

Loogiline programmeerimine

Loeng 5: Hulgateooria ja algebra mõistete
programmeerimine Prologis

Loengu plaan

- ▶ Hulgateooria ja algebra mõistete esitamine Prologis:
 - Hulk
 - Relatsioon
 - Relatsiooni transitiivsus
 - Transitiivne sulund
 - Ekvivalents
 - Faktorruum
 - Meetrika
 - Semantilise võrgu mõistete semantiline kaugus
- ▶ Teooria mõistete kasutamine ülesannete lahendamisel
- ▶ Takeaway: arusaam teooria kasulikkusest programmeerimises

Hulk

Eksplitsiitne defineerimine:

- ▶ List, mille elementideks on hulga elemendid

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

- ▶ Faktidena, kus funktoriks hulga nimi ja argumendiks element

```
hulk_a(el1).
```

```
hulk_a(el2).
```

```
...
```

```
hulk_a(eln).
```

- ▶ Üldistatud faktidena `hulk/2`, kus `ka` hulga nimi on argument

```
hulk(hulk_a, element).
```

Hulk

▶ Implitsiitne defineerimine (1):

- Baashulga ja kitsendava tingimusega:

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

```
even(X):-    0 is X rem 2.        % rem - jääk
```

Päring

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

genereerib hulga hulk_a paarisarvulised elemendid.

Hulk

- ▶ Implitsiitne defineerimine (2):
 - Baashulga ja kitsendava tingimusega:

```
set (Set) :-  
    findall (Element,  
            (member (Element, Superset), constraint (Element)),  
            Set).
```

vt. findall(+Template, :Goal, -Bag)

Hulk

- ▶ Implitsiitne defineerimine (3):

- Tingimus induktsiooniga:

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

- Genereeriva funktsiooniga:

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

Relatsioon

- ▶ Eksplitsiitne defineerimine (loendame relatsiooni ekstensiooni elemendid – paarid, kolmikud jne)

Näide 1: (binaarne relatsioon)

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

Näide 2: (3ne relatsioon)

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

Näide 2 (üldistatud esitus):

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

Relatsioon

Implitsiitne defineerimine (abstraktse relatsiooni ja selle ekstensiooni kitsendavate predikaatide kaudu):

Näide 1

```
relation(Rel, Arguments, Constraint) :-  
    alamklass(Rel, SuperRel),  
    TermSR = ..[SuperRel | Arguments],  
    TermSR,  
    forall(member(Arg, Arguments),  
            (TermCons = ..[Constraint, Arg],  
             TermCons)).
```

Päring

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


Relatsiooni transitiivsus

- ▶ Relatsioon R on transitiivne, kui mistahes x, y ja z korral
 xRy ja yRz järeljub xRz ,
kus $\{(x, y), (y, z), (x, z)\} \subseteq [R/]$
- ▶ Relatsiooni aste:
 - $xR^1y = xRy$ % 1. aste
 - xR^iy ja $yRz \Rightarrow xR^{i+1}z$ % $i+1$ - aste
- ▶ Binaarse relatsiooni R transitiivne sulund R^+ on relatsiooni R kõigi astmete ühendiga määratud binaarne relatsioon R^+ , kus
$$R^+: xR^+y = \bigcup_i xR^iy.$$

Transitiivne sulundi leidmine

```
transitive_closure(Rel):-          % 1. astme leidmine
    call(Rel,X,Y),
    assertz(closure(1,X,Y)),
    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(_).
```

Ekvivalentsi relatsioon

Relatsiooni \sim nimetatakse *ekvivalentsirelatsiooniks* (–seoseks), kui $\forall s, s', s'' \in [|\sim|]$ 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 S Eesti elanike hulk ja tähistagu $s \sim s'$, seost inimeste vahel, kes on sündinud samal aastal. Siis seos “ \sim ” on ekvivalentsisuhe hulgal S .

