Study of Markov Chains and Probabilistic Programming with Applications

Ghurye Sourabh Sunil and Nikhil Vyas
Guide : Prof. S. Akshay

IIT Bombay
# Table of Contents

**Introduction** .......................................................... 4

**Discrete Time Markov Chains** ..................................... 5
1 Definitions and Basic Properties .................................. 5
   1.1 Preliminaries .................................................. 5
   1.2 Definitions ................................................... 5
2 Class Structure and Periodicity ................................... 7
   2.1 Class structure as Equivalence relation .................... 7
   2.2 Periodicity .................................................... 8
3 Recurrence and Transience ......................................... 8
   3.1 Characterizations of Recurrence and Transience .......... 8
   3.2 Recurrence and transience as class properties .......... 10
4 Invariant Distributions ............................................ 10
   4.1 Existence and Uniqueness of Invariant Distribution .. 10
   4.2 Positive and Null Recurrence ............................... 14
   4.3 Ergodic Theorem ............................................. 15
   4.4 Convergence to an Equilibrium ............................ 16

**Continuous Time Markov Chains** ................................ 19
5 Definitions ......................................................... 19
6 Transient Behaviour of CTMC .................................... 19
   6.1 Uniformization ................................................ 20

**Markov Reachability Problem** .................................. 22
7 Cases with known solutions ....................................... 22
8 Cases with solutions not known .................................. 22

**Probabilistic Programming** ..................................... 23
9 Syntax and Semantics .............................................. 23
10 Examples ................................................................. 23
11 Encoding Observe Statement using Loops .......................... 24
12 Simulation of 6 faced fair Dice using DTMC and Probabilistic Programming ........................... 26
13 Simulation of Error .......................................................... 26
14 Sampling ........................................................................ 27
15 Inference for Probabilistic Program ....................................... 27
16 Challenges ..................................................................... 29
Introduction

A Markov chain (discrete-time Markov chain or DTMC), named after Andrey Markov, is a random process that undergoes transitions from one state to another on a state space. It must possess a property that is usually characterized as "memoryless": the probability distribution of the next state depends only on the current state and not on the sequence of events that preceded it. This specific kind of "memorylessness" is called the Markov property. Markov chains have many applications as statistical models of real-world processes.

A Markov chain is a stochastic process with the Markov property. The term "Markov chain" refers to the sequence of random variables such a process moves through, with the Markov property defining serial dependence only between adjacent periods (as in a "chain"). It can thus be used for describing systems that follow a chain of linked events, where what happens next depends only on the current state of the system.

Markov Chains are majorly of two types, Discrete and Continuous having transitions at fixed discrete times and continuous times respectively. We study them in Chapter 1 and 2 respectively.

Markov chains are an important concept in probability and many other areas of research. In this report, we formally define Markov chain and various concepts related to it. We also state and prove various theorems to study Markov chains.

Probabilistic Programming is a programming paradigm designed to describe probabilistic models and then perform inference in those models. It is closely related to graphical models and Bayesian networks, but is more expressive and flexible. Probabilistic programming represents an attempt to unify general purpose programming with probabilistic modeling. We describe it in Chapter 4.
1 Definitions and Basic Properties

We define some fundamental concepts related to Markov chain before proceeding to its formal definition. Reference to the upcoming sections are [3] and [4].

1.1 Preliminaries

Let $S = (s_0, s_1, \ldots)$ be a countable set then a distribution over $S$ is defined as function $f : S \rightarrow \mathbb{R}_{\geq 0}$ such that $\sum_{s \in S} f(s) = 1$. Generally $f(s)$ is written as $f_s$.

$\delta_i$ is a distribution over $\{0, 1, 2 \ldots k\}$ such that $(\delta_i)_j = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ if otherwise} \end{cases}$

1.2 Definitions

Let $I = \{i_0, i_1, \ldots\}$ be a countable set. Each $i \in I$ is known as a state and $I$ as the state space. Let $P = (p_{ij} : i, j \in I)$ be a row stochastic matrix, meaning that $p_{ij} \geq 0$ for all $i$ and $j$ and $\sum_{j \in I} p_{ij} = 1$.

**Definition 1.** We say that $(X_n)_{n \geq 0}$ is a Markov Chain with initial distribution $\lambda$ and Transition matrix $P$ if $j$ for all $n \geq 0$ and $i_0, i_1, \ldots, i_{n+1} \in I$.

1. $P(X_0 = i_0) = \lambda_{i_0}$
2. $P(X_{n+1} = i_{n+1}|X_n = i_n, \ldots, X_0 = i_0) = P(X_{n+1} = i_{n+1}|X_n = i_n) = p_{i_n i_{n+1}}$

Markov($\lambda, P$) denotes Markov chain with initial distribution $\lambda$ and Transition matrix $P$.

Now we formally define Markov chain and Markov property.

**Theorem 1.** $(X_n)_{n \geq 0}$ is Markov($\lambda, P$) iff for all $n \geq 0$ and $i_0, \ldots, i_n \in I$ then

\begin{equation}
P(X_0 = i_0, \ldots, X_n = i_n) = \lambda_{i_0}p_{i_0 i_1}\cdots p_{i_{n-1}i_n} \tag{1}
\end{equation}

**Proof.** Suppose $(X_n)_{n \geq 0}$ is Markov($\lambda, P$). Then

\begin{align*}
P(X_0 = i_0, \ldots, X_n = i_n) &= P(X_n = i_n|X_0 = i_0, \ldots, X_{n-1} = i_{n-1})P(X_0 = i_0, \ldots, X_{n-1} = i_{n-1}) \\
&= P(X_0 = i_0)P(X_1 = i_1|X_0 = i_0)\cdots P(X_n = i_n|X_0 = i_1, \ldots, X_{n-1} = i_{n-1}) \\
&= P(X_0 = i_0)P(X_1 = i_1|X_0 = i_0)\cdots P(X_n = i_n|X_{n-1} = i_{n-1}) \\
&= \lambda_{i_0}p_{i_0 i_1}\cdots p_{i_{n-1}i_n}
\end{align*}

For converse eq(1) is true, put $n = 0$ to get

\begin{equation}
P(X_0 = i_0) = \lambda_{i_0} \tag{2}
\end{equation}

Then summing (1) on $i_n$ we get

\begin{align*}
P(X_0 = i_0, \ldots, X_{n-1} = i_{n-1}) &= \lambda_{i_0}p_{i_0 i_1}\cdots p_{i_{n-2}i_{n-1}} \\
P(X_n = i_n|X_0 = i_0, \ldots, X_{n-1} = i_{n-1}) &= \frac{P(X_0 = i_0, \ldots, X_n = i_n)}{P(X_0 = i_0, \ldots, X_{n-1} = i_{n-1})} = p_{i_{n-1}i_n}
\end{align*}

