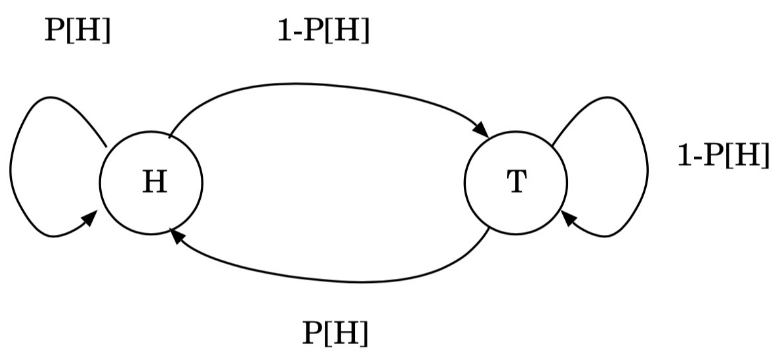
Machine Learning 2018.3

> - **Lista 5**
> - **Lucas Lopes Felipe** - *PPGI*
> - **Turma**: *CPS-863 / COS-623 / MAB-608*
> - **Professores**: Edmundo, Daniel S. Menascé e Rosa Leão

Questão 2

> In class we described the coin tossing toy example summarized in what follows. Suppose that a person is tossing a coin using 2 biased coins. Assume you know that the person changes from one coin to another with probability 0.2 after each coin flipping. Unfortunately, you do not know the probabilities that each coin flipping results in a head (H) or a tail (T).
>
> Your job in this question is to calculate the probability of observing a given sequence of heads and tails. Since you know nothing about the coins, you build a simple Markov model that should, you hope, explain the sequence of observations. The model is shown below:



In [1]:
```
import numpy as np

def matriz_transicao(p_H):
    #   [ H ,    T    ]
    h = [p_H, 1 - p_H]
    t = [p_H, 1 - p_H]
    return np.array([h,t])

def matriz_pi(num_estados):
    prob_identica = 1 / num_estados
    return np.repeat(prob_identica, num_estados)

def markov_prob(trans, pi, amostra):
    inicio = (amostra[0] == 'T') * 1
    prob   = pi[inicio]

    for i in range(1, len(amostra)):
        row = (amostra[i - 1] == 'T') * 1
        col = (amostra[i] == 'T') * 1
        prob *= trans[row, col]

    return prob
```

Suppose that you observe the sequence $HHHHHTTTTT$ and you decide to assign $P[H] = 0.5$ to your model.

In [2]:
```
transicao = matriz_transicao(0.5)
pi = matriz_pi(2)
```

1. **From the model, calculate the probability that the sequence $HHHHHTTTTT$ is observed.**

In [3]:
```
seq_1   = 'HHHHHTTTTT'
prob_1 = markov_prob(transicao, pi, seq_1)
print('\nA probabilidade da sequência HHHHHTTTTT ter sido gerado tal
 que P[H] = 0.5 é de:', prob_1)
```

```
A probabilidade da sequência HHHHHTTTTT ter sido gerado tal que P[H]
= 0.5 é de: 0.00048828125
```

1. **Repeat your calculations if, instead, you observe the sequence $HTHTHTHTHTH$.**

In [4]:
```
seq_2   = 'HTHTHTHTHTH'
prob_2 = markov_prob(transicao, pi, seq_2)
print('\nA probabilidade da sequência HTHTHTHTHTH ter sido gerado tal
 que P[H] = 0.5 é de:',prob_2)
```

```
A probabilidade da sequência HTHTHTHTHTH ter sido gerado tal que P[H]
= 0.5 é de: 0.00048828125
```

1. **Since you do not know if $P[H] = 0.5$ is a reasonable value to use, your job is to obtain $P[H]$, which maximizes the likelihood ($P(\mathcal{D} \mid \mathcal{M})$), for each of the sequences above.**

A probabilidade de ir do estado i para j é obtido através do número de vezes em que houve transição de i para j dividido pelo número total de vezes que saiu do estado i para qualquer outro estado. Representando pela seguinte [fórmula (https://www.stat.cmu.edu/~cshalizi/462/lectures/06/markov-mle.pdf)](https://www.stat.cmu.edu/~cshalizi/462/lectures/06/markov-mle.pdf):

$$p_{ij} = \frac{n_{ij}}{\sum_{j=1}^{m} n_{i,j}}$$

Como neste caso, todas as probabilidades de transição são em função da probabilidade de P[H], basta dividir o número de vezes que H aparece (exceto se este começar a sequência, pois não será uma transição e sim probabilidade de início), pelo tamanho da sequência -1 (total de transações).

