Markov Decision Process
Value Iteration

Anna Helena Reali Costa

Lecture slides partially extracted from:
http://aima.eecs.berkeley.edu/instructors.html
http://ocw.mit.edu/ (course 6.825)
http://www.laas.fr/planning/
Chapter 13: Machine Learning, Tom Mitchell

Characteristics of the problem

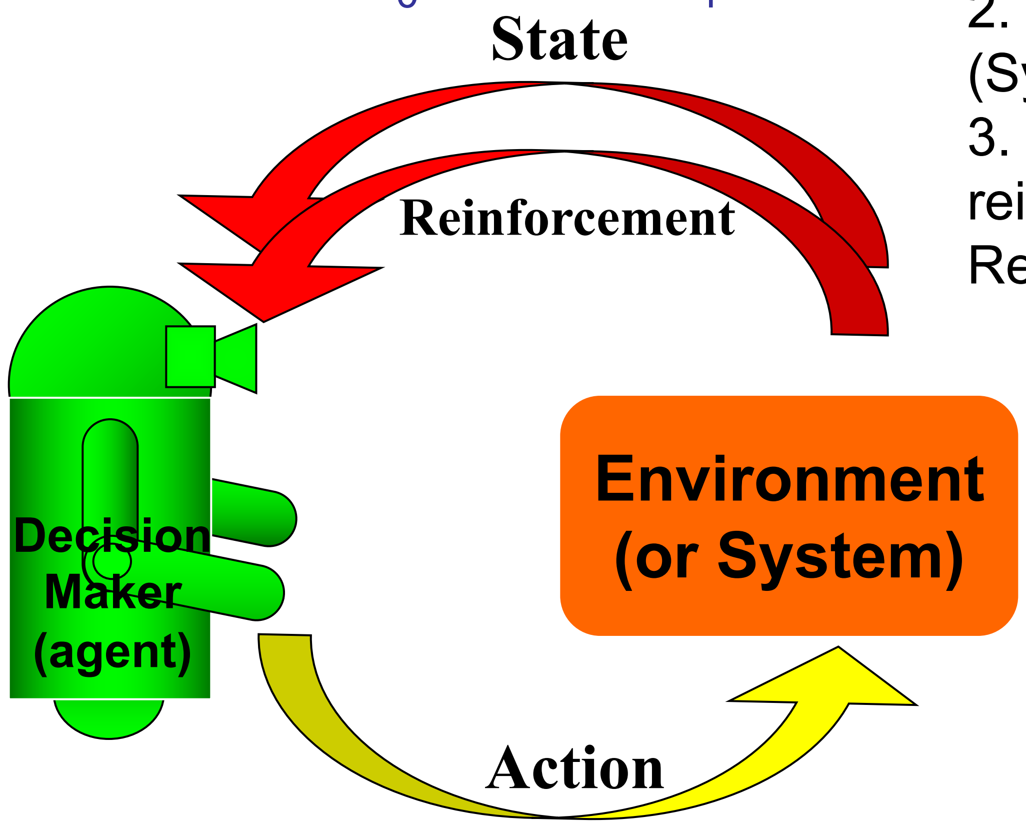
- The agent has a set of **sensors** to observe the ***state*** of its environment
- The agent has a set of ***actions*** it can perform to alter this state
- The agent perceives a **reward** (or penalty) to indicate the desirability of the resulting state

- The task of the agent is to learn from this <u>indirect</u>, <u>delayed</u> reward, to choose **sequences of actions** that produce the <u>greatest cumulative reward</u>

➔ it is a ***sequential decision problem***

Sequential decision problem

$$s_0 \xrightarrow[r_0]{a_0} s_1 \xrightarrow[r_1]{a_1} \cdots$$

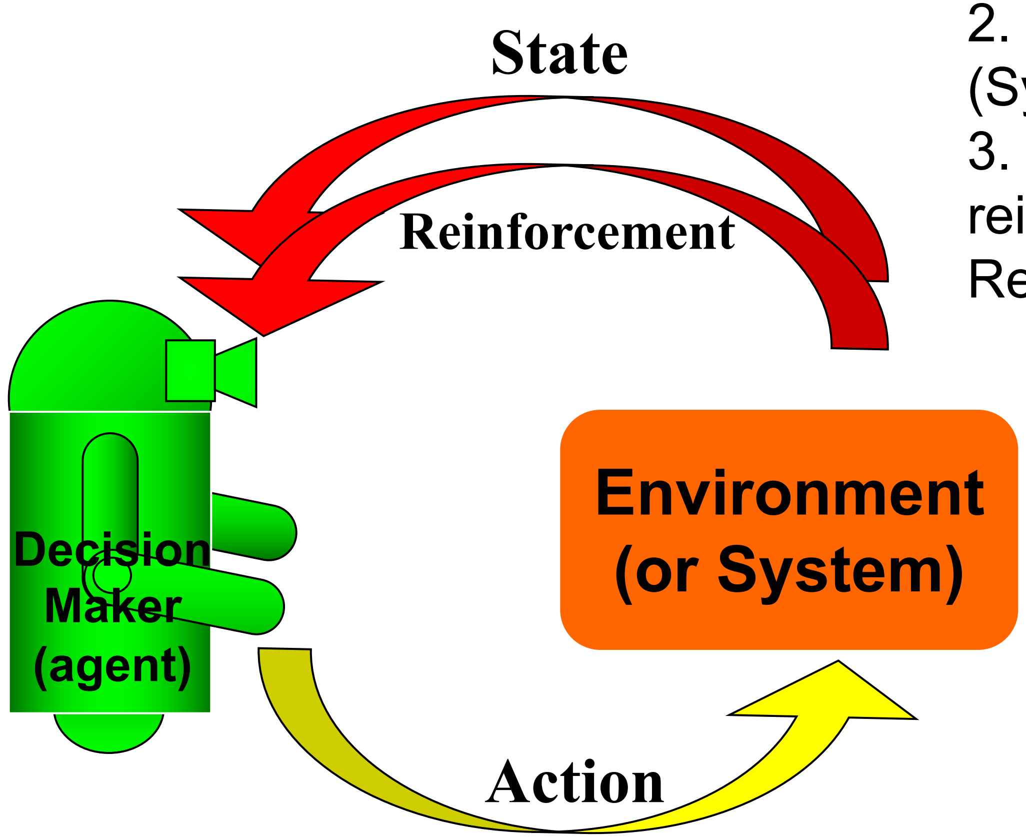


At each time step the decision maker:

1. Observes the state of the system;
2. Chooses an action and applies it;
(System evolves to a new state)
3. Observes an immediate reinforcement (reward or penalty);
Repeat 1 – 3

Sequential decision problem

$s_0 \xrightarrow[r_0]{a_0} s_1 \xrightarrow[r_1]{a_1} \cdots$



At each time step the decision maker:

1. Observes the state of the system;
2. Chooses an action and applies it;
(System evolves to a new state)
3. Observes an immediate reinforcement;
Repeat 1 – 3

This assumes discrete time.

Decisions are made at points of time referred to as ***decision epochs.***

The set of decision epochs can be **finite or infinite**: $T = \{0,1,2,\ldots,N\}$, $N \leq \infty$.

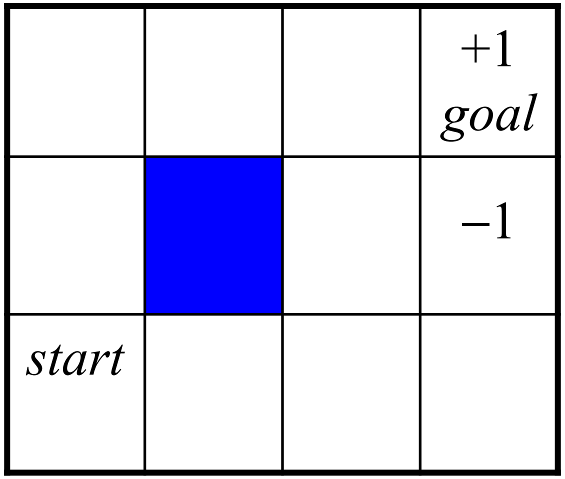
FAZER: Testinho 1 e 2

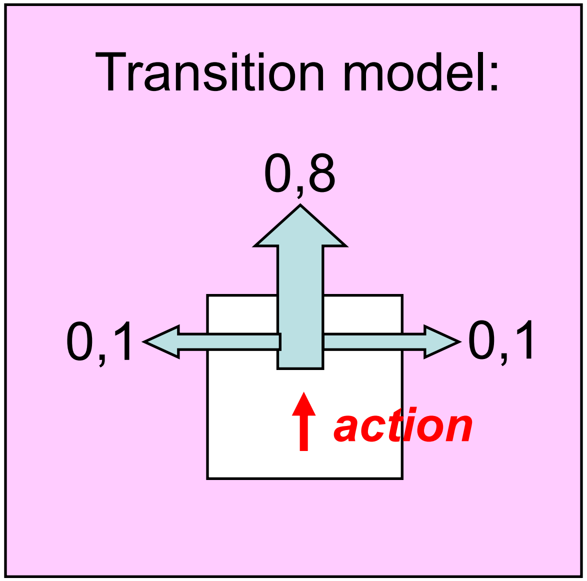
- 4×3 discrete fully-observed environment
- Actions: Up, Down, Left and Right
- Initial state: (1,1)
- Sequence **[U, U, R, R, R]**:

  (i) goes up around the barrier and reaches the goal state (4,3) with probability

  ………

  (ii) there is also a chance of accidentally reaching the goal by going **(1,1) → (2,1) → (3,1) → (3,2) → (3,3) → (4,3)** with probability ………





Utility function

- **Utility function** for the agent depends on a sequence of states (environment ***history***)
- In each state **s**, the agent receives a **reinforcement** **r(s)**, which may be positive or negative, but must be **bounded.**
- **Utility = sum of the rewards received**

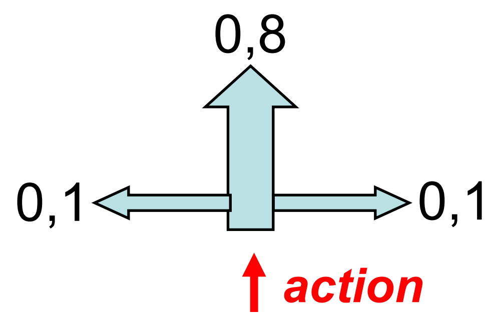
FAZER: testinho 3

- **Utility function**: sum of the rewards received
- Here: **r(s) = –0.04** ∀s except
  r(4,3) = +1 and r(4,2) = −1

➔ reach goal after **10 steps** (avoiding (4,2)):
utility = ...

