Discrete Events:

Perspectives from System Theory

J. M. Schumacher
Centre for Mathematics and Computer Science
P. O. Box 4079, 1009 AB Amsterdam, The Netherlands
Department of Economics, Tilburg University
P. O. Box 90153, 5000 LE Tilburg, The Netherlands

Mathematical system theory has traditionally been concerned with differential or difference equations. Recently, however, there has been increasing interest of system theorists for 'discrete event systems', whose evolution in time is marked by the occurrence of 'events' (arrivals of messages or customers, completion of tasks, etc.). This paper presents a brief survey of the approaches to discrete events that have been developed so far within the system theory community.

1. Introduction

Recently, there has been a surge of publications in the system theory literature promising to extend the working domain of this discipline to certain systems whose dynamics are not described by differential or difference equations. The term 'discrete event dynamic systems' (or just 'discrete event systems') is used to describe the area covered by these papers, and one speaks about 'system-theoretic approaches to discrete events'. The purpose of this paper is to give an introductory survey of what has been achieved under this heading so far. Exactly what is to be understood by a 'system-theoretic approach' can perhaps be subject to debate; we will take the easy way out here, and consider mainly papers that have appeared in the established systems and control journals, such as the *IEEE Transactions on Automatic Control* and the *SIAM Journal on Control and Optimization*. As a consequence of this limitation, several important approaches to discrete events will be covered only partly or not at all. This applies for instance to queueing networks, discrete event simulation, and theories of concurrency.

This paper is based in part on discussions that took place in the seminar on discrete event systems that was organized by the author at the Centre for Mathematics and Computer Science in Amsterdam in the spring semester of 1987. I would like to thank all the speakers that gave presentations in the seminar*: Ton de Kok (Centre for Quantitative Methods, Philips), Kees Praagman (Eindhoven University of Technology), Jan Willem Polderman (CWI, now at Twente University), Jan H. van Schuppen, Rein Smedinga (University of

* Affiliation omitted for members of CWI's Department of Operations Research, Statistics, and System Theory.

Groningen), Stef Smulders (CWI, now with the Dutch Department of Public Works), Michael Stöhr (visiting from the University of Kaiserslautern), Frits Vaandrager (CWI, Department of Software Technology), Remco de Vries (Delft University of Technology), Peter de Waal, and Henk Zijm (Centre for Quantitative Methods, Philips, and Eindhoven University of Technology). The statements below fall under the author's responsibility, however, so the people mentioned above cannot be blamed for these. In the ten two-hour sessions of the spring seminar, more was said than could be taken into this paper; our survey is intended to be representative rather than exhaustive. Of course, quite a bit has happened since the seminar took place in 1987. In fact, the literature on the subject has expanded almost explosively. Below, some of the recent literature will be covered, but certainly not all of it; for additional references, a good source is the January 1989 special issue of the Proceedings of the IEEE [6].

2. What are discrete events?

The dynamical nature of a process is often expressed by letting the process variables be functions of a parameter t, the time. However, there are also processes whose evolution in time cannot naturally be described in this way. Consider, for instance, a production line in which parts are produced and assembled into a final product. The state of the process may be specified in terms of the numbers of parts that are available at various stages, and the status of the machines (broken / active). The evolution of the state is determined by events such as a part being completed or a machine breaking down. These events are called 'discrete events'; the adjective is suggested by the facts that the states form a discrete (finite, or at most countable) set. At this level of description, real time is not present at all; a machine will produce no parts as long as it is broken, regardless of the amount of real time that is involved.

Other situations in which a 'discrete event' formulation is useful can be found in communication networks, multi-programmed computers, and traffic. Sometimes, one may want to include certain time intervals into the description - for instance, the time needed by a machine to produce a single part, or the time-out interval of a communication line that is waiting for a message. This would bring in a real-time aspect, but a full description in real time may still not be possible or even desirable.

There is a mutual relation between states and events: events may change the state, but the state determines which events may or may not occur. A state transition matrix might be used to model such a situation, but usually one would like to have models with more structure. A number of such models will be discussed below.

3. What is system theory?

Mathematical system theory is concerned with dynamical phenomena, and in particular with questions that arise from prediction and control of such phenomena. The origins of the field lie in electrical engineering (theory of electrical networks) and in mechanical engineering (design of servomechanisms). It now stands as a branch of applied mathematics that provides a link between

engineering problems and a great variety of mathematical disciplines, such as the theory of ordinary, partial, and stochastic differential equations, linear and commutative algebra, function theory, and differential and algebraic geometry.

The birth of mathematical system theory as a discipline of its own took place in the fifties. Let us mention here a few highlights and catchwords that may ring a bell to those who are not very familiar with the field. In the early years of system theory, the most appealing developments took place in the field of *optimal control*, with the development of Pontryagin's maximum principle and Bellman's dynamic programming method. Typical books for this period are [21] by Pontryagin et al. and [1] by Bryson and Ho. In the sixties, the work of Kalman led to the rediscovery of *linear systems* as a challenging research area; key topics were state space realization, linear-quadratic optimization, and the Kalman filter. This line of development is perhaps best characterized by the book [15] by Kalman, Falb, and Arbib. The seventies saw, among many other developments, a quickly increasing interest in a variety of *controller synthesis problems*. Techniques were developed to design controllers that could achieve various goals, not necessarily stated in the form of optimization criteria. A book which clearly represents this trend is [29] by Wonham. In the eighties, *robustness* and *adaptation* are to be counted among the most popular phrases in control theory.