Como desmontrado no seguinte função:

In [5]:
```
def mle(amostra):
    Hs = 0
    if amostra[0] == 'H': Hs -= 1
    for a in amostra:
        if a == 'H': Hs += 1
    return Hs / (len(amostra) - 1)
```

In [6]:
```
mle_1   = mle(seq_1)
trans_1 = matriz_transicao(mle_1)
print('\nSequência 1:\nP[H] =', mle_1,
      '\nLikelihood =', markov_prob(trans_1, pi, seq_1))
```

```
Sequência 1:
P[H] = 0.4
Likelihood = 0.0005971968
```

In [7]:
```
mle_2   = mle(seq_2)
trans_2 = matriz_transicao(mle_2)
print('\nSequência 2:\nP[H] =', mle_2,
      '\nLikelihood =', markov_prob(trans_2, pi, seq_2))
```

```
Sequência 2:
P[H] = 0.5
Likelihood = 0.00048828125
```

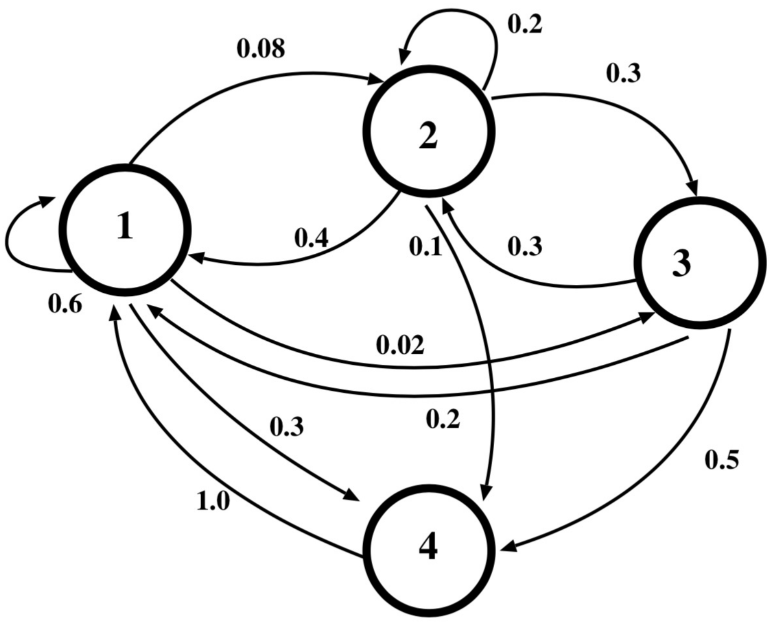
1. **Obtain the value for $P[H]$ that maximizes the likelihood for the sequence $HTHTHTTT$.**

In [8]:
```
seq_3   = 'HTHTHTTT'
mle_3   = mle(seq_3)
trans_3 = matriz_transicao(mle_3)
print('\nSequência 3:\nP[H] =', mle_3,
      '\nLikelihood =', markov_prob(trans_3, pi, seq_3))
```

```
Sequência 3:
P[H] = 0.2857142857142857
Likelihood = 0.007589160493137578
```

Questão 3

> Consider the Markov chain given below, that models some system. Suppose you observe the following sequence of states: $1, 2, 1, 3$ at instants t_1, t_2, t_3, t_4.



