Fault-Tolerant Computer System Design
ECE 60872

Topic: Simulation & Modeling

Saurabh Bagchi
ECE/CS
Purdue University

Markov Chains

- Markov process: Probability distribution for future state only depends on the present state, not on how the process arrived at the present state
- Markov chain: State space is discrete
- Discrete time Markov chain: Time is discrete
- $X_0, X_1, .., X_n, ..$:observed state at discrete times $t_0, t_1,..,t_n, ..$
- $X_n = j \Rightarrow$ system state at time step n is j.
- $P(X_n = i_n | X_0 = i_0, X_1 = i_1, \ldots, X_{n-1} = i_{n-1})$
  $= P(X_n = i_n | X_{n-1} = i_{n-1})$ (Markov Property)
- $p_{jk}(m,n) \equiv P(X_n = k \mid X_m = j), 0 \leq m \leq n$ (conditional pmf)
- $p_j(n) \equiv P(X_n = j)$ (unconditional pmf)

Homogeneous Markov Chain

- Homogeneous Markov chain: $p_{jk}(m,n)$ only depends on n-m
- n-step transition probability: $p_{jk}(n) = P(X_{m+n} = k \mid X_m = j)$
  - *1-step transition probability:* $p_{jk} = p_{jk}(1) = P(X_n = k \mid X_{n-1} = j)$
- Initial probability row vector: $\mathbf{p}(0) = [p_0(0), p_1(0), \ldots, p_k(0), \ldots]$
- Transition probability matrix:

$$P = [p_{ij}] = \begin{bmatrix} p_{00} & p_{01} & p_{02} & \cdot & \cdot \\ p_{10} & p_{11} & p_{12} & \cdot & \cdot \\ \vdots & \vdots & \vdots & \cdot & \cdot \end{bmatrix}$$

Markov Chain Example

1. System is in state 0 if it is operational, state 1 if it is undergoing repair. Homogeneous DTMC process.

$$P = \begin{bmatrix} 1-a & a \\ b & 1-b \end{bmatrix}, 0 \le a, b \le 1$$

IL: $a < b$
$1-a\uparrow$ and $b\uparrow$

2. A sequence of successive software runs, each run has two possible outcomes: success or failure. X_n denotes outcome of n^{th} run (0 for success, 1 for failure). Occurrence of failure in a run depends on the outcome of previous run.

$$P(X_{n+1}=1 \mid X_n=0) = 1-p$$
$$P(X_{n+1}=1 \mid X_n=1) = q$$

$1-p = q$
Independence

Computation of n-step Transition Probabilities

State i to j in n steps

- For a DTMC, find $p_{ij}(n) = P(X_{m+n} = j \mid X_m = i)$
- Events: State reaches *k* (from *i*) & reaches *j* (from *k*) are independent due to the Markov property (i.e. no history)
- Invoking the theorem of total probability:

$$p_{ij}(m+n) = \sum_k p_{ik}(m) p_{kj}(n)$$

- Let P(*n*) : *n*-step prob. transition matrix (*i,j*)th entry is $p_{ij}(n)$. Making $m=1$, $n=n-1$ in the above equation:

$$P(n) = P.P(n-1) = P^n$$

$$p_j(n) = P(X_n = j) = \sum_i P(X_0 = i) P(X_n = j | X_0 = i) = \sum_i p_i(0) p_{ij}(n)$$

Computation of n-step Transition Probabilities

- The pmf of the random variable X_n is

$$p_j(n) = P(X_n = j) = \sum_i P(X_0 = i) P(X_n = j | X_0 = i) = \sum_i p_i(0) p_{ij}(n)$$

- *j*, in general can assume countable values, 0,1,2, …. Defining,

$$\mathbf{p}(n) = [p_0(n), p_1(n), \ldots, p_j(n), \ldots]$$

- $p_j(n)$ for $j=0,1,2,..,k,\ldots$ can be written in the vector form as, $[p_0\ p_1\ p_2\ \ldots\ p_j\ \ldots](n) =$

$$\underbrace{\begin{bmatrix} p_0 & p_1 & p_2 & \cdots & p_j & \cdots \end{bmatrix}(0)}_{\mathbf{p(0)}} \begin{bmatrix} p_{00}(n) & p_{01}(n) & \ldots p_{0j}(n) & \ldots \\ p_{10}(n) & p_{11}(n) & \ldots p_{1j}(n) & \ldots \\ p_{20}(n) & p_{21}(n) & \ldots p_{2j}(n) & \ldots \\ \vdots & \vdots & \vdots & \vdots \\ p_{i0}(n) & p_{i1}(n) & \ldots p_{ij}(n) & \ldots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

