ML funkcionalno programiranje: rezime

Andrej Ivašković

Nedelja informatike, 2. april 2015.

1 Koncepti u funkcionalnom programiranju

```
(* rekurzija, liste, currying *)
fun nfact n = if n = 0 then 1
                       else n * nfact (n - 1);
fun fact n =
  let fun f (k, r) = if k = 0 then r
                              else f (k - 1, k * r);
  in f (n, 1)
  end;
fun obrni l =
  let fun akum ([], r) = r
        | akum (x::xs, r) = akum (xs, x::r);
  in akum (l, [])
  end;
fun mem [] _ = false
  | mem (x::xs) y = x = y orelse mem xs y;
(* funkcionali i primene *)
fun map _ [] = []
  | map f (x::xs) = (f x)::map f xs;
fun allcons x l = map (fn y => x::y) l;
fun filter _ [] = []
  | filter p (x::xs) =
      if p x then x::filter p xs
             else filter p xs;
fun intersec (l1, l2) = filter (mem l1) l2;
(* lenje liste *)
datatype 'a stream = Nil
                   | Cons of 'a * (unit -> 'a stream);
fun hdq (Cons(x, xs)) = x;
fun tlq (Cons(x, xs)) = xs();
fun from k = Cons(k, fn() => from (k + 1));
```

2 Zadaci za implementaciju

1. Napraviti ML funkciju koja određuje zbir $\sqrt{1} + \sqrt{2} + \ldots + \sqrt{n}$. Za rešavanje ovog zadatka je neophodno proučiti kako se radi sa bibliotekama sa implementiranim matematičkim funkcijama.

2. Implementirati algoritam brzog stepenovanja korišćenjem akumulatorskog argumenta.

3. (a) Napisati funkciju tipa $(\alpha \text{ list}) \times (\alpha \text{ list}) \to (\alpha \times \alpha \to \text{bool}) \to \alpha \text{ list}$ koja date dve sortirane liste spaja u jednu sortiranu listu. Kriterijum poretka je zadat funkcijom $(\alpha \times \alpha \to \text{bool})$.

   (b) Korišćenjem funkcije iz (a) napisati implementaciju *Merge sort* algoritma.

4. (a) Napraviti funkcional `lfold` tipa $(\alpha \times \beta \to \alpha) \times \alpha \to \beta \text{ list} \to \alpha$ koji, ukoliko mu se proslede argumenti (f, e) i $[x_1, \ldots, x_n]$, daje na izlazu $f\left(\ldots\left(f\left(e, x_1\right), x_2\right)\ldots, x_n\right)$.

   (b) Primenom ovog funkcionala odrediti zbir svih brojeva koji se nalaze u svim listama u nekoj datoj listi lista.

5. (a) Napisati funkciju za pristup n-tom članu lenje liste.

   (b) Za datu lenju listu čiji su elementi lenje liste odrediti j-ti element i-te liste.

   (c) Napraviti lenju listu lenjih lista celih brojeva koja predstavlja "tablicu množenja": vrednost j-tog elementa i-te liste je $i \cdot j$.

3 Pitanja za razmišljanje

1. Kako bi mogla da se implementiraju binarna stabla pretrage u ML-u? Šta je (možda) problematično?

2. Kako bi se implementirale funkcije višeg reda u C-u? Koja ograničenja postoje pri pravljenju ekvivalenta map (ukoliko pretpostavimo da je ekvivalent liste — niz)?

3. ML deluje kao da je jezik u kom nemamo ništa, ali se ispostavlja da već sada imamo previše stvari. Konkretno, brojevi su suvišni jer bismo mogli da ih *simuliramo*. Na koji način?