-0,04	-0,04	-0,04	**+1** *goal*
-0,04		-0,04	**–1**
-0,04 *start*	-0,04	-0,04	-0,04

Transition model:



Ex: (1,1),U,(1,2),D,(1,1),U,(1,2),U,(1,3),L,(1,3),R,(2,3),R,(3,3),L,(2,3),R,(3,3),R,(4,3)

Modeling sequential decision problems as Markov Decision Processes

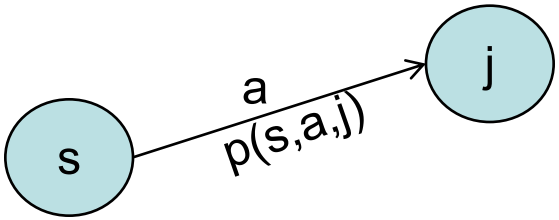
MDP – Model Formulation

An MDP is defined as $\langle S, A, p, r \rangle$:

- **S** is the set of possible system states (arbitrary finite set);
- **A** is the set of allowable actions (arbitrary finite set);
- **p**: $S \times A \times S \rightarrow [0,1]$ is the **transition probability function**;
- **r**: $S \times A \rightarrow \Re$ is the **reinforcement function**;

MDP

- The set **A** of allowable actions:
  - $A = \cup_{s \in S} A_s$ where A_s is the set of allowable actions in state $s \in S$
  - Or we might restrict the model: $A = A_s$ for all $s \in S$
- The transition probability function **p:**
  - $p(j \mid s, a)$ --- or $p(s, a, j)$ --- denotes the probability that the system is in state $j \in S$ at time $t+1$, when the decision maker performs action $a \in A_s$ in state $s \in S$ at time t.
  - $\sum_{j \in S} p(j \mid s, a) = 1$



MDP

- The reinforcement function **r:**
  - r(s,a), denotes the value of the reward (or reinforcement or cost) received when performing $a \in A_s$ in $s \in S$ at time t.
  - When positive, r(s,a) is an *income*, and when negative it is a ***cost***.
  - Can be:

    r(s);

    r(s,a);

    r(s,a,j) with $r(s,a) = \sum_{j \in S} r(s,a,j)\, p(j \mid s, a)$

Why "Markov"?

- The qualifier *Markov* is used because the transition probability function **p** and the reinforcement function **r** depend on the past through the current state of the system and the action selected by the decision maker in that state.

Notation: $X_{a:b} = X_a, X_{a+1}, \ldots, X_{b-1}, X_b$

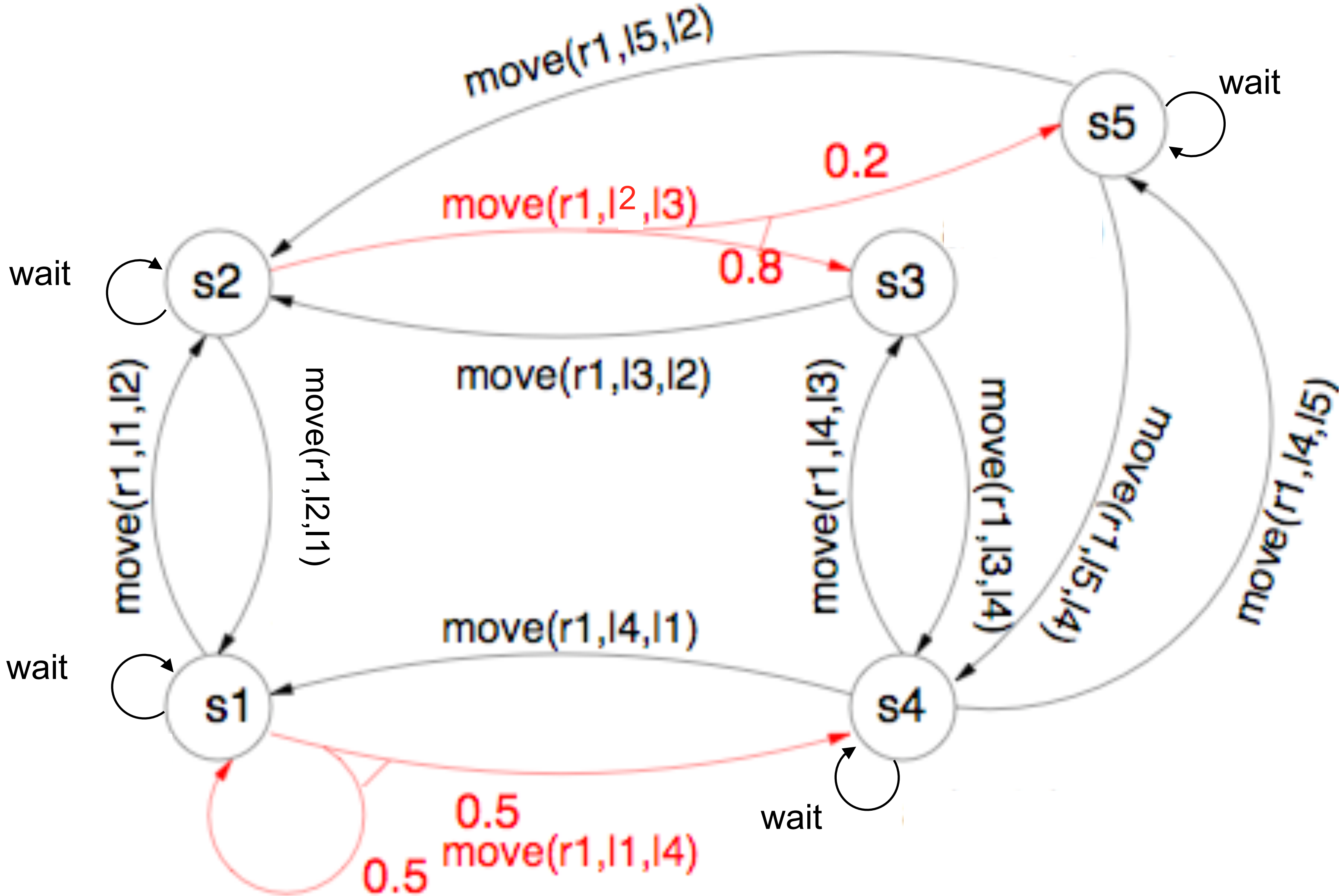
Markov assumption: X_t depends on <u>bounded</u> subset of $X_{0:t-1}$

- <u>First-order Markov process</u>: $\mathbf{p(X_t | X_{0:t-1}) = p(X_t | X_{t-1})}$

Example of an MDP

- A = {move(r1,l1,l2), move(r1,l2,l1), move(r1,l4,l1), move(r1,l1,l4), move(r1,l3,l2), move(r1,l2,l3), move(r1,l5,l2), move(r1,l4,l3), move(r1,l3,l4), move(r1,l5,l4), move(r1,l4,l5), wait}
- S = {s1, s2, s3, s4, s5}
- p(s1,move(r1,l1,l4),s4)=0.5; p(s1,move(r1,l1,l4),s1)=0.5;
  p(s2,move(r1,l2,l3),s3)=0.8; p(s2,move(r1,l2,l3),s5)=0.2;
  All others p(.) have a value of 1.
- r(s1,wait) = r(s2,wait) = -1; r(s4,wait)=0; r(s5,wait)= -100;
  r(s1,move(r1,l1,l2))=r(s2,move(r1,l2,l1))= -100;
  r(s3,move(r1,l3,l4))=r(s4,move(r1,l4,l3))= -100;
  r(s4,move(r1,l4,l5))=r(s5,move(r1,l5,l4))= -100;
  r(s1,move(r1,l1,l4))=r(s4,move(r1,l4,l1))= -1;
  r(s2,move(r1,l2,l3))=r(s3,move(r1,l3,l2))= -1;
  r(s5,move(r1,l5,l2))= -1; r(s1)=r(s2)=r(s3)=r(s5)=0; r(s4)=100

Example of an MDP

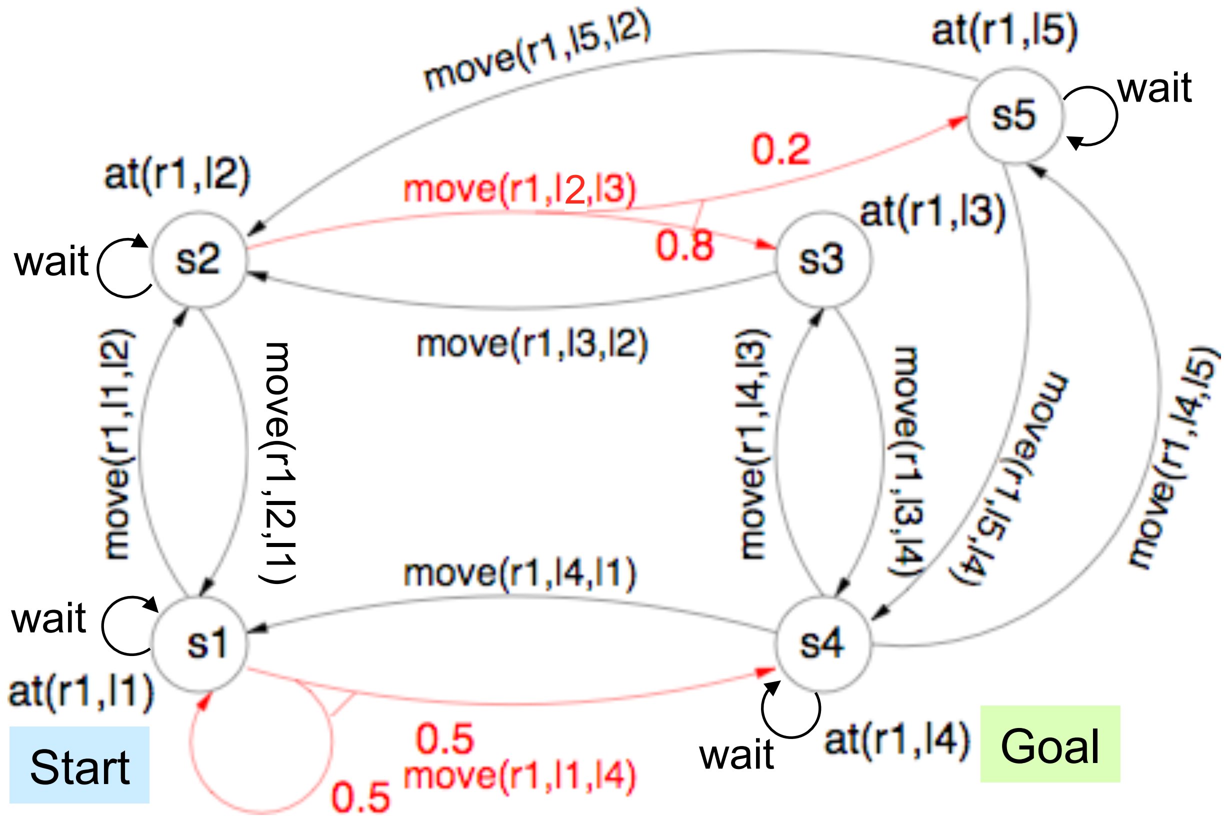


What is a Solution?

