UDK: 007.52:51
Znanstveni rad

MEHANIČKO RJEŠENJE JEDNE ARITMETIČKO - LOGIČKE GLAVOLOMKE

MECHANICAL SOLUTION OF THE RUNWAY PROBLEM

Mr. Mirko Čubrilo
Fakultet organizacije i informatike, Varaždin

U članku je opisana jedna aritmetičko-logička glavolomka, tzv. problem piste. Zatim je dano mehaničko rješenje glavolomke u jeziku logičkog programiranja PDC Prolog. To znači da je problem adekvatno formaliziran sredstvima jezika, a njegovo rješenje reprezentirano kao ciljni predikat pripadnog Prolog programa. Sam proces izračunavanja rješenja prepušten je deduktivnom mehanizmu PDC Prologa. U okviru formalne reprezentacije problema razvijen je niz predikata. Logika programa realizirana je sistemskim predikatima cut (rez) i fail (neuspjeh). Rezultati rada programa dani su u priloženoj .LOG datoteci. Iz podataka koje ona sadrži moguće je rekonstruirati proces rješavanja problema. Tekst datoteke je komentiran da bi se olakšalo njegovo razumijevanje.

Ključne riječi: *Glavolomka, reducirani problem piste, opći problem piste, mehaničko rješenje, Prolog*

In this article a nontrivial example of logico-arithmetical puzzle, so called runway problem is set and solved. The obtained solution is a mechanical one, in the sense that the problem is adequately formalized by means of the language (PDC Prolog), while the solution of the problem is acomplished by running Prolog's deductive mechanism. As a tools for controlling the logic of the program we have used the PDC Prolog's primitive predicates, namely the 'cut' and the 'fail' predicates.

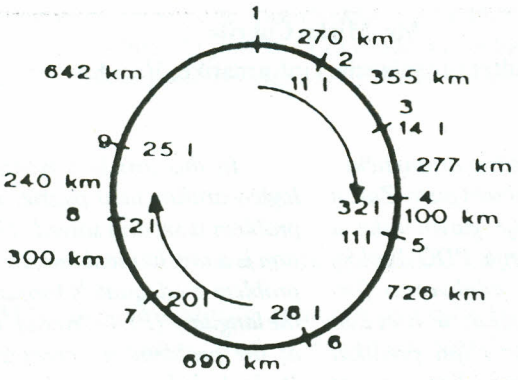
Keywords: *Runway problem, mechanical solution, Prolog*

1. FORMULACIJA PROBLEMA PISTE I OPĆE NAPOMENE

U knjizi [Bizam & Herczeg, 1972.] (problem br. 79) i članku [Čubrilo,1991.] opisana je slijedeća aritmetičko-logička glavolomka, na koju se u daljnjem tekstu pozivam kao na problem piste :

Na kružnoj automobilskoj pisti postavljene su benzinske stanice (pumpe) A _ I, tako da je u svakoj od njih pospremljena određena količina benzina. Nadalje, poznata je udaljenost između svake dvije susjedne stanice. Situacija je opisana na priloženom crtežu. Automobil troši 4 l benzina na svakih 100 km prevaljenog puta.

Problem je u slijedećem: odrediti bar jednu (ako postoji) benzinsku stanicu uz uvjet da polazeći iz nje automobil obiđe čitavu pistu i vrati se na polaznu točku. Pretpostavka je da se rezervoar za gorivo puni tek na polaznoj stanici i da je prije toga bio prazan.



Lako je uočiti da je ukupna količina benzina u svim stanicama jednaka 144 l, a da je duljina piste 3600 km. Uz zadanu potrošnju od 4 l benzina na 100 km prevaljenog puta to je upravo nužna i dovoljna količina benzina za obilazak čitave piste. Naravno, to je tek nužan uvjet za pozitivno rješenje problema. Da on bude i dovoljan, mora raspodjela benzina po stanicama biti takva da bar iz jedne stanice omogući obilazak čitave piste.

Bizam i Herczeg rješavaju ovaj problem metodom pokušaja i pogreške. Metoda se sastoji u nasumičnom odabiru polazne stanice i uzastopnoj provjeri mogućnosti prelaska u slijedeću (susjednu) stanicu. Tom metodom uzastopno se kao polazne eliminiraju stanice 1 - 5. Za stanicu br. 6 pokazuje se da polazeći iz nje auto može uzastopno prelaziti u susjednu stanicu i tako ponovo dostići istu stanicu. Uvjet za to je da rezerva benzina u svakoj stanici ne bude manja (po mjernom broju) od jedne dvadesetpetine udaljenosti između dotične i njoj susjedne stanice.

Riješiti problem u kontekstu programske sredine Prologa znači adekvatno opisati njegovu semantičku strukturu, a samo "izračunavanje" rješenja prepustiti deduktivnom mehanizmu Prologa. Zbog toga se govori o mehaničkom rješavanju problema.

U okviru tehnika za rješavanje problema važno mjesto zauzimaju tehnike verifikacije programa. Od tehnika verifikacije programa bit će korištena u PDC Prolog ugrađena tehnika trasiranja programa u njenoj reduciranoj varijanti (shorttrace directive).

