Veštačka inteligencija

Projekat – Blokiranje (Blockade)

Osnovne informacije

- Cilj projekta:
  - Formulacija problema
  - Implementacija algoritma za traženje (algoritma za igru)
  - Implementacija procene stanja korišćenjem pravila i zaključivanja
- Jezik: Python
- Broj ljudi po projektu: 3
- Datum objavljivanja projekta: 25.11.2021.
- Rok za predaju: 16.1.2022.

Ocenjivanje

- Broj poena:
  - Projekat nosi maksimalno 20% od konačne ocene
  - Poeni se odnose na kvalitet urađenog rešenja, kao i na aktivnost i zalaganje studenta
- Status:
  - Projekat je obavezan!
  - Minimalni broj poena koji se mora osvojiti je 5!
  - Očekuje od studenata da ozbiljno shvatite zaduženja!
  - Ukoliko ne uradite projekat u predviđenom roku, naredna prilika je tek sa sledećom generacijom, po pravilima koja će biti definisana za novi projekat!

Takmičenje/turnir

- Posle predaje projekta biće organizovano takmičenje.
- Planirani termin takmičenja je sredina januara.
- Prva tri mesta na turniru donose dodatne poene: 5 za prvo mesto, 3 za drugo i 2 za treće mesto (računaju se kao dodatni poeni za angažovanje u toku semestra).

Pravila ponašanja

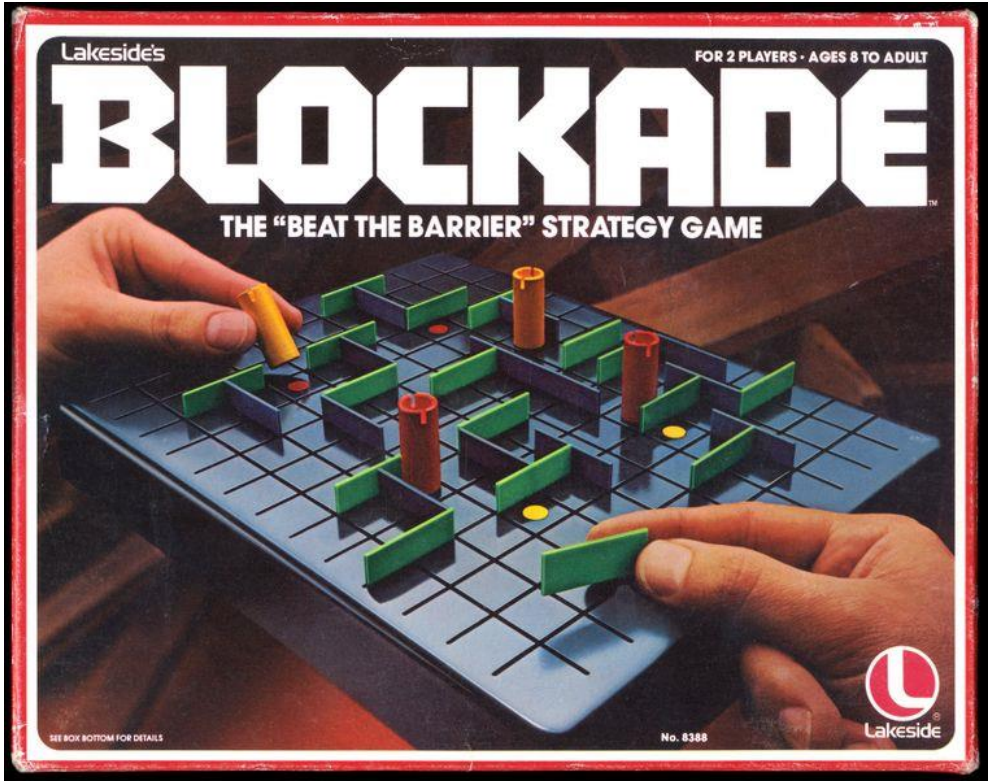
- Probajte da uradite projekat samostalno, bez pomoći kolega iz drugih timova i prepisivanja.
- Poštujte tuđi rad! Materijal sa Web-a i iz knjiga i radova možete da koristite, ali samo pod uslovom da za sve delove koda ili rešenja koje ste uzeli od nekog navedete referencu!
- Ne dozvolite da drugi prepisuje od vas, tj. da drugi koristi vaš rad i vaše rezultate!
- Ne dozvolite da član tima ne radi ništa! Dogovorite se i pronađite zaduženja koja on može da uradi. Ako mu ne ide, pronađite druga zaduženja.

