Proceedings of the 1996 Winter Simulation Conference
ed. J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain

VISUALIZING GENERALIZED SEMI-MARKOV PROCESSES

Lee W. Schruben
Eric L. Savage

School of Operations Research and Industrial Engineering
Cornell University
Ithaca, New York 14853, U.S.A.

ABSTRACT

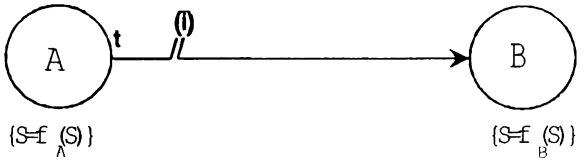
Generalized Semi-Markov Processes (GSMPs) are usually described by sets of variables, events and clock distributions. This kind of representation often lacks intuitive appeal. In this paper we propose a mapping from GSMPs to Event Graph Models. This mapping allows us to use an event graph to visualize a GSMP model as an intermediate step to implementation. By examining the event graph model, we can perform logic checking and verification more easily than if we try to interpret the GSMP description.

1 THE TWO MODELING PARADIGMS

Before presenting the mapping algorithm, we first introduce the two modeling schemes. S will denote a countable set of "physical" states and A equal a finite set of integers enumerating the events. Generic states are s and s'; a generic event is α.

1.1 Event Graph Models

Event graph models (EGMs) were first described by Schruben (1983) and later enriched by others, including Som and Sargent (1989). Pictorially the vertices of an EGM represent the various events in the simulation. The edges of the graph represent relationships between events. Basically, the edges define the conditions under which and the time delay after which one event will schedule another event to occur. Suppose the following edge is part of a simulation graph,

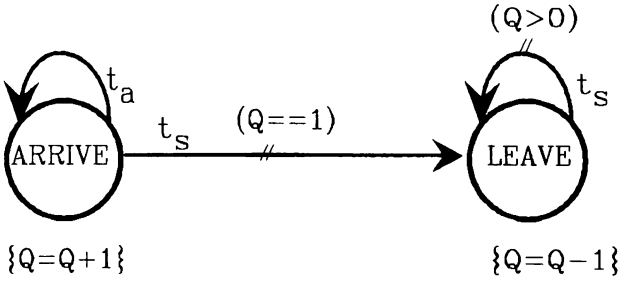


This edge is read as follows:

"whenever event A occurs, the system state, S, changes to $f_A(S)$. Then, if condition (i) is true, event B will be scheduled to occur after a delay of t."

Appropriate labels are omitted if the inter-event edge delay is zero or if the scheduling is unconditional.

One of the simplest examples of an EGM is a single server queue. Here the single state variable, Q, is the number of customers in the system (waiting in line or in service). The random time between customer arrivals is denoted as t_a and the random time of customer service is t_s. The EGM for a generic queue is as follows:



The ARRIVE event simply increments the queue and the LEAVE event decrements the queue.

We formally define an EGM using a directed graph G = {E,V} with edge set E and vertex set V and an associated state space, S. Generic vertices are denoted by v (perhaps with a subscript). Generic edges are denoted as $e = (v_o, v_d)$, which specifies the origin and destination of a directed edge. We label the graph with the following sets:

F = $\{f_v: \forall v \in V\}$ are the state changes associated with each event.

C = $\{c_e: S \to \Re, \forall e = (v_o, v_d) \in E\}$ when $c_e = 0$, the edge condition is false (as in the C programming language).

T = $\{t_e: \forall e = (v_o, v_d) \in E\}$ are the inter-event delay times.

$P = \{p_e: \forall e = (v_o,v_d) \in E\}$ are execution priority expressions used to break time ties.

The conditions in C specify whether or not an edge's destination event will be scheduled after the edge's origin event occurs. At any given time in the execution of the simulation, those edges where $c(s) \neq 0$ (i.e. the edge conditions are true) are referred to as *active* edges. Edges where $c(s) = 0$ can be thought of as being temporarily missing from the graph.

The basic notion of an event graph model, M = (V,E,S,F,C,T,P), is to represent the *indices* for above sets with the edges and vertices of a directed graph. It is this graph of indices that organizes the above sets into a simulation model.

