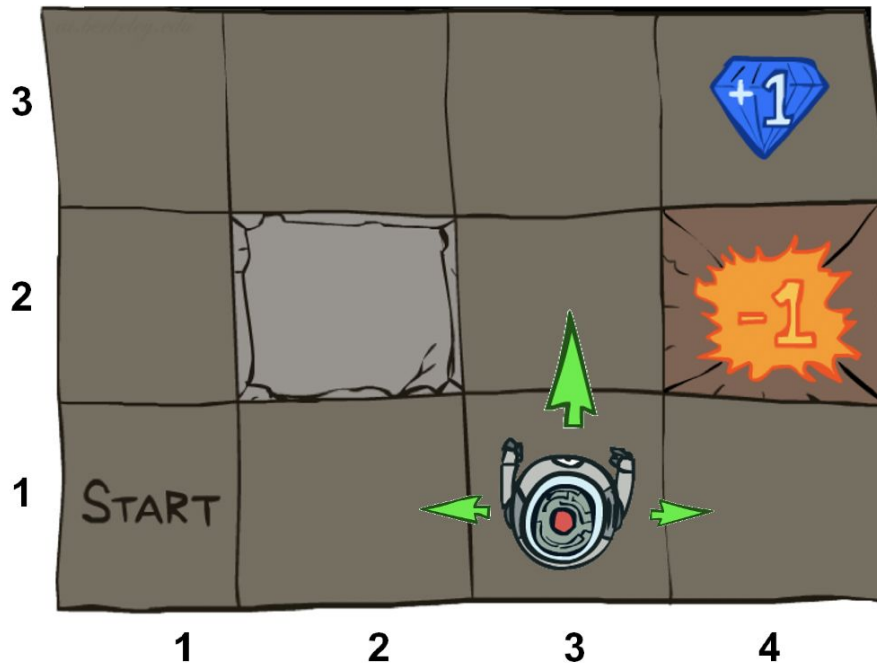























# Processos de Decisão de Markov

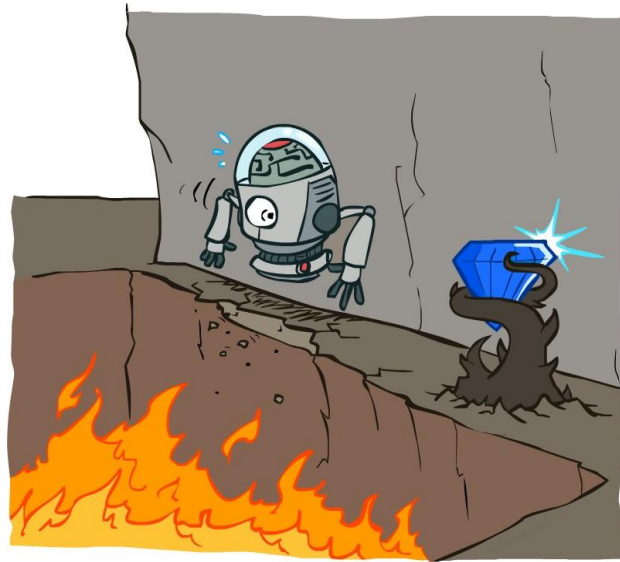


Baseado no curso CS188 de BERKLEY

# Empregos

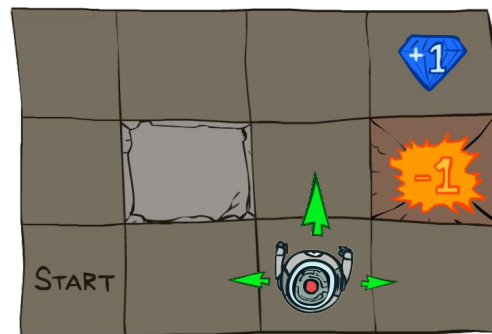
[r-list] Deep Learning / Reinforcement Learning Positions at Uber Advanced Technology.                           

