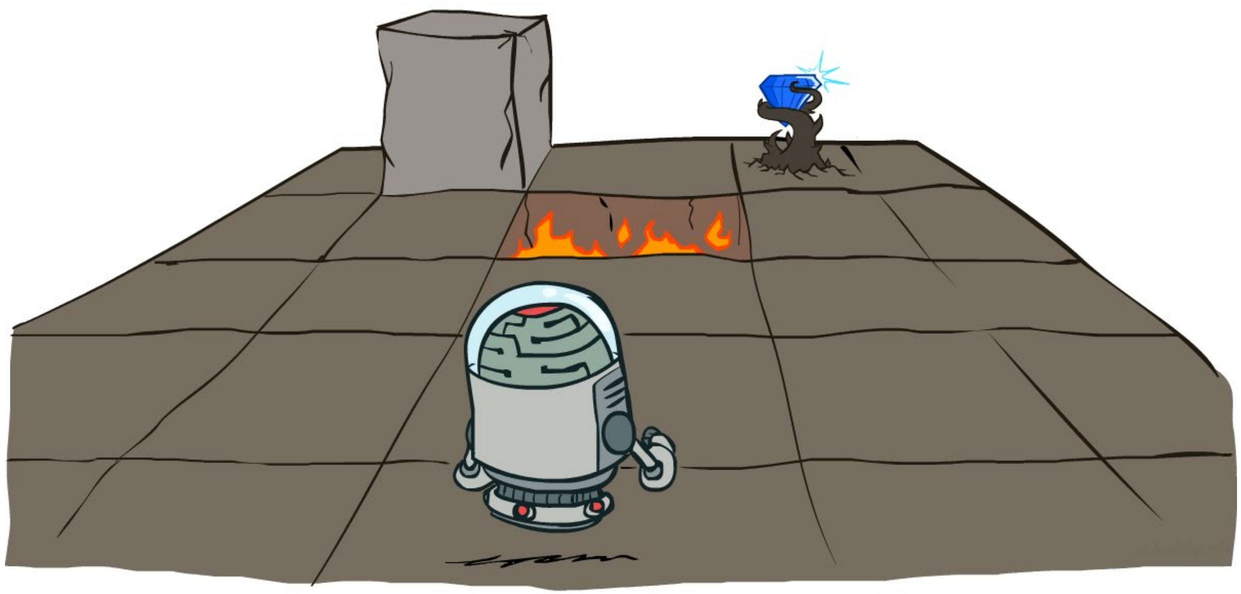
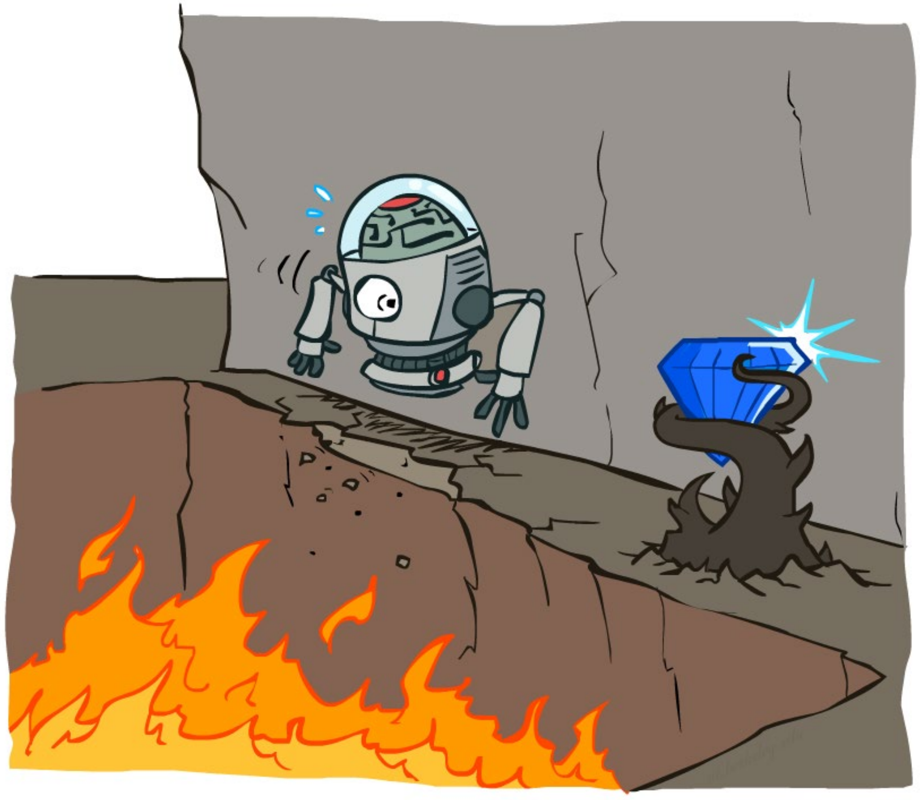
人工智能导论

马可夫决策过程

Markov Decision Processes

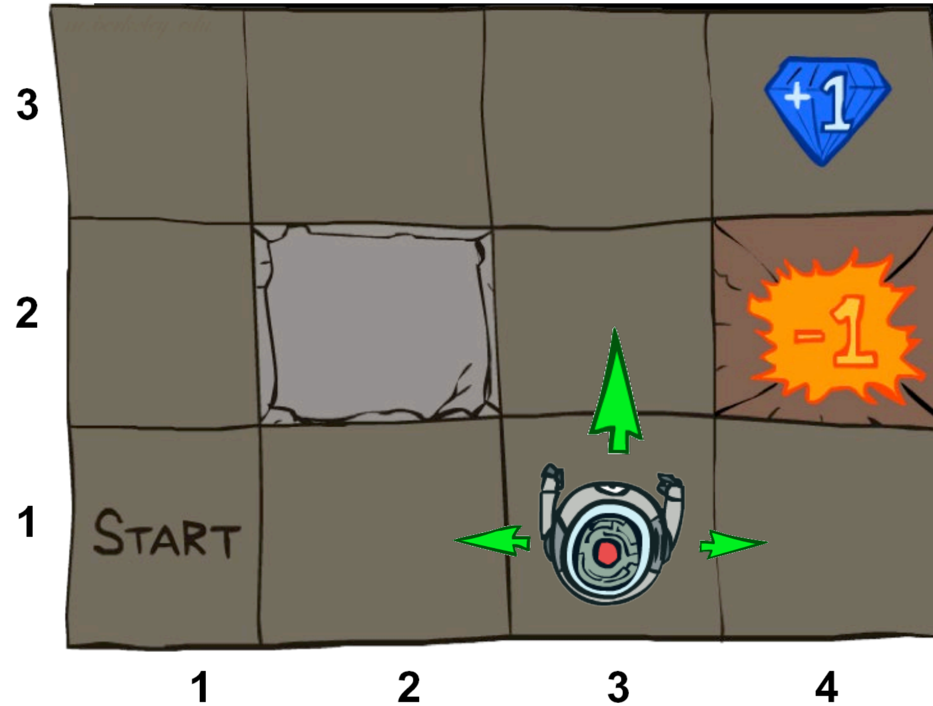


非确定性搜索
Non-Deterministic Search



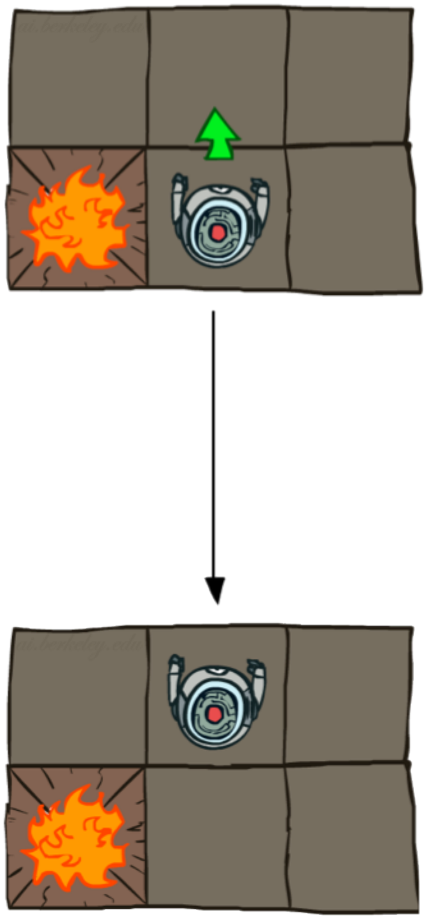
举例：Grid World

- 一个像迷宫的问题
- 粗糙的行动：行动不总是获得预先计划的结果
- 智能体会在每个时刻获得奖赏（得分）
  - 每时刻步长，少量的 “生存 living reward” 奖赏（也可能是负值）
  - 最大的奖赏分值在最后时刻（执行退出行动时）（可能是奖赏也可能是惩罚）
- 目标：最大化总的奖赏得分

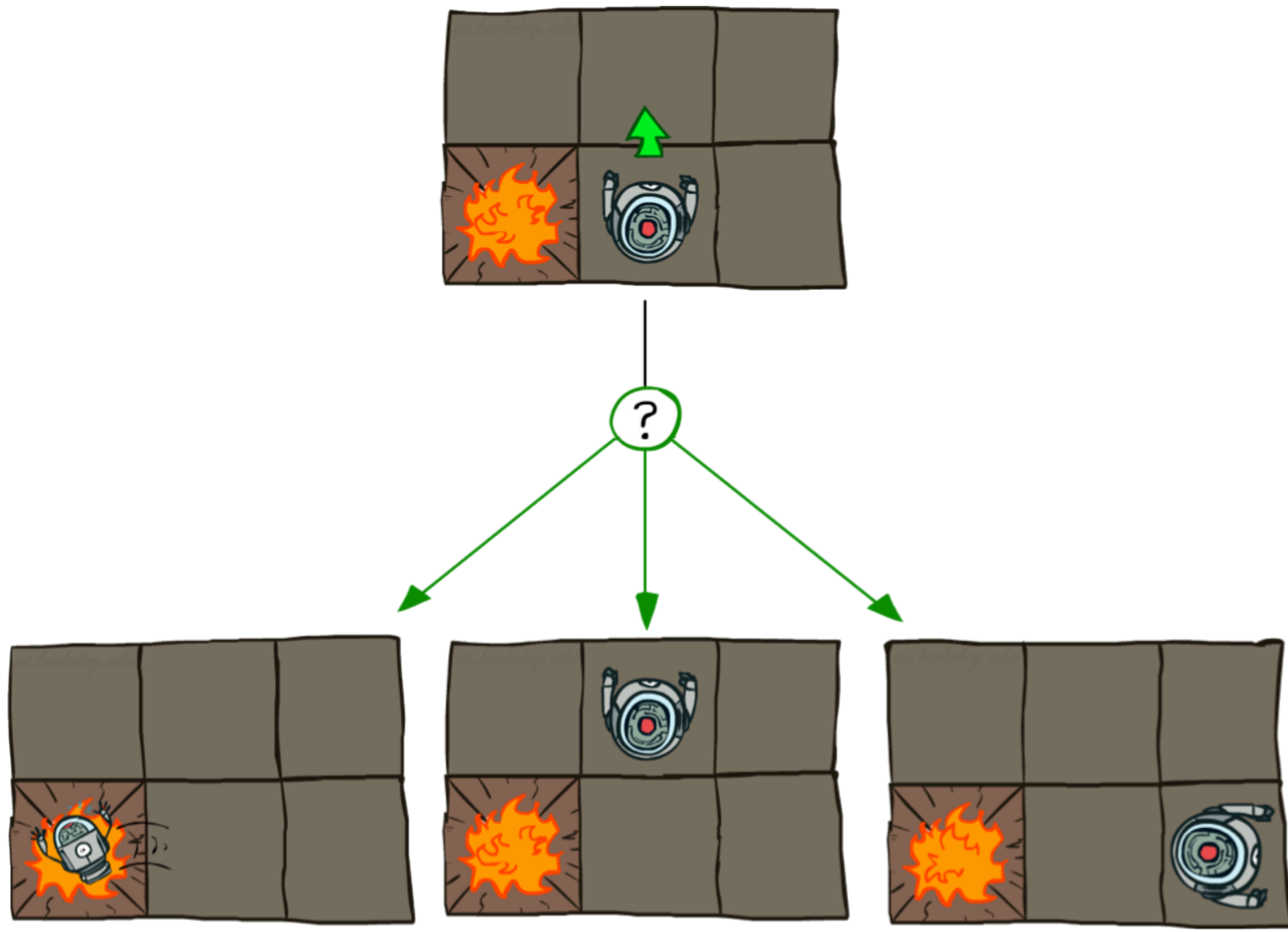


Grid World 行动

确定的 Grid World

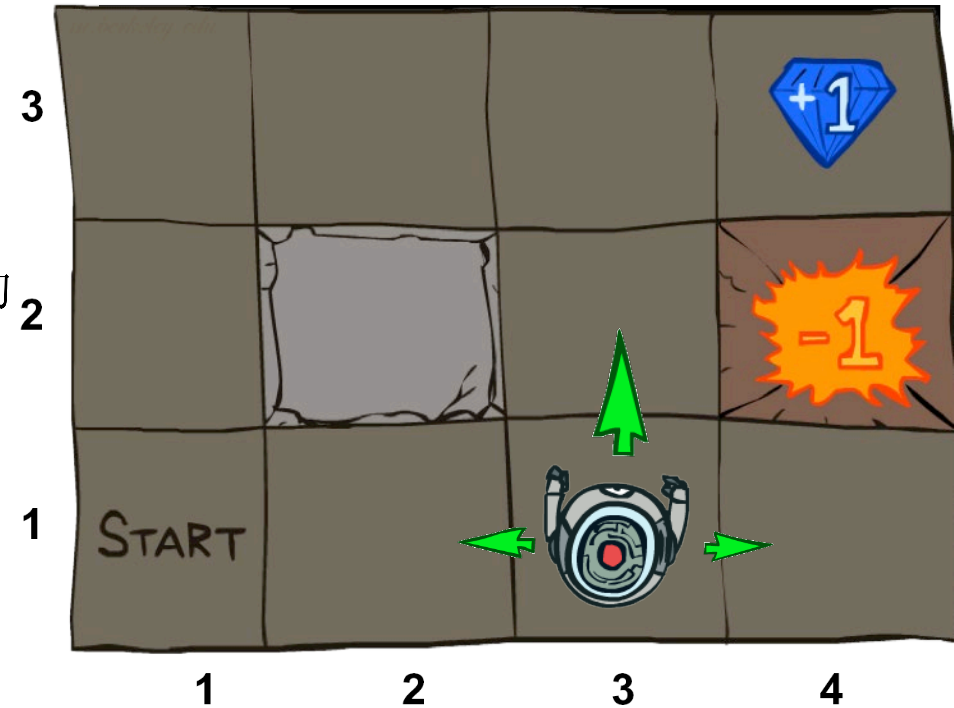


随机的(不确定的)
Grid World



马可夫决策过程(MDP)

- 一个MDP 由以下定义:
  - 状态集合 s ∈ S
  - 行动集合 a ∈ A
  - 转移函数 T(s, a, s’)
    - 从状态 s 采取行动 a 到达 s’ 的概率，即 P(s’ | s, a)
    - 也叫做动态模型
  - 奖赏函数 R(s, a, s’)
    - 有时只是 R(s) or R(s’)
  - 一个开始状态
  - 一个终点状态 （有可能）
- MDPs 是非确定性搜索问题
  - 一种求解方法是通过期望最大值搜索
  - 我们将看到其他的求解方法



MDPs中的马可夫性质

- "Markov" 通常意味着给定当前状态，未来状态独立于过去状态
- 对于马可夫决策过程，"Markov" 的含义是行动的输出结果只取决于当前状态

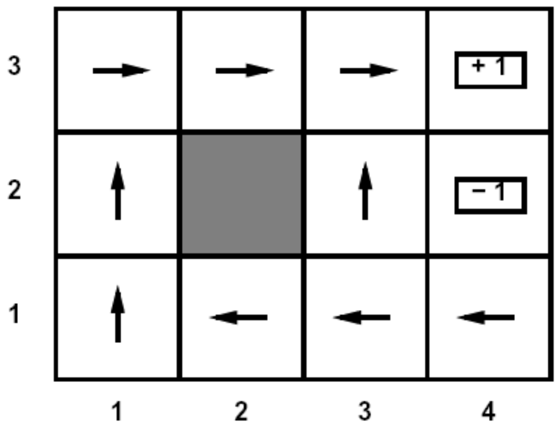
$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, A_{t-1}, \ldots S_0 = s_0)$$
$$=$$
$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t)$$

