

# Vještačka inteligencija

## Predavanje 12: Inteligentni agenti

*"Vjerujte onima koji traže istinu, sumnjajte u one koji je pronalaze ..."*  
- André Gide

Odgovorna nastavnica: Vanr. prof. dr Amila Akagić

Univerzitet u Sarajevu



# Uvodne informacije

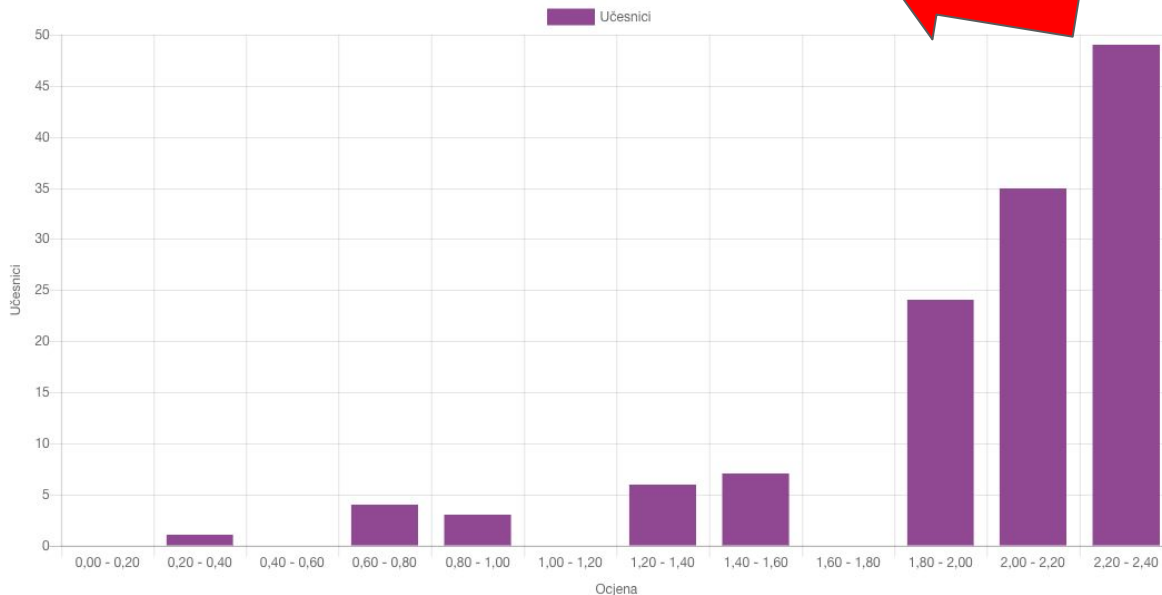
- This work is licensed under a Creative Commons 'Attribution-NonCommercial-ShareAlike 4.0 International' license. EN: <https://creativecommons.org/licenses/by-nc-sa/4.0/>



- Ovaj rad je licenciran pod međunarodnom licencom 'Imenovanje-Nekomercijalno-Dijeli pod istim uvjetima 4.0' od strane Creative Commons. HR: <https://creativecommons.org/licenses/by-nc-sa/4.0/deed.hr>

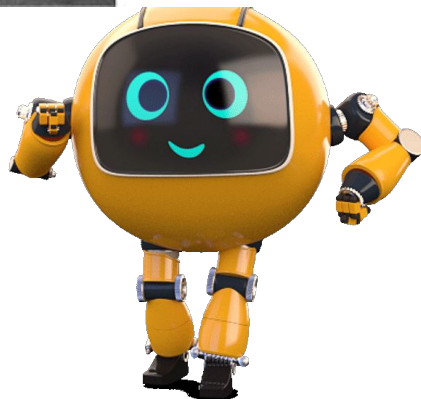
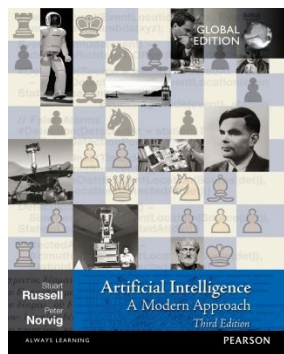
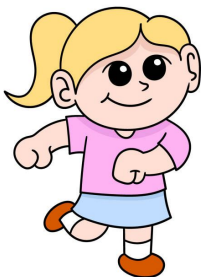
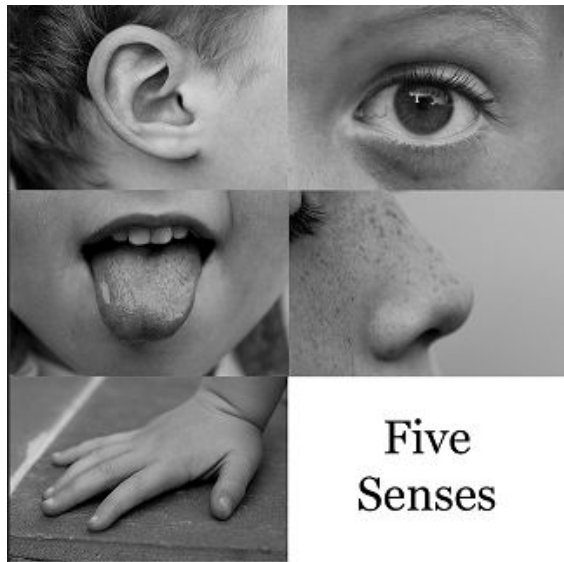
# Najave

- ❑ Kviz 9 održat će se 03.06.2021. u 14:02 preko c2.
- ❑ Sa predavanjima nastavljamo od 14:10+.
- ❑ Ovo je XIV sedmica nastave.
- ❑ U XV sedmici je II provjera znanja (10.06.2021. u 14:00).
- ❑ Kviz 10 održat će se 14.06.2021. (ponedjeljak) od 8:50.



