Offre n°2025-09005

Formalization and improvement of OCaml recursive modules

Le descriptif de l'offre ci-dessous est en Anglais

Type de contrat : CDD

Niveau de diplôme exigé : Bac + 5 ou équivalent

Fonction : Ingénieur scientifique contractuel

Contexte et atouts du poste

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

Mission confiée

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 $\mathsf{M}^?$, that mediates between the language OCaml and the higher-order polymorphic ?-calculus $\mathsf{F}^?$ extended with records, in which $\mathsf{M}^?$ programs are elaborated so as to ensure the soundness of $\mathsf{M}^?$ [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 $F^?$ 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.

Principales activités

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

Avantages

- 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

Informations générales

- **Thème/Domaine :** Preuves et vérification
  Ingénierie logicielle (BAP E)
- **Ville :** Paris
- **Centre Inria :** Centre Inria de Paris
- **Date de prise de fonction souhaitée :** 2025-10-01
- **Durée de contrat :** 6 mois
- **Date limite pour postuler :** 2025-07-03

Contacts

- **Équipe Inria :** CAMBIUM
- **Recruteur :**
  Remy Didier / Didier.Remy@inria.fr

A propos d'Inria

Inria est l'institut national de recherche dédié aux sciences et technologies du numérique. Il emploie 2600 personnes. Ses 215 équipes-projets agiles, en général communes avec des partenaires académiques, impliquent plus de 3900 scientifiques pour relever les défis du numérique, souvent à l'interface d'autres disciplines. L'institut fait appel à de nombreux talents dans plus d'une quarantaine de métiers différents. 900 personnels d'appui à la recherche et à l'innovation contribuent à faire émerger et grandir des projets scientifiques ou entrepreneuriaux qui impactent le monde. Inria travaille avec de nombreuses entreprises et a accompagné la création de plus de 200 start-up. L'institut s'e?orce ainsi de répondre aux enjeux de la transformation numérique de la science, de la société et de l'économie.

Attention: Les candidatures doivent être déposées en ligne sur le site Inria. Le traitement des candidatures adressées par d'autres canaux n'est pas garanti.

Consignes pour postuler

Sécurité défense :
Ce poste est susceptible d'être affecté dans une zone à régime restrictif (ZRR), telle

que définie dans le décret n°2011-1425 relatif à la protection du potentiel scientifique et technique de la nation (PPST). L'autorisation d'accès à une zone est délivrée par le chef d'établissement, après avis ministériel favorable, tel que défini dans l'arrêté du 03 juillet 2012, relatif à la PPST. Un avis ministériel défavorable pour un poste affecté dans une ZRR aurait pour conséquence l'annulation du recrutement.

Politique de recrutement :
Dans le cadre de sa politique diversité, tous les postes Inria sont accessibles aux personnes en situation de handicap.