Kodutöö

Ülesande kirjeldus

Koostada Prologis kabeprogramm, mis sooritab korraga ühe käigu või võtmise(d). Programm peab võistlema vastase programmiga. Predikaat `turniir/0` moodulis `arbiiter` kutsub korda-mööda välja mängijate programmide peapredikaadid. Mäng lõpeb, kui ühel mängijatest ei ole enam võimalik teha käike. Võitja on programm, mis sooritas viimase käigu. Kui 15 käigu korral ei toimu löömist, siis on tulemuseks viik. Arbiiter kontrollib käikude õigsust ja diskvalifitseerib sohki teinud programmi.

Kõik programmid peavad järgima järgmisi kokkuleppeid:

1. Kabelaua seis esitada faktidega ruut/3:

```
ruut(X,Y, Status).     % kus  X,Y ∈ [1,8],
      Status = 0       % tühi
      Status = 1       % valge
      Status = 2       % must
      Status = 10      % valge tamm
      Status = 20      % must tamm
```

NB! Valged alustavad väiksemate X-koordinaadi väärtusega ruutudest, mustad – suuremate X-koordinaadi väärtusega ruutudest st. valge nupu jaoks leidub algseisus fakt: `ruut(1,1,1).`

2. Mänija programmi vormistamise reeglid:

2.1 Käiku sooritav programm peab olema vormistatud mooduli kujul
Näide:

```
:- module(mooduli_nimi, peapredikaat/3).
```

2.2 Mooduli peapredikaat peab olema kujul

```
üliõpilaskood(Color,X,Y).
```

kus

- `üliõpilaskood` - peapredikaadi funktor
- `Color` – tähistab nuppude värvi, millega antud programm mängib.
- `X=Y=0` , kui mängijal ei ole kohustust käia kindla kabendiga ja `X,Y` omavad väärtusi 1-8, kui eelmisel käigul see kabend lõi (st kui sama kabendiga saab jätkata löömist, siis tuleb seda teha järgmisel käigul).

2.3 Moodulis ei tohi esineda staatilisi fakte ruut/3 ja deklaratsiooni

```
:- dynamic ruut/3.
```

2.4 Mängija programmist ei tohi väljuda *fail*-ga

2.5 Tagurdamine ei tohi jõuda teise mängija programmi. Seetõttu on soovitav kasutada mängija peapredikaadi järgmist struktuuri:

```
peapredikaat(Color,X,Y):-
...., !.
peapredikaat(_,_,_).
```

3 Mängu ettevalmistamine ja käivitamine:

3.1. Laadida esmalt Prologi töömällu programm `arbiter.pl` ja seejärel mängijate programmide failid.

3.2. Lisada käsurealt faktibaasi faktid:

```
assert(mustad(peapredikaat1)).
assert(valged(peapredikaat2)).
```

kus `predikaat1` ja `predikaat2` on vastavalt mustade ja valgetega mängivate programmide peapredikaatide nimed.
Uue mängu alustamisel ilma vahepeal Prologist väljumata tuleb vanad faktid eelnevalt kustutada - päring:

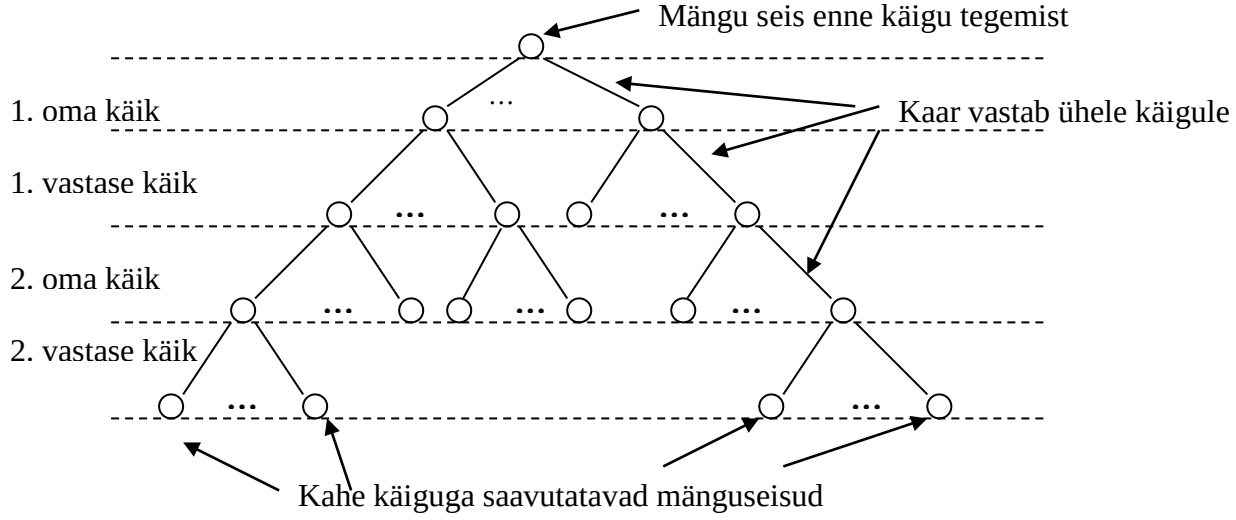
`abolish(mustad/1), abolish(valged/1).`

Soovitav on siiski iga mängu lõppedes ja uut mängu alustades Prolog sulgeda ja seejärel programmid uuesti laadida, see toimib prahikoristusena.

