CSCE 5760: Design For Fault Tolerance

Review:
- Reliability, Mean Time To Failure
- Availability, Mean Time for Repair
- Fault Models
  - What type of faults do we expect in Hardware
    - stuck at faults
  - Software: logic fault, range errors...
- Need a framework so that we can test for the faults
- Test sets, coverage of test sets
- Minimal test sets

Can be used to estimate reliability

Performability: when some components fail, the system may still operate but at a degraded performance.
- Different levels of acceptable performance
- Probability with each level

For example, if the performance is proportional to # processors, we can define the expected level of performance as

\[ C = \sum_{j=1}^{N} C_j P_j(t) \]

A canonical (or standard) structure is constructed out of N individual modules.

The basic canonical structures are:
- A series system
- A parallel system
- A mixed system

We will assume statistical independence between failures in the individual modules.

Reliability of a Series System

A series system - set of modules so that the failure of any one module causes the entire system to fail

\[ R_s(t) = \prod_{i=1}^{N} R_i(t) \]

\[ R_s(t) = e^{-\lambda_s t} = e^{-\sum \lambda_i t} \]

\[ MTTF_s = \frac{1}{\lambda_s} = \frac{1}{\sum \lambda_i} \]
CSCE 5760: Design For Fault Tolerance

Reliability of a Series System

Reliability of a series system - $R_s(t)$ - product of reliabilities of its $N$ modules

$$R_s(t) = \prod_{i=1}^{N} R_i(t)$$

$R_i(t)$ is the reliability of module $i$

Series System – Modules Have Constant Failure Rates

- Every module $i$ has a constant failure rate $\lambda_i$
  $$R_i(t) = e^{-\lambda_i t}$$
  $$R_s(t) = e^{-\sum_{i} \lambda_i t}$$

CSCE5760 September 10, 2019

CSCE 5760: Design For Fault Tolerance

Reliability of a Parallel System

- A Parallel System - a set of modules connected so that all the modules must fail before the system fails

$$R_p(t) = 1 - \prod_{i=1}^{N} [1 - R_i(t)]$$

$$R_p(t) = 1 - \prod_{i=1}^{N} [1 - e^{-\lambda_i t}]$$

$$MTTF_p = \sum_{i=1}^{N} \frac{1}{i \lambda}$$

CSCE5760 September 10, 2019
CSCE 5760: Design For Fault Tolerance

Non Series/Parallel Systems

(a): When C is working

\[ R_{\text{system}} = R_C [R_E (R_A + R_B) + (1 - R_C)] + (1 - R_C) R_A R_D R_F \]

(b): When C is not working

If \( R_A = R_B = R_C = R_D = R_E = R_F \)

\[ R_{\text{system}} = R^3 (R^3 - 3R^2 + R + 2) \]

If structure is too complicated - derive upper and lower bounds on \( R_{\text{system}} \)

An upper bound - \( R_{\text{system}} \leq 1 - \prod (1 - R_{\text{path}_i}) \)

\( R_{\text{path}_i} \) - reliability of modules in series along path \( i \)

Assuming all paths are in parallel

The lower bound is \( \rightarrow \) we define cut sets

\( R_{\text{system}} \geq \prod (1 - Q_{\text{cut}_i}) \)

\( Q_{\text{cut}_i} \) - probability that the minimal cut \( i \) is faulty (i.e., all its modules in the cut set are faulty)
CSCE 5760: Design For Fault Tolerance

Redundancy And Resiliency: Generic M-of-N Systems

An M-of-N system consists of N identical modules

Fails when fewer than M modules are functional

Best-known example - The Triplex (TMR)

Three identical modules whose outputs are voted on

This is a 2-of-3 system: as long as a majority of the processors produce correct results, the system will be functional

Reliability of M-of-N Systems

N identical modules

\( R(t) \) - reliability of an individual module

The reliability of the system is the probability that \( N - M \) or fewer modules have failed by time \( t \) (or at least \( M \) are functional)

\[
R_{m\text{-of} \text{-}n}(t) = \sum_{i=0}^{N-M} \binom{N}{i} (1 - R(t))^i R(t)^{N-i}
\]