Two state Markov chain example

- For a two state DTMC, say the transition prob. matrix is

$$P = \begin{bmatrix} 1-a & a \\ b & 1-b \end{bmatrix},\ 0 \le a, b \le 1,\ |1-a-b| < 1$$

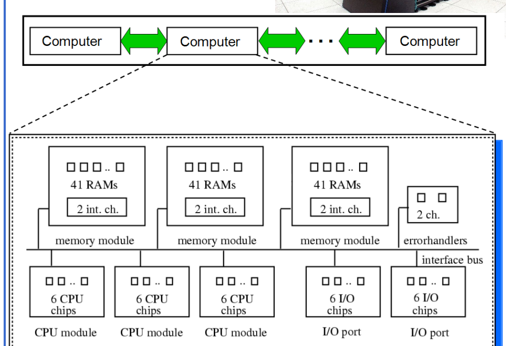
- Then, the n-step transition probability matrix $P(n) = P^n$ is

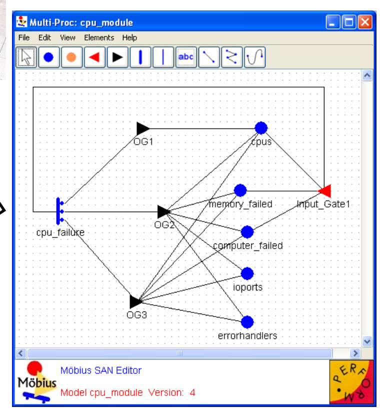
$$P(n) = \begin{bmatrix} \frac{b+a(1-a-b)^n}{a+b} & \frac{a-a(1-a-b)^n}{a+b} \\ \frac{b-b(1-a-b)^n}{a+b} & \frac{a+b(1-a-b)^n}{a+b} \end{bmatrix}$$

- Example: Consider a communication network consisting of a sequence of binary communication channels, with a=1/4, b=1/2. Here X_n denotes the bit leaving the nth stage and X_o represents the bit entering the system.

1. If a bit enters as 1, what is the probability of being correctly transmitted over (a) two stages (b) three stages?
2. Assuming initial probabilities $P(X_0=0) = 1/3$ and $P(X_0=1) = 2/3$, what is **p**(n)?

Modeling





Reliability: 0.9832 at 20 years

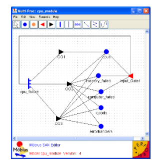
Stochastic Activity Networks

Stochastic activity networks, or SANs, are a convenient, graphical, high-level language for describing system behavior. SANs are useful in capturing the stochastic (or random) behavior of a system.

Examples:

- The amount of time a program takes to execute can be computed precisely if all factors are known, but this is nearly impossible and sometimes useless. At a more abstract level, we can approximate the running time by a random variable.
- Fault arrivals almost always must be modeled by a random process.

We begin by describing a subset of SANs: stochastic Petri nets.



Stochastic Petri Net Overview

One of the simplest high-level modeling formalisms is called *stochastic Petri nets*. A stochastic Petri net is composed of the following components:

- Places: which contain tokens, and are like variables

  NumCPU

- tokens: which are the "value" or "state" of a place

  NumCPU=3

- transitions: which change the number of tokens in places

  Timed Untimed

  every 30 seconds

- input arcs: which connect places to transitions

  CPU consumed every 30 seconds

- output arcs: which connect transitions to places

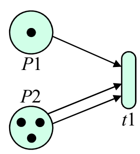
  CPU is placed in busy place

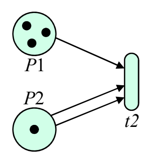
Firing Rules for SPNs

A stochastic Petri net (SPN) executes according to the following rules:

- A transition is said to be *enabled* if for each place connected by input arcs, the number of tokens in the place is ≥ the number of input arcs connecting the place and the transition.