4. Perturbation analysis

Discrete-event dynamical systems are often modeled as *networks of queues*, in which 'customers' (which may also be interpreted as parts or messages) are handled by 'servers' (machines, processing units) which are combined into a network. The arrival times of the customers and the service times needed by the servers are modeled as stochastic processes. There are quite a few degrees of freedom in specifying a network of queues (arrival time distributions, service time distributions, network topology, switching strategy, serving disciplines, buffer sizes, and more), and therefore there is rather a rich variety of models that fall into this class. An important goal of the analysis of such networks is to compute *performance measures* such as throughput, average queue length, and average sojourn time. A completely analytical solution is often not available and so the results are generally found by a mixture of analytical methods, approximations, and simulations.

The above already suggests that the analysis of a single network may present a formidable task. In practice, though, one is often interested in *optimization*; given a certain amount of freedom in adjusting various parameters and features of a network, one would like to optimize the performance measures. Using straightforward methods to search for an optimum calls for the evaluation of many different networks, and the computational effort involved may soon turn out to be quite forbidding. Analytical tools which help to reduce the amount of computation are, therefore, badly needed. One such tool is provided by the *perturbation analysis* of queueing networks, developed at Harvard University and other institutions by Y.-C. Ho and co-workers from the early eighties on. The first papers to expose the technique in general terms were [11] and [13]. The perturbation analysis technique allows, under certain conditions, to compute gradients of

performance measures with respect to network parameters from a single trajectory. The computational saving may easily reach a factor of ten, when the perturbation analysis is compared with a straightforward determination of these gradients by doing repeated simulations with small parameter changes. The gradient information can obviously be used in optimization schemes; the fact that only a single trajectory is needed means that one may even use the perturbation analysis technique to do optimization on-line, while the system is functioning.

Perturbation analysis of queueing networks exists, in fact, in various forms. The basic idea may be sketched as follows. Consider a long but finite nominal trajectory. If a change in a nominal network parameter (such as the service rate at a particular node) is small enough, the *order* of the events in the trajectory will not be changed. Of course, the event *occurrence times* will change, either as a direct consequence of the parameter perturbation, or because of propagation of perturbations through the network. The changes in occurrence times caused directly by perturbations can be computed analytically, whereas the propagation through the network can be followed by simple 'propagation rules'. For instance, if the arrival of a customer at a certain node is delayed but the nominal trajectory indicates that the customer would have had to wait there anyway, then this disturbance is not propagated beyond the node. Note that propagation can also take place in the backward direction: for instance, if the perturbation makes a certain server slower, and if this server has a finite buffer which at a certain moment is already full in the nominal trajectory, then the preceding servers will have to wait longer before they can output to the slowed server's buffer. Using the information about the perturbed event occurrence times, one can compute perturbed values of performance measures one is interested in. Since we assumed small perturbations in order to justify the assumption that the order of the events is not changed, the result of the computations is essentially the gradient of the performance measures with respect to the perturbed parameter.

Simple as the basic idea may be, the questions raised by the approach are manifold and intricate. For instance, to obtain an estimate of the sensitivity of the expectation of a performance measure with respect to a certain parameter, one would be led to average the results that one gets by the method described above from a large number of samples (or, with an appeal to ergodicity, from one very long sample). However, in this way one obtains an approximation of the expected gradient, which need *not* be the same as the gradient of the expectation - in particular, this situation will often prevail when the performance measure as a function of the parameter shows jumps at sample dependent points. This point is made very clear in [2], a paper for which the author, Xi-Ren Cao, received an Outstanding *Transactions* Paper Award at the 1987 Control and Decision Conference.

A simple example may illustrate the issue, which has given rise to a considerable research effort (see the survey in [10]). Suppose we are interested in computing the gradient of the expected time at which we arrive at our office (performance measure) with respect to the time we leave home (parameter). In order to get to the office, we have to catch a bus, which comes along at the bus stop at times that show some variance. To determine the derivative of the arrival time of

the office at a certain value of our parameter, the perturbation analysis approach would suggest that we take a number of trial runs in each of which we leave home at the same time, and that we determine for these runs what the difference in arrival time would have been if we would have left a very small time earlier or later. Of course, this difference is zero almost surely, since leaving a split second earlier will only very rarely lead to catching a bus that one would otherwise have missed; in other words, for sufficiently small perturbations the order of events (we arrive at the bus stop; the bus arrives at the bus stop) does not change. So, the answer that is obtained in this way is zero, the expected gradient. Did we prove that it is immaterial at which time we leave home? No, because in this case the expected gradient is not equal to the gradient of the expectation. In fact, there is no way to compute the latter quantity from simulation runs at a constant parameter value.

By the same token, the study of the statistical properties of estimates obtained by the perturbation analysis method is highly nontrivial. The assumption of no change in the order of events will almost surely be violated for any fixed perturbation when the number of samples considered goes to infinity (or when one considers a single sample that is sufficiently long). So the question of consistency has to be approached with great care. Nevertheless, consistency has been proved for a number of cases; see for instance [26].

Sometimes it may be necessary or preferable to take the possibility of event order change explicitly into account. Techniques for doing this have been developed under the name of *finite perturbation analysis* (as opposed to *infinitesimal* perturbation analysis, as discussed above). Propagation rules can be formulated for the zero-order case (no event order change), first-order case (interchange of two adjacent events), and so on. This work was started in [12], where it was shown by simulation that higher-order analysis leads to good estimates for a wider range of perturbations than zero-order analysis does. The approach of [12] ignores the state change associated with an event order change. A recent refinement, presented under the name *extended perturbation analysis* in [14], does take this state change into account by patching together pieces of the nominal trajectory that have the right starting states - an approach which is valid by the Markov property. The improved accuracy goes at the cost of extended simulation time, and so lessens the advantage of perturbation analysis methods over brute-force simulation.

