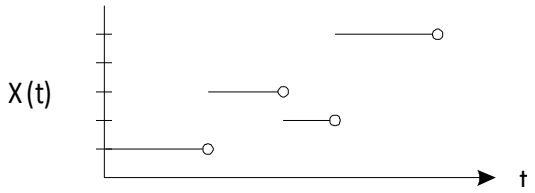
Discrete-Event Systems and Generalized Semi-Markov Processes

Ref: Section 1.4 in Shedler or Section 4.1 in Haas

1. Discrete-Event Stochastic Systems

Recall our previous definition of a discrete-event stochastic system: the system makes stochastic state transitions only at an increasing sequence of random times. If X(t) is the state of the system at time t, then a typical sample path of the underlying stochastic process of the simulation, i.e. (X(t): t ≥ 0), looks like this (for a finite or countably infinite state space):

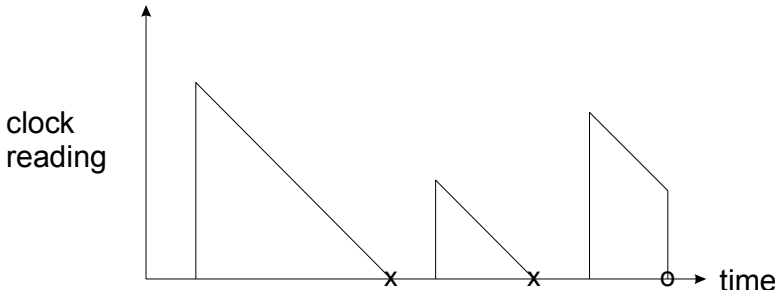


For a given simulation model, we need to precisely specify the process $(X(t): t \geq 0)$, so that we can generate sample paths of the process and obtain meaningful point and interval estimates of system characteristics based on well-defined properties of the process. The basic mathematical model for a discrete-event system is the *generalized semi-Markov process* (GSMP), which captures the key notions of state, events and clocks. After presenting the GSMP model, we will describe the general "variable time advance" algorithm that is used to simulate GSMPs. We then describe some special subclasses of GSMPs that can be simulated efficiently using special-purpose algorithms.

2. The GSMP Model

See the textbook by Gerald Shedler for a thorough treatment of GSMP's. Heuristically, a GSMP $(X(t): t \geq 0)$ makes stochastic *state transitions* when one or more *events* associated with the occupied state occur. (Associated events = events that can possibly occur in the state = events that are scheduled in the state.)

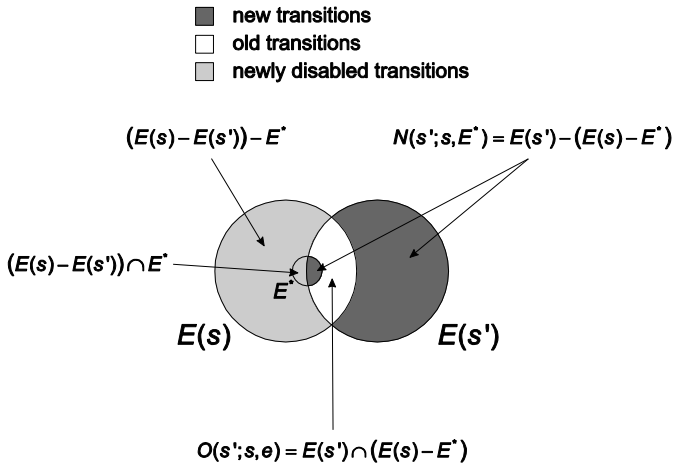
- events *associated with a state* "compete" to trigger the next state transition
- each event has its own *distribution* for determining the next state
- *new events* can be scheduled at each state transition---a new event with respect to a state transition is an event that is associated with the new state and either (a) is not associated with the old state or (b) is associated with the old state and also triggers the state transition
- for each new event, a *clock* is set with a reading that indicates the time until the event is scheduled to occur; when the clock runs down to 0 the event occurs (unless it is canceled in the interim)
- an *old event* with respect to a state transition is an event, associated with the old state, that does not trigger the state transition and is associated with the next state; its clock continues to run down
- a *cancelled event* with respect to a state transition doesn't trigger the state transition and is not associated with the next state; its clock reading is discarded
- clocks can run down at state-dependent speeds



