5. Prolog programmas struktūra

Nodaļas saturs

Nodaļā ievietotie attēli, programmu piemēri un tabulas

Koncentrēts Prolog programmas struktūras un darbināšanas principu apraksts dots 1. nodaļā. Šajā nodaļā tiek turpināts valodas Prolog apskats. Materiāls veidots, pieņemot, ka lasītājs jau prot palaist kādu no Prolog vidēm un darbināt tajā programmas.

5.1. Fakti, noteikumi un vaicājumi

Fakti, noteikumi un vaicājumi ir 3 konstrukciju veidi, kas ir iesaistīti Prolog programmu veidošanā un darbināšanā. Fakti un noteikumi ir Prolog programmas komponentes, bet vaicājumi ir Prolog programmas darbināšanas līdzeklis.

5.1.1. Fakti un noteikumi – Prolog programmas pamatelementi

Prolog programmu veido divu veidu konstrukcijas:

- **fakti** (*facts*),
- **noteikumi** (*rules*).

Abas šīs konstrukcijas atbilst vienkāršākām vai sarežģītākām aksiomām formālās teorijās (t.sk. predikātu loģikā).

Katrs fakts un katrs noteikums Prolog programmā ir viens **teikums**, kas beidzas ar punktu (.).

Fakts Prolog programmā ir pierakstāms formā:

```
P.
```

kur P – predikāts.

Noteikums Prolog programmā ir pierakstāms formā:

```
P :- Q1, Q2, ..., Qn.
```

kur P, Q_1..Q_n – predikāti, kas matemātiskās loģikas pierakstā izskatītos šādi:

$$Q_1 \,\&\, Q_2 \,\&\, \ldots \,\&\, Q_n \rightarrow P$$

vai, precīzāk, pieņemot, ka predikātos parādās mainīgie $x_1..x_m$:

$$\forall x_1 \ldots \forall x_n \,(Q_1 \,\&\, Q_2 \,\&\, \ldots \,\&\, Q_n \rightarrow P)$$

Nākošajās nodaļās uz vairāku piemēru bāzes tiks aprakstīta faktu un noteikumu sintakse un semantika, kā arī vaicājumu veidošana.

5.1.2. Vaicājumi – Prolog interaktīvās vides līdzeklis

> **Vaicājums** (*query*) ir konstrukcija, ar kuras palīdzību Prolog interaktīvajā vidē no programmas tiek iegūta informācija.

Prolog vaicājums ir līdzīgs teorēmai matemātiskajā loģikā, bet atbildes rēķināšana uz vaicājumu (ko dara sistēma) atbilst teorēmas pierādīšanai.

Atšķirībā no matemātiskās loģikas teorēmām, vaicājuma rezultāts vai starprezultāti nav tikai atbildes *true* un *false*, bet tie var saturēt arī noteiktas vērtības, kuras reprezentē vaicājumā ietvertie mainīgie.

5.2. Fakti – vienkāršākie Prolog programmas elementi

Prolog programma sastāv no teikumiem.

> **Teikums** (*clause*) ir pabeigta Prolog konstrukcija, kas definē kādu izteikumu.

Katrs teikums beidzas ar punktu.

> **Fakts** (*fact*) ir Prolog programmas teikums, kas definē vienkāršu patiesu izteikumu un sastāv no vienas predikāta galvas.

Viens fakts Prolog programmā izskatās šādi:

```
parent(minna,liesma).
```

Ar to Prolog programmā var pierakstīt, ka Minna ir vecāks Liesmai, tomēr semantiskā jēga šeit paliek lietotāja ziņā.

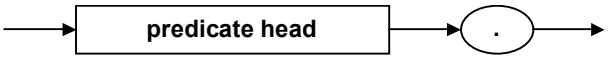
Predikāts (*predicate*) ir Prolog programmas jēdziens, kas definē noteiktu attiecību starp tā argumentiem.

Predikāts sintaktiski līdzinās tradicionālo programmēšanas valodu funkcijām, tomēr tā izmantošana daudzos gadījumos būtiski atšķiras.

