Loogiline programmeerimine

Loeng 11: Algebra ja hulgateooria mõistete programmeerimine Prologis

Põhimõisted

- Hulk
- Relatsioon
- Relatsiooni transitiivsus
- Transitiivne sulund
- Ekvivalents
- Faktorruum
- Meetrika
- Mõistete semantiline kaugus

Hulk

Eksplitsiitne defineerimine:

- Esitame listiga, mille elementideks on hulga elemendid

  $a([el_1,el_2,...,el_n])$.

- või faktidena, kus funktoriks hulga nimi ja parameetriks elemendi nimi

  $a(el_1)$.
  $a(el_2)$.
  ...
  $a(el_n)$.

- või universaalse hulga faktidena, kus ka hulga nimi on parameeter

  hulk(hulga_nimi, elemendi_nimi).

Hulk

- Implitsiitne defineerimine:
  - Baashulga ja kitsendava tingimusega (variant I):

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

  - Baashulga ja kitsendava tingimusega (variant II):

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

Hulk

- Implitsiitne defineerimine:
  - Tingimus induktsiooniga (variant III):

```
natural(X):-                % if not natural
      X < 0,!,fail.
natural(0):- !.   % if natural
natural(X):-
      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:

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

Näide 2:

```
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|Objects]):-
    is_a(Rel,SuperRel),
    relation([SuperRel|SuperObjects]),
    forall(member(Obj,SuperObjects),
                    constraint(Obj)).
```

Relatsiooni transitiivsus

- Relatsioon R on *transitiivne*, kui

  $$xRy \text{ ja } yRz \Rightarrow xRz.$$

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

- Binaarse relatsiooni R *transitiivne sulund* R^+ on relatsiooni R kõigi astmete ühendiga määratud binaarne relatsioon R^+: xR^+y, kus
  - $xR^+y = \cup_i xR^iy.$

Transitiivne sulundi arvutamine

```
transitive_closure(Rel):-     % 1. astme leidmine
      Relation =..[Rel,X,Y],
      call(Relation),
      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 *ekvivalentsisuhteks* (või *-seoseks* või *-relatsiooniks*), kui $\forall s, s', s'' \in dom(\sim)$ 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 tähendagu $s \sim s'$, et inimene s on sama vana kui s', siis on seos "~" ekvivalentsisuhe hulgal S.

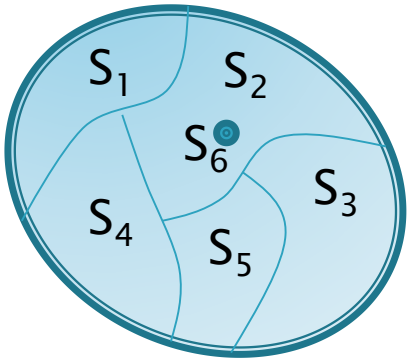
Ekvivalentsiklass

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

$S = \cup_i S_i$, 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.

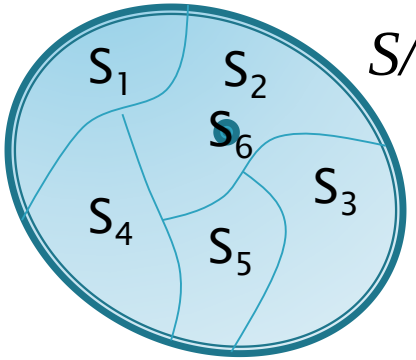
Faktorruum

Olgu ~ ekvivalentsisuhe (relatsioon), siis

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

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

Näide:



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

Ekvivalents objektide hulgal

- Vaatame ekvivalentsisuhet tüübiga objektide hulkal.
- Olgu
  - objektidel määratud *tüübid* (st väärtuste hulgad) ja
  - tüüpidel on defineeritud *meetrika.*

- Hulga Y *meetrika* on kujutus d: $Y \times Y \to R$, kus
  - $\forall x, y : \in Y, \quad d(x, y) \geq 0 \quad \text{ja} \quad d(x, x) = 0$
  - $\forall x, y : \in Y, \quad d(x, y) = d(y, x)$
  - $\forall x, y, z : \in Y, \quad d(x, y) + d(y, z) \geq d(x, z)$

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

Objektide hulga esitus Prologis

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

```
HULGA_NIMI(el_atrib_1, ... , el_atrib_n).
```