- 就像在搜索问题里，后继函数只依赖于当前的搜索状态（而不是历史状态）

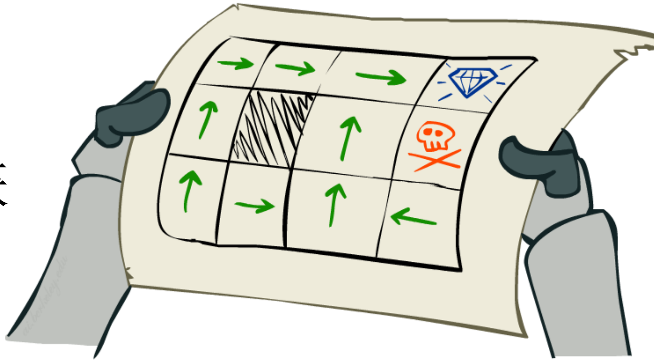
Andrey Markov
(1856-1922)

策略 Policies

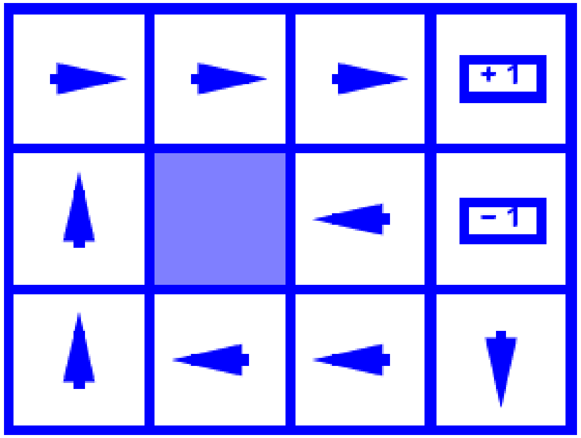
- 在确定性的，单一智能体的，搜索问题里，我们想获得一个最优的 规划 plan，或是一个从开始到目标状态的行动上的序列



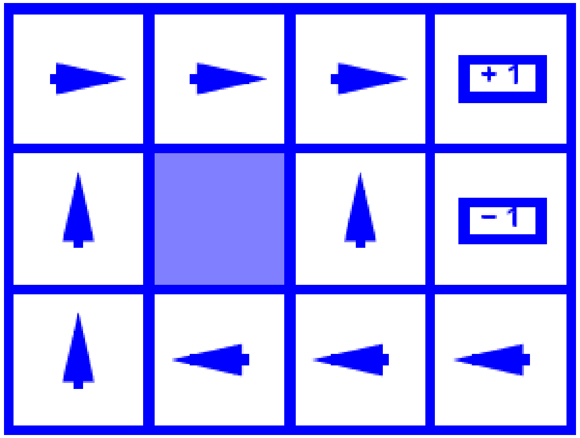
- 对于 MDPs，我们想获得的是一个最优的 策略 policy π*: S → A
  - 一个策略 π 对于每个状态给出了一个行动
  - 一个最优策略是这样一个策略，即遵循它可以获得最大化期望功效值
  - 一个明示的策略定义了一个反射型的智能体



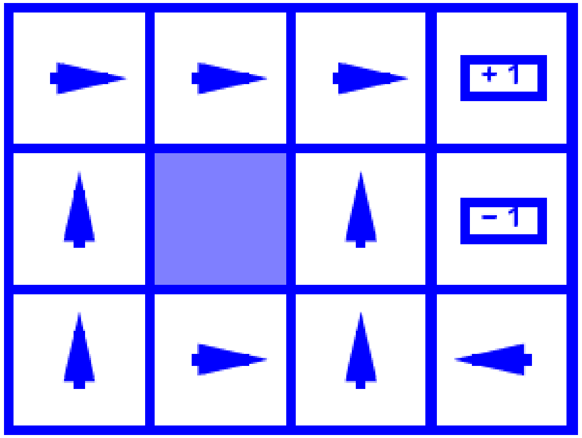
Optimal Policies



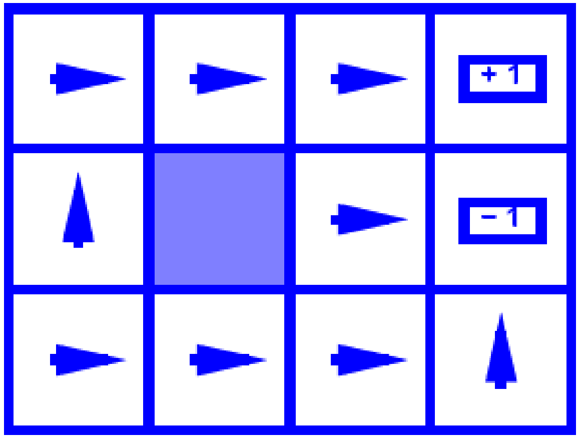
R(s) = -0.01



R(s) = -0.03

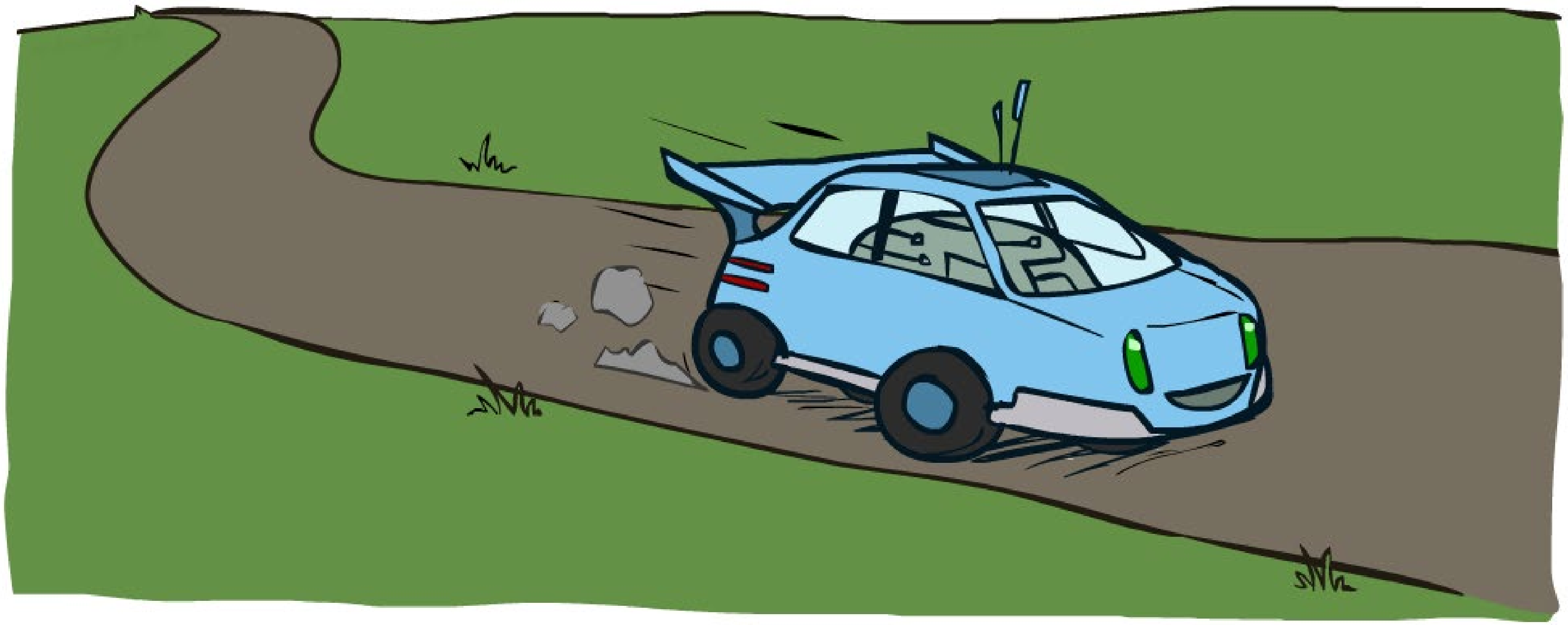


R(s) = -0.4



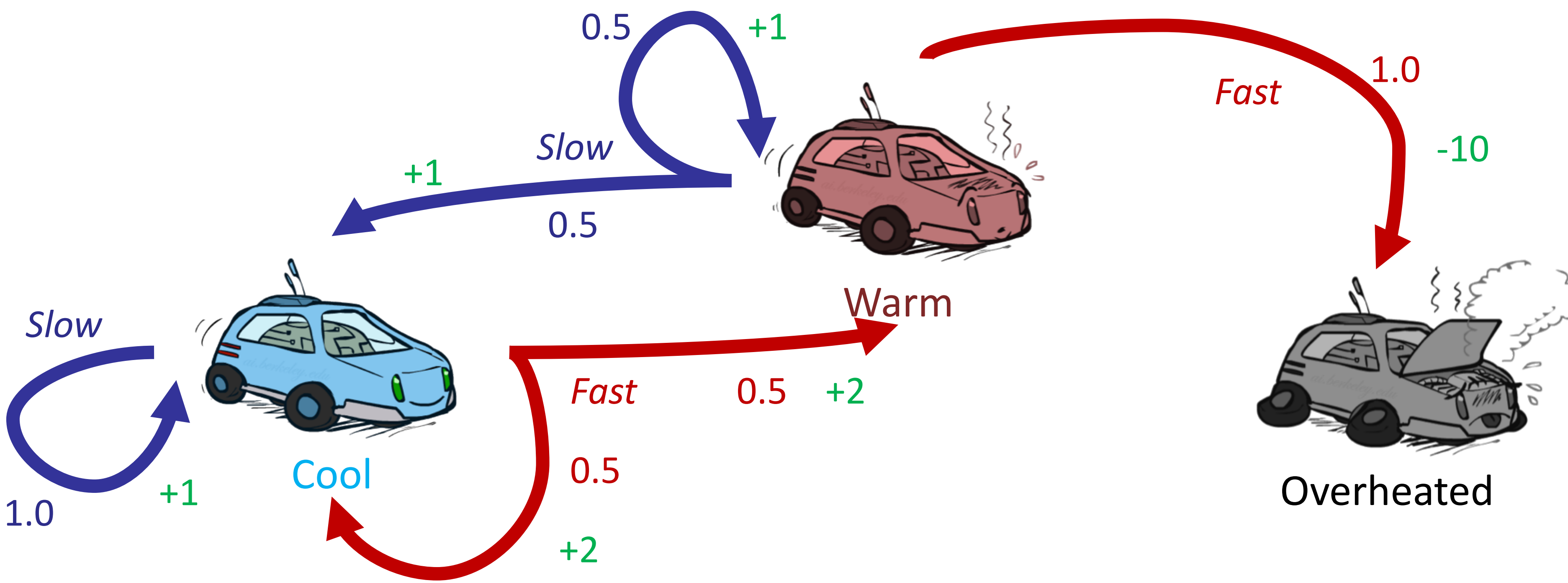
R(s) = -2.0

举例：赛车

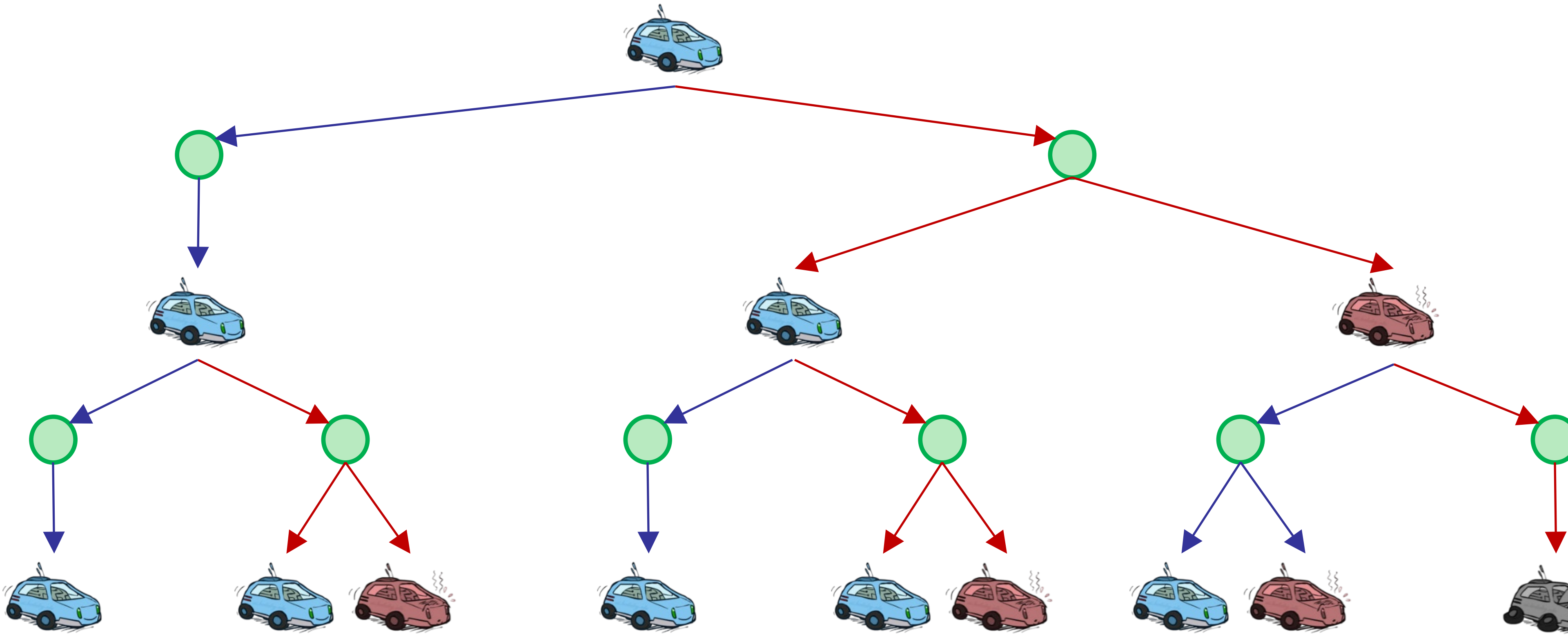


举例：赛车

- 一个机器人汽车想要开的既远又快
- 三个状态: Cool, Warm, Overheated
- 两个行动: *Slow*, *Fast*
- 速度快可以得到双倍的奖赏

