*Projekat iz veštačke inteligencije*

*Tema:*

Havannah

III Faza

*Članovi tima: Naziv tima:*

Ana Stojanović 15905 Anonymous Mink

Damjan Trifunović 15948

Milica Todorović 15937

Sadržaj

[Predstavljanje table 3](#_Toc532745055)

[Globalne promenljive I konstante 3](#_Toc532745056)

[Generisanje matrične reprezentacije table 4](#_Toc532745057)

[Učitavanje dimenzija table I postavljanje table 5](#_Toc532745058)

[Igranje poteza 5](#_Toc532745059)

[Odabir igrača 7](#_Toc532745060)

[Igra 7](#_Toc532745061)

[Štampanje table 8](#_Toc532745062)

[Pomoćne funkcije 9](#_Toc532745063)

[Funkcije za ispitivanje ciljnog stanja 11](#_Toc532745064)

[Operatori stanja 14](#_Toc532745065)

[Algoritmi za igranje poteza 15](#_Toc532745066)

[Reference: 16](#_Toc532745067)

# Predstavljanje table

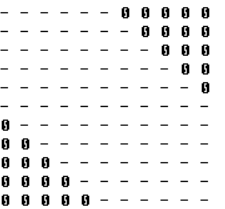
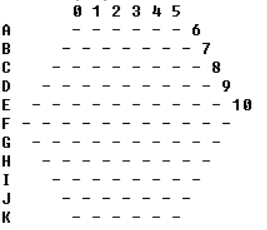
Tabla je interno predstavljena u vidu modifikovane matrice (videti sliku). Modifikacija je izvršena da bi elementima moglo da se pristupa direktno preko indeksa matrice. Polje [A,0] table se slika u polje [0,0] matrice i predstavlja validno polje. Validno polje može imati vrednost „X“, „O“ ili „-“, gde je „X“ prvi igrač, „O“ je drugi igrač a „-“ je prazno polje. Polje [G, 0] table se slika u polje [6, 0] matrice, sto je nevalidno polje. Širina matrice je [2\*n-1, 2\*n-1].

Tabla je zapamćena kao lisp matrica, tako da su validna polja strukture tipa „cell“, koje imaju vrednost „X“, „O“ ili „-“ a nevalidna polja „0“.

Primer:

#2A((#S(CELL :VALUE - :PARENT 0 :GROUPSIZE :ISEDGE 0 :ISCORNER 1 :RINGDEPTH 0)

#S(CELL :VALUE - :PARENT :GROUPSIZE :ISEDGE 1 :ISCORNER 0 :RINGDEPTH 0) . . . . ))

# Globalne promenljive I konstante

**File : globalvariables.lisp**

**emptyField** –konstanta koja definiše prazno polje u matrici kao znak “-”

**invalidField** – konstanta koja definiše nevalidno polje u matrici kao broj “0”

**N** – dimenzija table matrice koju unosi korisnik

**matrixDim** – dimenzija matrice u matričnoj reprezantaciji table (2n - 1)

**board** – globalna promenljiva koja čuva matričnu reprezentaciju table

**firstPlayer** - konstanta koja definiše simbol prvog igrača kao znak “X”

**secondPlayer** – konstanta koja definiše simbol drugog igrača kao znak “O”

**currentPlayer** – globalna promenljiva koja čuva trenutnog igrača

**human** –globalna promenljiva koja definiše igrača koji je čovek

**computer** - globalna promenljiva koja definiše igrača koji je računar

**numMoves** – broj do sada odigranih poteza

**maxNumMoves** –broj polja na tabli ( (2n-1)2 – n(n-1))

**gameState** – globalna promenljiva koja čuva stanje u kom se nalazi igra – “0” (igra traje), “1” (igra je završena), “2” (igra je završena i ishod je nerešeno)

**ringSize** – minimalna veličina prstena, koristi se za ispitivanje ring ciljnog stanja

**bitCount** –niz čiji element predstavlja broj jedinica u binarnoj reprezentaciji odgovarajućeg indexa, niz ima 64 elementa

**neighbours** – niz od 6 elemenata koji predstavljaju udaljenost suseda od datog elementa

**cell** – struktura koja predstavlja jedno polje matrice. Value je vrednost polja u matrici (“X”, “O”, “0”, “-“). Polja parent, groupSize, isEdge, isCorner su ubačena zbog algoritma union rank. Polje ringDepth dodato je zbog provere ring ciljnog stanja.

