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

Concepts of Programming Languages

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

Exercise Sheet 3

2009-05-07

Hints:

- When requested to modify or extend a language from the EOPL ("Essentials of Programming Languages") book, you should always extend the original code (available from the EOPL homepage `http://www.eopl3.com/code.html`). Please mark your changes clearly!
- The code from the EOPL book comes with an extensive test suite. When modifying code, you must adapt this test suite and extend it with your own tests!
- The SLLGEN parsing system (used by code from the EOPL book) is explained in Appendix B of the book.

Exercise 1 ((5+5) points)
Extend the language LET as presented in the EOPL book in the following ways:

(a) A `let` declaration can declare an arbitrary number of variables, using the following grammar:

$$\textit{Expression} ::= \texttt{let}\ \{\textit{Identifier}\ \texttt{=}\ \textit{Expression}\}^*\ \texttt{in}\ \textit{Expression}$$

As in Scheme's `let`, each of the right-hand sides is evaluated in the current environment, and the body is evaluated with each new variable bound to the value of its associated right-hand side. For example

```
let x = 30
in let x = -(x,1)
       y = -(x,2)
   in -(x,y)
```

should evaluate to 1.

(b) Extend the language with a `let*` expression that works like `let*` in Scheme. For example

```
let x = 30
in let* x = -(x,1)
        y = -(x,2)
   in -(x,y)
```

should evaluate to 2 because the variable `x` on the right-hand side of the binding for `y` is 29 not 30.

Exercise 2 ((1+5 points))

Dynamic binding (or *dynamic scoping*) is an alternative design in which the procedure body is evaluated in an environment obtained by extending the environment at the point of call. (In the lecture, we used *lexical binding*, in which the procedure body is evaluated in an environment obtained by extending the environment at the point where the procedure is defined.)

For example in

```
let a = 3
in let p = proc (x) -(x,a)
       a = 5
   in -(a, (p 2))
```

the `a` in the procedure body is bound to 5, not 3.

(a) What is the result of evaluating the expression above under dynamic binding? What is the result under lexical binding?

(b) Modify the PROC language from the EOPL book to support dynamic binding instead of lexical binding. Use the procedural representation for procedures when modifying the code.

Submission

Via email to `wehr@informatik.uni-freiburg.de`. Please submit in pairs of two. Your code must not raise errors when pressing DrScheme's "Run" bottom.

Please adhere to the following conventions:

- If your submission consists of a single scheme file, the name of this file should be $name_1$`-`$name_2$`-ex`n`.scm` where $name_1$ and $name_2$ are your names and n is the number of the sheet.
- If your submission consists of several file, you must place them into a `.zip` or `.tar.gz` archive. The name of this archive should be $name_1$`-`$name_2$`-ex`n`.zip` (replace `.zip` with `.tar.gz` if you are submitting a `.tar.gz` file), where $name_1$ and $name_2$ are your names and n is the number of the sheet. Inside the archive, their must be a single directory of name $name_1$`-`$name_2$`-ex`n ($name_1$, $name_2$, and n as before). This directory should then contain your actual submission files. **Do not place your submission files at the top-level of the archive!**

The strict submission deadline is **2009-05-14, 2 pm.**