Inteligência Artificial

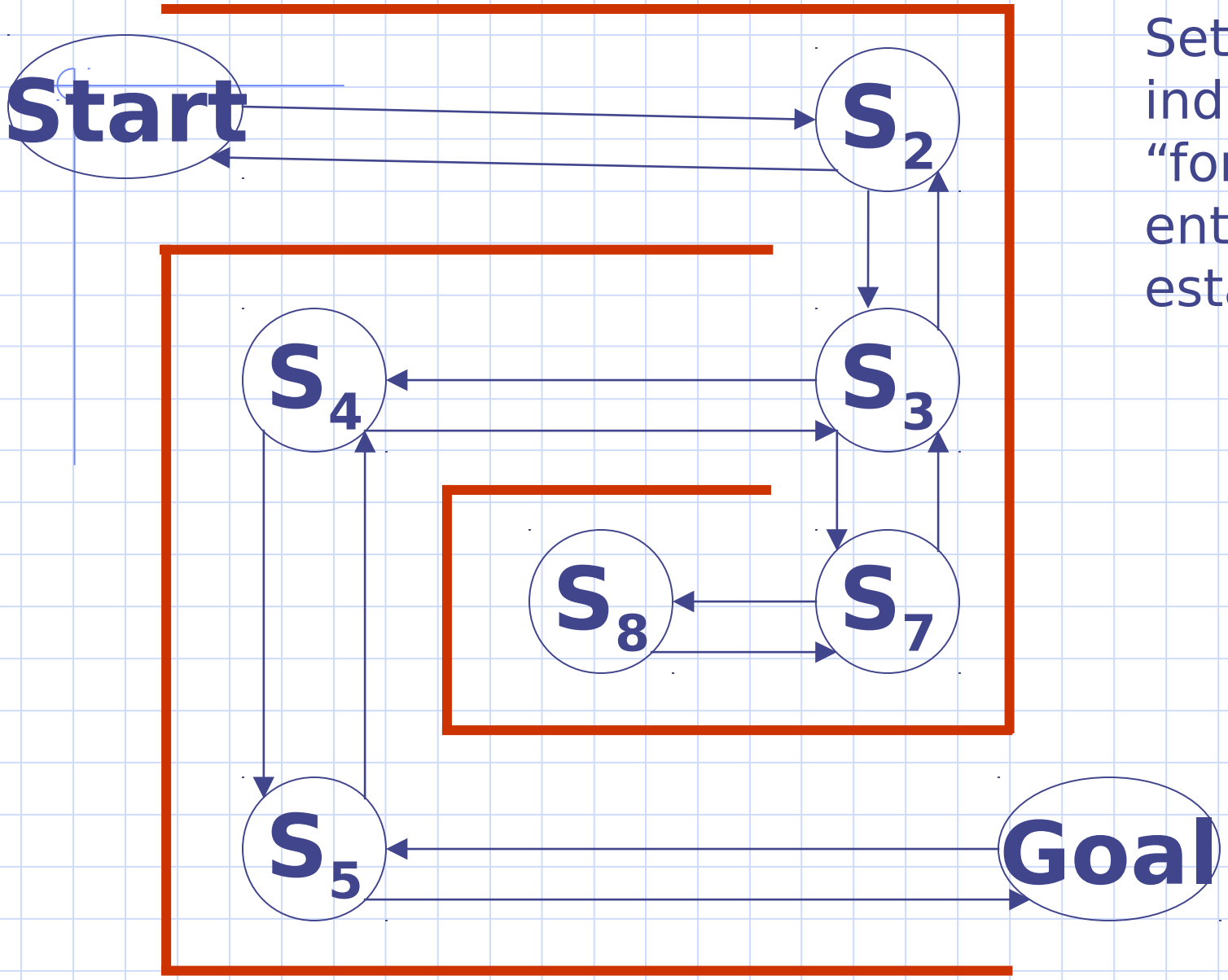
Aprendizado por reforço

Profa Debora Medeiros

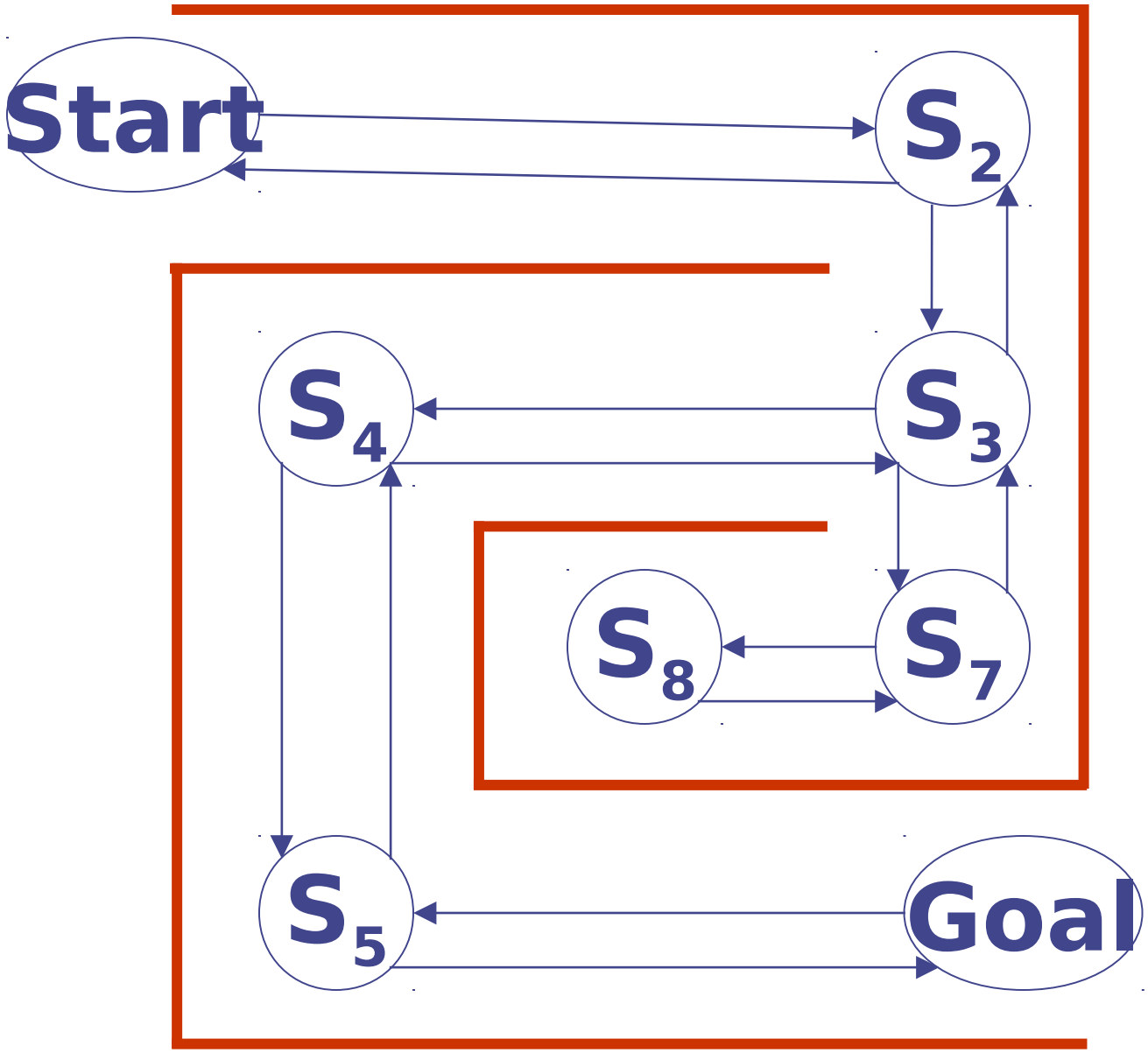
Motivação

- Como um agente aprende a escolher ações apenas interagindo com o ambiente?
  - Muitas vezes, é impraticável o uso de aprendizagem supervisionada ou busca
    - Como obter exemplos do comportamento correto e representativo para qualquer situação?
    - E se o agente for atuar em um ambiente desconhecido?
  - Exemplos:
    - Criança adquirindo coordenação motora
    - Robô interagindo com um ambiente para atingir objetivo(s)

