Procesos de Decisión de Markov

PDM

Procesos de decisión de Markov

- Aprendizaje por refuerzo
- Procesos de decisión de Markov:
  - Definición
  - Solución de un proceso de decisión de Markov
  - Ejemplo
- Algoritmos para resolver un PDM
  - Utilidad de una secuencia de acciones.
  - Iteración de valores
  - Iteración de políticas

Bibliografía de MDP y RL

UNIVERSITAT DE BARCELONA

- Consultar libro electrónico de Sutton and Barto

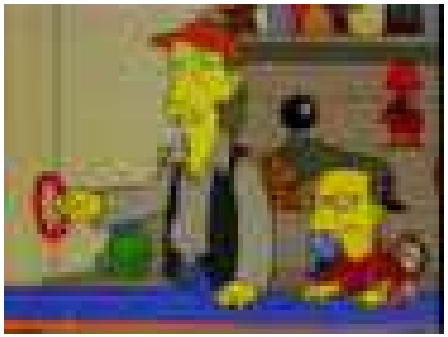
http://webdocs.cs.ualberta.ca/~sutton/book/ebook

  - capítulos 3, 4 y 6 (secciones 6.1, 6.2, 6.5)
- Las transparencias de MDP y RL también corresponden al libro de Russell and Norvig:
  - capítulos 17 y 21

Aprendizaje por refuerzo

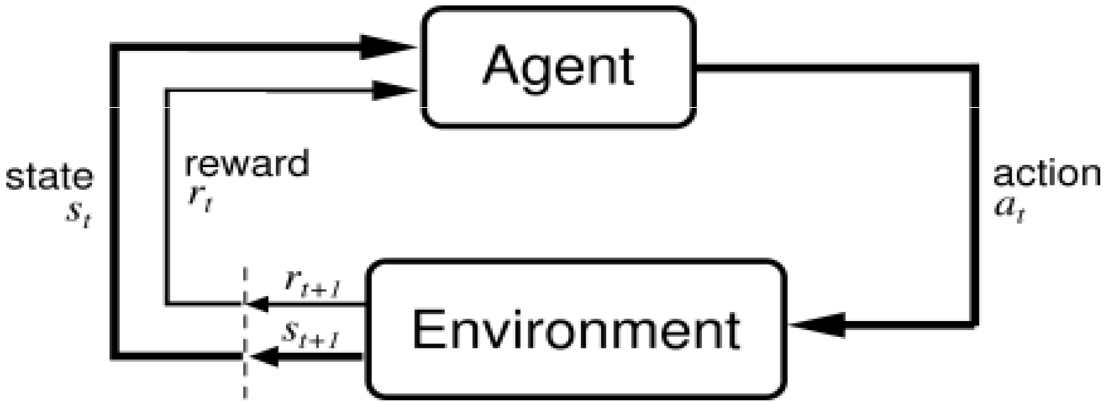
UNIVERSITAT DE BARCELONA

- Juguemos:
  - Véis una luz encendida, dos botones (uno a la derecha y otro a la izquierda) y un feriante con mala cara que os pide un euro por cada vez que presionéis un botón y os dice que juguéis. No parecéis tener otra opción jugar a su juego... así que le dais el primer euro y os pregunta que botón queréis apretar.



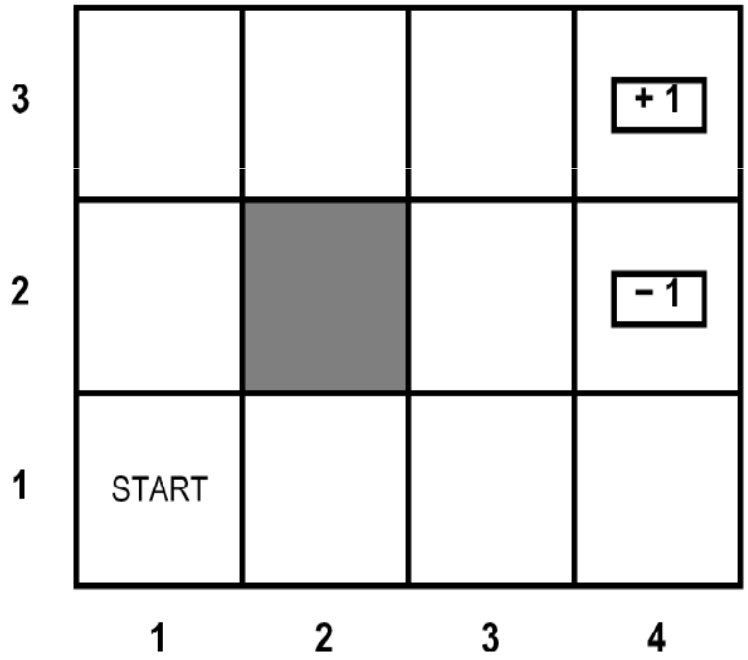
Aprendizaje por refuerzo

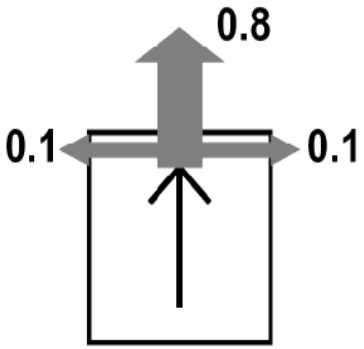
- Idea fundamental:
  - Recibimos información en forma de recompensa.
  - La utilidad del agente viene dada por la función de recompensa.
  - El agente debe aprender a actuar para maximizar la recompensa esperada.



Mundo rejilla

- El agente vive en una rejilla
- Los bloques oscuros impiden el paso del agente
- Las acciones del agente no siempre tienen el mismo resultado. Si quiere ir hacia el norte:
  - 80% de las veces va al norte
  - 10% al este
  - 10% al oeste
- Puede recibir una pequeña recompensa por ”sobrevivir”
- La mayor recompensa llega al final del juego
- Objetivo: maximizar la suma de recompensas.





Procesos de decisión de Markov

Un MDP se define por:

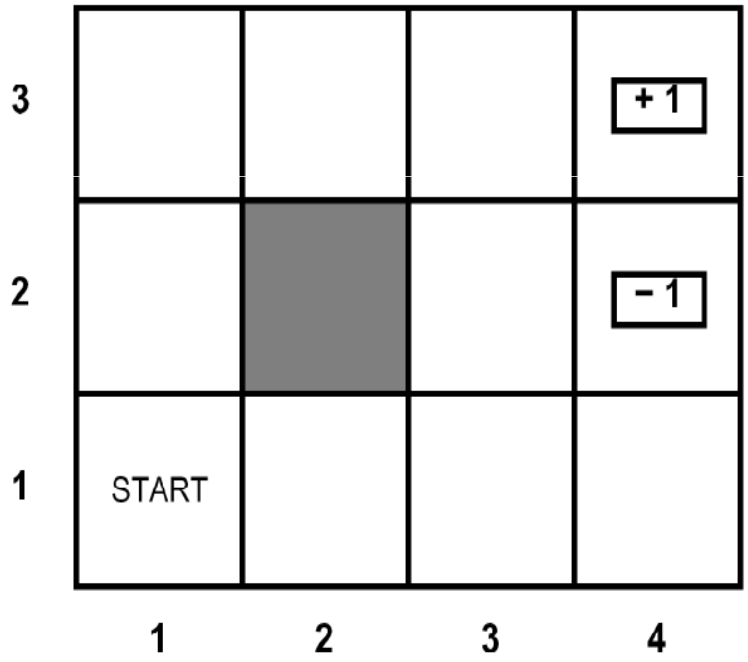
- Un conjunto de estados S.
- Un conjunto de acciones A.
- Una función de transición

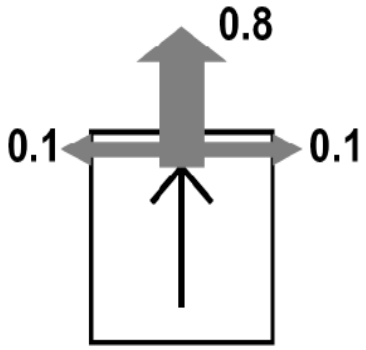
  T:A x S x S → R

  – T(a,s,s') = Probabilidad de ir al estado s' cuando estamos en el estado s y usamos la acción a.

  – T(a,s,s') = P(s'|s,a)
- Función de recompensa R(s,a,s')

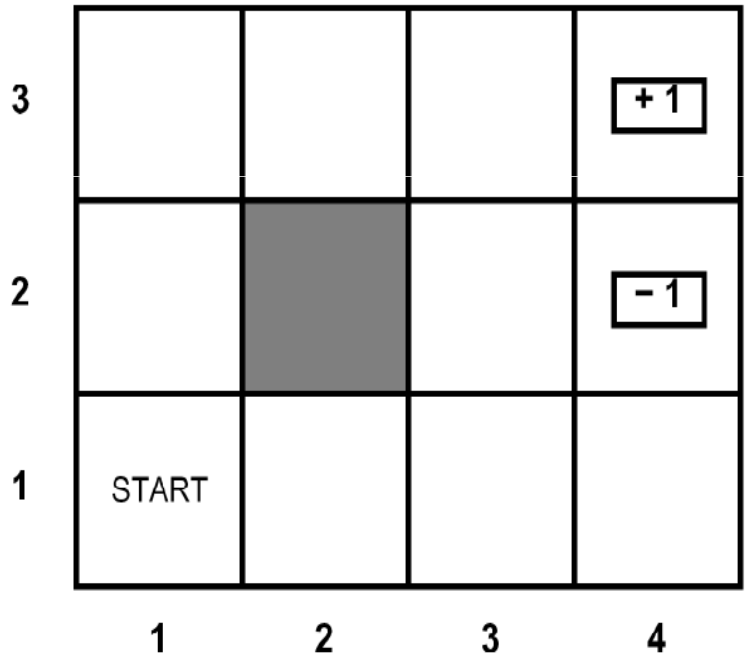
  – En ocasiones R(s)
- Un estado inicial
- Posiblemente un estado terminal

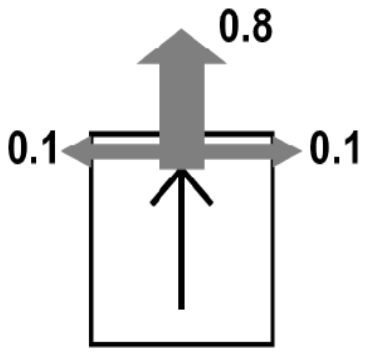




Procesos de decisión de Markov

- Los procesos de decisión de Markov son una familia de problemas de búsqueda no deterministas.
- El aprendizaje por refuerzo es un PDM del que desconocemos la función de recompensa o la función de transición.





Procesos de decisión de ¿Markov?

- Andrei Markov (1856-1922)
- La hipótesis de Markov:
  - El estado actual resume toda la información relevante del pasado.
- En el caso de los procesos de decisión de Markov, esto quiere decir que:

$$P(S_{t-1}|S_t, a_t, S_{t-1}, a_{t-1}, \ldots, S_0, a_0) = P(S_{t-1}|S_t, a_t)$$

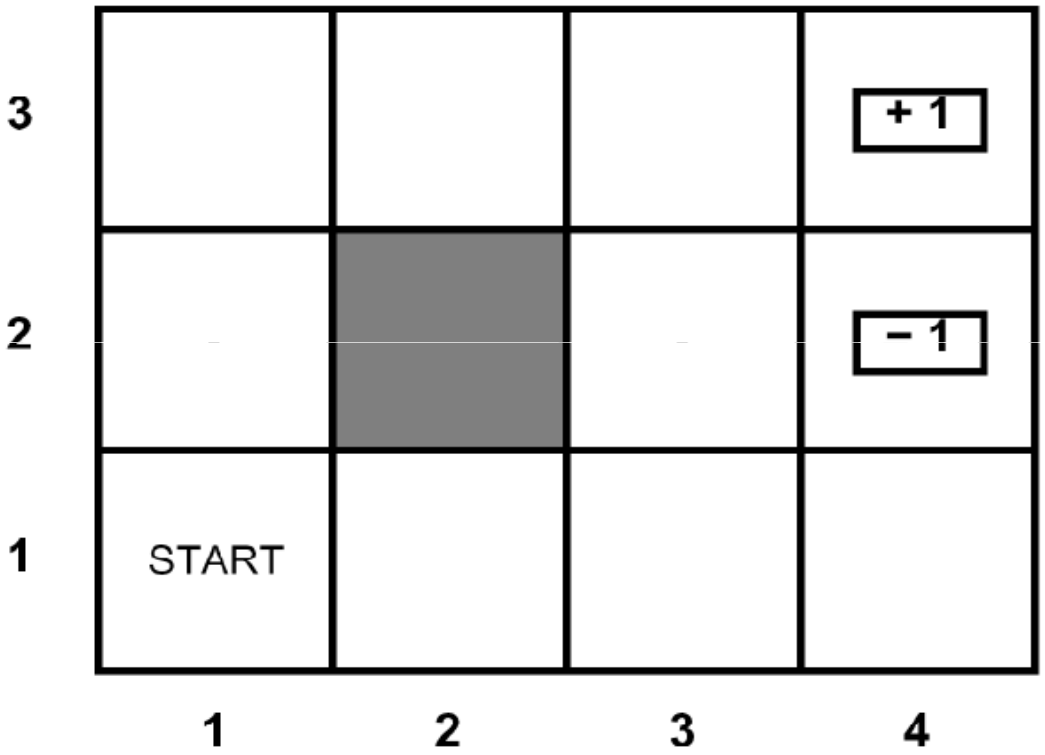
¿Qué es resolver un PDM?