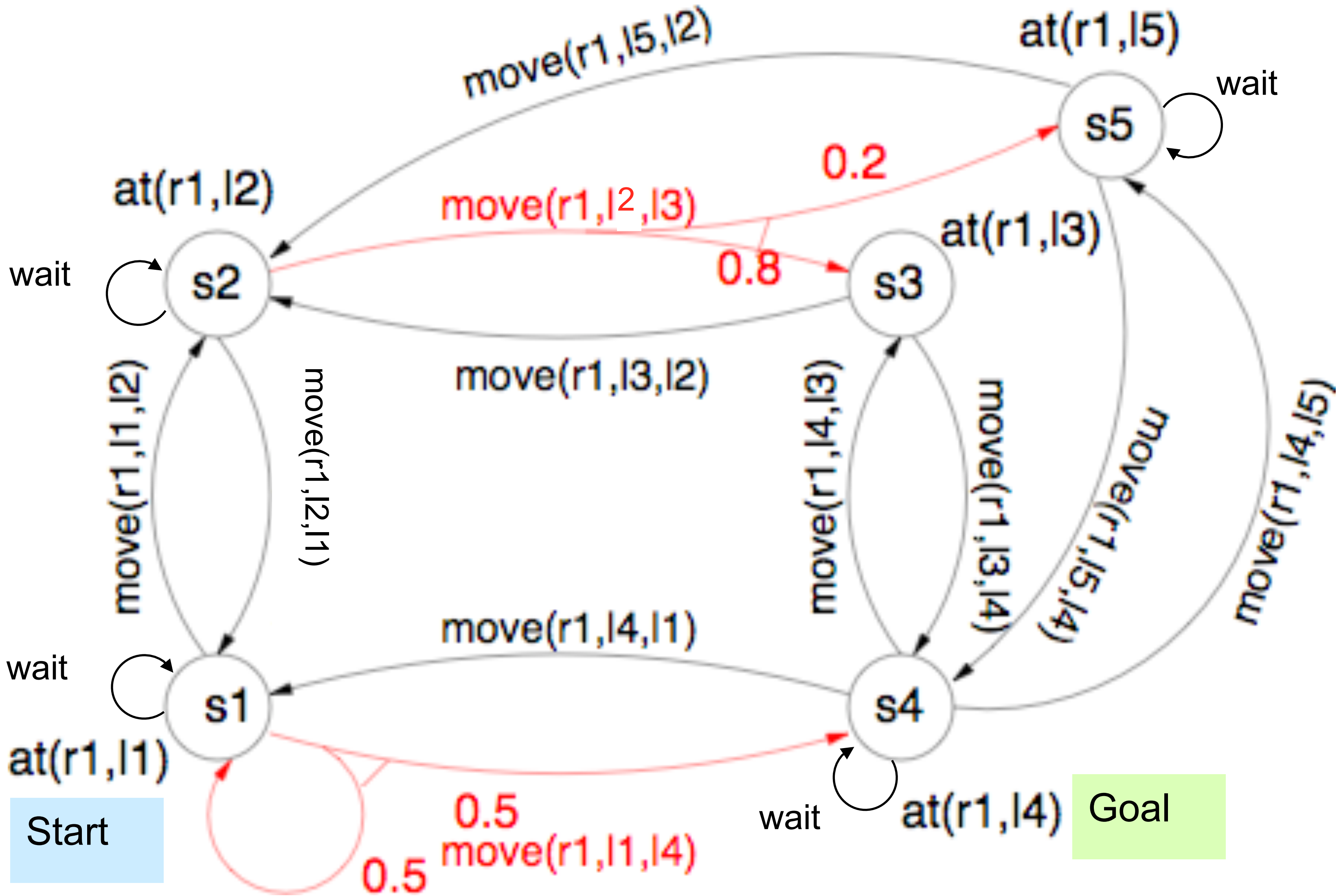
- What does a solution to the problem look like?
  - Any fixed action sequence (classical planning) will **not** solve the problem!

Example

- Robot r1 starts at location l1
  - State s1 in the diagram
- Objective is to get r1 to location l4
  - State s4 in the diagram



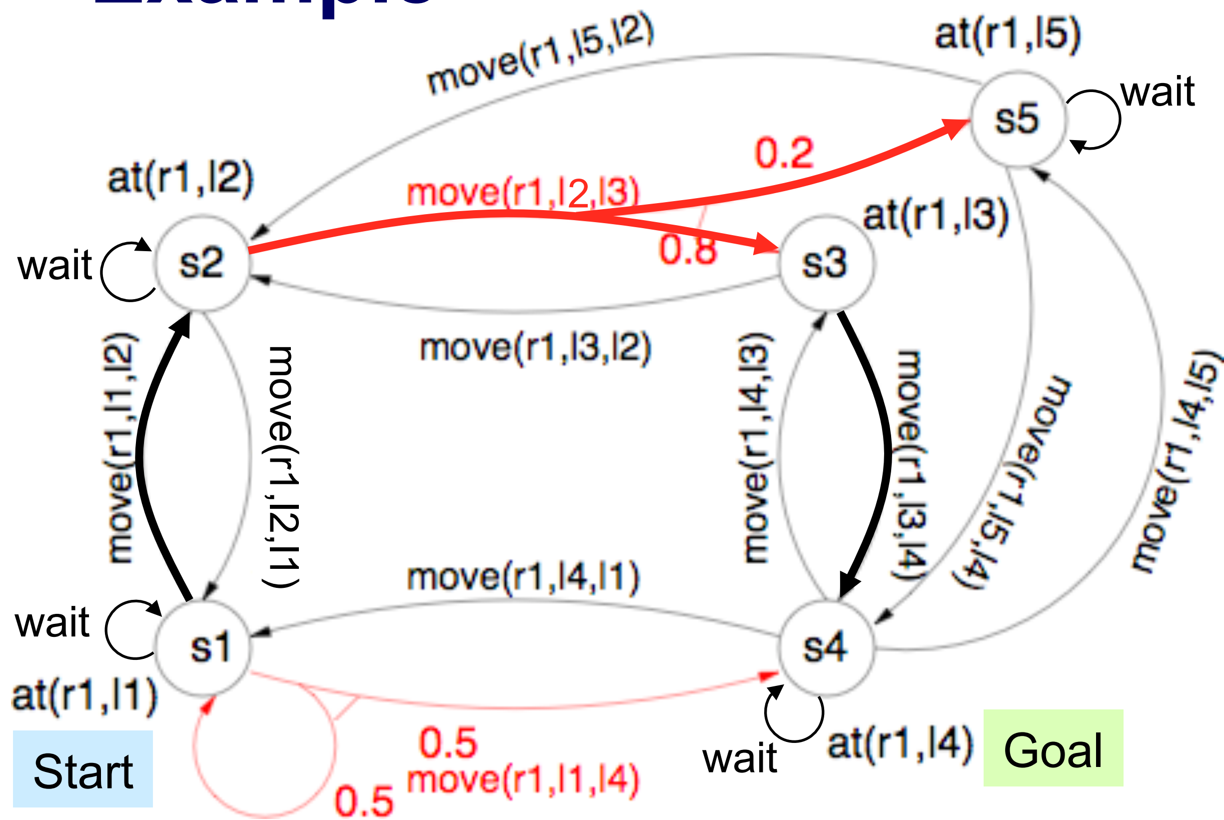
Is there a plan that will guarantee the solution?



Dana Nau: Lecture slides for Automated Planning

Example

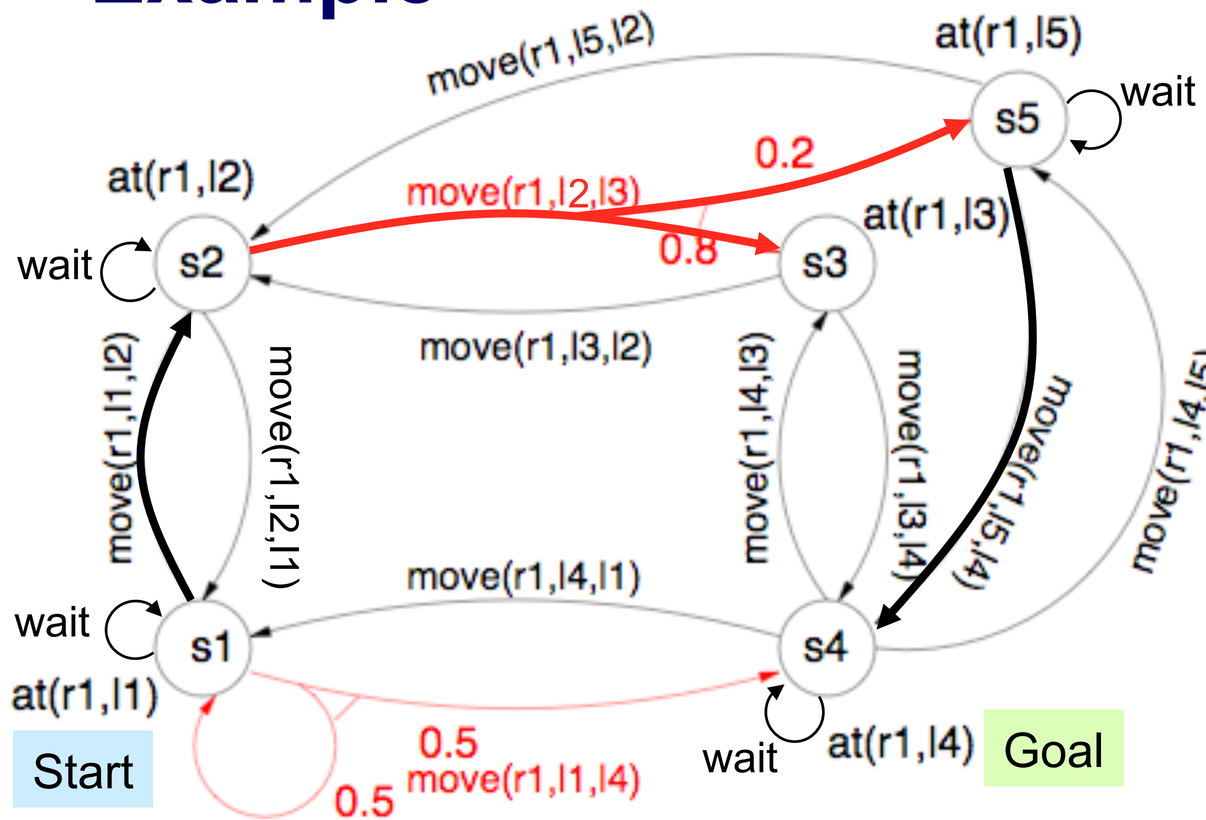
- Robot r1 starts at location l1
  - State s1 in the diagram
- Objective is to get r1 to location l4
  - State s4 in the diagram