3. GSMP Building Blocks

- S: a (finite or countably infinite) set of states
- $E = \{e_1, e_2, \ldots, e_M\}$: a finite set of events
- $E(s)$: the set of events scheduled to occur in state s ∈ S. Of course, E(s) ⊆ E. We say that event e is *active* in s if e ∈ E(s).
- $p(s';s,E^*)$: the probability that the new state is s' given that the events in E^* simultaneously occur in s. If $E^* = \{e^*\}$ for some $e^* \in E(s)$, then we simply write $p(s';s,e^*)$.
- $r(s,e)$: the nonnegative finite speed at which clock for e runs down in state s; typically r(s, e)=1, but can be set to other values in order to model "processor sharing" or "preempt resume" service discipline. (For modeling the latter we allow r(s, e) = 0.)
- $F(\cdot\,;s',e',s,E^*)$: the distribution function used to set the clock for the new event e' when the simultaneous occurrence of the events in E^* triggers a state transition from s to s'.
- μ: the initial distribution function for the state and clock readings. We will always assume that μ is such that the initial state s is chosen according to a distribution ν and for each event $e \in E(s)$ the clock is set independently according to $F_0(\cdot\,;e,s)$.

Sets of new and old events:



Example: GI/G/1 Queue

Assume that the interarrival-time distribution F_a and the service-time distribution F_s are continuous, so that an arrival and service completion never occur simultaneously. Also assume that a job arrives to an empty system at time 0.

Let $X(t)$ = number of jobs in service or waiting in queue at time t

Then $(X(t): t \geq 0)$ can be specified as a GSMP with

- S = {0, 1, 2, ...}
- E = {e_1, e_2}, where e_1 = "arrival" and e_2 = "completion of service"
- E(s) = {e_1} if s = 0 and E(s) = {e_1, e_2} if s > 0
- p(s+1; s, e_1) = 1 and p(s – 1; s, e_2) = 1
- F(x; s', e', s, e) = F_a(x) if $e' = e_1$ and F_s(x) if $e' = e_2$
- r(s,e) ≡ 1 for all s and e
- ν(1) = 1, $F_0(\cdot\,; e_1, s) = F_a(\cdot)$, and $F_0(\cdot\,; e_2, s) = F_s(\cdot)$

3. GSMP's and GSSMC's

A GSMP is formally defined in terms of a GSSMC {(S_n,C_n): n ≥ 0}, where

- S_n = state just after n^{th} transition
- C_n = ($C_{n,1}$, $C_{n,2}$, ... , $C_{n,M}$) = clock-reading vector after n^{th} transition (by convention, $C_{n,i} = 0$ if event e_i is not active in state S_n)

The transition kernel is given by a somewhat complicated-looking expression, namely

$$P((s,c), A) = p(s'; s, E^*) \prod_{e_i \in N(s')} F(a_i; s', s, E^*) \prod_{e_i \in O(s')} 1_{[0,a_i]}(c_i^*)$$

for sets $A = \{s'\} \times \{(c_1', \ldots, c_M') \in C(s): 0 \leq c_i' \leq a_i \text{ for } 1 \leq i \leq M\}$, where $E^* = E^*(s,c)$, $N(s') = N(s'; s, E^*)$, $O(s') = O(s'; s, E^*)$, and $c_i^* = c_i^*(s,c) = c_i - t^*(s,c)\, r(s,e_i)$.