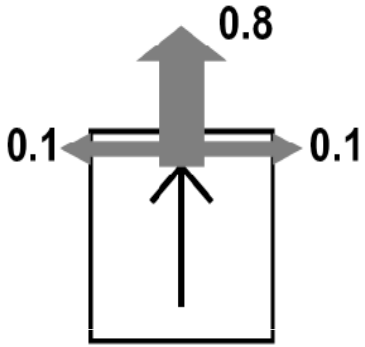
UNIVERSITAT DE BARCELONA

- En problemas de búsqueda deterministas en que interviene un único agente, queremos un plan óptimo, una secuencia de acciones desde el inicio hasta un objetivo.
- En un PDM queremos una política óptima $\pi^*:S \rightarrow A$
  - Una política asigna una acción a cada estado $\pi(s)=a$.
  - Una política óptima maximiza la utilidad esperada si se sigue.
  - El resultado es un agente reflejo.

Mundo rejilla

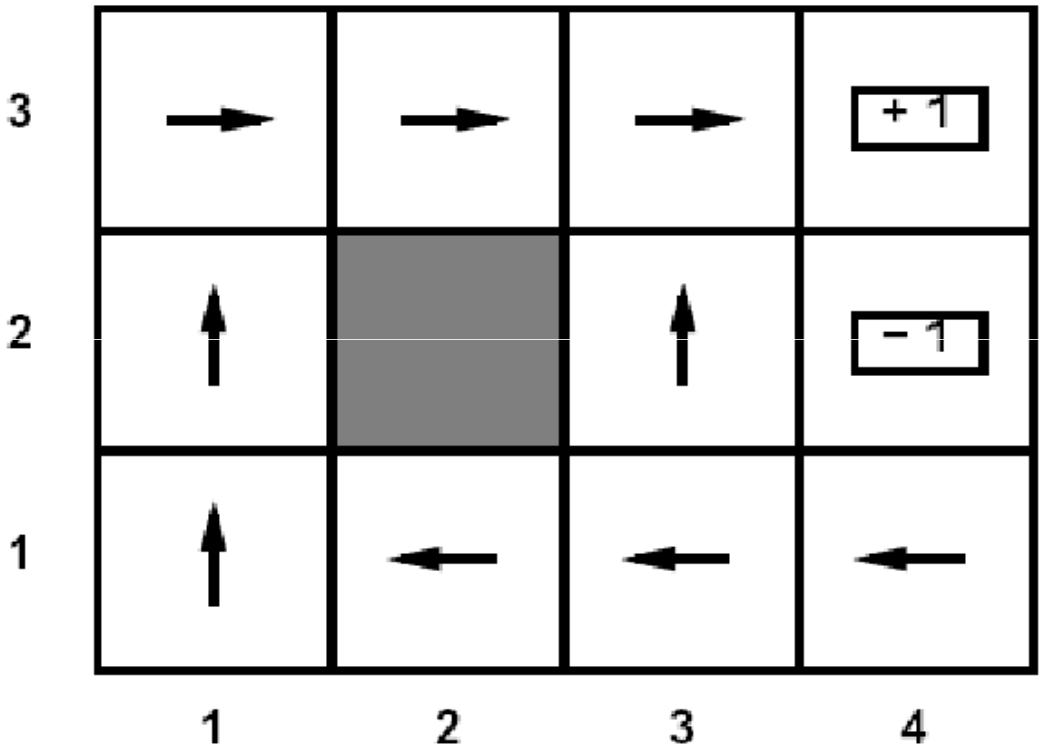
- Política óptima para el mundo rejilla cuando R(s)=-0.04 para todos los estados no terminales?

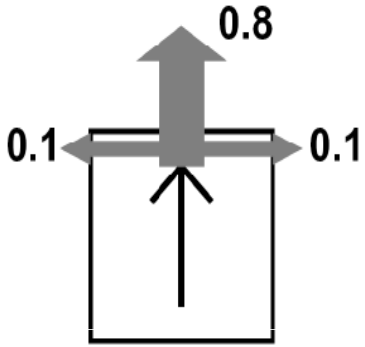




Mundo rejilla

- Política óptima para el mundo rejilla cuando R(s)=-0.04 para todos los estados no terminales

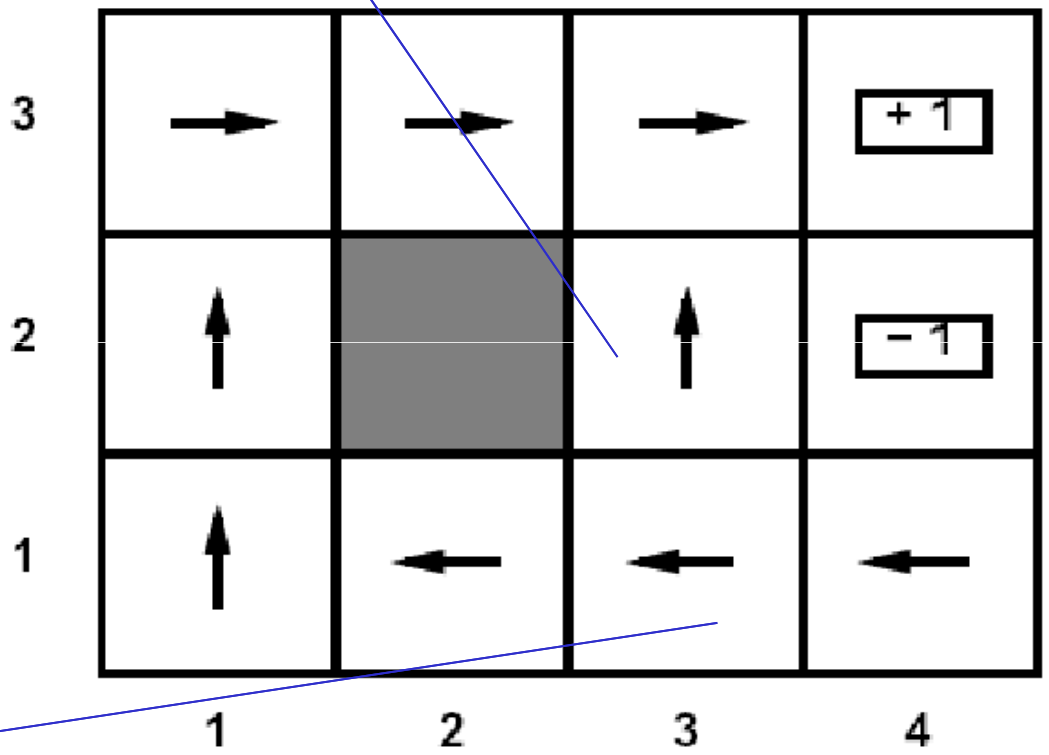


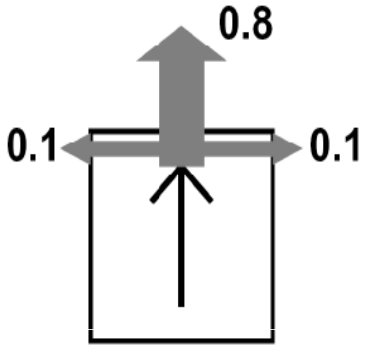


Mundo rejilla

- Política óptima para el mundo rejilla cuando R(s)=-0.04 para todos los estados no terminales

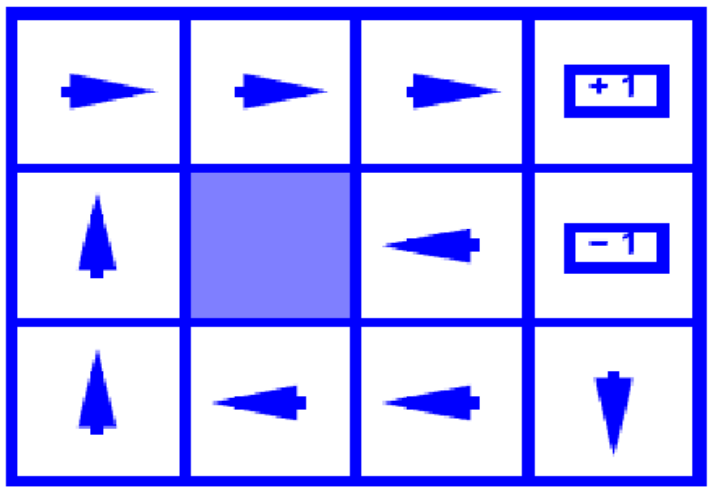
En el mejor de los casos -0.04+1=0,96
Tenemos posibilidad de que sea -1



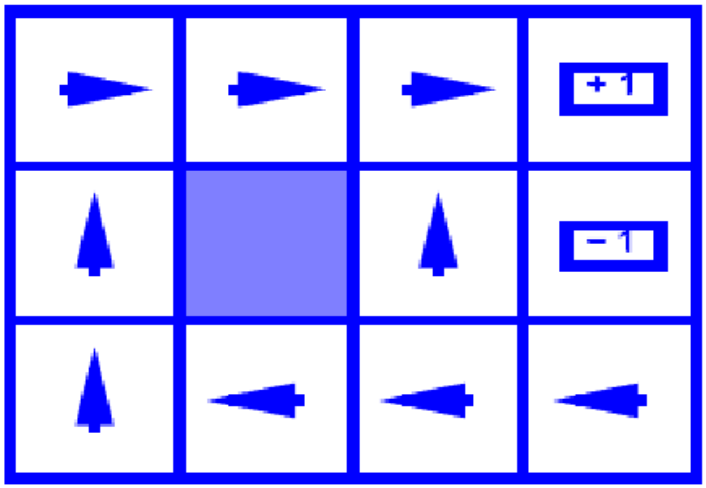


En el mejor de los casos -0.04*6+1=0,76
Si fuéramos hacia el N seria -0.04*2 + 1= 0.92 pero nos arriesgamos a tener -0.04-1= -1.04

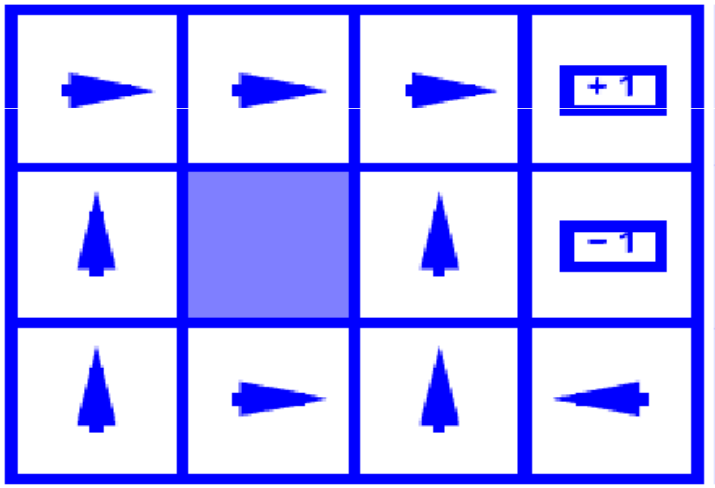
¿Qué pasa con otros valores de R(s)? (para estados no terminales)