$p_{i_{n-1}i_n}$ is independent of values $X_0, \ldots, X_{n-2}$

Hence, $P(X_{n+1} = i_{n+1}|X_n = i_n, \ldots, X_0 = i_0) = P(X_{n+1} = i_{n+1}|X_n = i_n) = p_{i_n i_{n+1}} \square$
Theorem 2. Markov property
Let \((X_n)_{n \geq 0}\) be Markov(\(\lambda, P\)). Then conditional on \(X_m = i\), \((X_{m+n})_{n \geq 0}\) is Markov\((\delta, P)\).

\[
P(X_{m+n} = i_{m+n}|X_m = i_m) = P(X_m = i_m|X_0 = i_m)
\]

(3)

Proof.

\[
P(X_{m+n} = i_{m+n}|X_m = i_m)
\]

\[
= \sum_{i_{m+1}, \ldots, i_{m+n-1} \in I} P(X_{m+n} = i_{m+n} = i_{m+n-1} \ldots X_m = i_m) P(X_{m+n-1} = i_{m+n-1}, \ldots X_{m+1} = i_{m+1}|X_m = i_m)
\]

\[
= \sum_{i_{m+1}, \ldots, i_{m+n-1} \in I} p_{i_{m+n-1}i_{m+n}} P(X_{m+n-1} = i_{m+n-1}, \ldots X_{m+1} = i_{m+1}|X_m = i_m)
\]

\[
= \sum_{i_{m+1}, \ldots, i_{m+n-1} \in I} p_{i_{m+n-1}i_{m+n}} \ldots p_{i_{m+1}i_{m+2}} P(X_{m+1} = i_{m+1}|X_m = i_m)
\]

\[
= \sum_{i_{m+1}, \ldots, i_{m+n-1} \in I} \sum_{i_{m+n}} p_{i_{m+n-1}i_{m+n}} \ldots p_{i_{m+1}i_{m+1}}
\]

Similarly

\[
P(X_n = i_{m+n}|X_0 = i_m) = \sum_{i_{m+1}, \ldots, i_{m+n-1} \in I} p_{i_{m+n-1}i_{m+n}} \ldots p_{i_{m+1}i_{m+1}}
\]

Hence \(P(X_{m+n} = i_{m+n}|X_m = i_m) = P(X_n = i_{m+n}|X_0 = i_m)\)

Here are some basic notations that we use in later part.

Definition 2. \(P_i(E)\) is the probability of occurrence of event \(E\) given the first state \(X_0 = i\).

\[
P_i(E) = P(E|X_0 = i)
\]

Definition 3. \(E_i(Z)\) is the expected value of random variable \(Z\) given the first state \(X_0 = i\).

\[
E_i(Z) = E(Z|X_0 = i)
\]

Theorem 3. Chapman-Kolmogorov Property Let, \(p_{ij} = P_i(X_1 = j)\) and \(p_{ij}^{(n)} = P_i(X_n = j)\)
Then for all \(i, j\) and \(n, m \geq 0\), Chapman-Kolmogorov equations hold :

\[
p_{ij}^{(n+m)} = \sum_{k \in I} p_{ik}^{(n)} p_{kj}^{(m)}
\]

(4)

Proof.

\[
p_{ij}^{(n+m)} = P(X_{n+m} = j|X_0 = i)
\]

\[
= \sum_{k \in I} P(X_{m+n} = j|X_n = k, X_0 = i) P(X_n = k|X_0 = i)
\]

\[
= \sum_{k \in I} P(X_{m+n} = j|X_n = k) P(X_n = k|X_0 = i)
\]

\[
= \sum_{k \in I} p_{ik}^{(n)} p_{kj}^{(m)}
\]
Corollary 1. For all $i, j$ and $n, m \geq 0$ the following inequality holds:

$$p_{ij}^{(n-m)} \geq \sum_{k \in I} p_{ik}^{(n)} p_{kj}^{(m)}$$

(5)

Proof. By Theorem 3

$$p_{ij}^{(n+m)} = \sum_{k \in I} p_{ik}^{(n)} p_{kj}^{(m)} \geq \sum_{k \in I} p_{ik}^{(n)} p_{kj}^{(m)} \text{ as } p_{ij} \geq 0 \text{ for all } i, j.$$  

(6)
Definition 6. Irreducibility
A Markov chain is said to be irreducible if for all i, j i ↔ j is true.

Definition 7. A equivalence class generated by the ↔ relation is known as a communicating class.

Definition 8. A Property is which if true for any element implies that it is true for all element of the corresponding communicating class is known as a class property.

2.2 Periodicity
Now we study how period of states in the same class are related. We also define aperiodic Markov chain.

Definition 9. The period of a state i denoted by $m_i$ is the greatest common divisor of the set \{n ∈ N : $p^n_{ij}(i,i) > 0$\}.

Theorem 6. Period is a class property
Proof. Let $S_k = \{n ∈ N : p^n_{kk} > 0\}$
Let i ↔ j. Then there exists paths of some lengths $a_{ij}$ and $a_{ji}$ from j to i and j to i respectively.
By the definition of period, $(a_{ij} + a_{ji}) \mod m_i = 0$
For any $s ∈ S_j (a_{ij} + s + a_{ji}) \mod m_i = 0$. Hence, $s \mod m_i = 0$
∴ $m_j \mod m_i = 0$. Hence $m_i = m_j$.

Definition 10. If any state of a closed class has a period 1 then that class is known as aperiodic class. An irreducible Markov Chain with a state of period 1 is known as aperiodic Markov Chain.

3 Recurrence and Transience
In this section we study the behaviour of return probability for a particular state. We also find the necessary and sufficient conditions for the sure return. We find the relation between return probabilities of states belonging to the same class.

3.1 Characterizations of Recurrence and Transience
Definition 11. Associated definitions :

\begin{align*}
H_i &= \inf \{n ≥ 0 : X_n = i\} = \text{hitting time on } i \\
T_i &= \inf \{n ≥ 1 : X_n = i\} = \text{first passage time to } i \\
V_i(k) &= \sum_{n=0}^{k} 1\{X_n=i\} = \text{number of visits to } i \text{ till } k \text{ steps} \\
V_i &= \sum_{n=0}^{\infty} 1\{X_n=i\} = \text{number of visits to } i \\
f_i &= P_i(T_i < \infty) = \text{return probability to } i \\
m_i &= E_i(T_i) = \text{mean return time to } i
\end{align*}
Definition 12. A state is said to be recurrent if the probability of return is 1 (i.e. \( f_i = 1 \)).

Definition 13. A state is said to be transient if the probability of return is less than 1 (i.e. \( f_i < 1 \)).

Lemma 1. Each state is either recurrent or transient.