5. A LINEAR-SYSTEM-THEORETIC VIEW

We have mentioned production lines (or flexible manufacturing systems, if you like) as one example of discrete-event dynamical systems. The operation of such production lines is often essentially deterministic in between machine breakdowns. Compared to other discrete-event systems such as communication networks, where it may be quite reasonable to assume an arrival rate with a definite (say, Poissonian) distribution, this leads to a situation which does not fit the queueing network model very well. A method to study the deterministic behaviour of discrete-event systems was proposed by Cohen et al. in [5]. The approach is based on a matrix representation of processing times in a network. A

surprising feature of this model is that it closely resembles the state equations for a linear system with continuous variables, be it that the underlying algebra is different.

How this 'state representation' of discrete-event systems is obtained can best be illustrated by an example. Consider the 'event graph' of Figure 1 below. In the production line interpretation of this picture, the bars marked x_i represent machines (processing units). The circles denote buffers; the numbers associated with the buffers are necessary waiting times (incorporating processing time by the preceding machine, transportation time, and such). The dots in the circles represent parts to be processed. The notational convention is such that every arrow going into a bar denotes a part that is needed for the job performed by the bar (or rather, by the machine that it represents), and every arrow going out represents a part that appears as a result of the completion of that job. So, for a machine to begin working on a job, it needs as many dots in the buffers preceding it as there are arrows from those buffers going into the corresponding bar. The effectuation of a job causes a redistribution of dots over the circles in the graph, the amount of dots before and after not necessarily being the same. Finally, there are parts that go in (resources) and parts that go out (completed products). Entrance of parts is marked by the bars denoted u_1 and u_2 in the figure, whereas exit takes place at the bar y.

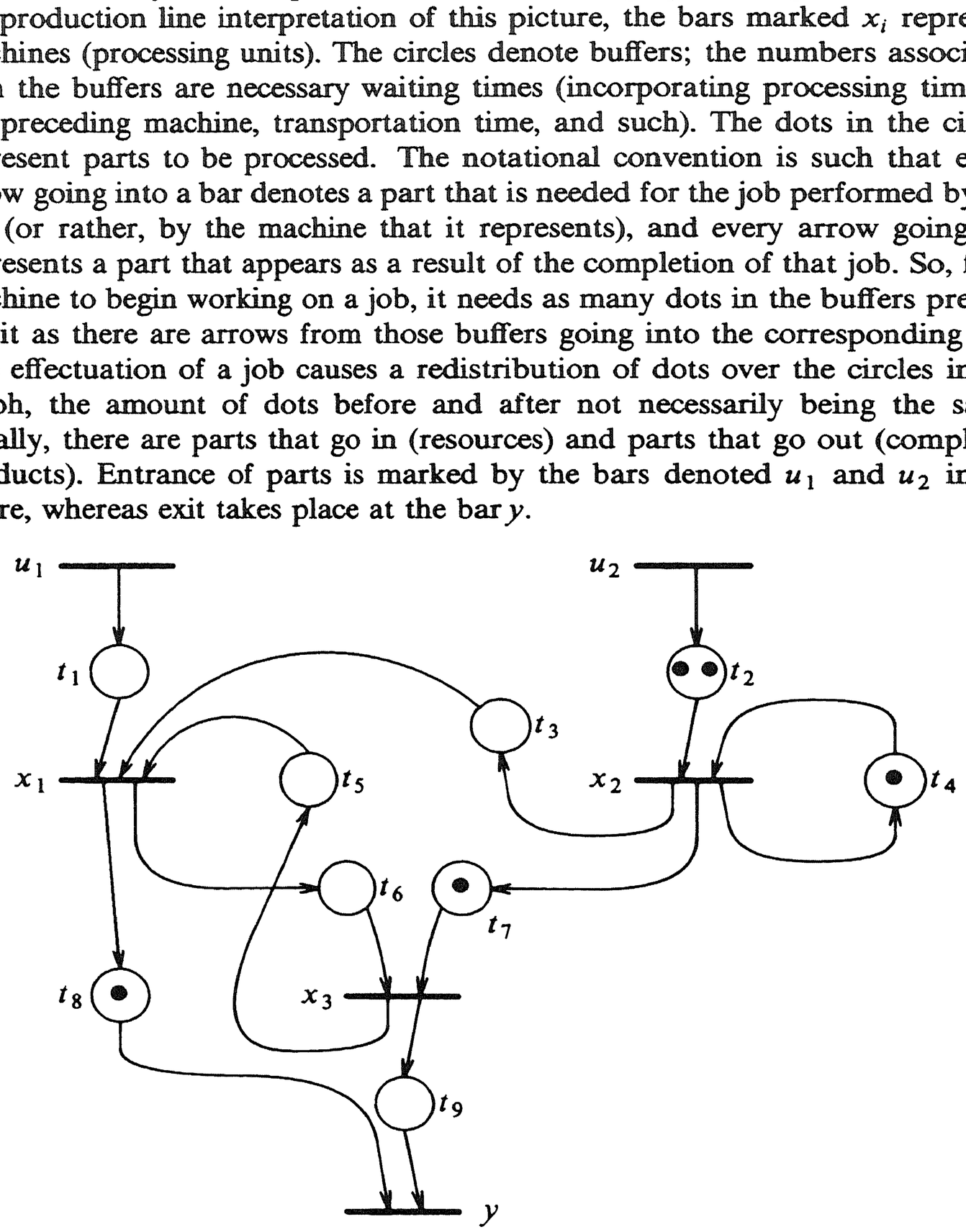

FIGURE 1

Now, let $d_j(c)$ be the time at which machine x_j starts to work on its c-th job. (The letter d is for 'date', c is for 'count'.) Also, let $u_j(c)$ denote the time at which copy number c of resource j becomes available, and let $y(c)$ be the time at which

the c-th completed product becomes available. Assuming that each machine will start processing as soon as all necessary preconditions are satisfied, and assuming that we start in the situation depicted in Figure 1, the following equations can be written down by inspection.