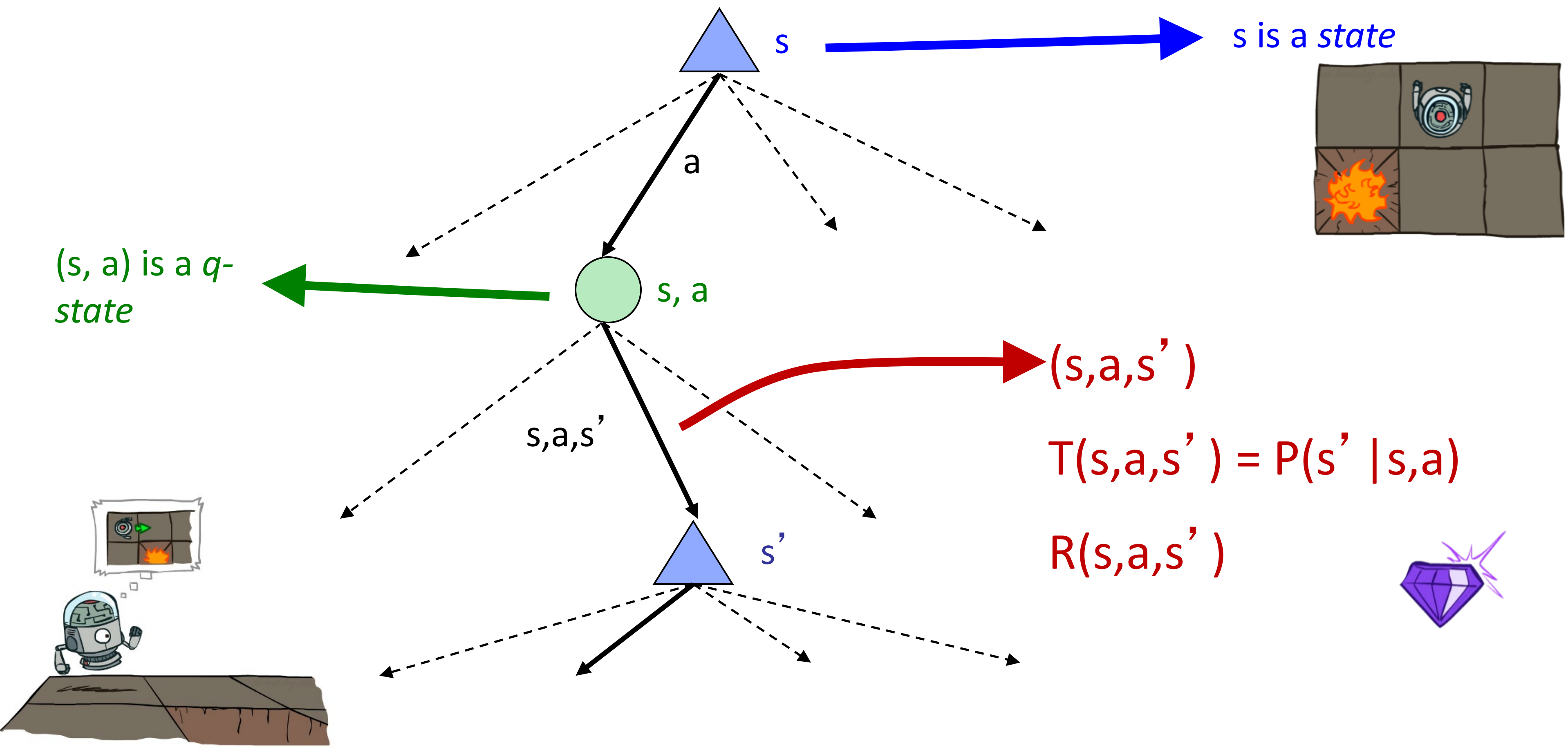


赛车例子里的搜索树

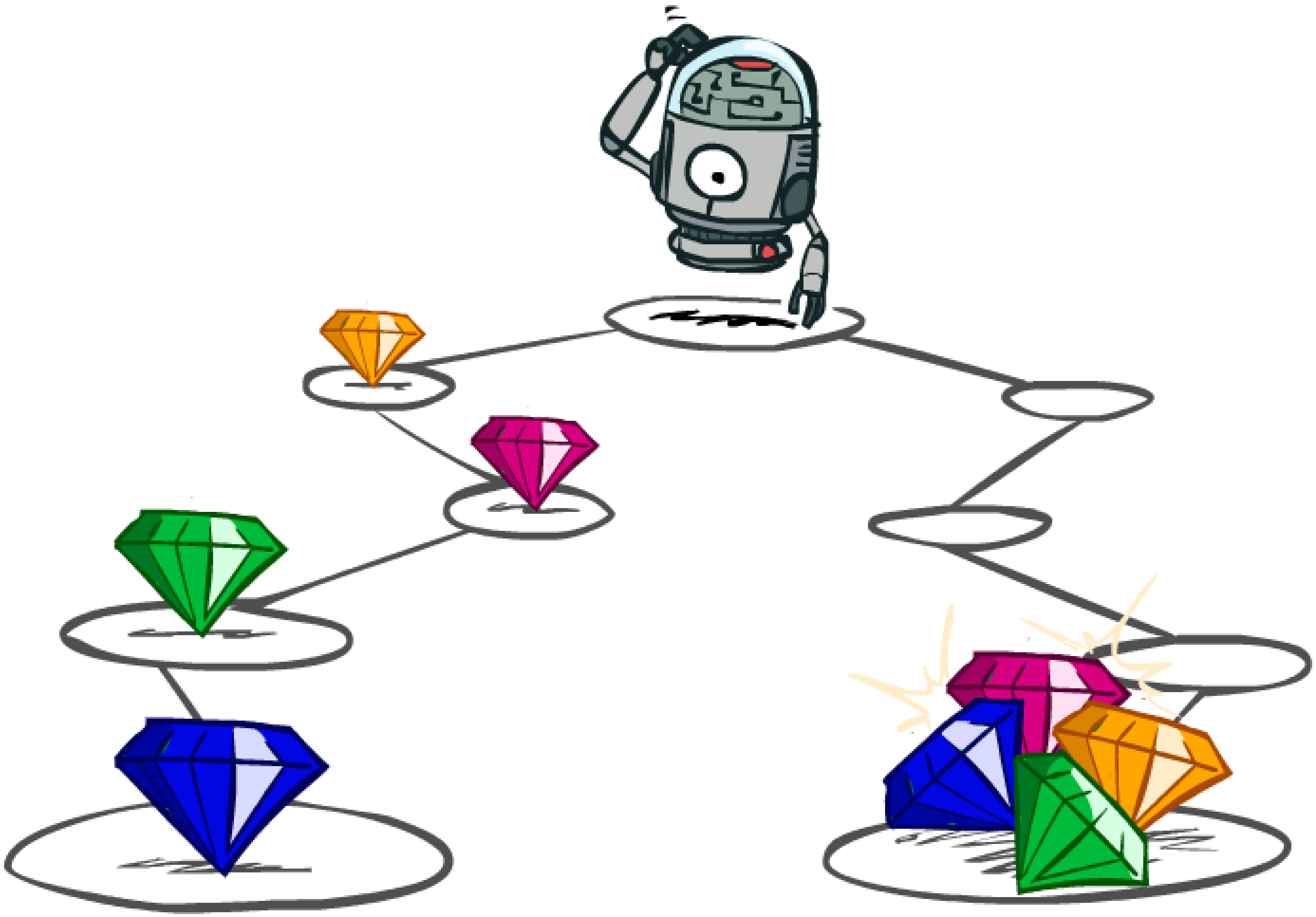


MDP 搜索树

- 一个 类似-期望最大的 搜索树



不同奖赏值序列的选择

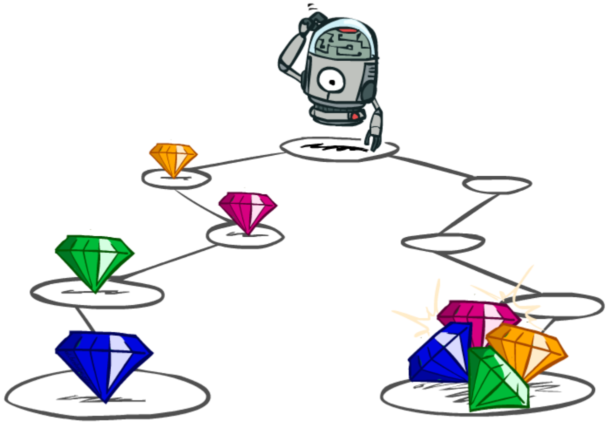


如何选择：不同的奖赏值序列

- 不同选择的偏好，智能体应如何选择?
- 多或是少?

  [1, 2, 2]   or   [2, 3, 4]
- 现在或是以后?

  [0, 0, 1]   or   [1, 0, 0]



奖赏的折扣 Discounting

- 最大化奖赏值的总和
- 同时，趋向于越早获得奖赏越好
- 一种方法是：奖赏值随时间成指数递减

1

现在的价值

γ

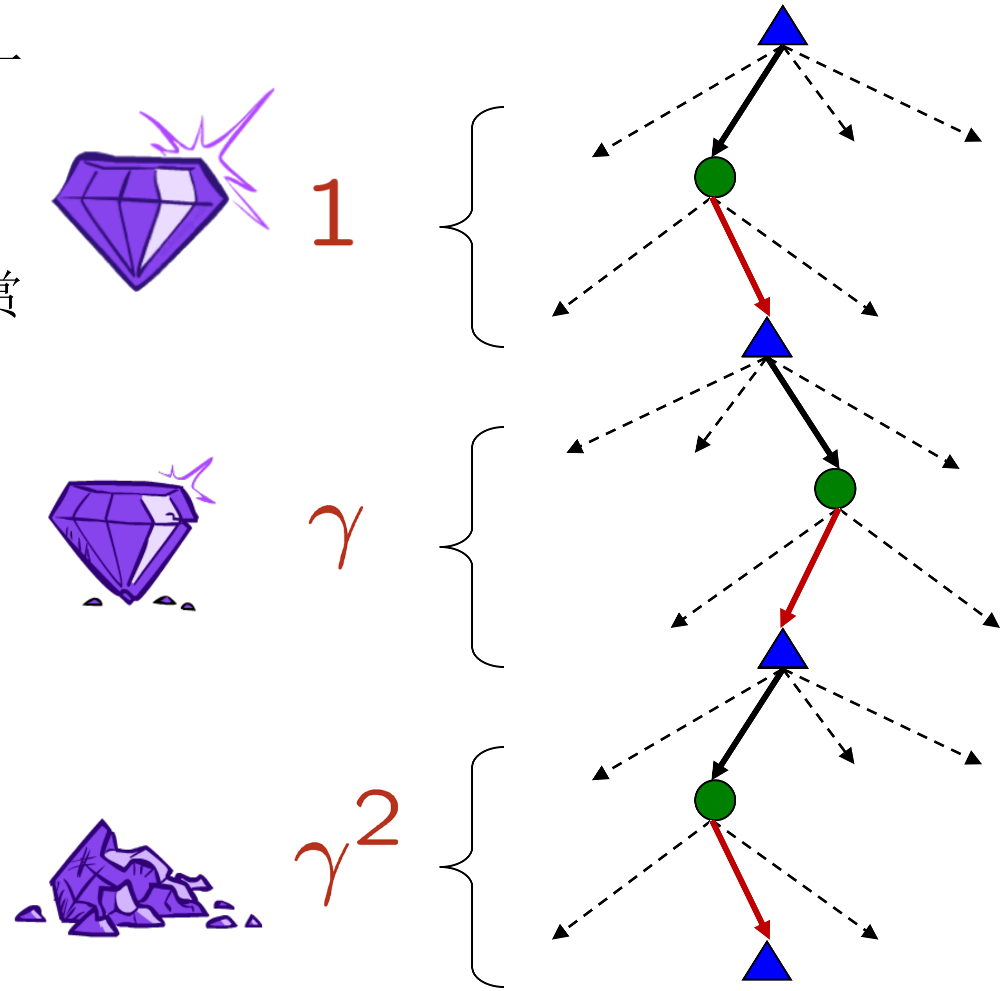
下一步的

γ^2

两步以后的

折扣的奖赏

- 如何打折?
  - 在搜索树上每下降一层，乘上一次折扣系数
- 为什么要进行折扣?
  - 为了实现倾向于更早地获得奖赏
  - 还能帮助算法收敛
- 例如：折扣系数 0.5
  - U([1,2,3]) = 1*1 + 0.5*2 + 0.25*3
  - U([1,2,3]) < U([3,2,1])



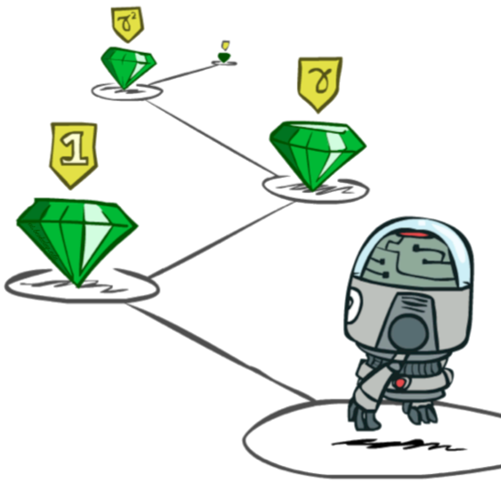
稳定的优先权
Stationary Preferences

- 定理：如果我们假设 稳定的优先权：

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

$$\Updownarrow$$

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



- 那么：只有两种方式定义其上的功效值 (utilities)
  - 相加功效值： $U([r_0, r_1, r_2, \ldots]) = r_0 + r_1 + r_2 + \cdots$
  - 折扣的功效值： $U([r_0, r_1, r_2, \ldots]) = r_0 + \gamma r_1 + \gamma^2 r_2 \cdots$

练习：折扣的功效值

10				1
a	b	c	d	e

- 给定:
  - 行动：向东走，向西走，和 退出（退出行动只在退出状态 a, e上有效）
  - 状态转移：确定性的
- Quiz 1: For $\gamma = 1$，最优策略是什么?

10				1

- Quiz 2: For $\gamma = 0.1$，最优策略?

10				1

- Quiz 3: γ 为何值时，在状态d上，向西和向东的行动是一样好的?

无穷大功效值?!

- 问题：如果游戏一直进行下去会怎么样？ 是否我们会得到无穷大的奖赏值?
- 解决方法:
  - 有限的游戏时间：（类似于有限深度的搜索）
    - 游戏终止在一个固定的 T 步长时间后
    - 使用非稳定的策略（π 依赖于所剩游戏时间）
  - 折扣的奖赏：使用 $0 < \gamma < 1$

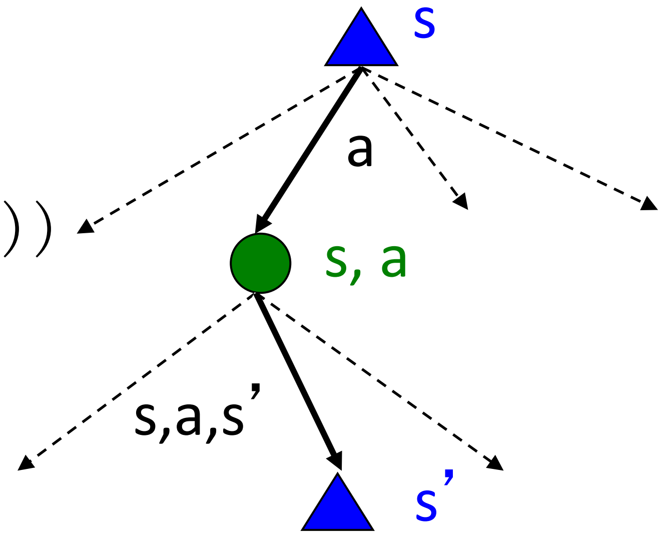
$$U([r_0, \ldots r_\infty]) = \sum_{t=0}^{\infty} \gamma^t r_t \leq R_{\max}/(1-\gamma)$$

$$\sum_{t=0}^{\infty} \gamma^t R_i \leq \sum_{t=0}^{\infty} \gamma^t R_{max} = R_{max} \sum_{t=0}^{\infty} \gamma^t = \frac{R_{max}}{1-\gamma} \text{（几何级数收敛于 } \frac{1}{1-\gamma}\text{）}$$

  - 吸收状态：对于每一个策略都保证，一个终点状态将会被最终达到（像在之前赛车例子里的 “overheated” 状态）

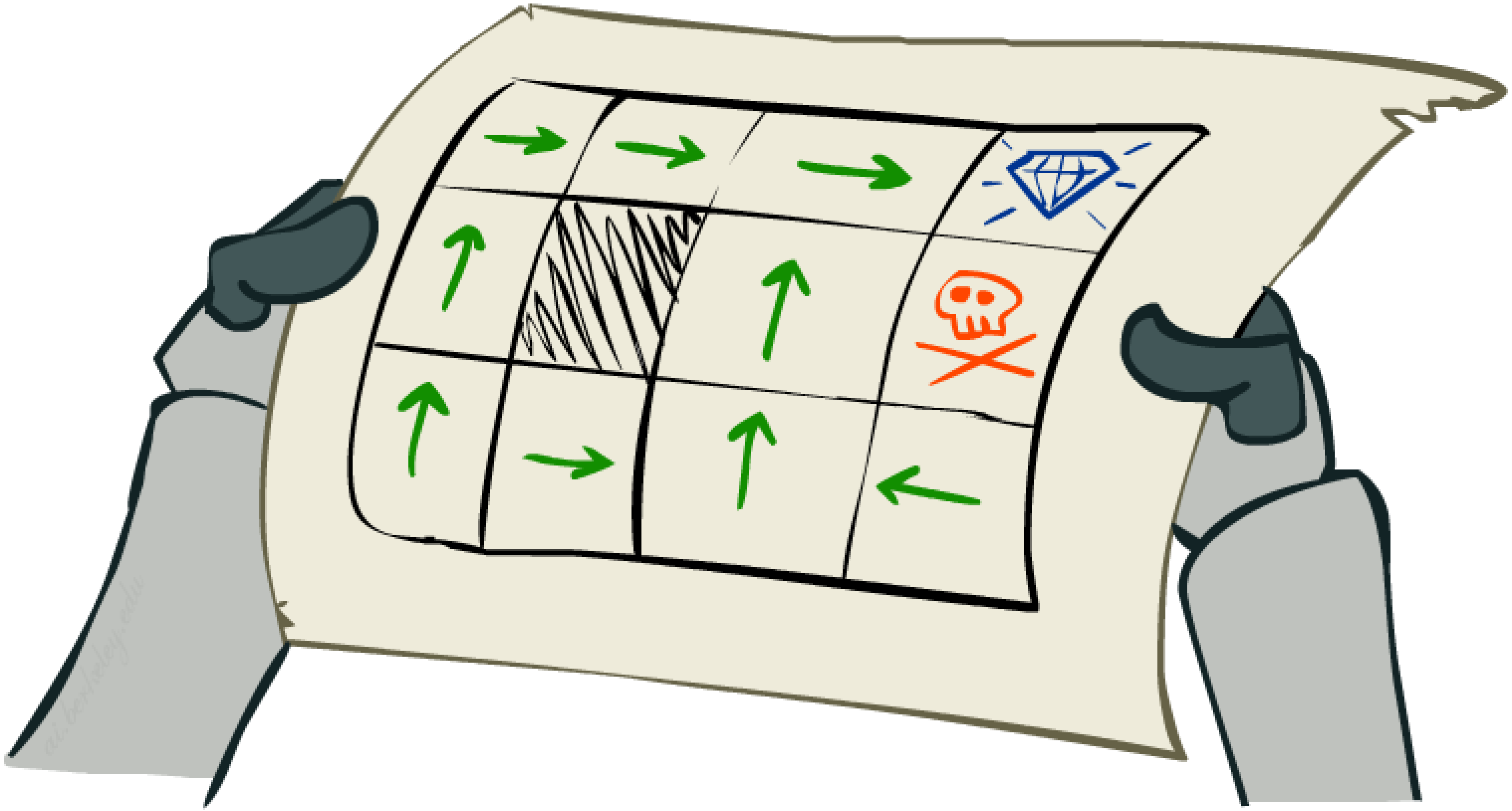
重温一下：MDPs的定义

- 马可夫决策过程:
  - 状态集合 S
  - 开始状态 s_0
  - 行动集合 A
  - 转移模型 P(s’|s, a) (或 T(s, a, s’))
  - 奖赏函数 R(s, a, s’) (和 折扣 γ)



- MDP 到目前为止所涉及的数量计算:
  - 策略 Policy = 对于每个给定状态的行动选择
  - 功效值 Utility = (折扣的) 奖赏值 之和

求解 MDPs



Racing Search Tree

doing way too much
with expectimax!

