7. Prolog vaicājumu izpilde

Nodaļas saturs

Nodaļā ievietotie attēli, programmu piemēri un tabulas

7.1. Vaicājumu izpildes pamatprincipi

Vaicājumu izpilde kaut kādā mērā atgādina teorēmu pierādīšanu matemātiskajā loģikā. Atšķirība ir tā, ka mēs nevis beigās iegūstam vienu rezultātu *true*/*fail* (vai *yes*/*no*), bet gan ir iespējami starprezultāti, kuros parādās objektu vērtības.

Vaicājumu izpilde nozīmē – mēģināt atrast vaicājumam atbilstošos faktus un noteikumus un savietot (unificēt) tos ar vaicājumu. Ja noteikumi ir rekursīvi, tad vaicājuma izpilde kļūst cikliska.

Katrā vaicājuma izpildes solī notiek šādas darbības:

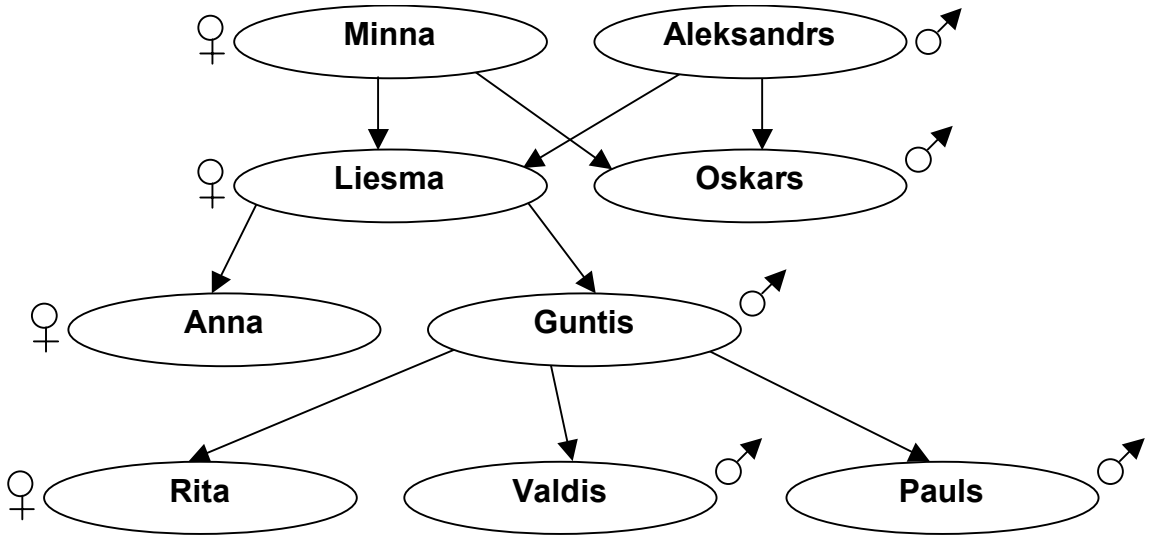
- attiecīga fakta vai noteikuma izvēle no programmas, kas atbilst dotajam programmas kontekstam (pašā sākumā – tieši pašam vaicājumam),
- mainīgo un objektu savietošana starp izvēlēto faktu/noteikumu un izpildes kontekstu,
- apakšvaicājumu izsaukšana un rezultāta iegūšana no tiem,
- lēmuma pieņemšana un, ja nepieciešams, rezultāta izvade.

Šajā nodaļā tiks apskatīta vaicājumu izpildes analīze vairākiem iepriekš skatītiem piemēriem uz programmas *family2* bāzes.

7.2. Bāzes programma

Lai uzskatāmi parādītu vaicājumu izpildi, visi programmas fakti un noteikumi tika numurēti (tieši tā, kā ir numurētas aksiomas matemātiskās loģikas teorijās).

Att. 7-1. Dzimtas koka fragments, kuru realizē programma Att. 7-2.



Att. 7-2. Prolog programmas piemērs (*family2.pl*) ar numurētiem teikumiem.

```
A1.   parent(minna,liesma).
A2.   parent(minna,oskars).
A3.   parent(aleksandrs,liesma).
A4.   parent(aleksandrs,oskars).
A5.   parent(liesma,anna).
A6.   parent(liesma,guntis).
A7.   parent(guntis,rita).
A8.   parent(guntis,valdis).
A9.   parent(guntis,pauls).
A10.  female(minna).
A11.  female(liesma).
A12.  female(anna).
A13.  male(aleksandrs).
A14.  male(oskars).
A15.  male(guntis).
A16.  male(valdis).
A17.  grandparent(X,Y):-parent(X,Z),parent(Z,Y).
A18.  sibling(X,Y):-parent(Z,X),parent(Z,Y),X\=Y.
A19.  ancestor(X,Y):-parent(X,Y).
A20.  ancestor(X,Y):-parent(X,Z),ancestor(Z,Y).
```