3.3. Kutsuda välja mängu käivitav predikaat `turniir/0` (päring: ?- `turniir.`)

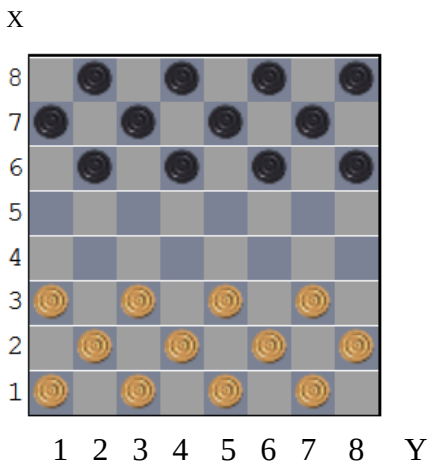
Abistavaid näpunäiteid käikude planeerimiseks

Käikude planeerimiseks on otstarbekas genereerida saavutatavate mänguseisude puu. Igale mänguseisule vastab puu üks tipp ja igale käigule kaar tippude vahel. Puu sügavus on määratud sellega kui mitu sammu antud käiku ette planeeritakse.



Joonis 1. Käikude planeerimispuu

Valged	Tühjad	Mustad
ruut(1,1,1).	ruut(4,2,0).	ruut(6,2,2).
ruut(1,3,1).	ruut(4,4,0).	ruut(6,4,2).
ruut(1,5,1).	ruut(4,6,0).	ruut(6,6,2).
ruut(1,7,1).	ruut(4,8,0).	ruut(6,8,2).
ruut(2,2,1).	ruut(5,1,0).	ruut(7,1,2).
ruut(2,4,1).	ruut(5,3,0).	ruut(7,3,2).
ruut(2,6,1).	ruut(5,5,0).	ruut(7,5,2).
ruut(2,8,1).	ruut(5,7,0).	ruut(7,7,2).
ruut(3,1,1).		ruut(8,2,2).
ruut(3,3,1).		ruut(8,4,2).
ruut(3,5,1).		ruut(8,6,2).
ruut(3,7,1).		ruut(8,8,2).



Joonis 2 Kabendite algseis faktidena ruut(X,Y,Color)

Käigu sooritamiseks vajalikud planeerimistegevused:

1. Planeerimiseks vajalike faktide loomine, näiteks musti ruute iseloomustav abifakt `ruut/7` omab järgmist vormingut:

```
ruut(X,Y,Color,Plan_step,Prev_state, Present_state,Cost).
```

kus

`X,Y` -- nupu koordinaadid [1,..,8]
`Color` -- ruudul oleva nupu värv [1,2]
`Plan_step` -- planeerimispuu korrus, millele antud `ruut/7` fakt vastab [0,..,n]
`Prev_state` -- eelmise seisu ID, millest tekib jooksev seis
`Present_state` -- käiguga tekkiva seisu ID
`Cost` -- käigu tulemusena tekkiva seisu hind (vastase nupu võtmisel: `Cost:=Cost+1`, oma nupu kaotamisel: `Cost:=Cost-1`)

2. Planeerimispuu genereerimine:
   a. Planeerimispuu koosneb `ruut/7` faktidest, mille parameeter `Prev_state` on viit antud seisu vahetu esivanemseisule ja võimaldab puud läbida terminal-tipust juur-tipu suunas.

3. Parima käigu valimine:
   a. Kasutades fakti ruut/7 parameetri `Cost` väärtusi, leida planeerimispuu terminaalsetele tippudele vastavate (, kus parameeter `Plan_step =` max planeerimissügavus, näiteks `Plan_step = 2`) faktide `ruut/7` hulgast niisugune, mille parameeter `Cost` omab suurimat väärtust.
   b. Kasutades fakti ruut/7 parameeterit `Prev_state` liikuda planeerimispuu juurtipuni ja kuulutada sellele teele jääva esimese käigu tulemus mängu uueks seisuks.
4. Kopeerida valitud käiguga tekkiv uus seis vastasele nähtavaks faktide hulgaks `ruut/3` ja anda juhtimine tagasi arbiiterprogrammile.

