

Loeng 2: Loogilise programmeerimise keel

Prolog



J.Vain





Loengu kava

- Prolog keele põhimõisted, mõistete süntaks ja semantika
- Tutvumine lihtsamate programmeerimisvõtetega
- Näided
- Takeaway
 - Vajalik teadmine lihtsamate Prolog programmide kirjutamiseks



Kuidas alustada?

- Predikaatide sisestamine käsurealt:

```
consult (user) .
```

```
  lause1.
```

```
  .....
```

```
  lausen.
```

```
end_of_file.
```

% Sisestusrežiimi sisenemine

% `user` - viit standardsisendile

% vaikimisi on see klaviatuur

%Sisestusrežiimist väljumine



Kuidas alustada?

- Programmi faili laadimine mällu käsurealt:

```
? - consult('c:\\...\\faili_nimi ').
```

Näide

```
?- consult('C:\\Users\\...\\Ained\\ITI0211\\proov.pl').
```

Praktilisi võtteid alustamiseks



- Kommentaar

/ *

Kommentaar mitmel real

Kommentaar mitmel real

Kommentaar mitmel real

* /

% *Kommentaar ühel real*

Prolog keele elemendid: sümbol, aatom



- **Sümbol** (16-bit Unicode sümbol)

- #, \$ &

- **Aatom**

Aatomid on andmete, programmide, failide **nimed**:

- alfanumbriline aatom

- seeOn_aatom9 % väikese algustähega aatom

- kvoteeritud aatom (ülakomade vahel)

- 'Aatom' % suure algustähega aatom

- Reserveeritud sümbolid, mida ei ole soovitatav kasutada aatomites

- ! ; [] , . . .



Prolog keele elemendid: term

- *Termideks* on:

- konstandid:

- täisarvud
- reaalarvud
- aatomid
- listid

- muutujad

% NB! muutujad on tüüpimata



Prolog keele elemendid: list

- *List* esitab elementide loendit:

- `['Ago', 'Peeter', 'Mai']` % elemendid on aatomid
- `[]` % tühilist
- `[12, [34, 2], [peep, []], 89]` % hierarhiline list

- Listi osadele saab viidata

- otseselt ja kaudselt
- elemendi ja alamlistide kaupa

- Näide (otsene viitamine elementidele):

```
[E11, E12 | _]      % Muutujad E11 ja E12 tähistavad  
listi esimest ja teist elementi
```

- Näide (kaudne viitamine elemendile):

```
[_ | Tail]      % Muutuja Tail väärtustatakse listi sabaga s.t.  
kogu list v.a. esimene element
```




Prolog keele elemendid: Horni lause

- Horni lause (clause) esineb *fakti*, *reegli* või *päringu* kujul.
- Iga lause algab predikaadi nimega ja lõpeb punktiga.
 - Näide (fakt):
`isa(peeter, mai) .`
 - Näide (reegel):
`vanavanem(X, Z) :- vanem(X, Y), vanem(Y, Z) .`
- Ühesuguse funktori ja aarsusega laused on ühe ja sama Horni lause alternatiivid.
 - Näide (ühe reegli alternatiivsed laused):
`vanem(X, Y) :- isa(X, Y) .`
`vanem(X, Y) :- ema(X, Y) .`



Prolog keele elemendid: muutuja

- Muutujatel puudub deklareeritav tüüp
- 2 alternatiivset tähistust:
 - Muutuja nimi algab suure algustähega:
Näit.: `A`, `Inimesed`, ...
 - Muutuja nimi algab allkriipsuga:
Näit.: `_loomastik`, `_cXz`



Prolog keele elemendid: loogikatehe

- Loogikatehted HL kehas

, - konjunktsioon (loogiline „ja“)

; - disjunktsioon (loogiline „või“)

not - eitus, eitus kehtib ainult Prologi teadmusbbaasi kontekstis, seda nimetatakse "suletud maailma" eelduseks.

b:- a.	} - implikatsioon $a \Rightarrow b$
s:- a -> b. (reegli kehas)	
s:- not (a) ; b.	



Prolog keele elemendid: fakt ja päring

Faktid

tootja(kalev) .
tootja(liviko) .
tootja(saku) .
vahendaja(abestock) .
vahendaja(hulgi) .
myyja(stockman) .
myyja(spar) .
myyja(selver) .
myyja(liviko) .

