MP24 @ II UWr 11 czerwca 2024 r.

Lista zadań nr 14

Zadanie 1. (2 pkt)

Na wykładzie zauważyliśmy, że wyliczenie rekurencyjnych definicji może się nie powieść, bo definiowana wartość może być potrzebna do wyliczenia jej samej. Można to zaobserwować np. w poniższym wyrażeniu.

```
let rec ones = Cons(1, ones) in ...
```

Celem zadania jest napisanie funkcji sprawdzającej, czy rekurencyjna definicja zawsze się policzy poprawnie. Oczywiście jest to problem nierozstrzygalny, więc zadowolimy się następującym przybliżeniem: grupa wzajemnie rekurencyjnych definicji

```
let rec x_1 = e_1
and x_2 = e_2
...
and x_n = e_n
in
...
```

jest poprawna, jeśli każde z wyrażeń $e_1, \ldots, e_n$ jest wartością (stałą, zmienną, funkcją, parą wartości, albo konstruktorem zaaplikowanym do wartości), a wszystkie wystąpienia zmiennych $x_1, \ldots, x_n$ w $e_1, \ldots, e_n$ znajdują się wewnątrz jakiejś funkcji. Dlaczego takie przybliżenie jest dobre?

Zadanie 2. (2 pkt)

Wyprowadź typy poniższych wyrażeń, a jeśli to niemożliwe, pokaż w którym miejscu algorytm rekonstrukcji typów pokazany na wykładzie zawiedzie.

- `fun x -> x`
- `fun x -> x x`
- `fun x y z -> x z (y z)`

- `fun f y -> f 42 (f true y)`
- `fun f x y -> f x (f x y)`

Zadanie 3. (2 pkt)

Rozszerz inferencję typów z wykładu (w którymkolwiek z wariantów) o obsługę operatorów binarnych.

Zadanie 4. (1 pkt)

Rozszerz inferencję typów z wykładu (w którymkolwiek z wariantów) o obsługę wyrażeń warunkowych.

Zadanie 5. (1 pkt)

Rozszerz inferencję typów z wykładu (w wariancie SIMPLE) o obsługę par (węzły `Pair(_, _)`, `Fst _` oraz `Snd _`).

Zadanie 6. (1 pkt)

Powtórz poprzednie zadanie w wariancie INFER.

Zadanie 7. (2 pkt)

Rozszerz inferencję typów z wykładu (w wariancie INFER) o obsługę rekurencyjnych definicji (węzeł `LetRec(_, _)`). Zauważ, że w tym wariancie można najpierw rozszerzyć środowisko, zgadując typy wzajemnie rekurencyjnych definicji, a potem je sprawdzić w już rozszerzonym środowisku.

Zadanie 8. (1 pkt)

Rozszerz inferencję typów z wykładu (w wariancie INFER) o obsługę wyjątków (konstrukcje `Try(_, _)` i `Raise`). A jak to zrobić w wariancie SIMPLE?