Exemplo de aprendizado por reforço



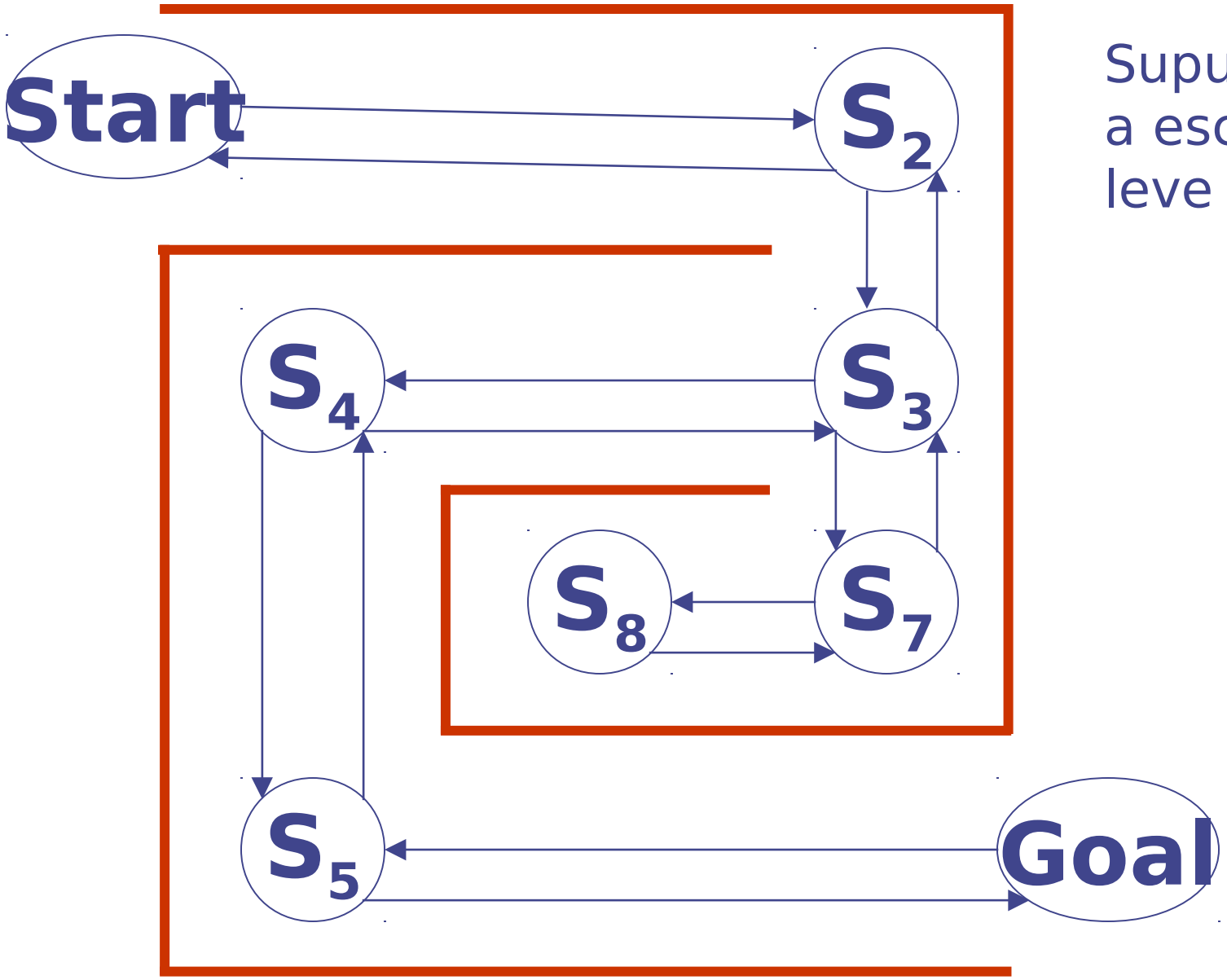
Setas indicam a "força" entre dois estados



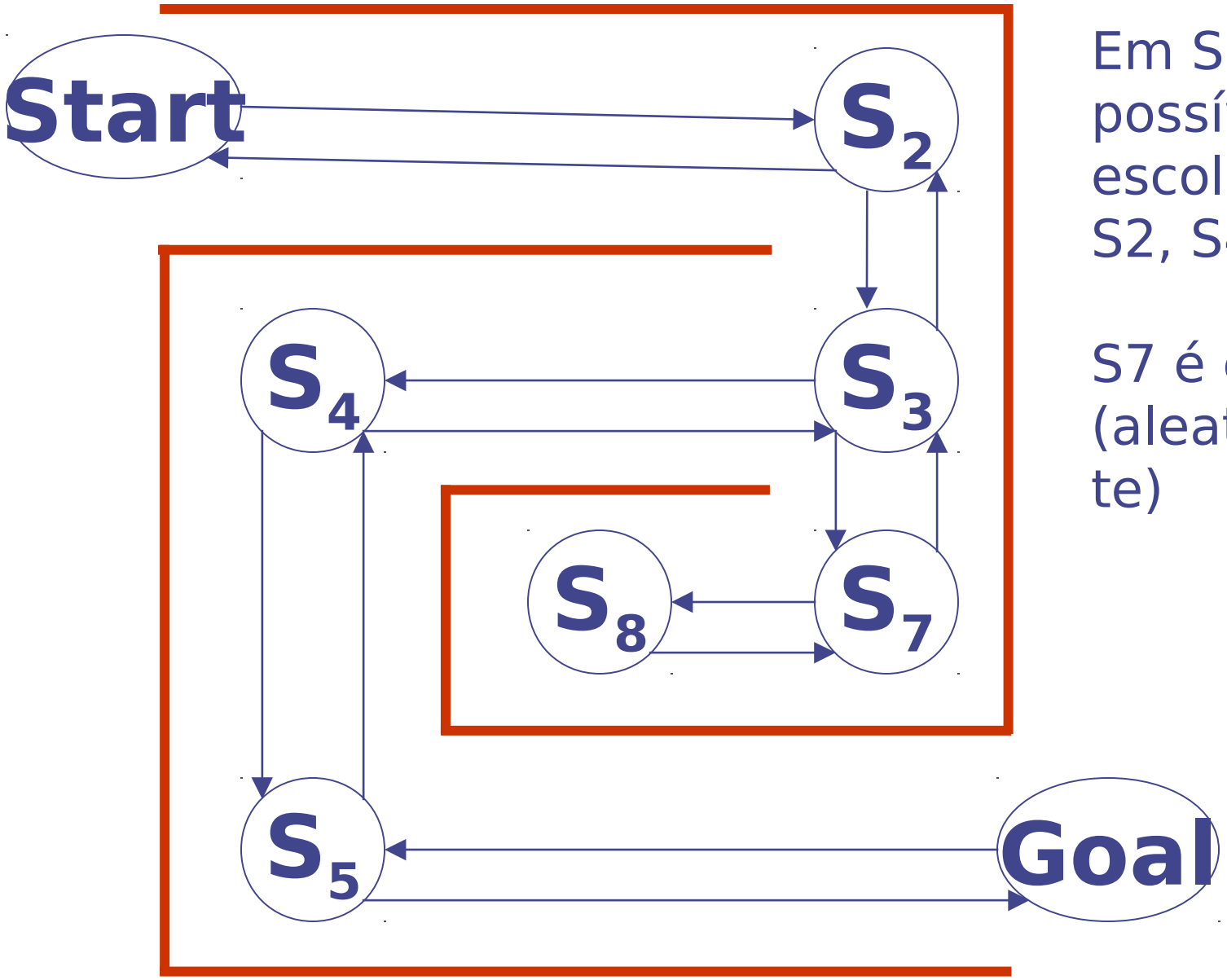
A primeira ação leva para S2 ...

Próximo estado é escolhido aleatoriamente de um dos possíveis estados, ponderado pela força da associação