Lihtpäring

? - tootja(Kes) .
? - myyja(Kes) .
? - vahendaja(Firma) .

Agregaatpäring

? - tootja(Kes), myyja(Kes) .
? - tootja(a_le_coq); myyja(saku) .
? - not (vahendaja(a_le_coq)) .



Prolog keele elemendid: reegel

Reegel on *tingimuslik Horni lause*.

Reegli üldkuju:

$$\text{järeldus} :- \underbrace{\text{eeldus}_1, \dots, \text{eeldus}_n}_{\text{tingimusosa}}$$

Näide: Kui sajab esmaspäeval **ja...ja** sajab pühapäeval, siis sajab vihma iga päev.

```
sajab(iga_päev) :-  
    sajab('Esmaspäev'),  
    sajab('Teisipäev'),  
    sajab('Kolmapäev'),  
    sajab('Neljapäev'),  
    sajab('Reede'),  
    sajab('Laupäev'),  
    sajab('Pühapäev').
```



Prolog keele elemendid: päring

Päring (*query*)

- Päring defineerib otsingu sihi (*goal*).
- Päring `?- call(Goal)` on semantiliselt samaväärne päringuga
`?- Goal`
- Päringu muutujad väärtustatakse päringu täitmise käigus, kui päringule leidub sobiv unifitseering

Näited:

```
?- call(isa(karl, martin)).  
?- isa(karl, martin).  
?- isa(karl, martin); isa(Keegi, peeter).
```



Prologi (süsteemi-) predikaadid

- Loogika-välised predikaadid:
 - otsingu juhtimise predikaadid (`repeat`, `!`, `fail`,...)
 - sisend-/väljundpredikaadid (`consult`, `reconsult`, `get`, `put`, `write`,...)
 - aritmeetika predikaadid (`div`, `max`, `min`, `>`, `=`, ...)
 - operaatorid
 - predikaadid tööks termidega



Predikaadid tööks termidega: termiteisendused

Näide 1: Muutujate leidmine termis

```
term_variables(+Term, -List).
```

Süntaksi metasümbolid '+' ja '-' näitavad kas tegemist on predikaadi sisend- või väljundparameetriga.

Näide: termist muutujate leidmine

```
?- term_variables(a(X, b(Y, X), Z), L).
```

```
L = [G367, G366, G371]
```

```
X = G367
```

```
Y = G366
```

```
Z = G371
```



Predikaadid tööks stringidega

Stringide teisendamine:

```
string_to_atom(?String, ?Atom)
string_to_list(?String, ?List)
string_length(+String, -Length)
string_concat(?String1, ?String2, ?String3)
sub_string(+String, ?Start, ?Length, ?After, ?Sub)
```

Märkus: metasümbol „?” näitab, et parameeter võib olla nii predikaadi sisend- kui väljundmuutuja



Predikaadid tööks stringidega

- Näide (alamstringi leidmine palindroomis):

```
?- sub_string(seebikiivikaupmees, 6, 4, After, Sub).
```

Mitu sümbolit eelneb alamstringile

Mitu sümbolit on alamstringis

Mitu sümbolit on
pärast alamstringi

```
After = 7
```

```
Sub = ivik
```

Predikaadid kõikide lahendite leidmiseks



```
findall(+Template, +Goal, -Bag)
```

```
bagof(+Template, +Goal, -Bag)
```

Näide: olgu teadmusbaasis faktid:

```
foo(a, b, c).
```

```
foo(a, b, d).
```

```
foo(b, c, e).
```

```
foo(b, c, f).
```

```
foo(c, c, g).
```

Päring `?- bagof(C, foo(A, B, C), Cs).`

annab järgmised alternatiivsed lahendid:

```
A = a, B = b, C = G308, Cs = [c, d] ;
```

```
A = b, B = c, C = G308, Cs = [e, f] ;
```

```
A = c, B = c, C = G308, Cs = [g]
```

sunnime Prologi tagurdamisega leidma uue lahendi

Operaator



- Prologi operaatorid
 - võimaldavad defineerida Prologi notatsioonile lisaks oma notatsiooni
 - aitavad parandada lähtekoodi loetavust
- Näide (aritmeetikatehted kui operaatorid):
 - + ja * prefiks-kujul: $+ (* (2, 3) , * (4, 5)) .$
 - + ja * infiks-kujul: $2 * 3 + 4 * 5 .$
- Kõik süsteemioperaatorid v.a. koma, on defineeritavad operaatoritena.
- Operaatorid kehtivad programmimooduli (käsitleme hiljem) piires, kuid neid saab ka moodulist välja eksportida.



