# VEŠTAČKA INTELIGENCIJA, 2020/2021

# INTELIGENTNI AGENTI


## Sadržaj

- Definicija agenta
- PEAS (Performance measure, Environment, Actuators, Sensors)
- Arhitekture agenata
- Tipovi okruženja

Prof. dr Leonid Stoimenov

# Šta je agent? (1)

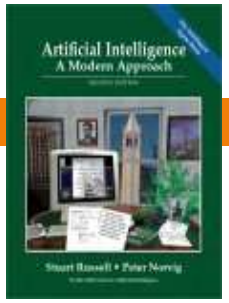

*"…agenti su softverski entiteti koji izvršavaju skup operacija u ime korisnika ili **drugog programa …**"* [IBM]

# Šta je agent? (2)

- **Inteligentni agent** (ili jednostavno agent) na Internetu, je program koji **uzima informacije** ili izvršava neke druge servise **bez vašeg prisustva odnosno intervencije,** na osnovu prethodno definisanog rasporeda.

- *„Agent je bilo šta što može da **percipira** svoje **okruženje** preko senzora i **deluje** na **okruženje** preko efektora."*

# Definicija agenta (4)




- *„Agent je bilo šta što može da **percipira** svoje **okruženje** preko senzora i **deluje** na **okruženje** preko efektora."*

Russell & Norvig

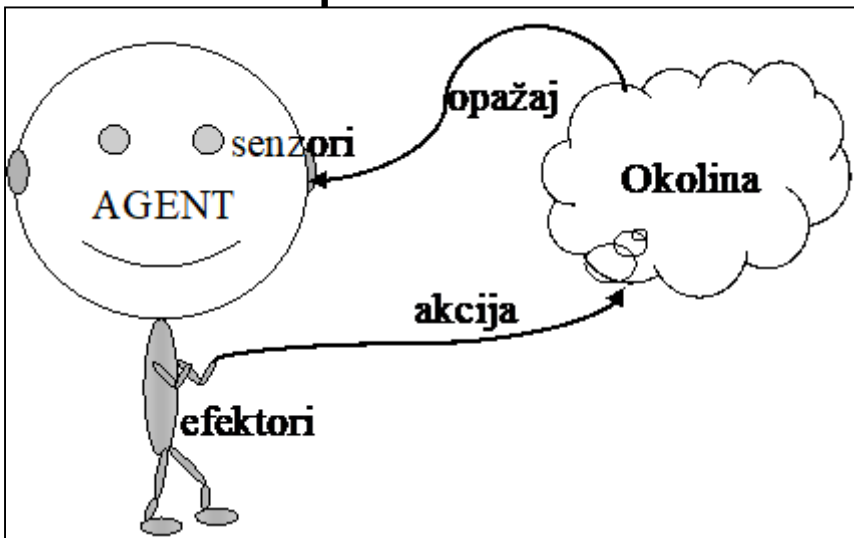
Agent je entitet koji saznaje o svom okruženju i izvršava razne akcije.

Agent je bilo koji sistem (sw, hw ili kombinacija), koji je sposoban da prima informacije iz okruženja

preko senzora i da

deluje na okruženje

preko efektora.



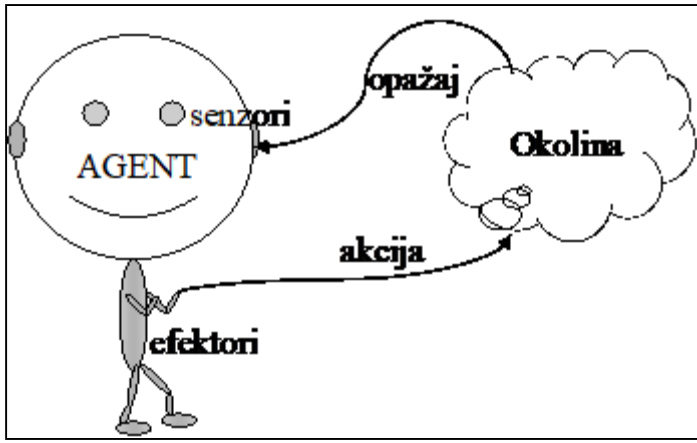
# Funkcija koja opisuje agente

Može se opisati pomoću funkcije (*funkcija agenta*) koja preslikava opažaje u akcije