Proof. As \( f_i \) is a probability hence either \( f_i = 1 \) (recurrent) or \( f_i < 1 \) (transient). \( \square \)

Theorem 7. A state is said to be recurrent iff any of the following equivalent conditions hold:

(i) \( P_i(V_i = \infty) = 1 \)

(ii) \( f_i = 1 \)

(iii) \( \sum_{n=0}^{\infty} p_{ii}^{(n)} = \infty \)

Proof. (i) \( \iff \) (ii)

\[
P_i(V_i = \infty) = 1 - \sum_{k=1}^{\infty} P_i(V_i = k)
= 1 - \sum_{k=1}^{\infty} (1 - f_i)f_i^{k-1}
= \begin{cases} 
1 & \text{if } f_i = 1 \\
0 & \text{if } f_i < 1
\end{cases}
\]

(ii) \( \iff \) (iii)

\[
\sum_{n=0}^{\infty} p_{ii}^{(n)} = E_i(V_i)
= \sum_{r=0}^{\infty} P_i(V_i > r)
= \sum_{r=0}^{\infty} f_i^r
= \frac{1}{1 - f_i}
= \begin{cases} 
\infty & \text{if } f_i = 1 \\
< \infty & \text{if } f_i < 1
\end{cases}
\]

\( \square \)

Theorem 8. A state is said to be transient iff any of the following equivalent conditions hold:
(i) \( P_i(V_i = \infty) = 0 \)

(ii) \( f_i < 1 \)

(iii) \( \sum_{n=0}^{\infty} p_{ii}^{(n)} < \infty \)

**Proof.** The conditions of Theorem 6 are necessary and sufficient for being recurrent. (i), (ii) and (iii) are complements of corresponding conditions of Theorem 6. Each state is either recurrent or transient by lemma 2. Hence, the conditions (i), (ii) and (iii) are necessary and sufficient for being transient. \( \square \)

### 3.2 Recurrence and transience as class properties

**Theorem 9.** *Transience is a class property*

**Proof.** Let C be a closed class and states \( i, j \in C \) and \( i \) is transient. As C is a closed class hence \( i \leftrightarrow j \) hence there exists \( n, m \) with \( p_{ij}^n > 0 \) and \( p_{ji}^m > 0 \). Then for all \( r \geq 0 \) by repeated application of corollary 1,

\[
p_{ii}^{n+m+r} \geq p_{ij}^n p_{jj}^r p_{ji}^m
\]

\[
\sum_{r=0}^{\infty} p_{jj}^r \leq \frac{1}{p_{ij}^n p_{ji}^m} \sum_{r=0}^{\infty} p_{ii}^{n+m+r} < \infty
\]

Hence by theorem 7, \( j \) is also transient. \( \square \)

**Theorem 10.** *Recurrence is a class property*

**Proof.** Let C be a communicating class and states \( i, j \in C \) and \( i \) is recurrent. Let \( j \) be transient. Hence \( i \) is transient by Theorem 8. Contradiction by lemma 2. Hence \( j \) is also recurrent. \( \square \)

### 4 Invariant Distributions

We study the equilibrium distribution of Markov chain in this section. We find the relation between the properties of Markov chain and existence of its equilibrium. We also try to find out if Markov chain has a unique fixed point. We prove some very important results in this section. References for this section are [3] and [4].

#### 4.1 Existence and Uniqueness of Invariant Distribution

**Definition 14.** \( \gamma_i^k \) represents mean number of visits to \( i \) between successive visits to \( k \).

\[
\gamma_i^k = \frac{T_k}{\sum_{n=0}^{T_k-1} 1 \{X_n = i\}}
\]

**Lemma 2.** \( \sum_{j \in I} \gamma_j^i = m_i \)
Proof. $m_i = E_i(T_i) = E_i \sum_{n=0}^{T_i-1} 1 = E_i \sum_{n=0}^{T_i-1} \{X_n=j\} = \sum_{j \in I} E_i(\sum_{n=0}^{T_i-1} 1_{\{X_n=j\}}) = \sum_{j \in I} \gamma_j^i$. \hfill \Box

**Theorem 11.** Let $P$ be irreducible and recurrent. Then

(i) $\gamma_k^i = 1$

(ii) $\gamma^k = (\gamma_k^i : i \in I)$ satisfies $\gamma^k P = \gamma^k$

(iii) $0 < \gamma_k^i < \infty$ for all $i$ and $k$

Proof. (i) $\gamma_k^i = E_k \sum_{n=0}^{T_k-1} 1_{\{X_n=i\}} = 1 + 0 + 0 + \ldots + 0 = 1$.

(ii) For $n = 1, 2, \ldots$ consider the event $n - 1 < T_k$

This event only depends on $X_0, X_1, \ldots, X_{n-1}$

So, by the Strong Markov property at $n - 1$,

$$P_k(X_{n-1} = i, X_n = j \text{ and } n - 1 < T_k) = P_k(X_{n-1} = i \text{ and } n - 1 < T_k) p_{ij}.$$

Since $P$ is recurrent, we have $P_k(T_k < \infty) = 1$. This means that we can partition the entire sample space by events of the form $\{T_k = t\}$, $t = 1, 2, \ldots$. Also, $X_0 = XT_k = k$ with probability 1.

So for all $j$ (including $j = k$)

$$\gamma_j^k = E_k \sum_{n=1}^{T_k} 1_{\{X_n=j\}}$$

$$= E_k \sum_{n=1}^{\infty} 1_{\{X_n=j \text{ and } n-1<T_k\}} \quad (\text{Since } T_k \text{ is finite})$$

$$= E_k \sum_{i \in I} \sum_{n=1}^{\infty} 1_{\{X_n=j, X_{n-1}=i \text{ and } n-1<T_k\}}$$

$$= \sum_{i \in I} \sum_{n=1}^{\infty} P_k(X_n = j, X_{n-1} = i \text{ and } n - 1 < T_k)$$

$$= \sum_{i \in I} p_{ij} \sum_{n=1}^{\infty} P_k(X_{n-1} = i \text{ and } n - 1 < T_k)$$

$$= \sum_{i \in I} p_{ij} E_k \sum_{n=1}^{\infty} 1_{\{X_{n-1}=i \text{ and } n-1<T_k\}}$$

$$= \sum_{i \in I} p_{ij} E_k \sum_{m=0}^{\infty} 1_{\{X_m=i \text{ and } m<T_k\}}$$

$$= \sum_{i \in I} p_{ij} E_k \sum_{m=0}^{T_k-1} 1_{\{X_m=i\}}$$

$$= \sum_{i \in I} \gamma_j^k p_{ij}$$

Hence, $\gamma_j^k = \gamma^k P$ \hfill \Box

**Definition 15.** An Invariant measure is any vector $\lambda$ such that $\lambda P = \lambda$. 
**Definition 16.** An Invariant distribution is any vector $\lambda$ such that $\lambda P = \lambda$ and $\lambda$ is a distribution.

**Theorem 12.** Let $P$ be irreducible and recurrent. Let $\lambda$ be an invariant measure for $P$ with $\lambda_k = 1$. Then $\lambda = \gamma_k$.