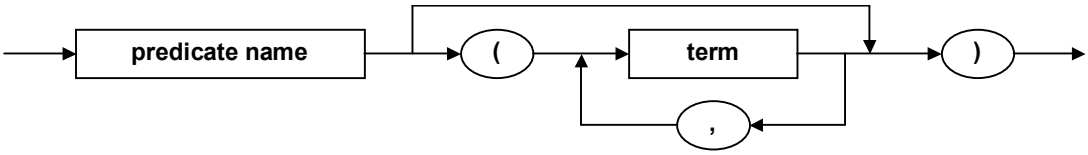
Predikātu programmā definē viens vai vairāki teikumi – arī ar to predikāts atšķiras no parastas funkcijas, kuru parasti definē vienā gabalā.

Sintaktiski fakts sastāv no predikāta vārda, kam iekavās seko ar komatiem atdalīts termu saraksts. Vienkāršākajā gadījumā terms ir vai nu konstante (atoms), vai nu mainīgais, beigās liekot punktu.

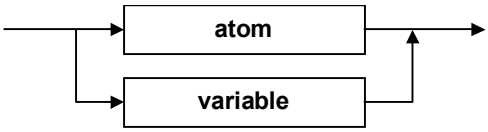
Att. 5-1. Sintakse: fakts (*fact*).



Att. 5-2. Sintakse: predikāta galva (*predicate head*).



Att. 5-3. Sintakse: terms (*term*).



5.3. Prolog programmas piemērs ar faktiem

5.3.1. Programma un tās reprezentētā problēma

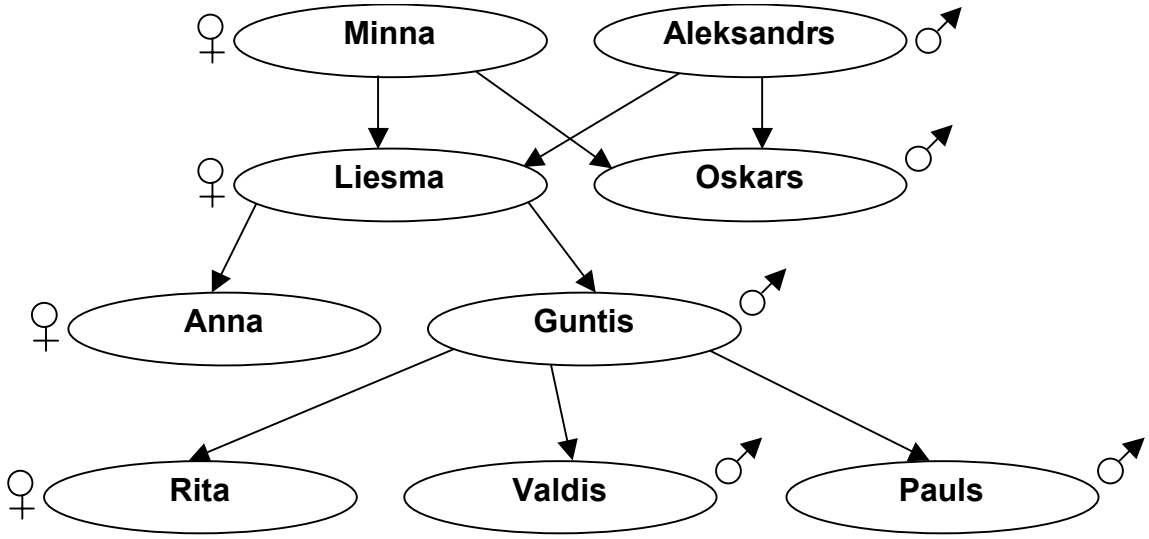
Piemērā parādītās programmas fakti reprezentē nelielu dzimtas koka fragmentu, kurā parādītas dažu personu attiecības "vecāks-bērns" (*parent*), kā arī norādīts dzimums (*female*, *male*).

Att. 5-4. Prolog programmas piemērs (*family.pl*).

```
parent(minna,liesma).
parent(minna,oskars).
parent(aleksandrs,liesma).
parent(aleksandrs,oskars).
parent(liesma,anna).
parent(liesma,guntis).
parent(guntis,rita).
parent(guntis,valdis).
parent(guntis,pauls).
female(minna).
female(liesma).
female(anna).
```

```
female(rita).
male(aleksandrs).
male(oskars).
male(guntis).
male(valdis).
male(pauls).
```

Att. 5-5. Dzimtas koka fragments, kuru realizē programma Att. 5-4.