In [9]:
```
t = [0, 1, 2, 1, 3]

#    [  s1,    s2,    s3,    s4]
s1 = [0.60, 0.08, 0.02, 0.30]
s2 = [0.40, 0.20, 0.30, 0.10]
s3 = [0.20, 0.30, 0.00, 0.50]
s4 = [1.00, 0.00, 0.00, 0.00]

m_transicao = np.array([s1, s2, s3, s4])
```

1. **What is the most probable state that you should observe at instant t_5? (In this item assume that there is not data observed after t_4.)**

Como em *Markov Chains* basta saber o estado atual para calcular a probabilidade do próximo, o estado no instante t_5 será o que possui maior probabilidade a partir do estado que está no instante t_4 (estado 3).

O estado 3 pode ir para tais estados com as seguintes probabilidades:

- $3 \rightarrow 1 = 0.02$
- $3 \rightarrow 2 = 0.3$
- $3 \rightarrow 4 = 0.5$

Portanto, o estado mais provável na sequência, partido do 3, é o estado 4.

In [10]:
```
# O estado que está no presente é o valor que está em t[4] (index inic
ia em 1)
presente = t[4]

# A coluna que possui o maior valor na linha do estado presente será o
 mais provável
estado_provavel = m_transicao[presente - 1].argmax() + 1
# Subtrai-se 1 pois o estado S está na linha S - 1 (index começa em 0)
# Soma-se 1 pois o estado da linha L é o estado L + 1 (index começa em
 0)

print('O estado mais provável a partir do estado', t[4], 'é o estado:'
, estado_provavel)
```

```
O estado mais provável a partir do estado 3 é o estado: 4
```

1. **What is the most probable state that you should observe at instant t_6? (In this item assume that there is not data observed after t_4.)**

Novamente, se quisermos responder qual o estado mais provável no instante t_6 precisamos olhar para o mais provável a partir do estado anterior, que está em t_5.

Porém, há apenas dados observados até t_4, portanto, para inferir qual o mais provável em t_6 precisamos multiplicar:

- Probabilidade de ir do estado em t_4 (estado 3) para t_5 (os possíveis estados saindo do 3); pela:
- Probabilidade de transição entre este possível estado em t_5 e seu respectivo mais prováavel estado de destido em t_6.

O par de estados em t_4 e t_5 que resultar no maior produtório $(prob(t_4 \rightarrow t_5) * prob(t_5 \rightarrow t_6))$, será a sequência mais provável.

Os estados que podem sair do estado 3 e suas respetivas probabilidades são:

- $3 \rightarrow 1 = 0.02$
- $3 \rightarrow 2 = 0.3$
- $3 \rightarrow 4 = 0.5$

Agora vamos pegar o mais provável de cada um destes, bem como seu valor:

- $1 \rightarrow 1 = 0.6$
- $2 \rightarrow 2 = 0.4$
- $4 \rightarrow 1 = 1.0$

Basta multiplicar e ver a combinação de maior probabilidade:

- $3 \rightarrow 1 \rightarrow 1 = 0.02 * 0.6 = 0.012$
- $3 \rightarrow 2 \rightarrow 2 = 0.30 * 0.4 = 0.12$
- $3 \rightarrow 4 \rightarrow 1 = 0.50 * 1.0 = 0.5$

Logo, a sequência mais provável é: $t_4 = 3 \rightarrow t_5 = 4 \rightarrow t_6 = 1$

1. **Imagine that you keep observing the system but you miss 1 observation. That is, you observe:** $1, 2, 1, 3, X, 2, 1$

a) _Is the most probable state you guessed for instant t_5 (above) the answer for this question? (yes/no). Justifv vour answer.

Não.

Na questão anterior precisávamos considerar apenas os prováveis estados a partir de $t_4 = 3$, escolhendo aquele de maior probabilidade de transição que é $t_5 = 4$.

