Programowanie funkcyjne 2014
Lista 1

kzi

14 października 2014

Wstęp. Załóżmy, że `T` jest typem w jakimś języku programowania, a $A_{\texttt{T}}$ jest zbiorem wszystkich wartości tego typu. W języku matematyki lista objektów typu `T` jest ciągiem elementów $A_{\texttt{T}}$. Ciągi nad zbiorem $A_{\texttt{T}}$ możemy zdefiniowany jako funkcja z N w $A_{\texttt{T}}$, gdzie $N = \{0, 1, \ldots, n\}$ dla pewnego $n \in \mathbb{N}$ lub $N = \mathbb{N}$.

Uwaga: Jeśli zadanie jest niedospecyfikowane, to braki w specyfikacji można uzupełnić w dowolny sensowny! sposób.

Zadania w OCaml'u

1. Zdefiniuj `'a lazylist` jako typ funkcji przyjmujących `int` i zwracajacych wartości typu `'a` oraz wyjątek `NIL` (Not In List). Zauważ, że każdą listę wartości typu `'a` możemy reprezentować funkcją typu `'a lazylist`, która zaaplikowana do k zwraca k'ty element tej listy, albo rzuca `NIL`, gdy k nie jest indeksem żadnego elementu z tej listy. `(1pkt)`
2. Zdefiniuj listę `ma : string lazylist` wszystkich napisów postaci $(\texttt{ma})^*$. Lista może się sypać, kiedy będziemy próbować wybrać z niej bardzo długi napis. Można skorzystać z funkcji `String.init`. `(2pkt)`
3. Zdefiniuj stałą `empty : 'a lazylist`, która reprezetuje listę pustą. `(1pkt)`
4. Zdefiniuj funkcję `append : 'a -> 'a lazylist -> 'a lazylist`, która zaaplikowana do wartości `e` i listy zwraca tą listę z `e` "dołożonym" na jej początku. `(1pkt)`
5. Zdefiniuj funkcję `head : 'a lazylist -> 'a`, która zwraca pierwszy element listy, a jeśli lista jest pusta rzuca `NIL`. `(1pkt)`
6. Zdefiniuj funkcję `tail : 'a lazylist -> 'a lazylist`, która zaaplikowana do listy zwraca tą listę, ale bez pierwszego elementu. `(1pkt)`
7. Zdefiniuj funkcję `is_empty : 'a lazylist -> bool`, która zwraca `true`, jeśli lista jest pusta i `false` wpp. `(1pkt)`
8. Zdefiniuj funkcję `length : 'a lazylist -> int`, która dla skończonej listy zwraca jej długość. `(2pkt)` (Czy da się napisać `length`, które dodatkowo będzie zwracało -1 dla wszystkich nieskończonych list? Odpowiedź na JFiZO.)
9. Zdefiniuj funkcję `eager_concat : 'a lazylist -> 'a lazylist -> 'a lazylist`, która zwraca konkatenację dwóch list. Dla wynikowej listy wybranie elementu powinno trwać asymptotycznie tyle, ile wybranie tego elementu z listy wejściowej, z której pochodzi. `(4pkt)`

Zadanie w Haskell'u

10. Zdefiniuj funkcję `bound_cut :: (a -> Bool) -> Integer -> [a] -> (Bool, [a], [a])`, która zaaplikowana do argumentów p k l zwraca trójkę:

    $$(b, \quad \text{prefiks listy } l \text{ do } m\text{'tego elementu włącznie}, \quad \text{reszta listy } l)$$

    gdzie m jest najmniejszą liczbą naturalną z przedziału $[0, k]$ taką, że m'ty element listy l spełnia warunek p lub $m = k$, oraz $b =$ `True`, gdy m'ty element l spełnia p, oraz $b =$ `False` wpp. Jedyne operacje na listach, których wolno Ci użyć to: `head`, `tail`, `null`, `:`, `[...]`. `(5pkt)`

    a) Podaj jakiś praktyczny przykład zastosowania tej funkcji. `(2pkt)`

Wskazówka: Może przydać się zerkniecie do dokumentacji biblioteki Data.List (Haskell).
Wskazówka: Listy w Haskell'u mogą być nieskończone, podobnie jak nasze `lazylist`*.*