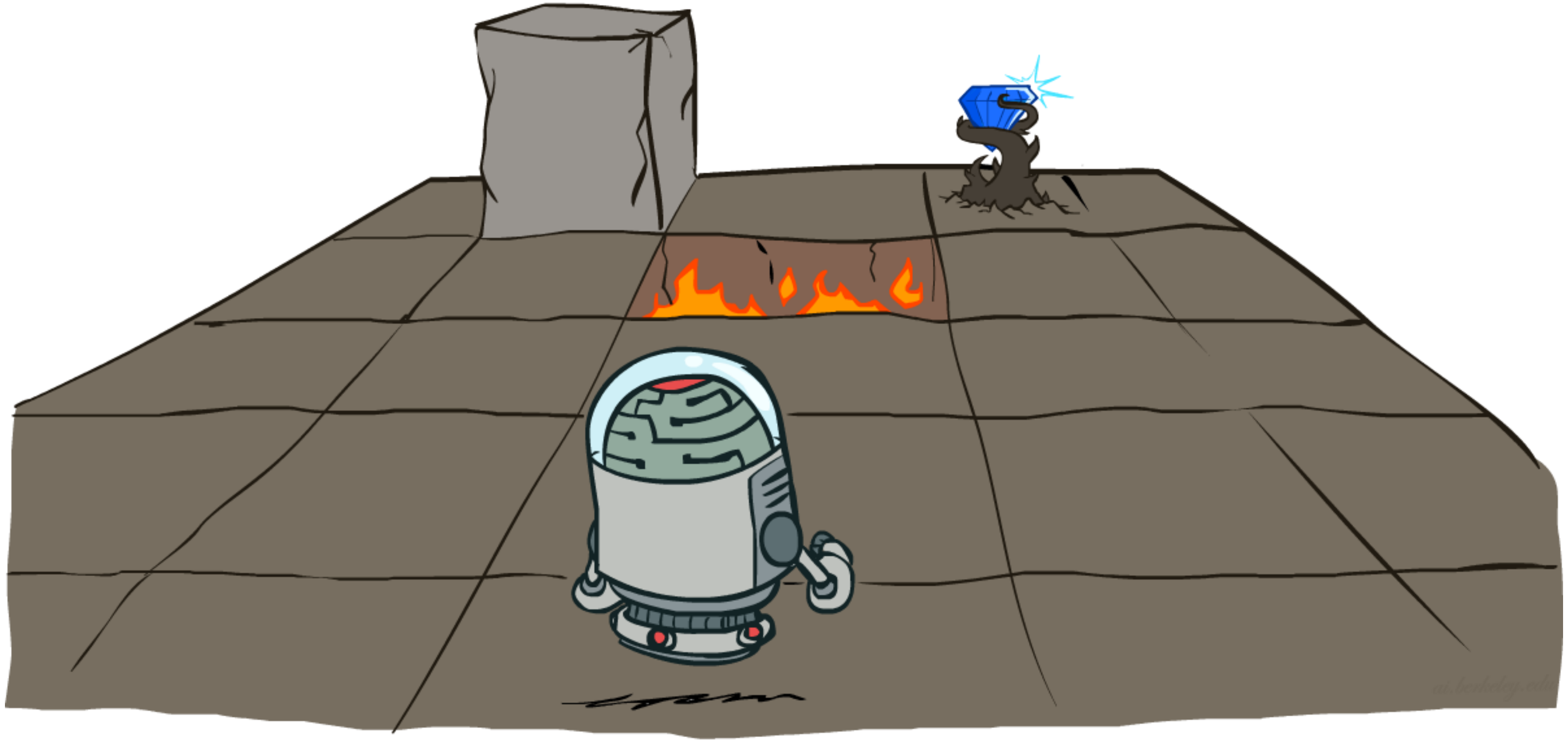
Intelligence Artificielle

Processus de decision de Markov

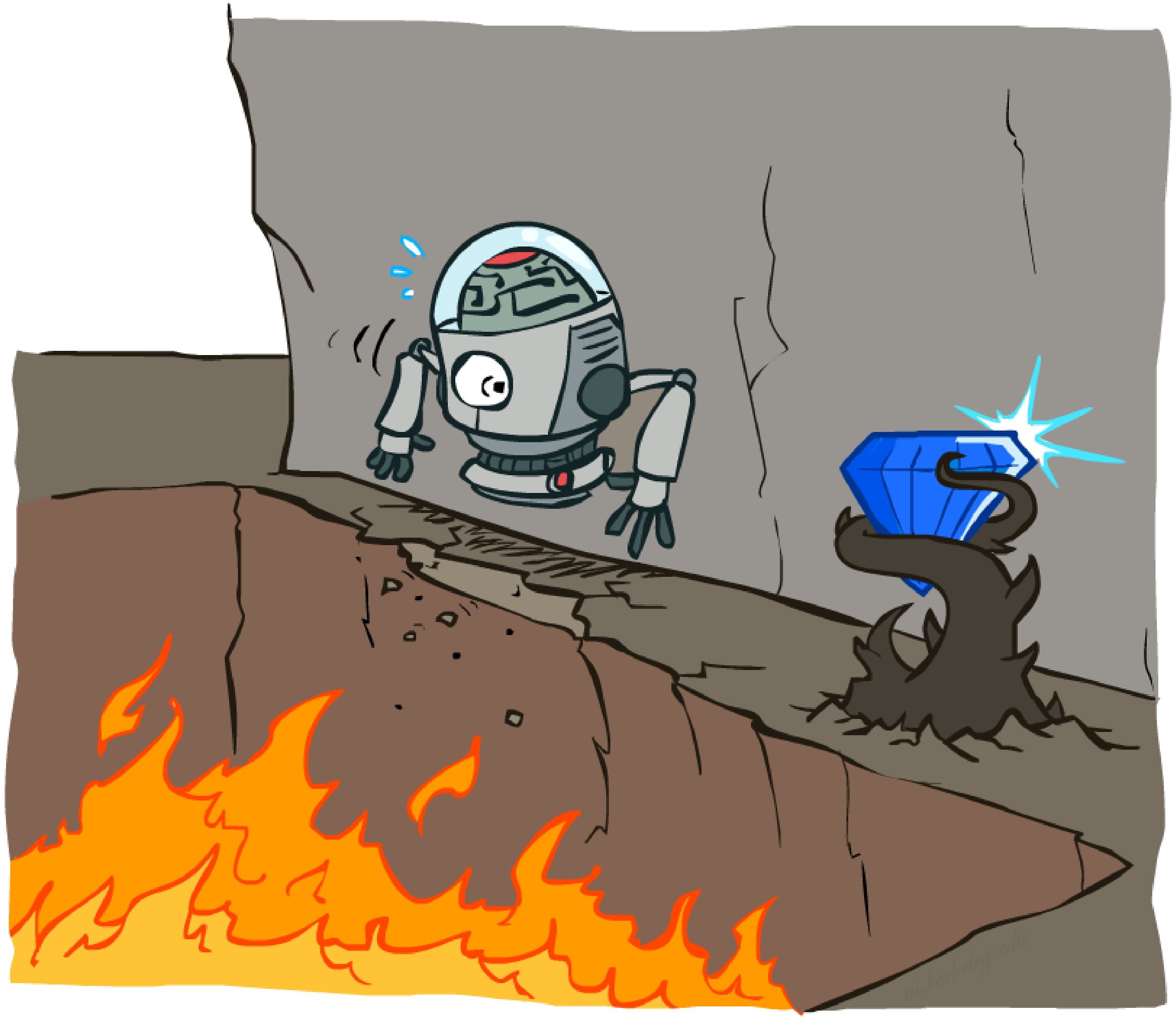


Prof: A.Belcaid

Ecole Nationale des Sciences Appliquées - Fès

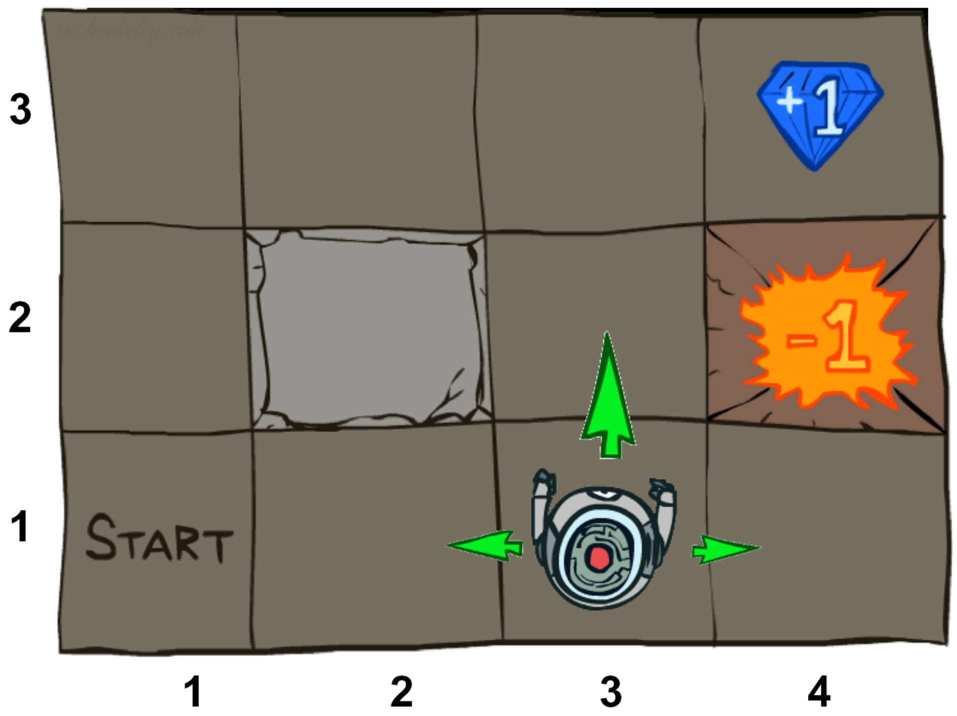
[Slides par Dan Klein and Pieter Abbeel du cours CS188 Intro to AI at UC Berkeley. All CS188 materials are available at http://ai.berkeley.edu.]

Recherche non déterministique



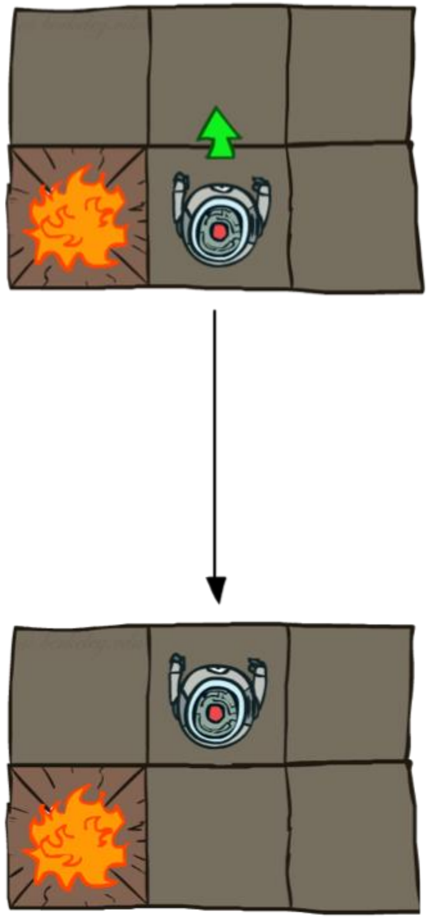
Exemple: Monde en Grille

- Un problème labyrinthe
  - L'agent vit dans une grille
  - Murs bloquent le chemin de l'agent
- Déplacement bruité: Actions ne mennent pas toujours au but
  - 80% l'action *North Prend le nord (si pas du mur)*
  - 10% *North prend l'agent Est* et 10% Ouest
  - S'il as un mur dans la direction choisie, l'agent gard sa position.
- L'agent recoit une récomponse
  - Mineure "living" recompense (peut être négative)
  - Majeure à la fin (selon l'état de terminaison)
- Objectif: Maximiser la somme des recompenses.

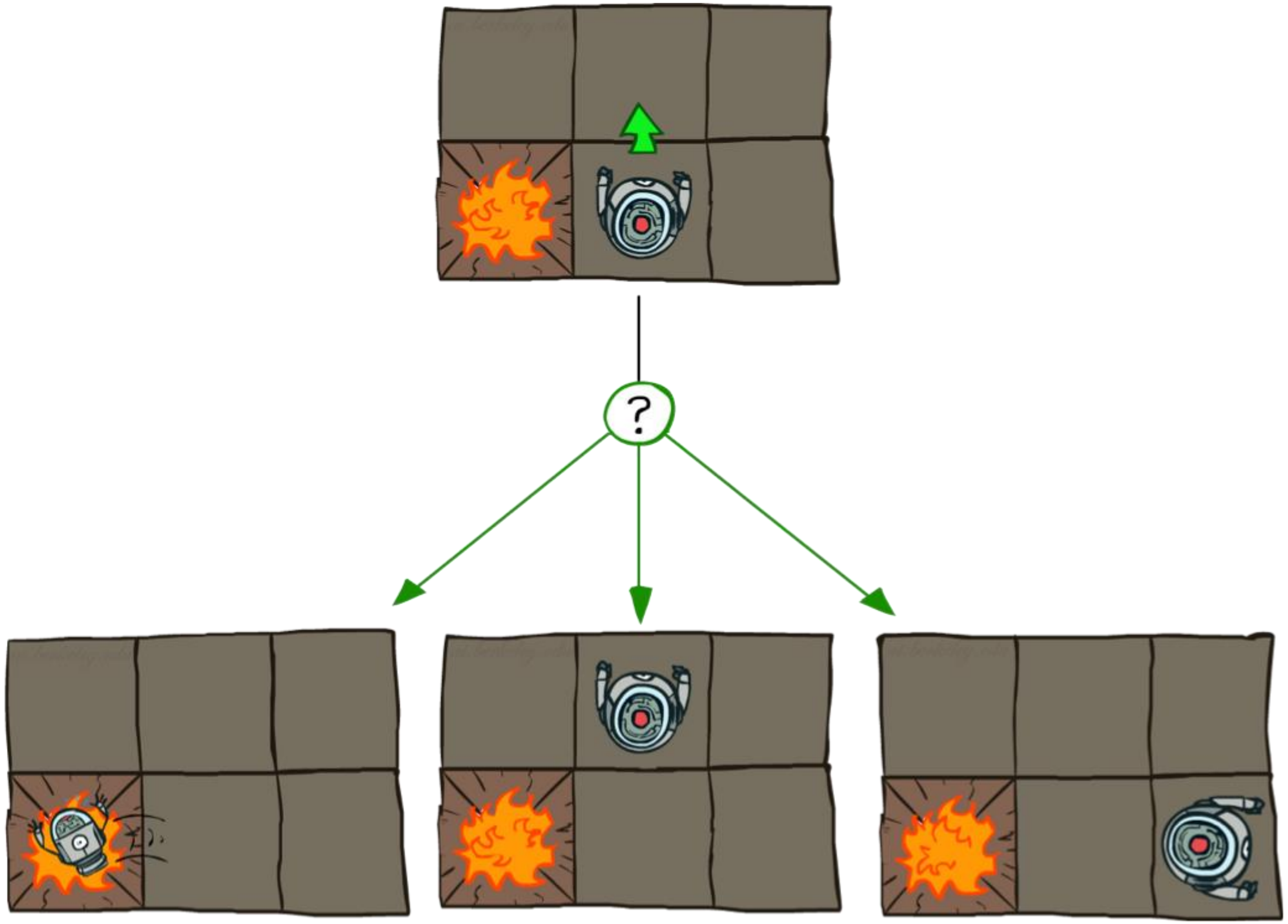


Actions dans la grille

Déterministique

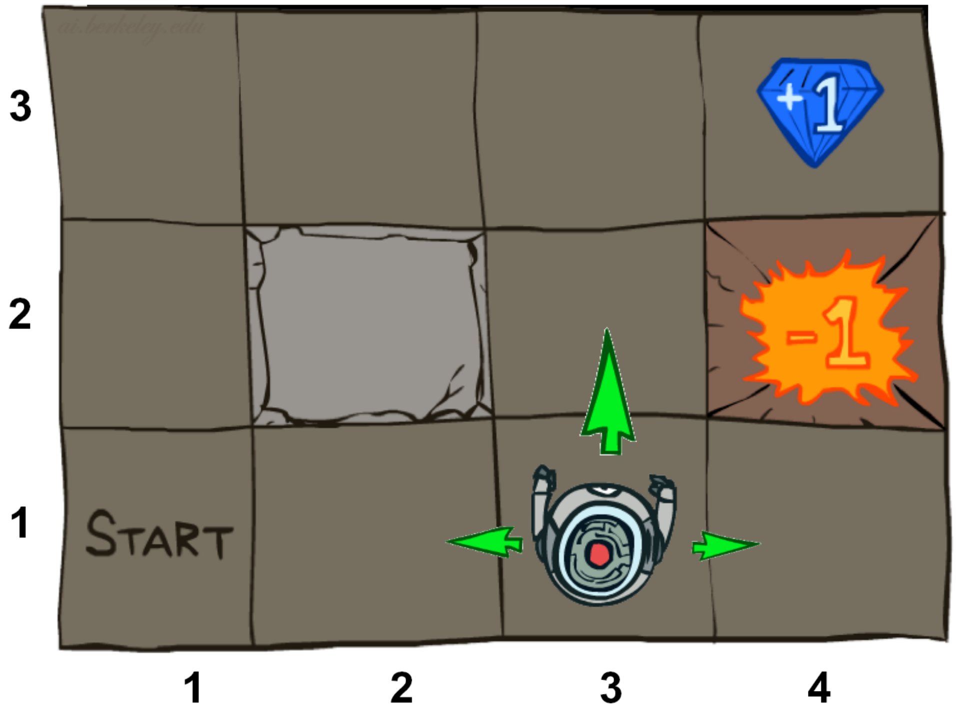


Stochastique



Processus de decision de Markov (MDP)

- Un MDP est défini par:
  - Un ensemble des états s ∈ S
  - Un ensemble des actions a ∈ A
  - Une function de transition T(s, a, s’)
    - Probabilité que **a** de **s** mène à **s’**, i.e., **P(s’| s, a)**
    - Appellée aussi **dynamique du modèle.**
  - Une function récompense R(s, a, s’)
    - Parfois juste R(s) ou R(s’)
  - Un état de départ
  - Possible aussi un état terminal.



