# Loeng 7: Hulgateooria ja algebra mõistete programmeerimine Prologis

J.Vain

Loogiline programmeerimine ITI0211

Sügis 2021

# Loengu eesmärk

- Meelde tuletada mõningaid hulgateooria ja algebra põhimõisteid ning õppida neid loogilise programmi kujul esitama.
- Näidata nende mõistete rakendusvõimalusi praktiliste ülesannet lahendamisel

# Loengu plaan

- Hulgateooria ja algebra mõistete esitamine Prologis:
  - Hulk
  - Relatsioon
  - Relatsiooni transitiivsus
  - Transitiivne sulund
  - Ekvivalents
  - Faktorruum
  - Meetrika
  - Semantilise võrgu mõistete semantiline kaugus
- Matemaatiliste konstruktsioonide kasutamine programmeerimis-ülesannete lahendamisel

# Hulga esitamine Prologis

Eksplitsiitne definitsioon:

1. List, mille elementideks on hulga elemendid

```
hulk_a([el1,el2,...,eln]).
```

2. Faktidena, kus funktor on hulga nimi ja argument hulga element

```
hulk_A(el1).
hulk_A(el2).
...
hulk_A(eln).
```

3. Tüüpi esitava faktina `hulk/2`, kus funktoriks on tüübi nimi, esimene argument on hulga nimi ja teine element on hulga element

```
hulk(hulk_A, element).
```

# Hulga esitamine Prologis: implitsiitne definitsioon (1):

4. Esitus reegliga, mille parameetrid on baashulk ja seda kitsendav tingimus:

```
set(BaasHulk, Tingimus, Element):-
        call(BaasHulk,Element),
        call(Tingimus,Element).
```

Näide: baashulga paarisarvuliste elementide leidmine

```
even(X):-                    % baashulk on täisarvud  (vaikimis)
     0 is X rem 2.           % rem – tagastab jagamise jäägi
```

Päring

```
?- set(hulk_A, even, Element).
```

Tagastab hulga `hulk_A` paarisarvulised elemendid.

# Hulga esitamine Prologis: implitsiitne definitsioon (2):

- Hulga kõigi elementide genereerimine kasutades süsteemipredikaati
  `findall(+Template, :Goal, -Bag) :`

```
set(+Superset,+Constraint,-Set):-
      findall(Element,(member(Element,Superset),
      call(Constraint(Element))),
      Set).
```

Päring tagastab hulga `Superset` niisuguste elementide listi, mis rahuldavad muutujas `Constraint` esitatud kitsendust.

# Hulga esitamine Prologis: implitsiitne definitsioon (3):

- Induktiivne definitsioon

  Näide: naturaalarvude hulk:

  - Naturaalarvude hulga elementi ära tundev predikaat:

```
natural(0):- !.          % baas
natural(X):-             % induktsiooni samm
          X > 0,
          XX is X-1,
          natural(XX).
```

  - Naturaalarvude hulga elemente genereeriv predikaat:

```
nat(0).
nat(X1):-
        nat(X), X1 is X+1.
```

# Relatsioonide defineerimine Prologis: eksplitsiitne definitsioon

- Esitus relatsiooni ekstensiooni kaudu
  - Relatsiooni ekstensiooni moodustavad relatsiooni kandjahulga elementide mitmikud, mille vahel relatsioon on defineeritud (paarid, kolmikud …)

Näide 1: binaarne relatsioon `connected/2`

```
connected('Tallinn', 'Keila').
connected('Tallinn', 'Saue').
connected('Keila', 'Saue').
```

Näide 2: ternaarne relatsioon `connected/3`

```
connected(['Tallinn','Keila','Saue']).
```

Näide 3: *n*-aarse relatsiooni esitus üldistava predikaadiga `relation/2`

```
relation(connected, ['Tallinn'| Linnad]).
```

Tüübi predikaat — Relatsiooni nimi — Relatsioonis olevad elemendid

# Relatsioonide defineerimine Prologis: implitsiitne definitsioon

Analoogne hulga defineerimisele - üldisema relatsiooni ja relatsiooni baashulka kitsendava predikaadi kaudu.