# Busca Estocástica



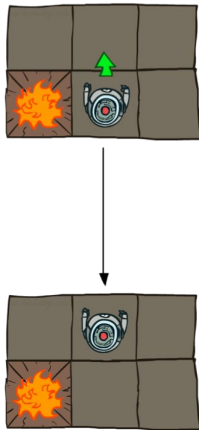
# Grid Estocástico

- Mapa com obstáculos - paredes bloqueiam a passagem do agente.
- Ações são estocásticas.
  - 80% das vezes o agente vai na direção intencionada.
  - 10% das vezes vai para a esquerda. 10% das vezes vai para a direita.
  - se existir parede na direção, o agente fica parado.
- “Recompensa de vida” em cada passo.
- Grandes recompensas ao final. **Objetivo: maximizar as recompensas.**

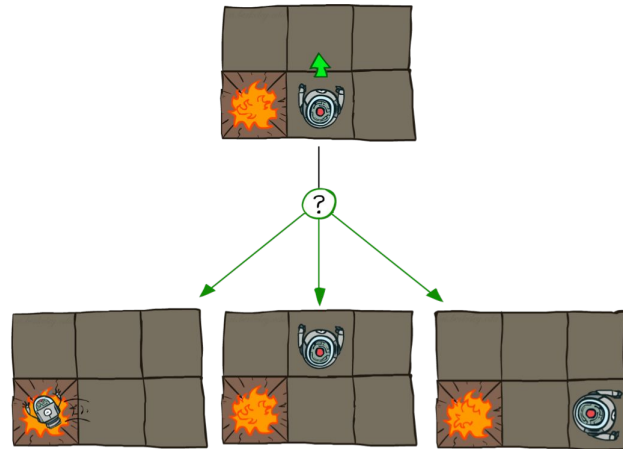


# Ações no Grid Estocástico

Grid Determinístico

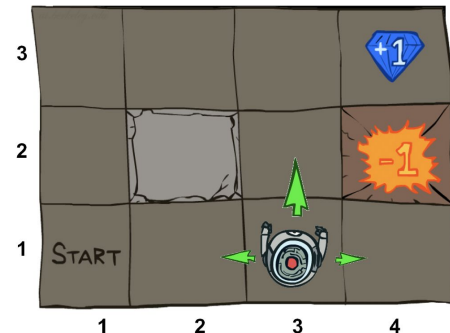


Grid Estocástico

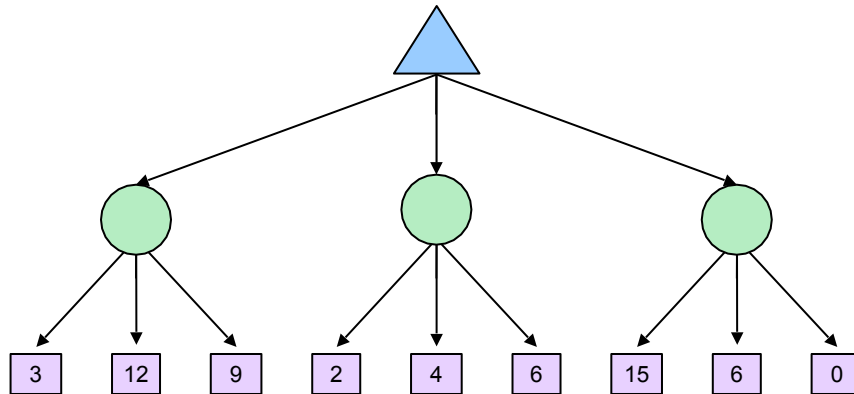


# Processos de Decisão de Markov

- Processos de Decisão de Markov (MDP).
  - Conjunto de estados  $s \in S$  Conjunto de ações  $a \in A$
  - Função de transição  $T(s, a, s')$  com a probabilidade de  $a$  nos levar a  $s'$  a partir de  $s$  i.e.,  $P(s'|s, a)$
  - Função de recompensa  $R(s, a, s')$  as vezes escrita como  $R(s')$
  - Um estado inicial
  - Às vezes um estado final.
- MDPs são problemas de busca não-determinísticos (expectimax soluciona).



# Expectimax



# Propriedade de Markov

- Dado o presente estado, o passado e o futuro são independentes.
- Em MDPs significa que o resultado de uma ação depende apenas do estado atual.

$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, A_{t-1}, \dots, S_0 = s_0)$$

=

$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t)$$

- Assim como em busca, em que a função sucessor depende apenas do estado atual.