- No fixed sequence of actions can be a solution, because we can not guarantee we will be in a state where the next action is applicable
  - e.g.,

Plan 1:⟨move(r1,l1,l2), move(r1,l2,l3), move(r1,l3,l4)⟩

Example

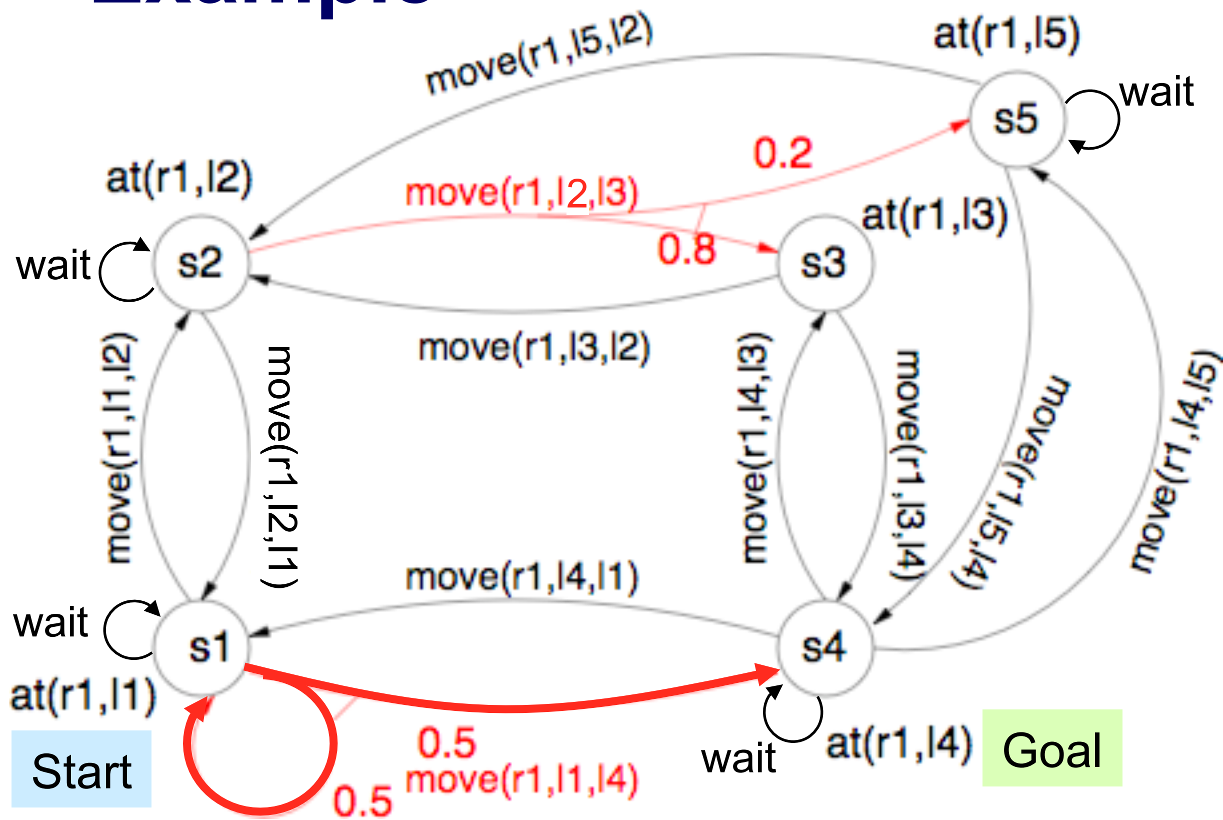


- Robot r1 starts at location l1
  - State s1 in the diagram
- Objective is to get r1 to location l4
  - State s4 in the diagram
- No fixed sequence of actions can be a solution, because we can not guarantee we will be in a state where the next action is applicable
  - e.g.,

Plan 2: ⟨move(r1,l1,l2), move(r1,l2,l3), move(r1,l5,l4)⟩

Example

- Robot r1 starts at location l1
  - State s1 in the diagram
- Objective is to get r1 to location l4
  - State s4 in the diagram



- No fixed sequence of actions can be a solution, because we can not guarantee we will be in a state where the next action is applicable
  - e.g.,

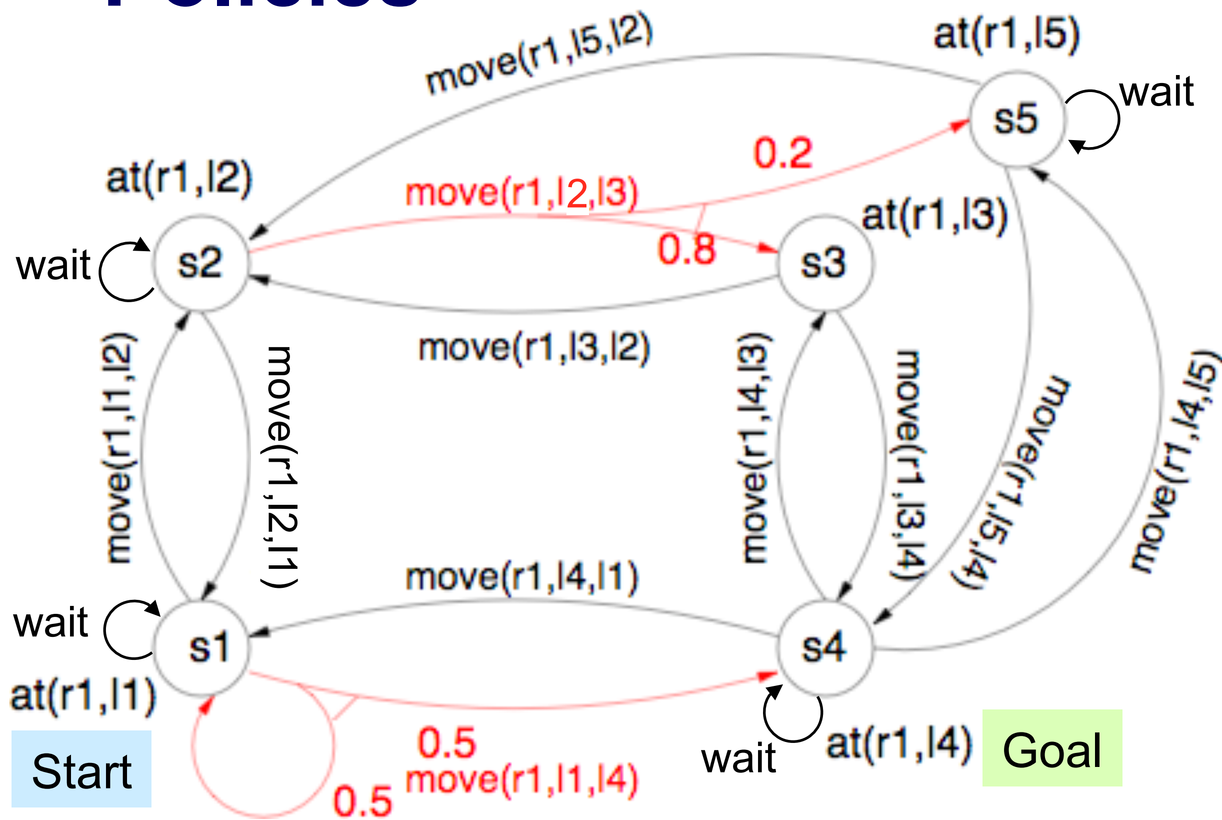
Plan 3: ⟨move(r1,l1,l4)⟩

What is a Solution?

- A solution must specify what the agent should do for ***any*** state that the agent might reach
  ➔ ***policy*** π

$$\Pi: S \rightarrow A, \quad \pi(s) = a \in A$$

Policies



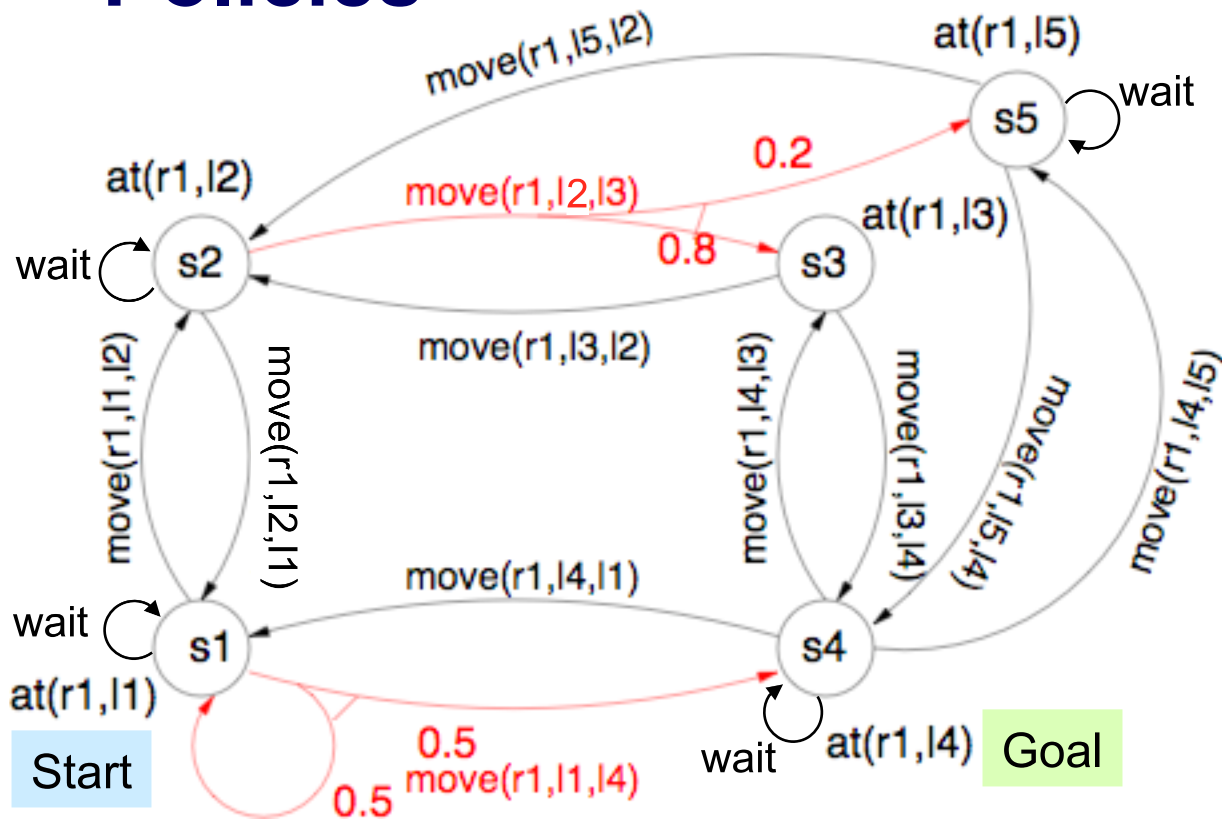
- Policy: a function that maps states into actions
- Write it as a set of state-action pairs