Faze izrade projekta

- Formulacija problema i implementacija interfejsa
  - Rok: 12.12.2021. godine
- Implementacija operatora promene stanja
  - Rok: 26.12.2021. godine
- Implementacija Min-Max algoritma za traženje sa alfa-beta odsecanjem
  - Rok: 5.1.2022. godine
- Definicija heuristike (procena stanja)
  - Rok: 16.1.2022. godine

- Rezultat svake faze je izveštaj koji sadrži dokument sa obrazloženjem rešenja i datoteku sa kodom.

Igra Blokiranje (*Blockade*)

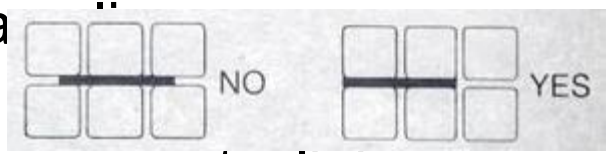


Opis problema *Blockade*

- Problem je igra *Blockade*
- Strateška igra zaobilaženja prepreka
- Tabla je dimenzija m×n, m vrsta (paran broj) i n kolona (neparan broj), što se definiše na početku igre (preporučeno je 11×14, a maksimalno 22×28)
- Dva igrača crni i beli (X i O) naizmenično odigravaju po jedan potez
- Svaki igrač ima dva pešaka i po k zelenih i plavih zidova što se definiše na početku igre (preporučeno je 9, a maksimalno 18)
- Tabla je na početku prazna (bez zidova) sa pešacima postavljenim na početna polja što se definiše na početku igre (preporučeno je da pešaci igrača X budu na poljima [4,4] i [8,4], a igrača O na poljima [4,11] and [8,11])
- Pobednik je igrač koji dovede jednog od svoja dva pešaka na početno polje protivnika
- Igra čovek protiv računara i moguće izabrati da prvi igra čovek ili računar

Pravila igre *Blockade*

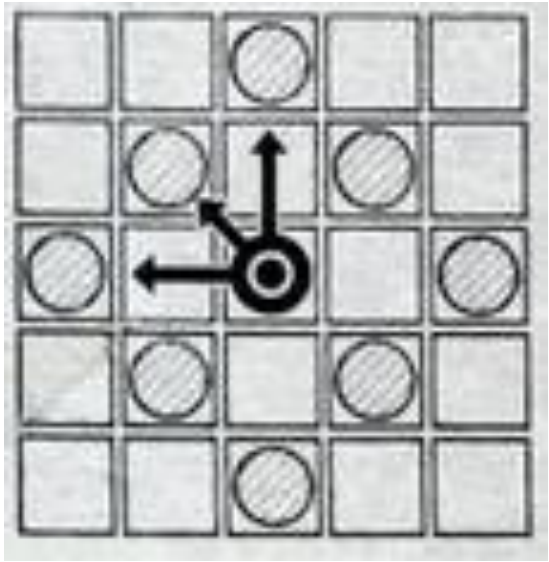
- Igrači povlače poteze naizmenično
- Igrač, u jednom potezu, pomera samo jednog pešaka u željenom pravcu i obavezno postavlja samo jedan zid (ako ima preostalih zidova)
- Pešak može da se pomeri tačno dva polja ulevo, udesno, nagore i nadole ili tačno jedno polje dijagonalno
- Zidovi se uvek nalaze između tačno četiri polja, tj. ne mogu počinjati na pola



- Zidovi plave boje se postavljaju u pravcu s leva udesno (horizontalno), a zidovi zelene boje u pravcu odozgo nadole (vertikalno).

