*Projekat iz veštačke inteligencije*

*Tema:*

Havannah

IV Faza

*Članovi tima: Naziv tima:*

Ana Stojanović 15905 Anonymous Mink

Damjan Trifunović 15948

Milica Todorović 15937

Sadržaj

[Predstavljanje table 3](#_Toc534482951)

[Ispitivanje ciljnog stanja 3](#_Toc534482952)

[Algoritmi za igranje poteza 4](#_Toc534482953)

[Heuristička provera stanja 4](#_Toc534482954)

[Pokretanje programa 5](#_Toc534482955)

[Globalne promenljive I konstante 5](#_Toc534482956)

[Generisanje matrične reprezentacije table 6](#_Toc534482957)

[Učitavanje dimenzija table I postavljanje table 7](#_Toc534482958)

[Igranje poteza 7](#_Toc534482959)

[Odabir igrača 8](#_Toc534482960)

[Igra 9](#_Toc534482961)

[Štampanje table 10](#_Toc534482962)

[Pomoćne funkcije 10](#_Toc534482963)

[Pomoćne funkcije vezane za heuristiku 12](#_Toc534482964)

[Fukcije za grupisanje elemenata 13](#_Toc534482965)

[Algoritmi za igranje poteza 18](#_Toc534482966)

[Heuristika 20](#_Toc534482967)

[Reference: 26](#_Toc534482968)

Opis programa

# Predstavljanje table

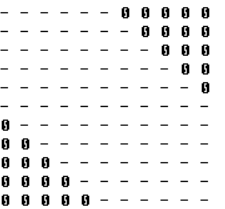
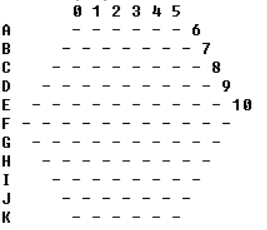
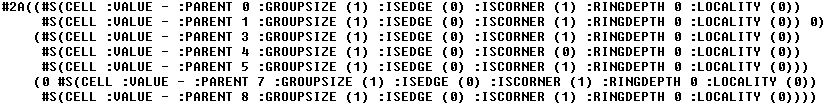
Tabla je interno predstavljena u vidu modifikovane matrice (videti sliku). Modifikacija je izvršena da bi elementima moglo da se pristupa direktno preko indeksa matrice. Polje [A,0] table se slika u polje [0,0] matrice i predstavlja validno polje. Validno polje može imati vrednost „X“, „O“ ili „-“, gde 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“. Izgled ćelije:

https://scontent-sof1-1.xx.fbcdn.net/v/t1.15752-9/49662976_758968171129357_7705642108979249152_n.png?_nc_cat=107&_nc_ht=scontent-sof1-1.xx&oh=361679b2c59fd9dc1debecb33d67804c&oe=5CC6928A

Izgled table dimenzije 2:



# Ispitivanje ciljnog stanja

Igra Havannah se može završiti nerešenim stanjem ili pobedom jednog od igrača (jedan od igrača napravi strukturu “fork”, “bridge” ili “ring”). Fork predstavlja vezu izmedju tri „edge-a“, bridge vezu izmedju dva “corner-a”.

Korišćen je algoritam “union-rank” radi efikasne provere stanja “bridge” i “fork”. Pri svakom odigranom potezu, polje koje je poslednje dodato se grupiše sa ostalim poljima istog igrača, tako da se pamti veličina grupe, “roditelj” grupe i broj “corner-a” i “edge-va” u grupi. Zbog ovakve strukture grupe se lako može utvrditi da li neka grupa sadrži dva “corner-a” ili tri “edge-a”.

Kada broj poteza bude veći od 11, proverava se da li je napravljena struktura tipa ring. Za proveru “ring-a” se koristi niz \*neighbours\*, pomoću koga se mogu izračunati indeksi komšija jednog polja. Pretražuju se komšije poslednjeg zadatog polja se u četri smera. Pretraga se završava ako se ponovo detektuje isto polje.

# Algoritmi za igranje poteza

Algoritam koji se koristi za igranje poteza je negamax. Algoritam ne kopira tablu svaki put, već stavlja potez na tablu i kasnije ga sklanja sa table. Da bi bilo moguće skidanje poteza, algoritam interno koristi drugačiju verziju algoritma koji grupiše polja (“union-find”). Sve strukture u okviru ćelije koje se menjaju pri dodavanju novog poteza su implementirane kao liste. Nova vrednost se dodaje na vrh liste, a zatim sklanja sa liste.