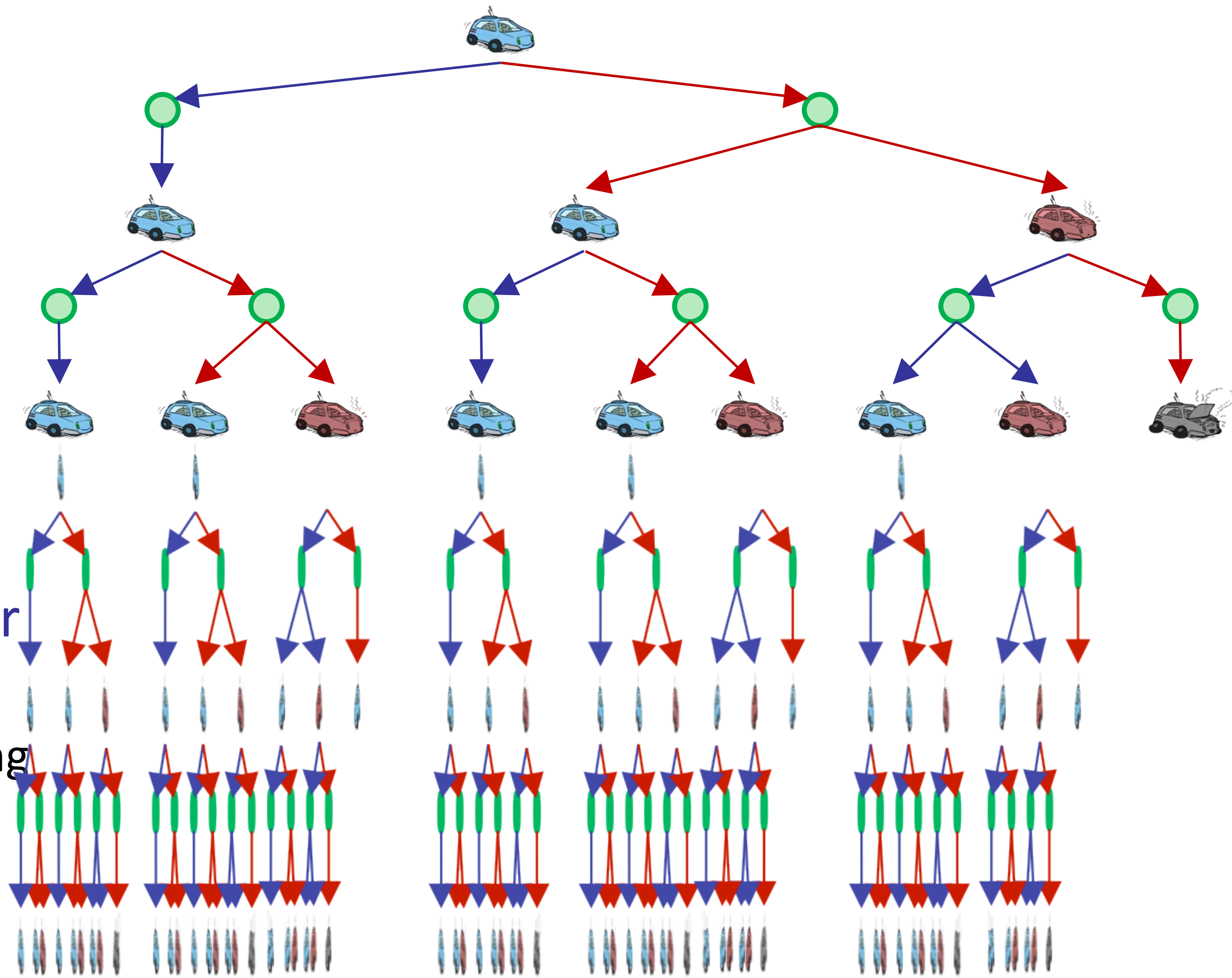
m: States are repeated

a: Only compute needed
ntities once

m: Tree goes on forever

a: Do a depth-limited
nputation, but with increasing
ths until change is small

e: deep parts of the tree
ntually don't matter if $\gamma < 1$



定义最优量值

- 一个状态 s 的(功效)值:

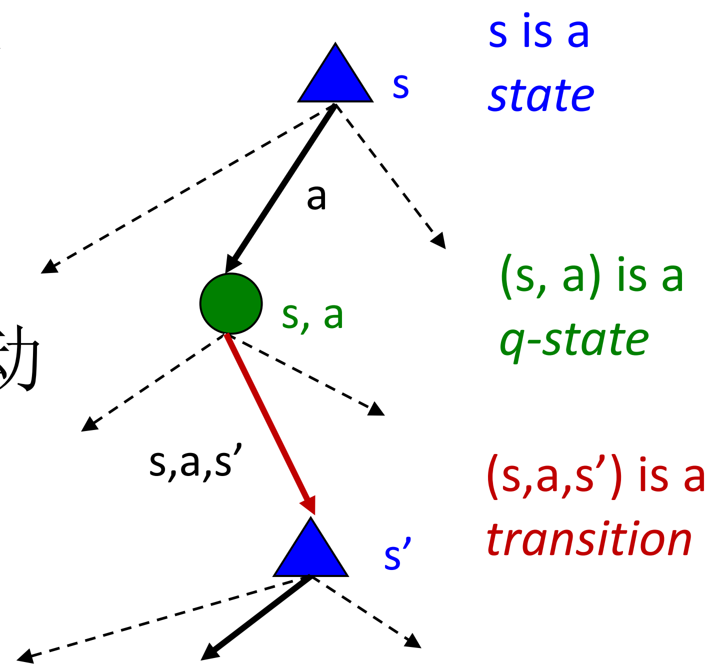
  V*(s) = 开始于状态s然后随后都采取最优化行动，最后获得的期望功效值

- 一个 q-状态 (s, a)的(功效)值:

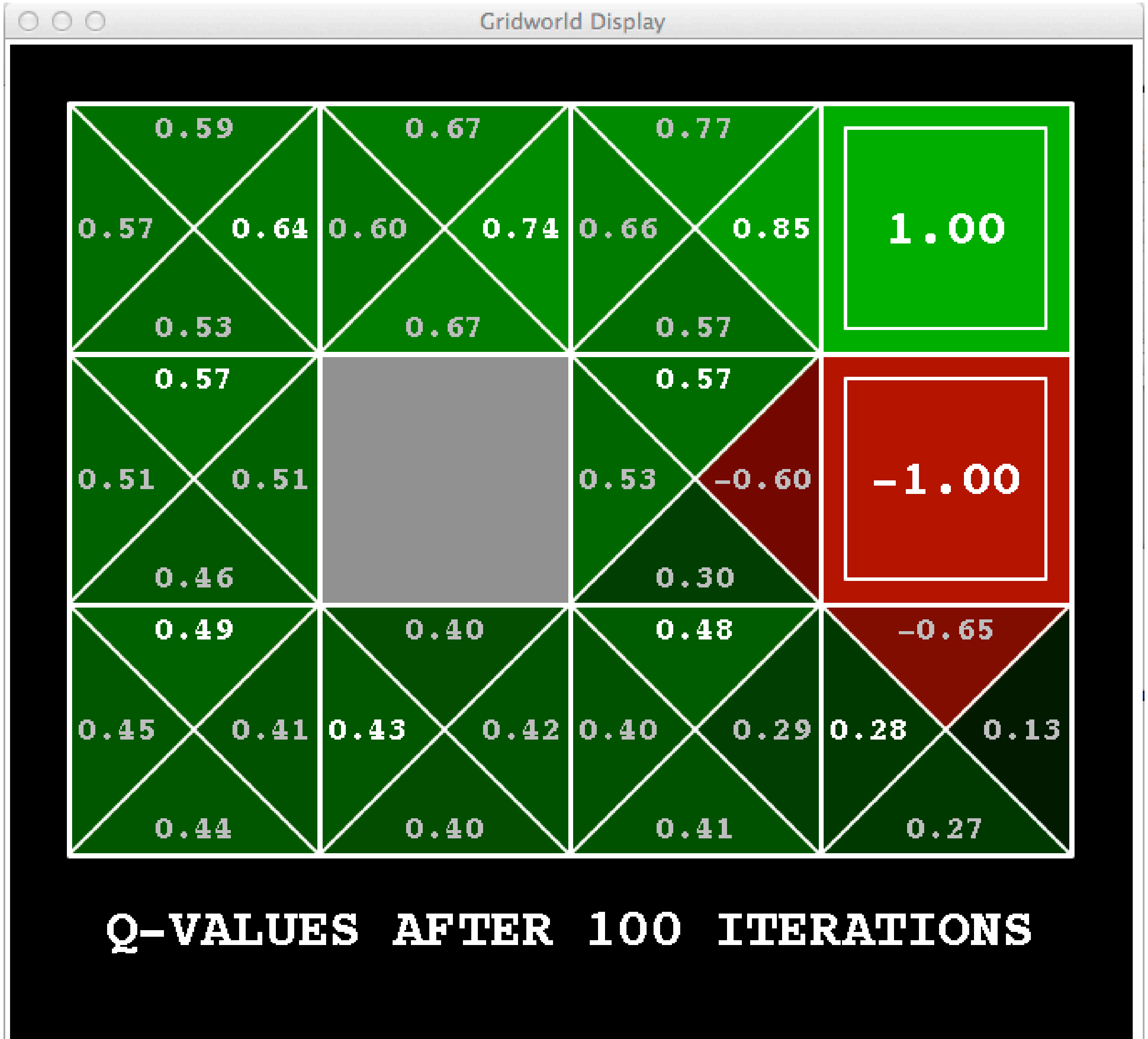
  Q*(s, a) = 开始于状态s并采取行动a, 随后都采取最优化行动，最后获得的期望功效值

- 最优策略:

  π*(s) =给定状态 s，返回最优的行动

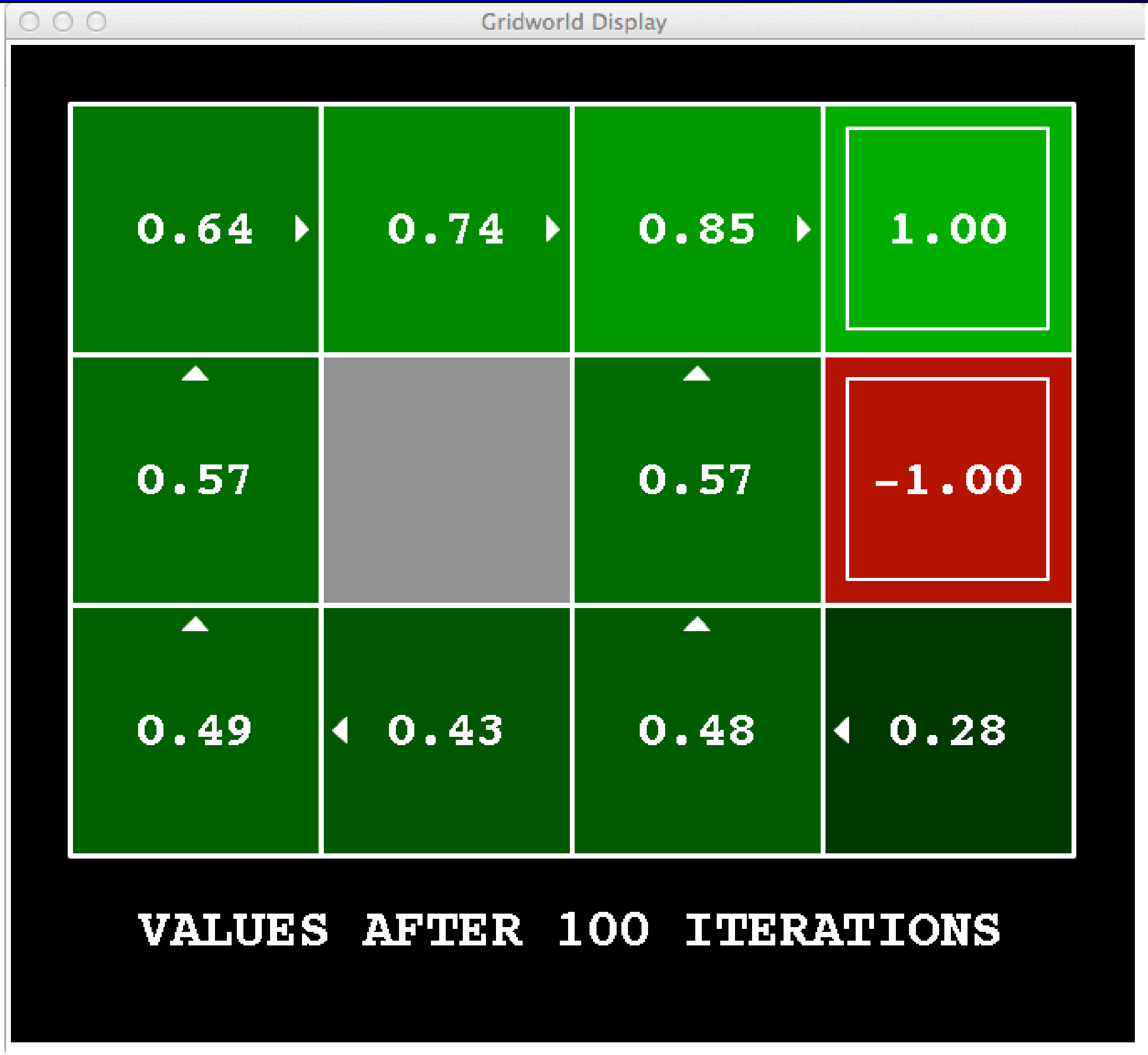


Snapshot of Gridworld - Q Values



Noise = 0.2
Discount = 0.9
Living reward = 0

Snapshot of Gridworld - V Values

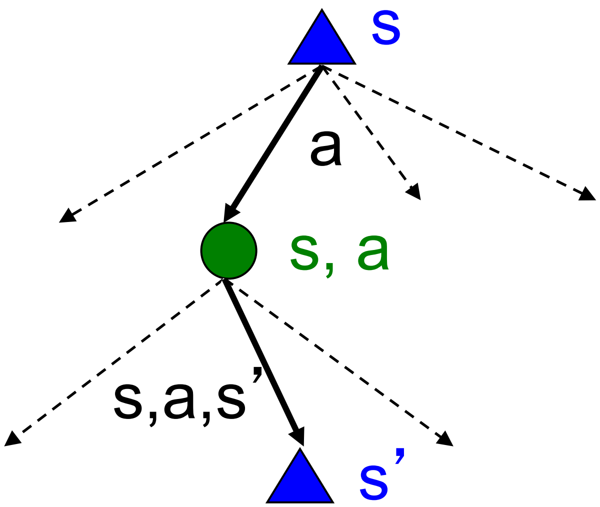


Noise = 0.2
Discount = 0.9
Living reward = 0

计算两种状态的值

- 基本操作：计算一个状态的（期望最大）值

- 递归定义：



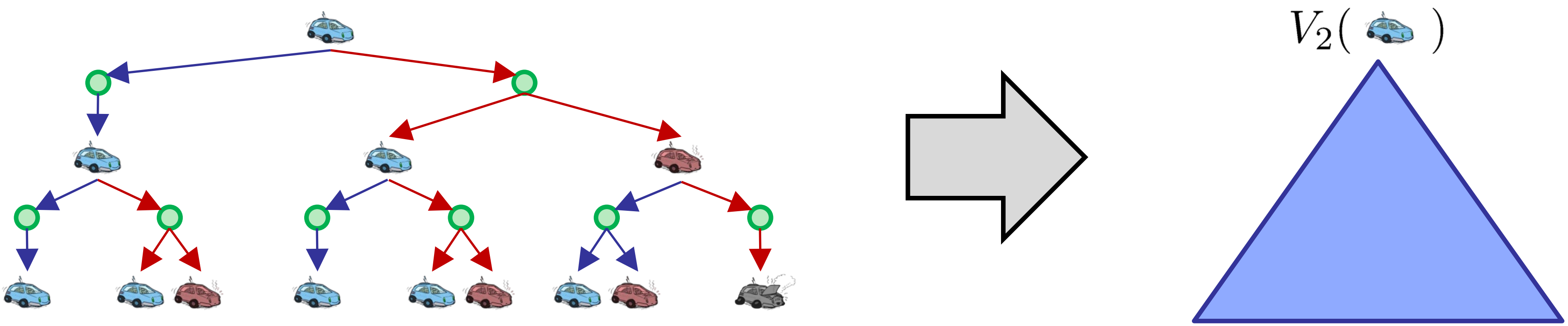
$$V^*(s) = \max_a Q^*(s,a)$$

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

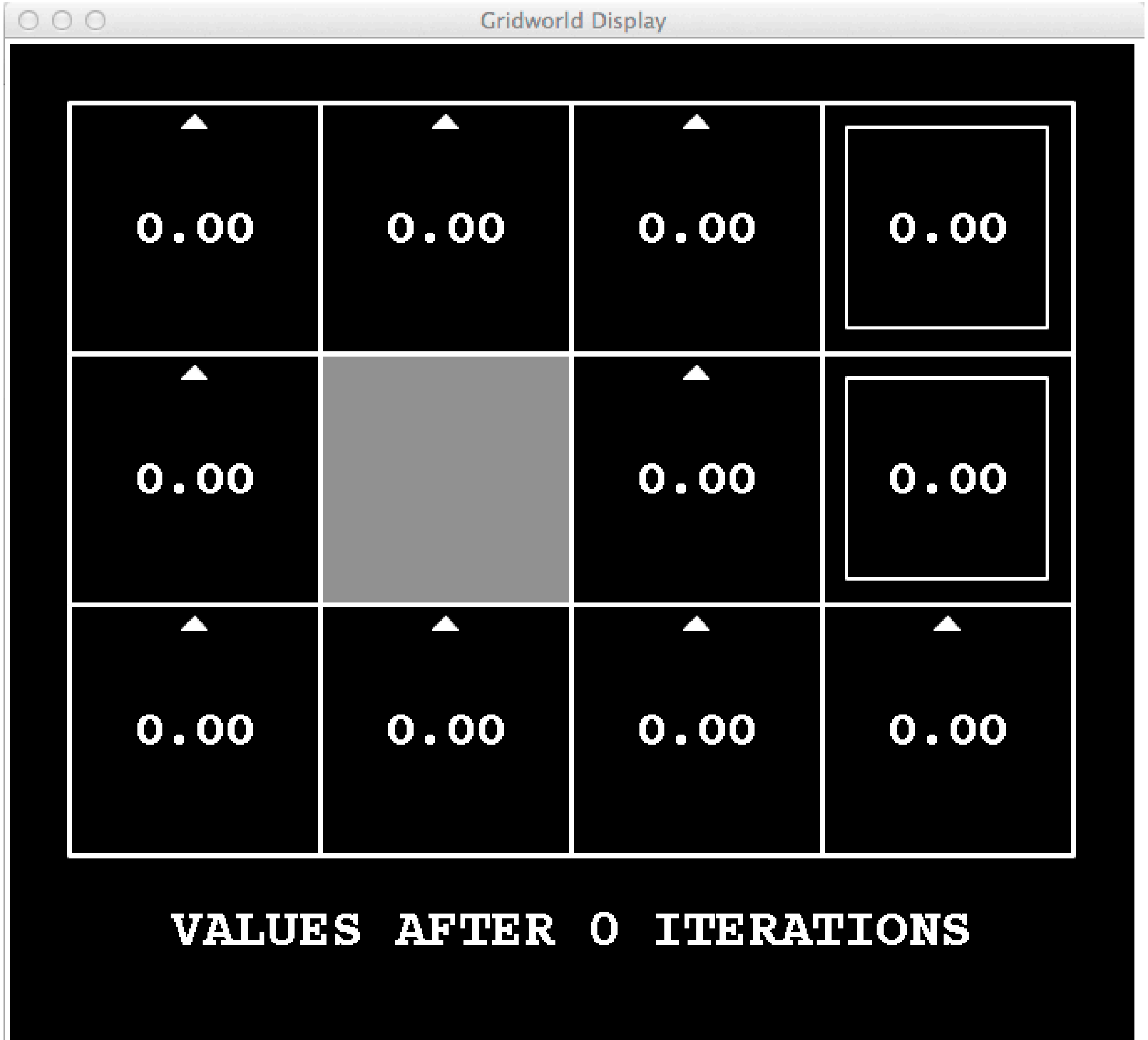
$$V^*(s) = \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s') \right]$$

有限时间步长下的V值

- Key idea: 限制时间步长
- Define $V_k(s)$ 状态s的最优值，如果游戏在K步长后结束
  - 相当于 it’ s what a depth-k expectimax would give from s

