CS2PLD: Programming Language Design Paradigms
Maribel Fernández & Odinaldo Rodrigues

Revision

Exercise 1:

1. Suppose a Prolog program contains the following clauses:

   ```
   sunny(mon).
   sunny(tue).
   sunny(wed).
   warm(mon).
   warm(wed).
   glorious(Day) :- sunny(Day), warm(Day).
   ```

   Write down all the responses which Prolog will give to the query

   ```
   :- glorious(Day).
   ```

   in the order they are given.

2. Draw the SLD-tree built by Prolog for the previous query.

Exercise 2:

1. Define a function *until* that takes a function $f :: \alpha \to \alpha$, a predicate $p :: \alpha \to Bool$ and an element $x :: \alpha$ as arguments, and iterates f until an element that satisfies p is obtained.
2. Give the type of *until*.

Exercise 3:

Consider the SFUN program:

$$double(x) \quad = \quad 2 * x$$

- Show by induction that for any natural number n, $double(n)$ is equivalent to $n+n$ (i.e. show that $double(n)$ and $n+n$ produce the same value, for any natural number n). You can use either call-by-value or call-by-name semantics.
- Explain why for terms of the form $double(t)$ where t is an arithmetic expression, the call-by-value and call-by-name strategies have the same behaviour and are equally efficient.
- Assume empty lists are represented by *nil* and non-empty lists by $x : l$ where x is the first element and l the rest of the list. Explain in your own words the behaviour of the function map defined below, and describe a call-by-name evaluation for

  $$map\ double\ list$$

  where *list* is a function that returns a list of numbers.

$$\begin{array}{lcl} map\ f\ nil & = & nil \\ map\ f\ (x : l) & = & (f\ x) : (map\ f\ l) \end{array}$$

Exercise 4:

Consider the following logic program:

```
even(0).
even(N) :- N > 1, N1 is N-2, even(N1).
```

and goals:

```
:- even(4).
:- even(3).
```

- Assuming `N > 1` succeeds when `N` is instantiated by a number greater than 1 and fails otherwise, describe the SLD-resolution tree for each goal, using the computation rule that always selects the leftmost literal as the one resolved upon.
- What are the answers that Prolog gives for these goals?
- We now replace the second clause by:

  ```
  even(N) :- N1 is N-2, even(N1), N > 1.
  ```

  Explain how Prolog will behave now with the same goals as above. What will be the answers for these goals?