Teoretyczne podstawy języków programowania

Lista 4

3.11.2004

Zadanie 1. Zdefiniuj w ET funkcje `div` i `mod`, obliczające odpowiednio iloraz i resztę z dzielenia całkowitego liczb naturalnych. Ponieważ mają to być funkcje całkowite, to przypadki dzielenia przez zero definiujemy następująco: `m div 0 = 0`, `m mod 0 = m`. Zauważ, że równość `m = n * (m div n) + (m mod n)` jest zachowana.

Zadanie 2. Napisz w ET funkcję `quicksort` sortującą listy z wykorzystaniem algorytmu sortowania szybkiego. Jakie inne algorytmy sortowania potrafisz zapisać w ET?

Zadanie 3. Zdefiniuj w ET funkcję `fib`, obliczającą wartość n-tej liczby Fibonacciego.

```
fib 0 = 0
fib 1 = 1
fib (n+2) = fib(n+1) + fib(n)
```

Zadanie 4. Typ dla nieskończonych list polimorficznych można zdefiniować w ET następująco:

```
codatatype STREAM 'a = Shd to 'a & Stl to STREAM 'a;
```

Napisz specyfikacje równościowe dla poniższych funkcji, zdefiniuj je w ET używając koiteratora lub korekursora dla strumieni i zweryfikuj poprawność swoich definicji.

a) `smap : ('a -> 'b) -> (STREAM 'a) -> (STREAM 'b)` aplikuje pierwszy argument do wszystkich elementów strumienia.

b) `natstr : N -> STREAM N` tworzy strumień kolejnych liczb naturalnych, zaczynający się od zadanej liczby.

c) `sprefix : 'a -> (STREAM 'a) -> STREAM 'a` dodaje pierwszy argument na początek strumienia (odpowiednik `Cons` dla zwykłych list).

Czy potrafisz w ET zdefiniować sito Eratostenesa?

Zadanie 5. Bezpośrednia definicja kombinatora $Y \equiv \lambda f.(\lambda x.f(xx))(\lambda x.f(xx))$ w języku SML lub OCaml spowoduje błąd typu. Możemy jednak zdefiniować typ danych (tu w notacji SML):

```
datatype 'a coded = Encode of ('a coded -> 'a);
```

a następnie funkcjonał `fixl : ('a -> 'a) -> 'a`

```
fun fixl f = let fun g(Encode x) = f (x (Encode x)) in g (Encode g) end;
```

który działa poprawnie dla dla języków z wartościowaniem leniwym. W SMLu (i OCamlu), w których programy są wartościowane gorliwie, wartościowanie `fixl` nie kończy się i nie uda się nam zdefiniować np. nieskończonej listy jedynek (sprawdź to!):

```
fixl (fn x => 1::x);
```

Można jednak nieco zmodyfikować ten funkcjonał i otrzymać funkcjonał

```
fix : (('a -> 'b) -> 'a -> 'b) -> 'a -> 'b,
```

który znajduje punkty stałe funkcji (ale już nie list!). Przekonaj się o tym, definiując funkcję obliczającą silnię zadanej liczby bez jawnego użycia rekursji.

Funkcjonał `fix : (('a -> 'b) -> 'a -> 'b) -> 'a -> 'b` można też zdefiniować bez użycia typu `'a coded`. Spróbuj znaleźć taką definicję.

Zadanie 6. Udowodnij, że w rachunku lambda regułę (η) można zastąpić poniższą regułą ekstensjonalości:

$$\frac{Mx = Nx}{M = N}\ (ext) \quad x \notin FV(M) \cup FV(N)$$

Pokaż, że w $\lambda\beta\eta$ można wywieść regułę (ext) i odwrotnie, w $\lambda\beta + (ext)$ można wywieść (η).

Zadanie 7. Udowodnij, że $\lambda\beta\eta \vdash (\lambda^* x.M)_\lambda = \lambda x.M$.

Wskazówka. Przez indukcję po strukturze M udowodnij, że $(\lambda^* x.M)_\lambda\, x = M$. Stąd wynika, że $\lambda x.(\lambda^* x.M)_\lambda\, x = \lambda x.M$ (dlaczego?) i na podstawie reguły (η) otrzymujemy $(\lambda^* x.M)_\lambda = \lambda x.M$.

Zadanie 8. Udowodnij, że $CL \vdash P = Q$ implikuje $\lambda\beta \vdash P_\lambda = Q_\lambda$.

Wskazówka. Indukcja po długości dowodu $P = Q$.