The state of OCaml, 2013

Xavier Leroy

INRIA Paris-Rocquencourt

OCaml Workshop, 2013-09-24

Outline

Recent releases

Major release 4.00.0: (June 2012)

- Generalized Algebraic Data Types
- Exposing rich typed ASTs and compiler internals (for IDEs and more)
- . . . and much more.

Minor release 4.00.1: (Oct 2012)

- 23 bugs fixed

Release 4.01.0: (Sept 2013)

What's new in OCaml 4.01.0

Type checking and inference:

- More clever typing of ambiguous record labels and datatype constructors.

Usability:

- A lot of new warnings.
- `-short-path` option to choose shorter, more readable names when printing inferred types.
- Suggested corrections for misspelt identifiers.
- Richer, more efficient API to record and display stack backtraces.

Ambiguous record labels

```
type t = { a: int }
type u = { a: int; b: int }
```

What is the type of `fun x -> x.a` ?

Last definition hides previous definitions: (OCaml $\leq$ 4.00)
label `a` is always associated with type `u`, never with `t`.

```
fun x -> x.a : u -> int
   { a = 1 } : ✘
```

Problem: programmers must make label names unique.

Polymorphic records: (using objects)

```
fun x -> x#a : < a:α, ... > -> α
```

Problem: high run-time cost of field accesses; no pattern-matching.

The new disambiguation strategy

```
type t = { a: int }
type u = { a: int; b: int }
```

- Use "last definition" approach if it type-checks.
- Otherwise, consider other definitions of the label of interest (based on type constraints and context). If one causes the term to type-check, choose it.

	In 4.01	Before
`fun x -> x.a`	`u -> int`	`u -> int`
`fun (x: t) -> x.a`	`t -> int`	✘
`{a = 1; b = 2}`	`u`	`u`
`{a = 1}`	`t`	✘

Also applies to constructors of sum types.

Development process

More external contributions, more careful PR triaging

→ 135 minor bugs fixed

→ 25 feature wishes granted.

Much improved & automated testing:

- Continuous integration for the core system (esp. all Windows ports)
- OCamlot testing of OPAM packages (under Linux & BSD)

This release brought to you by...

Damien Doligez,
release manager and general wizard.

The core Caml development team: Alain Frisch, Jacques Garrigue, Benedikt Meurer, Fabrice Le Fessant, Gabriel Scherer, Hongbo Zhang, Jonathan Protzenko, Wojciech Meyer, Xavier Clerc, Xavier Leroy.









With much appreciated contributions from: Anil Madhavapeddy, Benjamin Monate, Benoît Vaugon, Chet Murthy, Christoph Bauer, Christophe Papazian, Christophe Troestler, Dan Bensen, Daniel Bünzli, David Allsopp, François Berenger, Gabriel Kerneis, Gerd Stolpmann, Grégoire Henry, Jacques-Henri Jourdan, Jeffrey Scofield, Jérémie Dimino, Jérôme Vouillon, John Carr, Khoo Yit Phang, Leo P. White, Markus Mottl, Maxence Guesdon, Michel Mauny, Pierre Chambart, Pierre Weis, Tiphaine Turpin, Valentin Gatien-Baron, William Smith, ygrek.

Outline

The OPAM package manager

OPAM is taking off: from alpha one year ago to 512 packages today.

A great help for:

- beginners (one-stop shopping installation & upgrade)
- power users, library developers (e.g. support for multiple versions)
- the upcoming OCaml Platform
- . . . not to forget the core OCaml dev team (testing, and more).

Many thanks to OCamlPro, esp. Thomas Gazagnaire.

Dissemination

Not one but two new very good books in English:

- *Real-World OCaml*, Jason Hickey, Anil Madhavapeddy, and Yaron Minsky, O'Reilly.
- *OCaml from the very beginning*, John Whitington, Coherent Press.

New resources for beginners (OCamlPro):

- `tryocaml.ocamlpro.com` (the toplevel in your browser)
- OCaml-Top (at last a decent GUI for the toplevel)

The `ocaml.org` infrastructure (OCamlLabs):

- the new OCaml Web site
- consolidation of mailing lists, forge, etc.