2. OPIS I RJEŠENJE PROBLEMA PISTE U PDC PROLOGU

2.1. Reducirani problem piste

Problem piste bit će rješavan postupno. Za početak usvajam pretpostavku da je auto na početku dospio u stanicu iz koje može obići pistu. To je stanica br. 6. Predikati kojima se u PDC Prologu opisuje ova varijanta problema jesu:

1) **susjedne**(stanica, stanica,)

Domenu ovog predikata predstavljaju uređeni parovi susjednih stanica. Budući da je na pisti zadan smjer gibanja kazaljke na satu, predikat smatramo nesimetričnim.

2) **udaljenost**(stanica, stanica, udaljenost)

Ovim predikatom utvrđuje se udaljenost između svakog para susjednih stanica. Svojstvo simetričnosti, koje se obično pridaje predikatima tog tipa, ovdje se ne koristi, pa i ovaj predikat smatramo nesimetričnim s obzirom na prva dva argumenta.

3) **zalihaBenzina**(stanica, zaliha)

Ovaj predikat je definiran tako da zalihaBenzina(Stanica, Zaliha) bude u svakoj interpretaciji istinito ako, i samo ako je Zaliha početna zaliha benzina u stanici Stanica.

Navedeni predikati zadani su u samoj formulaciji problema piste. Predikati koji slijede su "korisnički definirani". Oni ocrtavaju strategiju rješavanja problema.

4) **startna**(stanica)

Samo ime predikata govori da se njime određena stanica proglašava startnom. Samo u tom slučaju je rezerva benzina nakon dolaska u stanicu jednaka početnoj zalihi benzina u toj stanici. Ovaj predikat bit će implictno definiran pomoću predikata susjedne.

5) **rezervaBenzina**(stanica, rezerva)

Ako stanica nije startna, onda će rezerva benzina nakon dolaska u stanicu biti jednaka zalihi benzina u toj stanici, uvećanoj za rezervu benzina u stanici koja je susjedna toj stanici, umanjenoj za količinu benzina potrebnu da se prevali put između tih stanica.

6) **autoObilaziPistuIzStanice**(stanica)

Ovo je predikat gornjeg nivoa. Njegovo ispunjenje (uspjeh) predstavlja rješenje problema.

Preostale predikate iz programa komentirat ću u samom tekstu programa. Delimiter komentara koji mogu stati u jedan redak teksta može biti znak za postotak (%), dok za komentare proizvoljne duljine kao delimiter za početak služi riječ /*, a za kraj riječ */.

U izvornim komentarima su tokom slaganja teksta na potrebnim mjestima izvršene intervencije umetanjem znakova hrvatskog jezika kojih nema u standardnom ASCII setu znakova.

```
/*********************************************************************/
/*   PISTA_1.PRO - program koji rjesava reducirani problem piste     */
/*********************************************************************/
domains
        stanica, udaljenost, zaliha = integer
        rezerva = real
database
        startna(stanica)
        susjedne(stanica,stanica)
predicates
        autoObilaziPistuIzStanice(stanica)
        autoPutuje
        autoPutujeDalje(rezerva, udaljenost)
        rezervaBenzina(stanica, rezerva)
        udaljenost(stanica,stanica, udaljenost)
        zalihaBenzina(stanica, zaliha)
        poruka(stanica, stanica)
clauses
autoObilaziPistuIzStanice(S):-
                susjedne(S,_),                          % (1)
                !,                                      % (2)
                asserta(startna(S)),                    % (3)
                autoPutuje.                             % (4)
/*
```

Da bi auto obišao pistu iz stanice S, on mora odabrati startnu stanicu i nakon toga mora uzastopno prelaziti u susjednu stanicu, sve dok ponovo ne dostigne startnu stanicu. Deduktivni mehanizam Prologa odabrat će prvi par susjednih stanica i lijevu će proglasiti za startnu (klauzule 1 i 3). Već je rečeno da ovaj program rješava problem pod pretpostavkom da je početno odabrana startna stanica ujedno i rješenje problema. Kako je poznato da je to stanica br. 6, među klauzulama iz opisa predikata **susjedne**, prva je susjedne(6,7). Predikat **startna** nastupa kao predikat u sekciji **database**, čime se osigurava mogućnost manipuliranja podacima tog tipa tokom realizacije programa. Predikat ! (rez) služi kao glavno sredstvo za kontrolu grananja u programu.

```
*/
autoPutuje:-susjedne(_,S1),                             % (5)
        startna(S2),
        poruka(S2,S1).
/*
```

Nakon što je odabrana startna stanica i utvrđen trenutni par susjednih stanica (u predikatu **autoPutuje**), program se grana u predikatu poruka.

```
*/
poruka(S2,S1):-S2S1,                                    % (6)
        !,
        susjedne(S,S1),
        rezervaBenzina(S,Rezerva),                      % (7)
        udaljenost(S,S1,Udaljenost),
        autoPutujeDalje(Rezerva,Udaljenost).            % (8)
poruka(S2,S1):-S2=S1,                                   % (9)
        !,
        susjedne(S,S1),
        rezervaBenzina(S,Rezerva),
        udaljenost(S,S1,Udaljenost),
        Rezerva=Udaljenost/25,
        write("Auto obilazi pistu iz stanice ",S2).
/*
```

