which satisfies all clauses. •

We now complete the description of G and of the supervisor specifications. Let there be no transitions possible out of x'_k. The transition diagram starting from the x_k's is depicted in Figure 3. We assume that $M(\tau_k) = \tau_k$ and $M(\sigma_k) = \sigma_k$, $M(w_k) = w_k$. This completes the description of

G and M.

The supervisor specifications are as follows. The transitions that are allowed to be disabled are all transitions labeled by σ_k and the transition w_k out of state $x_{k,K}$. However, we require that the particular transition σ_k leading into state $x_{k,k}$ is not disabled. These specifications are of the form $L(G,\gamma) \supset L(G_1)$, for a DES G_1 with the same transition diagram as G, except that certain arcs are deleted. Finally, we require that the transition w_k out of state $x_{k,K}$ be disabled. This requirement is of the form $L(G,\gamma) \subset L(G_2)$. Notice that our specifications for the w_k's amount to a requirement that the supervisor be able to tell apart the states $x_{1,K},...,x_{K,K}$.

We claim that a M–supervisor satisfying the above specifications exists if and only if we have started from a "no" instance of the 3SAT problem. Indeed suppose that we started with a "no" instance. Using Lemma 5.4, after the first $n+1$ transitions there is sufficient information available to exclude at least one state. Say that x_k is excluded. Then, the supervisor does the following: at each time $n+j$, $j=1,...,K$, it disables no transition, except that at time $n+k$ it disables σ_k. This is legal (conforms to the specification $L(G_1) \subset L(G,\gamma)$): since x_k has been excluded, the state at time $n+k$ is known to be different than $x_{k,k-1}$. With σ_k disabled, the transition that occurs at time $n+k$ reveals the identity of the current state: if τ_j is observed, the supervisor infers that the next state is $x_{j,k}$ and when eventually state $x_{j,K}$ is reached the supervisor knows that this is the case. Then, the supervisor is able to disable w_j, thus conforming to the specification $L(G,\gamma) \subset L(G_2)$.

For the converse, suppose that we were dealing with a "yes" instance of 3SAT. Then, if a string s^k of transitions occurs, where s^k is as in Lemma 5.4, then the supervisor has no information that could be used to discriminate between the x_k's. Since σ_1 cannot be disabled when the state is x_1 and since x_1 is a possible state given the available information, the supervisor cannot disable σ_1. (This is to conform to the specification $L(G_1) \subset L(G,\gamma)$.) Assuming that σ_1 actually occurs, then, at the next stage, the supervisor does not know whether the state is equal to $x_{2,1}$ and therefore cannot disable σ_2. By repeating this argument we see that there exists a scenario under which none of the σ_k's is disabled by the supervisor. Thus, at time $n+1+K$, the supervisor has no information useful in distinguishing between the states $x_{k,K}$ and therefore cannot disable the appropriate w_k, thus violating the specifications. This concludes the proof of Proposition 5.3. •

It is quite likely that a stronger result can be proved, although we have not been able to do so: that Problem A is actually PSPACE complete.

Problem B: Given two state DES's G and G_2, and a mask M, does there exist an M–supervisor γ such that $L(G,\gamma) \subset L(G_2)$ and such that "deadlock is impossible", meaning that we never come to a situation where all transitions out of the current state are disabled.

The "no deadlock" specification is equivalent to requiring that for every $s \in L(G,\gamma)$ there exists some $\sigma \in \Sigma$ such that $s\sigma \in L(G,\gamma)$. Such a specification cannot be expressed, in general, in the form $L(G_1) \subset L(G,\gamma)$. We will show that Problem B is intractable even for the special case where

$G_2 = G$. In this case, the equality $L(G,\gamma) \subset L(G_2)$ is trivially true and we are dealing with the following problem.

Problem B': Given a DES G does there exist an M–supervisor γ such that deadlock is impossible?

Proposition 5.5: Problem B' (and, a fortiori, Problem B) is PSPACE–hard.