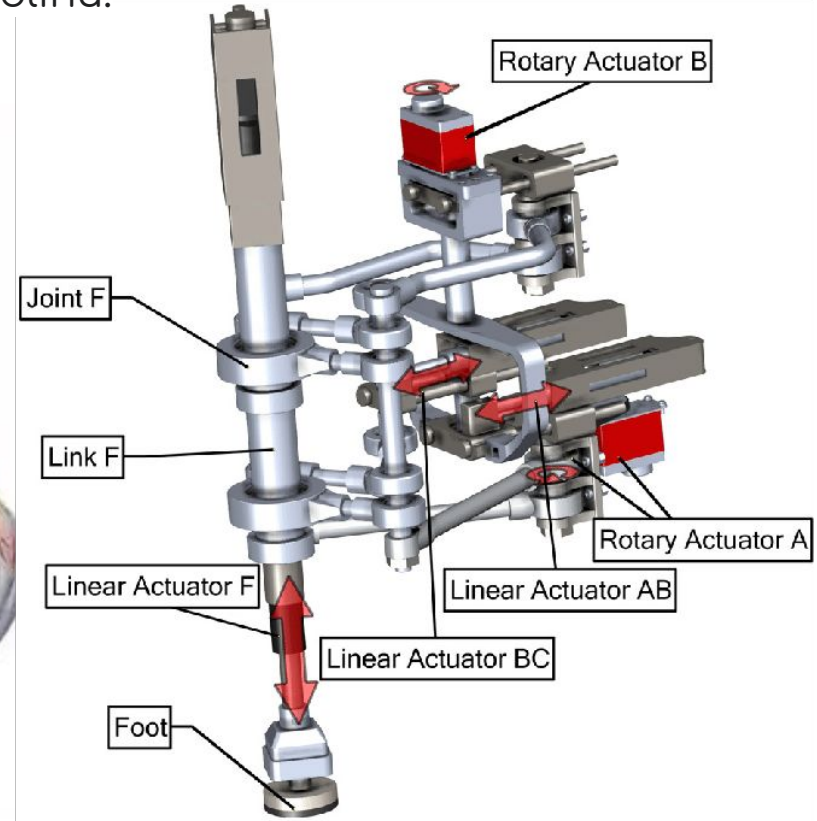
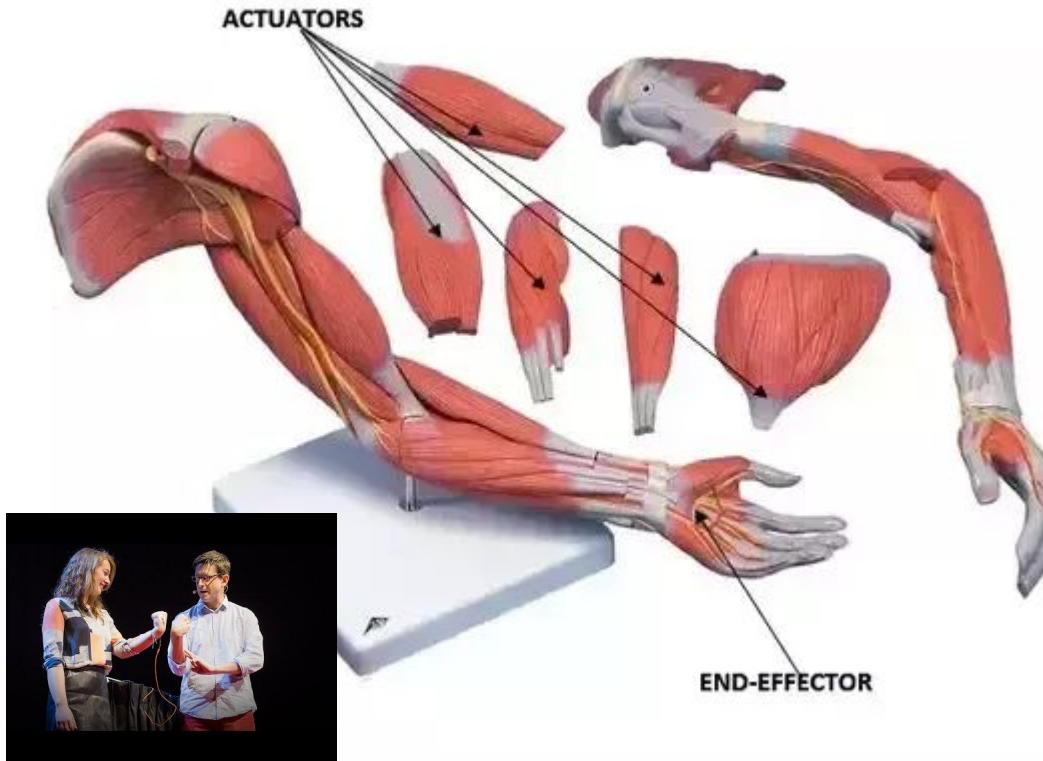
# Agent

- ❑ **Agent**: bilo šta za što možemo reći da može percipirati svoju okolinu putem senzora i djelovati na tu okolinu putem aktuatora.
- ❑ **Objekat zapažanja** (*percept*): objekat koji agent zapaža u svojoj okolini i uzima u obzir kao ulazni podatak.
- ❑ Sekvenca objekata zapažanja: historija svega što agent vidi.
  - ❑ Na primjer: sve što ste vidjeli u jednom danu.



# Aktuatori

Naprava/uređaj kojim se na pobudu upravljačkog (kontrolnog) signala pokretni dijelovi sistema dovode u željeni položaj, ostvaruje se njihovo gibanje ili razvija sila ili moment sile (zakretni moment) kojim ti dijelovi djeluju na okolinu.