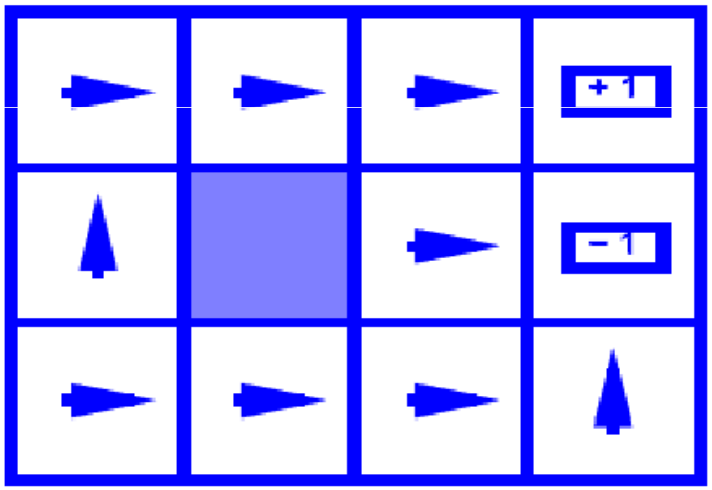
R(s) = -0.01



R(s) = -0.03

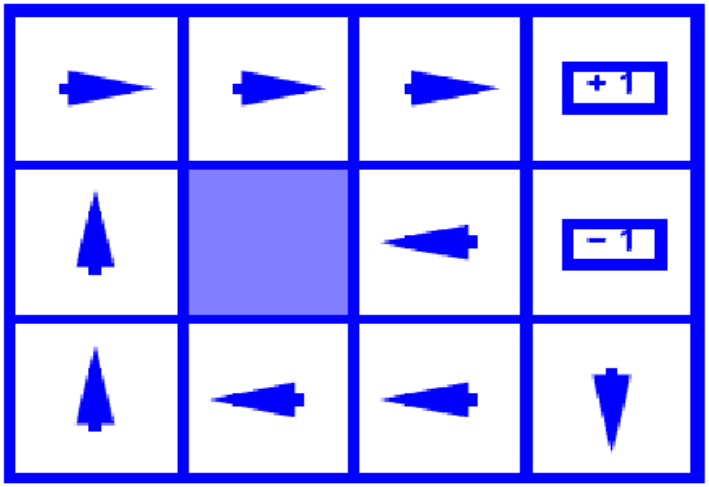


R(s) = -0.4

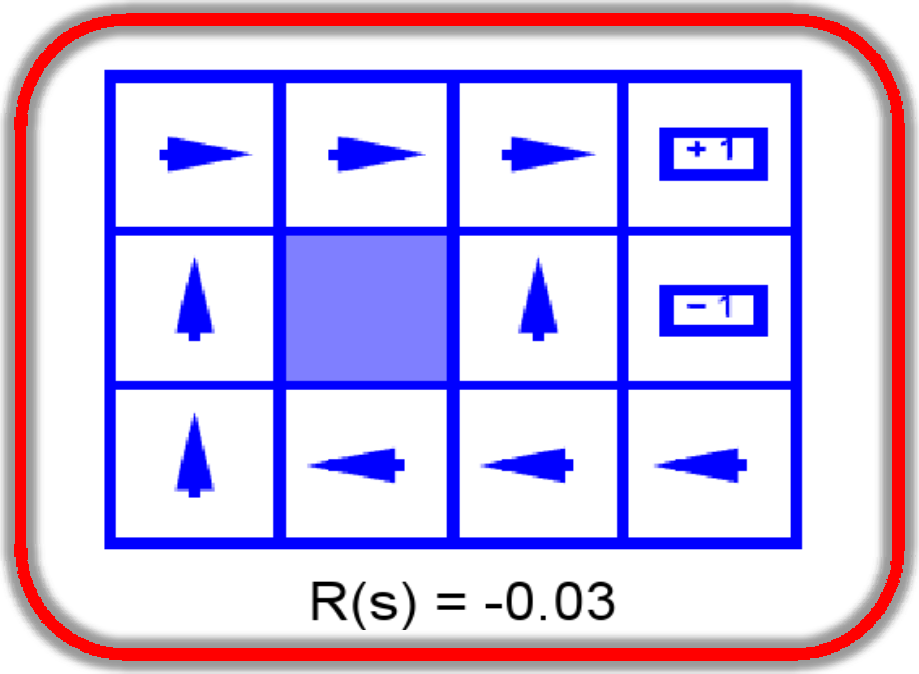


R(s) = -2.0

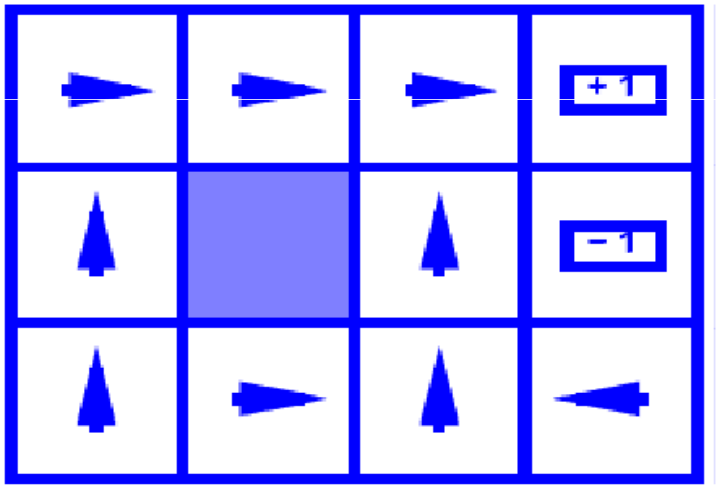
¿Qué pasa con otros valores de R(s)? (para estados no terminales)



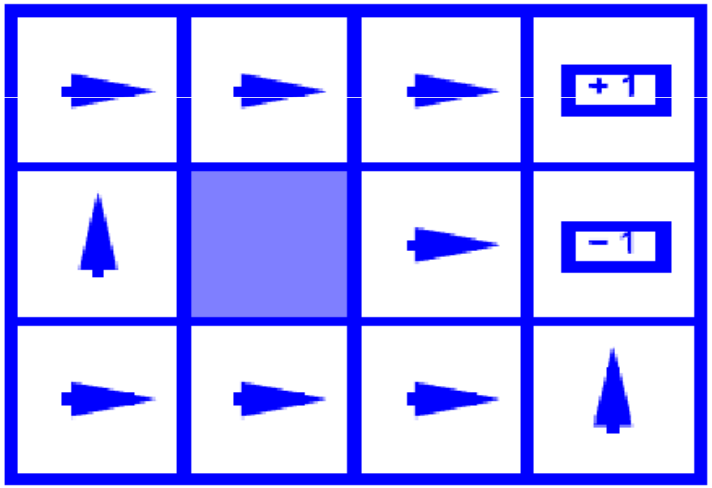
R(s) = -0.01



R(s) = -0.03

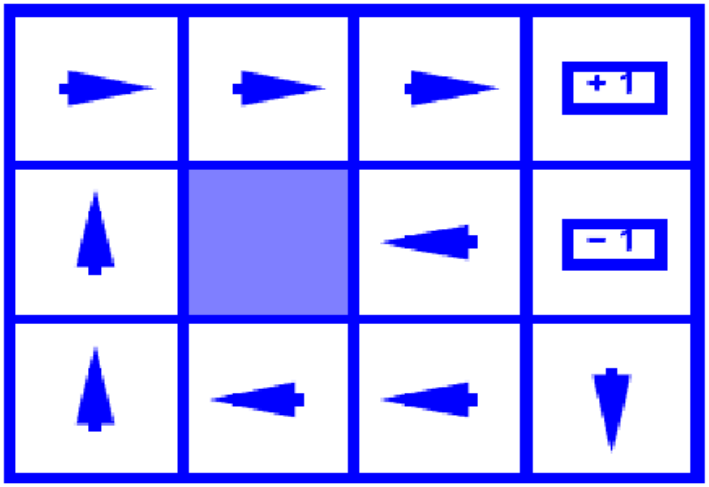


R(s) = -0.4

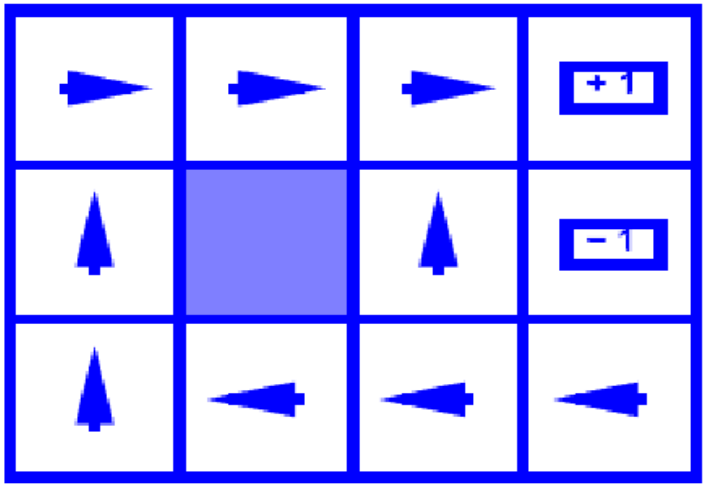


R(s) = -2.0

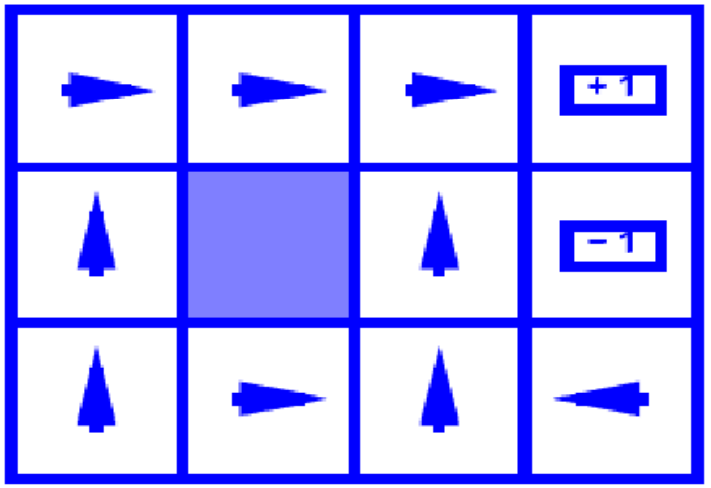
¿Qué pasa con otros valores de R(s)? (para estados no terminales)