Associação = largura da linha

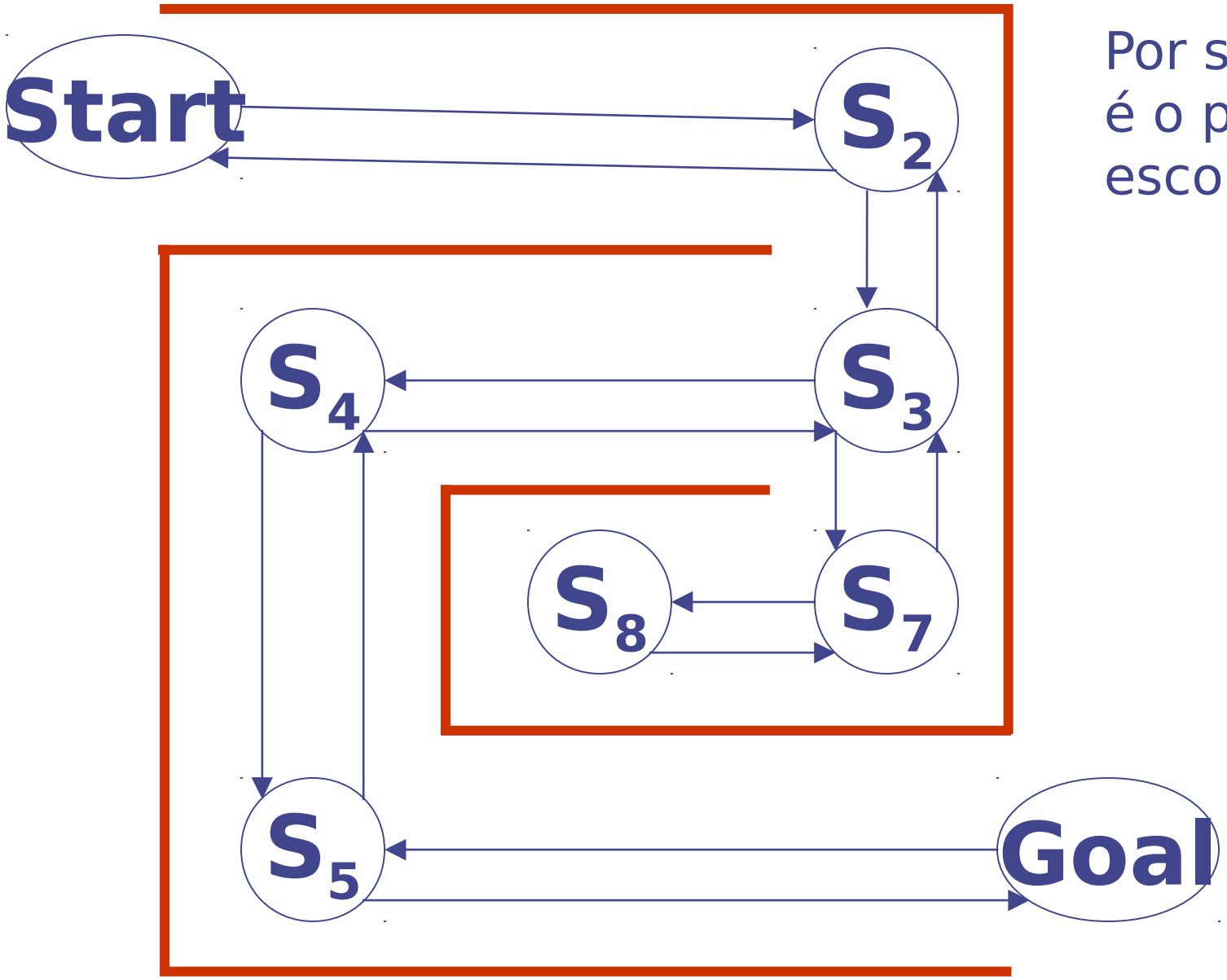


Supunha que
a escolha
leve a S3 ...



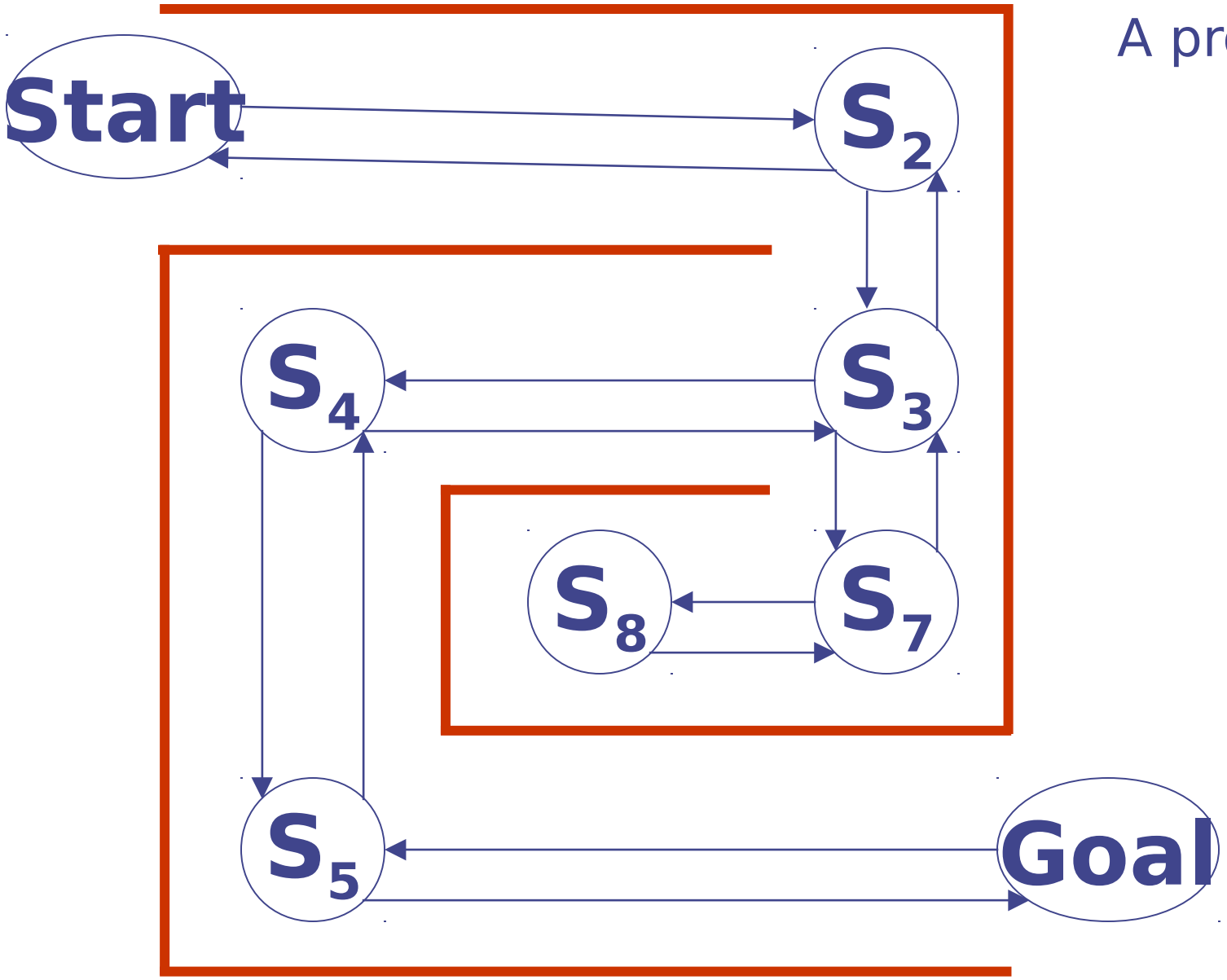
Em S3, as possíveis escolhas são S2, S4, or S7.

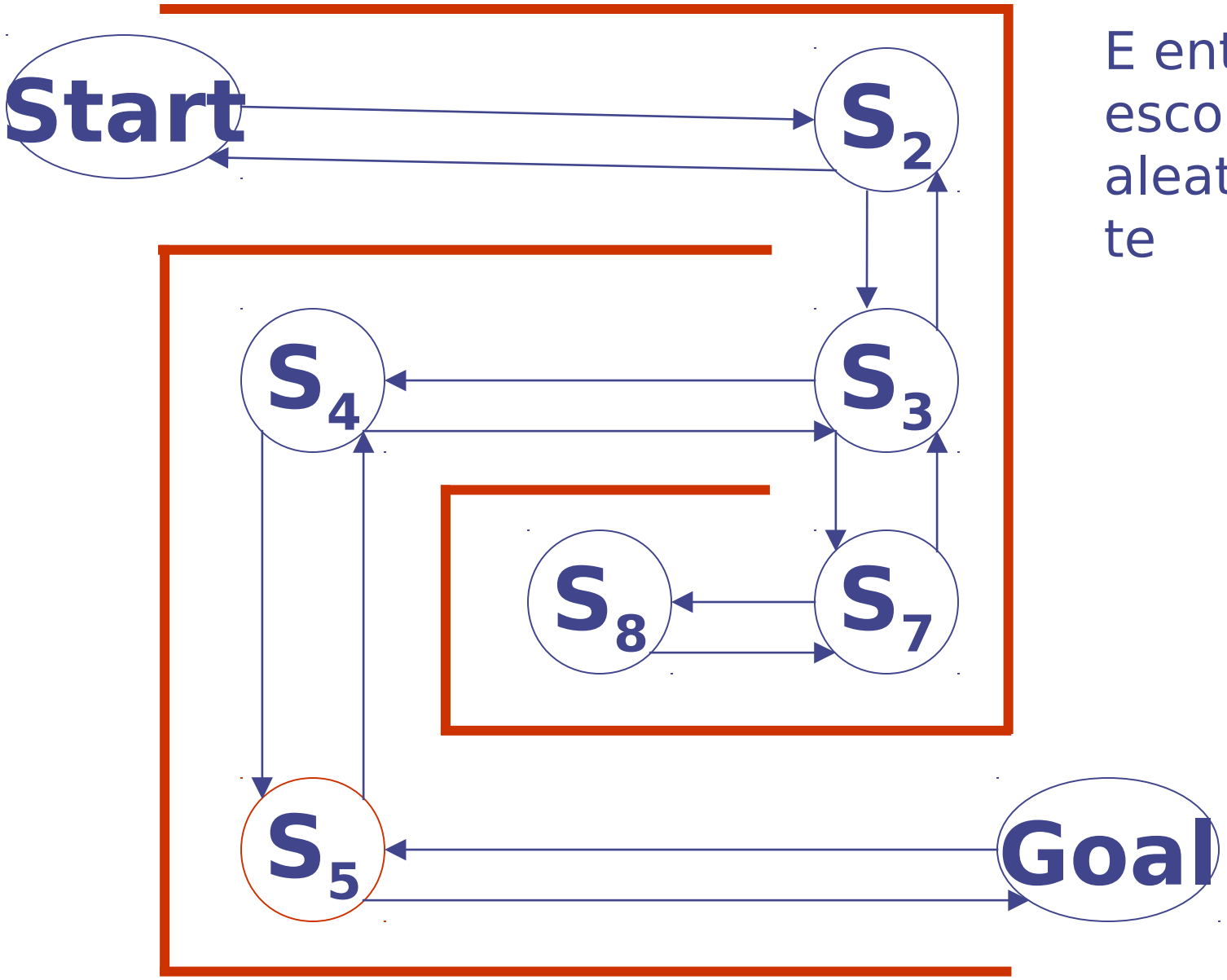
S7 é escolhido (aleatoriamente)