# Generisanje matrične reprezentacije table

**File: game.lisp**

**(generateValidRow rowIndex colIndex member count)**

*Ulaz:*

rowIndex – Indeks reda koji se generiše

colIndex - Kolona od koje počinje generisanje elemenata

member – Element koji se umeće u red

count – Koliko puta se ponavlja element

*Izlaz*:

Celokupni red matrice

*Opis:*

Generiše red matrice. Za validna polja generiše stukturu tipa cell, a nevalidna polja postavlja na \*invalidField\*.

**(generateRow rowIndex firstMember firstCount secondMember secondCount)**

*Ulaz:*

firstMember – Prvi element

firstCount – Broj ponavljanja prvog elementa

secondMember – Drugi element

secondCount – Broj ponavljanja drugog elementa

*Opis:*

Dvaput poziva funkciju generateValidRow, jednom za validna polja, jednom za nevalidna polja.

**(generateMatrix rowIndex firstMember firstCount secondMember secondCount)**

*Ulaz:*

rowIndex – Indeks reda koji se trenutno generiše

firstMember – Vrednost prvog elementa

firstCount – Broj ponavljanja prvog elementa

secondMember – Vrednost drugog element

secondCount – Broj ponavljanja drugog elementa

*Izlaz:*

Generisana matrica koja predstavlja tablu opisana u delu *Predstavljanje table*

*Opis:*

Generiše matricu opisanu u delu Predstavljanje table. Validna polja su strukture tipa cell, a nevalidna polja imaju vrednost \*invalidField\*.

**(initMatrix )**

*Ulaz:*

/

*Izlaz:*

Generisana matrica koja predstavlja tablu opisana u delu *Predstavljanje table*

*Opis:*

Funkcija koja poziva funkciju generate matrix.

# Učitavanje dimenzija table I postavljanje table

**File: game.lisp**

**(setDimension )**

*Ulaz:*

/

*Izlaz:*

Učitanje dimenzije table I generisana matrična reprezentacija table.

*Opis:*

Sa standardnog izlaze se učitava dimenzija table, a zatim se setuje globalna promenljiva koja predstavlja dimenziju matrice (2\*dimenzija\_table - 1) I generiše se tabla korišćenjem funkcije *initMatrix*.

# Igranje poteza

**File: game.lisp**

**(setElement el i j matrix)**

*Ulaz:*

el – element koji se ubacuje u matricu

i – indeks reda matrice

j – indeks kolone matrice

matrix – matrica u koju se element unosi

*Izlaz:*

Matrica sa unetim elementom na [i, j] poziciji

*Opis:*

Funkcija služi za ubacivanje prosleđenog elementa na poziciju [i, j] u prosleđenoj matrici i kao rezultat vraća izmenjenu matricu.

**(validateMove i j boardState)**

*Ulaz:*

i – index vrste

j – index kolone

boardState – trenutno stanje table

*Izlaz:*

True ili false u zavisnosti od toga da li je potez validan.

*Opis:*

Funkcija validira igranje poteza na datoj poziciji u matrici. Index vrste i index kolone j su validni ukoliko su pozitivni brojevi, dok je potez validan ukoliko se na poziciji [I,j] nalazi \**emptyField*\*.

**(playMove player)**

*Ulaz:*

player – oznaka igrača koji igra potez

*Izlaz:*

Globalna promenljiva \*board\* dobija vrednost matrice sa odigranim potezom.