Proof: The proof is patterned after the proof that the partial information Markov decision problem is PSPACE–complete [PT], with a few differences. It consists of reducing the QSAT problem (quantified satisfiability) of propositional calculus to Problem B. An instance of QSAT consists of $2n$ Boolean variables $v_1,...,v_{2n}$ and K disjunctive clauses $C_1,...,C_K$, with three literals per clause and the problem consists of deciding whether $\exists v_1 \forall v_2 ... \exists v_{2n-1} \forall v_{2n} [C_1 \wedge C_2 \wedge ... \wedge C_K]$, where $\wedge$ denotes conjunction. (The existential quantifiers $\exists$ are alternating with the universal quantifiers $\forall$ in the above formula.) We think of this problem as a game: an opponent assigns values to the even variables $v_2, v_4, ..., v_{2n}$ and we are to assign values the odd variables $v_1, v_3, ..., v_{2n-1}$, as functions of the past choices by the opponent; our objective is to have all the clauses satisfied.

Given an instance of QSAT we now construct an instance of Problem B'. The alphabet Σ is $\{u_0, u_1, ..., u_K\} \cup \{\tau_0, \tau_1, \tau_0', \tau_1'\}$. We let $\Sigma_c = \{\tau_0, \tau_1\}$. The mask M is such that $M(u_k) = \epsilon$, $\forall k$ and each τ_i, τ_i' is perfectly observed. We introduce an auxiliary function f defined by $f(\tau_0) = 0$ and $f(\tau_1) = f(\tau_1') = 1$. Accordingly, a sequence of $2n$ transitions not involving the u_k may be associated with a unique assignment of the variables $v_1, ..., v_{2n}$.

There is an initial state q_0 out of which any one of the transitions $u_1, ..., u_K$ may occur, leading to states $x_0, ..., x_K$, respectively. However, $M(u_k) = \epsilon$, $\forall k$, so that the state $q(1)$ after the first transition is unknown. (We may think of this as having the opponent choose a clause, but without revealing her choice.) The state transition diagram starting from any x_k, $k = 1, ..., K$ does not involve any u_j's and is such that the state $q(2n+1)$ is a "bad" state b_k (respectively, a "good" state g_k) if the assignments $v_k = f(\sigma(k))$ make the k–th clause true (respectively, false). (This is similar to the construction in Proposition 5.3 and is illustrated in Figure 4, for the case $n = 2$.) There are no transitions possible out of the bad states b_k, thus guaranteeing deadlock, whereas there is a transition w from every g_k to itself. Thus, deadlock avoidance is equivalent to guaranteeing that the state eventually reaches one of the states g_k, or, equivalently that the clause selected by the opponent is satisfied. Since the clause selected is not known (because $M(u_k) = \epsilon$, $\forall k$), we should satisfy all clauses. Notice that at odd times t the supervisor may disable τ_0 or τ_1, which corresponds to an assignment to the variable v_t, whereas at even times t the supervisor has no control, which corresponds to letting the opponent fix the value of v_t.

The proof will be completed by showing that such a supervisor exists if and only if we are dealing we a "yes" instance of QSAT. Suppose that we have a "yes" instance of QSAT. Thus, there exists a strategy through which at any odd time $t = 1, 3, ..., 2n-1$, we assign a value to v_t (equivalently, the supervisor enables exactly one of τ_0 or τ_1), as a function of past assignments, so that every clause is satisfied. This guarantees that the bad states b_k are avoided, for each k.

For the converse, suppose that we have a "yes" instance of Problem B; that is, there exists a supervisor conforming to the specifications. Since deadlock is prohibited, the supervisor enables at least one of r_0, r_1, at any odd time. Furthermore, if a supervisor exists, then there exists a supervisor which never enables both r_0, r_1. (Enabling both simultaneously amounts to leaving one more variable to the control of the "opponent" and this cannot be beneficial.) The strategy of this supervisor for deciding which r_i to enable at each odd time t then determines a strategy for assigning a value to the variable v_i so as to satisfy all clauses, which proves that we have a "yes" instance of QSAT and completes the proof. •

Problem C: Given a finite state DES G, two masks $M_i : \Sigma \mapsto \Pi \cup \{\epsilon\}$, $i = 1, 2$, and a finite state DES G_1 whose symbol alphabet is Π, does there exist an M_1–supervisor γ such that $M_2(L(G,\gamma)) = L(G_1)$?