CSCE5760 September 10, 2019

Reliability of M-of-N Systems

\[
R_{m\text{-of} \text{-}n}(t) = \sum_{i=0}^{N-M} \binom{N}{i} (1 - R(t))^i R(t)^{N-i}
\]

\[
= \sum_{i=M}^{N} \binom{N}{i} R(t)^i (1 - R(t))^{N-i}
\]

Where

\[
\binom{N}{i} = \frac{N!}{i!(N-i)!}
\]

Can we assume that all modules behave independently?

Is it possible to have correlated failures?

The same design failure may exist in several replicated components?
A common NMR (M out of N) system is the Triple Modular redundancy (TMR) system. Here N=3 and M=2 (2 out of 3). A voter picks the majority output.

Voter can fail - reliability of voter $R_{vot}(t)$

$$R_{TMR} = R_1 R_2 (1-R_3) + R_1 R_3 (1-R_2) + R_2 R_3 (1-R_1) + R_1 R_2 R_3$$

$$= 3R^2(1-R) + R^3 = 3R^2 - R^3$$

We can also derive this as $1 - \text{(failure probability)} = 1 - [\text{prob}(3 \text{ failing}) + \text{prob}(2 \text{ failing})]$ = $1 - [(1-R)^3 + 3(1-R)^2 R] = 3R^2 - R^3$

Reliability of TMR - Constant Failure Rates → as a probability distribution $R(t) = e^{-\lambda t}$

Assuming no voter failures - $R_{vot}(t) = 1$

$$R_{TMR}(t) = 3e^{-2\lambda t} - 2e^{-3\lambda t}$$

$$MTTF_{TMR} = \int_0^\infty R_{TMR}(t) \cdot dt = \frac{5}{6\lambda} < \frac{1}{\lambda} = MTTF_{simplex}$$

Not unexpected, the reliability is higher than a single unit for smaller time intervals [0,t]. But, the reliability is lower than a single unit in the longer run since there are more units that could fail. The $MTTF$ is lower (that is the life time of a TMR is less than a single unit). Also if the reliability of a single unit is low, TMR may not be beneficial.
CSCE 5760: Design For Fault Tolerance

Let us compute 5MR (3 out of 5 or majority)
We have to consider
(all working) + (4 working)(one failed) + (3 working)(2 failed)

\[ R_{5MR} = R_1 R_2 R_4 R_5 + (R_1 R_2 R_3 R_4 R_5 (1-R_5) + R_1 R_2 R_4 R_5 (1-R_4) + R_1 R_2 R_5 R_4 (1-R_3) + R_1 R_3 R_4 R_5 (1-R_2) + R_2 R_3 R_4 R_5 (1-R_1)) 
\]

\[ = 6R^3(1-R) + 5R^4(1-R) + R^5 \]

Below R=0.5 - redundancy becomes a disadvantage
Usually R >> 0.5 - triplex offers significant reliability gains

CSCE 5760: Design For Fault Tolerance

Correlated failure can greatly diminish reliability

Example: Let \( q_{cor} \) be the probability that the entire system suffers a global failure

\[ R_{m-of-n-corr}(t) = (1 - q_{cor}) \sum_{i=M}^{N} \binom{N}{i} R(t)^i (1 - R(t))^{N-i} \]

- If system is not designed carefully, the correlated failure factor can dominate the overall failure probability
- Different modes of correlation among modules exist - not necessarily a global failure
- Correlated failure rates are extremely difficult to estimate
- From now on we will assume statistically independent failures in modules
A voter receives inputs $X_1, X_2, \ldots, X_N$ from an $M$ of $N$ cluster and generates a representative output. Simplest voter - bit-by-bit comparison of the outputs producing the majority vote or exact match.

This only works when all functional processors generate outputs that match bit by bit. Processors must be identical, be synchronized and use the same software. Otherwise, two correct outputs can diverge slightly, in the lower significant bits.