*Opis:*

Funkcija zahteva od korisnika da unese koordinate polja (slovo za red, broj za kolonu). Ako su uneti podaci validni, matrica table se ažurira tako da sadrži potez koji je odigran i ta vrednost se upisuje u globalnu promenljivu koja čuva matricu table.

**(playMoveComputer)**

*Ulaz:*

/

*Izlaz:*

Ubacuje potez koji igra kompjuter na globalnu promenljivu \*board\*

*Opis:*

Funkcija poziva algoritam *negamax* da bi odabrala najbolji potez za kompjuter i inkrementira globalnu promenjlivu \*numMoves\*. U slučaju da je došlo do krajnjeg stanja, postavlja globalnu promenljivu \*gameState\*.

# Odabir igrača

**File: game.lisp**

**(choosePlayer)**

*Ulaz:*

/

*Izlaz:*

Postavljaju se globalne promenljive koje određuju da li je prvi igrač računar ili čovek.

*Opis:*

Funkcija zahteva od korisnika da unese da li će prvi igrač biti čovek ili računar (korisnik unosi “h” ili “c”). Postavljaju se globalne promenljive \*human\* i \*computer\* vrednostima \*firstPlayer\* (‘X) i \*secondPlayer\* (‘O), u zavisnosti od izbora korisnika.

**(switchCurrentPlayer)**

*Opis:*

Funkcija postavlja \*currentPlayer\* na suprotnog igraca u odnosu na prethodnog.

# Igra

**File: game.lisp**

**(isEndGame )**

*Ulaz:*

/

*Izlaz:* t – ukoliko je došlo do kraja igre; nil – ukoliko igra idalje traje

*Opis:* Funkcija proverava da li je došlo do kraja igre, ispitivanjem \**gameState*\* promenljive i štampa odgovarajuću poruku ukoliko je došlo do kraja igre.

**(playGameHumans )**

*Ulaz:*

/

*Izlaz:*

/

*Opis:* Funkcija štampa tablu i omogućava igranje partije između dva čoveka. Ispituje da li je došlo do ciljnog stanja pozivanjem funkcije *isEndGame*, ukoliko igra i dalje traje menja igrača koji je trenutno na potezu i ponavlja ceo proces.

**(playGameComputer)**

*Ulaz:*

/

*Izlaz:*

/

*Opis:* Funkcija štampa tablu i omogućava igranje partije između čoveka i kompjutera. Ispituje da li je došlo do ciljnog stanja pozivanjem funkcije *isEndGame*, ukoliko igra i dalje traje menja igrača koji je trenutno na potezu i ponavlja ceo proces.

**(havannah)**

*Ulaz:*

/

*Izlaz:*

/

*Opis:* Funkcija inicijalizuje gameState, currentPlayer i dimenzije. Omogućava odabir partije sa čovekom ili kompjuterom i poziva funkcije playGameHuman ili playGameComputer prvi put. Kad se igra završi, nudi igraču da restartuje igru. Ova funkcija je glavna funkcija koja pokreće celu igru.

# Štampanje table

**File: print-board.lisp**

**(printFirstRow count)**

*Ulaz:*

count – Redni broj elementa u redu koji se štampa. Uvek se prosledjuje -1.

*Izlaz:*

Funkcija vraća null i štampa na standardni izlaz.

*Opis:*

Funkcija štampa novi red (~%) i prvi red matrice. Prvi red se sastoji od n+1 blanko znakova i brojeva od 0 do n-1. Koristi globalnu promenljivu \*n\*.

**(printBoard)**

*Ulaz:*

board

*Izlaz:*

Null

*Opis:*

Funkcija štampa tablu sa indeksima kolone i slovima koja označavaju redove na standardni izlaz.