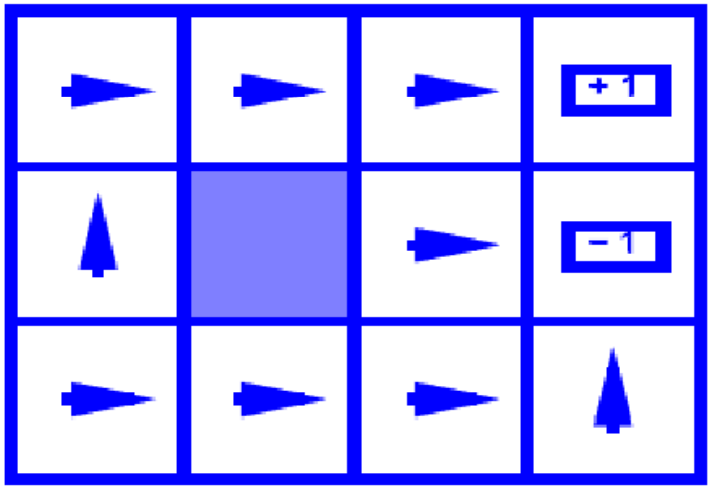
R(s) = -0.01



R(s) = -0.03

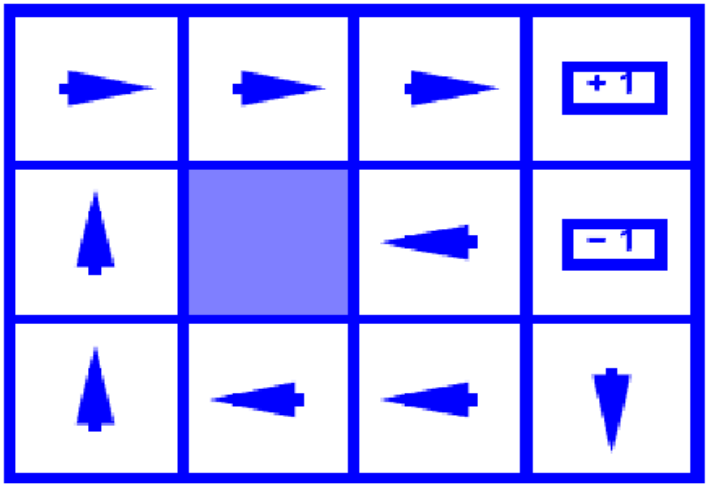


R(s) = -0.4

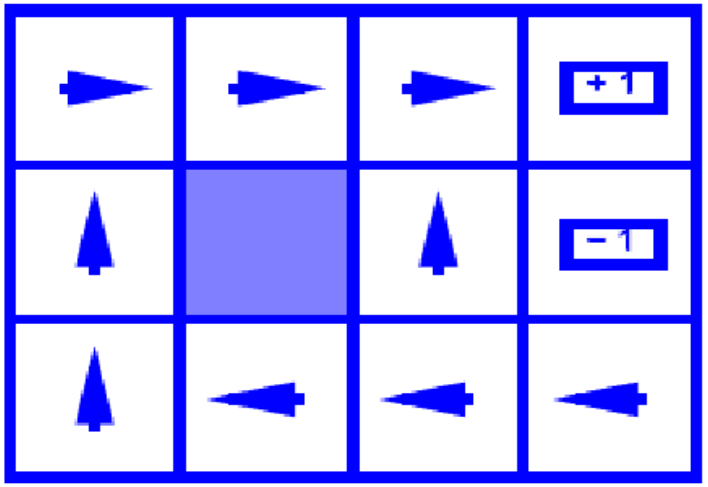


R(s) = -2.0

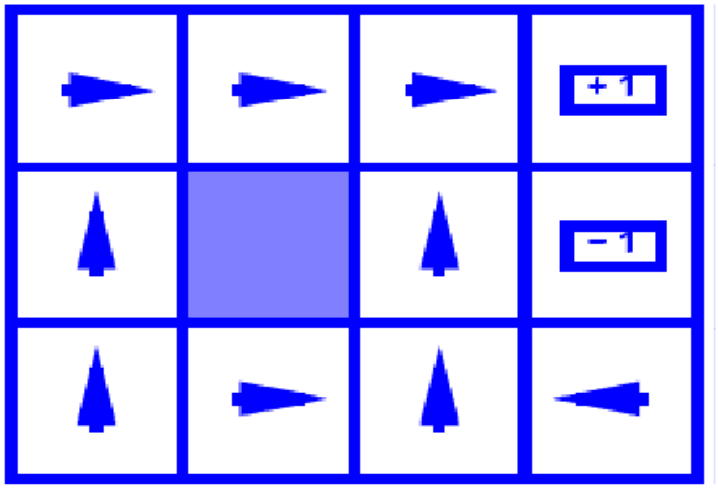
¿Qué pasa con otros valores de R(s)? (para estados no terminales)



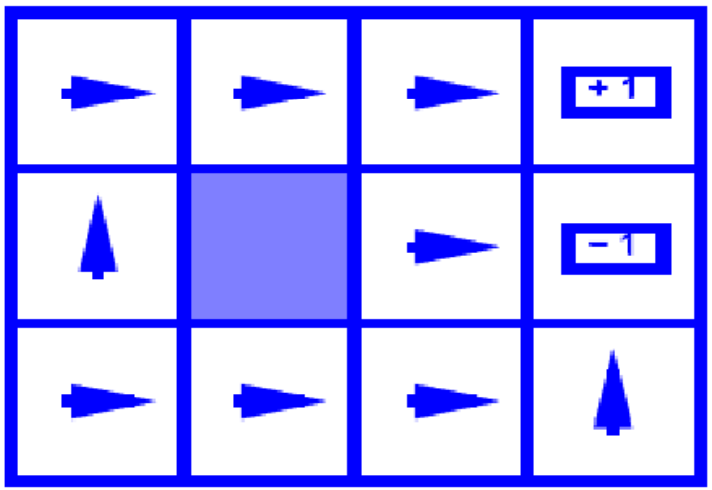
R(s) = -0.01



R(s) = -0.03

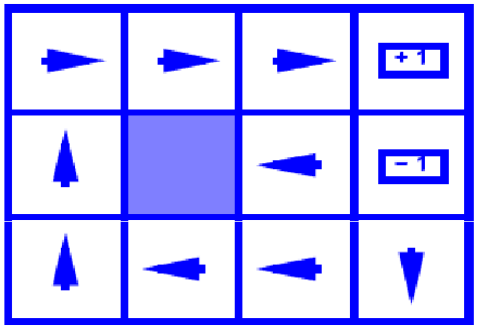


R(s) = -0.4

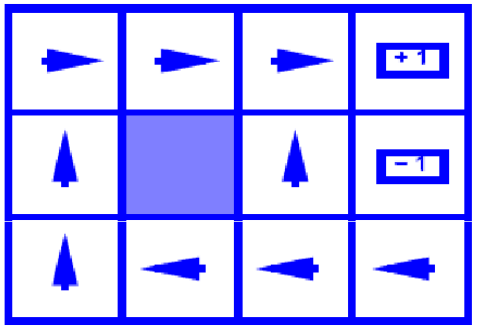


R(s) = -2.0

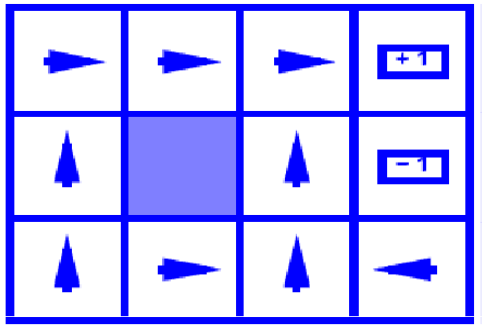
¿Qué pasa con otros valores de R(s)? (para estados no terminales)



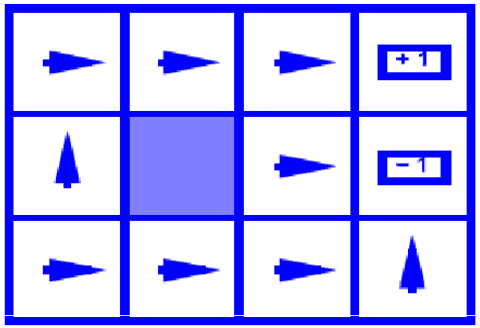
R(s) = -0.01



R(s) = -0.03



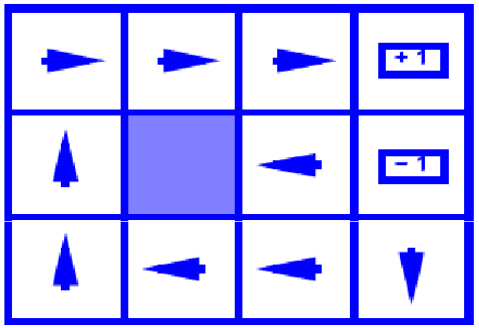
R(s) = -0.4



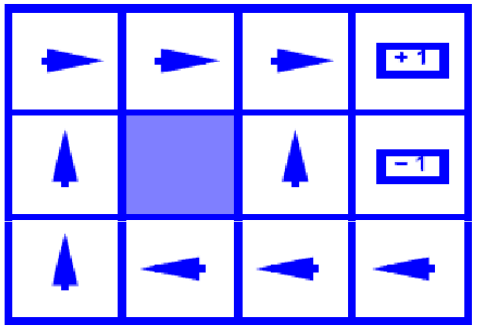
R(s) = -2.0

Y si R(s)>0 ?
(para estados no terminales)

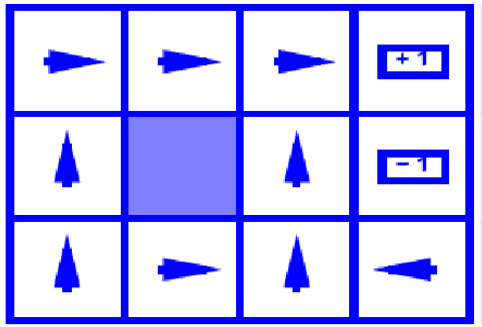
¿Qué pasa con otros valores de R(s)? (para estados no terminales)



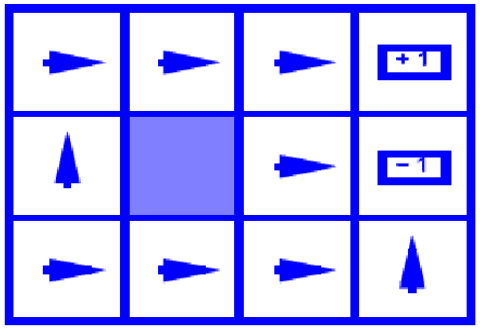
R(s) = -0.01



R(s) = -0.03

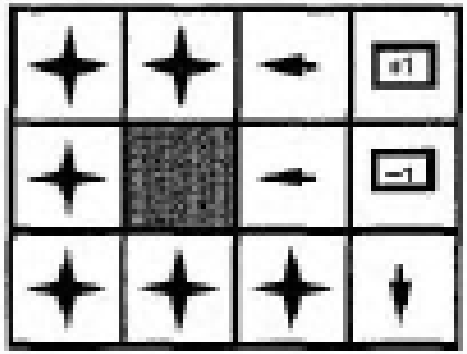


R(s) = -0.4



R(s) = -2.0

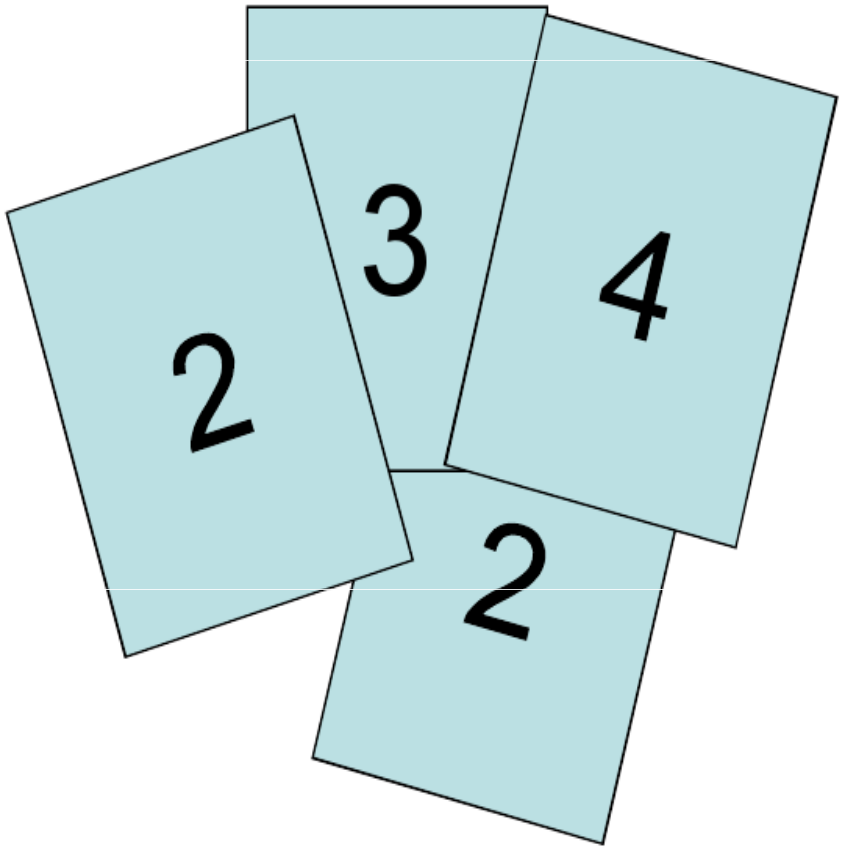
Y si R(s)>0 ?



R(s) > 0

Superior/Inferior

- Tres tipos de cartas: 2,3,4
- Mazo infinito
- Comenzamos con un 3
- Después de cada carta tenemos que decir: Superior o Inferior
- Se saca una nueva carta:
  - Si hemos acertado (o si sale la misma carta) ganamos tantos puntos como el valor de la carta.
  - El juego termina si hemos fallado.