Some new projects (not exhaustive)

Recently released:

- Merlin (Emacs and Vim-based IDE)
- SPOC (GPGPU programming)
- OCaml-Java (OCaml on the JVM)
- UCore (Unicode support library)
- Wodi (the GODI distribution for Windows)

Plus much activity on older projects (too many to list).

Outline

Reorganizing the core OCaml distribution

OPAM and the upcoming OCaml Platform make it possible to split off certain parts of the core OCaml distribution as separate projects, e.g.

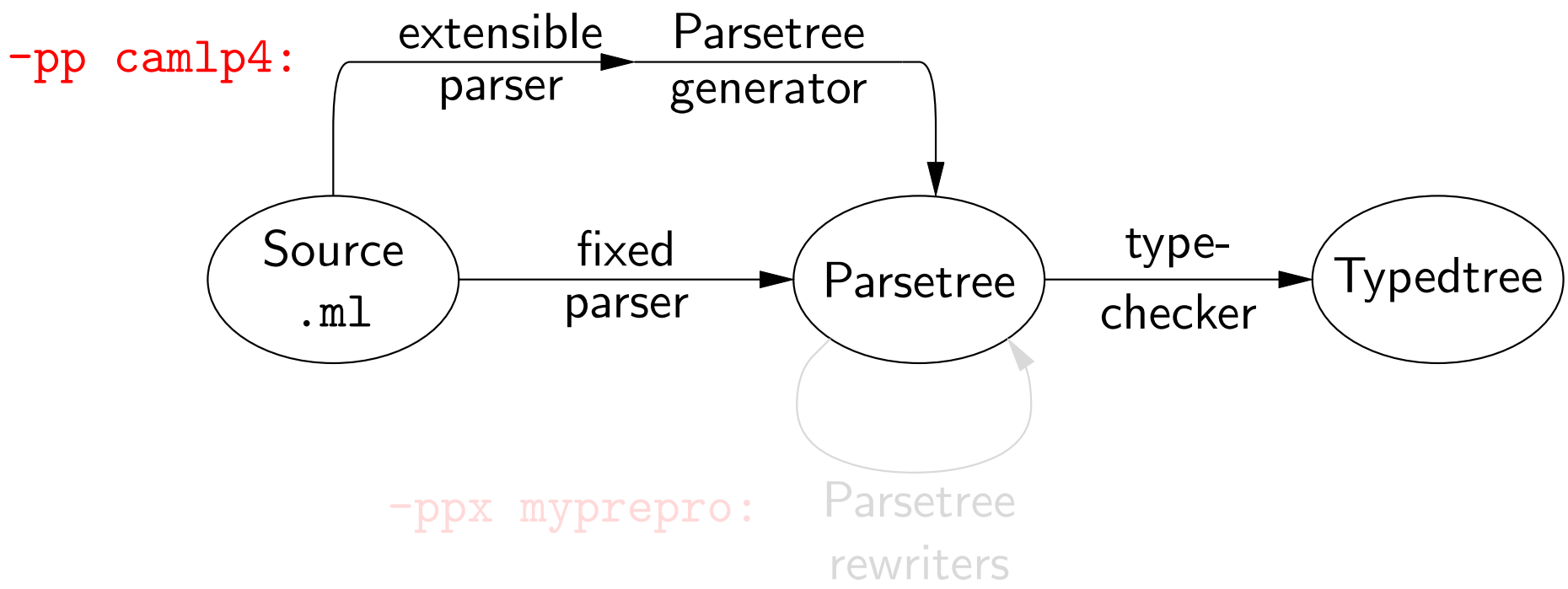
- the LablTK GUI library (done)
- Camlp4 (soon)
- OCamlbuild; the Num, Str, Graphics libraries (under discussion).

Expected benefits:

- Decoupling the development & release cycles of these projects.
- Attracting more contributors.
- Lightening up the burden on the core OCaml developers.

Vision: in the future, very few users should download and install the core OCaml distro themselves; instead, it will come as a component of the OCaml Platform.

