Universität des Saarlandes
FR Informatik

MPI INFORMATIK

Harald Ganzinger
Uwe Waldmann

June 13, 2002

Tutorials for "Logic in Computer Science"
Exercise sheet 9

Exercise 9.1:
If a list l has the form $[a_1, \dots, a_n]$, then a_{i+1} is called in successor of a_i in l. (For instance, `a`, `r`, and `n` are successors of `a` in the list `[s,a,a,r,l,a,n,d]`.) Implement a Prolog predicate `succ(`l`,`x`,`y`)` that succeeds if y is a successor of x in l. Can your implementation also be used to compute the predecessors of an element in a list?

> **Notes on Prolog programming:** You can find several Prolog implementations on the computers at the University campus. For instance, at the CIP pool in the first floor of Bldg. 45 you can find SWI-Prolog (`/usr/local/bin/pl`); on the MPI computers, GNU Prolog is available (`/opt/gnu/bin/gprolog`). Sources and binaries (Linux, MS-Windows, MacOS X) for SWI-Prolog can also be downloaded from `http://www.swi-prolog.org/`.
>
> To use Prolog, write your clauses into a file *filename*`.pl`, start Prolog, and type
>
> `['`*filename*`.pl'].`
>
> (including the square brackets, the quotes, and the period at the end). Queries can be entered directly at the Prolog prompt (also terminated with a period). If a Prolog system has found a solution (i. e., a success node of the SLD-tree), you can type a semicolon to start the search for another solution.
>
> A remark on the Prolog syntax: To prevent typos, most Prolog implementations issue a warning message if a program clause contains a *singleton variable* (i. e., a variable that occurs only once). To avoid this, use an underscore (`_`) instead of a regular variable name for singleton variables.

Exercise 9.2:
Define a *rotation* of a list l as follows:

- If l is the empty list $[\,]$, then l is a rotation of l.
- If l is a list $[a_1, \dots, a_n]$ with $n \geq 1$, then l is a rotation of l, and every list $[a_i, \dots, a_n, a_1, \dots, a_{i-1}]$ with $1 < i \leq n$ is a rotation of l.

The following Prolog program computes rotations of a list. (We use a predicate `append1` rather than `append`, since the latter is predefined in most Prolog systems.)

```
append1([],L2,L2).
append1([X|L1],L2,[X|L3]) :- append(L1,L2,L3).

rotate(L,L).
rotate([X|L],R) :- append1(L,[X],R1), rotate(R1,R).
```

Describe the SLD-tree for the query `rotate([a,b],X)` and show that it is infinite.

Exercise 9.3:
Give an alternative implementation for `rotate` so that, given any ground list l, the SLD-tree for the query `rotate(`l`,X)` is finite.

Hint: `append1` can not only be used to append two lists, but also to split a list in two sublists.

Challenge: Implement `rotate` in such a way that every rotation of a list is computed exactly once (i.e., if l is a ground list of length n, then the SLD-tree for `rotate(`l`,X)` is finite and contains exactly one success node for $n = 0$ and exactly n success nodes for $n > 0$.)

Exercise 9.4:
Is it possible that a Prolog system terminates for a query G_1, terminates for a query G_2, and loops for the query G_1, G_2 ? If so, how?

Put your solution into the mail box at the door of room 627 in the MPI building (46.1) before June 21, 11:00 (Group D: before June 24, 11:00). Don't forget to write your name and the name of your tutorial group (B, C, D) on your solution.