



Introduction to O'Cam1 for Compiler people

Zhou Shuchang

2009-9-4

Outline

- ❖ History
- ❖ Features

History

- ❖ ML, 1970s
 - Designed for writing theorem prover,
 - Robin Milner University of Edinburgh
- ❖ Caml
- ❖ O'Caml 1996
 - Xavier Leroy INRIA, France
- ❖ F#
 - Visual Studio 2010 Microsoft

Features

- ❖ An interactive top-level loop
- ❖ Statically typed
- ❖ Module system
- ❖ (may be) Object oriented
- ❖ Imperative/Functional

An interactive top-level loop

❖ `let () =
 Printf.printf "Hello World!\n"`

An interactive top-level loop

- ❖ `# let x = 3+4;;`
- ❖ `val x : int = 7`
- ❖ `#let rec fib n = if n < 2 then 1 else fib(n-1) + fib(n-2);;`
- ❖ `val fib : int -> int = <fun>`
- ❖ `# fib 33;;`
- ❖ `- : int = 5702887`

An interactive top-level loop

```
❖ # let rec sum xs =  
    match xs with  
    | [] -> 0  
    | x :: xs -> x + sum xs  
- : sum : int list -> int = <fun>  
# sum [1;2;3;4;5];;  
- : int = 15
```

An interactive top-level loop

```
❖ let rec quicksort = function
  | [] -> []
  | pivot :: rest ->
    let left, right =
      List.partition ((<) pivot) rest in
    quicksort left @ [pivot] @ quicksort right
```

Statically typed

- ❖ Polymorphic type

- `List.length : 'a list -> int`

- ❖ Type inference

- No need to specify all types

- ❖ `# fun x -> x + 1;;`

- ❖ `- : int -> int = <fun>`

Statically typed

❖ Type checking

□ Detect errors at compile time

❖ `List.length "abc";;`

❖ Error: This expression has type string but an expression was expected of type 'a list

Module system

- ❖ The standard library is organized as modules

module List :

sig

val length : 'a list -> int

val hd : 'a list -> 'a

val tl : 'a list -> 'a list

val nth : 'a list -> int -> 'a

val rev : 'a list -> 'a list

...

Module system

- ❖ `# [(fun x -> x);(fun x -> x*x)];;`
- ❖ `- : (int -> int) list = [<fun>; <fun>]`
- ❖ `# List.length [(fun x -> x);(fun x -> x*x)];;`
- ❖ `- : int = 2`

- ❖ `open List`
- ❖ `let () = Printf.printf "%d" (length [])`

Object oriented

```
❖ # class point x y = object
  val x : int = x
  val y : int = y
  method getX = x
  method getY = yend;;
# let p = new point 2 3;;
val p : point = <obj>
# p#getX;;
- : int = 2
```

Imperative/Functional

❖ Functional

- Modification applied to a value yields a new one, but with sharing like that of SVN

```
# List.tl [1;2;3]
```

```
:- [2;3]
```

❖ Imperative

- Modification in place

```
# let i = ref 0 in incr i;!i;;
```

```
- : int = 1
```

Imperative/Functional

- ❖ Functional
 - Persistence arguably improves robustness
- ❖ Imperative
 - Allow efficient implementation for hot parts
- ❖ Up to you to strike the balance!

More features

- ❖ Byte code/native code
- ❖ A plugin for Eclipse
- ❖ Powerful debugger
- ❖ Ecosystem

Byte code/native code

- ❖ Byte code is interpreted
 - generated fast
 - Can be loaded into the toplevel
- ❖ Native code undergoes optimizations to reach near C level speed

Byte code/native code

- ❖ Both allow exception backtrace
export OCAMLRUNPARAM="b"

```
zsc@moon:~ 15:29$ ./a.out
```

```
Fatal error: exception Failure("hd")
```

```
Raised at file "pervasives.ml", line 22, characters  
22-33
```

A plugin for Eclipse

O'Caml - select/t31.ml - Eclipse (dzz)