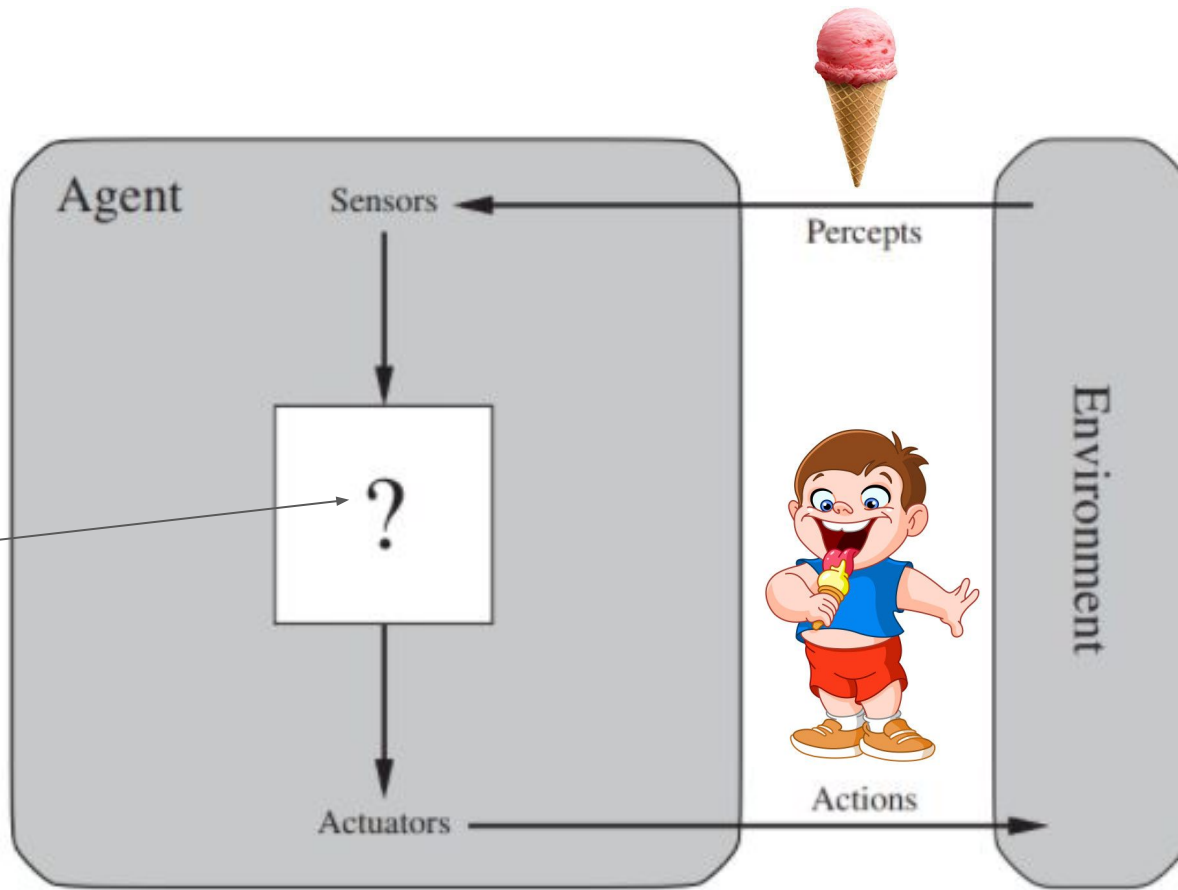
# Agent

- ❑ Inteligentni agenti trebaju donositi odluke na osnovu onoga što dobijaju iz svoje okoline (objekata zapažanja).
- ❑ Definisanjem svih mogućih odluka, izbora i akcija definiše kako jedan agent funkcioniše.
- ❑ Definisanjem funkcije agenta moguće je mapirati/preslikati percepciju u akcije.
- ❑ Program agenta: interna implementacija funkcije agenta.

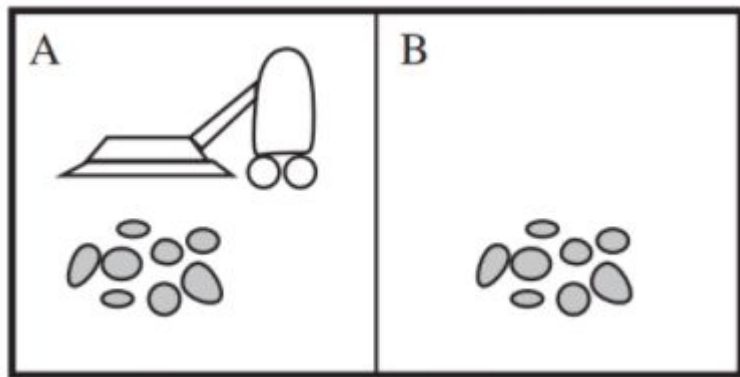


# Agent

Program  
agenta:  
implementacij  
a funkcije  
agenta



# Primjer: svijet robot-usisavača



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
⋮	⋮
[A, Clean], [A, Clean], [A, Clean]	Right
[A, Clean], [A, Clean], [A, Dirty]	Suck
⋮	⋮



# Koncept racionalnosti

- ❑ **Racionalni agent:** agent koji radi “ispravne (dobre) stvari”.
  - ❑ Tabela preslikavanja sadrži sekvencu ispravnih akcija.
  - ❑ Šta znači “ispravno” ili “pogrešno”? Koje su konsekvence?
- ❑ Postoji sekvenca događaja koje treba pratiti:
  - ❑ Agent percipira (opaža) svoju okolinu.
  - ❑ Percepcija okoline će dovesti do nekih akcija.
  - ❑ Akcije dovode do promjene okoline.
- ❑ Ako je sekvenca promjena agenta koje mijenjaju okolinu poželjna, onda je agent uradio ispravne stvari. Npr. očistio svoju okolinu.
- ❑ Performanse agenta se mogu mjeriti sa nekom metrikom: **mjera performansi**
  - ❑ Svaki put kada agent uradi nešto dobro dobija nagradu: +1
  - ❑ Akumulacija bodova.
  - ❑ Koje akcije nagrađivati (primjer robot-usisavač)?
    - ❑ Pod je čist? Ekifasno očišćen pod? Količina koju očisti?

# Racionalnost

- ❑ Racionalnost zavisi od:
  - ❑ Mjere performansi
  - ❑ Agentovog prethodnog znanja
  - ❑ Akcija koje agent može da poduzme
  - ❑ Agentove trenutne percepcije (ili liste objekata zapažanja)
- ❑ Definicija racionalnog agenta:

Za svaku moguću sekvencu opažanja, racionalni agent trebao bi odabrati onu akciju (radnju) za koju se očekuje da bi mogla maksimizirati mjeru performansi, s obzirom na dokaze pružene sekvencom opažanja i bez obzira na ugrađeno (prethodno) znanje koje agent ima.

# Primjer

- ❑ Robot usisivač:
  - ❑ 1 bod za čišćenje prostora (do max 1000 akcija/radnji)
  - ❑ Okolina je poznata, očišćeni prostor ostaje čist, usisavanje čisti prostor.
  - ❑ Akcije/radnje: lijevo, desno, usisaj.
  - ❑ Agent može da percipira čistoću i lokaciju.
- ❑ Šta ako se uvede mjera kazne? Može li robot postati iracionalan?
- ❑ Šta se dešava kada agent očisti sve površine?
  - ❑ Potrebno napraviti mogućnost detekcije ovog stanja i ugasiti usisivač ili ga staviti u stanje mirovanja.
- ❑ Agenti ne mogu znati sve, ne mogu znati ishod svake akcije.
- ❑ **Racionalne akcije** nisu uvijek savršene ili korektne i ne moraju da imaju pozitivan ishod, iako na prvu ruku mogu da izgledaju korektne.



*"Put do pakla je popločan dobrim namjerama."*

# Racionalnost

- ❑ Racionalnost maksimizira očekivane performanse.
  - ❑ Perfektnost nije racionalnost. Perfektnost vrži maksimiziranje stvarnih performansi.
  - ❑ Da li je moguće projektovati perfektnog agenta?
  - ❑ S druge strane, moguće je minimizirati neefikasne akcije. (Spriječiti agenta da pravi neadekvatne akcije).
  - ❑ Agent može prikupljati informacije koje će omogućiti zapaženje objekata za bolje odlučivanje u narednim koracima.
- 
- ❑ Autonomija agenta: agent koji zavisi od znanja koje je ugradio njegov kreator nema autonomnost.
  - ❑ Racionalni agenti trebaju imati autonomiju: u tom slučaju ne moraju se eksplicitno programirati za svaki novi slučaj.

# Okruženje agenta

- ❑ Okruženje zadatka (*task environment*): okolina koja se definiše za rješavanje nekog problema.
- ❑ PEAS (Performance, Environment, Actuators, Sensors): četiri stvari koje je potrebno opisati agentu.
- ❑ Okolina zadatka sačinjena je od podataka o PEAS.

Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard



# Okruženje agenta

- ❑ Okruženje može biti stvarno ili virtuelno.
- ❑ Koliko je okruženje kompleksno u odnosu na ponašanje agenta, sekvencu opažanja i mjeru performansi?
- ❑ Potrebno je što bolje razumijeti okruženje kako bi ponašanje agenta bilo što bolje.
- ❑ Softverski agenti (softbot): na primjer agent koji pretražuje Twitter za interesantnim novostima. Okruženje je izuzetno dinamično.