$$d_1(c) = \max(u_1(c)+t_1, d_3(c)+t_5, d_2(c)+t_3) \tag{1}$$

$$d_2(c) = \max(u_2(c-2)+t_2, d_2(c-1)+t_4) \tag{2}$$

$$d_3(c) = \max(d_1(c)+t_6, d_2(c-1)+t_7) \tag{3}$$

$$y(c) = \max(d_1(c-1)+t_8, d_3(c)+t_9). \tag{4}$$

This can be written in matrix form if we agree to interpret matrix multiplication to involve maxima of sums rather than sums of products, and to read vector addition accordingly as maximization:

$$\begin{aligned}\begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}(c) = {} & \begin{bmatrix} -\infty & t_3 & t_5 \\ -\infty & -\infty & -\infty \\ t_6 & -\infty & -\infty \end{bmatrix}\begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}(c) + \\ & + \begin{bmatrix} -\infty & -\infty & -\infty \\ -\infty & t_4 & -\infty \\ -\infty & t_7 & -\infty \end{bmatrix}\begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}(c-1) + \\ & + \begin{bmatrix} t_1 & -\infty \\ -\infty & -\infty \\ -\infty & -\infty \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}(c) + \begin{bmatrix} -\infty & -\infty \\ -\infty & t_2 \\ -\infty & -\infty \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}(c-2)\end{aligned} \tag{5}$$

$$\begin{aligned} y(c) = {} & (-\infty \quad -\infty \quad t_9)\begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}(c) + \\ & + (t_8 \quad -\infty \quad -\infty)\begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}(c-1). \end{aligned} \tag{6}$$

In general, an event graph such as the one in Figure 1 may be given a matrix representation of the form

$$\begin{aligned} d(c) = {} & A_0 d(c) + A_1 d(c-1) + \cdots + A_k d(c-k) + \\ & + B_0 u(c) + \cdots + B_k u(c-k) \end{aligned} \tag{7}$$

$$\begin{aligned} y(c) = {} & C_0 d(c) + \cdots + C_k d(c-k) + \\ & + D_0 u(c) + \cdots + D_k u(c-k). \end{aligned} \tag{8}$$

This description is just an alternative representation for the event graph that has been given in pictorial form in Figure 1; the relation between the two representations is quite direct and comparable to the connection between a graph and its

incidence matrix. The above equations look very much like systems of difference equations like one finds, for instance, in ARMA models. But, of course, the resemblance is stimulated somewhat artificially by the re-interpretation of the arithmetic operations. Essentially, the real field $\mathbb{R}$ with the usual operations of addition and multiplication has been replaced by the 'max algebra' $\mathbb{R} \cup \{-\infty\}$ with the operations of addition and maximization. The max algebra does not have such nice properties as the real field; for instance, the equation $a + x = b$ in the max algebra (corresponding to $\max(a,x) = b$ in ordinary notation) does not always have a solution. In other words, we cannot in general 'subtract' in the max algebra. Nevertheless, its properties of associativity, distributivity and commutativity do make the max algebra an example of an algebraic structure called *dioid* [7]. So, the description of the event graph in matrix terms above may be seen as a linear system over a dioid. Given the amount of knowledge that is available for linear systems over a field, this opens a rather wide perspective of possible research directions.

The first basic steps are taken in [5]; in particular, the analog is studied of the closed (autonomous) linear system $\dot{x}(t) = Ax(t)$. For this equation, one knows that, if there is a simple real eigenvalue λ of A such that all other eigenvalues have real parts smaller than λ, then the long-term behaviour of the solution is dictated by the eigenvector belonging to λ. It turns out that the analogous equation $x(k+1) = Ax(k)$, where addition and multiplication are now interpreted in the max-algebra, shows a similar and perhaps even simpler behaviour. If the weighted graph corresponding to the matrix A is irreducible (meaning that every node can be reached from every other node), then one can show that A has exactly *one* eigenvalue (defined in the obvious way, as a number λ such that $Ax = \lambda x$ for some nontrivial x). Moreover, if the associated eigenvector is unique, then the evolution of the closed system becomes periodical after a finite number of steps, with the operating mode being given by the eigenvector. In fact, the eigenvalue of A is the maximum of the average weights of the circuits in the graph associated with it, and the eigenvectors can be found from the circuits with the largest average weights (critical circuits).

The max algebra has been used before for graph-related problems, notably by R. A. Cuninghame Green [8]. However, the connection with *dynamic* problems as explored in [5] seems to be new. The theory of linear dynamic systems is quite a bit more extensive than the theory of linear equations, and if one is justified in likening the theory of Cuninghame Green to the latter and that of Cohen et al. to the first, then a considerable development of this approach is still to be expected. The 'event graphs' discussed above are also well known in computer science; they form a subcategory of the class of *Petri nets* [20] which find use in the study of concurrency. The graph shown above is an example of a *timed* Petri net; this is one of the many species of Petri nets that have been developed. The real-time aspect is often left out of consideration when basic properties such as liveness and safety are being studied.

The approach in [5] enables one to compute cycle times for closed networks. (In automated factories, production 'lines' are often more like closed curves, if one considers the pallets on which the parts move around as the basic entities.) It

also allows one to find the dominant cyclic regime, so that it is possible to identify the bottleneck machine or machines. The same result could also be obtained by brute-force simulation, of course, but the computational effort involved in this might be large if there are close-to-critical circuits. An important point of interest in more recent research following up on the work in [5] has been the representation of the linear system equations associated with a timed event graph in *state space form*. The theory associated with this problem requires a generalization of the Cayley-Hamilton theorem to matrices over dioids; one can imagine that this is non-trivial, since the usual definition of the determinant involves subtraction, and this operation is not available in dioids. In fact, several different solutions have been proposed [18, 19].