Šajā programmā ietvertajos faktos visi termi (predikātu argumenti) ir atomi (konstantas vērtības) un nav neviena mainīgā, kuru izmantošana tika apskatīta vēlāk.

parent/2

Predikāts *parent/2* nosaka vecāka-bērna attiecības (tagad un turpmāk aiz predikāta pēc slīpsvītras tiks uzrādīts predikāta argumentu skaits jeb aritāte). Pirmie 9 teikumi (fakti) definē vecāka-bērna attiecības 9 personu pāriem.

Predikātu nosaukumi un atomi ar mazo burtu

Ievērojiet, ka atšķirībā no matemātiskās loģikas, šeit:

- predikātu nosaukums sākas ar mazo burtu, piemēram, *parent*,
- atomi sākas ar mazo burtu, piemēram, *anna*.

female/1, male/1

Ar šiem predikātiem nosaka objekta piederību attiecīgi sieviešu vai vīriešu dzimumam. Šos predikātus definē, attiecīgi izmantojot 3 un 4 faktus.

Tradicionālā nozīmē šo būtu grūti nosaukt par programmu, tomēr, izrādās, ka, izmantojot **vaicājumus**, no šīs programmas iespējams iegūt daudz informācijas.

5.3.2. Vaicājumi dotajai programmai

Šajā nodaļā tiks apskatīta jautājumu, t.i. vaicājumu uzdošana, ar kuru palīdzību no programma glabātajiem faktiem tiks iegūta informācija.

Tālāk tiks apskatīti konkrēti vaicājumu piemēri, tomēr detalizēti vaicājums kā konstrukcija tiks apskatīts nākošajās nodaļās.

Pirms sākt strādāt ar programmu, tā ir jāielādē sistēmā. To veic, piemēram, šādi:

```
?- consult(family).
```

Sīkāks apraksts par programmas ielādēšanu dots nodaļā 1.3.3.

5.3.2.1. Pārbaude uz fakta eksistenci – vienkāršākais vaicājumu veids

Vienkārši un eleganti.

Vienkāršākie vaicājumi ir fakta esamības noskaidrošana programmā.

Mēs varam pajautāt, vai Pauls ir Gunta bērns (t.i., vai Guntis ir Paula vecāks) un iegūt atbildi *yes*.

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

yes
```

Tāpat mēs varam pārliecināties, ka Pauls nav Gunta vecāks:

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

no
```

Nepārliecinošs eksistences apstiprinājums – Prolog īpatnība.

Ar Aleksandru un Liesmu varētu būt grūtāk, kaut arī no loģikas viedokļa tas ir tieši tas pats:

```
?- parent(aleksandrs,liesma).

true ? ;

no
```

GNU-Prolog dod atbildi *true*, nevis noslēdzošo *yes*, kas nozīmē, ka viens šāds fakts jau ir atrasts, bet šobrīd nav zināms, vai gadījumā nav vēl kāds šāds fakts (tātad, teorētiski programmā varētu būt uzrādīti vairāki šādi fakti). Nospiežot semikolu, iegūstam *no*, tātad, otra šāda fakta sistēmā nav. Šāda Prolog uzvedība var likties muļķīga – un, ja programma sastāvētu tikai no faktiem (kā tas ir mūsu gadījumā), tad tā arī varētu uzskatīt, bet programmas tikai ar faktiem (bez noteikumiem) ir gandrīz tas pats, kas algoritmi bez cikla konstrukcijām, tāpēc tālākās programmas būs krietni sarežģītākas.

Atbildi šai atšķirībai var ieraudzīt programmā – kopā ir 3 fakti, kuros Guntis ir vecāks un fakts *parent(guntis,pauls)* ir pēdējais no tiem, savukārt fakts *parent(aleksandrs,liesma)* nav pēdējais starp faktiem, kur Aleksandrs ir vecāks. Šī īpatnība parāda, ka valodā Prolog, atšķirībā no matemātiskās loģikas, faktu (vispārīgā gadījumā – teikumu) **secībai ir nozīme**.

Neeksistējošs predikāts.