Policies

π_1 = {(s1, move(r1,l1,l2)),
(s2, move(r1,l2,l3)),
(s3, move(r1,l3,l4)),
(s4, wait),
(s5, wait)}

π_2 = {(s1, move(r1,l1,l2)),
(s2, move(r1,l2,l3)),
(s3, move(r1,l3,l4)),
(s4, wait),
(s5, move(r1,l5,l4))}

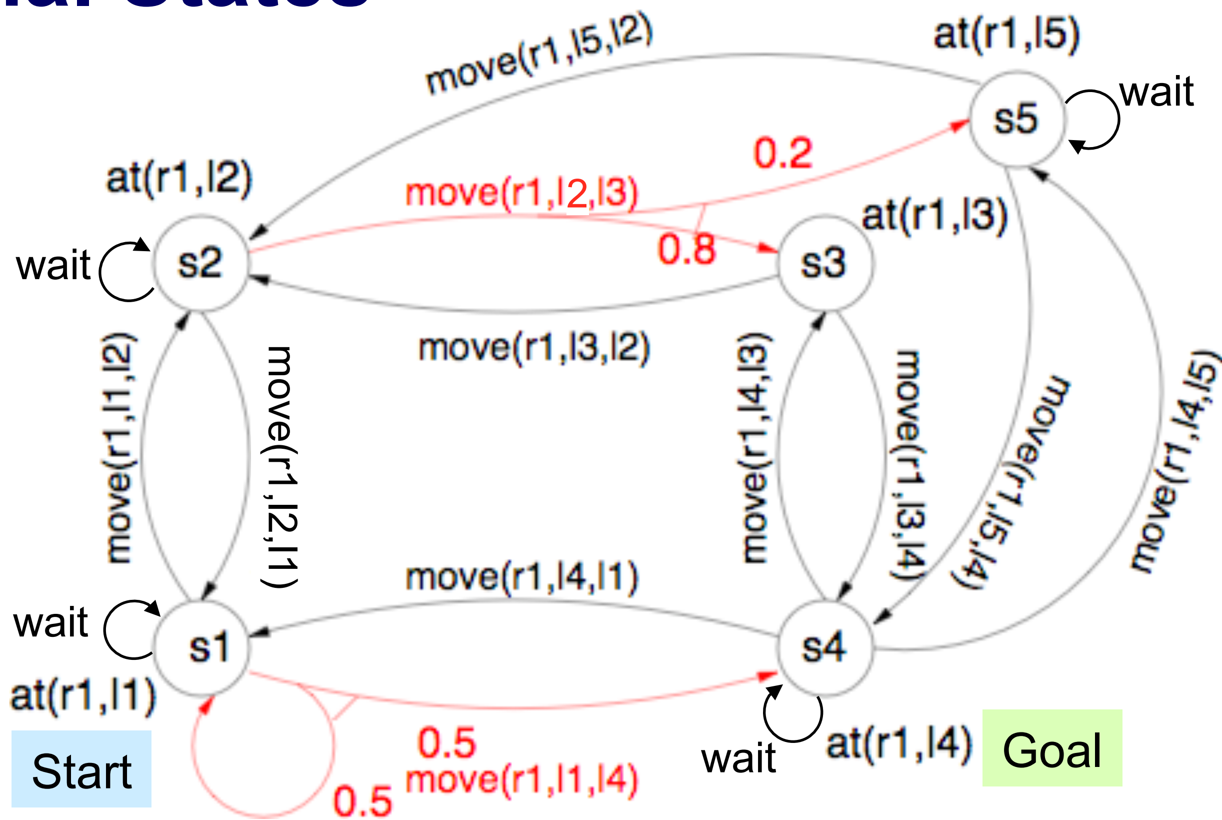
π_3 = {(s1, move(r1,l1,l4)),
(s2, move(r1,l2,l1)),
(s3, move(r1,l3,l4)),
(s4, wait),
(s5, move(r1,l5,l4)}



- Policy: a function that maps states into actions
- Write it as a set of state-action pairs

Initial States

- For every state s, there will be a probability $P(s)$ that the system begins in the state s
  - ◆ We assume the system starts in a unique initial state s_0
    - » $P(s_0) = 1$
    - » $P(s_i) = 0$ for $i \neq 0$



- In the example, $P(s_1) = 1$, and $P(s) = 0$ for all other states

Histories

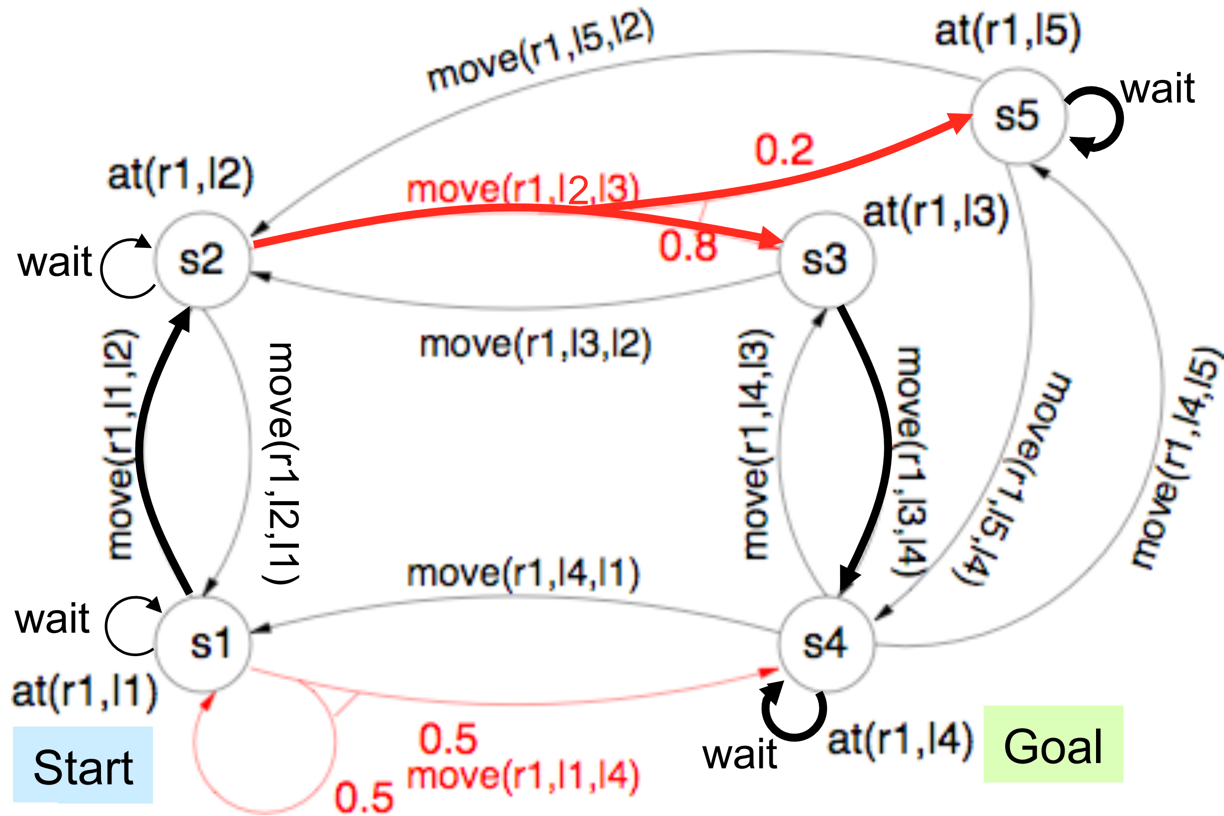
- Each time a given policy is executed starting from the initial state, the stochastic nature of the environment will lead to a different environment <u>history</u>.

