Advanced Topics in Software Engineering: Markov Chains

Prof. Michele Loreti

Advanced Topics in Software Engineering
Corso di Laurea in Informatica (L31)
Scuola di Scienze e Tecnologie

Stochastic Process

- Formally, a stochastic model is one represented as a stochastic process;

Stochastic Process

- Formally, a stochastic model is one represented as a stochastic process;
- A stochastic process is a set of random variables $\{X(t), t \in T\}$.

Stochastic Process

- Formally, a stochastic model is one represented as a stochastic process;
- A stochastic process is a set of random variables $\{X(t), t \in T\}$.
- T is called the index set usually taken to represent time.

Stochastic Process

- Formally, a stochastic model is one represented as a stochastic process;
- A stochastic process is a set of random variables $\{X(t), t \in T\}$.
- T is called the index set usually taken to represent time.
- Since we consider continuous time models $T = \mathbb{R}^{\geq 0}$, the set of non-negative real numbers.

State Space

The state space of a stochastic process is the set of all possible values that the random variables $X(t)$ can assume.

State Space

The state space of a stochastic process is the set of all possible values that the random variables $X(t)$ can assume.

Each of these values is called a state of the process.

State Space

The state space of a stochastic process is the set of all possible values that the random variables $X(t)$ can assume.

Each of these values is called a state of the process.

Any set of instances of $\{X(t), t \in T\}$ can be regarded as a path of a particle moving randomly in the state space, S, its position at time t being $X(t)$.

State Space

The state space of a stochastic process is the set of all possible values that the random variables $X(t)$ can assume.

Each of these values is called a state of the process.

Any set of instances of $\{X(t), t \in T\}$ can be regarded as a path of a particle moving randomly in the state space, S, its position at time t being $X(t)$.

These paths are called sample paths or realisations of the stochastic process.

Properties of Stochastic Processes

In this course we will focus on stochastic processes with the following properties:

Properties of Stochastic Processes

In this course we will focus on stochastic processes with the following properties:

$\{X(t)\}$ is a Markov process.
This implies that $\{X(t)\}$ has the Markov or memoryless property: given the value of $X(t)$ at some time $t \in T$, the future path $X(s)$ for $s > t$ does not depend on knowledge of the past history $X(u)$ for $u < t$, i.e. for $t_1 < \cdots < t_n < t_{n+1}$,

$$\Pr(X(t_{n+1}) = x_{n+1} \mid X(t_n) = x_n, \ldots, X(t_1) = x_1) = \Pr(X(t_{n+1}) = x_{n+1} \mid X(t_n) = x_n)$$

Properties of Stochastic Processes

In this course we will focus on stochastic processes with the following properties:

$\{X(t)\}$ **is irreducible**.

This implies that all states in S can be reached from all other states, by following the transitions of the process. If we draw a directed graph of the state space with a node for each state and an arc for each event, or transition, then for any pair of nodes there is a path connecting them, i.e. the graph is strongly connected.

Properties of Stochastic Processes

In this course we will focus on stochastic processes with the following properties:

$\{X(t)\}$ **is stationary**:
for any $t_1, \ldots t_n \in T$ and $t_1 + \tau, \ldots, t_n + \tau \in T$ $(n \geq 1)$, then the process's joint distributions are unaffected by the change in the time axis and so,

$$F_{X(t_1+\tau)\ldots X(t_n+\tau)} = F_{X(t_1)\ldots X(t_n)}$$

Properties of Stochastic Processes

In this course we will focus on stochastic processes with the following properties:

$\{X(t)\}$ **is time homogeneous**:
the behaviour of the system does not depend on when it is observed. In particular, the transition rates between states are independent of the time at which the transitions occur. Thus, for all t and s, it follows that

$$\Pr(X(t+\tau) = x_k \mid X(t) = x_j) = \Pr(X(s+\tau) = x_k \mid X(s) = x_j).$$

Exit rate and sojourn time

In any stochastic process the time spent in a state is called the sojourn time.

Exit rate and sojourn time

In any stochastic process the time spent in a state is called the sojourn time.

In a Markov process the rate of leaving a state x_i, q_i the exit rate, is exponentially distributed with the rate which is the sum of all the individual transitions that leave the state, i.e. $q_i = \sum_{j=1, j \neq i}^{N} q_{i,j}$.

Exit rate and sojourn time

In any stochastic process the time spent in a state is called the sojourn time.

In a Markov process the rate of leaving a state x_i, q_i the exit rate, is exponentially distributed with the rate which is the sum of all the individual transitions that leave the state, i.e. $q_i = \sum_{j=1, j\neq i}^{N} q_{i,j}$.

This follows from the superposition principle of exponential distributions.

Exit rate and sojourn time

In any stochastic process the time spent in a state is called the sojourn time.

In a Markov process the rate of leaving a state x_i, q_i the exit rate, is exponentially distributed with the rate which is the sum of all the individual transitions that leave the state, i.e. $q_i = \sum_{j=1, j \neq i}^{N} q_{i,j}$.

This follows from the superposition principle of exponential distributions.

It follows that the sojourn time will be $1/q_i$.

Exit rate and sojourn time

In any stochastic process the time spent in a state is called the sojourn time.

In a Markov process the rate of leaving a state x_i, q_i the exit rate, is exponentially distributed with the rate which is the sum of all the individual transitions that leave the state, i.e. $q_i = \sum_{j=1, j \neq i}^{N} q_{i,j}$.

This follows from the superposition principle of exponential distributions.

It follows that the sojourn time will be $1/q_i$.

Note: by the Markov property, the sojourn times are memoryless.

Transition rates and transition probabilities

At time τ, the probability that there is a state transition in the interval $(\tau, \tau + dt)$ is $q_i dt + o(dt)$.

Transition rates and transition probabilities

At time τ, the probability that there is a state transition in the interval $(\tau, \tau + dt)$ is $q_i dt + o(dt)$.

When a transition out of state x_i occurs, the new state is x_j with probability p_{ij}, which must depend only on i and j (Markov).

Transition rates and transition probabilities

At time τ, the probability that there is a state transition in the interval $(\tau, \tau + dt)$ is $q_i dt + o(dt)$.

When a transition out of state x_i occurs, the new state is x_j with probability p_{ij}, which must depend only on i and j (Markov).

Thus, for $i \neq j, i, j \in S$, $\Pr(X(\tau + dt) = j \mid X(\tau) = i) = q_{ij} dt + o(dt)$ where the $q_{ij} = q_i p_{ij}$, by the decomposition property.

Transition rates and transition probabilities

At time τ, the probability that there is a state transition in the interval $(\tau, \tau + dt)$ is $q_i dt + o(dt)$.

When a transition out of state x_i occurs, the new state is x_j with probability p_{ij}, which must depend only on i and j (Markov).

Thus, for $i \neq j, i, j \in S$, $\Pr(X(\tau + dt) = j \mid X(\tau) = i) = q_{ij} dt + o(dt)$ where the $q_{ij} = q_i p_{ij}$, by the decomposition property.

The q_{ij} are called the instantaneous transition rates.

Transition rates and transition probabilities

At time τ, the probability that there is a state transition in the interval $(\tau, \tau + dt)$ is $q_i dt + o(dt)$.

When a transition out of state x_i occurs, the new state is x_j with probability p_{ij}, which must depend only on i and j (Markov).

Thus, for $i \neq j, i, j \in S$, $\Pr(X(\tau + dt) = j \mid X(\tau) = i) = q_{ij} dt + o(dt)$ where the $q_{ij} = q_i p_{ij}$, by the decomposition property.

The q_{ij} are called the instantaneous transition rates.

The transition probability p_{ij} is the probability, given that a transition out of state i occurs, that it is the transition to state j. By the definition of conditional probability, this is $p_{ij} = q_{ij}/q_i$.

Infinitesimal Generator Matrix

The state transition diagram of a Markov process captures all the information about the states of the system and the transitions which can occur between then.

Infinitesimal Generator Matrix

The state transition diagram of a Markov process captures all the information about the states of the system and the transitions which can occur between then.

We can capture this information in a matrix, $\mathbf{Q}$, termed the infinitesimal generator matrix.

Infinitesimal Generator Matrix

The state transition diagram of a Markov process captures all the information about the states of the system and the transitions which can occur between then.

We can capture this information in a matrix, $\mathbf{Q}$, termed the infinitesimal generator matrix.

For a state space of size N, this is a $N \times N$ matrix, where entry $q(i, j)$ or $q_{i,j}$, records the transition rate of moving from state x_i to state x_j.

Infinitesimal Generator Matrix

The state transition diagram of a Markov process captures all the information about the states of the system and the transitions which can occur between then.

We can capture this information in a matrix, $\mathbf{Q}$, termed the infinitesimal generator matrix.

For a state space of size N, this is a $N \times N$ matrix, where entry $q(i,j)$ or $q_{i,j}$, records the transition rate of moving from state x_i to state x_j.

By convention, the diagonal entries $q_{i,i}$ are the negative row sum for row i, i.e.

$$q_{i,i} = -\sum_{j=1, j\neq i}^{N} q_{i,j}$$

Steady state probability distribution

In performance modelling we are often interested in the probability distribution of the random variable $X(t)$ over the state space S, as the system settles into a regular pattern of behaviour.

Steady state probability distribution

In performance modelling we are often interested in the probability distribution of the random variable $X(t)$ over the state space S, as the system settles into a regular pattern of behaviour.