$f$: P* → A

- Program agenta se izvršava na fizičkoj arhitekturi da bi realizovao $f$

Agent = Arhitektura + Program

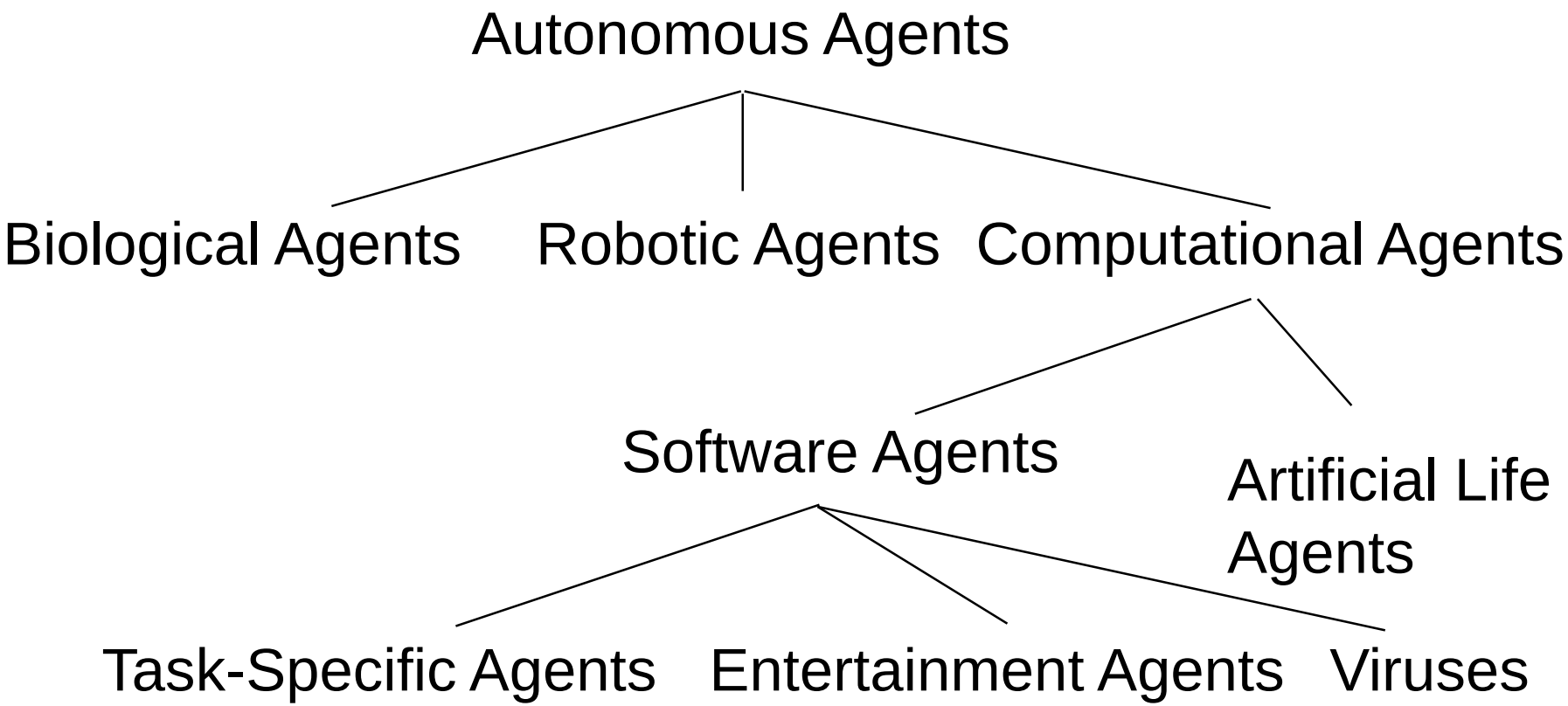


# Agenti

- **Čovek** kao agent:
  - oči, uši, i ostali organi kao senzori;
  - ruke, noge, usta i ostali delovi tela kao aktuatori
- **Robot**-agent:
  - kamere i infrared senzori za dodir;
  - različiti motori kao aktuatori
- **Sofverski agent**
  - Ulazni parametri
  - Rezultati

# Klasifikacija agenata

## (Franklin and Graesser)



# Šta je *inteligentni* agent?

**Inteligentni agent** *je računarski sistem koji može da izvrši* ***autonomno fleksibilnu*** *akciju u nekom okruženju za dostizanje definisanog cilja*

Pod *fleksibilnom* akcijom podrazumevamo:

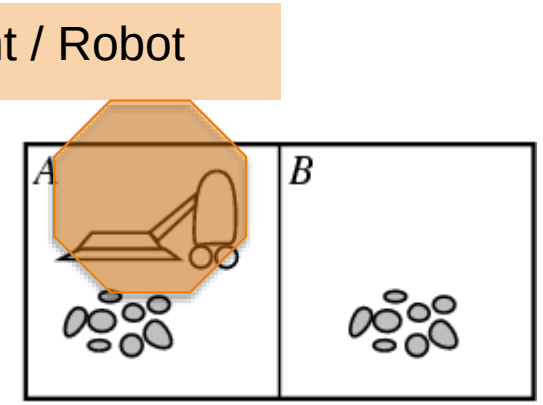
- *Reaktivnost* – postoji interakcija sa okruženjem i sposoban je da odgovori na promene koje se dešavaju u njemu
- *Pro-aktivnost* – sposoban je za ponašanje definisano ciljem,
- *Socijalni aspekt* – sposoban je da interaguje sa drugim agentima (i ljudima)

# Evaluacija agenata

- Agent treba da “radi prave stvari", u zavisnosti od onoga što prima preko percepcije i akcija koje može da izvršava.
- Prava akcija je ona koja obezbeđuje da agent bude najuspešniji
- **Racionalnost**:
  - da li se agent ponaša tako da **maksimizira performanse,** na osnovu zadatog znanja i opažanja iz okruženja?
- Autonomnost:
  - da li su agentove akcije odredjene iskustvom

# Mera performansi

- **Mera performansi:** Objektivni kriterijum kojim se meri uspešnost agenta
- Određena je ciljevima koji su definisani za agenta
  Primer: mera performasi za *Vacuum-cleaner* agenta može biti očišćena količina smeća, potrošeno vreme za čišćenje, količina energije, količina generisane buke, ...



# Evaluacija agenata

- Kako meriti performanse agenta?
- Merenje performansi (zavisi od okruženja)
  - **Eksterno merenje performansi** (kako agent deluje na okolinu)
  - **Samo-evaluacija** – agent je sposoban da sam vrši procenu svojih akcija
  - **kontinualno, periodično ili jedna evaluacija**

- Specifikacija mera performansi:
  - mere performansi su eksterne
  - okruženje obezbeđuje povratnu informaciju agentu u obliku funkcija za preslikavanje iz mogućih stanja u realne brojeve
  - povratne informacije se mogu obezbediti nakon svake akcije, periodično ili na kraju delovanja agenta

# Idealni racionalni agent

Za svaku sekvencu opažanja, idealni racionalni agent izvršava akciju koja maksimizira performanse, na osnovu ulaza i ugrađenog znanja koje agent poseduje.