# Primjeri agenata

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, reduced costs	Patient, hospital, staff	Display of questions, tests, diagnoses, treatments, referrals	Keyboard entry of symptoms, findings, patient's answers
Satellite image analysis system	Correct image categorization	Downlink from orbiting satellite	Display of scene categorization	Color pixel arrays
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, joint angle sensors
Refinery controller	Purity, yield, safety	Refinery, operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical sensors
Interactive English tutor	Student's score on test	Set of students, testing agency	Display of exercises, suggestions, corrections	Keyboard entry

# Osobine okruženja

- ❑ Raspon zadatka je izuzetno veliki.
- ❑ Mogu se klasificirati na različite načine. Na primjer:
  - ❑ **Potpuno definisana**: ako agent može da dobije sve informacije iz okruženja za donošenje odluka preko svojih senzora kroz čitav period vremena.
  - ❑ **Dijelimično definisana**: važi suprotno. 😄
  - ❑ **Nedefinisana**: agent koji nema senzore.
- ❑ Okruženja se mogu klasificirati u odnosu na:
  - ❑ Broj agenata
  - ❑ Vrste komunikacije (kompetitivnost, kooperativnost)
  - ❑ Dinamičnosti
  - ❑ ...






# Vrste okruženja

- ❑ Vrste okruženja u odnosu na broj agenata:
  - ❑ **Jedan agent:** agent samostalno rješava neki problem. Primjer: rješavanje zagonetke
  - ❑ **Više agenata:** agenti komuniciraju međusobno u istom okruženju. Primjer: igranje šaha.
- ❑ Agenti mogu da rade jedan protiv drugog: **kompetitivno okruženje.**
- ❑ Agenti koji rade zajedno: **kooperativno okruženje.**
- ❑ Okruženje može biti:
  - ❑ **Determinističko:** bazirano samo na predefinisanim stanjima i akcijama agenta.
  - ❑ **Stohastičko:** suprotno od determinističkog, postoji određenja vjerovatnoća djelovanja.
  - ❑ **Neizvjesno:** ne može se potpuno opažati i nije determinističko.



# Vrste okruženja

- ❑ Epizodno i sekvencijalno okruženje (prethodne akcije?):
  - ❑ **Epizodno okruženje** akcije su diskretne - atomske. Ne uzimaju se obzir prethodne akcije, nego samo trenutna opažanja.
    - ❑ Primjer: mašina treba da utvrdi da li je neki dio u kvaru.
  - ❑ **Sekvencijalno okruženje**: uzimaju se u obzir prethodne akcije i na bazi toga se donosi nova odluka.
    - ❑ Primjer: igranje šaha.
- ❑ Statično i dinamičko okruženje:
  - ❑ **Dinamičko okruženje**: ukoliko se okruženje mijenja u vremenu donošenja (razmišljanja o) naredne odluke
    - ❑ Primjer: vožnja autonomnog vozila. 
  - ❑ **Statičko okruženje**: suprotno od dinamičkog.
    - ❑ Primjer: Rješavanje puzzle. 
  - ❑ **Poludinamično okruženje**: okruženje se ne mijenja, ali se mijenja bodovanje performansi. Primjer: igranje šaha sa aspektom vremena. 

# Vrste okruženja

- ❑ Diskretno ili kontinulano (vremensko) okruženje: odnosi se stanje okruženje i na pojam vremena u okruženju, tj. način opažanja objekata i agentovih akcija.

- ❑ **Diskretno okruženje:**

- ❑ Primjer: igra dame, konačan broj koraka pa tako i stanja.

- ❑ **Kontinualno okruženje:**

- ❑ Agent prelazi iz jednog okruženja u drugo vrlo glatko (kontinuirano).
  - ❑ Primjer: brzina vozila se konstantno mijenja, kao i lokacija.



# Vrste okruženja

- ❑ Poznato i nepoznato okruženje: odnosi se na agentova poznavanje okruženja u kojem se nalazi, kao i na stvari fizičke prirode.
  - ❑ **Poznato okruženje:** Ishodi i vjerovatnoća ishoda akcija je poznata/data.
    - ❑ Primjer: imam 30% šanse da budem uspješan u nekom koraku.
    - ❑ Mogu biti djelimično definisao (posmatrana/uočljiva).
  - ❑ **Nepoznato okruženje:** suprotno od poznatog. 😄

# Vrste okruženja

- ❑ Okruženje nije jednostavno definisati.
- ❑ Uspješnost projekta u velikom mjeri zavisi od uspješno definisanog okruženja.

Task Environment	Observable	Agents	Deterministic	Episodic	Static	Discrete
Crossword puzzle	Fully	Single	Deterministic	Sequential	Static	Discrete
Chess with a clock	Fully	Multi	Deterministic	Sequential	Semi	Discrete
Poker	Partially	Multi	Stochastic	Sequential	Static	Discrete
Backgammon	Fully	Multi	Stochastic	Sequential	Static	Discrete
Taxi driving	Partially	Multi	Stochastic	Sequential	Dynamic	Continuous
Medical diagnosis	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Image analysis	Fully	Single	Deterministic	Episodic	Semi	Continuous
Part-picking robot	Partially	Single	Stochastic	Episodic	Dynamic	Continuous
Refinery controller	Partially	Single	Stochastic	Sequential	Dynamic	Continuous
Interactive English tutor	Partially	Multi	Stochastic	Sequential	Dynamic	Discrete

# Struktura agenata

- ❑ Zadatak AI programera je da napiše/dizajnira **program agenta** na način da izvršava **agentsku funkciju** (mapiranje percepcije u akcije/radnje).
- ❑ Program se izvršava na nekom hardverskom uređaju sa senzorima i aktuatorima, koje se naziva **arhitektura**.

$$\text{agent} = \text{arhitektura} + \text{program}$$

# Program agenta

- ❑ Program treba da uzme (jedan) objekat opažanja i da akciju koju zatim izvode aktuatori.
- ❑ Ako agent donosi odluke na osnovu sekvence prethodnih koraka, onda mora imati memoriju.

**function** TABLE-DRIVEN-AGENT(*percept*) **returns** an action

**persistent:** *percepts*, a sequence, initially empty

*table*, a table of actions, indexed by percept sequences, initially fully specified

append *percept* to the end of *percepts*

*action* ← LOOKUP(*percepts*, *table*)

**return** *action*

Nažalost, upotreba agenata koji su bazirani na lookup tabelama je veoma ograničena.

Teško je predvidjeti svaku moguću sekvencu zapažanja.