This is termed the steady state probability distribution.

Steady state probability distribution

In performance modelling we are often interested in the probability distribution of the random variable $X(t)$ over the state space S, as the system settles into a regular pattern of behaviour.

This is termed the steady state probability distribution.

From this probability distribution we will derive performance measures based on subsets of states where some condition holds.

Existence of a steady state probability distribution

For every time-homogeneous, finite, irreducible Markov process with state space S, there exists a steady state probability distribution

$$\{\pi_k, x_k \in S\}$$

Existence of a steady state probability distribution

For every time-homogeneous, finite, irreducible Markov process with state space S, there exists a steady state probability distribution

$$\{\pi_k, x_k \in S\}$$

This distribution is the same as the limiting or long term probability distribution:

$$\pi_k = \lim_{t \to \infty} \Pr(X(t) = x_k \mid X(0) = x_0)$$

Existence of a steady state probability distribution

For every time-homogeneous, finite, irreducible Markov process with state space S, there exists a steady state probability distribution

$$\{\pi_k, x_k \in S\}$$

This distribution is the same as the limiting or long term probability distribution:

$$\pi_k = \lim_{t \to \infty} \Pr(X(t) = x_k \mid X(0) = x_0)$$

This distribution is reached when the initial state no longer has any influence.

Probability flux

In steady state, π_i is the proportion of time that the process spends in state x_i.

Probability flux

In steady state, $\boldsymbol{\pi_i}$ is the proportion of time that the process spends in state x_i.

Recall q_{ij} is the instantaneous probability that the model makes a transition from state x_i to state x_j.

Probability flux

In steady state, $\boldsymbol{\pi_i}$ is the proportion of time that the process spends in state x_i.

Recall q_{ij} is the instantaneous probability that the model makes a transition from state x_i to state x_j.

Thus, in an instant of time, the probability that a transition will occur from state x_i to state x_j is the probability that the model was in state x_i, $\boldsymbol{\pi_i}$, multiplied by the transition rate q_{ij}.

Probability flux

In steady state, π_i is the proportion of time that the process spends in state x_i.

Recall q_{ij} is the instantaneous probability that the model makes a transition from state x_i to state x_j.

Thus, in an instant of time, the probability that a transition will occur from state x_i to state x_j is the probability that the model was in state x_i, π_i, multiplied by the transition rate q_{ij}.

This is called the probability flux from state x_i to state x_j.

Global balance equations

In steady state, equilibrium is maintained so for any state the total probability flux out is equal to the total probability flux into the state.

$$\boxed{\underbrace{\boldsymbol{\pi_i} \times \sum_{x_j \in S, j \neq i} q_{ij}}_{\text{flux } \textbf{out} \text{ of } x_i} = \underbrace{\sum_{x_j \in S, j \neq i} (\boldsymbol{\pi_j} \times q_{ji})}_{\text{flux } \textbf{into } x_i}}$$

Global balance equations

In steady state, equilibrium is maintained so for any state the total probability flux out is equal to the total probability flux into the state.

$$\boxed{\underbrace{\boldsymbol{\pi_i} \times \sum_{x_j \in S, j \neq i} q_{ij}}_{\text{flux } \textbf{out} \text{ of } x_i} = \underbrace{\sum_{x_j \in S, j \neq i} (\boldsymbol{\pi_j} \times q_{ji})}_{\text{flux } \textbf{into } x_i}}$$

(If this were not true the distribution over states would change.)

Global balance equations

Recall that the diagonal elements of the infinitesimal generator matrix $\mathbf{Q}$ are the negative sum of the other elements in the row, i.e.
$q_{ii} = -\sum_{x_j \in S, j \neq i} q_{ij}$.

Global balance equations

Recall that the diagonal elements of the infinitesimal generator matrix $\mathbf{Q}$ are the negative sum of the other elements in the row, i.e. $q_{ii} = -\sum_{x_j \in S, j \neq i} q_{ij}$.

We can use this to rearrange the flux balance equation to be:

$$\sum_{x_j \in S} \boldsymbol{\pi_j} q_{ji} = 0.$$

Global balance equations

Recall that the diagonal elements of the infinitesimal generator matrix $\mathbf{Q}$ are the negative sum of the other elements in the row, i.e. $q_{ii} = -\sum_{x_j \in S, j \neq i} q_{ij}$.

We can use this to rearrange the flux balance equation to be:

$$\sum_{x_j \in S} \boldsymbol{\pi_j} q_{ji} = 0.$$

Expressing the unknown values $\boldsymbol{\pi_i}$ as a row vector $\boldsymbol{\pi}$, we can write this as a matrix equation:

$$\boxed{\boldsymbol{\pi}\ \mathbf{Q} = 0}$$

Normalising constant

The π_i are unknown — they are the values we wish to find.

Normalising constant

The π_i are unknown — they are the values we wish to find.

If there are N states in the state space, the global balance equations give us N equations in N unknowns.

Normalising constant

The $\boldsymbol{\pi_i}$ are unknown — they are the values we wish to find.

If there are N states in the state space, the global balance equations give us N equations in N unknowns.

However this collection of equations is irreducible.

Normalising constant

The $\boldsymbol{\pi_i}$ are unknown — they are the values we wish to find.

If there are N states in the state space, the global balance equations give us N equations in N unknowns.

However this collection of equations is irreducible.

Fortunately, since $\{\boldsymbol{\pi_i}\}$ is a probability distribution we also know that the normalisation condition holds:

$$\boxed{\sum_{x_i \in S} \boldsymbol{\pi_i} = 1}$$

Normalising constant

The $\boldsymbol{\pi_i}$ are unknown — they are the values we wish to find.

If there are N states in the state space, the global balance equations give us N equations in N unknowns.

However this collection of equations is irreducible.

Fortunately, since $\{\boldsymbol{\pi_i}\}$ is a probability distribution we also know that the normalisation condition holds:

$$\boxed{\sum_{x_i \in S} \boldsymbol{\pi_i} = 1}$$

With these $n+1$ equations we can use standard linear algebra techniques to solve the equations and find the n unknowns, $\{\boldsymbol{\pi_i}\}$.

Example

- Consider a system with multiple CPUs, each with its own private memory, and one common memory which can be accessed only by one processor at a time.

Example

- Consider a system with multiple CPUs, each with its own private memory, and one common memory which can be accessed only by one processor at a time.
- The CPUs execute in private memory for a random time before issuing a common memory access request. Assume that this random time is exponentially distributed with parameter λ.

Example

- Consider a system with multiple CPUs, each with its own private memory, and one common memory which can be accessed only by one processor at a time.
- The CPUs execute in private memory for a random time before issuing a common memory access request. Assume that this random time is exponentially distributed with parameter λ.
- The common memory access duration is also assumed to be exponentially distributed, with parameter μ (the average duration of a common memory access is $1/\mu$).

Example

If the system has only one processor, it has only two states:

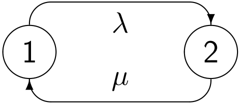
1. The processor is executing in its private memory;
2. The processor is accessing common memory.

Example

If the system has only one processor, it has only two states:

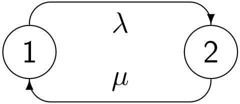
1. The processor is executing in its private memory;
2. The processor is accessing common memory.

The system behaviour can be modelled by a 2-state Markov process whose state transition diagram and generator matrix are as shown below:



$$\mathbf{Q} = \begin{pmatrix} -\lambda & \lambda \\ \mu & -\mu \end{pmatrix}$$

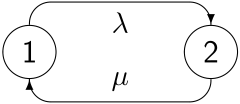
Example



$$\mathbf{Q} = \begin{pmatrix} -\lambda & \lambda \\ \mu & -\mu \end{pmatrix}$$

If we consider the probability flux in and out of state 1 we obtain: $\pi_1 \lambda = \pi_2 \mu$. Similarly, for state 2: $\pi_2 \mu = \pi_1 \lambda$.

Example

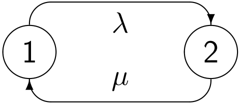


$$\mathbf{Q} = \begin{pmatrix} -\lambda & \lambda \\ \mu & -\mu \end{pmatrix}$$

If we consider the probability flux in and out of state 1 we obtain: $\boldsymbol{\pi_1} \lambda = \boldsymbol{\pi_2}\mu$. Similarly, for state 2: $\boldsymbol{\pi_2}\ \mu = \boldsymbol{\pi_1}\lambda$.

We know from the normalisation condition that: $\boldsymbol{\pi_1} + \boldsymbol{\pi_2} = 1$.

Example



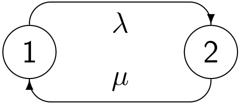
$$\mathbf{Q} = \begin{pmatrix} -\lambda & \lambda \\ \mu & -\mu \end{pmatrix}$$

If we consider the probability flux in and out of state 1 we obtain: $\boldsymbol{\pi_1}\ \lambda = \boldsymbol{\pi_2}\mu$. Similarly, for state 2: $\boldsymbol{\pi_2}\ \mu = \boldsymbol{\pi_1}\lambda$.

We know from the normalisation condition that: $\boldsymbol{\pi_1} + \boldsymbol{\pi_2} = 1$.

Thus the steady state probability distribution is $\boldsymbol{\pi} = \left(\frac{\mu}{\mu+\lambda}, \frac{\lambda}{\mu+\lambda}\right)$.