Operaatori deklareerimine

Operaatori deklaratsiooni parameetrid:

- **prioriteet** (1, ..., 1500) – väiksem number annab kõrgema prioriteedi.
- **operaatori tüüp** – määratakse 2 atribuudiga:
 - **assotsiatiivsus** (tehete järjekord avaldise täitmisel)
 - **kuju** (prefiks, infiks, postfiks)
- operaatorite tüübideklaratsioon:

$f x$	mitte-assotsiatiivne	prefiks kuju
$f y$	parem-assotsiatiivne	prefiks kuju
$x f$	mitte-assotsiatiivne	postfiks kuju
$y f$	vasak-assotsiatiivne	postfiks kuju
$x f x$	mitte-assotsiatiivne	infiks kuju
$x f y$	parem-assotsiatiivne	infiks kuju
$y f x$	vasak-assotsiatiivne	infiks kuju



Operaatori deklaratsiooni näide

Defineerime operaatori, mille tähistame sümboliga „#“.

- Kui operaatori # tüüp on $y \rightarrow x$, siis täidetakse operaatori esinemisi avaldises *vasakult paremale* st kehtib :

$$P\#Q\#R\#S = \#(\#(\#(P, Q), R), S)$$

- Kui operaatori # tüüp on $x \rightarrow y$, siis täidetakse # esinemisi avaldises *paremalt vasakule*:

$$P\#Q\#R\#S = \#(P, \#(Q, \#(R, S)))$$



Prologi standardoperaatorid

1200	xfx	-->, :-
1200	fx	:-, ?-
1150	fx	dynamic, discontinuous, initialization, module_transparent, multifile, thread_local, volatile
1100	xfy	;
1050	xfy	->, op*->
1000	xfy	,
900	fy	\+
900	fx	~
700	xfx	<, =, =.., =@=, =:=, =<, ==, =\=, >, >=, @<, @=<, @>, @>=, \=, \==, is
600	xfy	:
500	yfx	+, -, /\, \/, xor
500	fx	?
400	yfx	*, /, //, rdiv, <<, >>, mod, rem
200	xfx	**
200	xfy	^
200	fy	+, -, \



Operaatorite deklareerimine

Deklaratsiooni üldkuju

```
:- op(Priority, Type, Name).
```

Näide:

```
:- op(700, xfy, likes).  
    juhan likes mari.  
    mari likes peeter.  
    jane likes juhan.  
    kati likes X:- not(Y likes X).  
    inimene likes Y:- Y=loom;Y=auto.  
?- inimene likes auto.  
   true  
?- inimene likes ratas.  
   false
```

Prolog keele elemendid: termide võrdus



- **Konstante** sisaldavate termide ja predikaatide võrdlemine unifitseerimisel:

- $\text{arg1} = \text{arg2}$ või $=(\text{arg1}, \text{arg2})$

Näited:

```
?- a = a.
```

```
    true
```

```
?- a = b.
```

```
    false
```

```
?- location(apple, kitchen) = location(apple, kitchen).
```

```
    true
```

```
?- location(apple, kitchen) = location(pear, kitchen).
```

```
    false
```

```
?- a(b, c(d, e(f, g))) = a(b, c(d, e(f, g))).
```

```
    true
```

```
?- a(b, c(d, e(f, g))) = a(b, c(d, e(g, f))).
```

```
    false
```



Prolog keele elemendid: termide võrdus

- *Muutujaid* sisaldavate termide võrdus

```
?- X = a.
```

```
    X = a
```

```
?- 4 = Y.
```

```
    Y = 4
```

```
?- ruum(esik, köök) = ruum(esik, X).
```

```
    X = köök
```

```
?- ruum(X, Y) = ruum(esik, köök).
```