The initial distribution μ(A) is of the form

$$\mu(A) = \nu(s') \prod_{e_i \in E(s')} F_0(a_i; e_i, s').$$

for a set A as above.)

To construct the process {X(t): t ≥ 0} from {(S_n, C_n): n ≥ 0}, let ζ_n be the time of the n^{th} state transition:

$$\zeta_n = \sum_{k=0}^{n-1} t^*(S_k, C_k)$$

where $t^*(s,c)$ is the holding time in state s starting with clock-reading vector $c = (c_1, c_2, \ldots, c_M)$:

$$t^*(s,c) = \min_{\{i: e_i \in E(s)\}} \{c_i / r(s,e_i)\}$$

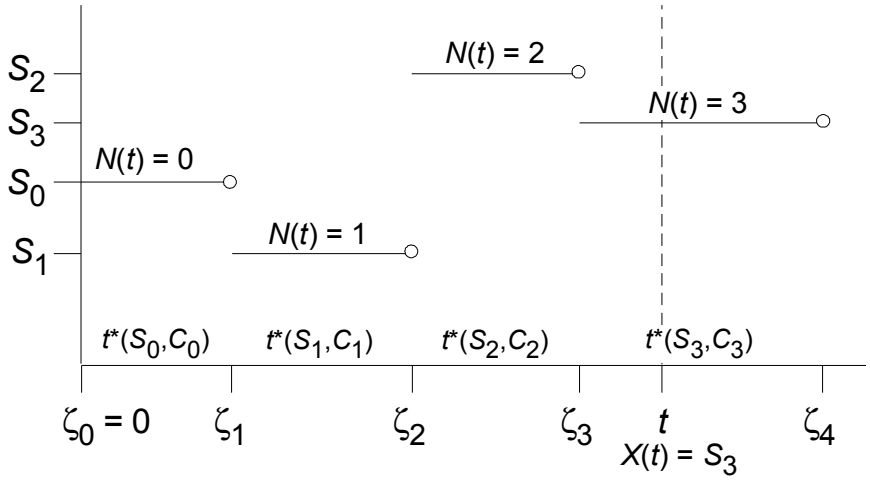
and $N(t)$ is the number of state transitions in $[0,t]$:

$$N(t) = \max\{n \geq 0 : \zeta_n \leq t\}$$

Then let $\Delta \notin S$ and set

$$X(t) = \begin{cases} S_{N(t)} & \text{if } N(t) < \infty \\ \Delta & \text{if } N(t) = \infty \end{cases}$$

for $t \geq 0$. (If S is finite then $N(t)$ is finite for each $t < \infty$.) The process $\{X(t): t \geq 0\}$ is the GSMP. Thus, we can use results from GSSMC theory to study GSMP's.



4. Sample Path Generation

The GSMP definition leads directly to an algorithm for sample-path generation:

Sample-Path Generation Algorithm

1. (Initialization) Select $s \in S$ according to ν. For each $e_i \in E(s)$ generate a clock reading c_i according to $F_0(\cdot; e_i, s)$. Set $c_i = 0$ for $e_i \notin E(s)$.
2. Determine the time $t^*(s,c)$ until the next state transition and determine the set of events $E^* = E^*(s,c) = \{e_i : c_i / r(s,e_i) = t^*(s,c)\}$ that will trigger the next state transition.
3. Generate the next state s' according to the probability mass function $p(\cdot; s, E^*)$.

4. For each $e_i \in N(s';s,E^*)$, generate c_i' according to $F(\cdot;s',e_i,s,E^*)$.
5. For each $e_i \in O(s';s,E^*)$, set $c_i' = c_i - t^*(s,c)r(s,e_i)$.
6. For each $e_i \in$ (E(s) – E*) – E(s'), set $c_i' = 0$ (i.e., cancel event e_i).
7. Set $s = s'$ and $c = c'$, and go to Step 2. (Here $c = (c_1, c_2, \ldots, c_M)$ and similarly for c'.)