**Proof.** For each $j \in I$ we have

$$
\lambda_j = \sum_{i_0 \in I} \lambda_{i_0} P_{i_0 j} \\
= p_{kj} + \sum_{i_0 \neq k} \lambda_{i_0} P_{i_0 j} \\
= p_{kj} + \sum_{i_0 \neq k} p_{k i_0} P_{i_0 j} + \sum_{i_0, i_1 \neq k} \lambda_{i_1} P_{i_1 i_0} P_{i_0 j} \\
= p_{kj} + \sum_{i_0 \neq k} p_{k i_0} P_{i_0 j} + \sum_{i_0, i_1 \neq k} p_{k i_1} P_{i_1 i_0} P_{i_0 j} + ... \\
+ \sum_{i_0, i_1, ..., i_{n-1} \neq k} p_{k i_{n-1}} ... P_{i_1 i_0} P_{i_0 j} + \sum_{i_0, i_1, ..., i_n \neq k} \lambda_{i_n} P_{i_n i_{n-1}} ... P_{i_1 i_0} P_{i_0 j} + ...
$$

$$
\geq P_k (X_1 = j \text{ and } T_k \geq 1) + P_k (X_2 = j \text{ and } T_k \geq 2) + ... \\\n+ P_k (X_n = j \text{ and } T_k \geq n) \\
\rightarrow \gamma_k^j \text{ as } n \rightarrow \infty
$$

So $\lambda \geq \gamma_k$.

As $P$ is recurrent $\gamma_k$ is invariant by Theorem 10, so $\mu = \lambda - \gamma_k$ is also invariant and $\mu \geq 0$.

Since $P$ is irreducible, given $i \in I$, we have $p_{ik}^{(n)} > 0$ for some $n$, and

$$
0 = \mu_k = \sum_j \mu_j p_{jk} \geq \mu_i p_{ik}
$$

So $\mu_i = 0$ for all $i \implies \lambda = \gamma_k$.

Note: Theorem 10 and 11 also work for a finite irreducible Markov chain as any finite irreducible Markov chain is recurrent. In next section we provide alternative proofs for finite irreducible Markov chains.

**Lemma 3.** Transition matrix $P$ has an left eigenvalue of 1

**Proof.** 1 is an right eigenvalue of $P$ as $P \mathbf{1} = \mathbf{1}$ where $\mathbf{1}$ is a vector of 1’s. The set of right eigenvalues and left eigenvalues are same hence 1 is a left eigenvalue.

**Lemma 4.** All eigenvalues of Transition matrix $P$ satisfy $|\lambda| \leq 1$

**Proof.** Let $x$ be an eigenvector with eigenvalue $\lambda$ and $y_m$ be the max value (w.r.t. modulus) of $y$. Then
\(|(Px)_i| = |\sum_{j\in I} p_{ij}x_j|\\ 
\leq \sum_{j\in I} |p_{ij}x_j|\\ 
\leq (\sum_{j\in I} p_{ij})x_m\\ 
\leq x_m\\ 
\therefore |\max(Px)| \leq |\max(x)|\\ 
By definition, Px = \lambda x.\\ 
\therefore |\max(\lambda x)| \leq |\max(x)|\\ 
\implies |\lambda||\max(x)| \leq |\max(x)|\\ 
\implies |\lambda| \leq 1\\ 
\square

**Theorem 13. Perron-Frobenius Theorem**

Let T be a non negative irreducible matrix, then

- T has a positive (real) eigenvalue \(\lambda_{\max}\) such that all other eigenvalues of T satisfy \(|\lambda| \leq \lambda_{\max}\)
- The eigen vector \(x\) corresponding to \(\lambda_{\max}\) has algebraic and geometric multiplicity of 1.

**Proof.** Let \(P = (I + T)^k\) where k is choosen such that P is a positive matrix. This can be done for some \(k < n\) as T is irreducible.

Let \(z\) be any distribution. Then define

\[ L(z) = \max\{s : sz \leq Tz\} \]

By definition, \(L(rz) = L(z)\) for any \(r \geq 0\).

As \(u \leq v\) implies \(Pu \leq Pv\) hence,

\[ L(z)z \leq Tz \]
\[ L(z)Pz \leq PTz \]
\[ L(z)Pz \leq TPz \]
\[ L(z) \leq L(Pz) \]

Furthermore, equality is only achieved if \(z\) is an eigenvector with \(L(z)\) as the corresponding eigenvalue. Hence maximum of \(L\) over all distributions will be achieved at an eigenvector.

Let the corresponding eigenvalue be \(\lambda_{\max}\)

Let \(y\) be an eigenvector and \(\lambda\) be the corresponding eigenvalue then,

\[ Ty = \lambda y \]
\[ \lambda y_i = \sum_j T_{ij}y_j \]
\[ |\lambda||y_i| \leq \sum_j T_{ij}|y_j| \]
\[ |\lambda||y| \leq T|y| \]
Hence $|\lambda| \leq L(|y|) \leq \lambda_{\text{max}}$.

If $S \leq T$, $S \neq T$ and $z$ is a distribution then $Sz < Tz$. Hence $L_S(z) < L_T(z)$ which implies that all eigenvalues of $S$ are less than $\lambda_{\text{max}}$.

\[
\frac{d}{d\lambda} \det(\lambda I - T) = \sum_i \det(\lambda I - T_i)
\]

where $T_i$ is the matrix formed by setting the $i$-th row and column to 0. At $\lambda = \lambda_{\text{max}}$ RHS is positive as each term in sum is strictly positive. This is due to the fact that magnitude of each eigenvalue of $T_i$ is less than $\lambda_{\text{max}}$.

Hence the derivative of characteristic polynomial is positive at $\lambda_{\text{max}}$ hence the algebraic and geometric multiplicities corresponding to $\lambda_{\text{max}}$ is 1.

**Theorem 14.** Any irreducible markov chain has an unique stationary distribution.

*Proof.* The transition matrix $P$ is a non negative irreducible matrix.

1 is an eigenvalue of $P$ by Lemma 3

$|\lambda| \leq 1$ and 1 is an eigenvalue $\therefore \lambda_{\text{max}} = 1$

Hence by Theorem 12, $\pi P = \pi$ has a solution with geometric multiplicity 1.

In other words there exists a unique invariant distribution. \(\Box\)

### 4.2 Positive and Null Recurrence

We classify the recurrent state as positive or null on the basis of expected return time. We also characterise them as class properties.

**Positive Recurrent:**

The state is recurrent iff $P_i(T_i < \infty) = 1$.

If in addition the **mean return time** $m_i = E_i(T_i)$ is finite then we say that the state is positive recurrent.

**Null Recurrent:**

A recurrent state with $m_i = \infty$ is said to be null recurrent.

**Lemma 5.** A recurrent state is either positive recurrent or null recurrent.

*Proof.* As in a recurrent state either $m_i = \infty$ (positive recurrent) or $m_i < \infty$ (null recurrent). \(\Box\)

**Theorem 15.** Let $P$ be irreducible then following are equivalent:

(i) every state is positive recurrent;

(ii) some state $i$ is positive recurrent;

(iii) $P$ has an invariant distribution, $\pi$ say.

Moreover, when (iii) holds we have $m_i = 1/\pi_i$ for all $i$.

*Proof.* (i) $\implies$ (ii)

As every state is recurrent some state $i$ will be recurrent.

