8. Datu struktūras un aritmētika

Nodaļas saturs

Nodaļā ievietotie attēli, programmu piemēri un tabulas

8.1. Vienkāršie datu objekti

Līdz šim esam iepazinušies ar divu veidu datu objektiem – atomiem un mainīgajiem. Tie abi ir vienkāršie datu objekti, taču vēl pie vienkāršajiem datu objektiem pieder skaitļi. Kaut arī valoda Prolog pamatā domāta simboliskās informācijas apstrādei, tomēr zināmas operācijas ar skaitļiem arī, protams, ir iespējamas, tomēr, kā vēlāk būs redzams, tās veikt būs nedaudz neērtāk nekā tradicionālajās valodās.

Skaitļi var būt veseli un ar peldošo komatu. Skaitļu iespējamais pieraksts vispārīgā gadījumā var atšķirties starp dažādām Prolog realizācijām (vairāk šeit ir runa par zinātniskā pieraksta iespējamību), tomēr standarta variantā tas ir tradicionāls.

Tab. 8-1. Prolog vienkāršo datu objektu kopsavilkums.

Nosaukums	Piemērs	Komentārs
Mainīgais	`A _X3`	Sākas ar lielo burtu vai pasvītrojuma zīmi.
Atoms	`anna 'Anna'`	Sākas ar mazo burtu vai tiek likts vienkāršajās pēdiņās, reprezentē tekstuālu vērtību.
Vesels skaitlis	`87 -55`	
Skaitlis ar peldošo komatu	`1.234 -0.001`	

8.2. Datu apvienošana struktūrās, izmantojot funktorus

Datu apvienošana struktūrās veicama ļoti vienkārši:

- vairākus datu objektus apvieno virknē, atdalot ar komatiem,
- virkni ieliek iekavās un pieliek priekšā funktoru.

Piemēram, *date (1, january, 2008)*.

Sintaktiski funktors ir identisks predikāta galvai, tomēr tam ir pilnīgi cita nozīme – datu struktūras veidošana.

Nākošais piemērs demonstrē funktoru izmantošanas iespējas, demonstrējot arī to, ka predikāta pozīcijām nav noteikts datu tips, kuram jāatbilst visiem tiem piesaistītajiem datu objektiem.

Att. 8-1. Funktoru izmantošanas piemērs (*persons.pl*) ar numurētiem noteikumiem.

```
A1.   person(anda,date(1985,12,13),female).
A2.   person(juris,date(1984,5,1),male).
A3.   person(laila,date(1984,3,19),female).
A4.   person(guntis,date(1986,8,8),male).
A5.   person(valda,date(1985,10,10),female).
A6.   person(gatis,null,male).
A7.   person(ginta,null,female).
```

Šajā programma ierakstīta informācija par 7 personām, uzrādot personas vārdu, dzimšanas datumu un dzimumu. Divām personām dzimšanas datums nav zināms, un datu struktūru aizvietot *null*. Šajā gadījumā *null* ir atoms bez kādas Prolog predefinētas nozīmes.

Programmā dzimšanas datuma glabāšanai tiek izmantots funktors *date* ar 3 argumentiem.

Tālāk sekos daži vaicājumu piemēri, lai demonstrētu datu struktūru izmantošanas specifiku.

Pirmais vaicājums ir iegūt to personu vārdus, kurām sistēmā ir reģistrēts dzimšanas datums:

```
?- person(X,date(_,_,_),_).

X = anda ? a

X = juris

X = laila

X = guntis

X = valda

no
```

Tāpat var iegūt to personu vārdus, kam dzimšanas datums nav reģistrēts:

```
?- person(X,null,_).

X = gatis ? a

X = ginta

yes
```

To var izdarīt arī citādi (“\=” nozīmē aizliegumu unificēties):

```
?- person(X,D,_),D\=date(_,_,_).

D = null
X = gatis ? a

D = null
```

```
X = ginta

yes
```

Atrast sievietes, kas dzimušas gada otrajā pusē:

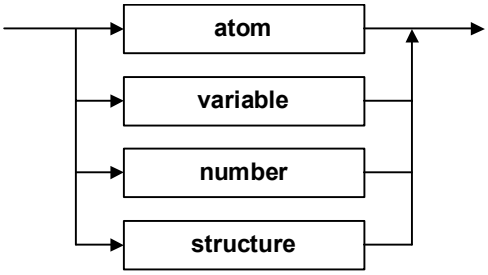
```
?- person(X,date(_,M,_),female),M>6.

M = 12
X = anda ? a

M = 10
X = valda

no
```

Alternatīvs veids salikto termu (struktūru) izmantošanai ir saraksti (*lists*), kas tiks apskatīti nākošajā nodaļā.

8.3. Terma jēdziena paplašinājums

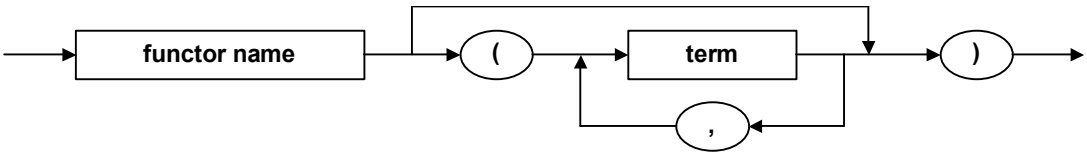
Ieviešot jēdzienus par skaitli un struktūru, mainās arī terma jēdziens, kas vienkāršotajā variantā tika definēts 5. nodaļā.

Att. 8-2. Sintakse: terms (*term*).



Struktūras definīcija sintaktiski pilnīgi sakrīt ar predikāta galvu (sk. 5. nodaļu):

Att. 8-3. Sintakse: struktūra (*structure*).