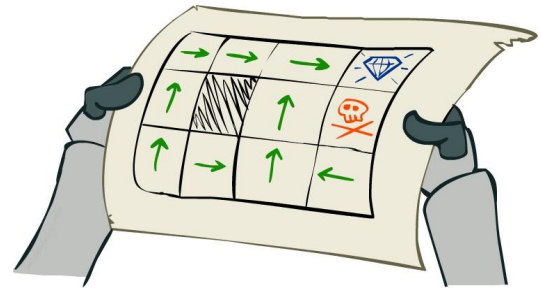
Andrey Markov  
(1856-1922)



# Políticas

- Em busca encontramos uma sequência de ações ótima, que transforma o estado inicial no objetivo.
- Para MDPs, definimos uma política ótima como  $\pi^*: S \rightarrow A$

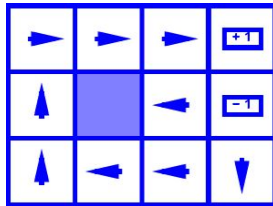
política  $\pi$  fornece uma ação para cada estado política ótima  $\pi^*$  maximiza a utilidade esperada



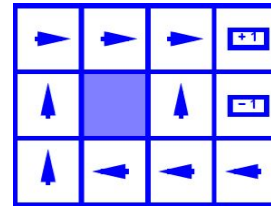
- Expectimax não computa uma política, mas apenas uma ação para um estado.

política ótima se recompensa de vida é igual a -0.3 para estados não terminais.