To overcome this problem, we can use **Plurality Voting**

We declare two outputs $X$ and $Y$ as practically identical if $|x - y| < \delta$ for some specified $\delta$.

A k-plurality voter looks for a set of at least $k$ practically identical outputs, and picks any of them (or their median) as the representative.

Example - $\delta = 0.1$, five outputs $1.10, 1.11, 1.32, 1.49, 3.00$. The subset $\{1.10, 1.11\}$ would be selected by a 2-plurality voter.

If a voter can fail, we can introduce redundant voters. We can do this at each stage of the computation/system. Here we a voter voting on other voters.

How to computing reliability? We need two modules and two voters plus final voter working.

**Unit-level Modular Redundancy**

Voters no longer critical - a single faulty voter is no worse than a single faulty unit. The level of replication and voting can be lowered using additional voters - increasing the size and delay of the system.
Triplicated Processor/Memory System

All communications (in either direction) between triplicated processors and triplicated memories go through majority voting

Higher reliability than a single majority voting of triplicated processor/memory structure

We can explore other topologies and configurations to improve reliability

---

Active/Dynamic Redundancy

In previous examples - considerable extra hardware used to instantaneously mask errors

In many cases, temporary erroneous results may be acceptable if the system can detect an error, replace the faulty module by a fault-free spare, or reconfigure itself

This is called dynamic (or active) redundancy

Example:

How do we analyze the reliability of such a system?
How do we analyze the reliability of such a system?

All N spare modules are active (powered) and have the same failure rate – resulting in a basic parallel system with N+1 modules.

System reliability is:

\[ R_{\text{dynamic}}(t) = R_{\text{dru}}(t)[1 - (1 - R(t))^{N+1}] \]

- \( R(t) \): reliability of module
- \( R_{\text{dru}}(t) \): reliability of Detection & Reconfiguration unit

Dynamic Redundancy with Unpowered (Standby) Spares

Spare modules are not powered (e.g., to conserve energy) and cannot fail until they become active.

C - coverage factor - probability that faulty active module is correctly diagnosed and disconnected, and good spare successfully connected.

Calculating exact reliability for the general case is complicated.

Reliability for a special case:
- Very large N (never run out of spares)
- Constant failure rate \( \lambda \) per active module
- Failure rate of nonrecoverable faults is \((1-C)\lambda\).

Reliability at time \( t \) - probability of no nonrecoverable faults up to time \( t \):

\[ R_{\text{dynamic}}(t) = R_{\text{dru}}(t)e^{-(1-C)\lambda t} \]
Hybrid Redundancy

NMR masks permanent and intermittent failures but its reliability drops below that of a single module for very long mission times. Hybrid redundancy overcomes this by adding spare modules to replace active modules once they become faulty.

A hybrid system consists of a core of N processors (NMR), and K spares.

Reliability of a hybrid system with a TMR core and K spares is:

\[ R_{\text{hybrid}}(t) = R_{\text{vot}}(t)R_{\text{rec}}(t)[1 - mR(t)(1 - R(t))^{m-1} - (1 - R(t))^m] \]

\( m = K + 3 \) - total number of modules

\( R_{\text{vot}}(t) \) and \( R_{\text{rec}}(t) \) - reliability of voter and comparison & reconfiguration circuitry

Assuming: any fault in voter or comparison & reconfiguration circuit will cause a system fault.

In practice, not all faults in these circuits will be fatal: the reliability will be higher.
CSCE 5760: Design For Fault Tolerance

Sift-Out Modular Redundancy

- Like NMR all N modules are active but simpler than hybrid redundancy
- Comparing output of each module to outputs of other still operational modules
  - A module whose output disagrees with other is switched out

- Sift-out should not be too aggressive - most failures are transient
- Purge a module only if it produces incorrect outputs over a sustained period of time

![Diagram of modular redundancy system]