Problem: perfektno racionalno ponašanje je limitirano hardverskim mogućnostima

# Racionalni agent

Racionalni agent je generalizacija programa:

- Softverski sistem => Agent
- Ulaz => Opažanja
- Izlaz => Akcije
- Specifikacije => Mera performansi

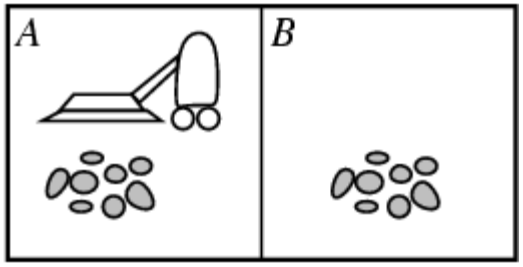
Problem: perfektno racionalno ponašanje je limitirano hardverskim mogućnostima

# Osnovni koncepti

**P**ercepts,
**A**ctions,
**G**oals,
**E**nvironment

- **O**pažanja (ulaz) / **S**enzori (ulazni uredjaji)
- **A**kcije (izlaz) / **E**fektori (izlazni uredjaji)
- **C**ilj / **M**era performansi
- **O**kruženje

# Primer: Vacuum-cleaner world




- Opažanja: lokacija i sadržaj, napr., [A, Dirty]
- Akcije: *Left*, *Right*, *Suck*, *NoOp*

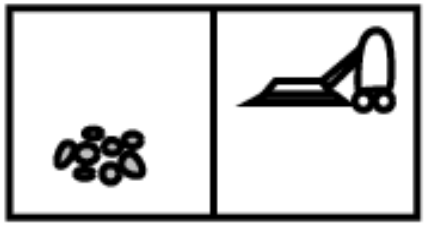
**iRobot Roomba® 400**
Vacuum Cleaning Robot


**iRobot Corporation**

**Founder Rodney Brooks (MIT)**

- Powerful suction and rotating brushes
- Automatically navigates for best cleaning coverage
- Cleans under and around furniture, into corners and along wall edges
- Self-adjusts from carpets to hard floors and back again
- Automatically avoids stairs, drop-offs and off-limit areas
- Simple to use— just press the Clean button and Roomba does the rest

# Osnovni koncepti: Primer Vacuum-cleaner world



- **Okruženje**: zadati prostor (dve prostorije, međusobno povezane), zidovi, prepreke, nečistoća
- **Opažanja**: odnose se na lokaciju i sadržaj, tj detekcija nečistoća, zid/dodir
  - **Senzori**: odgovaraju opažanjima – senzor za nečistoću, senzor za dodir
- **Akcije**: čisti/stani, gore, dole, levo, desno
  - **Efektori**: za izvršenje akcija – motor za pokretanje, mehanizam za zaustavljanje, mehanizam za okretanje,...
- **Cilj/Mera performansi**: Ciljevi zavise od okoline: čiste obe prostorije!!

# Agent za automatski taksi:

- Okruženje: ??
- Senzori: ??
- / Opažanja (ulaz): ??
- Akcije: ??
- / Efektori: ??
- Mere performansi: ??
- / Ciljevi: ??

**Ralph**: Vision based vehicle steering
analiza slika autoputa ispred vozila u pokretu
određuje zakrivljenost puta (krivine)
određuje bočni offset pomeranja vozila u odnosu na centar trake puta
komande nad upravljačkim sistemom vozila
Komercijalno dostupan 1996: Assistwrae Technology Inc., cena $1975,


# Agent za automatski taksi

- **Okruženje**: putevi, drugi saobraćaj, pešaci, korisnici
- **Senzori i opažanja**: kamere, sonar, brzinomer, GPS, merač pređenog puta, senzori motora, tastatura
- **Akcije i Efektori**: upravljački deo, akcelerator, kočnica, signalizacija, i sl
- **Mere performansi** / **Cilj**: siguran, brz, legalan, udoban put, maksimizacija profita

# Medicinski dijagnostički sistem

- **Okruženje**: pacijenti, bolnica, osoblje
- **Senzori i opažanja**: tastatura (unos simptoma, nalaza, odgovora pacijenata)
- **Akcije i Efektori**: monitor (pitanja, testovi, dijagnoze, tretman, terapije) i sl
- **Mere performansi** / **Cilj**: zdrav pacijent, minimizacija cene, tužbe, ...

# Robot na postavljanju delova

- **Okruženje:** pokretna traka sa delovima, delovi, spremišta delova
- **Senzori i opažanja:** kamere, senzori za ugao
- **Akcije i Efektori:** ruka robota
- **Mere performansi / Cilj:** procenat delova na svom mestu

# Interaktivni tutor za učenje jezika

- **Okruženje**: studenti
- **Senzori i opažanja**: tastatura, mikrofon
- **Akcije i Efektori**: Ekran (primeri, sugestije, korekcije)
- **Mere performansi / Cilj**: maksimizacija rezultata studenta na testu, zaposlenje u Nemačkoj

# Primeri tipova agenata i prikaz osnovnih koncepata

| Agent Type | Percepts | Actions | Goals | Environment |
|---|---|---|---|---|
| Medical diagnosis system | Symptoms, findings, patient's answers | Questions, tests, treatments | Healthy patient, minimize costs | Patient, hospital |
| Satellite image analysis system | Pixels of varying intensity, color | Print a categorization of scene | Correct categorization | Images from orbiting satellite |
| Part-picking robot | Pixels of varying intensity | Pick up parts and sort into bins | Place parts in correct bins | Conveyor belt with parts |
| Refinery controller | Temperature, pressure readings | Open, close valves; adjust temperature | Maximize purity, yield, safety | Refinery |
| Interactive English tutor | Typed words | Print exercises, suggestions, corrections | Maximize student's score on test | Set of students |

**Figure 2.3** Examples of agent types and their PAGE descriptions.

# Struktura inteligentnih agenata

- Agent = arhitektura + program
- Agent je specificiran pomoću **funkcije** koja preslikava niz opažanja u akciju.

$$f: P^* \rightarrow A$$

# Osnovni algoritam:

- Ažuriraj memoriju (nova opažanja)
- Izaberi najbolju akciju
- Ažuriraj memoru (nova akcija)