Example:





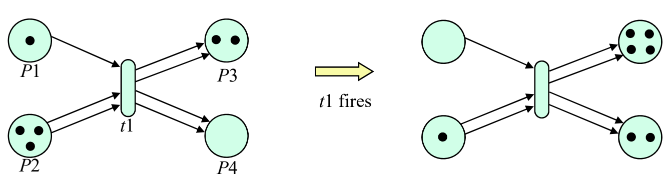
Transition $t1$ is enabled.

Transition $t2$ is not enabled.

Firing Rules, cont.

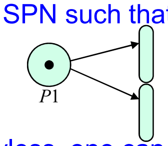
- A transition may *fire* if it is enabled. (More about this later.)
- If a transition fires, for each input arc, a token is removed from the corresponding place, and for each output arc, a token is added to the corresponding place.

Example:



Specification of Stochastic Behavior of an SPN

- A stochastic Petri net is made from a Petri net by
  - Assigning an exponentially distributed time to all transitions.
  - Time represents the “delay” between enabling and firing of a transition.
  - Transitions “execute” in parallel with independent delay distributions.
- Since the minimum of multiple independent exponentials is itself exponential, time between transition firings is exponential.
- If a transition *t* becomes enabled, and before *t* fires, some other transition fires and changes the state of the SPN such that *t* is no longer enabled, then *t* *aborts*, that is, *t* will not fire.



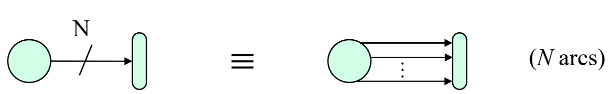
- Since the exponential distribution is memoryless, one can say that transitions that remain enabled continue or restart, as is convenient, without changing the behavior of the network.

SPN Example: Readers/Writers Problem

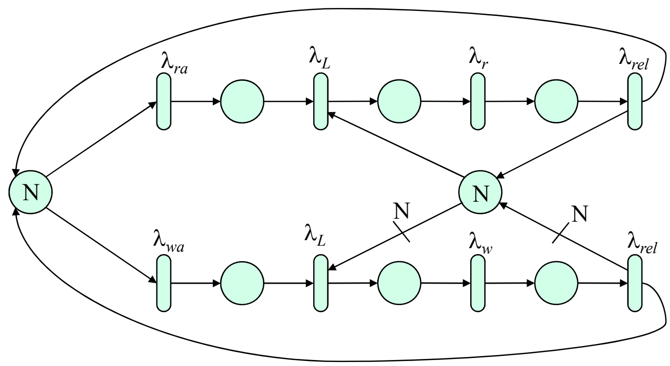
There are at most *N* requests in the system at a time. Read requests arrive at rate λ_{ra}, and write requests at rate λ_{wa}. Any number of readers may read from a file at a time, but only one writer may write at a time. A reader and writer may not access the file at the same time.

Locks are obtained with rate λ_L (for both read and write locks); reads and writes are performed at rates λ_r and λ_w respectively. Locks are released at rate λ_{rel}.

Note:

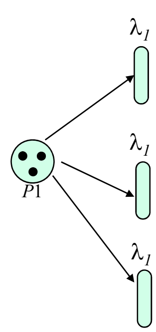


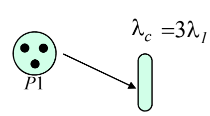
SPN Representation of Reader/Writers Problem



SPN Example: Expected Timings

- Since the minimum of multiple independent exponentials is itself exponential, time between transition firings is exponential.





Stochastic Activity Networks

The need for more expressive modeling languages has led to several extensions to stochastic Petri nets. One extension that we will examine is called *stochastic activity networks*. Because there are a number of subtle distinctions relative to SPNs, stochastic activity networks use different words to describe ideas similar to those of SPNs.

Stochastic activity networks have the following properties:

- A general way to specify that an activity (transition) is enabled
- A general way to specify a completion (firing) rule
- A way to represent zero-timed events
- A way to represent probabilistic choices upon activity completion
- State-dependent parameter values
- General delay distributions on activities

SAN Symbols

Stochastic activity networks (hereafter SANs) have four new symbols in addition to those of SPNs:

- Input gate: used to define complex enabling predicates and completion functions
- Output gate: used to define complex completion functions
- Cases: (small circles on activities) used to specify probabilistic choices
- Instantaneous activities: used to specify zero-timed events

SAN Enabling Rules

An input gate has two components:

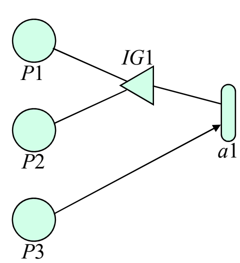
- enabling_function (state) → boolean; also called the *enabling predicate*
- input_function(state) → state; rule for changing the state of the model

An activity is *enabled* if for every connected input gate, the enabling predicate is true, and for each input arc, the number of tokens in the connected place ≥ number of arcs.