8.4. Aritmētisko operāciju izpilde

Aritmētisko darbību izpilde Prolog sistēmā no pirmā acu uzmetiena liekas nedaudz dīvaina, un tam pamatā ir Prolog darbības princips – dažādu pozīciju savietošana (unifikācija), atstājot aritmētiku otrajā plānā.

Galvenā "dīvainība ir tā", ka aritmētiskās operācijas ne vienmēr izpildās programmā vai vaicājumā – komplektā ar aritmētisku operāciju obligāti jābūt arī kādai speciālai operācijai vai komandai, kas izsauc aritmētisko izpildi (*evaluation*).

Lūk, vienkāršākais "dīvainības" piemērs, kas parāda, ka Prolog uztver "1+2" kā simbolu virkni:

```
?- X=1+2.

X = 1+2

yes
```

Operācija "=", kā jau iepriekš ir ticis pieminēts, ir salīdzināšana uz unificēšanos jeb savietošanos, kas nenes līdzi aritmētisku darbību izpildes izsaukšanu. Lai piespiestu sistēmai rēķināt, = vietā jālieto *is:*

```
?- X is 1 + 2.

X = 3

yes
```

Vēl daži piemēri – dalīšana, veselo skaitļu dalīšana un atlikums:

```
?- X is 13 / 5.

X = 2.6000000000000001

yes
?- X is 13 // 5.

X = 2

yes
?- X is 13 mod 5.

X = 3

yes
```

Veselo skaitļu dalīšanas operators // citās Prolog realizācijās varētu atšķirties un būt, piemēram, *div*.

Vēl viena "dīvainība" ir tā, ka piešķiršanas operācijai viens un tas pats mainīgais nedrīkst būt gan labajā, gan kreisajā pusē – jāizmanto cits mainīgais:

```
?- X = 1, X is X + 2.

no
```

bet:

```
?- X = 1, Y is X + 2.

X = 1
Y = 3

yes
```

Zinot šo īpatnību mēs varam uzrakstīt priekšteča predikāta otro variantu *ancestor/3*, kas divām personām pasaka, cik paaudzes tās šķir.

```
ancestor(X,Y,1):-parent(X,Y).
ancestor(X,Y,N):-parent(X,Z),ancestor(Z,Y,M), N is M + 1.
```

Predikāts jāizmanto komplektā ar programmu *family2*, pilna programmas versija kopā ar predikātu *ancestor/3* atrodama programmā *family3*.

Predikāta *ancestor/3* izmantošanas piemēri:

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

X = 2 ? ;

no
| ?- ancestor(aleksandrs,rita,X).

X = 3 ? ;

no
| ?- ancestor(liesma,guntis,X).

X = 1 ? ;

no
| ?- ancestor(oskars,anna,X).

no
```

8.5. Skaitliskā salīdzināšana

Mums ir zināmas divas salīdzināšanas operācijas "=" un "\=", kuras ir domāts salīdzināšanai uz unificēšanos. Lai salīdzinātu skaitliskas izteiksmes pēc to vērtības ir nepieciešamas skaitliskās salīdzināšanas operācijas.

```
?- 1+2 = 2+1.

no
```

Šeit rezultāts *no* ir loģisks, jo 1+2 un 2+1 tiek salīdzinātas kā simbolu virknes.

Skaitliskās vienādības operācija ir "=:=", bet nevienādības – "=\=":

```
?- 1+2 =:= 2+1.

yes
?- 1+2 =\= 2+1.

no
?- 1+2 =\= 4.

yes
```

Tab. 8-2. Populārāko Prolog skaitlisko operāciju kopsavilkums.

Operators	Piemērs	Komentārs
`is`	`A is 1 + 2`	Izteiksmes izpilde
`=:= =\=`	`1+2 =:= 2+1` `1+2 =\= 4`	Aritmētiskā vienādība un nevienādība
`< =< >= >`	`5 =< 6`	Ievērojiet, ka mazāks vai vienāds ir =<, nevis <=
`+ - * /`	`5 + 6`	"/" ir parastā dalīšana (nevis veselo skaitļu)
`//`	`12 // 5`	Veselo skaitļu dalīšana, dažādās realizācijās var atšķirties
`mod`	`12 mod 5`	Veselo skaitļu dalījuma atlikums
`abs`	`abs(-5)`	Absolūtā vērtība
`sign`	`sign(-10)`	Zīme
`min max`	`min(5,6)`	Minimālā/maksimālā vērtība no 2 skaitļiem
`**`	`2 ** 4`	Kāpināšana
`sqrt exp log`	`sqrt(2)`	Kvadrātsakne, e pakāpe, naturālais logaritms

`floor`
`ceiling`
`round`
`truncate`
	`round(3.14)`	Apaļošana

`sin cos asin` `acos atan`	`sin(3.14259/6)`	Trigonometriskās funkcijas

8.6. Vingrinājumi

Vingrinājums #1.

Uzrakstīt vaicājumu (programmas *persons* ietvaros), kas atgriež tās sievietes, kuru dzimšanas datuma dienas numurs sakrīt ar mēneša numuru.

```
?- ...

A_ = valda
D = 10 ? a

no
```

Vingrinājums #2.

Uzrakstīt predikātu *sumof/4*, kas ceturtajā pozīcijā atgriež pirmo 3 pozīciju skaitļu summu.

```
?- sumof(2,3,4,X).

X = 9

yes
```

Vingrinājums #3.

Uzrakstīt (rekursīvu) predikātu *fact/2*, kas izrēķina faktoriālu. *fact(n) = n*(n-1)*...*2*1.*

```
?- fact(5,X).

X = 120

yes
```