SUPERVISORY CONTROL OF ELECTRIC POWER TRANSMISSION NETWORKS

Joseph Prosser[1] John Selinsky[1] Harry Kwatny[2] Moshe Kam[1]

[1]Data Fusion Laboratory, ECE Department
[2]Mechanical Engineering and Mechanics Department
Drexel University
32nd & Chestnut St.
Philadelphia, PA 19104

ABSTRACT

The methodology of discrete-event systems (DESs) and supervisory control is applied to a line-restoration problem, aiming to increase the steady-state security level of a power network during restoration. An example using the IEEE 14-bus system serves to demonstrate the potential of the DES formulation in the control of power networks, and to introduce the basic tools and techniques that this formulation offers.

Unlike other types of controllers, a DES-based supervisory controller does not serve to specify the exact control action at each state of the network. Rather, it defines an envelope of allowable actions. Within that envelope local controllers make specific decisions to satisfy local performance indices. This approach allows the creation of multi-level non-conflicting hierarchical control procedures, which are particularly attractive to managing large-scale systems. Each level in the hierarchical control structure defines the envelope of operation for the lower-level controllers through the enabling and disabling of controllable *events*. Component failure such as line or generator outages are formulated as uncontrollable events. Synthesis procedures for DES-based supervisory controllers can then be applied to synthesize hierarchical control for large power networks.

Keywords: Discrete Event Systems, Static Security Assessment, Line Restoration, Supervisory Control.

1. INTRODUCTION

Supervisory control using the formulation of discrete-event systems (DESs) has become an area of active and growing research [1]. The basic idea is to provide a formulation for analysis and control of large-scale systems where "events" occur at unpredictable times and cause changes in the state of the system and in the set of allowable future trajectories. Unlike stochastic control, where the objective is often to optimize a combination of moments and ensure good performance on average, the objective of DES-based supervisory control is to define an envelope within which the system can evolve so as to guarantee some performance and avoid "forbidden regions" (representing, for example, instability, critical malfunctions, or plant failure).

94 SM 547-0 PWRS A paper recommended and approved by the IEEE Power System Engineering Committee of the IEEE Power Engineering Society for presentation at the IEEE/PES 1994 Summer Meeting, San Francisco, CA, July 24 - 28, 1994. Manuscript submitted January 4, 1994; made available for printing June 1, 1994.

This DES methodology lends itself naturally to hierarchical structures, where each level of control defines the allowable envelope of operation for the subsequent lower levels. To accomplish this task, new definitions of controllability and observability and new controller-synthesis procedures have been developed during the last decade [1].

While most applications of DES analysis and control were so far limited to manufacturing processes and communication networks, there are strong indications that DES-based approaches are applicable and can be useful in the control of large power networks. This paper provides examples to support this claim and presents the methodology required for supervisory control in the context of network restoration. We show how a system with a number of faulty lines is gradually mended, so as to guarantee the highest possible security level throughout the restoration process. Specifically we use a 14-bus, 40-line transmission network; it is restored from any state of 36 operating lines to the highest possible steady-state security level, within a restoration envelope dictated by a four-step look-ahead supervisory controller.

The paper is organized as follows. In section 2 we provide the basic formulation of discrete-event systems and supervisory control. In section 3 we demonstrate a DES model of a power transmission network. In section 4 we show how the supervisory control guides a restoration process in a 14-bus 40-line system with four faulty lines.

2. DISCRETE EVENT SYSTEMS

A *discrete event system* is a dynamic system whose evolution in time is governed by the abrupt occurrence of physical events, at possibly irregular intervals. For example, an event could be the arrival of a packet in a communication system, the completion of a job or the arrival of a part in a production line, an error detection or disturbance in a complex control system, or the occurrence of a binary input pattern in a digital logic circuit. Discrete event system models have been employed in the control and scheduling of manufacturing systems [2], in logical models such as queues and communication protocols [3,4,5], and in the study of decentralized control theory [4,6,7,8]. Discrete-event models are generally used to describe systems where coordination and control is required to ensure the orderly flow of events, or to avoid the occurrence of certain chains of events.