Example



$$\mathbf{Q} = \begin{pmatrix} -\lambda & \lambda \\ \mu & -\mu \end{pmatrix}$$

If we consider the probability flux in and out of state 1 we obtain: $\boldsymbol{\pi_1}\ \lambda = \boldsymbol{\pi_2}\mu$. Similarly, for state 2: $\boldsymbol{\pi_2}\ \mu = \boldsymbol{\pi_1}\lambda$.

We know from the normalisation condition that: $\boldsymbol{\pi_1} + \boldsymbol{\pi_2} = 1$.

Thus the steady state probability distribution is $\boldsymbol{\pi} = \left(\frac{\mu}{\mu+\lambda}, \frac{\lambda}{\mu+\lambda}\right)$.

From this we can deduce, for example, that the probability that the processor is executing in private memory is $\mu/(\mu+\lambda)$.

Solving the global balance equations

- In general the systems of equations will be too large to solve by hand.
- Instead we take advantage of linear algebra packages which can solve matrix equations of the form $\mathbf{A}\mathbf{x} = \mathbf{b}$.
- Here
  - $\mathbf{A}$ is an $N \times N$ matrix,
  - $\mathbf{x}$ is a column vector of N unknowns, and
  - $\mathbf{b}$ is a column vector of N values.

Solving the global balance equations

First we must resolve two problems:

1. Our global balance equation is expressed in terms of a row vector of unknowns $\boldsymbol{\pi}$, $\boldsymbol{\pi}\,\mathbf{Q} = 0$: the unknowns.

Solving the global balance equations

First we must resolve two problems:

1. Our global balance equation is expressed in terms of a row vector of unknowns $\boldsymbol{\pi}$, $\boldsymbol{\pi}\,\mathbf{Q} = 0$: the unknowns.

   This problem is resolved by transposing the equation, i.e. $\mathbf{Q}^T\boldsymbol{\pi} = 0$, *where the right hand side is now a column vector of zeros, rather than a row vector.*

Solving the global balance equations

2 We must eliminate the redundancy in the global balance equations and add in the normalisation condition.

Solving the global balance equations

2 We must eliminate the redundancy in the global balance equations and add in the normalisation condition.

We replace one of the global balance equations by the normalisation condition. In $\mathbf{Q}^T$ we replace one row (usually the last) by a row of 1's. We denote the modified matrix $\mathbf{Q}_N^T$.

Solving the global balance equations

2 We must eliminate the redundancy in the global balance equations and add in the normalisation condition.

We replace one of the global balance equations by the normalisation condition. In $\mathbf{Q}^T$ *we replace one row (usually the last) by a row of 1's. We denote the modified matrix* $\mathbf{Q}_N^T$.

We must also make the corresponding change to the "solution" vector $\mathbf{0}$, *to be a column vector with 1 in the last row, and zeros everywhere else. We denote this vector,* $\mathbf{e}_N$.

Solving the global balance equations

2. We must eliminate the redundancy in the global balance equations and add in the normalisation condition.

   We replace one of the global balance equations by the normalisation condition. In $\mathbf{Q}^T$ *we replace one row (usually the last) by a row of 1's. We denote the modified matrix* $\mathbf{Q}_N^T$.

   We must also make the corresponding change to the "solution" vector $\mathbf{0}$*, to be a column vector with 1 in the last row, and zeros everywhere else. We denote this vector,* $\mathbf{e}_N$.

Now we can use any linear algebra solution package, such as `MatLab` to solve the resulting equation:

$$\mathbf{Q}_N^T \boldsymbol{\pi} = \mathbf{e}_N$$

Example

Consider the two-processor version of the multiprocessor with processors A and B.

Example

Consider the two-processor version of the multiprocessor with processors A and B.

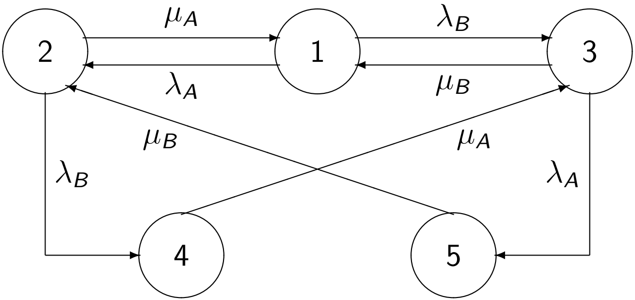
We assume that the processors have different timing characteristics, the private memory access of A being governed by an exponential distribution with parameter λ_A, the common memory access of B being governed by an exponential distribution with parameter μ_B, etc.

Example: state space

Now the state space becomes:

1. A and B both executing in their private memories;
2. B executing in private memory, and A accessing common memory;
3. A executing in private memory, and B accessing common memory;
4. A accessing common memory, B waiting for common memory;
5. B accessing common memory, A waiting for common memory;

Example: state space



Example: generator matrix

$$\mathbf{Q} = \begin{pmatrix} -(\lambda_A + \lambda_B) & \lambda_A & \lambda_B & 0 & 0 \\ \mu_A & -(\mu_A + \lambda_B) & 0 & \lambda_B & 0 \\ \mu_B & 0 & -(\mu_B + \lambda_A) & 0 & \lambda_A \\ 0 & 0 & \mu_A & -\mu_A & 0 \\ 0 & \mu_B & 0 & 0 & -\mu_B \end{pmatrix}$$

Example: modified generator matrix

$$\mathbf{Q}_N^T = \begin{pmatrix} -(\lambda_A + \lambda_B) & \mu_A & \mu_B & 0 & 0 \\ \lambda_A & -(\mu_A + \lambda_B) & 0 & 0 & \mu_B \\ \lambda_B & 0 & -(\mu_B + \lambda_A) & \mu_A & 0 \\ 0 & \lambda_B & 0 & -\mu_A & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Example: steady state probability distribution

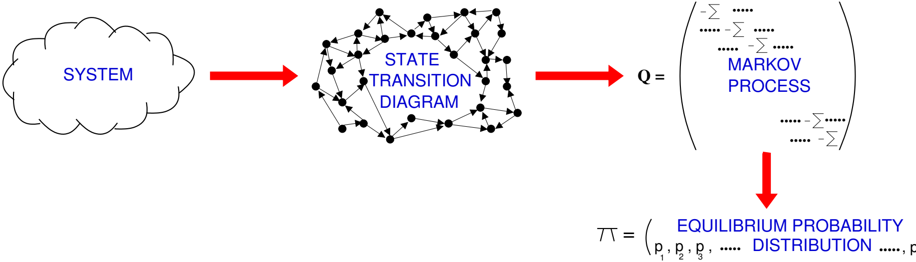
If we choose the following values for the parameters:

$$\lambda_A = 0.05 \qquad \lambda_B := 0.1 \qquad \mu_A = 0.02 \qquad \mu_B = 0.05$$

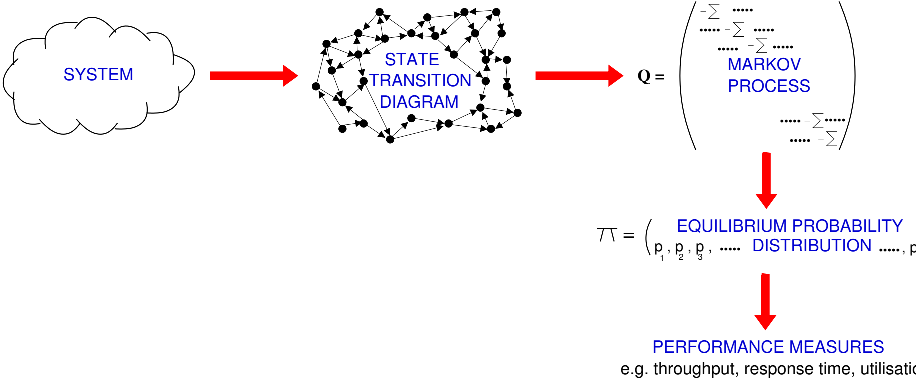
solving the matrix equation, and rounding figures to 4 significant figures, we obtain:

$$\boldsymbol{\pi} = (0.0693,\ 0.0990,\ 0.1683,\ 0.4951,\ 0.1683)$$

Deriving Performance Measures



Deriving Performance Measures



Deriving Performance Measures

Broadly speaking, there are three ways in which performance measures can be derived from the steady state distribution of a Markov process.

Deriving Performance Measures

Broadly speaking, there are three ways in which performance measures can be derived from the steady state distribution of a Markov process.

These different methods can be thought of as corresponding to different types of measure:

- state-based measures, e.g. utilisation;

Deriving Performance Measures

Broadly speaking, there are three ways in which performance measures can be derived from the steady state distribution of a Markov process.

These different methods can be thought of as corresponding to different types of measure:

- state-based measures, e.g. utilisation;
- rate-based measures, e.g. throughput;

Deriving Performance Measures

Broadly speaking, there are three ways in which performance measures can be derived from the steady state distribution of a Markov process.

These different methods can be thought of as corresponding to different types of measure:

- state-based measures, e.g. utilisation;
- rate-based measures, e.g. throughput;
- other measures which fall outside the above categories, e.g. response time.

State-based measures

State-based measures correspond to the probability that the model is in a state, or a subset of states, which satisfy some condition.

State-based measures

State-based measures correspond to the probability that the model is in a state, or a subset of states, which satisfy some condition.

For example, utilisation will correspond to those states where a resource is in use.