CSCE 5760: Design For Fault Tolerance

Duplex systems

Replicated but we need to be able to identify failed unit

Both processors execute the same task
- If outputs are in agreement - result is assumed to be correct
- If results are different - we can not identify the failed processor
- A higher-level software has to decide how failure is to be handled
- This can be done using one of several methods

Reliability

Two active identical processors with reliability R(t)

Lifetime of duplex - time until both processors fail

\[ R_{\text{duplex}}(t) = R_{\text{comparator}}(t) \cdot [R^2(t) + 2C R(t)(1-R(t))] \]

C - Coverage Factor - probability that a faulty processor will be correctly diagnosed, identified and disconnected
CSCE 5760: Design For Fault Tolerance

Duplex - Constant Failure Rates

- Each processor has a constant failure rate $\lambda$.
- Ideal comparator - $R_{\text{comp}}(t)=1$
- Duplex reliability:
  \[ R_{\text{duplex}}(t) = e^{-2\lambda t} + 2Ce^{-\lambda t}(1-e^{-\lambda t}) \]
  \[ \text{MTTF}_{\text{duplex}} = \frac{1}{2\lambda} + \frac{C}{\lambda}. \]

Pair and Spare

- Avoid disruption of operation upon a mismatch between the two modules in a duplex.
- Disconnect duplex and transfer task to spare pair.
- Test offline, and if fault is transient - mark duplex as a good spare.

We can use duplex with TMR: if a pair disagree, switch out both. Reliability? In TMR, replace $R$ with $R_{\text{duplex}}$.

CSCE 5760: Design For Fault Tolerance

For most reliability models, including Markov processes (continuous) or Markov Chains (discrete), we assume that the random variable(s) behave like a Poisson Process.

So, what is a Poisson Process?

Non-deterministic events (i.e., random variable outcomes) of some kind occurring over time with the following probabilistic behavior:

For some constant $\lambda$ and a very short interval of length $\Delta t$:

1. Probability of one event occurring during $\Delta t$ is $\lambda\Delta t$ plus a negligible term.
   That is, the probability of something happening in a small interval is directly proportional to the time interval and failure rate.

2. Probability of more than one event occurring during $\Delta t$ is negligible.

3. Probability of no events occurring during $\Delta t$ is $1-\lambda\Delta t$ plus a negligible term.
CSCE 5760: Design For Fault Tolerance

We view the random variable as the number of the events $N(t)$ occurring during $[0,t]$.

$$P_k(t) = \text{Prob}(N(t) = k)$$

- probability of $k$ events occurring during a time period of length $t$ ($k = 0, 1, 2, \ldots$)

$$P_k(t + \Delta t) = P_k(t)(1 - \lambda \Delta t) + P_{k-1}(t)\lambda \Delta t$$

That is: $k$ events happened by time $t$, and no new event happened during $[t, t+\Delta t]$

Or $k-1$ events happened by time $t$, and one more event happened during $[t, t+\Delta t]$

$$P_0(t + \Delta t) \approx P_0(t)(1 - \lambda \Delta t) \quad \text{Probability of zero events}$$

This results in the following differential equations:

$$\frac{dP_k(t)}{dt} = -\lambda P_k(t) + \lambda P_{k-1}(t)$$

$$\frac{dP_0(t)}{dt} = -\lambda P_0(t)$$

With initial condition $P_k(0) = 0$ (for $k \geq 1$) and $P_0(0) = 1$

---

CSCE 5760: Design For Fault Tolerance

The solution to the differential equation (for $k = 0, 1, 2, \ldots$) gives us

$$P_k(t) = e^{-\lambda t}(\lambda t)^k/k!$$

For all values of $t$, $N(t)$ is a Poisson process with rate $\lambda$.

We can think of as the average number of events in a unit time.

For example: number of transactions or tasks arriving at a computer.

Number of customers arriving at a bank teller.

Number of failures occurring in a unit time.

Some properties of a Poisson process:

- For a Poisson process with rate $\lambda$:
  - Expected number (or mean) of events in an interval of length $t$ is $\lambda t$.
  - Length of time between consecutive events has an exponential distribution with parameter $\lambda$, and mean $1/\lambda$.
  - Numbers of events in disjoint intervals are statistically independent.

