Stages OCamlPro 2023

Définitions récursives et schémas de compilation en ML

Niveau requis: M2

Durée du stage: 6 mois

Contexte du stage

L'équipe compilation d'OCamlPro, qui travaille sur le compilateur OCaml, s'intéresse à spécifier formellement les définitions récursives qui ont du sens et la façon de les compiler.

Le langage OCaml est relativement permissif sur ses définitions récursives, autorisant non seulement les fonctions récursives et mutuellement récursives mais aussi des définitions telles que des structures de données cycliques (contenant une référence vers elles-mêmes).

Mais il existe des définitions qui n'ont pas de sens (comme let rec x = x + 1), et que le compilateur rejette. Une définition formelle de quels termes sont autorisés est fournie dans l'article *A Practical Mode System for Recursive Definitions*, par A. Reynaud, G. Scherer et J. Yallop.

Sujet du stage

Le but de ce stage est de formaliser et d'implémenter un schéma de compilation plus efficace que le schéma par *global store* présenté dans l'article, qui est utilisé actuellement dans le compilateur. Un prototype d'implémentation a été proposé (https://github.com/ocaml/ocaml/pull/8956), mais il n'est pas correct: certains des termes acceptés par la définition précédente ne sont pas compilés correctement.

La spécificité de ce schéma de compilation est qu'il garantit que toutes les fonctions définies dans un groupe de définitions récursives sont compilées en une seule fois, sans patch a postériori. Exemple, tiré de l'article en question:

```
let rec mfib x =
  if x <= 1 then x
  else remember mfibs (x-1) + remember mfibs (x-2)
and mfibs = { f = mfib; values = empty_table () }
```

Le schéma de compilation standard produit:

```
let mfib = Uninitialized size_of_mfib
let mfibs = Uninitialized size_of_mfibs
let ((), ()) =
  (mfib <- fun x ->
    if x <= 1 then x
    else remember mfibs (x-1) + remember mfibs (x-2);
   mfibs <- { f = mfib; values = empty_table () })
```

Avec le nouveau schéma, cela devient:

```
let mfibs = Uninitialized size_of_mfibs
let rec mfib = fun x ->
  if x <= 1 then x
  else remember mfibs (x-1) + remember mfibs (x-2)
let () =
  mfibs <- { f = mfib; values = empty_table () }
```

La principale raison pour ce choix est que nous souhaitons appliquer la transformation plus tôt dans le compilateur, sur des langages intermédiaires où il n'est pas possible de calculer la taille de valeurs fonctionnelles ou de les modifier en place.

Il s'agira donc, pour le stagiaire, de reprendre ces travaux, et de produire une preuve de correction du nouveau schéma de compilation (éventuellement adapté) par rapport aux définitions récursives acceptées (avec possibilité de restreindre les définitions acceptées).

Si le temps le permet, le stagiaire pourra aussi travailler à l'intégration de ce schéma de compilation au compilateur OCaml.

Compétences requises

- Théorie des langages, systèmes de types, preuves de correction
- Programmation en OCaml

Compétences acquises

- Compilation de langages fonctionnels
- Détails du compilateur OCaml et de la sémantique du langage

Lieu du stage

Le stage se déroulera dans les locaux d'OCamlPro: 21, rue de Châtillon 75014 Paris.

Contact

Vincent Laviron, vincent.laviron@ocamlpro.com

À propos d'OCamlPro

OCamlPro SAS est une société issue de l'INRIA, créée en avril 2011, pour promouvoir l'utilisation de langages de programmation à l'état de l'art tels qu'OCaml et Rust dans le milieu industriel. Elle participe activement à des programmes de recherche et de développement visant à améliorer la sûreté et la sécurité des applications informatiques en général. Plus d'informations sur notre site web : https://www.ocamlpro.com/