Slides to Accompany *Programming Languages and Methodologies*

R. J. Schalkoff

Chapter 11: `ocaml` Profiling

Notes

- In modern times, *profiling* is politically incorrect.
- However, we make an exception for functions in `ocaml`.

Why Profile?

- To check distribution of function use
- To identify computationally costly functions based upon aggregate use
- To investigate distribution of branches
- To empirically determine scalability

Finding Out What We Have, Part 1

```
$ ocamlc -config
version: 3.12.1
standard_library_default: /usr/local/lib/ocaml
standard_library: /usr/local/lib/ocaml
standard_runtime: /usr/local/bin/ocamlrun
ccomp_type: cc
bytecomp_c_compiler: gcc -fno-defer-pop -Wall -D_FILE_OFFSET_BITS=64 -D_REEN
bytecomp_c_libraries: -lm  -ldl -lcurses -lpthread
native_c_compiler: gcc -Wall -D_FILE_OFFSET_BITS=64 -D_REENTRANT
native_c_libraries: -lm  -ldl
native_pack_linker: ld -r  -o
ranlib: ranlib
cc_profile: -pg
architecture: amd64
model: default
system: linux
asm: as
ext_obj: .o
```

```
ext_asm: .s
ext_lib: .a
ext_dll: .so
os_type: Unix
default_executable_name: a.out
systhread_supported: true
$
```

Finding Out What We Have, Part 2

```
$ ocamlcp -v
The Objective Caml compiler, version 3.12.1
Standard library directory: /usr/local/lib/ocaml
$
```

What Can be Profiled?

You can see other options via:

```
$ ocamlcp -help
Usage: ocamlcp <options> <files>
options are:
  -p [afilmt]  Profile constructs specified by argument (default fm):
       a  Everything
       f  Function calls and method calls
       i  if ... then ... else
       l  while and for loops
       m  match ... with
       t  try ... with
...
```

An Example

```
(* odd-even5.ml *)
(** main (mutually recursive) functions (with if-then-else) *)

let rec even n =
if (n==0) then true
          else odd (n-1)
and (* here's the mutual recursion *)
odd m =
if (m==0) then false
          else even (m-1);;

(* some evaluations *)

Printf.printf "\neven(10000) is %b \n" (even 10000);;
```

The Mechanics

```
 ocamlcp -o oddEven5prof oddeven5.ml
./oddEven5prof
ocamlprof oddeven5.ml
```

Now Look at 'Original' Source File After Profiling

```
$  ocamlcp -o oddEven5prof oddeven5.ml
$ ./oddEven5prof

even(10000) is true
$ ocamlprof oddeven5.ml

(* odd-even5.ml *)
(** main (mutually recursive) functions (with if-then-else) *)

let rec even n =
(* 5001 *) if (n==0) then true
           else odd (n-1)
and (* here's the mutual recursion *)
odd m =
(* 5000 *) if (m==0) then false
           else even (m-1);;

(* some evaluations *)
```

```
Printf.printf "\neven(10000) is %b \n" (even 10000);;
```

Another Example (You Can Generate Your Own)

```
$ ocamlcp -p i oddeven5.ml
$ ocamlcp -p i -o oddEven5prof oddeven5.ml
$ ./oddEven5prof

even(10000) is true
$ ocamlprof oddeven5.ml
(* odd-even5.ml *)
(** main (mutually recursive) functions (with if-then-else) *)

let rec even n =
if (n==0) then (* 1 *) true
          else (* 5000 *) odd (n-1)
and (* here's the mutual recursion *)
odd m =
if (m==0) then (* 0 *) false
          else (* 5000 *) even (m-1);;
```

```
(* some evaluations *)

Printf.printf "\neven(10000) is %b \n" (even 10000);;
```