Polymorphic variant types

Variant types can be polymorphic in one or more type variables

Example: type `'a option` of optional values of type `'a`

```
type 'a option = None | Some of 'a;; (* bult-in type in OCaml *)
```

Module Option

Some useful functions defined in the module:

```
let is_none = function (* is_none : 'a option -> bool *)
    None -> true
  | _ -> false;;

let is_some = function (* is_some : 'a option -> bool *)
    Some _ -> true
  | _ -> false;;

let get = function (* get : 'a option -> 'a *)
    Some v -> v
  | _ -> raise (Invalid_argument "get");;
```

Polymorphic variant types

Example with Option

- implementation of the function:
  `find : ('a -> bool)-> 'a list -> 'a option`
- specification: `find p ls` returns
  - `Some e` if `e` is the first element in `ls` such that `p e = true`
  - `None` if there are no elements `e` in `ls` such that `p e = true`

```
let find p =
    let rec aux = function
        hd::tl -> if p hd then Some hd else aux tl
      | _ -> None
    in aux;;
```

Polymorphic variant types

Example with Option

```
# let v=find ((<) 0) [-1;-2;3];; (* (<) 0 means fun x -> 0 < x  *)
val v : int option = Some 3
# Option.is_some v;;
- : bool = true
# Option.get v;;
- : int = 3
# let v=find ((<) 0) [-1;-2;-3];;
val v : int option = None
# Option.is_none v;;
- : bool = true
# Option.get v;;
Exception:  Invalid_argument "get".
```

Polymorphic and recursive variant type

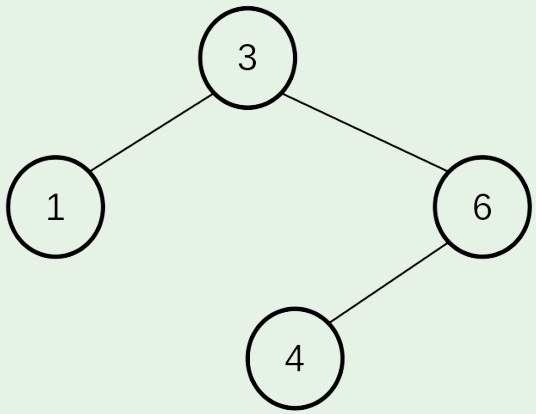
Polymorphic variant types can be recursive

Example: type `'a btree` of binary trees labeled with values of type `'a`

```
type 'a btree = Empty | Node of 'a * 'a btree * 'a btree;;
```

Example of value of type `int btree`

```
# let t=Node(3,Node(1,Empty,Empty),Node(6,Node(4,Empty,Empty),Empty));;
val t : int btree = ...
```



Polymorphic and recursive variant type

Example: function member on binary search trees

```
# let member el =
    let rec aux = function
        Node(label,left,right) ->
          el=label || if el<label then aux left else aux right
      | _ -> false
    in aux ;;
val member : 'a -> 'a btree -> bool = <fun>

#let t=Node(3,Node(1,Empty,Empty),Node(6,Node(4,Empty,Empty),Empty));;
val t : int btree = ...

member 4 t=true;;

member 5 t=false;;
```

Polymorphic and recursive variant type

Example: function insert on binary search trees

```
# let insert el =
   let rec aux = function
       Node(label,left,right) as t ->
         if el=label then t
         else if el<label then Node(label,aux left,right)
         else Node(label, left, aux right)
     | _ -> Node(el,Empty,Empty)
   in aux;;
   val insert : 'a -> 'a btree -> 'a btree = <fun>

#let t=Node(3,Node(1,Empty,Empty),Node(6,Node(4,Empty,Empty),Empty));;
val t : int btree = ...

insert 2 t = Node(3,Node(1,Empty,Node(2,Empty,Empty)),Node(6,Node(4,
    Empty,Empty),Empty))
```