# Pomoćne funkcije

**File: helpers.lisp**

**(isCorner i j)**

*Ulaz:*

I – index vrste

J – index kolone

*Izlaz:*

0 ili 1

*Opis:*

Funkcija proverava da li je polje zadato sa *[i,j]* polje u uglu table, ukoliko jeste vraća 1 u suprotnom 0.

**(isEdge i j)**

*Ulaz:*

I – index vrste

J – index kolone

*Izlaz:*

Broj koji je stepen dvojke, od 0 - 32

*Opis:*

Funkcija proverava da li je polje zadato sa *[i,j]* polje na stranici table, ukoliko jeste vraća 2x, pri čemu je x odgovarajuća stranica table (od 0 – 5), u suprotnom vraća 0.

**(parentIndex i j dim)**

*Ulaz:*

I – index vrste

J – index kolone

Dim – dimenzija table

*Izlaz:*

I\*dim + j

*Opis:*

Funkcija vraća index niza ekvivalentan indexiranju matrice. (index niza = *i*\**dim* + *j*)

**(getColumnFromIndex index dim)**

*Ulaz:*

Index – index, indexiranje niza

Dim – dimenzija table

*Izlaz:*

Index mod dim

*Opis:*

Funkcija vraća index kolone matrice iz indexa niza. (kolona matrice = *index* mod *j*)

**(getRowFromIndex index column dim)**

*Ulaz:*

Index – index, indexiranje niza

Column – index kolone

Dim – dimenzija table

*Izlaz:*

(Index – column)/ dim

*Opis:*

Funkcija vraća index vrste matrice iz indexa niza.

(vrsta matrice =(*Index* – *column*)/ *dim* )

**(findNeighbours board row col currentPlayer)**

*Ulaz:*

board – tabla

row – index vrste

col – index kolone

currentPlayer – igrač čiji se susedi traže

*Izlaz:*

Primer: (0 2)

*Opis:*

Funkcija vraća listu suseda datog elementa na kojima se nalaze žetoni datog *currentPlayer*-a. Elementi liste su u formatu indexiranja niza (koristi se funkcija *parentIndex*)

**(validateNeighbourIndex row col indexNb currentPlayer board)**

*Ulaz:*

board – tabla

row – index vrste

col – index kolone

currentPlayer – igrač čiji se susedi traže

indexNb – index suseda

*Izlaz:*

Primer: (0 2)

*Opis:*

Funkcija vraća index vrste i kolone odgovarajućeg suseda datog elementa ukoliko se na njemu nalazi žeton datog *currentPlayer*-a. Sused koji se traži dat je promenljivom *indexNb*.

# Funkcije za ispitivanje ciljnog stanja

**File: union-rank.lisp**

**(root index dim boardState)**

*Ulaz:*

Index – index elementa čiji root element se traži

BoardState – Tabla

Dim – Dimenzija table

*Izlaz:*

Root datog elementa

*Opis:*

Funkcija vraća root element datog elementa, koristeći union-rank (disjoint set) sa path compression algoritmom.

**(isInUnion firstNodeIndex secondNodeIndex board dim)**

*Ulaz:*

FirstNodeIndex – Indeks prvog elementa

SecondNodeIndex – Ideks drugog elementa

Board – Trenutno stanje table

Dim – Dimenzija table

*Izlaz:*

True ili False

*Opis:*

Funkcija proverava da li su elementi sa indeksima fistNodeIndex i secondNodeIndex u uniji.

**(union-rank firstNodeIndex secondNodeIndex dim boardState)**

*Ulaz:*

firstNodeIndex, secondNodeIndex – indexi elemenata koje treba spojiti

BoardState – Tabla

Dim – Dimenzija table

*Izlaz:*

Root unije nastale spajanjem datih elemenata

*Opis:*

Funkcija spaja data dva elementa u jedan skup (uniju) koristeći disjoint set algoritam. Manji skup pridružuje većem i setuje novi broj corner-a i edge-eva koji se sada nalaze u novonastaloj uniji. Povratna vrednost je root novonastale unije.