Like continuous-time systems, a discrete-event system can be in any one of a set of internal configurations or *states*. The state of a system summarizes all the available information about past events, and is all that is needed to determine the behavior of the system upon subsequent events. Transitions between states are made instantaneously and are associated with the occurrence of *events*. Events occur spontaneously — but only one at a time — and cause transitions in the system from the present state to the next state.

To specify a DES model, it is necessary to identify the set of states (including an initial state), the set of events, and the transition structure of the system. Formally, a DES is represented by an *automaton* or *generator* $G = (Q, \Sigma, \delta, q_0, Q_m)$, which consists of a finite set of states Q, with $q_0 \in Q$ being the initial state; a finite set of events Σ; a transition map $\delta: \Sigma \times Q \rightarrow Q$; and a set of *final* or *marked* states $Q_m \subset Q$. The marked states are a subset of the system states that may represent desirable states or the completion of certain tasks.

An event $\sigma \in \Sigma$ may occur while the automaton is in state $q \in Q$ only if $\delta(\sigma, q)$ is defined. In that case, the automaton may make the transition to the state defined by $\delta(\sigma, q) \in Q$. In our analysis, G will play the role of the *plant*, with its states, events and transition structure modeling a physical process. This mathematical description of a discrete-event model can be represented by an associated *directed graph*, as shown in the following example.

Example 1

Suppose we model a system of two transmission lines, where each line has two possible states: either it is *in service* (state 1) or it is *out of service* (state 0). The possible system states consist of all combinations of possible line states, so Q={11, 01, 10, 00}; the first digit represents the state of line 1, and the second digit represents the state of line 2. Transitions between states are made upon the occurrence of events: either a line is restored and makes the transition from 0 to 1, or a line trips and makes a transition from 1 to 0. Let f_i represent the event 'line i restored' and let b_i represent the event 'line i has tripped'. The event set is $\Sigma = \{f_1, b_1, f_2, b_2\}$. The system is started with all lines in service, so the initial state q_0=11. State q_0 is also the most desirable state, so we set Q_m={11}. The transition function $\delta(\sigma,q)$ is defined by table 1. The directed graph associated with $G=(Q, \Sigma, \delta, q_0, Q_m)$ appears in figure 1. States which are marked appear with a double border.

Table 1 Transition function of example 1.

present state q	event σ	next state δ(σ,q)
11	b_1	01
11	b_2	10
10	b_1	00
10	f_2	11
01	f_1	11
01	b_2	00
00	f_1	10
00	f_2	01

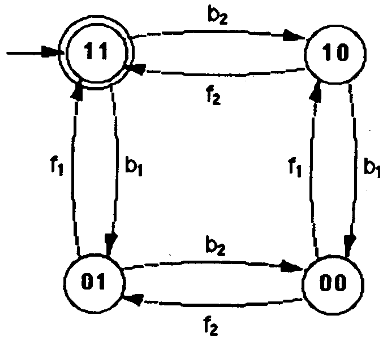

Figure 1: A directed graph associated with the generator of example 1.

The generator G is *deterministic* in the sense that if an event occurs, the system moves to a unique state. A *deterministic finite automaton*, or DFA, will form the basis of our discrete-event model for a power network.

The behavior of a discrete-event system is characterized by the event sequences that are produced during its operation. A sequence of events is called a *string*, and is formed by concatenating events. For example, the events a, b, and c can be concatenated to form the string abc. This represents the event a, followed by the event b, followed by the event c. The *null event*, denoted ε, represents the identity element for the concatenation operator, e.g. if a is an event or string, $a\varepsilon = \varepsilon a = a$. The length of the string s is denoted |s| and is the number of events in that string. For example, if s=abaca, then |s|=5. We define |ε|=0. The transition function can be extended over strings. This is accomplished by (i) defining $\hat{\delta}(\varepsilon,q) = q$; and (ii) defining $\hat{\delta}(\sigma,q)$ for $s=a_1a_2..a_k$, as the function $\hat{\delta}(s,q) = \delta(a_k, \delta(a_{k-1}, \ldots \delta(a_2, \delta(a_1,q))\ldots))$, provided all the functions involved are defined.

The *Kleene closure* of a set is the set of all strings which are formed by concatenating elements of the set, in any number or combination, with the null string included. The Kleene closure of a set A is denoted A^*. For example, if A = {a, b} (with a and b representing either strings or events), then $A^* = \{$ ε, a, b, aa, ab, ba, bb, aaa, aab, aba, ... }. The set Σ^* represents all of the strings which can be formed with the plant's event set (Σ).