Ja mēs mēģinātu vaicājumā izmantot predikātu, kas vispār nav definēts programmā (piemēram, *vecaks/2* predikāta *parent/2* vietā), Prolog sistēma, atkarībā no realizācijas pateiktu vai nu to pašu *no*, vai izdotu kļūdas paziņojumu, kā to dara GNU-Prolog:

```
?- vecaks(guntis,pauls).
uncaught exception:
    error(existence_error(procedure,vecaks/2),top_level/0)
```

5.3.2.2. Objektu vērtību iegūšana – vaicājuma papildus rezultāts

Iepriekš apskatītajos vaicājumos tika jautāts – "ir vai nav taisnība, ka...", iegūstot atbildi: *yes* vai *no*. Tas lielā mērā atbilst teorēmas pierādījumam matemātiskajā loģikā. Tomēr Prolog sistēmā ir daudz lielākas iespējas iegūt rezultātus. Piemēram, mēs varam paprasīt, kas ir Paula vecāks:

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

X = guntis

yes
```

Ievērojiet, ka šeit pirmo reizi **vaicājumā parādās mainīgais** (X; Prolog sistēmā ir noteikts, ka ar lielo burtu sākas mainīgā vārds). Tādējādi, sasniedzot rezultātu (*yes*), tiek parādīta arī mainīgā/mainīgo vērtība (*X = guntis*).

Atbilžu variantu uz vaicājumu var būt vairāk nekā viens (turklāt, kā jau iepriekš vienā gadījumā tika ievērots, arī dodot beidzamo no atbildēm, sistēma tajā brīdī var vēl nezināt, ka šī atbilde ir beidzamā). Piemēram, vaicājums: kas ir Gunta bērni:

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

X = rita ? ;

X = valdis ? a

X = pauls

yes
```

Sistēma neatgriež uzreiz visus rezultātus, bet tikai vienu (jo teorētiski rezultātu var būt ļoti daudz).

- Lai iegūtu nākamo rezultātu spiež **semikolu (;)**.
- Lai iegūtu visus nākamos rezultātus, spiež **burtu a**.
- Lai pārtrauktu rezultātu izvadi, spiež **ENTER**.

(Dažādās Prolog sistēmās tas var nedaudz atšķirties.)

Tāpat mēs varam uzzināt, kādi mums vispār ir reģistrētie vecāku-bērnu pāri. To dara, abos argumentos vaicājumā liekot mainīgos:

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

X = minna
Y = liesma ? a

X = minna
Y = oskars
```

```
X = aleksandrs
Y = liesma
...
```

5.3.2.3. Saliktie vaicājumi

Sasaiste pēc vienāda mainīgo nosaukuma.

Vaicājumam formai nav jāaprobežojas tikai ar vienu predikāta galvu, kā tas ir faktu gadījumā – to var veidot kā vienkāršu vaicājumu kopumu, apvienojot un sasaistot dažādu veidu informāciju no programmas. Vispārīgā gadījumā vaicājums ir ar komatiem atdalītu predikātu galvu vai citu konstrukciju kopums. Izmantojot saliktu vaicājumu, var, piemēram, noskaidrot, kas ir Minnas mazbērni:

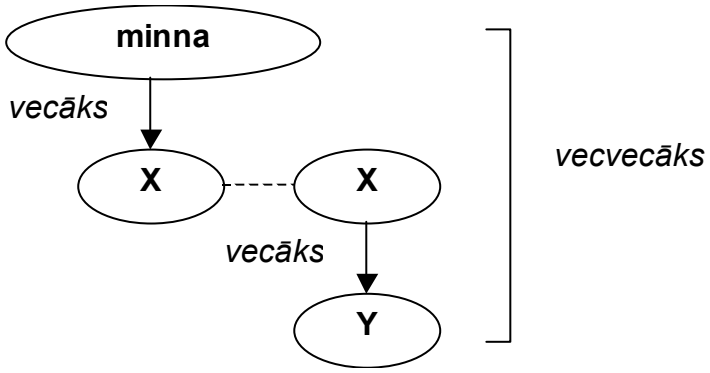
```
?- parent(minna,X),parent(X,Y).

