Simple OCaml Introduction

Florian E. Muecke

August 17, 2006

Abstract

This is a simple Objective CAML introduction that shows some basic elements of this object oriented functional programming language.

1 Introduction

A complete tutorial can be found here: www.ocaml-tutorial.org/.

2 Examples

Brackets are used in OCaml only to pack arguments together, never to seperate the function from its parameters like in `C`. Comments are also defined by brackets with stars like `(* this is a comment *)`. Complete instructions (functions) should always end with '`;;`'.

2.1 A simple function

Program:
```
let average a b = (a +. b) /. 2.0;;
```
Output:
```
val average : float -> float -> float = <fun>
```
Run:
```
# average 3.0 4.0;;
- : float = 3.5
```

The `+.` and `\.` in the function definition means that the arguments `a` and `b` have to be of type float. In OCaml there are different functions to convert one type to another (like explicit type casts in `C`): `float_of_int, int_of_float, char_of_int, int_of_char, string_of_int` and so on.

2.2 Recursive function

You want to define a recursive function, you have to do so using `let rec`.

```
let rec range a b =
  if a > b then []
  else a :: range (a+1) b
  ;;
```

2.3 Polymorphic functions (with output)

```
# let returnString x = "Hossa!";;
val returnString : 'a -> string = <fun>
# returnString 5;;
- : string = "Hossa!"
```

2.4 References

You can get the *reference* of an object `x` by using `ref x`. You can assign references with '`:=`' like in `x := "abc"` and dereference objects by using an '`!`' like in `!x`.