Projekat iz vestacke inteligencije

Dokumentacija-II faza

Tim:Julijan

1.Pavle Marinkovic 15705

2.Aleksa Simic 15862

3.Selena Simic 15864

**Dodate funkcije za trazenje oblika prsten:**

1.

(defun nadji\_put\_d\_prsten (l cvorovi put\_d)

//telo fje

)

* Ova fja vraca sve covorove koji su povezani iz put\_d pocevsi od cvora l .
* Parametri fje su:
  + l – pocetni cvor i formatu liste npr: ’( (1 2) )
  + cvorovi – su obradjeni cvorovi
  + put\_d – lista cvorova u kome se traze svi spojeni cvorovi

2.

(defun nadji\_presek (lista1 lista2)

// telo fje

)

* Ova fja vraca presek prve i druge liste
* Parmateri fje su:
  + lista1 – lista cvorvoa u obliku ’( (1 2) (2 1) (3 5))
  + lista2 – isto sto i lista1

3.

(defun nadji\_put\_izmedju\_i1\_i2 (g\_ivica3 d\_ivica3 put\_d )

// telo fje

)

* Ova fja vraca put izmedju clana ivice od tri elementa i to put imedju ivice1 i ivice2
* Parametri fje su:
  + g\_ivica3 – gornja ivica u prstenu koja ima tri clana ’( (cvor ivica1 ivica2))
  + d\_ivica3 – donja ivica u prstenu koja ima tri clana
* Ova fja se poziva sa parametrima koji su na suprotnim stranama, npr gornja leva i donja desna ivica...

(defun nadji\_put\_izmedju\_i2\_i1 (g\_ivica3 d\_ivica3 put\_d )

// telo fje )

* Ista kao i gornja samo nalazi put za suprotne ivice.
* Nacin funkcionisanja fja je iskomentarisan u kodu...

4.

(defun dodaj\_potomke\_prsten(cvor obradjeni put\_d)

//telo fje

)

* Fja praca potomke za cvor koji se nalaze u listi put\_d i koji se ne nalaze u listi obradjeni
* Parametri:
  + cvor – za koji se traze potomci – oblik (1 2) – indexi gde je odigran potez
  + obradjeni – lista obradjenih cvorova
  + put\_d – lista u kojoj se pronalaze potomci

5.

(defun izbaci\_nil (lista) // telo fje)

* Vraca listu koja ne sadriz nijedan nil element
* Parmatri:
  + lista – iz koje se izbacuju elementi koji us nil

6.

(defun npl8 (c1 c2 c3 c4 c5 c6 c7 c8) ;; pravi listu od 8 cvora )

7.

(defun proveri\_prsten\_cvorovi (cvorovi) )

* Vraca T ako cvorovi cine oblik prsten ili u suprotnom NIL
* Parametri:
  + cvorovi – lista cvorova koji su ustvari indexi odigranih poteza npr. ’( (1 2) (2 3) (3 4) ... )

8.

(defun proveri\_prsten\_za\_sve\_gd\_dl\_ivice (sve\_g\_ivice3 sve\_d\_ivice3 cvorovi)

* Vraca T ako barem jedan par (gornja desna i donja leva ivica od po 3 elementa) se nalaze na korektnim mestima i na pravom su rastojanju
* Parametri:
  + sve\_g\_ivice3 – sve gornje desne ivice u grupama od po 3 elementa
  + sve\_d\_ivice3 – sve donje leve ivice u grupama od po 3 elementa

(defun proveri\_prsten\_za\_sve\_gl\_dd\_ivice (sve\_g\_ivice3 sve\_d\_ivice3 cvorovi)

* Ista fja koa gore samo za zuprotne ivice

9.

