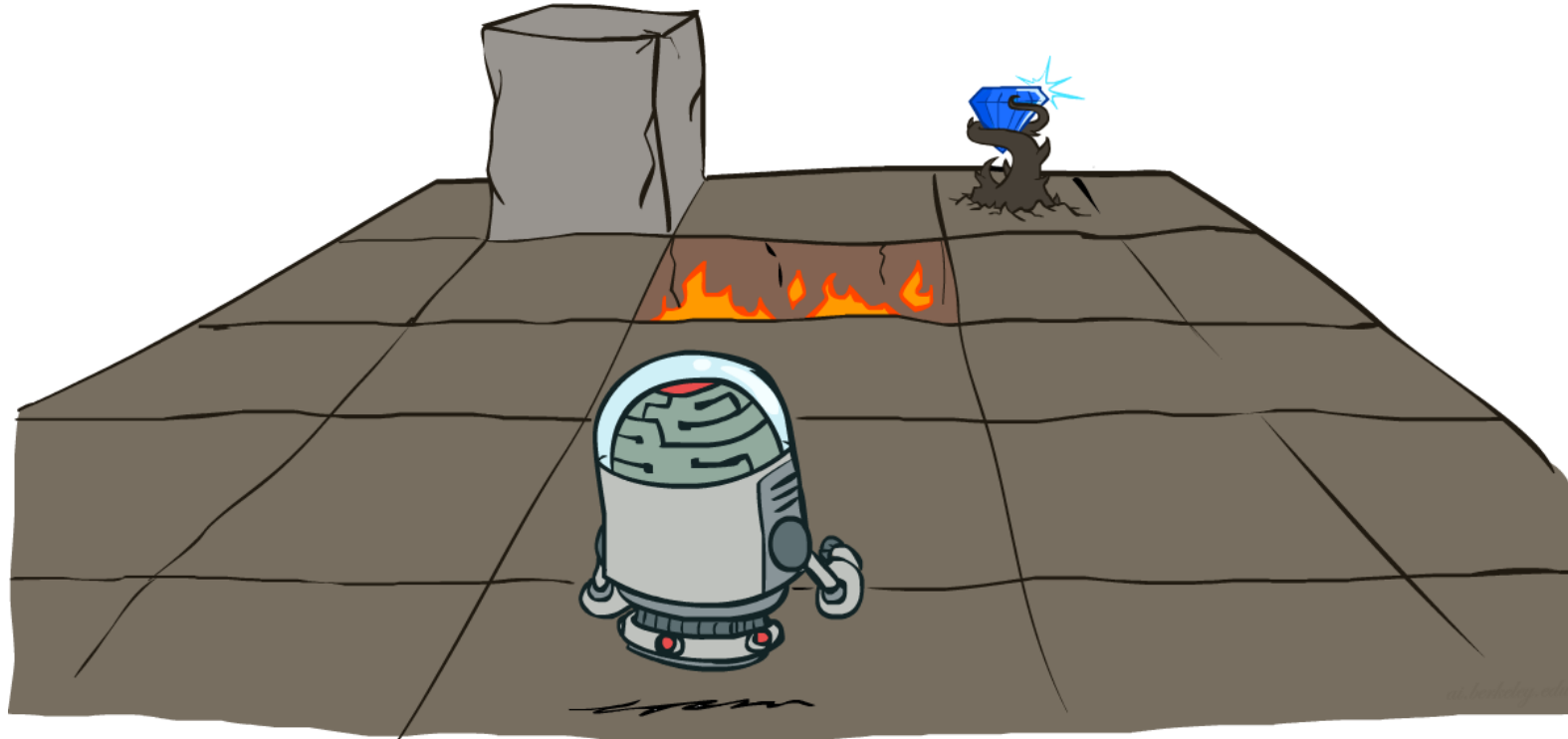


Osnovi Računarske Inteligencije

Markovljevi Procesi Odlučivanja

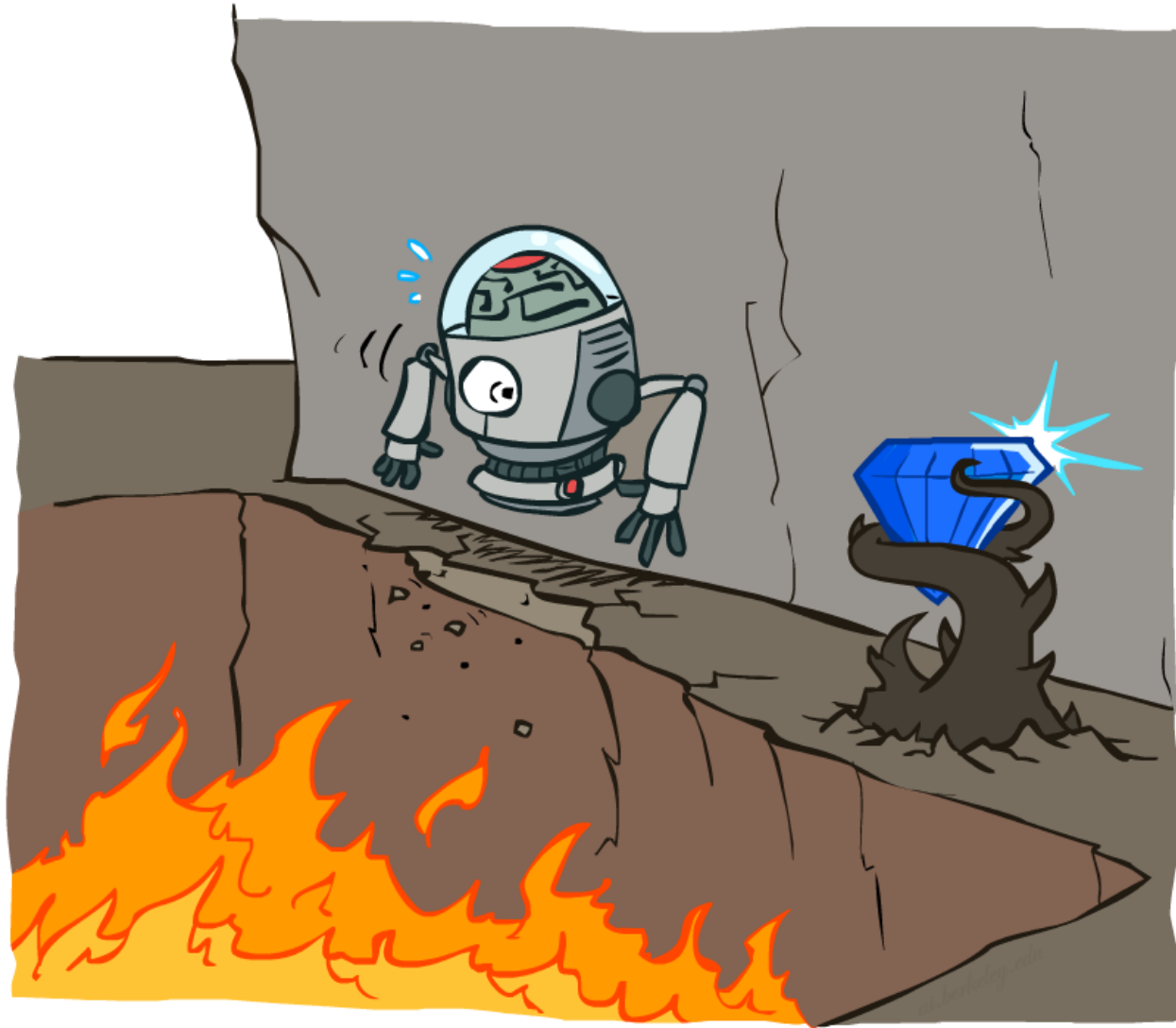


Predavač: Aleksandar Kovačević

Slajdovi preuzeti sa kursa CS188, University of California, Berkeley

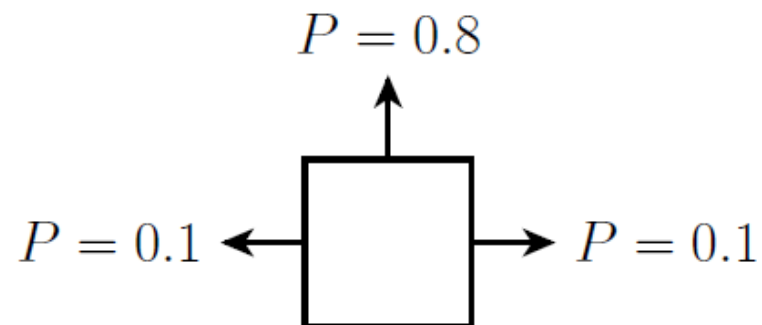
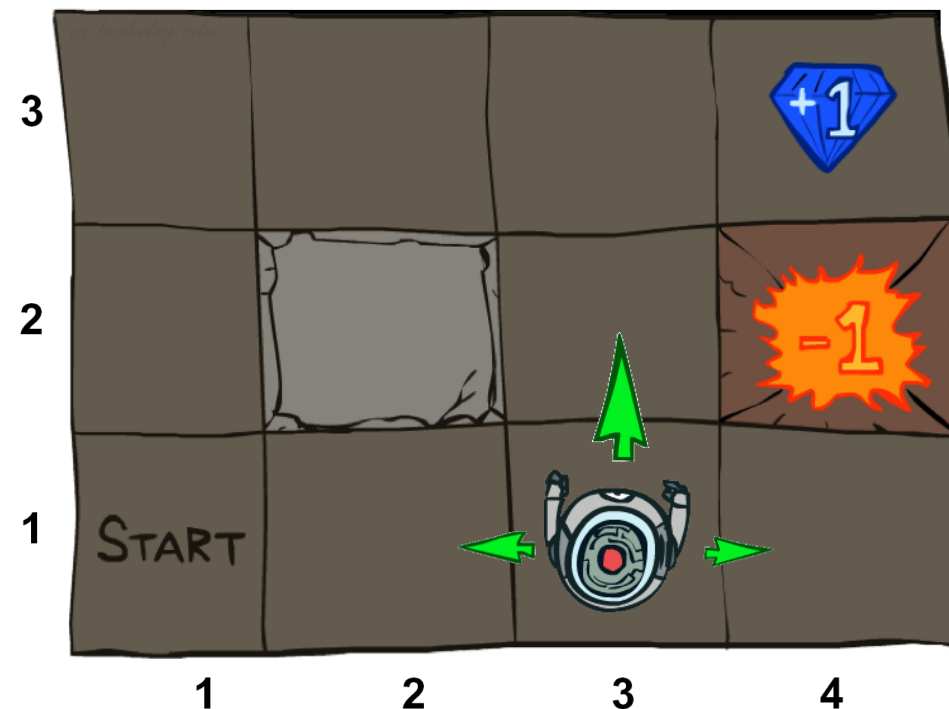
<http://ai.berkeley.edu/>

Ne-Determinističke Pretrage



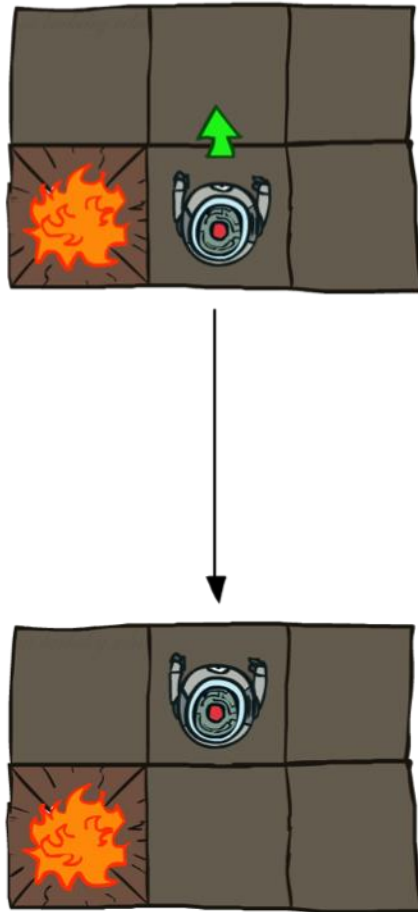
Primer: Mrežasti Svet (*Grid World*)

- Problem sličan lavirinintu
 - Agent živi na mreži polja
 - Zidovi mu blokiraju put
- Kretanje uz smetnje: ishodi akcija nisu uvek onakvi kakve očekujemo
 - Za naš primer: 80% puta akcija Napred vodi nas na Napred (ako tamo nema zida)
 - 10% puta akcija Napred nas vodi na Levo, a 10% na Desno (akcija Napred nas nikad ne vodi unazad).
 - Ako je zid tamo gde treba da nas odvede rezultat akcije, agent ostaje u mestu.
- Agent dobija nagradu za svaki korak koji napravi
 - Malu nagradu "postojanja" (*living reward*) za svaki korak (nagrada može biti negativna tj. kazna)
 - Velike nagrade dolaze na kraju (pozitivne ili negativne)
- Cilj: maksimizovati sumu nagrada