7.3. Vaicājumi, kas ekspluatē tikai faktus

7.3.1. Fakta eksistence ar apstiprinošu atbildi

```
?- parent(guntis,pauls).

yes
```

Izvedums.

```
S1. parent(guntis,pauls).
  A9. guntis=guntis, pauls=pauls
```

```
yes
```

Izveduma analīzes 2. rindiņā tiek izvēlēts noteikums A9 (izveduma analīzē vienkāršības labad arī faktus sauksim par noteikumiem). Pēc tam notiek pozīciju (šeit: vērtību) savietošana starp izvēlēto noteikumu un vaicājumu. Visas pozīcijas tiešā veidā ir savietojamas, un atbilde ir **yes**. Acīmredzami nederīgie noteikumi izveduma analīzē vienkāršības dēļ vispār netiek uzrādīti, kaut arī sistēmai teorētiski ir jāizskata visi noteikumi. Ar S1, S2, utt. tiks apzīmēti izpildes soļi.

7.3.2. Fakta eksistence ar noliedzošu atbildi

```
?- parent(pauls,guntis).

no
```

Izvedums.

```
S1. parent(pauls,guntis).
  no
```

7.3.3. Vērtību uzskaitījums

```
?- parent(guntis,X).

X = rita ? ;

X = valdis ? a

X = pauls

yes
```

Izvedums.

```
S1. parent(guntis,X?).
  A7. guntis=guntis, rita=X
  X=rita
  A8. guntis=guntis, valdis=X
  X=valdis
  A9. guntis=guntis, pauls=X
  X=pauls
  yes
```

Parādot savietošanu starp izvēlēto noteikumu un vaicājumu, rīkosimies tā (bet tas ir gaumes jautājums), ka kreisajā pusē no salīdzināšanas operācijas ir pozīcija (mainīgais vai objekts) no attiecīgā noteikuma, bet labajā – no vaicājuma, piemēram, *rita=X*. Vēl nekonkretizētam mainīgajam beigās pievieno jautājuma zīmi, piemēram, X?.

Sistēmas pieņemtais lēmums un sniegtais rezultāts parādās treknināti (*bold*) un atsevišķā rindā, piemēram, **X=valdis**.

Tikai faktus ekspluatējošo vaicājumu izpildes analīze ir ļoti vienkārša un parasti acīmredzama, tāpēc kaut kāda jēga no tā ir tikai tādu izvedumu analīzes, kuros ietverti arī noteikumi.

7.4. Vaicājumi, kas ekspluatē vienkāršus noteikumus

Šajā sadaļā tiks apskatīti divi vaicājumi – mazbērna noskaidrošana un māsu/brāļu noskaidrošana. Abi vaicājumi tikuši apskatīti iepriekšējā nodaļā.

```
?- grandparent(minna,X).

X = anna ? a

X = guntis

no
```

Izvedums.

```
S1. grandparent(minna,X?).
  A17. X=minna, Y?=X?, parent(minna,Z?), parent(Z?,Y?)
    S2. parent(minna,Z?), parent(Z?,Y?)
      A1. Z=liesma, parent(liesma,Y?)
        S3. parent(liesma,Y?)
          A5. anna=Y, true
          anna=S2.Y, S2.Y=S1.X
          X=anna
          A6. guntis=Y, true
          guntis=S2.Y, S2.Y=S1.X
          X=guntis
      A2. Z=oskars, parent(oskars,Y?)
        S3. parent(oskars,Y?)
          no
```

Pēc tam, kad, piemēram, solī S3 pēc noteikuma A5 izvēles notiek veiksmīga mainīgā Y savietošana ar *anna*, notiek arī unifikācija hierarhiski uz augšu līdz pat S1 līmenim (*anna=S2.Y, S2.Y=S1.X*), tāpēc atbilde ir *X=anna*, nevis *Y=anna*.

Izveduma koka būvēšanas nianses ir gaumes jautājums – to var darīt dažādos veidos, t.sk. grafiski. Pats galvenais, lai būtu skaidra secība, kā tiek pieņemts lēmums un iegūts rezultāts.

```
sibling(rita,A).

A = valdis ? a

A = pauls

no
```

Izvedums.

```
S1. sibling(rita,A?).
  A18. X=rita, Y?=A?, parent(Z?,rita), parent(Z?,Y?), rita≠Y
    S2. parent(Z?,rita), parent(Z?,Y?), rita≠Y
      A7. guntis=Z, parent(guntis,Y?), rita≠Y
        S3. parent(guntis,Y?), rita≠Y
          A7. rita=Y, rita≠rita, fail
          A8. valdis=Y, rita≠valdis, true
          valdis=s2.Y, s2.Y=s1.A
          A=valdis
          A9. pauls=Y, rita≠pauls, true
          pauls=s2.Y, s2.Y=s1.A
```

```
        A=pauls
    no
```

Ja konkrētā solī ir pieņemts negatīvs lēmums (rita≠rita ⇒ *fail*), tas nenozīmē, ka izpilde kopumā beidzas neveiksmīgi, bet gan, ka izpildē jāpaiet vienu hierarhisko līmeni uz augšu un tur jāmeklē nākošais variants. Tikai tad, kad šāda neveiksme aiziet līdz pašam augšējam līmenim, sistēma beidz darbu ar **no**. *"fail"* atšķirībā no *"no"* nozīmē lokālu neveiksmi, nevis visa vaicājuma izgāšanos.