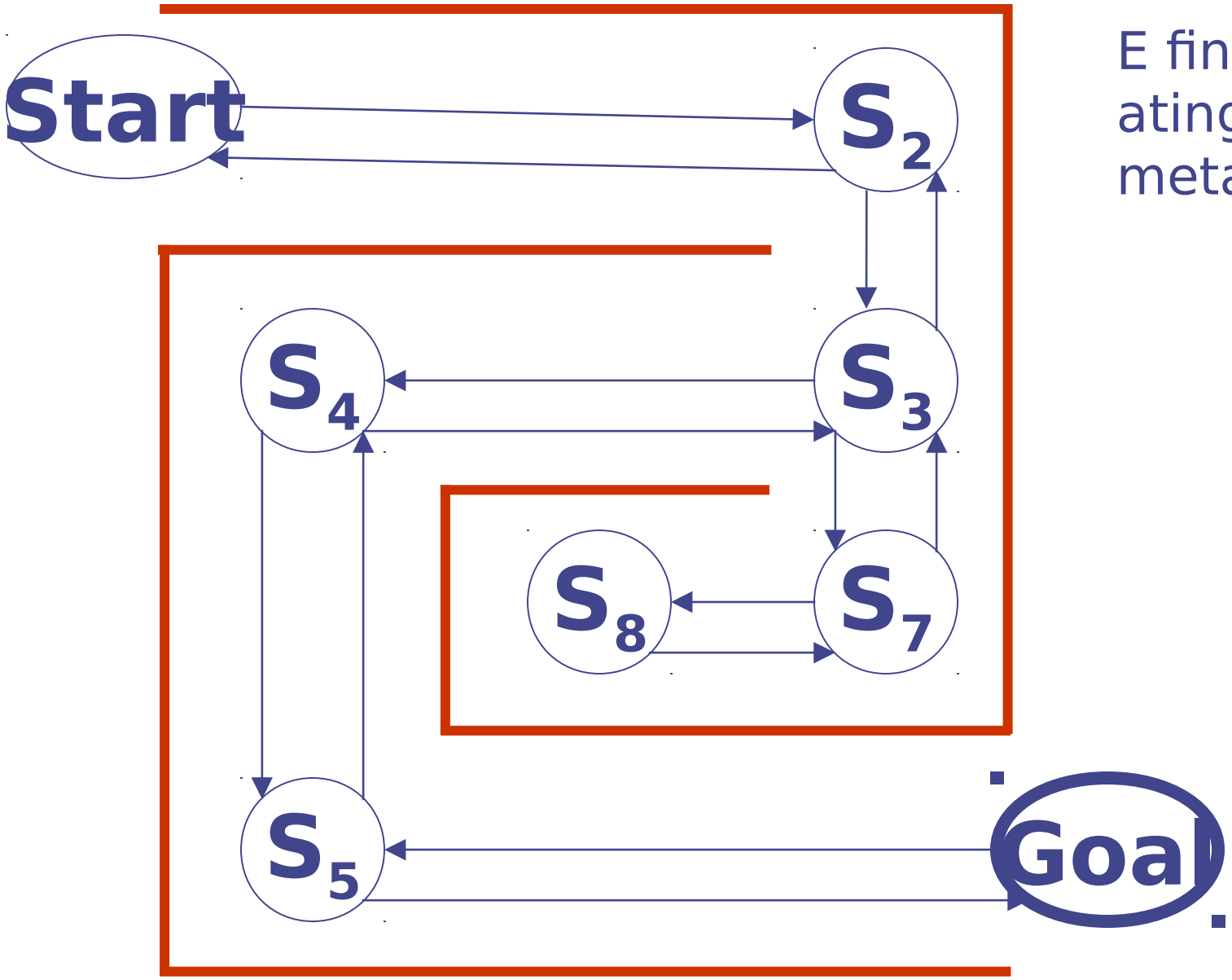
Por sorteio, S3 é o próximo escolhido...

