Programowanie Funkcyjne 2018

Lista zadań nr 7

28 listopada 2018

Zadanie 1 (5p). 1. Zdefiniuj operator stałopunktowy `fix` typu `(('a -> 'b)-> 'a -> 'b)-> 'a -> 'b`, który pozwoli na wyznaczanie punktu stałego funkcji typu `('a -> 'b)-> 'a -> 'b)`, a co za tym idzie na definiowanie rekurencyjnych funkcji bez użycia konstrukcji `let rec`. Np. silnię można wyrazić przy użyciu `fix` następująco:

```
fix (fun f -> fun n -> if n = 0 then 1 else n * (f (n-1)))
```

2. Nie używając rekursji (tj. konstrukcji `let rec`) zdefiniuj funkcję obliczającą silnię (użyj referencji). W podobny sposób zdefiniuj funkcję `fix`.

Zadanie 2 (3p). Rozważmy modyfikowalne listy zdefiniowane następująco:

```
type 'a list_mutable = LMnil | LMcons of 'a * 'a list_mutable ref
```

Zaimplementuj konkatenację list typu `'a list_mutable` na dwa sposoby:

1. funkcja `concat_copy` buduje listę wynikową kopiując pierwszy argument;
2. funkcja `concat_share` buduje listę wynikową bez kopiowania argumentów.

Zadanie 3 (6p). Technika memoizacji pozwala wykorzystać cechy imperatywne języka w celu zwiększenia efektywności działania funkcji, która sama jest czysto funkcyjna, tj. kolejne wywołanie takiej funkcji dla tego samego argumentu zwróci tę samą wartość. Memoizacja polega na zapamiętywaniu wartości wywołań funkcji dla konkretnych argumentów w pewnej strukturze danych, i na wyszukiwaniu potrzebnych wartości przy kolejnych wywołaniach tej funkcji. Aby umożliwić memoizację dowolnej jednoargumentowej funkcji (o typie argumentu, którego wartości OCaml potrafi porównywać), zaimplementuj następujący schemat:

- zdefiniuj typ polimorficzny służący jako tablica wartości wywołań dowolnej funkcji;
- napisz funkcję tworzenia pustej tablicy;
- napisz funkcję wyszukiwania w tablicy wartości funkcji dla zadanego argumentu;
- napisz funkcję dopisującą do tablicy nową wartość wywołania funkcji.

Wykorzystaj ten schemat do memoizacji funkcji wyznaczającej kolejne liczby Fibonacciego: napisz funkcję `fib : int -> int` według standardowej definicji oraz funkcję `fib_memo : int -> int` wykorzystującą memoizację. Porównaj czasy działania obu funkcji.

Czy funkcja `fib_memo` spełnia Twoje oczekiwania w kwestii efektywności? Zaimplementuj taką wersję funkcji `fib_memo`, która lepiej wykorzysta technikę memoizacji.

Zadanie 4 (3p). Napisz funkcję fresh typu `string -> string` generującą świeże nazwy, której kolejne wywołania mają następujący efekt:

```
# fresh "x";;
- : string = "x1"
# fresh "x";;
- : string = "x2"
# fresh "x";;
- : string = "x3"
# fresh "x";;
- : string = "x4"
```

itd... oraz funkcję `reset` typu `int -> unit`, która ustawia początkową wartość generowanego indeksu dla następnych wywołań funkcji `fresh`, np.

```
# fresh "x";;
- : string = "x1"
# fresh "x";;
- : string = "x2"
# reset 5;;
- : unit = ()
# fresh "x";;
- : string = "x6"
# fresh "x";;
- : string = "x7"
```

Uwaga! Funkcje nie mogą wykorzystywać żadnych zmiennych globalnych.

Zadanie 5 (3p). Rozwiąż problem Józefa, tj. zadanie 4 z listy kontrolnej do wykładu 6.