Решения на задачи от второ контролно по Логическо програмиране

11 януари 2020

1 Първа задача на пролог

За естествено число n с представяне в 8-ична бройна система $n = \sum_{i=0}^{\infty} a_i 8^i$, където $a_i \in \{0, 1, 2, 3, 4, 5, 6, 7\}$, ще казваме, че е *байт-дървовидно* тогава и само тогава, когато ако:

- Вариант 1: $v = \lfloor u/8 \rfloor$ и $a_u \not\equiv 0 \pmod 7$, то $a_u \equiv a_v + 1 \pmod 6$.
- Вариант 2: $v = \lfloor u/8 \rfloor$ и $a_u \not\equiv 0 \pmod 6$, то $a_v \equiv a_u + 1 \pmod 7$.

Да се дефинира на предикат на пролог $byteTreeNum(N)$, който по дадено естествено число N проверява дали то е байт-дървовидно.

1.1 Примерно решение

```
% Helper predicates: nth0 (a.k.a. nthElement)
condition11(L) :-
    not(( nth0(U, L, AU),
          nth0(V, L, AV),
          V=:=U div 8,
          AU mod 7=\=0,
          (AU-AV-1)mod 6=\=0
        )).

condition12(L) :-
    not(( nth0(U, L, AU),
          nth0(V, L, AV),
          V=:=U div 8,
          AU mod 6=\=0,
          (AV-AU-1)mod 7=\=0
        )).

genList(0, []).
genList(N, [Last|R]) :-
    N>0,
    Last is N mod 8,
    N1 is N div 8,
    genList(N1, T).

byteTreeNum(N) :-
    genList(N, L),
    condition11(L).
```

2 Втора задача на пролог

Ще казваме, че един списък е *анаграма* на друг, ако е съставен от същите елементи, но в евентуално различен ред.

Да се дефинира предикат на пролог $maxAnagrams(L, M)$ на пролог, който по даден списък от списъци L, генерира в M най-голямото число, за което има поне M на брой:

- M-елементни списъка от L, които са анаграми един на друг.
- $(M + 2)$-елементни списъка от L, които са анаграми един на друг.

2.1 Примерно решение

```
% Helper predicates: member, length, permutation
subsequence([], []).
subsequence([H|T], [H|R]) :-
    subsequence(T, R).
subsequence([_|T], R) :-
    subsequence(T, R).

anagrams(L) :-
    not(( member(X, L),
          member(Y, L),
          not(permutation(X, Y))
        )).

condition21(S, M) :-
    anagrams(S),
    S=[H|_],
    length(H, M),
    length(S, M1),
    M1>=M.

condition22(S, M) :-
    anagrams(S),
    S=[H|_],
    length(H, M),
    length(S, M1),
    M1>=M-2.

maxAnagrams(L, M, S) :-
    subsequence(L, S),
    condition21(S, M),
    not(( subsequence(L, S1),
          condition21(S1, M1),
          M1>M
        )).
```