- MDPs sont des problems de recherch stochastiques.

[Demo – gridworld manuel (L8D1)]

Relation Markov et MDP?

- "Markov" implique généralement que la future est **indépendant** du passé étant donné le present.

- Au point de vue des MDS, la "Markov" implique que l'état à t+1, depend suelement de l'état actuel t.

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

$$=$$

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

- Similaire au problème de recherché, la transition depend seulement de l'état actuel et non pas du chemin.

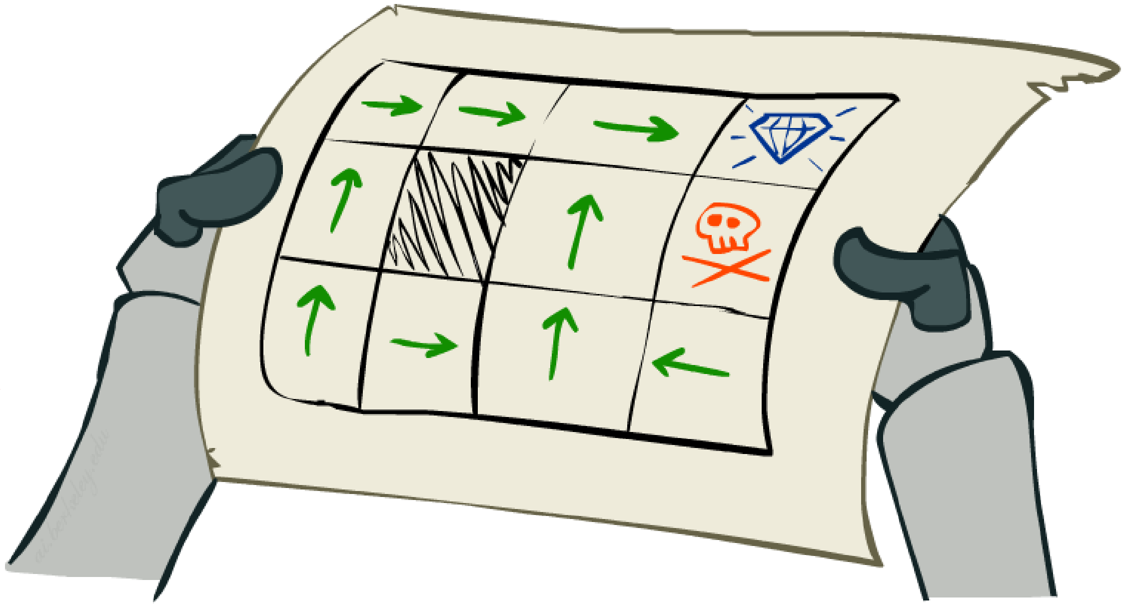
Andrey Markov
(1856-1922)

Stratégies

- Pour les problemes déterministique, on cherche une sequence d'action (plan) qui ne ramène à l'objectif.

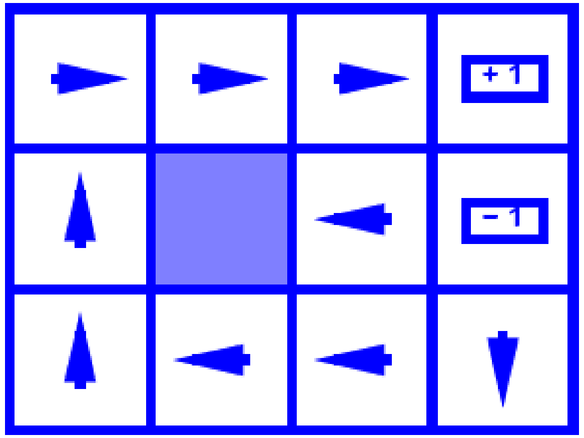
- Pour les MDPs, On cherche stratégie $\pi^*: S \rightarrow A$
  - Une stratégie π associe une action à chaque état.
  - Une stratégie optimale maximize l'espérance des récomponses.
  - Une stratégie explicite définit un agent de reflèxe.

- Expectimax ne calculi pas la stratégie en intégrité

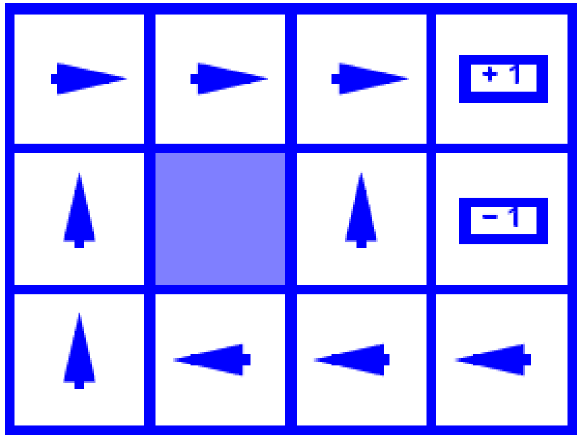


Stratégie optimale avec R(s, a, s') = -0.03.

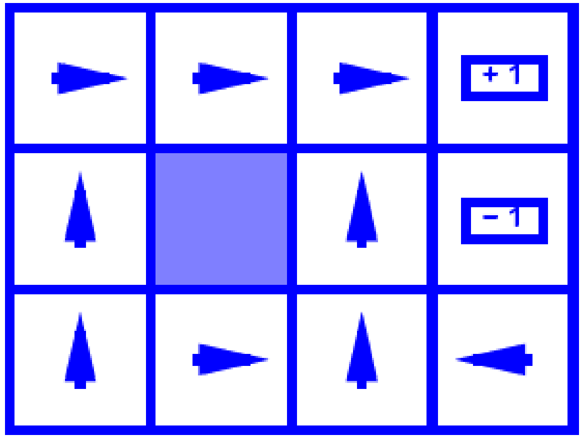
Stratégies optimales.



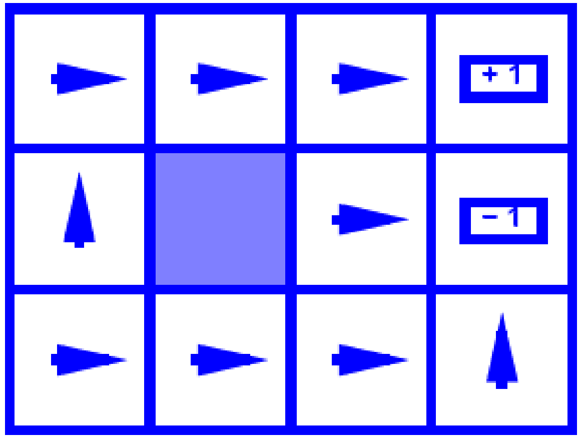
R(s) = -0.01



R(s) = -0.03

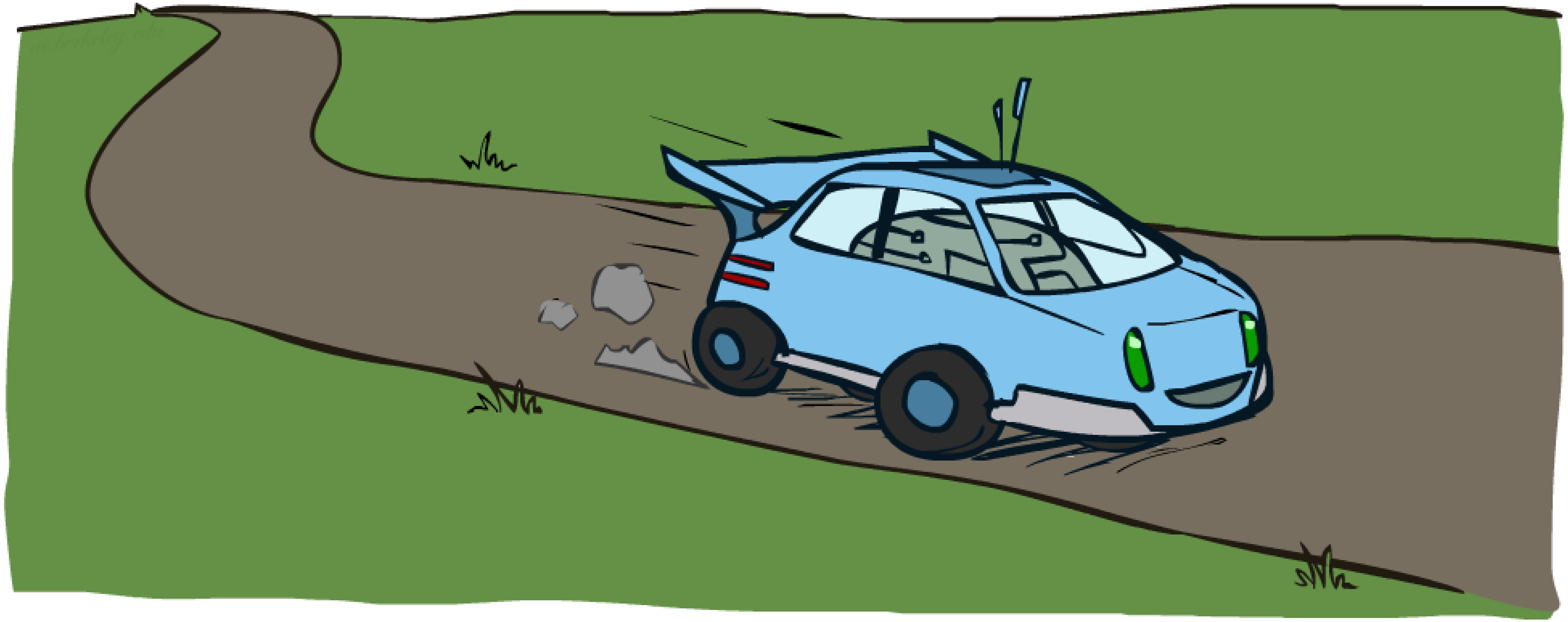


R(s) = -0.4



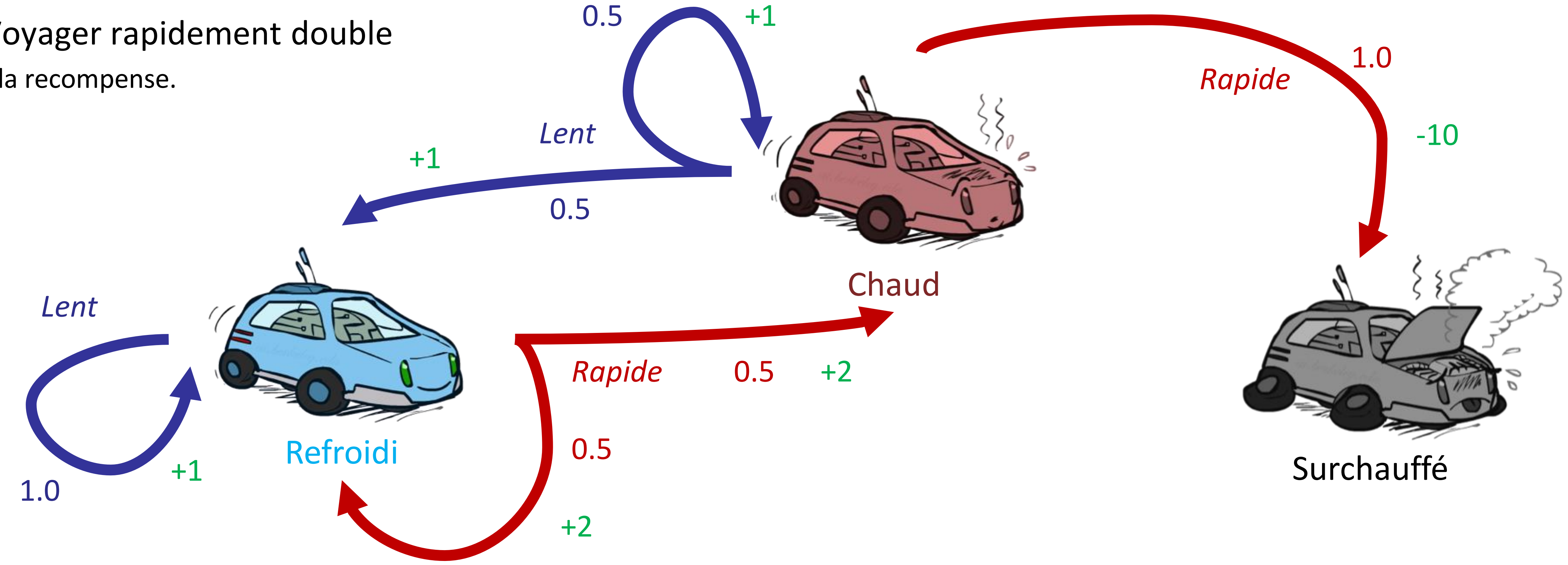
R(s) = -2.0

Exemple: Course

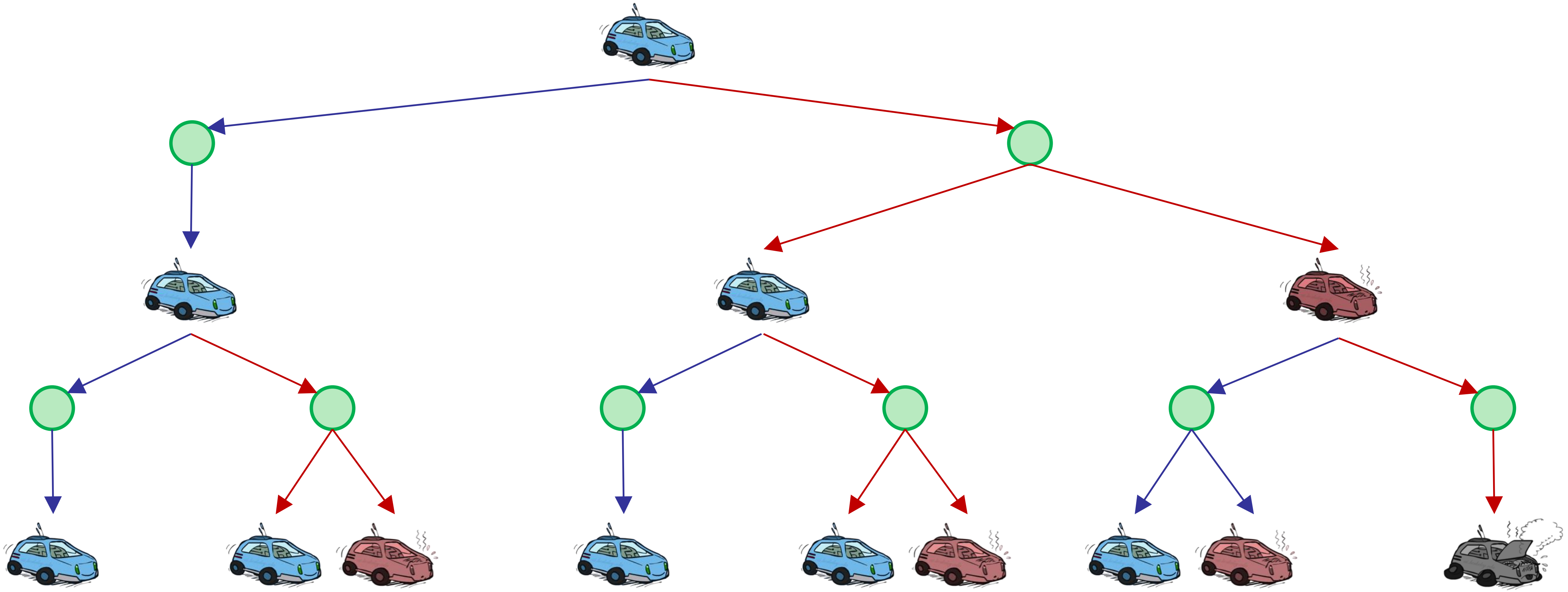


Exemple: Course

- Un robot voiture veut voyager **rapidement.**
- Trois états: Refroidi, Chaud, Surchauffé
- Deux actions: *Lent*, *Rapide*
- Voyager rapidement double la recompense.