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
⋮	⋮
[A, Clean], [A, Clean], [A, Clean]	Right
[A, Clean], [A, Clean], [A, Dirty]	Suck
⋮	⋮

# Program agenta

- ❑ Izazov: Kako napisati program agenta koji će pokazati racionalno ponašanje na način da koristimo skup manjih programa umjesto jedne velike tabele?
- ❑ Postoje četiri pristupa:
  - ❑ Jednostavni refleksi agenti
  - ❑ Model-bazirani refleksi agenti
  - ❑ Ciljno-bazirani agenti
  - ❑ Korisno-bazirani agenti



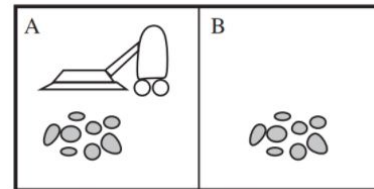
# Jednostavni refleksni agenti

**function** REFLEX-VACUUM-AGENT([*location, status*]) **returns** an action

**if** *status* = *Dirty* **then return** *Suck*

**else if** *location* = *A* **then return** *Right*

**else if** *location* = *B* **then return** *Left*



- ❑ Ova implementacija može sagledati samo jedno opažanje i na osnovu toga djelovati (pokrenuti akciju). Ignorišu se i sva prethodna opažanja (nema memoriju).
- ❑ Beneficije: manji programski kod, nema tabele.
- ❑ **Pravilo uslov-akcije:** koristi if/then logiku, detektuj objekat i reaguj.
- ❑ Primjer: self-driving car

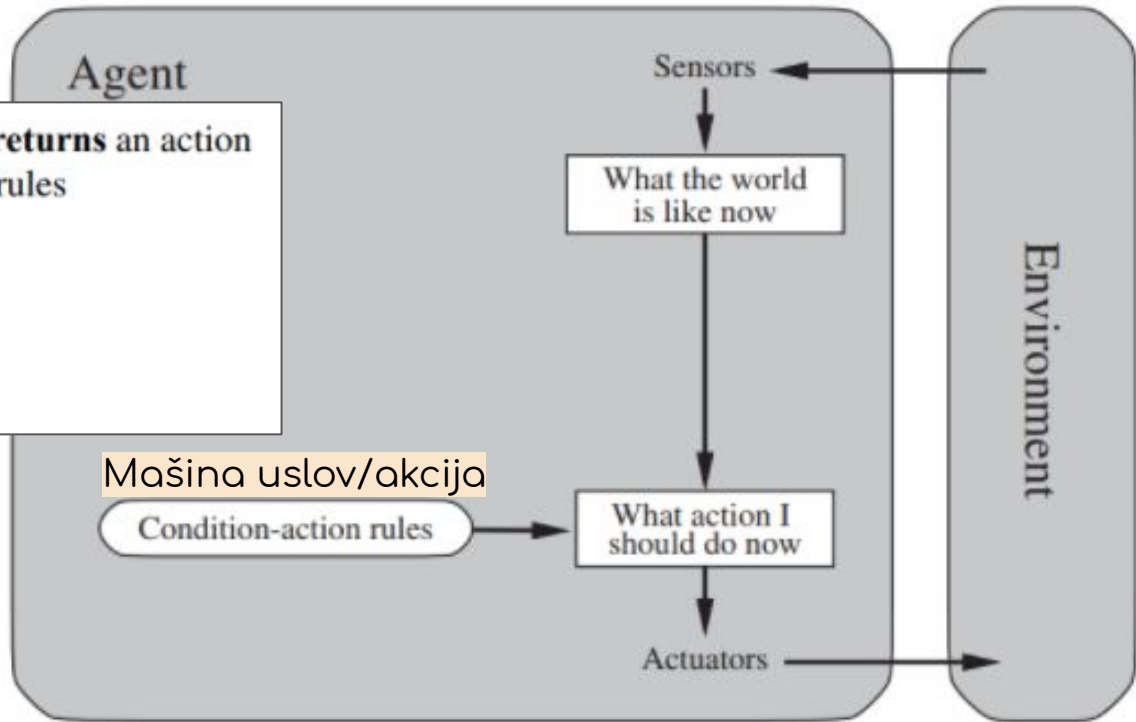


**IF** auto naprijed koči  
**THEN** koči

# Jednostavni refleksni agenti

```
function SIMPLE-REFLEX-AGENT(percept) returns an action  
persistent: rules, a set of condition–action rules  
  
state ← INTERPRET-INPUT(percept)  
rule ← RULE-MATCH(state, rules)  
action ← rule.ACTION  
return action
```

- ❑ Mašina može da se programira za različite situacije.
- ❑ Jednostavna implementacija, ali postoje ograničenja u nivou inteligencije.
- ❑ Agent će raditi dobro jedino u okruženje koje se ne mijenja.
- ❑ Primjer: self-driving cars: agent mora donijeti odluku na osnovu jedno frame-a (slike).



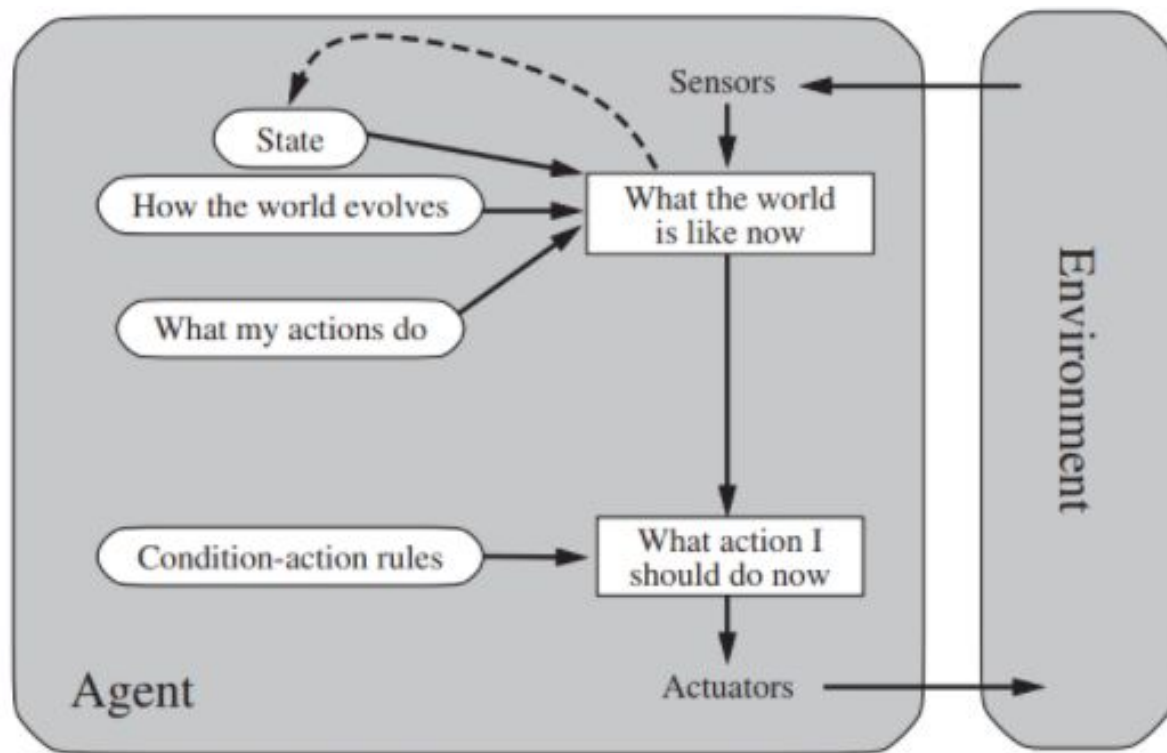
Može se desiti da agent završi u petlji. U tom slučaju potrebno je dodati neku vrstu slučajnosti.  
Šta se dešava kada agent nema senzor lokacije?