State-based measures

State-based measures correspond to the probability that the model is in a state, or a subset of states, which satisfy some condition.

For example, utilisation will correspond to those states where a resource is in use.

If we consider the multiprocessor example, the utilisation of the common memory, U_{mem}, is the total probability that the model is in one of the states in which the common memory is in use:

$$U_{mem} = \boldsymbol{\pi_2} + \boldsymbol{\pi_3} + \boldsymbol{\pi_4} + \boldsymbol{\pi_5} = 93.07\%$$

State-based measures

Other examples of state-based measures are idle time, or the number of jobs in a system.

State-based measures

Other examples of state-based measures are idle time, or the number of jobs in a system.

Some measures such as the number of jobs will involve a weighted sum of steady state probabilities, weighted by the appropriate value (expectation).

State-based measures

Other examples of state-based measures are idle time, or the number of jobs in a system.

Some measures such as the number of jobs will involve a weighted sum of steady state probabilities, weighted by the appropriate value (expectation).

For example, if we consider jobs waiting for the common memory to be queued in that subsystem, then the average number of jobs in the common memory, N_{mem}, is:

$$N_{mem} = (1 \times \pi_2) + (1 \times \pi_3) + (2 \times \pi_4) + (2 \times \pi_5) = 1.594$$

Rate-based measures

Rate-based measures are those which correspond to the predicted rate at which some event occurs.

Rate-based measures

Rate-based measures are those which correspond to the predicted rate at which some event occurs.

This will be the product of the rate of the event, and the probability that the event is enabled, i.e. the probability of being in one of the states from which the event can occur.

Example: rate-based measures

In order to calculate the throughput of the common memory, we need the average number of accesses from either processor which it satisfies in unit time.

Example: rate-based measures

In order to calculate the throughput of the common memory, we need the average number of accesses from either processor which it satisfies in unit time.

X_{mem} is thus calculated as:

$$X_{mem} = (\mu_A \times (\boldsymbol{\pi_2} + \boldsymbol{\pi_4})) + (\mu_B \times (\boldsymbol{\pi_3} + \boldsymbol{\pi_5})) = 0.0287$$

or, approximately one access every 35 milliseconds.

Other measures

The other measures are those which are neither rate-based or state-based.

Other measures

The other measures are those which are neither rate-based or state-based.

In these cases, we usually use one of the operational laws to derive the information we need, based on values that we have obtained from solution of the model.

Other measures

The other measures are those which are neither rate-based or state-based.

In these cases, we usually use one of the operational laws to derive the information we need, based on values that we have obtained from solution of the model.

For example, applying Little's Law to the common memory we see that

$$W_{mem} = N_{mem}/X_{mem} = 1.594/0.0287 = 55.54 \text{ milliseconds}$$

Assumptions

Stochastic Hypothesis

"The behaviour of a real system during a given period of time is characterised by the probability distributions of a stochastic process."

Assumptions

- All delays and inter-event times are exponentially distributed.

Assumptions

- All delays and inter-event times are exponentially distributed.
- (This will often not fit with observations of real systems.)

Assumptions

- All delays and inter-event times are exponentially distributed.
- (This will often not fit with observations of real systems.)
- We make the assumption because of the nice mathematical properties of the exponential distribution, and because it is the only distribution giving us a Markov process.

Assumptions

- All delays and inter-event times are exponentially distributed.
- (This will often not fit with observations of real systems.)
- We make the assumption because of the nice mathematical properties of the exponential distribution, and because it is the only distribution giving us a Markov process.
- Plus only a single parameter to be fitted (the rate), which can be easily derived from observations of the average duration.

Assumptions

- All delays and inter-event times are exponentially distributed.
- (This will often not fit with observations of real systems.)
- We make the assumption because of the nice mathematical properties of the exponential distribution, and because it is the only distribution giving us a Markov process.
- Plus only a single parameter to be fitted (the rate), which can be easily derived from observations of the average duration.
- The Markov/memoryless assumption — future behaviour is only dependent on the current state, not on the past history — is a reasonable assumption for computer and communication systems, if we choose our states carefully.

Assumptions

- All delays and inter-event times are exponentially distributed.
- (This will often not fit with observations of real systems.)
- We make the assumption because of the nice mathematical properties of the exponential distribution, and because it is the only distribution giving us a Markov process.
- Plus only a single parameter to be fitted (the rate), which can be easily derived from observations of the average duration.
- The Markov/memoryless assumption — future behaviour is only dependent on the current state, not on the past history — is a reasonable assumption for computer and communication systems, if we choose our states carefully.
- We generally assume that the Markov process is finite, time homogeneous and irreducible.

Exercise

- Consider the multiprocessor example, but with three processors, A, B and C sharing the common memory instead of two.
- List the states of the system, and draw the state transition diagram for this case.
- What is the difficulty in doing this and what further information do you need?

To be continued...

Advanced Topics in Software Engineering: Discrete Time Markov Chains

Prof. Michele Loreti

Advanced Topics in Software Engineering
Corso di Laurea in Informatica (L31)
Scuola di Scienze e Tecnologie

Discrete time Markov chains...

Many (distributed) algorithms can be expressed in terms of Discrete Time Markov Chains.

Discrete time Markov chains...

Many (distributed) algorithms can be expressed in terms of Discrete Time Markov Chains.

This is a family of random variable $\{X(k) | k \in \mathbb{N}\}$, where $X(k)$ are the observations at (discrete) time k.

Discrete time Markov chains...

Many (distributed) algorithms can be expressed in terms of Discrete Time Markov Chains.

This is a family of random variable $\{X(k) | k \in \mathbb{N}\}$, where $X(k)$ are the observations at (discrete) time k.

Markov property is satisfied:

$$Pr(X(k) = s_k | X(k-1) = s_{k-1}, \ldots, X(0) = s_0) = Pr(X(k) = s_k | X(k-1) = s_{k-1})$$

Discrete time Markov chains...

Many (distributed) algorithms can be expressed in terms of Discrete Time Markov Chains.

This is a family of random variable $\{X(k) | k \in \mathbb{N}\}$, where $X(k)$ are the observations at (discrete) time k.

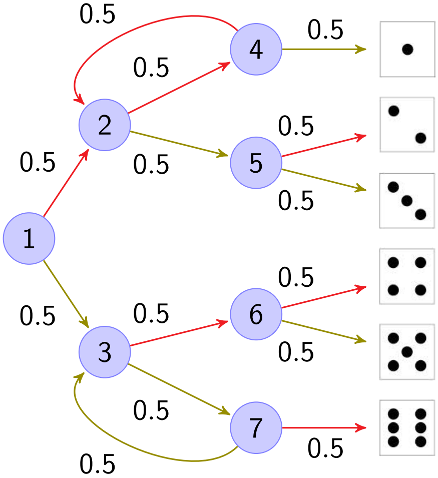
Markov property is satisfied:

$$Pr(X(k) = s_k | X(k-1) = s_{k-1}, \ldots, X(0) = s_0) = Pr(X(k) = s_k | X(k-1) = s_{k-1})$$

We can consider a state based view of a DTMC.

Example: Knut-Yao Algorithm

We can use a (fair) coin to mimic a dice (red edges stands for *head*, green for *tail*):



Questions. . .

Is the algorithm correct?

Questions. . .

Is the algorithm correct?

Is the outcome fair?

Questions. . .

Is the algorithm correct?

Is the outcome fair?

What is the probability of needing more than 4 coin tosses?

Questions...

Is the algorithm correct?

Is the outcome fair?

What is the probability of needing more than 4 coin tosses?

On average, how many coin tosses are needed?

Discrete Time Markov Chains

Can be seen as a state-transition systems augmented with probabilities

Discrete Time Markov Chains

Can be seen as a state-transition systems augmented with probabilities

States represent the possible configurations of the system being modelled.

Discrete Time Markov Chains

Can be seen as a state-transition systems augmented with probabilities

States represent the possible configurations of the system being modelled.

Transitions describe how system evolve from one state to the others in a discrete-time step.

Discrete Time Markov Chains

Can be seen as a state-transition systems augmented with probabilities

States represent the possible configurations of the system being modelled.

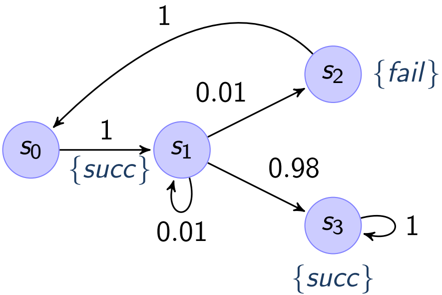
Transitions describe how system evolve from one state to the others in a discrete-time step.

Probabilities of transitions is given by discrete probability distributions.

Simple DTMC example

Modelling a very simple communication protocol:

- after one step a process starts trying to send a message;
- with probability 0.01, channel unready so wait a step;
- with probability 0.98, send message successfully and stop;
- with probability 0.01, send message fails, restart.