(ii) $\implies$ (iii) If $i$ is positive recurrent, it is certainly recurrent, so $P$ is recurrent by Theorem 9.

By Theorem 10, $\gamma_i$ is then invariant. But

$\sum_{j \in \mathcal{I}} \gamma_{ij} = m_i < \infty$ by lemma 1

so $\pi_k = \gamma_{ik} / m_i$ defines an invariant distribution.
(iii) $\implies$ (i) Since $\sum_{i \in I} \pi_i = 1$ we have $0 < \pi_k < \infty$ for some $k$.

Pick that $k$. Set $\lambda_i = \pi_i/\pi_k$.

Then $\lambda$ is an invariant measure with $\lambda_k = 1$. So by Theorem 5, $\lambda = \gamma_k$. Hence,

$$m_k = \sum_{i \in I} \gamma_i^k \leq \sum_{i \in I} \frac{\pi_i}{\pi_k} = \frac{1}{\pi_k} < \infty$$

hence the $k$th state is recurrent. \hfill \qed

**Theorem 16.** Positive recurrence is a class property

**Proof.** As (ii) $\implies$ (i) in Theorem 12. \hfill \qed

**Theorem 17.** Null recurrence is a class property

**Proof.** Let $C$ be a communicating class and states $i, j \in C$ and $i$ is null recurrent. By Theorem 11 $j$ is also recurrent. Let $j$ be positive recurrent then $i$ is also positive recurrent by Theorem 13. Contradiction.

Hence $j$ is also Null recurrent. \hfill \qed

### 4.3 Ergodic Theorem

**Theorem 18.** Ergodic Theorem

Let $P$ be irreducible and let $\lambda$ be any distribution. Suppose $(X_n)_{0 \leq n \leq N}$ is Markov$(\lambda, P)$. Then

$$P\left( \frac{n}{V_i(n)} \to m \text{ as } n \to \infty \right)$$

where $m_i = E_i(T_i)$ is expected return time to state $i$.

**Note:** We do not need the state to be aperiodic.

**Proof.** Let $i$ be a transient state then by definition total visits to $i$ is finite hence

$$\frac{V_i(n)}{n} \leq \frac{V_i}{n} \to 0 = \frac{1}{m_i} \text{ as } n \to \infty$$

Where $V_i(n) = \text{Number of visits to } i \text{ till } n \text{ steps}$ and $V_i = \text{Total visits to } i$.

Let $P$ be recurrent and $i$ be any state $P(T_i < \infty) = 1$ by theorem definition and $(X_{T_i+n})_{n \geq 0}$ is Markov$(\delta_i, P)$ by strong Markov property. so the long run portion of time spent in a state is independent of initial distribution hence in suffices to consider $\lambda = \delta_i$.

Let,

$$T_i^1 = \inf\{n \geq 1 : X_n = i\}$$

and inductively,

$$T_i^{r+1} = \inf\{n \geq T_i^r + 1 : X_n = i\}$$

The length of the $r$th excursion $(S_i^r)$ is then,

$$S_i^r = \begin{cases} T_i^r - T_i^{r-1} & \text{if } T_i^r < \infty \\ 0 & \text{if otherwise} \end{cases}$$
By Strong Markov property all $S_i^r$ are i.i.d. random variables for fixed $i$ with $E_i[S_i^r] = m_i$. Now,

$$S_i^1 + \ldots + S_i^{V_i(n)-1} \leq n - 1 \text{ and,}$$

$$S_i^1 + \ldots + S_i^{V_i(n)} \geq n$$

Hence,

$$\frac{S_i^1 + \ldots + S_i^{V_i(n)-1}}{V_i(n)} \leq \frac{n}{V_i(n)} \leq \frac{S_i^1 + \ldots + S_i^{V_i(n)}}{V_i(n)}$$

as $P$ is recurrent $\lim_{n \to \infty} V_i^n = \infty$. Hence as $n \to \infty$, by law of large numbers

$$P\left(\frac{n}{V_i(n)} \to m_i \text{ as } n \to \infty\right) = 1$$

which implies

$$P\left(\frac{V_i(n)}{n} \to \frac{1}{m_i} \text{ as } n \to \infty\right) = 1$$

$\Box$

### 4.4 Convergence to an Equilibrium

In this part, we study how the Markov chain converges to its equilibrium. We prove that the distance between current and equilibrium distribution will decrease if Markov chain possess certain properties.

**Lemma 6.** Let $\{m_i\}_{1 \leq i \leq r}$ be a finite subset of Natural Numbers whose GCD is 1.

$$\exists n_0 \text{ s.t. } \forall n \geq n_0 \quad n = \sum_{i=1}^{r} a_i m_i \quad \text{and} \quad a_i \geq 0$$

**Proof.** By Euclid’s Lemma :

1 = $\sum_{i=1}^{r} t_i m_i$ where $t_i \in \mathbb{Z}$

$t^* = \max_{1 \leq i \leq r} |t_i|$

$m^* = \sum_{i=1}^{r} m_i$

Any integer $n$ can be represented as
\[
\begin{align*}
    n &= km^* + s \\
    &= k \sum_{i=1}^{r} m_i + s \cdot 1 \\
    &= \sum_{i=1}^{r} km_i + s \sum_{i=1}^{r} t_i m_i \\
    &= \sum_{i=1}^{r} (k + st_i)m_i
\end{align*}
\]

If \( k \geq m^*t^* \) then \((k + st_i) \geq 0 \)

\( \forall n \geq (m^*)^2t^* \) \( \exists a_1, a_2, \ldots, a_r \geq 0 \) s.t. \( n = \sum_{i=1}^{r} a_i m_i \)

For any Probability distributions \( \nu, \mu \) distance function is defined as
\[
    d(\mu, \nu) = \frac{1}{2} \sum_{i \in x} |\mu_i - \nu_i|
\]

**Theorem 19.** Assume that the entries of \( P \) are all strictly positive. Then the mapping \( \nu^T \rightarrow \nu^T P \) is a strict contraction of the simplex relative to total variation distance, that is, there exists \( 0 < \alpha < 1 \) such that for any two probability vectors \( \mu, \nu \)
\[
    d(\mu^T P, \nu^T P) \leq \alpha d(\mu^T, \nu^T)
\]

**Proof.** Since every entry of \( P \) is strictly positive, there is a real number \( \epsilon > 0 \) such that \( p_{ij} \geq \epsilon \) for every pair of states \( i, j \). Notice that \( N \epsilon \leq 1 \), where \( N \) is the total number of states, because the row sums of are all 1.

We may assume (by choosing a slightly smaller value of \( \epsilon > 0 \)) that \( 1 - N \epsilon > 0 \).

Define \( q_{ij} = (p_{ij} - \epsilon)/(1 - N \epsilon) \), and let \( Q \) be the matrix with entries \( q_{ij} \).

Then \( Q \) is a stochastic matrix, because its entries are nonnegative (by the choice of \( \epsilon \)), and for every state \( i \),
\[
    \sum_{j} q_{ij} = (1 - N \epsilon)^{-1} \sum_{j} p_{ij} - (1 - N \epsilon^{-1}) \sum_{j} \epsilon = 1
\]

Let \( J \) be \( N \times N \) matrix with all entries 1.