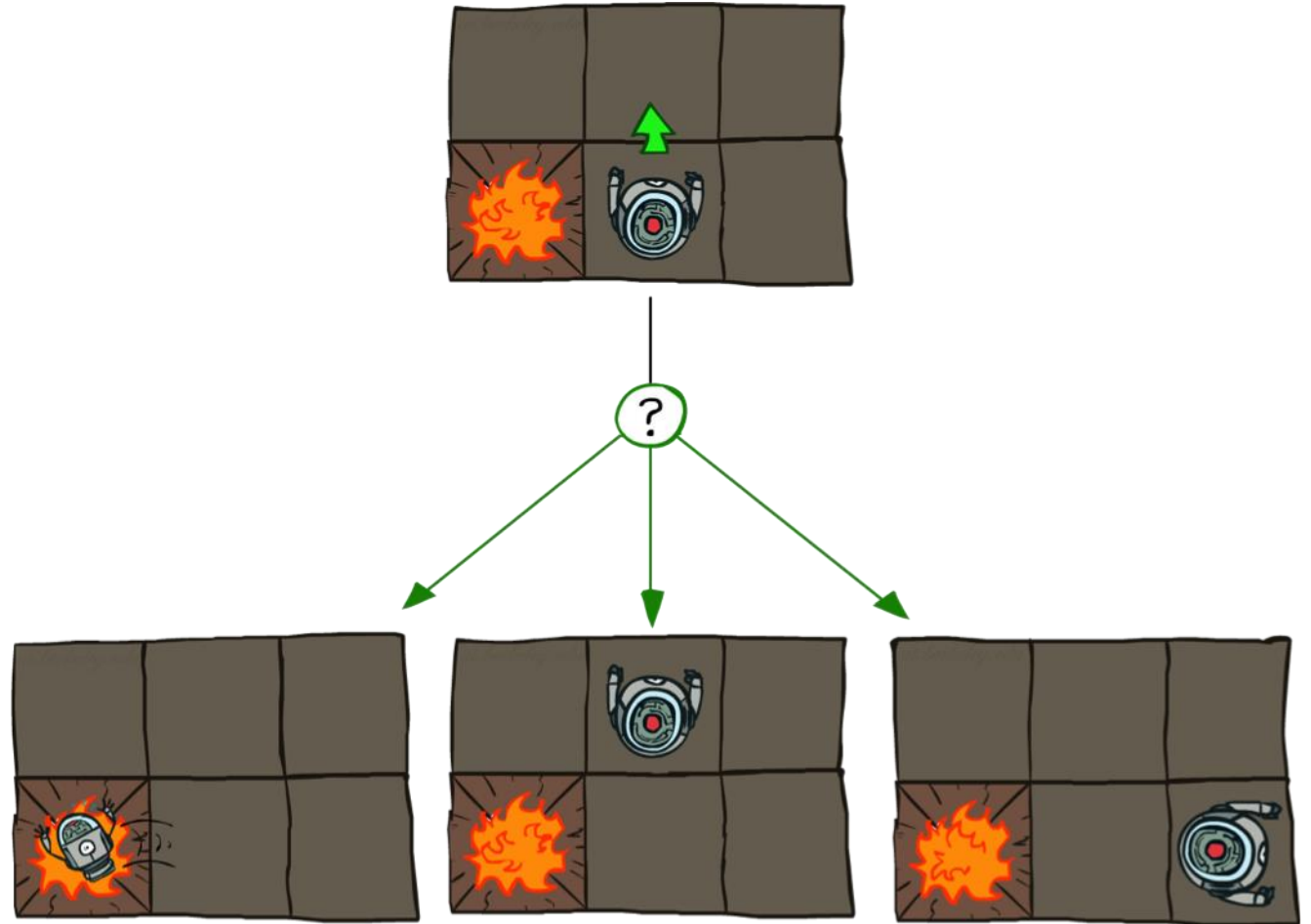


Mrežasti Svet - Akcije

Deterministički Mrežasti Svet



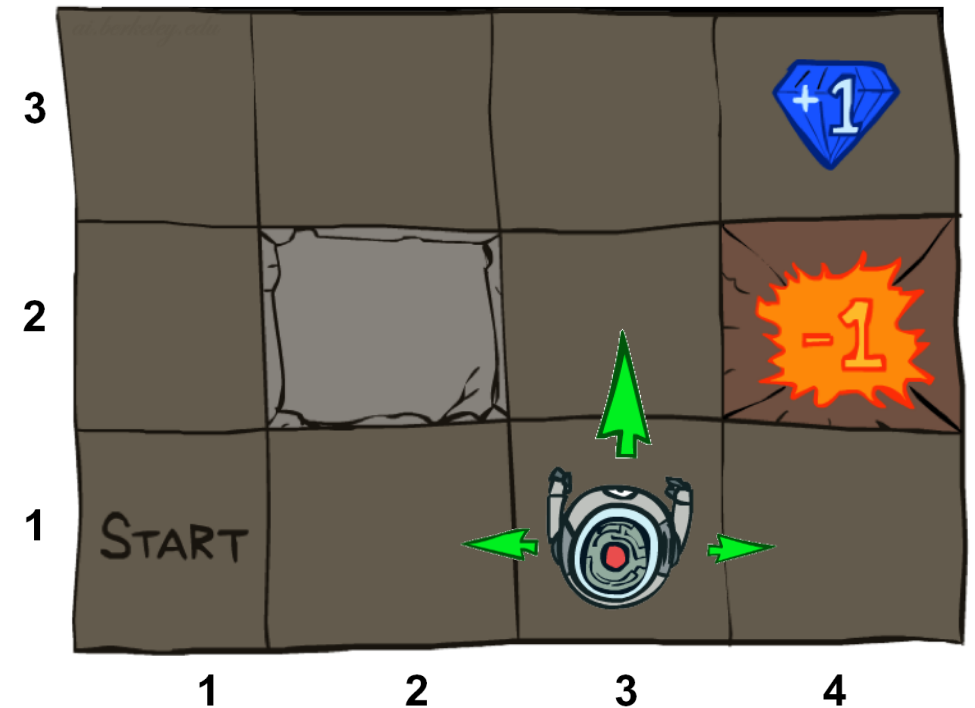
Stohastički Mrežasti Svet



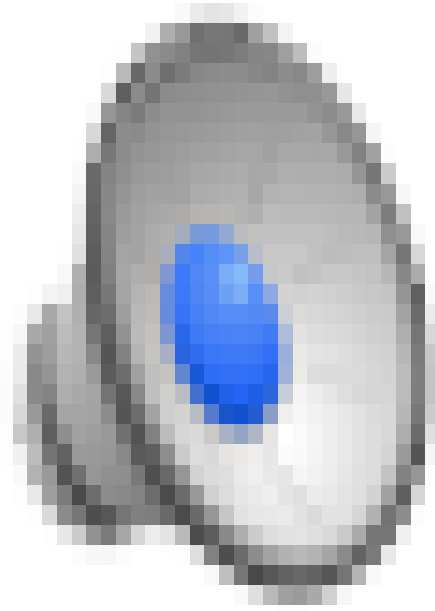
Markovljevi Procesi Odlučivanja

Markov Decision Processes, MDPs

- MDP je definisan sa:
 - Skupom stanja $s \in S$
 - Skupom akcija $a \in A$
 - Funkcijom prelaza (*transition function*) $T(s, a, s')$
 - Verovatnoća da a iz s vodi u s' tj. $P(s' | s, a)$
 - Funkcija nagrade (*reward function*) $R(s, a, s')$
 - Početnim stanjem
 - Može, ali ne mora, krajnjim (terminalnim) stanjem
- MDPs su ne-deterministički problemi pretrage
 - Jedan način da se reše je Expectimax
 - Uskoro ćemo prikazati drugi način



Demo Mrežasti Svet – Ručno Kretanje



Šta je Markovljevo Svojstvo u MDP?

- “Markovljevo” generalno znači da ako posmatramo trenutno stanje, prošlost i budućnost su nezavisni (ishod akcije koju sad radimo ne zavisi toga koje smo akcije radili pre nje).
- Za MDP, “Markovljevo svojstvo” znači da ishodi akcija zavise samo od stanja u kome ih radimo

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

=

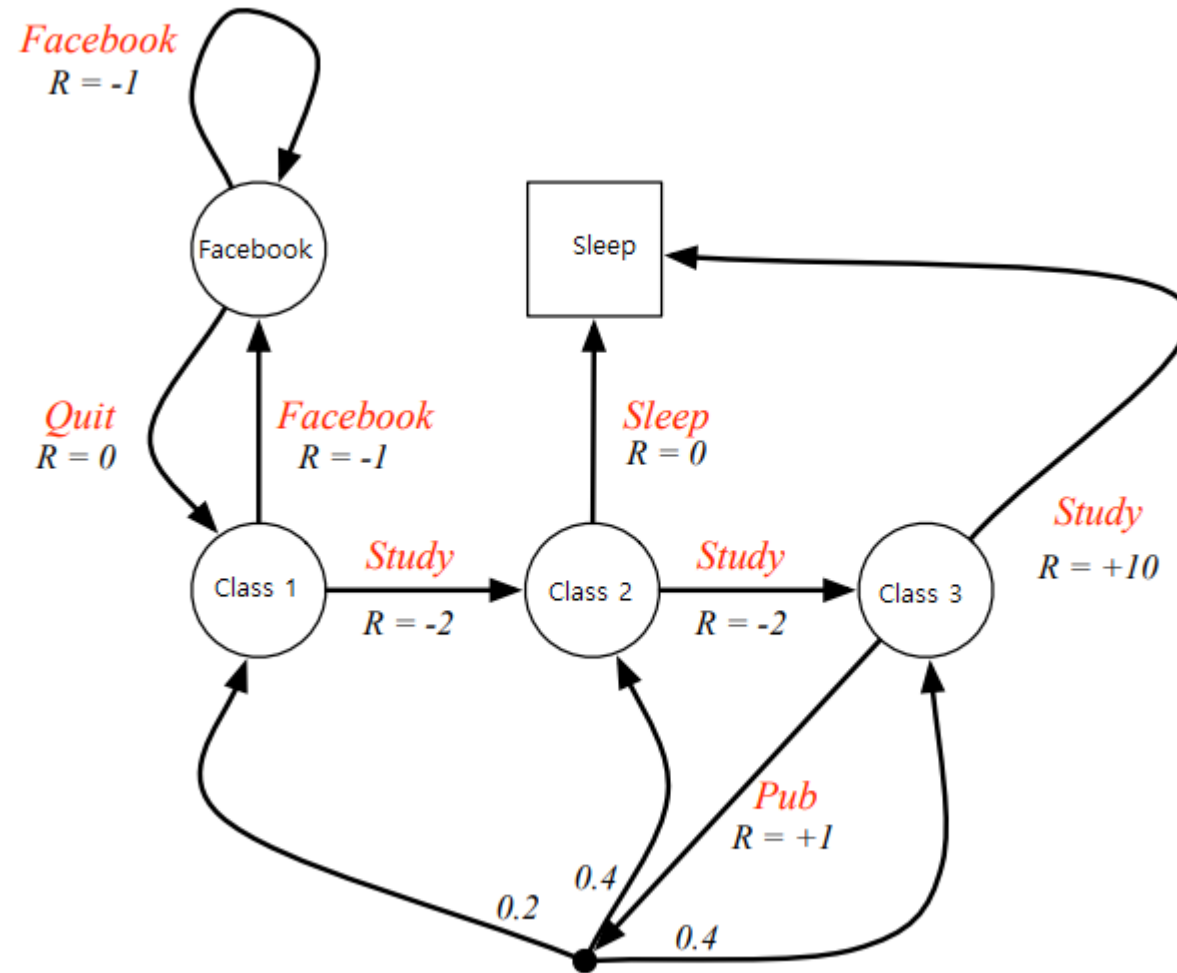
$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t)$$

- Ovo je kao kod stabla pretraživanja, čvorovi naslednici zavise samo od akcija u trenutnom čvoru, ne istorije akcija do njega.



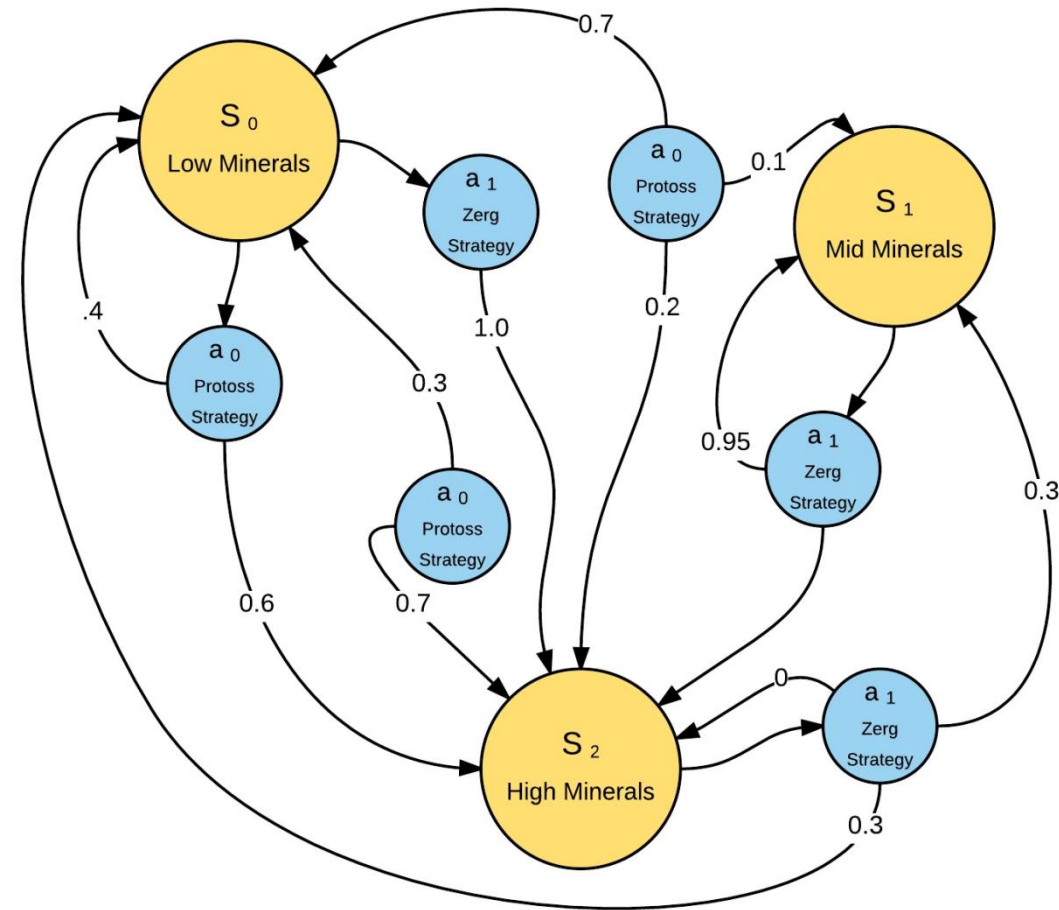
Andrey Markov
(1856-1922)