Agora, por sabermos que o estado posterior à t_5 é $t_6 = 2$, precisamos considerar a probabilidade não somente de $t_4 \rightarrow t_5$ mas também de $t_5 \rightarrow t_6$. Multiplicando as probabilidade de transição de ($t_4 = 3 \rightarrow t_5 = X) * (t_5 = X \rightarrow t_6 = 2)$. Onde X são todos os estados possíveis que saem do estado 3 e possam ir em seguida para o estado 2.

b) *Show how to obtain the most probable missing state observation.*

O estado perdido X precede do estado 3 e antecede do 2, portanto os possíveis estados estão na intercessão entre:

Estados possíveis a partir de 3:

- $3 \rightarrow 1 = 0.02$
- $3 \rightarrow 2 = 0.3$
- $3 \rightarrow 4 = 0.5$

Estados possíveis com destino ao 2:

- $1 \rightarrow 2 = 0.08$
- $2 \rightarrow 2 = 0.2$
- $3 \rightarrow 2 = 0.3$

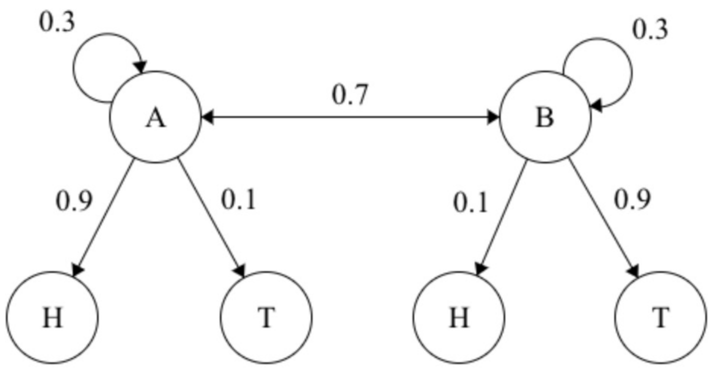
Os estados que pertencem a estes 2 grupos descritos acima são:

- $3 \rightarrow X \rightarrow 2$
- $3 \rightarrow 1 \rightarrow 2 = 0.02 * 0.08 = 0.0016$
- $3 \rightarrow 2 \rightarrow 2 = 0.30 * 0.20 = 0.06$

Logo, o mais provável é: $X = 2$ com probabilidade 0.06

Questão 4

> Refer to *Question 2*. A friend of yours said that she knows the bias of each coin. She said that, if one coin is tossed, the result is head with probability 0.9 (refer to this coin as coin A). The other coin (B) has only a 0.1 probability of resulting a head when tossed. In addition, you know that the person that tosses the coins changes from one coin to another with probability 0.7, after showing you the result.



> Since you have just learned about HMMs, you build a HMM and use the learned material to calculate the probability that each of the above sequences (sequence $HHHHHTTTTTT$ and $HTHTHTHTHTH$) is observed. Since you do not know which coin is tossed first your initial guess is that each coin is equally probable of being used for the first toss. To answer the questions below you may either implement the appropriate recursions your learned in class or **use the MTK module of the Tangram-II tool** (the manual is available), or an appropriate software package for HMMs (R for instance).

In [11]:
```
sq_1 = 'HHHHHTTTTTT'
sq_2 = 'HTHTHTHTHTH'

#      [   A,    B]
a_t = [0.3, 0.7]
b_t = [0.7, 0.3]

#      [   H,    T]
a_e = [0.9, 0.1]
b_e = [0.1, 0.9]

m_transicao = np.array([a_t, b_t])
m_emissao   = np.array([a_e, b_e])
p_inicio    = matriz_pi(2)
```

1. **Describe your calculations and compare the results with that obtained from the Markovian model. In other words, you should indicate which of the 2 models (MC or HMM) is the best to explain each of the 2 sequences above.**

In [12]:
```
def prob_amostra(T, E, pi, amostra):
    forward = np.zeros((len(amostra), len(pi)))

    for i in range(len(pi)):
        forward[0,i] = pi[i] * E[i, (amostra[0] == 'T') * 1]

    for t in range(len(amostra) - 1):
        for i in range(len(pi)):
            soma = 0
            for j in range(len(pi)):
                soma += forward[t,j] * T[j, i]
            forward[t+1,i] = soma * E[i, (amostra[t+1] == 'T') * 1]

    prob = 0
    for i in range(len(pi)):
        prob += forward[-1,i]

    return prob, forward
```