Pravila igre *Blockade*

- Dozvoljeni potezi

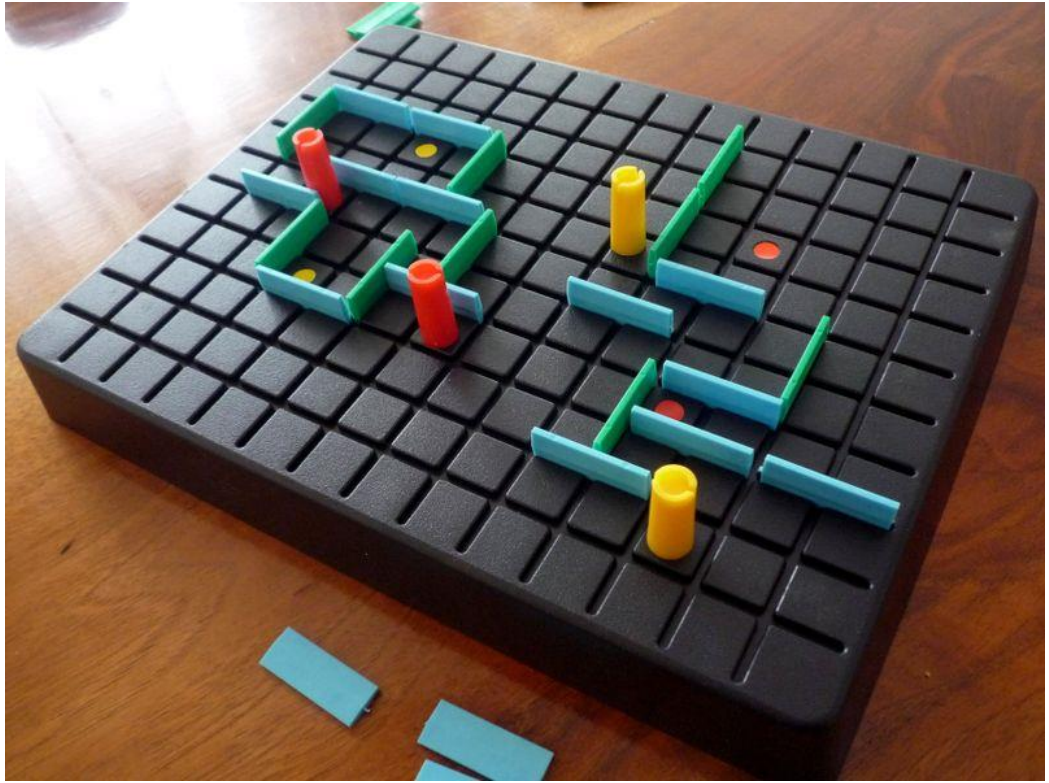


Pravila igre *Blockade*

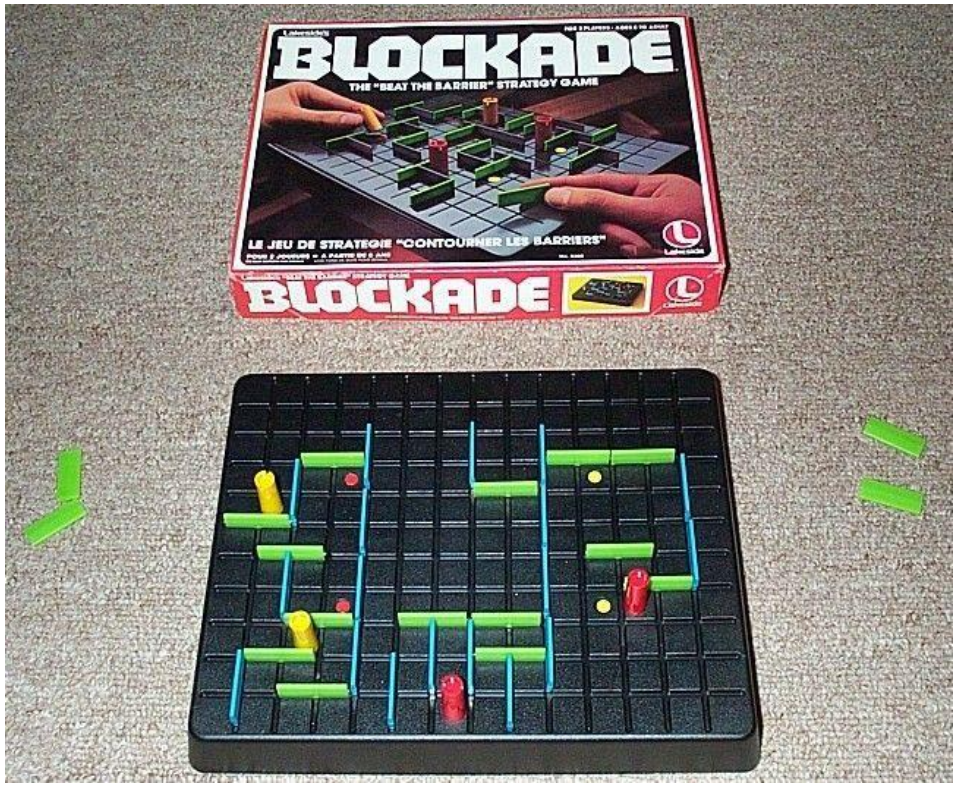
- Dva pešaka ne mogu se naći na istom polju
- Pešak ne može da ukloni protivničkog pešaka (osim sa početnog polja)
- Pešak ne može preskočiti zid
- Pešak može preskočiti drugog pešaka koje se nalazi na susednom polju
- Pešak se može pomeriti samo jedno polje ako je blokiran drugim pešakom
- Početna polja ne smeju biti ograđena zidovima
- Za svako do četiri početnog polja mora da postoji barem jedan put kojim se može stići do njega
- Početno polje se ne može zaštititi sopstvenim pešakom, jer je protivniku dozvoljeno da ukloni pešaka sa početnog polja
- Pešak koji se nalazi na polju susednom početnom polju protivnika može da stane na njega