A *prefix* of a string s is an event sequence which is an initial sequence of s, i.e. if w and s are any strings in Σ^*, u is a prefix of s if uw=s. If s = abcd, the set of prefixes of s is {ε, a, ab, abc, abcd}. A set which contains all of the prefixes of all of its elements is said to be *prefix closed*. Clearly, Σ^* is prefix closed. As some sets of strings may not contain all of their prefixes, we define the *prefix closure* of a set A, denoted $\overline{A}$, as the set containing all prefixes of every element of A. The set A is prefix-closed if $A=\overline{A}$. If A is not prefix-closed, then $A \subset \overline{A}$ (strictly).

A *language* is a set of strings (or words) formed by concatenating events (like letters of an alphabet). A *language over* Σ is any subset of Σ^*. The language *generated* by the plant G, denoted L(G), is the set of strings $\{s \mid s \in \Sigma^*, \hat{\delta}(s, q_0)$ is defined$\}$. The language L(G) contains all event sequences which are physically possible in the plant. For instance, in example 1, the strings b_1, $b_2f_2b_1$, and $b_1b_2f_2b_2f_1$ are all in L(G). Clearly, L(G) is a subset of Σ^*, and, since no event sequence in the plant can occur without its prefix occurring first, L(G) is also prefix-closed.

The *marked* language, denoted $L_m(G)$ is a subset of L(G), and consists of all those strings which can be extended to a marked state. More formally, $L_m(G) = \{s \mid s \in L(G)$, and there exist $w \in \Sigma^*$ with $\hat{\delta}(sw, q_0) \in Q_m\}$. A discrete-event system is said to be *nonblocking* if $L_m(G)=L(G)$. If Q_m represents a set of desirable marked states, and if G is nonblocking, it means that there always exists a sequence of events which takes the plant from any state to a marked state.

In some applications of discrete-event models, it is necessary to account for several independent and asynchronous processes simultaneously. For example, we may have two independent processes that must be coordinated. For this case, there exists an obvious procedure to generate a *shuffle product* of two DFAs, that combines two independent asynchronous processes (described by two generators G_1 and G_2) into a single new process described by a new generator $G_3 = G_1 \parallel G_2$ (see, for example [1]). The procedure in essence defines new states (of G_3) as ordered pairs of states from G_1 and G_2 and events of G_3 as the union of events in G_1 and G_2. The initial (correspondingly, marked) state of G_3 is the ordered pair of initial (marked) state in G_1 and G_2.

The control of discrete-event systems is generally performed through the *enabling* and *disabling* of events. Enabled events are allowed to occur in the plant, while disabled events are not allowed to occur. Events which may be disabled by a controller are called *controllable*, and events which may not be disabled are called *uncontrollable*. In example 1, for instance, we may not be able to prevent line 1 from tripping. Therefore b_1 is an uncontrollable event. We can, on the other hand, prevent line 1 from being restored once it tripped, so f_1 is a controllable event. A controlling agent for a discrete-event system is called a *supervisor*.

The supervisor controls the behavior of a discrete-event system by enabling and disabling events, hence affecting the actual event sequences and state trajectories of the plant. The supervisor's input is the string of events which occurred in the plant; its output is an *enable/disable map*. The supervisor assigns to each event a zero (0) or one (1). Assignment of zero means that the event is currently disabled; assignment of one means that the event is currently enabled. Often, the supervisor uses the state of the plant to determine which events to enable. In this case, the supervisor is a function $f: Q \rightarrow \Gamma$ which maps the plant's state to another map $\Gamma: \Sigma \rightarrow \{0,1\}$; $\Gamma(\sigma)=0$ means that the event $\sigma \in \Sigma$ is disabled and $\Gamma(\sigma)=1$ indicates that the event σ is enabled.

To separate events that are controllable from events that are not, a *controllable event set*, is defined. It is denoted Σ_c and is the subset of Σ which may be disabled by the supervisor. If $\sigma \notin \Sigma_c$, then $\Gamma(\sigma) \equiv 1$ for every $q \in Q$. Uncontrollable events are thus permanently enabled. Formally, a supervisor S is defined by the pair $S=(f, \Sigma_c)$ containing the supervisor's enable/disable map, and the set of events which it may control.