1.2 Generalized Semi-Markov Processes

GSMPs are a useful probability model for discrete event simulation: Comprehensive treatments appear in (Glynn and Iglehart, 1988) and in (Glynn, 1989). We define GSMPs following the development in (Glasserman, 1991). Define the following: $E(s)$ = non-empty set of possible events in state $s \in S$; $p(s';s,\alpha)$ = probability of jumping from state s to s' when event, α, occurs; $F_\alpha(.)$ = distribution of interval until event α occurs. The dynamics of a particular GSMP model may require the two doubly-indexed independent sequences of random variables: (i) $\{X(\alpha,k);\ \alpha \in A,\ k=1,2,...\}$ distributed according to $F_\alpha(.)$ (these are the successive inter-event times for the GSMP sample path) and (ii) $\{U(\alpha,k);\ \alpha \in A,\ k=1,2,...\}$ uniformly distributed on the semi-open interval (0,1] (these are used for random state changes when events occur). Glasserman [1991] also defines the mappings ϕ for each $\alpha \in A$, $s' = \phi_\alpha(s,U(\alpha,k))$ so that $Prob(s' = \phi_\alpha(s,U(\alpha,k)) = p(s';s,\alpha)$. In an EGM of the same system both these input processes might be mappings of the random number sequence, U. The algorithm or scheme by which the dynamics of a GSMP evolve is typically specified by mimicking the execution of a typical event-scheduling discrete event simulation code.

2. A PROPOSED GSMP→EGM MAPPING

We start with a directed graph G = (V,E) and use the same physical state space, S, with a unique vertex, $v \in V$ in the event graph corresponding to each event, α, in the GSMP. The vertices are labeled with the state changes associated with each event $f_v = \phi_\alpha, \alpha = v$. For each pair of vertices, $e = (v_o,v_d)$, define:

$C_e = \{s \to s' \in S : v_o \in E(s)$ and $v_d \in \{E(s') - (E(s) - \{v_o\})\}\}$ where $s \to s'$ indicates that the present state of the system is s' and the state immediately preceding s' was s.

In other words if v_o can cause a transition out of state s and v_d is a new event when the state changes to s' then it must be possible for v_o to schedule v_d (i.e. if the systems enters state s' from state s).

If s' and v_o uniquely determine s (i.e f_v is invertible) the we can write a *static* definition:

$C_e = \{s' \in S : v_o \in E(s)$ and $v_d \in \{E(s') - (E(s) - \{v_o\})\}$ $\forall s \in S \ni p(s';s,v_o) > 0\}$.

Here, if it is possible to get from state s to s' when v_o is executed and v_d is a new event in state s' then v_o must schedule v_d if the system is in state s'.

The edges of G are $E = \{e = (v_o,v_d): C_e \neq \varnothing\}$. Each edge is labeled with:

$t_e = \{X(v_d,k), k=1,2,...\}$, its delay time

$p_e: S \to R$, its execution priority,

and $c_e: S \to \{true,false\}$, its edge condition, where $c_e(s') = true$ iff $s' \in C_e$; that is we label each edge with a membership rule, C_e. Note that E can be countably infinite if S is, typically $|E| < |S|$.

3 EXAMPLE 1: A SINGLE-SERVER QUEUE

GSMP:

$S = \{0,1,2,...\} = Q$ = number of customers in the system

$A = (1, 2)$ = (Arrive, Leave)

$F_1(.)$ is the distribution of the inter-arrival time, t_a,

$F_2(.)$ is the distribution of the service time, t_s.

$E(s) = \{1,2\}$ if $s>0$, and $E(0) = \{1\}$, finally $p(s+1,s,1) = 1$, $p(s-1,s,2) = 1$

EGM:

V = {Arrive, Leave}

note that the state changes f_1 and f_2 are invertible so we can use the static definition of C_e. $f_1^{-1}(s)=s-1$ for $s > 0$, $f_2^{-1}(s)=s+1$

$C_{(1,1)} = \{s' \in Q : 1 \in E(s'-1)$ and $1 \in \{E(s') - (E(s'-1) - \{1\})\}, s>0\} = \{Q>0\}$