Grananje je ovisno o tome da li je dolazna stanica (S1) različita od startne stanice (S2) ili ne. Ako jest, onda je trenutna dolazna stanica različita od startne stanice i ovisno o rezervi benzina u trenutnoj stanici i udaljenosti od nje do dolazne stanice auto putuje dalje, što se izriče istoimenim predikatom. Ako nije, onda je trenutna dolazna stanica istovjetna startnoj i vrši se neposredna provjera mogućnosti prelaska posljednje etape puta. Ako je prelazak moguć, a ovdje iz rečene pretpostavke slijedi da jest, slijedi poruka o tome da je trenutna dolazna stanica rješenje problema piste i program okončava rad.

```
*/
autoPutujeDalje(Rezerva,Udaljenost):-Rezerva=Udaljenost/25,   % (10)
                        susjedne(S,S1),                       % (11)
                        assert(susjedne(S,S1)),               % (12)
                        retract(susjedne(S,S1)),              % (13)
                        autoPutuje.
/*
```

Pretpostavka da je Rezerva =Udaljenost/25 (zbog potrošnje od 4 l na 100 km) ovdje je ispunjena, opet zbog polazne pretpostavke da je izbrana startna stanica upravo ona koja predstavlja rješenje problema. Nakon što klauzula 10 bude zadovoljena, klauzula 11 pokazuje trenutno procesirani par susjednih stranica. Klauzula 12 uvijek uspijeva. Njenim izvršenjem trenutni par susjednih stanica postavlja se na začelje baze podataka. Uspijevanjem klauzule 13 dobavlja se naredni par susjednih stanica koji dalje procesira predikat autoPutuje.

```
*/
rezervaBenzina(S,Rezerva):-retract(startna(S)),          % (14)
                zalihaBenzina(S,Zaliha),!,
```

```
                Rezerva=Zaliha,
                asserta(startna(S)).                                    % (15)
/*
```

Uspijevanjem klauzule 14 potvrđuje se da je startna stanica ujedno trenutna polazna stanica. Zbog toga je rezerva benzina u toj stanici jednaka zalihi benzina. Jer sistemski predikat (primitiv) **retract** odstranjuje dobavljenu startnu stanicu iz baze podataka, ona se mora obnoviti. Obnavljanje se ostvaruje uspijevanjem klauzule 15.

```
*/
rezervaBenzina(S,Rezerva):-
        susjedne(S1,S),
        rezervaBenzina(S1,Rezerva1),
        zalihaBenzina(S,Zaliha),
        udaljenost(S1,S,Udaljenost),
        Rezerva = Rezerva1 + Zaliha - Udaljenost/25.                    % (16)
/*
```

Kad trenutna polazna stanica nije startna, rezerva benzina u njoj prije polaska u susjednu stanicu dobije se kao zbroj zalihe benzina u toj stanici i rezerve benzina u stanici kojoj je ona dolazna, umanjen za potrošnju benzina na put između tih stanica (klauzula 16).

```
*/
zalihaBenzina(1,11).
zalihaBenzina(2,14).
zalihaBenzina(3,11).
zalihaBenzina(4,32).
zalihaBenzina(5,1).
zalihaBenzina(6,28).
zalihaBenzina(7,20).
zalihaBenzina(8,2).
zalihaBenzina(9,25).
/*  Činjenice koje opisuju predikat zalihaBenzina. */
udaljenost(1,2,270).
udaljenost(2,3,355).
udaljenost(3,4,277).
udaljenost(4,5,100).
udaljenost(5,6,726).
udaljenost(6,7,690).
udaljenost(7,8,300).
udaljenost(8,9,240).
udaljenost(9,1,642).
/*  Činjenice koje opisuju predikat udaljenost . */
susjedne(6,7).
```

```
susjedne(7,8).
susjedne(8,9).
susjedne(9,1).
susjedne(1,2).
susjedne(2,3).
susjedne(3,4).
susjedne(4,5).
susjedne(5,6).
/*   Činjenice koje opisuju predikat susjedne. */
goal
   autoObilaziPistuIzStanice(_).
```

2.2. Trasiranje programa i rješenja reduciranog problema piste

Većina implementacija Prologa ima ugrađena sredstva za trasiranje programa. U tu svrhu služi jedna ili više varijanti primitiva trace, koji se može pozvati iz razvojnog okruženja Prologa ili iz programa kao direktiva prevodiocu (compiler directive). PDC Prolog posjeduje primitive **trace** i **shorttrace** i svaki od njih se može koristiti za trasiranje bilo čitavog programa, nekog njegovog fragmenta ili samo pojedinih predikata.

Rezultat trasiranja programa je niz "poruka" slijedećih tipova:

```
CALL:<text>
RETURN:<text>
FAIL: <text>
REDO: <text>
```

Poruka tipa CALL: <text> pojavljuje se kad god se u programu poziva neki predikat (sistemski ili korisnički). Tada <text> sadrži ime predikata koji se poziva i njegove eventualne parametre. Varijable koje nisu poprimile vrijednost označavaju se crticom "_".

Poruka tipa RETURN: <text> vraća vrijednosti varijabli koje zadovoljavaju pojedine klauzule programa. Ako postoje i druge mogunosti zadovoljenja iste klauzule, to se označava zvjezdicom "*", što znači da je klauzula nedeterministička.

Poruka tipa FAIL: <text> označava neuspjeh predikata koji je prethodno bio pozvan porukom tipa CALL: <text>.

Poruka tipa REDO: <text> označava da slijedi novi pokušaj zadovoljenja predikata čiji prethodni pokušaj zadovoljenja nije uspio.