Therefore, \( P = (1 - N \epsilon)Q + \epsilon J \) Now consider the total variation distance between \( \nu^T P \) and \( \mu^T P \). Using the fact that \( \sum_{j} \nu_j = \sum_{j} \mu_j = 1 \),

\[
    \sum_{i} (\mu^T P)_i - (\nu^T P)_i = \sum_{j} |(\mu^T P)_j - (\nu^T P)_j| \\
    = \sum_{j} |\sum_{i} (\mu_i p_{ij} - \nu_i p_{ij})| \\
    = \sum_{j} |\sum_{i} (\mu_i - \nu_i) q_{ij} (1 - N \epsilon)|
\]
Factor out $(1 - N\epsilon) =: \alpha$. What’s left is

\[
\sum_j | \sum_i (\mu_i - \nu_i)q_{ij} | \leq \sum_j \sum_i |(\mu_i - \nu_i)q_{ij} |
\]
\[
= \sum_i |(\mu_i - \nu_i)| \sum_j q_{ij}
\]
\[
= \sum_i |(\mu_i - \nu_i)|
\]
\[
= \sum_i (\mu_i^T - \nu_i^T)
\]
\[
= d(\mu^T, \nu^T)
\]

Hence, proved. □
Continuous Time Markov Chain

A continuous time Markov Chain has the same memorylessness property as discrete time Markov chains but the transitions can happen at any instant rather than at fixed equally separated points. Reference for this section is [4].

5 Definitions

Definition 17. A Continuous Time Markov Chain (CTMC) is a DES with the following features:

- The time interval between events $T_n$ is a random variable whose distribution is the exponential one and whose rate only depends on state $S_n$.
  More formally:
  \[
  \Pr(T_n \leq \tau | S_0 = s_0, ..., S_n = s, T_0 \leq \tau_0, ..., T_{n-1} \leq \tau_{n-1}) = \Pr(T_n \leq \tau | S_n = s_i) = 1 - \exp^{-\lambda_i \tau}
  \]
- The selection of the state that follows the current state only depends on that state and the transition probabilities remain constant along the run:
  \[
  \Pr(S_{n+1} = s_j | S_0 = s_{i_0}, ..., S_n = s_i, T_0 \leq \tau_0, ..., T_n \leq \tau_n) = \Pr(S_{n+1} = s_j | S_n = s_i) = p_{ij}
  \]

Definition 18. Embedded Chain The DTMC defined by transition matrix $P$ is called the embedded chain.

Embedded Chain observes the state changes of the CTMC without taking into account the time elapsed.
A state of the CTMC is absorbing if it is absorbing w.r.t. the embedded DTMC. The chain is said irreducible if the embedded chain is irreducible.

6 Transient Behaviour of CTMC

In a CTMC, due to the memoryless property of the exponential distribution, the evolution of the DES only depends on its current state.
Let $\pi_{ij}(\tau)$ denote the probability that at time $\tau$, the CTMC is in state $s_j$ knowing that at time 0 the CTMC is in state $s_i$.
Then by the memoryless property, $\pi_{ij}(\tau)$ is also the probability that at time $\Delta + \tau$, the CTMC is in state $s_j$ knowing that at time $\Delta$ the CTMC is in state $s_i$.
Thus the following equation is satisfied:
\[
\pi_{ij}(\Delta + \tau) = \sum_k \pi_{ik}(\Delta) \pi_{kj}(\tau)
\]

Let us define matrix $Q$ by:
\[
q_{ij} = \lambda_i p_{ij} \quad \text{for } i \neq j \quad \text{and} \quad q_{ii} = (p_{ii} - 1) \lambda_i = - \sum_{i \neq j} q_{ij}.
\]
Matrix $Q$ is called the infinitesimal generator of the CTMC.
The next proposition shows that the behaviour of the CTMC fulfills a backward differential equation system related to $Q$.

**Proposition:**
Let $C$ be a CTMC with $P$ its transition matrix and $\lambda$ its rate vector. The family of functions $\{\pi_{ij}\}_{ij}$ satisfies the following properties:

- For all $i = j$, $\lim_{\tau \to 0} \pi_{ii}(\tau) = 1$ and $\lim_{\tau \to 0} \pi_{ij}(\tau) = 0$
- For all $i, j$, $\pi_{ij}$ is differentiable and fulfills: $\frac{d \pi_{ij}(\tau)}{d\tau} = \sum_k q_{ik}\pi_{kj}(\tau)$

Introducing matrix $\Pi$ whose item $\Pi[i, j]$ is $\pi_{ij}$, the previous equation can be rewritten as: $\frac{d\Pi}{d\tau} = Q.\Pi$

### 6.1 Uniformization

It is simpler to analyse a CTMC if all the states have same transition rate. To achieve this we use the process called Uniformization.

Uniformization is a process of creating an equivalent CTMC with same $\lambda_i$ for all $i$ by adding self transitions.

**Construction:**

1. Choose a $\lambda$ such that $\lambda \geq \lambda_i$ for all $i$.
2. $P^*_{ii} = 1 - \frac{\lambda_i}{\lambda}$
3. $P^*_{ij} = \frac{\lambda}{\lambda} P_{ij}$ $(j \neq i)$

Observe that for $i \neq j$, $\lambda P^*_{ij} = \lambda \left(\frac{\lambda}{\lambda} P_{ij}\right) = \lambda_i P_{ij}$

For $i = j$ new CTMC has extra transitions while the old CTMC has none but as it does not affect the state hence is not of any concern. The self transitions are necessary for equalizing the $\lambda_i$’s.

Hence any CTMC with transition matrix $P$ with variable out rate $\lambda_i$ can be thought as a CTMC with transition matrix $P^*$ with a uniform out rate of $\lambda$.

Programming the new CTMC is more efficient and faster as all rates are same hence only values from one exponential distribution are needed.

**Theorem 20.** The stationary distribution of the CTMC $(P, \lambda_i)$ and the uniformized DTMC $(P^*)$ is the same.

**Proof.** By the property of Stationary Distribution in the DTMC,

$$\pi_j = \sum_{i \in I} \pi_i P^*_{ij}$$

By substituting the values, we get

$$\pi_j = \pi_j \left(1 - \frac{\lambda_j}{\lambda}\right) + \sum_{i \neq j} \pi_i P_{ij} \frac{\lambda_i}{\lambda}$$

$$\lambda_j \pi_j = \sum_{i \neq j} \pi_i P_{ij} \lambda_i$$
This is the equation for the stationary distribution of the CTMC. Hence, proved.
Markov Reachability Problem

Definition 19. Makov Reachability Problem
Given a finite row stochastic matrix $P$ with rational entries and a rational number $r \in [0, 1]$, does there exist an $n \in \mathbb{N}$ such that $(P^n)_{1,2} = r$
We can restate the problem to ask if there exists an $n \in \mathbb{N}$ such that $(\delta_1 P^n)_2 = r$

7 Cases with known solutions
For $r \neq \pi^2_1$ : For irreducible Markov Chains if $\pi^2_1 \neq r$ then there exists an $n_0$ such that for all $n > n_0 |P_1(X_n = 2) - r| \geq \epsilon$ where $\epsilon > 0$
Hence we can answer yes if there is an $n \leq n_0$ such that $P_1(X_n = 2) = r$ and no otherwise.