$C_{(1,2)} = \{s' \in Q : 1 \in E(s'-1)$ and $2 \in \{E(s') - (E(s'-1) - \{1\})\}, s>0\} = \{Q=1\}$

$C_{(2,1)} = \{s' \in Q : 2 \in E(s'+1)$ and $1 \in \{E(s') - (E(s'+1) - \{2\})\}\} = \{\varnothing\}$

$C_{(2,2)} = \{s' \in Q : 2 \in E(s'+1)$ and $2 \in \{E(s') - (E(s'+1) - \{2\})\}\} = \{Q>0\}$

The resulting event graph is identical to the single server queue model pictured in section 1.1 with the addition of the redundant condition (Q > 0) on the edge generating the arrivals.

4 EXAMPLE 2: A COLLISION-FREE BUS NETWORK

An interesting example of a larger GSMP is the collision-free bus network modeled by Iglehart & Shedler (1983). In this model there are N ports connected by a passive bilateral bus on which message packets are transmitted and received. In addition to the bus, there is a one-way logic control wire that links the ports. This wire transmits send flipflops which are essentially requests by a port to use the bus for transmission. The signal tapped at the control wire input to a port is the inclusive OR of the send flipflops from upstream ports, i.e., the port sees whether some upstream port is waiting to transmit.

When a message packet arrives at a port for transmission, the port sets the send flipflop (requests the bus), waits for a time interval to make sure that its flipflop is received and that it has been notified of any occurring transmission. After that delay, the port waits until the bus is observed to be idle and there are no bus requests from upstream. When those conditions are satisfied, it begins transmission and resets its send flipflop to 0.

Propagation times between ports i and j are denoted as T(i,j) for the actual propagation time along the bus and R(i,j) for the propagation time along the control wire. S(j) represents the send flipflop at port j and P(j) is the inclusive OR signal tapped at port j. For simplicity the model assumes that there is at most one packet in queue at each port; A(j), a random variable, represents the time between completion of transmission and arrival of the next message packet at port j. L(j) is a random variable representing the transmission time for port j. Since event interval times are constant, we omit their description by sampling function.

The variables for this GSMP are:

$W(t) = (W_1(t), ..., W_N(t))$ where $W_j(t)$ is the state of port j at time t. $W_j(t)=1$ if port j has set the flipflop but is not ready to transmit (because it is still waiting), $W_j(t)=2$ if port j has completed the waiting time but has not begun transmitting, $W_j(t)=3$ if port j is transmitting, $W_j(t)=4$ if transmission is complete and port j is waiting for the next message packet.

$U(t) = (U_1(t), ..., U_N(t))$ where $U_j(t)=1$ if port j observes the bus to be busy at time t, and equals 0 otherwise.

$V(t) = (V_{2,1}(t), V_{3,1}(t), V_{3,2}(t), V_{4,1}(t), ..., V_{N,N-1}(t))$ where $V_{j,k}(t)=1$ if port j has observed that port k has set its flipflop and equals zero otherwise.

The following are the events of the GSMP: "arrival of new packet at port j", "end of waiting period for port j", "end of transmission by port j", "observation by port j of the setting (to 1) of the flipflop by port k upstream (k<j)", "observation by port j of the resetting (to 0) of the flipflop by port k upstream (k<j)", "observation by port j of the start of transmission", "observation by port j of the end of transmission." Table 1 displays the events (with abbreviated names) along with their state changes, interval times and the subset of S for which the event is a member of E(s).

Table 1: Attributes of the Events of the Collision-Free Bus Network GSMP

Event	State Changes	time delay	s: event ∈ E(s)
New Packet(j)	W(j) = 1	A(j)	W(j) = 4
Clear(j)	W(j) = 2 or 3*	R(j,N)+T(1,N)	W(j) = 1
End Transmission(j)	W(j) = 4	L(j)	W(j) = 3
Observe Set(j,k)	V(k,j) = 1	R(j,k)	∃ k<j ∋ W(k) = 1, V(k,j) = 0
Observe Reset(j,k)	V(k,j) = 0, W(j) = 2 or 3*	R(j,k)	∃ k<j ∋ W(k) = 3, V(k,j) = 1
Observe Start(j)	U(j) = 1	T(k,j)	∃ k ∋ W(k) = 3, U(j) = 0
Observe End(j)	U(j) = 0, W(j) = 2 or 3*	T(k,j)	∃ k ∋ W(k) = 4, U(j) = 1