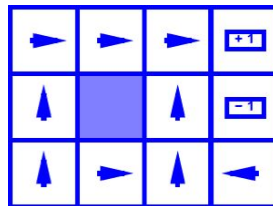
# Políticas Ótimas



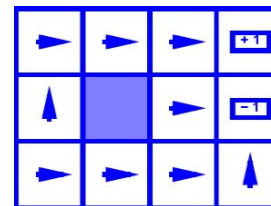
$R(s) = -0.01$



$R(s) = -0.03$



$R(s) = -0.4$



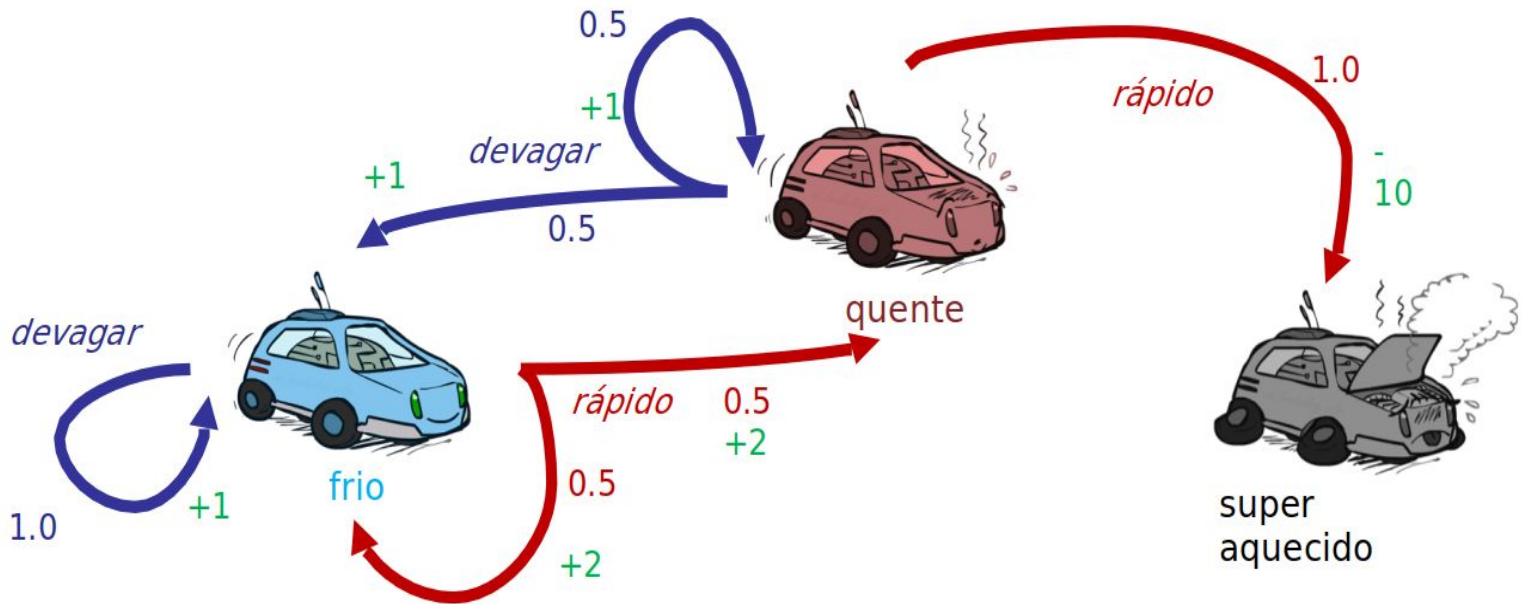
$R(s) = -2.0$

# Exemplo: Corrida

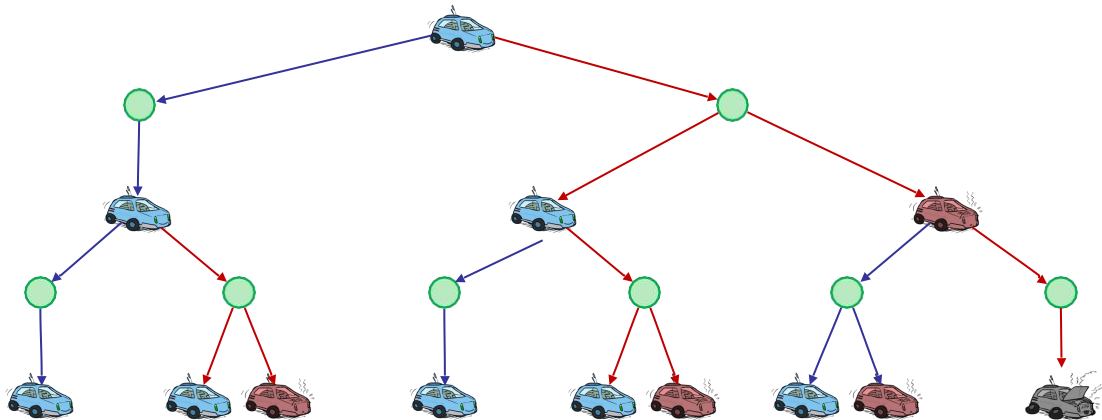


# Exemplo: Corrida

- Carro quer ir longe e rápido.
- Três estados: **frio**, **quente** e super aquecido.
- Duas ações: **rápido** e **devagar**.

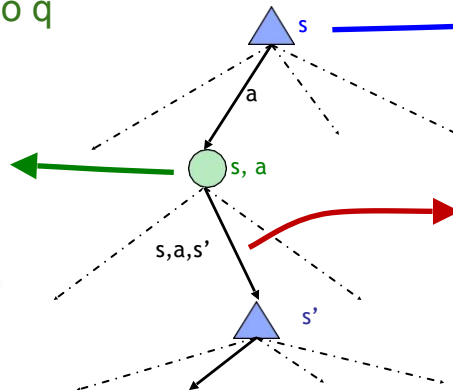
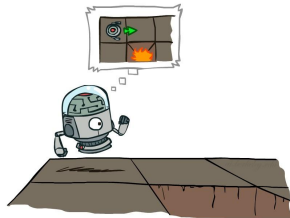