MDP – Primer



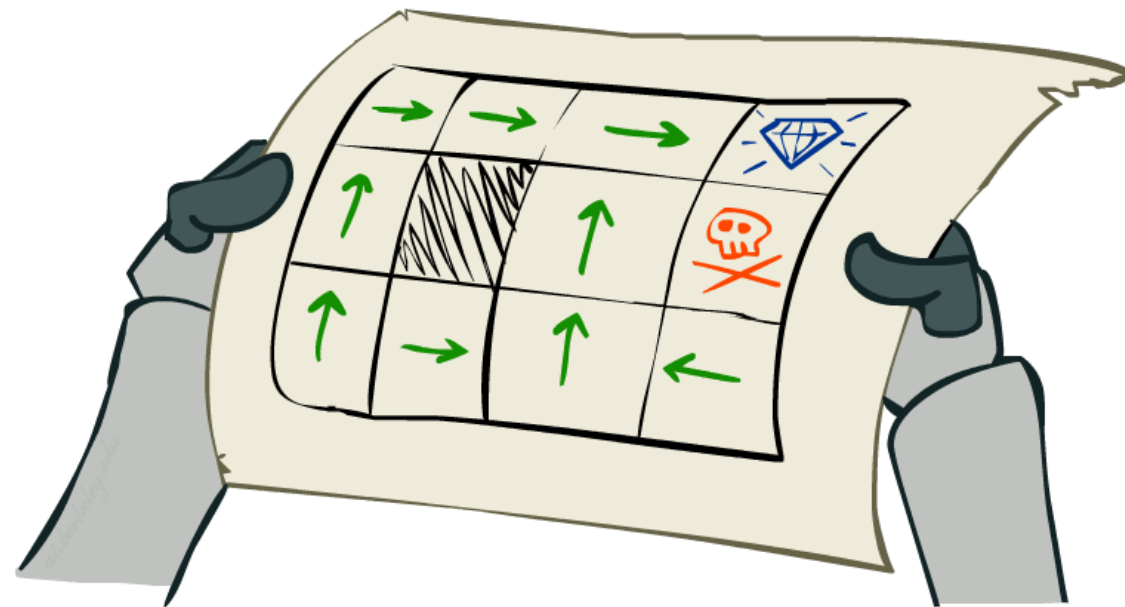
MPD - Primer

Example of a Markov Decision Process for Starcraft



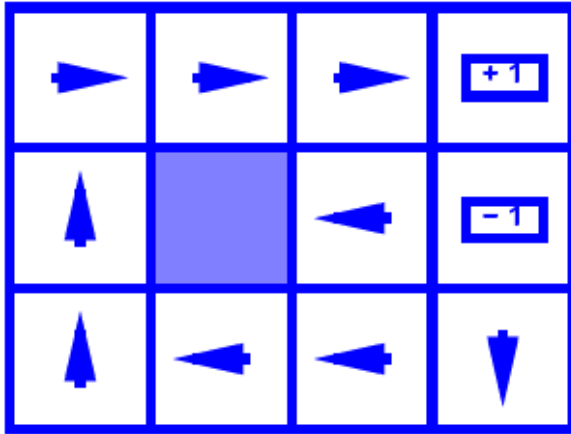
Politike (*Policies*)

- Kod determinističkih pretraga tražili smo **plan** tj. niz akcija koje nas vode od starta do cilja.
- Za MDPs, treba nam optimalna
- **politika** $\pi^*: S \rightarrow A$
 - Politika π za svako stanje kaže koju akciju treba da radimo.
 - Optimalna politika je ona koja, ako je pratimo, maksimizuje očekivanu korisnost.
 - Ako imamo eksplicitnu politiku onda je agent koji je prati refleks agent (*reflex agent*) – ne re-planira samo prati politiku.

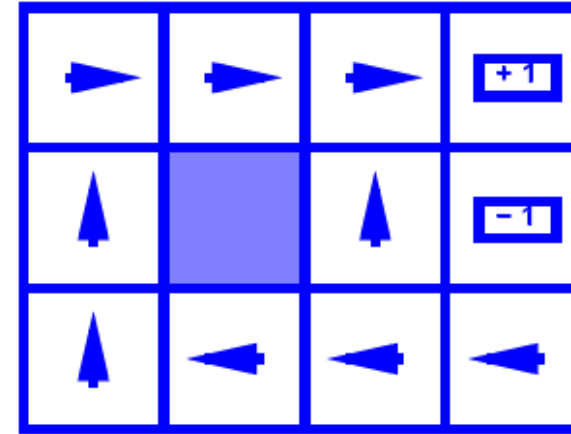


Na slici je optimalna politika kad važi $R(s, a, s') = -0.03$ za sva ne-terminalna stanja

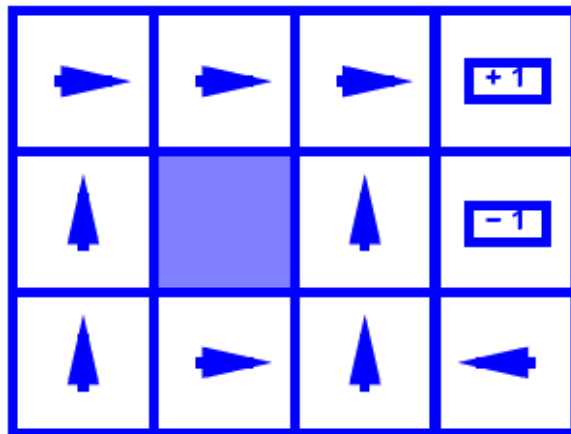
Optimalne Politike Za Date Nagrade (tačnije kazne)