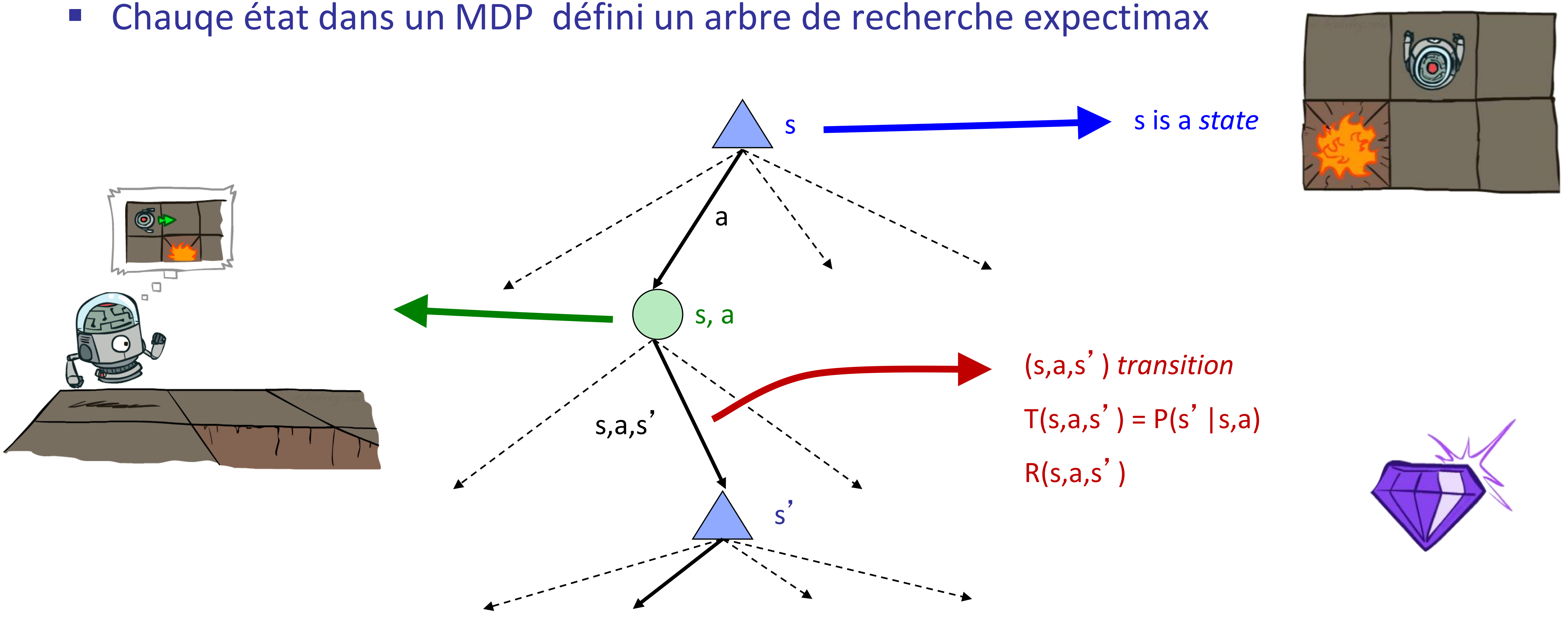


Arbre de recherche



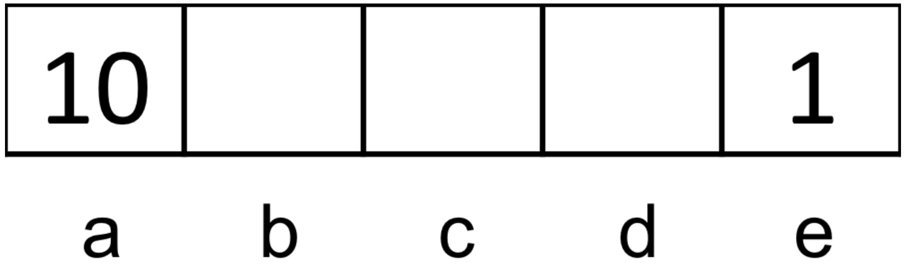
Arbre de recherche d’un MDP

- Chauqe état dans un MDP défini un arbre de recherche expectimax



Quiz: Représentation MDP

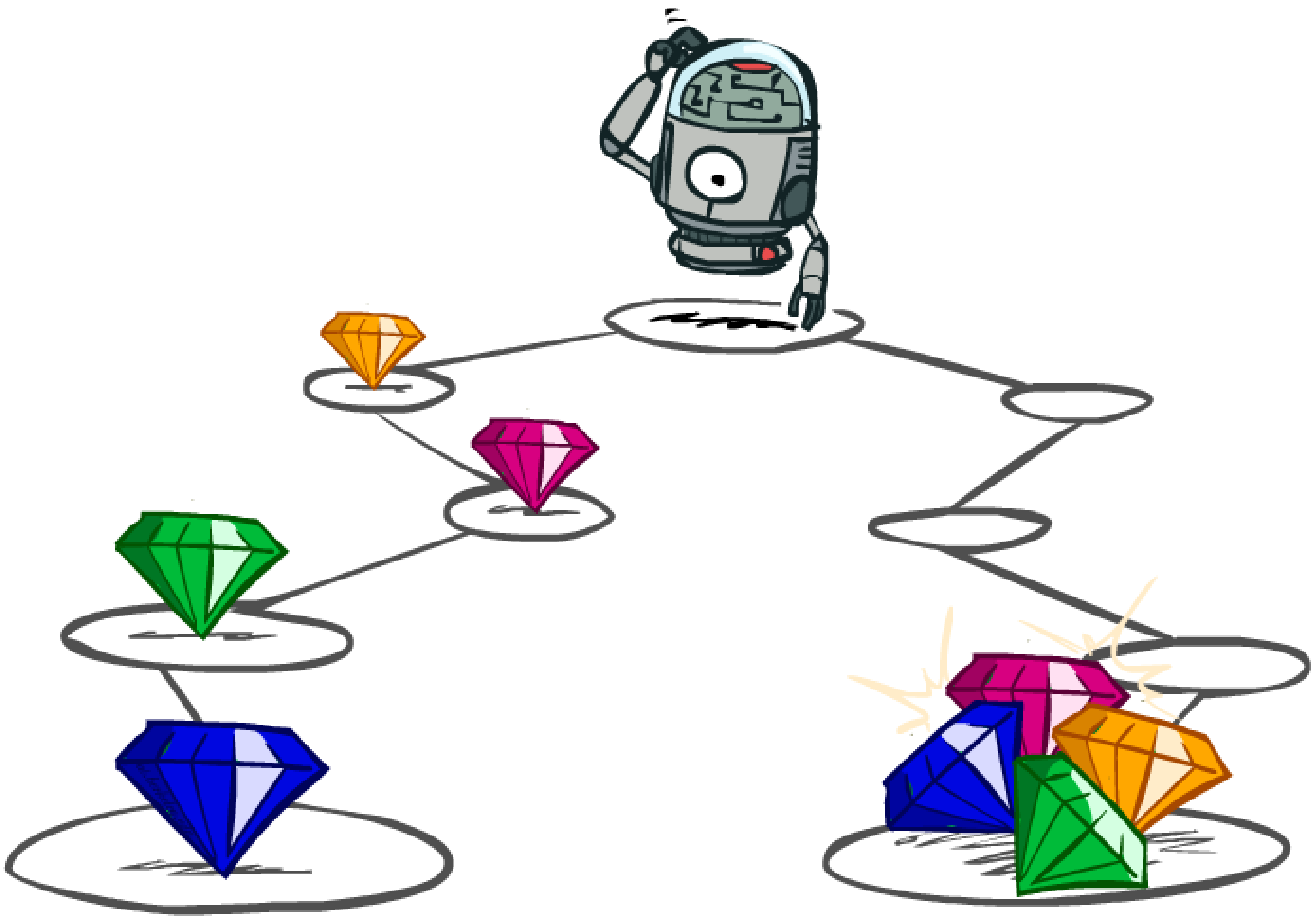
On considère le **MDP** représentée dans la figure, On possède deux point terminaux **a** avec une récompense 10 et **e** avec une récompense de 1. Les deux actions possible dans chaque état sont: **Left, Right** avec une probabilité de réussite **80%.** Au cas d'echec l'agent reste à ca place. Remplir les valeur demandées?



- T(c,Right,d)
- T(c,Right,e)
- T(c,Right, c)
- T(c,Right, b)
- T(c, Right, a)
- T(c,Right, Etat terminal)

- T(a,Exit, Etat terminal)
- T(a, Right, b)
- T(a,Right, a)
- R(a, Right, a)
- R(a,Right, Etat terminal)
- R(c, Right, d)

Utilité d’une Séquence

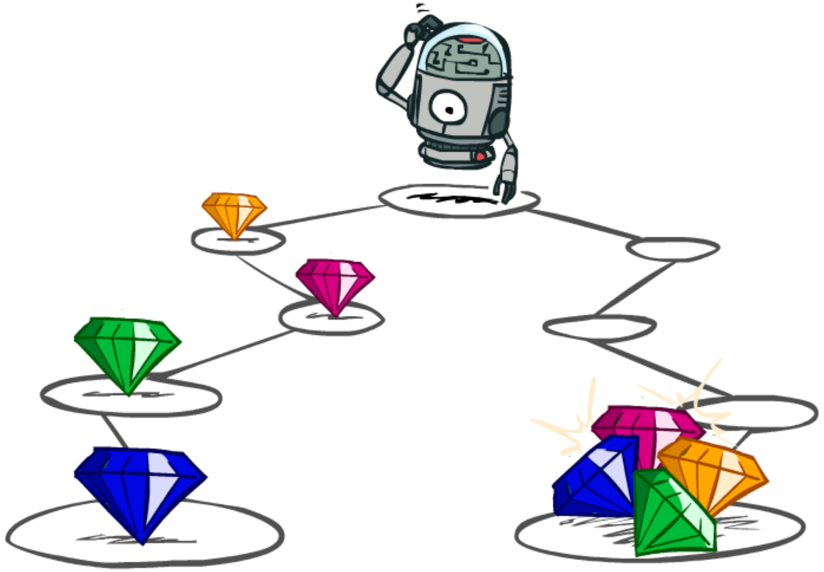


Utilité d'une Séquence

- Quelle préférences pour un agent entre deux séquences?

- Plus or Moins? [1, 2, 2] or [2, 3, 4]

- Temps? [0, 0, 1] or [1, 0, 0]



Remise

- Raisonnable de maximiser la somme des recompenses.
- Aussi raisonnable de préférer les recompenses au temps actuel.
- Une solution: valeurs des recompenses décroit exponentielle.

1

actuelle

γ

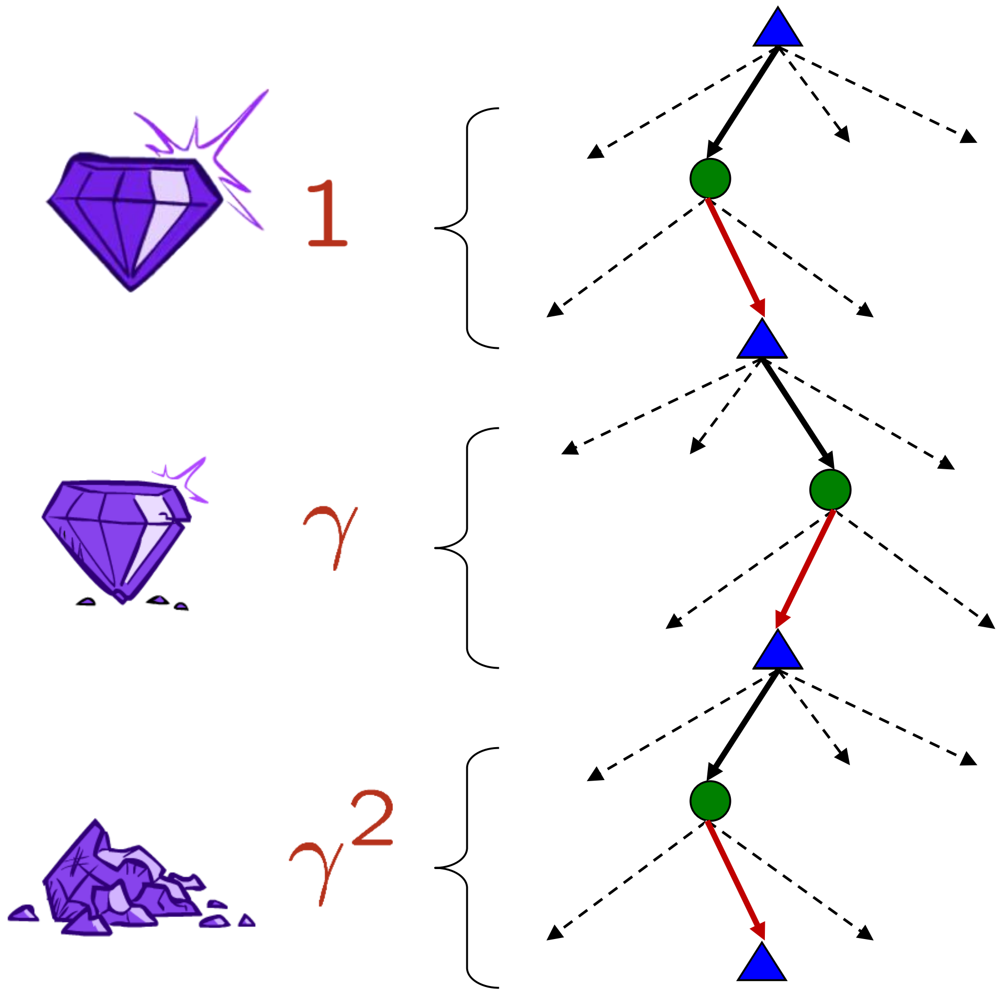
Une itération

γ^2

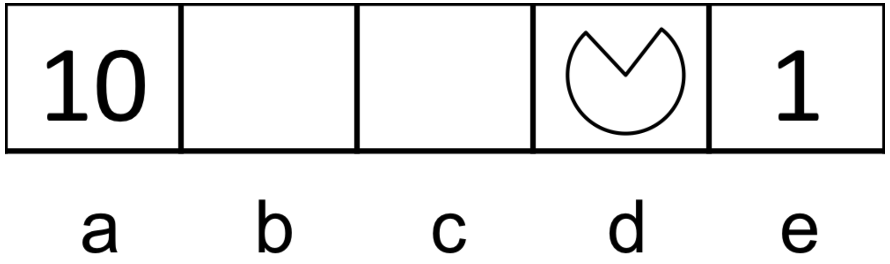
Deux itérations

Discounting

- Comment appliquer la remise?
  - Chaque iteration dans l'arbre, on multiplie par γ.
- Pourqu'oi?
  - Préférences des recompenses proches dans le temps.
  - Aussi aide l'algorithme à converger
- Exemple: remise à 0.5
  - U([1,2,3]) = 1*1 + 0.5*2 + 0.25*3
  - U([1,2,3]) < U([3,2,1])



Quiz: Remise



On considère le même MDP du quiz 1 avec un agent Pacman dans la position **d.**

1. Pour une remise $\gamma = 0.1$, Quelle est l'action choisie par la stratégie optimal pour le nœud d.

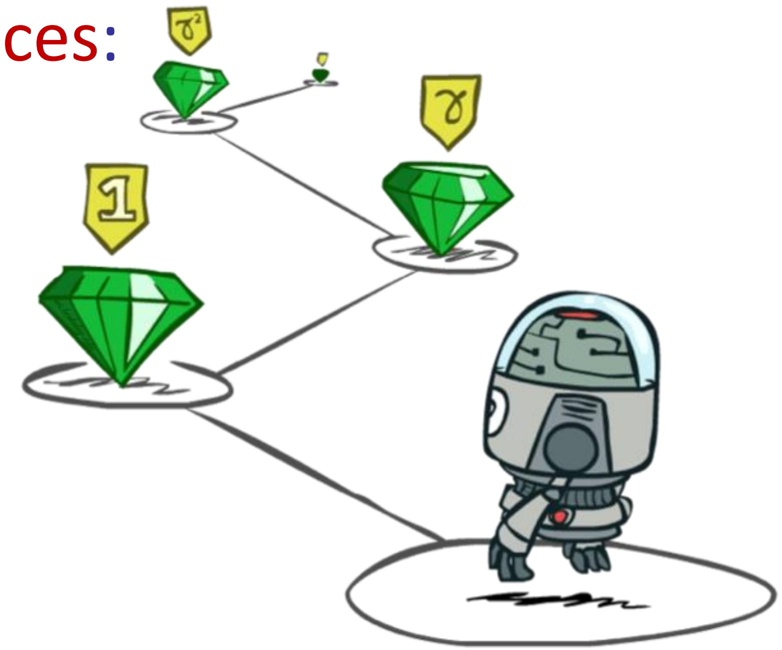
- ❑ Left
- ❑ Right

2. Répéter la question 1 pour une remise $\gamma = 0.999$?

- ❑ Left.
- ❑ Right.

Stationarité des Préférences

- Theorème: Si on suppose la stationarité des préférences:



$$[a_1, a_2, \ldots] \succ [b_1, b_2, \ldots]$$
$$\Updownarrow$$
$$[r, a_1, a_2, \ldots] \succ [r, b_1, b_2, \ldots]$$

- Alors: Seulement deux alternatives pour définir les utilities
  - Utilité additive: $U([r_0, r_1, r_2, \ldots]) = r_0 + r_1 + r_2 + \cdots$
  - Utility avec remise: $U([r_0, r_1, r_2, \ldots]) = r_0 + \gamma r_1 + \gamma^2 r_2 \cdots$

Quiz: Remise

- Etant donne :

10				1
a	b	c	d	e

  - Actions: *East, West, and Exit* (disponible seulement dans les noeuds a, e)
  - Transitions: déterministique

- Quiz 1: Pour γ = 1, Quelle la stratégie optimale?

10				1

- Quiz 2: Pour γ = 0.1, Stratégie optimale?

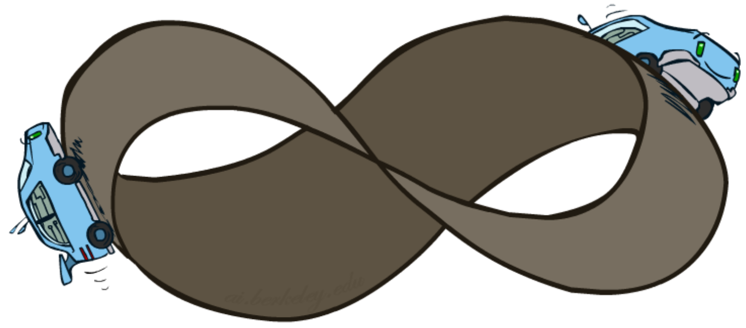
10				1

- Quiz 3: Pour quelle valeur de γ les actions West and East sont égales

Utilitiés Infinie?!

- Problème: Que se passe il si le jeu itère à l’infini?