# Model-bazirani refleksni agenti

- ❑ Ukoliko je okruženje djelimično definisano onda je najbolje sačuvati historiju prethodnih dešavanja (opažanja, akcija).
- ❑ Agent ima definisano interno stanje
  - ❑ Kako se okruženje mijenja bez agenta?
  - ❑ Kako agent može izmijeniti okruženje?
- ❑ **Model**: znanje kojim se opisuje kako svijet funkcioniše.
- ❑ Agenti koji se baziraju na ovom pristupu nazivaju se: **model-bazirani agenti**.
- ❑ Primjer: self-driving car
  - ❑ Auto ima trenutnu sliku okruženja u kojem se nalazi. Tu sliku može uporediti sa slikama koje je prethodno pohranio i na osnovu toga detektovati promjene u okolini.

# Model-bazirani refleksni agenti

- ❑ **Ažurirano okruženje:**  
originalno stanje +  
kako se svijet može  
promijeniti (fizika) +  
kako moje akcije utiču  
na svijet
- ❑ **Odluka** se donosi na  
bazi **pravila**  
**uslov-akcije** i  
**ažuriranog okruženja**



# Model-bazirani refleksni agenti

```
function MODEL-BASED-REFLEX-AGENT(percept) returns an action
  persistent: state, the agent's current conception of the world state
               model, a description of how the next state depends on current state and action
               rules, a set of condition–action rules
               action, the most recent action, initially none

  state ← UPDATE-STATE(state, action, percept, model)
  rule ← RULE-MATCH(state, rules)
  action ← rule.ACTION
  return action
```

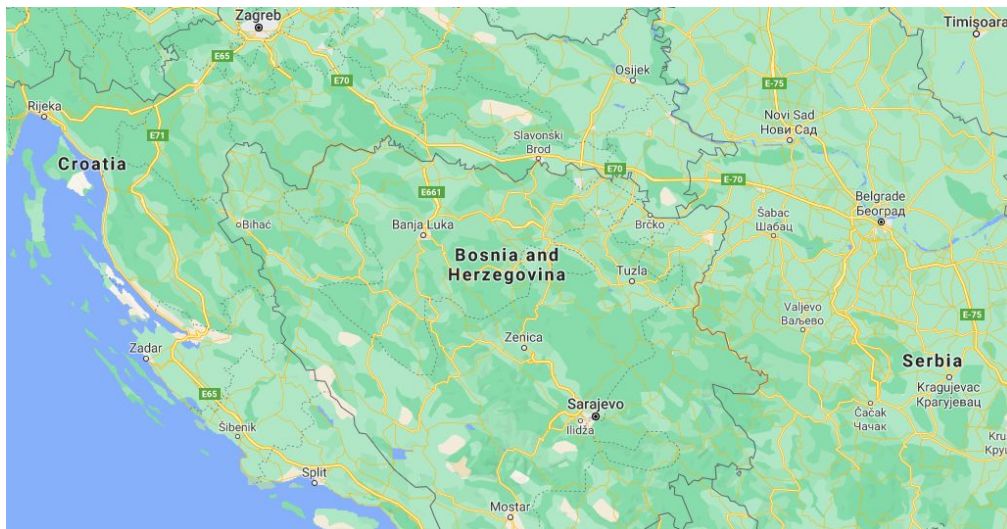
# Ciljno-bazirani agenti

- ❑ Kod određenih problema nije dovoljno poznavati samo okruženje.
- ❑ Agenti ponekad moraju imati određeni cilj.
  - ❑ Na primjer: self-driving car mora zna ciljnu destinaciju, tj. **cilj**.
- ❑ Cilj se može kombinovati sa okolinom, kako bi se došlo do što bolje odluke.
- ❑ **Pretraživanje** za sekvencom akcija koje dovode do cilja je posebno polje istraživanje, kao i **planiranje** dolaska do cilja.
  - ❑ Primjer: stanja sistema se mogu predstaviti kao čvorovi drveta, pri čemu putanja predstavlja promjenu stanja.
- ❑ Agenti bazirani na dostizanju cilja su više **fleksibilni**, ali manje efikasni (potrebno više resursa).
- ❑ Mogu modifikovati odluke na bazi prethodnih dešavanja.
- ❑ Ponašanje agenta se može lako modifikovati, na primjer promjenom destinacije.