Razlika između primitiva **trace** i **shorttrace** je u tome što posljednji ne pokazuje rezultate repne rekurzije predikata koji su tako građeni. Grubo govoreći, ne pokazuje računanje nekih međurezultata.

Poruke koje se tokom interaktivnog testiranja programa šalju u posebno za tu namjenu kreiran prozor na ekranu računala moguće je preusmjeriti u posebnu ASCII

datoteku sistemskog imena PROLOG.LOG. Ona se dalje može pregledavati u klasičnim editorima teksta.

Vezano za izloženi program, dajem fragmente pripadne datoteke PROLOG.LOG uz odgovarajuće komentare

```
Execute program CALL:_PROLOG_Goal()   % Pozivanje sistemskog cilja
CALL:autoobilazipistuizstanice(_)     % programa
CALL:susjedne(_,_)                    % Traži se prvi par susjednih stanica
RETURN:Đsusjedne(6,7)
assert(startna(6))                    % Proglašava se startna stanica
CALL:autoputuje()
CALL:susjedne(_,_)
RETURN:Đsusjedne(6,7)
CALL:startna(_)
RETURN:Đstartna(6)
CALL:poruka(6,7)              % Poruka ovisi o startnoj i trenutnoj dolaznoj stanici
6><7                          % Uspijeva prva klauzula definicije
CALL:susjedne(_,7)
RETURN:Đsusjedne(6,7)
CALL:rezervabenzina(6,_)      % Računa se rezerva benzina u polaznoj stanici
retract(startna(6))           % Utvrđuje se da je to startna stanica
CALL:zalihabenzina(6,_)       % Traži se zaliha benzina u startnoj stanici i
REDO:zalihabenzina(6,_)       % redom se prolaze istovjetne činjenice za
REDO:zalihabenzina(6,_)       % prethodnih pet stanica. U daljnjem takve
REDO:zalihabenzina(6,_)       % fragmente izbacujem jer oduzimaju puno
REDO:zalihabenzina(6,_)       % prostora.
REDO:zalihabenzina(6,_)
RETURN:zalihabenzina(6,28)    % Izračunata zaliha u stanici 6
28=28
assert(startna(6))
RETURN:rezervabenzina(6,28)   % Izračunata rezerva benzina u stanici 6
CALL:udaljenost(6,7,_)
RETURN:udaljenost(6,7,690)
CALL:autoputujedalje(28,690)
28=27.6                       % rezerva je dostatna za prijelaz u susjednu stanicu
CALL:susjedne(_,_)
RETURN:Đsusjedne(6,7)
assert(susjedne(6,7))         % Slanje trenutnog para susjednih stanica na kraj baze
retract(susjedne(6,7))        % Dobavljanje novog para susjednih stanica
CALL:autoputuje()             % Početak pokušaja novog prelaska u (novu) dolaznu
CALL:susjedne(_,_)            % stanicu.
RETURN:Đsusjedne(7,8)
CALL:startna(_)
```

```
RETURN:Đstartna(6)
CALL:poruka(6,8)
6
CALL:susjedne(_,8)
RETURN:Đsusjedne(7,8)
CALL:rezervabenzina(7,_)
FAIL:rezervabenzina(7,_)
REDO:rezervabenzina(7,_)
CALL:susjedne(_,7)
RETURN:Đsusjedne(6,7)
CALL:rezervabenzina(6,_)
retract(startna(6))
CALL:zalihabenzina(6,_)
RETURN:zalihabenzina(6,28)
28=28
assert(startna(6))
RETURN:rezervabenzina(6,28)
CALL:zalihabenzina(7,_)
RETURN:zalihabenzina(7,20)
CALL:udaljenost(6,7,_)
RETURN:udaljenost(6,7,690)
20.4=20.4
RETURN:rezervabenzina(7,20.4)
CALL:udaljenost(7,8,_)
RETURN:udaljenost(7,8,300)
CALL:autoputujedalje(20.4,300)
20.4>=12
/* Pokušaj prelaska završava uspjehom. Fragment datoteke u kojoj se prethodni ciklus
ponavlja za stanice 8,9,1,2,3,4 izostavljam jer je u potpunosti analogan fragmentu izloženom
za stanicu 7. Za kraj, dajem posljednji fragment datoteke PROLOG.LOG koji se odnosi na
prijelaz iz stanice 5 u stanicu 6 */
assert(susjedne(4,5))
retract(susjedne(4,5))
CALL:autoputuje()
CALL:susjedne(_,_)
RETURN:Đsusjedne(5,6)
CALL:startna(_)
RETURN:Đstartna(6)
CALL:poruka(6,6)
6><6                    % Startna i trenutna dolazna stanica nisu različite
FAIL:poruka(6,6)        % pa prva klauzula predikata poruka ne uspijeva.
REDO:poruka(6,6)        % Pokušaj ponovnog zadovoljenja.
6=6                     % Pokušaj uspijeva.
```

```
/* U ostatku datoteke utvrđuje se mogućnost prijelaza iz stanice 5 u stanicu 6 i daje se
poruka da je stanica 6 rješenje problema. Slijedi sistemska poruka da je program uspješno
okončao s radom. I ovdje je izbačen fragment datoteke u kojem se rekurzivno računa rezer-
va benzina u stanici 5. */
CALL:susjedne(_,6)
RETURN:Đsusjedne(5,6)
CALL:rezervabenzina(5,_)
RETURN:rezervabenzina(5,29.04)
CALL:udaljenost(5,6,_)
RETURN:udaljenost(5,6,726)
29.04=29.04
write("Auto obilazi pistu iz stanice ")
Auto obilazi pistu iz stanice write(6)
6
RETURN:poruka(6,6)
RETURN:_PROLOG_Goal()
Program terminated with success
```

2.3. Dijagnostika programa

Na kraju izlaganja vezanih za reducirani problem piste dajem zapis njegovog dijagnostičkog modula. Zapis je također preusmjeren u datoteku PROLOG.LOG.

