

7. Prologi andmestruktuurid

7.1. Listid

Esitavad järjestatud elementide korteeže

`[a, d, f, [s, f, []], d]`

List unifitseerub

- ühe muutujaga

`List = [a, d, f, [s, f, []], d]`

- Listi erinevaid osi adresseerivate muutujatega, kui on mitte-tühi list

`[Head|Tail]`

kus

`Head` (listi pea) - listi ilmutatult viidatavad esimesed elemendid

`Tail` (listi saba) – ülejäänud listi elemendid

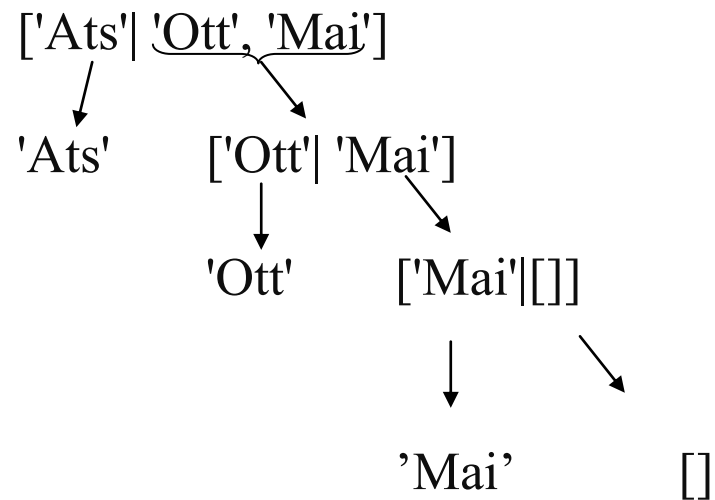
| - eraldussümbol.

Näiteid: `[H|T]`, `[_|T]`, `[H|_]`, `[E11, E12, E13|Tail]` jne.

[Type text]

[Type text]

Listi esitus kahendpuuna:



[Type text]

[Type text]

Loogiline programmeerimine ITI0021

Sügis 2014

J.Vain

Näited:

```
assert(pere(['Ats', 'Mai', 'Ott'])).
```

```
?- pere(Liikmed).
```

```
Liikmed = ['Ats','Mai','Ott']
```

```
?- pere([Pea|Saba]).
```

```
Pea = 'Ats',
```

```
Saba = ['Mai','Ott']
```

```
?- pere([Isa,Ema|Laps]).
```

```
Isa = 'Ats'
```

```
Ema = 'Mai'
```

```
Laps = ['Ott']
```

- Listioperatsioonid

1. Termi konverteerimine listiks ja vastupidi ”= ..”.

Tulemuseks list, mille peaks on predikaadi nimi ja sabaks predikaadi argumendid.

Näide 1:

```
?- isa(juku,peeter) =.. L.  
   L = [isa, juku, peeter]
```

Näide 2:

```
?- isa(Kes, Kellele) =.. L.  
   Kes = _G492  
   Kellele = _G493  
   L = [isa, _G492, _G493]
```

Näide 3:

```
?- [peep, ats, ott, mai] =.. L.  
   L = ['.', peep, [ats, ott, mai]]
```

2. Listide konkatenatsioon (*append*):

```
append([], A, A).  
append([A|B], C, [A|D]) :-  
    append(B, C, D).
```

```
?- append([s,d,f],[e,r,t],A).  
    A = [s, d, f, e, r, t]
```

3. Elemendi eemaldamine listist:

```
remove(_, [], []).  
remove(S, [S|T], L) :-  
    remove(S, T, L), !.  
remove(S, [U|T], [U|L]) :-  
    remove(S, T, L).
```

Näide:

```
?- remove(ats,[reet,pets,ott,ats], Uus_list).  
Uus_list = [reet, pets, ott]
```

4. Listi pikkuse määramine

```
length([], 0).  
length([S|T], M):-  
    length(T, N),  
    M is N + 1.
```

Näide:

```
?- length([e,r,t,w], A).  
    A = 4
```

5. Listi pööramine

```
reverse([], []).  
reverse([S|T], L):-  
    reverse(T, V),  
    append(V, [S], L).
```

Näide:

```
?- reverse([1,2,3,4,5], R_list).  
    R_list = [5,4,3,2,1]
```

6. Elemendi sisalduvuse kontroll listis:

```
member(S, [S|_]) .  
member(S, [_|_]) :-  
    member(S, _) .
```

7. Listi n-da elemendi leidmine:

```
nth_member(S, 1, [S|_]) .  
nth_member(S, N, [_|L]) :-  
    T is N - 1,  
    nth_member(S, T, L) .
```