7.5. Vaicājumi, kas ekspluatē rekursīvus noteikumus

Šādu vaicājumu izpilde vispārīgā gadījumā ir sarežģītāk izsekojama, jo izpildes apakšvaicājumi var veidot dziļu hierarhisku struktūru, tomēr izpildes princips ir tieši tāds pats, kā iepriekš.

```
ancestor(X,rita).

X = guntis ? a

X = minna

X = aleksandrs

X = liesma

no
```

Izvedums.

```
S1. ancestor(X?,rita).
  A19. Y=rita, parent(X?,rita)
    S2. parent(X?,rita)
      A7. guntis=X, true
      guntis=s2.X, s2.X=S1.X
      X=guntis
  A20. Y=rita, parent(X?,Z?), ancestor(Z?,rita)
    S3. parent(X?,Z?), ancestor(Z?,rita)
      A1. minna=X, liesma=Z, ancestor(liesma,rita)
        S4. ancestor(liesma,rita)
          A19. fail
          A20. X=liesma, Y=rita, parent(liesma,Z?),
                         ancestor(Z?,rita)
            S5. parent(liesma,Z?), ancestor(Z?,rita)
              A5. anna=Z, parent(anna,rita)
                S6. ancestor (anna,rita), fail
                  A19. fail
                  A20. X=anna, Y=rita, parent(anna,Z?),
                               ancestor(Z?,rita)
                    S7. parent(anna,Z?), ancestor(Z?,rita)
                      parent(anna,Z?)⇒fail
              A6. guntis=Z, ancestor (guntis,rita)
                S8. ancestor (guntis,rita)
                  A7. parent(guntis,rita), true
                  guntis=S5.Z, S4⇒true, minna=S3.X, minna=S1.X
                  X=minna
```

```
      A2. minna=X, oskars=Z, ancestor(oskars,rita)
        S9. ancestor(oskars,rita)
          A19. parent(oskars,rita), fail
          A20. X=oskars, Y=rita, parent(oskars,Z?), parent(Z?,rita)
            S10. parent(oskars,Z?), parent(Z?,rita)
              parent(oskars,Z?)⇒fail
      A3. aleksandrs=X, liesma=Z, ancestor(liesma,rita)
        /*izvedums, tāds pats kā Minnas gadījumā, tiek izlaists*/
                  X=aleksandrs
      A4. aleksandrs=X, oskars=Z, ancestor(oskars,rita)
        /*izvedums tiek izlaists*/
        fail
      A5. liesma=X, anna=Z, ancestor(anna,rita)
        /*izvedums tiek izlaists*/
        fail
      A6. liesma=X, guntis=Z, ancestor(guntis,rita)
        S11. ancestor(guntis,rita)
          A7. parent(guntis,rita), true
          guntis=S3.Z, liesma=S3.X, liesma=S1.X
          X=liesma
    /*tālākā neveiksmīgā izveduma daļa tiek izlaista*/
  no
```

Šādu izvedumu veidošana nav tik daudz sarežģīta, cik ķēpīga, tomēr to veidošana ļauj izprast veidu, kā Prolog sistēma izpilda programmas, kas ļauj veidot efektīvākas programmas un izvairīties no kļūdām, kas varētu novest pie izpildes ieciklošanās.

7.6. Vingrinājumi

Izveidot izpildes izvedumus dotajiem vaicājumiem.

Vingrinājums #1.

```
?- parent(X,oskars).

X = minna ? a

X = aleksandrs

no
```

Vingrinājums #2.

```
?- grandparent(liesma,pauls).

yes
```

Vingrinājums #3.

```
?- grandparent(oskars,valdis).

no
```

Vingrinājums #4.

```
sibling(anna,guntis).

true ? ;

no
```

Vingrinājums #5.

```
sibling(anna,guntis).

true ? ;

no
```

Vingrinājums #6.

```
?- sibling(X,guntis).

X = anna ? ;

no
```

Vingrinājums #6.

```
?- grandparent(X,anna).

X = minna ? a

X = aleksandrs

no
```

Vingrinājums #7.

```
?- ancestor(minna,anna).

true ? ;

no
```