A próxima é S4

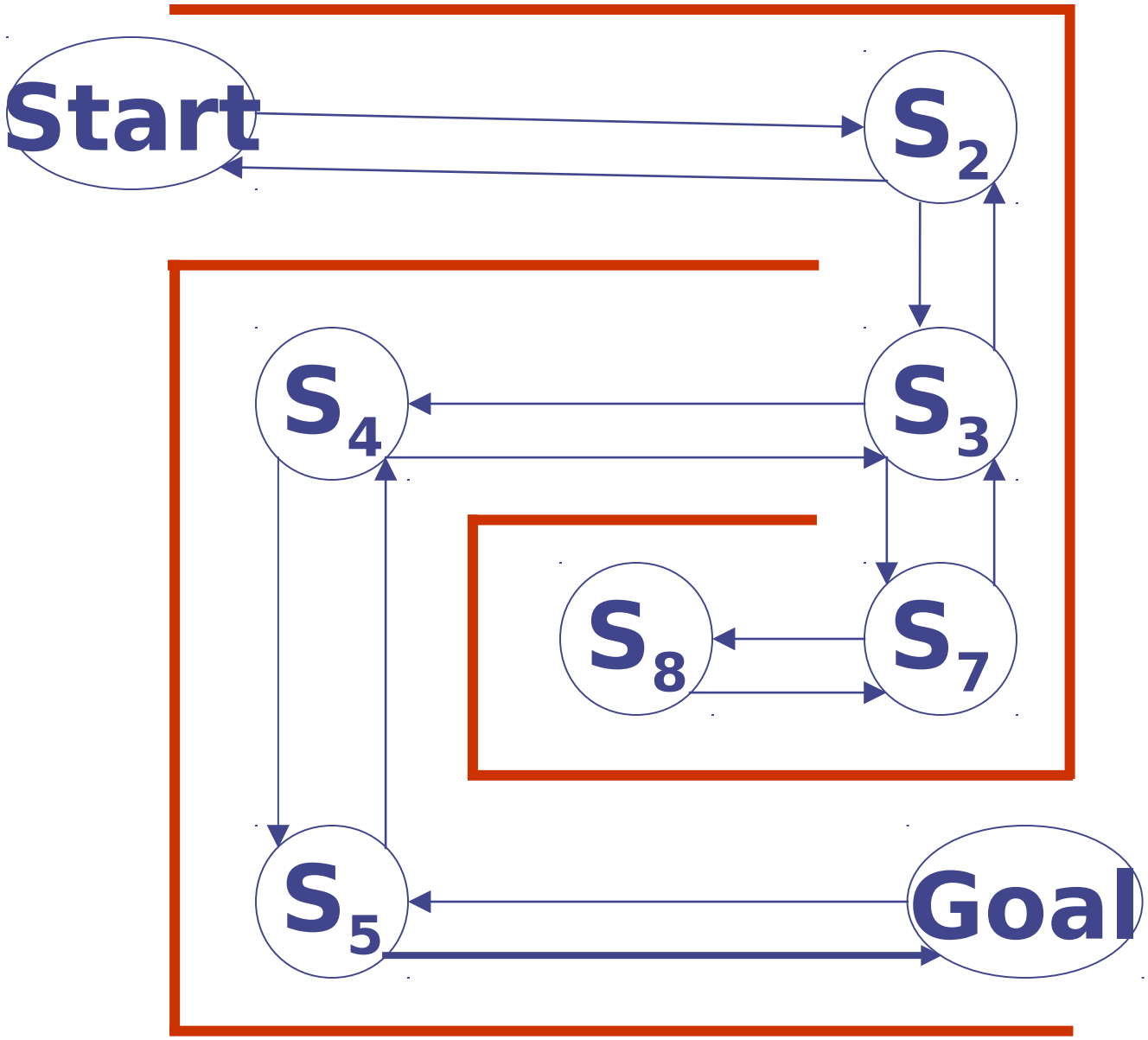




E então S5 é escolhido aleatoriamente

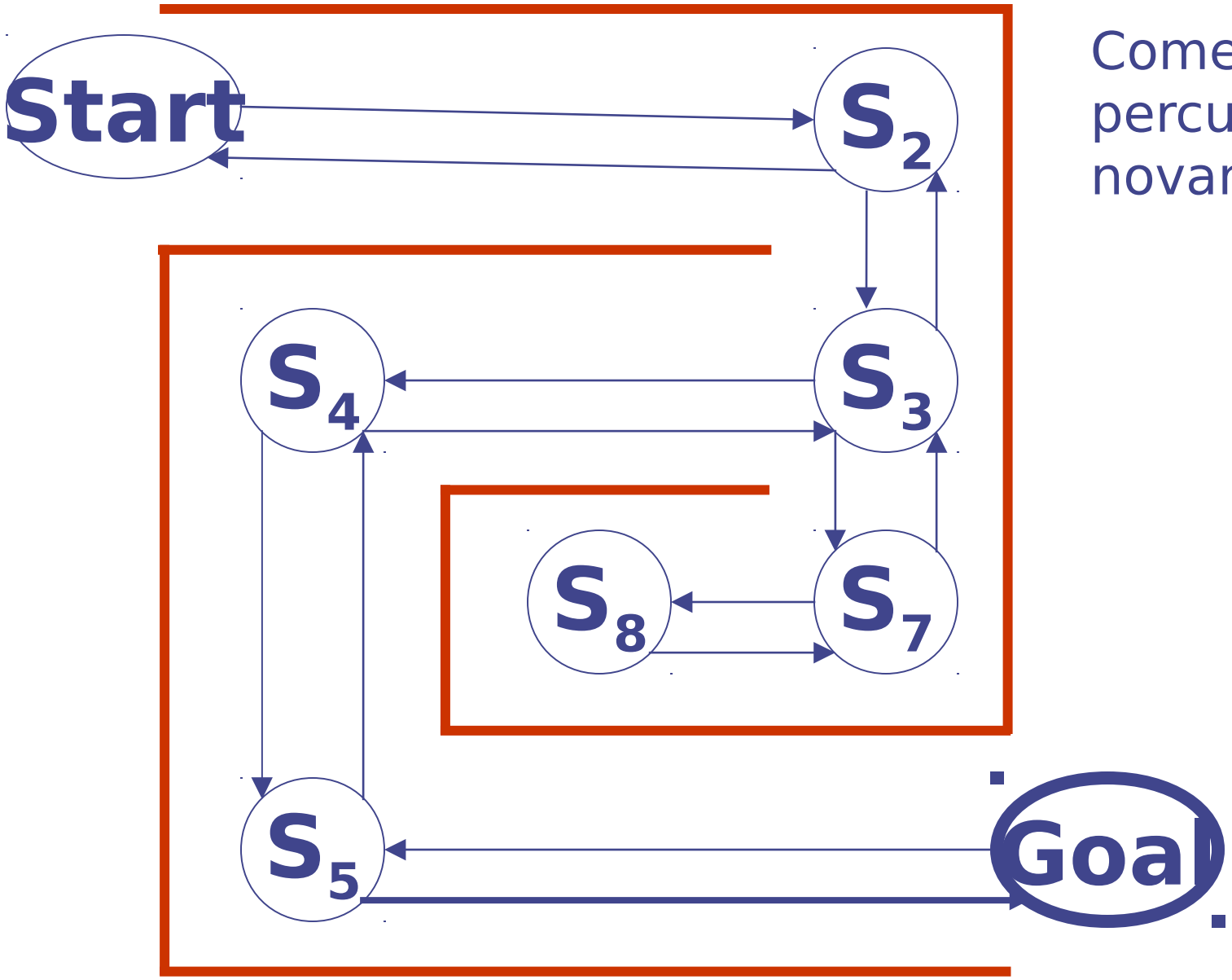


E finalmente atingimos a meta ...

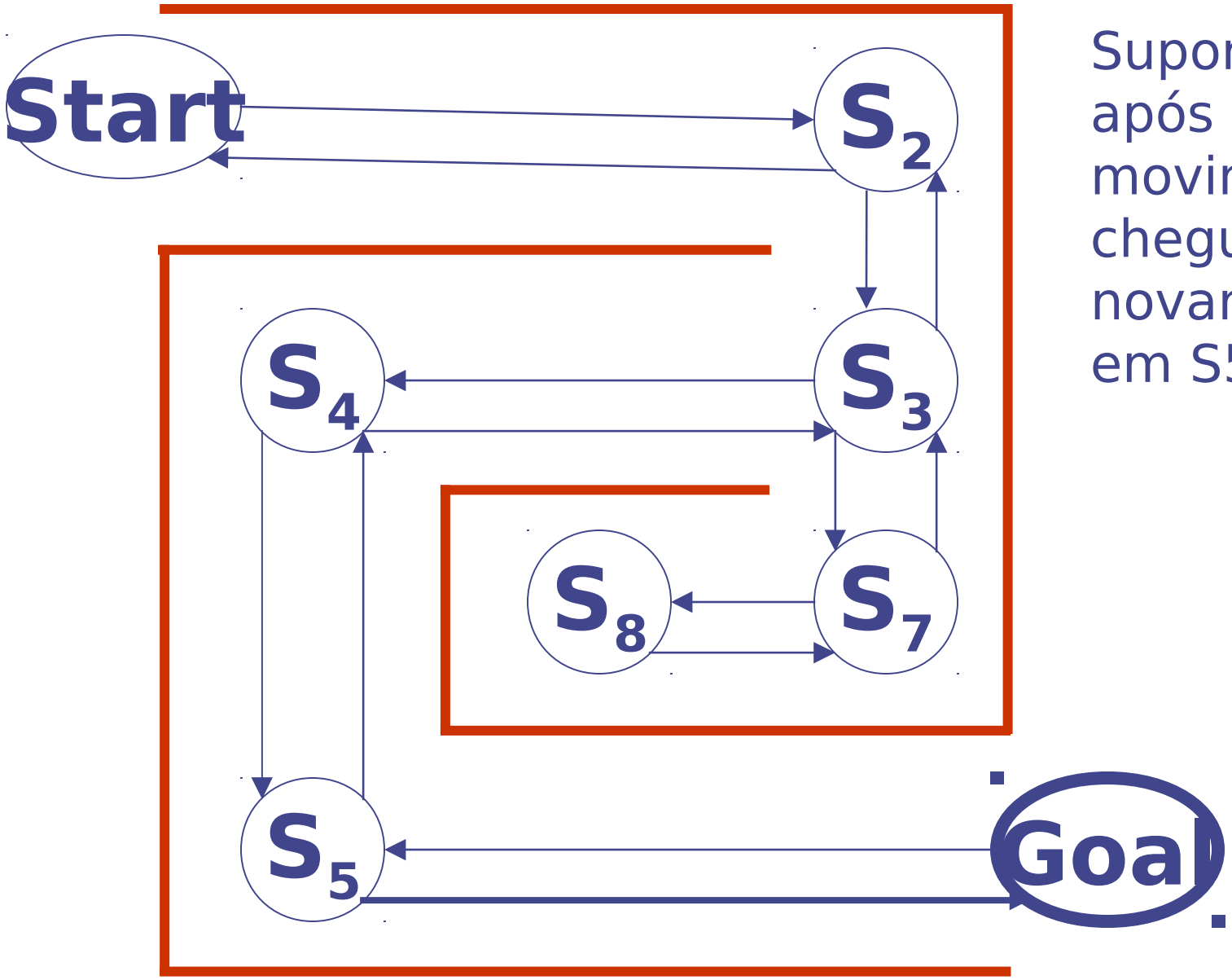


Quando a meta é atingida, reforce a conexão entre ele e o estado que levou a ele

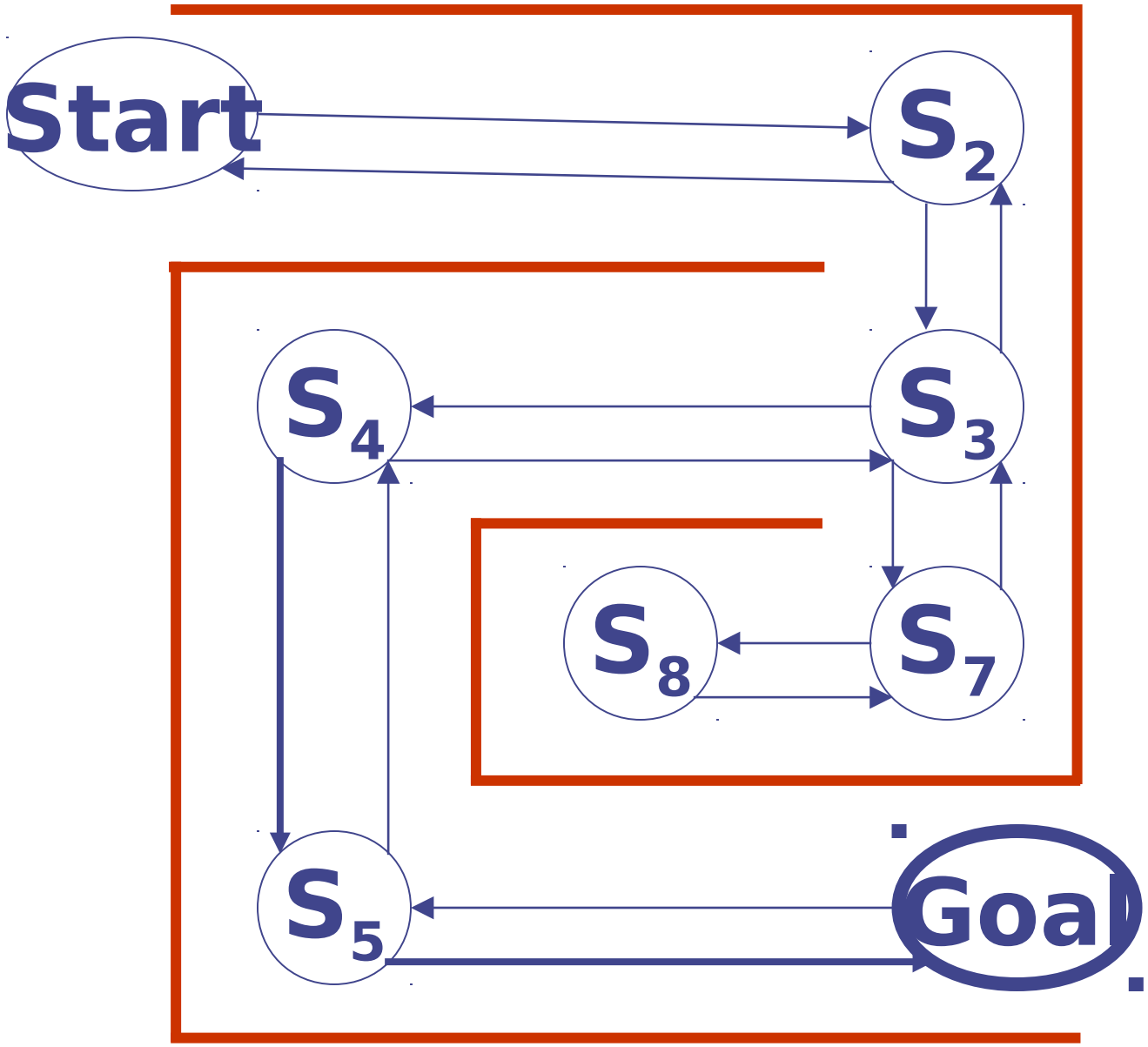
Na próxima vez que S5 for alcançado, parte da força de associação será passada para S4...



