

# Principles of Programming, Fall 2009

## Practice 10

### OCaml functors

Woosuk Lee, Suwon Jang, Sungkeun Cho  
Programming Research Lab.@SNU

November 16, 2009

We will define a functor for manipulating two-dimensional vectors (pairs of (x,y) coordinates) that can be instantiated with different types for the coordinates.<sup>1</sup>

Arguments have the following type.

```
type arg = Rational of int * int
         | Float of float
         | Complex of float * float
```

Numbers have the following signature.

```
module type NUMBER =
sig
  type num
  val create : arg -> num
  val add : num -> num -> num
  val string_of : num -> string
end
```

1. Define three structures `Rational`, `Float` and `Complex` implementing the signature `NUMBER`.

e.g.

```
module Complex : NUMBER =
struct
  type num = float * float (* e.g. 2.0 + 3.0i *)
  let create = ...
  let add = ...
  let string_of = ...
end
```

---

<sup>1</sup>Emmanuel Chailloux et al., *Developing Applications With Objective Caml*, p431, O'Reilly, Paris, 2000.

2. Vectors have following signature `VECTOR`.

```
module type VECTOR =
sig
  type atom (* Vector atom type *)
  type vector
  val create : atom list -> vector
  val add : vector -> vector -> vector
  val string_of : vector -> string
end
```

Define the functor `MakeVector`, parameterized by a module of signature `NUMBER` and output structure has signature `VECTOR`.

e.g.

```
module MakeVector(Number : NUMBER) : VECTOR
  with type atom = Number.num =
struct
  ...
end
module ComplexVector = MakeVector(Complex)
```

3. Use these structures to define (by functor application) three modules for vectors of rationals, reals and complex.