Here $M_2(L(G,\gamma))$ stands for the language consisting of the images of all strings in $L(G,\gamma)$, under the mapping M_2. This is a reasonable design criterion if our performance specifications depend on the string of transitions s through some function M_2. That is, instead of constraining s directly, we only constrain $M_2(s)$. Such a specification may be used, for example, if want to impose a condition that the state of G eventually reaches a special state q^*, but we do not care about how it gets there.

Proposition 5.6: Problem C is PSPACE–hard.

Proof: We use exactly the same reduction as in the proof of Proposition 5.5. We
by $M_2(w) = w$ and $M_2(\sigma) = \epsilon$, $\forall \sigma \neq w$. We impose the specification $M_2(L(G$
is easy to see that this specification is equivalent to the no–deadlock specification we nad in the context of Proposition 5.5. Therefore Problem C is also PSPACE–hard, for the same reasons.•

REFERENCES

[B] Bertsekas, D.P., *Dynamic Programming and Stochastic Control,* Academic Press, 1976.

[CDFV] Cieslak, R., Desclaux, C., Fawaz, A., Varaiya, P., "Supervisory Control of Discrete–Event Processes with Partial Observations", Memorandum M86/63, Electronics Research Laboratory, University of California at Berkeley, 1986.

[PS] Papadimitriou, C.H., Steiglitz, K., *Combinatorial Opimization: Algorithms and Complexity*, Prentice Hall, 1982.

[PT] Papadimitriou, C.H., Tsitsiklis, J.N., "On the Complexity of Markovian Decision Processes", to appear in *Mathematics of Operations Research.*

[RW1] Ramadge, P.J., Wonham, W.M., "Supervisory Control of a Class of Discrete Event Processes", Systems Control Group Report No. 8515, University of Toronto, 1985.

[RW2] Ramadge, P.J., Wonham, W.M., "On the Supremal Controllable Sublanguage of a Given Language", Systems Control Group Report 8312, University of Toronto, 1984.

[VW] Vaz, A.F., Wonham, W.M., "On Supervisor Reduction in Discrete–Event Systems", *Interna-*

tional Journal of Control, 44, 2, 1986, pp. 475–491.