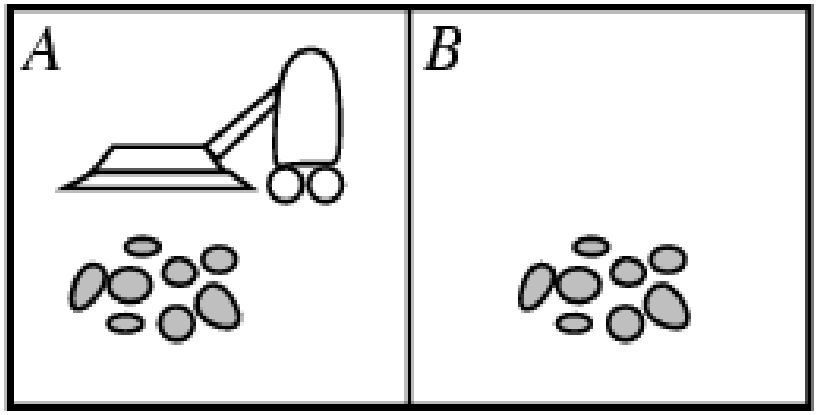
```
function SKELETON-ACTION (percept) returns action
    static: memory

    memory ←UPDATE-MEMORY(memory, percept)
    action ←CHOOSE-BEST-ACTION(memory)
    memory ←UPDATE-MEMORY(memory, action)
    return action
```

# Specificiranje funkcije preslikavanja

Jednostavno rešenje:

- Tabela koja sadrži sva moguća preslikavanja (može biti prevelika)
- Program agenta (predstavlja preslikavanje na koncizan način)
  - Uzima jedno opažanje
  - Pamti interno stanje

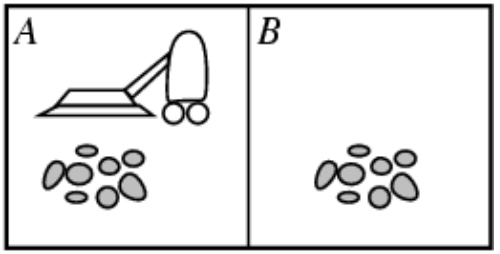
# Primer: Vacuum-cleaner world




- **Opažaji**: lokacija i sadržaj, napr, [A,Dirty]
- **Akcije**: *Left*, *Right*, *Suck*, *NoOp*

# Funkcija agenta: Preslikavanje opažaja u akcije

| Percept sequence | Action |
|---|---|
| $[A, Clean]$ | $Right$ |
| $[A, Dirty]$ | $Suck$ |
| $[B, Clean]$ | $Left$ |
| $[B, Dirty]$ | $Suck$ |
| $[A, Clean], [A, Clean]$ | $Right$ |
| $[A, Clean], [A, Dirty]$ | $Suck$ |
| ⋮ | ⋮ |

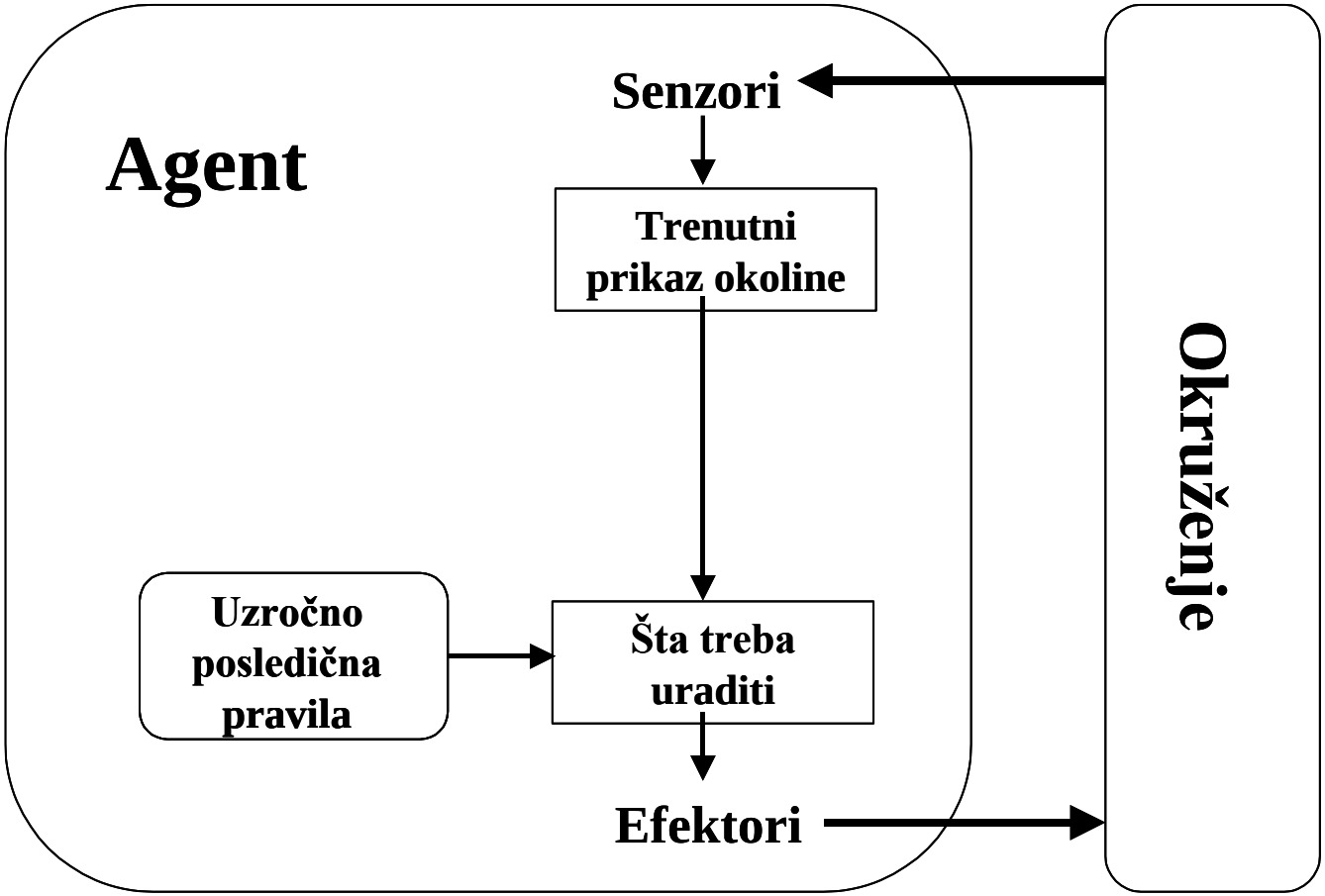



- Problemi:
  - za neke agente ova tabela je prevelika!!
  - Vreme potrebno za genrisanje tabele
  - Nema autonomije
  - Čak i sa elementima učenja, zahteva se dodatno vreme za sve podatke u tabeli