Discrete Time Markov Chains

A Discrete Time Markov Chain (DTMC) is a pair (S, P) where

- S is a set of states;
- $P : S \times S \rightarrow [0, 1]$ is a transition probability matrix:

$$\sum_{s' \in S} P(s, s') = 1 \qquad \text{(for all } s \in S\text{)}$$

A Discrete Time Markov Chain (DTMC) is a pair (S, P) where

- S is a set of states;
- $P : S \times S \to [0, 1]$ is a transition probability matrix:

$$\sum_{s' \in S} P(s, s') = 1 \qquad \text{(for all } s \in S\text{)}$$

$$S = \{s_0, s_1, s_2, s_3\}$$

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{tikzcd} s_0 \arrow[r, "1"] & s_1 \arrow[loop below, "0.01"] \arrow[r, "0.01"] \arrow[dr, "0.98"'] & s_2 \arrow[ll, bend right, "1"'] \\ & & s_3 \arrow[loop right, "1"] \end{tikzcd}$$

Definitions...

P is a stochastic matrix if and only if:

- $\forall s, s' \in S : P(s, s') \in [0, 1]$;
- $\forall s \in S : \sum_{s' \in S} P(s, s') = 1$.

Definitions...

P is a stochastic matrix if and only if:

- $\forall s, s' \in S : P(s, s') \in [0, 1]$;
- $\forall s \in S : \sum_{s' \in S} P(s, s') = 1$.

P is a sub-stochastic matrix if and only if:

- $\forall s, s' \in S : P(s, s') \in [0, 1]$;
- $\forall s \in S : \sum_{s' \in S} P(s, s') \leq 1$.

Definitions...

P is a stochastic matrix if and only if:

- $\forall s, s' \in S : P(s, s') \in [0, 1]$;
- $\forall s \in S : \sum_{s' \in S} P(s, s') = 1$.

P is a sub-stochastic matrix if and only if:

- $\forall s, s' \in S : P(s, s') \in [0, 1]$;
- $\forall s \in S : \sum_{s' \in S} P(s, s') \leq 1$.

A state $s \in S$ is absorbing if and only if:

$$P(s, s') = \begin{cases} 1 & \text{if } s = s' \\ 0 & \text{otherwise} \end{cases}$$

States and transitions

Current state of a DTMC can be rendered in terms of a probability distributions π over states ($Dist(S)$).

States and transitions

Current state of a DTMC can be rendered in terms of a probability distributions π over states ($Dist(S)$).

If $\pi \in Dist(\pi)$ is the probability distribution of current state, next state can be compute via a vector-matrix multiplication:

$$\pi P = \pi'$$

States and transitions

Current state of a DTMC can be rendered in terms of a probability distributions π over states ($Dist(S)$).

If $\pi \in Dist(\pi)$ is the probability distribution of current state, next state can be compute via a vector-matrix multiplication:

$$\pi P = \pi'$$

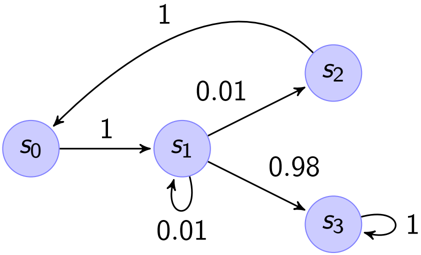
Let $x \in [0,1]^S$ (that is a column vector associating each element in S with a value in $[0,1]$), a matrix-vector multiplication can be use to compute the reverse flow of x:

$$x' = Px$$

Example

$$S = \{s_0, s_1, s_2, s_3\}$$

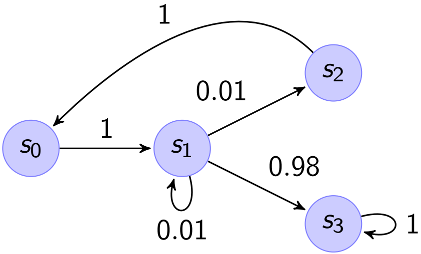
$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Example