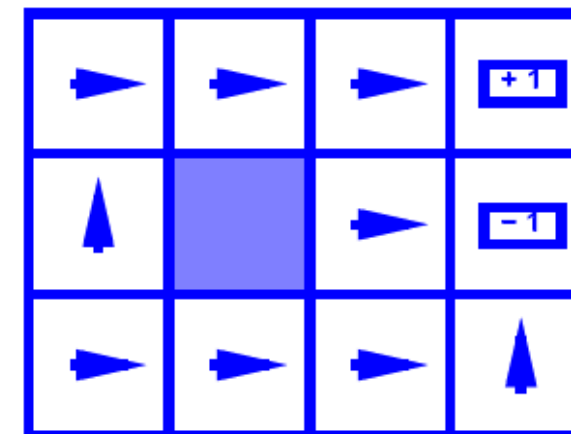
$$R(s) = -0.01$$



$$R(s) = -0.03$$

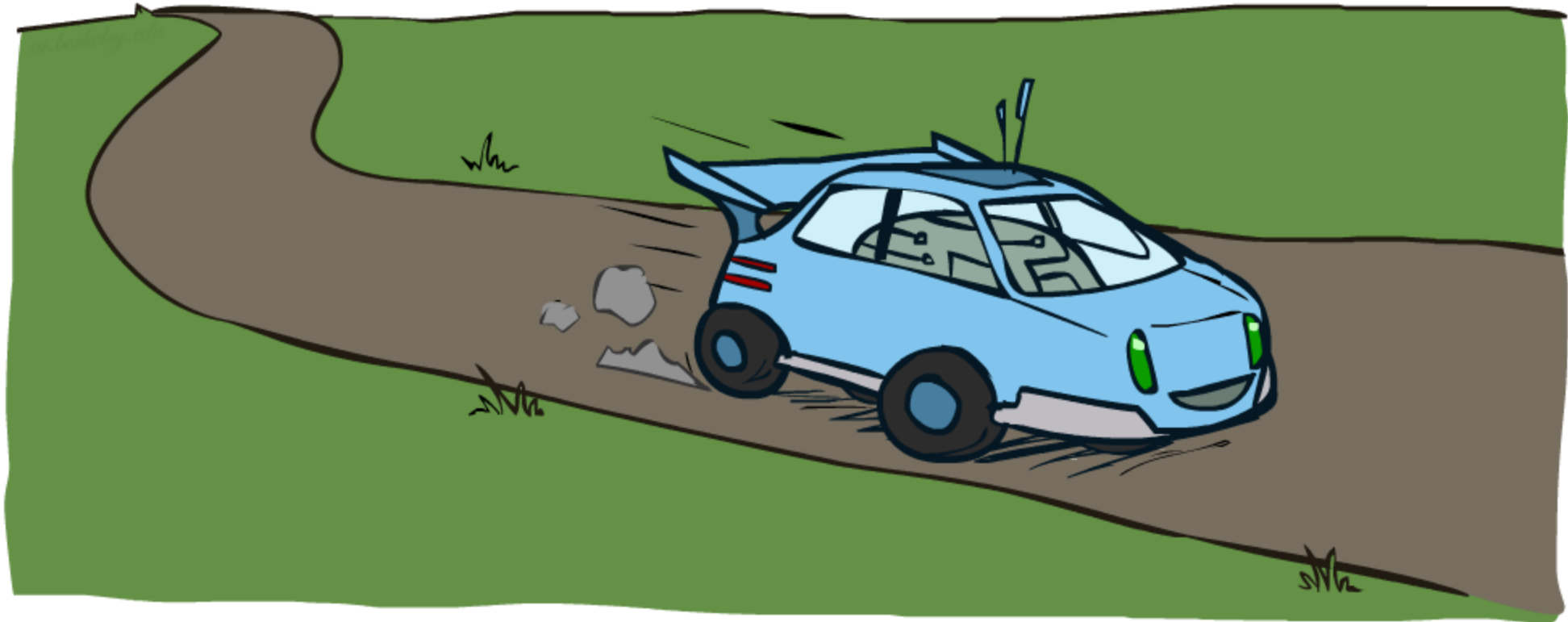


$$R(s) = -0.4$$



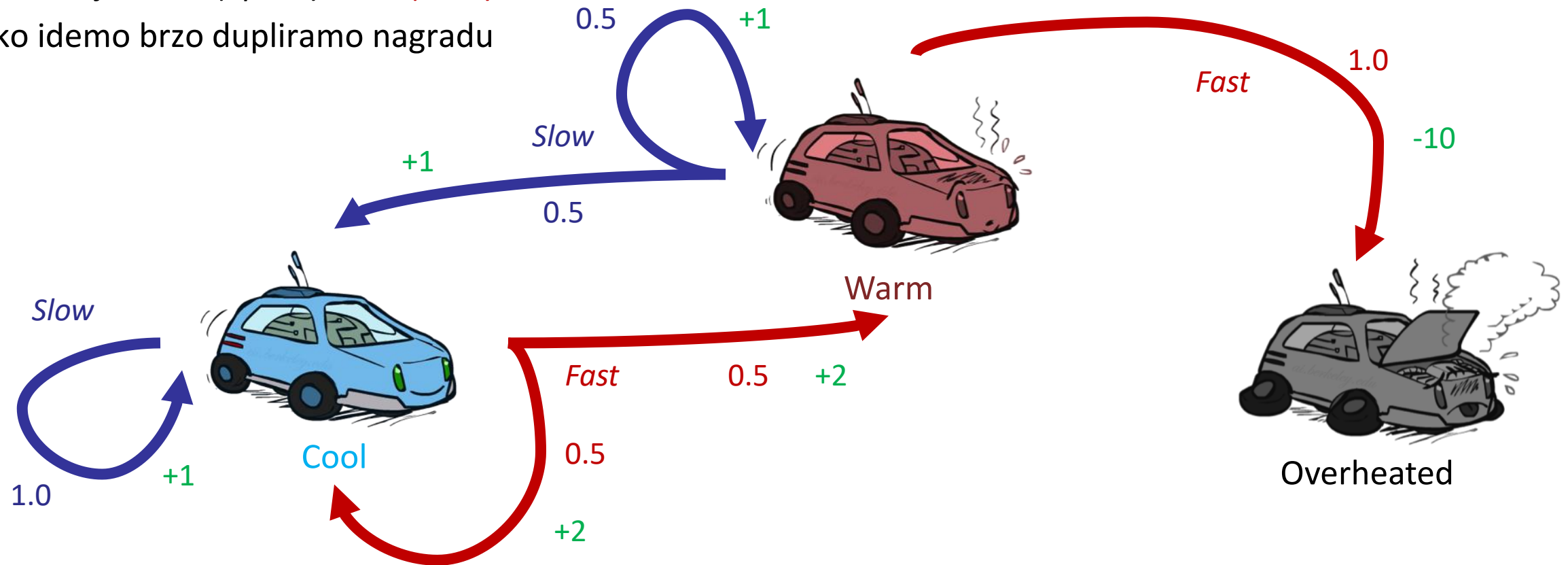
$$R(s) = -2.0$$

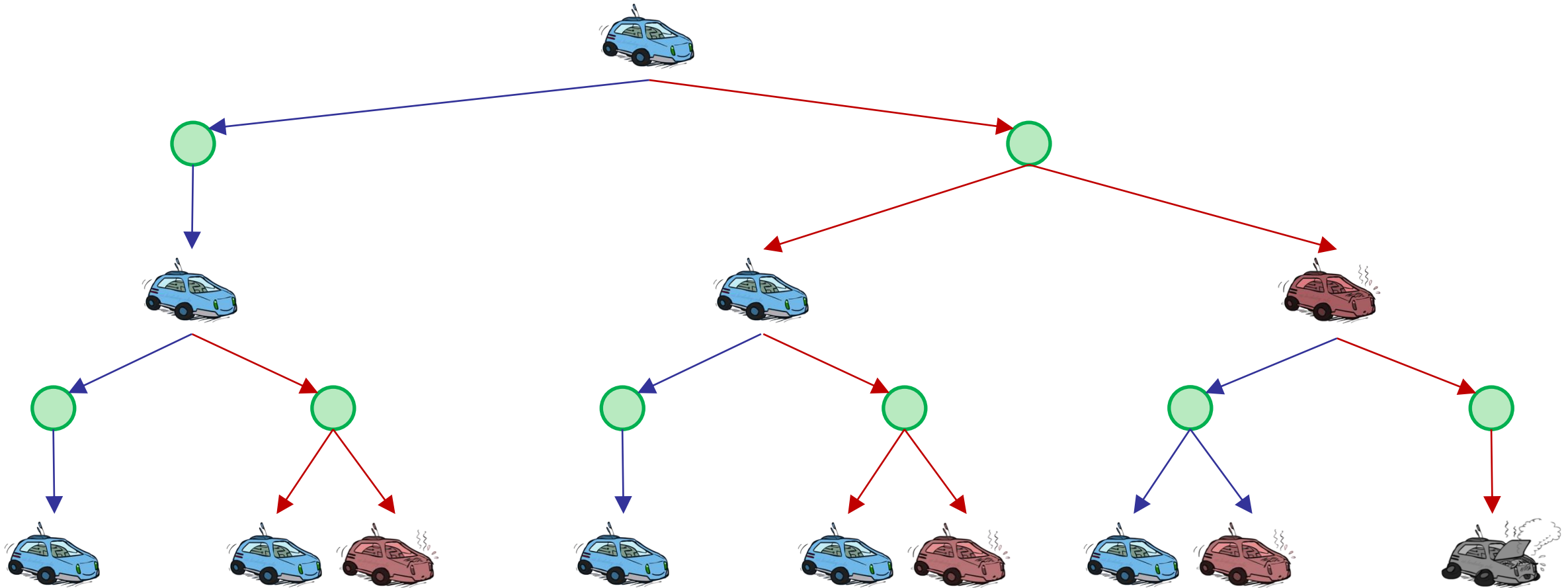
Primer: Trka Automobilom



Primer: Trka Automobilom

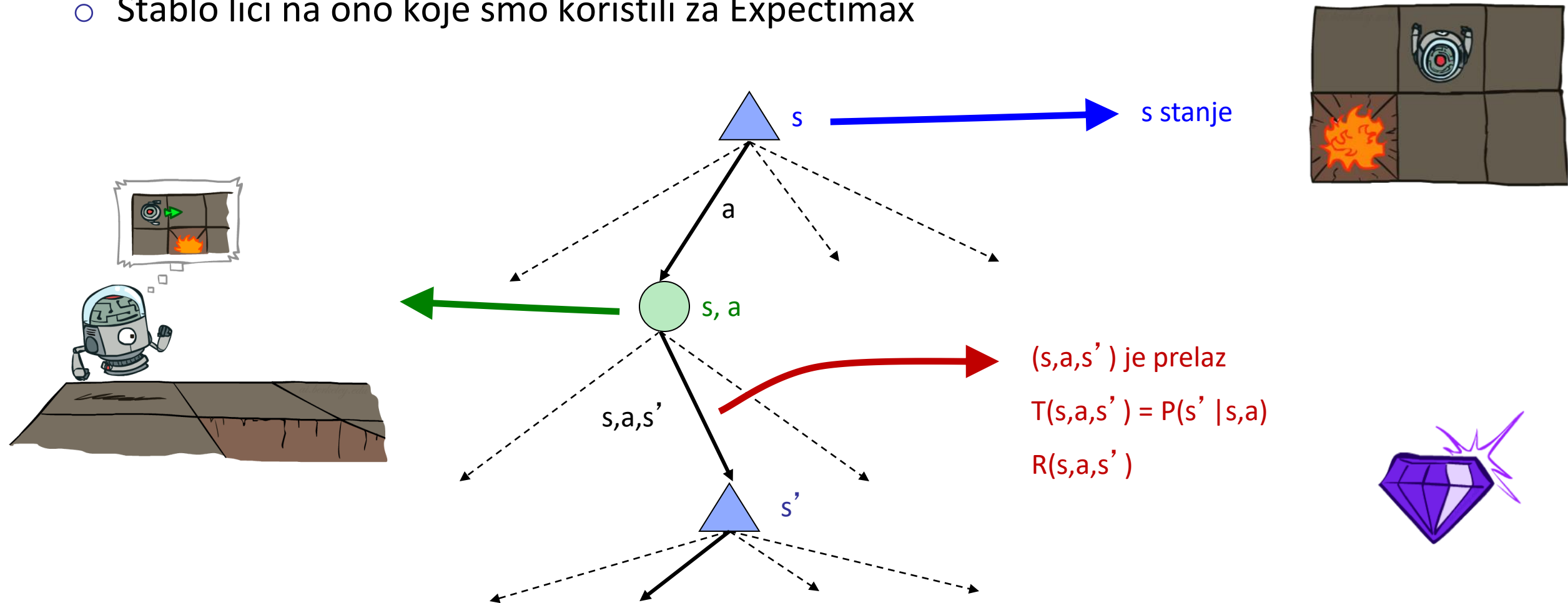
- Autonomni automobil želi da ode što dalje što brže
- Tri stanja: **Cool (Hladan)**, **Warm (Topao)**, Overheated (Pregrejan)
- Dve akcije: *Slow (Sporo)*, *Fast (Brzo)*
- Ako idemo brzo dupliramo nagradu



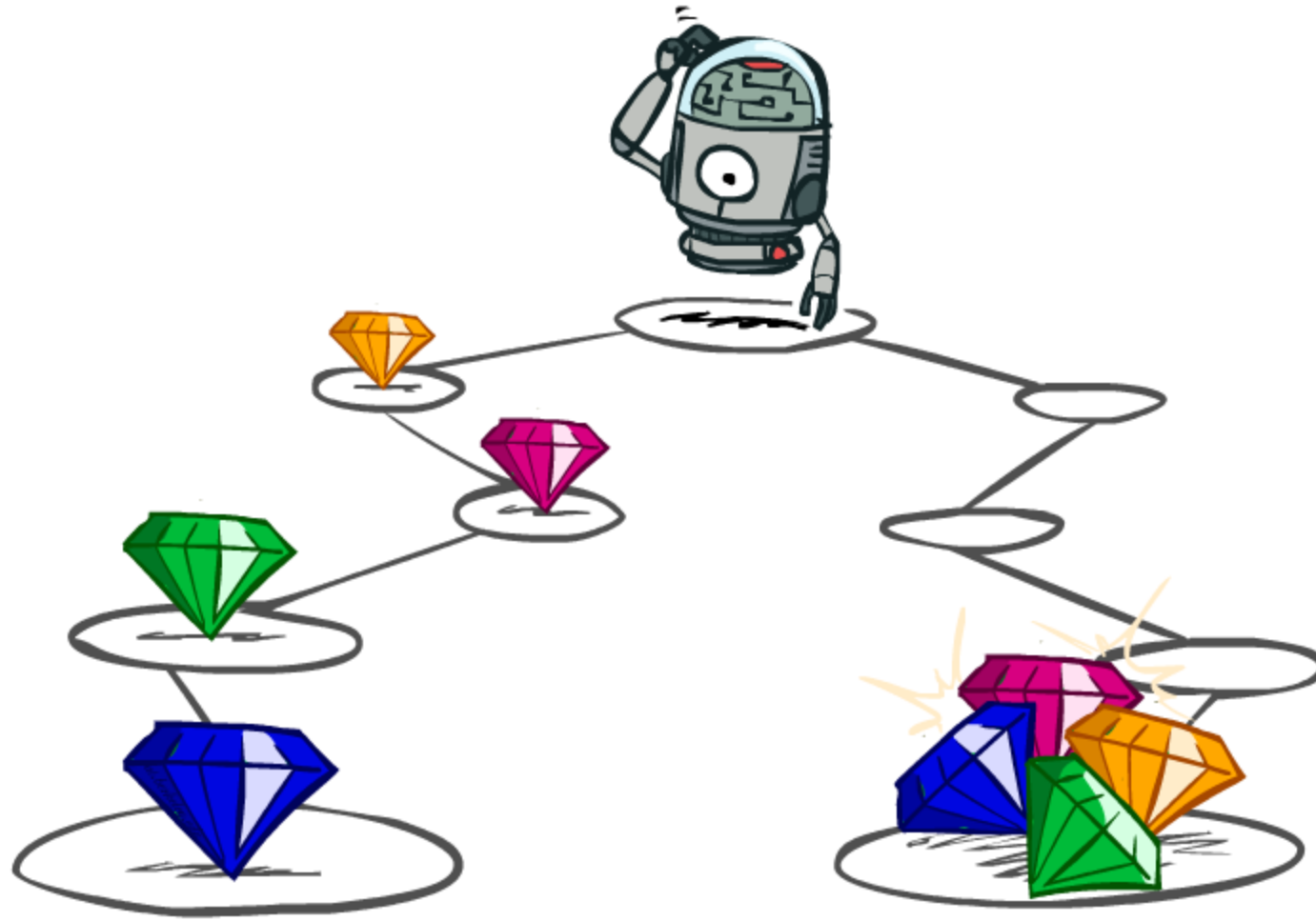


MDP – Stablo Pretraživanja

- Stablo liči na ono koje smo koristili za Expectimax

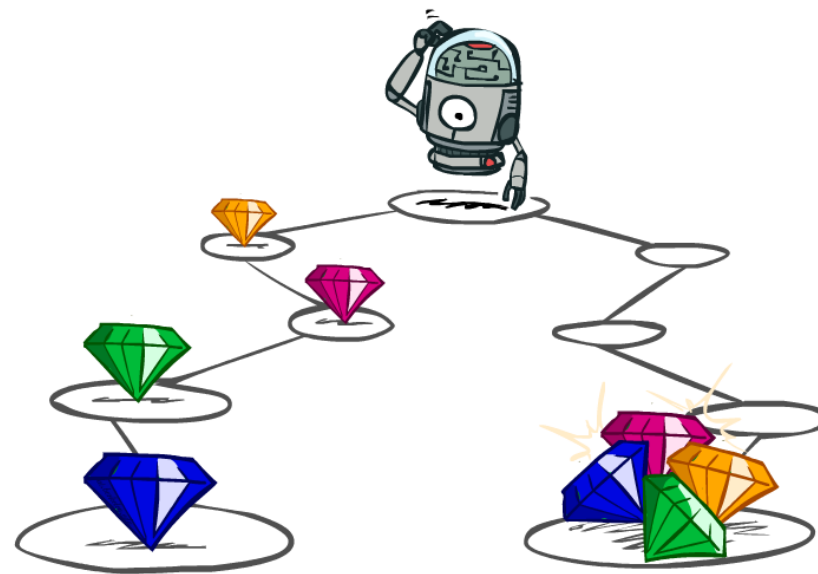


Korisnosti Sekvenci Akcija



Korisnosti Sekvenci Akcija

- Ako imamo neke sekvence nagrada koje dobijamo, koje od sekvenci bi naš agent trebalo da preferira?
- Manje ili više? $[1, 2, 2]$ ili $[2, 3, 4]$
- Sad ili kasnije? $[0, 0, 1]$ ili $[1, 0, 0]$
- Preference su vrlo zdravo-razumske
- Hoćemo što veću nagradu što pre moguće



Zanemarivanje nagrada (*Discounting*)

- Razumno je da želimo da maksimizujemo sumu nagrada
- Razumno je da su nam nagrade koje dobijamo sad važnije od onih koje dobijamo kasnije (1M\$ danas vs. 1M\$ kada imamo 90 godina)
- Jedno rešenje: vrednosti nagrada opadaju eksponencijalno



1

Vrednost sad



γ

Vrednost u
sledećem koraku

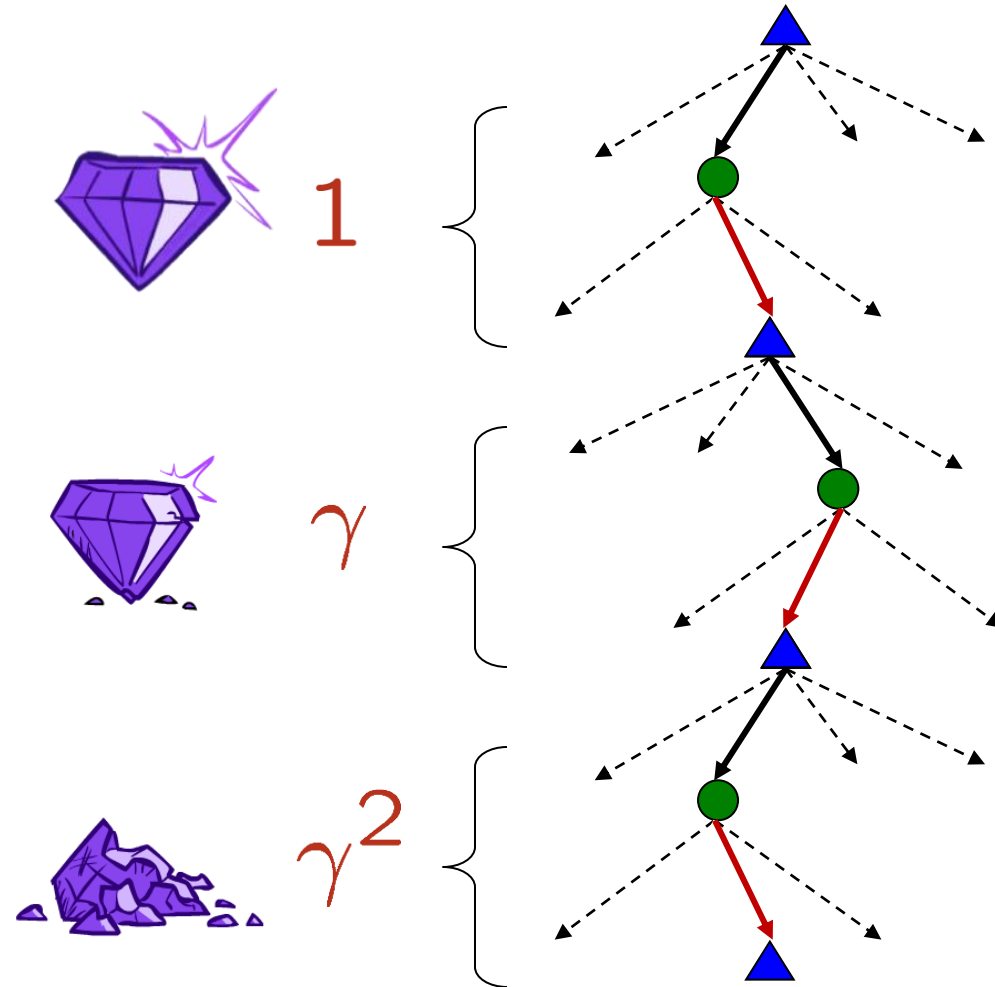


γ^2

Vrednost posle dva
koraka

Zanemarivanje

- Kako radimo zanemarivanje?
 - Svaki put kada se spustimo za nivo množimo faktorom zanemarivanja
- Zašto radimo zanemarivanje?
 - Nagrade koje dobijamo pre verovatno više doprinose ukupnoj korisnosti nego one kasnije
 - Pomaže nam kod konvergencije algoritama (uzimamo bez dokaza, a algoritmi će biti dati u nastavku)
- Primer: faktor zanemarivanja 0.5
 - $U([1,2,3]) = 1*1 + 0.5*2 + 0.25*3$
 - $U([1,2,3]) < U([3,2,1])$



Kviz: Zanemarivanje

- Ako je dato:

10				1
a	b	c	d	e

- Akcije su: Istok, Zapad i Izlaženje (moguće samo u stanjima izalaza a i e)
- Prelazi: deterministički
- Napomena: bodove u stanjima a i e dobijamo tek kada uradimo akciju Izlaženje koja sigurno uspeva.

