Tik-79.144 **Syksy 1997**
Logiikka tietotekniikassa: perusteet
Laskuharjoitus 9
Ratkaisut

1. Kirjoitetaan `otter`-ohjelmalle syötetiedosto `monkey.otr`:

```
% Monkey example

set(binary_res).             % Resoluutiostrategia
assign(max_mem, 1500).       % 1.5 MB
assign(max_seconds, 1800).   % 30 min
set(free_all_mem).

formula_list(sos).

(all x (close(m,x) -> reach(m,x))).
(all x ((tall(x) & on(m,x) & under(x,b)) -> close(m,b))).
tall(c).
climb(m,c).
(all z moves(m,c,z)).
(all x (climb(m,x) -> on(m,x))).
(all x (all y (all z (moves(x,y,z) ->
         (close(y,z) | under(y,z)))))).
-close(c,b).
-(exists x reach(m,x)).

end_of_list.
```

Kutsutaan ohjelmaa komennolla `otter < monkey.otr > monkey.out`
Tulostiedostosta löytyy nyt mm. kaavojen klausuuliesitys:

```
list(sos).
1 [] -close(m,x) | reach(m,x).
2 [] -tall(x) | -on(m,x) | -under(x,b) | close(m,b).
3 [] tall(c).
4 [] climb(m,c).
5 [] moves(m,c,z).
6 [] -climb(m,x) | on(m,x).
7 [] -moves(x,y,z) | close(y,z) | under(y,z).
```

```
8 [] -close(c,b).
9 [] -reach(m,x).
end_of_list.
```

sekä binääriresoluutiolla haettu todistus tyhjälle klausuulille:

```
1 [] -close(m,x) | reach(m,x).
2 [] -tall(x) | -on(m,x) | -under(x,b) | close(m,b).
3 [] tall(c).
4 [] climb(m,c).
5 [] moves(m,c,z).
6 [] -climb(m,x) | on(m,x).
7 [] -moves(x,y,z) | close(y,z) | under(y,z).
8 [] -close(c,b).
9 [] -reach(m,x).
10 [binary,1,9] -close(m,x).
11 [binary,6,4] on(m,c).
12 [binary,7,5] close(c,x) | under(c,x).
15 [binary,12,8] under(c,b).
17 [binary,2,11] -tall(c) | -under(c,b) | close(m,b).
24 [binary,17,15] -tall(c) | close(m,b).
26 [binary,24,3] close(m,b).
27 [binary,26,10] .
```

Näinollen klausuulijoukko on toteutumaton ja $\exists x \mathrm{Reach}(m,x)$ seuraa loogisesti premisseistä. Klausuuliesityksestä selviää, että kaikki premissit eivät ole esitettävissä Horn-klausuuleina, joten suora käännös `PROLOG`-ohjelmaksi ei onnistu.

2.
```
make_evaluable(_+_, $SUM(_,_)).
make_evaluable(_-_, $DIFF(_,_)).
make_evaluable(_<=_, $LE(_,_)).
make_evaluable(_>_, $GT(_,_)).

set(auto).

list(usable).
-j(x, y) | j(3, y).  % täytetään kolmen litran kannu
-j(x, y) | j(0, y).  % tyhjennetään kolmen litran kannu
-j(x, y) | j(x, 4).  % täytetään neljän litran kannu
-j(x, y) | j(x, 0).  % tyhjennetään neljän litran kannu
```

```
% kaadetaan pienempi kannu suurempaan
-j(x, y) | -(x+y <= 4) | j(0, y+x).
% kaadetaan pienemmästä suurempaan kunnes täysi
-j(x, y) | -(x+y >  4) | j(x - (4-y), 4).
% kaadetaan suurempi kannu pienempään
-j(x, y) | -(x+y <= 3) | j(x+y, 0).
% kaadetaan suuremmasta pienempään kunnes täysi
-j(x, y) | -(x+y >  3) | j(3, y - (3-x)).


j(0, 0).   % alkutila, molemmat tyhjiä

-j(x, 2).  % lopputila, neljän litran kannu
           % sisältää kaksi litraa

end_of_list.



---------------- PROOF ----------------

2 [] -j(x,y)|j(0,y).
3 [] -j(x,y)|j(x,4).
7 [] -j(x,y)| -(x+y<=3)|j(x+y,0).
8 [] -j(x,y)| -(x+y>3)|j(3,y- (3-x)).
9 [] -j(x,2).
10 [] j(0,0).
11 [hyper,10,3] j(0,4).
13 [hyper,11,8,eval,demod] j(3,1).
16 [hyper,13,2] j(0,1).
18 [hyper,16,7,eval,demod] j(1,0).
20 [hyper,18,3] j(1,4).
22 [hyper,20,8,eval,demod] j(3,2).
23 [binary,22.1,9.1] $F.

------------ end of proof -------------
```