For $r = 0$ :
In any markov chain in a finite number of steps either a scc will have positive probability for all nodes. Hence if the probability is zero in any of the first $n^2$ steps then answer yes otherwise no.

For $r = 1$ :
If the probability is 1 after $n^2$ steps then answer yes otherwise no.

8 Cases with solutions not known
For $r = \pi^2_1$ the Markov Reachability Problem is equivalent to the Skolem Problem[1] and is yet unsolved.
Probabilistic Programming

Probabilistic programs are "usual" programs (written in languages like C, Java or ML) with two added constructs:

1. The ability to draw values at random from distributions.
2. The ability to condition values of variables in a program via observe statement.

We explain the terms used above in the next section syntax and semantics. Reference for this section is [2].

9 Syntax and Semantics

In this section, we proceed to give a precise syntax and semantics.

The meaning of a probabilistic program is the expected value of its return expression. The return expression of a program is a function from program states to non-negative reals. The denotational semantics \[ S(f)(\sigma) \] gives the expected value returned by a program with statement \( S \), return expression \( f \), and initial state \( \sigma \).

The probabilistic programming language that we consider is a C-like imperative programming language with two additional statements:

1. The probabilistic assignment " \( x \sim \text{Dist}(\theta) \) " draws a sample from a distribution " \( \text{Dist} \) " with a vector of parameters \( \theta \), and assigns it to the variable \( x \). For instance, the statement " \( x \sim \text{Gaussian}(\mu, \sigma) \) " draws a value from a Gaussian distribution with mean \( \mu \) and standard deviation \( \sigma \), and assigns it to the variable \( x \).

\[
\mathbb{J}_{x \sim \text{Dist}(\theta)} K(f)(\sigma) := \int_{v \in \text{Val}} \text{Dist}(\sigma(\theta))(v) \times f(\sigma[x \leftarrow v]) dv
\]

2. The observe statement " \( \text{observe}(\varphi) \) " conditions a distribution with respect to a predicate or condition \( \varphi \) that is defined over the variables in the program. In particular, every valid execution of the program must satisfy all conditions in observe statements that occur along the execution.

\[
\mathbb{J}_{\text{observe}(\varphi)} (f)(\sigma) := \begin{cases} 
  f(\sigma) & \text{if } \sigma(\varphi) = \text{true} \\
  0 & \text{otherwise}
\end{cases}
\]

Importance:

We get the output statistics by running the program large number of times. Each instance of output is nothing but the value of the return statement in a particular simulation run. For each run the values are selected from distributions without any bias. Hence the distribution of output is same as the distribution of the return statement. Hence by law of large numbers the observed distribution of output over finite runs tends to the distribution of return statement as the number of runs tend to infinity. Thus, Probabilistic programming provides very powerful semantics and therefore the correctness of results can be verified theoretically. This is the reason why the probabilistic programming is such an interesting and growing field.

10 Examples

Let us go through some examples.
1. The program tosses two fair coins (Bernoulli Distribution with 0.5) and assigns the value to the boolean variables c1, c2.

```c
bool c1, c2;
c1 = Bernoulli(0.5);
c2 = Bernoulli(0.5);
return(c1, c2);
```

The semantics of this program is the expectation of its return value. In this case, this is equal to (1/2, 1/2). Since we have that

\[
\Pr(c_1 = \text{false}, c_2 = \text{false}) = \Pr(c_1 = \text{false}, c_2 = \text{true}) = \Pr(c_1 = \text{true}, c_2 = \text{false}) = \Pr(c_1 = \text{true}, c_2 = \text{true}) = 1/4,
\]

Expectation on the return value is given by (treating true as 1 and false as 0):

\[
\frac{1}{4} \times (0, 0) + \frac{1}{4} \times (0, 1) + \frac{1}{4} \times (1, 0) + \frac{1}{4} \times (1, 1) = (1/2, 1/2).
\]

2. The program tosses two fair coins (Bernoulli Distribution with 0.5) and simulates the result given that at least one of them is Heads (true or 1).

```c
bool c1, c2;
c1 = Bernoulli(0.5);
c2 = Bernoulli(0.5);
observe(c1 || c2);
return(c1, c2);
```

The `observe` statement blocks runs which do not satisfy the boolean expression `c1 || c2` and does not permit those executions to happen. Executions that satisfy `c1 || c2` are permitted to happen. The semantics of the program is the expected return value, conditioned by permitted executions.

Since conditioning by permitted executions yields \( \Pr(c_1 = \text{false}, c_2 = \text{false}) = 0, \) \( \Pr(c_1 = \text{false}, c_2 = \text{true}) = \Pr(c_1 = \text{true}, c_2 = \text{false}) = \Pr(c_1 = \text{true}, c_2 = \text{true}) = 1/3, \) we have that the expected return value is

\[
0 \times (0, 0) + 1/3 \times (0, 1) + 1/3 \times (1, 0) + 1/3 \times (1, 1) = (2/3, 2/3).
\]

11 Encoding Observe Statement using Loops

This program also counts the number of coin tosses that result in the value true, and stores this count in the variable `count`. The semantics of the program is the expected return value, conditioned by permitted executions.

```c
bool c1, c2;
int count = 0;
c1 = Bernoulli(0.5);
if (c1) then
    count = count + 1;
c2 = Bernoulli(0.5);
if (c2) then
    count = count + 1;
observe(c1 || c2);
return(count);
```

Since conditioning by permitted executions yields \( \Pr(c_1 = \text{false}, c_2 = \text{false}) = 0, \) \( \Pr(c_1 = \text{false}, c_2 = \text{true}) = \Pr(c_1 = \text{true}, c_2 = \text{false}) = \Pr(c_1 = \text{true}, c_2 = \text{true}) = 1/3, \) we have that the expected return value is

\[
0 \times 0 + 1/3 \times 1 + 1/3 \times 1 + 1/3 \times 2 = 4/3.
\]
Loopy Probabilistic Program:
bool c1, c2;
int count = 0;
c1 = Bernoulli(0.5);
if (c1) then
count = count + 1;
c2 = Bernoulli(0.5);
if (c2) then
count = count + 1;
while !(c1 || c2) {
count = 0;
c1 = Bernoulli(0.5);
if (c1) then
count = count + 1;
c2 = Bernoulli(0.5);
if (c2) then
count = count + 1;
}
return(count);

The above program is equivalent to the previous one. The observe statement is encoded using While loop. The loop exists if the condition (c1||c2) is true. If not, it merely re-samples c1 and c2, re-calculates count and checks the condition (c1||c2) again.
12 Simulation of 6 faced fair Dice using DTMC and Probabilistic Programming

In this example, we simulate 6 faced dice as DTMC where bottom nodes represent the faces of dice.

