CS520 Programming Assignment 1

Posted: 12 Sept 2006
Due: 26 Sept 2006

Overview The purposes of this assignment are:

1. Learn how to program in Ocaml.
2. Implement an evaluator for a small un-typed lambda calculus.

Definitions In this assignment, we are dealing with an un-typed lambda calculus called λ_0 whose abstract syntax is defined in Ocaml as follows:

```
type term0 = TmVar0 of string           (* variables *)
           | TmLam0 of string * term0   (* abstraction *)
           | TmApp0 of term0 * term0    (* application *)
```

A parser is provided to parse the concrete syntax of λ_0 to the abstract syntax tree (as defined above). For example, the *identity* term can be written as

```
lam (x) => x
```

(which is $\lambda x.x$ in the notation of the book) while the abstract syntax tree is

```
TmLam0 ("x", TmVar0 "x")
```

As another example, the term

```
lam (x) => lam (y) => y x
```

has the following abstract syntax tree:

```
TmLam0 ("x", TmLam0 ("y", TmApp0 (TmVar0 "y", TmVar0 "x")))
```

Problem 1 (40pts): Implement a function called *subst* in Ocaml which performs substitution for λ_0. Note that the *subst* function should apply to arbitrary λ_0 terms, therefore, α*-renaming* must be implemented correctly. The *subst* function should be assigned the following type in Ocaml:

$$subst \ : \ term0 \rightarrow string \rightarrow term0 \rightarrow term0$$

Given two terms t_0, t_1 and a string s, $subst(t_0)(s)(t_1)$ denotes $[s \mapsto t_1]t_0$, namely, substituting each free occurrence of the variable named s in t_0 by t_1.

Problem 2 (20pts): Implement a function called *isClosed* in Ocaml which determines whether a λ_0 term is *closed* or not. Given some term t, $isClosed(t)$ returns *true* if t is

closed, otherwise, returns *false*. The *isClosed* function should be assigned the following type in Ocaml:

$$isClosed \ : \ term0 \rightarrow bool$$

Problem 3 (30pts): Implement a function called *eval* in Ocaml which *evaluates closed* λ_0 *terms through the call-by-value strategy*. The *eval* function should have type

$$eval \ : \ term0 \rightarrow term0$$

in Ocaml.

Implementation notes A few files (in **prog1.tar.gz**) are provided to start the assignment. You need to provide the actual implementations of the above functions based on the given code. Once all the code are ready, type **make** under the directory. If no error reported, an executable file called **evaluator** will be produced. You can test your code by typing

```
./evaluator filename
```

where *filename* should be replaced by some actual file path.

Grading The grading of the assignment is based on whether the required functionalities are correctly implemented. Please make sure your code can be compiled and tested on **csa2** because all submissions will be tested on **csa2**. There are **10pts** for

1. if the code is well organized.
2. if errors are properly handled.
3. if the code has necessary comments.