10				1
----	--	--	--	---

- Pitanje 1: Za $\gamma = 1$, koja je optimalna politika?

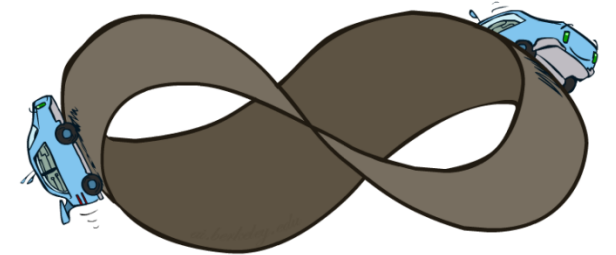
10				1
----	--	--	--	---

- Pitanje 2: Za $\gamma = 0.1$, koja je optimalna politika?

- Pitanje 3: Za koje γ su Zapad i Istok jednako dobre kad smo u stanju d?

Beskonače Korisnosti?!

- Problem: Šta ako igra nema kraj? Da li onda dobijamo beskonačne nagrade?
- Rešenje:
 - Konačni horizont: (slično kao pretraga za zadatu dubinu)
 - Završavamo sekvence akcija (epizode) posle T koraka (npr. negativna nagrada „postojanja“)
 - Politika π sad zavisi od vremena koje nam „otkucava“



- Zanemarivanje: koristimo faktor $0 < \gamma < 1$

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

- Manje γ znači manji “horizont” – fokusiramo se samo na ono što nam je sada blizu
 - Ovako smo ograničili korisnost tj. obezbedili da je ona uvek manja od neke konstante pa nije beskonačna
- Apsorbujuće stanje: garancija da će svaka politika stići u terminalno stanje kad tad (npr. Pregrejano kod primera sa Trkom)

Beskonače Korisnosti?!

- Dokaz za:

- Zanemarivanje: koristimo faktor $0 < \gamma < 1$

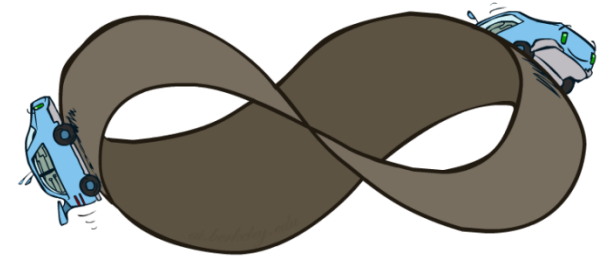
$$U([r_0, \dots, r_\infty]) = \sum_{t=0}^{\infty} \gamma^t r_t \leq R_{\max}/(1 - \gamma)$$

- Na osnovu sledećeg tvrđenja vezanog za beskonačne geometrijske redove:

$$\sum_{t=0}^{\infty} \gamma^t a = \frac{a}{1 - \gamma}, a = \text{const}, |\gamma| < 1$$

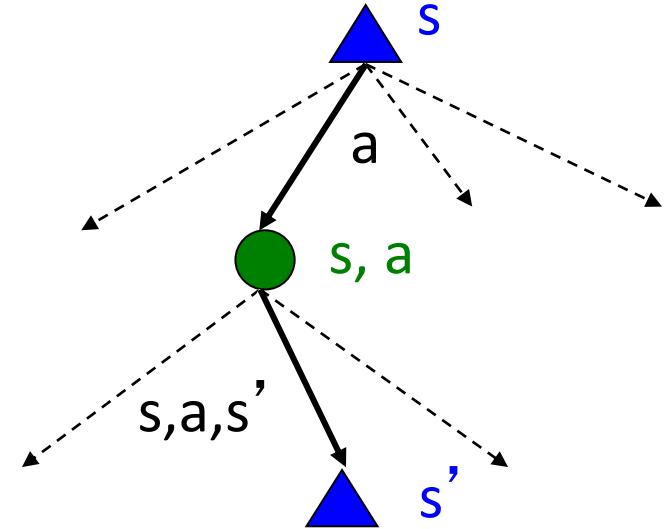
- i čijenice da možemo da r_t ograničimo (tj. zamenimo) sa nekom maksimalnom nagradom R_{\max} dobijamo:

$$\sum_{t=0}^{\infty} \gamma^t r_t \leq \sum_{t=0}^{\infty} \gamma^t R_{\max} \leq \frac{R_{\max}}{1 - \gamma}$$

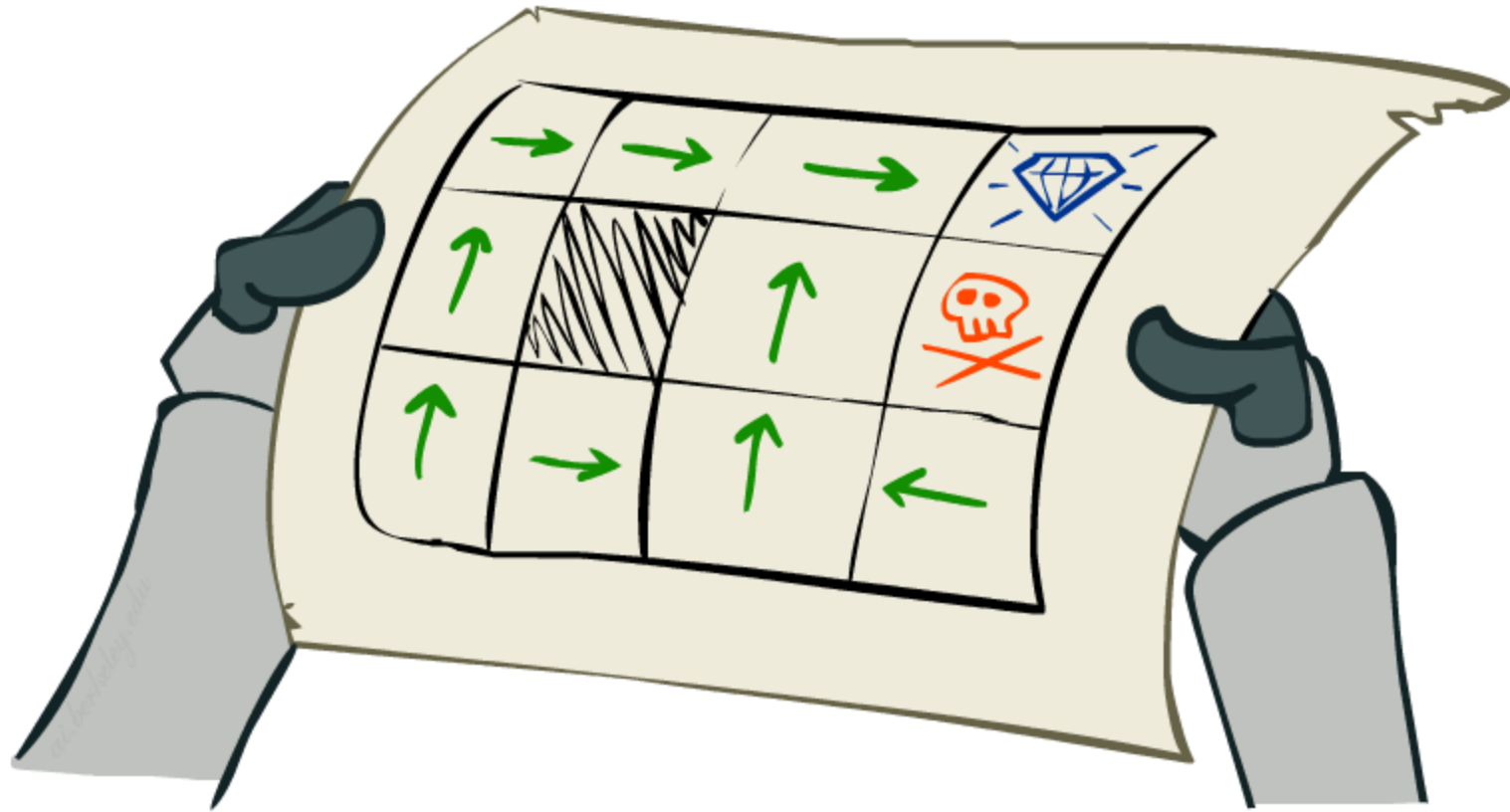


Rezime: Definicija MDP

- MDP je definisan sa:
 - Skupom stanja S
 - Početnim stanjem s_0
 - Skupom akcija A
 - Prelazima $P(s' | s, a)$ (ili $T(s, a, s')$)
 - Nagradama $R(s, a, s')$ (i zanemarivanjem γ)
- MDP vrednosti koje izračunavamo:
 - Politika = koju akciju odabrati u kom stanju
 - Korisnost = suma nagrada uz zanemarivanje



Rešavanje MDP



Optimalne Vrednosti za MDP

- Optimalna Vrednost (korisnost) za stanje s :

$V^*(s)$ = očekivana korisnost ako krenemo iz s i dalje radimo uvek optimalne akcije

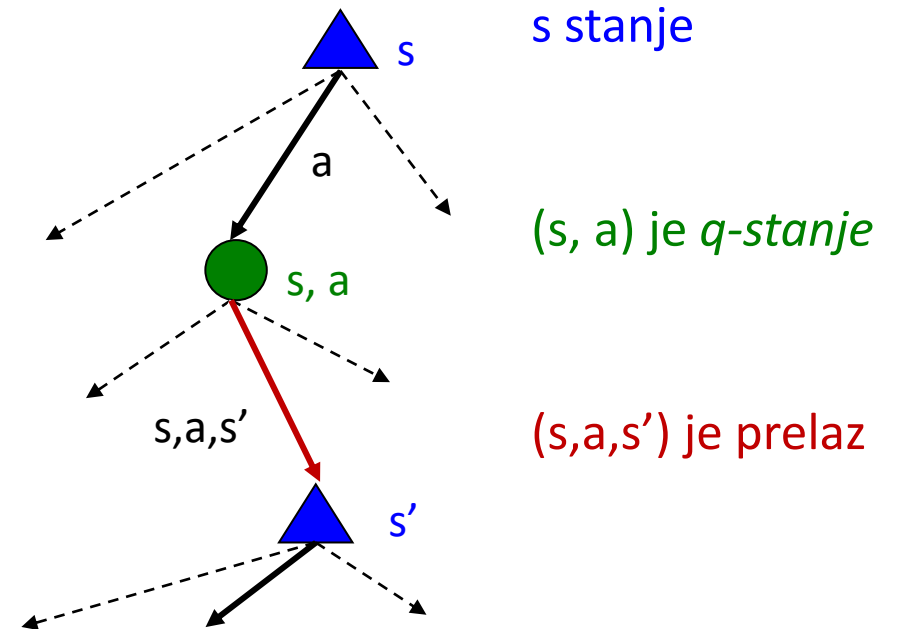
- Optimalna Vrednost (korisnost) q -stanja (s,a) :

$Q^*(s,a)$ = očekivana korisnost ako uradimo akciju a u stanju s (koja ne mora da bude optimalna), ali dalje radimo uvek optimalne akcije

- Optimalna politika:

$\pi^*(s)$ = optimalna akcija u stanju s

Drugim rečima uvek vraća akciju koja je optimalna baš za stanje u kome se nalazimo



