6. Noteikumi – faktu inteliģents paplašinājums

Nodaļas saturs

Nodaļā ievietotie attēli, programmu piemēri un tabulas

6.1. Noteikumu definēšana

Iepriekšējā nodaļā apskatītā programma bija ļoti vienkārša un sastāvēja tikai no faktiem. Tomēr arī no tik vienkāršas programmas, izmantojot saliktos vaicājumus, varēja iegūt netriviālus rezultātus.

Saliktajos vaicājumos esošo "sarežģītību" var "pārnest" uz programmu, tādējādi no faktiem "izveidojot" noteikumus.

> **Noteikums** (*rule*) ir Prolog programmas teikums, kas definē patiesu izteikumu, kas atkarīgs no citu nosacījumu izpildes.

Pirmais salikta vaicājuma piemērs iepriekšējā nodaļā bija vecvecāka-mazbērna attiecības noskaidrošana divām personām:

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

Pēc līdzības ar šo vaicājumu, var uzbūvēt jaunu noteikumu, ar kuru papildināt programmu *family*.

```
grandparent(X,Y):-parent(X,Z),parent(Z,Y).
```

Ar šo noteikumu tiek definēts predikāts *grandparent/2*, kas nosaka, vai divas personas ir attiecībās vecvecāks-mazbērns.

Kā redzams, noteikums sastāv no divām daļām – galvas (*head*) un ķermeņa (*body*), starp kurām ir operators ":-".

Esot definētam predikātam *grandparent/2*, vaicājums noskaidrot Minnas mazbērnus izskatās šādi:

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

X = anna ? a

X = guntis

no
```

Tikpat viegli ir noskaidrot, kas ir Gunta vecvecāki:

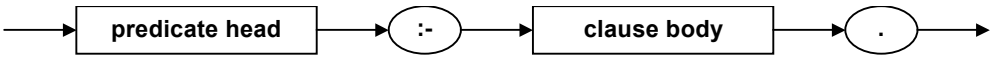
```
grandparent(X,guntis).

X = minna ? ;

X = aleksandrs ? ;

no
```

6.2. Noteikumu un vaicājumu pieraksta sintakse

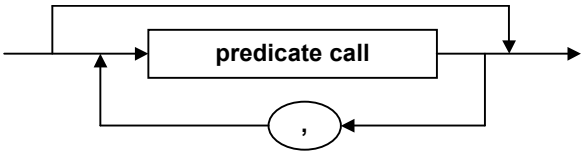
Noteikums sastāv no 2 daļām:

- **galva** (*head*) apraksta definējamo predikātu,
- **ķermenis** (*body*) apraksta nosacījumus, kam jābūt spēkā, lai izpildītos dotais predikāts.

Att. 6-1. Sintakse: noteikums (*rule*).



Att. 6-2. Sintakse: teikuma ķermenis (*clause body*).

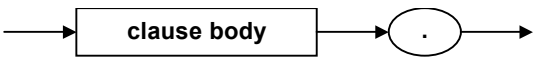


Predikāta izsaukums (*predicate call*) parasti atgādina predikāta galvu, tomēr tas var reprezentēt arī citas konstrukcijas, piemēram, salīdzināšanu, aritmētiskas operācijas u.c., kas tiks apskatīts tuvākajos piemēros.

Vizuāli varētu teikt, ka noteikums ir konstrukcija, kas it kā apvieno faktu un vaicājumu. Praktiski var teikt, ka fakts gan ir noteikuma speciālgadījums, tomēr vaicājumus gluži nē – kaut vai tādēļ, ka vaicājuma pielietošanas apgabals ir interaktīvā sistēma, kamēr noteikums ir programmas sastāvdaļa.

Tomēr, ja ir nodefinēta noteikuma sintakse, tad ir triviāli nodefinēt arī vaicājuma sintaksi.

Att. 6-3. Sintakse: vaicājums (*query*).



6.3. Salīdzināšana uz unificēšanos

Jau iepriekš tika demonstrēts, ka vienādi mainīgo nosaukumi dažādās viena teikuma pozīcijās nodrošina šo pozīciju sasaisti jeb unifikāciju programmas izpildes laikā, un tas attiecas gan uz noteikumiem, gan vaicājumiem.

Tomēr vienādu mainīgo nosaukumu lietošanas vietā var izmantot arī salīdzināšanas operāciju "=", tādējādi, programmas fragments:

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

funkcionāli atbilst šādam fragmentam:

```
parent(minna,X1),parent(X2,Y),X1=X2.
```

Ja šajā piemērā tas ir tikai alternatīvas iespējas demonstrācija (kur labāk tomēr lietot pirmo variantu), tad citādāk tas ir salīdzināšanas operāciju uz nevienādību, kas valodā Prolog ir pierakstāms kā divu simbolu kombinācija "\=".

Tipisks piemērs nevienādības operācijai ir māsas-brāļa attiecības (*sibling*) noskaidrošana – māsa vai brālis otrai personai ir tas, kam ir kopīgs vecāks un kas **nav** šī pati persona. Bez salīdzināšanas operācijas Prolog programmā to pateikt nav iespējams, jo Prolog programmā dažādos mainīgajos drīkst unificēties vienādi objekti.

Nākošais noteikums, kurš jāpievieno programmai definē brāļa-māsas attiecību:

```
sibling(X,Y):-parent(Z,X),parent(Z,Y),X\=Y.
```

Šeit ir arī redzams pirmais piemērs, kur predikāta izsaukums ieņem citu formu nekā predikāta galva.

Šāda noteikuma ieviešana dod iespēju uzdot šādu jautājumu par Ritas brāļiem un māsām:

```
sibling(rita,A).

A = valdis ? a

A = pauls

no
```

Salīdzināšanas operācija "\=" nodrošina to, ka šajā sarakstā nav pašas Ritas.

Jautājumu var uzdot arī otrādi, jo abi predikāta argumenti darbojas simetriski:

```
sibling(A,guntis).

A = anna ? ;

no
```

Ar salīdzināšanas operācijām (šeit "=" un "\=") Prologā jābūt ļoti uzmanīgam, jo valodā Prolog kopā ir 3 dažādu veidu salīdzināšanas operācijas.

Ja parastajās programmēšanas valodās ir tikai viena veida salīdzināšanas – uz objektu sakritību, tad Prologā tas ir tikai viens no veidiem, turklāt ne tas izplatītākais. Šajos programmu fragmentos izmantotās salīdzināšanas operācijas realizē **salīdzināšanu uz unificēšanos**, t.i. sasaisti starp pozīcijām.

6.4. Rekursīva noteikumu definēšana

Pirmais šajā nodaļā apskatītais noteikums definē predikātu *grandparent/2*. Ja *parent/2* apzīmē radniecību pirmajā pakāpē, tad *grandparent/2* – otrajā pakāpē. Abos gadījumos pakāpe ir fiksēta, un šādā veidā nevar uzkonstruēt vienu noteikumu dažādām radniecības pakāpēm. Piemēram, predikātu priekštecis rekursīvi varētu definēt šādos divos punktos:

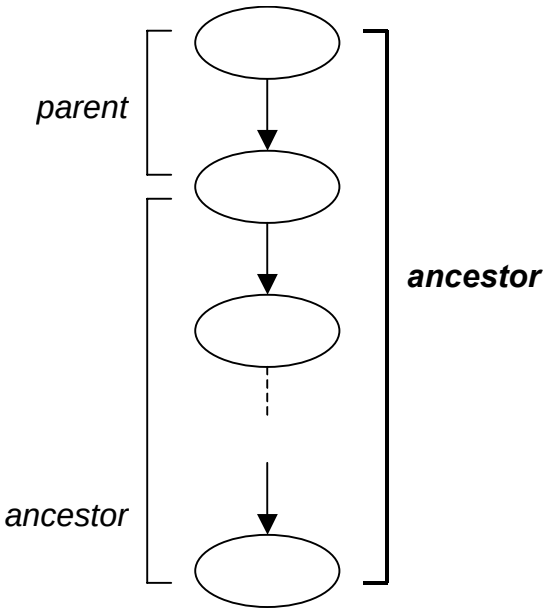
- jebkurš vecāks ir arī priekštecis,
- priekštecis ir tas, kurš ir vecāks tam, par kuru ir zināms, ka tas ir priekštecis.

Prolog programmā to analoģiskā veidā apraksta divi šādi noteikumi, definējot predikātu *ancestor/2*:

```
ancestor(X,Y):-parent(X,Y).
ancestor(X,Y):-parent(X,Z),ancestor(Z,Y).
```

Lai programma pareizi darbotos, ir svarīgi, ka abi noteikumi ir tieši šādā secībā un ka otrā noteikuma ķermenī predikātu izsaukumi arī ir tieši norādītajā secībā, citādi predikāta darbināšanai draud ieciklošanās/steka pārpildīšanās. Šīs īpatnības tiks sīkāk aplūkotas nākamajās nodaļās, kur tiks analizēta vaicājumu izpildes secība.

Att. 6-4. Ar predikātu *ancestor/2* reprezentētais priekšteča jēdziens.



Papildinātā programma, pievienojot noteikumus, kas definē predikātus *grandparent/2*, *sibling/2* un *ancestor/2*, izskatās šādi:

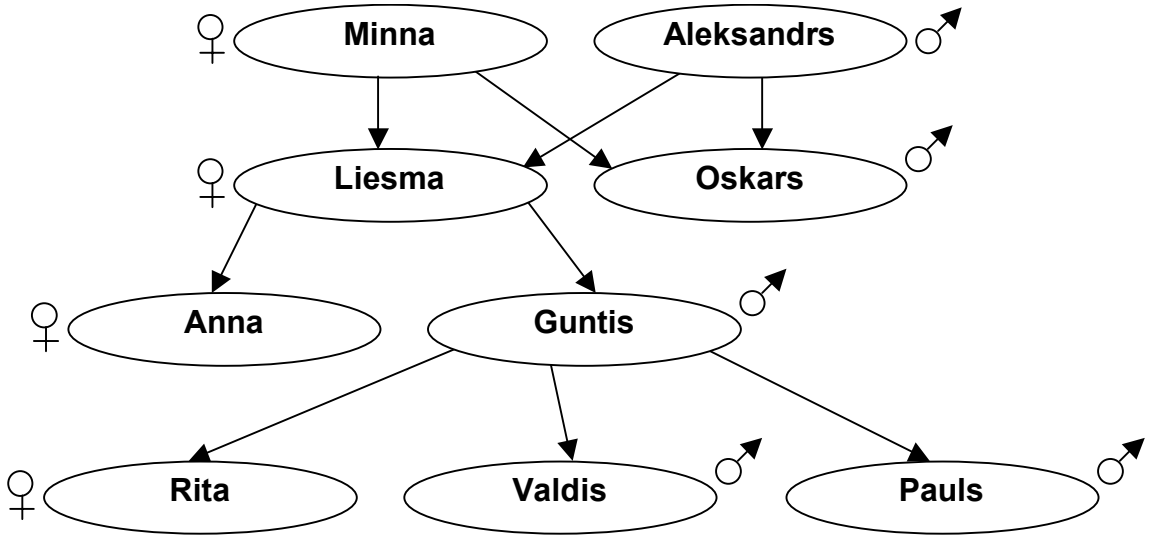
Att. 6-5. Prolog programmas piemērs (*family2.pl*).

```
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).
grandparent(X,Y):-parent(X,Z),parent(Z,Y).
sibling(X,Y):-parent(Z,X),parent(Z,Y),X\=Y.
ancestor(X,Y):-parent(X,Y).
ancestor(X,Y):-parent(X,Z),ancestor(Z,Y).
```

Atgādinājumam, dzimtas koks, kuru realizē programmas fakti.

Att. 6-6. Dzimtas koka fragments, kuru realizē programma Att. 6-5.



Kā jau tas "pierasts", arī jaunieviestais predikāts darbojas simetriski un spēj noskaidrot tiklab priekštečus, kā pēctečus:

```
ancestor(X,rita).

X = guntis ? a

X = minna

X = aleksandrs

X = liesma

no
```

```
?- ancestor(aleksandrs,X).

X = liesma ? a

X = oskars
```

```
X = anna
...
```

6.5. Mainīgie un atomi – kopsavilkums

Mainīgie un **atomi** ir svarīgākie vienkāršie termi (elementi) Prolog programmā.

6.5.1. Mainīgie

Mainīgā (*variable*) galvenais uzdevums ir savietot (sasaistīt) dažādas viena teikuma daļas (unifikācija), lai nodrošinātu noteiktu vērtību ķēdi un tādējādi nonāktu līdz rezultātam. Par Prolog programmu **nevar teikt**, ka mainīgais glabā noteiktu informāciju – mainīgais ir tikai līdzeklis paraugu savietošanas (*pattern matching*) mehānisma nodrošināšanā.

Tam kā pierādījums ir **anonīmo mainīgo** (_) lietošana – ja mainīgais neveic sasaistes funkciju, tad viņa uzdevums ir tikai aizpildīt noteiktu predikāta argumenta pozīciju, kuru tikpat labi var izdarīt arī anonīms (nekonkretizēts) mainīgais.

6.5.2. Atomi

Atoms (*atom*) Prolog programmā reprezentē konstantu vērtību un tā būtība ir vistuvāk simbolu virknei (*string*) tradicionālajās programmēšanas valodās.

Atšķirība no parastajām valodām ir tā, ka ja vērtība sākas ar mazo burtu, tad to var nelikt vienkāršajās pēdiņās: anna, guntis. Citos gadījumos vienkāršās pēdiņas ir nepieciešamas: 'Anna', '123', '123a', lai simbolu virkne netiktu uztverta par mainīgo, skaitli vai vispār sintaktiski nekorektu simbolu sakopojumu.

6.6. Vingrinājumi

Vingrinājums #1.

Definēt (programmas *family2* ietvaros) predikātu *grandmother/1* (vecmāmiņa), kas par doto personu pasaka, vai tā ir vai nav kādam vecmāmiņa.

Vingrinājums #2.

Definēt (programmas *family2* ietvaros) predikātu *cousin/2*, kas definē attiecību starp divām personām, kam ir kopējs vecvecāks.

Vingrinājums #3.

Definēt (programmas *family2* ietvaros) predikātu *sister/2*, kas definē attiecību starp divām personām, nosakot, ka pirmā ir māsa otrajam.

Vingrinājums #4.

Definēt (programmas *family2* ietvaros) predikātu *hasson/1*, kas pasaka, vai dotajai personai ir dēls.