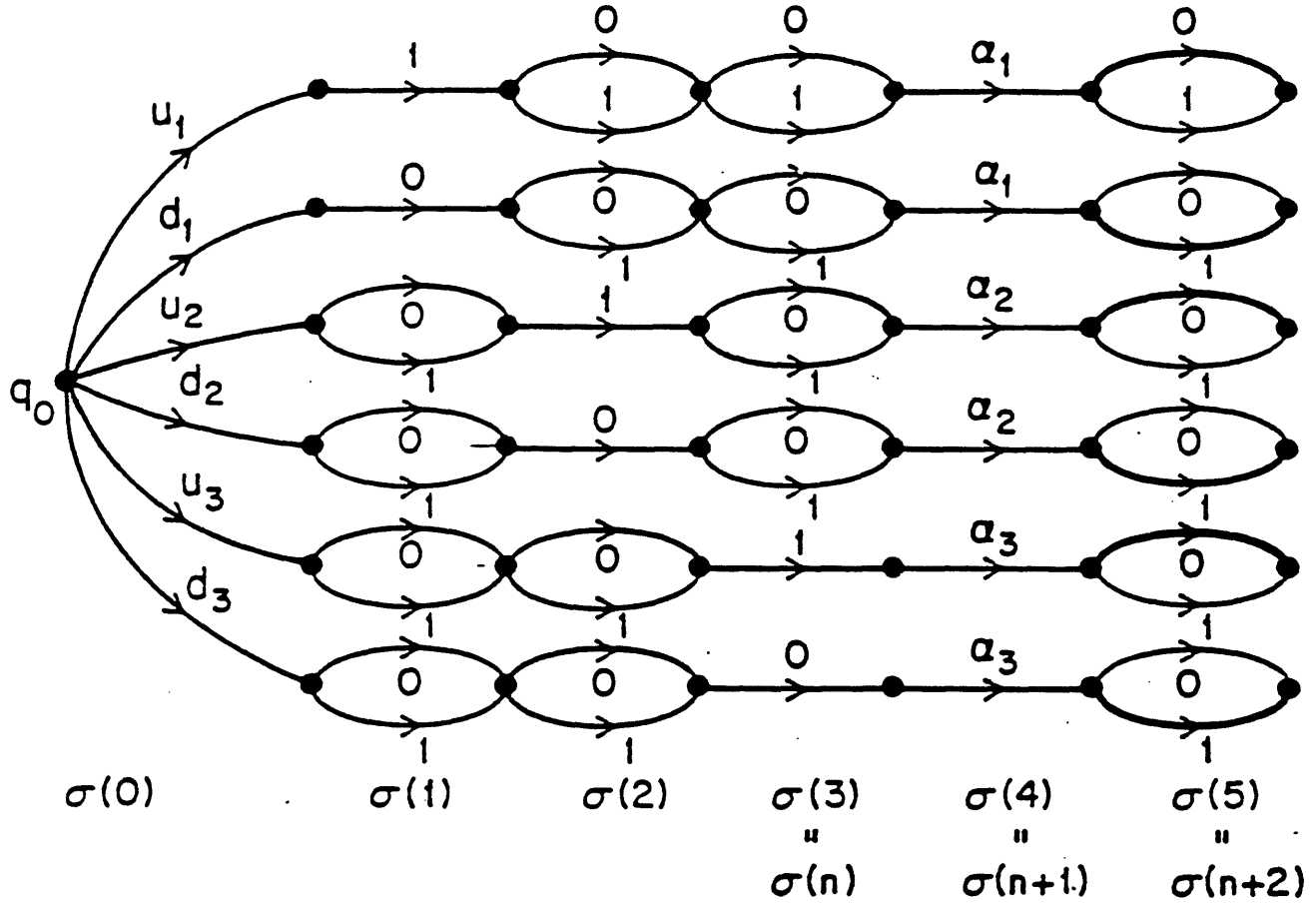
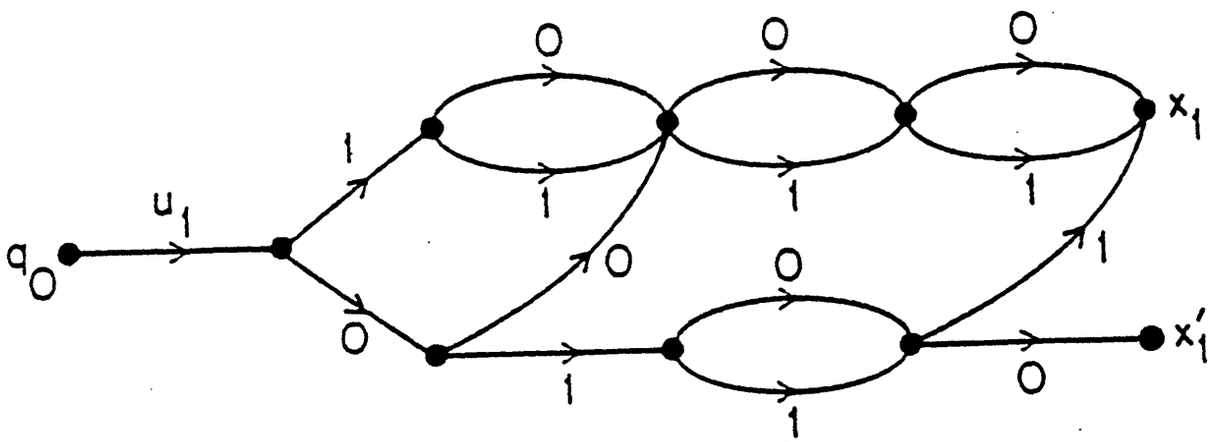
