Tik-79.144 **Syksy 1998**
Logiikka tietotekniikassa: perusteet
Laskuharjoitus 11
Ratkaisut

1. `PROLOG`-tulkki toimii pinoperiaatteella: annettu kysely laitetaan pinoon, ja lähdetään käymään läpi ohjelman sääntöjä järjestyksessä läpi. Tarkoituksena on löytää sääntö, jonka pää unifioituu kyselyn kanssa. Kun tällainen sääntö löytyy, poistetaan alkuperäinen kysely pinosta, ja lisätään löydetyn säännön häntä alikyselyiksi pinoon. Tehtävän a-kohdan kysely unifioituu säännön

```
reverse(Xs,Ys) :- reverse(Xs, [], Ys)
```

kanssa (substituutiolla `Xs/[a,b,c]`), joten pinoon lisätään alikysely `reverse([a,b,c], [], Ys)`.

Mikäli pino saadaan tyhjennettyä säännöistä, on kyselylle löytynyt todistus ja tulkki palauttaa tehdyt muuttujasidonnat. Mikäli jokin alikysely epäonnistuu, tulkki yrittää löytää jonkin toisen säännön, jota käyttämällä todistus onnistuu.

a)
```
| ?- reverse([a,b,c],Xs).
 + 1  1  Call: reverse([a,b,c],_83) ?
 + 2  2  Call: reverse([a,b,c],[],_83) ?
 + 3  3  Call: reverse([b,c],[a],_83) ?
 + 4  4  Call: reverse([c],[b,a],_83) ?
 + 5  5  Call: reverse([],[c,b,a],_83) ?
 + 5  5  Exit: reverse([],[c,b,a],[c,b,a]) ?
 + 4  4  Exit: reverse([c],[b,a],[c,b,a]) ?
 + 3  3  Exit: reverse([b,c],[a],[c,b,a]) ?
 + 2  2  Exit: reverse([a,b,c],[],[c,b,a]) ?
 + 1  1  Exit: reverse([a,b,c],[c,b,a]) ?

Xs = [c,b,a] ?

yes
```

b)
```
| ?- reverse([a,b,c],[c,b]).
 + 1  1  Call: reverse([a,b,c],[c,b]) ?
 + 2  2  Call: reverse([a,b,c],[],[c,b]) ?
```

```
 + 3  3  Call: reverse([b,c],[a],[c,b]) ?
 + 4  4  Call: reverse([c],[b,a],[c,b]) ?
 + 5  5  Call: reverse([],[c,b,a],[c,b]) ?
 + 5  5  Fail: reverse([],[c,b,a],[c,b]) ?
 + 4  4  Fail: reverse([c],[b,a],[c,b]) ?
 + 3  3  Fail: reverse([b,c],[a],[c,b]) ?
 + 2  2  Fail: reverse([a,b,c],[],[c,b]) ?
 + 1  1  Fail: reverse([a,b,c],[c,b]) ?

no
```

2. a)

```
qmove(X1,Y1,X2,Y2) :- diff(X1,X2,_), equal(Y1,Y2).
qmove(X1,Y1,X2,Y2) :- diff(Y1,Y2,_), equal(X1,X2).
qmove(X1,Y1,X2,Y2) :- diff(X1,X2,N), diff(Y1,Y2,N).

equal(1,1).
equal(X,Y) :- succ(A,X), succ(B,Y), equal(A,B).

diff(X,Y,N) :- add(X,N,Y).
diff(X,Y,N) :- add(Y,N,X).

add(X,1,Y) :- succ(X,Y).
add(X,Y,Z) :- succ(A,Y), succ(B,Z), add(X,A,B).

succ(1,2).
succ(2,3).
succ(3,4).
succ(4,5).
succ(5,6).
succ(6,7).
succ(7,8).
```

b)

```
nqmove(X1,Y1,X2,Y2) :- diff(X1,X2,N), diff(Y1,Y2,M),
                       diff(N,M,_).
```

c)

```
nocc(X,Y,[]) :- equal(X,Y).
nocc(X,Y,[]) :- diff(X,Y,_).
nocc(X1,Y1,[pos(X2,Y2)|L]) :- nqmove(X1,Y1,X2,Y2),
                              nocc(X1,Y1,L).

noccupied(X,8,[]) :- nocc(X,8,[]).
noccupied(X1,Y1,[pos(X2,Y2)|L]) :- succ(Y1,Y2),
```

```
  noccupied(X2,Y2,L), nocc(X1,Y1,[pos(X2,Y2)|L]).

solution([pos(X,1)|L]) :- noccupied(X,1,L).
```