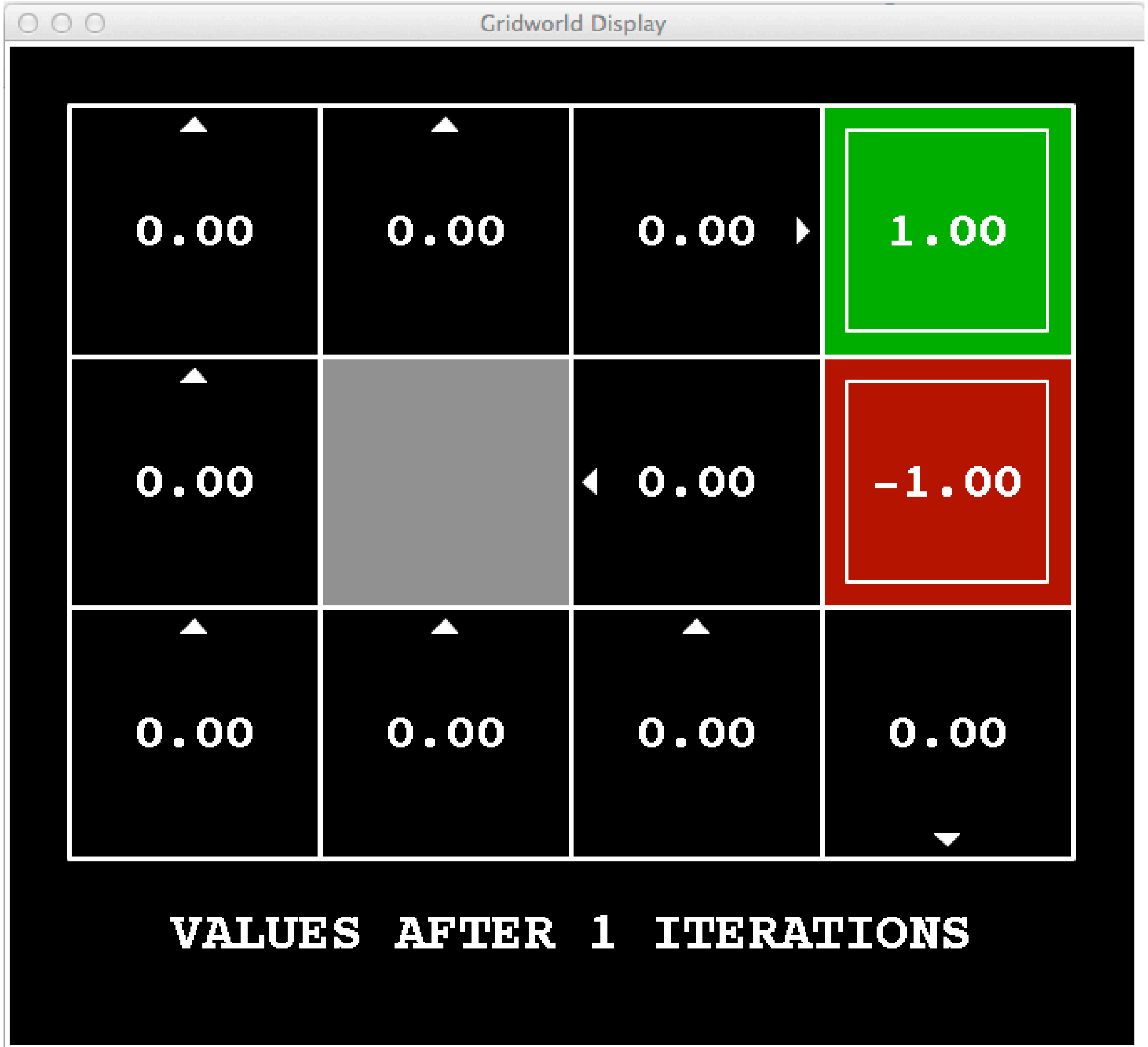


k=0



Noise = 0.2
Discount = 0.9
Living reward = 0

k=1



Noise = 0.2
Discount = 0.9
Living reward = 0

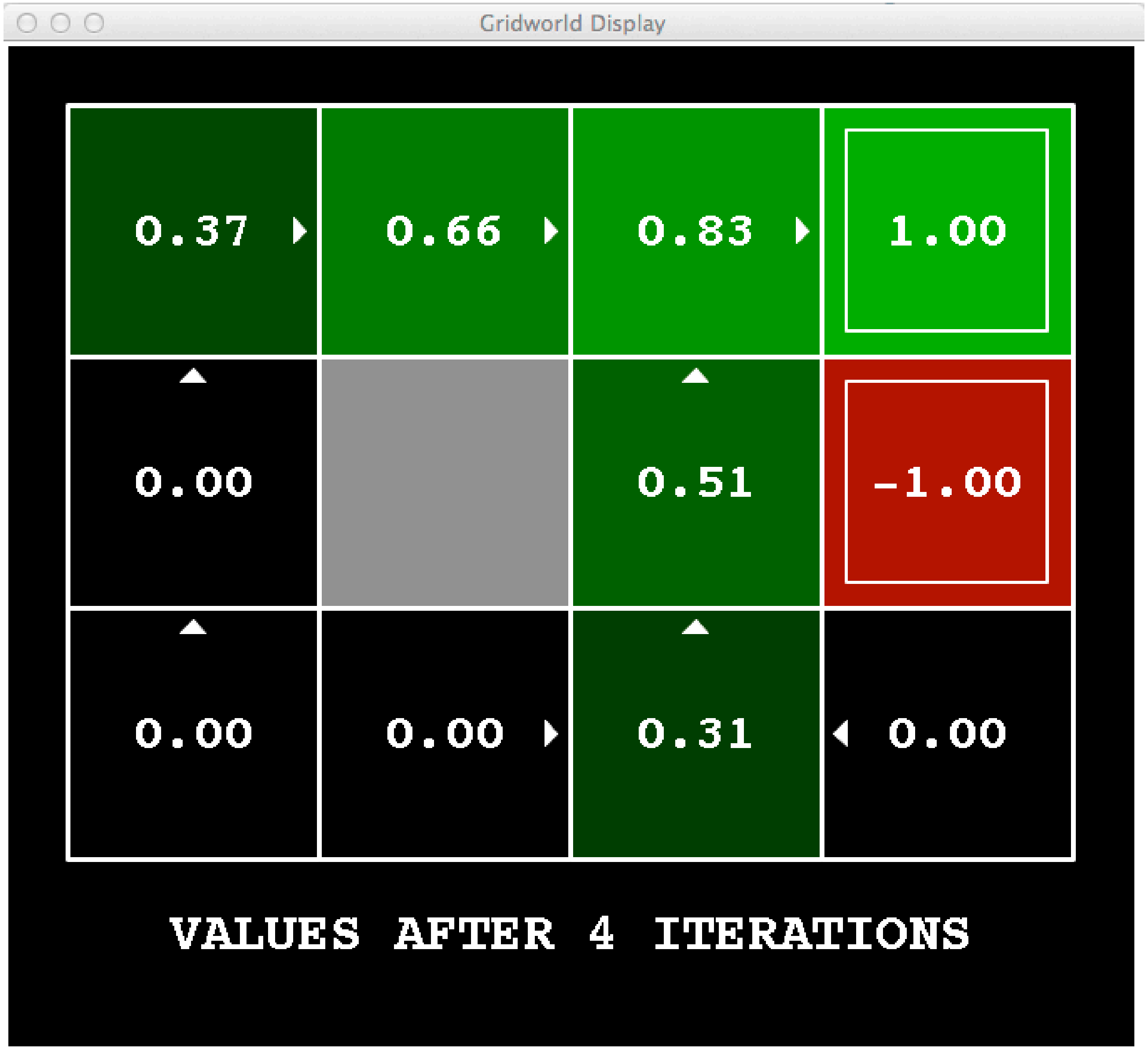
k=2

Noise = 0.2
Discount = 0.9
Living reward = 0

k=3

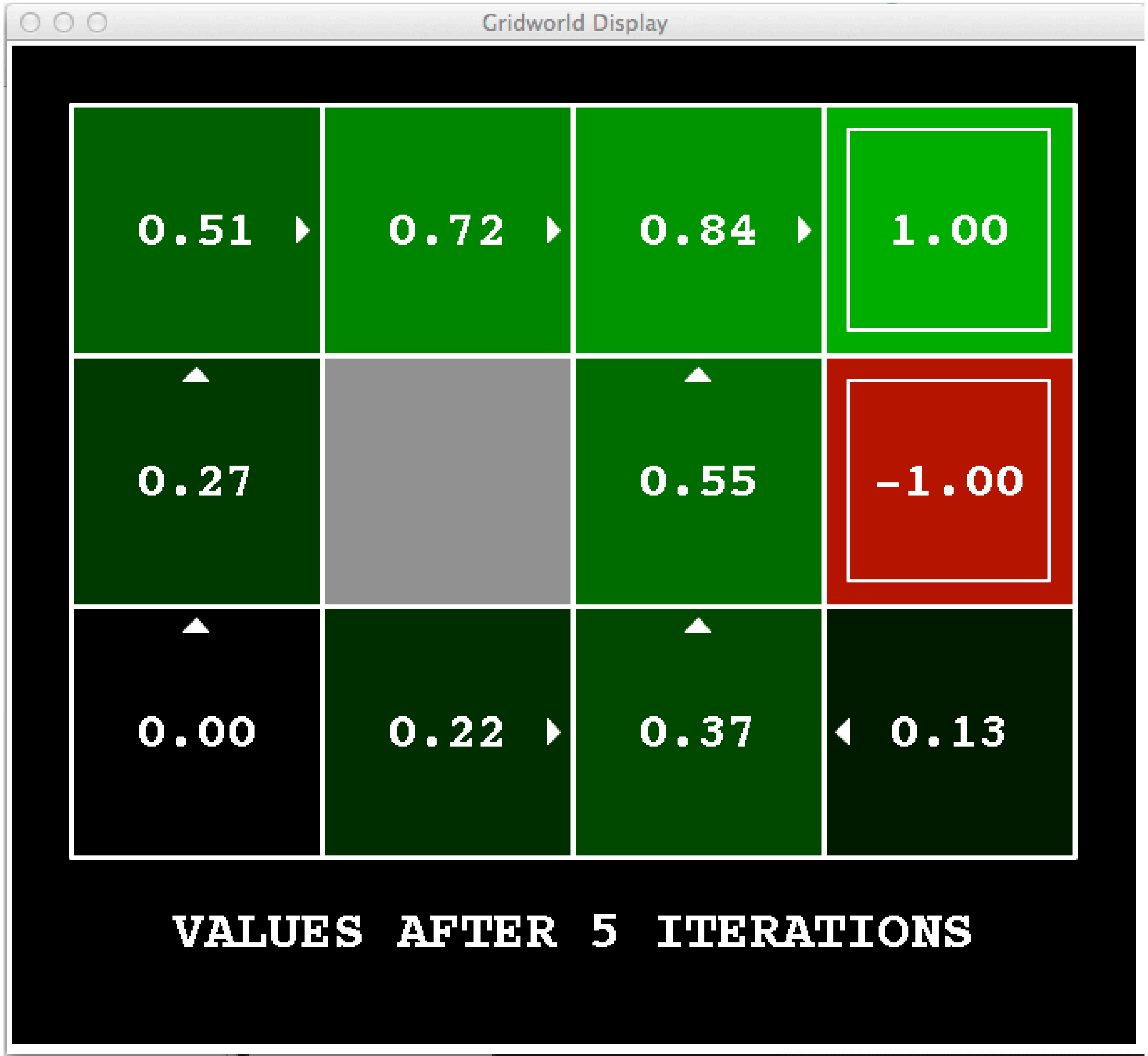
Noise =
Discour
Living r

k=4



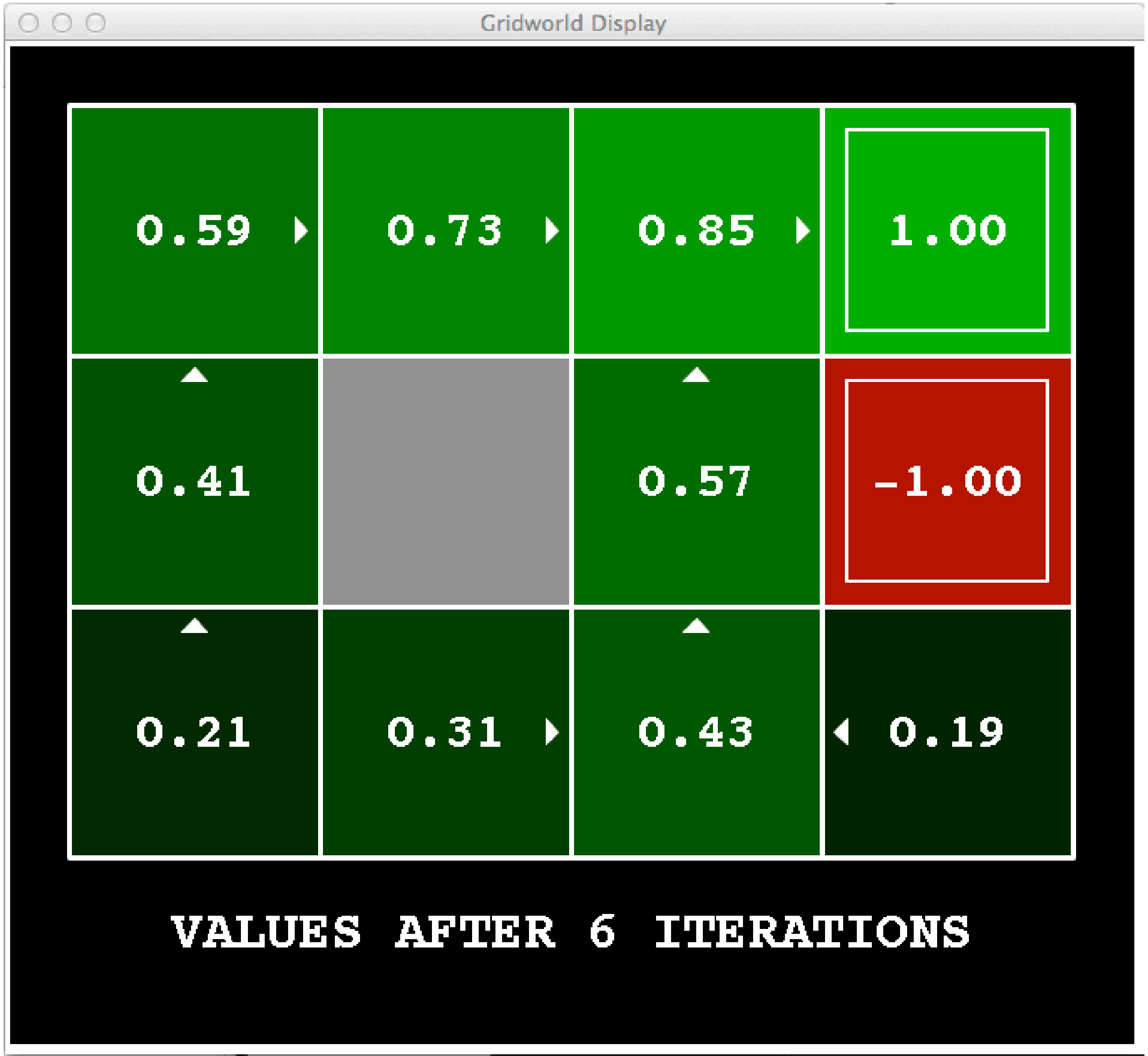
Noise =
Discour
Living r

k=5



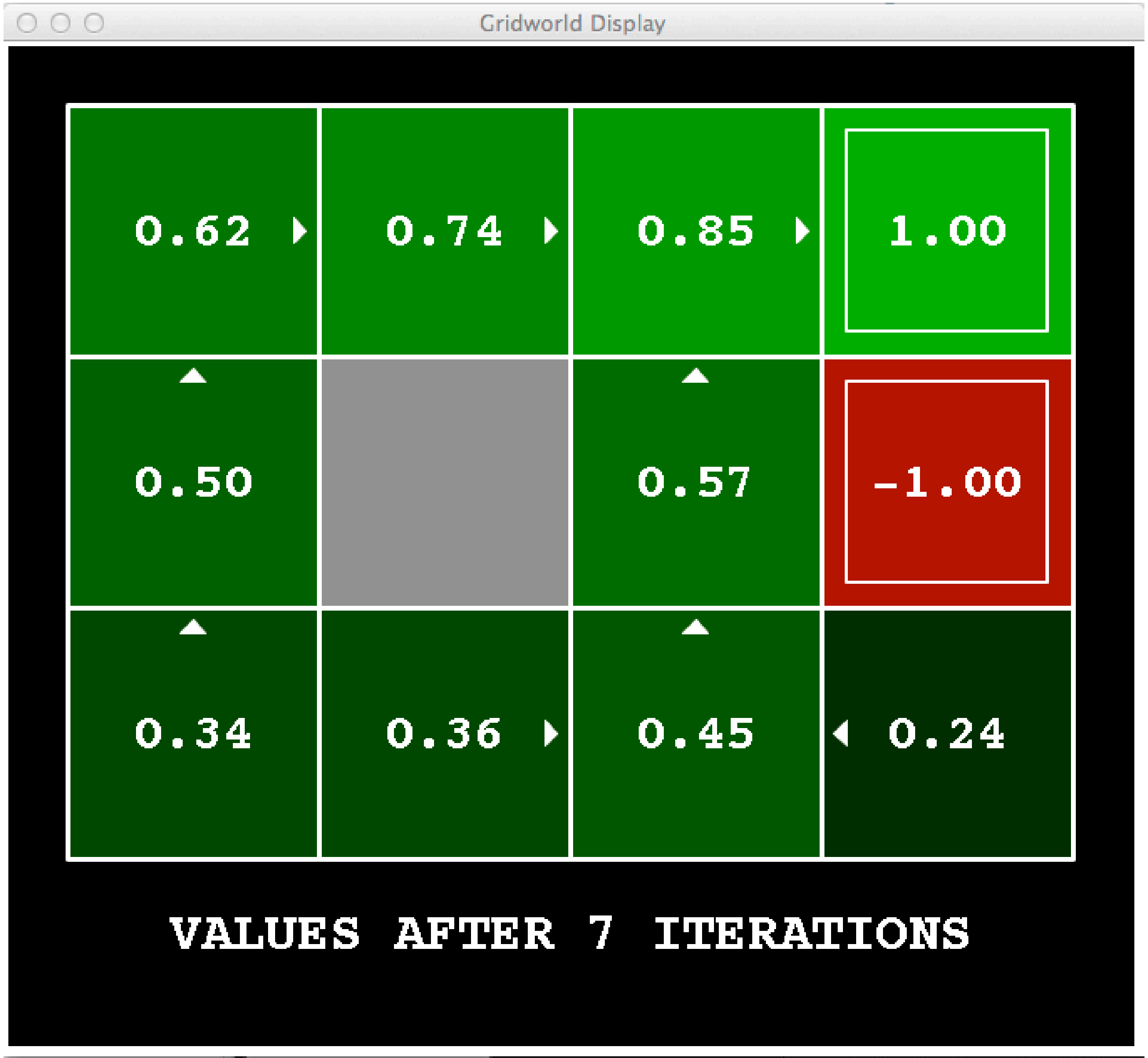
Noise =
Discour
Living r

k=6



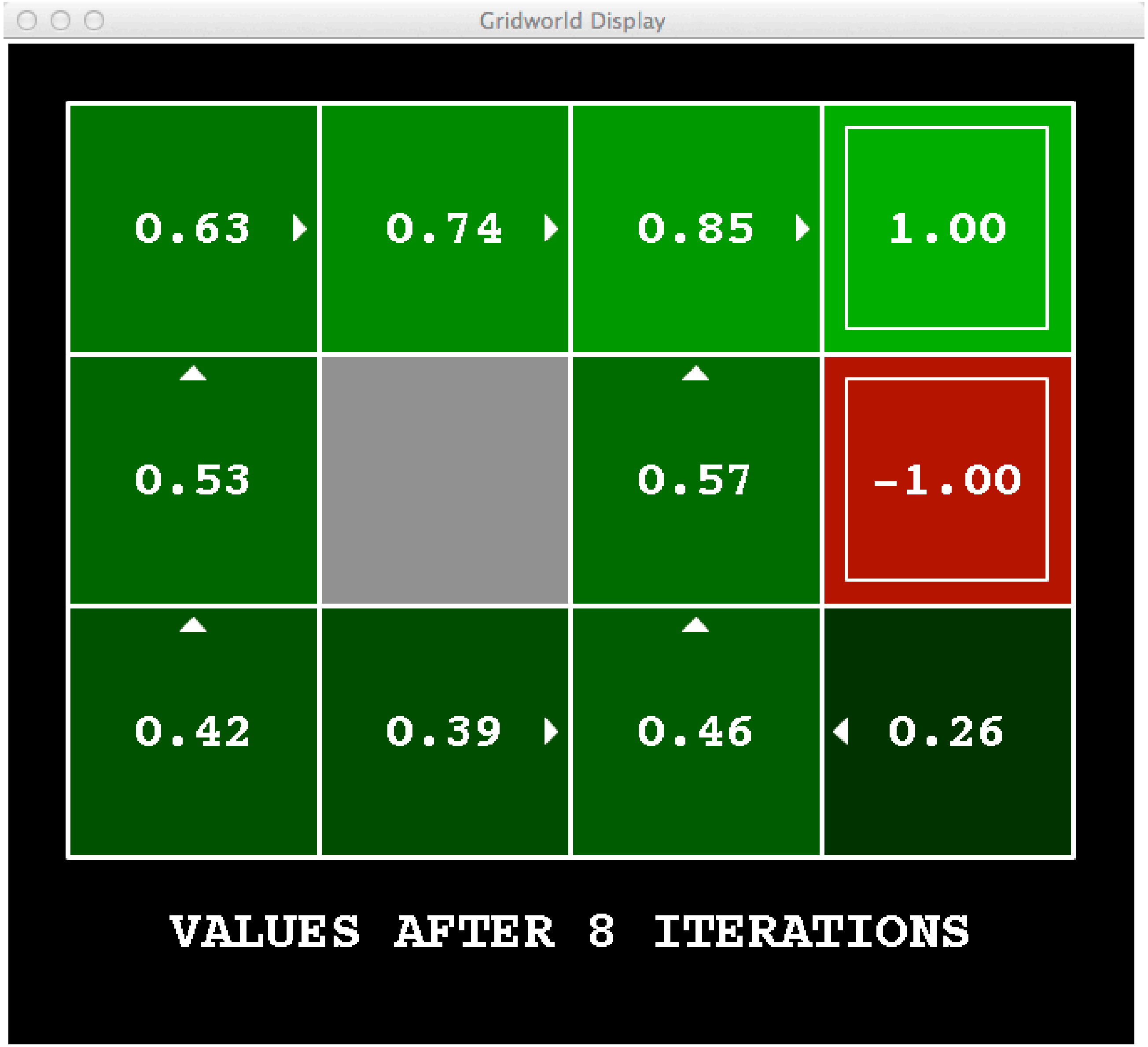
Noise =
Discour
Living r

k=7



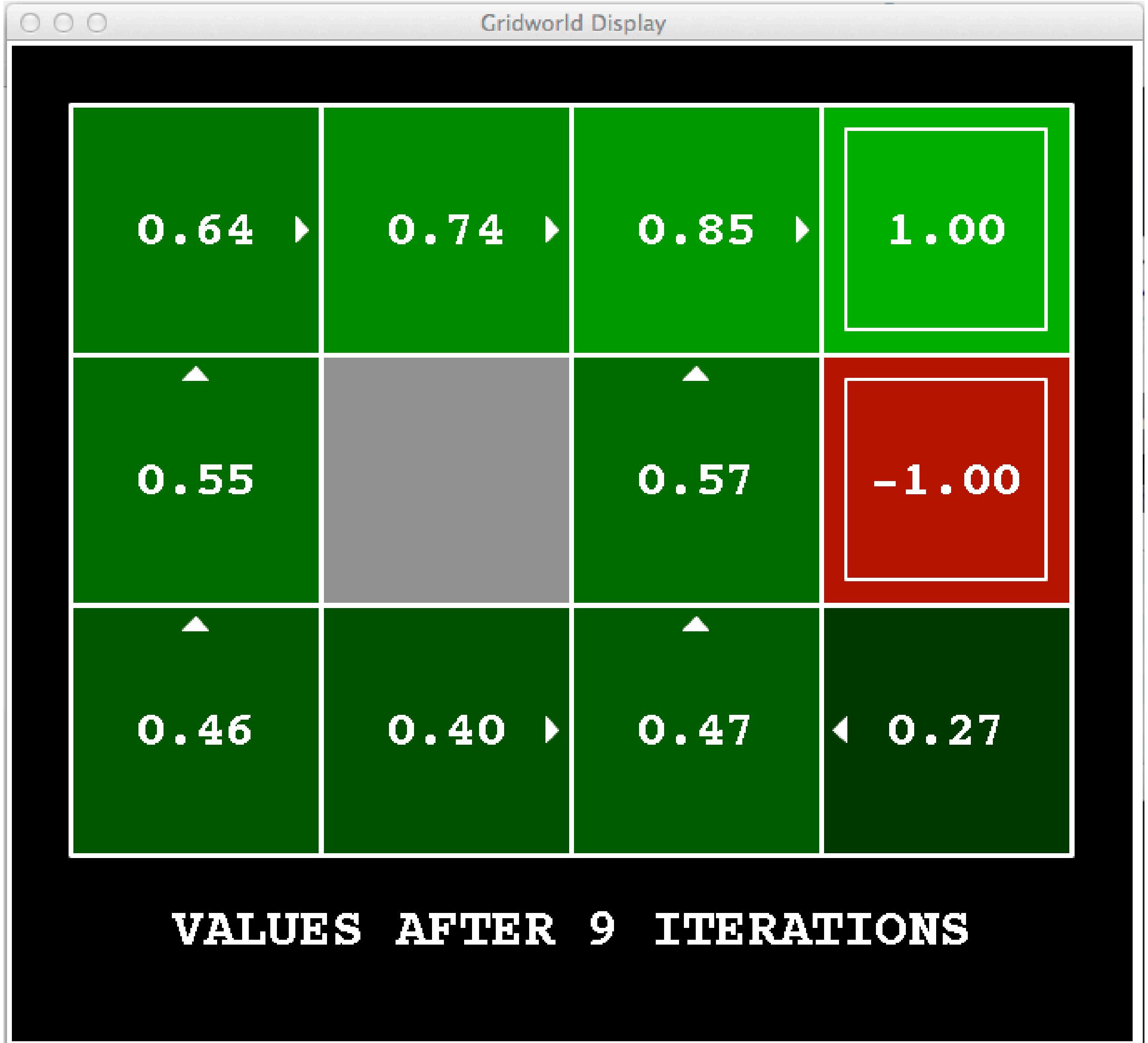
Noise =
Discour
Living r

k=8



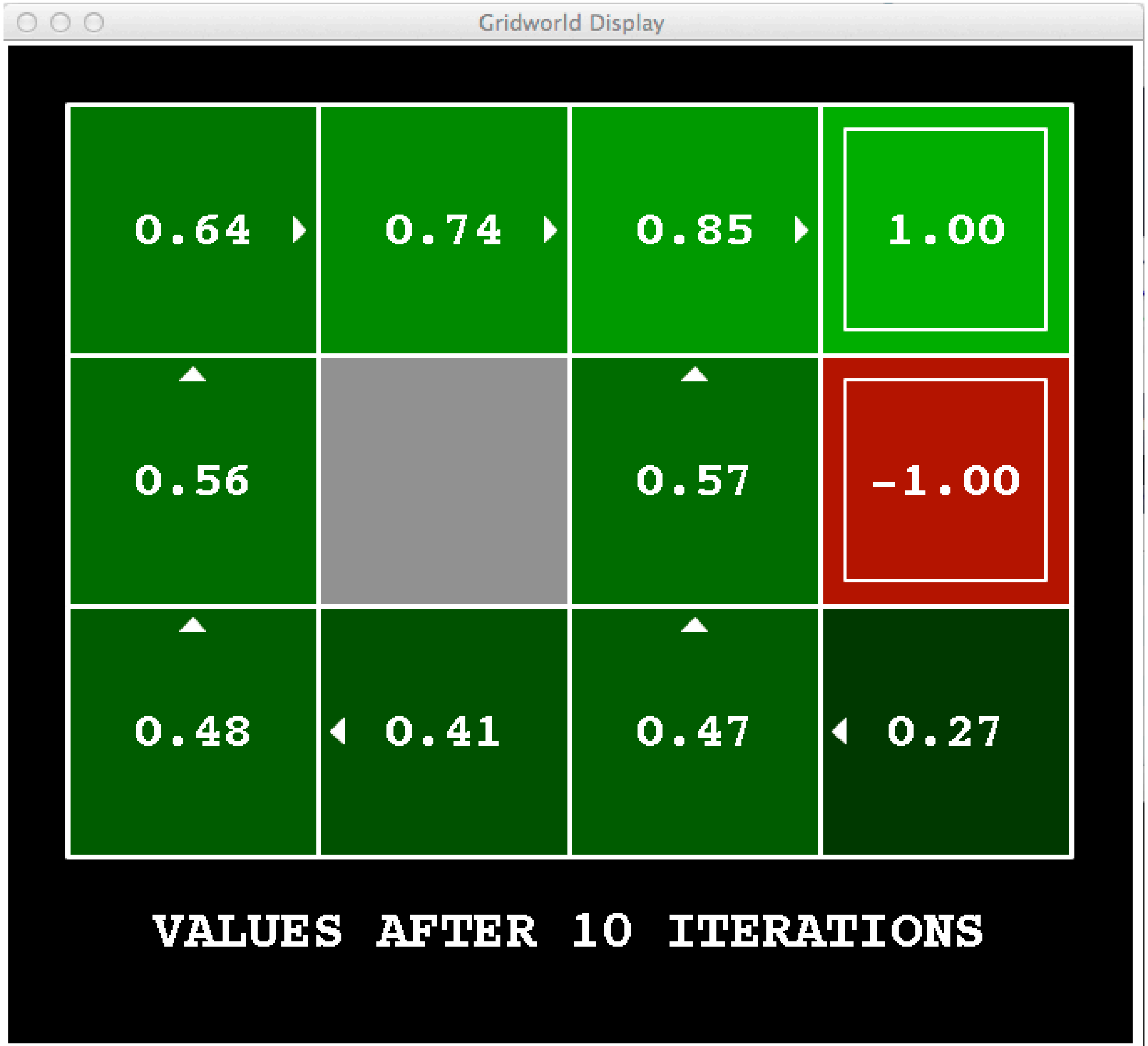
Noise =
Discour
Living r

k=9



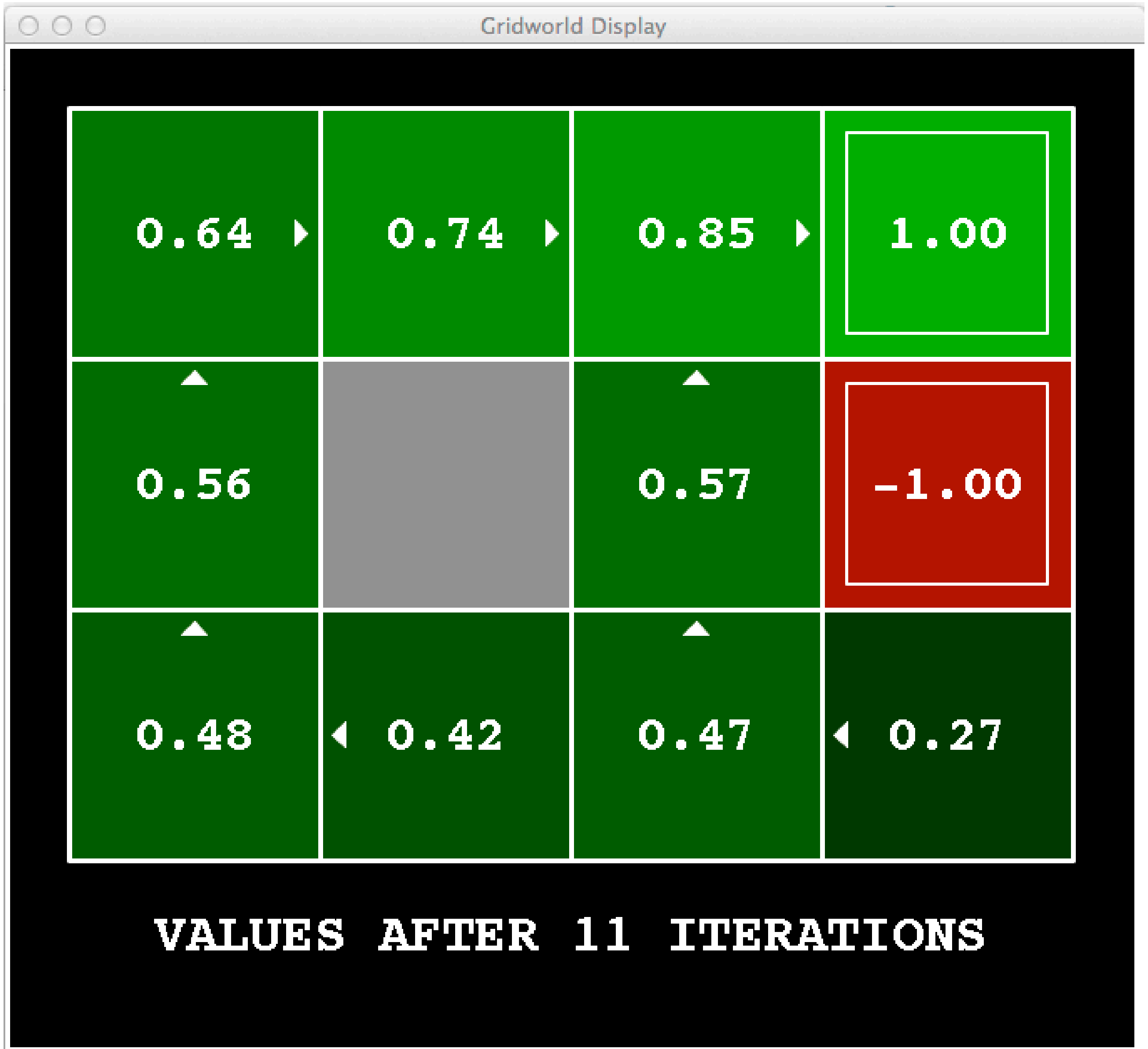