* W(j) = 3 if P(j) = 0 and U(j) = 0, W(j) = 2 otherwise

4.1 The Event Graph Translation

In this GSMP all of the state changes are invertible so we can use the static definition of C_e, $C_e = \{s' \in S : v_o \in E(s)$ and $v_d \in \{E(s') - (E(s) - \{v_o\})\}\ \forall s \in S\ \ni p(s';s,v_o) > 0\}$. For example consider v_o=Clear(j), v_d=Observe Start(k). Now Clear(j) $\in$ E(s) if and only if W(j)=1 and Observe Start (k) $\in$ E(s) if and only if W(j)=3 for some j and U(k)=0. So the only possible combinations of s and s' are those where s includes W(j)=1 and s' includes W(j)=3 and U(k)=0 and $p(s';s,v_o) > 0$. C_e therefore contains {s: W(j)=3 and U(k)=0}. C_e is not empty so we would draw an edge from Clear(j) to Observe Start(k); its edge condition would be (W(j)=3 and U(k)=0) and its time delay, from the GSMP, would be T(j,k). Table 2 lists all of the edges for which C_e is non-empty with the edge conditions and time delays. The state changes are the same as given in Table 1. Figure 1 shows the event graph.

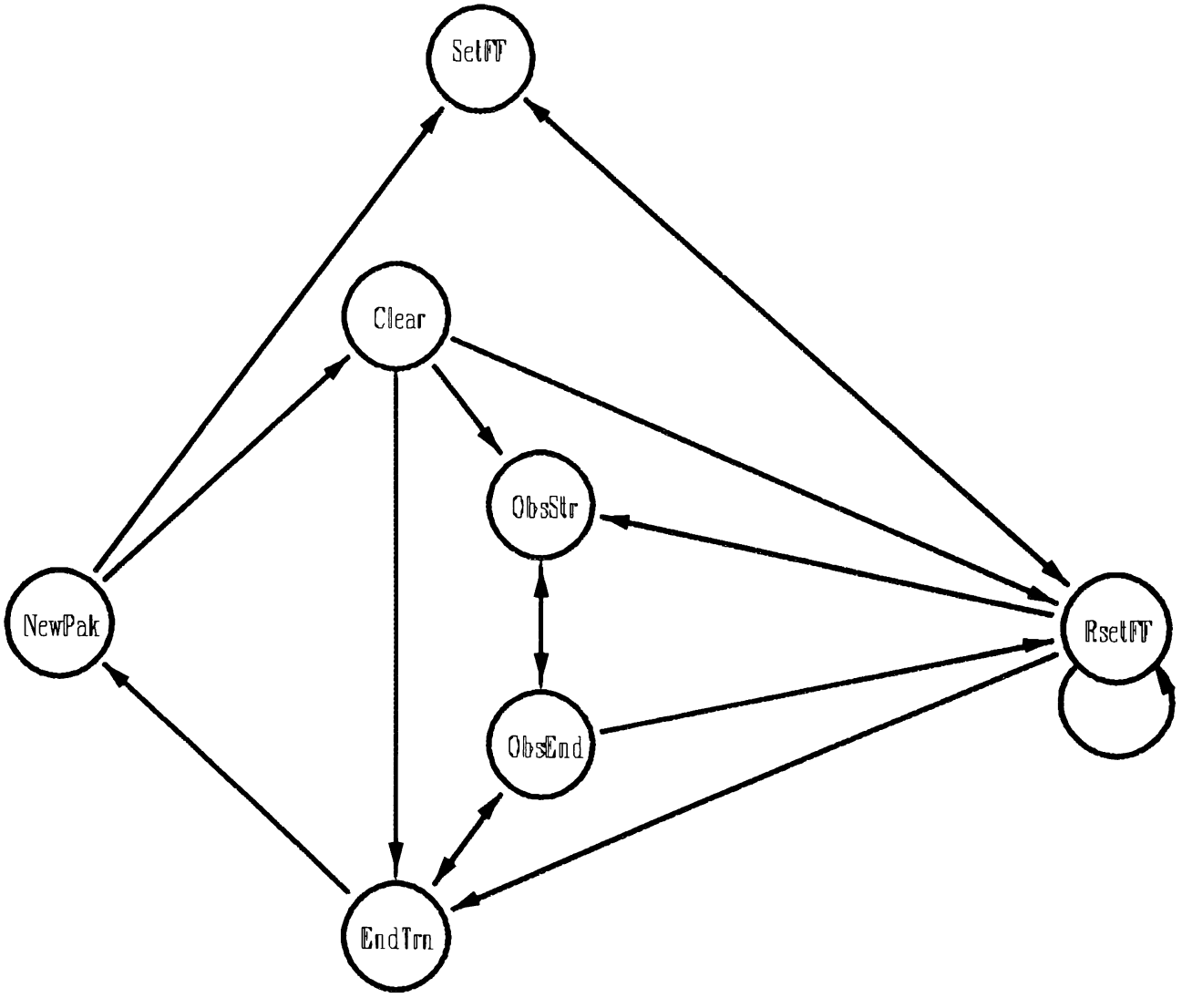


Figure 1: Event Graph of the Collision-Free Bus Network

Table 2: Edges of the Event Graph Translation of the Collision-Free Bus Network GSMP

Edge	Cc	time delay
New Packet(j) → Clear(j)	W(j) = 1	R(j,N)+T(1,N)
New Packet(j) → Observe Set(j,k), k>j	W(j) = 1, V(k,j) = 0	R(j,k)
Clear(j) → End Transmission(j)	W(j) = 3	L(j)
Clear(j) → Observe Reset(j,k), k>j	W(j) = 3, V(k,j) = 1	R(j,k)
Clear(j) → Observe Start(k), ∀k≠j	W(j) = 3, U(k) = 0	T(j,k)
End Transmission(j) → New Packet(j)	W(j) = 4	A(j)
End Transmission(j) → Observe End(k), ∀k≠j	W(j) = 4, U(k) = 1	T(j,k)
Observe Set(j,k) → Observe Reset(l,k), l<k	W(l) = 3, V(k,l) = 1	R(l,k)
Observe Reset(j,k) → End Transmission(j)	W(j) = 3	L(j)