File Edit Navigate Search Project Ocaml Run Window Help

Navigator

- cil
- cil137
- cil2
- euler
- lido
- linda
- linda2
 - _build
 - _darcs
 - External files
 - .Makefile.swp
 - deprecated.ml
 - deprecated.mli
 - linda_test.exe
 - linda_test.ml
 - linda_test.mli
 - linda.ml
 - linda.mli
 - Makefile
 - linda3
 - ocamlgraph
 - parser
 - parser2
 - parser3
- peak
 - _darcs
 - a.out.ipakeep
 - equake_kernel
 - .cproject
 - .pydevproject

```
?(recordFileName = "RUNTIME") ?(timeout = 3600000) bin =
let fn = Filename.basename bin in
timeoutExec recordFileName timeout
((if copy then copyToRemote bin else "") <+>
deleteRemote recordFileName <+>
execRemoteCmd (sprintf "%s" fn) <+>
(if isDeleteRemote then deleteRemote fn else "") <+>
copyFromRemote recordFileName)
let timeitBin ?(recordFileName = "RUNTIME") ?(timeout = 3600000) bin =
remoteTimeitBin ~recordFileName ~timeout bin
let nTimeitBin ?(recordFileName = "RUNTIME") ?(timeout = 3600000) n bin =
(if n > 0 then [remoteTimeitBin ~recordFileName ~timeout bin] else []) @
map (fun _ -> remoteTimeitBin ~copy:false ~recordFileName ~timeout bin) (range 0 (n - 1))
end
module LocalExec = struct
open ExtList
let timeitBin ?(recordFileName = "RUNTIME") ?(timeout = 3600000) bin =
timeoutExec recordFileName timeout (sprintf "timeout %d 100000 %s" timeout bin)
let nTimeitBin ?(recordFileName = "RUNTIME") ?(timeout = 3600000) n bin =
map (fun _ -> timeitBin ~recordFileName ~timeout bin) (range 0 n)
end
module LE = LocalExec
module RE = RemoteExec(struct type t = string let p = "root@10.3.0.182" end)
(*let () =
*)
(* printf "%s\n" @$ show_list Score.show @$ RE.nTimeitBin 30 "a.out"*)
(* printf "%s\n" @$ ExtUnix.exec "ssh root@10.3.0.182 \"timeout 3600 :1 ./a.out\""*)
let () = List.length "abc"
```

Outline

- Print
- Linda
- Algebra
- RunScore
- Score
- EXEC
- timeoutExec : string → int → string →
- RemoteExec
- LocalExec
- LE
- RE

Problems

```
;;
val splitToPair : char -> string -> string * string = <fun>
# splitToPair ' ' "ab,efe,ef,,";;
- : string * string = ("ab", "efe,ef,,")
# List.length "abc";;
List.length "abc";;
Error: This expression has type string but an expression was expected of type
'a list
#
```

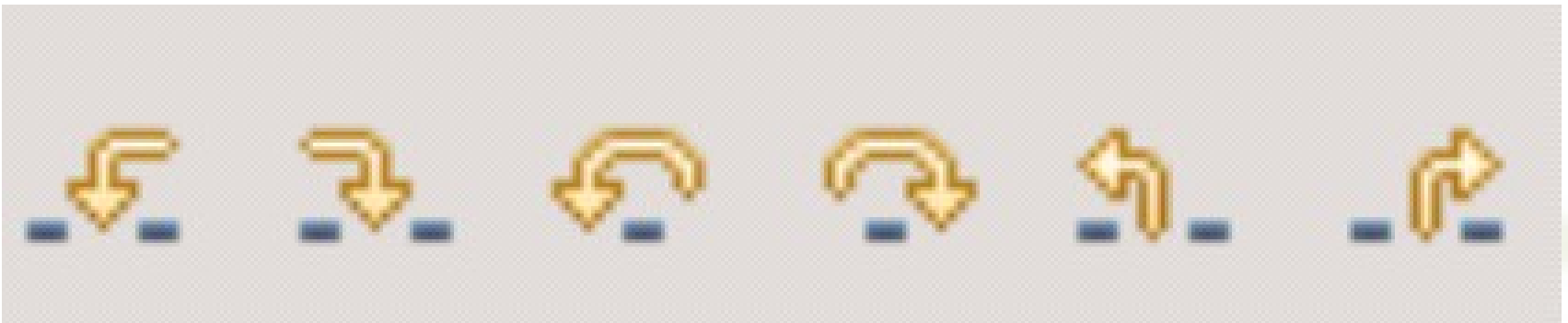
Writable Insert 109 : 26

A plugin for Eclipse

- ❖ compiler type check when saving file
- ❖ A toplevel to try out code snippets
- ❖ Automatic location of errors
- ❖ Organizing files as projects
- ❖ Prompting for members of modules
- ❖ Auto completion

Powerful debugger

❖ With backstep!



Ecosystem

- ❖ Popular in INRIA etc.
 - CIL, C Intermediate Language, with numerous tools based on it
 - Frama-C, actively developed static analyzer
- ❖ F# as its cousin
- ❖ Janestreet Capital, Microsoft, etc.

Ecosystem

❖ You are now part of It!

Advertisement

- ❖ module Linda
- ❖ 4000+ lines
- ❖ Extends standard library
 - Tree
 - Algebra
 - Monad



Thank you!