Programowanie Funkcyjne 2018

Lista zadań nr 1 dla grupy TWI

Na zajęcia 11 października 2018

Zadanie 1 (2p). Jaki jest typ wyrażenia `fun x -> x`? Napisz wyrażenie, którego wartością też jest funkcja identycznościowa, ale które ma typ `int -> int`. Napisz wyrażenia, których typami są:

```
('a -> 'b) -> ('c -> 'a) -> 'c -> 'b
'a -> 'b
'a -> 'b -> 'a
'a -> 'a -> 'a
```

Czy potrafisz napisać wyrażenie typu `'a`?

Zadanie 2 (2p). Napisz dwie wersje funkcji (w tym jedną za pomocą rekursji ogonowej), która oblicza n-ty wyraz ciągu zdefiniowanego wzorami:

$$\begin{aligned} a_0 &= 0 \\ a_{n+1} &= 2a_n + 1, \end{aligned}$$

a następnie porównaj szybkość ich działania dla dużych n.

Zadanie 3 (2p). Zdefiniuj funkcję `<.>` wyznaczającą złożenie funkcji, tj. spełniającą tożsamość:

$$(f \mathbin{\texttt{<.>}} g)\; x \;=\; f(g(x))$$

dla dowolnych funkcji f i g oraz argumentu x odpowiednich typów. Wykorzystaj ją do zdefiniowania takiej funkcji `<^>`, że

$$(f \mathbin{\texttt{<\^{}>}} n)\; x \;=\; \underbrace{f(f(\ldots(f(}_{n \text{ razy}}x))\ldots))$$

dla dowolnej funkcji f i argumentu x odpowiednich typów oraz nieujemnej liczby całkowitej n. Za pomocą tej funkcji zdefiniuj operację mnożenia liczb wykorzystując operację dodawania oraz operację `<**>` potęgowania liczb wykorzystując operację mnożenia. (Zauważ, że operatory infiksowe są zwykłymi funkcjami dwóch zmiennych!)

Zadanie 4 (8p). Strumień (tj. nieskończony ciąg) elementów typu `t` możemy reprezentować za pomocą funkcji `s: int -> t` w taki sposób, że wartością wyrażenia `s 0` jest pierwszy element strumienia, wyrażenia `s 1` — drugi itd. Używając powyższej reprezentacji zdefiniuj następujące funkcje działające na strumieniach (tam, gdzie to możliwe, funkcje te powinny być polimorficzne, tj. powinny działać na strumieniach o elementach dowolnego typu):

- `hd`, `tl` — funkcje zwracające odpowiednio głowę i ogon strumienia,
- `add` — funkcja, która dla zadanego strumienia tworzy nowy strumień, którego każdy element jest większy o zadaną stałą od odpowiadającego mu elementu oryginalnego strumienia,
- `map` — funkcja, która dla zadanego strumienia tworzy nowy strumień, którego każdy element jest wynikiem obliczenia zadanej funkcji dla argumentu będącego odpowiadającym mu elementem oryginalnego strumienia (tak, jak `map` dla list skończonych),

- `map2` — jak wyżej, ale dla podanej funkcji dwuargumentowej i dwóch strumieni,
- `replace` — funkcja, która dla zadanego indeksu `n`, wartości `a` i strumienia `s` zastępuje co `n`-ty element strumienia `s` przez wartość `a` i zwraca powstały w ten sposób strumień,
- `take` — funkcja, która dla zadanego indeksu `n` i strumienia `s` tworzy nowy strumień złożony z co `n`-tego elementu strumienia `s`,
- `scan` — funkcja, która dla zadanej funkcji `f: 'a -> 'b -> 'a`, wartości początkowej `a: 'a` i strumienia `s` elementów typu `'b` tworzy nowy strumień, którego każdy element jest wynikiem „zwinięcia" początkowego segmentu strumienia `s` aż do bieżącego elementu włącznie za pomocą funkcji `f`, tj. w strumieniu wynikowym element o indeksie `n` ma wartość

  ```
  (f (...(f (f a (s 0)) (s 1)) ...) (s n)),
  ```

- `tabulate` — funkcja tablicowania strumienia, której wynikiem powinna być lista elementów strumienia leżących w zadanym zakresie indeksów.

Zdefiniuj przykładowe strumienie i przetestuj swoją implementację.

W definicji funkcji `tabulate` wykorzystaj możliwość definiowania parametrów opcjonalnych dla funkcji — niech początek zakresu indeksów będzie opcjonalny i domyślnie równy 0. *Przykład:* pierwszy argument funkcji `f` w deklaracji

```
let f ?(x=0) y = x + y
```

ma etykietę `x` i jest opcjonalny, a jego wartość domyślna wynosi 0. Wyrażenie `(f 3)` jest równoważne wyrażeniu `(f ~x:0 3)` (i ma wartość 3), zaś wartością wyrażenia `(f ~x:42 3)` jest 45.

Zadanie 5 (3p). Ile różnych wartości mają zamknięte wyrażenia typu `'a -> 'a -> 'a`, których obliczenie nie wywołuje żadnych efektów ubocznych (tj. które zawsze kończą działanie, nie wywołują wyjątków, nie używają operacji wejścia/wyjścia itp.)? Okazuje się, że jest ich dokładnie tyle, ile potrzeba, by reprezentować za ich pomocą wartości logiczne prawdy i fałszu! Zdefiniuj odpowiednie wartości `ctrue` i `cfalse` typu `'a -> 'a -> 'a` oraz funkcje o podanych niżej sygnaturach implementujące operacje koniunkcji i alternatywy oraz konwersji między naszą reprezentacją a wbudowanym typem wartości logicznych.

- `cand, cor: ('a -> 'a -> 'a) -> ('a -> 'a -> 'a) -> 'a -> 'a -> 'a`
- `cbool_of_bool: bool -> 'a -> 'a -> 'a`
- `bool_of_cbool: (bool -> bool -> bool) -> bool`

Zastanów się, czemu typy niektórych z powyższych funkcji znalezione przez algorytm rekonstrukcji typów różnią się od podanych powyżej.

Zadanie 6 (3p). Wartościami zamkniętych wyrażeń typu `('a -> 'a) -> 'a -> 'a` są wszystkie funkcje postaci $(f, x) \mapsto f^n(x)$ dla $n \in \mathbb{N}$. Wartości tego typu mogą więc reprezentować liczby naturalne. Zdefiniuj liczbę zero, operację następnika, operacje dodawania i mnożenia, funkcję sprawdzającą, czy dana liczba jest zerem, a także konwersje między naszą reprezentacją a wbudowanym typem liczb całkowitych (nie przejmuj się liczbami ujemnymi):

- `zero : ('a -> 'a) -> 'a -> 'a`
- `succ : (('a -> 'a) -> 'a -> 'a) -> ('a -> 'a) -> 'a -> 'a`
- `add, mul : (('a -> 'a) -> 'a -> 'a) ->`
  `(('a -> 'a) -> 'a -> 'a) -> ('a -> 'a) -> 'a -> 'a`
- `isZero : (('a -> 'a) -> 'a -> 'a) -> 'a -> 'a -> 'a`
- `cnum_of_int : int -> ('a -> 'a) -> 'a -> 'a`
- `int_of_cnum : ((int -> int) -> int -> int) -> int`

Zastanów się, czemu typy niektórych z powyższych funkcji znalezione przez algorytm rekonstrukcji typów różnią się od podanych powyżej.

Niniejszy tekst jest kompilacją listy zadań przygotowanej przez Filipa Sieczkowskiego.