DISCRETE EVENT SYSTEMS

Christos G. Cassandras
Dept. of Manufacturing Engineering, 15 St. Mary's St., Boston University, Brookline, MA 02446, USA

Keywords: Time-driven System, Event-driven System, Language, Automaton, Supervisory Control, Sample Path Analysis, Petri Net, Dioid Algebra, Simulation.

Contents

Summary

Discrete Event Systems (DES) are characterized by the occurrence of discrete events asynchronously over time which are responsible for driving all dynamics. Such systems are ubiquitous in modern technological environments, ranging from communication networks and manufacturing to transportation and logistics. This class of event-driven dynamic systems is first compared to traditional time-driven systems described by differential (or difference) equations. We subsequently overview some of the major modeling frameworks for DES, based on automata, Petri nets, and dioid algebras. The use of these models is illustrated through queuing systems, a common class of DES.

1. Introduction

The term "discrete event system" was introduced in the early 1980s to identify an increasingly important class of dynamic systems in terms of their most critical feature: the fact that their behavior is governed by *discrete events* occurring asynchronously over time and solely responsible for generating state transitions. In between event occurrences, the state of such systems is unaffected. Examples of such behavior abound in technological environments such as computer and communication networks, automated manufacturing systems, air traffic control systems, C^3I (Command, Control, Communication, and Information) systems, advanced monitoring and control systems in automobiles or large buildings, intelligent transportation systems, distributed software systems, and so forth. The operation of such environments is largely

regulated by human-made rules for initiating or terminating activities and scheduling the use of resources through controlled events, such as hitting a keyboard key, turning a piece of equipment "on", or sending a message packet. In addition, there are numerous uncontrolled randomly occurring events, such as a spontaneous equipment failure or a packet loss, which may or may not be observable through sensors.

We shall henceforth use the acronym DES for "Discrete Event System", but must point out that the acronym DEDS, for "Discrete Event Dynamic System", is also commonly used to emphasize that it is the dynamics of such systems that render them particularly interesting.

The conceptual and practical challenges in the development of the DES field may be summarized as follows

1. The types of variables involved in the description of a DES are both continuous and discrete, sometimes purely symbolic, i.e., non-numeric (as in describing the state of a piece of equipment as "on" or "off"). This renders traditional mathematical models based on differential (or difference) equations inadequate, and methods relying on the power of calculus are, consequently, of limited use. New modeling frameworks, analysis techniques, and control procedures are required to suit the structure of a DES.

2. Because of the asynchronous nature of events that cause state transitions in DES, it is neither natural nor efficient to use time as a synchronizing element driving the system dynamics (details on this issue are provided in Section 2). It is for this reason that DES are often referred to as *event-driven*, to contrast them to classical *time-driven* systems based on the laws of physics; in the latter, as time evolves state variables such as position, velocity, temperature, pressure, current, voltage, etc. also continuously evolve. In order to capture event-driven state dynamics, however, new mathematical models are necessary.

3. Uncertainties are inherent in the technological environments where DES are encountered. Therefore, the mathematical models used for DES and all associated methods for analysis and control must incorporate this element of uncertainty, sometimes by explicitly modeling nondeterministic behavior and often through the inclusion of appropriate stochastic model components.

4. Complexity is also inherent in DES of practical interest, usually manifesting itself in the form of combinatorially explosive state spaces. Purely analytical methods for DES design, analysis, and control have proved to be limited. A large part of the progress made in this field has relied on the development of new paradigms characterized by a combination of mathematical techniques and effective processing of experimental data.

The remainder of this chapter is organized as follows. In Section 2, we contrast time-driven and event-driven systems as a means of highlighting the characteristics of DES that motivate the development of the main modeling frameworks and methodologies

for design and control of such systems. In Section 3, we describe the different levels of abstraction used for modeling DES, depending on the objectives of the analysis and the problems to be addressed. This will also help to understand the nature of the subsequent three chapters that focus on the supervisory control framework and on methodologies based on sample path analysis. Finally, in Section 4 we provide an overview of the main modeling approaches for DES. Without burdening the reader with excessive technical details, we compare different models for a class of DES that constitutes a basic building block in many applications.