The behavior of the plant G under supervision of the supervisor S is a language denoted L(S/G), and is sometimes called the *closed-loop behavior*. The closed-loop behavior must take into account the flow of events in the plant as they are governed by the enabling and disabling actions of the supervisor. Clearly strings which can not possibly occur in the plant can also not occur under supervision. Therefore $L(S/G) \subset L(G)$. To calculate the closed-loop behavior, we construct a DFA which generates L(S/G) in the following way. Let $G=(Q, \Sigma, \delta, q_0, Q_m)$ and $S=(f, \Sigma_c)$ be as defined above. Let the generator of the closed-loop language be denoted S/G. Since $L(S/G) \subset L(G)$, both S/G and G have the same event sets. Also, since f depends only on the plant's state, both G and S/G have the same state sets and the same initial state. Let $S/G=(Q, \Sigma, \delta_f, q_0, Q_m)$, where the new transition function δ_f takes into account the enable/disable map f. The transition function must be modified so that transitions only occur when the event is enabled. Define $\delta_f(\sigma,q)$ only if $\delta(\sigma,q)$ is defined and if $f(q)(\sigma)=1$; in that case set $\delta_f(\sigma,q) = \delta(\sigma,q)$. Leave $\delta_f(\sigma,q)$ undefined in all other cases. The resulting DFA recognizes the closed-loop behavior.

3. DES MODEL OF AN ELECTRIC POWER TRANSMISSION NETWORK

Let the state of line i in an n-line power system network be denoted as $L_i \in \{0,1\}$, i= 1,...,n where

$L_i = 1$ denotes that line i is in service; and
$L_i = 0$ denotes that line i is out of service (contingency).

Transitions between states are due to the occurrence of events; either a line 'trips' and goes from being in service to being out of service, or a line is 'restored' and goes from being out of service to being in service. The desirable state is for the line to be in service. We also assume that on system start-up, all lines are in service. Let f_i denote the event 'line i is restored' and let b_i denote the event 'line i trips'. The following generator, represented by G_i, is the discrete-event model of this single power transmission line.

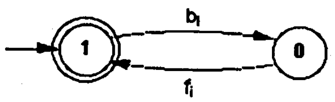

Figure 2: Discrete-event model of a single power transmission line.

To extend the model to cover all the lines in the network, we take the shuffle product $G=G_1||G_2||...||G_n$. The new model has 2^n states and 2n events (n trip events, n line-restoring events). The *configuration state* of the power system network, denoted N_s, is the vector of all the line states, $N_s= L_1L_2...L_n$. Both the initial state (denoted N_{s0}) and the (single) marked state (denoted N_{sm}) have $L_i=1$, i=1,...,n. The discrete-event model encompasses all possible configuration states and all transitions between them. We define $\Sigma_c=\{f_1, f_2, ... f_n\}$, meaning that all line trips are uncontrollable events.

Security Assessment

During operation, a power system is in a *secure state* when all load demands are met and all operating parameters are within limits. These limits may include generating capacity, voltage levels, and current flows in lines and transformers. The power system will be in an *insecure state* if any of the limits are violated. The system will be in *failure state* if any of the load demands cannot be met. While we realize that this (steady-state) categorization of system states is simplistic, it will suffice to introduce the methodology of the DES approach. The use of more accurate and more sophisticated security-assessment algorithms is a straight forward (if computationally-heavy) extension.

The construction of our control algorithm starts by determination of contingencies that will cause a power system, operating in the normal secure state, to enter an insecure state. The insecure state is defined in our example as either a violation of thermal limits of any transmission line or the creation of isolated subnetworks ("islands") due to line-trips.

There are 2^n possible configuration states. We will consider contingencies for which a maximum of m of the n lines in the network can be out of service. Then the number of configuration states is $C_n=\sum_{r=0}^{m} \binom{n}{r}$.
Given a configuration state, we need to test whether or not it is secure. Any exact or approximating technique for contingency selection [9] is a candidate to achieve this goal. In our examples we have first modeled the network as an undirected graph, and used a depth-first search of the graph [10] to quickly determine whether or not the given set of contingencies resulted in "islanding". Any configuration state which resulted in islanding was declared insecure. For the remaining questionable configuration states, power flows were computed and configuration states which resulted in thermal limit violations were declared insecure.