- Sum of two Poisson processes with parameters $\lambda_1$ and $\lambda_2$ is a Poisson process with parameter $\lambda_1 + \lambda_2$. 
Example uses of Poisson to compute reliability
Consider the Duplex system with two units and a comparator

We have unlimited spares, so that the comparator detects failures and replaces failed units with spares

Spare do not fail (not powered)

Each processor has a constant failure rate $\lambda$.

Lifetime of a processor - Exponential distribution with parameter $\lambda$.

Time between two consecutive failures of same logical processor - Exponentially distributed with a parameter $\lambda$.

$N(t)$ - number of failures in one logical processor during $[0,t]$

$M(t)$ - number of failures in the duplex system during $[0,t]$

Reliability model for duplex using Poisson Process

Duplex has two processors - failure rate is $2\lambda$.

Comparator failure rate - negligible

Probability of $k$ failures in duplex in $[0,t]$ -

$$\Pr(\text{failures}) = e^{-2\lambda t} \left( \frac{2\lambda t}{k!} \right)^k$$

(for $k=0,1,2,\ldots$)

For the duplex not to fail, each of these failures must be detected and successfully replaced - probability $C$

For $k$ failures - probability

$$R_{\text{duplex}}(t) = \sum_{k=0}^{\infty} \Pr(\text{failures}) C^k = \sum_{k=0}^{\infty} e^{-2\lambda t} \left( \frac{2\lambda t}{k!} \right)^k C^k$$

$$= e^{-2\lambda t} \sum_{k=0}^{\infty} \left( \frac{2\lambda t C}{k!} \right)^k = e^{-2\lambda t} e^{2\lambda t C} = e^{-2\lambda (1-C)t}$$
Markov processes – are based on exponential distribution
memory less property
Future behavior is dependent on the current “state”
and on not past history

Discrete processes are called Markov Chains and continuous processes are called Markov Processes

A formal definition of Markov Chain
A stochastic process \(X(t)\) is Markov if for any \(t_1 < t_2 < \ldots < t_n < t\) and any set \(\{A, A_1, A_2, \ldots, A_n\}\) of states

\[
\text{Prob}\{X(t) - A \mid X(t_1) = A_1, X(t_2) = A_2, \ldots, X(t_n) = A_n\} = \text{Prob}(X(t) = A \mid X(t_n) = A_n)
\]

What are we saying?

The probability of being in a specific state in the future depends only the current state
\(\rightarrow\) not previous states \(\rightarrow\) memory less property

Discrete Markov Chains. We assume that the process changes state only at the end of a discrete time step (or we only observe the process at discrete time intervals)

Let us assume \(S = \{S_1, S_2, \ldots, S_n\}\) represent the set of states
At sometime, assume that the process is in state \(S_i\)
At the next time of observation the process may transition to \(S_j\)
Note \(S_j\) may be \(S_i\)
The probability of transition from state \(S_i\) to \(S_j\) is \(P_{ij}\)

\[
\sum_j P_{ij} = 1 \quad \rightarrow \text{the process must transition to some state}
\]

Consider a simple example: A computer is shared by 2 users.
At any minute (that is our time scale) a user may disconnect with a probability of 0.5; and the user may reconnect at the next time unit with a probability of 0.2.

Can we describe this as a Markov chain (we will use discrete time)?
How to define a state with such a Markov process?

Number of users on the computer: \( \{0, 1, 2\} \)

Now we need to describe transition probabilities

```
 0 1 2
0 0.64 0.32 0.04
1 0.4 0.5 0.1
2 0.25 0.5 0.25
```

How did we get these transition probabilities?
Easy to see from 2 to 1 (one user disconnects)
What about the others?
We need to assume that the users are independent with same probability

So, from 0 – 1, one user connects while the other remains disconnected

```
P_{00} = 0.8^2 = 0.64
P_{01} = 0.2*0.8+0.2*0.8 = 0.32
P_{02} = 0.2*0.2 = 0.04
P_{10} = 0.8*0.5 = 0.4
P_{11} = 0.8*0.5+0.2*0.5 = 0.5
P_{12} = 0.5*0.2 = 0.1
P_{20} = 0.5*0.5 = 0.25
P_{21} = 0.5
P_{22} = 0.5*0.25 = 0.25
```

In such systems (called irreducible), what we are interested in is finding the steady state probability.