X = liesma
Y = anna ? a

X = liesma
Y = guntis
```

Vaicājums atgriež divas atbildes – katrā no tām bez mazbērna tiek uzrādīts attiecīgais Minnas bērns. Nākošajā attēlā parādīts princips, kā tiek veikts dotais vaicājums.

Att. 5-6. Vaicājums mazbērna noskaidrošanai.



Attēls demonstrē, ka tas, ka abās vaicājuma komponentēs tiek izmantots **viens un tas pats mainīgais X**, nav nejauši. Viena mainīgā lietošana viena teikuma, t.sk. viena vaicājuma, ietvaros, nozīmē, ka attiecīgās teikuma pozīcijas ir unificējamas, t.i., izpildes laikā tām atbilst vieni un tie paši objekti vai vērtības. Tādējādi ar šādu vaicājumu mēs meklējam, divas tādas attiecības "vecāks-bērns", kur tai personai, kas vienā no attiecībām ir bērna lomā, otrā attiecībā jābūt vecāka lomā.

Anonīmie mainīgie.

Nākošais uzdevums ir atrast visas mātes – t.i., tādas personas, kuras ir vecāki un kuras ir sievietes.

```
?- parent(X,Y),female(X).

X = minna
```

```
Y = liesma ? a

X = minna
Y = oskars

X = liesma
Y = anna

X = liesma
Y = guntis

no
```

Šajā vaicājumā tiek atgriezti 4 rezultāti, kas reprezentē tos atbilstošos 4 faktus ar predikātu *parent/2*, kur vecāks ir sieviete. Tomēr rezultāts parāda arī liekus rezultātus – t.i., ar šo māšu bērnus (mainīgais Y). Valodā Prolog ir iespēja izvairīties no lieko pozīciju parādīšanas, marķējot tās ar t.s. anonīmajiem mainīgajiem, kuri tiek reprezentēti ar pasvītrojuma zīmi (_). To demonstrē nākošais piemērs.

```
?- parent(X,_),female(X).

X = minna ? a

X = minna

X = liesma

X = liesma

no
```

Tas jau ir tuvu tam, ko vēlējāmies. Pēdējā problēma ir tā, ka viena persona rezultātos var tikt parādīta vairākas reizes – šajā gadījumā tik, cik tai programmā atbilst faktu ar *parent/2*, kur šī persona ir vecāka lomā. Arī šo problēmu iespējams risināt, tomēr tas tiks apskatīts nākamajās nodaļās.

5.4. Prolog svarīgāko elementu kopsavilkums

Tab. 5-1. Prolog svarīgāko elementu un mehānismu kopsavilkums.

Nosaukums	Piemērs	Komentārs
Mainīgais	`A`	Sākas ar lielo burtu.
Atoms	`anna`	Sākas ar mazo burtu, reprezentē tekstuālu vērtību.
Terms	`A anna`	Mainīgais vai atoms.
Predikāts	`parent(A,B)`	Predikāts definē noteikt attiecību. Predikāta nosaukums sākas ar mazo burtu, predikātam ir noteikts argumentu skaits.
Fakts	`parent(anna,leo).`	Definē vienkāršu patiesu izteikumu.
Vaicājums	`?- parent(X,Y).`	Informācijas iegūšana no programmas.
Teikums		Pabeigta Prolog konstrukcija, kas beidzas ar punktu, piemēram fakts vai vaicājums.

Sasaiste pēc mainīgā nosaukuma	`?- parent(minna,X), parent(X,Y).`	Mainīgie ar vienādu nosaukumu viena teikuma (šeit, vaicājuma) ietvaros nosaka sasaistāmās pozīcijas.

5.5. Vingrinājumi

Vingrinājums #1.

Uzrakstīt vaicājumu programmai *family*, kas atgriež tās personas, kas pēc sistēmā reģistrētās informācijas ir uzskatāmas par vectētiņiem.

Vingrinājums #2.

Uzrakstīt vaicājumu programmai *family*, kas atgriež tās personas, kas ir brāļi vai māsas Valdim (tās, kam ar Valdi ir kopīgs vecāks, resp., Rita un Pauls). Neuztraukties, ka programma atgriezīs arī pašu Valdi.

Vingrinājums #3.

Uzrakstīt vaicājumu programmai *family*, kas atgriež visus tos vīriešus, kam ir kāda māsa (šeit: Pauls, Valdis, Guntis, Oskars).

Vingrinājums #4.

Papildināt programmai *family* ar predikātu *teacher/2* (skolotājs), definējot, ka Oskars ir skolotājs Ritai un Valdim. Uzrakstīt vaicājumu, kas atgriež visas skolnieces. (Pēc programmas koda papildināšanas, lai izmaiņas stātos spēkā, vēlreiz izsaukt komandu *consult(family)*.)