- Listide rakendusi: üldistatud sorteerimine (*Clocksin&Mellish insertion sort*, 1987)

```
insort([], [], _).
insort([X|L], Sorted_list, Ordering):-
    insort(L, N, Ordering),
    insortx(X, N, Sorted_list, Ordering).
```

```
insortx(X, [A|L], [A|L], Ordering):-
    P =.. [Ordering, A, X], call(P), !.
insortx(X, L, [X|L], Ordering).
```

Sorteerimispredikaadi defineerimine:

- Ordering := < , kui aritmeetiline järjestamine
- Ordering := aless , kui alfabeetiline järjestamine, kus

```
aless(X, Y):-
    name(X, L), name(Y, M), alessx(L, M).
```

```
alessx([], [_|_]).
alessx([X|_], [Y|_]):- X < Y.
alessx([H|Q], [H|S]):- alessx(Q, S).
```

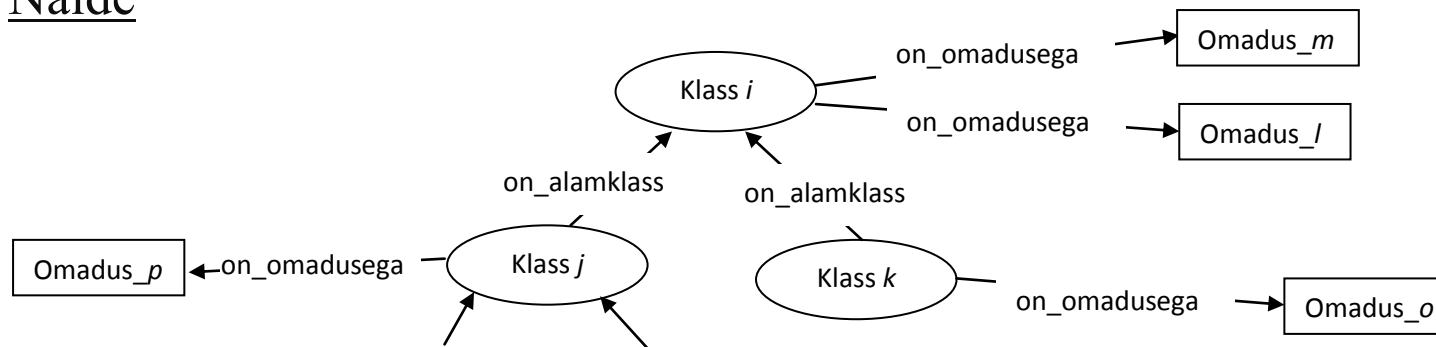

7.2 Semantilised võrgud

Semantiline võrk (SV) on diagramm, mis esitab objekte, nende omadusi ja objektide vahelisi seoseid. SV annab mõistete konteksti ja aitab selgitada valdkonna mõistete tähendust.

SV graafiline esitus:

- tipud: objektid ja nende omadused
- kaared (suunatud): näitab objektidevahelisi seoseid ja objektide seost omadustega

Näide



[Type text]

[Type text]

Loogiline programmeerimine ITI0021

Sügis 2014

J.Vain

Näide:

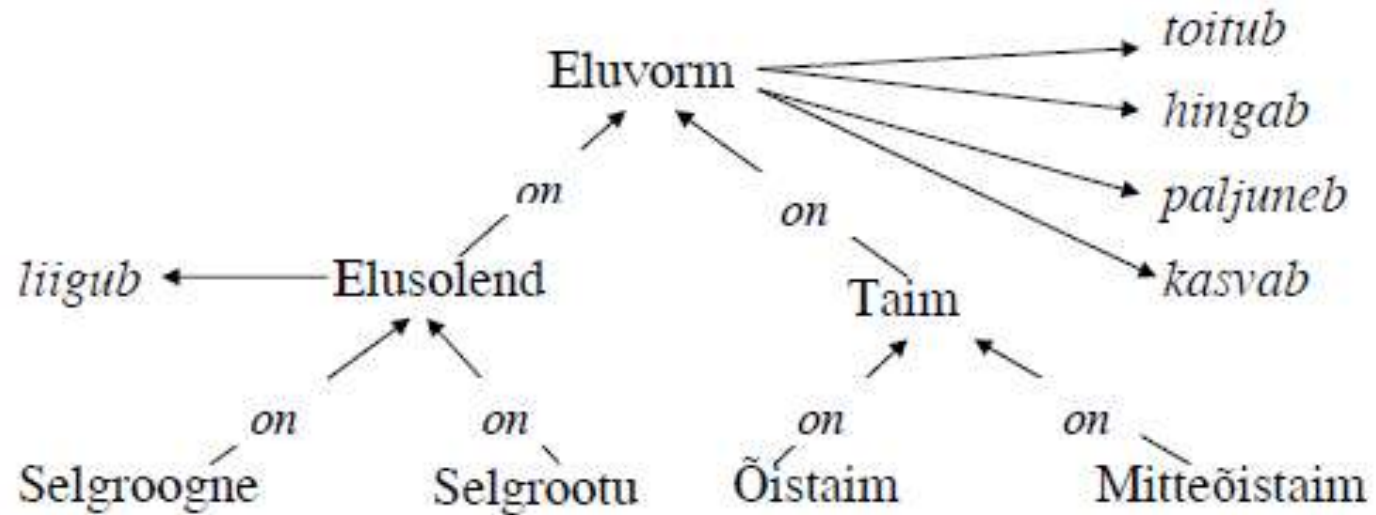
Elusorganismide klassifitseerimine

Klassid:

- elusolend/taim;
- selgroogne/ selgrootu;
- õistaim/mitteõistaim.

Reeglid:

- eluvorm on kas *elusolend* või *taim*;
- eluvorm *toitub, hingab, paljuneb* ja *kasvab*;
- elusolend on *selgroogne* või *selgrootu*;
- elusolend *liigub*;
- taimed on kas õistaimed või mitteõistaimed.



- Predikaat "on" tähistab relatsiooni "on alamklass".
- Predikaadid "toitub", "hingab", "paljuneb", "kasvab" on rakendatavad objektidele, millest vastavad nooled lähtuvad (omaduste pärimine).

NB! Omadused, mis kehtivad objektide kohta, peavad kehtima ka kõigi tema alamobjektide kohta (, mis seotud predikaadi "on" ("is_a") kaudu).

```
% Eluslooduse klassifikaator
  is_a(elusolend, eluvorm) .
  is_a(taim, eluvorm) .
  is_a(selgroogne, elusolend) .
  is_a(lehm, selgroogne) .
  is_a(selgrootu, elusolend) .
  is_a(õistaim, taim) .
  is_a(mitteõistaim, taim) .
  is_a(kapsas, õistaim) .
  is_a(maasi, lehm) .

  toitub(eluvorm) .
  hingab(eluvorm) .
  paljuneb(eluvorm) .
  kasvab(eluvorm) .
  liigub(elusolend) .
  eats(lehm, kapsas) .
  eats(selgrootu, taim) .
```

[Type text]

[Type text]

Loogiline programmeerimine ITI0021

Sügis 2014

J.Vain

Abireeglid:

```
%alamhulk(X,X):- !.
```

```
alamhulk(X,Y):- is_a(X,Y), !.
```

```
alamhulk(X,Y):- is_a(W,Y), alamhulk(X,W).
```

Reeglid:

```
hingab(M):- alamhulk(M,N), hingab(N).
```

```
liigub(P):- alamhulk(P,Q), liigub(Q).
```

```
toitub(R):- alamhulk(R,S), toitub(S).
```

```
kasvab(V):- alamhulk(V,W), kasvab(W).
```

```
paljuneb(X):- alamhulk(X,Y), paljuneb(Y).
```

```
eats(X,Y):- alamhulk(X,V), alamhulk(Y,W), eats(V,W).
```

7.3 Freimid

Freime kasutatakse andmestruktuuride üldistusena, kus struktuur on paljudel andmetel ühine, kuid elemendid on erinevat tüüpi või tüüp on täpsustamata.

Freim – abstraktne skelett, milles on vahetatavate elementide tarvis lahtrid (*slots*).