Superior/Inferior

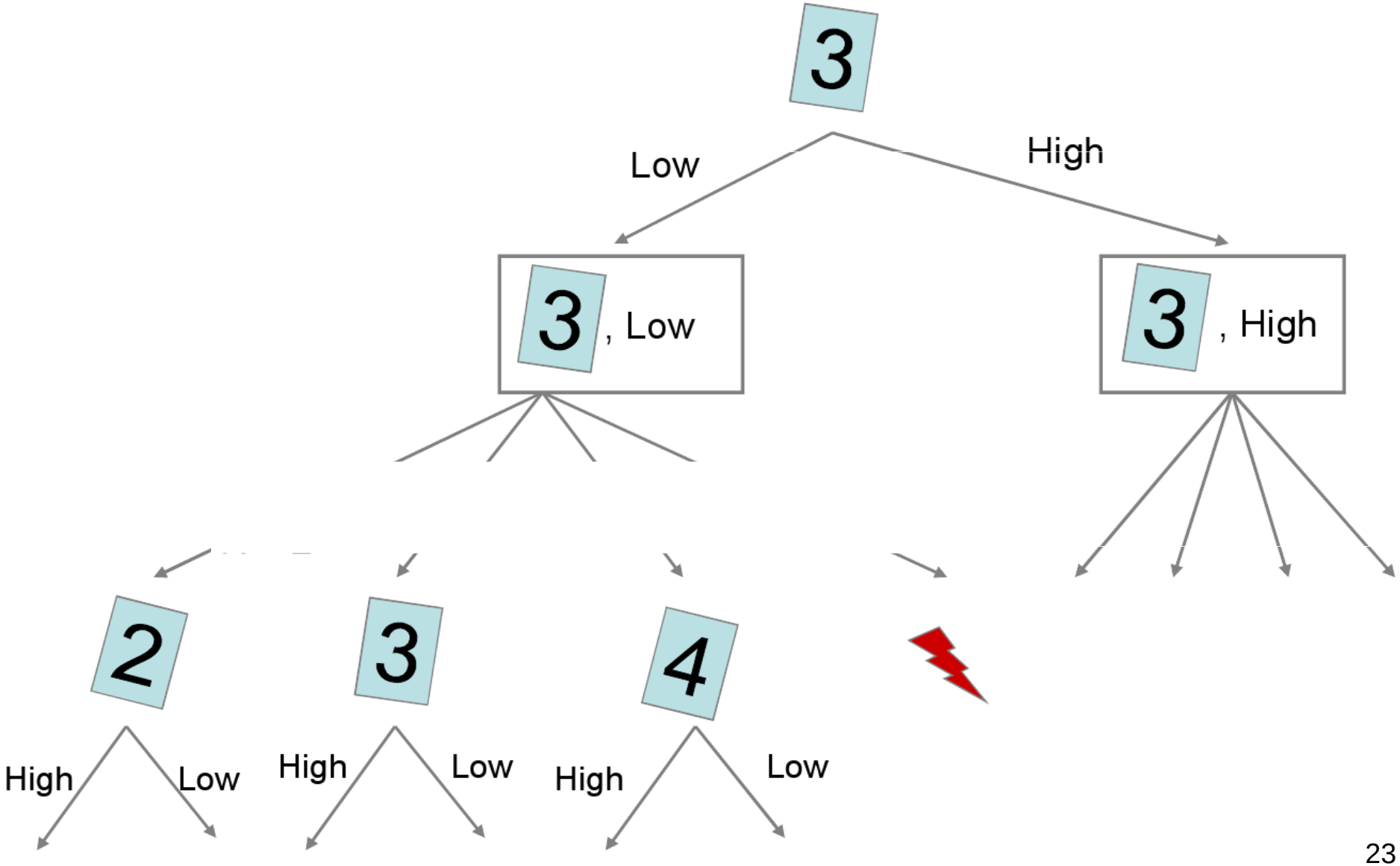
- ¿Podemos usar expectimax?
- No, porque:
  - En expectimax las recompensas llegan únicamente al final del juego
  - En expectimax el juego termina en algún momento.
- Formalicemos Superior/Inferior como un PDM...

Superior/Inferior como PDM

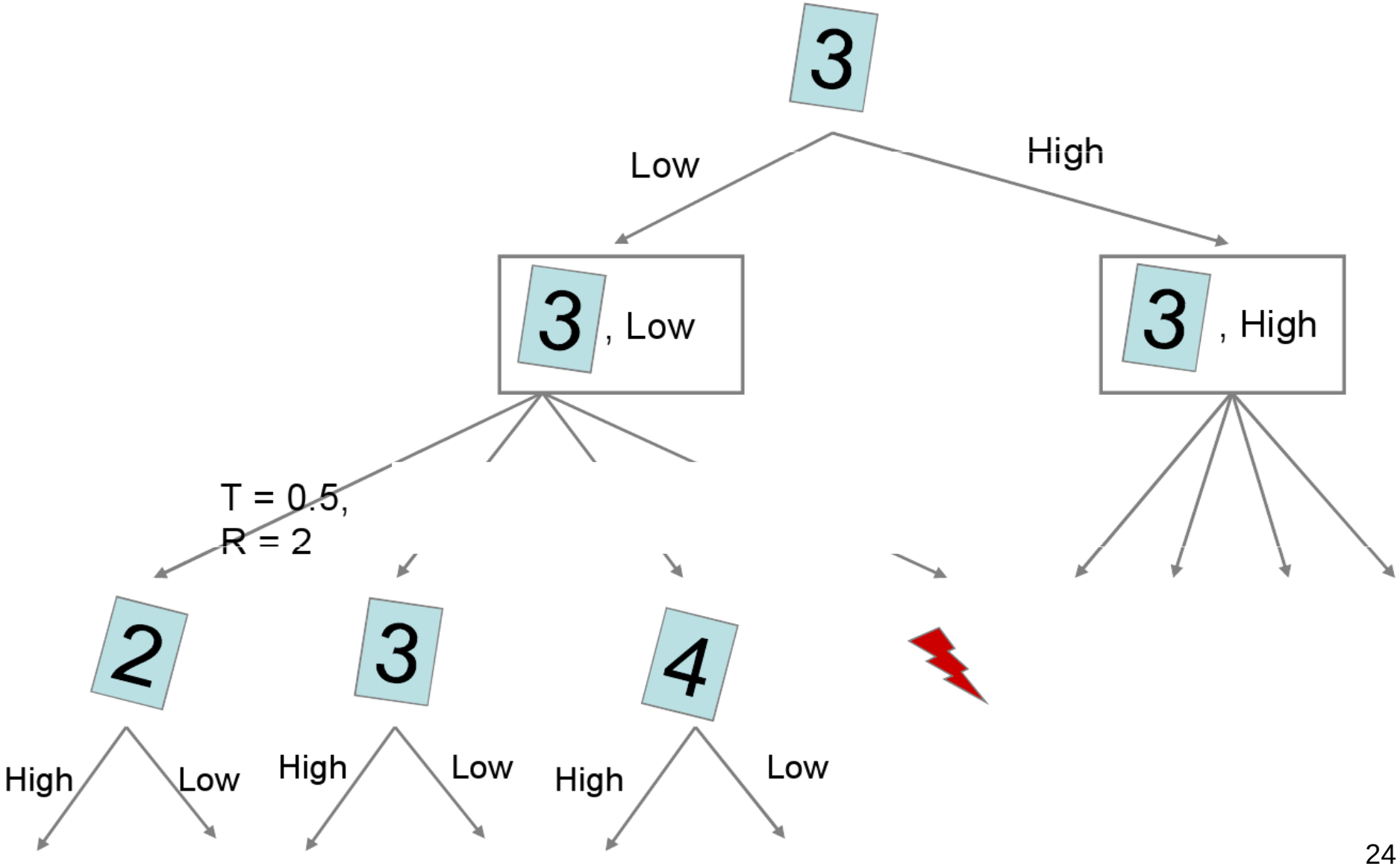
UNIVERSITAT DE BARCELONA

- Estados: 2,3,4,fin
- Acciones: Superior, Inferior
- Modelo: T(s,a,s')
  - T(s=4,a=Inferior,s'=4)=0.25
  - T(s=4,a=Inferior,s'=3)=0.25
  - T(s=4,a=Inferior,s'=2)=0.5
  - T(s=4,a=Inferior,s'=fin)=0
  - ...
  - T(s=4,a=Superior,s'=4)=0.25
  - T(s=4,a=Superior,s'=3)=0
  - T(s=4,a=Superior,s'=2)=0
  - T(s=4,a=Superior,s'=fin)=0.75
- Retorno: R(s,a,s') = valor(s') si s'≠fin,
  = 0 si s'=fin
- Estado inicial: 3

