Nurturing OCaml in 2019-2020

Florian Angeletti
Inria
florian.angeletti@inria.fr

21 September 2020

Seven releases since July 2019

August 5 2019 OCaml 4.08.1

September 18 2019 OCaml 4.09.0

February 21 2020 OCaml 4.10.0

March 18 2020 OCaml 4.10.0

Aug 19 2020 OCaml 4.11.0

Aug 20 2020 OCaml 4.10.1

Aug 31 2020 OCaml 4.11.1

Learning from the past

4.08
Painful ppx migration, 3 months late

4.09

- broken dune: daring definitions of dependencies
- broken ocamlfind

4.10

- C API change in preparation to multicore
- (Too) many packages were using the runtime internal
- Internal runtime API fixed during the beta

How to catch those problems early?

Opamcheck and opam-health check

opamcheck

- Checking breakage early on the opam ecosystem
- Before merging PRs, new features

opam-health-check

- Developed at ocamllabs
- check the health of the opam ecosystem

Improved opamcheck

Recent improvements

- Docker version
- PR and branch mode for testing prototypes or PR under review
- online summary
- http://gallium.inria.fr/blog/an-ocaml-release-story-1/

Not the future but usable right now.

Stabilised cycle

3 months feature development phase

3 months Bug fixing

alpha releases stable features and IRs, no guarantee for users

beta releases stable internal API, ready for library writers

rc release opam ecosystem ready

Towards quiet releases

4.10 and 4.11 releases continuous Work with Kate Deplaix to check and improve the opam ecosystem state

4.11 every core package in the opam ecosystem works, most packages work

4.11.0 too permissive, broken typechecker, fixed in two weeks $\Leftarrow$ 4.11.1

Future work: merlin synchronisation

- Merlin is usually released late in the cycle
- Costly maintenance
- Reduce the cost by upstreaming part of the typechecking API used by merlin

A fresh documentation

- Anchors
- Highlighted code
- Availability on ocaml.org (work done by Xavier Leroy)

Future documentation improvements

- work on progress: ocaml.org specific version by sanette
- an odoc version
  - with a new latex back-end for odoc
  - and a new manpage back-end (written by Gabriel Radanne)

Explaining the typechecker

- for users: error messages
- for developers: comments and regular flow control

A spell of regularity

```
type ('a,'b) a = [ `A of ('b,'a) b ]
and ('a,'b) b = [ `B of ('a,'b) a ]

Error: In the definition of b,
  type ('b, 'a) a should be ('a, 'b) a
```

Explanation
The typechecker was checking the *regularity* of type definition, the typechecker find a non-regular use of the type constructor a

(After a short discussion with Jacques Garrigue)

```
type ('a,'b) a = [ `A of ('b,'a) b ]
and ('a,'b) b = [ `B of ('a,'b) a ]

Error: This recursive type is not regular.
       The type constructor a is defined as
         type ('a, 'b) a
       but it is used as
         ('b, 'a) a
       after the following expansion(s):
         ('b, 'a) b = [ `B of ('b, 'a) a ]
       All uses need to match the definition for
       the recursive type to be regular.
```

Where are my polyvariants?

```
module Def: sig
  type t = private [< `A|`B]
  type u = private [> `A ]
end = struct
   type t = [ `A ]
   type u = [`A | `B]
end
```

```
let x: Def.u = `B

Error: This expression has type [> `B ]
       but an expression was expected of type Def.u
       The second variant type does not allow tag(s) `B
```

```
let is_b (x: Def.u) = function
  | `B -> true
  | _ -> false
```

```
let y: Def.t = `A

Error: This expression has type [> `A ]
       but an expression was expected of type Def.t
       Types for tag `A are incompatible
```

Explanation

- Private types enforces a constraint on the hidden row variable.
- The error printer does not know that
- The error printer tries to understand the error a posteriori.
- It failed

Solution
Explain to the error printer what went wrong

Tiptoeing around fixed row variables

```
let x: Def.u = `B

Error: This expression has type [> `B ]
       but an expression was expected of type Def.u
       The second variant type is private,
       it may not allow the tag(s) `B

let y: Def.t = `A

Error: This expression has type [> `A ]
       but an expression was expected of type Def.t
       The second variant type is private,
       it may not allow the tag(s) `A
```

For the sake of counter-examples

Before 4.11:

```
type nothing = |
type t = | A of int  | B of nothing | C
let f (A n) = n
let crash = f C
```

```
let g = function
  | A n ->  n
  | _ -> .

Error: This match case could not be refuted.
       Here is an example of a value that would reach it: C
```

Why?

- Two counter-example type-checking strategies
- Empty types mishandled in one strategy

Counter-examples and strategies

- Counter-example typechecking share the same code path as pattern type-checking
- Two strategies for counter-examples checking
- Complex control flow

In OCaml 4.11 (work with Jacques Garrigue, Thomas Refis, Gabriel Scherer)

- Simpler control flow: less exceptions
- Documented counter-example checking strategies
- Documented counter-example specific code paths

Patterns compilation

- efficient compilation of pattern is important
- pattern compilation operates on a matrix of patterns

```
 A(_)      P21  P31  _
  _        P22  P32  _
B | C(0)   P23  P33  _
   _        _    _   _
```

- Work columns by columns
- complex pipeline with tacit invariants
- How to maintain or extend it without introducing bugs

Typing the invariants

- Work by Thomas Refis and Gabriel expressing to lift those invariants to the type system without altering the semantics
- Review started in 4.09, partially integrated in 4.10, done in 4.11
- `https://github.com/ocaml/ocaml/pull/9321` and 9321, 9322, 9359, 9361, 9417, 9447, 9464, 9493, 9520, 9563, 9599, 9608, 9647.
- (semantic change: a small optimisation was let in)

A future of fearless refactoring?

- Few more types to add before hitting diminishing return
- A better place to start exploring new features

Conclusion

- Smoother release cycles
- A slightly improved documentation, with more changes in sight
- More documentation for the typechecker, externally and internally
- A more self-documented pattern compiler
- And more bug fixes ...

Thank you for your attention