OCaml Evolution

September 21st, 2020 Gabriel Scherer

Principal Investigators:

- Gabriel Scherer
- François Pottier

Problem

Being a good programming language

OCaml wants to be both:

- a real-world language enables productive development of high-quality programs
- an innovative language improve programming through programming-language research

Both are important to our past, present and future users.

But

Since the mid-2000s, innovation in OCaml (the language) has been slowing down; today we are trailing behind GHC Haskell.

Lack of ideas/interest?

Various ambitious proposals, including:

- static contract verification
- powerful module systems (MixML, 1ML...)
- modular implicits (type-classes with abstraction)
- separation logic in types (Mezzo)
- effect typing
- algebraic effects
- linear types
- static capabilities and resource ownership

So why?

Various factors, including:

1. Lack of manpower to work on advanced language features. (few key people trying to handle everything at once)
2. Lack of coordination among researchers. (separate prototypes without collaboration strategies)

3. Technical debt in the type-checker codebase.
4. Real scientific challenges hidden under the carpet. (Coherent specification? Typed core language? Principality? Soundness?)

Grant

"OCaml Evolution" grant from Nomadic Labs, aimed at providing answers to some of these problems. Provide funding to:

- train new engineers and researchers on these problems
- coordinate projects: fund visits, etc.
- work on the technical difficulties, attack scientific challenges

Details

Four focus area proposed... all fairly ambitious.

- Rebuilding the type-checker (ongoing)
- Multicore OCaml (ongoing)
- Effect typing
- Modular implicits (ongoing)

Hires: Gabriel Radanne Visits:

- Jacques Garrigue from Nagoya, Japan (done)
- KC Sivaramakrishnan from Madras, India (todo)

This talk: rebuilding the type-checker.

Rebuilding the type-checker

Problem:

- the type-checker codebase is low-level, very complex, under-documented
- only one person (Jacques Garrigue) knows it well

Not good for extending the language, or even fixing type-checker bugs when they show up.

Our plan of attack

Follow two paths **together**:

- "high road": clean-room redesign of what we need try first on a core language, scale to OCaml later (constraint-based type inference, typed core language, type soundness proofs, etc.)
- "low road": incremental changes share knowledge, document, create abstraction boundaries, understand what we need to formalize

Results so far, from a distance

- high road:
  - expression level: plan to extend Inferno (constraint-based type inference by François Pottier) preliminary work by Gabriel Scherer and Olivier Martinot
  - module level: understand the theory from the code fix blind spots for implicits Gabriel Radanne's talk.
- low road: understanding and clarifying the type-checker
  - type-checking patterns (Florian Angeletti, Jacques Garrigue, Thomas Refis, Gabriel Scherer)
  - typing of as-patterns (Jacques Garrigue, Thomas Refis, Gabriel Scherer)
  - principality in pattern-matching (Jacques Garrigue, Thomas Refis, Leo White)
  - typing error messages (Florian Angeletti, Gabriel Radanne)