2. Event-Driven and Time-Driven Systems

As previously mentioned, today's technological and increasingly computer-dependent world includes numerous examples of dynamic systems with the following two features. First, many of the quantities we deal with are "discrete", typically involving counting integer numbers (how many parts are in an inventory, how many planes are in a runway, how many telephone calls are active). Second, what drives many of the processes we use and depend on is a variety of instantaneous "events" such as the pushing of a button, hitting a keyboard key, or a traffic light turning green. In this section, we will explain how these features amount to fundamental differences between traditional dynamic systems modeled through differential (or difference) equations and the new class of DES.

Let us begin with the concept of "event". We do not attempt to formally define it, since it is a primitive concept with a good intuitive basis. We only wish to emphasize that an event should be thought of as occurring instantaneously and causing transitions from one system state value to another. An event may be identified with a specific action taken (e.g., somebody presses a button). It may be viewed as a spontaneous occurrence dictated by nature (e.g., a computer goes down for whatever reason too complicated to figure out) or it may be the result of several conditions which are suddenly all met (e.g., the fluid level in a tank exceeds a given value). For the purpose of developing a model for a DES, we will use the symbol e to denote an event. When considering a system affected by different types of events, we will assume we can define an *event set* E whose elements are all these events. Clearly, E is a discrete set.

Let us next concentrate on the nature of the state space of a system. In continuous-state systems the state generally changes as time changes. This is particularly evident in discrete-time models: the "clock" is what drives a typical sample path. With every "clock tick" the state is expected to change, since *continuous* state variables *continuously* change with time. It is because of this property that we refer to such systems as *time-driven*. In this case, time is a natural independent variable which appears as the argument of all input, state, and output functions involved in modeling a system.

In DES, at least some of the state variables are discrete and their values change only at certain points in time through instantaneous transitions which we associate with "events". What is important is to specify the timing mechanism based on which events take place. Let us assume there exists a clock through which we will measure time, and consider two possibilities: (*i*) At every clock tick an event e is selected from the

event set E (if no event takes place, we can think of a "null event" as being a member of E, whose property is that it causes no state change), and (*ii*) At various time instants (not necessarily known in advance and not necessarily coinciding with clock ticks), some event e "announces" that it is occurring. There is a fundamental difference between (*i*) and (*ii*) above. In (*i*), state transitions are *synchronized* by the clock: there is a clock tick, an event (or no event) is selected, the state changes, and the process repeats. Thus, the clock alone is responsible for any possible state transition. In (*ii*), every event $e \in E$ defines a distinct process through which the time instants when e occurs are determined. State transitions are the result of combining these *asynchronous* concurrent event processes. Moreover, these processes need not be independent of each other. The distinction between (*i*) and (*ii*) gives rise to the terms *time-driven* and *event-driven* systems respectively. Continuous-state systems are, by their nature, time-driven. However, in discrete-state systems this depends on whether state transitions are synchronized by a clock or occur asynchronously as in scheme (*ii*) above. Clearly, event-driven systems are more complicated to model and analyze, since there are several asynchronous event-timing mechanisms to be specified as part of our understanding of the system.

In view of this discussion, let us now turn our attention to mathematical models one can use for time-driven and event-driven systems. In the former case, the field of systems and control has based much of its success on the use of well-known differential-equation-based models, such as

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \quad \mathbf{x}(t_0) = \mathbf{x}_0, \tag{1}$$

$$\mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), t), \tag{2}$$

where (1) is a (vector) state equation with initial conditions specified, and (2) is a (vector) output equation. As is common in system theory, $\mathbf{x}(t)$ denotes the state of the system, $\mathbf{y}(t)$ is the output, and $\mathbf{u}(t)$ represents the input, often associated with controllable variables used to manipulate the state so as to attain a desired output. In order to use this type of model, the two key properties that the system must satisfy are: (*i*) it has a continuous state space, and (*ii*) the state transition mechanism is time-driven. The first property allows us to define the state by means of continuous variables, which can take on any real (or complex) value. It is because of this reason that this class of systems is often referred to as *Continuous Variable Dynamic Systems* (CVDS). Common physical quantities such as position, velocity, acceleration, temperature, pressure, flow, etc. fall in this category. Since we can naturally define time derivatives for these continuous variables, differential equation models like (1) can be used.