Blockade – Početak igre

Blockade – Primer stanja igre



Blockade – Primer kraja igre



Blockade – Korisni linkovi

- Originalna pravila igre:
  - https://boardgamegeek.com/boardgame/2559/blockade
  - https://en.wikipedia.org/wiki/Blockade_(board_game)
  - https://en-academic.com/dic.nsf/enwiki/6995480

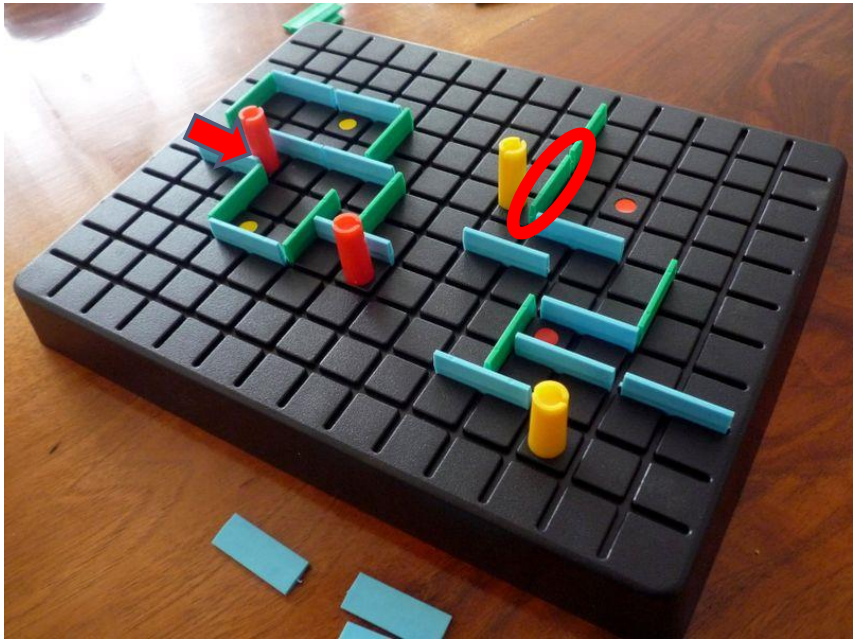
Zadatak I – Formulacija problema i interfejs (1)

- Definisati način za predstavljanje stanja problema (igre)
  - Tabla, pozicije pešaka na tabli, pozicije zidova na tabli, broje preostalih zidova
- Napisati funkciju za postavljanje početnog stanja
  - Definiše se na osnovu zadate veličine table, broja zidova i početnih pozicija pešaka
- Napisati funkcije za proveru kraja igre
  - Kraj je kada je pešak na protivničkom početnom polju
- **VAŽNO: NIJE POTREBNO realizovati funkcije koje obezbeđuju odigravanje partije (faza II)**

Zadatak I – Formulacija problema i interfejs (1)