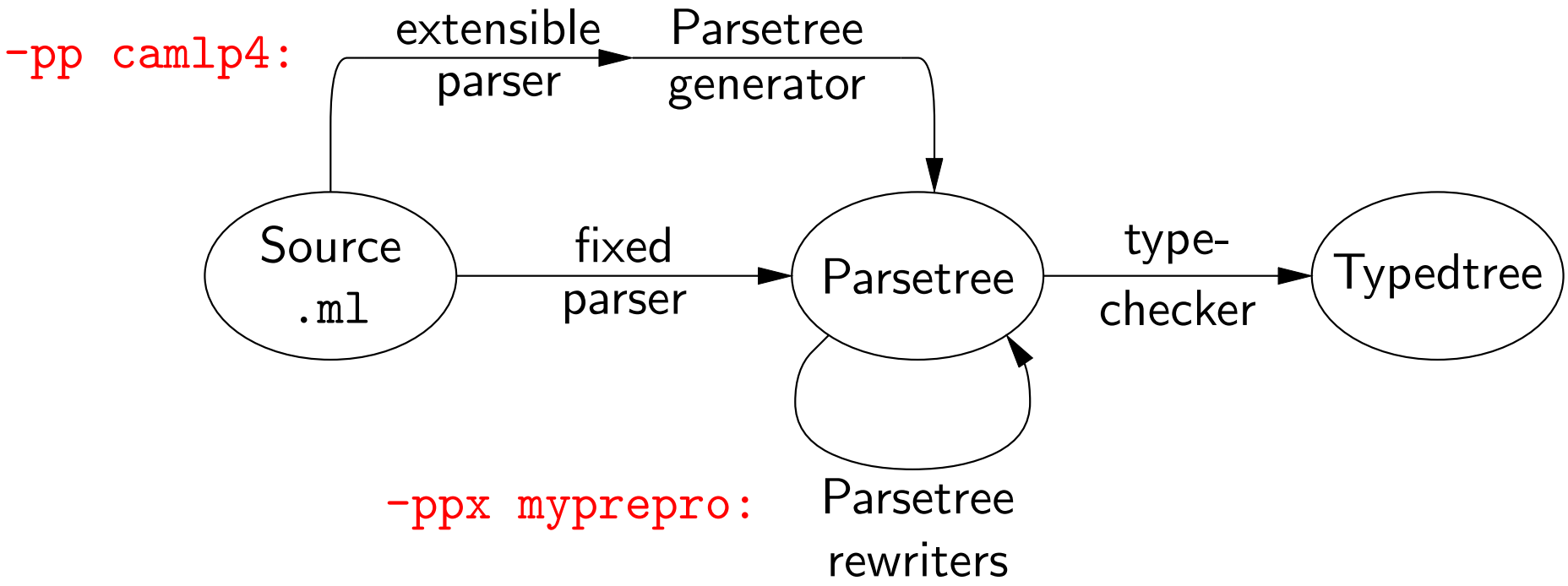
Extension points and -ppx preprocessing



The Camlp4 way: a special parser; each preprocessor extends the syntax.

The -ppx way: parsetree-to-parsetree rewriting; use the standard parser from ocamlc/ocamlopt, which supports "extension points" (a.k.a. attributes, annotations).

Extension points and -ppx preprocessing



The Camlp4 way: a special parser; each preprocessor extends the syntax.

The -ppx way: parsetree-to-parsetree rewriting; use the standard parser from ocamlc/ocamlopt, which supports "extension points" (a.k.a. attributes, annotations).

Extension points and -ppx preprocessing

Extension points = free-form annotations that are attached to the parsetree, ignored by the compiler, exploited by preprocessors.

Example: generating functions from type definitions. The Camlp4 way:

```
type t = {
  x : int with default(42);
  y : int with default(3), sexp_drop_if(y_test);
} with sexp
```

With extension points:

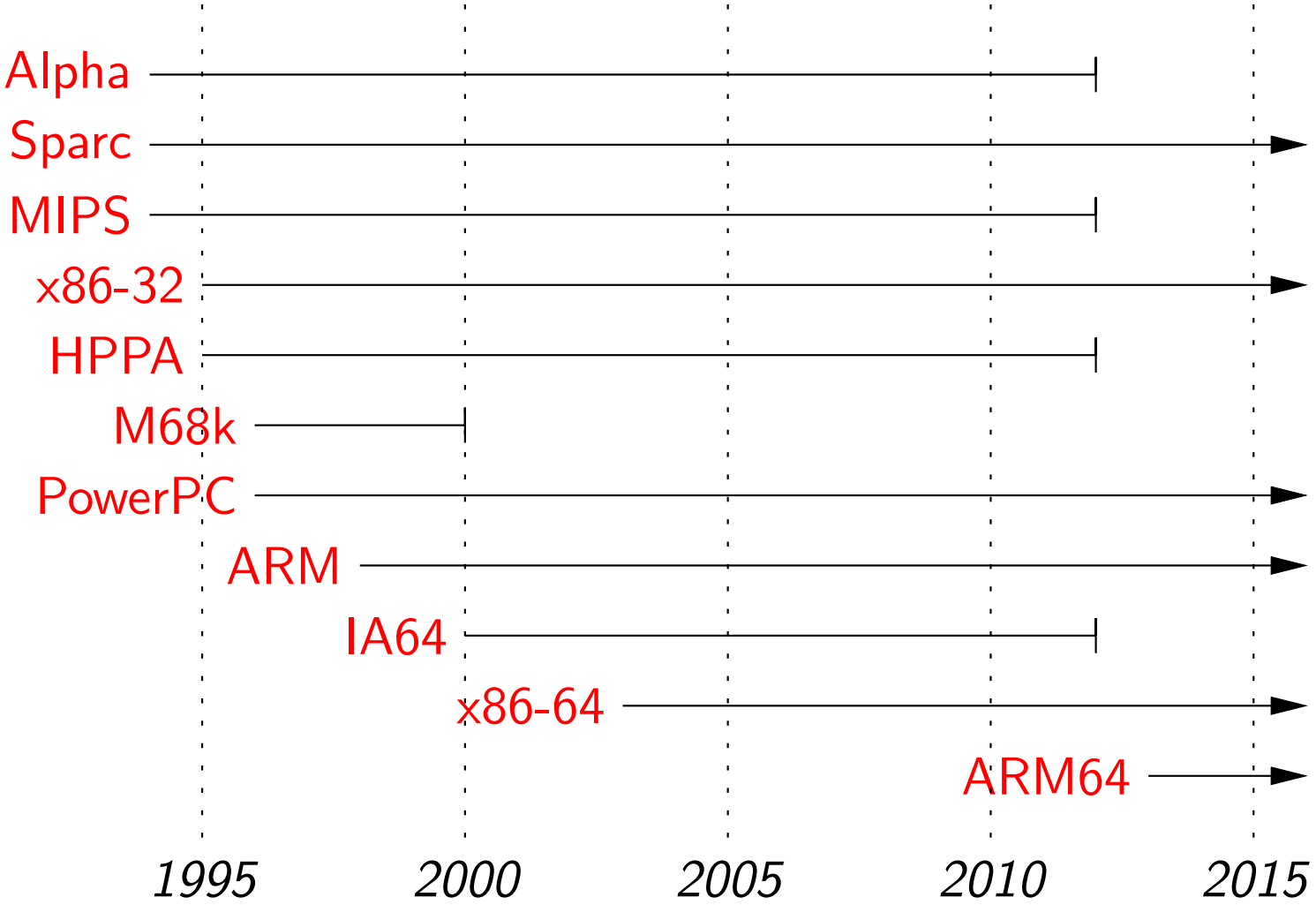
```
type t = {
  x : int [@default 42];
  y : int [@default 3] [@sexp_drop_if y_test];
} [@@sexp]
```

Status: first proposal in SVN trunk; ongoing discussions on syntax & contents of extension points.

A code generator for the ARM 64-bits platform

(a.k.a. AArch64)

The first new target architecture since x86-64, ten years ago.



Improving performance

Several ongoing experiments:

- Middle-end: inlining (P. Chambard), unboxing (A. Frisch)
- Back-end: CSE, aggressive constant propagation (X. Leroy)
- Run-time system: more lightweight write barriers, page table, major heap allocation, ...
- Profiling tools: better `perf` support (OCamlPro), memory usage profiling (OCamlPro, M. Shinwell).

A prerequisite: building a benchmark suite, ideally as part of Platform packages.

In closing...

A lively language; a very lively community.

Two milestones reached this year (OPAM, *Real World OCaml*).

Next milestone: the OCaml Platform. Support it!

Thanks for all the contributions. Keep them flowing!