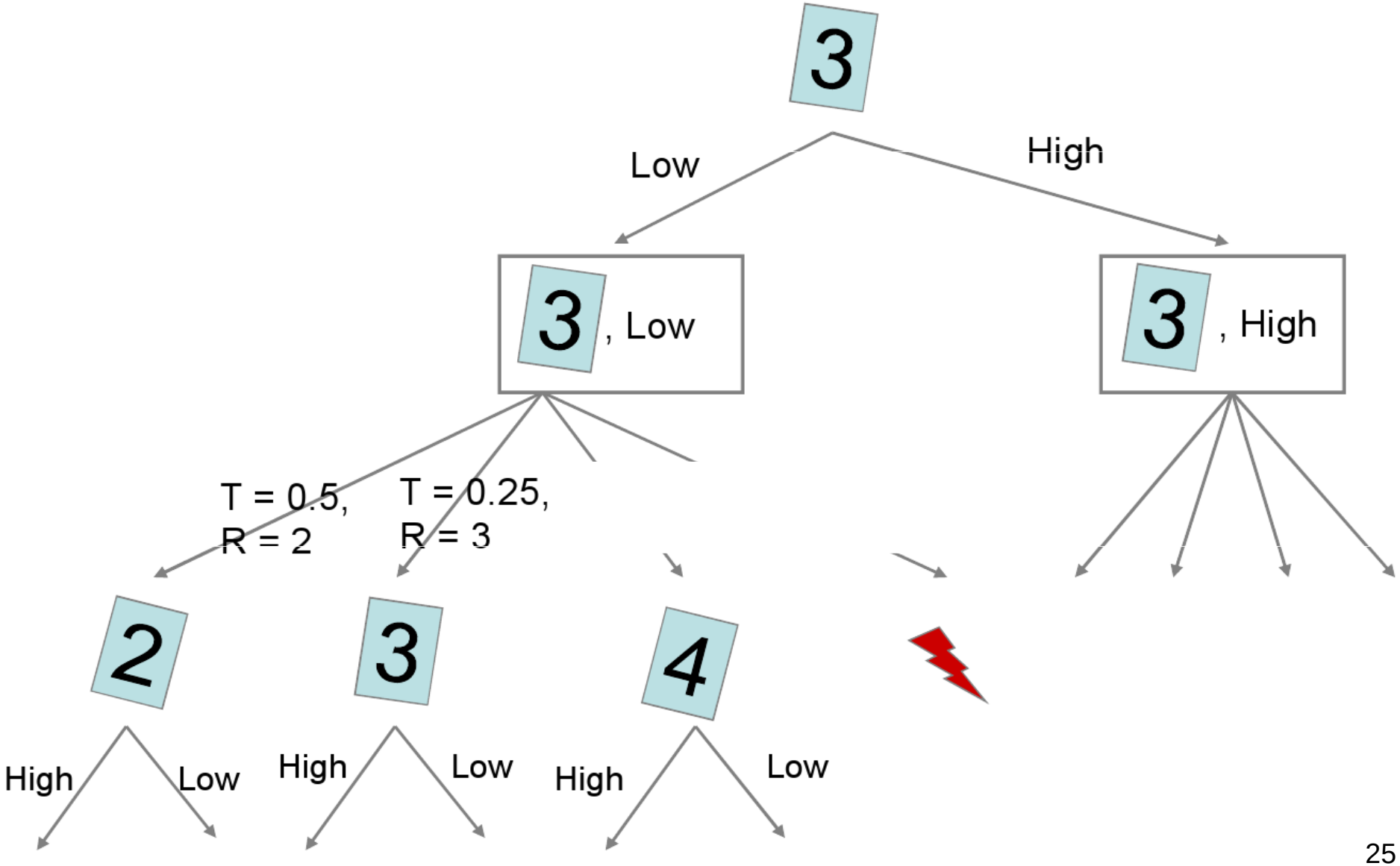
Superior/Inferior



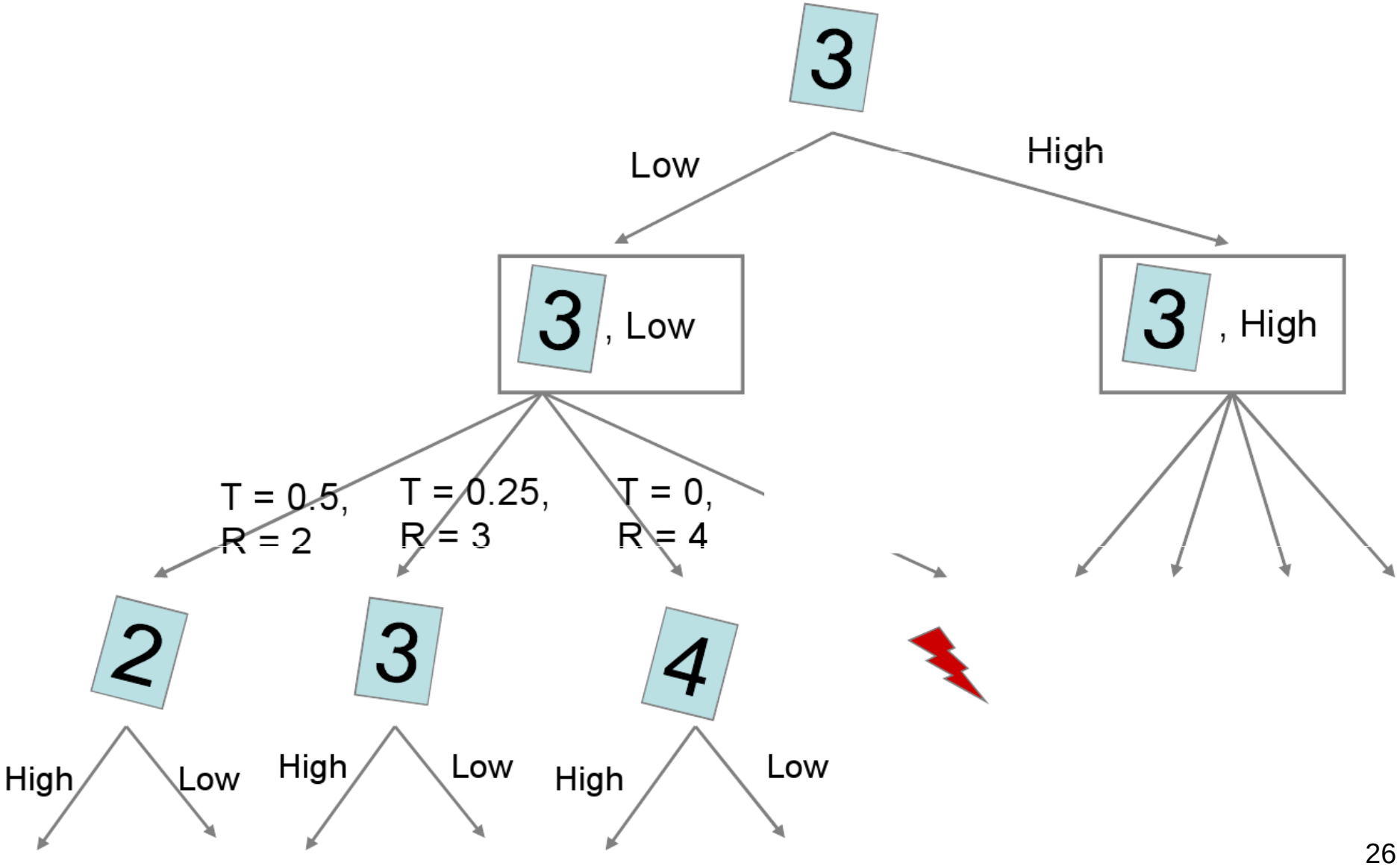
Superior/Inferior



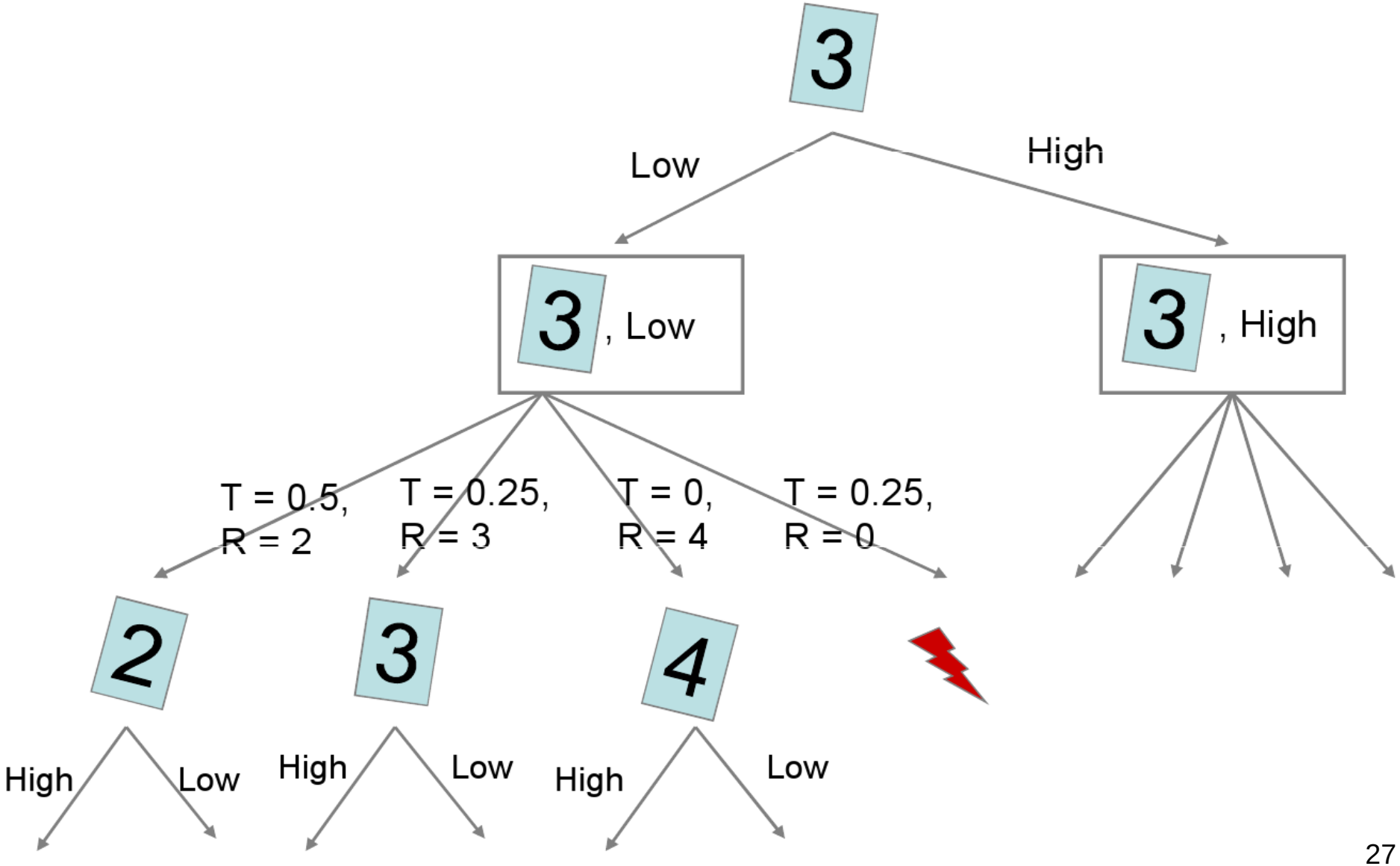
Superior/Inferior



Superior/Inferior

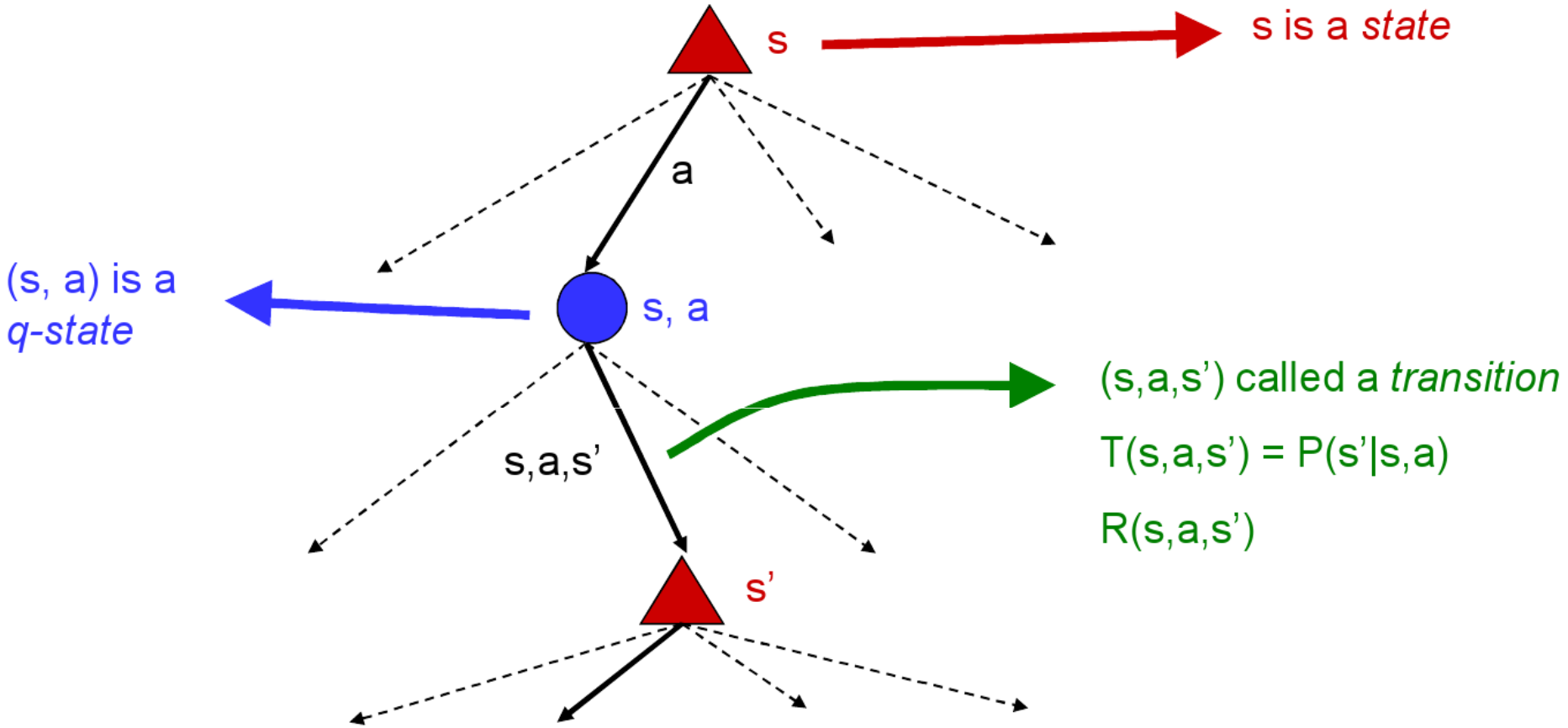


Superior/Inferior



Árboles de búsqueda PDM

- Cada estado del PDM genera un árbol de búsqueda expectimax.



Nos interesa la utilidad de hacer una acción a en un estado s

Utilidad de una secuencia de acciones

- Para formalizar la optimalidad de una política, necesitamos entender la optimalidad de una secuencia de acciones.
- Típicamente consideramos preferencias estacionarias sobre secuencias de acciones.

$$[r, r_0, r_1, r_2, \ldots] \succ [r, r'_0, r'_1, r'_2, \ldots]$$
$$\Leftrightarrow$$
$$[r_0, r_1, r_2, \ldots] \succ [r'_0, r'_1, r'_2, \ldots]$$

Utilidad de una secuencia de acciones

- Teorema: Si las preferencias son estacionarias únicamente existen dos maneras de definir la utilidad de una secuencia de acciones:
  - Utilidad aditiva:

$$U([r_0, r_1, r_2, \ldots]) = r_0 + r_1 + r_2 + \cdots$$

  - Utilidad aditiva con descuento:

$$U([r_0, r_1, r_2, \ldots]) = r_0 + \gamma r_1 + \gamma^2 r_2 \cdots$$

¿Utilidad infinita?

