*Proceedings of the 1991 Winter Simulation Conference*
*Barry L. Nelson, W. David Kelton, Gordon M. Clark (eds.)*

# TOWARDS A LOGICAL FOUNDATION OF DISCRETE EVENT MODELING AND SIMULATION

Ashvin Radiya

Computer Science Department
Wichita State University
Wichita, KS 67220

## ABSTRACT

This paper presents the first step in developing a logical foundation of Discrete Event Modeling and Simulation (DEMS) by defining some of the fundamental terms and concepts of DEMS from a logical view point. Two fundamental notions of events and states are defined in terms two types of propositions. The notion of "behavior of a system/model over a time interval is formalized by defining the concept of discrete scenario. The modeling capabilities of a logical language are discussed using two simple examples. The logical language enhances modeling capabilities because it provides new pregram connectives and generalizes the program connectives of many DEMS languages.

## 1 INTRODUCTION

DEMS has been widely applied to reason about and understand complex systems such as computer systems, communication networks, and manufacturing systems. The methodology of reasoning about a system in DEMS requires (i) a suitable computer programming language, called a DEMS language, for expressing (discrete event) models (ii) an appropriate model of the system which is constructed by viewing the system in a particular way, and (iii) a simulation procedure to simulate the model expressed in the DEMS language. The particular way in which systems must be viewed plays a central role in determining the form of a DEMS language and its simulation procedure.

In this paper the way in which systems must be viewed in order to apply the methodology of DEMS is formalized from a logical view point. The term "logical" refers to the general approach and perspective of logicians (Dowty, Wall, and Peters 1981). In Radiya (1990) a logical language $L_{PDE}$ and its simulation procedure are formally defined based upon the concepts and terms discussed here. $L_{PDE}$ generalizes some of the ways in which events and state-changes are related in many DEMS languages based on traditional world views of event scheduling, activity scanning, and process interaction (Zeigler 1976). This is illustrated here by discussing two simple examples (see Radiya (1990) for more examples).

There are several advantages of developing a logical foundation of DEMS. First, the fundamental concepts and terms of DEMS are formally and logically defined. Second, the logical foundation has led to the logical DEMS language $L_{PDE}$ which allows program connectives of many DEMS languages in a more generalized form and it contains new program connectives. Third, proof systems can be developed for $L_{PDE}$ for proving properties of models such as equivalence of two models or events occurring in a model satisfy certain relationships. Fourth, by combining the relevant research work on causality in Artificial Intelligence (AI) with the work presented in this paper and Radiya (1990), both DEMS and AI will mutually benefit. Note that most of the work in AI, unlike in DEMS, is couched in the framework of logics.

## 2 FUNDAMENTAL CONCEPTS

For constructing a model, a system must be viewed in terms of events and states such that the behavior of the system over an (time) interval can be represented by a piecewise constant state trajectory (Figure 1). Note that events as defined in this paper do not occur at every (time) instant at which the state changes. In the following the notions of event and state are formalized. To formalize these notions, two types of propositions are defined. A *proposition* is, roughly, what a sentence asserts. A proposition is an abstract entity and cannot be perceived (Dowty 1981). A truth value t(rue) or f(alse) can be associated with a proposition.

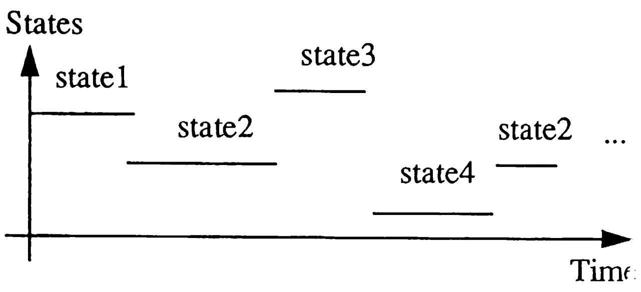

Fig. 1. A typical state trajectory of a system

### 2.1 Events and Instantaneous Propositions

The term "event" has several different meanings. For example, Franta (1977) considers events to be instantaneous state-changes, whereas Reiter (1971) considers events to be particular instants during a simulation when something happens or should have happened. This multiplicity of definitions arises from the lack of distinction between an event and an occurrence of an event. The difference between an event and its occurrence becomes evident from the typical examples of events such as customer_arrival, service_begin, and cpu_begins_a_job. In these events neither state-changes nor instants are associated with events per se. However, an instant must be associated with an event occurrence. State-changes may or may not be associated with event occurrences.

The most fundamental characteristic of an event is that, at any instant, it is meaningful to ask whether the event has occurred. Another property of events is that they can occur only at finitely many instants over any bounded interval. Thus, an event can be considered to be a conceptual entity called instantaneous proposition as defined below.

*DEF* : An *instantaneous proposition* is a proposition that is true only at finitely many instants in any bounded interval.

To say that an event has (not) occurred at an instant is the same as asserting that the corresponding instantaneous proposition is true (false) at that instant.

### 2.2 States and Static Propositions

State is usually represented by a finite set of (*variable*, *value*) pairs. A (*var*, *val*) pair can be thought of as a proposition asserting the fact that an entity represented by the variable *var* has the value *val*. Such propositions, called static propositions, can take the truth value t at infinitely many instants, unlike instantaneous propositions, in a bounded interval.

*DEF* : A proposition p is called a *static proposition* if the truth values of the proposition p over any bounded interval changes only finitely many time.

