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

Concepts of Programming Languages

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

Exercise Sheet 6

2009-05-28

Exercise 1 ((4+4) points)
Consider the following language supporting top-level functions and conditionals:

$$\begin{array}{lcll} & v & \in & \mathsf{Var} \\ & f & \in & \mathsf{Fun} \\ \mathsf{Exp} & e & ::= & v \mid \mathtt{n} \mid e\mathtt{+}e \mid e\mathtt{*}e \mid \mathtt{zero?}(e) \mid \mathtt{if}\ e\ \mathtt{then}\ e\ \mathtt{else}\ e \mid f\mathtt{(}e\mathtt{)} \\ \mathsf{Def} & d & ::= & f\mathtt{(}v\mathtt{)}\ \mathtt{=}\ e \\ \mathsf{Prog} & p & ::= & d^*\ e \end{array}$$

Var and Fun are disjoint, unspecified sets of variable and function symbols. Expressions e comprise variables v, constants n for numbers in $\mathbb{Z}$, addition e+e and multiplication e*e, tests against null zero?(e), conditionals if e then e else e, and function calls of the form f(e). For simplicity, a function definition d binds exactly one variable. A program p consists of a list of definitions d^* and a "main expression" e.

By convention, we encode booleans as elements from $\mathbb{Z}$ such that 0 corresponds to *false* and all $x \neq 0$ correspond to *true*. We assume that the set of constant symbols consists of $\ldots, \mathtt{-2}, \mathtt{-1}, \mathtt{0}, \mathtt{1}, \mathtt{2}, \ldots$. Moreover, there exists a function $\mathcal{C}$ mapping constant symbols to their corresponding number in $\mathbb{Z}$.

(a) Define a big-step style semantics for this language using call-by-value for parameter passing. The evaluation relation has the form $d^*, \rho \vdash e \hookrightarrow y$, where d^* is a list of function definition, $\rho \in \mathsf{Env} = \mathsf{Var} \mapsto \mathbb{Z}$, $e \in \mathsf{Exp}$, and $y \in \mathbb{Z}$.

(b) Consider the function definition:

```
fac(v) = if zero?(v) then 1 else fac(v + -1) * v
```

Using your big-step semantics to prove that fac(n) evaluates to $\mathcal{C}(\mathtt{n})!$ for all n such that $\mathcal{C}(\mathtt{n}) \in \mathbb{N}$.

Exercise 2 (4 points)
Define a small-step semantics for the language defined in the preceding exercise. This time, use call-by-name for parameter passing.

Exercise 3 (4 points)
Extend the language from exercise 1 with explicit references as described in chapter 4.2 of the EOPL book. Define a small-step semantics using call-by-value parameter passing.

Submission
On paper (please don't send me emails). The strict submission deadline is **2009-06-15, 2:15 pm** (before the lecture).