Semantics and Implementation of Actors in Multicore OCaml

Gabriel RADANNE, Ludovic Henrio, Inria CASH/LIP

1 Context

To program parallel systems efficiently and easily, a wide range of programming models have been proposed, each with different choices concerning synchronization and communication between parallel entities. Among them, futures [2] and the actor model [1] is based on loosely coupled parallel entities that communicate by means of asynchronous messages and mailboxes. Some actor languages provide a strong integration with object-oriented concepts ; these are often called active object languages. Actors and active object languages feature interesting properties and programming abstractions, including the interaction between entities by request/reply mechanisms, and some determinism guarantees making programming easy and efficient in a distributed environment.

On the other hand, OCaml 5.0 [3] provides a brand new feature to OCaml : multicore programming. For this purpose, OCaml 5.0 will contain a new runtime and garbage collector that support parallelism [5], and a new language feature to easily support concurrency : algebraic effects [6]. Traditionally, complex control flow structures such as `raise` for exceptions, `yield` for iterators, or `async`/`await` for concurrency are *built-in* constructs of the language that are specially handled by the compiler. Algebraic effects [4] define a general mechanism which allow programmers to define all these constructs directly as libraries. These libraries can then define how exactly these constructs are implemented, for instance by providing a dedicated scheduler.

Algebraic effects gives us a great opportunity to experiment with varied concurrency constructs. Notably, we have recently developed an initial prototype implementing parallel actors with a syntax extension for OCaml. However, so far, this library has no support for distributed systems, no well-defined semantics, and merely serves as a proof of concept.

2 Internship objective

The objective of the internship is to develop a library of distributed actors in multicore OCaml. This work includes both theoretical and implementation aspects. The internship should cover both aspects but can be oriented more towards theoretical contribution or implementation depending on the student.

2.1 A library for distributed actors

On the practical side, the objective of the internship is the following.

— Based on our initial prototype, develop distributed actors based on OCaml Multicore. A first step is the development of a runtime library for actors with both multicore and distributed capabilities. A second step is the design of a syntax extension providing a nice syntax for actors, and of a compiler that uses the runtime library developed in the first step.

— Develop practical usecases showing the effectiveness of the proposed library. Depending on the interest of the student, this can range from computationally intensive application that leverage parallelism, to distributed systems with complex synchronisations.

LIP – UMR CNRS / ENS Lyon / UCB Lyon 1 / Inria – 69007 Lyon
E-mail : gabriel.radanne@inria.fr and ludovic.henrio@ens-lyon.fr

2.2 Proving the correctness of the implementation of actors

From the theoretical standpoint, our objective is the following :

— Define the semantics of our actor language. Several semantics for actors exist, but they should be adapted our context.

— Define the semantics of the target subset of OCaml with algebraic effects.

— Define the translation from actors to effects and prove its correctness.

This theoretical work should either be designed in a distributed setting or extended with distributed actors in a second time.

3 Internship

The internship will take place in the CASH Team, LIP lab, in Lyon France.

Candidate profile The candidate should ideally be familiar with formal approaches in programming language design, notably type systems, semantics, and logic.

From the practical point of view, a basic experience in software programming and usage of collaborative tools such that `git`. Knowledge of the OCaml programming language is mandatory.

This internship strongly relies on the fact that practical implementation should have strong theoretical foundations and that further refinements of the theory should get inspiration from the practical side. We expect the candidate to agree with this philosophy.

Références

[1] Frank De Boer, Vlad Serbanescu, Reiner Hähnle, Ludovic Henrio, Justine Rochas, Crystal Chang Din, Einar Broch Johnsen, Marjan Sirjani, Ehsan Khamespanah, Kiko Fernandez-Reyes, and Albert Mingkun Yang. A survey of active object languages. *ACM Comput. Surv.*, 50(5), oct 2017. ISSN 0360-0300. doi : $10.1145/3122848$. URL https://doi.org/10.1145/3122848.

[2] Kiko Fernandez-Reyes, Dave Clarke, Elias Castegren, and Huu-Phuc Vo. Forward to a Promising Future. In Giovanna Di Marzo Serugendo and Michele Loreti, editors, *20th International Conference on Coordination Languages and Models (COORDINATION)*, volume LNCS-10852 of *Coordination Models and Languages*, pages 162–180, Madrid, Spain, June 2018. Springer International Publishing. doi : $10.1007/978\text{-}3\text{-}319\text{-}92408\text{-}3\_7$. URL https://hal.inria.fr/hal-01821490.

[3] KC Sivaramakrishnan. Ocaml 5.0. https://speakerdeck.com/kayceesrk/ocaml-5-dot-0, 2022.

[4] Gordon D. Plotkin and Matija Pretnar. Handlers of algebraic effects. In Giuseppe Castagna, editor, *Programming Languages and Systems, 18th European Symposium on Programming, ESOP 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings*, volume 5502 of *Lecture Notes in Computer Science*, pages 80–94. Springer, 2009. doi : $10.1007/978\text{-}3\text{-}642\text{-}00590\text{-}9\backslash\_7$. URL https://doi.org/10.1007/978-3-642-00590-9_7.

[5] K. C. Sivaramakrishnan, Stephen Dolan, Leo White, Sadiq Jaffer, Tom Kelly, Anmol Sahoo, Sudha Parimala, Atul Dhiman, and Anil Madhavapeddy. Retrofitting parallelism onto ocaml. *Proc. ACM Program. Lang.*, 4(ICFP) : 113 :1–113 :30, 2020. doi : $10.1145/3408995$. URL https://doi.org/10.1145/3408995.

[6] K. C. Sivaramakrishnan, Stephen Dolan, Leo White, Tom Kelly, Sadiq Jaffer, and Anil Madhavapeddy. Retrofitting effect handlers onto ocaml. In Stephen N. Freund and Eran Yahav, editors, *PLDI '21 : 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, Virtual Event, Canada, June 20-25, 2021*, pages 206–221. ACM, 2021. doi : $10.1145/3453483.3454039$. URL https://doi.org/10.1145/3453483.3454039.