This algorithm generates a sequence of states $\{S_n: n \geq 0\}$, clock-reading vectors $\{C_n: n \geq 0\}$, holding times $\{t^*(S_n,C_n): n \geq 0\}$. The state-transition times $\{\zeta_n: n \geq 0\}$, the continuous-time process $\{X(t): t \geq 0\}$, etc., are then computed as described previously. We can then use our usual techniques to estimate quantities of the form $E[f(X(t))]$ or even

$$\alpha = E\left[\frac{1}{t}\int_0^t f(X(u))\,du\right] = E\left[\frac{1}{t}\left(\sum_{n=0}^{N(t)-1} f(S_n)t^*(S_n,C_n) + f(S_{N(t)})\left(t-\zeta_{N(t)}\right)\right)\right],$$

The algorithm is an example of a *variable time-advance procedure* (and is given at a high level of abstraction). We will discuss some more detailed computational issues for such procedures shortly. (Exercise: What other kinds of time advance procedures can you think of?)

Flowcharts and figures: When programming a simulation of a complex system such as a GSMP, it can be very helpful to use diagrams and flowcharts to increase your understanding of the system and avoid programming errors. Flowcharts can correspond to the overall simulation logic or to a component of a simulation (such as the processing of an arrival event in a queue), and can specify which random variables to generate, and so on. See Law, pp. 30-31 for a simple example involving a queueing simulation.

5. Generating Clock Readings

How do we generate clock readings from a general clock-setting distribution during the sample-path generation algorithm? We'll discuss this more fully later, but right now we will present the most basic technique. Let's start with a specific example:

Generation of Exponential Random Variables:

The exponential distribution with rate (or intensity) λ has density function f and distribution function F given by

$$f(x;\lambda) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \quad \text{and} \quad F(x;\lambda) = \begin{cases} 1-e^{-\lambda x} & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}.$$

The mean of the distribution is $1/\lambda$.

Suppose that we have a uniform random variable U, and set V = $\frac{-\ln U}{\lambda}$. We claim that V $\stackrel{D}{=}$ exp(λ).

Proof:

$$P\{V > x\} = P\left\{\frac{-\ln U}{\lambda} > x\right\} = P\{\ln U < -\lambda x\} = P\{U < e^{-\lambda x}\} = e^{-\lambda x}$$

This is an instance of a more general method for generating non-uniform RV's from uniform(0,1) RV's called the *inversion method*.

The Inversion Method:

The idea is to generate a random variable V having continuous increasing distribution function F(x) = P{V ≤ x} by setting

$$V = F^{-1}(U),$$

where F^{-1} is the inverse of F and U is uniform(0,1). Since F is non-decreasing, we have

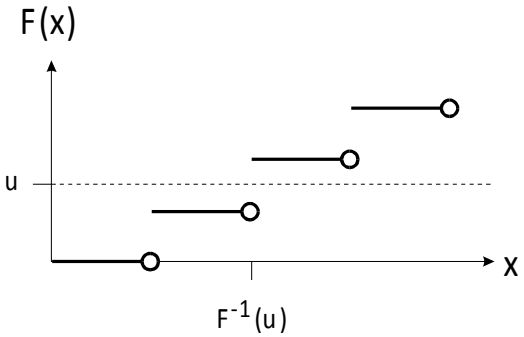
$$P\{V \le x\} = P\{F^{-1}(U) \le x\} = P\{F(F^{-1}(U)) \le F(x)\} = P\{U \le F(x)\} = F(x)$$

For example, in the case of an exp(λ) random variable, we have $F(x) = 1 - e^{-\lambda x}$, so that

$F^{-1}(u) = \frac{-\ln(1-u)}{\lambda}$, and we can set $V = \frac{-\ln(1-U')}{\lambda}$, where $U' \stackrel{D}{=}$ uniform(0,1). Observing that $U = 1 - U'$ also is a uniform(0,1) RV, we obtain our previous formula.