We use the notation *MARK*(*P*) to denote the number of tokens in place *P*.

Example SAN Enabling Rule

Example:



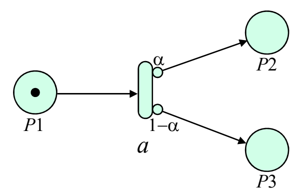
IG1 Predicate:

```
if((MARK(P1)>0 && MARK(P2)==0)||
        (MARK(P1)==0 && MARK(P2)>0))
                return 1;
else return 0;
```

Activity *a*1 is enabled if *IG*1 predicate is true (1) and *MARK*(*P*3) > 0. (Note that “1” is used to denote true.)

Cases

Cases represent a probabilistic choice of an action to take when an activity completes.



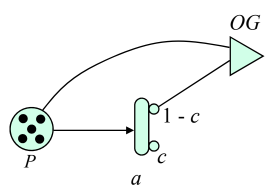
When activity *a* completes, a token is removed from place *P*1, and with probability α, a token is put into place *P*2, and with probability 1 - α, a token is put into place *P*3.

Note: cases are numbered, starting with 1, from top to bottom.

Output Gates

When an activity completes, an output gate allows for a more general change in the state of the system. This output gate function is usually expressed using pseudo-C code.

<u>Example</u>

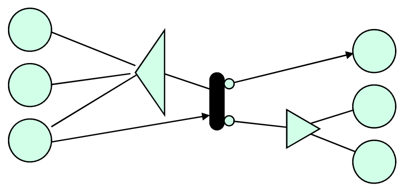


<u>OG Function</u>

MARK(P) = 0;

Instantaneous Activities

Another important feature of SANs is the instantaneous activity. An *instantaneous activity* is like a normal activity except that it completes in zero time after it becomes enabled. Instantaneous activities can be used with input gates, output gates, and cases.

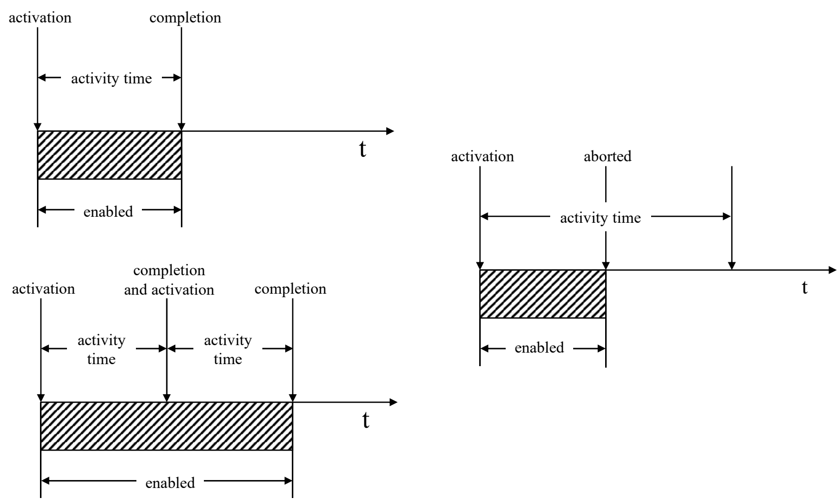


Instantaneous activities are useful when modeling events that have an effect on the state of the system, but happen in negligible time, with respect to other activities in the system, and the performance/dependability measures of interest.

SAN Terms

1. *activation* - time at which an activity begins
2. *completion* - time at which activity completes
3. *abort* - time, after activation but before completion, when activity is no longer enabled
4. *active* - the time after an activity has been activated but before it completes or aborts.

Illustration of SAN Terms



Completion Rules

When an activity *completes*, the following events take place (in the order listed), possibly changing the marking of the network:

1. If the activity has cases, a case is (probabilistically) chosen.
2. The functions of all the connected input gates are executed (in an unspecified order).
3. Tokens are removed from places connected by input arcs.
4. The functions of all the output gates connected to the chosen case are executed (in an unspecified order).
5. Tokens are added to places connected by output arcs connected to the chosen case.