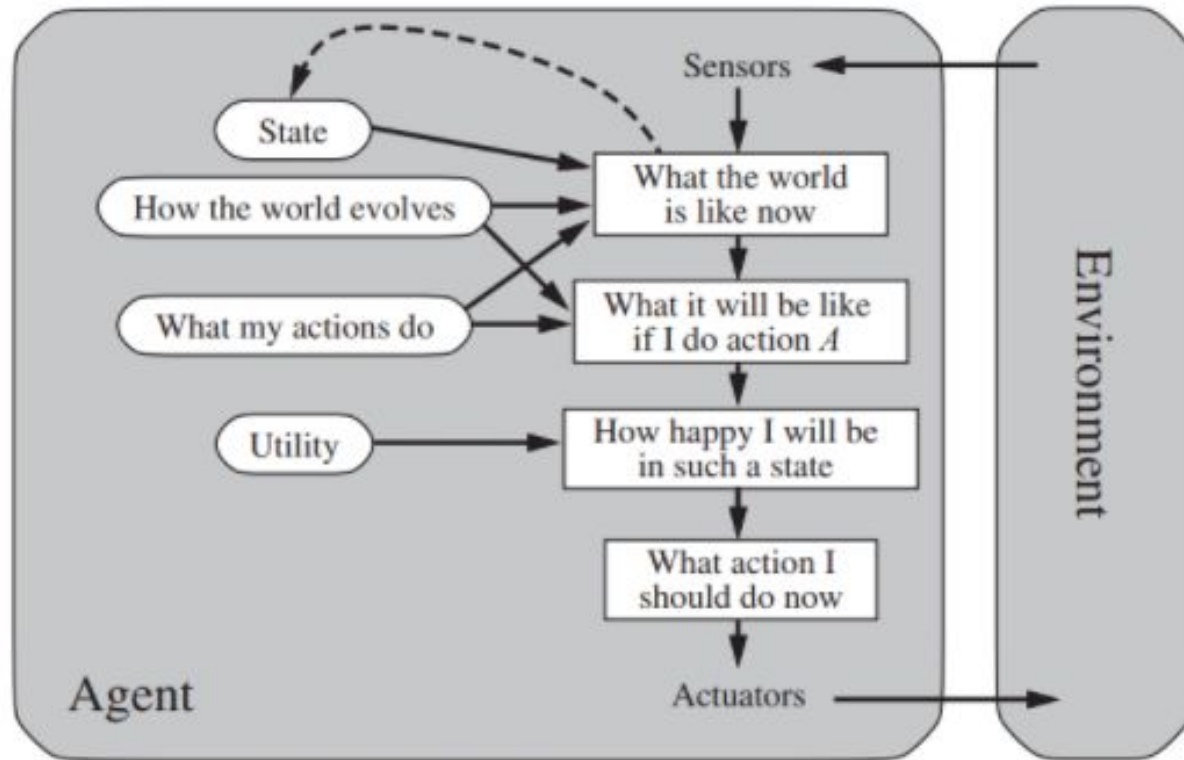
# Korisno-bazirani agenti

- ❑ Ciljevi nisu dovoljni da bi se došlo do nekog kvalitetnog rješenja.
  - ❑ Mnogo puteva vodi do jednog cilja. Koji od njih je optimalan?
  - ❑ Da li je cilj stići brzo, sigurno, pouzdano ili jeftinije?
  - ❑ Uvodi se mjera korisnosti koja se koristi za poređenje različitih puteva.
  - ❑ Sekvena prelaza stanja ocjenjuje se kako bi se stvorila slika o tome koji je put bolji.

Definiše se **funkcija korisnosti**.  
Agent se smatra racionalnim  
ukoliko napravi najbolju odluku u  
odnosu na zadati cilj ili postigne  
**očekivanu korisnost**.