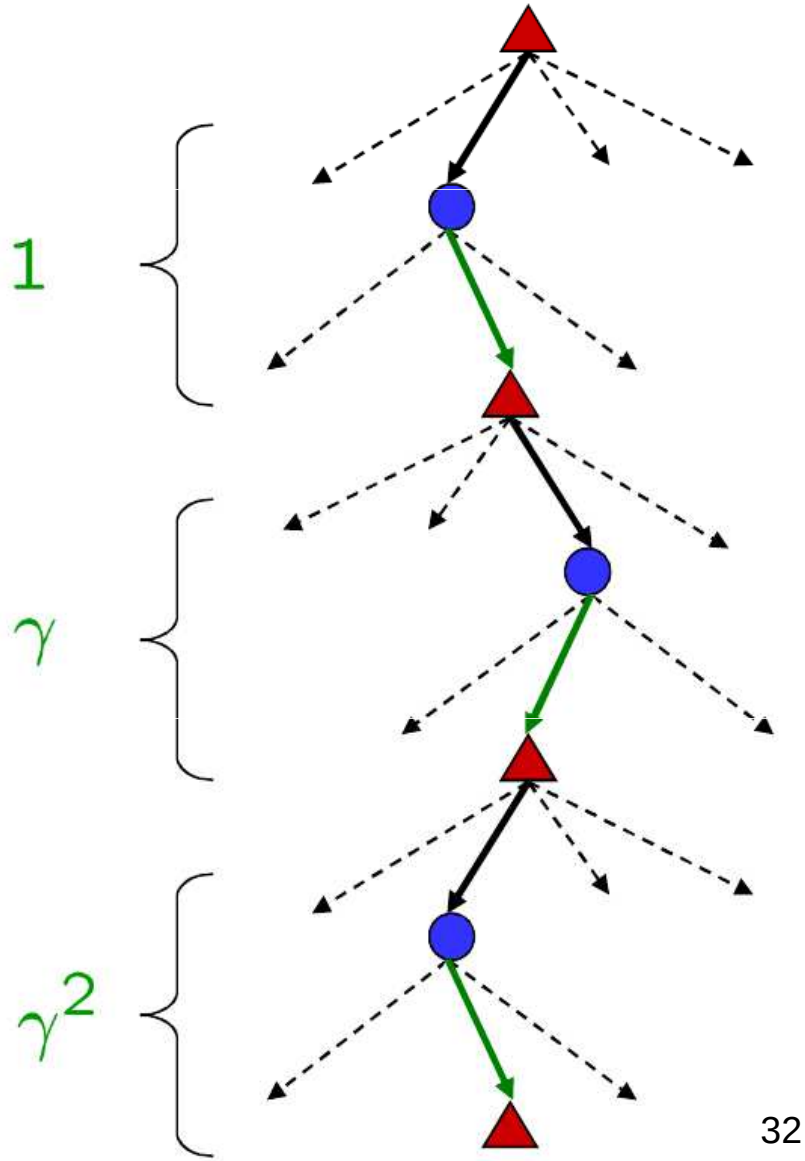
- Problema: Las secuencias de acciones infinitas pueden tener una utilidad infinita.
- Soluciones:
  - Horizonte finito. Establecemos que sólo se jugará al juego durante T unidades de tiempo. Da lugar a políticas no estacionarias (π* depende del tiempo que queda)
  - Estado absorbente: Garantizamos que para cualquier política se alcance siempre un nodo terminal.
  - Descontar: Dado un $0 < \gamma < 1$

$$U([r_0, \ldots r_\infty]) = \sum_{t=0}^{\infty} \gamma^t r_t \leq R_{\max}/(1-\gamma)$$

    - Un γ más pequeño indica que estamos mucho más interesados en los estados más inmediatos que en los que llegarán más tarde.

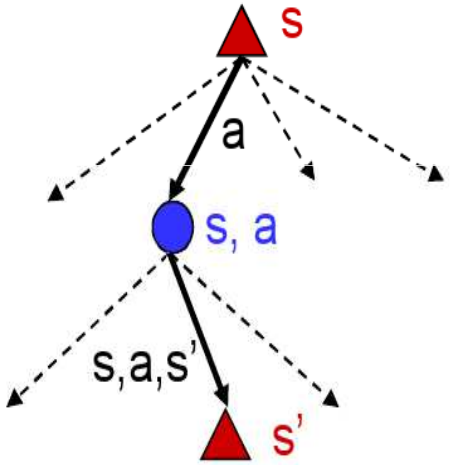
Descontar

- Normalmente se descuenta por $\gamma < 1$ cada paso de tiempo
- Las recompensas más cercanas en el tiempo tienen una utilidad mayor que las que tardarán más en llegar.
  - ”Más vale pájaro en mano...”
- Ayuda a los algoritmos a converger.



Valor óptimo de un estado

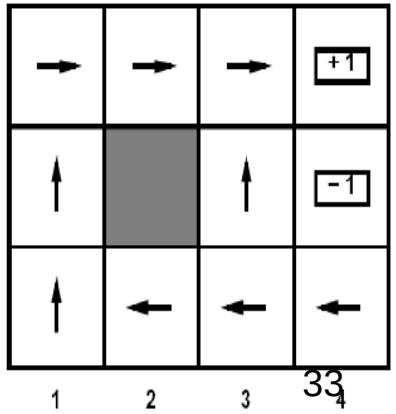
- Operación fundamental: Calcular el valor óptimo de cada estado.
- V*(s) = Valor que obtendríamos si empezásemos en s y aplicásemos la política óptima
- Q*(s,a) = Valor que obtendríamos si empezásemos en s, tomáramos la acción a y después aplicásemos la política óptima
- π*(s) = Acción óptima cuando nos encontramos en el estado s.



V*(s)

3	0.812	0.868	0.918	+1
2	0.762		0.660	-1
1	0.705	0.655	0.611	0.388
	1	2	3	4

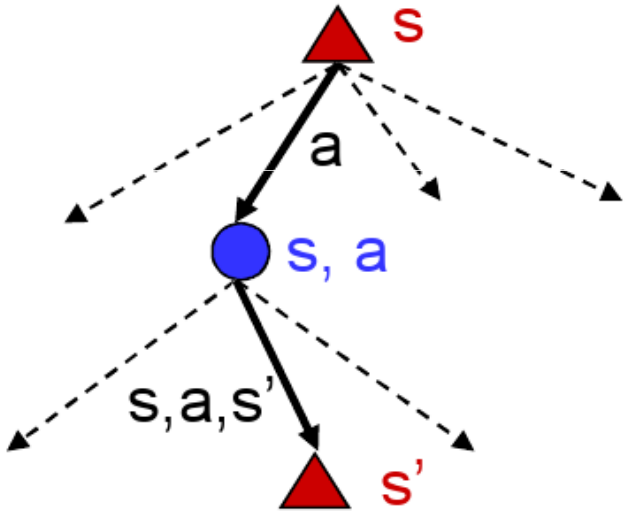
π*(s)



$\gamma = 1$ y $R(s) = -0.04$ para estados no terminales

Ecuaciones de Bellman

- La definición de ”utilidad óptima” nos lleva a pensar que obtenemos la recompensa óptima maximizando sobre la primera acción y siguiendo la política óptima a partir de ahí.



- Formalmente:

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

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

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

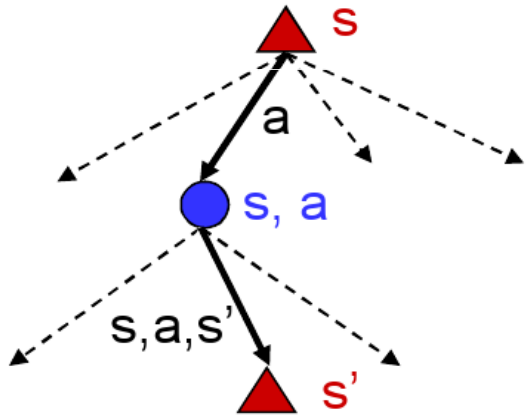
Resolviendo PDM

- Queremos encontrar una política óptima π*
- Propuesta 1: Podemos hacer una búsqueda expectimax modificada empezando en el estado s.

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

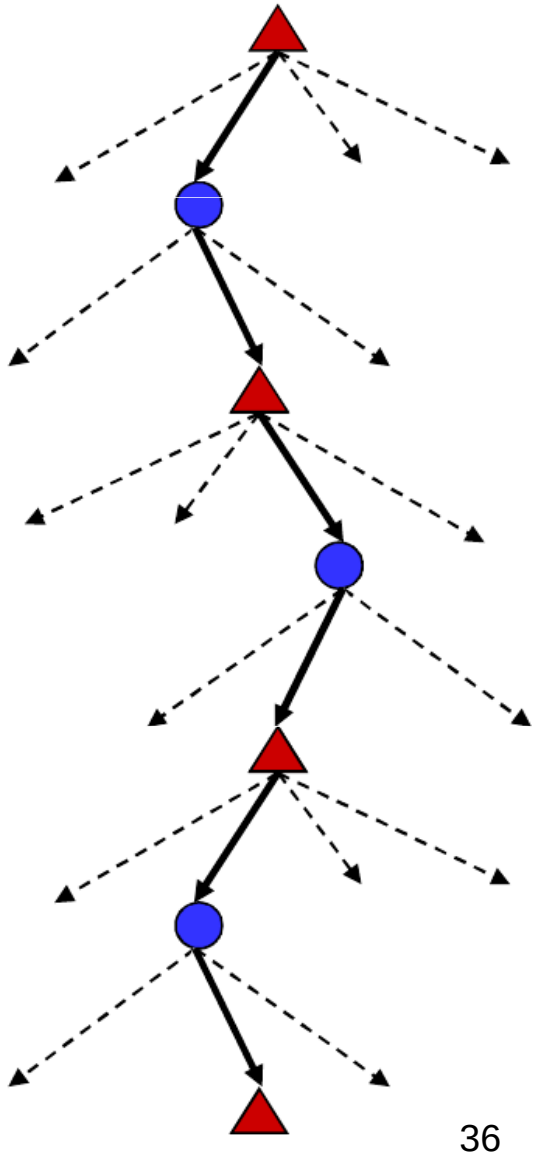
$$Q^*(s,a) = \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s') \right]$$

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



¿Por qué no utilizar búsqueda?

- El árbol es usualmente infinito.
- Los mismos estados aparecen una y otra vez.
- Buscaríamos una vez por estado.



Iteración de valores

- Computar los valores óptimos para todos los estados al mismo tiempo utilizando aproximaciones sucesivas.
- $V^*_0, V^*_1, \ldots, V^*_k, \ldots$
- Una vez terminamos, no necesitamos replanificar (toda la planificación se realiza offline).

Estimación de valor

- Para cada estado s calculamos estimaciones $V^*_k(s)$:
  - No son el valor óptimo de s!!
  - Son el valor óptimo considerando k recompensas.
  - Cuando $k \rightarrow \infty$, se aproxima al valor óptimo
- ¿Por qué?
  - Cuando descontamos, las recompensas que están lejos se vuelven negligibles.
  - Si desde cualquier parte se puede alcanzar un estado terminal, la fracción de episodios que no terminan se convierte en negligible.
  - En otro caso, podríamos tener utilidad infinita y la aproximación no funcionará.

Iteración de valores

- Idea (algoritmo):
  - Empezar con $V^*_0(s)=0$
  - Dado V^*_i, calcular el valor de todos los estados a profundidad i+1

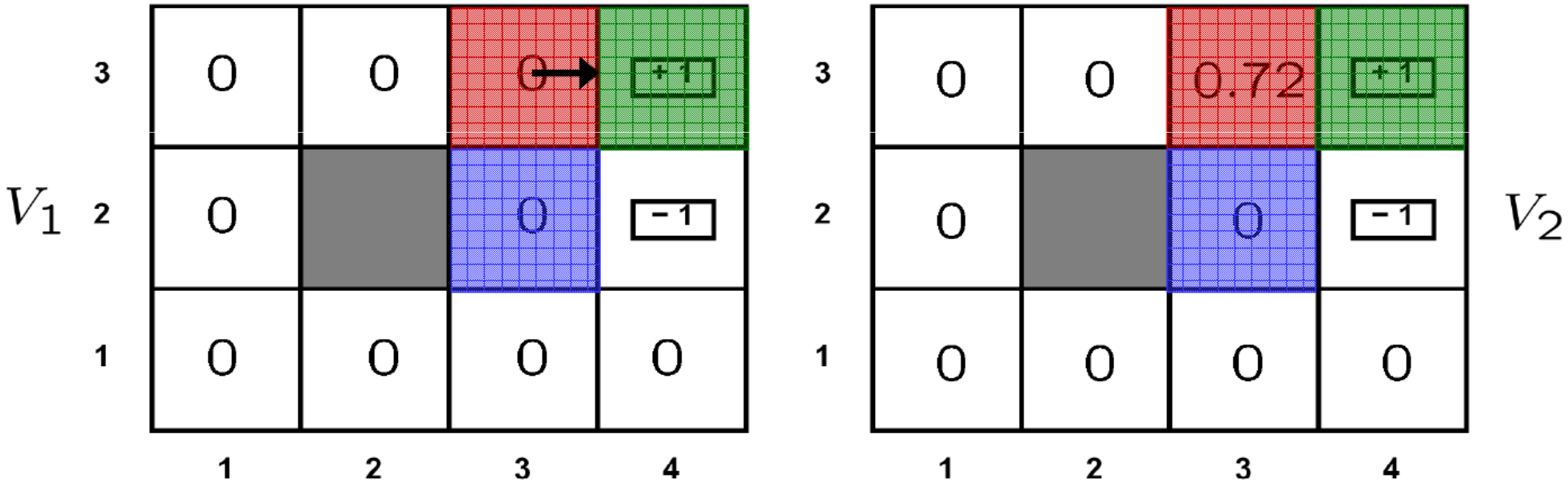
$$V_{i+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i(s') \right]$$