Freimi iga lahter võib olla omakorda freim, kusjuures kõik lahtrite omadused päritakse tema alamfreimide poolt.

Slot	Value	Type
ALEX	_	(This Frame)
NAME	Alex	(key value)
IS_A	Boy	(parent frame)
SEX	Male	(inheritance value)
AGE	IF-NEEDED: Subtract(current,BIRTHDATE);	(procedural attachment)
HOME	100 Main St.	(instance value)
BIRTHDATE	8/4/2000	(instance value)
FAVORITE_FOOD	Spaghetti	(instance value)
CLIMBS	Trees	(instance value)
BODY_TYPE	Wiry	(instance value)
NUM_LEGS	1	(exception)

[Type text]

[Type text]

Loogiline programmeerimine ITI0021
Sügis 2014
J.Vain

Näide1:

```
% on_eriliik(See, Selle)
    on_eriliik(romb, nelinurk).
    on_eriliik(ruut, romb).
% Rombile iseloomulikud lahtri väärtused
    romb(kyljed, vordsed).
    romb(nurgad, ebavordsed).
% Ruudule iseloomulikud lahtri väärtused
    ruut(symmeetriline, 4).

% Pärimisreegel freimi lahtrite väärtustamiseks
    ruut(+Atribuut, -Vaartus):-
        on_eriliik(ruut, Freim),           % Leiab esivanemklassi nime
                                           % millel on viidatud atribuut
Subgoal =.. [Freim,Atribuut,Vaartus],
    Subgoal.                               % Saab atribuudi päritud väärtuse

?- ruut(kyljed, Missugused).
    Missugused = vordsed
```

[Type text]

[Type text]

Loogiline programmeerimine ITI0021
Sügis 2014
J.Vain

Näide 2: (sobib sügavate pärimispuude korral)

```
on_eriliik_reegel(Esivanem) :-  
    repeat,  
    jooksev(Klass),  
    on_eriliik(Klass, Esivanem),  
    retract(jooksev(Klass)),  
    assertz(jooksev(Esivanem)).
```

```
% on_eriliik(See, Selle)  
on_eriliik(romb, nelinurk).  
on_eriliik(ruut, romb).  
on_eriliik(nelinurk, hulknurk).
```

```
hulknurk(koosneb, [loigud, nurgad]).
```

```
% Rombile iseloomulikud lahtri väärtused  
romb(kyljed, vordsed).  
romb(nurgad, ebavordsed).
```

```
% Ruudule iseloomulikud lahtri väärtused  
ruut(symmeetriline, 4).
```


[Type text]

[Type text]

Loogiline programmeerimine ITI0021
Sügis 2014
J.Vain

```
% Pärimisreegel objekti omaduste väärtustamiseks
ruut(+Atribuut, -Vaartus):-
    assert(jooksev(ruut)),
    on_eriliik_reegel(Freim),          % Leiab esivanemklassi nime
    Subgoal =.. [Freim, Atribuut,Vaartus],
                                     % Saab atribuudi päritud väärtuse
    Subgoal,
    retractall(jooksev(_)).

?- ruut(kyljed, Missugused).
Missugused = vordsed
```

7.4 "If... then ..." reeglid ekspertsüsteemides

Võimaldavad spetsifitseerida põhjus-tagajärg seoseid ning (läbi transitiivsuse) pikemaid põhjuslikkuse ahelaid.

Näide:

Valdkonnas "Auto diagnostika" kirjeldame auto seisundit faktidega:

```
mootor(ei_käivitu) .
```

```
...
```

```
tuled_on(tuhmid) .
```

```
...
```

ja järelduste tegemist rikke põhjustest reeglitega:

```
aku_on(tyhi) :-
```

```
    mootor(ei_käivitu) ,
```

```
    tuled_on(tuhmid) .
```

```
aku_on(täis) :-
```

```
    mootor(käivitub) , tuled_on(heledad) .
```

```
?-aku(Seisund) .
```

[Type text]

[Type text]

Loogiline programmeerimine ITI0021
Sügis 2014
J.Vain

Ülesanne:

Kirjutada Prologis reeglid:

- "Kui raadiojaam ei ole korralikult kuuldav või häälestusindikaator vilgub, jätkka häälestusnupu keeramist".
- "Kui raadiojaam on hästi kuuldav ja häälestusindikaator põleb pidevalt, lõpeta nupu keeramine".