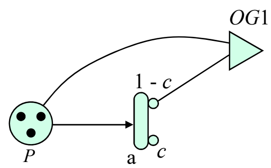
Ordering is important, since effect of actions can be marking-dependent.

Marking Dependent Behavior

Virtually every parameter may be any function of the state of the model. Examples of these are

- rates of exponential activities
- parameters of other activity distributions
- case probabilities

An example of this usefulness is a model of three redundant computers where the coverage (probability that a single computer crashing crashes the whole system) increases after a failure.



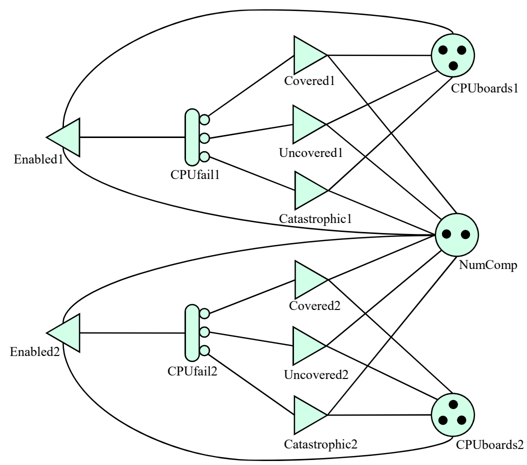
	a
case 1	0.1 + 0.02 * MARK(P)
case 2	0.9 – 0.02 * MARK(P)

Fault-Tolerant Computer Failure Model Example

A fault-tolerant computer system is made up of two redundant computers. Each computer is composed of three redundant CPU boards. A computer is operational if at least 1 CPU board is operational, and the system is operational if at least 1 computer is operational.

CPU boards fail at a rate of $1/10^6$ hours, and there is a 0.5% chance that a board failure will cause a computer failure, and a 0.8% chance that a board will fail in a way that causes a catastrophic system failure.

SAN *computer* for Computer Failure Model



Activity Case Probabilities and Input Gate Definition

Activity	*Case*	*Probability*
CPUfail1	*1*	*0.987*
	2	*0.005*
	3	*0.008*

Gate	*Definition*
Enabled1	Predicate *MARK(CPUboards1 > 0) && MARK(NumComp) > 0*
	Function *MARK(CPUboards1)– –;*

Output Gate Definitions

Gate	*Definition*
Covered1	Function *if (MARK(CPUboards1) == 0)* *MARK(NumComp)--;*
Uncovered1	Function *MARK(CPUboards1) = 0;* *MARK(NumComp)--;*
Catastrophic1	Function *MARK(CPUboards1) = 0;* *MARK(NumComp) = 0;*

Reward Variables

- Reward variables are a way of measuring performance- or dependability-related characteristics about a model.
- Examples:
  - Expected time until service
  - System availability
  - Number of misrouted packets in an interval of time
  - Processor utilization
  - Length of downtime
  - Operational cost
  - Module or system reliability
- Types:
  - A model may be in a certain state or states for some period of time, for example, "CPU idle" states. This is called a *rate reward*.
  - An activity may complete. This is called an *impulse reward*.

Reward Variables for Example

Reliability	
	<u>Rate rewards</u> *Subnet = computer* <u>Predicate:</u> *MARK(NumComp) > 0* <u>Function:</u> *1*
	<u>Impulse reward</u> *none*
NumBoardFailures	
	<u>Rate reward</u> *none*
	<u>Impulse reward</u> *Subnet = computer* activity = CPUfail1, value = 1 activity = CPUfail2, value = 1

Reward Variables for Example

Performability	
	<u>Rate rewards</u> *Subnet = computer* <u>Predicate:</u> *1* <u>Function:</u> *MARK(NumComp)*
	<u>Impulse reward</u> *none*
NumBoards	
	<u>Rate reward</u> *Subnet = computer* <u>Predicate:</u> *1* <u>Function:</u> *MARK(CPUBboards1) + MARK(CPUboards2)*
	<u>Impulse reward</u> *none*