In [13]:
```
prob_amostra(m_transicao, m_emissao, p_inicio, sq_1)[0]
```

Out[13]:
```
5.92535142432e-05
```

In [14]:
```
prob_amostra(m_transicao, m_emissao, p_inicio, sq_2)[0]
```

Out[14]:
```
0.005977272905936001
```

1. **Calculate the probability that coin A is the last one being tossed.**

In [15]:
```
def prob_ultimo(f):
    return f[-1,0] / (f[-1,0] + f[-1,1])
```

In [16]:
```
f_1 = prob_amostra(m_transicao, m_emissao, p_inicio, sq_1)[1]
f_2 = prob_amostra(m_transicao, m_emissao, p_inicio, sq_2)[1]

print('Probabilidade da moeda A ser a última da sequência', sq_1,
      'é de:', prob_ultimo(f_1))
print('Probabilidade da moeda A ser a última da sequência', sq_2,
      'é de:', prob_ultimo(f_2))
```

```
Probabilidade da moeda A ser a última da sequência HHHHHTTTTTT é de:
0.16216421630895447
Probabilidade da moeda A ser a última da sequência HTHTHTHTHTH é de:
0.9503400731145438
```

Questão 5

> Consider the model of *Question 4* to answer the questions below you may either implement the appropriate recursions your learned in class or **use the MTK module of the Tangram-II tool** (the manual is available).

1. **Suppose you observe the sequence $HTHHT$.**

In [17]:
```
seq_5_1 = 'HTHHT'
```

a) *What is the most probable sequence of coins if you assign the initial state probabilities as $\langle 0.5, 0.5 \rangle$ for coin A and B, respectively.*

Para calcular a probabilidade de uma sequência de observações ter sido gerada por uma sequência de amostras, basta multiplicar:

- Probabilidade do estado inicial; pela:
- Probabilidade de transição entre o estado anterior e atual; e
- Probabilidade de emissão deste estado para a observação no t atual.

E ir repetindo os 2 últimos passos (pois a probabilidade do estado inicial basta apenas uma vez) até chegar ao final da sequência.

Como por exemplo, supondo que queremos calcular a probabildiade da sequência observada ($H \rightarrow T$) ter sido gerada pela sequência de estados ($A \rightarrow B$), para isso devemos multiplicar:

- Probabilidade de iniciar com o primeiro da amostra (neste caso A, que tem a mesma probabilidade de início de B) $= 0.5$
- Probabilidade deste estado inicial (A) emitir a amostra visível (H) em $t_1 = 0.9$
- Probabilidade de transacionar entre estados (trocar da moeda A para B) $= 0.7$
- Probabilidade deste estado atual (B) emitir a amostra visível (T) em $t_2 = 0.9$

Multiplicando estes valores ($0.5 * 0.9 * 0.7 * 0.9$) teremos a probabilidade da amostra HT ter sido gerada pela sequência de estados AB que é 0.2835

Portanto, se quiseremos saber a sequência de estados que melhor produz a sequência de observações, devemos efetuar o cáculo descrito acima para cada uma das possíveis combinações de estados, onde, neste exemplo, como temos 2 estados (moeda A e moeda B) e 2 observações (H e T), a quantidade de combinações possíveis será $2^2 = 4$.

Já para as sequências dadas pelo problema, ambas possuem 11 amostras coletadas, logo, há $2^{11} = 2048$ diferentes combinações de estados possíveis que poderiam descrever tais amostra, cada um com sua probabilidade, e a que tiver o maior, será a mais provável.

Nota-se que seria extremamente custoso calcular a probabilidade de todas, pois é um problema que cresce exponecialmente. Para tanto, há uma forma menos custosa e que obtém o resultado precisando realizar menos cálculos, que é o algoritmo de *Viterbi*:

In [18]:
```
def viterbi(T, E, pi, amostra):

    probabilities = []
    if amostra[0] == 'H':
        probabilities.append((pi[0] * E[0,0], pi[1] * E[1, 0]))
    else:
        probabilities.append((pi[0] * E[0,1], pi[1] * E[1, 1]))

    for i in range(1,len(amostra)):
        prev_A, prev_B = probabilities[-1]

        if amostra[i] == 'H':
            now_A = max(prev_A * T[0,0] * E[0,0], prev_B * T[1,0] * E[
0,0])
            now_B = max(prev_A * T[0,1] * E[1,0], prev_B * T[1,1] * E[
1,0])
            probabilities.append((now_A, now_B))
        else:
            now_A = max(prev_A * T[0,0] * E[0,1], prev_B * T[1,0] * E[
0,1])
            now_B = max(prev_A * T[0,1] * E[1,1], prev_B * T[1,1] * E[
1,1])
            probabilities.append((now_A, now_B))

    coins = []
    probs = []
    for p in probabilities:
        if p[0] > p[1]:
            coins.append('A')
            probs.append(p[0])
        else:
            coins.append('B')
            probs.append(p[1])

    return coins, probs
```

In [19]:
```
prob_seq_1 = viterbi(m_transicao, m_emissao, p_inicio, seq_5_1)
print('\nA sequência de moedas que melhorer descreve a sequência', seq
_5_1, 'é:\n', *prob_seq_1[0])
```

```
A sequência de moedas que melhorer descreve a sequência HTHHT é:
 A B A A B
```

1. **Suppose you observe the sequence** $HTHTHTHTHTH$

In [20]:
```
seq_5_2 = 'HTHTHTHTHTH'
```

a) *What is the most probable sequence of coins if you assign the initial state probabilities as* $\langle 0.5, 0.5 \rangle$ *for coin* A *and* B*, respectively.*

In [21]:
```
prob_seq_2_a = viterbi(m_transicao, m_emissao, p_inicio, seq_5_2)
print('\nA sequência de moedas que melhorer descreve a sequência', seq
_5_2, 'é:\n', *prob_seq_2_a[0])
```

```
A sequência de moedas que melhorer descreve a sequência HTHTHTHTHTH
é:
 A B A B A B A B A B A
```

b) *Repeat the item above if the initial state probabilities are* $\langle 0.2, 0.8 \rangle$

In [22]:
```
new_pi = [0.2, 0.8]
prob_seq_2_b = viterbi(m_transicao, m_emissao, new_pi, seq_5_2)
print('\nA sequência de moedas que melhorer descreve a sequência', seq
_5_2,
      'com probabilidade inicial', new_pi, 'é:\n', *prob_seq_2_b[0])
```

```
A sequência de moedas que melhorer descreve a sequência HTHTHTHTHTH c
om probabilidade inicial [0.2, 0.8] é:
 A B A B A B A B A B A
```

c) *Compare the likelihood of the sequence in both models above (i.e., models with different initial state probabilities). Which model is the most appropriate to explain the observed sequence?*

In [23]:
```
from functools import reduce

likelihood_2_a = reduce(lambda x, y: x * y, prob_seq_2_a[1])
likelihood_2_b = reduce(lambda x, y: x * y, prob_seq_2_b[1])

print('\nLikelihood da sequência', seq_5_2,
     'com probabilidade inicial', p_inicio, 'é:\n', likelihood_2_a)
print('\nLikelihood da sequência', seq_5_2,
     'com probabilidade inicial', new_pi, 'é:\n', likelihood_2_b)
print('\nLogo, o modelo mais apropriado para explicar a observação', s
eq_5_2,
      'é o com probabilidade inicial:', p_inicio if likelihood_2_a > l
ikelihood_2_b else new_pi)
```

```
Likelihood da sequência HTHTHTHTHTH com probabilidade inicial [0.5 0.
5] é:
 1.409509059072285e-15

Likelihood da sequência HTHTHTHTHTH com probabilidade inicial [0.2,
0.8] é:
 5.911909484503104e-20

Logo, o modelo mais apropriado para explicar a observação HTHTHTHTHTH
é o com probabilidade inicial: [0.5 0.5]
```

Questão 6

[Dendroclimatology (http://en.wikipedia.org/wiki/Dendroclimatology)](http://en.wikipedia.org/wiki/Dendroclimatology) is the science of inferring past climates from trees, observing the properties of the annual tree rings. Studies have shown that there is a strong correlation among the size of a tree ring and the expected annual temperature. Scientist have used these findings to calculate the temperatures from thousands of years in the past. Since there is no temperature records from distance past, mathematical models such as that used below are very useful. In this question, our purpose is to use HMMs in this application.