What is the probability that in the long run the computer is idle
That is probability of having zero users \( (P_0) \)

or Probability of having one and probability of having 2 two uses

Note the difference: the matrix above shows the probability of transitioning from state \( S_i \) to \( S_j \)
Steady state probability is the probability of finding the system in a state
CSCE 5760: Design For Fault Tolerance

Two types of Markov processes (or chains)

With “Absorbing” states. Once you reach these states, you cannot get out
Without Absorbing states – you can move between all states

And we use different approaches to analyze the two type of Markov processes

We can classify states into

Absorbing states. A state \( s_i \) is absorbing if \( p_{ij} = 0 \) for any \( j \neq i \) and \( p_{ii} = 1 \). 

you remain in the same state and not transition out

For example, failed state is an absorbing state. Once we get into this state, we will not get out of the state.

Recurrent states. A state \( s_i \) is recurrent if, there is a probability of 1 that when the system leaves the state \( s_i \) that the system will reenter the state (eventually).

Transient State. A state \( s_i \) is transient, that if the probability that the system will return to the state (after it leaves the state) is less than 1. That is, there is a finite probability that it will not return to state \( s_i \) eventually.

If we represent the Markov process as a directed graph (edges represent probability of transitions greater than 0.0); then

Absorbing states are those vertices with no outgoing paths;
Recurrent states are those that form a clique;
Transient represent collective states with no incoming edges
-- from other states.

Subclasses of Recurrent states -- and equivalent classes. That is if state \( i \) is a recurrent state, and state \( i \) communicates with state \( j \), then state \( j \) is also recurrent state.

Null recurrent: The mean time to return to a recurrent state is infinite.

Non-null recurrent if the mean time to recur is finite.

Periodic recurrent: A state recurs periodically after certain number of transitions.
Aperiodic recurrent means non periodic. So, if there is a probability that state \( i \) recurs after \( k \) steps, there is a probability that the state \( i \) recurs after \( k+1, k+2, .. \) steps also.
CSCE 5760: Design For Fault Tolerance

If a Markov process/chain only contains recurrent states then we call the Markov process/chain **irreducible**

If it contains transient or absorbing, we call it **reducible**

We can reduce the process into groups of states

And in the steady state we have *zero probability* of being in a transient state and probability of 1 being in an absorbing state

Our example with 2 users on a computer is an irreducible process since we only have recurrent states

![Markov chain example]

CSCE 5760: Design For Fault Tolerance

Let us look at irreducible Markov chains (no absorbing states)

Let us say $P$ is the state transition matrix. What does $P^n$ mean?

$P \times P \rightarrow p[j,j]$ = probability of going from state $s_i$ to state $s_j$ in exactly 2 time units

Likewise, $P^n$ gives us the probabilities of going from one state to another in 3 steps

And so on.

We can also find $\sum_{j=1}^{n} P^n$ which gives us the probability of going from one state to another in 1,2,...n steps

Steady State: The long term behavior

If the system reaches steady state say after $n$ steps, then the probability of going from state $i$ to state $j$ in $n+1$ steps should be same as the probability of going from $i$ to $j$ in $n$ steps.