Mobius Software

- Graphically builds and simulates SAN's
- Includes abstraction:
  - Models
    - These can be either atomic models or other composed models
    - Cycles are not allowed
  - Join nodes
    - Join nodes define a set of shared variables that are joined across the child models and nodes
    - Shared variables must all have the same type and initial value
  - Rep nodes
    - Rep nodes define a set of identical replicas and a set of shared variables across those replicas

Mobius Examples (v 2.5)

- Running an Existing Example:
  - Projects -> Unarchive (Multi-Proc)
  - Projects -> Open (Multi-Proc) , re-save if necessary
  - Documentation: https://www.mobius.illinois.edu/wiki/index.php/Fault-Tolerant_Multiprocessor_Model
- Note the rewards for tracking performance
- Note the study variable sweeps for the experiments
- Open either the simulation or transformer
- Collect the output (ASCII) and map it to a plot (Excel)

Duplicating Example in Presentation

- Projects -> New
- Double click Atomic
  - SAN Model, give it a name "combined_pc_model"
  - Replicate the system diagram
    - (Use Variable->Mark() instead of MARK(Variable))
  - Setup the reward
  - Create an experiment/study
  - Use a simulated solver

Markov Chains

- Markov process: Probability distribution for future state only depends on the present state, not on how the process arrived at the present state
- Markov chain: State space is discrete
- Discrete time Markov chain: Time is discrete
- $X_0, X_1, .., X_n, ..$:observed state at discrete times $t_0, t_1,..,t_n, ..$
- $X_n = j \Rightarrow$ system state at time step n is j.
- $P(X_n = i_n | X_0 = i_0, X_1 = i_1, \ldots, X_{n-1} = i_{n-1})$
  $= P(X_n = i_n | X_{n-1} = i_{n-1})$ (Markov Property)
- $p_{jk}(m,n) \equiv P(X_n = k \mid X_m = j),\ 0 \leq m \leq n$ (conditional pmf)
- $p_j(n) \equiv P(X_n = j)$ (unconditional pmf)

Homogeneous Markov Chain

- Homogeneous Markov chain: $p_{jk}(m,n)$ only depends on $n-m$
- n-step transition probability: $p_{jk}(n) = P(X_{m+n} = k \mid X_m = j)$
  - 1-step transition probability: $p_{jk} = p_{jk}(1) = P(X_n = k \mid X_{n-1} = j)$
- Initial probability row vector: $\mathbf{p}(0) = [p_0(0), p_1(0), \ldots, p_k(0), \ldots]$
- Transition probability matrix:

$$P = [p_{ij}] = \begin{bmatrix} p_{00} & p_{01} & p_{02} & \cdot & \cdot \\ p_{10} & p_{11} & p_{12} & \cdot & \cdot \\ \vdots & \vdots & \vdots & \cdot & \cdot \end{bmatrix}$$

Markov Chain Example

1. System is in state 0 if it is operational, state 1 if it is undergoing repair. Homogeneous DTMC process.

$$P = \begin{bmatrix} 1-a & a \\ b & 1-b \end{bmatrix}, 0 \le a, b \le 1$$

2. A sequence of successive software runs, each run has two possible outcomes: success or failure. X_n denotes outcome of n^{th} run (0 for success, 1 for failure). Occurrence of failure in a run depends on the outcome of previous run.

$$P(X_{n+1}=1 \mid X_n=0) = 1-p$$
$$P(X_{n+1}=1 \mid X_n=1) = q$$

Computation of n-step Transition Probabilities

- For a DTMC, find $p_{ij}(n) = P(X_{m+n} = j \mid X_m = i)$
- Events: State reaches *k* (from *i*) & reaches *j* (from *k*) are independent due to the Markov property (i.e. no history)
- Invoking the theorem of total probability:

$$p_{ij}(m+n) = \sum_k p_{ik}(m) p_{kj}(n)$$

- Let P(*n*) : *n*-step prob. transition matrix (*i,j*)th entry is $p_{ij}(n)$. Making *m=1, n=n-1* in the above equation:

$$P(n) = P.P(n-1) = P^n$$

$$p_j(n) = P(X_n = j) = \sum_i P(X_0 = i) P(X_n = j | X_0 = i) = \sum_i p_i(0) p_{ij}(n)$$

Computation of n-step Transition Probabilities

- The pmf of the random variable X_n is

$$p_j(n) = P(X_n = j) = \sum_i P(X_0 = i) P(X_n = j | X_0 = i) = \sum_i p_i(0) p_{ij}(n)$$