Given the set of states, we defined the *security level* of a configuration state q, denoted P(q), as the minimum number of line trips which will bring the system into an insecure state. Let E denote the set of insecure states. $P(q):=\min \{ |s| \mid s\in(\Sigma-\Sigma_c)^*, \hat{\delta}(s,q)\in E\}$. To each state a security level is assigned. Security level A is higher than security level B if more line trips must occur in A in order to bring the system into an insecure state.

Control Objective

The supervisor may disable the restoration event of a transmission line (the trip event of a line in service is uncontrollable). When the system is in a given state q, any line which is out of service may be restored. However, the restoration of some lines may increase the security level of the current state while the restoration of others may not. By allowing only increases in security level, we accelerate (at least in the short run) the movement of the system towards higher security levels. Let q_c denote the current configuration state and let Q_1 represent the set of states which are reachable from q_c through the restoration of a single line. We calculate the security level of each state in Q_1, and enable transitions into Q_1 according to the following rules:

1) if $P(\delta(f_i,q_c))>P(q_c)$, then set $f(q_c)(f_i)=1$;
2) if there are no states in Q_1 such that $P(\delta(f_i,q_c))>P(q_c)$, then let $f(q_c)(f_i)=1$ for all f_i where $\delta(f_i,q_c)$ is defined.

The interpretation of these rules is as follows: suppose it is possible for the system to increase its security level by restoring a certain line. Then the supervisor will allow the restoration of that line before it allows the restoration of any other line that will leave the system at the present security level. Thus, the supervisor enables the restoring of a line if it produces an increase in security level. If it is impossible to increase the security level by restoring any line, all restorations are allowed.

For example, consider a system with three lines, and let the insecure states be {000, 010}. Suppose the system is in state 100. The supervisor must decide which of the events {f_2, f_3} to enable. The state 100 has security level 1. If line 1 trips, the result will be 000, an insecure state. It is possible (through f_2) to move to state 110, and it is possible (through f_3) to move to state 101. The state 110 has security level 1, the state 101 has security level 2. The event f_3 is therefore enabled and f_2 is disabled. Once the system moves to 101, it may move to 111 by restoring line 2. Since both 101 and 111 have security level 2 and there are no possible restoration events which result in a security level of three, the event f_2 is now enabled. The order in which the lines are restored move the system to a more secure state as quickly as possible. By doing so, the supervisor removes some event-sequences or strings from the plant language. We are guaranteed, however, that the resulting supervisor is nonblocking, since there will always be at least one line-restoration event enabled, no matter what the configuration state of the system (except for q=111 ... 1, when all lines are in service.) Hence the system will always be able to reach the marked state eventually.

4. IMPLEMENTATION AND RESULTS

An electric power transmission network based on the IEEE 14-bus system was used as an example (see [11]), and is shown in figure 3.

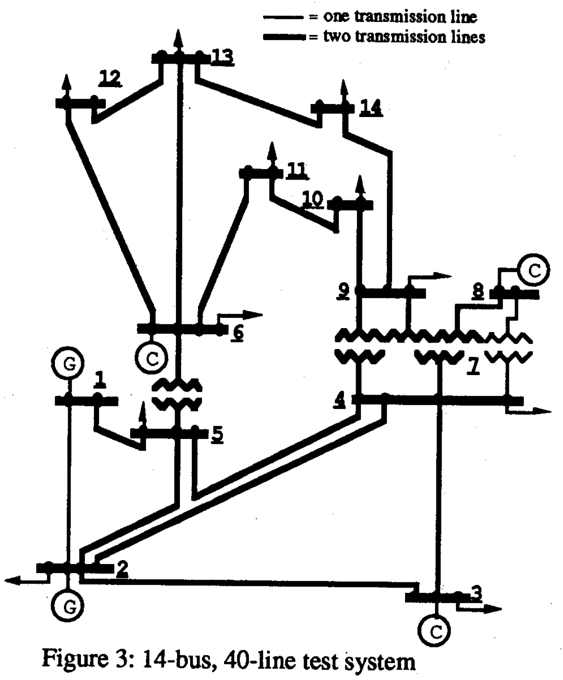

Figure 3: 14-bus, 40-line test system

This network is an extension of the standard system in that several transmission lines were added for a total of 40 lines. In the test system, each of the 14 buses is connected to the rest of the network by at least three transmission lines. Islanding can now occur only once a contingency consisting of at least three lines being out of service has occurred.