Slika Demoa – Mrežasti Svet V Vrednosti

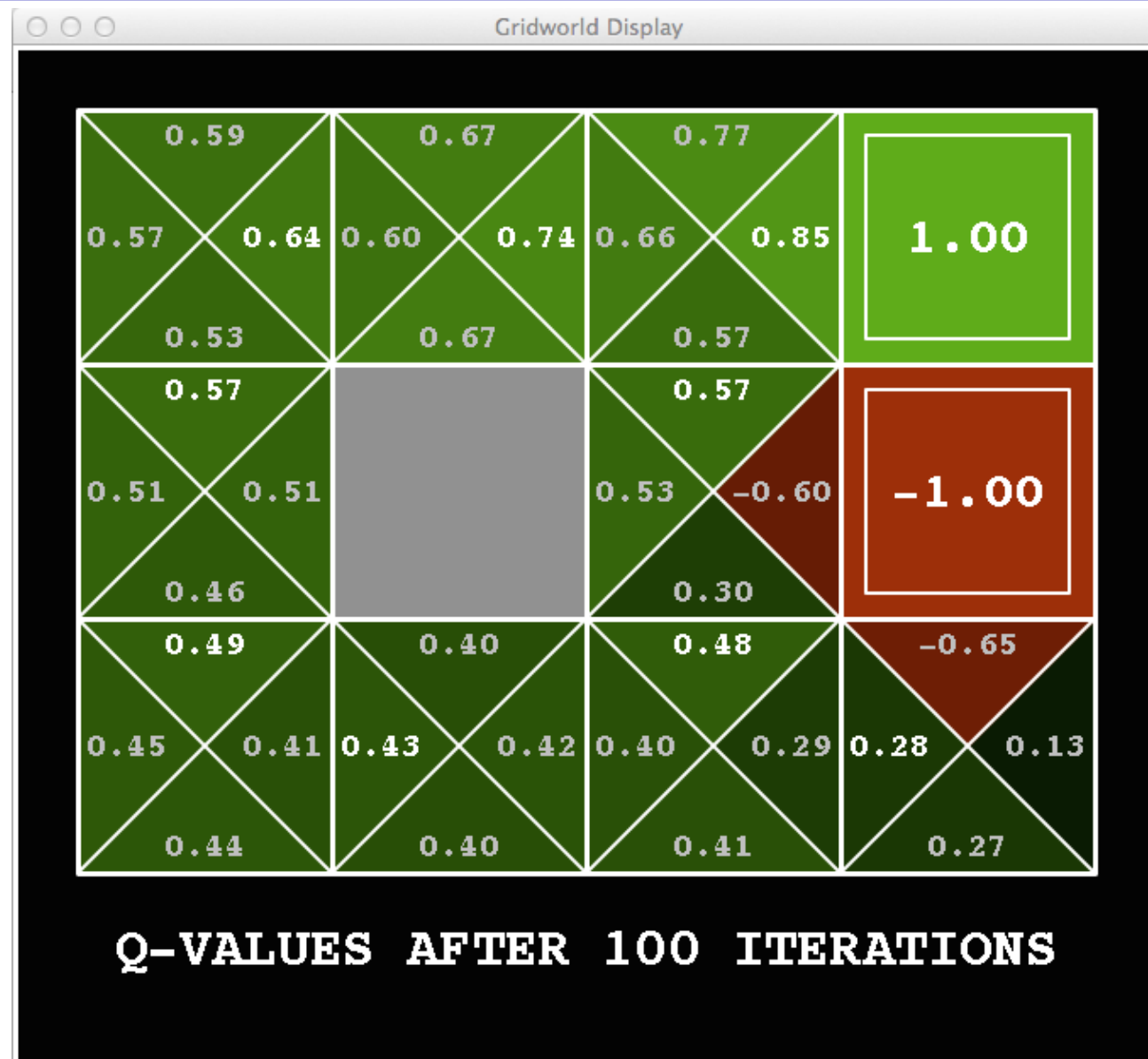


Šum (slučajnost za akcije) = 0.2

Zanemarivanje = 0.9

Nagrada postojanja = 0

Slika Demoa – Mrežasti Svet Q Vrednosti



Šum (slučajnost za akcije) = 0.2
Zanemarivanje = 0.9
Nagrada postojanja = 0

Vrednosti Stanja

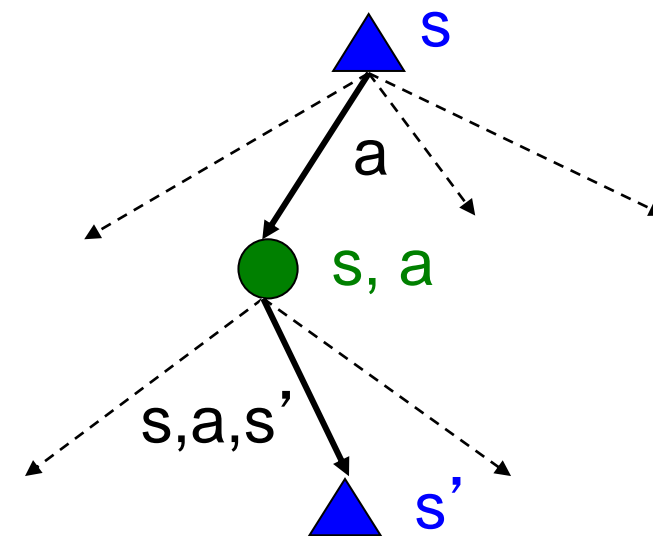
- Važna operacija: izračunavamo (expectimax) vrednost stanja
 - Očekivanu nagradu uz optimalne akcije
 - To je prosečna suma nagrada uz zanemarivanje
 - Ovo je vrednost koju je izračunavao expectimax!

- Rekurzivna definicija vrednosti:

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

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

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



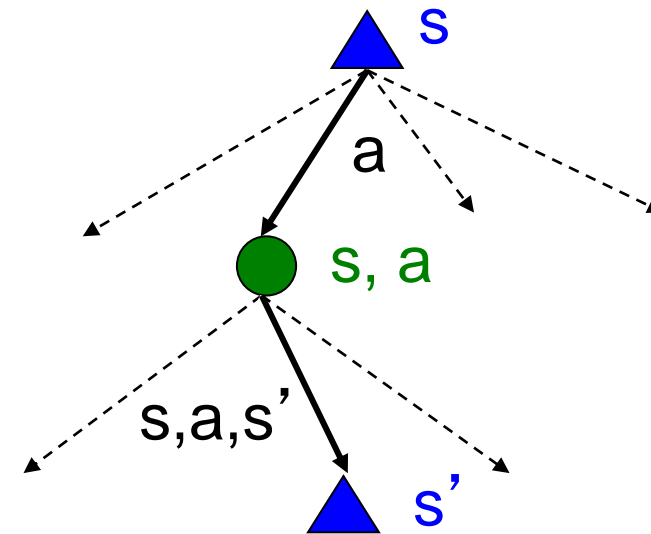
Vrednosti Stanja

- Rekurzivna definicija vrednosti:

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

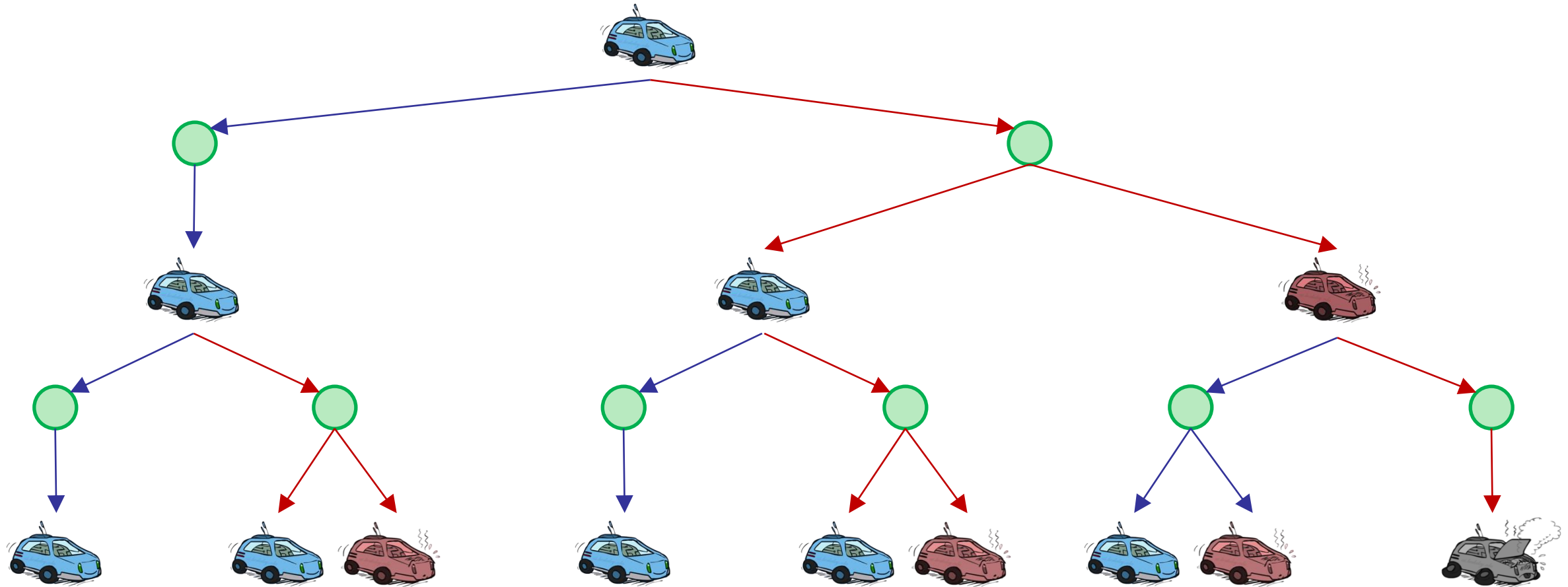
$$Q^*(s, a) = \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$

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

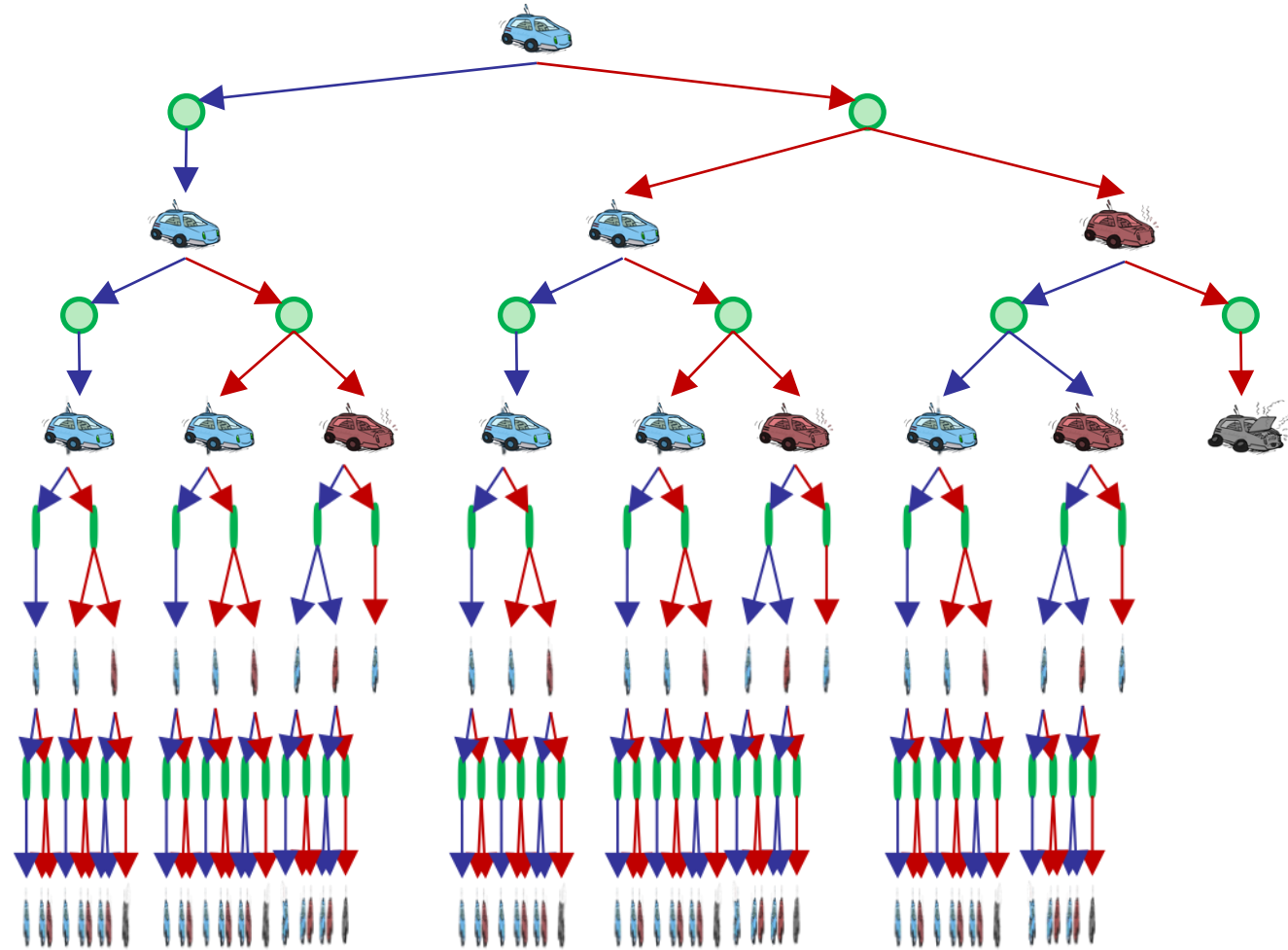


- Date formule nazivaju se Belmanove (*Bellman*) jednakosti ili Belmanove promene (*Bellman updates*)