Noise =
Discour
Living r

k=10



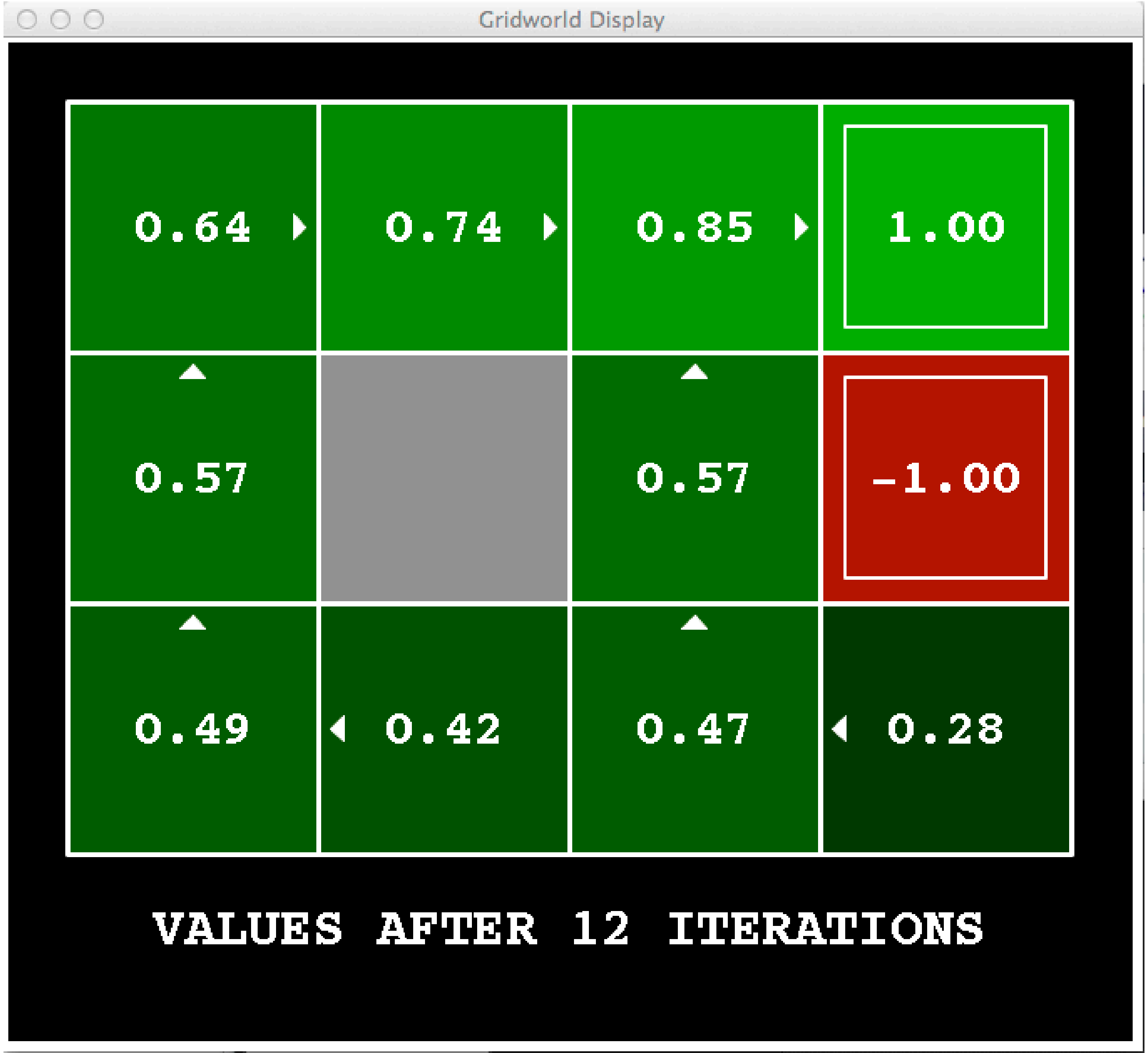
Noise =
Discour
Living r

k=11



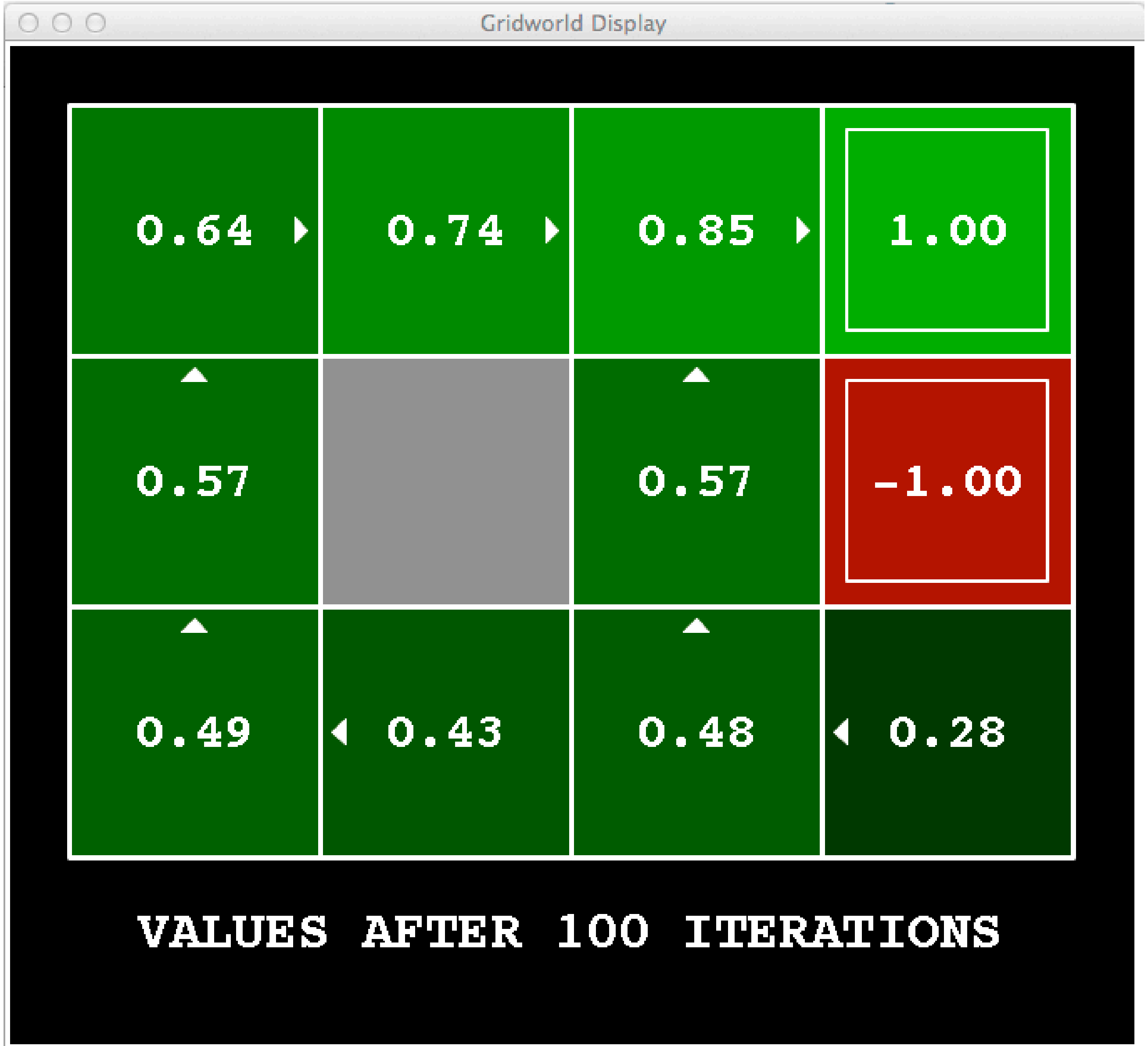
Noise =
Discour
Living r

k=12



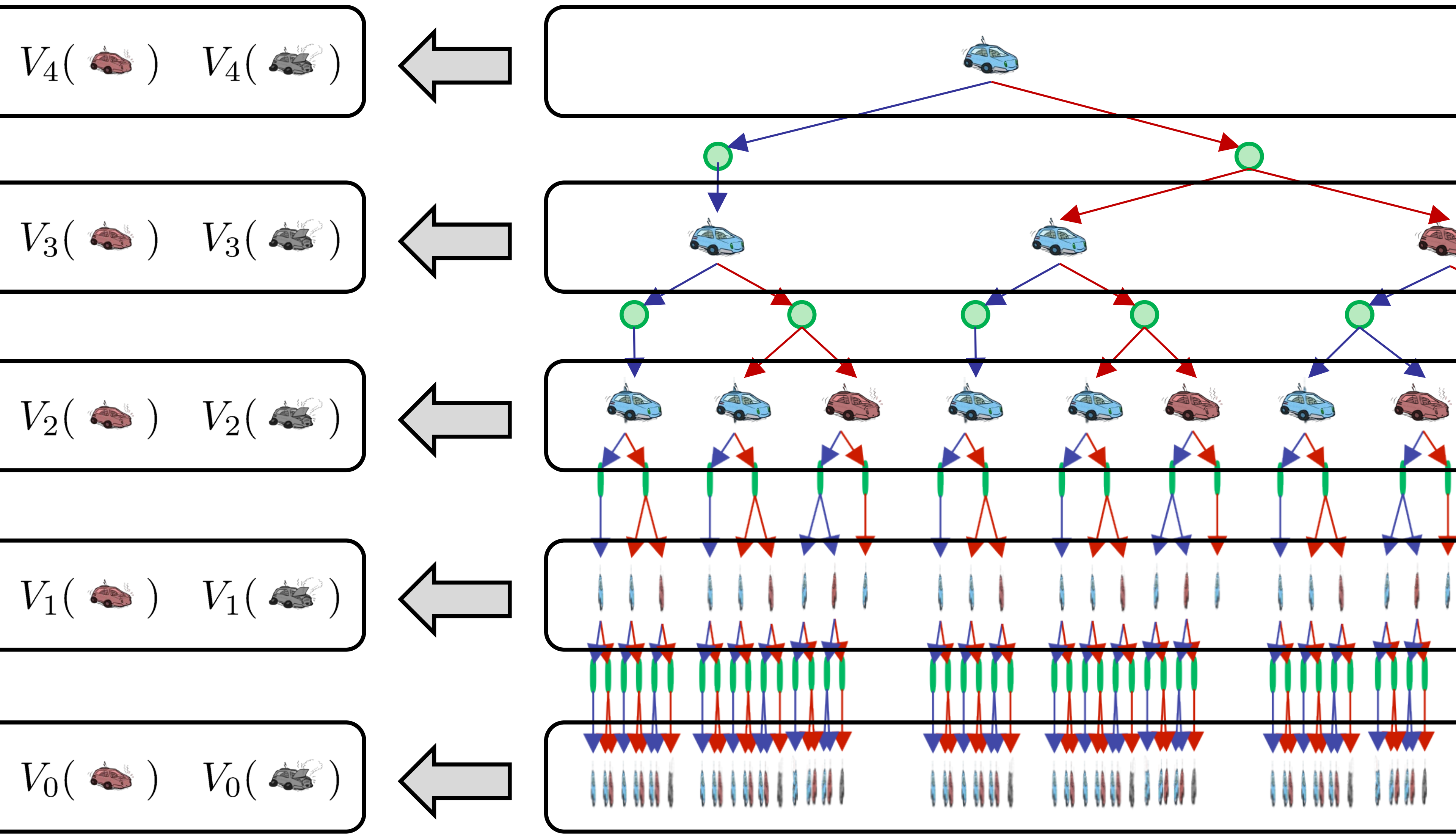
Noise =
Discour
Living r

k=100

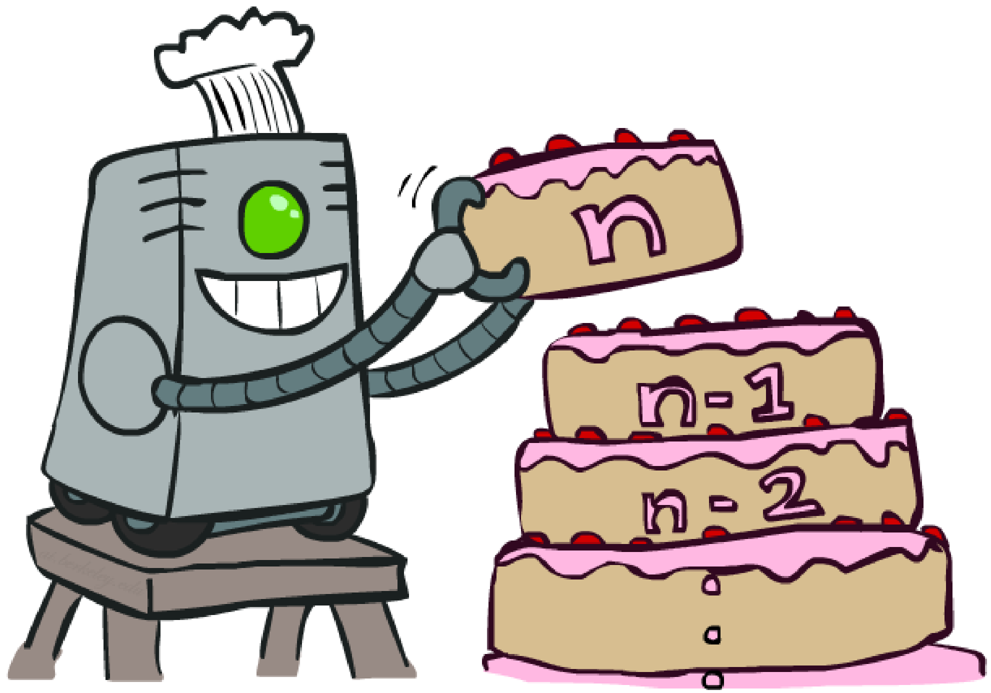


Noise = 0.2
Discount = 0.9
Living reward = 0

Computing Time-Limited Values



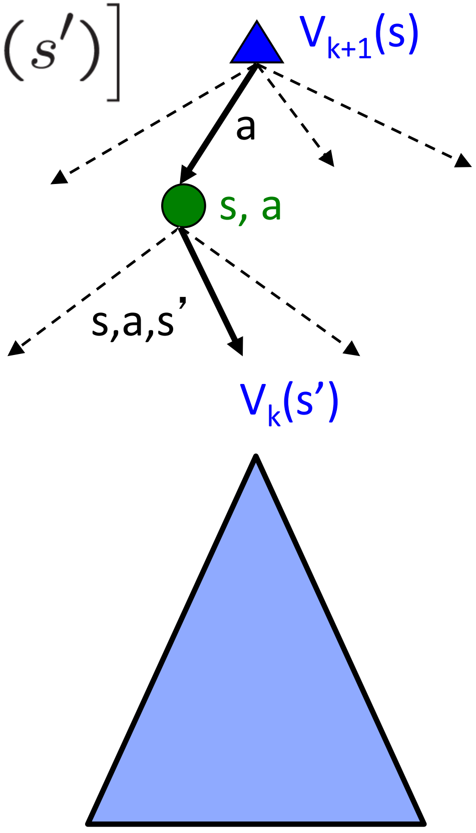
状态赋值迭代 Value Iteration



状态赋值迭代

- 开始于 $V_0(s) = 0$
- 给定向量 $V_k(s)$ 值，计算一步期望最大值，根据以下公式：

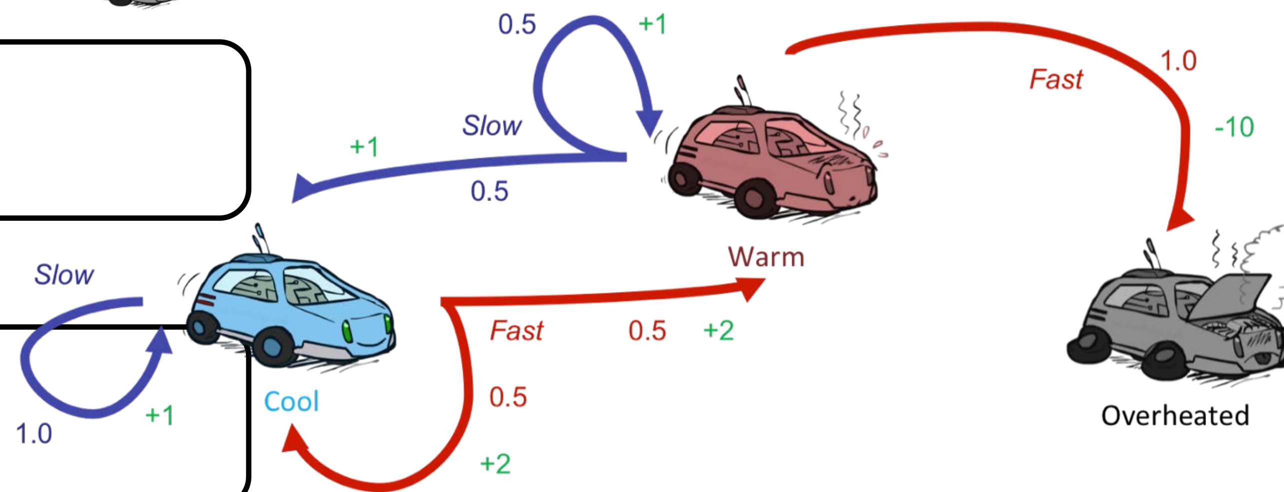
$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_k(s') \right]$$



- 重复这个过程，直到收敛
- 每步迭代的计算时间复杂度：$O(S^2A)$
- 定理：这个过程将会收敛于唯一的最优值

举例：赋值迭代

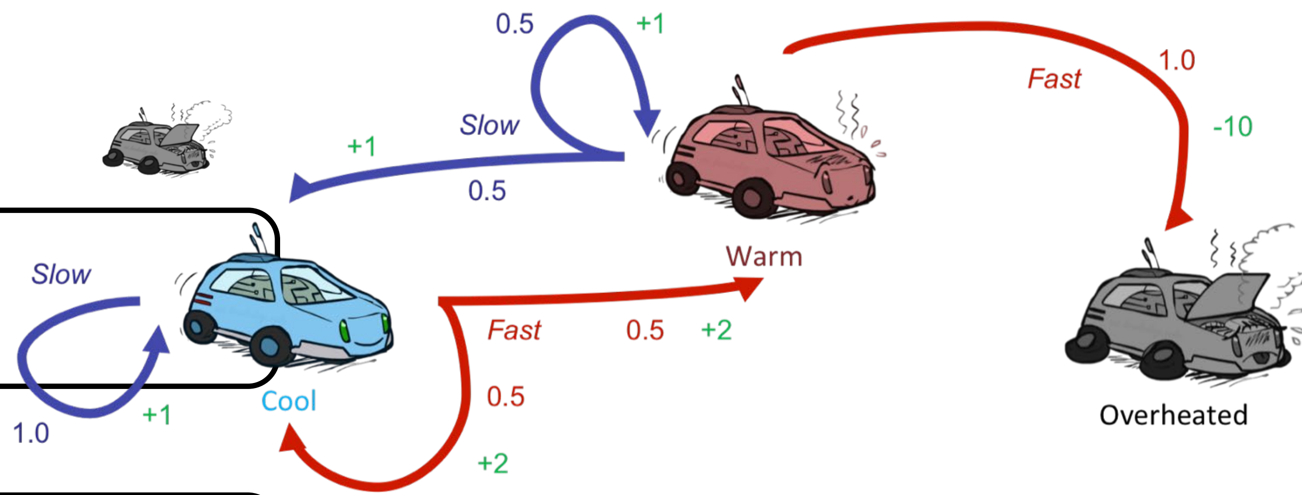