# Árvore de Busca



# Árvore de Busca MDP

$(s,a)$  é um estado  $q$



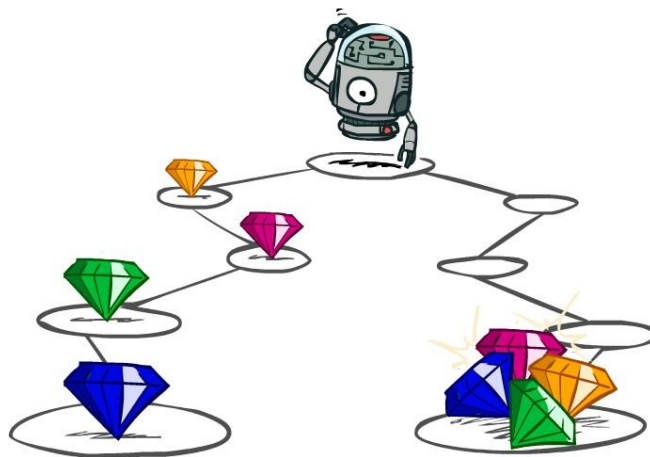
$s$  é um estado



$(s,a,s')$  é uma transição  
 $T(s,a,s') = P(s' | s,a)$   
 $R(s,a,s')$



# Sequência de Utilidades

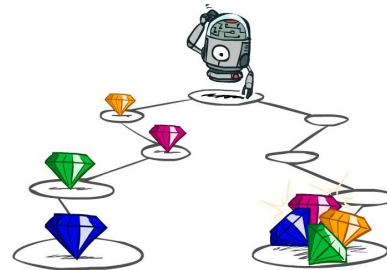


# Sequência de Utilidades

- Mais ou menos?

[2, 3, 4] ou [1, 2, 2]

- Agora ou mais tarde? [1, 0, 0] ou [0, 0, 1]





# Desconto

- É natural maximizar a soma das recompensas. É natural também preferir
- recompensas agora. **Solução: descontar!**
- 



1

valor agora



$\gamma$

valor no próximo  
passo



$\gamma^2$

valor em dois passos

# Desconto

- Como descontar?

Multiplica-se a recompensa pelo fator de desconto em cada nível da árvore.

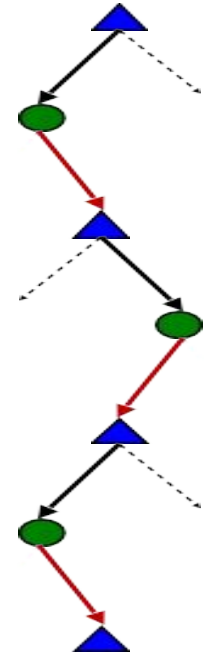
- Por que descontar?

Recompensas próximas valem mais. Ajuda a fazer os algoritmos convergirem.

- Exemplo: desconto de 0.5

$$U(1, 2, 3) = 1 + 0.5 \cdot 2 + 0.25 \cdot 3$$

$$U(1, 2, 3) < U(3, 2, 1)$$



# Quiz

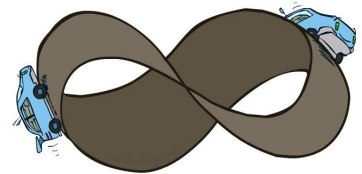
- Ações: leste, oeste, ambas determinísticas.

10				1
a	b	c	d	e

1. Para  $\gamma = 1$  qual a política ótima?
2. Para  $\gamma = 0.1$  qual a política ótima?
3. Para qual  $\gamma$  as ações leste e oeste são igualmente boas no estado d?

# Utilidade Infinita?

- Se o problema durar para sempre, teremos recompensa infinita?



- Soluções:

1. Horizonte finito (similar a busca truncada), onde um episódio termina em  $t$  passos de tempo.

2. Desconto:  $0 < \gamma < 1$  
$$U([r_0, \dots, r_\infty]) = \sum_{t=0}^{\infty} \gamma^t r_t \leq R_{\max}/(1 - \gamma)$$

3. Estado absorvente: MDP termina eventualmente (super aquecimento na corrida)

# Recapitulando: MDPs

- MDP:

Conjunto de estados  $s \in S$

Conjunto de ações  $a \in A$  Função de transição  $T(s, a, s')$

probabilidade de  $a$  nos levar a  $s'$  a partir de  $s$  i.e.,  $P(s'|s, a)$

Função de recompensa  $R(s, a, s')$

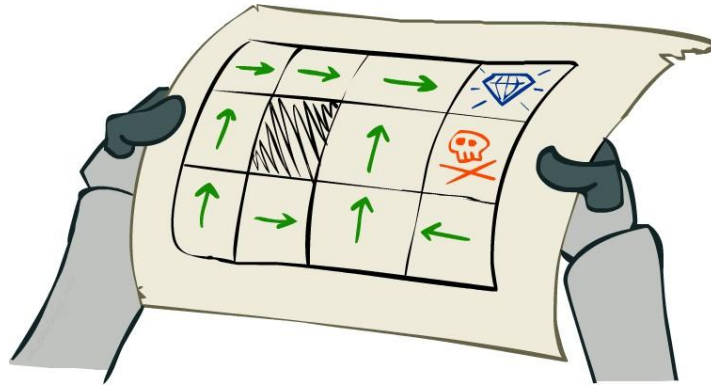
as vezes escrita como  $R(s')$

Um estado inicial

As vezes um estado final.

