Harjutus 2. kontrolltööks

ITI0211

J.Vain

23.11.2021

DCG (loeng 8)

- Terminal- ja mitteterminalsümbolite mõiste
- Mis on asendusreegel?

  Vastus: Reegel defineerib kuidas saab tuletada lauseid ja fraase asendades mitteterminalsümboleid fraasis teise sümboli või sümbolite jadaga
- DCG reeglite esitus ja kitsendused reeglite kirjutamisel:

  Vastus:

  Asendada saab ainult mitteterminalsümbolit, DCG mitteterminalsümboliks võivad olla Prologi termid v.a. listid.

DCG

- Kirjutada DCG reeglid etteantud lausete parsimiseks.
- Näide:

„Kaks meest viisid klaveri lavale“

DCG

- Kirjutada DCG reeglid etteantud lausete parsimiseks.
- Näide:

„Kaks meest viisid klaveri lavale.“

„Üks naine viis lilled lavale.“

  - lause --> nimisõnafraas, tegusõnafraas.
  - nimisõnafraas --> arvsõna, nimisõna_alusena.
  - tegusõnafraas --> tegusõna, nimisõna_sihitises, kohamäärsõna.
  - arvsõna --> [üks];[kaks].
  - nimisõna_alusena --> [meest];[naine].
  - tegusõna --> [viisid];[viis].
  - nimisõna_sihitises --> [klaveri];[lilled].
  - kohamäärsõna --> [lavale].

DCG

- Esitada DCG reeglid Prolog programmina:

lause --> nimisõnafraas, tegusõnafraas.

nimisõnafraas --> arvsõna, nimisõna_alusena.

tegusõnafraas --> tegusõna, nimisõna_sihitises,
kohamäärsõna.

arvsõna --> [üks];[kaks].

...

```
lause(A, B) :-
        nimisonafraas(A, C),
        tegusonafraas(C, B).
 nimisonafraas(A, B) :-
        arvsõna(A,C),
        nimisõna_alusena(C, D).
tegusõnafraas(A, B):-
        tegusõna(A, C),
        nimisõna_sihitises(C,D),
        kohamäärsõna(D,B).
arvsõna(A,B):-
        A=[üks|B];
        A=[kaks|B].
```

Tagurdamise kasutamine reeglites

- Otsingu juhtimise predikaadid *repeat*, *fail*, *cut* ja nende kasutamine
- Tagurdamisega otsing faktide hulgal
- Ülesanne (5 punkti):
  - Olgu antud hulk *h* faktidega `hulk/2`, kus esimene parameter on hulga nimi ja teine parameter on selle hulga elemendi väärtus.

    ```
    :- dynamic hulk/2.
    hulk(h,1).
    hulk(h,2).
    hulk(h,3).
    hulk(h,4).
    ```

    Programmeerige tagurdamisega reegel `delete_smaller(H, Const)`, mis kustutab kõik need hulga *h* elemendid, mis on väiksemad selles päringus etteantud konstandist `Const`.

Näide:

```
?- delete_smaller(h, 2).
```

kustutab fakti `hulk(h,1).`

Tagurdamise kasutamine reeglites: ülesande lahendused

- **Lahendus 1:** `repeat`-ga

```
delete_smaller(H,Const):-
        repeat,
        retractall(tunnus),
        hulk(H,El),
        (El<Const,
                retract(hulk(H,El)),
                assert(tunnus)
         ;
                true),
        not(tunnus).
```

Väljumiseks kasutame abipredikaati `tunnus/0`

Lahendus 2: `fail`-ga

```
delete_smaller2(H, Const):-
        hulk(H,El),
        El<Const,
        retract(hulk(H,El)),
        fail.
delete_smaller2(_,_).
```

Rekursioon listidel

- Ülesanne (5 punkti) :
  - Koostada programm, mis tuvastab kas list L2 on listi L12 sufiks (lõpuosa) ja tagastab listi L1, kus L1 on listi L12 algusosa (prefiks)
  - Päring:

```
?-  suffix(-L1, +L2,+L12).
```

  - Märkus: Lubatud on kasutada süsteemipredikaati `reverse`

Rekursioon listidel

- Lahendus

```
suffix(L2, [], L2):- !.                 % Peatumistingimus
suffix(_, _, []):- !,fail.              % Peatumistingimus
suffix([El], L2, [El|L2]):- !.          % Peatumistingimus
suffix([El|L1],L2,[El|L21]):-           % Rekursiooni samm
        suffix(L1,L2,L21).
```

Rekursioon listidel: ülesanne

- Ülesanne (5 punkti):
  - Hulgad h1 ja h2 on listide kujul. Koostada programm, mis leiab hulkade h1 ja h2 vahe.
  - (Hulk h3 on hulkade h1 ja h2 vahe, kui selles sisalduvad ainult need hulga h1 elemendid, mis ei sisaldu hulgas h2.
  - Programmis on lubatud kasutada süsteemipredikaati `member/2`.
  - Päring:

```
?- vahe(H1,H2,H3).
```

Rekursioon listidel: ülesanne

- Ülesanne (5 punkti):
  - Hulgad h1 ja h2 on listide kujul. Koostada programm, mis leiab hulkade h1 ja h2 vahe.
  - (Hulk h3 on hulkade h1 ja h2 vahe, kui selles sisalduvad ainult need hulga h1 elemendid, mis ei sisaldu hulgas h2.
  - Programmis on lubatud kasutada süsteemipredikaati `member/2 ja select/2`
  - Päring:

```
?- vahe(H1,H2,H3).
```

```
vahe(H1,[],H1):- !.
vahe(H1,[El|H2],H3):-
          vahe(H1,H2,H31),
          (member(El,H31),select(El,H31,H3);
          H3=H31),!.

% Test: vahe([a,s,d],[],L3).
% Test: vahe([a,s,d],[d,s],L3).
% Test: vahe([a,s,d],[f,c],L3).
% Test: vahe([],[f,c],L3).
```

Hierarhilised listid: ülesanne

- Ülesanne:
  - Koostada predikaat `list_partition/3`, mis jagab tasapinnalise eraldajate järgi (1. parameeter) alamlistide listiks .
  - Päring: `list_partition(+Eraldaja,+ListIn,-ListOut),`
  - kus `ListIn` on tasapinnaline list ja `ListOut` on hierarhiline list.

  - Näide:

```
?- list_partition('\n',[s,d,'\n',d,f,'\n','See'],ListOut).
ListOut = [[s,d],[d,f],['See']]
```

  - Soovitus: kasutada reeglis sabarekursiooni

Hierarhilised listid: lahendus ülesandele

```
list_partition(_,[],[[]]).
list_partition(Separator,[Separator|ListIn],[[]|ListOut]):-
      list_partition(Separator, ListIn, ListOut).
list_partition(Separator,[Element|ListIn],[[Element|InternalL]|ListOut]):-
      list_partition(Separator,ListIn,[InternalL|ListOut]).

% Test: list_partition('\n',[s,d,'\n',d,f,'\n','See'],ListOut).
```