The method can be extended to any distribution function F (not necessarily continuous) if we define

$$F^{-1}(u) = \min\{x: F(x) \ge u\}.$$



The proof is almost the same as above, and hinges on the fact that $F^{-1}(u) \le x$ iff $u \le F(x)$ by definition of the function $F^{-1}(u)$.

Exercise: Show that the inversion method, when applied to generate a discrete random variable, coincides with the naive method given earlier.

6. Markovian and Semi-Markovian GSMP's

Certain subclasses of GSMPs can be simulated more efficiently than by using the general sample-path generation algorithm.

Definition: An event e' is *simple* if

$$F(\cdot;s',e',s,E^*) \equiv F(\cdot;e') \text{ and } F_0(\cdot;e';s) \equiv F(\cdot;e')$$

for some function $F(\cdot;e')$ and all s', s, and E*.

Suppose that each clock-setting distribution is simple and exponential, so that $F(x;e_i) = 1 - e^{-\lambda_i x}$ for all $x \geq 0$ and $e_i \in E$. The exponential distribution has three special properties that we can exploit: if $X \stackrel{D}{=} \exp(\lambda)$ and $Y \stackrel{D}{=} \exp(\mu)$ with X and Y independent, then

$$\min(X,Y) \stackrel{D}{=} \exp(\lambda+\mu) \quad \text{(independent of whether } X < Y \text{ or } Y < X\text{)},$$

$$P(X < Y) = \frac{\lambda}{\lambda+\mu},$$

and

$$P(X > a+b \mid X > a) = e^{-\lambda b}.$$

The first two properties generalize to an arbitrary number of exponentially distributed random variables. The third property is called the "memoryless" property of the exponential distribution.

As a consequence of the memoryless property, it can be shown that, whenever the GSMP jumps into a state s, the clock readings for the events in $E(s)$ are distributed as independent exponential random variables. Using the remaining two properties, we obtain the following simulation algorithm.

1. (Initialization) Select $s \in S$ according to ν.
2. Generate a holding time t^* as a sample from an $\exp(\lambda)$ distribution, where $\lambda = \lambda(s) = \sum_{e_i \in E(s)} \lambda_i$
3. Select $e_i \in E(s)$ as the trigger event with probability λ_i / λ.
4. Generate the next state s' according to the probability mass function $p(\cdot;s,e_i)$
5. Set $s = s'$ and go to step 2.

Observe that the sequence $(S_n : n \geq 0)$ of states visited by the GSMP is a DTMC with transition matrix given by $R(s,s') = \sum_{e_i \in E(s)} p(s';s,e_i)(\lambda_i / \lambda)$. Given the sequences of states, the holding times are mutually independent, with the holding time in state S_n distributed as $\exp\big(\lambda(S_n)\big)$.

It is often the case that the occurrence of event e_i in s will cause a transition from s to a unique state $y_i = y_i(s)$ with probability 1. Then the algorithm simplifies even further:

1. (Initialization) Select $s \in S$ according to ν.
2. Generate a holding time t^* as a sample from an $\exp(\lambda)$ distribution, where $\lambda = \sum_{e_i \in E(s)} \lambda_i$
3. Set $s' = y_i(s)$ with probability λ_i / λ
4. Set $s = s'$ and go to step 2.

As can be seen, there is no need to maintain individual clock readings as in the general case.

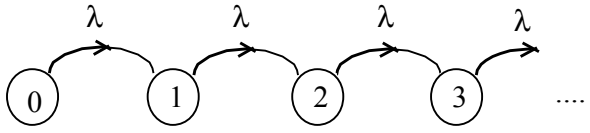
It can be shown that a GSMP {X(t): t ≥ 0} with simple and exponentially distributed clock-setting distributions is in fact a *continuous-time Markov chain* (CTMC) in that it has a finite or countable state space and obeys a continuous-time version of the Markov property:

$$P\{X(t+u) = s \mid X(s) : 0 \leq s \leq t\} = P\{X(t+u) = s \mid X(t)\}.$$

(See Ross, Ch. 6, for a general discussion of CTMCs.) All CTMCs have the sample path structure described above: the sequence of states visited is a CTMC and, given the states, the holding times are mutually independent and exponentially distributed with a rate that depends only on the current state.