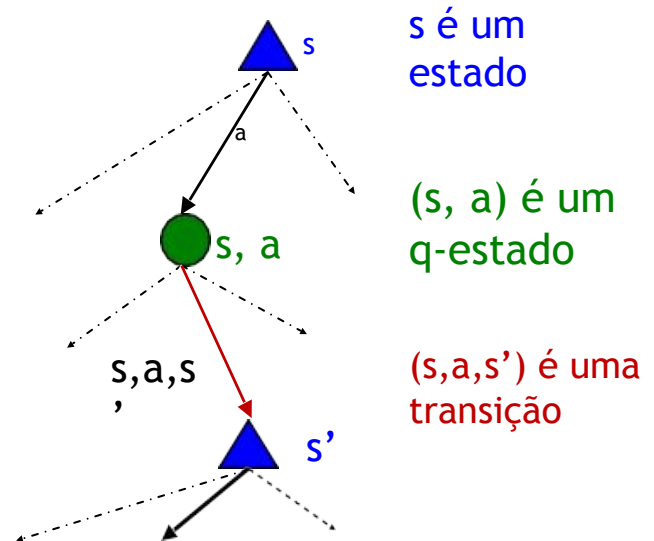
- Política: escolha de ação em um dado estado
- Utilidade: soma de recompensas descontadas.

# Solucionando MDPs



# Valores Ótimos

- $V^*(s)$  é a utilidade esperada começando-se em  $s$  e agindo de forma ótima.
- $Q^*(s, a)$  é a utilidade esperada ao tomar ação  $a$  a partir de  $s$  e assumindo-se ações ótimas futuras.
- $\pi^*(s)$  é a ação ótima em  $s$ .

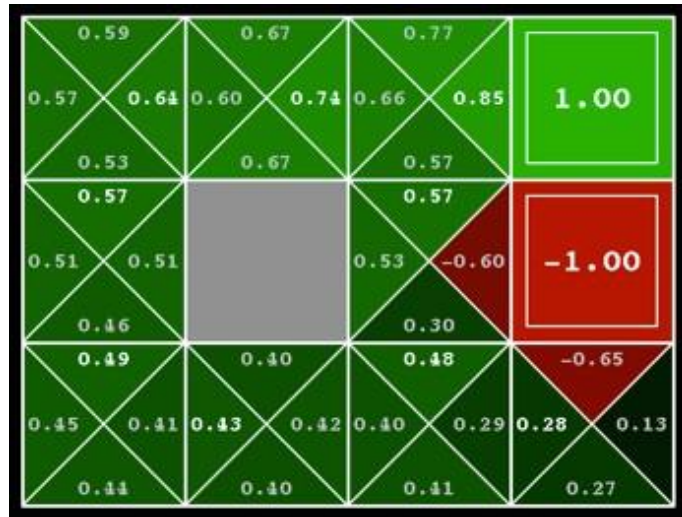


# Valores de Utilidade

0.64 ▶	0.74 ▶	0.85 ▶	1.00
▲ 0.57		▲ 0.57	-1.00
▲ 0.49	◀ 0.43	▲ 0.48	◀ 0.28



# Valores Q



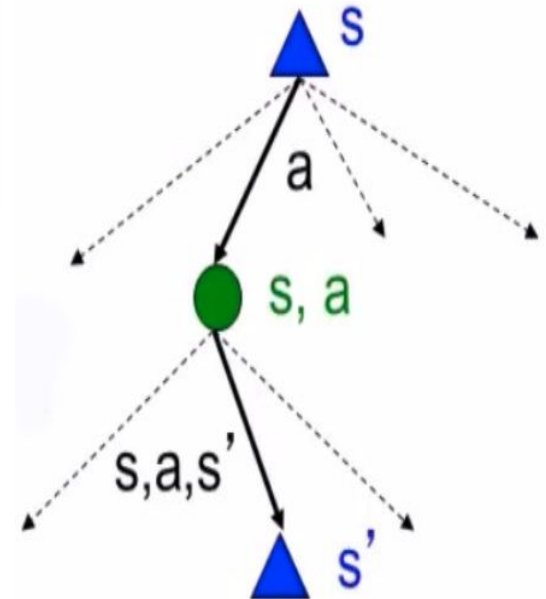
# Valores de Estados

O valor  $V^*(s)$  era justamente o que expectimax calculava. Definição recursiva (Equações de Bellman)