# Arhitektura inteligentnih agenata

0. (Table-driven agenti)
1. Jednostavni agent baziran na refleksima
2. Refleks agenti koji pamte stanja
3. Agenti bazirani na cilju
4. Agenti bazirani na korisnosti

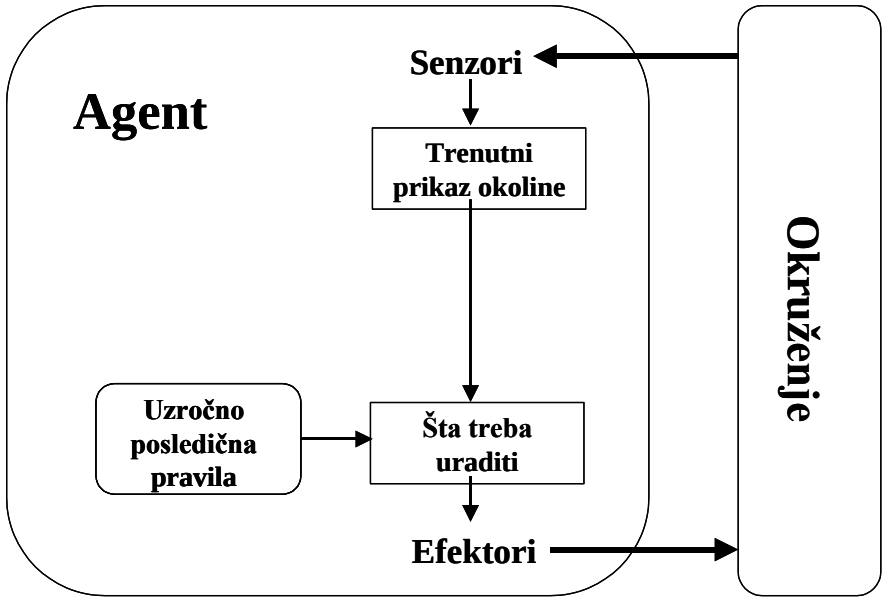
# Jednostavni refleks agent



# Jednostavni refleks agent

**Algoritam**

- Interpretacija ulaza
- Izbor pravila (poklapanje se situacijom)
- Nova akcija na osnovu pravila



# Pseudo-kod agenta baziranog na refleksima

**function** SIMPLE-REFLEX-AGENT(*percept*) **returns** action

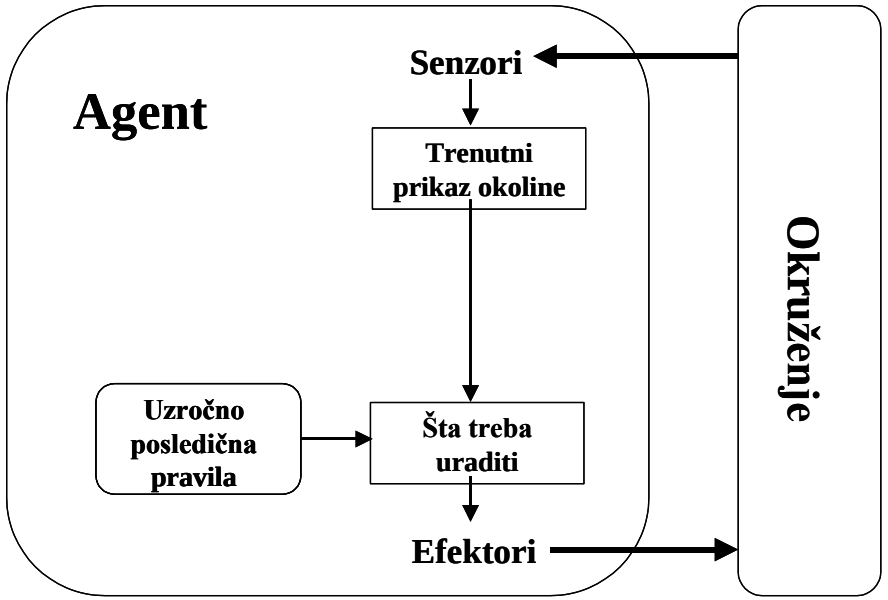
  **static:** *rules*, a set of condition-action rules

  *state* ← INTERPRET-INPUT(*percept*)

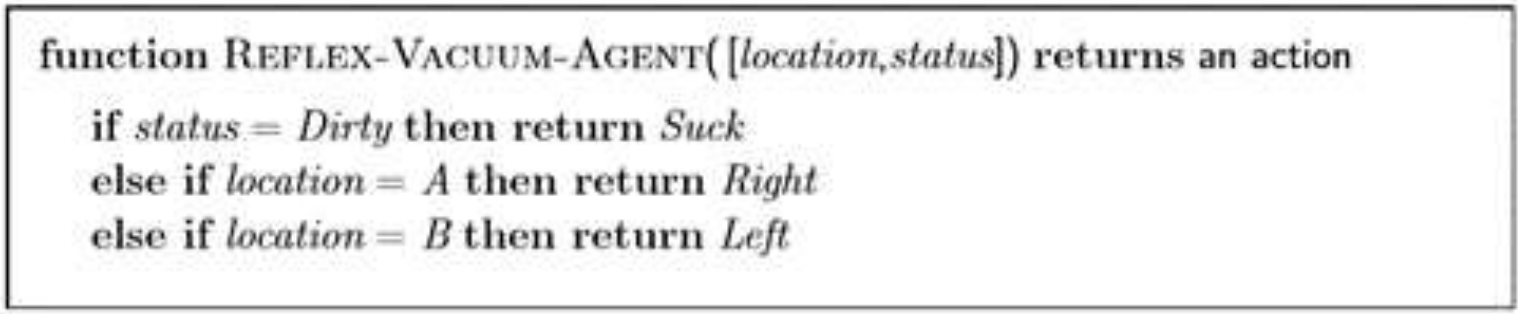
  *rule* ← RULE-MATCH(*state*,*rules*)