- Solutions:
  - Horizon fini: Recherche avec profendeur finie
    - Terminer les iterations apèrs T iteration
    - Si on as pas la stationarité (π depends du temps)



  - Remise: utiliser $0 < \gamma < 1$

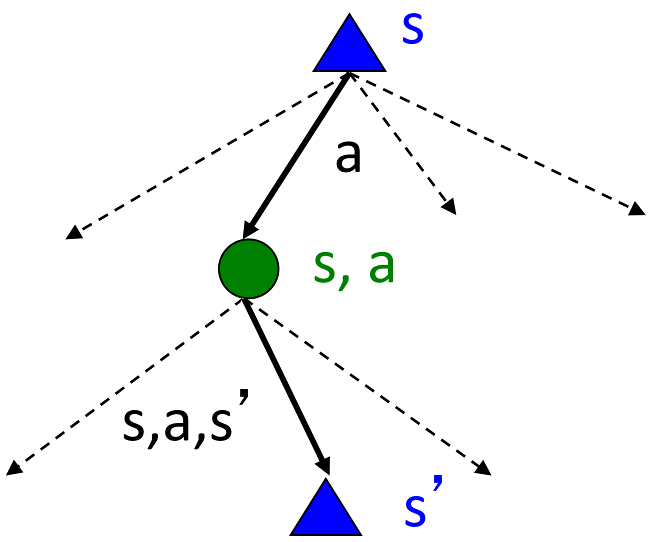
$$U([r_0, \ldots r_\infty]) = \sum_{t=0}^{\infty} \gamma^t r_t \leq R_{\max}/(1-\gamma)$$

    - Valeurs petites de γ affecte l’“horizon” (plus court)

  - Etat absorbant: Stratégie assure d’atteindre un état de terminaison (.e.g Surchaufé pour le cas de la race des voiture.

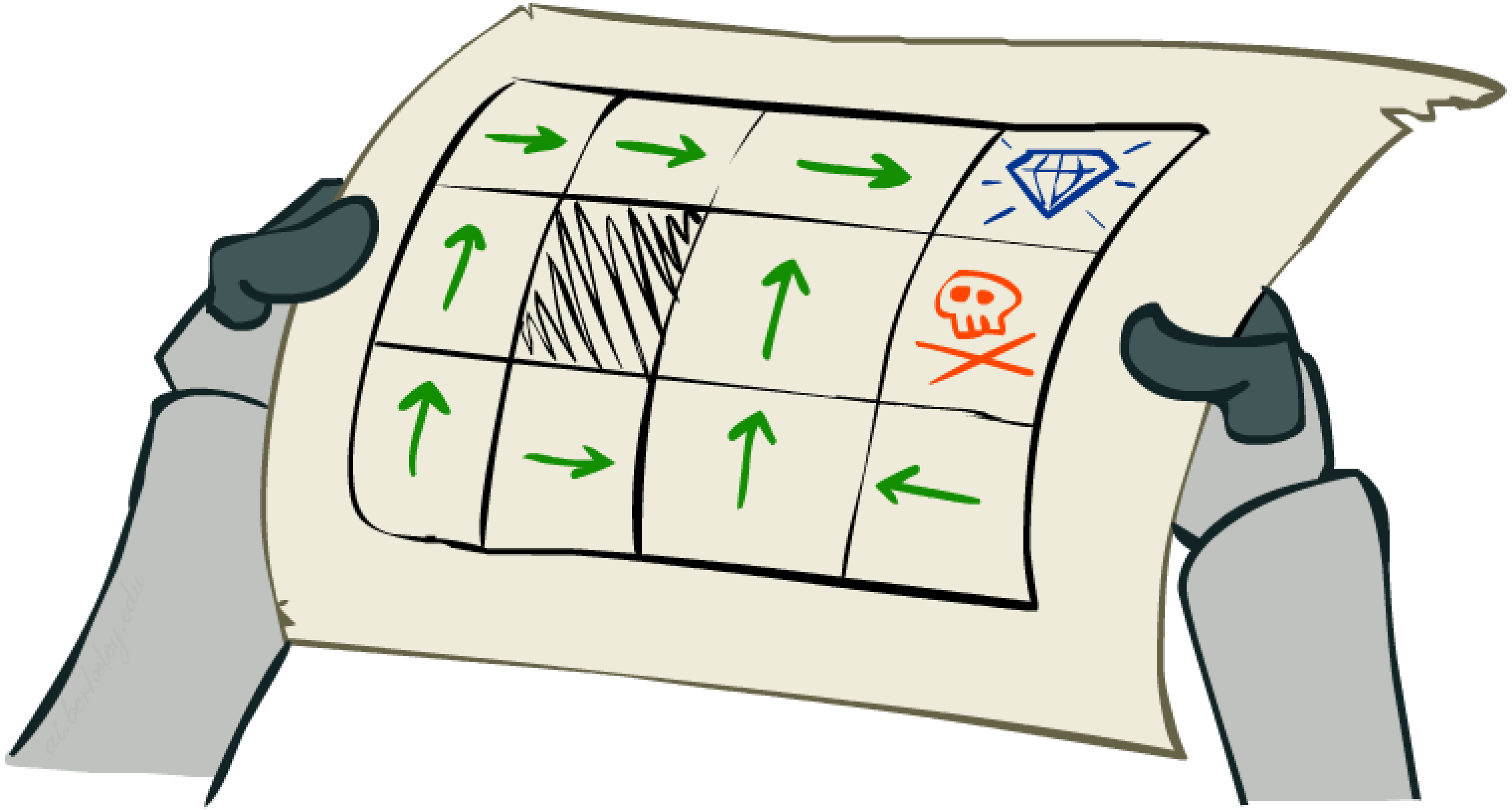
Résumé: Definition d'un MDP

- Processus de Décision de Markov :
  - Ensemble des états S
  - Etat de départ s_0
  - Ensemble d'actions A
  - Transitions P(s'|s,a) (or T(s,a,s'))
  - Récompense R(s,a,s') (et remise γ)



- Eléments d'un MDP :
  - Stratégie = Choix d'une action pour chaque état.
  - Utilité = somme des recompenses avec remise.

Résoudre un MDP



Eléments Optimaux

- La valeur utilité d'un état s:

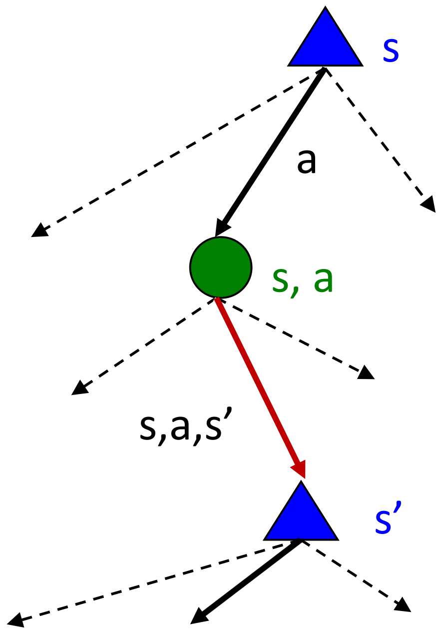
  $V^*(s)$ = Espérance d' l'utilité en commençant par s et agissant optimalement.

- La veleur utilité d'un état q: (s,a):

  $Q^*(s,a)$ = Espérance de l'utilité de létat s si on est engage à choisir l'action a.

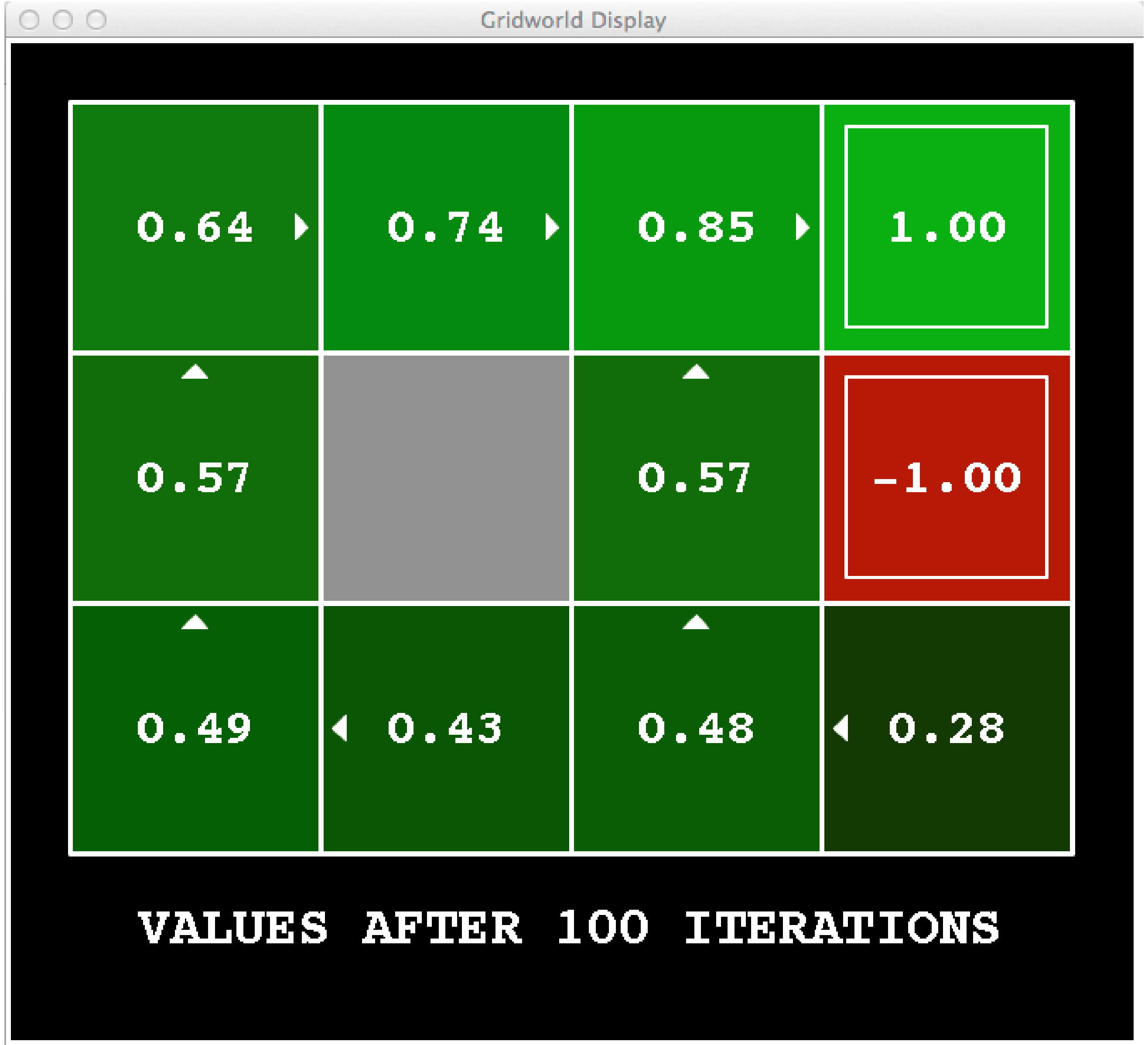
- Stratégie optimale:

  $\pi^*(s)$ = Action optimale pour le noeud s.



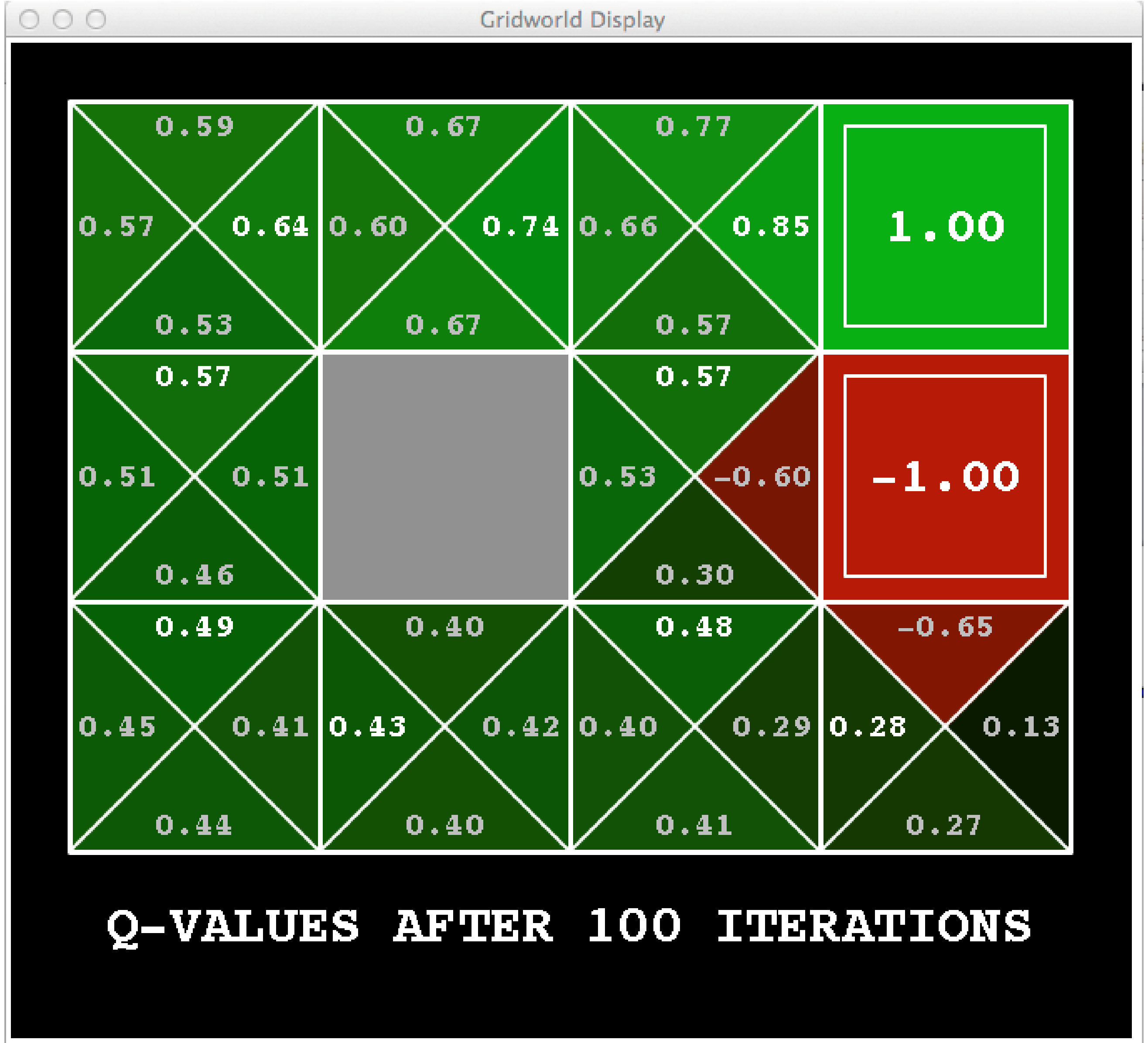
[Demo – gridworld values (L8D4)]

Snapshot of Demo – Gridworld V Values



Noise = 0.2
Discount = 0.9
Living reward = 0

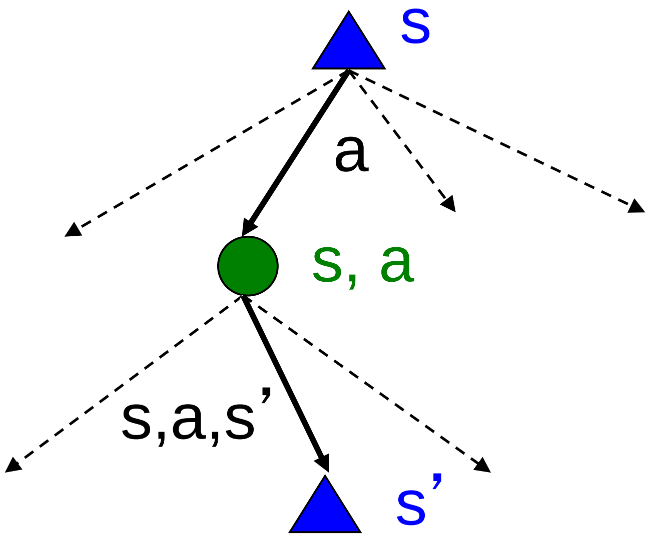
Valeurs Q



Bruit = 0.2
Remise = 0.9
Recompense vie = 0

Valeurs d’un état

- Opération fondamentale: calculer la valeur (expectimax) d’un état
  - Espérance de l’utilité avec des actions optimaux.
  - Moyenne des sommes des recompenses avec remise.
  - Exact valeur calculée par expectimax!