6. Supervisory control

In control theory, a crucial notion is the distinction between *plant* and *controller*. The term 'plant' is used as a generic phrase referring to whatever process one wants to control. The plant represents the data in a control problem; the controller represents the solution. The problem of *controller synthesis* is not so much to design a controller for some specific plant, but rather to find an algorithm which will, for every plant in a given class, produce a controller which will meet specified design goals when applied to that plant. Of course, different plants will in general require different controllers to achieve the same design goals, and so the synthesis algorithm is nothing else but a map from the given set of plants to a given set of controllers, assigning to every plant a suitable controller.

This point of view has been very fruitful in control theory. To formulate a synthesis problem, one has to specify:

- ☐ a set of plants;
- ☐ a set of controllers;
- ☐ the way interaction takes place between plant and controller;
- ☐ a set of design goals.

For example, the set of plants may be the class of all constant-parameter linear systems with a strictly proper rational transfer matrix, assumed to be given in state space form:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{9}$$

$$y(t) = Cx(t). \tag{10}$$

Here, A, B and C are real matrices of appropriate dimensions. Controllers may be taken from the same class:

$$\dot{z}(t) = Nx(t) + Lr(t) \tag{11}$$

$$v(t) = Mz(t). \tag{12}$$

The interaction between plant and controller can now be specified as the one that arises from a *feedback connection*:

$$r(t) = y(t) \tag{13}$$

$$u(t) = v(t). \tag{14}$$

After 'closing the loop', one obtains a closed dynamical system:

$$\frac{d}{dt}\begin{bmatrix} x \\ z \end{bmatrix}(t) = \begin{bmatrix} A & BL \\ MC & N \end{bmatrix}\begin{bmatrix} x \\ z \end{bmatrix}(t). \tag{15}$$

The objective could be to design the controller in such a way that the above system is stable, i. e., all the eigenvalues of the matrix appearing in the above equation are in the left half plane. This is called the problem of *stabilization by dynamic output feedback*. One could rephrase this as the problem of determining matrices L, M and N for given matrices A, B and C such that the composite matrix above is stable, but such a reformulation would miss the point that the *order* of the controller (the length of the vector $z(t)$) is also a design variable. In fact, the problem of finding a characterization in reasonably simple terms of the set of all systems (A, B, C) that can be stabilized by a controller of a given order has resisted all attempts for a solution. Quite in contrast, a design principle called the *observer plus state feedback scheme* solves the problem of stabilization by dynamic output feedback with elementary means.

The idea of controller design is not clearly present either in the perturbation analysis approach to discrete-event systems or in the approach using the max algebra. Still it is clear that there is a place for control in the context of discrete events - for instance, a network protocol can very well be seen as a controller which acts on a 'plant' (the agents in the network) in order to reach specified design goals, such as avoidance of deadlock. Definitions of notions like 'plant', 'controller' and 'plant-controller interaction' in the discrete-event context were proposed by W. M. Wonham and P. J. Ramadge in [25]. They used the formalism of *automata* for this purpose. So, a plant is not given by a set of differential equations such as above, but by a finite or infinite state machine*. An event takes place when there is a transition from one state to another, with the possible transitions being determined by the transition function which is a partial function from $Q \times \Sigma$ to Q (Q the set of states, Σ the set of events). The controller is of the same form, and the crucial point that remains is, how to define the plant-controller interaction.

This is done as follows. The event sets of plant and controller are taken to be the same, so that both machines are driven by the same events. A *control pattern* is, by definition, an assignment of the attributes 'enabled' and 'disabled' to the event set; the transition function of both plant and controller is modified, for each particular control pattern, by allowing only enabled events to take place. Now, the connection between plant and controller is established by a map ϕ (called the *feedback mapping*) which assigns to every controller state a control pattern. The range of ϕ is restricted to a predetermined set of 'feasible' control patterns; this restriction adds considerably, of course, to the nontriviality of the

* Consideration of automata is in itself not new in system theory; see for instance Arbib's contribution in [15].

control problem. The set-up leads to a closed-loop situation: the events that may take place depend on the present state of the plant but (through the mapping ϕ) also on the present state of the controller; the event that actually happens determines the next state of the plant as well as the next state of the controller, which then determine the next events that may take place, and so on.

To illustrate this, let us see how an obvious protocol for two users who want to transmit messages over the same channel can be formulated in this language. We assume that the users can each put the next message they want to send in a single-place buffer, and that messages are to be sent in turns unless the user whose turn it is has an empty buffer. The 'plant' is a four-state machine: the states consist of two bits indicating whether the first or the second user's buffer is empty or full. There are also four events: a_1 and a_2 indicate the arrival of a message from user 1 and user 2 respectively, and s_1 and s_2 indicate that a message from user 1 or user 2 is being sent. The state transition diagram is as in Figure 2 below.

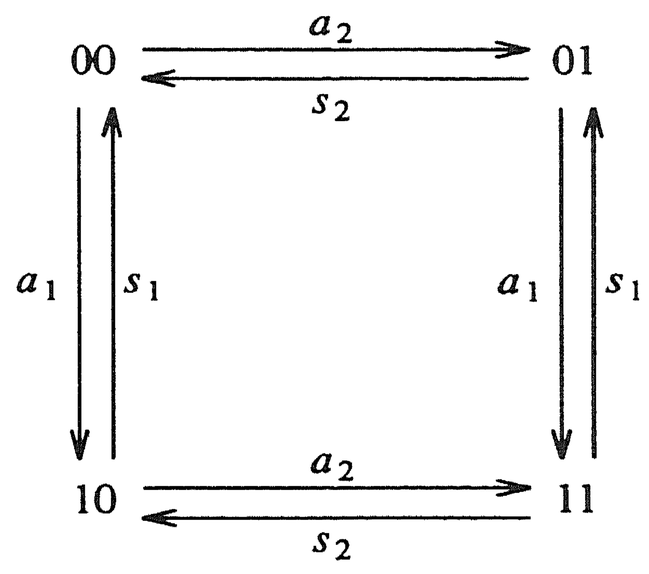

FIGURE 2

There are four feasible control patterns, corresponding to the events s_1 and s_2 being enabled or disabled; the events a_1 and a_2 are always enabled. (In the Wonham-Ramadge terminology, s_1 and s_2 are called *controlled events*, whereas a_1 and a_2 are *uncontrolled events*.) The controller is an eight-state machine; the state consists of three bits of which the first two have the same meaning as in the plant states, and the third indicates whether the previous message sent was from user 1 or from user 2. The state transition diagram is as in Figure 3.