  *action* ← *RULE*-ACTION[*rule*]

  **return** *action*

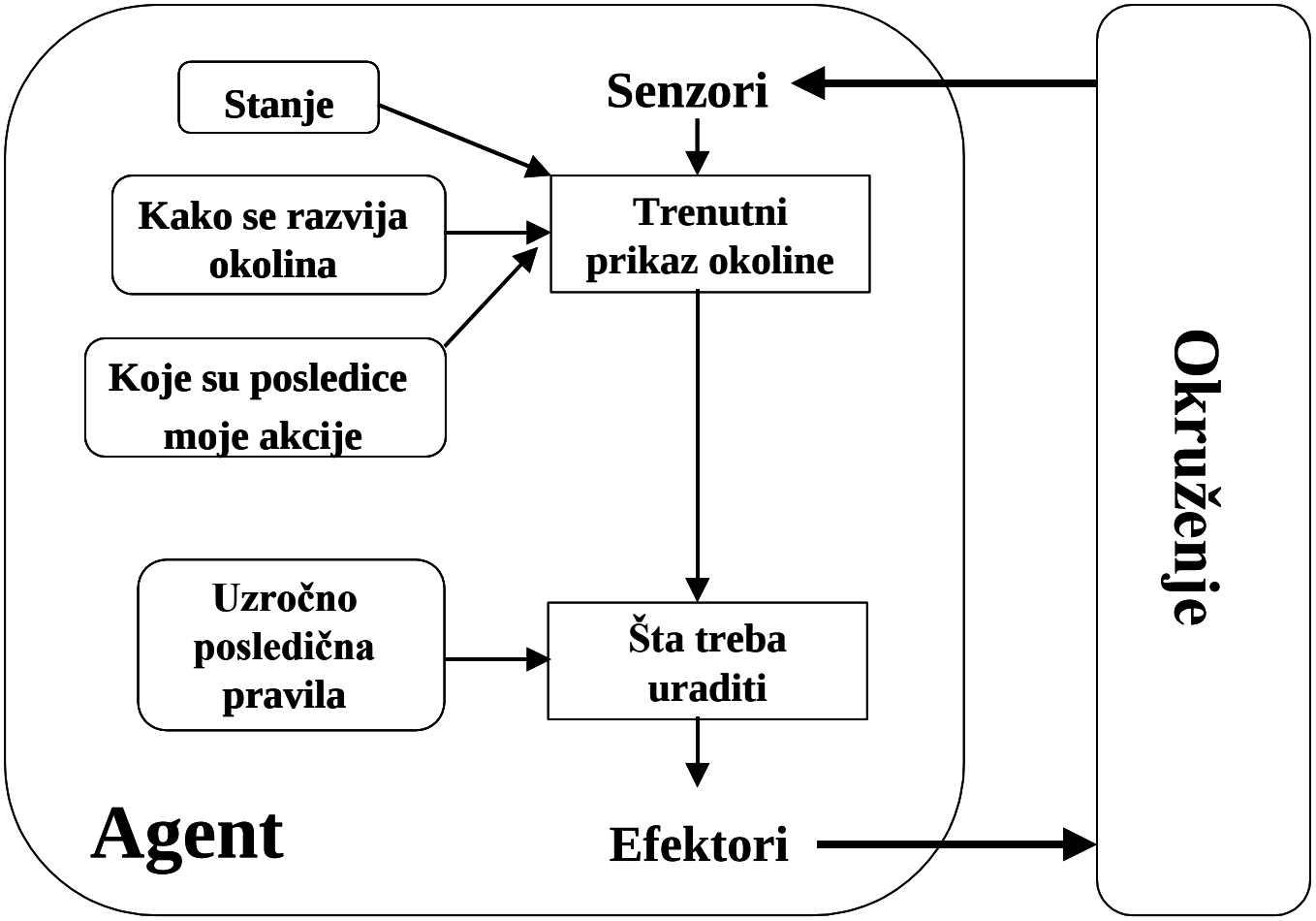


# Primer implementacije u Lisp-u




```
(setq joe (make-agent :name 'joe :body (make-agent-body)
                      :program (make-reflex-vacuum-agent-program)))

(defun make-reflex-vacuum-agent-program ()
    #'(lambda (percept)
        (let ((location (first percept)) (status (second percept)))
            (cond ((eq status 'dirty) 'Suck)
                  ((eq location 'A) 'Right)
                  ((eq location 'B) 'Left)))))
```