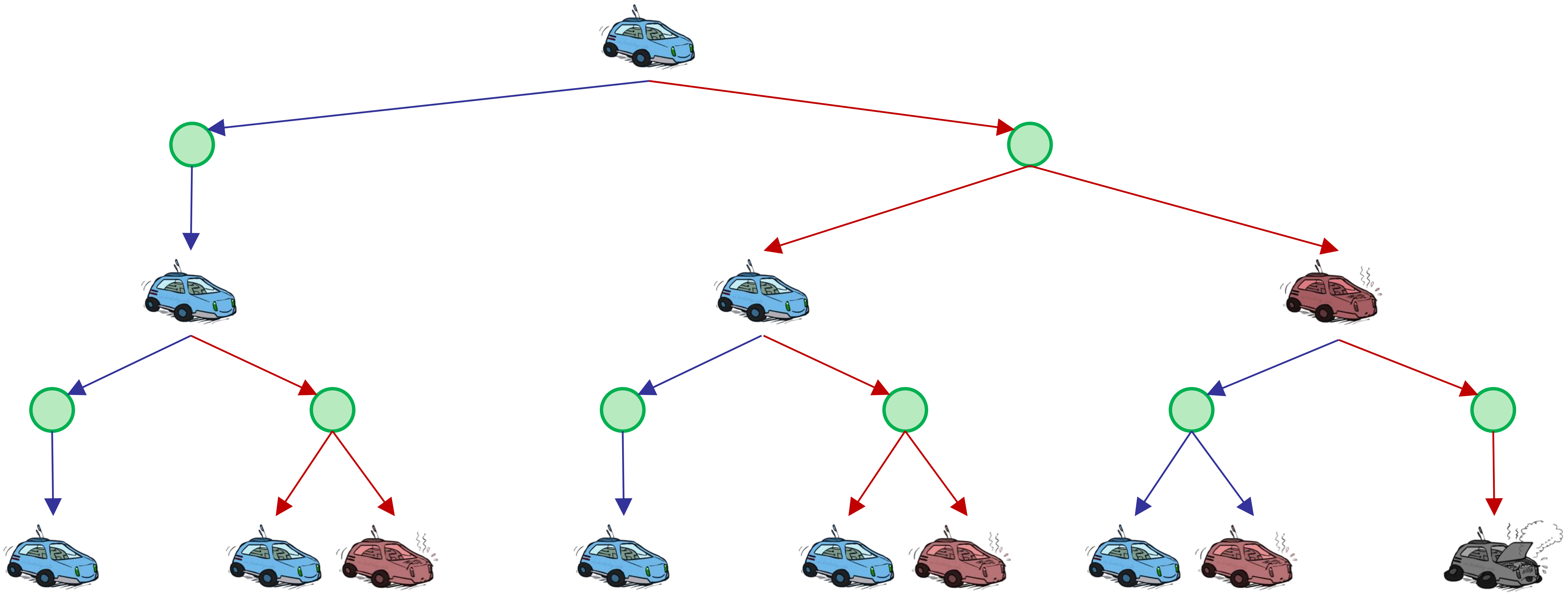
- Définition récursive:

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

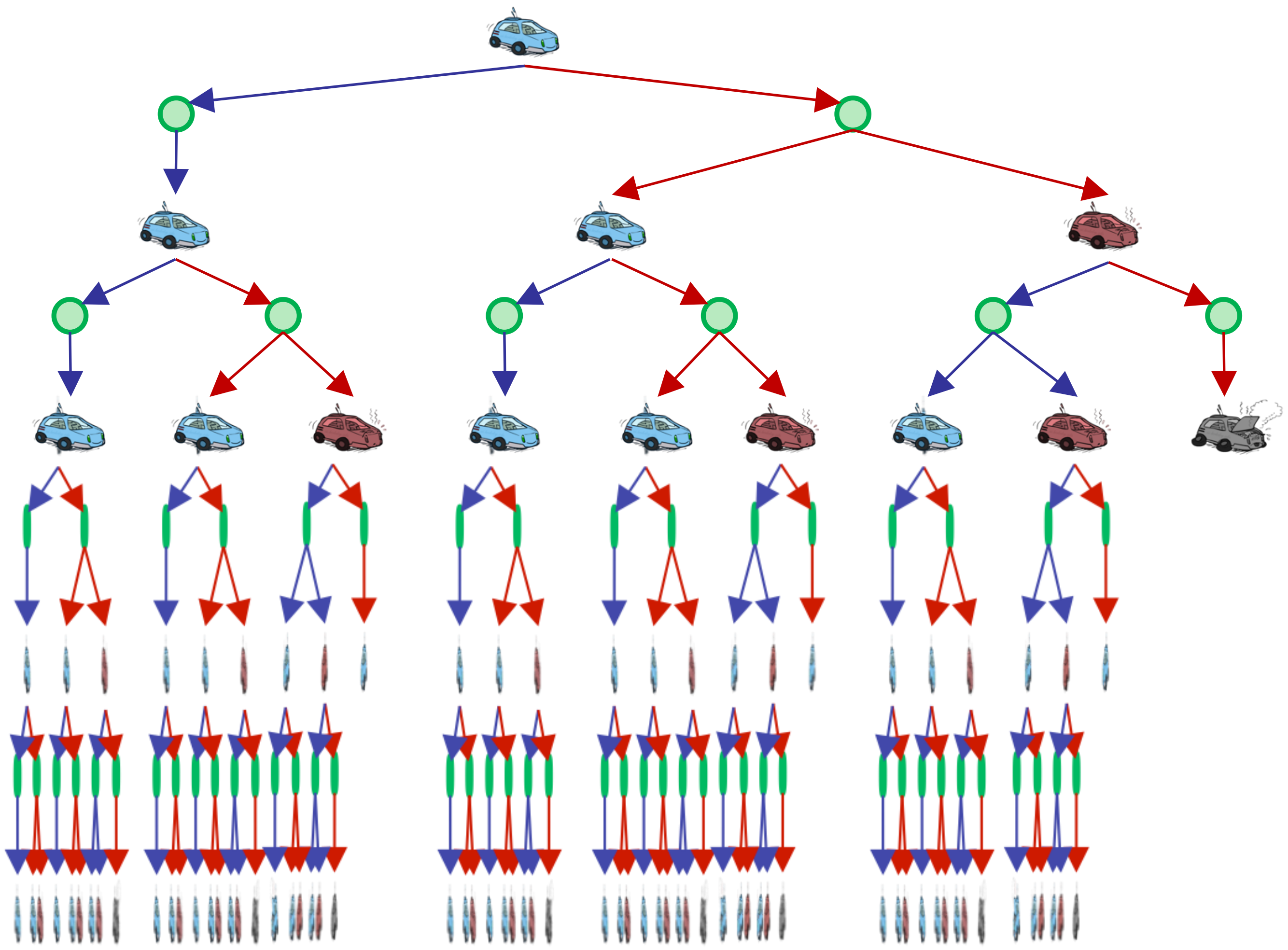
$$Q^*(s,a) = \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s')\right]$$

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

Arbre de recherche Course

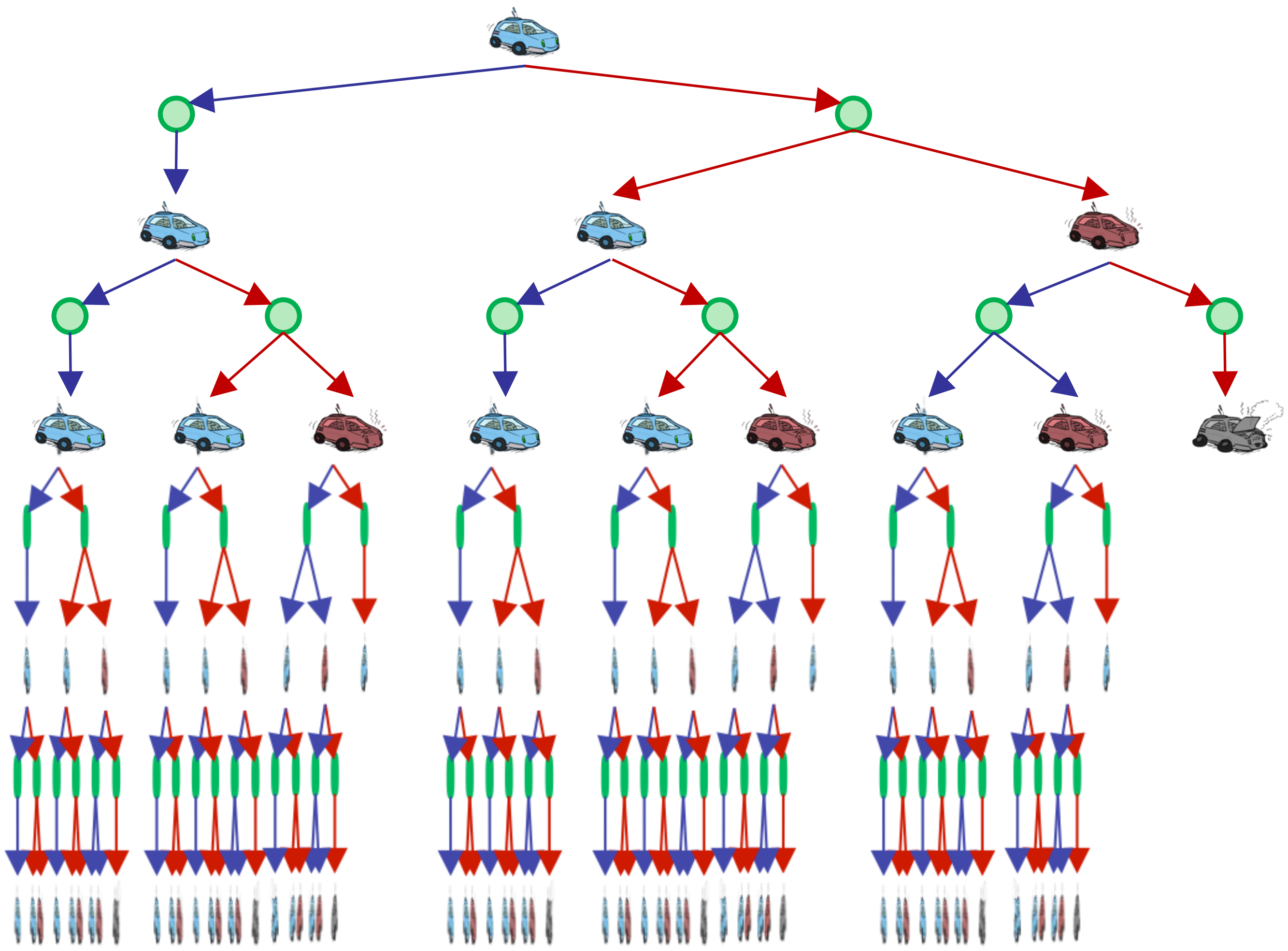


Explosion rapide de l’arbre



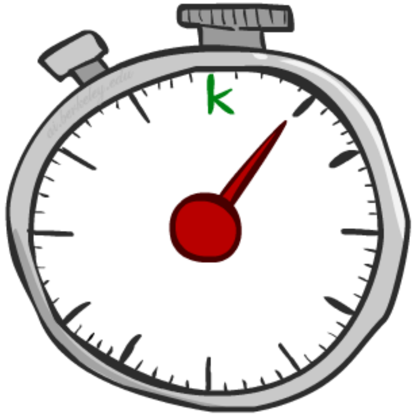
Racing Search Tree

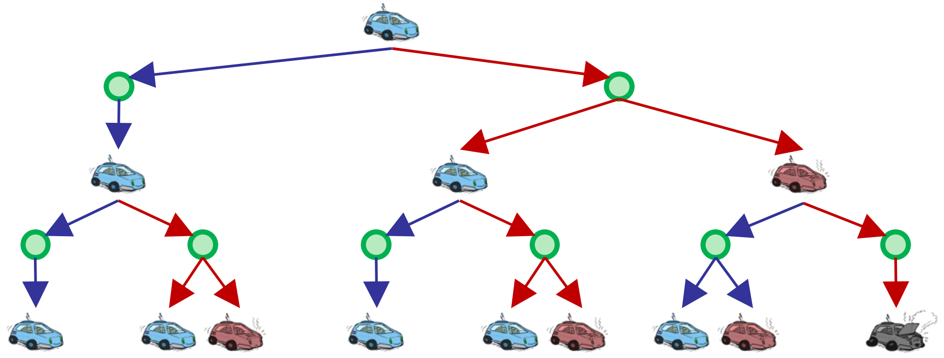
- Calcul répétitif avec Expectimax!
- Problème: les Etats sont répétés
  - Idée: calculer la valeur d'un noeud un seule fois.
- Problème: arbre allant à l'infini
  - Idée: Recherche limitée en profendeur.

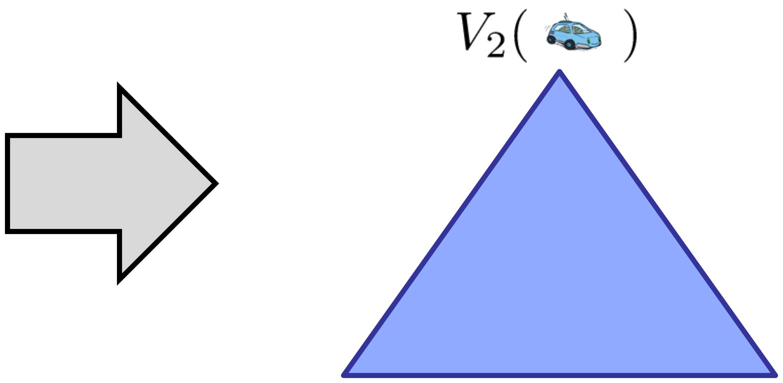


Valeurs limitées en temps

- Idée clé: valeurs limités en temps.
- Definir $V_k(s)$ comme la valeuroptimale de s, si le jeu se termine forcémenet après **k** iteration.
  - Réaliser une recherché expectimax limitée en profendeur.

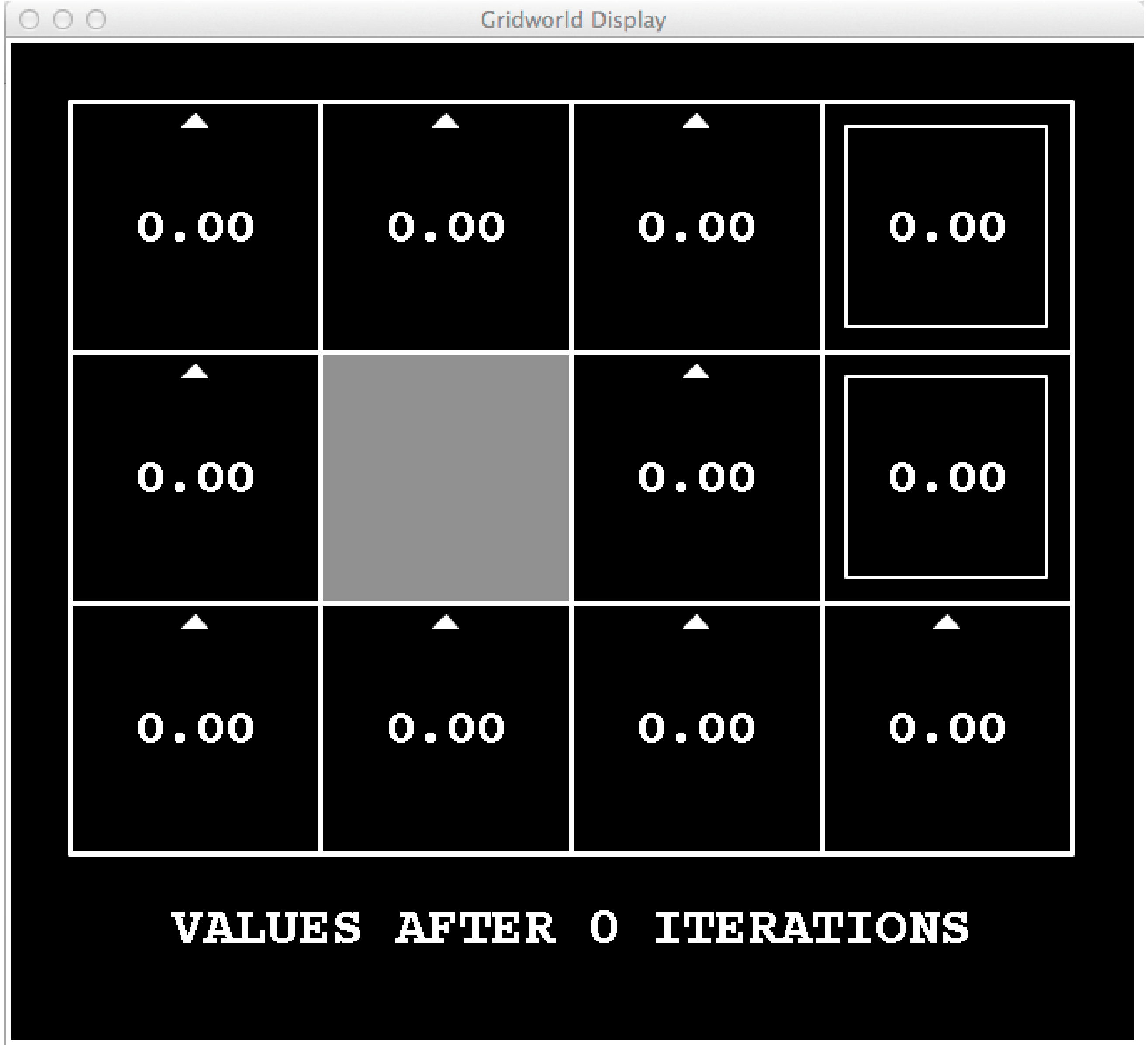






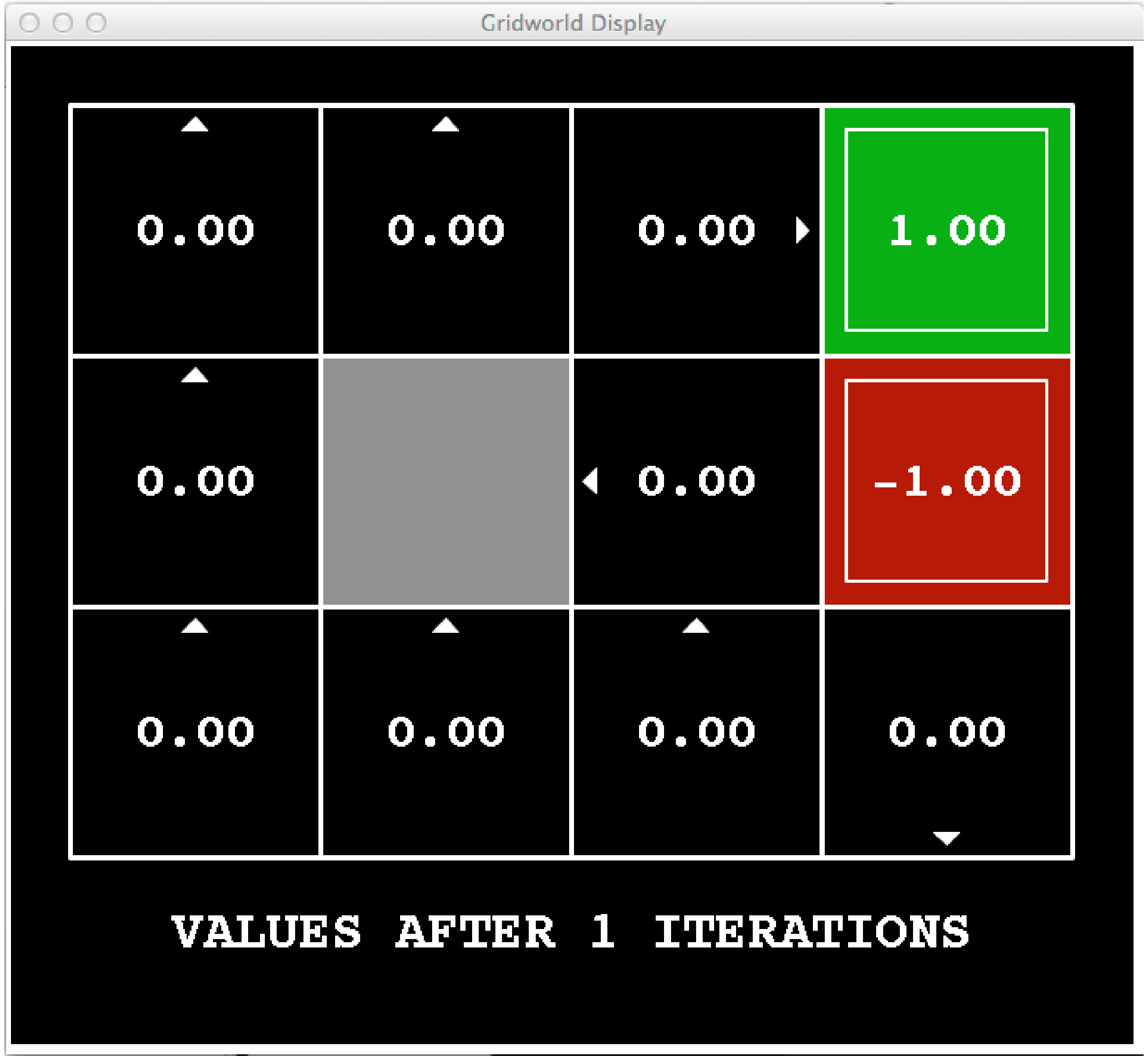
[Demo – time-limited values (L8D6)]