There are 40 lines in this system, resulting in 2^{40} possible configuration states. We will be working with a subset of the total number of configuration states consisting of contingencies with a maximum of 4 lines out of service. There are 102,090 configuration states in this subset.

The transmission line and transformer specifications for the test system are shown in table 2. Impedance and line-charging susceptance are shown in p.u. on a 100 MVA base. The line charging is one-half of the total line charging. The bus data for the system is the same as specified in [11].

Table 2: Transmission line and transformer data for the 14 bus system.

Line	Resistance (p.u.)	Reactance (p.u.)	Line Charging	Maximum MVar
1-2	0.01938	0.05917	0.0264	300
1-5	0.05403	0.22304	0.0246	300
2-3	0.04699	0.19797	0.0219	125
2-4	0.05811	0.17632	0.0187	125
2-5	0.05695	0.17388	0.0170	125
3-4	0.06701	0.17103	0.0173	125
4-5	0.01335	0.04211	0.0	125
4-7	0.0	0.20912	0.0	125
4-9	0.0	0.55618	0.0	125
5-6	0.0	0.25202	0.0	125
6-11	0.09498	0.19890	0.0	125
6-12	0.12291	0.25581	0.0	125
6-13	0.06615	0.13027	0.0	125
7-8	0.0	0.17651	0.0	125
7-9	0.0	0.11001	0.0	125
9-10	0.03181	0.08450	0.0	125
9-14	0.12711	0.27038	0.0	125
10-11	0.08205	0.19207	0.0	125
12-13	0.22092	0.19988	0.0	125
13-14	0.17093	0.34802	0.0	125
1-5	0.05403	0.22304	0.0246	300
2-3	0.04699	0.19797	0.0219	125
2-4	0.05811	0.17632	0.0187	125
2-5	0.05695	0.17388	0.0170	125
3-4	0.06701	0.17103	0.0173	125
4-5	0.01335	0.04211	0.0	125
4-7	0.0	0.20912	0.0	125
4-9	0.0	0.55618	0.0	125
5-6	0.0	0.25202	0.0	125
6-11	0.09498	0.19890	0.0	125
6-12	0.12291	0.25581	0.0	125
6-13	0.06615	0.13027	0.0	125
7-8	0.0	0.17651	0.0	125
7-9	0.0	0.11001	0.0	125
9-10	0.03181	0.08450	0.0	125
9-14	0.12711	0.27038	0.0	125
10-11	0.08205	0.19207	0.0	125
12-13	0.22092	0.19988	0.0	125
13-14	0.17093	0.34802	0.0	125
7-8	0.0	0.17651	0.0	125

The power flow analysis for the contingencies was performed using the Philadelphia Electric Company Power System Analysis Program (PSAP6 version) [12]. Normal operating conditions were established using line power flows and were calculated by AC load flow analysis. Combinations of contingencies were then simulated using distribution factors that were calculated during the power flow analysis.

The procedure to calculate the security level of a given state was executed as follows:

```
(0) i=1
(1) find all states reachable through i line trips
(2) are any of the states found in step (1) insecure?
        if yes, security level=i.
        if no, increment i by 1 and go to step (1).
```

Suppose that a certain contingency in our 40-line system consists of lines 34 and 40 being out of service. Then the current configuration state is

q_c = 1111111111111111111111111111111110111110.

From analysis of the network we found that q_c has a security level of two. The reason for this is that if line 15 and line 29 go out of service while we are in q_c, then a thermal limit violation occurs on line 10.

There are two possible states which can result from placing one line back in service in state q_c. These are

q_{f1}=1111111111111111111111111111111111111110,

which results from the restoration of line 34; and

q_{f2}=1111111111111111111111111111111110111111

which results from the restoration of line 40.

State q_{f1} was analyzed to have a security level of three. Indeed, there would be an increase in system security when line 34 is put back into service. The supervisor would therefore enable the transition to q_{f1}.