Probabilistic Program to simulate above DTMC :

```c
int x = 0;
while (x < 11) {
    bool coin = Bernoulli(0.5);
    if(x=0)
        if (coin) x = 1 else x = 2;
    else if (x=1)
        if (coin) x = 3 else x = 4;
    else if (x=2)
        if (coin) x = 5 else x = 6;
    else if (x=3)
        if (coin) x = 1 else x = 11;
    else if (x=4)
        if (coin) x = 12 else x = 13;
    else if (x=5)
        if (coin) x = 14 else x = 15;
    else if (x=6)
        if (coin) x = 16 else x = 2;
}
return (x);
```

13 Simulation of Error

In real life situations, there can be an error in the data received. We can simulate the error by adding the extra variable(drawn from some known distribution) to the data. For example, the Age data with the error of $\pm 1$ year can be used as :

```
Age=age+Uniform(-1,1);
```
14 Sampling

Markov Chain Monte Carlo (MCMC) algorithms are widely used for performing the sampling. Metropolis-Hastings (MH) Algorithm is one of the most common MCMC algorithms.

Metropolis-Hastings Algorithm:

MH algorithm takes a target distribution $P(x)$ as input (in some implicit form where $P$ can be evaluated at every point) and returns samples that are distributed according to this target distribution. These samples can be used to compute any estimator such as expectation of a function with respect to the target distribution $P(x)$. Two key steps of the MH algorithm are:

1. Every sample for a variable $x$ is drawn from a proposal distribution $Q(x_{old} \rightarrow x_{new})$. This is used to pick a new value $x_{new}$ for the variable $x$ by appropriately perturbing its old value $x_{old}$.
2. A parameter $\beta$ is used to decide whether to accept or reject a new sampled value for $x$, and is defined as follows:
   $\beta = \min\left\{1, \frac{P(x_{new})Q(x_{new} \rightarrow x_{old})}{P(x_{old})Q(x_{old} \rightarrow x_{new})}\right\}$

The sample is accepted if a random value drawn from the uniform distribution over $[0, 1]$ is less than $\beta$, otherwise it is rejected.

15 Inference for Probabilistic Program

A variety of inference techniques have been implemented in probabilistic programming systems.

- Static Inference:
  The semantics of a probabilistic program can be calculated using data flow analysis. The data flow facts here are probability distributions and they can be propagated by symbolically executing each statement, merging the data flow facts at join points, and performing fix points at loops.

  We study static analysis with the example of Bayesian Network but let us define it first. We will define it in short for our example and won’t go in details.

Bayesian Network:

A Bayesian network, Bayes network, belief network, Bayes(ian) model or probabilistic directed acyclic graphical model is a probabilistic graphical model (a type of statistical model) that represents a set of random variables and their conditional dependencies via a directed acyclic graph (DAG).

Here is an example:

```
Difficulty  Intelligence

Grade   SAT
```

Let D, I, G, S be the random variables associated with the states Difficulty, Intelligence, Grade and SAT respectively.

Now we simulate the above Bayesian Networks using the Probabilistic program.

Consider the following two programs. The second program is obtained after applying static inference technique.
bool i, d, s, g;
i = Bernoulli(0.3);
d = Bernoulli(0.4);
if (!i && !d)
g = Bernoulli(0.7)
else if ( i && !d)
g = Bernoulli(0.95)
else if (i && d)
g = Bernoulli(0.1)
else
g = Bernoulli(0.5)
if (!i)
s = Bernoulli(0.05)
else
s = Bernoulli(0.8)
return(s);

The above code can be converted into the code below because SAT score(s) is independent of Difficulty and Grade.

bool i, d, s, g;
i = Bernoulli(0.3);
d = Bernoulli(0.4);
if (!i)
s = Bernoulli(0.05)
else
s = Bernoulli(0.8)
return(s);

This was about the static inference.

- **Dynamic Inference**:
  Dynamic approaches (which are also called sampling based approaches) are widely used, since running a probabilistic program is natural, regardless of the programming language used to express the program. The main challenge in this setting is that many samples that are generated during execution are ultimately rejected for not satisfying the observations. This is analogous to rejection sampling for standard probabilistic models. In order to improve efficiency, it is desirable to avoid generating samples that are later rejected, to the extent possible.

  Dynamic Inference is consist of two steps:
  1. Propagation of observations back through the program using the pre-image operation to place an observe statement immediately next to every probabilistic assignment. This transformation preserves program semantics and helps perform efficient sampling.
  2. Perform a modified Metropolis-Hastings (MH) sampling over the transformed probabilistic program. The modifications exploit the structure in the transformed programs that observe statements immediately follow probabilistic assignments, and sample from sub-distributions in order to avoid rejections.
Let us see one example. Here is the burglar alarm simulation.

```c
bool earthquake, burglary, alarm, phoneWorking, maryWakes, called;
earthquake = Bernoulli(0.001);
burglary = Bernoulli(0.01);
alarm = earthquake || burglary;
if (earthquake)
    phoneWorking = Bernoulli(0.6);
else
    phoneWorking = Bernoulli(0.99);
if (alarm && earthquake)
    maryWakes = Bernoulli(0.8);
else if (alarm)
    maryWakes = Bernoulli(0.6);
else
    maryWakes = Bernoulli(0.2);
called = maryWakes && phoneWorking;
observe(called);
return burglary;
```

After pushing the observe statements up without changing the semantics, we get

```c
bool earthquake, burglary, alarm, phoneWorking, maryWakes, called;
earthquake = Bernoulli(0.001);
burglary = Bernoulli(0.01);
alarm = earthquake || burglary;
if (earthquake)
    phoneWorking = Bernoulli(0.6);
    observe(phoneWorking);
else
    phoneWorking = Bernoulli(0.99);
    observe(phoneWorking);
if (alarm && earthquake)
    maryWakes = Bernoulli(0.8);
    observe(maryWakes && phoneWorking);
else if (alarm)
    maryWakes = Bernoulli(0.6);
    observe(maryWakes && phoneWorking);
else
    maryWakes = Bernoulli(0.2);
    observe(maryWakes && phoneWorking);
called = maryWakes && phoneWorking;
return burglary;
```

16 Challenges

There are some challenges we face in the field of Probabilistic Programming.

– Modelling Continuous Time:

We simulate time in Continuous time model e.g., CTMC using uniformisation and time as a random variable. But this approach doesn’t seem to work in all cases. More native encoding of time will be beneficial but it seems to complicate semantics and cause the explosion of overall complexity.
– **Non-Determinism**:
Non-determinism is a powerful modelling tool to deal with unknown information, as well as to specify abstractions in situations where details are unimportant. Currently, most probabilistic programming systems are unable to represent (and hence analyse) such a model.
References

1. S. Akshay, Timos Antonopoulos, Joel Ouaknine, and James Worell. Reachability problems for markov chains.