# Korisno-bazirani agenti

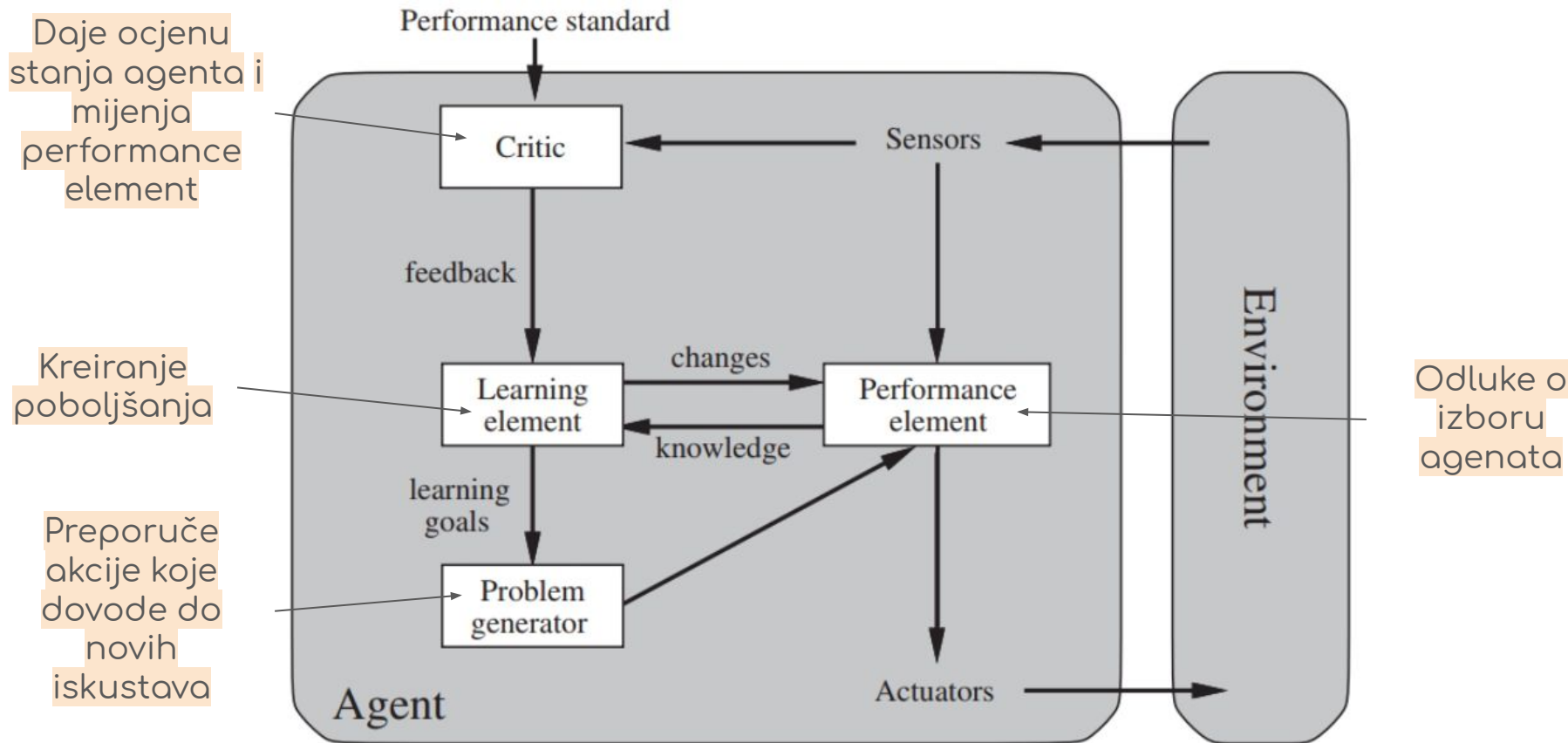




# Agenti za učenje

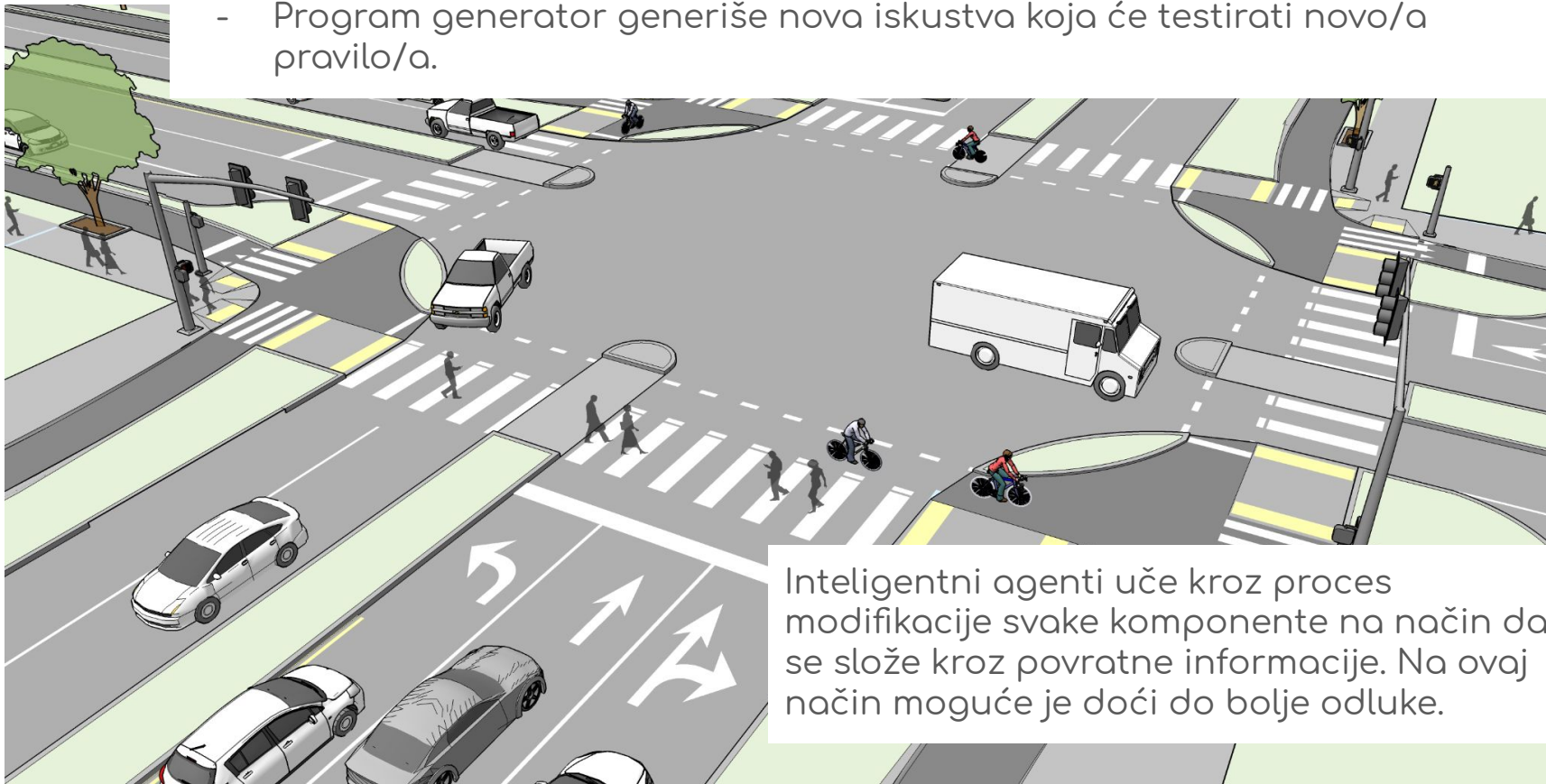
- ❑ Kako nastaju agenti?
- ❑ Programiranje nije jednostavno.
- ❑ Turing je predložio kreiranje mašine koja može da uči (sama se programira). Mašinu je zatim moguće podučavati.

# Agenti za učenje



# Primjer

- Taxi pokušava da pređe preko prelaza (performance element)
- Critic posmatra negativne reakcije okoline/drugih vozača
- Learning element definiše novo pravilo koje omogućuje sigurniji prolaz.
- Program generator generiše nova iskustva koja će testirati novo/a pravilo/a.



Inteligentni agenti uče kroz proces modifikacije svake komponente na način da se slože kroz povratne informacije. Na ovaj način moguće je doći do bolje odluke.

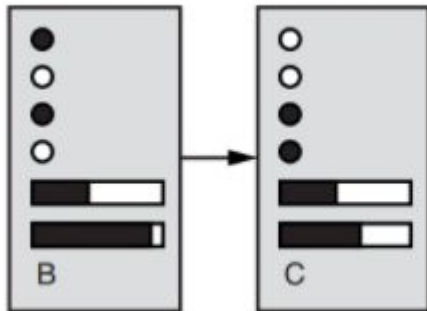
# Komponente agenta za opis okruženja



## Atomske

Stanje je crna kutija bez interne reprezentacije  
(Boolean vrijednosti, stanje je nedjeljivo)

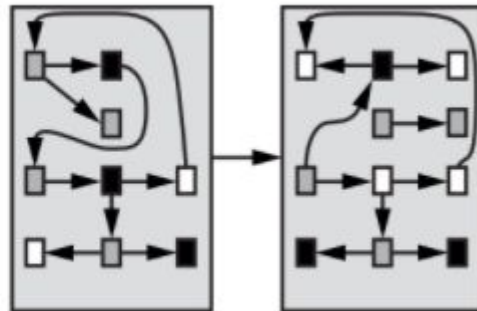
Primjeri: search and game-playing, Hidden Markov models, Markov decision processes.



## Faktorske

Stanje je vektor parametara (boolean, realne vrijednosti, skup simbola)

Primjeri: constraint satisfaction algorithms, propositional logic, planning, Bayesian networks, machine learning.

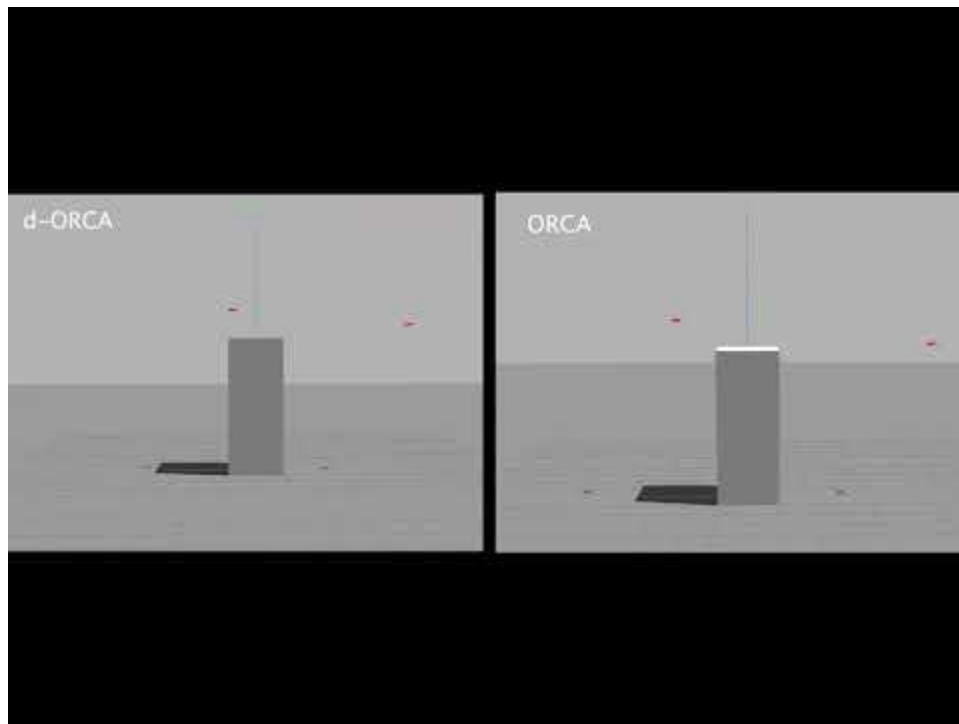


## Strukturne

Stanje uključuje objekte; svaki objekat može da ima atribute kao i relacije između objekata.

Primjeri: relational databases, first-order logic, first-order probability models, knowledge-based learning, natural language understanding, ...

# Primjer



# Primjer