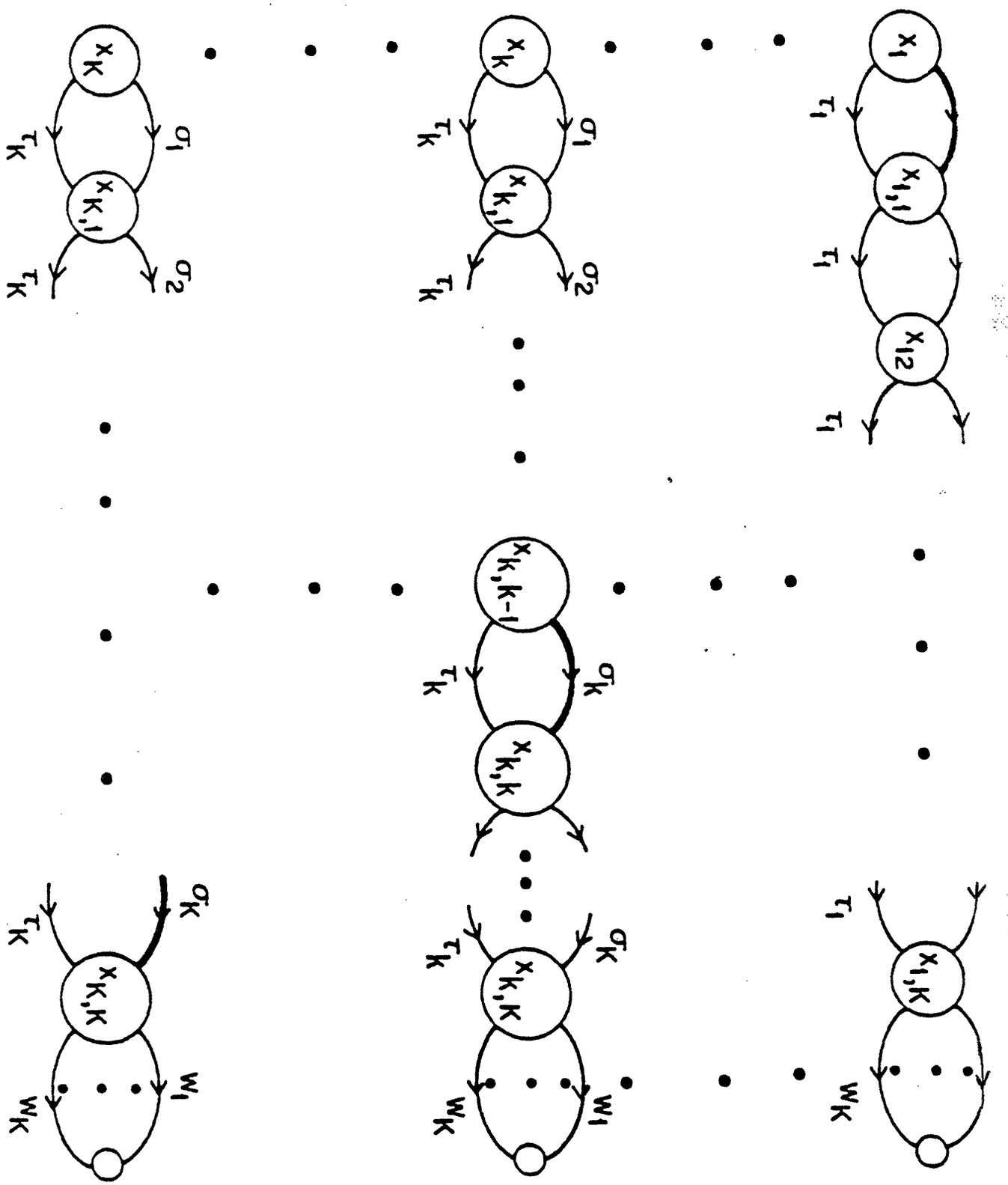

