IIT CS440: Programming Languages and Translators

Homework 6: STLC and Continuations

Prof. Stefan Muller
TA: Xincheng Yang

Out: Tuesday, Apr. 13
Due: Thursday, Apr. 27 11:59pm CST

This assignment contains 5 written tasks and 2 programming tasks for a total of 42 points.

0 Logistics and Submission - Important

1. Make sure you read and understand the collaboration policy on the course website.
2. This assignment contains a mix of written and programming questions. For the written questions, submit typed or *neatly* handwritten and scanned answers on in .pdf, .doc, or .docx format.
3. You will answer the programming questions in `hw6.ml`. As usual, don't touch any line beginning with `(*>*`, and don't change the names or types of any functions in the file.
4. Submit both `hw6.ml` and your written answers on Blackboard under HW6.

1 STLC

In this section, refer to the syntax and typing rules for the Simply-typed λ calculus, given in lecture.

Task 1.1 (Written, 12 points).

Give the STLC types of the following expressions:

(a) $\lambda x : \mathsf{unit}.\lambda y : \mathsf{unit} \times \mathsf{unit}.x$

(b) $(\lambda x : \mathsf{unit}.x, ())$

(c) $\lambda x : \mathsf{unit} \times \mathsf{unit}.\mathsf{fst}\ x$

(d) $(\mathsf{fst}\ (\lambda x : \mathsf{unit}.x, \lambda x : \mathsf{unit}.x))\ ()$

Task 1.2 (Written, 9 points).

Give a typing derivation for

$$\emptyset \vdash (\lambda x : \mathsf{unit}.(x, x))\ () : \mathsf{unit} \times \mathsf{unit}$$

Continued on the next page.

2 Continuation-Passing Style

Task 2.1 (Written, 9 points).

For each of the functions below, say whether it's in CPS. If not, explain why not in a sentence or two.

(a) `let callfst f g k = f 0 k`

(b) `let exn a k e = match a with Some a -> k a | None -> e ()`

(c) `let app f x k = k (f x)`

Task 2.2 (Programming, 6 points).

Write a function `compose` that takes two CPS functions `f: 'a -> ('b -> 'k) -> 'k` and `g: 'b -> ('c -> 'k) -> 'k`, and returns their composition, which is also a CPS function of type `'a -> ('c -> 'k) 'k`. The composed function should take its input `a`, call `f` on it, then call `g` on the result before passing it to `k`.

For example,

`compose (fun a k -> k (a + 1)) (fun b k -> k (b * 2)) 1 (fun x -> x) = 4`

Your function must be in CPS; all call must be tail calls.

Task 2.3 (Programming, 6 points).

Write a function `prod: int list -> (int -> 'k) -> 'k`. The call `prod l k` should call `k` with the result of multiplying together all of the integers in `l`. We will assume the product of the empty list is 1, so `prod [] k` will call `k` with 1, and, e.g., `prod [1; 2; 3] k` will call `k` with $1 * 2 * 3 = 6$.

Here's the catch: If at any point in the list, you encounter a 0, you should **immediately call** `k` with 0, since you know that the final result will be 0 regardless of what is in the rest of the list.

Your function must be in CPS, with the exception that you can call the normal `*` (integer multiplication) function and use its result as usual; **all other calls must be tail calls.**

3 Standard Final Questions

Task 3.1 (Written, 0 points).

How long (approximately, in hours/minutes of actual working time) did you spend on this homework, total? Your honest feedback will help us with future homeworks.

Task 3.2 (Written, 0 points).

Who, if anyone, did you collaborate with (and in what way), and what outside sources, if any, did you consult in working on this homework?