Tammi programmeerimine:

Antud tammi positsioon (X_0, Y_0)

- Samm1: 2 diagonaali määramine võrranditega:
  - $X = Y + X_0 - Y_0$ (Rel*)
  - $X = -Y + (X_0 + Y_0)$ (Rel**)

  mis lõikuvad positsioonis (X_0, Y_0).
- Samm 2: FORALL Rel $\in$ {Rel*, Rel**} Tamm saab võtta, kui diagonaalil, mis rahuldab võrrandit Rel leidub vastase nupuga ruut (X_v, Y_v), st.

  $$\exists P(X_v, Y_v) \models Rel \wedge \mathrm{COLOR}(X_v, Y_v) =/= \mathrm{COLOR}(X_0, Y_0) =/= 0$$

  - tammi ja vastase nupu vahel on kõik ruudud tühjad st.
    $\forall P(X_i, Y_i) \models Rel$: $\mathrm{sign}(X_v-X_0)=\mathrm{sign}(X_i-X_0) \wedge d(P_0,P_i) < d(P_0,P_v) \Rightarrow \mathrm{COLOR}(X_i, Y_i) = 0$
  - sellel diagonaalil vahetult peale vastase nuppu leiduvate vabade ruutude hulk on mittetühi > 0.

    $$T = \{(X_t, Y_t) \models Rel, 0 < X_t, Y_t < 9 \wedge \mathrm{COLOR}(X_t, Y_t) = 0 \wedge \mathrm{sign}(X_v-X_0) = \mathrm{sign}(X_t-X_0) \wedge \exists!(X_t, Y_t) \models Rel$$
- Samm 3: Võtmisel tekib igast vabast järelruudust uus planeerimisharu
- Samm 4: Kas Sammud 1 – 3 andsid püsipunkti?
  - Kui “jah”, siis return;
  - Kui “ei”, siis goto Samm 1

Kabemängu lühireeglid vene kabe jaoks

Kabelaud ja selle asetus. Vene kabet mängitakse ruudukujulisel laual, mis on jagatud 64-ks ühesuguseks ruuduks (8×8). Ruudud on vaheldumisi tumedad ja heledad, laud asetatakse mängijate vahele nii, et peatee algab kummagi mängija vasakult käelt, seega vasem alumine ruut on must. Mängitakse mustadel ruutudel.

Käigud. Kivid (nn tavalised nupud) ja kabed (nn tammid) liiguvad mööda musti väljasid/diagonaale. Kivid, mis on jõudnud viimasele horisontaalile, muutuvad kabeks. Esimese käigu teeb alati valgetega mängija. Mängijad sooritavad käike oma kabenditega kordamööda. Kiviga saab käiku sooritada ainult edasisuunas (mööda diagonaali) järgmise rea tühjale väljale. Kabe võib liikuda edasi või tagasi mööda järjestikulisi vabu välju diagonaalil, millel ta asub.

Löömine. Vastase kabendeid saab lüüa nii edasi- kui ka tagasisuunas (ei saa lüüa iseenda kabendeid). Löömine on kohustuslik (sundlöömine).

Löömine kiviga. Kui kivi asub vastase kabendiga sama diagonaali kõrvalväljal ja vastase kabendi taga on tühi väli, siis peab ta hüppama vabale väljale üle vastase kabendi, mis seejärel eemaldatakse laualt. Sellist operatsiooni sooritamist nimetatakse löömiseks kiviga.

Löömine kabega. Kabega löömisel võib vastase löödav kabend asuda samal diagonaalil (kas vahetult vastas või sellest eemal) ja selle kabendi taga on üks või mitu vaba välja. Lööv kabe peab hüppama üle vastase kabendi sobivale vabale väljale. Samal käigul tuleb löömist jätkata kuni selleks võimalust on.

Võit. Võitjaks loetakse mängija, kes saavutab esimesena seisu, milles tema vastane ei suuda teha järjekordset käiku: selle tõttu, et kõik tema kabendid on löödud (kabelaualt kõrvaldatud) või on säilinud kivid liikumisvõimetud so kinnistatud.

Viik. Partii lõpeb viigiga, kui kumbki pool ei suuda võita.

Punktid. Turniiridel antakse võidu eest 2 punkti, viigi eest 1 punkt, kaotuse eest 0-punkti.

Ajakontroll. Kabeprogrammi käigule antakse maksimaalselt 10 sekundit. (NB! See ei ole võistluskabe reegel!)

Tähtsamad reeglid kabevõistlustel

- Vastase kabendite löömine nii edasi- kui ka tagasisuunas on kohustuslik.
- Löömisel on lubatud vastase kabendist üle hüpata vaid üks kord, mille järel tagastab kabeprogramm juhtimise arbiitrile. Arbiiter väljastab vahetulemuse ja annab käiguõiguse tagasi löönud programmile koos löönud kabendi X,Y koordinaadiga.
- Kui löömise käigus satub lööki teostav kivi viimasele reale, muutub ta kohe kabeks ja jätkab löömist kabena.
- Partii loetakse lõppenuks viigiga, kui
    - mängijad 15 käigu jooksul tegid käike ega teinud ühtegi löömiskäiku.

NB! Tegu on lühendatud ja lihtsustatud reeglitega, mis kehtivad ainult loogilise programmeerimise kursuse kabeprogrammide kohta.

Allikad: Urmo Ilvese kaberaamat (64), Kabekoodeks, FMJD Sektsioon 64 vene kabe reeglid