We can use the ideas of steady state to find the probability that the system can be found in any given state during steady state behavior.
CSCE 5760: Design For Fault Tolerance

Let us consider the probability that the system is in state $p_i$ in the steady state.
We can obtain these steady state probabilities as follows (called Eigen value solution):

$$(p_1, p_2, \ldots, p_n) P = (p_1, p_2, \ldots, p_n)$$

$P$ is the $n \times n$ state transition probability matrix.
This only gives $n-1$ equations with $n$ unknowns.
But we also know that $(p_1 + p_2 + \ldots + p_n) = 1$

Coming back to our example of two users on a computer, let us find the steady state probabilities using the above formula:

$$P = \begin{bmatrix}
0 & 0.64 & 0.32 & 0.04 \\
1 & 0.4 & 0.5 & 0.1 \\
2 & 0.25 & 0.5 & 0.25
\end{bmatrix}$$

We have to solve

$$(P_0, P_1, P_2) \times P = (P_0, P_1, P_2)$$

Given $P_0 + P_1 + P_2 = 1$

Steady state probabilities:

$P_0 = 0.509$; $P_1 = 0.408$; $P_2 = 0.082$

So, there is a 50.9% probability that the computer is idle!

Another example, given the following probability transition matrix:

We have 5 states. How do we know that this is an irreducible chain?

We need to make sure that every state is reachable from every other state.

To find reachability, we can compute

$$\sum_{i=1}^{n} p_{ij}$$

And then make sure that all numbers in the matrix > 0.
Steady state probabilities can be obtained from

\[(p_1, p_2, p_3, p_4, p_5) P = (p_1, p_2, p_3, p_4, p_5)\]

\[p_1 = 0.21554; p_2 = 0.380389, p_3 = 0.190194; p_4 = 0.116653; p_5 = 0.09721\]

Let us use a different example.

Kavi is trying to lose weight and has the following behavior in terms of lunch.

If he eats a heavy lunch one day, he is likely to eat light lunch next day with 60% probability (and heavy lunch with a probability of 40%).

But if he eats a light lunch one day, he is likely to each a light lunch next day with 50% probability.

So in the steady state, on any given day what is the probability he will eat a light lunch?

How do we model this?

This is a discrete time Markov chain where our time step is 1 day

We can develop models of market share

Say a customer who shopped at Target is 0.5 likely to return to Target next week but is likely to go to Wallmart with 0.25 probability to K-Mart with 0.25 probability

And so on

We can then obtain steady state probabilities of a customer shopping at each of these places

Let us consider systems where a steady state means just a failed state

<table>
<thead>
<tr>
<th></th>
<th>HL</th>
<th>LL</th>
</tr>
</thead>
<tbody>
<tr>
<td>HL</td>
<td>0.4</td>
<td>0.6</td>
</tr>
<tr>
<td>LL</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

This is the state transition probability matrix

\[(P_{HL}, P_{LL})P = (P_{HL}, P_{LL})\] and \[(P_{HL} + P_{LL}) = 1\]

Solving we get \(P_{HL} = 54.55\%\) and \(P_{LL} = 45.45\%\)

In the steady state, on any day the probability that Kavi eats heavy lunch is 54.5\% ✓ no chance of losing weight

We can develop models of market share

Say a customer who shopped at Target is 0.5 likely to return to Target next week but is likely to go to Wallmart with 0.25 probability to K-Mart with 0.25 probability

And so on

We can then obtain steady state probabilities of a customer shopping at each of these places

Let us consider systems where a steady state means just a failed state
3W, 2W, 1W, F

We have absorbing states
Namely, Failed states

What if we need at least two working States: 3W, 2W and F
Noe once a unit failed it remains failed (NO REPAIR)
So, we have absorbing states

Another example

Let us consider reducible Markov chains (with absorbing states)
And find probabilities associated with absorbing failures
And in turn obtain reliabilities
We are dealing with discrete time Markov chains
CSCE 5760: Design For Fault Tolerance

Consider the following example of a discrete Markov Chain:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0.25</td>
<td>0.25</td>
<td>0.25</td>
<td>0</td>
<td>0</td>
<td>0.25</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1.00</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

States 6-7 and States 3-4-5 form recurrent states.
3-4-5 are periodic (no self loops so must leave the state).
States 1-2 are transient – once they leave these states, no path back.

Reducible vs Irreducible Markov Chains.

A Markov chain is irreducible if every state is reachable from other states. In terms of graphs, “strongly connected” (no absorbing or transient states).