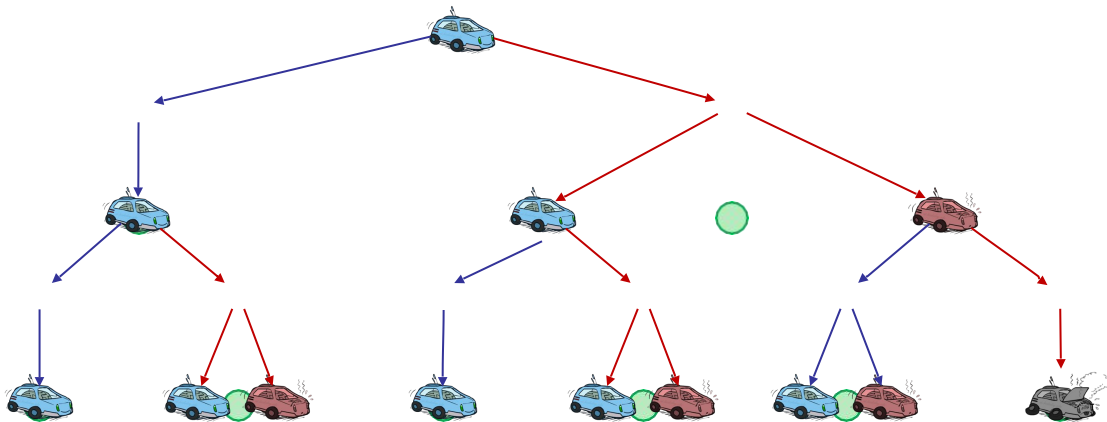
$$V^*(s) = \max_a Q^*(s, a)$$

$$Q^*(s, a) = \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$

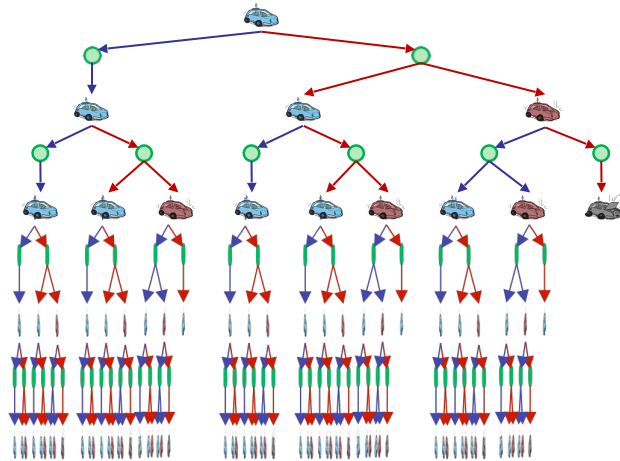
$$V^*(s) = \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$