V_2			
V_1	S: 1 F: .5*2+.5*2=2		
V_0	0	0	0



假设没有折扣值 ！

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

举例：赋值迭代

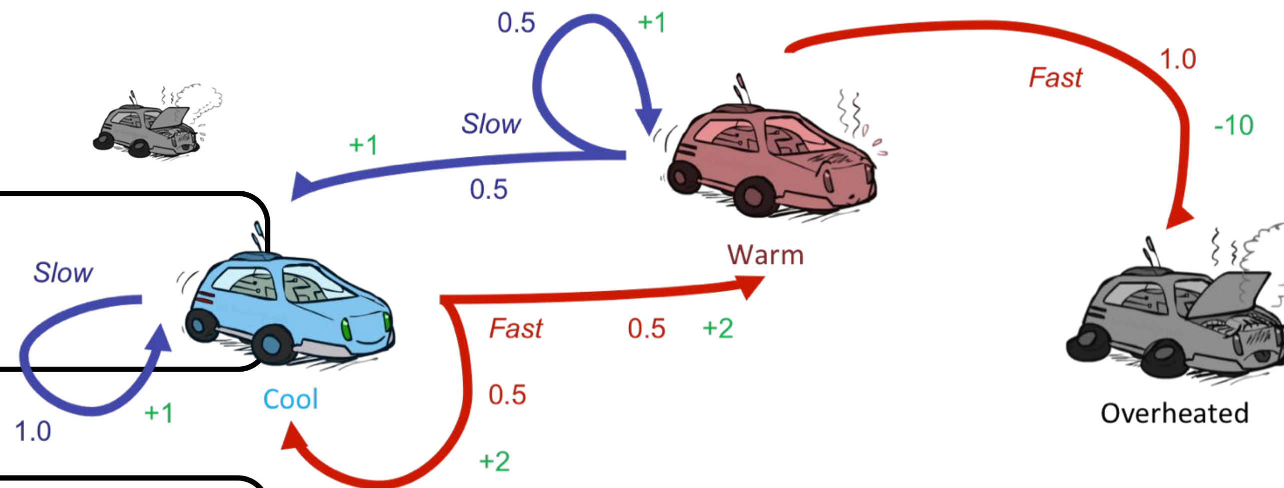


V_2			
V_1	2	S: .5*1+.5*1=1 F: -10	
V_0	0	0	0

假设没有折扣值 ！

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

举例：赋值迭代



V_2			
V_1	2	1	0
V_0	0	0	0

假设没有折扣值 ！

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

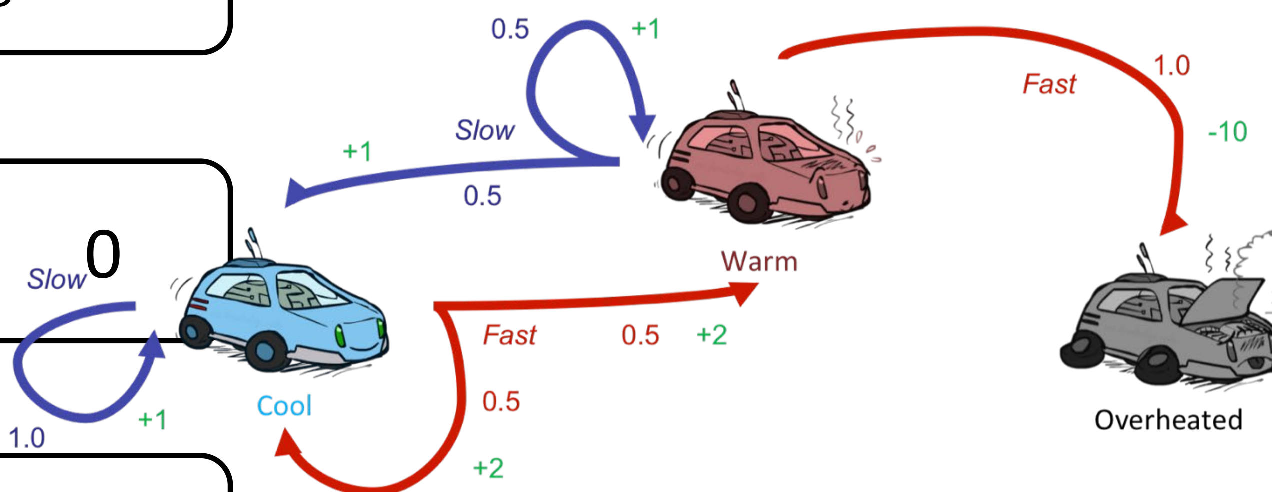
举例：赋值迭代

V_2

S: 1+2=3

F: .5*(2+2)+.5*(2+1)=3.5

V_1	2	1	0
V_0	0	0	0

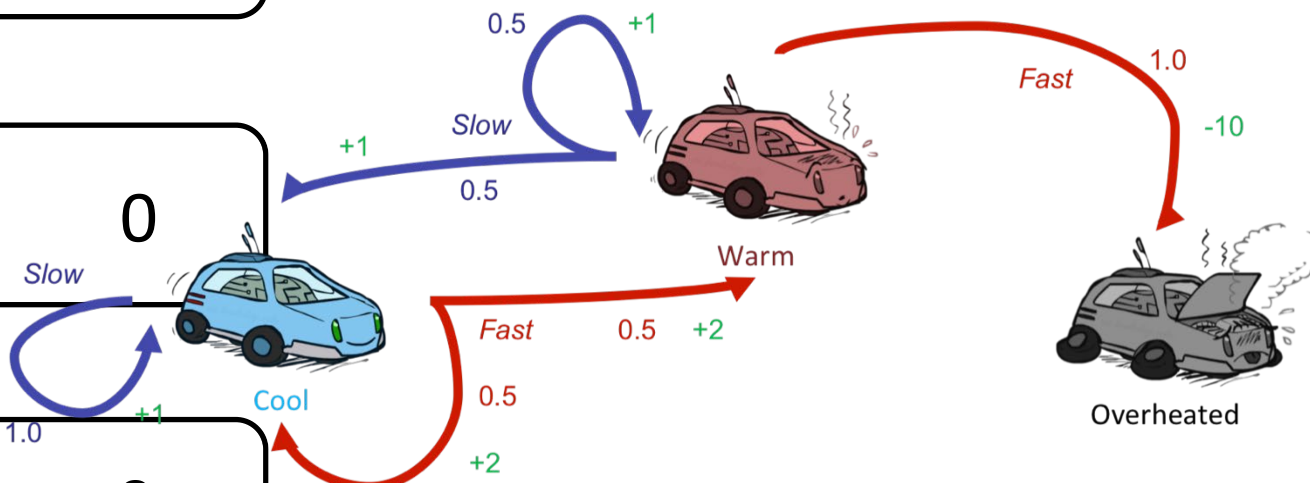


假设没有折扣值 ！

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

举例：赋值迭代

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



假设没有折扣值 ！

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