Comece o percurso novamente...



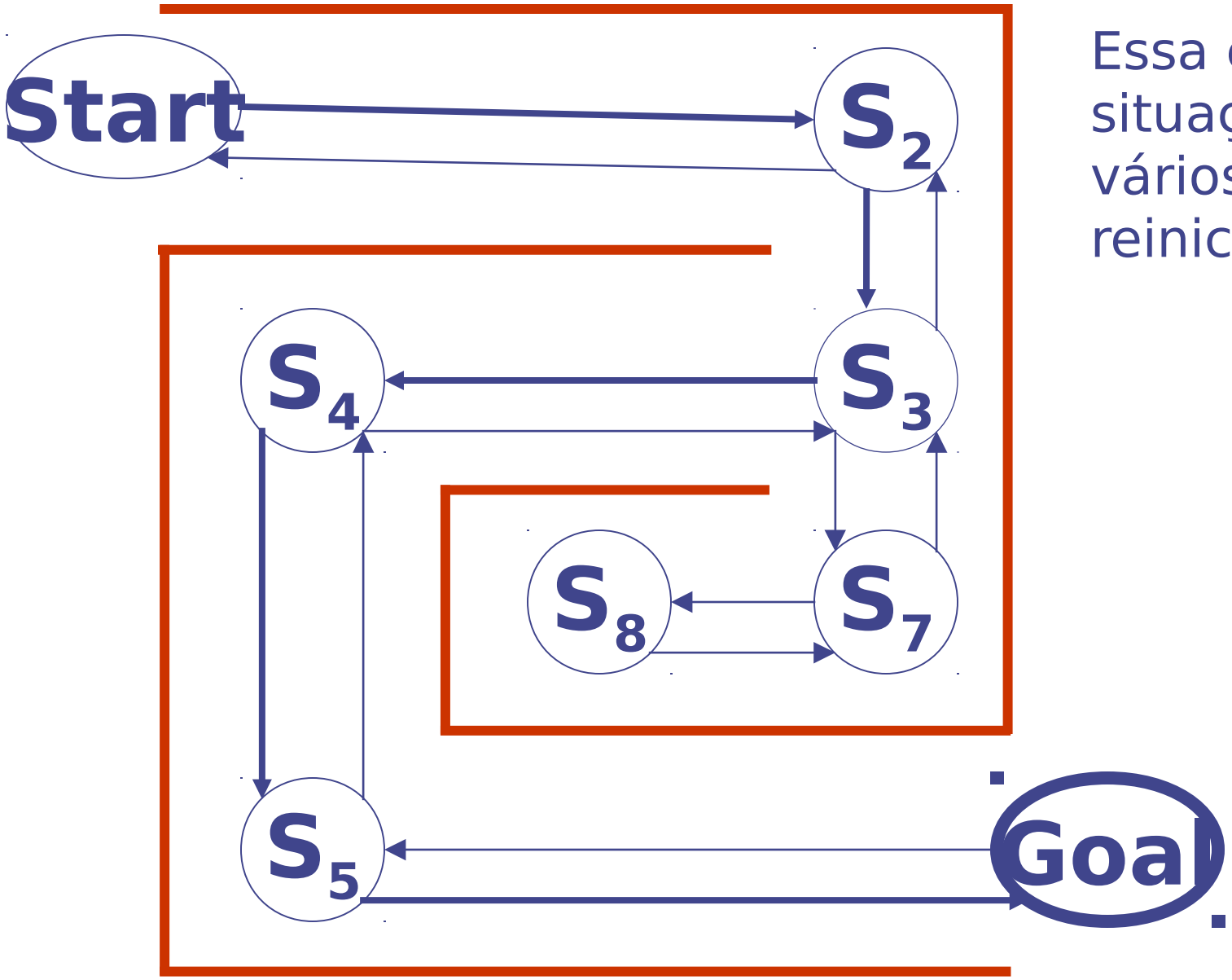
Suponha que após alguns movimentos, cheguemos novamente em S5



S5 tem grande chance de atingir a meta pela rota com mais força

Em aprendizado por reforço, a “força” é passada de volta para o estado anterior

Esse processo leva a criar um caminho entre o início e a meta

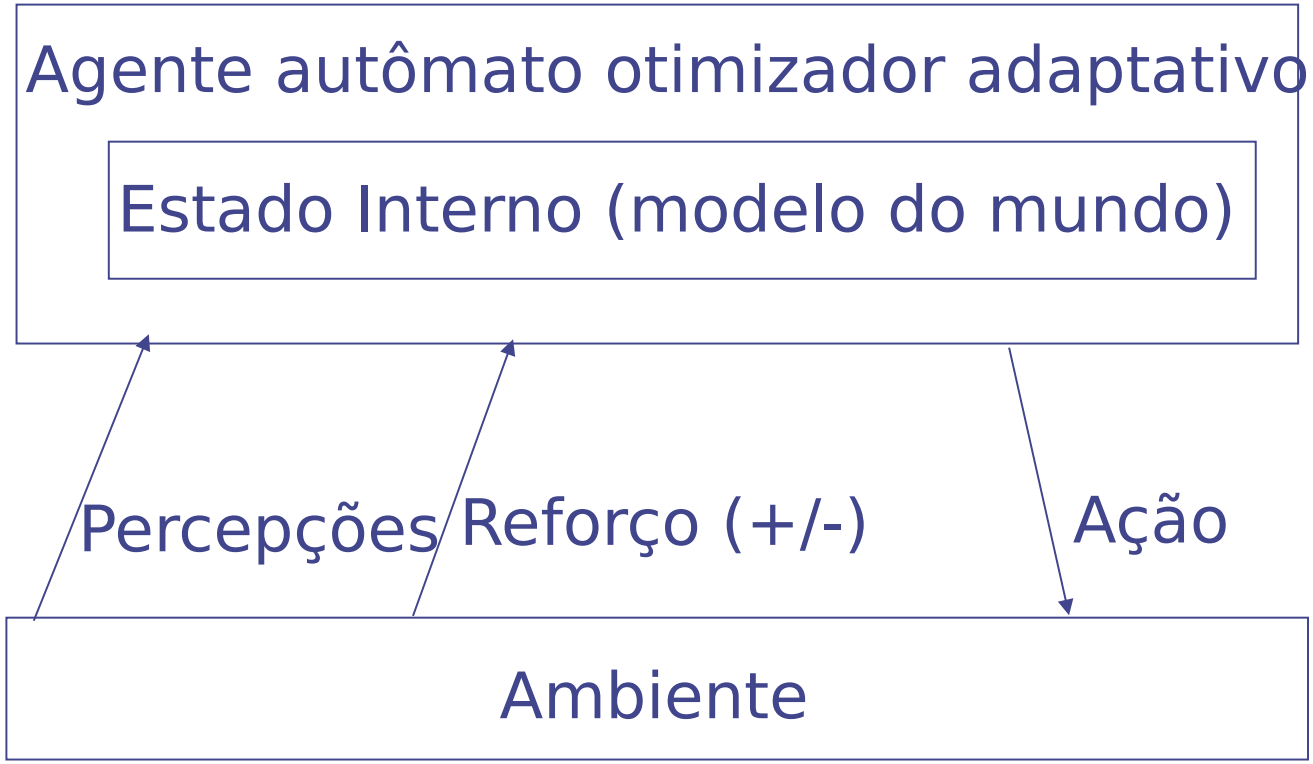


Essa é a situação após vários reinicios...

O que é aprendizagem por reforço (tradicional)?