Observe Reset(j,k) → Observe Set(l,k), l<k	W(l) = 1, V(k,l) = 0	R(l,k)
Observe Reset(j,k) → Observe Reset(k,l), l>k	W(k) = 3, V(l,k) = 1	R(k,l)
Observe Reset(j,k) → Observe Start(l), ∀l≠k	W(k) = 3, U(l) = 0	T(k,l)
Observe Start(j) → Observe End(j)	∃ k ∋ W(k) = 4, U(j) = 1	T(j,k)
Observe End(j) → End Transmission(j)	W(j) = 3	L(j)
Observe End(j) → Observe Reset(j,k), k>j	W(j) = 3, V(k,j) = 1	R(j,k)
Observe End(j) → Observe Start(j)	∃ k ∋ W(k) = 3, U(j) = 0	T(j,k)
Observe End(j) → Observe Start(k), ∀k≠j	W(j) = 3, U(k) = 0	T(j,k)

4.2 Checking the Logic

Now that we have an event graph model, we can code it up using available software. In this case we used SIGMA (Schruben 1995). In debugging this model a number of implementation issues arise.

The first concern was that some of the events would become inactive without being executed. For example, when port j begins transmission, an Observe Start event is scheduled for the other ports (because W(j) = 3). If the transmission is finished before port k observes the start then Observe Start(k) disappears from the active event set (W(j) ≠ 3 for any j) although it was never executed.

This means that events are being canceled in the original GSMP. While some GSMP definitions explicitly allow for canceling, others do not. If this were a problem for the modeler then certain assumptions would have to be made to prevent this occurrence

Iglehart and Shedler explicitly state that R(i,j)>T(i,j) ∀i,j. Additional assumptions are required however to prevent cancellation in the GSMP. L(j)>R(j,N) will allow the reset flipflop to finish propagating before the port finishes transmission and A(j)>max$_i$T(j,i) will allow the end of transmission to propagate before a new packet arrives at the port.