Dijagnostički modul za dani PDC Prolog program nastaje realizacijom direktive **diagnostics** Prolog prevodiocu ili izborom opcije **diagnostics/on** u radnom okruženju Prologa.

Dijagnostički modul PDC Prolog programa sadrži slijedeće podatke o njemu:

- imena svih predikata u programu
- neke karakteristike predikata (da li je predikat lokalan ili globalan, da li je interno definiran sistemom, da li je deklariran u internoj bazi podataka, da li je deterministički ili nedeterministički)
- veličinu memorijskog zapisa svakog predikata
- tipove domena svakog predikata
- tokove podataka svakog predikata.

Dijagnostički zapis koristan je iz više razloga. Na primjer, on pokazuje da neki u programu definirani predikati nisu (ako nisu) uistinu i korišteni. Nadalje, analizom tokova podataka za pojedine predikate mogu se isti ili adekvatnije definirati ili se mogu ukloniti pogreške nastale baš zbog neadekvatnih tokova podataka u programu, itd.

DIAGNOSTICS FOR MODULE: D:ĐPROLOGĐPISTA_1.PRO

Predicate Name	Type	Determ	Size	Domains -- flowpattern
_PROLOG_Goal	local	YES	118	--
startna	dbase	NO	152	stanica
susjedne	dbase	NO	163	stanica,stanica -- o,o
susjedne	dbase	NO	162	stanica,stanica -- o,i
autoobilazipistuizstanice	local	NO	198	stanica -- o
autoputuje	local	NO	160	--
autoputujedalje	local	NO	540	rezerva,udaljenost -- i,i
rezervabenzina	local	NO	992	stanica,rezerva -- i,o
udaljenost	local	YES	596	stanica,stanica,udaljenost -- i,i,o
zalihabenzina	local	YES	542	stanica,zaliha -- i,o
poruka	local	NO	583	stanica,stanica -- i,i
Total size			4206	

Size of symbol table = 445 bytes

2.4. Druga verzija programa PISTA_1.PRO

Program PISTA_1.PRO pisan je tako da maksimalno odražava deklarativni aspekt logičkog programiranja. S konceptualnog aspekta to znači povećan obim rekurzivnih izračunavanja vrijednosti iz domena predikata. Sa stanovišta realizacije programa to može značiti uzastopno izračunavanje nekad izračunatih vrijednosti. Primjer takvog predikata u programu PISTA_2.PRO je predikat **rezervaBenzina**. Da bi se na primjer izračunala rezerva benzina u stanici 5, mora se, rekurzivnim pozivanjem predikata, računati rezerva benzina za stanice 4,3,2,1,9,8,7 i 6. Tek za stanicu 6 kao polaznu stanicu ona je zadana eksplicitno kao činjenica preko predikata **zalihaBenzina**. Jasno je, međutim, da je za računanje rezerve benzina u desnoj u paru susjednih stanica dovoljno znati zalihu benzina u istoj stanici i rezervu benzina u lijevoj stanici para.

Spomenuti problem može se ponekad ublažiti modeliranjem u Prologu nekih tehnika proceduralnog programiranja. Rečeni problem bi se u svakom programu pisanom u proceduralnom stilu riješio pamćenjem već izračunatih vrijednosti rezerve benzina u pojedinim stanicama piste. Da bi se to uradilo u Prologu, potrebno je predikat **rezerva-Benzina** proglasiti za predikat interne baze podataka jer se u takvu bazu podataka tokom realizacije programa mogu unositi i iz nje uklanjati činjenice o danom predikatu.

Usporednom analizom programa PISTA_1.PRO i PISTA_2.PRO uočava se da posljednji ne sadrži eksplicitnu definiciju predikata **rezervaBenzina**. Umjesto toga, zaliha benzina u startnoj stanici proglašena je za rezervu benzina u toj stanici (klauzula 1) nakon čega je sistemskim predikatom **assert** ta činjenica dodana u internu bazu podataka. U prvoj točci definicije predikata **poruka** očitava se rezerva benzina u trenutnoj polaznoj stanici (klauzule 3 i 4) i zatim se ponovo baza podataka obnavlja tom činjenicom. U drugoj točci definicije istog predikata dovoljno je samo očitati rezervu benzina u trenutnoj polaznoj stanici (klauzula 7). Predikat **autoPutujeDalje** preuzima izračunatu rezervu benzina i provjerava mogućnost prelaska u susjednu stanicu. Nakon te provjere računa se rezerva benzina u trenutnoj dolaznoj stanici (klauzule 8,9 i 10) i proces se nastavlja dalje ponovnim pozivanjem predikata **autoPutuje**.