If we have transient states, then the Markov chain is reducible.

Canonical Representation of the reducible Markov processes

We renumber states: All absorbing states are numbered first (lower numbers).
Then we number recurrent states.
Then we number transient states.
Then our probability transition matrix looks like:

\[
P = \begin{bmatrix}
P_{1} & 0 \\ R & Q \\
\end{bmatrix}
\]

We will see how we can use this representation to determine how long (how many steps) it takes reach an absorbing state.
Consider our previous example

Here we have states 1 and 2 as transient → will not return to these states
States (3, 4, 5), and (6, 7) as recurrent

We cannot use the method we used to find steady state probabilities for systems with only recurrent states (irreducible)

In the steady state
The probability of being in one of the absorbing state (at steady state) is 1
And the probability of being in a transient state is zero

But this is what we need to model failures (absorbing states)
Remember how we renumbered states in a canonical form giving the transient matrix

Now let us define a matrix called Fundamental matrix

\[ M = (I - Q)^{-1} \]
(take the inverse of \( I - Q \))

The elements of \( M \) (of \( m_{ij} \)) will give the average (or expected) number of visits to state \( j \) if we started in state \( i \), before entering an absorbing or recurrent state.

If we sum the values of each row, \( M_i \) – the row \( i \) gives the average number of visits to other transient states (including \( i \) itself) before the system reaches a recurrent or absorbing state
CSCE 5760: Design For Fault Tolerance

Note \( m_i \) or \( M_i \) do not indicate which absorbing or recurrent state will be reached.

Let us define \( F = M \cdot R \) (\( M \) is the fundamental matrix).

Now the values in \( F \) give you the probability of reaching a specific absorbing or recurrent state.

\[
P_1 \begin{array}{c}
0 \\
R \\
Q
\end{array}
\]

Example: Consider a simple model for passage of a phosphorus molecule through a pasture ecosystem.

We will use only 4 states for the molecule: in the soil (Soil), as grass grows, moves from soil to grass (Grass), the grass eaten by cattle and the molecule moves into the cattle (Cattle), and the products of the cattle (milk, beef) carry the molecule out of the pasture system (Out).

Out is an absorbing state.

\[
\begin{array}{cccc}
S & G & C & O \\
S & 3/5 & 3/10 & 0 & 1/10 \\
G & 1/10 & 2/5 & 1/5 & 0 \\
C & 3/4 & 0 & 1/5 & 1/20 \\
O & 0 & 0 & 0 & 1 \\
\end{array}
\]

\[
Q = \begin{bmatrix}
3/5 & 3/10 & 0 \\
1/10 & 2/5 & 1/2 \\
3/4 & 0 & 1/5 \\
\end{bmatrix}
\]

\[
I - Q = \begin{bmatrix}
2/5 & -3/10 & 0 \\
-1/10 & 3/5 & -1/2 \\
-3/4 & 0 & 4/5 \\
\end{bmatrix}
\]

\[
M = (I - Q)^{-1} = \begin{bmatrix}
8.6 & 4.3 & 2.7 \\
8.2 & 5.8 & 3.6 \\
8.1 & 4.1 & 3.8 \\
\end{bmatrix}
\]

If you added rows we get:

\[
\begin{array}{cccc}
S & G & C \\
S & 15.6 & 17.6 & 16.0 \\
\end{array}
\]
CSCE 5760: Design For Fault Tolerance

If we assumed the unit of time is a day, then the molecule stays in soil for 15.6 days (if we started there) before leaving the system.

Likewise it stays 17.6 in grass and 16 days in the cattle

\[
F = \begin{bmatrix}
0.995 \\
1.0 \\
1.0
\end{bmatrix}
\]

The probability of the phosphorus leaving the system is 0.995 if it started initially in the soil.

The probability of leaving the system is 100%, if the molecule started in either the Grass or in the Cattle.

We can use similar formulation to find MTTF. Assume that we use a unit time (hour, day or week) in defining transition probabilities.