- Omogućiti izbor ko će igrati prvi (čovek ili računar)
- Prvi igra uvek igrač X, a drugi igrač O
- Implementirati funkcije koje obezbeđuju unos početnih parametara igre
  - Dimenzija table i broja zidovai početnih pozicija pešaka
- Implementirati funkcije koje obezbeđuju prikaz proizvoljnog stanja problema (igre)
- Realizovati funkcije koje na osnovu zadatog poteza igrača menjaju trenutno stanje problema (igre) i prelaze u novo
  - Potez je oblika igrač (X ili O) broj pešaka (1 ili 2) potez (6 3 – vrsta i kolona) i boje i pozicije gornjeg levog polja od ćetiri koja okružuje zid (p 4 3 – zid je između polja [4,3], [4,4], [5,3], [5,4])
- Realizovati funkcije koje proveravaju da li je potez valjan
  - Proveriti da li je moguće pomeriti pešaka u odgovarajućem pravcu zadati broj polja
  - Proveriti da li je moguće postaviti zid na zadato mesto na tabli
- **VAŽNO: NIJE POTREBNO proveravati da li se postavljanjem zida na zadato mesto zatvara put do bilo kog početnog polja (faza II)**

Zadatak I – Interfejs

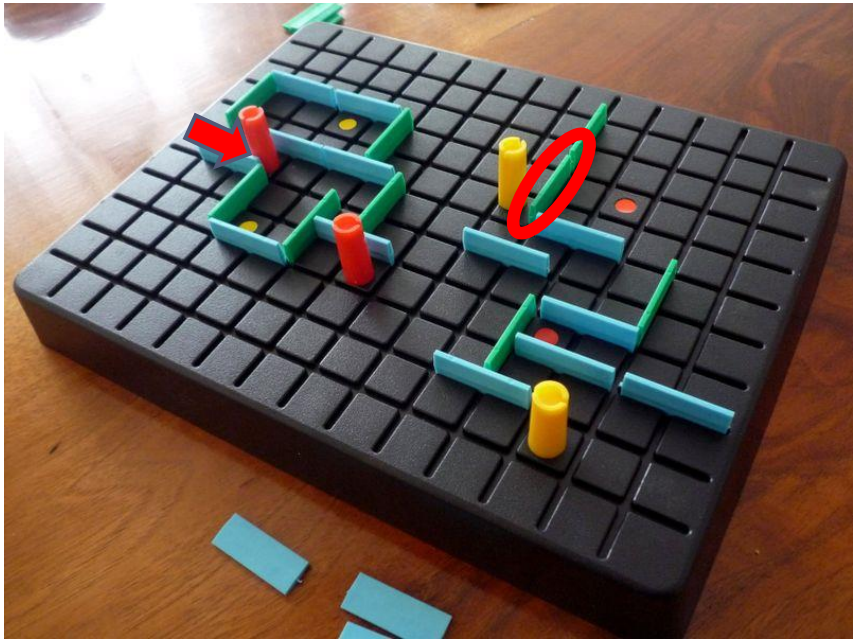


```
  1 2 3 4 5 6 7 8 9 A B C D E
  = = = = = = = = = = = = = =
1‖ |   |   |   |   |   |   |   |   |   |   |   |   ‖1
  - - - - - - - - - - - - - -
2‖ |   |   |   |   |   |   |   ‖   |   |   |   |   ‖2
  - - - - - - - - - - - - - -
3‖ |   |   |   |   |   |   |   ‖   |   |   |   |   ‖3
  - = = = = - - - - - - - - -
4‖ ‖   |   |   ‖   |   |   |   |   |   |   |   |   ‖4
  - - - - - - - - - - - - - -
5‖ ‖   |   |   ‖   |   |   |O  |   |   |   |   |   ‖5
  - - = = = = - - - - = = - - -
6‖X   |   |   ‖   |   |   |   |   ‖   |   |   ‖6
  - - = = - - - - - = = - - - -
7‖ |   |   ‖   |   ‖   |   |   |   |   ‖   |   |   ‖7
  - - - - - - = = - - - = = - -
8‖ |   |   ‖   |   ‖   |X  |   |   |   ‖   |   |   ‖8
  - - - = = - - - - - - = = = =
9‖ |   |   |   |   |   |   |   |   |   ‖   |   |   ‖9
  - - - - - - - - - - = = - - -
A‖ |   |   |   |   |   |   |   |   |   |O  |   |   ‖A
  - - - - - - - - - - - - - - -
B‖ |   |   |   |   |   |   |   |   |   |   |   |   ‖B
  = = = = = = = = = = = = = =
  1 2 3 4 5 6 7 8 9 A B C D E

Uspravni zid: ‖ (0x01C1) ili I

Potez: [X 2] [6 3] [Z 4 9]
```