# Árvore de Busca

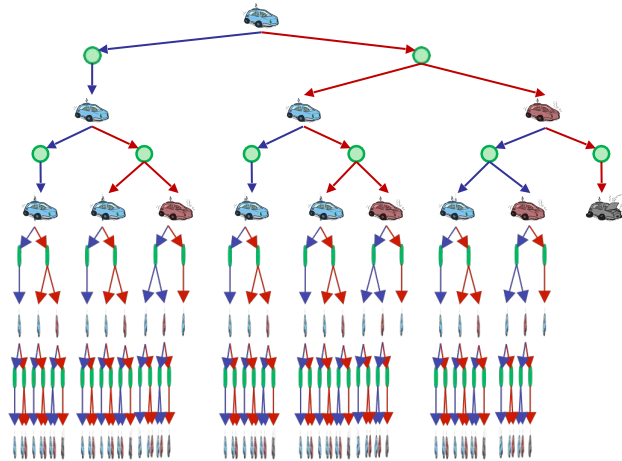


# Árvore de Busca



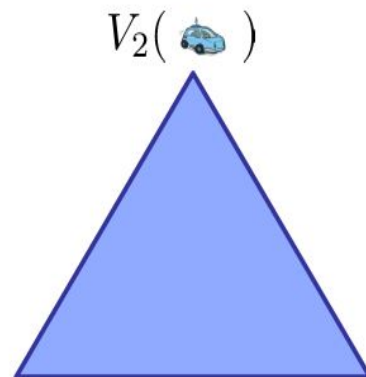
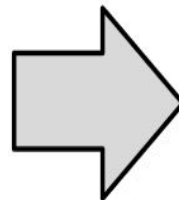
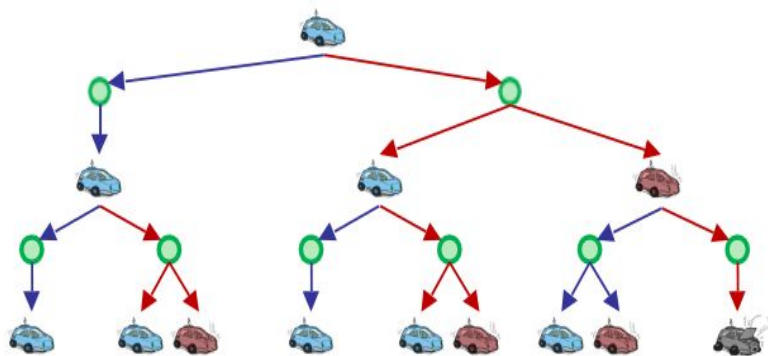
# Árvore de Busca

Expectimax faria trabalho redundante pois vários estados se repetem.

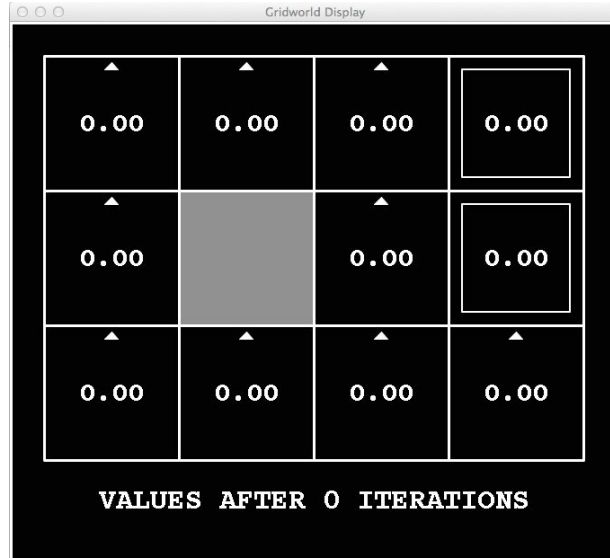


# Valores Limitados por Tempo

- Defina  $V_k(s)$  o valor ótimo de  $s$  se o jogo termina em  $k$  passos.
- Equivalente a uma busca até profundidade  $k$  com expectimax.



# k=0



# k=1





# k=2



# k=3



# k=4



# k=5



# k=6



# k=7



# k=8



# k=9





# k=10



# k=11



# k=12



# k=100






# Iteração de Valor

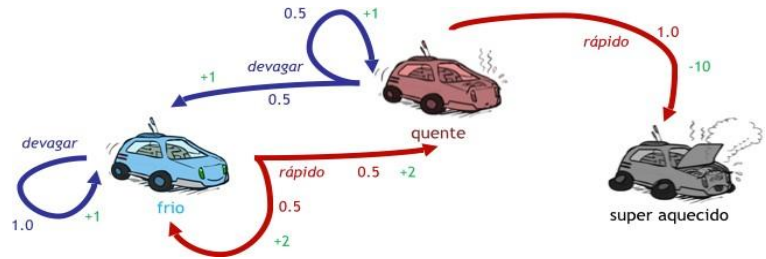
1. Inicializa-se  $V_0(S) = 0$  para todos os estados  $s$
2. Faça  $t = 0$
3. Repita até convergir:
  - a. para todo  $s \in S$

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') \left[ R(s, a, s') + \gamma V_k(s') \right]$$

- Complexidade iteração:  $O(S^2A)$
- Teorema:  $V_k(s)$  converge para  $V^*(s)$

# Exemplo: Iteração de Valor

			
$V_2$	3.5	2.5	0
$V_1$	2	1	0
$V_0$	0	0	0



Assuma  $\gamma = 1.0$

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V_k(s')]$$

FIM