# Refleks agent koji pamti stanje

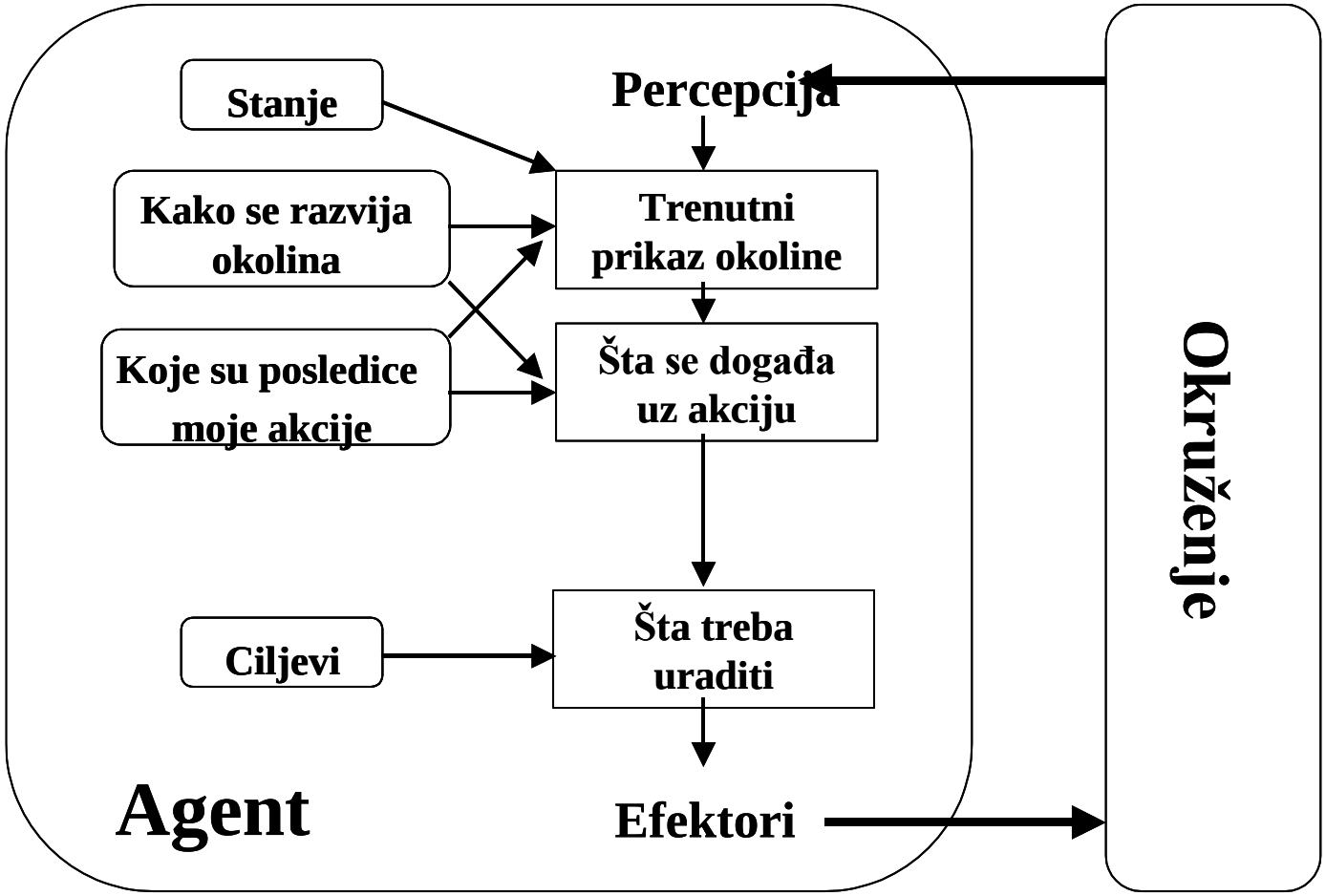


# Refleks agent koji pamti stanje Pseudo-kod

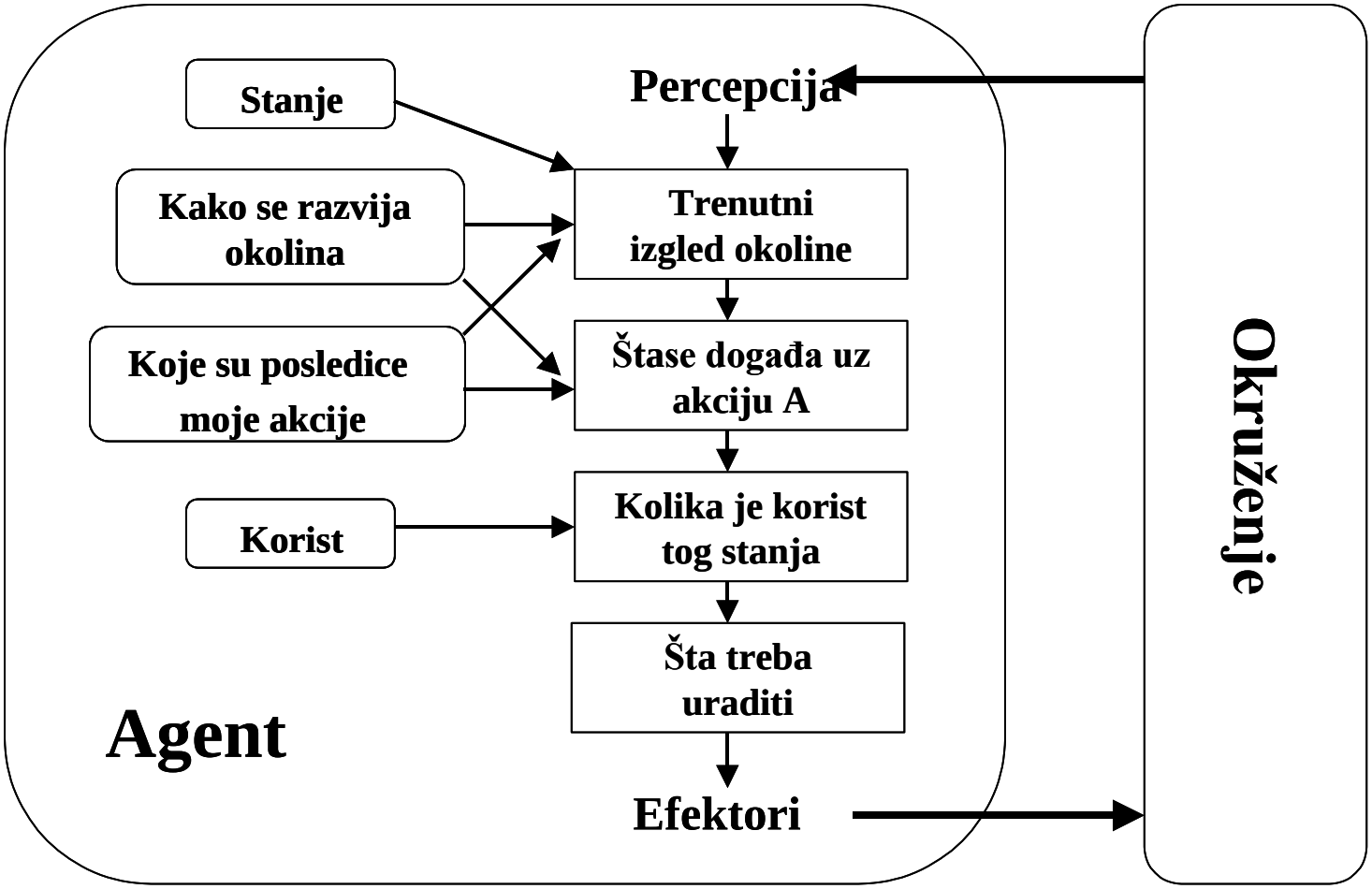
- Ažuriraj stanje na osnovu ulaza
- Poklapanje situacije sa pravilima
- Izbor akcije na osnovu pravila
- Ažuriraj stanje na osnovu akcije