The second property points to the fact that the state generally changes as time changes. As a result, the time variable t (or some integer $k = 0,1,2,\ldots$ in discrete time) is a natural independent variable for modeling such systems.

In contrast to a CVDS, in a DES the state space is discrete (or at least includes several discrete variables) and the state transition mechanism is event-driven. From a

modeling point of view, the latter property has the following implication. If we can identify a set of "events" any one of which can cause a state transition, then time no longer serves the purpose of driving such a system and may no longer be an appropriate independent variable.

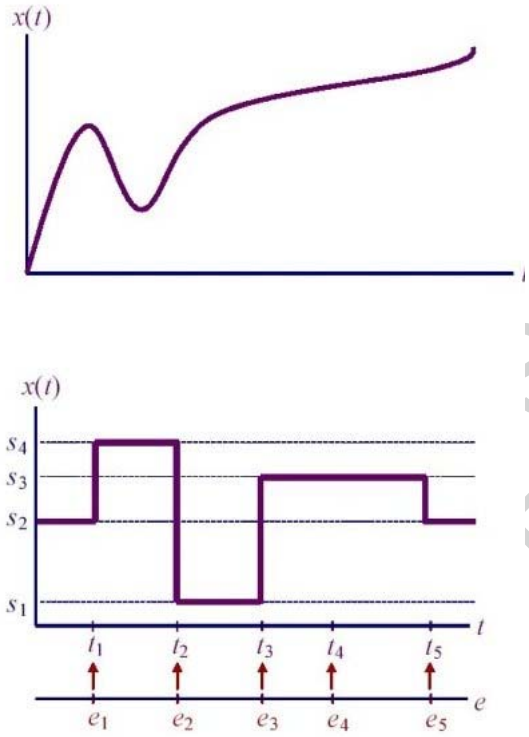

Figure 1: Comparison of time-driven and event-driven sample paths

These two characteristics are the ones used in defining a DES as a discrete-state, event-driven system, that is, we view a DES as one whose state evolution depends entirely on the occurrence of asynchronous discrete events over time. In fact, comparing state trajectories (sample paths) of CVDS and DES is useful in understanding the differences between the two and setting the stage for DES modeling frameworks. Thus, comparing typical sample paths from each of these system classes, as in Fig. 1, we observe the following:

- For the time-driven CVDS shown, the state space X is the set of real numbers $\mathbb{R}$, and $x(t)$ can take any value from this set. The function $x(t)$ is the solution of a differential equation of the general form $\dot{x}(t) = f(x(t), u(t), t)$, where $u(t)$ is the input.

- For the event-driven DES, the state space is some discrete set $X = \{s_1, s_2, s_3, s_4\}$. The sample path can only jump from one state to another whenever an event occurs. Note that an event may take place, but not cause a state transition, as in the case of e_4. There is no immediately obvious analog to $\dot{x}(t) = f(x(t), u(t), t)$, i.e., no mechanism to specify how events might interact over time or how their time of occurrence might be determined. Thus,

a large part of the early developments in the DES field has been devoted to the specification of an appropriate mathematical model containing the same expressive power as (1)-(2).

We should point out that discrete *event* systems should not be confused with discrete *time* systems. The class of discrete time systems contains both CVDS and DES. In other words, a DES may be modeled in continuous or in discrete time, just like a CVDS can.