Finally, a mapping ϕ can be defined from the controller states to the feasible control patterns such that s_1 is disabled in states 011 and 111 and enabled otherwise, and such that s_2 is disabled in states 102 and 112, and enabled otherwise. This ensures that the coupled system consisting of plant, controller, and feedback mapping will show the desired behaviour.

The combination of a controller plus a feedback mapping as in the example above is referred to by Wonham and Ramadge as a *supervisor*, and the method of control is denoted as *supervisory control*. The above example was of course simple

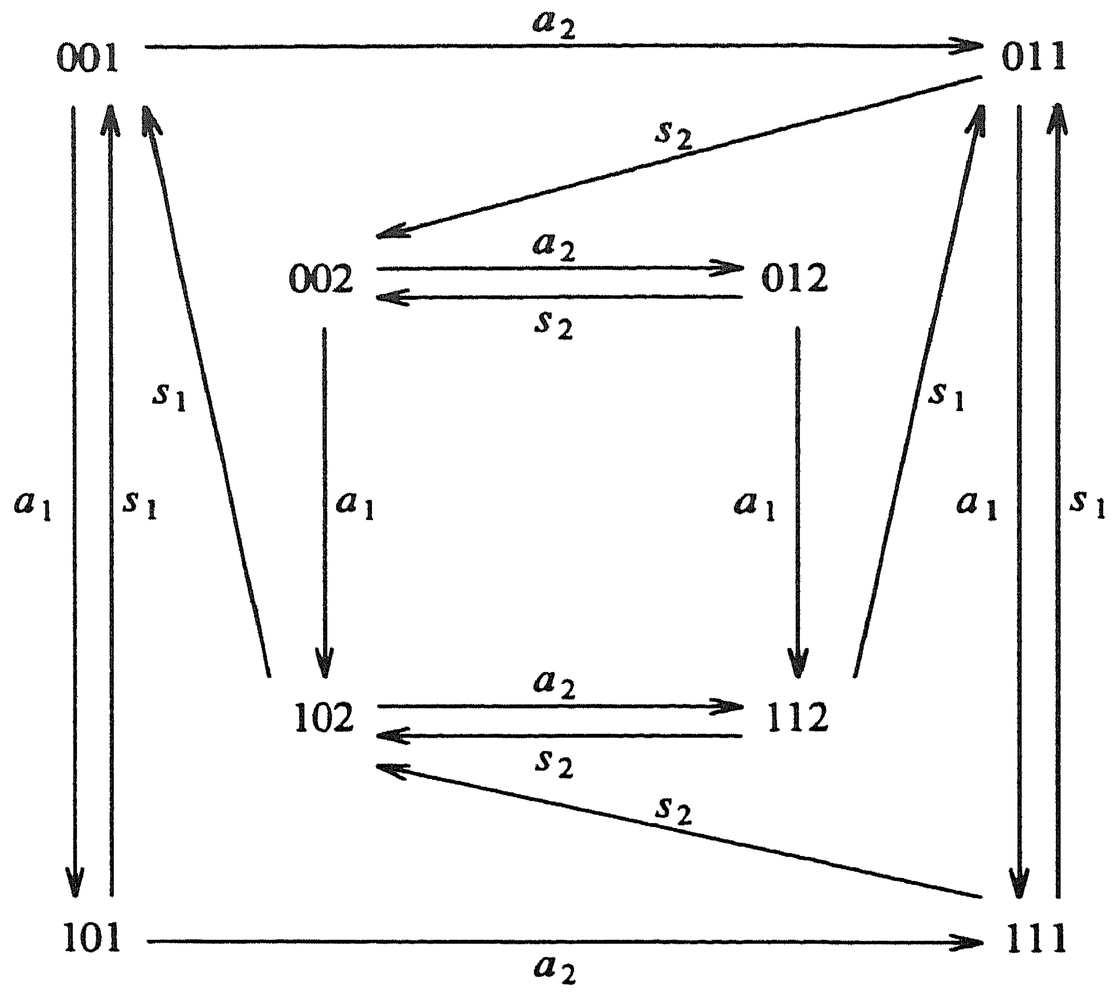

FIGURE 3

enough to allow construction from scratch of a supervisor that meets the specifications. The interesting problem is to give *general* methods for constructing supervisors to fulfill given specifications. In [30], Wonham and Ramadge assume the specifications to be given by an automaton that determines a sublanguage of the language generated by the plant automaton. The purpose of the supervisory control is to ensure that events take place in a proper order, without however imposing restrictions that do not necessarily follow from this demand. In other words, the task is to select, from all languages that are controllable in the sense that they can be enforced by a supervisor using feasible control patterns, the largest one that is contained in the given sublanguage.Wonham and Ramadge show that the solution to this problem (or actually, a somewhat more refined version of it) can be given a fixpoint characterization. Moreover, they give a constructive algorithm to compute the solution in the regular case, when both the plant automaton and the automaton that describes the 'orderly' sequences have a finite number of states.