Poisson Process

An important special case is when $S = \{0,1,2,\ldots\}$, there is a single $\exp(\lambda)$ event, and the occurrence of the event in state s causes a transition to state $s+1$ with probability 1. A graphical representation of the process, which we denote by $\{N(t) : t \geq 0\}$ is as follows.



The process $\{N(t) : t \geq 0\}$ is called a *Poisson process* with rate λ. The holding times in successive state are i.i.d. according to an exp(λ) distribution. It can be shown that

$$P\big(N(t+s) = m+n \mid N(t) = m\big) = \frac{e^{-\lambda s}(\lambda s)^n}{n!} \text{ for all } t \text{ and } m.$$

That is, the number of upward jumps that occur within a specified time interval follows a Poisson distribution.

A common example of a Poisson process occurs when the successive interarrival times to a queue are i.i.d. according to an exp(λ) distribution. Then $N(t)$ is the number of arrivals to the queue in the time interval [0, t].

Poisson processes arise in other settings also. For example, if one has a component with an exponentially distributed lifetime that is immediately replaced with an identical component, then the total number of replaced components in $[0,t]$ is given by a Poisson process $\{N(t): t \geq 0\}$.

Remark:

A Poisson process is a special type of *renewal (counting) process*. Specifically, suppose that T_n, the time at which the *n*th event occurs, is given by $T_n = \tau_1 + \tau_2 + \cdots + \tau_n$, where the τ_i's are i.i.d. positive random variables. Then $N(t) = \max\{n \geq 0 : T_n \leq t\}$ counts the number of events that occur in $[0,t]$, and is called a renewal counting process. A renewal counting process is a special case of a GSMP in which there is exactly one simple event e and the transition probability function is given by $p(s+1;s,e) = 1$ for $s \geq 0$. A Poisson process with rate λ is the special case in which each τ_i is an exponential random variable with mean λ^{-1}.

Semi-Markov process

A GSMP $\{X(t): t \geq 0\}$ with simple events in which exactly one event is active in a state, so that $|E(s)| = 1$ for each $s \in S$, is a *semi-Markov process* (SMP). A semi-Markov process $\{X(t): t \geq 0\}$ with discrete state space *S* is similar to a CTMC in that the sequence $(X_n : n \geq 0)$ of states visited by the process is a DTMC with transition matrix, say, *R*. The holding time in state $s \in S$, however, is distributed according to an arbitrary distribution function $F(\cdot; s)$ that can depend on *s*. (This definition is a little narrower than some, in which the distribution of the holding time in state *s* can also depend on s', the next state hit by the process.)

Example (Renewal Counting Process):

When

- $S = \{0,1,2,\ldots\}$
- $R(s, s+1) = 1$ for all $s \in S$
- $F(\cdot; s) = G(\cdot)$ for some function $G(\cdot)$.

then $\{X(t): t \geq 0\}$ coincides with a renewal counting process.

Simulation of a semi-Markov process:

The algorithm for simulation of a CTMC applies almost unchanged, except that the holding time in state X_n is generated according to $F(\cdot; X_n)$ rather than $\exp(\lambda(X_n))$.

7. A More Complicated GSMP Example

Patrolling Repairman:

- N machines
- single repairman visits machines in order $1 \to 2 \to \cdots \to N \to 1 \to 2 \to \cdots$
- repairs stopped machine, passes running machine
- repair times for machine j are i.i.d. as a random variable R_j
- lifetimes for machine j are i.i.d. as a continuous random variable L_j
- walking time from machine j to next machine is a constant $W_j > 0$
- at time 0, the repairman has just finished repairing machine 1 and all other machines are broken.