Näide (siin eeldame, et relatsioon on elementide suhtes sümmeetriline st kitsendus `Constraint` kehtib kõigile relatsioonis olevatele kandjahulga elementidele):

```
relation(Rel,Arguments,Constraint):-
      alamklass(Rel,SuperRel),                  % Leiame esivanemrelatsiooni
      TermSR =..[SuperRel|Arguments],           % Moodustame sellega päringu
      TermSR,
      forall(member(Arg,Arguments),
            (TermCons =..[Constraint,Arg],      % Kas päringust saadud argumendi
             TermCons)).                        % väärtused rahuldavad kitsendust
```

Päring

```
?- relation(suurem,[2,4],even).          % Kahe paarisarvu „>“ - seos
```

# Relatsioonide omadused: **transitiivsus**

- Relatsioon $R$ on *transitiivne*, kui mistahes $x$, $y$ ja $z$ korral

$$xRy \wedge yRz \Rightarrow xRz,$$

kus $(x, y), (y, z), (x, z) \in \underbrace{[\![ R ]\!]}_{\text{Relatsiooni } R \text{ interpretatsioon}}$

- Relatsiooni *aste*:
  - $xR^1y = xRy$ % 1. aste
  - $xR^iy$ ja $yRz \Rightarrow x\,R^{i+1}z$ % $i$+1 - aste

- Binaarse relatsiooni $R$ *transitiivne sulund* $R^+$ on relatsiooni $R$ kõigi astmete ühendiga määratud binaarne relatsioon $R^+$ , kus

$$R^+: xR^+y = \cup_i\, xR^iy.$$

# Transitiivne sulundi leidmine

```
transitive_closure(Rel):-          % relatsiooni  1. astme leidmine
     call(Rel(X,Y)),
     assertz(closure(1,X,Y)),      % closure/3 faktide loomine
     fail.
transitive_closure(_):-            % i+1. astme leidmine
     call(closure(I,A,B)),
     call(closure(1,B,C)),
     I1 is I+1,
     assertz(closure(I1,A,C)),
     fail.
transitive_closure(_).
```

# Relatsioonide omadused: ekvivalents

Relatsiooni $\sim$ nimetatakse *ekvivalentsirelatsiooniks* (-s*eoseks*), kui

$$\forall s, s', s'' \in [\![ \sim ]\!] \text{ korral kehtib}$$

- Refleksiivsus: $s \sim s$
- Sümmeetria: $s \sim s' \Rightarrow s' \sim s$
- Transitiivsus: $s \sim s' \wedge s' \sim s'' \Rightarrow s \sim s''$

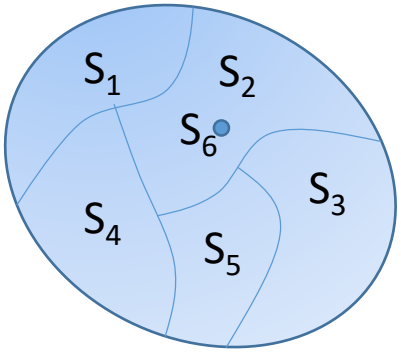
- Näide:

  Olgu Eesti elanike hulk $S$ ja tähistagu $s \sim s'$, seost inimeste vahel, kes on sündinud samal kümnendil. Siis seos "~" on ekvivalentsisuhe hulgal $S$.

# Ekvivalentsiklass

Ekvivalentsisuhe tükeldab hulga $S$, millel ta on defineeritud, *ekvivalentsiklassideks*

$$S = \cup_i S_i \text{ , nii et } \forall s, s' : s, s' \in S_i \Leftrightarrow s \sim s'$$




**Omadused:**

- Ekvivalentsiklassid katavad kogu hulga
- Ekvivalentsiklassidel puudub ühisosa

- **Näide:**

  Ekvivalentsiklassideks Eesti elanike hulgal on vanuserühmad sünniaastaga samal kümnendil.

# Faktorruum

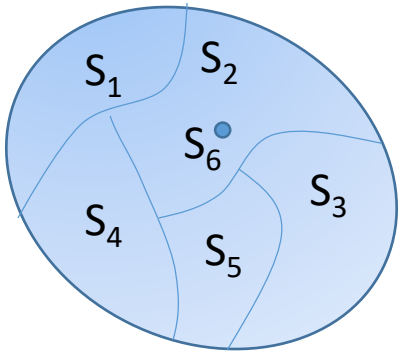
Olgu ~ ekvivalentsisuhe (-relatsioon), siis

$$S/_{\sim} = \{ S_i \}$$

tähistab faktorruumi s.o. hulka, mille elementideks on ekvivalentsiklassid.