# Heuristička provera stanja

U programu je implementirana jedna heuristika koja koristi mašinu zaključivanja i jedna bez nje. Zbog zahteva zadatka, u finalnoj verziji programa se koristi heuristika koja koristi mašinu zaključivanja.

Prva heuristika vrši evaluaciju stanja table na osnovu sledećih vrednosti:

* Broj edge-va i corner-a u grupi u kojoj je odigran poslednji potez – Skor se dobija množenjem empirijske konstante zbirom edge –va i corner-a u grupi.
* Veličina grupe kojoj je dodat poslednji potez
* Lokalnost – Blizina odigranog poteza poljima iste boje. Pri svakom odigranom potezu se setuje “lokalni” skor komšija poteza. Neposrednim komšijama se dodeljuje lokalni skor 3, komšijama sa kojima polje ima vezu preko dva polja se daje vrednost 2, a komšijama sa kojima polje ima vezu preko jednog polja se daje lokalni skor 1. Lokalni skor se računa na osnovu svih okolnih polja ali maksimalno može biti 3.

Vrednost poteza se računa kao zbir gorenavedenih vrednosti.

Druga heuristika koristi mašinu zaključivanja. Dinamički se generiše skup činjenica koji predstavlja koji igrač okupira koje polje, kao i to koji je potez poslednji odigran.

Koristi se sledeće stavke u pravilima zaključivanja:

* Broj “corner-a” koje okupira trenutni igrač
* Broj “corner-a” koje okupira protivnik
* Broj “edge-va” koje okupira trenutni igrač
* Broj “edge-va” koje okupira protivnik
* Lokalni skor polja na kom je odigran poslednji potez
* Da li poslednji odigran potez pripada grupi ili ne

Vrednost table se računa kao težinski zbir i razlika ovih vrednosti.

Dokumentacija

# Pokretanje programa

**Fajl: main.cl**

Program se pokreće učitavanjem svih potrebnih fajlova, i zatim pozivanjem funkcije (havannah).

# 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, koju unosi korisnik

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

**board** – globalna promenljiva koja čuva tablu u matričnom obliku

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

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

**edgeConnectivity** – konstanta koja se koristi pri računanju heurističke vrednosti table

**currentPlayer** – globalna promenljiva koja čuva igrača koji je trenutno na potezu

**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 18 elemenata koji služi za računanje indeksa suseda jednog polja (prvi i drugi krug suseda) i sadrži vrednost koja se koristi pri računanju heurističke vrednosti table

**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. Polje locality je dodato zbog računanja heurističke vrednosti table.

**move** – struktura koja čuva jedan odiran potez i njegovu vrednost

**T1-RULES** – pravila zaključivanja, dodata zbog računanja heurističke vrednosti table pomoću mašine zaključivanja

**T1-FACTS** – činjenice, dodate zbog računanja heurističke vrednosti table pomoću mašine zaključivanja

# Generisanje matrične reprezentacije table

**File: game.cl**

(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:* 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.cl**

(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.cl**

(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 računar i inkrementira globalnu promenjlivu \*numMoves\*. U slučaju da je došlo do krajnjeg stanja, postavlja globalnu promenljivu \*gameState\*.

# Odabir igrača

**File: game.cl**

(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.cl**

(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.cl**

(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.cl**

(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* )

(findNeighboursNew 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*.

# Pomoćne funkcije vezane za heuristiku

**File: helpers.cl**

(validateNeighbourSecond row col indexNb board)

*Ulaz:*

row – index vrste

col – index kolone

indexNb – index suseda

board – tabla

*Izlaz:*

Lista u obliku (row col score)

*Opis:*

Funkcija validira indeks komšije i vraća listu koja opisuje red, kolonu i skor komšije.

(findNeighboursSecond board row col)

*Ulaz:*

board – tabla

row – index vrste

col – index kolone

*Izlaz*:

Lista svih komšija

*Opis*:

Funkcija vraća listu svih komšija jednog polja (prvi i drugi prsten).

(setLocality board row col currentPlayer)

*Ulaz:*

board – tabla

row – index vrste

col – index kolone

currentPlayer – igrač koji je trenutno na potezu

*Izlaz:*

/

*Opis:*

Funkcija postavlja lokalni skor svih komšija datog polja.

(setLocalityComputer board row col currentPlayer)

*Ulaz:*

board – tabla

row – index vrste

col – index kolone

currentPlayer – igrač koji je trenutno na potezu

*Izlaz:*

/

*Opis:*

Funkcija dodaje novi lokalni skor svih komšija datog polja na listu skorova polja.