- **Each policy induces a probability distribution over histories**
  - If $h = \langle s_0, s_1, \ldots \rangle$ then

$$P(h \mid \pi) = P(s_0) \prod_{i \geq 0} p_{\pi(s_i)} (s_{i+1} \mid s_i, a_i)$$

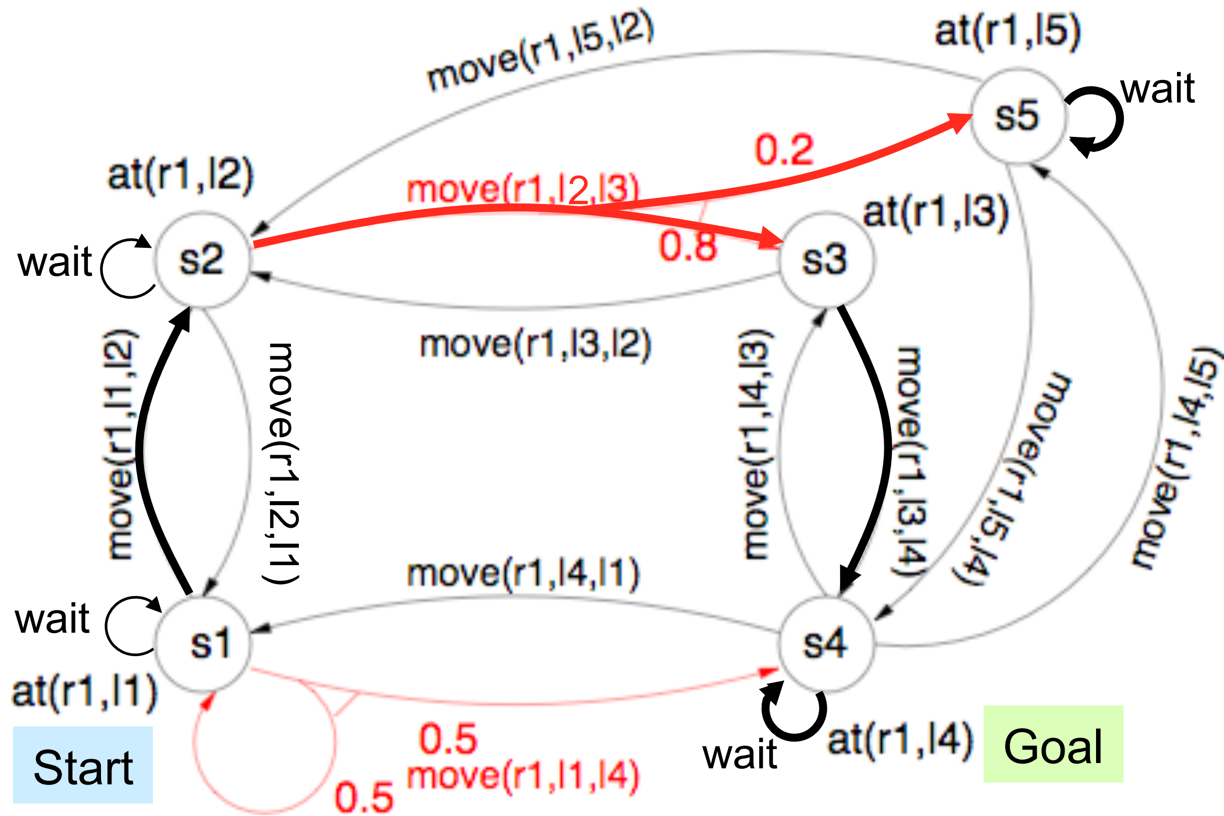
FAZER Testinho

$\pi_1 = \{$(s1, move(r1,l1,l2)),
(s2, move(r1,l2,l3)),
(s3, move(r1,l3,l4)),
(s4, wait),
(s5, wait)$\}$



Qual(is) história(s) é(são) desenvolvida(s) por esta política neste MDP?

FAZER TESTINHO!!!



$\pi_1 = \{$(s1, move(r1,l1,l2)),
(s2, move(r1,l2,l3)),
(s3, move(r1,l3,l4)),
(s4, wait),
(s5, wait)$\}$

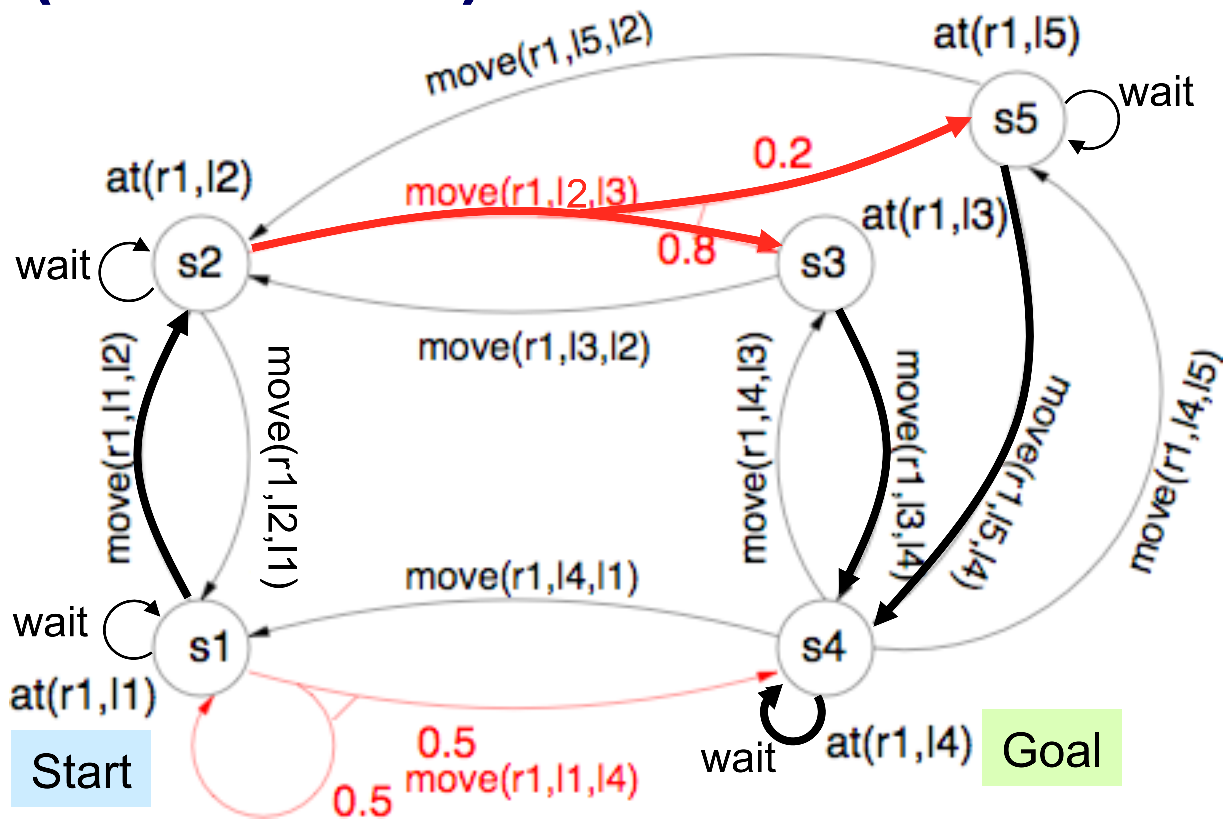
goal

$h_1 = \langle$s1, s2, s3, s4, s4, … $\rangle$

$h_2 = \langle$s1, s2, s5, s5 … $\rangle$

Qual a probabilidade de desenvolver h1 neste MDP?

Example (continued)



$\pi_2 = \{$(s1, move(r1,l1,l2)),
(s2, move(r1,l2,l3)),
(s3, move(r1,l3,l4)),
(s4, wait),
(s5, move(r1,l5,l4))$\}$

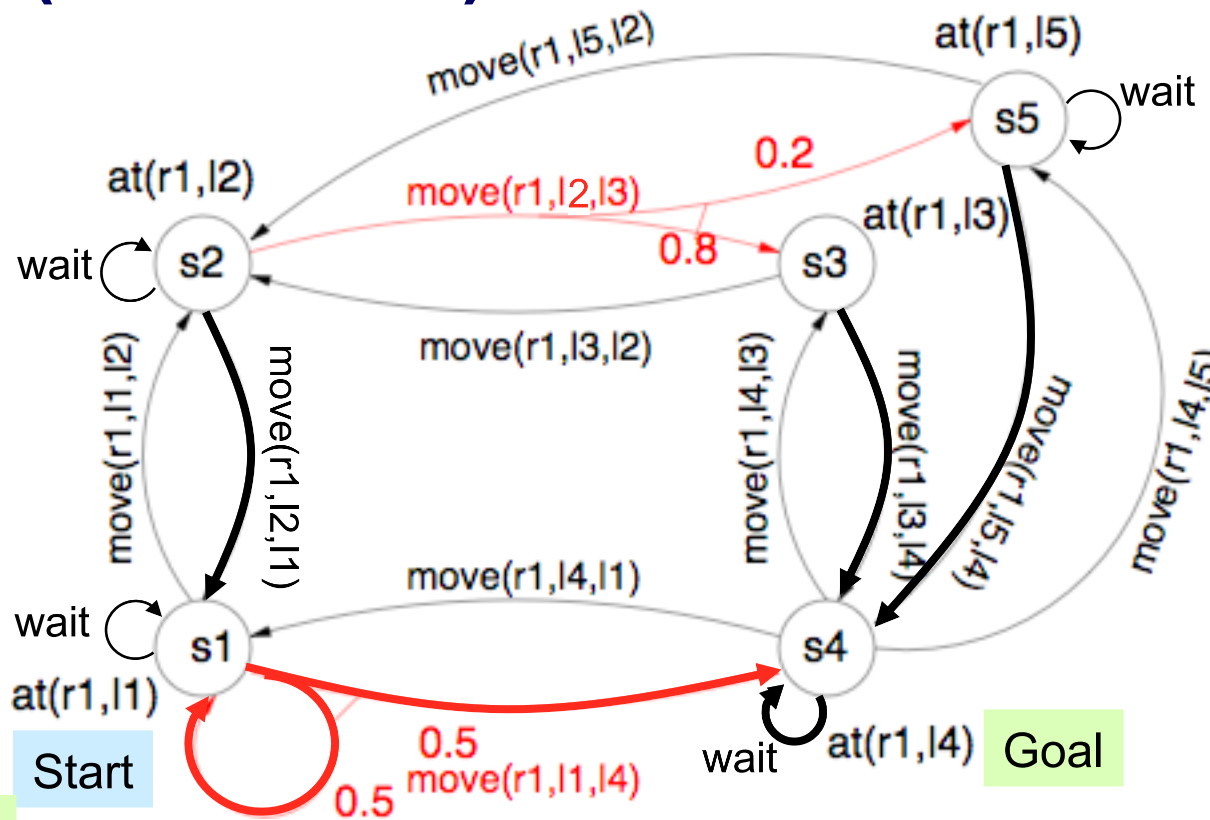
goal

$h_1 = \langle$s1, s2, s3, s4, s4, … $\rangle$

$h_3 = \langle$s1, s2, s5, s4, s4, … $\rangle$

$$\begin{cases} P(h_1 \mid \pi_2) = 1 \times 0.8 \times 1 \times \ldots = 0.8 \\ P(h_3 \mid \pi_2) = 1 \times 0.2 \times 1 \times \ldots = 0.2 \\ P(h \mid \pi_2) = 0 \text{ for all other } h \end{cases}$$

Example (continued)