<u>Näide:</u>

$$S/_{\sim} = \{ S_i \}, i=1,6$$



# Meetrika objektide hulgal

- Vaatame ekvivalentsi tüübiga objektide hulgal.
- Olgu
  - objektidel määratud *tüübid* (lubatud väärtused)
  - tüüpidel on defineeritud *meetrika.*

- Üldiselt, *meetrika d* hulgal $Y$ on kujutus $d$: $Y \times Y \rightarrow \mathbb{R}^+$, kus kehtivad seosed
  - $\forall x, y : \in Y, \quad d(x, y) \geq 0 \quad$ ja $\quad d(x, x) = 0$ % Obj. kaugus endast on 0
  - $\forall x, y : \in Y, \quad d(x, y) = d(y, x)$ % Kaugus ei olene suunast
  - $\forall x, y, z : \in Y, \quad d(x, y) + d(y, z) \geq d(x, z)$ % Kaugus kahe obj. vahel ei % ole suurem kaugusest läbi % kolmanda objekti

# Meetrika (tüübiga) objektide hulgal

- Kui loenduval hulgal on defineeritud (sümmetriline) binaarne seos $\mathcal{R}$, siis selle kaudu saab defineerida *diskreetse meetrika*.
- Kauguse $d(x, y)$ määrab siis selle seose $\mathcal{R}$ *minimaalne aste*, s.t.

$$d(x, y) = i \ , \ \ if \qquad (x, y), (y, x) \in [\![ \ min \mathcal{R}^i \ ]\!] \qquad (*)$$

*NB! Semantilise kauguse määramisel semantilise võrgu mõistete vahel tingimust* (*) ei saa otseselt kasutada juhul, kui $\mathcal{R}$ ei ole sümmeetriline s.t. kui

$$d(x, y) \neq d(y, x)$$

Seega ebasümmeetrilise seose korral sõltub kaugus suunast.

# Meetrika (tüübiga) objektide hulgal

Orientatsiooniga seoste korral, näiteks *is_a* seos semantilisel võrgul, saab mõistete vahelise kauguse defineerida näiteks:

- $d(x, y) = min(d(\overrightarrow{x, y}), d(\overrightarrow{y, x}))$.

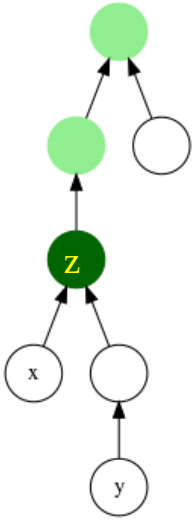
või

- $d(x, y) = \begin{cases} i + j \quad if \ \exists\, z.\ (x, z) \in [\![ min_i\ is\_a^i ]\!] \text{ and } (y, z) \in [\![ min_j\ is\_a^j ]\!] \\ \\ sup\ d(x, y), \ \text{otherwise} \end{cases}$

$sup\ d(.)$ – kauguse funktsooni $d$ ülemine tõke

# Semantiline kaugus: näide

- Semantiline kaugus on kasutusel ontoloogia objektide, geneetiliste objektide jm võrdlemisel.
- Kauguse defineerimiseks semantilisel võrgul kasutame mõistet *lowest common subsumer* (*lcs*)
- Näide: tippude $x$ ja $y$ puhul on $lcs(x,y) = z$




- Kehtib seos: $d^{sem}(x,y) = d(x, lcs(x,y)) + d(y, lcs(x,y))$

- Prologi programmina:

```
sem_distance(+X,+Y,-D):-
      findall(D,(closure(Dx,X,LCS),closure(Dy,Y,LCS),
                  D is Dx+Dy), Ds),
      sort(Ds,[D|_])).
```

# Objektide hulga esitus Prologis

- Objektide hulk → faktide hulk
- Objekti atribuudi väärtustus → fakti argumendi väärtus

Märkus: ka objekti nime võib käsitleda kui tema klassi üht atribuuti.

```
klassi_NIMI(Atrib_1, ... , Atrib_n).
```

- Väärtustatud parameetritega faktid esitavad selle klassi karakteristliku predikaadi interpretatsiooni hulka ehk ekstensiooni.
- Näide: inimeste hulk

```
% inimene(Nimi, Sünniaasta, Sugu, Silmade_värv, …).
inimene('John Smith', 1990, male, gray, …).
…
inimene('Betty Joung', 1998, female, brown, …).
```

# Tüübid

- Objekti tüüp on määratud kitsendustega tema atribuutide tüüpide ristkorrutisel:

```
TYYP(TYYBI_NIMI, ATRIB1_TYYP, ..., ATRIBnTYYP).
```