k=0



Bruit = 0.2
Remise = 0.9
Récompense vie = 0

k=1



Bruit = 0.2
Remise = 0.9
Récompense vie = 0

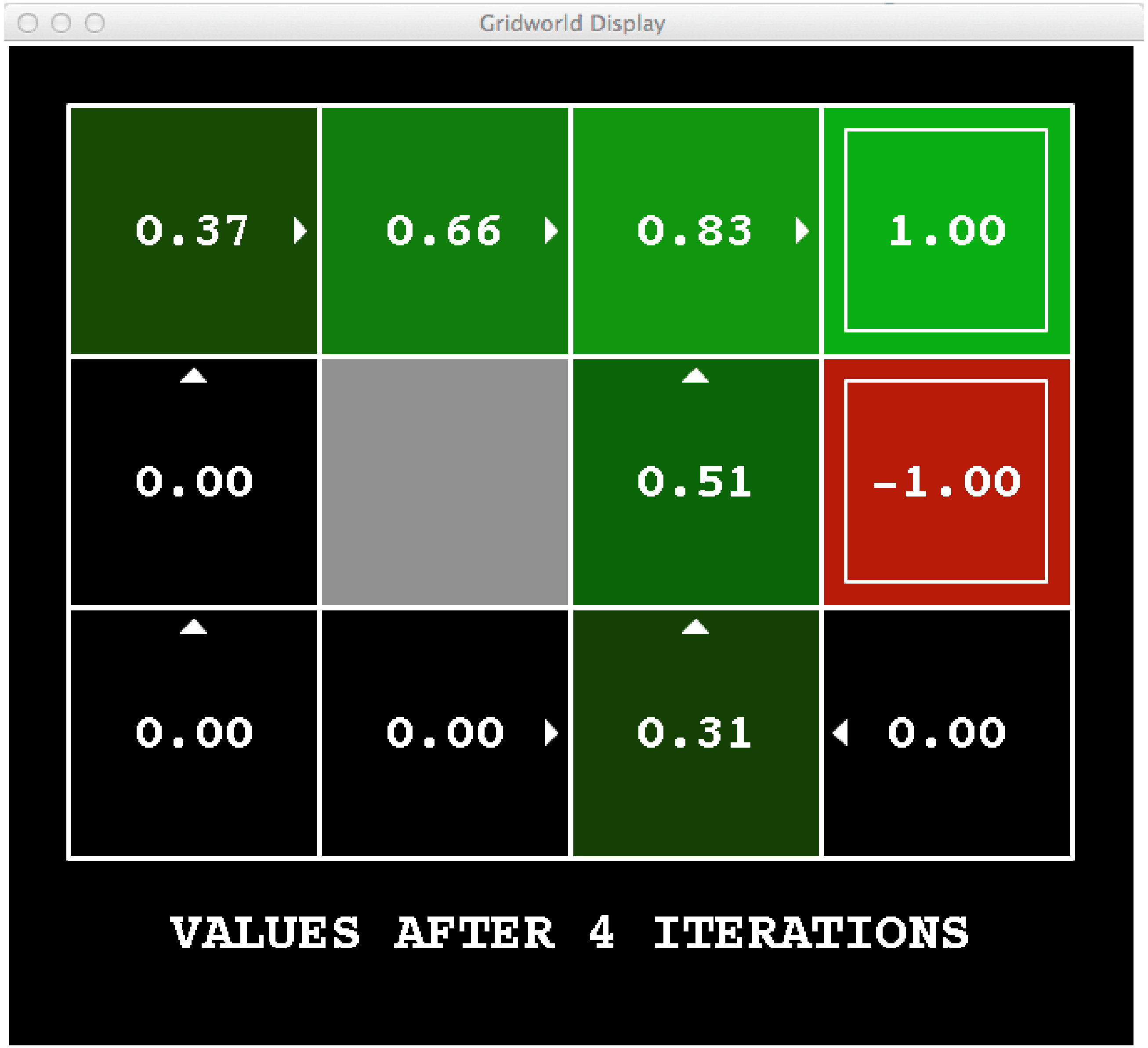
k=2

Bruit= 0.2
Remise = 0.9
Récompense vie = 0

k=3

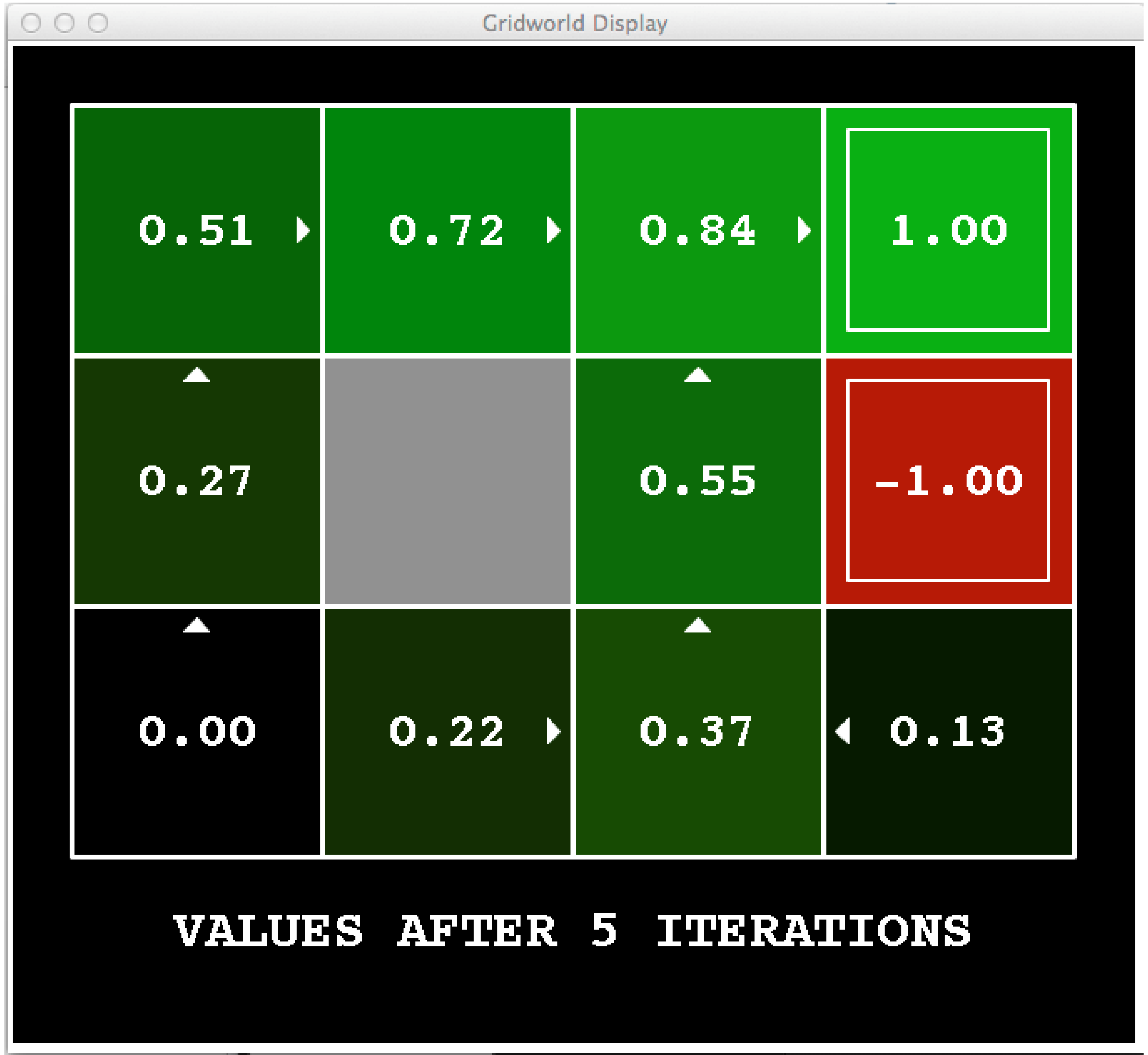
Bruit= 0.2
Remise = 0.9
Récompense vie = 0

k=4



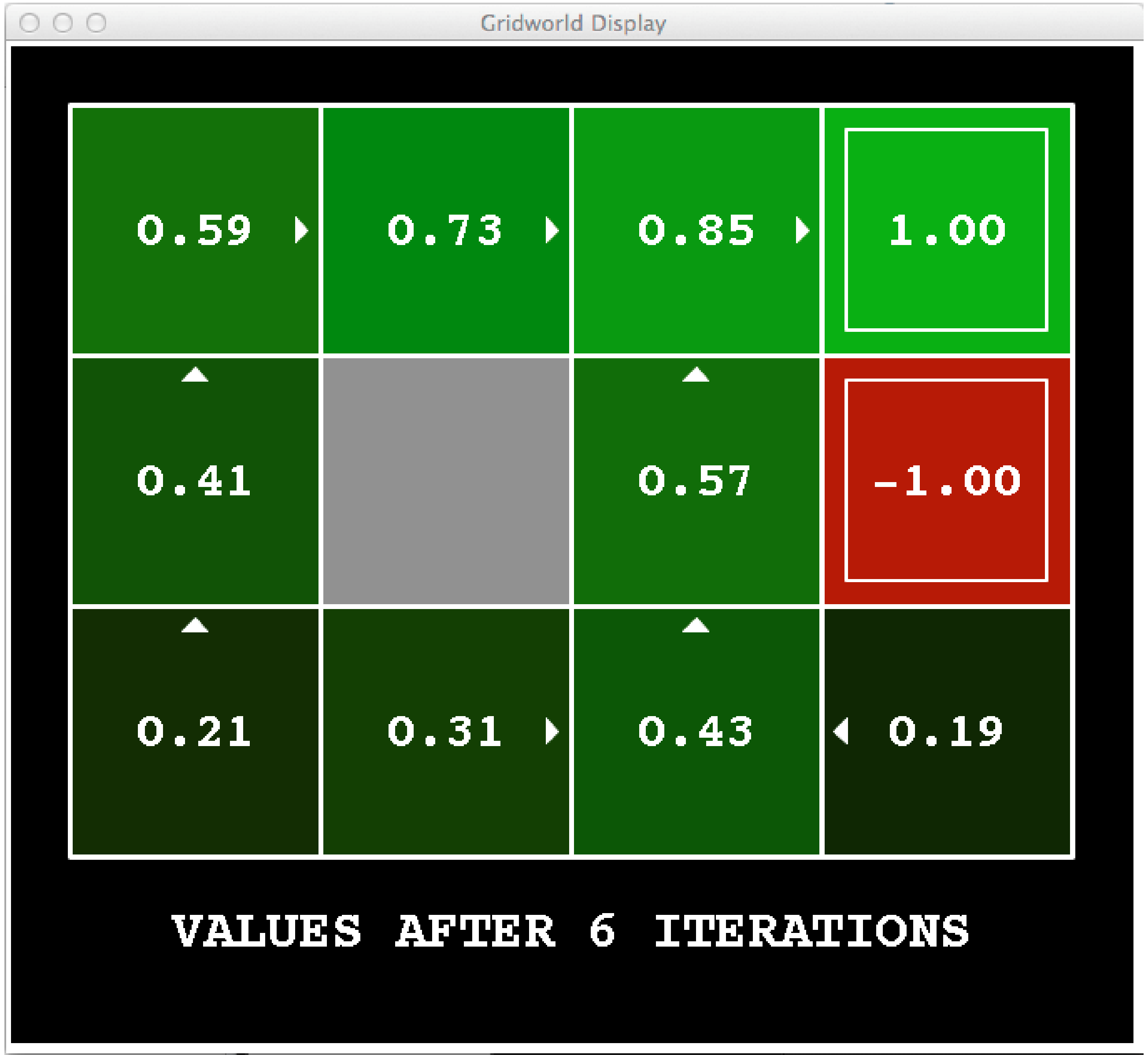
Bruit = 0.2
Remise = 0.9
Récompense vie = 0

k=5



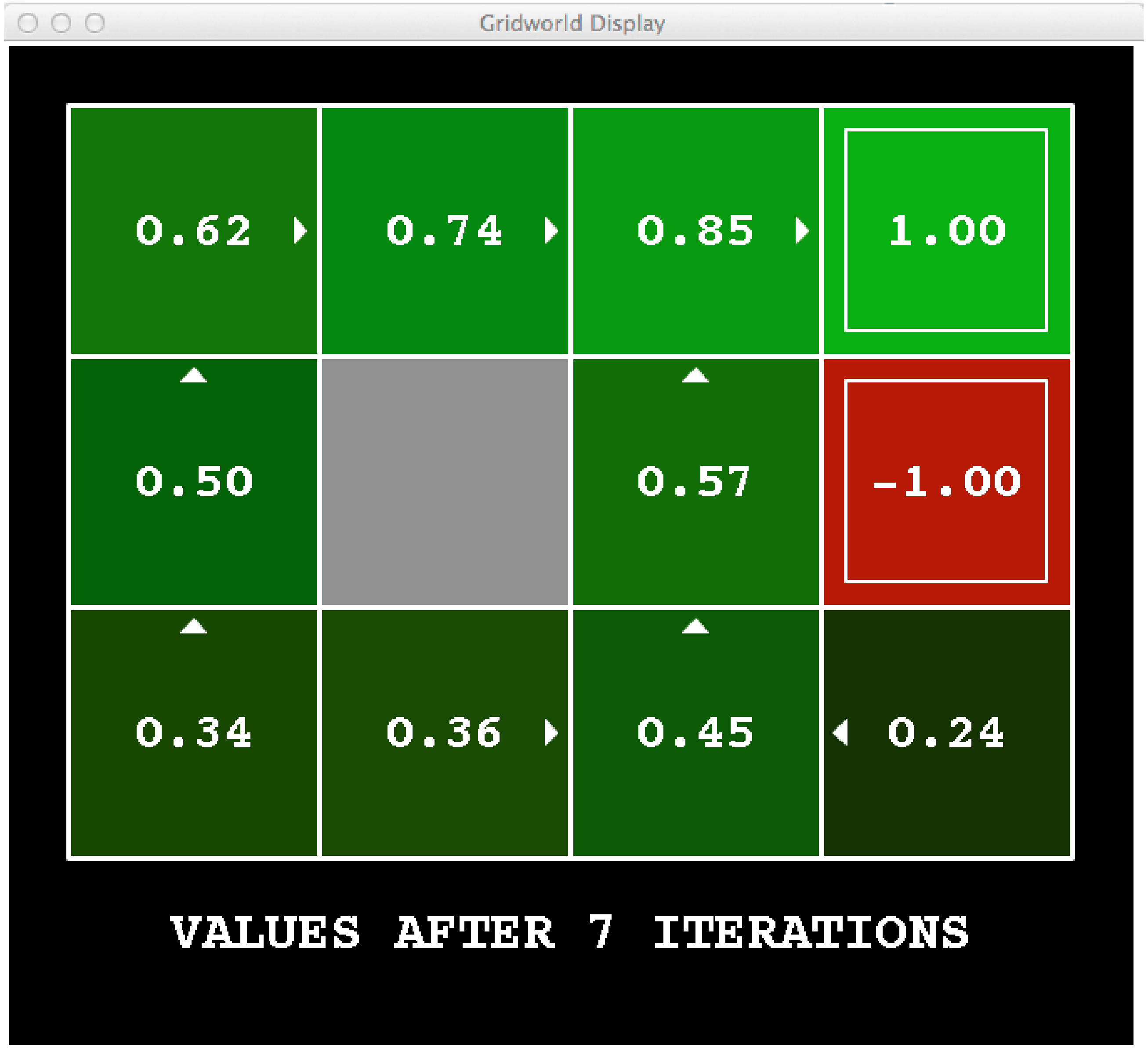
Bruit= 0.2
Remise = 0.9
Récompsense vie = 0