Uštedu koju daje program PISTA_2.PRO u odnosu na program PISTA_1.PRO najlakše je uočiti usporedbom veličina pripadnih .LOG datoteka:

```
PISTA_1.LOG   25892  byta
PISTA_2.LOG   7814   byta
Omjer je gotovo 4:1 u "korist" programa PISTA_1.PRO.
```

```
/*************************************************************************/
/* PISTA_2.PRO - nova verzija programa PISTA_1.PRO, iz kojeg je          */
/* eliminiran predikat rezervaBenzina                                    */
/*************************************************************************/
domains
  stanica,udaljenost=integer
  zaliha,rezerva=real
database
  startna(stanica)
  susjedne(stanica,stanica)
  rezervaBenzina(stanica,rezerva)
predicates
  autoObilaziPistuIzStanice(stanica)
  autoPutuje
  autoPutujeDalje(rezerva,udaljenost)
  udaljenost(stanica,stanica,udaljenost)
  zalihaBenzina(stanica,zaliha)
  poruka(stanica,stanica)
clauses
autoObilaziPistuIzStanice(S):-
  susjedne(S,_),
  !,
  asserta(startna(S)),
  zalihaBenzina(S,Zaliha),
  Rezerva=Zaliha,                              % (1)
  assert(rezervaBenzina(S,Rezerva)),           % (2)
```

```
  autoPutuje.
autoPutuje:-susjedne(_,S1),startna(S2),poruka(S2,S1).
poruka(S2,S1):-S2S1,!,
        susjedne(S,S1),
        retract(rezervaBenzina(S,Rezerva)),                    % (3)
        Rezerva1=Rezerva,                                      % (4)
        asserta(rezervaBenzina(S,Rezerva)),                    % (5)
        udaljenost(S,S1,Udaljenost),
        autoPutujeDalje(Rezerva1,Udaljenost).                  % (6)
poruka(S2,S1):-S2=S1,!,
        susjedne(S,S1),
        retract(rezervaBenzina(S,Rezerva)),                    % (7)
        udaljenost(S,S1,Udaljenost),      Rezerva=Udaljenost/25,
        write("Auto obilazi pistu iz stanice ",S2).
autoPutujeDalje(Rezerva1,Udaljenost):-Rezerva1=Udaljenost/25,
              susjedne(S,S1),
              zalihaBenzina(S1,Zaliha1),                       % (8)
              Rezerva2=Rezerva1+Zaliha1-Udaljenost/25,         % (9)
              asserta(rezervaBenzina(S1,Rezerva2)),            % (10)
              assert(susjedne(S,S1)),
              retract(susjedne(S,S1)),
              autoPutuje.
zalihaBenzina(1,11).
zalihaBenzina(2,14).
zalihaBenzina(3,11).
zalihaBenzina(4,32).
zalihaBenzina(5,1).
zalihaBenzina(6,28).
zalihaBenzina(7,20).
zalihaBenzina(8,2).
zalihaBenzina(9,25).
udaljenost(1,2,270).
udaljenost(2,3,355).
udaljenost(3,4,277).
udaljenost(4,5,100).
udaljenost(5,6,726).
udaljenost(6,7,690).
udaljenost(7,8,300).
udaljenost(8,9,240).
udaljenost(9,1,642).
susjedne(6,7).
susjedne(7,8).
```

```
susjedne(8,9).
susjedne(9,1).
susjedne(1,2).
susjedne(2,3).
susjedne(3,4).
susjedne(4,5).
susjedne(5,6).
goal
    autoObilaziPistuIzStanice(_).
```

2.5. Opće rješenje problema piste

U ovom odjeljku dajem PDC Prolog program koji rješava opći problem piste. U odnosu na program PISTA_1.PRO uvedeni su neki novi predikati, a neki postojeći su preoblikovani (prošireni). Prije komentiranja koda programa i pojedinih njegovih predikata dajem nekoliko napomena vezanih za konceptualno rješenje problema piste.

Kao prvo, za razliku od prve verzije problema, ne možemo općenito pretpostaviti da će prva startna stanica koju program odabere biti upravo ona koja rješava problem, tj. ona iz koje auto polazi i uspijeva obići čitavu pistu. To znači da će, polazeći iz neke stanice kao startne, auto dospjeti u stanicu iz koje (zbog nedovoljne rezerve benzina) neće moći prijeći u njoj susjednu stanicu. Kada se to desi, potrebno je trenutnu startnu stanicu zamijeniti za njoj susjednu i ponoviti pokušaj obilaska piste.

Ono što je rečeno u prethodnoj napomeni može se desiti više puta, sve dok auto ne izabere pravu startnu stanicu. Kako se nakon svakog uspješnog prijelaza u susjednu stanicu dobavlja novi par susjednih stanica za novi pokušaj prijelaza, startna stanica i prvi dostupni par susjednih stranica u internoj bazi podataka moraju biti usklađeni. Što to znači, detaljnije objašnjavam na primjeru. Trasiranjem programa PISTA2.PRO lako je ustanoviti da polazeći iz stanice 1 kao startne auto dospijeva u stanicu 3, ali da iz nje ne može prijeći u stanicu 4 (jer je ispunjeno: startna(1), rezervaBenzina(3,11), udaljenost(3,4,277) i rezerva-benzina-u-stanici-3 = 11 11.08 = 277/25 = potrošnja- benzina-na-put-od-stanice-3-do-stanice-4). Tokom uspješnog dijela putovanja s vrha baze parova susjednih stanica uklonjeni su parovi (1,2) i (2,3) i tim redoslijedom smješteni na kraj baze. Kako nova startna stanica mora biti stanica 2, onda na vrhu baze parova susjednih stanica mora biti par (2,3). No, par (2,3) nalazi se na dnu baze podataka. Dakle, prije novog ciklusa putovanja potrebno je ažurirati vrh baze podataka.

