The core Caml system, 2010–2011

Xavier Leroy

INRIA Paris-Rocquencourt

OCaml users meeting, 2011-04-15

Outline

Previous release

Major release 3.12.0 (august 2010):

- Surprisingly many new language features!
  - First-class modules
  - Polymorphic recursion
  - Expression-local `open`
  - New operations over module signatures
  - . . . and more.
- Small improvements to the compilers and tools, e.g.
  - Finer control of warnings.
  - Options `-strict-sequence` and `-no-app-funct`.
  - The `ocamlobjinfo` tool.
- About 20 bug fixes.

Zoom: First-class modules

Encapsulate a module as a core language value (with an explicit type), then recover the module from this value.

$$\begin{aligned}
\textit{expr} &::= \ldots \mid \texttt{(module } \textit{module-expr} \texttt{ : } \textit{package-type}\texttt{)} \\
\textit{module-expr} &::= \ldots \mid \texttt{(val } \textit{expr} \texttt{ : } \textit{package-type}\texttt{)} \\
\textit{type} &::= \ldots \mid \texttt{(module } \textit{package-type}\texttt{)} \\
\textit{package-type} &::= \textit{modtype-path} \texttt{ with } t_1 = \tau_1 \texttt{ and } \ldots t_n = \tau_n
\end{aligned}$$

(An extension of Claudio Russo's proposal, part of Moscow ML.)

Zoom: First-class modules

Typical use: selecting at run-time among several implementations of a signature.

```
module type DEVICE = sig ... end
let devices : (string, (module DEVICE)) Hashtbl.t
            = Hashtbl.create 17

module SVG = struct ... end
let _ = Hashtbl.add devices "SVG" (module SVG : DEVICE)

module PDF = struct ... end
let _ = Hashtbl.add devices "PDF" (module PDF: DEVICE)

module Device =
  (val (try Hashtbl.find devices (parse_cmdline())
        with Not_found -> eprintf "Unknown device %s\n"; exit 2)
   : DEVICE)
```

Zoom: First-class modules

More advanced uses:

- Functors that take a list of structures as argument.
- Encodings of first-class values with existential types.
- (Painful) encodings of some Generalized Algebraic Data Types.

See papers at ML Workshop 2010:

First-class modules and composable signatures in Objective Caml 3.12, A. Frisch and J. Garrigue.

First-class modules: hidden power and tantalizing promises, J. Yallop and O. Kiselyov.

Next releases

Minor release 3.12.1 (Real Soon Now):

- About 20 bug fixes in the core system
  + 20 in Camlp4 and OCamlbuild (thanks to X. Clerc).
- Bumped up default minor heap size.
- Small floating-point performance improvements in x86-64 bits.

Next releases

Major release 3.13.0:

- No release date fixed.
- Several language extensions under consideration (see next slides)

Ongoing work 1: more lightweight first-class modules

(Jacques Garrigue; already merged in SVN trunk)

Omit ": package-type" explicit type annotations when they are determined by the context:

```
let f (x: module PT) = ... val x ...
                (* no "(val x: PT)" *)
```

Patterns to match and bind first-class module values:

```
let sort (type s) (module M : Set.S with type elt = s) l = ...
```

instead of

```
let sort (type s) m l =
  let module M = (val m: Set.S with type elt = s) in ...
```

Ongoing work 2: Generalized Algebraic Data Types

(Jacques Le Normand, with advice from Jacques Garrigue and Alain Frisch)

(SVN branch `branches/gadts`. More details at `https://sites.google.com/site/ocamlgadt/`)

An example of a GADT: type-safe interpreters.

```
type _ t =
  | IntLit : int -> int t
  | Pair : 'a t * 'b t -> ('a * 'b) t
  | App : ('a -> 'b) t * 'a t -> 'b t
  | Abs : ('a -> 'b) -> ('a -> 'b) t

let rec eval : type s . s t -> s = function
  | IntLit x -> x                    (* s = int here *)
  | Pair (x,y) -> (eval x,eval y)    (* s = 'a * 'b here *)
  | App (f,a) -> (eval f) (eval a)
  | Abs f -> f
```

Ongoing work 2: Generalized Algebraic Data Types

Interests:

- Expressing and enforcing invariants in data structures
  (a restricted form of dependent types).
- Reflecting types into values $\rightarrow$ generic programming.
- Matching Haskell's GADTs...

