Prof. O. Nierstrasz
Boris Spasojević, Jan Kurš

Serie 11 - Applications of Logic Programming

Exercise 1: General questions

a. What are definite clause grammars (DCG) and why are they particularly useful in conjunction with Prolog?

b. How are DCG specifications translated into Prolog?

c. What exactly does the 'C' predicate do? And what would be a possible explanation for its rather not meaningful name?

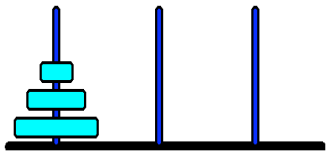
d. Why are left-associative grammar rules problematic?

e. How can we represent syntax trees in Prolog?

f. Why must DCG side conditions be put in curly brackets {}?

Exercise 2: Hanoi Towers

The objective of this famous puzzle is to move N disks from the left peg to the right peg using the center peg as an auxiliary holding peg. It is not allowed to place a larger disk on a smaller disk, and only one disk can be taken away at once from the top. The following diagram depicts the starting setup for $N = 3$ disks: Define a predicate `hanoi(N, A, B, C, Moves)` that solves the hanoi-towers



problem. `Moves` holds the list of moves that represent the process of moving `N` disks from `A` to `B` with the help of `C`. If `N` is bigger than 1, then `N-1` disks will be shifted to `C`, so that the move from `A` to `B` can be accomplished. The move of a disk from `A` to `B` will be represented as `[a to b]`. The binary operator *to* is loaded into the knowledge base by the following commands:

```
:- ensure_loaded(library(operators)). % load readable operators
:- op(900, xfy, to).                  % define new infix operator 'to'
```

Examples:

```
?- hanoi(1,a,b,c,X).
X = [a to b] ?
yes

?- hanoi(2,a,b,c,X).
X = [a to c, a to b, c to b] ?
yes
```