Some typical examples of static propositions are length_of_queue_is_1, machine_is_free, and machine_is_busy. A state is defined by associating the truth value t to finitely many static propositions out of all the static propositions under consideration. A state can be represented by a finite set consisting of true static propositions. Hence, difference between any two states, called *state-changes*, can also be represented by a finite set.

## 3 MODELING

A model is a syntactic entity, an expression, of a DEMS language such that state trajectories similar to the one in Figure 1 can be generated from the model using a simulation procedure. Different initial conditions may lead to different state trajectories. In $L_{PDE}$, a model is a finite set of formulas and it relates the truths of two types of propositions at different instants. The underlying assumptions of DEMS are such that over any bounded interval of time there are only finitely many instants at which the truths of a proposition can change. Thus, even though the time is represented by the set of positive real numbers R+ , the behavior of a system/model over a bounded interval can be represented by finite means and there are only finitely many instants that are of interest in that interval. This can be formally stated by the concept of discrete scenario. Let $\dot{P}$ and $\bar{P}$ be the sets of instantaneous and static propositions.

*DEF:* A discrete scenario over an interval $I \subseteq$ R+ is a function of the type $(\dot{P} \cup \bar{P}) \mathrm{x} I \rightarrow \{\mathrm{t}, \mathrm{f}\}$ such that over any bounded subinterval of $I$, the truths of propositions in $\dot{P}$ and $\bar{P}$ change their truth values only finitely many times.

A propositional discrete event logic $L_{PDE}$ which allows the truths of propositions to be related at different instants is defined in [Radiya 1990]. In $L_{PDE}$, quantifiers connect the truths of propositions at different instants. In the terminology of programming languages, quantifiers correspond to program connectives. $L_{PDE}$ contains infinitely many program connectives including **next**, **if**, **when**, **whenever**, **until**, **while**, and **at**. These connectives are used in a limited way in the DEMS languages based on traditional world views. For example, event scheduling world view utilizes formulas of the following kind which characterizes event routines — { ( **whenever** $\dot{P}$) ... }. This formula refers to all the instants at which the instantaneous proposition $\dot{P}$ is true, i.e., all the instants at which the event occurs. The event routine of the above form changes the truth values of static propositions at the referred instants; it may also assert truths of some instantaneous propositions in the future. $L_{PDE}$ allows a more generalized form of formula — { ( **whenever** *f*) ... }, where *f* is obtained by combining instantaneous propositions, static propositions, and logical operators **and** and **not**.

## 4 EXAMPLES

The modeling capability of $L_{PDE}$ is illustrated by considering two simple examples. In the first example, the following new policy is to be added to the existing model of a bank. Every time the manager steps out of the bank before 11:00 am, the first customer to arrive in the bank while the manager is stepping out or after the manager has stepped out receives a gift of $2 in the customer's account. A customer can get a gift of at most $2. Assume that at most one customer arrives at an instant. If the model was represented in $L_{PDE}$ then the policy can be added by adding the following formula in the existing model. {((**whenever** manager_steps_out & t* < 11am)) {((**when** customer_arrives)) {[customer_gets_a_gift_of_$_2]} } }. In this formula, quantifiers/program connectives are **whenever** and **when**. manager_steps_out and customer_arrives are names of events or instantaneous propositions. customer_gets_a_gift_of_$_2 denotes a transition (state-changes) corresponding to a customer getting a gift of $2.

Second example consists of adding yet another policy in the model of the first example. The new additional policy is that every time a customer arrives when the manager is out of the bank, the customer gets a flower. This policy can be incorporated in a model by adding the following formula to the model of the above example — {((**whenever** manager_steps_out)) {((**until** manager_comes_back)) {((**if** customer_arrives)) **then** {[customer_gets_a_flower]} } } }. Note that this example utilizes the program connective until which is not available in the languages based on traditional world views. Radiya (1990) shows that extensions illustrated by the above examples are difficult to achieve in the languages based on event scheduling world view.

## 5 SUMMARY

Two types of propositions — instantaneous and static — are defined to develop a logical foundation of DEMS. The concept of discrete scenario is defined which formalizes the notion of the "behavior of a system/model" over an interval. Two simple examples are presented to illustrate the modeling capabilities of the logic $L_{PDE}$. $L_{PDE}$ enhances modeling capabilities because it provides infinitely many program connectives which allow events and state-changes to be related in a generalized and logical way.

## REFERENCES

Dowty, D. R., Wall, R. E., and Peters, S. *Introduction to Montague Semantics.* Dordrecgt, Holland, D. Reidel Pub. Co., 1981.

Franta, W. R. *The Process View of Simulation.* Elsevier-North Holland, New York, 1977.

Radiya, A. A Logical Approach to Discrete Event Modeling and Simulation. *PhD Dissertation.* School of Computer and Information Science, Syracuse University, 1990.

Rivett, P. *Principles of Model Building.* New York: John Wiley, 1972.

Zeigler, B. P. *Theory of Modelling and Simulation.* John Wiley & Sons, 1976.

## AUTHOR BIOGRAPHY

**ASHVIN RADIYA** is an assistant professor in the Computer Science Department at The Wichita State University. His research interests are discrete event modeling and simulation, logical foundation of procedural programming languages, visual programming, distributed computer graphics and animation, and machine learning.