3. Abstraction Levels in the Study of Discrete Event Systems

Let us return to the DES sample path shown in Fig.1. Instead of plotting the piecewise constant function $x(t)$ as shown, it is often convenient to simply write the timed sequence of events

$$(e_1,t_1),(e_2,t_2),(e_3,t_3),(e_4,t_4),(e_5,t_5), \qquad (3)$$

which contains the same information as the sample path depicted in Fig. 1. The first event is e_1 and it occurs at time $t=t_1$; the second event is e_2 and it occurs at time $t=t_2$, and so forth. When this notation is used, it is implicitly assumed that the initial state of the system, s_2 in this case, is known and that the system is "deterministic" in the sense that the next state after the occurrence of an event is unique. Thus, from the sequence of events in (3), we can recover the state of the system at any point in time and reconstruct the DES sample path in Fig. 1.

Consider the set of all possible timed sequences of events that a given system can ever execute. We call this set the *timed language* model of the system. The word "language" comes from the fact that we can think of the event E as an "alphabet" and of (finite) sequences of events as "words". We can further refine our model of the system if some statistical information is available about the set of sample paths of the system. Let us assume that probability distribution functions are available about the "lifetime" of each event type $e \in E$, that is, the elapsed time between successive occurrences of this particular e. We call a *stochastic timed language* a timed language together with associated probability distribution functions for the events. The stochastic timed language is then a model of the system that lists all possible sample paths together with relevant statistical information about them.

Stochastic timed language modeling is the most detailed in the sense that it contains event information in the form of event occurrences and their orderings, information about the exact times at which the events occur (and not only their relative ordering), and statistical information about successive occurrences of events. If we omit the statistical information, then the corresponding timed language enumerates all the possible sample paths of the DES, with timing information. Finally, if we delete the timing information from a timed language we obtain an *untimed language*, or simply *language*, which is the set of all possible orderings of events that could happen in the given system. Deleting the timing information from a timed language means deleting the time of occurrence of each event in each timed sequence in the timed language.

For example, the untimed sequence corresponding to the timed sequence of events in (3) is

$\{e_1, e_2, e_3, e_4, e_5\}$.

Languages, timed languages, and stochastic timed languages represent different levels of abstraction at which DES are modeled and studied: untimed (or logical), timed, and stochastic. The choice of the appropriate level of abstraction clearly depends on the objectives of the analysis. In many instances, we are interested in the "logical behavior" of the system, that is, in ensuring that a precise *ordering of events* takes place which satisfies a given set of specifications (e.g., first-come first-served in a job processing system). Or we may be interested in finding if a particular state (or set of states) of the system can be reached or not. In this context, the actual timing of events is not required, and it is sufficient to model only the untimed behavior of the system, that is, consider the language model of the system. *Supervisory control* is the term established for describing the systematic means (i.e., enabling or disabling events which are controllable) by which the logical behavior of a DES is regulated to achieve a given specification. This topic is further discussed in another chapter (see *Supervisory Control of Discrete Event Systems*)

Next, we may become interested in *event timing* in order to answer questions such as: "How much time does the system spend at a particular state?" or "How soon can a particular state be reached?" or "Can this sequence of events be completed by a particular deadline?" These and related questions are often crucial parts of the design specifications. More generally, event timing is important in assessing the performance of a DES often measured through quantities such as *throughput* or *response time*. In these instances, we need to consider the timed language model of the system. The fact that different event processes are concurrent and often interdependent in complex ways presents great challenges both for modeling and analysis of timed DES. Moreover, since we cannot ignore the fact that DES frequently operate in a stochastic setting (e.g., the time when some equipment fails is unpredictable), an additional level of complexity is introduced, necessitating the development of probabilistic models and related analytical methodologies for design and performance analysis based on stochastic timed language models. *Sample path analysis* refers to the study of sample paths of DES, focusing on the extraction of information for the purpose of efficiently estimating performance sensitivities of the system and, ultimately, achieving on-line control and optimization (see *"Sample Path Analysis of Discrete Event Dynamic Systems"* for more details).