First you need to build a HMM. For that, assume that you collected data on tree rings of many trees and have also available the temperature of each year in the recent past (say the last 50 years). To make this example simple, we assume that there are only two temperature levels: $C = cold$ and $H = hot$. In addition, we measure only three different ring sizes: $S = small$ and $M = medium$ and $L = large$. The following table lists the probabilities of the sizes of tree rings according to the temperature:

	S	M	L
H	0.05	0.40	0.55
C	0.80	0.10	0.10

An important observation is that recent data indicates that the sequence of temperatures at each year are correlated with that of previous year. For our problem assume that the probability of a hot (cold) year followed by another hot (cold) year is 0.75 (respectively 0.6)

1. If there was *no correlation* at all between the temperatures from one year to another, what would be the most probable sequence of temperatures if you observe a sequence of rings *SMSL*?

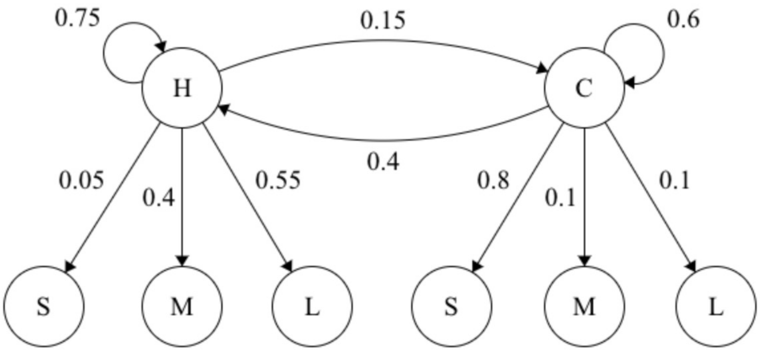
Seria a temperatura mais právovel para cada observação:

- $S \rightarrow C = 0.8$
- $M \rightarrow H = 0.4$
- $L \rightarrow H = 0.55$

Logo, para a sequência $SMSL$ a melhor sequência de temperatura seria $CHCH$

2. Well, you know that the temperatures are correlated from one year to another, and you need to build a HMM! What would be the choice for the hidden states? Note that you have available the size of tree rings from the distant past, but not the temperatures!

- Se não possuimos as temperaturas, logo, estas serão os estados ocultos, formando a matriz de transição.
- Se possumos os aneis das árvores, logo, estas serão os estados visíveis, formando a matriz de emissão.



In [24]:
```
#       [    H,     C]
h_t = [0.75, 0.15]
c_t = [0.40, 0.60]

#       [    S,     M,     L]
h_e = [0.05, 0.40, 0.55]
c_e = [0.80, 0.10, 0.10]

m_trasicao = np.array([h_t, c_t])
m_emissao  = np.array([h_e, c_e])
p_inicial  = matriz_pi(2)
```

3. We have already given you a model, but you should briefly describe the necessary steps to build such model. That is, what sequences you need if no model is available to you? Are you solving problem $1, 2$ or $\ldots$?

Se o modelo não fosse já dado, seria preciso obter uma sequência onde, para cada ano, tivesse a temperatura e a largura do anel das árvores, pois desta forma, bastaria calcular a frequência de transição entre estados, e a frequência de cada emissão associada ao estado, para gerar as matrizes necessárias do modelo.

Quanto ao Problema, considerando que é sabido:

- Matriz de Transição
- Matriz de Emissão
- Probabilidade Inicial

Logo concluímos que temos, além das observações $\mathcal{O}_T$, também o modelo $\mathcal{M}$. Isso já elimina os Problemas 3 e 4, pois nestes devemos construir um modelo. Agora, dentre os problemas 1 e 2 o que se adequa à esta questão é o 2, pois queremos descobrir, a partir das observações, a sequência de temperatura que melhor descreve tais dados coletados.