Tik-79.144 **Syksy 1998**
Logic in Computer Science: Foundations
Exercise 10
Solutions

1. a) Define predicates $R_n(x, y)$ as follows:

$$\begin{gathered}\forall x R_0(x, x) \\ \forall x \forall y \forall z (R_0(x, y) \wedge K(y, z) \rightarrow R_1(x, z)) \\ \forall x \forall y \forall z (R_1(x, y) \wedge K(y, z) \rightarrow R_2(x, z)) \\ \vdots \\ \forall x \forall y \forall z (R_{k-1}(x, y) \wedge K(y, z) \rightarrow R_k(x, z))\end{gathered}$$

The rules mean that each node has a route of length 0 to itself, and if there is a route of length $k-1$ from node x to node y and there is an edge from y to z, then there is a route of length k from x to z.

Graph:

$$a \quad \begin{array}{c}\longrightarrow \\ \longleftarrow\end{array} \quad b \quad \longrightarrow \quad c$$

can be formalized as follows:

$$\begin{gathered}K(a, b) \\ K(b, a) \\ K(b, c)\end{gathered}$$

b) To use resolution we have to transform the set of sentences into clausul form.

From the definition of the graph we get clauses

$$\{K(a, b)\}, \{K(b, a)\}, \{K(b, c)\}$$

Because we want to examine routes of length 3 we have to define predicates R_n only up to value $n = 3$.

$$\begin{gathered}\{R_0(x, x)\} \\ \{\neg R_0(x, y), \neg K(y, z), R_1(x, z)\} \\ \{\neg R_1(x, y), \neg K(y, z), R_2(x, z)\} \\ \{\neg R_2(x, y), \neg K(y, z), R_3(x, z)\}\end{gathered}$$

Finally add the negation of the desired consequence to the graph

$$\begin{aligned} & \neg\exists x(R_2(x,x) \wedge R_3(x,c)) \\ \equiv\ & \forall x \neg(R_2(x,x) \wedge R_3(x,c)) \\ \equiv\ & \forall x(\neg R_2(x,x) \vee \neg R_3(x,c)) \\ \Rightarrow\ & \{\neg R_2(x,x), \neg R_3(x,c))\} \end{aligned}$$

Informally, the desired consequence is "there is some node such that there is a loop of length 2 back to the node and a path of length 3 to the node c". When we look at the graph we see that node b fulfills this condition. The easiest way to construct the proof is by doing it in following steps:

1. Prove that there is a route of length 1 from b to a.
2. Prove that there is a route of length 2 from b to b.
3. Prove that there is a route of length 3 from b to c.

1	$\{K(b,a)\}$	P
2	$\{\neg R_0(x_1,y_1), \neg K(y_1,z_1), R_1(x_1,z_1)\}$	P
3	$\{\neg R_0(x_1,b), R_1(x_1,a)\}$	$[1,2]$ MGU: $\{y_1/b, z_1/a\}$
4	$\{R_0(x_2,x_2)\}$	P
5	$\{R_1(b,a)\}$	$[3,4]$ MGU: $\{x_1/b, x_2/b\}$
6	$\{\neg R_1(x_3,y_3), \neg K(y_3,z_3), R_2(x_3,z_3)\}$	P
7	$\{\neg K(a,z_3), R_2(b,z_3)\}$	$[5,6]$ MGU: $\{x_3/b, y_3/a\}$
8	$\{K(a,b)\}$	P
9	$\{R_2(b,b)\}$	$[7,8]$ MGU: $\{z_3/b\}$
10	$\{\neg R_2(x,y), \neg K(y,z), R_3(x,z)\}$	P
11	$\{\neg K(b,z_4), R_3(b,z_4)\}$	$[9,10]$ MGU: $\{x_4/b, y_4/b\}$
12	$\{K(b,c)\}$	P
13	$\{R_3(b,c)\}$	$[11,12]$ MGU: $\{z_4/c\}$
14	$\{\neg R_2(x_5,x_5), \neg R_3(x_5,c)\}$	P
15	$\{\neg R_3(b,c)\}$	$[9,14]$ MGU: $\{x_5/b\}$
16	$\square$	$[13,15]$

2. Write the input file `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.
```

Start the program with command `otter < monkey.otr > monkey.out`
The output file contains first the clausul form of the sentences.

```
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.
```

Next there is a binary resolution proof of the claim:

```
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] .
```