- Väärtustatud parameetritega faktid esitavad hulga karakteristliku predikaadi interpretatsiooni.
- 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 tema atribuutide tüüpide ristkorrutisega:

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

Näide

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

- Veel tüübi defineerimise võimalusi Prologis :
  - elementaartüüp: `tyyp(tüübi_nimi,tüübi_väärtused).`
  - tüübikonstruktoriks on ristkorrutis:

Näide (atribuudi tüüp $Type_i$ on eksplitsiitne hulk):

```
tyyp(obj_tüübinimi, Attr1, Attr2,..., Attrn):-
        member(Attr1,Type_1),
        …
        member(Attrn,Type_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).
```

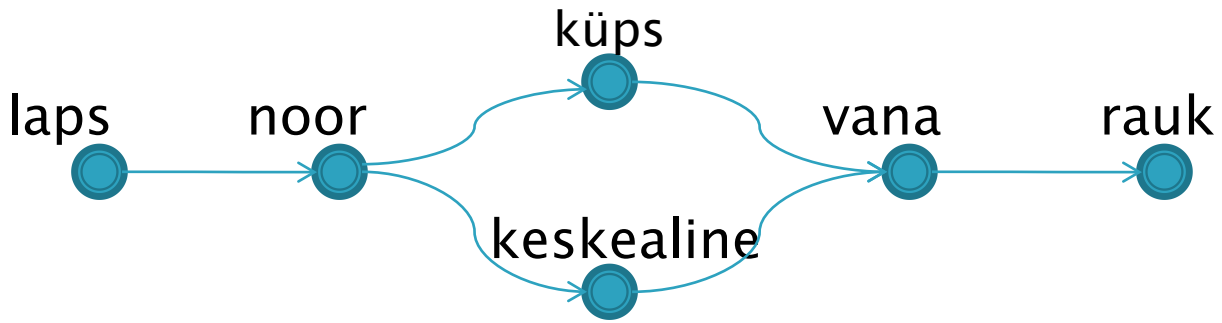
Näide (meetrika hulgal Inimesed):

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:

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

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

Näide

- Ühe vanusegrupi moodustavad inimesed, kelle vanuse 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.

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.
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 leidud kaugus < teadaolevast?
  retract(closure(DD,A,C)).
```

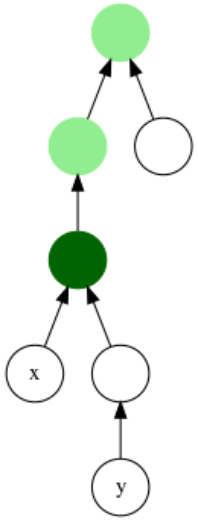
```
%=============================================================
% Reegel klasterdab hulga ekvivalentsiklassideks – fakt eq_class/2
% eq_class(KLASSI NIMI, BAASHULGA_NIMI(ELEMENDI NIMI, ATRIBUUT_mille pohjal)).
%=============================================================
equivalence_class(Hulk, ArityNV, Nr, Distants):-
    functor(TermV, Hulk, ArityNV),     % moodustab termi nimega Hulk aarsusega ArityNV
    call(TermV),
    arg(Nr,TermV,Val),                 % leiab termi TermV Nr-nda parameetri 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 ekvival. kl. kuuluva elem-ga
    abs(D, D1), D1 =< Distants,
    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(_,_,_),
      (closure(Val, Obj1, Obj2);
      (closure(Val1,Obj2,Obj1), Val is 0 - Val1);
      Val=999), !.     % Kui tr. sulundis ei leidu paari
distants(Obj1,Obj2, Val):-    % Kui sulund veel leidmata
      transitive_closure(jarjestus),  % genereeri tr sulund
      distants(Obj1,Obj2, Val),!.
```

Objektide semantiline kaugus

- Semantiline kaugus kasutusel ontoloogia objektide, geneetiliste objektide jm võrdlemisel
- Lihtsaim juhtum on lowest common subsumer (*lcs*) kaugus semantilisel võrgul.
- Näide: tippude x ja y puhul $lcs(x,y) = tumeroheline\_tipp$



$$d^{sem}(x,y) = d(x, lcs(x,y)) + d(y, lcs(x,y))$$

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