State q_{f2} was analyzed to have a security level of two. From a system security point of view there is no immediate gain in restoring line 40; the transition to state q_{f2} would therefore be disabled.

To illustrate the advantage of using our supervisor we have compared the average security level of the system as it was restored from various contingency states to the marked state. The two methods that we compared were

a) using a supervisory controller to enable the restoration only of those out-of-service lines which resulted in a higher security level; and

b) enabling all line-restoration events.

In both cases, restoration within the permitted envelope was performed in arbitrary order.

The system was restored from a set of 150 test states. Each state consisted of 4 lines being out of service. Each test state was restored 1000 times using both methods. A line was selected from among those that had their restoration events enabled. The selected line was restored, and the corresponding transition to the next state was made. The procedure continued until all four out-of-service lines had been restored, and the system arrived at the marked state. The average security level during the restoration from each test state was recorded. By that we mean that we averaged the security levels of all the states that the system has visited during restoration.

For 144 of the 150 test states, use of the supervisory controller resulted in higher average security levels. For the remaining six test states, use of the supervisory controller resulted in average security levels which were equal to those obtained using unrestricted restoration order. In none of the test states did the use of the supervisory controller result in lower average security levels. The average security level obtained with the supervisory controller was 2.17, while the average security level obtained without it was 1.98.

CONCLUSION

We have presented the basic terms and methods used in discrete-event systems, and applied them to a simple line-restoration problem with average steady state security levels as the performance index. In our example the supervisory controller enabled or disabled the restoration of out-of-service lines, and the local controllers restored the lines within the allowable-restoration envelope in arbitrary order. In spite of the simplicity of our example, it demonstrates how hierarchical supervisory controllers for networks can be constructed, and how control actions in power networks can be formulated in the terminology of discrete-event systems. The formulation of power-network control problems in this terminology will allow the importation of DES-based algorithms for controller synthesis into the area of power system control.

In the present exposition we exemplified the role of *event controllability*, and modeled component failure as uncontrollable. More sophisticated models would use the notion of *event observability*, and will include the possibility that some events will be unobservable to supervisory controllers at certain levels of the control hierarchy. Inclusion of unobservable events in the model would allow us to address sensor selection and sensor placement. They will also allow the study of control networks under partial uncertainty (as to the network's state), and under inaccurate and conflicting measurement data.

ACKNOWLEDGEMENTS

This research was jointly supported by the National Science Foundation and the Electric Power Research Institute (grants ECS-9216588 and RP 8030).

REFERENCES

[1] Ramadge, P. J. and Wonham, W. M. "The Control of Discrete Event Systems," *Proceedings of the IEEE*, Vol. 77, No. 1, Jan. 1989, pp. 81-98.

[2] Sharfina, A. "Production Control of a Manufacturing System with Multiple Machine States," *IEEE Transactions on Automatic Control*, Vol. 33, No. 7, Jul. 1988, pp. 620-625.

[3] Osroff, J. S. "A Logic for Real-Time Discrete Event Processes," *IEEE Control Systems Magazine*, Jun. 1990, pp. 95-102.

[4] Rudie, K. and Wonham, W. M. "Protocol Verification Using Discrete-Event Systems," *Proceedings of the 31st Conference on Decision and Control*, Tuscon, AZ, Dec. 1992. pp. 3770-3777.

[5] Rudie, K. and Wonham, W. M. "Supervisory Control of Communicating Processes," in *Protocol Specification, Testing, and Verification*, X. L. Logrippo, R. L. Robert, and H. Ural (editors), Elsevier Science Publishers B. V.(North-Holland) 1990.

[6] Lin, F. and Wonham, W. M. "Decentralized Supervisory Control of Discrete-Event Systems," *Information Science*, Vol. 44, 1988, pp. 199-224.

[7] Ramadge, P. J. and Wonham, W. M. "Modular Feedback for Discrete Event Systems," *SIAM Journal of Control and Optimization.*, Vol. 25, No. 5, Sept. 1987, pp. 1202-1218.

[8] Wonham, W. M. and Ramadge, P. J. "Modular Supervisory Control of Discrete Event Systems," *Mathematics of Control, Signals and Systems*, Jan. 1988, pp.13-30.