Baastüüp:

```
TYYBI_NIMI × ATRIB1_TYYP × ... × ATRIBnTYYP
```

Näide (tüübi kitsenduste programmeerimine):

Selgitus: Atribuutide tüübid antakse reegli päises sisendparameetritena, tüübikitsendused reegli kehas.

```
TYYP(inimene,Nimi,Sünniaasta,Sugu,Silmade_värv, ...):-
     (Sugu=male; Sugu=female),
     (Sugu -> nimekitsendused(Nimi)),
     sünniaastakitsendused(Sünniaasta),
                    …  .
```

# Tüübid

- Veel tüübi defineerimise võimalusi Prologis :
  - elementaartüüp:
    `tyyp(tüübi_nimi,tüübi_väärtused_listina).`
  - kõige üldisemaks tüübikonstruktoriks on ristkorrutis,
  - tingimuslikud kitsendused atribuudi tüüpidel, näiteks:

$$Constr_k(Attr_i) \text{ -> } Constr_l(Attr_j)$$

Näide:

```
tyyp(obj_tüübinimi, Atr1, Atr2,..., Atrn):-
        tyyp(Atr1_id,Type_1), member(Atr1,Type_1),
        …
        tyyp(Atrn_id,Type_n), member(Atrn,Type_n),
        …
        Constr_k(Attr_i) -> Constr_n(Attr_n).
```

# Meetrikaga tüübid

- Kui atribuudil on numbriline tüüp, siis on meetrika defineeritud selle tüübi endaga
- Kui mittenumbriline loenduv tüüp, siis defineerime meetrika selle tüübi määramispiirkonnal:
  - Võrreldavate väärtuste vahel defineerime (osalise)järjestussuhte predikaadiga `meetrika`:

```
meetrika(tüübi_nimi, väiksem_väärtus, suurem_väärtus).
```

Näide (meetrika hulgal `Inimesed`):

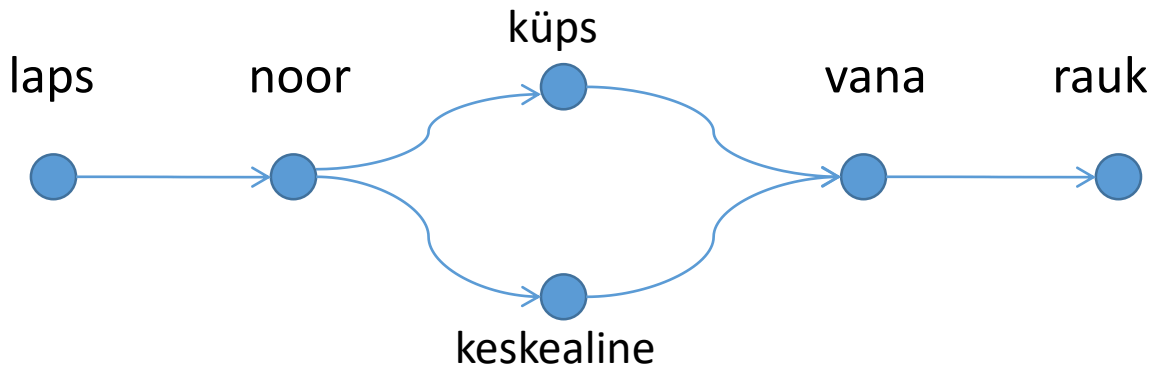
Olgu igal elemendil 2 atribuuti: nimi ja vanus

Prologis:

```
inimene(Nimi, Vanus).
type(vanus,[laps,noor,kyps,keskealine,vana,rauk]).
```

% so. loendav defefinitsioon

Defineerime tüübil järjestuse (võimalik ka osaline järjestus):




- Järjestussuhte esitus Prologis :

  Fakti mall: `meetrika(vanus, Noorem, Vanem).`

  `meetrika(vanus, laps, noor).`
  `meetrika(vanus, noor, kyps).`
  `meetrika(vanus, kyps, vana).` jne

Semantiline kaugus: $d(o_i, o_j) = k\text{-}1$, kus $k$ on relatsiooni `meetrika/2` *vähim aste*, nii et $\langle o_i, o_j \rangle \in meetrika^k$

# Näide

- Ühe vanusegrupi moodustavad inimesed, kelle sünniaasta erinevus on $\leq k$ aastat. Siis
  - vanusegrupid moodustavad ekvivalentsiklassid
  - vanuse astmete järjestus annab semantilise kauguse arvutamiseks vajaliku meetrika
  - kaugus meetrikal on vanuse erinevus
  - Ekvivalentsi seos on määratud tingimusega, et vanuse erinevus on $\leq k$ aastat ja kui objekt kuulub ühte ekvivalentsi klassi, siis ta ei kuulu samal ajal teise klassi (mittelõikuvuse tingimus).
  - Erinevad klasterdusalgoritmid annavad hulgal erinevad tükeldused, mis sõltuvad tsentroididest ja nende valimise järjekorrast.