$$S = \{s_0, s_1, s_2, s_3\}$$

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

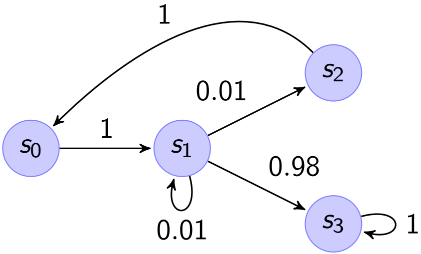


$$(1, 0, 0, 0) \to$$

Example

$$S = \{s_0, s_1, s_2, s_3\}$$

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

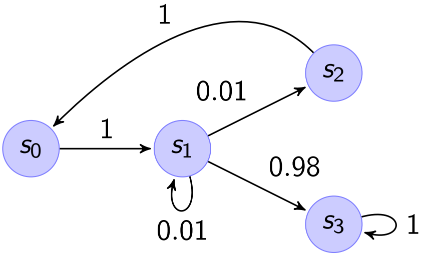


$$(1, 0, 0, 0) \rightarrow (\ ,\ ,\ ,\ )$$

Example

$$S = \{s_0, s_1, s_2, s_3\}$$

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

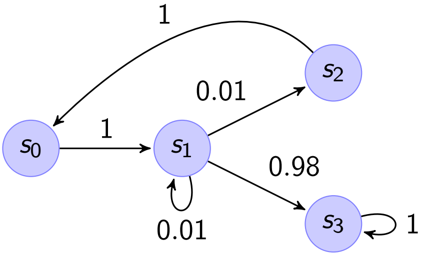


$$(1, 0, 0, 0) \rightarrow ({\color{red}0}, \quad, \quad, \quad)$$

Example

$$S = \{s_0, s_1, s_2, s_3\}$$

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

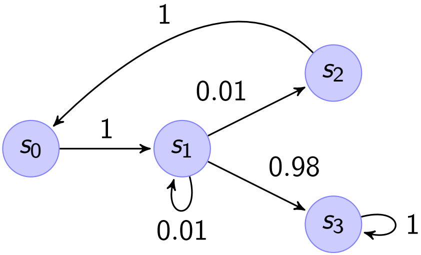


$$(1, 0, 0, 0) \to (0, \ , \ , \ )$$

Example

$$S = \{s_0, s_1, s_2, s_3\}$$

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

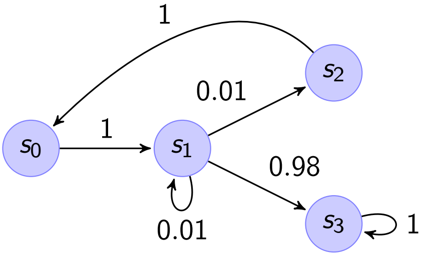


$$(1, 0, 0, 0) \rightarrow (0, 1, \;\; , \;\;)$$

Example

$$S = \{s_0, s_1, s_2, s_3\}$$

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

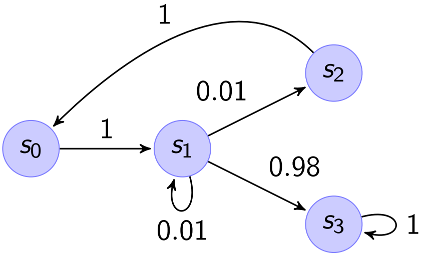


$$(1, 0, 0, 0) \to (0, 1, \quad , \quad)$$

Example

$$S = \{s_0, s_1, s_2, s_3\}$$

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

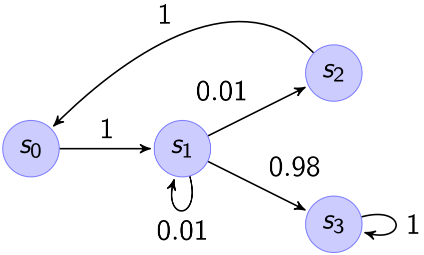


$$(1, 0, 0, 0) \rightarrow (0, 1, 0, \quad)$$

Example

$$S = \{s_0, s_1, s_2, s_3\}$$

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

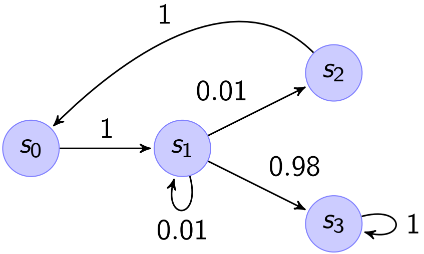


$$(1, 0, 0, 0) \to (0, 1, 0, \;\;)$$

Example

$$S = \{s_0, s_1, s_2, s_3\}$$

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

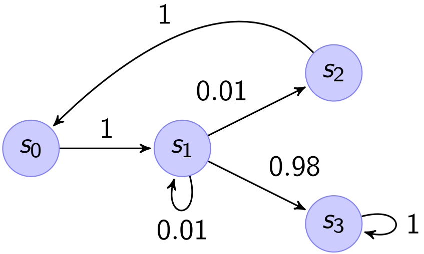


$$(1, 0, 0, 0) \rightarrow (0, 1, 0, \textcolor{red}{0})$$

Example

$$S = \{s_0, s_1, s_2, s_3\}$$

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

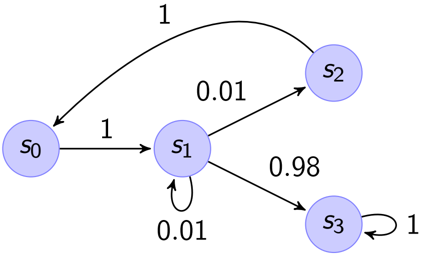


$$(1,0,0,0) \rightarrow (0,1,0,0) \rightarrow (\quad,\qquad,\qquad,\qquad)$$

Example

$$S = \{s_0, s_1, s_2, s_3\}$$

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

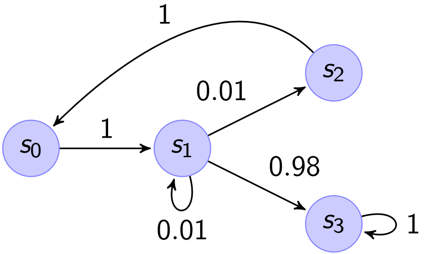


$$(1,0,0,0) \to (0,1,0,0) \to (\ ,\ \ ,\ \ ,\ \ )$$

Example

$$S = \{s_0, s_1, s_2, s_3\}$$

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

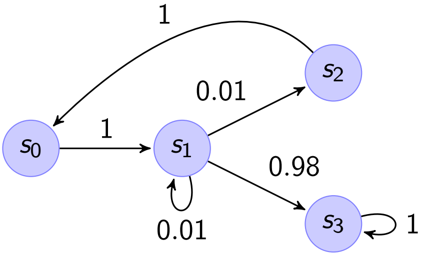


$$(1, 0, 0, 0) \to (0, 1, 0, 0) \to ({\color{red}0}, \quad , \quad , \quad)$$

Example

$$S = \{s_0, s_1, s_2, s_3\}$$

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

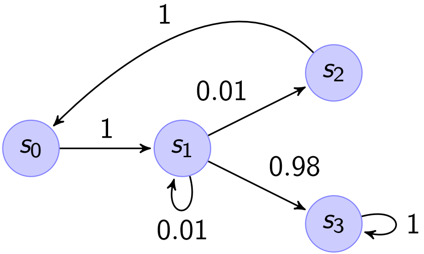


$$(1,0,0,0) \rightarrow (0,1,0,0) \rightarrow (0, \quad , \quad , \quad)$$

Example

$$S = \{s_0, s_1, s_2, s_3\}$$

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

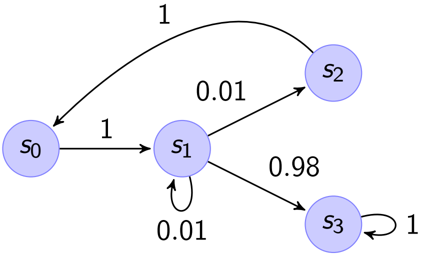


$$(1,0,0,0) \to (0,1,0,0) \to (0, {\color{red}0.01}, \quad\quad , \quad\quad)$$

Example

$$S = \{s_0, s_1, s_2, s_3\}$$

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

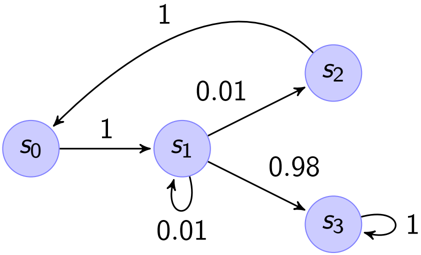


$$(1,0,0,0) \to (0,1,0,0) \to (0, 0.01, \quad , \quad)$$

Example

$$S = \{s_0, s_1, s_2, s_3\}$$

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

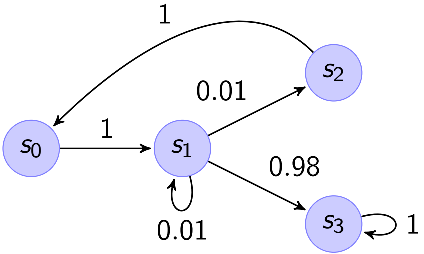


$$(1,0,0,0) \to (0,1,0,0) \to (0, 0.01, {\color{red}0.01}, \quad)$$

Example

$$S = \{s_0, s_1, s_2, s_3\}$$

$$P = \begin{pmatrix} 0 & 1 & 0 & \color{red}{0} \\ 0 & 0.01 & 0.01 & \color{red}{0.98} \\ 1 & 0 & 0 & \color{red}{0} \\ 0 & 0 & 0 & \color{red}{1} \end{pmatrix}$$

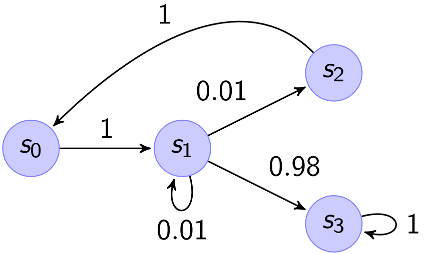


$$(1,0,0,0) \rightarrow \color{red}{(0,1,0,0)} \rightarrow (0, 0.01, 0.01, \qquad)$$

Example

$$S = \{s_0, s_1, s_2, s_3\}$$

$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



$$(1,0,0,0) \rightarrow (0,1,0,0) \rightarrow (0,0.01,0.01,{\color{red}0.98})$$

Paths in DTMCs

A path in a DTMC represents an execution, that is one possible behaviour, of the system being modelled.

Paths in DTMCs

A path in a DTMC represents an execution, that is one possible behaviour, of the system being modelled.

Formally a path is a infinite sequence of states $s_0 s_1 s_2 \ldots$ such that for any $i > 0$, $P(s_i, s_{i+1}) > 0$.

Paths in DTMCs

A path in a DTMC represents an execution, that is one possible behaviour, of the system being modelled.

Formally a path is a infinite sequence of states $s_0 s_1 s_2 \ldots$ such that for any $i > 0$, $P(s_i, s_{i+1}) > 0$.

Notation:

- $Path(s)$ denotes the set of paths starting in a state s;
- $Path_{fin}(s)$ denotes the set of finite path starting in a state s.

Paths and probabilities

To speak about the probability of paths, we need to define a probability space on $Path(s)$:

Paths and probabilities

To speak about the probability of paths, we need to define a probability space on $Path(s)$:

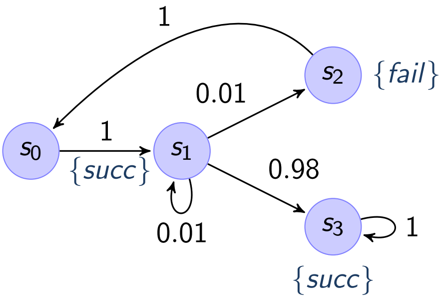
- $\Omega = Path(s)$
- $\Sigma_{Path(s)}$ is the σ-algebra on $Path(s)$ containing $Cyl(\omega)$, for any $\omega \in Paths_{fin}(s)\}$:

$$Cyl(\omega) = \{\omega' \in Path(s) | \omega \text{ is a prefix of } \omega'\}$$

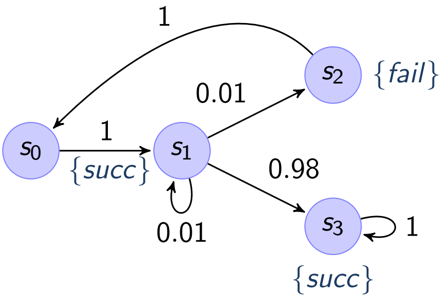
- Probability measure Pr_s is defined as follows:
  - $Pr_s(Cyl(\omega)) = P_s(\omega)$ where

$$\begin{array}{rcl} P_s(s) & = & 1 \\ P_s(ss'\omega) & = & P(s,s') \cdot P_{s'}(s'\omega) \end{array}$$

Example...

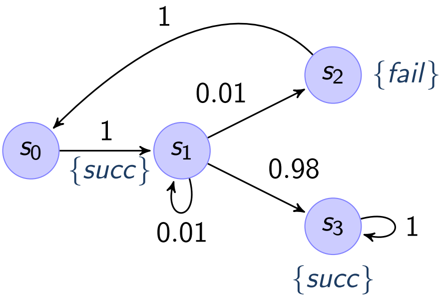


Example...



All the computations where sending immediately fails...

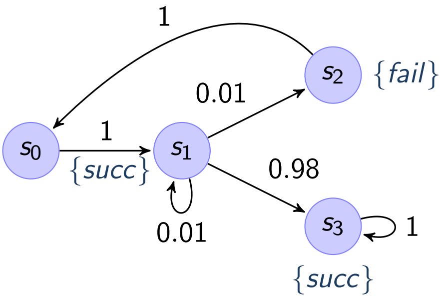
Example...



All the computations where sending immediately fails...

... all paths starting with $s_0 s_1 s_2$

Example...



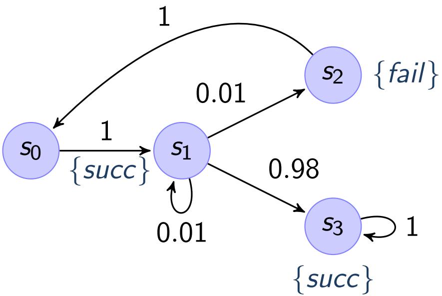
All the computations where sending immediately fails...

... all paths starting with $s_0 s_1 s_2$

... this is $Cyl(s_0 s_1 s_2)$:

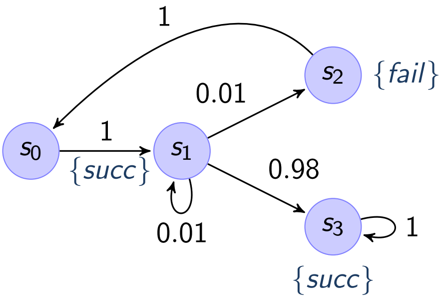
$$Pr_{s_0}(s_0 s_1 s_2) = P(s_0, s_1) \cdot P(s_1, s_2) = 1 \cdot 0.01 = 0.01$$

Example. . .



All the computations that are eventually successful with no failures

Example...