(unsetLocalityComputer board row col currentPlayer)

*Ulaz*:

board – tabla

row – index vrste

col – index kolone

*Izlaz*:

/

*Opis*:

Funkcija skida poslednji dodati lokalni sa liste lokalnih skorova.

# Fukcije za grupisanje elemenata

**File: union-rank.cl**

(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*.

(rootComputer 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) bez path compression algoritam. Path compression se ne koristi radi vraćanja odigranog poteza.

(union-find-computer firstNodeIndex secondNodeIndex board dim)

*Ulaz*:

firstNodeIndex, secondNodeIndex – indexi elemenata koje treba spojiti

board – Trenutno stanje table

dim – Dimenzija table

*Izlaz*:

/

*Opis*:

Funkcija spaja data dva elementa u jedan skup (uniju) koristeći disjoint set algoritam. Uvek pridružuje drugi element prvom (drugi ukazuje na prvi) i algoritam ne vrši path compression.

(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:*

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

(find-parents isFirst index dim boardState)

*Ulaz:*

isFirst – Pomoćna promenljiva kojom se preskače element za koji tražimo roditelje

index – Indeks elementa za koje tražimo roditelje

dim – Dimenzija table

boardState – Trenutno stanje table

*Izlaz*:

Lista roditelja prosleđenog elementa

*Opis*:

Funkcija vraća listu svih elemenata na putu od prosleđenog elementa do root-a.

(un-union row col dim boardState)

*Ulaz*:

row, col – Indeksi reda i kolone odigranog poteza.

dim – Dimenzija table

boardState – Trenutno stanje table

*Izlaz*:

/

*Opis*:

Funkcija raskida vezu između prosleđenog elementa i unije kojoj pripada. Koristi se za vraćanje poteza.

(remove-move row col boardState dim)

*Ulaz:*

row, col – Indeksi reda i kolone odigranog poteza.

boardState – Trenutno stanje table

dim – Dimenzija table

*Izlaz:*

/

*Opis:*

Funkcija vraća odigran potez. Poziva funkciju „un-union“ kako bi se raskinule veze sa unijama koje su nastale pri odigravanju tog poteza.

Funkcije za ispitivanje ciljnog stanja - Prsten

(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.cl**

(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”.

(checkWinRetMove 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”. Za razliku od funkcije “checkWin”, ova funkcija se poziva tokom izvršavanja algoritma negamax i ne vrši se path compression.

(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.

(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.*

(negamaxRetMove 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 *checkWinRetMove*. Pri dostizanju dubine depth, funkcija poziva funkciju *INFERENCE MACHINE.* Algoritam ne generiše nove table za različita stanja koja evaluira, već se svi mogući potezi igraju na jednoj tabli, evaluiraju i tabla se vraća u pređašnje stanje „vraćanjem“ poteza. Ovaj algoritam se trenutno koristi u igri.

# Heuristika

**File: heuristika.cl**

(evaluate board lastMove numMoves player)

*Ulaz:*

board – tabla

lastMove – poslednji odigran potez

numMoves – broj odigranih poteza

player – igrač trenutno na potezu

*Izlaz:*

/

*Opis:*

Funkcija koja evaluira tablu. Poziva pojedinačne komponente evaluacije i pravi kumulativni skor.

(edgeConnectivity rootEl)

*Ulaz:*

rootEl – koren grupe za koji se traži skor

*Izlaz:*

Skor grupe koja se ispituje

*Opis:*

Funkcija računa skor na osnovu broja “edge-va” i “corner-a”, koristeći konstantu \*edgeConnectivity\*.

(groupSize rootEl)

*Ulaz*:

rootEl – koren grupe za koji se traži skor

*Izlaz*:

Veličina grupe

*Opis*:

Funkcija vraća veličinu grupe čiji je roditelj rootEl.

(localityScore board lastMove player)

Ulaz:

board– tabla na kojoj se igra

lastMove – poslednji odigran potez

player – trenutni igrač

Izlaz:

Lokalni skor poteza

Opis:

Funkcija vraća lokalni skor poteza. Lokalni skor svakog polja se računa svaki put kad se odigra potez i prikazuje blizinu odigranog poteza postojećim poljima iste boje. Neposredne komšije polja dobijaju skor 3, komšije koje imaju dve konekcije s poljem skor 2, i komšije koje imaju jednu konekciju sa poljem skor 1. Maksimalni lokalni skor je 3.