- Problema de aprendizagem (não é uma técnica)
  - **Um agente, em um ambiente**
  - **A cada instante de tempo t:**
    - **o agente está em um *estado* s**
    - **executa uma *ação* a**
    - **vai para um *estado* s'**
    - **recebe uma *recompensa* r**
  - **Problema da aprendizagem por reforço:**
    - **Como escolher uma *política de ações* que maximize o total de recompensas recebidas pelo agente**

O problema da aprendizagem por reforço



Algumas aplicações

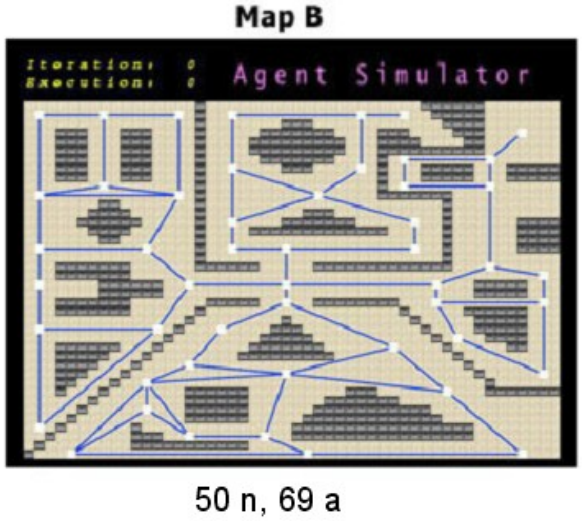
- [Tesauro, 1995] Modelagem do jogo de gamão como um problema de aprendizagem por reforço:
  - Vitória: +100
  - Derrota: – 100
  - Zero para os demais estados do jogo (*delayed reward*)
  - Após 1 milhão de partidas contra ele mesmo, joga tão bem quanto o melhor jogador humano

Algumas aplicações

- Time Brainstormers da Robocup (entre os 3 melhores em 3 anos seguidos)
  - Objetivo: Time cujo conhecimento é obtido 100% por técnicas de aprendizagem por reforço
- Inúmeras aplicações em problemas de otimização, de controle, jogos e outros...

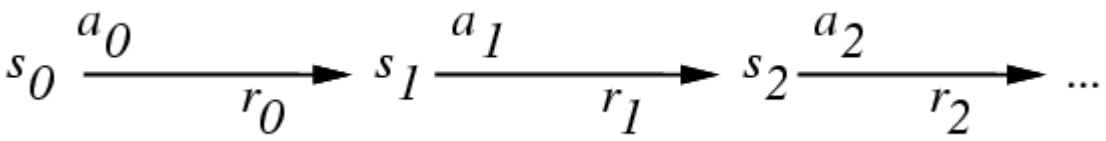
Patrulha multi-agente

- Dado um mapa, um grupo de agentes deve visitar continuamente locais específicos deste mapa de maneira a minimizar o tempo que os nós ficam sem serem visitados
- Recompensa: ociosidade dos nós visitados
- Coordenação emergente (mesmo sem comunicação explícita)



Conceitos Básicos

- Processo de decisão de Markov (MDP)
  - Conjunto de estados **S**
  - Conjunto de ações **A**
  - Uma função de recompensa **r(s,a)**
  - Uma função de transição de estados (pode ser estocástica) **δ(s,a)**
- Política de ações **π(s)** :
  - π: **S** → **A**



Estados e Ações

- Estado: conjunto de características indicando como está o ambiente
  - Formado pelas percepções do agente + modelo do mundo
  - Deve prover informação para o agente de quais ações podem ser executadas
- A representação deste estado deve ser suficiente para que o agente tome suas decisões (satisfaz a propriedade de Markov)
  - A decisão de que ação tomar não pode depender da sequência de estados anteriores
  - Ex: Um tabuleiro de dama satisfaz esta propriedade, mas de xadrez não

A função de recompensa

- Feedback do ambiente sobre o comportamento do agente
- Indicada por $r:(S \times A) \rightarrow R$
  - r(s,a) indica a recompensa recebida quando se está no estado **s** e se executa a ação **a**
  - Pode ser determinística ou estocástica

Função de transição de estados

- δ: (**S** $\times$ **A)** $\rightarrow$ **S**
- δ(s,a) indica em qual estado o agente está, dado que:
  - Estava no estado **s**
  - executou a ação **a**
- Ambientes não-determinísticos:
  - escrita como δ(s,a,s')
  - indica a probabilidade de ir para um estado **s'** dado que estava em **s** e executou **a**

Exemplos de MDPs

Problema	Estados	Ações	Recompensas
Agente jogador de damas	Configurações do tabuleiro	Mover uma determinada peça	#capturas – #perdas
Agente em jogo de luta	Posições/energia dos lutadores, tempo, se está sendo atacado ou não, etc...	Mover-se em uma determinada direção, lançar magia, dar golpe, etc...	(Sangue tirado – sangue perdido)
Agente patrulhador	Posição no mapa (atual e passadas), ociosidade da vizinhança, etc...	Ir para algum lugar vizinho do mapa	Ociosidade (tempo sem visitas) do lugar visitado atualmente

Política de ações (π)

- Função que modela o comportamento do agente
  - Mapeia estados em ações
- Pode ser vista como um conjunto de regras do tipo $s_n \rightarrow a_m$
  - Exemplo:
    - Se estado **s = (inimigo próximo, estou perdendo e tempo acabando)** então
      ação **a = (usar magia)**;
      Se estado **s =(outro estado)** então
      ...

Função valor dos **estados** $V\pi(s)$ $(S \rightarrow R)$

- Como saber se um determinado estado é bom ou ruim?
  - A função valor expressa esta noção, em termos das recompensas e da política de ações
  - Representa a recompensa a receber em um estado **s**, mais as recompensas futuras se seguir uma política de ações π
    - ex. tornar-se diretor, vale pelo que o cargo permite e permitirá nas próximas promoções (não interessa de onde veio - chefe de seção)
  - $V\pi(s_0) = r_0 + r_1 + r_2 + r_3 + ...$
    - Problema: se o tempo for infinito, a função valor do estado tende a infinito

Função Valor dos **estados**

- Para garantir convergência e diferenciar recompensas distantes do estado atual, usa-se um fator de desconto
  $0 \leq \gamma \leq 1$
- $V\pi(s_t) = r_t + \gamma\, r_{t+1} + \gamma^2\, r_{t+2} + \gamma^3\, r_{t+3} \ldots$
- $V\pi(s_t) = r_t + \gamma V\pi(s')$, onde:
  - $r_t = r(s_t, \pi(s_t))$
  - $s' = \delta(s_t, \pi(s_t))$
- Ex. Se $\gamma = 90\%$, então:
  - $V\pi(s_t) = r_t + 0.9\, r_{t+1} + 0.81\, r_{t+2} + 0.729\, r_{t+3} \ldots$

Função valor das **ações**
$Q\pi(s,a) : (S \times A) \rightarrow R$

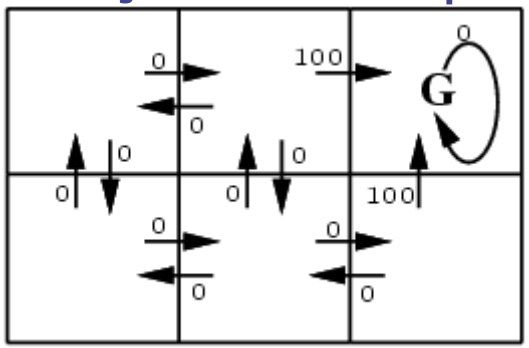
- Analogamente, ela diz a soma das recompensas a obter dado que:
  - o agente está no estado **s**
  - executou uma ação **a**
  - a partir daí, seguiu uma política de ações π
- $Q\pi(s,a) = r(s,a) + \gamma V\pi(s')$, onde:
  - $s' = \delta(s,a)$
    - o valor da ação é a recompensa da ação mais o valor do estado para onde o agente vai devido à ação

Aprendizagem por reforço

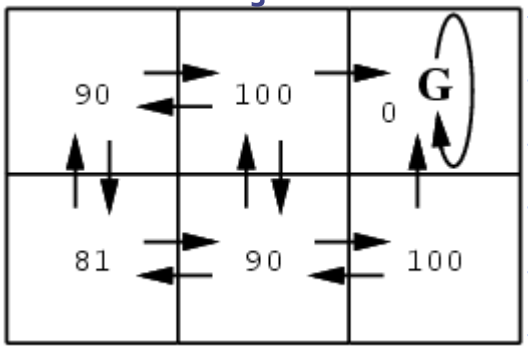
- Tarefa de aprendizagem por reforço:
  - Aprender uma política de ações **π*** ótima**,** que maximiza a função **Vπ (V*)** ou a função **Qπ (Q*)**
    - $\pi^* = \text{argmax}_\pi[V\pi(s)]$
- Em outras palavras, de que maneira o agente deve agir para maximizar as suas recompensas futuras

Exemplo: Labirinto (c/γ=0.9)