**(uniteNeighboursWithList neighbourList row col board dim currentPlayer)**

*Ulaz:*

neighbourList – lista suseda datog elementa

row, col – index reda i kolone

board – Tabla

dim – Dimenzija table

currentPlayer – igrač koji je na potezu

*Izlaz:*

/

*Opis:*

Nakon što se odigra potez funkcija pravi uniju datog novog elementa i njegovih suseda. Funkcija takođe proverava da li je došlo do nekog od ciljnih stanja i ukoliko jeste setuje promenljivu \**gameState*\* na 1.

**(uniteNeighbours row col board dim currentPlayer)**

*Ulaz:*

row, col – index reda i kolone

board – Tabla

dim – Dimenzija table

currentPlayer – igrač koji je na potezu

*Izlaz:*

/

*Opis:*

Funckija poziva funkciju *uniteNeighboursWithList* i prosleđuje joj listu suseda koristeći funkciju *findNeighbours*.

**(uniteNeighboursWithListComputer neighbourList rowIndex colIndex board dim currentPlayer)**

*Ulaz:*

neighbourList – Lista suseda datog elementa

rowIndex, colIndex – Index reda i kolone u matrici

board – Trenutno stanje table

dim – Dimenzija table

currentPlayer – igrač koji je na potezu

*Izlaz:*

/

*Opis:*

Nakon što se odigra potez, funkcija pravi uniju datog novog elementa i njegovih suseda. Funkcija je predviđena za korišćenje u okviru algoritma koji računa najbolji potez za kompjuter.

**(uniteNeighboursComputer row col board dim currentPlayer)**

*Ulaz:*

row, col – Index reda i kolone u matrici

board – Trenutno stanje table

dim – Dimenzija table

currentPlayer – igrač koji je na potezu

*Izlaz:*

/

*Opis:*

Funckija poziva funkciju *uniteNeighboursWithListComputer* i prosleđuje joj listu suseda koristeći funkciju *findNeighboursNew.*

**(checkRing row col currentPlayer ringSize board)**

*Ulaz:*

row – Red u kom je poslednji odigran potez

col – Kolona u kojoj je poslednji odigran potez

currentPlayer – Igrač koji je poslednji igrao potez

ringSize – Minimalna veličina prstena (Podrazumevano 6)

board – Tabla na kojoj se igra

*Izlaz:*

True ako postoji ring na tabli a false ako ne postoji.

*Opis:*

Funkcija traži prsten na tabli, na osnovu poslednje dodatog poteza. U odnosu na poslednji potez, pretražuje komšije polja u četri smera (gore-levo, levo, dole-levo, dole-desno), i za svakg komšiju koji je validan i pripada istom igraču poziva funkciju followRing.

**(followRing row col dir currentPlayer depth ringSize board)**

*Ulaz:*

row – Red u kom se nalazi komšija za kog je pozvana funkcija

col – Kolona u kojoj se nalazi komšija za kog je pozvana funkcija

dir – Indeks komšije za kog je pozvana fukncija

currentPlayer – Igrač koji je poslednji igrao potez

depth – Trenutna dubina detektovanog prstena

ringSize – Minimalna veličina prstena (Podrazumevano 6)

board – Tabla na kojoj se igra

*Izlaz:*

True ako postoji ring a false ako ne postoji.

*Opis:*

Funkcija traži u tri smera u onosu na prosledjeno polje (smerovi se odredjuju na osnovu indeksa dir). Funkcija se rekurzivno poziva se dok se ne stigne do polja koje je već obrađeno ili se ne obrade sva validna polja.

# Operatori stanja

**File: operatori-stanja.lisp**

**(copyMatrix boardState dim)**

*Ulaz:*

BoardState – Tabla koja se kopira

Dim – Dimenzija table koja se kopira

*Izlaz:*

Kopija table

*Opis:*

Funkcija pravi kopiju table koja joj se prosledi.