Another possibility is to redefine the event sets in the original GSMP. In fact without the above assumptions or the introduction of new state variables it is possible that the model will fail to be a GSMP at all; the system can reach a state where the active event set cannot be fully determined.

For example, suppose port j ends transmission and very soon afterwards a new packet arrives at port j. There may be a port k that has not observed that end of transmission yet; W(j) = 1 but Observe Start(k) should still be an active event. We also have the usual case however, where if W(j) ≠ 4 for any j then there are no Observe Start(k) events active. Therefore if W(j) = 1 we can't tell whether Observe Start(k) should be active or not.

Another situation is the existence of simultaneous events. In running the event graph of this GSMP it became clear that this situation exists in the model despite efforts and assumptions aimed at preventing simultaneous events. Further inspection revealed that the problem was caused by the fact that the end of transmission and start of transmission propagate at the same speed. If a start of transmission by port j is enabled by the observation that port k has ended transmission then the end of k's transmission and start of j's transmission now travel together down the bus. If they are executed in the wrong order (Observe Start then Observe End) then an error results because the other ports will think the bus is free when it is not.

In running the event graph version of the model, it also became clear that there were other problem in the definitions of the event sets for certain states. Specifically Observe End(j) is supposed to be an active

event whenever W(k)=4 for some k and U(j)=1. What happens is that whenever port j observes a start of transmission, U(j) will become 1 and if any other port is waiting for a new packet (W(k)=4) then an Observe End(j) will be scheduled. If port j observes a start of transmission while the transmission is still happening an Observe End may be scheduled because of some other port which is waiting for a new packet, clearly this should not happen.

5 IMPLICATIONS FOR SIMULATION MODELING

Mapping from a GSMP to an EGM may uncover logical errors that are not apparent in the GSMP description. In addition, implementing and running the event graph model may uncover other concerns that need to be addressed in the original GSMP.

We can see in this example that EGMs offer an effective and efficient method for model development and structural analysis and permits development and enrichment of GSMP results and models. GSMPs offer a path to proofs of validity of simulation methodologies and potential algorithms for implementation. A GSMP→EGM mapping bridges a basic application-theory gap in that EGMs are very popular for model building and GSMPs are very popular for developing DEDS theory.

ACKNOWLEDGMENT

The authors are grateful to the Division of Design, Manufacture and Industrial Innovation of the National Science Foundation for their generous support through a research grant to Cornell University.

REFERENCES

Glasserman, P., (1991). "Structural Conditions for Perturbation Analysis Derivative Estimation: Finite Time Performance Indices." *Operations Research* **39**, pp. 724-738

Glynn, P. W., and D. L. Iglehart, (1988) "Simulation Methods for Queues: an Overview", *Queueing Systems*, Vol. 3, pp. 221-255, 1988.

Glynn, P., (1989). "A GSMP Formalism for Discrete Event Systems." *Proceedings of the IEEE*, Vol. 77, No. 1, pp. 14-23.

Iglehart, D. L. and G. S. Shedler, (1983). "Simulation of Non-Markovian Systems." *IBM Journal of Research and Development*, Vol. 27, No.5, pp. 472-479.

Schruben, L. W., (1983). "Simulation Modeling with Event Graphs." *Communications of the ACM*, Vol. 26, No. 11, pp. 957-963.

Schruben, L. W., (1995). *Graphical Simulation Modeling and Analysis: Using SIGMA for Windows*. Boyd & Fraser. Danvers, MA.

Som, T. K. and R. G. Sargent, (1989). "A Formal Development of Event Graphs as an Aid to Structured and Efficient Simulation Programs." *ORSA Journal on Computing*, Vol. 1, No. 2, pp. 107-125.

AUTHOR BIOGRAPHIES

LEE W. SCHRUBEN is a Professor in the School of Operations Research at Cornell University. He received his undergraduate degree in engineering from Cornell and his Ph.D. from Yale. His research interests are in statistical design and analysis of simulation experiments and in graphical simulation modeling methods.

ERIC L. SAVAGE is a Ph.D. candidate in the School of Operations Research at Cornell University. He received his undergraduate degree in mathematics from Rensselaer Polytechnic Institute and an M.S. degree in Operations Research from Cornell. His research interests include modeling logic and methodology in graphical representations.