  - Esta es la actualización de Bellman.
  - Repetir hasta que converja
- Teorema: El algoritmo converge a valores óptimos únicos.
  - Nota: Las políticas pueden converger mucho antes de que lo hagan los valores

Ecuaciones de Bellman: Ejemplo

(extraído del curso de Dan Klein – UC Berkeley)

Example: γ=0.9, living reward=0, noise=0.2



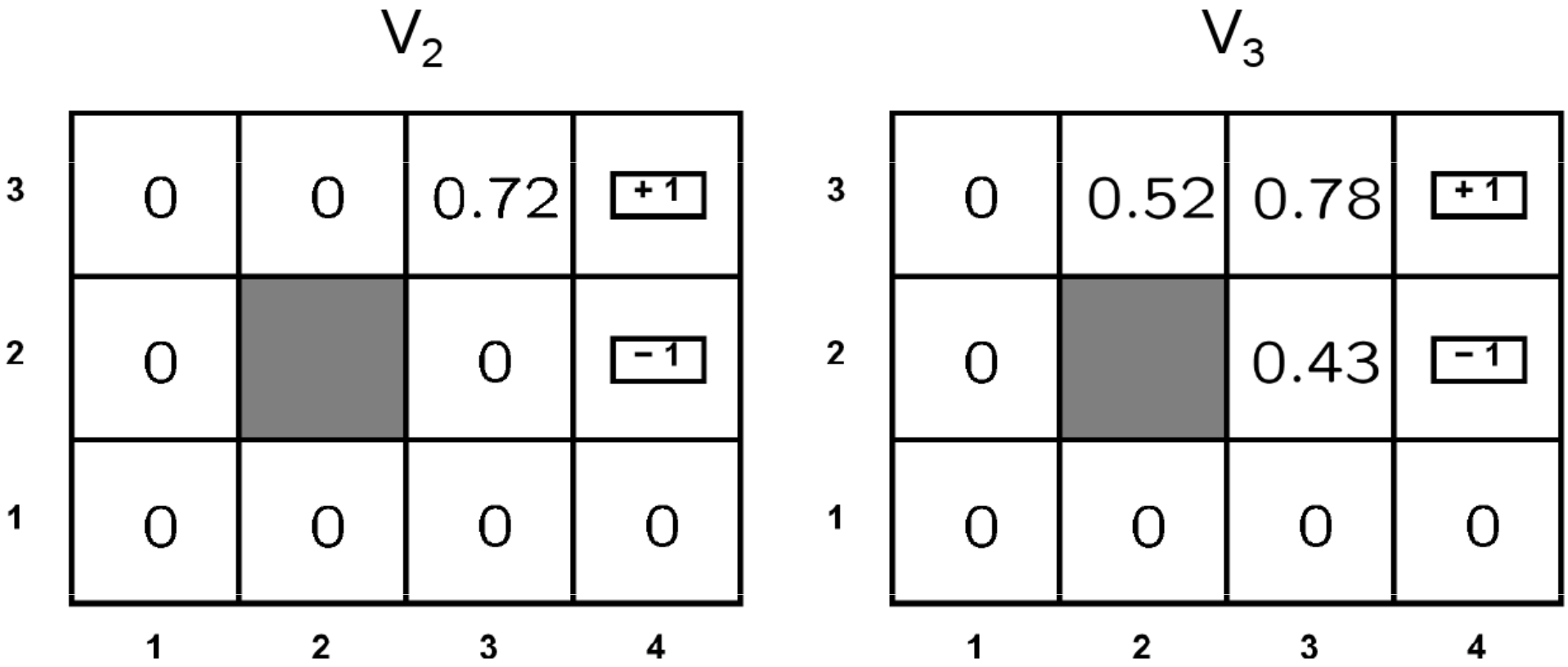
$$V_{i+1}(s) = \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i(s') \right]$$

$$V_2(\langle 3,3 \rangle) = \sum_{s'} T(\langle 3,3 \rangle, \text{right}, s') \left[R(\langle 3,3 \rangle) + 0.9 V_1(s') \right]$$

max happens for a=right, other actions not shown

$$= 0.9 \left[0.8 \cdot 1 + 0.1 \cdot 0 + 0.1 \cdot 0 \right]$$

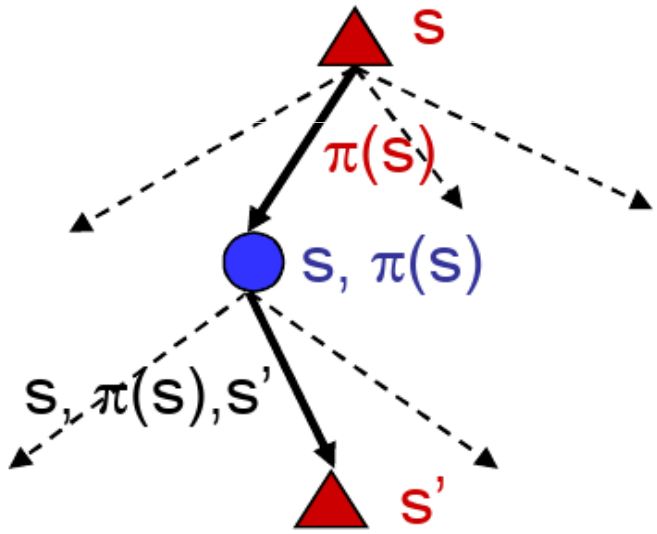
Ejemplo: Iteración de valores



- La información se propaga hacia atrás desde los estados terminales. Al final todos los estados tienen estimaciones de valor correctas

Evaluación de políticas

- Otra operación básica: calcular el valor de un estado en una política (en general, no necesariamente óptima) dada.
- $V^{\pi}(s)$ = Suma total de recompensas esperadas a partir de s si seguimos la política π.



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

Evaluación de políticas

- ¿Cómo calculamos las V's para una política π determinada?
- Idea 1: Modifiquemos las ecuaciones de Bellman

$$V_0^\pi(s) = 0$$

$$V_{i+1}^\pi(s) \leftarrow \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V_i^\pi(s')]$$

- Es simplemente un sistema lineal, se puede resolver con Matlab (por ejemplo).

Iteración de políticas

- Problemas de la iteración de valores:
  - Considerar todas las acciones en cada iteración es lento. Tarda |A| veces más tiempo que la evaluación de una política
  - En ese tiempo la política no cambia... tiempo perdido.
- Alternativa a la iteración de valores:
  - Iteración de políticas:
    - Repetir hasta que converja:
      - Paso 1: Evaluar una política
      - Paso 2: Mejorar la política

Iteración de políticas

- Repetir hasta que converja la política
  - Evaluación de política: Dada la política actual π encontrar los valores asociados aplicando las ecuaciones de Bellman simplificadas de forma iterativa

$$V_{i+1}^{\pi_k}(s) \leftarrow \sum_{s'} T(s, \pi_k(s), s') \left[R(s, \pi_k(s), s') + \gamma V_i^{\pi_k}(s') \right]$$

  - Mejora de política: Determinar una nueva política a partir de los valores encontrados en el paso anterior
  - $\pi_{k+1}(s) = \arg\max_a \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V^{\pi_k}(s') \right]$

Comparación

- Iteración de valores:
  - Cada paso modifica tanto los valores de cada estado (explícitamente) como la política.
- Iteración de políticas:
  - Varias iteraciones para calcular las utilidades de una política determinada.
  - Cada iteración se modifica la política.

$$\pi_0 \xrightarrow{\text{E}} V^{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} V^{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi^* \xrightarrow{\text{E}} V^*,$$

Iteración de valores
(libro Russell and Norvig)

function VALUE-ITERATION(mdp, ϵ) returns a utility function
  inputs: mdp, an MDP with states S, transition model T, reward function R, discount γ
    ϵ, the maximum error allowed in the utility of any state
  local variables: U, U', vectors of utilities for states in S, initially zero
    6, the maximum change in the utility of any state in an iteration

repeat
  $U \leftarrow U'; \delta \leftarrow 0$
  for each state s in S do
    $U'[s] \leftarrow R[s] + \gamma \max_a \sum_{s'} T(s, a, s') U[s']$
    if $|U'[s] - U[s]| > \delta$ **then** $\delta \leftarrow |U'[s] - U[s]|$
until $\delta < \epsilon(1-\gamma)/\gamma$
return U

Figure 17.4 The value iteration algorithm for calculating utilities of states. The termination condition is from Equation (17.8).

Actualización de Bellman usada: $U_{i+1}(s) \leftarrow R(s) + \gamma \max \sum_{s'} T(s,a,s') U_i(s')$.

Iteración de valores

(e-libro Sutton and Barto)

Initialize V arbitrarily, e.g., $V(s) = 0$, for all $s \in \mathcal{S}^+$

Repeat

$\Delta \leftarrow 0$

For each $s \in \mathcal{S}$:

$v \leftarrow V(s)$

$V(s) \leftarrow \max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that

$\pi(s) = \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma V(s')\right]$

Figure 4.5: Value iteration.

Iteración de políticas
(libro Russell and Norvig)


```
function POLICY-ITERATION(mdp) returns a policy
  inputs: mdp, an MDP with states S, transition model T
  local variables: U, a vector of utilities for states in S, initially zero
                   π, a policy vector indexed by state, initially random

  repeat
    U ← POLICY-EVALUATION(π, U, mdp)
    unchanged? ← true
    for each state s in S do
      if max_a Σ_s' T(s,a,s') U[s'] > Σ_s' T(s,π[s],s) U[s'] then
        π[s] ← argmax_a Σ_s' T(s,a,s') U[s']
        unchanged? ← false
  until unchanged?
  return π
```

given a policy π_i calculate $U_i = U^{\pi_i}$, the utility of each state if π_i were to be executed.

Figure 17.7 The policy iteration algorithm for calculating an optimal policy.

Iteración de políticas

(e-libro Sutton and Barto)

UNIVERSITAT DE BARCELONA

1. Initialization

   $V(s) \in \Re$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation

   Repeat

   $\Delta \leftarrow 0$

   For each $s \in \mathcal{S}$:

   $v \leftarrow V(s)$

   $V(s) \leftarrow \sum_{s'} \mathcal{P}_{ss'}^{\pi(s)} \left[\mathcal{R}_{ss'}^{\pi(s)} + \gamma V(s') \right]$

   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$

   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement

   policy-stable $\leftarrow$ *true*

   For each $s \in \mathcal{S}$:

   $b \leftarrow \pi(s)$

   $\pi(s) \leftarrow \arg\max_a \sum_{s'} \mathcal{P}_{ss'}^{a} \left[\mathcal{R}_{ss'}^{a} + \gamma V(s') \right]$

   If $b \neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*

   If *policy-stable*, then stop; else go to 2

Figure 4.3: Policy iteration (using iterative policy evaluation) for V^*. In the "arg max" step in 3, it is assumed that ties are broken in a consistent order.

Iteración asíncrona de valores

- En la iteración de valores se actualiza cada estado en cada iteración
- De hecho, cualquier secuencia de actualizaciones de Bellman convergerá si cada estado es visitado un número infinito de veces.
- Podemos actualizar la política tan a menudo como queramos que seguirá convergiendo
- Idea: actualizar los estados cuyos valores esperamos que cambien:

  IF $|V_{i+1}(s) - V_i(s)|$ es grande THEN

  actualiza los predecesores de s

Algunas consideraciones

- Notación:
  - en las transparencias casi siempre se considera R(s,a,s'):
    - R: S x A x S → $\Re$
  - en el libro de AI (Russell and Norvig) se dice R(s)
    - R: S → $\Re$
    - "*does not change the problem in any fundamental way*"
  - U(s) para denotar V(s)
    - Utilidades de los estados (es decir, Valores)

Video: swing-up and balance from scratch
http://www.youtube.com/watch?v=Lt-KLtkDlh8
Video: robotic soccer goalkeeper
http://www.youtube.com/watch?v=CIF2SBVY-J0&NR=1

Bibliografia (MDP y RL)

- http://webdocs.cs.ualberta.ca/~sutton/book/ebook