**function** REFLEX-AGENT-WITH-STATE(*percept*) **returns** an action
  **static:** ***state***, a description of the current world state
    ***rules***, a set of condition-action rules
  ***state*** ← UPDATE-STATE(***state***, ***percept***)
  ***rule*** ← RULE-MATCH(***state***, ***rules***)
  ***action*** ← *RULE-ACTION*(***rule***)
  ***state*** ← *UPDATE-STATE*(***state***, ***action***)
  **return** ***action***

# Agent baziran na cilju



# Agent baziran na korisnosti



# Osobine okruženja: Tipovi okruženja (1)

- **Dostupnost:** Potpuno ili delimično dostupna:
  - da li agent ima podatke o celokupnom okruženju ili samo o nekom delu okruženja.
- **Determinisanost:** Deterministička ili stohastička:
  - determisnitičko, ako sledeće stanje okruženja kompletno zavisi od tekućeg stanja i izabrane akcije od strane agenta,
  - inače stohastičko.

# Tipovi okruženja (2)

- **Diskretnost:** Diskretna ili kontinualna:
  - da li su **stanja** (i **akcije**) diskretna ili kontinualna.
  - Šah je diskretno, vožnja taksija je kontinualno okruženje.
  - Kod diskretnih okruženja postoji konačan broj jasno definisanih koncepata i akcija.

# Tipovi okruženja (3)

- **Epizodnost**: Bazirana na epizodama ili sekvencijalna:
  - u okruženju baziranom na epizodama iskustvo agenta je podeljeno na epizode.
  - Kvalitet akcije zavisi od samo od epizode; naredne epizode ne zavise od prethodnih.
  - Kod sekvencijalnih to nije slučaj, tj. “epizode” su međusobno zavisne.

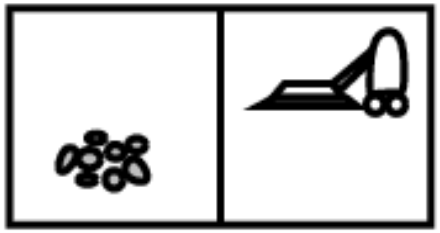
# Tipovi okruženja (4)

- Statička ili dinamička:
  - okruženje se ne menja (ili menja u drugom slučaju) dok agent "razmišlja".

- **Jedan ili više agenata u okruženju:**
  - konkurentna (suparnička) ili
  - komparativna (uporediva) okruženja

  (u originalu: *competitive* ili *comparative*).
- Definišu da li agenti međusobno sarađuju ili se nadmeću; u oba slučaja postoji komunikacija između agenata.

# Okruženje: Primer



Agent: *vacuum-cleaner*

- Tip(ovi) ovog okruženja: ??
  - Potpuno ili delimično dostupna **??**
  - Deterministička ili stohastička **??**
  - Diskretna ili kontinualna **??**
  - Bazirana na epizodama ili sekvencijalna **??**
  - Statička ili dinamička **??**
  - Konkurentna ili komparativna **??**

# Primeri okruženja i njihove osobine

| Environment | Accessible | Deterministic | Episodic | Static | Discrete |
|---|---|---|---|---|---|
| Chess with a clock | Yes | Yes | No | Semi | Yes |
| Chess without a clock | Yes | Yes | No | Yes | Yes |
| Poker | No | No | No | Yes | Yes |
| Backgammon | Yes | No | No | Yes | Yes |
| Taxi driving | No | No | No | No | No |
| Medical diagnosis system | No | No | No | No | No |
| Image-analysis system | Yes | Yes | Yes | Semi | No |
| Part-picking robot | No | No | Yes | No | No |
| Refinery controller | No | No | No | No | No |
| Interactive English tutor | No | No | No | No | Yes |

# Programiranje okruženja

- Simulatori okruženja:
  - Za jednog ili više agenata
  - Daju agentu/agentima opažaje
  - Izvršava program agenta i dobija akciju agenta
  - Ažurira okruženje
  - Može se dodati kod koji vrši merenje performansi

# Simulator okruženja

```
procedure RUN-ENVIRONMENT(state, UPDATE-FN, agents, termination)
  inputs: state, the initial state of the environment
          UPDATE-FN, function to modify the environment
          agents, a set of agents
          termination, a predicate to test when we are done

  repeat
    for each agent in agents do
      PERCEPT[agent] ← GET-PERCEPT(agent, state)
    end
    for each agent in agents do
      ACTION[agent] ← PROGRAM[agent](PERCEPT[agent])
    end
    state ← UPDATE-FN(actions, agents, state)
  until termination(state)
```

# Simulator okruženja sa merenjem performansi

```
function RUN-EVAL-ENVIRONMENT(state, UPDATE-FN, agents,
                              termination, PERFORMANCE-FN) returns scores
  local variables: scores, a vector the same size as agents, all 0

  repeat
    for each agent in agents do
      PERCEPT[agent] ← GET-PERCEPT(agent, state)
    end
    for each agent in agents do
      ACTION[agent] ← PROGRAM[agent](PERCEPT[agent])
    end
    state ← UPDATE-FN(actions, agents, state)
    scores ← PERFORMANCE-FN(scores, agents, state)
  until termination(state)
  return scores                                          /* change */
```

# PITANJA?

Dileme?

Komentari?