S obzirom na raspoložive primitive za ažuriranje internih baza podataka (aserta, assertz i retract), one po tipu strukture podataka predstavljaju DECK s ograničenim izlazom. Takvim strukturama podataka mogu se realizirati cikličke permutacije danog konačnog skupa objekata, što je u ovom slučaju i dovoljno.

Slijedi komentar programa i njegovih predikata.

```
/*****************************************************************/
/* PISTA_3.PRO -program koji rjesava opći problem piste     */
/*****************************************************************/
domains
    stanica,udaljenost,zaliha=integer
    rezerva=real
database
    startna(stanica)
    susjedne(stanica,stanica)
predicates
    autoObilaziPistuIzStanice(stanica)
    autoPutujeDalje(rezerva,udaljenost)
    rezervaBenzina(stanica,rezerva)
    udaljenost(stanica,stanica,udaljenost)
    zalihaBenzina(stanica,zaliha)
    autoPutuje
    autoObilaziPistuIzStanice1(stanica)
    obnoviSusjedne(stanica,stanica)
    obnoviSusjedne1(stanica,stanica)
    poruka(stanica,stanica)
clauses
autoObilaziPistuIzStanice(S):-
    susjedne(S,_),
    !,
    asserta(startna(S)),
    autoPutuje.
autoPutuje:-
    susjedne(S,S1),
    startna(S2),
    !,
    poruka(S2,S1).
poruka(S2,S1):-
    S2S1,
    !,
    susjedne(S,S1),
    rezervaBenzina(S,Rezerva),
    udaljenost(S,S1,Udaljenost),
    autoPutujeDalje(Rezerva,Udaljenost).
poruka(S2,S1):-
    S2=S1,
    !,
    susjedne(S,S1),
```

```
    rezervaBenzina(S,Rezerva),
    udaljenost(S,S1,Udaljenost),
    Rezerva=Udaljenost/25,
    write("Auto obilazi pistu iz stanice ",S2).
autoPutujeDalje(Rezerva,Udaljenost):-
    Rezervaaljenost/25,
    !,
    startna(S2),
    susjedne(S2,S3),
    retract(startna(S2)),
    autoObilaziPistuIzStanice1(S3).
/* Nova klauzula predikata autoPutujeDalje(rezerva,udaljenost). Nakon neuspjeha prijelaza u
susjednu stanicu, uklanja se trenutna startna stanica iz baze startnih stanica i poziva predikat
autoObilaziPistuIzStanice1(stanica). */
autoPutujeDalje(Rezerva,Udaljenost):-
    Rezerva=Udaljenost/25,
    susjedne(S,S1),
    assert(susjedne(S,S1)),
    retract(susjedne(S,S1)),
    autoPutuje.
autoObilaziPistuIzStanice1(S3):-
    obnoviSusjedne(S3,_),
    autoObilaziPistuIzStanice(S3).
/* Novi predikat. Nakon uklanjanja trenutne neuspješne startne stanice ovaj predikat poziva
predikat obnoviSusjedne, čijim uspjehom se obnavlja (permutira) baza susjednih stanica da
bi mogao započeti novi ciklus putovanja. */
obnoviSusjedne(S3,S4):-
    retract(susjedne(S4,S5)),
    asserta(susjedne(S4,S5)),
    obnoviSusjedne1(S3,S4).
/* Ovaj predikat ispituje vrh baze parova susjednih stanica, nakon čega predikat obnovi-
Susjedne1 obnavlja bazu susjednih stanica na potrebni nivo */
obnoviSusjedne1(S3,S4):-S3=S4,!.
/* Ako je startna stanica istovjetna lijevoj stanici u paru susjednih stanica na vrhu baze, onda
je obnavljanje baze parova susjednih stanica uspješno okončano */
obnoviSusjedne1(S3,S4):-
    retract(susjedne(S4,S5)),
    assertz(susjedne(S4,S5)),
    !,
    obnoviSusjedne1(S3,S5).
/* Svaki put kad uspiju prve dvije klauzule iz tijela gornje klauzule predikata obnovi-
Susjedne1, gornji par susjednih stanica u bazi seli na začelje. Zatim se rekurzivno poziva isti
predikat u primjeni na novi par susjednih stanica na vrhu baze susjednih stanica. */
```

```
rezervaBenzina(S,Rezerva):-
    retract(startna(S)),
    zalihaBenzina(S,Zaliha),
    Rezerva=Zaliha,
    asserta(startna(S)).
rezervaBenzina(S,Rezerva):-
    susjedne(S1,S),
    rezervaBenzina(S1,Rezerva1),
    zalihaBenzina(S,Zaliha),
    udaljenost(S1,S,Udaljenost),
    Rezerva=Rezerva1+Zaliha-Udaljenost/25.
zalihaBenzina(1,11).
zalihaBenzina(2,14).
zalihaBenzina(3,11).
zalihaBenzina(4,32).
zalihaBenzina(5,1).
zalihaBenzina(6,28).
zalihaBenzina(7,20).
zalihaBenzina(8,2).
zalihaBenzina(9,25).
udaljenost(1,2,270).
udaljenost(2,3,355).
udaljenost(3,4,277).
udaljenost(4,5,100).
udaljenost(5,6,726).
udaljenost(6,7,690).
udaljenost(7,8,300).
udaljenost(8,9,240).
udaljenost(9,1,642).
susjedne(1,2).
susjedne(2,3).
susjedne(3,4).
susjedne(4,5).
susjedne(5,6).
susjedne(6,7).
susjedne(7,8).
susjedne(8,9).
susjedne(9,1).
goal
autoObilaziPistuIzStanice(_).
```

Dodatna verifikacija programa provedena je trasiranjem programa za sve ciklične permutacije baze parova susjednih stanica.