$\pi_3 = \{$(s1, move(r1,l1,l4)),
(s2, move(r1,l2,l1)),
(s3, move(r1,l3,l4)),
(s4, wait),
(s5, move(r1,l5,l4)$\}$

goal

$h_4 = \langle$s1, s4, s4, … $\rangle$

$h_5 = \langle$s1, s1, s4, s4, … $\rangle$

$h_6 = \langle$s1, s1, s1, s4, s4, … $\rangle$

…

$h_7 = \langle$s1, s1, s1, s1, s1, s1, … $\rangle$

$$P(h_4 \mid \pi_3) = 1 \times 0.5 \times 1 \times 1 \times 1 \times \ \ldots = 0.5$$

$$P(h_5 \mid \pi_3) = 1 \times 0.5 \times 0.5 \times 1 \times 1 \times \ \ldots = 0.25$$

$$P(h_6 \mid \pi_3) = 1 \times 0.5 \times 0.5 \times 0.5 \times 1 \times \ldots = 0.125$$

$$P(h_7 \mid \pi_3) = 1 \times 0.5 \times 0.5 \times 0.5 \times 0.5 \times \ldots = 0$$

Quality of a policy

- The quality of the policy is measured by the ***expected utility (***or ***value)*** of the possible environment histories generated by that policy:

$$E[V_\pi(h)] = \sum_h P(h|\,\pi)\, V_\pi(h)$$

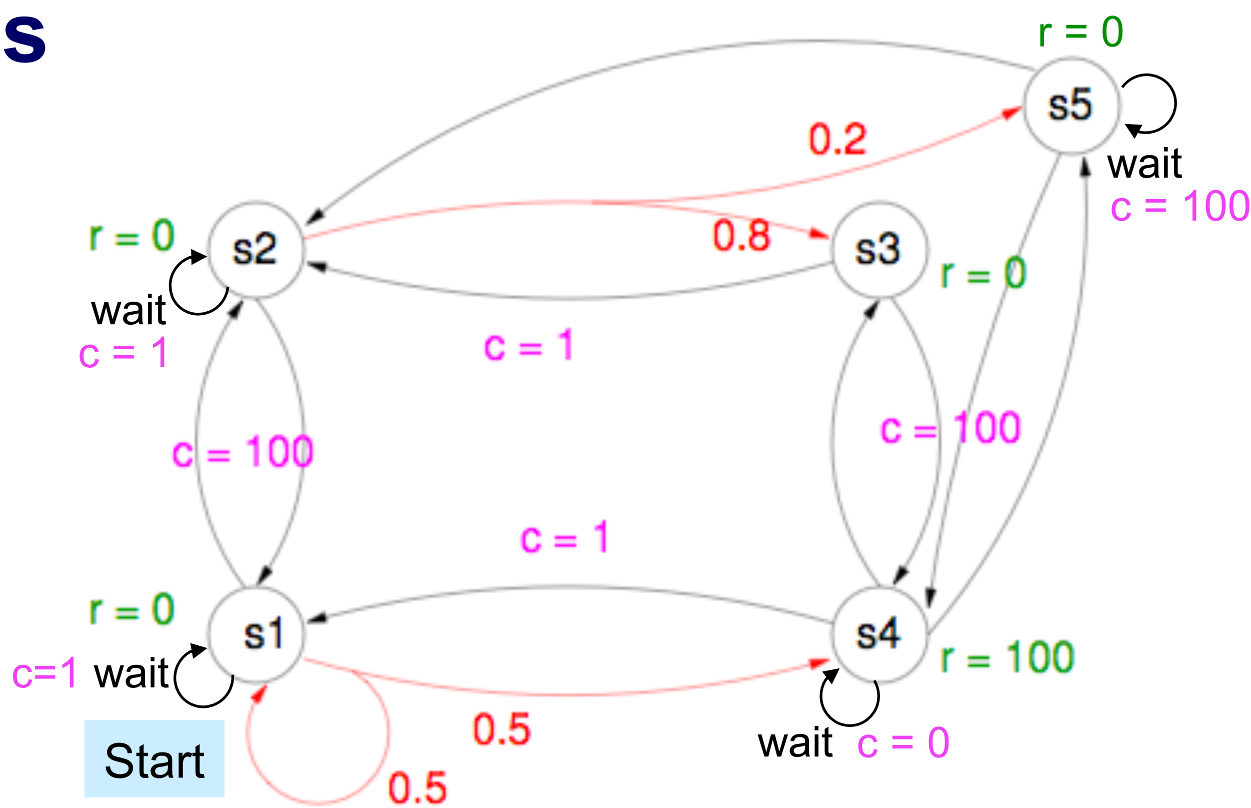
- An **optimal policy π*** is a policy that yields the **highest expected utility**.

Discounted reinforcements:

$$V_\pi(h) = \mathrm{r}(s_0) + \gamma\,\mathrm{r}(s_1) + \gamma^2\mathrm{r}(s_2) + \gamma^3\mathrm{r}(s_3) + \ldots$$

- A **discount factor *γ*:** $0 \le \gamma \le 1$

Utility Functions

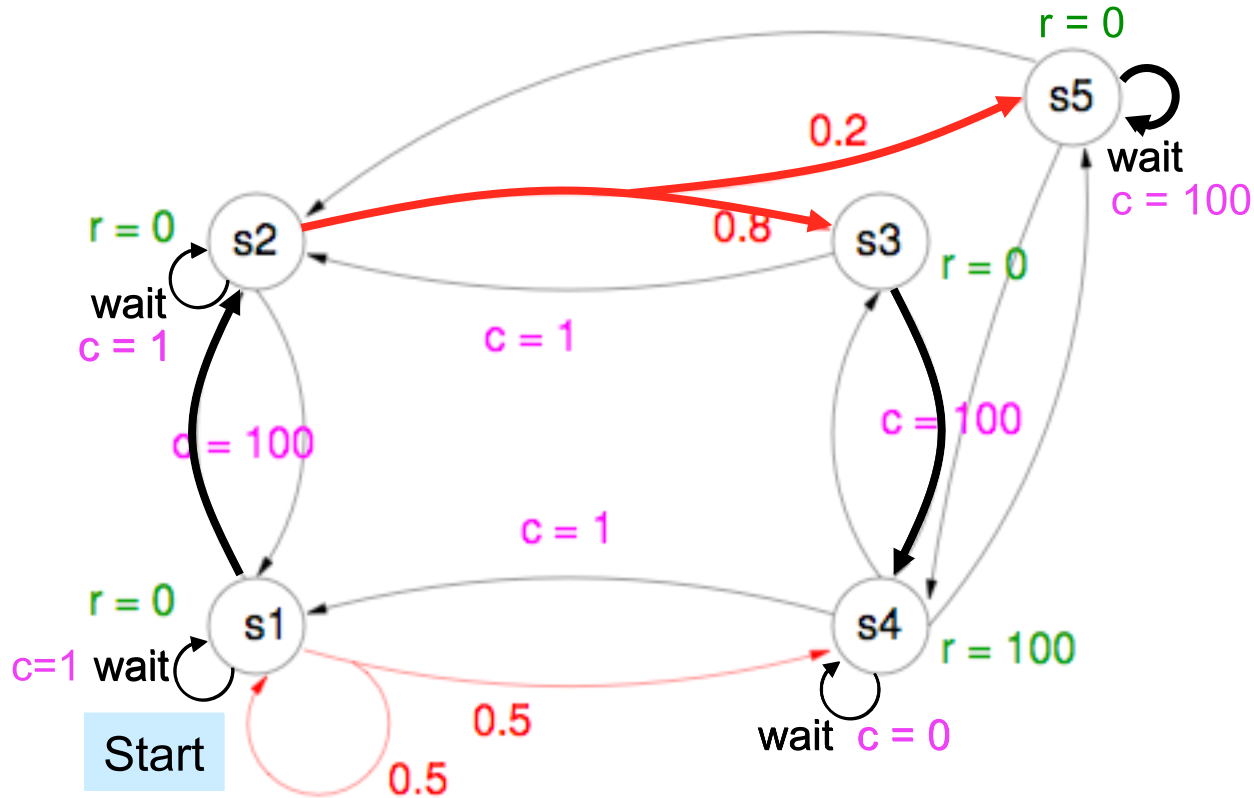


- Reinforcement $r(s)$ for each state s:
  - $-$: *cost $C(s,a)$*
  - $+$: *reward $R(s)$*
- Example:
  - $C(s,a) = 1$ for each "horizontal" action
  - $C(s,a) = 100$ for each "vertical" action
  - $C(s_1,wait) = 1$; $C(s_2,wait) = 1$; $C(s_4,wait) = 0$; $C(s_5,wait) = 100$
  - R as shown: $r(s_1)=r(s_2)=r(s_3)=r(s_5)= 0$; $r(s_4) = 100$
- Utility function: **generalization of a goal** (additive rewards)
  - If $h = \langle s_0, s_1, \ldots \rangle$, then $V_\pi(h) = \sum_{i \geq 0} \gamma^i (R(s_i) - C(s_i, \pi(s_i)))$

Example

$\pi_1 = \{$(s1, move(r1,l1,l2)),
(s2, move(r1,l2,l3)),
(s3, move(r1,l3,l4)),
(s4, wait),
(s5, wait)$\}$

$\gamma = 0.9$



$h_1 = \langle$s1, s2, s3, s4, s4, … $\rangle$

$V_{\pi 1}(h_1) = .9^0(0 - 100) + .9^1(0 - 1) + .9^2(0 - 100) + .9^3\,100 + .9^4\,100 + \ldots = 547.9$

$h_2 = \langle$s1, s2, s5, s5 … $\rangle$

$V_{\pi 1}(h_2) = .9^0(0 - 100) + .9^1(0 - 1) + .9^2(-100) + .9^3(-100) + \ldots = -910.1$

TESTINHO: qual a utilidade (ou valor) esperada de π_1 ?