As is often the case with important theorems, the result just mentioned gives rise to a host of new issues and possible lines of development. For instance, there is the problem of *supervisor complexity*, as measured by the number of states of the supervisor automaton. Sometimes, a supervisor may be reduced in the sense that some of its states may be lumped with no effect on the imposed behaviour. The resulting concept of a 'quotient supervisor' is discussed in [30]. There is also

the issue of *computational complexity*. Although it has been shown in [24] that the construction algorithm of [30] is polynomial in the number of states associated with the plant and with the prescribed behaviour, this is often not the result that one wants. If the 'plant' actually consists of k machines each with n states, then what one gets is that the number of steps in the supervisor construction method is polynomial in n^k, whereas one would like to have methods that are polynomial in n and k. Some recent results on this problem are mentioned in [23].

Yet another point is the assumption, made in [25], that the event sets of plant and supervisor are the same. This means that the supervisor 'sees' every event that takes place. Of course, such an assumption would be unrealistic in many contexts, in particular when the plant is actually a distributed system. A parallel may be drawn with the distinction between 'state feedback' and 'output feedback' in control theory. The recognition of this observability problem calls for a redefinition of the plant-controller interaction, and several proposals have been made to handle this [4, 17, 22]. In [4], the authors introduce a 'mask', which is a mapping from the set of plant events to the set of supervisor events. Plant events that have the same image under this mapping cannot be distinguished by the supervisor; there is also a 'null event' corresponding to plant events that the supervisor doesn't notice at all. It is shown that the classical case of the *alternating bit protocol* can be fitted into this framework. While it is therefore suggested that the set-up of [4] would be suitable for the analysis and synthesis of communication protocols, the discussion in [16] of transaction executions in database systems indicates that the presence of partial information in this context needs a model that is different from the ones proposed in [4, 17, 22]. In [4], an algorithm is given to construct a solution to the minimally restrictive supervisor problem under partial observation, if the solution is required to lie in a specific subclass; an alternative algorithm, which applies under weaker conditions, has been given in [3]. The issue of computational complexity for systems with partial observations is discussed in [28].

7. Final remarks

In the beginning of this paper, the question was asked what contributions can be expected from system theory towards the study of discrete events. To answer this, we have looked at a number of approaches to discrete events that have been published in the system theory literature. Blending in a good deal of personal prejudice, I would like to offer the following as aspects of discrete events on which some additional light could be shed from a system-theoretic perspective.

a. Questions of *representation* and *parametrization*. In system theory, the representation of a system is adapted to the goal one has in mind. For the representation of linear systems, for instance, several 'nice' forms exist which each are particularly suited for expressing certain properties. Algorithms exist to go from one of these forms to another, and such algorithms are very useful to make clear the meaning of various system properties in each of the possible representations. Also, there is the *modeling* problem of going from a given (unstructured) representation to a better one. There is an obvious parallel here

with, for instance, the relation between timed Petri nets and state space representations as discussed in Section 4 of this paper. Another example may be found in the specification of constraints: in the supervisory control setting, the 'legal' behaviour is assumed to be specified through a sublanguage as determined by an automaton, whereas it is often more natural to use temporal logic formulas to describe desired behaviour (as stressed by Wonham himself: see for instance [27]). This can also be interpreted as a representation problem.

b. The notions of *plant* and *controller*. This distinction makes it possible to see controller synthesis as the problem of constructing a mapping between a *class* of plants and a *class* of controllers. In this way, one is automatically led to studying the essential features of plants with regard to a given set of specifications. The issue of verification is also affected in a crucial way. The correctness of a solution to a specific problem will follow from the correctness of the algorithm that has produced it, so that the problem of verification of solutions (communication protocols, for instance) will be solved if we can prove the correctness of the synthesis algorithm.

c. Specific *design principles.* One example of a design principle in system theory is the *internal model principle*, which says that every controller that is able to regulate a plant must contain an 'internal model' of the signals it can cope with. (See [29], p. 210 for a more precise formulation.) The 'quotient structure theorem' of [25] can be interpreted as an attempt to formulate a similar internal model principle in the context of discrete events. The *observer plus state feedback scheme*, which has already been mentioned above, constitutes another design principle which could find possible use, in particular in situations where one has to do with partial observations. Many problems in the design of protocols which must ensure proper functioning of a noisy channel call for a *disturbance decoupling* property: the occurrence of disturbances should not have an effect on the essence of the communication process. A similar 'disturbance decoupling' property has been investigated quite extensively for linear as well as nonlinear systems, and the principles developed here might turn out to be useful also in the context of discrete-event systems.

The approaches we have discussed are quite different in scope. The perturbation analysis approach is quantitative in nature; it concentrates on performance measures that are given by numbers. The 'supervisory control' set-up, on the other hand, is concerned with qualitative features; it guarantees that events will take place in an orderly sequence, but there is no indication how long a particular sequence will take. On this scale from quantitative to qualitative, the approach using state space representations of timed Petri nets is somewhere between the other two. There is no intrinsic reason why there should be just one level of analysis which would provide a suitable platform for discussion, and it may well be that qualitatively oriented approaches will remain co-existent with quantitatively oriented ones.

In a recent editorial in the IEEE Transactions on Automatic Control [9], Y.-C.

Ho warned against a process of formalization and mathematization of models for discrete-event dynamical systems, without regard to *'what the real problems are'*. This, of course, is to be taken seriously. The phrase 'discrete-event systems' actually covers a rather wide variety of problems, stemming from manufacturing, communication networks, database systems, transportation systems, and other sources. Fixation on too specific problems will not help very much to develop the theory, and neither are attempts likely to be successful to carry over concepts from other fields - such as system theory - in a mechanical way. There is room for a variety of models, and for truly creative mathematization. The approaches discussed above foreshadow many interesting developments that are still to take place.

