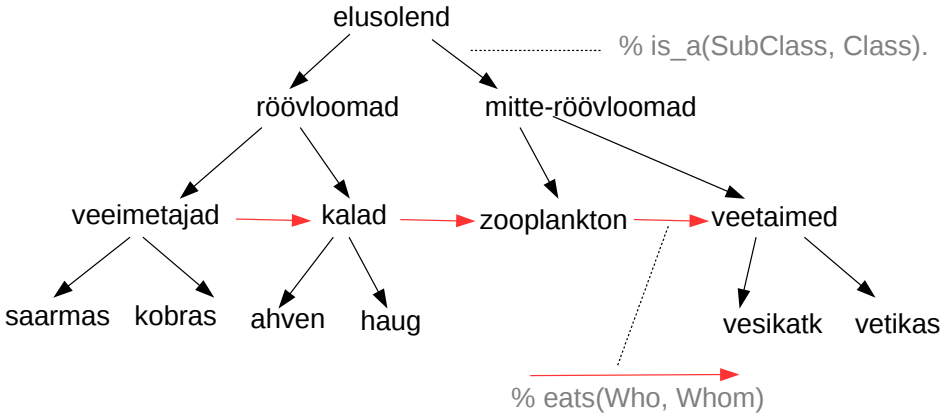
Praktikum PR08

Semantilised võrgud

Eesmärk: Omandada praktiline oskus kodeerida klasside omadusi, seoseid ning klassidevahelisi pärimisreegleid.

Olgu meil järgnev bioloogiast inspireeritud semantiline võrk, mis kirjeldab elavate organismide taksonoomiat:



Praktikumi eesmärgiks on kirjutada reegel, mis leiab väljasurevad liigid ja nende arvu, kui anda ette algselt häviv liik.

1) Defineerige elusloodusest inspireeritud semantiline võrk kasutades predikaati `is_a/2` ja lisage faktibaasile juurde fakte loomade ja taimede kohta.

Näiteks:

```
% is_a(SubClass, Class).
is_a(roovloomad,elusolend).
is_a(mitte-roovloomad,elusolend).
is_a(veeimetajad,roovloomad).
is_a(kalad,roovloomad).
is_a(saarmas,veeimetajad).
is_a(kobras,veeimetajad).
is_a(ahven,kalad).
```

```
is_a(haug,kalad).
is_a(zooplankton,mitte-roovloomad).
is_a(veetaimed,mitte-roovloomad).
is_a(vesikatk,veetaimed).
is_a(vetikas,veetaimed).
```

3) Defineerige seosed kes keda sööb kasutades predikaati `eats/2`.

```
% eats(Who, Whom).
eats(zooplankton,veetaimed).
eats(kalad,zooplanton).
eats(veeimetajad,kalad).
```

3) Programmeerige predikaat `count_terminals(Node,Terminals,Count)`, mis leiab terminaalsete tippude arvu.

```
?- count_terminals(mitte-roovloomad, T, C).
T = [zooplankton, vesikatk, vetikas],
C = 3.
```

4) Programmeerige predikaat `extinction(Who,What_spieces,How_many)`, mis leiab väljasurevad liigid ja nende arvu, kui anda ette algselt häviv liik.

```
?- extinction(veetaimed, T, C).
T = [saarmas, kobras, ahven, haug, zooplankton, vesiatk, vetikas],
C = 7.
```

5) Lisa reegel `find_most_sensitive_species/3`. Lisatud reegel peab leidma liigi, mille väljasuremine tekitaks toitumisahela kaudu liigilisele mitmekesisusele kõige suuremat kahju.

Ülesande lahendamiseks tuleks otsida predikaadile `extinction/3` lahendit, mis maksimeerib väljasurevate liikide arvu. Soovitav on lahenda see dünaamilise faktiga `max/3` ja tagurdamisega otsinguga.

```
?- find_most_sensitive_species(L, C, T).
L = veetaimed,
T = [saarmas, kobras, ahven, haug, zooplankton, vesiatk, vetikas],
C = 7.
```

Dünaamilised faktid:

```
:-dynamic fakt/2, fakt2/4.      % dünaamilise fakti defineerimine

assert(fakt(14, 56)).           % fakti lisamine

retract(fakt(14, 56)).          % fakti kustutamine

retractall(fakt(14, A).         % kustutab kõik faktid nimega fakt, mille esimene
                                % parameeter omab väärtust 14

retractall(fakt(_, _)).         % kustutab kõik faktid fakt/2
```

Näide: parameetri tingimust rahuldavate kõikide faktide otsimine

Olgu defineeritud faktid step/3:

```
step(a,b,1).
step(c,b,3).
step(a,c,2).
step(b,c,3).

main:-
        step(From,To,X),
        X == 3,
        assert(fakt(From,To)),
        fail.
main.
```

NB! Kui tagurdamine puudutab rekursiivset reeglit, siis on soovitav kasutada **cut** operaatorit rekursiooni mittesoovitatavasse harusse mineku vältimiseks.