Stablo Pretraživanja za Trku

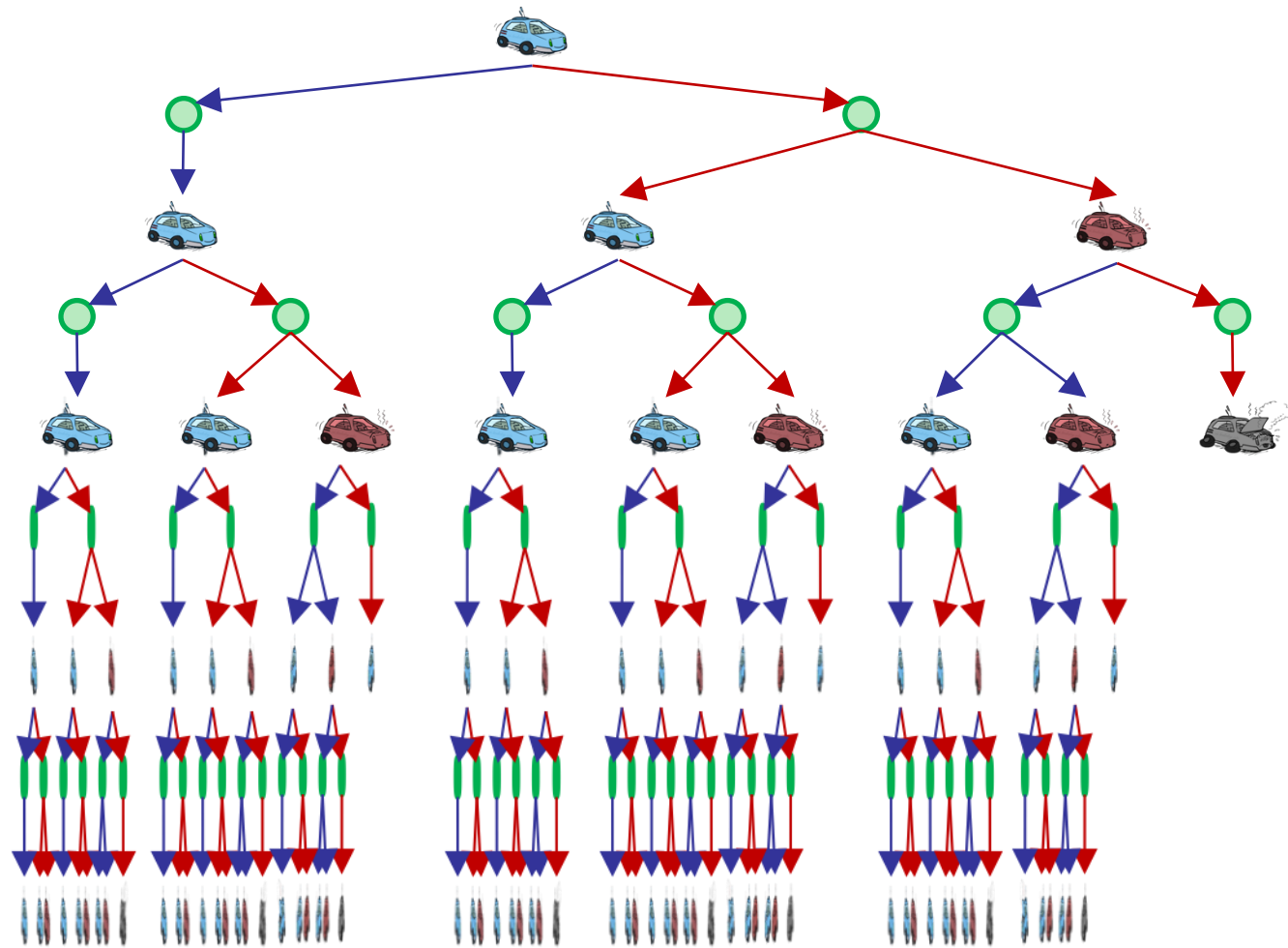


Stablo Pretraživanja za Trku



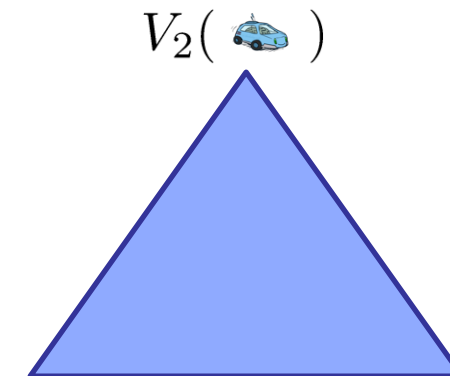
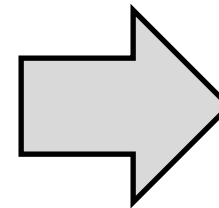
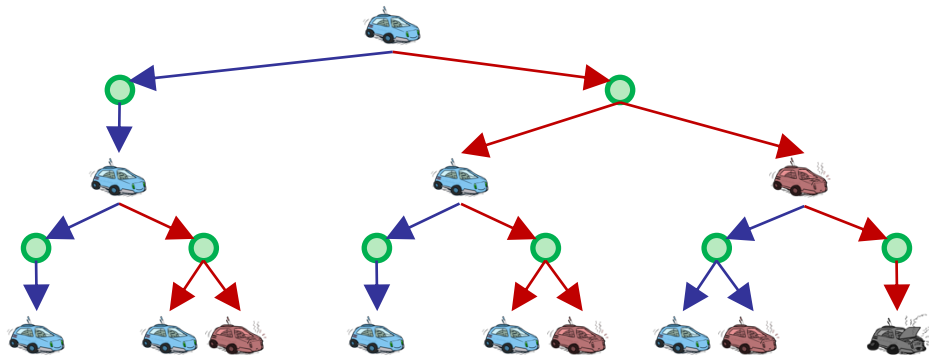
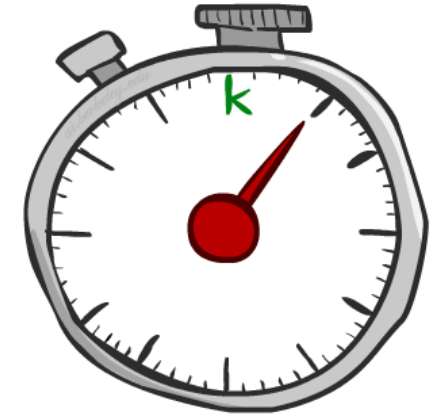
Stablo Pretraživanja za Trku

- Ako bi koristili expectimax, imali bi previše posla!
- Problem: Stanja se ponavljaju
 - Ideja: Izračunavati potrebne vrednosti samo jednom
- Problem: Stablo ide do beskonačnosti
 - Ideja: Radimo izračunavanja do ograničene dubine, ali malo po malo povećavamo dubinu dok promena vrednosti ne bude zanemarljiva
 - Napomena: duboki nivoi stabla nam nisu mnogo važni ako je $\gamma < 1$

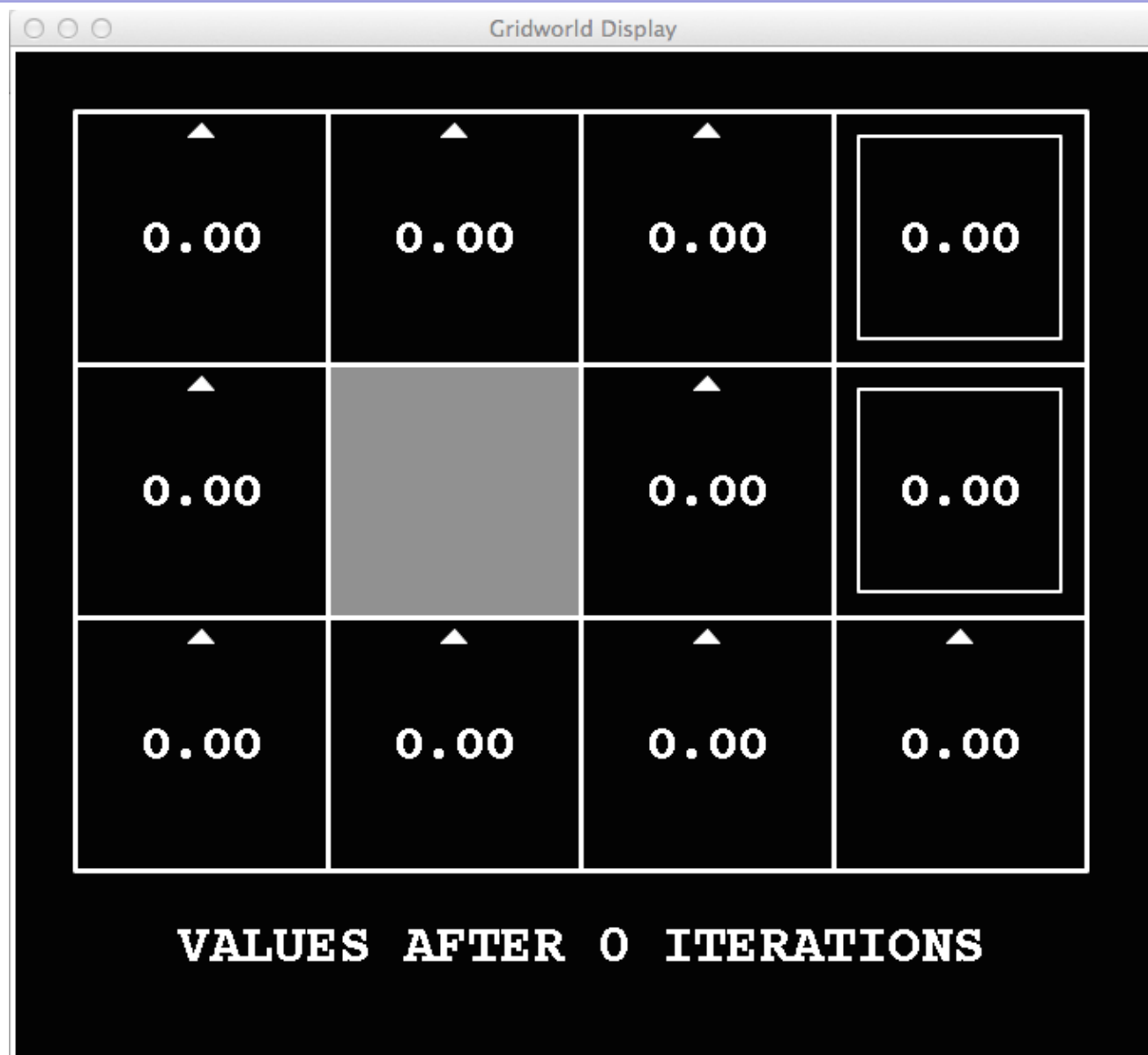


Vrednosti Ograničene Vremenom

- Ideja: vezujemo vrednosti za vreme
- Definišemo $V_k(s)$ kao optimalnu vrednost u s ako se igra završava za k koraka
 - To je vrednost koju bi s dobio od expectimax algoritma sa ograničenom dubinom k



$k=0$

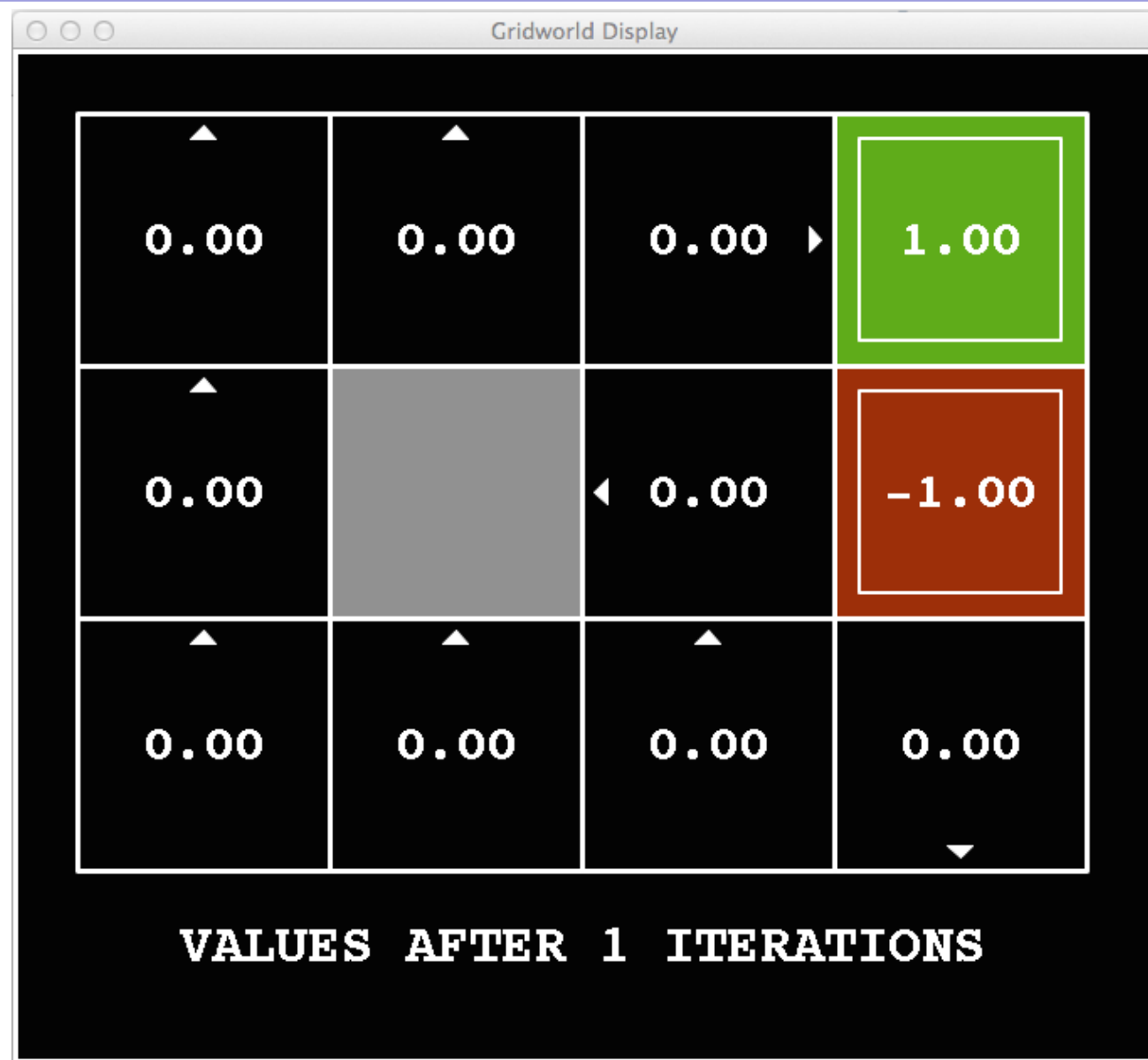


Šum = 0.2

Zanemarivanje = 0.9

Nagrada postojanja = 0

$k=1$



Šum = 0.2

Zanemarivanje = 0.9

Nagrada postojanja = 0

k=2

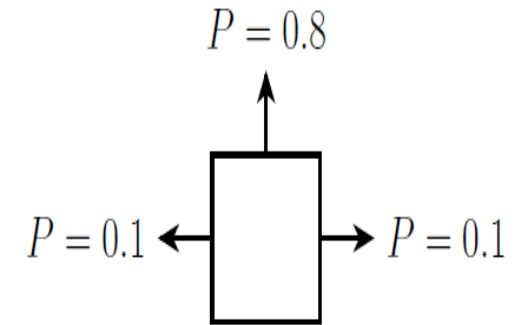
Posmatramo stanje s koje
sa vrednošću 0.72.
Objasnićemo kako smo
dobili ovu rednost.