2.6. Drugo rješenje općeg problema piste

Analogno programu PISTA_2.PRO koji rješava reducirani problem piste vrednovanjem predikata rezervaBenzina na uvjetno rečeno proceduralni način, program PISTA_4.PRO rješava opći problem piste. Kod programa nije potrebno komentirati.

```
/*************************************************************/
/* PISTA_4.PRO -program koji rjesava opci problem piste      */
/* s predikatom rezervaBenzina kao predikatom interne baze   */
/* podataka                                                  */
/*************************************************************/
domains
stanica,udaljenost,zaliha=integer
rezerva=real
database
startna(stanica)
susjedne(stanica,stanica)
rezervaBenzina(stanica,rezerva)
predicates
    autoObilaziPistuIzStanice(stanica)
    autoPutujeDalje(rezerva,udaljenost)
    udaljenost(stanica,stanica,udaljenost)
    zalihaBenzina(stanica,zaliha)
    autoPutuje
    autoObilaziPistuIzStanice1(stanica)
    obnoviSusjedne(stanica,stanica)
    obnoviSusjedne1(stanica,stanica)
    poruka(stanica,stanica)
clauses
autoObilaziPistuIzStanice(S):-
    susjedne(S,_),
    !,
    asserta(startna(S)),
    zalihaBenzina(S,Zaliha),
    Rezerva=Zaliha,
    assert(rezervaBenzina(S,Rezerva)),
    autoPutuje.
autoPutuje:-
    susjedne(_,S1),
    startna(S2),
    !,
    poruka(S2,S1).
poruka(S2,S1):-
    S2S1,
```

```
    !,
    susjedne(S,S1),
    retract(rezervaBenzina(S,Rezerva)),
    Rezerva1=Rezerva,
    asserta(rezervaBenzina(S,Rezerva)),
    udaljenost(S,S1,Udaljenost),
    autoPutujeDalje(Rezerva1,Udaljenost).
poruka(S2,S1):-
    S2=S1,
    !,
    susjedne(S,S1),
    retract(rezervaBenzina(S,Rezerva)),
    udaljenost(S,S1,Udaljenost),
    Rezerva=Udaljenost/25,
    write("Auto obilazi pistu iz stanice ",S2).
autoPutujeDalje(Rezerva1,Udaljenost):-
    Rezerva1aljenost/25,
    !,
    startna(S2),
    susjedne(S2,S3),
    retract(startna(S2)),
    autoObilaziPistuIzStanice1(S3).
autoPutujeDalje(Rezerva1,Udaljenost):-
    Rezerva1=Udaljenost/25,
    susjedne(S,S1),
    zalihaBenzina(S1,Zaliha1),
    Rezerva2=Rezerva1+Zaliha1-Udaljenost/25,
    asserta(rezervaBenzina(S1,Rezerva2)),
    assert(susjedne(S,S1)),
    retract(susjedne(S,S1)),
    autoPutuje.
autoObilaziPistuIzStanice1(S3):-
    obnoviSusjedne(S3,_),
    autoObilaziPistuIzStanice(S3).
obnoviSusjedne(S3,S4):-
    retract(susjedne(S4,S5)),
    asserta(susjedne(S4,S5)),
    obnoviSusjedne1(S3,S4).
obnoviSusjedne1(S3,S4):-S3=S4,!.
obnoviSusjedne1(S3,S4):-
    retract(susjedne(S4,S5)),
    assertz(susjedne(S4,S5)),
```

```
    !.
    obnoviSusjedne1(S3,S5).
zalihaBenzina(1,11).
zalihaBenzina(2,14).
zalihaBenzina(3,11).
zalihaBenzina(4,32).
zalihaBenzina(5,1).
zalihaBenzina(6,28).
zalihaBenzina(7,20).
zalihaBenzina(8,2).
zalihaBenzina(9,25).
udaljenost(1,2,270).
udaljenost(2,3,355).
udaljenost(3,4,277).
udaljenost(4,5,100).
udaljenost(5,6,726).
udaljenost(6,7,690).
udaljenost(7,8,300).
udaljenost(8,9,240).
udaljenost(9,1,642).
susjedne(1,2).
susjedne(2,3).
susjedne(3,4).
susjedne(4,5).
susjedne(5,6).
susjedne(6,7).
susjedne(7,8).
susjedne(8,9).
susjedne(8,9).
susjedne(9,1).
goal
    autoObilaziPistuIzStanice(_).
```

LITERATURA:

1. **D. Bizam, J. Herczeg**: *JATEK ES LOGIKA*, 85 FELADATBAN, ruski prijevod, MIR, Moskva, 1973.

2. **M. Čubrilo:** *Neki aspekti modeliranja ljuski ekspertnih sistema*, Zbornik Fakulteta organizacije i informatike Varaždin 1991., br. 15, str. 43 - 66.