Offer #2025-09005

Formalization and improvement of OCaml recursive modules

Contract type : Fixed-term contract

Level of qualifications required : Graduate degree or equivalent

Fonction : Temporary scientific engineer

Context

The work will be conducted in the Inria Cambium team under the main supervision of Didier Rémy

Assignment

Context

One objective of the Cambium team is to establish a precise formalization of the OCaml language, encompassing both its dynamics semantics and its static semantics, that is, its type system. We recently formalized the static semantics of a module language *à la OCaml* [2] that covers most of the OCaml module features, but not all: it does not include recursive modules nor abstract signatures.

The actual formalization uses an intermediate language, called $M^?$, that mediates between the language OCaml and the higher-order polymorphic ?-calculus $F^?$ extended with records, in which $M^?$ programs are elaborated so as to ensure the soundness of $M^?$ [3]. The formalization of recursive modules will likely follow the same schema.

Overall objective

The objective is twofold. First, it aims to extend the actual formalization to include recursive modules. Second, it is to study possible improvements to the actual OCaml design of recursive modules.

The formalization will mainly be on paper, with potentially mechanized proofs for selected components.

Main challenge

A first mandatory step is to extend the actual variant of $\mathsf{F}^?$ with equi-recursive types to encode recursive signatures (at least in a limited form that covers all of OCaml features related to recursive modules) and ensure type soundness of this extension.

This should enable, as a first step, an explicit form of recursive modules with fully explicit signature annotations. However, the necessity of fully explicit signature annotations impedes the utilization of recursive modules. In fact, the OCaml implementation already performs some form of signature inference for recursive modules. Therefore, it will be necessary to examine the current state of the art regarding partial inference of recursive signatures, in particular [5,4], and adapt it to the language OCaml. At this juncture, the whole OCaml language will be covered, with the exception of abstract signatures, a feature that may be deliberately left aside.

Further challenges

Nevertheless, there is room for several enhancements.

A known challenging problem for recursive modules is the *double vision*: typically, two recursively defined modules A and B may define types tA and tB that are exported as abstract types tA? and tB? in the external signatures SA and SB of A and B so as to preserve their internal invariants. Consequently, A may call a function of B with the external abstract view tA?. However, if B calls back A with the same object, then it should be seen from A with its original internal type tA rather than its abstract view tA?.

Another highly desired feature regarding recursive modules is the ability to define them in separate files. Perhaps a solution to this problem could be found half-way between recursive modules and mixin modules [6].

Recursive modules also raise a compilation issue related to their initialization. In order to achieve sufficient expressiveness, it is necessary to permit recursive modules whose initialization may not always be statically proved to be well-defined. In such cases, OCaml introduces a dynamic check that may raise a runtime exception. The situation could be enhanced by incorporating a more rigorous static analysis, for instance drawing parallels with the initialization of objects [1].

References

1. Clément Blaudeau and Fengyun Liu. A conceptual framework for safe object initialization: a principled and mechanized soundness proof of the celsius model. *Proc. ACM Program. Lang.*, 6 (OOPSLA2), October 2022. doi: 10.1145/3563314. URL `https://doi.org/10.1145/3563314`.
2. Clément Blaudeau, Didier Rémy, and Radanne. Avoiding signature avoidance in ml modules with zippers. *Proc. ACM Program. Lang.*, 9 (POPL), January 2025. doi: 10.1145/3704902. URL `https://cambium.inria.fr/~remy/ocamod/`.
3. Clément Blaudeau, Didier Rémy, and Gabriel Radanne. Fulfilling ocaml modules with transparency. *Proc. ACM Program. Lang.*, 8 (OOPSLA1), apr 2024. doi: 10.1145/3649818. URL `https://doi.org/10.1145/3649818`.

4. Derek Dreyer. A type system for recursive modules. In *Proceedings of the 12th ACM SIGPLAN International Conference on Functional Programming*, ICFP '07, pages 289–302, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595938152. doi: 10.1145/1291151.1291196. URL `https://doi.org/10.1145/1291151.1291196`.
5. Keiko Nakata and Jacques Garrigue. Recursive modules for programming. In *Proceedings of the Eleventh ACM SIGPLAN International Conference on Functional Programming*, ICFP '06, pages 74–86, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933093. doi: 10.1145/1159803.1159813. URL `https://doi.org/10.1145/1159803.1159813`.
6. Andreas Rossberg and Derek Dreyer. Mixin' up the ml module system. *ACM Trans. Program. Lang. Syst.*, 35 (1), April 2013. ISSN 0164-0925. doi: 10.1145/2450136.2450137. URL `https://doi.org/10.1145/2450136.2450137`.

Main activities

Working on bibliography, writing papers, programming prototypes, writing machine-checked proofs,
giving talks.

Benefits package

- Subsidized meals
- Partial reimbursement of public transport costs
- Leave: 7 weeks of annual leave + 10 extra days off due to RTT (statutory reduction in working hours) + possibility of exceptional leave (sick children, moving home, etc.)
- Possibility of teleworking (after 6 months of employment) and flexible organization of working hours
- Professional equipment available (videoconferencing, loan of computer equipment, etc.)
- Social, cultural and sports events and activities
- Access to vocational training
- Social security coverage

General Information

- **Theme/Domain :** Proofs and Verification
  Software engineering (BAP E)
- **Town/city :** Paris
- **Inria Center :** Centre Inria de Paris
- **Starting date :** 2025-10-01
- **Duration of contract :** 6 months
- **Deadline to apply :** 2025-07-03

Contacts

- **Inria Team :** CAMBIUM
- **Recruiter :**
  Remy Didier / Didier.Remy@inria.fr

About Inria

Inria is the French national research institute dedicated to digital science and technology. It employs 2,600 people. Its 200 agile project teams, generally run jointly with academic partners, include more than 3,500 scientists and engineers working to meet the challenges of digital technology, often at the interface with other disciplines. The Institute also employs numerous talents in over forty different professions. 900 research support staff contribute to the preparation and development of scientific and entrepreneurial projects that have a worldwide impact.

Warning : you must enter your e-mail address in order to save your application to Inria. Applications must be submitted online on the Inria website. Processing of applications sent from other channels is not guaranteed.

Instruction to apply

Defence Security :
This position is likely to be situated in a restricted area (ZRR), as defined in Decree No. 2011-1425 relating to the protection of national scientific and technical potential (PPST).Authorisation to enter an area is granted by the director of the unit, following a favourable Ministerial decision, as defined in the decree of 3 July 2012 relating to the PPST. An unfavourable Ministerial decision in respect of a position situated in a ZRR would result in the cancellation of the appointment.

Recruitment Policy :
As part of its diversity policy, all Inria positions are accessible to people with disabilities.