These different levels of abstraction are complementary, as they address different issues about the behavior of a DES. Indeed, the literature in DES is quite broad and varied as extensive research has been done on modeling, analysis, control, optimization, and simulation at all levels. Although the language-based approach to discrete event modeling is attractive in presenting modeling issues and discussing system-theoretic properties of DES, it is by itself not convenient to address verification, controller synthesis, or performance issues; what is also needed is a convenient way of *representing* languages, timed languages, and stochastic timed languages. If a language (or timed language or stochastic timed language) is finite, we

could always list all its elements, that is, all the possible sample paths that the system can execute. Unfortunately, this is rarely practical. Preferably, we would like to use *discrete event modeling formalisms* that would allow us to represent languages in a manner that highlights structural information about the system behavior and that is convenient to manipulate when addressing analysis and controller synthesis issues. Discrete event modeling formalisms can be untimed, timed, or stochastic, according to the level of abstraction of interest. In the next section, we provide a brief introduction to one of the major modeling formalisms, based on *automata*, which also forms the foundation for supervisory control and sample path analysis (further discussed in *"Supervisory Control of Discrete Event Systems"* and *"Sample Path Analysis of Discrete Event Dynamic Systems"*). Some more details on the manipulation of automata for the construction of system models from component models and comparisons with other modeling approaches are given in "*Modeling of Discrete Event Systems"*. We will use automata to illustrate the construction of models for a common class of DES and contrast it to two other modeling frameworks.

4. Modeling Overview

The introduction to DES in the previous sections has served to point out the main characteristics of these systems. Two elements which have emerged as essential in defining a DES are: a discrete state space, which we denote by X , and a discrete event set, which we denote by E . We can now build on this basic understanding in order to develop some formal models for DES.

-
-
-

Bibliography

Baccelli, F., Cohen, G., Olsder, G.J., Quadrat, J.-P. (1992). *Synchronization and Linearity: An Algebra for Discrete Event Systems*, 514pp., New York, NY, Wiley, Chichester,. [Good reference for the dioid algebra approach to discrete event systems]

Cassandras, C.G., and Lafortune, S. (1999). *Introduction to Discrete Event Systems*, 822pp., Boston, MA Kluwer Academic Publishers. [Textbook with a comprehensive coverage of all aspects of discrete event systems, including modeling frameworks, supervisory control, discrete event simulation, and sample paths analysis methods]

Glasserman, P., Yao, D.D. (1991). *Monotone Structure in Discrete-Event Systems*, 297pp., New York, NY, Wiley. [This book emphasizes structural properties of discrete event systems that lend themselves to methods for performance analysis, control, and optimization]

Ho, Y.C. (Ed.) (1991). *Discrete Event Dynamic Systems: Analyzing Complexity and Performance in the Modern World*, 291pp. New York, NY, IEEE Press. [A collection of papers reflecting different approaches to the modeling and analysis of discrete event systems, summarizing the first decade of research accomplishments in the field]

Ho, Y.C., Cao, X. (1991). *Perturbation Analysis of Discrete Event Dynamic Systems*,, 437pp., Boston, MA, Kluwer Academic Publishers. [This book covers the most important class of sample-path-based methods for the performance analysis, control, and optimization of discrete event systems]

Peterson, J.L. (1981). *Petri Net Theory and the Modeling of systems*, Englewood Cliffs, NJ, Prentice Hall. [Good textbook reference of Petri nets]

Biographical Sketch

Christos G. Cassandras is Professor of Manufacturing Engineering and Professor of Electrical and Computer Engineering at Boston University. He received a B.S. from Yale University, M.S.E.E. from Stanford University, and S.M. and Ph.D. degrees from Harvard University. In 1982-84 he was with ITP Boston, Inc. where he worked on the design of automated manufacturing systems. In 1984-1996 he was a faculty member at the Department of Electrical and Computer Engineering, University of Massachusetts/Amherst. He specializes in the areas of discrete event and hybrid systems, stochastic optimization, and computer simulation, with applications to computer networks, manufacturing systems, and transportation systems. He has published over 200 refereed papers in these areas, and two textbooks. He has guest-edited several technical journal issues, serves on several editorial boards, and he is currently Editor- in-Chief of the IEEE Transactions on Automatic Control. He has received several awards, including the 1999 Harold Chestnut Prize (IFAC Best Control Engineering Textbook) for Discrete Event Systems: Modeling and Performance Analysis and a 1991 Lilly Fellowship, and was elected Fellow of the IEEE in 1996.