Loogiline programmeerimine

Loeng 7: Algebra ja hulgateooria mõistete programmeerimine Prologis

Põhimõisted

- Hulk
- Relatsioon
- Relatsiooni transitiivsus
- Transitiivne sulund
- Ekvivalents
- Faktorruum
- Meetrika
- Mõistete 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 predikaadiga:

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

- Genereeriva funktsiooniga:

```
natural(X):-          % if not natural
        X < 0,!,fail.
natural(0):- !.       % if natural
natural(X):-
        XX is X-1,
        natural(XX).
```

Relatsioon

- **Eksplitsiitne defineerimine**

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 ja $yRz \Rightarrow xRz.$

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

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

Transitiivne sulundi arvutamine

```
transitive_closure(Rel):-
        Relation =..[Rel,X,Y],
        call(Relation),
        assertz(closure(1,X,Y)),   fail.
transitive_closure(_):-
        call(closure(N,A,B)),
        call(closure(1,B,C)),
        N1 is N+1,
        assertz(closure(N1,A,C)), fail.
transitive_closure(_).
```

Ekvivalents

Relatsiooni ~ nimetatakse ekvivalentsisuhteks, kui $\forall s, s', s'' \in \text{dom} \sim$ kehtib

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

- <u>Näide</u>:

  Olgu S Eesti elanike hulk ja tähendagu $s \sim s'$, et inimene s on sama vana kui s', siis on seos "~" ekvivalentsisuhe.

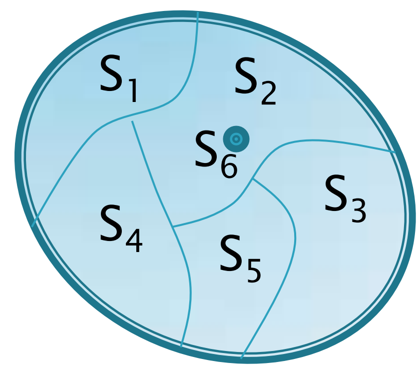
Ekvivalentsiklass

Ekvivalentsisuhe tükeldab hulga, 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

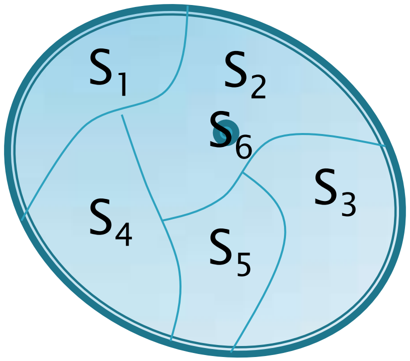
Faktorruum

Olgu ~ ekvivalentsisuhe (relatsioon), siis

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

tähistab faktorruumi s.o. hulka, mis koosneb kõikidest ekvivalentsiklassidest.

Näide:



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

Ekvivalents objektide hulgal

- Vaatame ekvivalentsisuhet tüübiga objektide hulkal,
- Olgu
  - objektidel *tüübid*
  - tüüpidel defineeritud *meetrika.*

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

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

Objektide hulga esitus Prologis

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

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

faktid esitavad hulga karakteristliku predikaadi interpretatsiooni hulka

Tüübid

- Hulga elemendi tüüp on tema atribuutide tüüpide ristkorrutis

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

- Tüübi defineerimise võimalusi Prologis :
  - elementaartüüp:

    `tyyp(tüübi_nimi,[määramispk]).`
  - tüübikonstruktoriks on ristkorrutis:

    `tyyp(tüübinimi, tyyp1, tyyp2,..., tyypn).`

Meetrikaga tüübid

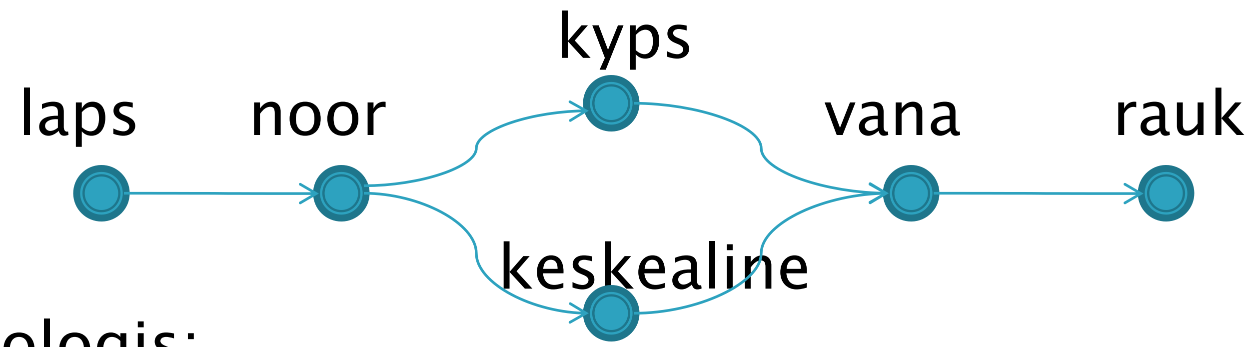
- Kui numbriline tüüp, siis on meetrika defineeritud tüübi endaga
- Kui mittenumbriline loenduv tüüp, siis defineerime meetrika selle tüübi määramispiirkonnal
  - Võrreldavate väärtuste vahel defineerime järjestussuhte predikaadiga
    MEETRIKA(tüübi_nimi, väiksem_väärtus, suurem_väärtus)

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

$d(o_i, o_j) = k-1$, kus k on relatsiooni "*meetrika*" min. astak, nii et $\langle o_i, o_j \rangle \in meetrika^k$ või $\langle o_j, o_i \rangle \in meetrika^k$

Näide

- Kasutades ekvivalentsi relatsiooni defineerimiseks meetrikat ja kaugust, siis ühe vanusegrupi moodustavad inimesed, kelle vanuse erinevus on $\leq k$ aastat.

Näiteid Prologi predikaatidest

```
% Transitive closure
%-----------------------------------------
% TEST1: transitive_closure(pr).
% TEST2: transitive_closure(jarjestus).
%-----------------------------------------
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).
```

Transitive closure with metrics (minimal 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 juba olemas niisugune fakt?
m_test(A,C,D):-
    closure(DD,A,C),                  % kas paar esineb madalamas astmes?
    D < DD,
    retract(closure(DD,A,C)).
```

```
%========================================
% Ekvivalentsiklassid
% genereerib faktid:
% 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.

% TEST: equivalence_class(inimene,2,2,1).
```

Näiteid Prologi predikaatidest

```
%----------------------------------------
% Kauguse leidmine etteantud meetrikas
% TEST: distants(noor, rauk,V).
%----------------------------------------
distants(Obj1, Obj2, Val):-
   closure(_,_,_),
        (closure(Val, Obj1, Obj2);
        (closure(Val1,Obj2,Obj1), Val is 0 - Val1);
        Val=999), !.
distants(Obj1,Obj2, Val):-    % Kui sulund veel leidmata
        transitive_closure(jarjestus),
        distants(Obj1,Obj2, Val),!.
```