Suppose we wish to estimate μ_r, the expected fraction of time in [0, t] that the repairman spends repairing machines. If we define our system state by X(t) = A(t), where

$$A(t) = \begin{cases} 1 & \text{if repairman is repairing a machine} \\ 0 & \text{otherwise} \end{cases}$$

then $\mu_r = E\left[\frac{1}{t}\int_0^t A(u)\,du\right]$. We might also want to estimate μ_s, the expected number of stopped machines at time t, or μ_w, the long-run average wait for repair for machine 1.

Problems:

- Can't determine number of stopped machines from observing A(t)
- Not clear how to generate sample paths of $\{A(t): t \geq 0\}$

$\Rightarrow$ need to put more information into state definition

Here's another attempt at a state definition:

$$X(t) = (Z_1(t), Z_2(t), \ldots, Z_N(t), M(t), N(t)),$$

where

$$Z_j(t) = \begin{cases} 1 & \text{if machine j is waiting for repair at time t} \\ 0 & \text{otherwise} \end{cases}$$

$$M(t) = \begin{cases} j & \text{if machine j is under repair at time t} \\ 0 & \text{if no machine is under repair at time t} \end{cases}$$

N(t) = j if at time t the repairman will next arrive at machine j

Then we can generate sample paths of {X(t): t ≥ 0} (because this process is a well-defined GSMP as shown below and, as was shown earlier, there is a well-defined algorithm for generating sample paths of a GSMP). Also, all of the system characteristics of interest can be precisely expressed in terms of {X(t): t ≥ 0}:

$$\mu_r = E\left[\frac{1}{t}\int_0^t f_r(X(u))\,du\right] \quad \text{and} \quad \mu_s = E[f_s(X(t))]$$

where

$$f_r(z_1, \ldots, z_N, m, n) = 1_{\{1,2,\ldots,N\}}(m)$$

$$f_s(z_1, \ldots, z_N, m, n) = z_1 + z_2 + \ldots + z_N + 1_{\{1,2,\ldots,N\}}(m)$$

(Recall that $1_A(x) = 1$ if $x \in A$ and $1_A(x) = 0$ otherwise.)

Also, we can express μ_w in terms of {X(t): t ≥ 0}. To see this, set $B_0 = 0$ and recursively define the start and termination of the n^{th} waiting time for machine 1 by

$A_n = \min\{\zeta_k > B_{n-1}: Z_1(\zeta_{k-1}) = 0 \text{ and } Z_1(\zeta_k) = 1\}$ and $B_n = \min\{\zeta_k > A_n: M(\zeta_{k-1}) \neq 1 \text{ and } M(\zeta_k) = 1\}$

where ζ_n is the time of the n^{th} state transition. We can also define these times in terms of the continuous time process by setting

$$A_n = \min\{t > B_{n-1}: Z_1(t-) = 0 \text{ and } Z_1(t) = 1\} \text{ and } B_n = \min\{t > A_n: M(t-) \neq 1 \text{ and } M(t) = 1\},$$

where X(t-) indicates the stat of the system just before time t.

In either case, we can then write the n^{th} waiting time as $D_n = B_n - A_n$, and hence

$$\mu_w = \lim_{n\to\infty} \frac{1}{n}\sum_{k=1}^{n} D_k \quad \text{(assuming that it exists)}$$

The process {X(t): t ≥ 0} can be specified as a GSMP as follows:

- S consists of all $(z_1, \ldots, z_N, m, n) \in \{0,1\}^N \times \{0, 1, \ldots, N\} \times \{1, 2, \ldots, N\}$ such that
  - n = m + 1 if 0 < m < N
  - n = 1 if m = N
  - m = j only if $z_j = 0$ $(1 \le j \le N)$