- *j,* in general can assume countable values, 0,1,2, …. Defining,

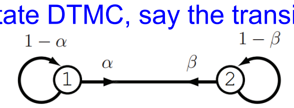
$$\mathbf{p}(n) = [p_0(n), p_1(n), \ldots, p_j(n), \ldots]$$

- $p_j(n)$ for *j=0,1,2,..,k,…* can be written in the vector form as, $[p_0\ p_1\ p_2\ \ldots\ p_j\ \ldots](n) =$

$$\underbrace{\begin{bmatrix} p_0 & p_1 & p_2 & \cdots & p_j & \cdots \end{bmatrix}(0)}_{\mathbf{p(0)}} \begin{bmatrix} p_{00}(n) & p_{01}(n) & \ldots p_{0j}(n) & \ldots \\ p_{10}(n) & p_{11}(n) & \ldots p_{1j}(n) & \ldots \\ p_{20}(n) & p_{21}(n) & \ldots p_{2j}(n) & \ldots \\ \vdots & \vdots & \vdots & \vdots \\ p_{i0}(n) & p_{i1}(n) & \ldots p_{ij}(n) & \ldots \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

Two state Markov chain example

- For a two state DTMC, say the transition prob. matrix is



$$P = \begin{pmatrix} 1-\alpha & \alpha \\ \beta & 1-\beta \end{pmatrix}$$

The eigenvalues are 1 and $1-\alpha-\beta$. So we can write

$$P = U\begin{pmatrix} 1 & 0 \\ 0 & 1-\alpha-\beta \end{pmatrix} U^{-1} \implies P^n = U\begin{pmatrix} 1 & 0 \\ 0 & (1-\alpha-\beta)^n \end{pmatrix} U^{-1}$$

So $p_{11}^{(n)} = A + B(1-\alpha-\beta)^n$ for some A and B.

But $p_{11}^{(0)} = 1 = A+B$ and $p_{11}^{(1)} = 1-\alpha = A+B(1-\alpha-\beta)$. So $(A,B) = (\beta,\alpha)/(\alpha+\beta)$, i.e.

$$P_1(X_n = 1) = p_{11}^{(n)} = \frac{\beta}{\alpha+\beta} + \frac{\alpha}{\alpha+\beta}(1-\alpha-\beta)^n.$$

Note that this tends exponentially fast to a limit of $\beta/(\alpha+\beta)$.

Other components of P^n can be computed similarly, and we have

$$P^n = \begin{pmatrix} \frac{\beta}{\alpha+\beta} + \frac{\alpha}{\alpha+\beta}(1-\alpha-\beta)^n & \frac{\alpha}{\alpha+\beta} - \frac{\alpha}{\alpha+\beta}(1-\alpha-\beta)^n \\ \frac{\beta}{\alpha+\beta} - \frac{\beta}{\alpha+\beta}(1-\alpha-\beta)^n & \frac{\alpha}{\alpha+\beta} + \frac{\beta}{\alpha+\beta}(1-\alpha-\beta)^n \end{pmatrix}$$

Source: http://www.statslab.cam.ac.uk/~rrw1/markov/M.pdf

Two state Markov chain example

- Then, the n-step transition probability matrix *P(n)* = P^n is

$$P^n = \begin{pmatrix} \frac{\beta}{\alpha+\beta} + \frac{\alpha}{\alpha+\beta}(1-\alpha-\beta)^n & \frac{\alpha}{\alpha+\beta} - \frac{\alpha}{\alpha+\beta}(1-\alpha-\beta)^n \\ \frac{\beta}{\alpha+\beta} - \frac{\beta}{\alpha+\beta}(1-\alpha-\beta)^n & \frac{\alpha}{\alpha+\beta} + \frac{\beta}{\alpha+\beta}(1-\alpha-\beta)^n \end{pmatrix}$$

- Example: Consider a communication network consisting of a sequence of binary communication channels, with a=1/4, b=1/2. Here X_n denotes the bit leaving the nth stage and X_o represents the bit entering the system.

1. If a bit enters as 1, what is the probability of being correctly transmitted over (a) two stages (b) three stages?
2. Assuming initial probabilities P(X_0=0) = 1/3 and P(X_0=1) = 2/3, what is **p**(n)?