(defun proveri\_prsten\_za\_dd\_ivice (gl\_ivica3 sve\_dd\_ivice3 cvorovi)

* Vraca T ako nadje barem jednu donju desnu ivicu na dobrom rastojanju koja je povezana sa gornjom levom ivicom preko cvorova
* Parmatri:
  + gl\_ivica3 – jedna graupa ivica ( cvor ivica1 ivica2)
  + sve\_dd\_ivica3 – sve donje desne grupe ivica
  + cvorovi – u kojima se traze ove dve grupe ivica

(defun proveri\_prsten\_za\_dl\_ivice (gd\_ivica3 sve\_dl\_ivice3 cvorovi)

* Radi isto kao i gronja samo za donje leve ivice i gornje desne

Sve ostale fje koje se upotrebljavaju za proveru oblika prsten su opisane u predhodnim dokementima.   
Alfa-Beta algoritam koji se koristi u projektu je nadjen na internetu i blago modifikovan kako bi se uklopio.

**MASINA ZA ZAKLJUCIVANJE**

Masina za zakljucivanje je preduzeta sa sajta predmeta uz koju su uklopljene nasa baza pravila i cinjenica kako bi preko nje mogao da se odredi kraj igra.

1. Baza pravila

(defparameter \*JULIJAN-RULES\*

'(

     (if (NaPotezu ?igrac) and (KrajIgre ?stanje) then (Pobednik ?igrac))

)

)

1. Baza cinjenica

(defparameter \*JULIJAN-FACTS\*

'(

     (KrajIgre 'Viljuska)

     (KrajIgre 'Most)

     (KrajIgre 'Prsten)

(NaPotezu 'X)

(NaPotezu 'O)

(Pobednik 'X)

(Pobednik 'O)

)

)

1. Poziv interfejsne funkcije prepare-knowlaedge

(prepare-knowledge \*JULIJAN-RULES\* \*JULIJAN-FACTS\* 10)

1. Koriscenje iterfejsne funkcije infer za zadate upite sa predefinisanim predikatima

(defun proveri\_infere()

(or

(infer '(AND (Pobednik 'X) (!viljuska)))

(infer '(AND (Pobednik 'X) (!most) ))

(infer '(AND (Pobednik 'X) (!prsten) ))

(infer '(AND (Pobednik 'O) (!viljuska)))

(infer '(AND (Pobednik 'O)(!most) ))

(infer '(AND (Pobednik 'O) (!prsten) ))

)

)

1. Predefinsiani predikati

(defun !viljuska()

fork

)

(defun !most()

most

)

(defun !prsten()

prsten

)

* Vracaju globalni bool projekta koje se postavlja na T kada je neki od obilka napravljen u toku igre sto ce da dovede do kraja igre.

**HEURISTIKA**

1.

(defun heuristika(stanje simbol))

* Vraca heuristiku stanja za odredjeni simbol
* Paramteri:
  + stanje – stanje tabele
  + simbol – igraca za kog proveravamo (’X ili ’O)

heuristika poziva u sebi fju procenji-stanje

(defun proceni-stanje (stanje pomstanje))

* Vraca procenu trenutnog stanja
* Paramteri
  + stanje – stanje za koje se proverava
  + pomstanje – takodje, kada se zove fja , isto stanje

2.

(defun proceni-stanje-vrsta(rbrvrste vrsta pomstanje))

* Vraca procenu vrste
* Paramteri:
  + rbrvrste – redi broj vrste
  + vrsta – lista
  + pomstanje – pomocno stanje

Ovo je fja gde se trenutno nalazi logika kopjutera.   
Mesto na kome se ne nalazi nista ili je nevalidno polje u stanju ima vrednost 0.  
Ako prilikom procene stanja fja naidje na polje gde nije nista odrigrano ili gde bi bio nevalidan potez u stanju, njega preskace bez dodavanja vrednosti heuristici. Ako protivnik proba da napravi most kompjuter ce da ga blokira... Ako nema pretnje da se napravi most, kompjuter ce da daje prednost potezima koji ce da se redjaju jedan do drugog, ondnosno potezu koji ce da ima najvise potomaka oko sebe.