All the computations that are eventually successful with no failures

$$X = Cyl(s_0 s_1 s_3) \cup Cyl(s_0 s_1 s_1 s_3) \cup Cyl(s_0 s_1 s_1 s_1 s_3) \cup \cdots$$

$$Pr_{s_0}(X) = \sum_{i=0}^{\infty} 1 \cdot 0.01^i \cdot 0.98 = \frac{0.98}{0.99}$$

Reachability

Reachability is a key property:

- probability of a path reaching a target state in $T \subseteq S$;

Reachability

Reachability is a key property:

- probability of a path reaching a target state in $T \subseteq S$;
- Examples:

Reachability

Reachability is a key property:

- probability of a path reaching a target state in $T \subseteq S$;
- Examples:
  - the algorithm terminates;

Reachability

Reachability is a key property:

- probability of a path reaching a target state in $T \subseteq S$;
- Examples:
  - the algorithm terminates;
  - an error occurs during execution.

Reachability

Reachability is a key property:

- probability of a path reaching a target state in $T \subseteq S$;
- Examples:
  - the algorithm terminates;
  - an error occurs during execution.

Reachability

Reachability is a key property:

- probability of a path reaching a target state in $T \subseteq S$;
- Examples:
  - the algorithm terminates;
  - an error occurs during execution.

Invariant is the dual of reachability:

- probability of remaining within a class of states $T \subseteq S$;

Reachability

Reachability is a key property:

- probability of a path reaching a target state in $T \subseteq S$;
- Examples:
  - the algorithm terminates;
  - an error occurs during execution.

Invariant is the dual of reachability:

- probability of remaining within a class of states $T \subseteq S$;
- $Pr(\text{"remain in } T\text{"}) = 1 - Pr(\text{"reach } S - T\text{"})$.

Reachability

Reachability is a key property:

- probability of a path reaching a target state in $T \subseteq S$;
- Examples:
  - the algorithm terminates;
  - an error occurs during execution.

Invariant is the dual of reachability:

- probability of remaining within a class of states $T \subseteq S$;
- $Pr(\text{"remain in } T\text{"}) = 1 - Pr(\text{"reach } S - T\text{"})$.
- Example: an error never occurs.

Reachability probability

Let (S, P) be a DTMC and $T \subseteq S$, we let

$$ProbReach(s, T) = Pr(Reach(s, T))$$

where $Reach(s, T) = \{s_0 s_1 \ldots \in Path(s) | \exists i : s_i \in T\}$

Reachability probability

Let (S, P) be a DTMC and $T \subseteq S$, we let

$$ProbReach(s, T) = Pr(Reach(s, T))$$

where $Reach(s, T) = \{s_0 s_1 \ldots \in Path(s) | \exists i : s_i \in T\}$

Question: Is $Reach(s, T)$ measurable?

Reachability probability

Let (S, P) be a DTMC and $T \subseteq S$, we let

$$ProbReach(s, T) = Pr(Reach(s, T))$$

where $Reach(s, T) = \{s_0 s_1 \ldots \in Path(s) | \exists i : s_i \in T\}$

Question: Is $Reach(s, T)$ measurable? **Yes!**

Reachability probability

Let (S, P) be a DTMC and $T \subseteq S$, we let

$$ProbReach(s, T) = Pr(Reach(s, T))$$

where $Reach(s, T) = \{s_0 s_1 \ldots \in Path(s) | \exists i : s_i \in T\}$

Question: Is $Reach(s, T)$ measurable? **Yes!**

Let $Path_{fin}(s, T) = \{\omega \in Path(s) : \exists i . \omega[i] \in T \wedge \forall j < i . \omega[j] \notin T\}$:

Reachability probability

Let (S, P) be a DTMC and $T \subseteq S$, we let

$$ProbReach(s, T) = Pr(Reach(s, T))$$

where $Reach(s, T) = \{s_0 s_1 \ldots \in Path(s) | \exists i : s_i \in T\}$

Question: Is $Reach(s, T)$ measurable? **Yes!**

Let $Path_{fin}(s, T) = \{\omega \in Path(s) : \exists i. \omega[i] \in T \wedge \forall j < i. \omega[j] \notin T\}$:

- $Path_{fin}(s, T)$ is enumerable;

Reachability probability

Let (S, P) be a DTMC and $T \subseteq S$, we let

$$ProbReach(s, T) = Pr(Reach(s, T))$$

where $Reach(s, T) = \{s_0 s_1 \ldots \in Path(s) | \exists i : s_i \in T\}$

Question: Is $Reach(s, T)$ measurable? **Yes!**

Let $Path_{fin}(s, T) = \{\omega \in Path(s) : \exists i. \omega[i] \in T \wedge \forall j < i. \omega[j] \notin T\}$:

- $Path_{fin}(s, T)$ is enumerable;
- for each $\omega_1, \omega_2 \in Path_{fin}(s, T)$ $(\omega_1 \neq \omega_2)$, $Cyl(\omega_1) \cap Cyl(\omega_2) = \emptyset$;

Reachability probability

Let (S, P) be a DTMC and $T \subseteq S$, we let

$$ProbReach(s, T) = Pr(Reach(s, T))$$

where $Reach(s, T) = \{s_0 s_1 \ldots \in Path(s) | \exists i : s_i \in T\}$

Question: Is $Reach(s, T)$ measurable? **Yes!**

Let $Path_{fin}(s, T) = \{\omega \in Path(s) : \exists i.\omega[i] \in T \wedge \forall j < i.\omega[j] \notin T\}$:

- $Path_{fin}(s, T)$ is enumerable;
- for each $\omega_1, \omega_2 \in Path_{fin}(s, T)$ $(\omega_1 \neq \omega_2)$, $Cyl(\omega_1) \cap Cyl(\omega_2) = \emptyset$;
- $Reach(s, T) = \bigcup_{\omega \in Path_{fin}(s,T)} Cyl(\omega)$;

Reachability probability

Let (S, P) be a DTMC and $T \subseteq S$, we let

$$ProbReach(s, T) = Pr(Reach(s, T))$$

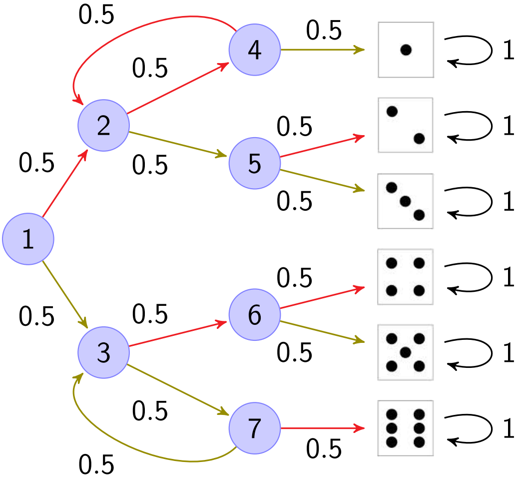
where $Reach(s, T) = \{s_0 s_1 \ldots \in Path(s) | \exists i : s_i \in T\}$

Question: Is $Reach(s, T)$ measurable? **Yes!**

Let $Path_{fin}(s, T) = \{\omega \in Path(s) : \exists i.\omega[i] \in T \wedge \forall j < i.\omega[j] \notin T\}$:

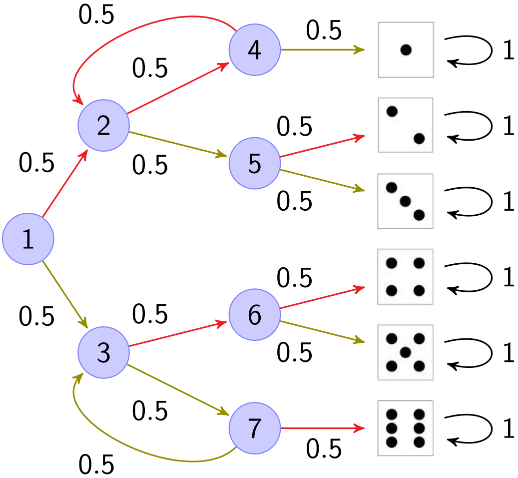
- $Path_{fin}(s, T)$ is enumerable;
- for each $\omega_1, \omega_2 \in Path_{fin}(s, T)$ $(\omega_1 \neq \omega_2)$, $Cyl(\omega_1) \cap Cyl(\omega_2) = \emptyset$;
- $Reach(s, T) = \bigcup_{\omega \in Path_{fin}(s,T)} Cyl(\omega)$;
- $Pr(Reach(s, T)) = \sum_{\omega \in Path_{fin}(s,T)} Pr(Cyl(\omega))$.