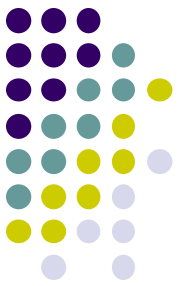
```
    X = esik
```

```
    Y = köök
```



Prolog ütleb mis tingimustel on termid võrdsed

Prolog keele elemendid: termide võrdus



- *Väärtustamata muutujaid* sisaldavate termide võrdus:

```
?- X = Y.
```

```
    X = _01
```

```
    Y = _01
```

```
?- ruum(X, köök) = ruum(Y, köök).
```

```
    X = _01
```

```
    Y = _01
```

```
?- X = Y, Y = hello.
```

```
    X = hello
```

```
    Y = hello
```

```
?- X = Y, a(Z) = a(Y), X = hello.
```

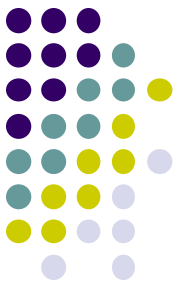
```
    X = hello
```

```
    Y = hello
```

```
    Z = hello
```

Erineva nimega „vabased“ muutujaid interpreteeritakse võrduses kui ühte ja sama muutajat (omistatakse sama töömälu aadress)

Prolog keele elemendid: termide võrdus



- Keerulisemaid näiteid termide võrdusest:

```
?- a(b,X) = a(b,c(Y,e)), Y = hello.  
   X = c(hello, e)  
   Y = hello
```

```
?- food(X,Y)= Z, write(Z),nl,  
X = broccoli,Y = apple, write(Z).  
   food(_01,_02)  
   food(broccoli, apple)  
   X = broccoli  
   Y = apple  
   Z = food(broccoli, apple)
```

- Interpretaator väljastab konstantide väärtused, mille omistamisel muutujatele termide võrdus kehtib.

Prolog keele elemendid: rekursiivne reegel



- Rekursiivses reeglis toimub sama reegli poole pöördumine reegli enda kehast.

Näide:

```
esivanem(Vanem, Noorem) :-
```

```
    vanem(Vanem, Noorem) .
```

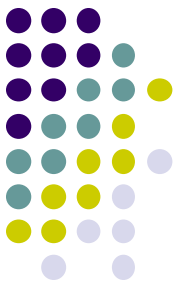
```
esivanem(Vanem, Noorem) :-
```

```
    vanem(Vanem, Vahepealne),
```

```
    esivanem(Vahepealne, Noorem) .
```

rekursiivne pöördumine

Prolog keele elemendid: dünaamiline predikaat



- Dünaamilist predikaati saab luua ja muuta programmi täitmise käigus.
- Dünaamilise predikaadi deklaratsioon (koodis enne predikaadi kasutust):

```
:- dynamic funktor1/aarsus1, ..., funktorN/aarsusn.
```

- Predikaatide loomine `assert`-käsuga:

```
assert (Clause) .
```

```
asserta (Clause) .      % lisatakse mälus samanimeliste faktide ette
```

```
assertz (Clause) .     % lisatakse mälus samanimeliste faktide järele
```

- Dünaamiliste predikaatide kustutamine:

```
retract (Clause) .     % kustutab argumendiga unifitseeruva predikaadi
```

```
retractall (Clause) . % kustutab kõik argumendiga unifitseeruvad predikaadid
```

```
abolish (nimi/aarsus) . % kustutab kõik antud nime ja aarsusega predikaadid
```

Prolog keele elemendid: päringute genereerimine



- Päringus esinev term `Term` konstrueeritakse operaatoriga „`=..`“

Süntaks: `?Term =.. ?List,`

kus argumendi `List` esimene element on loodava termi funktor ja ülejäänud elemendid on loodava termi argumendid. Argumendiks võib olla ka muutuja.

Näited:

```
?- foo(hello, X) =.. List.           % päripidi teisendus s.t.  
   List = [foo, hello, X]          % argumendiks on term  
  
?- Term =.. [baz, foo(1)]          % argumendiks on list  
   Term = baz(foo(1))              % nuripidi
```

Ettevaatust!

- Dünaamiliste predikaatide kasutamine on *riskantne tagurdamisega otsingul*
- vältida dünaamiliste faktide loomist/kustutamist otsingu ajal.