k=6



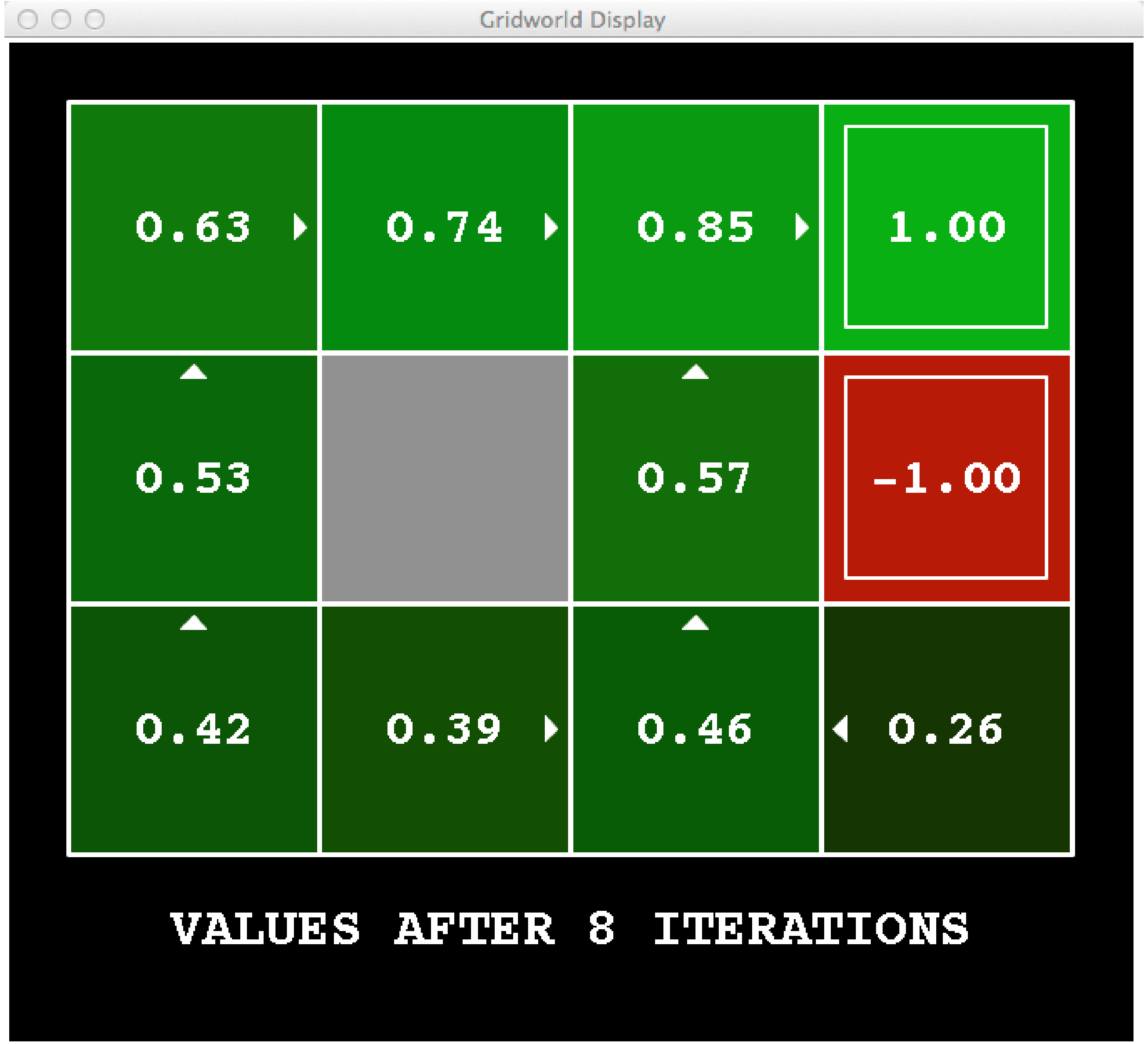
Noise = 0.2
Discount = 0.9
Living reward = 0

k=7



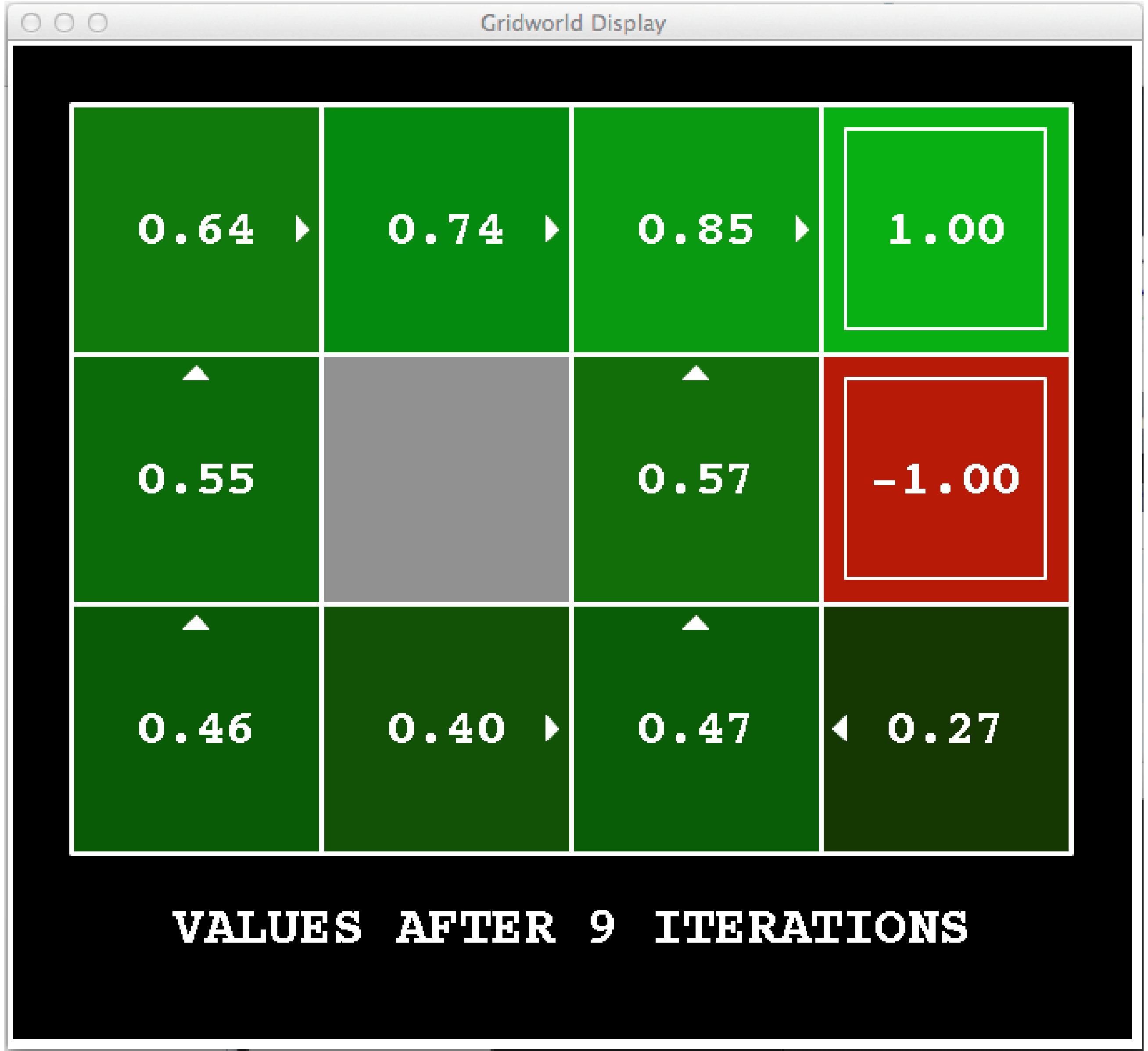
Bruit = 0.2
Remise = 0.9
Récompense vie = 0

k=8



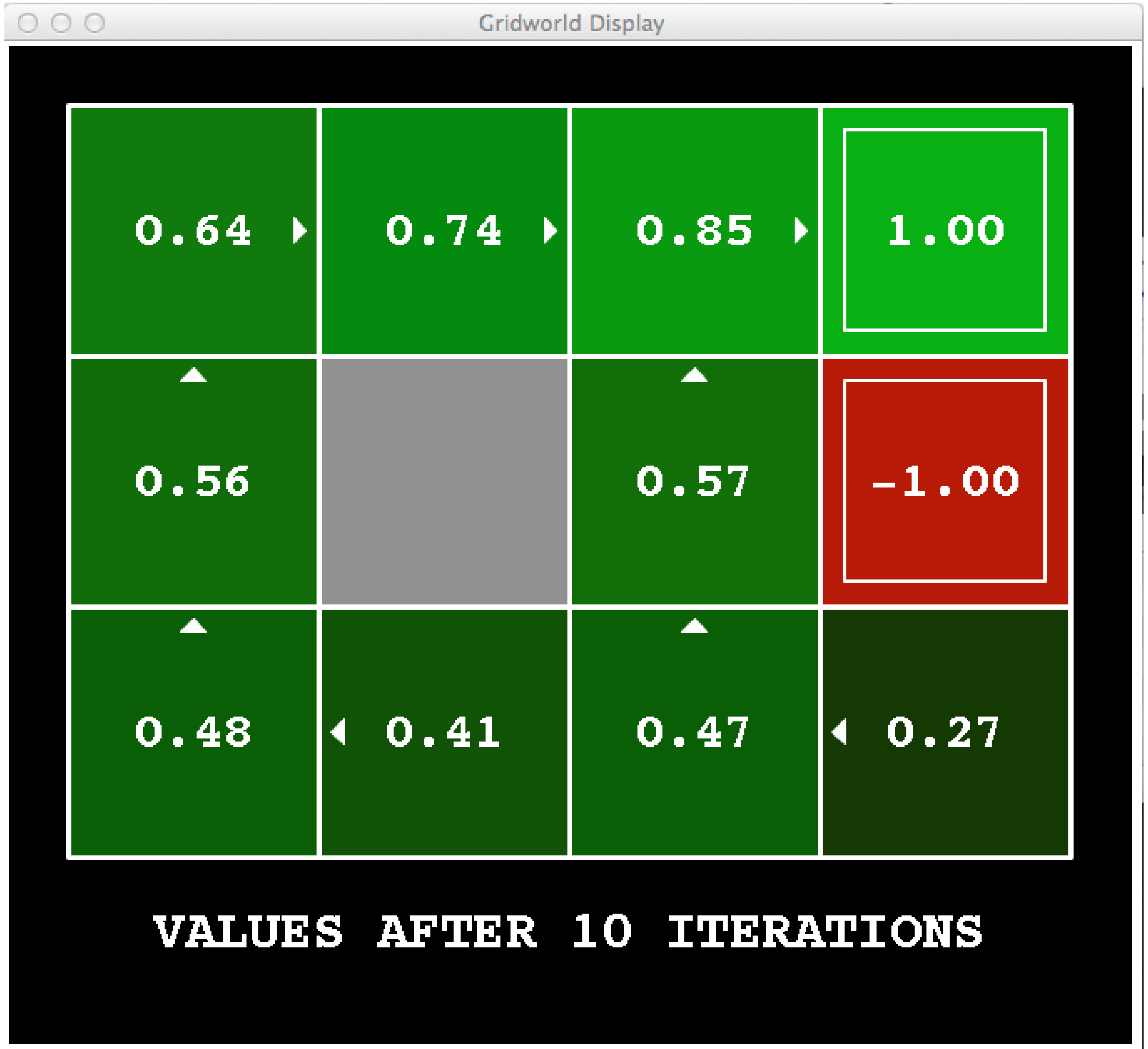
Noise = 0.2
Discount = 0.9
Living reward = 0

k=9



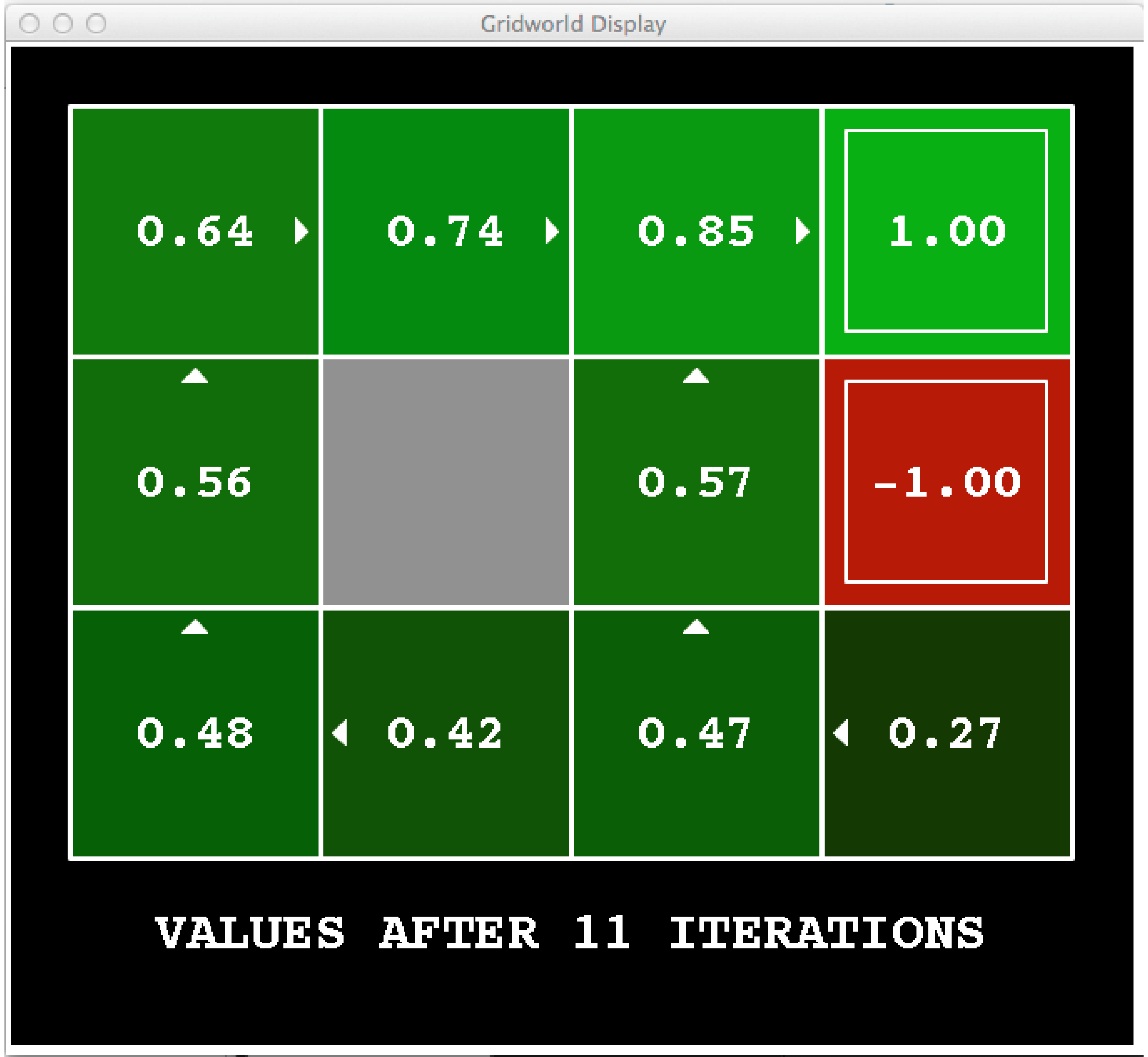
Bruit = 0.2
Remise = 0.9
Récompense vie = 0

k=10



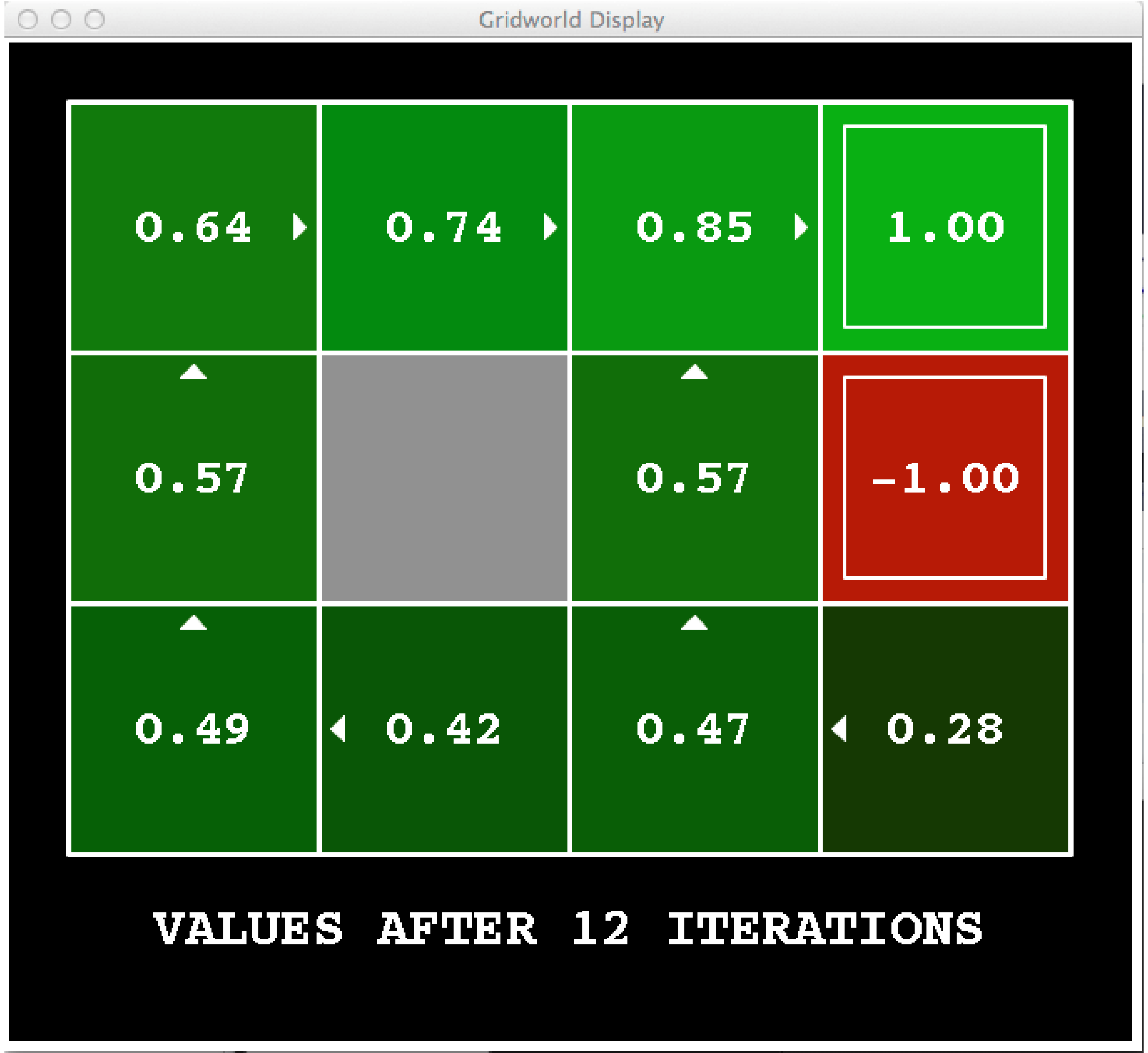
Noise = 0.2
Discount = 0.9
Living reward = 0