Example: Knut-Yao Algorithm



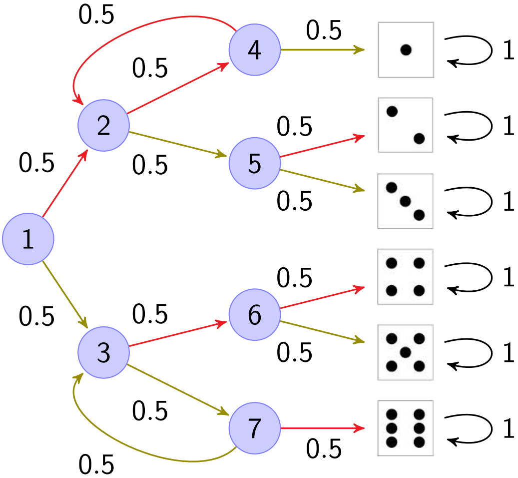
Example: Knut-Yao Algorithm

$ProbReach(s_1, ⚅) =$



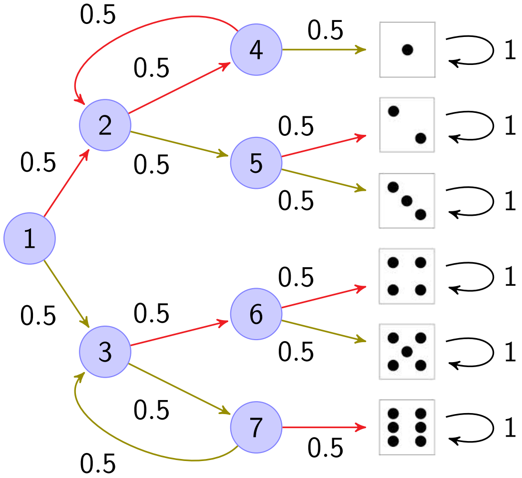
Example: Knut-Yao Algorithm

$ProbReach(s_1, ⚅) =$
$Pr(Cyl(s_1 s_3 s_7 ⚅))$



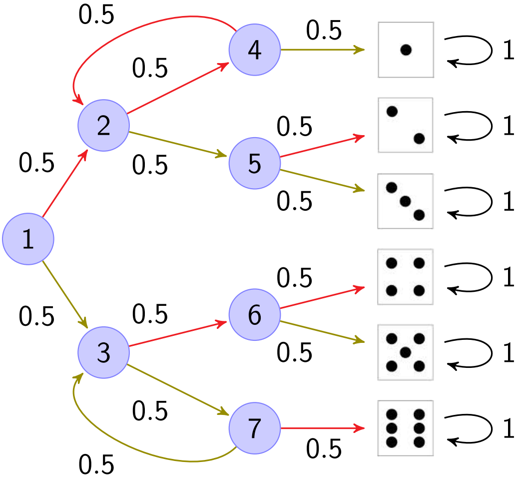
Example: Knut-Yao Algorithm

$$
\begin{aligned}
&ProbReach(s_1, \text{⚅}) = \\
&\quad Pr(Cyl(s_1 s_3 s_7 \text{⚅})) \\
&\quad + Pr(Cyl(s_1 s_3 s_7 s_3 s_7 \text{⚅}))
\end{aligned}
$$



Example: Knut-Yao Algorithm

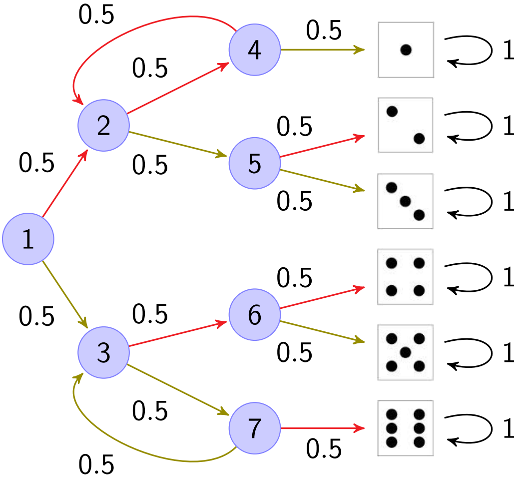
$$
\begin{aligned}
&ProbReach(s_1, \text{⚅}) = \\
&\quad Pr(Cyl(s_1 s_3 s_7 \text{⚅})) \\
&\quad + Pr(Cyl(s_1 s_3 s_7 s_3 s_7 \text{⚅})) \\
&\quad + \ldots
\end{aligned}
$$



Example: Knut-Yao Algorithm

$$ProbReach(s_1, ⚅) =$$
$$Pr(Cyl(s_1 s_3 s_7 ⚅))$$
$$+Pr(Cyl(s_1 s_3 s_7 s_3 s_7 ⚅))$$
$$+\ldots$$

We have to compute an infinite sum!



Computing Reachability Properties

We can use a system of linear equations to compute the reachability properties.

Computing Reachability Properties

We can use a system of linear equations to compute the reachability properties.

Following this approach we compute the reachability property for all the states in the system at the same time!

Computing Reachability Properties

We can use a system of linear equations to compute the reachability properties.

Following this approach we compute the reachability property for all the states in the system at the same time!

We let x_s denote the (unknown) value $ProbReach(s, T)$.

Computing Reachability Properties

We can use a system of linear equations to compute the reachability properties.

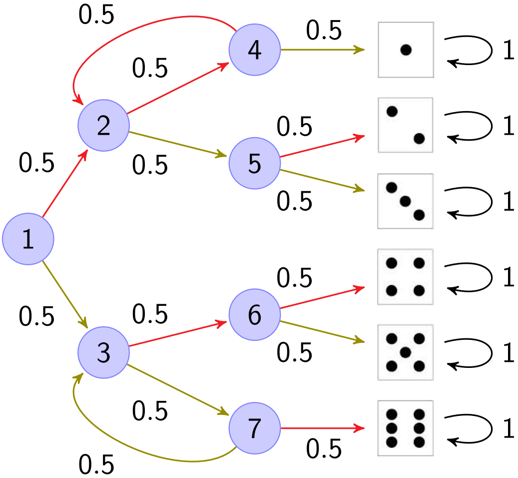
Following this approach we compute the reachability property for all the states in the system at the same time!

We let x_s denote the (unknown) value $ProbReach(s, T)$.

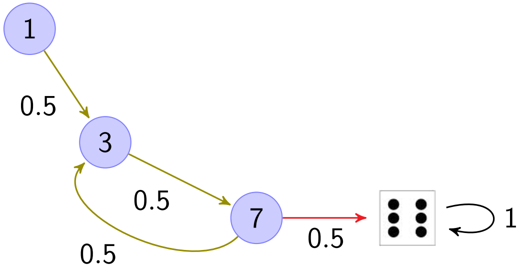
We solve the system of linear equations:

$$x_s = \begin{cases} 1 & s \in T \\ 0 & s \text{ cannot reach } T \\ \sum_{s' \in S} P(s, s') \cdot x_{s'} & \text{otherwise} \end{cases}$$

Example: Knut-Yao Algorithm

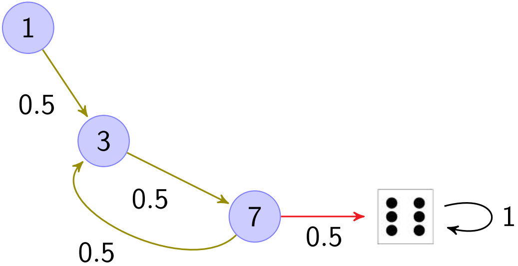


Example: Knut-Yao Algorithm



Example: Knut-Yao Algorithm

$$
\begin{aligned}
x_1 &= 0.5 \cdot x_3 \\
x_3 &= 0.5 \cdot x_7 \\
x_7 &= 0.5 \cdot x_3 + 0.5 \cdot x_{\text{⚅}} \\
x_{\text{⚅}} &= 1 \\
x_i &= 0
\end{aligned}
$$

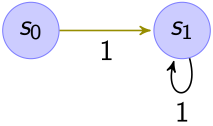


Unique solution

To guarantee the existence of an unique solution, we have to identify the states that cannot reach T.

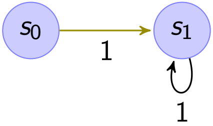
Unique solution

To guarantee the existence of an unique solution, we have to identify the states that cannot reach T.



Unique solution

To guarantee the existence of an unique solution, we have to identify the states that cannot reach T.

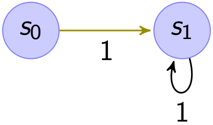


Let us compute the probability to reach $\{s_1\}$. If we do not remove states that are not able to reach s_0 we have the following system:

$$
\begin{aligned}
x_0 &= 1.0 \\
x_1 &= x_1
\end{aligned}
$$

Unique solution

To guarantee the existence of an unique solution, we have to identify the states that cannot reach T.



Let us compute the probability to reach $\{s_1\}$. If we do not remove states that are not able to reach s_0 we have the following system:

$$\begin{aligned} x_0 &= 1.0 \\ x_1 &= x_1 \end{aligned}$$

Any assignment $(x_0, s_1) = (0, p)$ (with $p \in [0, 1]$) is a valid solution!

Bounded Reachability probability

Let (S, P) be a DTMC and $T \subseteq S$, we let

$$ProbReach^{\leq k}(s, T) = Pr(Reach^{\leq k}(s, T))$$

where $Reach^{\leq k}(s, T) = \{s_0 s_1 \ldots \in Path(s) | \exists i \leq k : s_i \in T\}$

Bounded Reachability probability

Let (S, P) be a DTMC and $T \subseteq S$, we let

$$ProbReach^{\leq k}(s, T) = Pr(Reach^{\leq k}(s, T))$$

where $Reach^{\leq k}(s, T) = \{s_0 s_1 \ldots \in Path(s) | \exists i \leq k : s_i \in T\}$

Function $ProbReach^{\leq k}(s, T)$ can be recursively defined:

$$ProbReach^{\leq k}(s, T) = \begin{cases} 1 & s \in T \\ 0 & k = 0 \ \wedge s \notin T \\ \sum_{s' \in S} P(s, s') \cdot ProbReach^{\leq k-1}(s', T) & \end{cases}$$

To be continued...