Mašina zaključivanja

**File: masina-zakljucivanja.cl**

(!eq a b)

*Ulaz:*

a, b – elementi čija se jednakost proverava

*Izlaz:*

True ili false

*Opis:*

Predikat proverava jednakost elemenata a i b.

(!ne a b)

*Ulaz:*

a, b – elementi čija se nejednakost proverava

*Izlaz:*

True ili false

*Opis:*

Predikat proverava nejednakost elemenata a i b.

(!isEdge row col)

*Ulaz:*

row – red

col - kolona

*Izlaz:*

True ili false

*Opis:*

Predikat proverava da li je element sa indeksima row i col „edge“.

(!isCorner row col)

*Ulaz:*

row – red

col - kolona

*Izlaz:*

True ili false

*Opis:*

Predikat proverava da li je element sa indeksima row i col „corner“.

(!locality1 row col board)

*Ulaz:*

row – red

col – kolona

board – trenutno stanje table

*Izlaz:*

True ili false

*Opis:*

Predikat proverava da li element sa datim indeksima ima lokalni skor 1.

(!locality2 row col board)

*Ulaz:*

row – red

col – kolona

board – trenutno stanje table

*Izlaz:*

True ili false

*Opis:*

Predikat proverava da li element sa datim indeksima ima lokalni skor 2.

(!locality3 row col board)

*Ulaz:*

row – red

col – kolona

board – trenutno stanje table

*Izlaz:*

True ili false

*Opis:*

Predikat proverava da li element sa datim indeksima ima lokalni skor 3.

(!edgeConnectivity1 row col board)

*Ulaz:*

row – red

col – kolona

board – trenutno stanje table

*Izlaz:*

True ili false

*Opis:*

Predikat proverava da li grupa koja sadrži element sa indeksima row i col ima jedan “edge” ili „corner“.

(!edgeConnectivit2 row col board)

*Ulaz:*

row – red

col – kolona

board – trenutno stanje table

*Izlaz:*

True ili false

*Opis:*

Predikat proverava da li grupa koja sadrži element sa indeksima row i col ima dva “edge-a” ili „corner“.

(!edgeConnectivit3 row col board)

*Ulaz:*

row – red

col – kolona

board – trenutno stanje table

*Izlaz:*

True ili false

*Opis:*

Predikat proverava da li grupa koja sadrži element sa indeksima row i col ima tri “edge-a” ili „corner“.

(!groupSize row col board)

*Ulaz:*

row – red

col – kolona

board – trenutno stanje table

*Izlaz:*

True ili false

*Opis:*

Predikat proverava da li element sa indeksima row i col pripada grupi većoj od jedan.

(get-cell-state i j player board)

*Ulaz:*

i – red

j – kolona

player – igrač trenutno na potezu

tabla - trenutno stanje table

*Izlaz:*

Jedna činjenica

*Opis:*

Funkcija generiše jednu činjenicu u obliku (move i j) ako polje pripada trenutnom igraču ili (move-opponent i j) ako polje pripada protivniku.

(get-states board index matrixDim states cp)

*Ulaz:*

board – trenutno stanje table

index – indeks trenutnog elementa koji se obradjuje

matrixDim – dimenzija matrice

states – lista činjenica koja se generiše

cp – igrač koji je trenutno na potezu

*Izlaz:*

Lista činjenica

*Opis:*

Funkcija generiše listu činjenica koje se odnose na to koji igrač okupira koje polje na tabli.

(generate-facts board matrixDim lastMove)

*Ulaz:*

board – trenutno stanje table

matrixDim – dimenzija matrice

lastMove – poslednji odigran potez

*Izlaz:*

/

*Opis:*

Funkcija postavlja listu činjenica \*T1-FACTS\* pozivanjem funkcije get-states i dodaje činjenicu koja se odnosi na poslednji odigran potez.

(set-rules)

*Ulaz:*

/

*Izlaz:*

/

*Opis:*

Funkcija postavlja listu pravila \*T1-RULES\*.

(get-locality-score row col board)

*Ulaz:*

row – red

col – kolona

board – trenutno stanje table

*Izlaz:*

Lokalni skor

*Opis:*

Funkcija vraća lokalni skor elementa sa indeksima row i col.

(heuristika-inference-engine board matrixDim lastMove)

*Ulaz:*

/

*Izlaz:*

board – trenutno stanje table

matrixDim – dimenzija matrice

lastMove – poslednji odigran potez

*Opis:*

Funkcija koja evaluira trenutno stanje table korišćenjem mašine zaključivanja. Ovo je heuristika koja se koristi u finalnoj vrednosti algoritma.

# Reference:

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