k=11



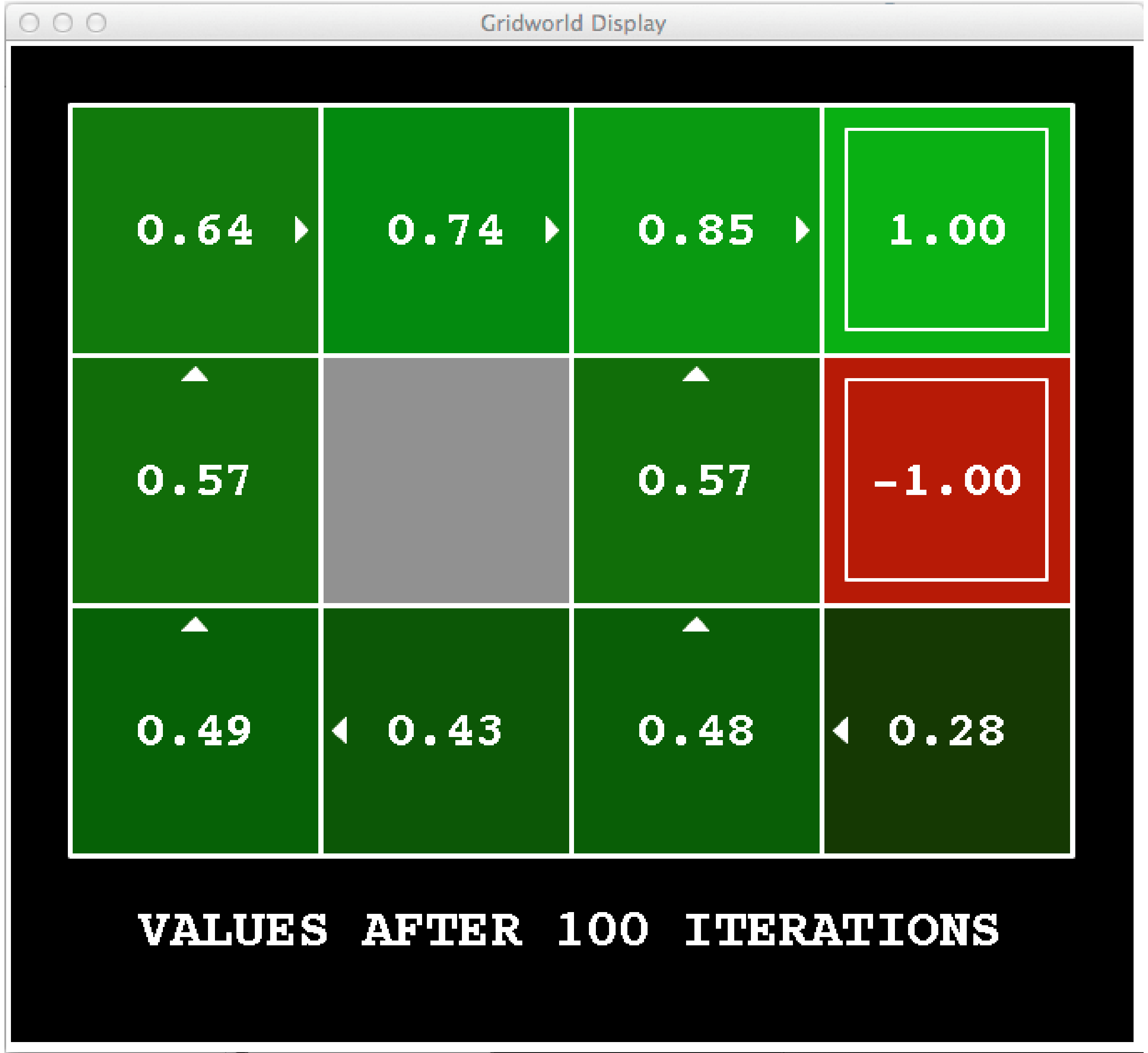
Bruit= 0.2
Remise = 0.9
Récompense vie = 0

k=12



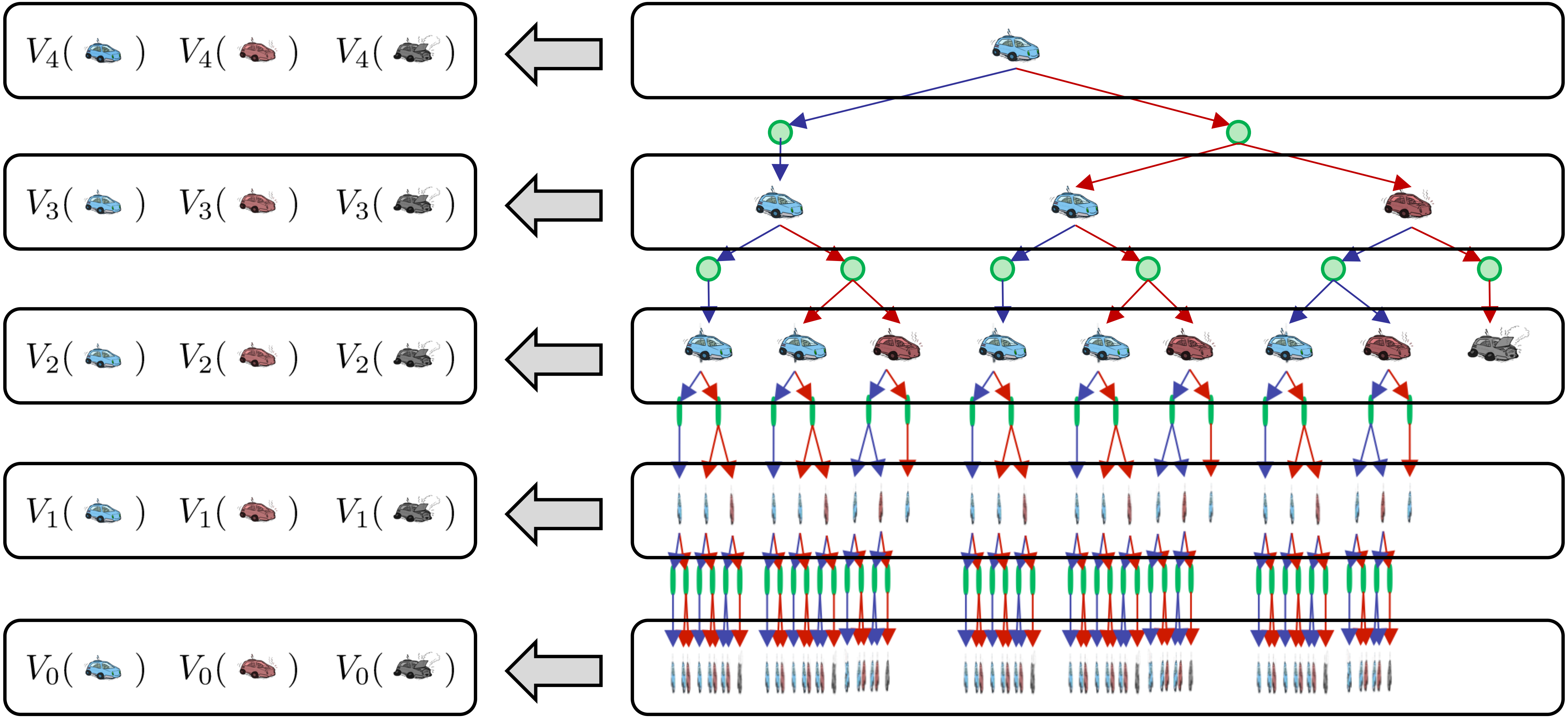
Bruit = 0.2
Remise = 0.9
Récompense vie = 0

k=100

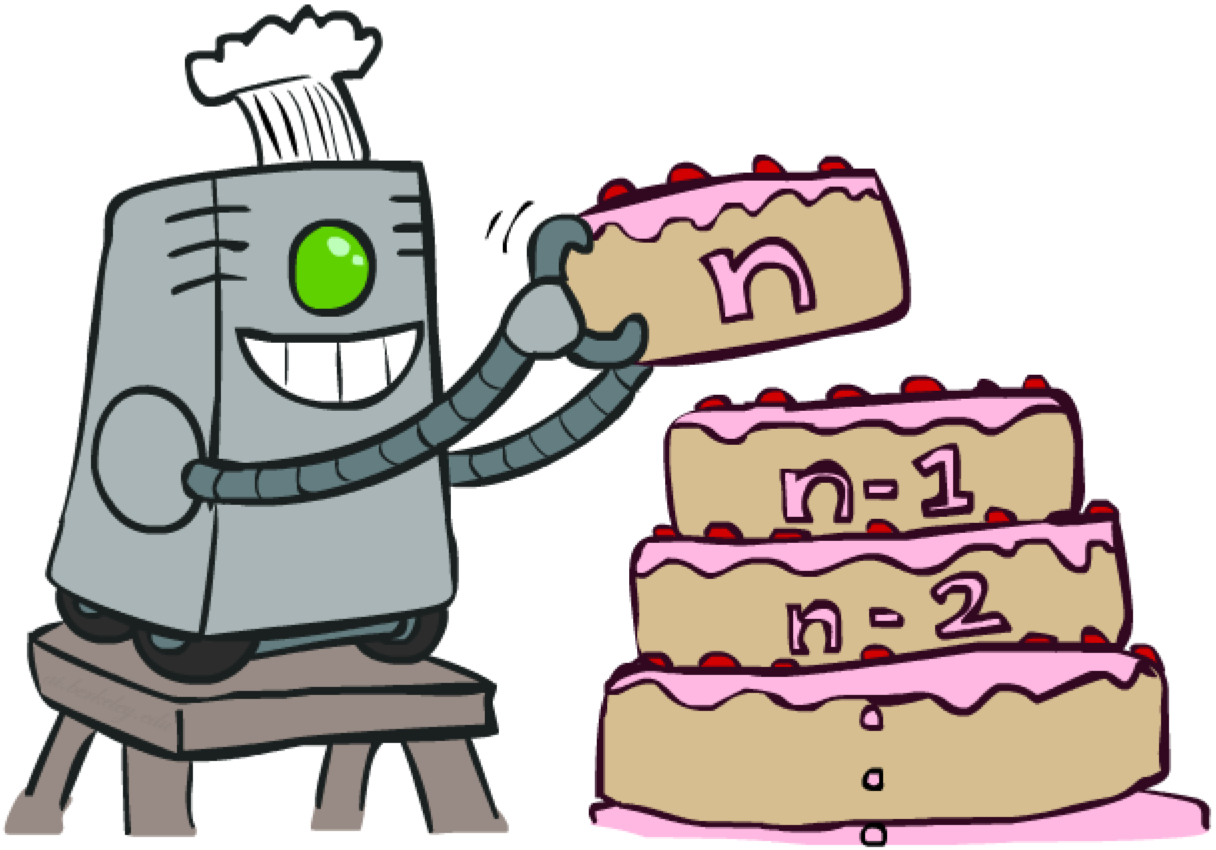


Noise = 0.2
Discount = 0.9
Living reward = 0

Calcul des valeurs limitées en temps



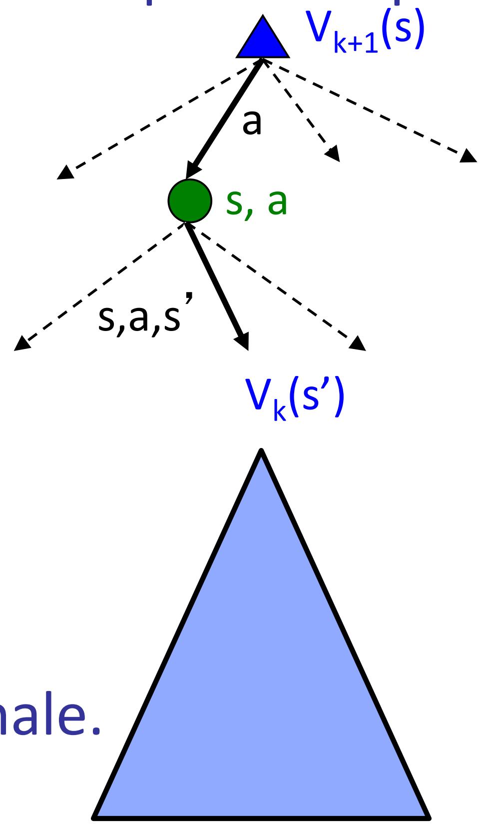
Itération de la valeur



Itération de la valeur

- Commencer avec $V_0(s) = 0$: aucune iteration.
- Etant donnée un vecteur $V_k(s)$ des valeurs, réaliser un **pli d'**expectimax pour chaque état:

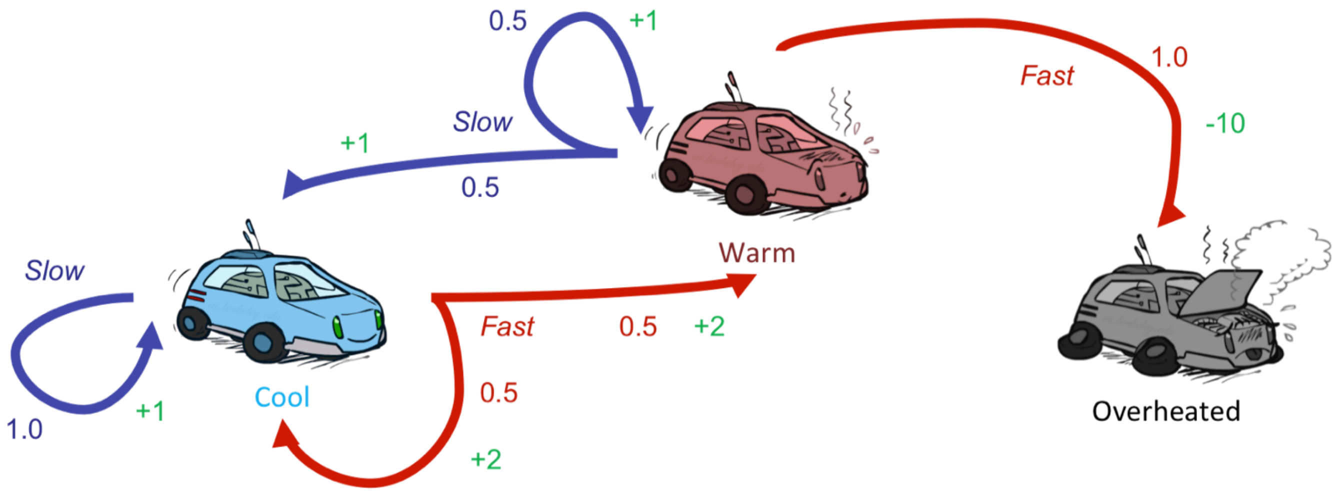
$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_k(s') \right]$$



- Répéter jusqu'à convergence.
- Complexityé de chaque iteration est : $O(S^2A)$
- Theorème: Itération de la valeur converge vers une stratégie optimale.
  - Idée basique: Les valeurs convergent vers leurs valeur optimaux.
  - Problème: Stratégie souvent converge **avant** les valeurs.

Example: Value Iteration

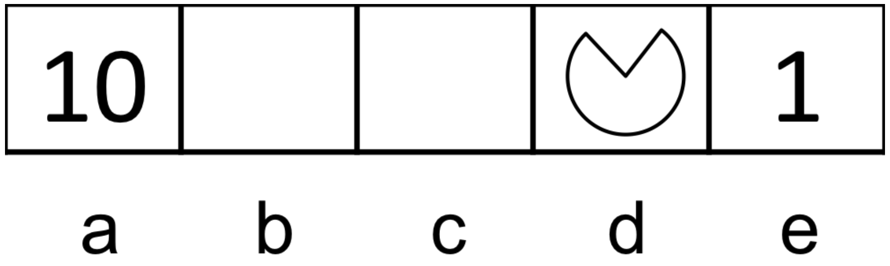
V_2	3.5	2.5	0
V_1	2	1	0
V_0	0	0	0



On considère le cas sans remise!

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

Quiz: Itération de la valeur



On considère que toutes les actions sont réussies sans remise. Calculer les valeurs suivantes:

- $V_0(d)$
- $V_1(d)$
- $V_2(d)$
- $V_3(d)$
- $V_4(d)$
- $V_5(d)$