**(getNewState player i j boardState)**

*Ulaz:*

Player – Igrač za kog se igra potez

i – Red u kome se igra potez

j – Kolona u kojoj se igra potez

BoardState – Tabla na kojoj se igra potez

*Izlaz:*

Tabla sa odigranim potezom

*Opis:*

Funkcija igra potez na kopiji table, bez menjanja \*board\* promenljive.

**(getPossibleStates boardState player dim)**

*Ulaz:*

BoardState – Trenutno stanje table za koje se generišu sva moguća sledeća stanja.

Player – Igrač koji je na potezu

Dim – Dimenzija table

*Izlaz:*

Sva moguća sledeća stanja za datu tablu.

*Opis:*

Funkcija generiše sva moguća sledeća stanja na osnovu trenutnog stanja table i trenutnog igrača.

# Algoritmi za igranje poteza

**File: alpha-beta.cl**

**(playerToCurrent player)**

*Ulaz:*

Player – Igrac koji je trenutno na potezu. [ 1 – computer, -1 čovek]

*Izlaz:*

Simbol odgovarajućeg igrača [‘X’ ili ‘O’]

*Opis:*

Funkcija pretvara kod igrača u simbol koji se vidi na tabli.

**(checkWin board lastMove numMoves maxPlayer)**

*Ulaz:*

Board - Trenutno stanje table

LastMove – Poslednji odigran potez

NumMoves – Ukupno odigranih poteza

MaxPlayer - Igrač za kog se proverava pobeda

*Izlaz:*

True ili False

*Opis:*

Funkcija proverava da li je došlo do pobede za prosleđenog igrača (ring, fork i bridge). Poziva funkciju *checkRing* i proverava broj “corner-a” i “edge-eva”.

**(evaluate board lastMove numMoves)**

*Ulaz:*

Board - Trenutno stanje table

LastMove – Poslednji odigran potez

NumMoves – Ukupno odigranih poteza

*Izlaz:*

Evaluacija table

*Opis:*

Funkcija vrši heurističku evaluaciju table nakon dostizanja određene dubine. U ovoj

fazi vraća 0, i ostavljena je za buduće faze.

**(negamax depth board lastMove alpha beta maxPlayer numMoves)**

*Ulaz:*

Depth – Trenutna dubina algoritma

Board – Trenutno stanje table

LastMove – Poslednji odigran potez

Alpha, Beta – Vrednosti potrebne za alfa beta odsecanje

maxPlayer – Igrač trenutno na potezu

numMoves - Broj odigranih poteza

*Izlaz:*

Najbolji potez

*Opis:*

Funkcija računa najboji mogući potez koji kompjuter može da odigra, koristeći algoritam negamax (varijanta minmaksa) i alfa beta odsecanje. Funkcija proverava da li je došlo do pobede korišćenjem funkcije *checkWin*. Pri dostizanju dubine depth, funkcija poziva funkciju *evaluate.* Ovaj algoritam se trenutno koristi u igri.

**(alpha-beta depth board lastMove alpha beta maxPlayer numMoves)**

*Ulaz:*

Depth – Trenutna dubina algoritma

Board – Trenutno stanje table

LastMove – Poslednji odigran potez

Alpha, Beta – Vrednosti potrebne za alfa beta odsecanje

maxPlayer – Igrač trenutno na potezu

numMoves - Broj odigranih poteza

*Izlaz:*

Najbolji potez

*Opis:*

Funkcija računa najboji mogući potez koji kompjuter može da odigra, koristeći algoritam minmaks sa alfa beta odsecanjem. Funkcija proverava da li je došlo do pobede korišćenjem funkcije *checkWin*. Pri dostizanju dubine depth, funkcija poziva funkciju *evaluate.* Ovaj algoritam se trenutno ne koristi.

# Reference:

<http://havannah.ewalds.ca/static/thesis.pdf>