Types and Programming Languages, Exercise 1

Philip Wadler

Issued: Monday 29 January 2007. Due: Monday 19 February 2007.

TAPL refers to *Types and Programming Languages* by Benjamin Pierce.

1. TAPL, Exercise 3.2.4
2. TAPL, Exercise 3.2.5
3. TAPL, Exercise 3.5.17
4. TAPL, Exercise 3.5.18
5. TAPL, Exercise 4.2.2
6. Write an implementation of untyped call-by-value lambda calculus based on the following principles:
   - use a big-step semantics;
   - use an environment rather than substitution at each step;
   - use variable names rather than de Bruijn indices;
   - include booleans and naturals as base types, with the operations given in the text (boolean constants, conditionals, zero, successor, predecessor, test for zero).
7. Using your implementation above, write and test the following:
   (a) addition, multiplication, and exponentiation on naturals. (Hint: you will need to use the fixpoint combinator Y to support recursion.)
   (b) addition, multiplication, and exponentiation on Church numerals. (Hint: do not use the fixpoint combinator.)
   (c) functions to convert a Church numeral to the corresponding natural, and vice versa.
8. Update your answer to Exercise 6 to typed lambda calculus, plus the typed fixpoint operator `fix` (as in TAPL 11.11).
9. Update your answer to Exercise 7 to typed lambda calculus. (Hint: you may need to assign Church numerals different types depending on how they are used.)