# Näiteid Prologi predikaatidest

```
% Transitive closure
%-----------------------------------------------------
% TEST1: transitive_closure(pr).
%-----------------------------------------------------
transitive_closure(Relation):-
        Clause =..[Relation, X, Y],
        call(Clause),
        assertz(closure(1, X, Y)), fail.
transitive_closure(_):-
        call( closure(N, A, B)),
        call( closure(1, B, C)),
        not (closure(_, A, C)), % kas juba olemas niisugune fakt?
        N1 is N + 1,
        assertz(closure(N1, A, C)), fail.
transitive_closure(_).

   % Katsehulk binaarseid predikaate
        pr(s,f).
        pr(d,f).
        pr(f,g).
        pr(r,s).
```

# Meetrikaga transitiivne sulund (min closure)

```
m_transitive_closure(Relation):-
  Clause =..[Relation,D,X,Y],
  call(Clause),
  assertz(closure(D,X,Y)),fail.
m_transitive_closure(_):-
  call(closure(D1,A,B)),
  call(closure(D2,B,C)),
  D is D1 + D2,
  m_test(A,C,D),
  assertz(closure(D,A,C)), fail.
m_transitive_closure(_):- !,      listing(closure) .

m_test(A,A,_):- !, fail.                % välistame tsüklid
m_test(A,C,_):- not closure(_,A,C),!.  % kas olemas niisugune fakt?
m_test(A,C,D):-
  closure(DD,A,C),      % kas paar esineb juba mõnes astmes?
  D < DD,               % kas leitud kaugus < teadaolev?
  retract(closure(DD,A,C)).
```

```
%=================================================
% Reegel klasterdab hulga ekvivalentsiklassideks – faktid eq_class/2
% eq_class(KLASSI NIMI, BAASHULGA_NIMI(ELEMENDI NIMI, ATRIBUUT_mille pohjal)).
%=================================================
equivalence_class(Hulk, ArityNV, Nr, Distants):-
   functor(TermV, Hulk, ArityNV),          % moodustab termi nimega Hulk aarsusega ArityNV
   call(TermV),                            % omistab selle muutujale TermV
   arg(Nr, TermV, Val),                    % leiab termi TermV Nr-nda argumendi väärtuse
   assert(eq_class(Val,TermV)),
   functor(TermV1, Hulk, ArityNV),         % moodustab termi nimega Hulk aarsusega ArityNV
   call(TermV1),
   arg(Nr, TermV1, Val1),          % leiab termi TermV1 Nr-nda argumendi väärtuse
   distants(Val, Val1, D),         % võrdleb, kas tegemist on samasse klassi
   abs(D, D1), D1 =< Distants,     % kuuluvate elementidega
   assert(eq_class(Val, TermV1)),
   fail.

% Päring: ?- equivalence_class(Hulga_nimi, AtribuutideArv, MitmesAtrib, MaxKaugusTsentroidist).
% TEST: equivalence_class(inimene, 2, 2, 1).
```

# Veel näiteid Prologi predikaatidest

```
%-----------------------------------------------
% Kauguse leidmine etteantud meetrikas
% TEST: distants(+noor, +rauk, -VanuseErinevus).
%-----------------------------------------------
distants(Obj1, Obj2, Val):-       % Kui transit. sulund olemas
     (closure(Val, Obj1, Obj2);   % kui paar (Obj1,Obj2) on sulundis
     (closure(Val1,Obj2,Obj1),    % kui paar (Obj2,Obj1) on sulundis
                Val is -Val1;
     Val=999), !.                  % Kui tr. sulundis ei leidu paari
distants(Obj1,Obj2, Val):-        % Kui sulund on veel leidmata
     m_transitive_closure(jarjestus), %genereeri tr sulund
     distants(Obj1,Obj2, Val),!. % Kutsu välja reegli 1. alternatiiv
```