Zadatak I – Interfejs



```
  1 2 3 4 5 6 7 8 9 A B C D E
  = = = = = = = = = = = = = =
1‖ | | | | | | | | | | | | ‖1
  - - - - - - - - - - - - -
2‖ | | | | | | | | ‖ | | | ‖2
  - - - - - - - - - - - - -
3‖ | | | | | | | | ‖ | | | ‖3
  - = = = = - - - - = - - - -
4‖ ‖ | | | ‖ | | | | ‖ | | | ‖4
  - - - - - - - - = - - - -
5‖ ‖ | | | ‖ | | | |O‖ | | | ‖5
  - - = = = - - - = = - - -
6‖ | |X| | | ‖ | | | | ‖ | ‖6
  - = = - - - - - = = - - - -
7‖ | | ‖ | ‖ ‖ | | | | | ‖ | ‖7
  - - - - - = = - - - = = - -
8‖ | | ‖ | ‖ |X| | | ‖ | | | ‖8
  - - - = = - - - - - = = = =
9‖ | | | | | | | | | ‖ | | | ‖9
  - - - - - - - - - = = - - -
A‖ | | | | | | | | | |O| | ‖A
  - - - - - - - - - - - - - -
B‖ | | | | | | | | | | | | ‖B
  = = = = = = = = = = = = = =
  1 2 3 4 5 6 7 8 9 A B C D E
```

Zadatak II – Operatori promene stanja

- Realizovati funkcije koje proveravaju da li se postavljanjem zida na zadato mesto zatvara put do bilo kog početnog polja
- Napisati funkcije za operatore promene stanja problema (igre) u opštem slučaju (proizvoljno stanje na tabli)
  - Na osnovu trenutne (proizvoljne) situacije na tabli (stanja) i zadatog (validnog) poteza formirati novu situaciju na tabli (stanje). Ne menjati postojeće stanje već napraviti novo i na njemu odigrati potez.
  - Na osnovu trenutne (proizvoljne) situacije u kocki (stanja) i igrača koji je na potezu formira listu svih mogućih situacija na tabli (stanja) povlačenjem samo jednog poteza, korišćenjem funkcije iz prethodne tačke
- Realizovati funkcije koje obezbeđuju odigravanje partije između dva igrača (**dva čoveka, ne računara i čoveka**)
  - unos poteza i provera da li je potez moguć
  - ukoliko nije moguć zahtevati unos novog poteza
  - ukoliko je moguć odigrati ga i promeniti trenutno stanje
  - prikazati novonastalo stanje sistema
  - proveru kraja i određivanje pobednika u igri

Zadatak III – Min-max algoritam

- Implementirati Min-Max algoritam sa alfa-beta odsecanjem za zadati problem (na osnovu zadatog stanja problema)
- Obezbediti da funkcija Min-Max sa alfa-beta odsecanjem ima ulazni parametar kojim se definiše dubina pretraživanja
- Obezbediti da funkcija Min-Max sa alfa-beta odsecanjem vrati potez koji treba odigrati ili stanje u koje treba preći
- Realizovati funkcije koje obezbeđuju odigravanje partije između čoveka i računara
- **VAŽNO: NIJE POTREBNO implementirati funkciju za određivanje heuristike (faza III)**
  - Napraviti funkciju koja za odgovarajuća stanja vraća karakteristične vrednosti samo u svrhu testiranja ispravnosti napravljenog Min-Max algoritma

Zadatak IV – Heuristika

- U implementaciju Min-Max-a sa alfa-beta odsecanjem dodati funkciju za procenu stanja koja se poziva kada se dostigne zadata dubina traženja.
- Implementirati funkciju koja vrši procenu stanja na osnovu pravila zaključivanja
- Funkcija za procenu stanja kao parametre treba da ima oznaku igrača za kojeg računa valjanost stanja, kao i samo stanje za koju se računa procena.
- Procena stanja se mora vršiti isključivo korišćenjem mehanizma zaključivanja nad prethodno definisanim skupom pravila. Zadatak je formulisati skup pravila i iskoristiti ih na adekvatan način za izračunavanje heuristike.
- Za izvođenje potrebnih zaključaka (izvršavanje upita nad skupom činjenica kojima se opisuje stanje) koristiti mašinu za zaključivanje.
- Implementirati funkciju koja prevodi stanje u listu činjenica …