q_0
u_1
d_1
u_2
d_2
u_3
d_3
α_1
α_2
α_3
$\sigma(0)$
$\sigma(1)$
$\sigma(2)$
$\sigma(3)$
$\sigma(4)$
$\sigma(5)$
$\sigma(n)$
$\sigma(n+1)$
$\sigma(n+2)$

Figure 1



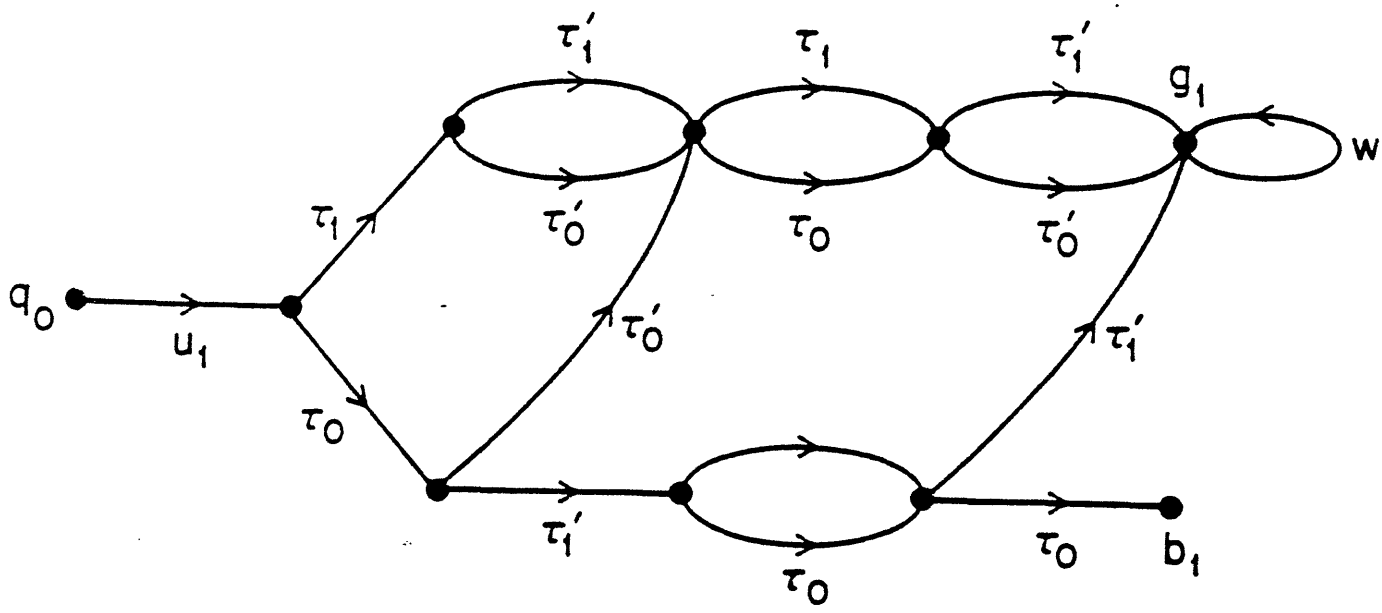
An example with $n = 4$, $K = 1$. The clause is $(v_1 \vee \bar{v}_2 \vee v_4)$. Notice that the state moves to the upper row once the clause is satisfied.

Figure 2



The supervisor cannot disable the r_k's. It may disable all σ_k's, except for those corresponding to heavy lines.

Figure 3



An example of the reduction in Proposition 5.5. Here $n = 2$ and there is a single clause $(v_1 \vee \bar{v}_2 \vee v_4)$.

Figure 4