References

1. A. E. Bryson, Y.-C. Ho (1969). *Applied Optimal Control*, Ginn, Waltham, Mass.
2. X. R. Cao (1985). Convergence of parameter sensitivity estimates in a stochastic experiment. *IEEE Trans. Automat. Contr. AC-30*, 834-843.
3. H. Cho, S. I. Marcus (1989). On supremal languages of classes of sublanguages that arise in supervisor synthesis problems with partial observation. *Math. Control Signals Systems 2*, 47-69.
4. R. Cieslak, C. Desclaux, A. S. Fawaz, P. Varaiya (1988). Supervisory control of discrete-event processes with partial observations. *IEEE Trans. Automat. Contr. AC-33*, 249-260.
5. G. Cohen, D. Dubois, J. P. Quadrat, M. Viot (1985). A linear-system-theoretic view of discrete-event processes and its use for performance evaluation in manufacturing. *IEEE Trans. Automat. Contr. AC-30*, 210-220.
6. Y. C. Ho (guest ed.) (1989). *Proc. IEEE 77-1*. (Special issue on dynamics of discrete event systems.)
7. M. Gondran, M. Minoux (1984). *Graphs and Algorithms*, Wiley, Chichester.
8. R. A. Cuninghame Green (1979). *Minimax Algebra*, Lect. Notes Econ. Math. Syst. 166, Springer, New York.
9. Y.-C. Ho (1987). Basic research, manufacturing automation, and putting the cart before the horse. *IEEE Trans. Automat. Contr. AC-32*, 1042-1043.
10. Y.-C. Ho (1988). A selected and annotated bibliography on perturbation analysis. P. Varaiya, A. B. Kurzhanski (eds.). *Discrete Event Systems: Models and Applications* (Proc. IIASA Conf., Sopron, Hungary, August 3-7, 1987), Lect. Notes Contr. Inf. Sci. 103, Springer, Berlin, 217-224.
11. Y.-C. Ho, X. R. Cao (1983). Perturbation analysis and optimization of queueing networks. *J. Opt. Th. Appl. 40*, 559-582.
12. Y.-C. Ho, X. R. Cao, C. G. Cassandras (1983). Infinitesimal and finite perturbation analysis for queueing networks. *Automatica 19*, 439-445.
13. Y.-C. Ho, C. G. Cassandras (1983). A new approach to the analysis of discrete event dynamic systems. *Automatica 19*, 149-167.
14. Y.-C. Ho, S. Li (1988). Extensions of infinitesimal perturbation analysis.

IEEE Trans. Automat. Contr. AC-33, 427-438.
15. R. E. KALMAN, P. L. FALB, M. A. ARBIB (1969). *Topics in Mathematical System Theory*, McGraw-Hill, New York.
16. S. LAFORTUNE (1988). Modeling and analysis of transaction execution in database systems. *IEEE Trans. Automat. Contr. AC-33*, 439-447.
17. F. LIN, W. M. WONHAM (1987). *On observability of discrete-event systems*, Syst. Control Group Rep. 8701, Univ. of Toronto.
18. P. MOLLER (1986). Théorême de Cayley-Hamilton dans les dioïdes et application à l'étude des systèmes á évènements discrets. A. BENSOUSSAN, J. L. LIONS (EDS.). *Analysis and Optimization of Systems* (Proc. 7th Int. Conf., Antibes, June 1986), Lect. Notes Contr. Inform. Sci. 83, Springer, Berlin, 215-226.
19. G. J. OLSDER (1986). On the characteristic equation and minimal realizations for discrete-event dynamic systems. A. BENSOUSSAN, J. L. LIONS (EDS.). *Analysis and Optimization of Systems* (Proc. 7th Int. Conf., Antibes, June 1986), Lect. Notes Contr. Inform. Sci. 83, Springer, Berlin, 189-201.
20. J. L. PETERSON (1981). *Petri Net Theory and the Modeling of Systems*, Prentice Hall, Englewood Cliffs, NJ.
21. L. S. PONTRYAGIN, V. G. BOLTYANSKII, R. V. GAMKRELIDZE, E. F. MISHCHENKO (1962). *The Mathematical Theory of Optimal Processes*, Interscience, New York.
22. P. J. RAMADGE (1986). Observability of discrete event systems. *Proc. 25th IEEE Conf. Dec. Contr.* (Athens, Greece, 1986), IEEE, New York, 1108-1112.
23. P. J. RAMADGE (1988). Supervisory control of discrete event systems: A survey and some new results. P. VARAIYA, A. B. KURZHANSKI (EDS.). *Discrete Event Systems: Models and Applications* (Proc. IIASA Conf., Sopron, Hungary, August 3-7, 1987), Lect. Notes Contr. Inf. Sci. 103, Springer, Berlin, 69-80.
24. P. J. RAMADGE, W. M. WONHAM (1986). Modular supervisory control of discrete event systems. A. BENSOUSSAN, J. L. LIONS (EDS.). *Analysis and Optimization of Systems* (Proc. 7th Int. Conf., Antibes, June 1986), Lect. Notes Contr. Inf. Sci. 83, Springer, Berlin, 202-214.
25. P. J. RAMADGE, W. M. WONHAM (1987). Supervisory control of a class of discrete-event processes. *SIAM J. Contr. Optimiz. 25*, 206-230.
26. R. SURI, M. A. ZAZANIS (1988). Perturbation analysis gives strongly consistent sensitivity estimates for the M/G/1 queue. *Management Science 34*, 39-64.
27. J. G. THISTLE, W. M. WONHAM (1986). Control problems in a temporal logic framework. *Int. J. Contr. 44*, 943-976.
28. J. N. TSITSIKLIS (1989). On the control of discrete-event dynamical systems. *Math. Control Signals Systems 2*, 95-107.
29. W. M. WONHAM (1979). *Linear Multivariable Control: a Geometric Approach* (2nd ed.), Springer, New York.
30. W. M. WONHAM, P. J. RAMADGE (1987). On the supremal controllable sublanguage of a given language. *SIAM J. Contr. Optimiz. 25*, 637-659.