Prof. P. Thiemann, S. Wehr Summer Term 2009

Concepts of Programming Languages

http://proglang.informatik.uni-freiburg.de/teaching/konzepte/2009ss/

Exercise Sheet 9

2009-06-25

Exercise 1 (4 points)
On the homepage of the lecture, you find code of the procedures `list-length`, `remove-first`, `occurs-free?`, and the pair of mutual recursive procedures `subst` and `subst-in-s-exp`. Rewrite each of these procedures f into a procedure `f/k` such that `f/k` is in CPS.

Exercise 2 (3 points)
Figure 6.3 on page 204 of EOPL defines the grammar of a language called CPS-IN. Write a Scheme procedure `tail-form?` that takes the syntax tree of a program in CPS-IN and checks whether the program is in tail form.

Exercise 3 ((3+6) points)
Transforming a program into CPS mainly serves two purposes: (1) make control contexts explicit, and (2) make evaluation order explicit by introducing names for intermediate results. The latter is called *sequentialization*.

If we do not care about explicit control contexts, we can sequentialize a program by converting it into *A-normal form* (short: *ANF*). In a nutshell, programs in ANF form use `let` expressions to name all intermediate results. Here is a procedure in ANF for computing the n-th fibonacci number:

```
(define fib/anf
  (lambda (n)
    (if (< n 2)
        1
        (let ((val1 (fib/anf (- n 1)))
              (val2 (fib/anf (- n 2))))
          (+ val1 val2)))))
```

(a) Design the grammar of a language ANF-OUT in which evaluation order is explicit.

(b) Write a program that translates expressions of the language CPS-IN into ANF-OUT.

Submission
Via email to `wehr@informatik.uni-freiburg.de`. The strict submission deadline is **2009-07-02, 1 pm.**