Détection de fonction identité à la compilation

Stage 6 mois, niveau M2 Recherche.

Présentation d'OCamlPro :

OCamlPro SAS est une société issue de l'INRIA, créée en avril 2011, pour promouvoir l'utilisation du langage de programmation OCaml 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. Vous trouverez plus d'informations sur notre site web: https://www.ocamlpro.com/

Présentation d'OCaml :

OCaml est un langage fonctionnel avec typage fort, et il est courant d'avoir de nombreuses définitions de types, chacune correspondant à une structure de données et un invariant associé. Il est donc relativement courant de convertir d'un type à un autre, soit pour passer d'une représentation à une autre, soit pour refléter un changement d'invariant. Dans certains cas, la conversion oblige à parcourir une structure de données en entier pour reconstruire une nouvelle structure identique à la précédente,modulo le nom des constructeurs si les types sont différents.

Exemples :

```
(* Expression simple *)
 type 'a s_expr =
| SVar of 'a
| SAbs of 'a * 'a s_expr
| SApp of 'a s_expr * 'a s_expr

(* Expression étendue *)
 type 'a expr =
| EVar of 'a
| EAbs of 'a * 'a expr
| EApp of 'a expr * 'a expr
| ELet of 'a * 'a expr * 'a expr

let rec extend = function
| SVar v -> EVar v
| SAbs (x, body) -> EAbs (x, extend body)
| SApp (f, x) -> EApp (extend f, extend x)
```

```
(* From https://github.com/ocaml/ocaml/issues/9459 *)
 type empty = |
type 'a ast = Feature of 'a | Normal of int | Operation of 'a ast * 'a

 ast let rec conv : empty ast -> 'a ast = function

| Feature _ -> .
| Normal n -> Normal n
| Operation (a, b) -> Operation (conv a, conv b)
```

Dans les deux cas, il serait intéressant d'indiquer au compilateur par une annotation que la conversion devrait être une fonction triviale (l'identité) et qu'il peut donc optimiser l'implémentation. Plusieurs difficultés apparaissent dans ce problème. Tout d'abord, dans le premier exemple la sémantique de haut niveau du langage ne garantit pas que les représentations sont compatibles. Il faut donc faire attention à la sémantique que l'on choisit pour le langage, afin que les conditions de déclenchement de l'optimisation soient suffisamment souples mais restent compréhensibles pour un développeur.

De plus, détecter qu'une fonction est l'identité est un problème non trivial.Il faut prendre en compte les effets de bord potentiels, la terminaison, et potentiellement d'autres propriétés. On s'intéressera aussi à la complexité algorithmique des différentes solutions, et à la complétude (formalisation des conditions sous lesquelles l'analyse détecte correctement une fonction identité). Il sera particulièrement intéressant d'étudier les interactions entre cette optimisation et différents types d'autres optimisations.

Sujet de stage :

L'objectif de ce stage est d'étudier la faisabilité de cette analyse, y compris documenter les choix possibles et leurs avantages et désavantages respectifs. Le stagiaire pourra effectuer les tâches suivantes (dans un ordre au choix) :

- Étudier le compilateur pour trouver la représentation interne la plus appropriée pour cette analyse
- Implémenter un prototype, sur une ou plusieurs des représentations internes
- Réfléchir à comment exposer cette fonctionnalité aux développeurs (cela implique de communiquer avec la communauté OCaml en général)
- Formaliser l'analyse, et en prouver la correction
- Étudier les cas d'application sur le code existant

Le stage se déroulera au sein de l'équipe 'flambda' chez OCamlPro qui développe des optimisations sur le langage OCaml. Le travail réalisé lors de ce stage a vocation à être intégré au compilateur officiel par l'équipe 'flambda', mais l'intégration ne fait pas partie des objectifs du stage.