- $E = \{e_1, e_2, \ldots, e_{N+2}\}$, where
  - e_j = "stoppage of machine j" $(1 \le j \le N)$
  - e_{N+1} = "completion of repair"
  - e_{N+2} = "arrival of repairman"
- E(s) is defined as follows for $s = (z_1, \ldots, z_N, m, n)$:
  - $e_j \in E(s)$ $(1 \le j \le N)$ iff $z_j = 0$ and $m \ne j$
  - $e_{N+1} \in E(s)$ iff $m > 0$
  - $e_{N+2} \in E(s)$ iff $m = 0$
- $p(s'; s, e^*)$ is defined as follows:
  - if $e^* = e_j$ $(1 \le j \le N)$, then $p(s'; s, e^*) = 1$
    when $s = (z_1, \ldots, z_{j-1}, 0, z_{j+1}, \ldots, z_N, m, n)$ and $s' = (z_1, \ldots, z_{j-1}, 1, z_{j+1}, \ldots, z_N, m, n)$
  - if $e^* = e_{N+2}$, then $p(s'; s, e^*) = 1$
    when $s = (z_1, \ldots, z_{j-1}, 1, z_{j+1}, \ldots, z_N, 0, j)$ with $j < N$ and $s' = (z_1, \ldots, z_{j-1}, 0, z_{j+1}, \ldots, z_N, j, j+1)$;
    when $s = (z_1, z_2, \ldots, z_{N-1}, 1, 0, N)$ and $s' = (z_1, z_2, \ldots, z_{N-1}, 0, N, 1)$;
    when $s = (z_1, \ldots, z_{j-1}, 0, z_{j+1}, \ldots, z_N, 0, j)$ with $j < N$ and $s' = (z_1, \ldots, z_{j-1}, 0, z_{j+1}, \ldots, z_N, 0, j+1)$;
    and when $s = (z_1, z_2, \ldots, z_{N-1}, 0, 0, N)$ and $s' = (z_1, z_2, \ldots, z_{N-1}, 0, 0, 1)$
  - exercise: do the case $e^* = e_{N+1}$
- $F(x; s', e', s, e^*)$ is defined as follows
  - if $e' = e_j$ $(1 \le j \le N)$, then $F(x; s', e', s, e^*) = P\{L_j \le x\}$
  - if $e' = e_{N+1}$ and $s' = (z_1, \ldots, z_N, m, n)$ then $F(x; s', e', s, e^*) = P\{R_m \le x\}$
  - if $e' = e_{N+2}$ and $s' = (z_1, \ldots, z_N, 0, n)$ then $F(x; s', e', s, e^*) = 1_{[0,x]}(W_{n-1})$ if $n > 1$ and $1_{[0,x]}(W_N)$ if $n = 1$
- $r(s, e) \equiv 1$ for all s and e
- initial dist'n: $\nu(s) = 1$, where s = (0,1,1,...,1,0,2), $F_0(x; e_1, s) = P\{L_1 \le x\}$ and $F_0(x; e_{N+2}, s) = 1_{[0,x]}(W_1)$

Note: Specifying a GSMP can be complex and time-consuming. Some reasons to do this are:

1. A GSMP description as above, together with the general GSMP simulation algorithm, given previously, provide direct guidance when coding up the model, and help ensure that "corner cases" are not overlooked.

2. A GSMP description can be used to communicate the model to others at a high level (rather than requiring people to grope their way through your code). This helps to separate questions of correct logical model specification from correct implementation in a specific programming language.

3. Theory developed for GSMPs can help in establishing important properties of the simulation, such as stability, i.e., convergence to steady state, so that steady-state estimation problems are well defined. Other results can be used to ensure that simulation output-analysis methods (such as the regenerative and batch-means methods that we will cover later in the course) can validly be applied to the specific model of interest, so that the resulting estimates are correct.