[9] Blau, N., T. Bertram, A. Bose, V. Brandwajn, G. Cauley, D. Curtice, A. Fouad, L. Fink, M.G. Lauby, B.F. Wollenberg, and J.N. Wrubel "On-line Power System Security Analysis," *Proceedings of the IEEE*, Vol. 80, No. 2, Feb. 1992, pp. 262-280.

[10] Sedgewick, R. *Algorithms*, Addison-Wesley, 1983, pp. 381-387.

[11] Freris, L.L. and Sasson, A.M. "Investigation of the Load Flow Problem," *Proc. IEE*, Vol. 115, No. 10, Oct. 1968, pp. 1459-1470.

[12] Philadelphia Electric Co., *PSAP6: Users Manual.*

Joseph Prosser received the B.S. and M.S. degrees in Electrical Engineering from Drexel University in 1991 and 1993, respectively. His M.S. thesis was entitled *Centralized Supervisory Control of Discrete-Event Systems*. At the present time he is a candidate for the Ph.D. degree at the Data Fusion Laboratory (DFL), ECE Department, Drexel University. He is investigating limited-lookahead policies in the synthesis and control of discrete-event systems and Petri nets. Mr. Prosser's professional interests also include modeling and synthesis of robotic systems, and he participates in the development of the DFL's autonomous tracked vehicle and robotic hopping machines. His work in these areas was documented in the Proceedings of the IEEE Conferences on Decision and Control (CDC), the Conferences on Information Sciences and Systems (CISS) and the SPIE Conferences on Mobile Robots. He is a member of Sigma Xi.

John Selinsky received the B.S. and M.S. degrees in Electrical Engineering from Drexel University in 1987 and 1990, respectively. He is currently a candidate for the Ph.D. degree at the Data Fusion Laboratory (DFL), ECE Department, Drexel University. He is investigating Supervisory Control policies for large-scale systems with emphasis on the control of interconnected power systems. His research interests include the modeling and control of nonlinear systems, pattern recognition, neural networks, and robotics. His past studies were described in the proceedings of the International Joint Conferences on Neural Networks, the Journal of Robotic Systems, and the Journal of Intelligent and Robotic Systems. He is a member of Eta Kappa Nu and Tau Beta Pi.

Harry Kwatny received the B.S. degree in Mechanical Engineering from Drexel University, the S.M. degree in Aeronautics and Astronautics from the Massachusetts Institute of Technology, and the Ph.D. degree from the University of Pennsylvania, in 1961, 1962, and 1967, respectively.

In 1963 he joined Drexel University as an instructor and currently he is the S. Herbert Raynes Professor of Mechanical Engineering. His current research interests include modeling, analysis, and control of nonlinear and distributed dynamical systems. In recent years, he has focused on the stability and bifurcation analysis of electric power networks and the control of nonlinear, flexible spacecraft and aircraft. He has been a consultant to industrial and government organizations in the general area of dynamic system analysis and control, especially in the operation and control of interconnected power systems, electric generating plants, industrial extrusion and drying processes, shipboard systems for aircraft operation, missile guidance and control, heating and air conditioning systems and large rotating machinery.

Dr. Kwatny was an Associate Editor of the *IEEE Transactions on Automatic Control* for several years and is currently an Associate Editor of the IFAC journal *Automatica*. He is a member of Pi Tau Sigma, Tau Beta Pi, Phi Kappa Phi, and Sigma Xi.

Moshe Kam received his B.Sc. degree from Tel Aviv University in 1977 and his M.S. and Ph.D. degrees from Drexel University in 1985 and 1987 respectively. From 1976 to 1983 he was with the Israeli Defense Forces, and since 1986 he has been with the ECE Department at Drexel University where he is now an Associate Professor and the Director of the Data Fusion Laboratory.

Dr. Kam's primary professional interests are analysis and design tools for large-scale systems, especially multi-sensor detection architectures. He has investigated and written about sensor fusion, neural networks, decision theory, and robotics. He is an associate editor of *IEEE Transactions on Systems, Man, and Cybernetics* and of *Pattern Recognition.* In 1990 he received a NSF Presidential Young Investigator award, and in 1991 he received the Eta Kappa Nu C. Holmes MacDonald award for the outstanding Electrical Engineering educator. He is a member of Eta Kappa Nu, Tau Beta Pi, and Sigma Xi.