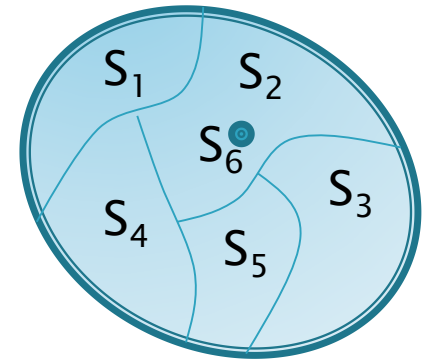
Ekvivalentsiklass

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 sünniaasta järgi.

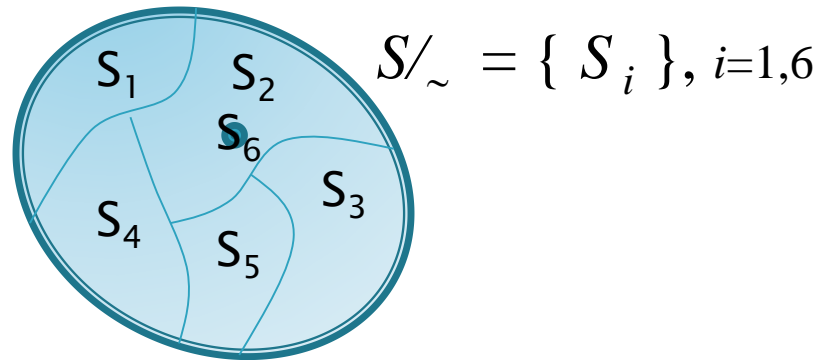
Faktorruum

Olgu \sim ekvivalentsuhe (relatsioon), siis

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

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

Näide:



Meetrika objektide hulgal

- ▶ Vaatame ekvivalentsisuhet tüübiga objektide hulgal.
- ▶ Olgu
 - objektidel määratud tüübid (lubatud väärtused) ja
 - tüüpidel on defineeritud meetrika.
- ▶ Üldiselt, meetrika d hulgal Y on kujutus $d: Y \times Y \rightarrow R$, kus
 - $\forall x, y : \in Y, \quad d(x, y) \geq 0$ ja $d(x, x) = 0$
 - $\forall x, y : \in Y, \quad d(x, y) = d(y, x)$
 - $\forall x, y, z : \in Y, \quad d(x, y) + d(y, z) \geq d(x, z)$

Meetrika tüübiga objektide hulgal

- ▶ Kui loenduv hulgal on defineeritud (sümmeetriline) binaarne seos \mathcal{R} , siis selle kaudu saab defineerida diskreetse meetrika.
- ▶ Kauguse $d(x, y)$ määrab siis selle seose \mathcal{R} aste, st

$$d(x, y) = \mathcal{R}^i,$$

kus $(x, y) \in [|\mathcal{R}^i|]$

Objektide hulga esitus Prologis

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

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

- ▶ Väärtustatud parameetritega faktid esitavad klassi karakteristikliku 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(HULGA_NIMI, ATRIB1_TYYP, ..., ATRIBnTYYP).
```

Näide

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

Tüübid

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

$$\text{Constr}_k(\text{Attr}_i) \rightarrow \text{Constr}_l(\text{Attr}_j)$$

Näide (atribuudi tüüp Type_i on eksplitsiitne hulk):

```
tyyp(obj_tüübinimi, Attr1, Attr2, ..., Attrn) :-  
    member(Attr1, Type_1),  
    ...  
    member(Attrn, Type_n),  
    Constr_k(Attr_i) -> Constr_n(Attr_n).
```

Meetrikaga tüübid

- ▶ Kui numbriline tüüp, siis on meetrika defineeritud tüübi endaga
- ▶ Kui mittenumbriline loenduv tüüp, siis defineerime meetrika selle tüübi määramispiirkonnal
 - Võrreldavate väärtuste vahel defineerime (osalise)järjestussuhte predikaadiga `meetrika(tüübi_nimi, väiksem_väärtus, suurem_väärtus)`.

Näide (meetrika hulgal `Inimesed`):

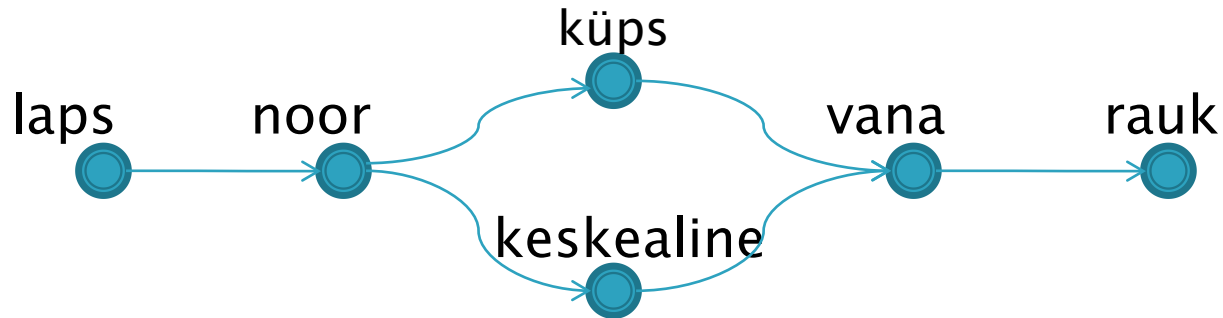
Olgu igal elemendil 2 atribuuti: nimi ja vanus

Prologis:

```
inimene(Nimi, Vanus).
```

```
type(vanus, [laps, noor, kyps, keskealine, vana, rauk]). % Loendav  
def
```

Defineerime tüübil järjestuse (võimalik ka osaline järjestus):



► Prologis Järjestussuhte mall:

```
% meetrika(Väiksem, Suurem).  
meetrika(laps, noor).  
meetrika(noor, kups).  
meetrika(kups, vana). jne
```

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

Näide

- ▶ Ühe vanusegrupi moodustavad inimesed, kelle sünniaasta erinevus on $\leq k$ aastat.
 - ekvivalentsiklass on vanusegrupp
 - meetrika on vanus
 - kaugus meetrikal on vanuse erinevus
 - ekvivalentsiseos 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).
 - tehisintellektis klasterdusalgoritmid.
 - klastrid sõltuvad tsentroidide loomise järjestusest

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(Closure),
    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.
m_test(A,C,_):- not closure(_,A,C),!. % kas olemas niisugune fakt?
m_test(A,C,D):-
    closure(DD,A,C), % kas paar esineb madalamas 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, Distant):-
    functor(TermV, Hulk, ArityNV),    % moodustab termi nimega Hulk aarsusega ArityNV
    call(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 =< Distant,      % kuuluvate elementidega
    assert(eq_class(Val, TermV1)),
    fail.

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

```

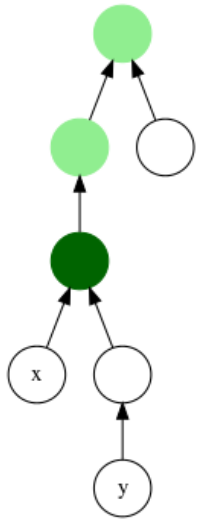

Näiteid Prologi predikaatidest

```
%-----  
% Kauguse leidmine etteantud meetrikas  
% TEST: distants(noor, rauk,V).  
%-----  
distants(Obj1, Obj2, Val):-    % Kui transit. sulund olemas  
    closure(_,_,_),           % Kas sulund on arvutatud  
    (closure(Val, Obj1, Obj2);  
    (closure(Val1,Obj2,Obj1), Val is -Val1;  
    Val=999), !.              % Kui tr. sulundis ei leidu paari  
distants(Obj1,Obj2, Val):-    % Kui sulund veel leidmata  
    m_transitive_closure(jarjestus), %genereeri tr sulund  
    distants(Obj1,Obj2, Val),!.
```

Objektide semantiline kaugus

- ▶ Semantiline kaugus on kasutusel ontoloogia objektide, geneetiliste objektide jm võrdlemisel
- ▶ Lowest common subsumer (*lcs*) kaugus semantilisel võrgul.
- ▶ Näide: tippude x ja y puhul $lcs(x,y) = \textit{tumeroheline_tipp}$

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



```
sem_distance(+X,+Y,-D):-
```

```
    findall(D,(closure(Dx,X,LCS),closure(Dy,Y,LCS),  
                D is Dx+Dy),Ds),
```

```
    sort(Ds,[D|_])).
```