$V(s) = \max Q(s, a)$
 $a = \text{north, south, east, west}$

Nećemo računati sve Q
(iako bi trebalo) već samo
ono koje će u ovoj situaciji
sigurno biti maksimalno, a
to je $Q(s, \text{east})$.

$$Q(s, \text{east}) = 0.8 * (0 + 0.9 * 1) + 0.1 * 0 + 0.1 * 0 = 0.72$$

Akcija east uspeva 0.8
puta, a 0.1 će biti north ili
south.



Šum = 0.2
Zanemarivanje = 0.9
Nagrada postojanja = 0

$k=3$



Šum = 0.2

Zanemarivanje = 0.9

Nagrada postojanja = 0

$k=4$



Šum = 0.2

Zanemarivanje = 0.9

Nagrada postojanja = 0

k=5



Šum = 0.2

Zanemarivanje = 0.9

Nagrada postojanja = 0

k=6



Šum = 0.2
Zanemarivanje = 0.9
Nagrada postojanja = 0

$k=7$



Šum = 0.2

Zanemarivanje = 0.9

Nagrada postojanja = 0

$k=8$



Šum = 0.2

Zanemarivanje = 0.9

Nagrada postojanja = 0

$k=9$



Šum = 0.2

Zanemarivanje = 0.9

Nagrada postojanja = 0

k=10



Šum = 0.2

Zanemarivanje = 0.9

Nagrada postojanja = 0

k=11



Šum = 0.2

Zanemarivanje = 0.9

Nagrada postojanja = 0

k=12



Šum = 0.2

Zanemarivanje = 0.9

Nagrada postojanja = 0

k=100

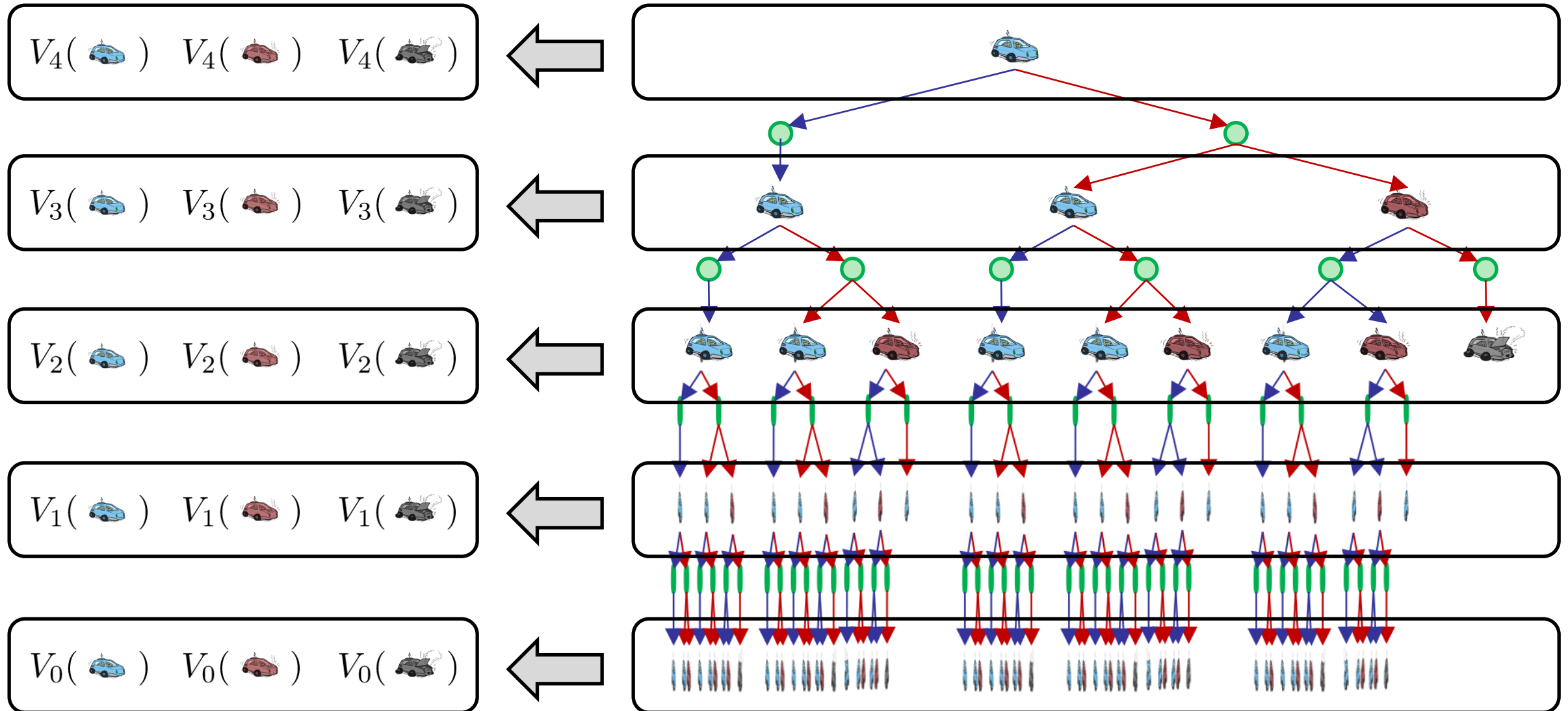


Šum = 0.2

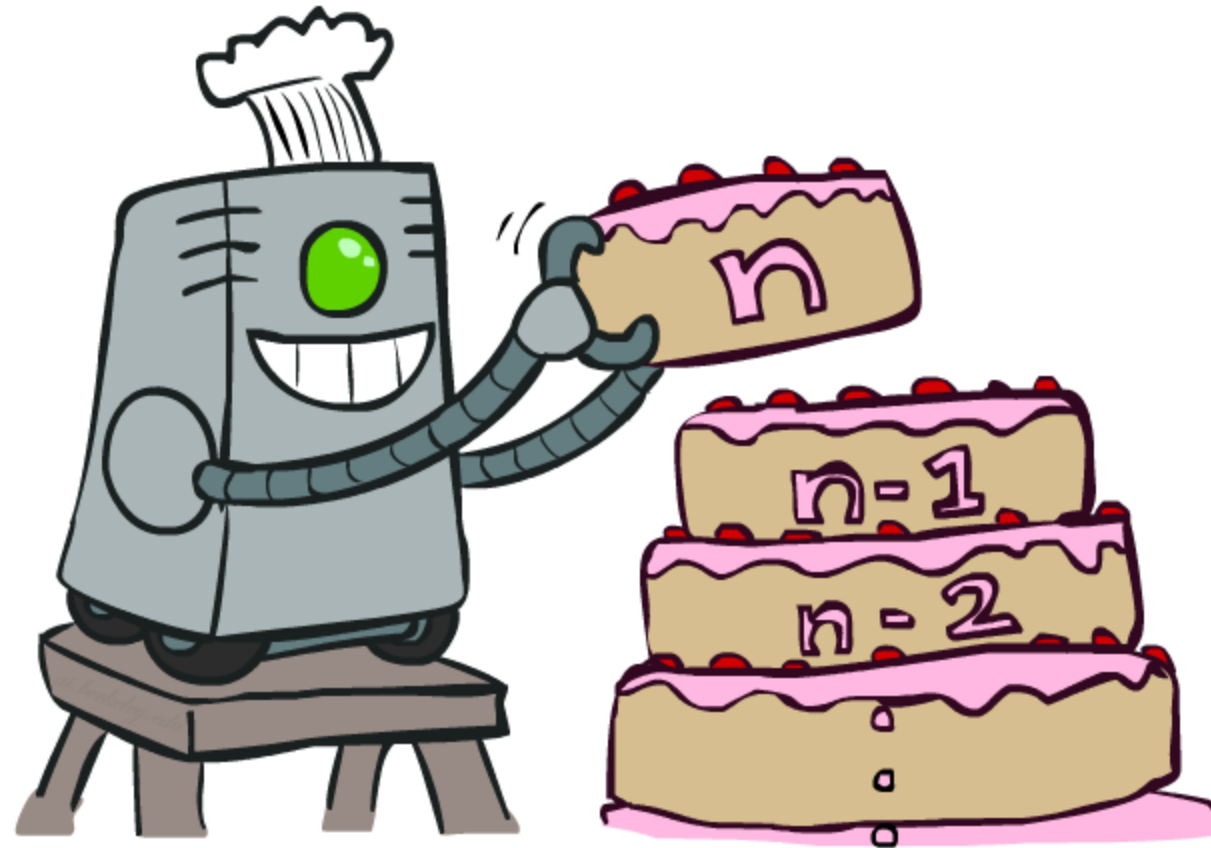
Zanemarivanje = 0.9

Nagrada postojanja = 0

Izračunavanje Vrednosti Ograničenih Vremenom



Algoritam: Iteriranje Vrednosti (*Value Iteration*)

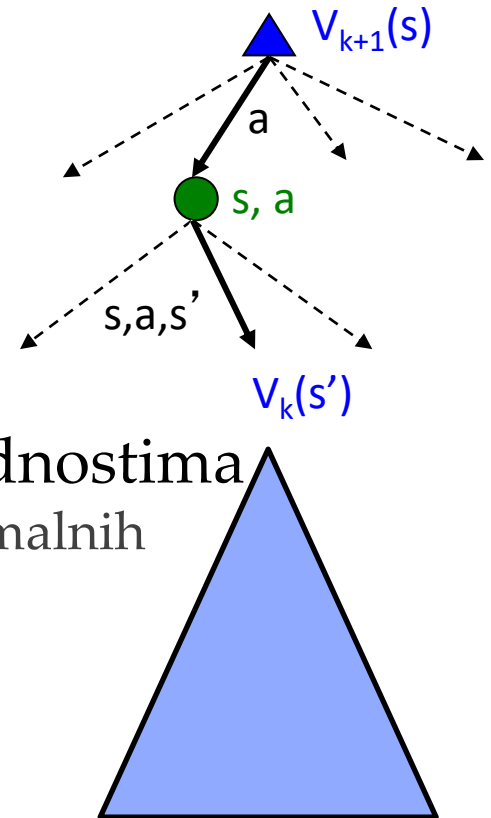


Iteriranje Vrednosti




- Krećemo sa $V_0(s) = 0$: broj koraka $k=0$ pa igra traje 0 poteza pa imamo 0 vrednosti
- Za izračunate vrednosti $V_k(s)$ računamo jedan nivo dole expectimax algoritma:

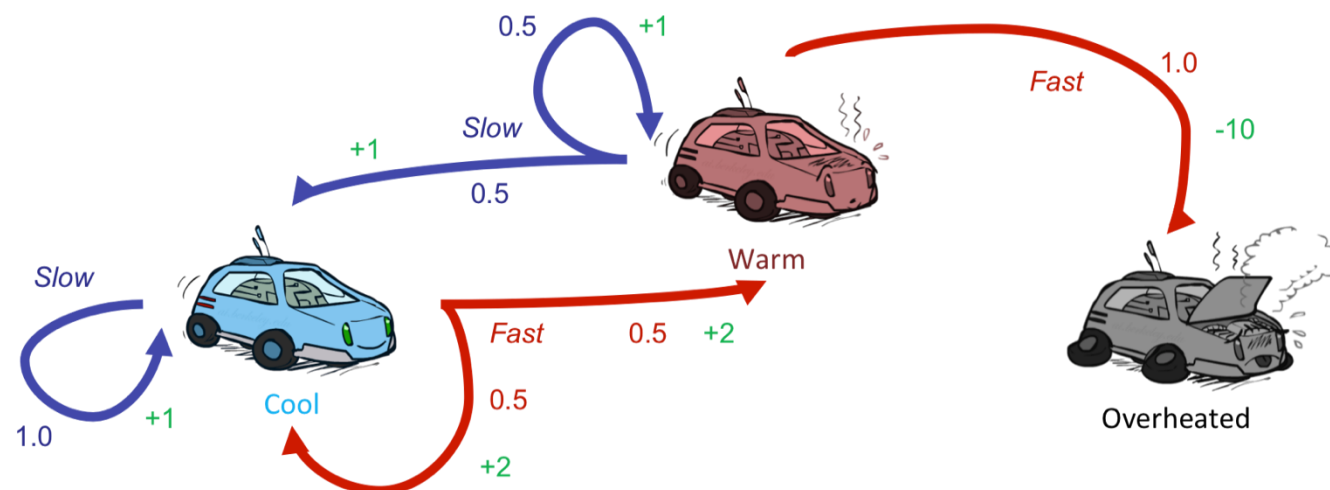
$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V_k(s')]$$

- Ponavljamo do konvergencije
- Teorema (bez dokaza): algoritam konvergira ka optimalnim vrednostima
 - Generalna ideja: aproksimacije vrednosti se polako „popravljaju“ do optimalnih
 - Politika može da konvergira mnogo pre vrednosti (o tome kasnije)



Primer: Iteriranje Vrednosti

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



Pretpostavimo da nemamo zanemarnivanje.

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

$$V_2(\text{cool}) = \max(Q(\text{cool}, \text{slow}), Q(\text{cool}, \text{fast}))$$

$$Q(\text{cool}, \text{slow}) = 1 * (1 + 1 * 0) = 2$$

$$Q(\text{cool}, \text{fast}) = 0.5(2 + 1 * 0) + 0.5(2 + 1 * 0) = 2.5$$

$$V_2(\text{cool}) = \max(Q(\text{cool}, \text{slow}), Q(\text{cool}, \text{fast})) = \max(2, 2.5) = 2.5$$