Optimal Policy

- Utility of a state – defined in terms of the utility of state sequences:

$$V_\pi(s) = E\left[\sum_{t=0}^{\infty} \gamma^t r(s_t) \,\middle|\, \pi, s_0 = s\right]$$

The true utility of a state, V(s), is just $V_{\pi^*}(s)$, which allows the agent to choose the action that <u>maximizes</u> the expected utility of the ***subsequent*** state:

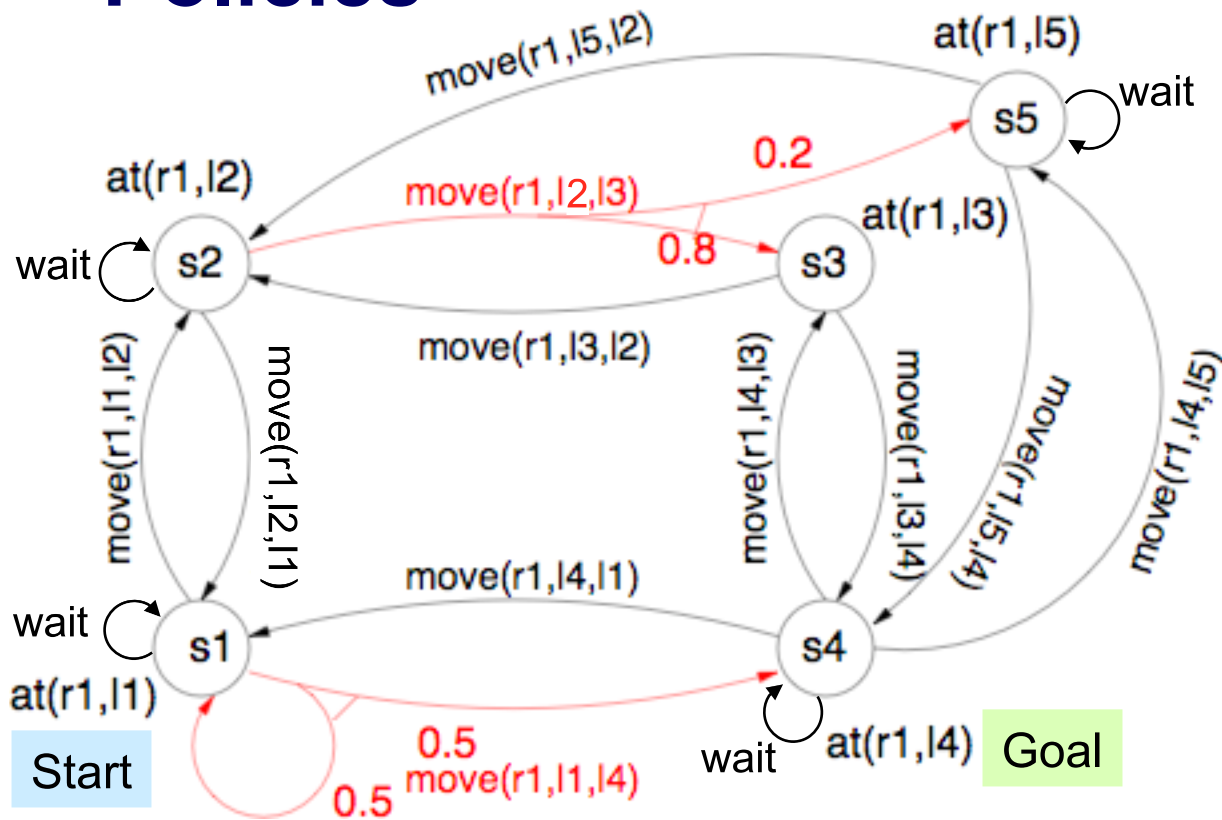
$$\boxed{\pi^*(s) = \arg\max_a \sum_{s'} p(s' \mid s, a) V^*(s')}$$

If we know V*, then it's easy to find the optimal policy.

Policies

$\pi_1 = \{$(s1, move(r1,l1,l2)),
(s2, move(r1,l2,l3)),
(s3, move(r1,l3,l4)),
(s4, wait),
(s5, wait)$\}$

$\pi_2 = \{$(s1, move(r1,l1,l2)),
(s2, move(r1,l2,l3)),
(s3, move(r1,l3,l4)),
(s4, wait),
(s5, move(r1,l5,l4))$\}$



Qual é melhor? π_1 ou π_2 ?

Computing V* Approaches

- **Value iteration**
- Policy iteration
- Linear programming

Value Iteration

1. Initialize $V_0(s) = 0$, for all s.
2. Loop until a stop criterion is met:
   - Loop for all s:

$$V^{t+1}(s) \leftarrow r(s) + \max_a \gamma \sum_{s'} p(s' \mid s, a) V^t(s')$$

This algorithm is guaranteed to converge to V*.
The influence of **r** and **p**, which we know, drives the successive Vs to get closer and closer to V*.

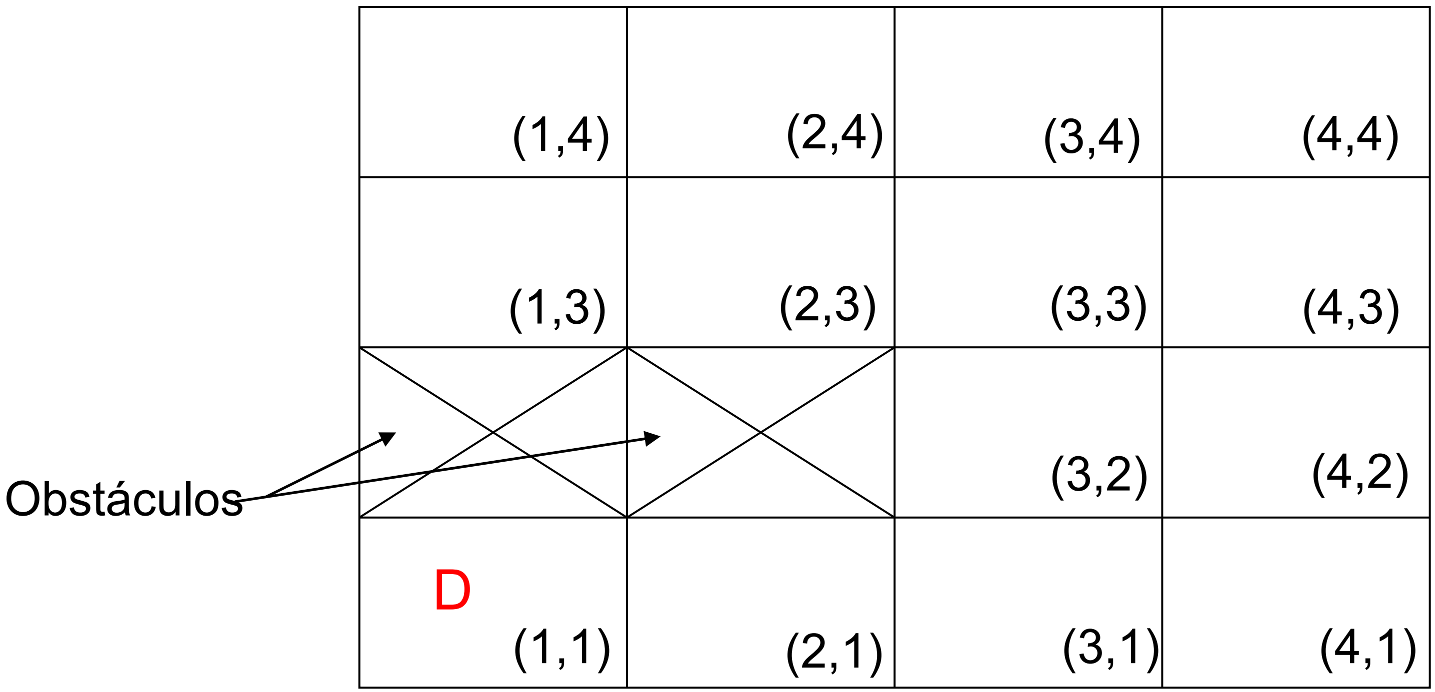
VI: Discussion

- VI computes new values in each iteration, and chooses a policy based on those values
- This algorithm converges in a polynomial number of iterations
  - But the variable in the polynomial is the number of states
  - The number of states is usually **huge**
- Need to examine the **entire state space** in each iteration
  - Thus, this algorithm takes huge amounts of time and space

TAREFA

- Ambiente discreto 4x4, com obstáculos.
- Agente deve alcançar posição destino D a partir de **qualquer lugar** do ambiente.
- D é um estado absorvente (ao atingir D, o episódio termina): $V^*(D) = 0$
- Ações que o agente pode realizar: N, S, L, O
- Penalidade por executar uma **ação** (qualquer) $= -1$
  - Melhor política => caminho mais curto
- Considerar $\gamma = 1$ e MDP determinístico ($p=1$)

Ambiente



Ações = {N, S, L, O}

Tarefa: algoritmo de iteração de valor para MDP determinístico

- Cálculo iterativo da função valor ótima.

$$V(s) \leftarrow r_{s,a} + \max_a (V(s'))$$

Repetir até V(s) estabilizar.

Sendo:

s – estado atual, s' – próximo estado,

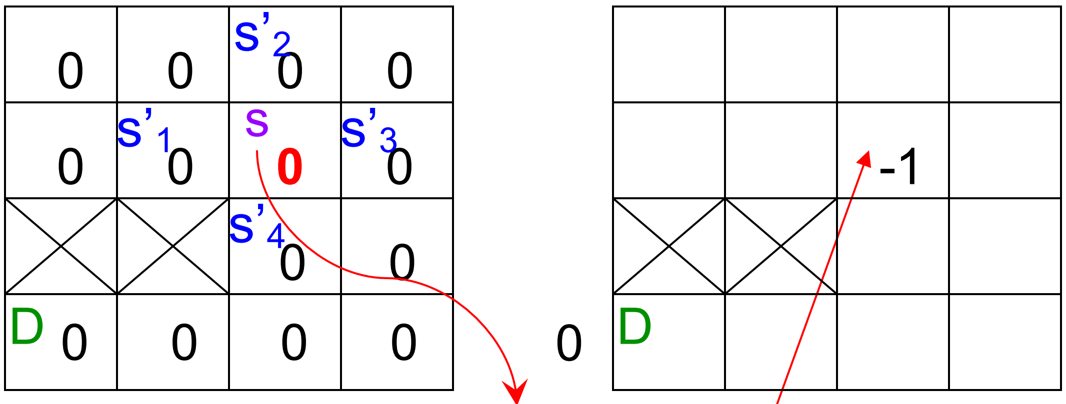
$r_{s,a}$ – reforço recebido por executar a em s

V(.) – valor do estado

Exemplo de cálculo de V(s)

Início

Iteração 1 (quando calculous para todos os estados)



$$V(s) = \max_a \big((r(s, O) + V(s'_1)),$$
$$(r(s, N) + V(s'_2)),$$
$$(r(s, L) + V(s'_3)),$$
$$(r(s, S) + V(s'_4)) \big)$$
$$= \max_a ((-1+0), (-1+0), (-1+0), (-1+0))$$
$$= -1$$

. . .

Tarefa

- Entrega: Mostrar o valor no espaço de estados (grade com o valor em cada célula) após CADA ITERAÇÃO, até a convergência do algoritmo VI
- Responder:
  - Qual estado tem valor **mínimo**? Qual o valor deste estado?
  - Qual estado tem valor **máximo**? Qual o valor deste estado?
  - Mostrar na grade qual é a **política ótima** em cada célula.