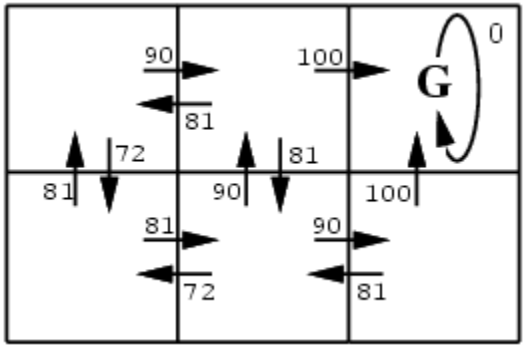
Função recompensa



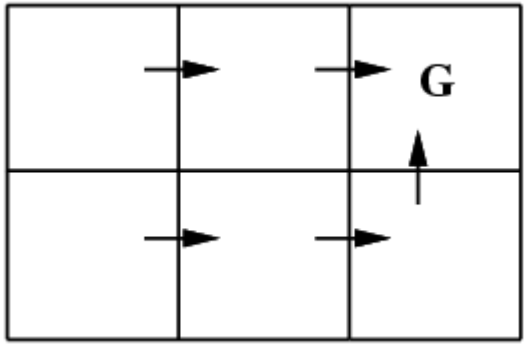
Função V*



Função Q*



Uma política de ações ótima



Aprendendo uma política ótima

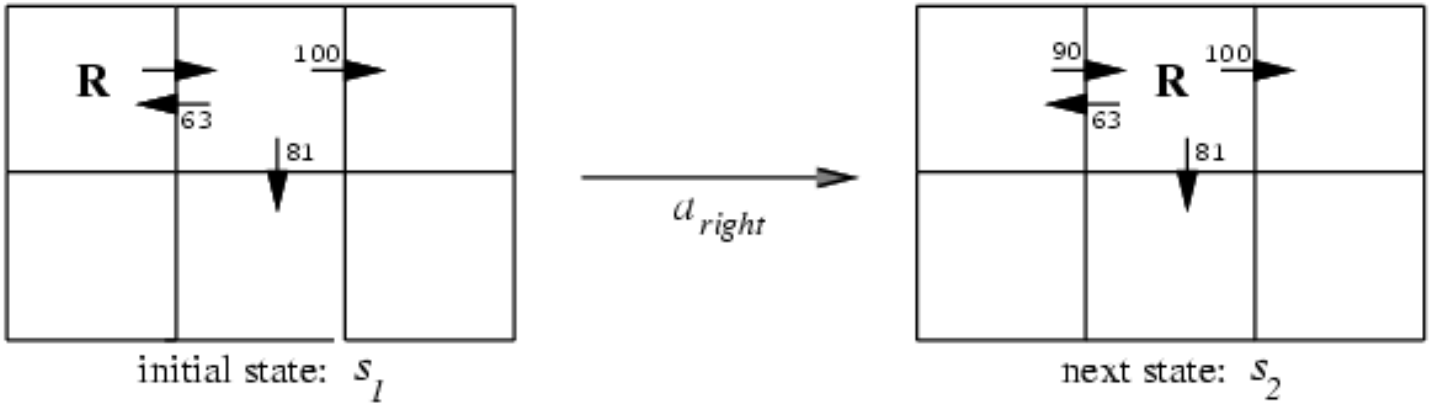
- Se o ambiente é determinístico ($\delta(s,a) = s'$ é conhecida) e r(s,a) é conhecida, a programação dinâmica computa uma política ótima :
  - $V^*(s) = \max_a [r(s,a) + \gamma V^*(\delta(s,a))]$
  - $\pi^*(s) = \text{argmax}_a [r(s,a) + \gamma V^*(\delta(s,a))]$
  - Tempo polinomial
  - Problema: se não temos conhecimento prévio das recompensas e transição de estados
- Se o ambiente é não-determinístico mas a função de probabilidade de transição de estados for conhecida, também é possível computar π^*
  - problema: É difícil estimar estas probabilidades

Q Learning

- É possível determinar π^* se eu conheço Q^*
  - não precisando conhecer δ (função de transição de estados) nem **r**
  - $\pi^*(s) = \text{argmax}_a[Q(s,a)]$
    - não é função de δ nem de r
- Então, vamos aprender a função Q ótima (valor das ações) sem considerar V
  - $Q(s_t,a_t) = r(s_t,a_t) + \gamma\, V^*(\, \delta(s_t,a_t)\,)$
    $= r(s_t,a_t) + \gamma \max_{a'} [Q(s_{t+1},a')]$
    - o valor do próximo estado é o melhor Q nele
    - Como atualizar Q ?

Q-Learning

- Atualiza-se $Q(s_t)$ após observar o estado s_{t+1} e recompensa recebida



- $Q(s_1,a_{right}) = r + \gamma \max_{a'} Q(s_2,a')$
  $= 0 + 0.9 \max\{63,81,100\}$
  $= 90$

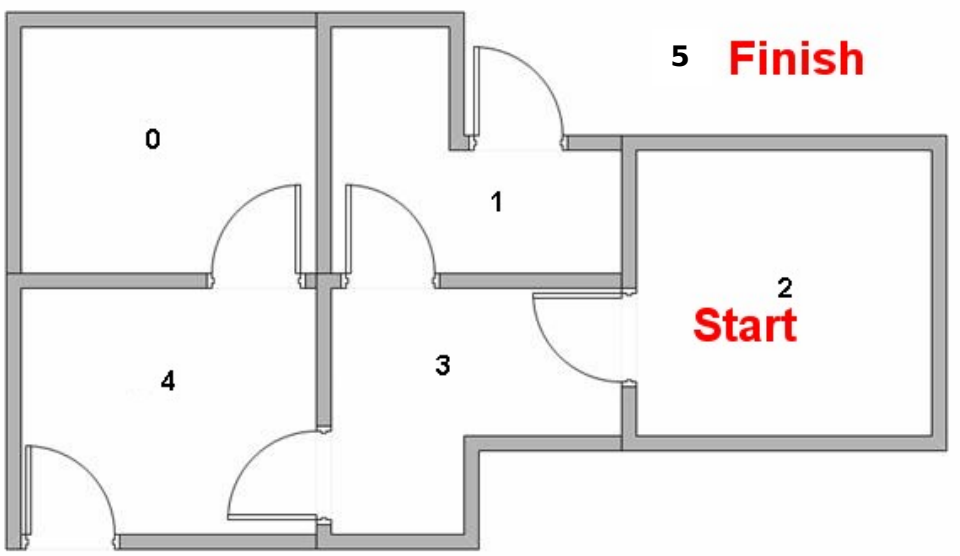
Algoritmo Q-Learning para mundos determinísticos

- Para todo estado **s** e ação **a**, inicialize a tabela Q[s][a] = 0;
- Para sempre, faça:
  - Observe o estado atual **s;**
  - Escolha uma ação **a** e execute;
  - Observe o próximo estado **s'** e recompensa **r**
  - Atualize a tabela Q:
    - $Q[s][a] = r + \gamma \max_{a'} (Q[s'][a'])$

Usufruir de valores conhecidos ou explorar valores não computados?

Exemplo

- 6 estados, máximo 6 ações possíveis por estado (ação descrita pelo novo estado)
- R inicial:



http://mnemstudio.org/path-finding-q-learning-tutorial.htm

Exemplo

- 6 estados, máximo 6 ações possíveis por estado (ação descrita pelo novo estado)
- Q inicial:

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Exemplo

- Selecionando aleatoriamente estado 1 como inicial e a ação que leva ao estado 5

Q(1, 5) = R(1, 5) + 0.8 * Max[Q(5, 1), Q(5, 4)]

= 100 + 0.8 * 0

= 100

- **O próximo estado seria o 5, porém ele é final, então, o processo deve continuar a partir de um novo estado inicial**

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Exemplo

- Selecionando aleatoriamente 3 como estado inicial e a ação que leva ao estado 2

$$Q(3, 1) = R(3, 1) + 0.8 * \text{Max}[Q(1, 3), Q(1, 5)]$$
$$= 0 + 0.8 * \text{Max}(0, 100)$$
$$= 80$$

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \quad \Rightarrow \quad Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Dilema de explorar ou usufruir (exploration x exploitation)

- Usufruir
  - Escolher a ação que atualmente está com maior valor Q(s,a)
- Explorar
  - Escolher uma ação randômica, para que seu valor Q(s,a) seja atualizado
- Dilema
  - Dado que eu aprendi que Q(s,a) vale 100, vale a pena tentar executar a ação a' se Q(s,a') por enquanto vale 20 ?
    - Depende do ambiente, da quantidade de ações já tomadas e da quantidade de ações restantes

Métodos para balancear exploration e exploitation

- E-Greedy
  - A cada iteração, escolhe uma ação exploratória(randômica) com probabilidade E

Semi-MDP

- Como o agente pode levar em conta o tempo de suas ações?
  - Ex. no jogo de luta: É melhor dar vários socos fracos ou um soco forte?
    - Soco forte provavelmente traria maior recompensa
    - Demoraria mais tempo para ser executado
  - No problema da patrulha: como levar em conta o a distância entre os nós?

Semi-MDP

- O formalismo SMDP engloba este conceito
- Prova-se que a atualização de Q passa a ser dada por:
  - $Q[s][a] = r + \gamma^t \max_{a'} (Q[s´][a'])$
  - Onde **t** pode ser:
    - número de unidades de tempo que o agente executou a ação (caso discreto)
    - alguma função contínua do tempo
  - Desta maneira, as recompensas futuras passam a valer menos se o agente passar muito tempo executando uma ação

Aprendizagem por reforço multi-agente - Cooperação

- Abordagens usando RL tradicional:
  - White box agent
    - Representação de estado global
    - Encontra a ação conjunta (a_1, a_2, ..., a_n) que maximiza uma função de reforço global (única)
    - Problemas
      - Complexidade exponencial no número de agentes
      - Como aprender as ações de maneira distribuída ?
  - Black box agent
    - O reforço é individual, mas é alguma função do bem estar global
    - O agente não toma conhecimento dos outros agentes
      - Outros agentes passam a ser ruído no ambiente

Referências

- Slides de Hugo Pimentel de Santana (CIN/UFPE)
- Lecture slides do livro *Machine Learning*, do Tom Mitchell
  - http://www-2.cs.cmu.edu/~tom/mlbook-chapter-slides.html
- Livro “Reinforcement Learning: An introduction”, de Sutton & Barto disponível *online*
  - http://envy.cs.umass.edu/~rich/book/the-book.html