Challenges:

- Partial type inference and principality.
  (Same solution as for first-class polymorphism in records and objects.)
- Pattern-matching: exhaustiveness, unused cases. (Solved?)
- Syntax wars ...

Most welcome: experimentation with Le Normand's branch + feedback.

Ongoing work 3: compiler-produced annotated ASTs

(Xavier Clerc, Damien Doligez, Alain Frisch, Fabrice Le Fessant, . . .)

Idea: have the OCaml compilers save (in a file) a full abstract syntax tree annotated with

- source locations;
- scoping information;
- types (both declared and inferred);
- and possibly user-inserted source-level annotations.

(A generalization of today's `-annot` flag.)

Intended uses:

- For IDEs (integrated development environments). Unifying the needs of OCamlSpotter, OCamlWizard, OcaIDE, etc.
- For code generators driven by type declarations, and documentation generators.

Outline

Membership

Recently joined: MLstate, MyLife.com.

Currently joining: Esterel Technologies, OCamlPro.

Recently left: Intel.

13 members (+ INRIA) total:

before	2008	2009	2010	2011
Citrix	CEA	SimCorp	MLstate	Esterel Tech
Dassault Aviation	OCamlCore		MyLife.com	OCamlPro
Dassault Systèmes			– Intel	
Intel				
Jane Street				
Lexifi				
Microsoft				

Actions of the Consortium

What the Consortium does:

- Sell permissive licensing conditions on the Caml code base.
- Enable lightweight corporate sponsoring.
- A place to discuss needs with power users from industry.
- A sounding board for new features and future directions.
- Two members contributing directly to the Caml code base.
- Good public relation.
- Brings "pocket money" e.g. for sponsoring this meeting.

A grant application under examination to get matching funds from an *Institut de Recherche Technologique*.

Outline

News from the community

(not exhaustive)

New projects:

- OASIS package management (S. Le Gall)
- Mirage: OCaml as a guest Xen OS
  (A. Madhavapeddy and T. Gazagnaire)
- Plasma map-reduce and distributed file system (G. Stolpmann)
- Bitstring manipulation (Rich Jones)

News from the community

(not exhaustive)

New releases of interesting libraries and tools, such as:

- Batteries 1.0 (E. Friendly et al)
- OCamlnet 3.0 (G. Stolpmann)
- OUnit (S. Le Gall)
- OCamlJava (X. Clerc)
- Ocsigen (V. Balat et al)
- LWT 2.2 (J. Vouillon et al)
- Delimited Continuations (O. Kiselyov)

News from the community

(not exhaustive)

Cool factor:

- OCaml iPhone and iPad apps (Jonathan Kimmitt, Jeffrey Scoffield)

Summer(s) of code:

- Google's via Xen.org
- Jane Street's ?

Still missing:

- A good, published textbook in English.

Consulting & custom development services

Just created: the OCamlPro company (Fabrice Le Fessant), bringing to 3 the number of companies offering consulting and development services in Caml:

Company	*Specialties*	*Community contributions*
Gerd Stolpmann (Darmstad)	Distributed apps, clusters & grids	Many libraries
OCamlCore (S. Le Gall, Paris)	OCaml consulting, custom developments	OASIS package infrastructure, OCamlCore.org forge
OCamlPro (F. Le Fessant, Paris)	Custom modifications to the OCaml system, long-term support	Compiler improvements

Outline

The state of OCaml

What we (core Caml development team) can commit on:

- Fanatical support of OCaml as it is today.
- "Sustainable development" of language and tools.

What you (Caml community) are expected to do:

- Take initiative for libraries, additional tools, packaging, distribution, etc.
- Join community efforts.
- Test & provide quick feedback.