Fachbereich Mathematik und Informatik
Prof. Dr. K. Ostermann

Sebastian Erdweg, seba@informatik
Tillmann Rendel, rendel@informatik

Philipps Universität Marburg

November 13, 2009

Programming Languages and Types
Homework Assignment 5

Please hand in your homework by email to `mailto:pllecture@informatik.uni-marburg.de` until November 19.

Use `http://www.informatik.uni-marburg.de/~kos/teaching/pl/h05.ss` from the course homepage as starting point for this homework assignment. This file contains a version of the FAE and OOE interpreters with `FAE-` or `OOE-` prefixed to every function, so that the interpreters can be used in the same scheme session.

H5.1 Free Variables

Remember that the free variables of a term are all variables which occur in the term, but are not bound by `fun` (or `with`) in the term. For example, `{fun {x} {f (y + x)}}` has the free variables `f` and `y`, but `x` is not free in `{fun {x} {f (y + x)}}`, because it is bound by `fun`.

Write a function `FAE-free-variables` which returns a list of the free variables found in an `FAE` expression. Add some testcases to check your implementation.

H5.2 Encoding Functions as Objects

A function (in `FAE`) can be encoded as an object (in `OOE`) with a single method `apply`, which does whatever the function would have done. For example, the `FAE`-term

```
{{fun {x} {x + 3}} 15}
```

can be transformed into the `OOE`-term

```
new EncodedFunction().apply(15)
```

in the following class environment.

```
class EncodedFunction {
  apply(x) {
    return (x + 3);
  }
}
```

please turn the page!

Transform the following FAE terms into an OOE term and a class environment each.

1. ```
   {with {f {fun {x} {x + 9}}}
     {f 16}}
   ```
   Hint: Desugar first.

2. ```
   {with {x 3}
     {with {f {fun {y} {x + y}}}
       {with {x 5}
         {f 4}}}}
   ```
   Hint: Remember to close over the free variables.

H5.3 FAE to OOE translation

Write a function FAE->OOE which translates from FAE terms to OOE terms and class environments.

```
(define-type Exp*Classenv
  [e*c (exp OOE?) (classenv (list-of? OOE-Class?))])

;; FAE->OOE : FAE -> Exp*Classenv
(define (FAE->OOE exp)
  ...)
```

You can test your translation on programs which produce numbers. The original FAE program and the resulting OOE program after translation should produce the same numbers with their respective interpreters.

```
(define original
  (FAE-with 'x (FAE-num 3)
         (FAE-with 'f (FAE-fun 'y (FAE-add (FAE-id 'x) (FAE-id 'y)))
               (FAE-with 'x (FAE-num 5)
                     (FAE-app (FAE-id 'f) (FAE-num 4))))))

(define transformed
  (FAE->OOE (desugar original)))

(test (FAE-numV-n (FAE-interp (desugar original)
                              (FAE-emptyEnv)))
      (OOE-numV-n (OOE-interp (e*c-exp transformed)
                              (e*c-classenv transformed)
                              (OOE-emptyEnv))))
```

Hint: You may need to add additional arguments to FAE->OOE.