# Proposition de Stage de M1/M2 Recherche: Analyses “légères” de pointeurs pour programmes C

Laure Gonnord – Maître de conférences Université Lille 1
co-encadrement Paul Feautrier – Professeur Émérite Éns Lyon
Laure.Gonnord@lifl.fr

## 1. Contexte

Ce sujet se place à la frontière des thématiques de recherche Émeraude (Université Lille1 / Laboratoire d'Informatique Fondamentale de Lille) et Compsys (Inria / Éns Lyon).

L'avancée technologique des plateformes matérielles, avec des caractéristiques toujours plus complexes, rend nécessaire des analyses toujours plus fines du code source, que ce soit pour des buts d'optimisation à la compilation (optimisation de taille mémoire, de l'usage du cache...) ou d'optimisations lors de l'exécution.

L'équipe Compsys a une expérience dans la transformation de programme source à source, mais les programmes traités sont un noyau de C. L'équipe Émeraude a des expériences dans l'utilisation et l'optimisation des ressources à l'exécution, mais pour l'instant trop peu d'informations sur les programmes exécutés ne sont utilisées. L'implémentation d'une analyse de pointeurs est donc cruciale pour le passage à l'échelle de nos travaux.

## 2. Sujet

L'analyse d'un programme utilisant des pointeurs est connu pour être un problème difficile, qui n'admet en général que des solutions approximatives et conservatives, lesquelles le plus souvent inhibent toute optimisation. Pour un panorama de la recherche sur le sujet, on consultera [1].

Or, on constate que très souvent, les pointeurs ne sont utilisés - en C - que de façon rudimentaire :

- transmission d'arguments par adresses
- allocation dynamique de tableaux
- descripteurs de tableaux
- pseudo-indices.

Le sujet du stage est la réalisation d'un outil de détection de ces usages simples. Le résultat de cette analyse préliminaire sont destinées aux phases suivantes d'un analyseur du genre de l'outil `c2fsm` [2] qui transforme le programme en un automate interprété. Pour éviter d'avoir à se plonger dans un logiciel complexe et peu documenté, on procédera de préférence en mode source-à-source, c'est-à-dire que l'on engendrera un nouveau programme C, débarassé d'un maximum de pointeurs, et sémantiquement équivalent au programme original. Dans une étape unltérieure – qui ne fait pas partie du stage – on essaiera de générer directement un représentation intermédiaire (RI) conforme à celle de l'outil, et incluant les résultats de l'analyse.

## 3. Méthode

On propose d'utiliser un outil libre – par exemple LLVM-CLANG – pour une première analyse syntaxique, suivie d'une exécution symbolique de la RI. L'idée de base est que si la valeur d'un pointeur n'est jamais définie, ou si elle n'est définie qu'une fois, on peut remplacer l'objet pointé par un scalaire ou un tableau, suivant le contexte. Dans le cas contraire, on déclarera que la valeur du pointeur est inconnue.

On s'efforcera d'étendre le plus loin possible ces indications rudimentaires. Par exemple, on essaiera de reconnaître des codes de la forme :

```
  while(n-- > 0)
    *p++ = *q++;
```

en suivant les indications de l'article [3]. On ne cherchera cependant pas à reconstituer le tableau proprement dit, ou à identifier des boucles simples : ce travail est à la charge de `c2fsm`. Un autre cas intéressant est :

```
for(i=0; i<n; i++){
  p = malloc(N);
  for(j=0; j<N; j++)
    p[i] = ...
}
```

ce qui revient à construire un tableau à deux dimensions.

On cherchera enfin à traiter le cas d'un pointeur constant sur une structure qui contient elle-même des pointeurs constants ... recursivement. Ce style de programmation est souvent utilisé en milieu industriel pour suppléer à l'absence de descripteur de tableau en C.

## 4. Prérequis

Bonnes connaissances en C et en théorie de la compilation. Une familiarité avec LLVM et son langage d'implémentation (C++) serait appréciée.

## 5. Mots-clefs

Analyse statique de programmes C, interprétation abstraite, pointeurs, modèle mémoire, compilation source à source, LLVM.

## 6. Informations Pratiques

Le stagiaire pourra indifféremment travailler à Lyon (ENS Lyon, Lyon 7) ou Lille (Université Lille1, Villeneuve d'Ascq). Si il ne bénéficie pas déjà d'un traitement, la gratification de stage sera de 420 euros par mois. Certains frais d'hébergement pourront éventuellement être pris en charge si le stagiaire s'y prend assez tôt.

## Références

[1] Michael Hind. Pointer analysis : Haven't we solved this problem yet ? In *PASTE'01*, pages 54–61. ACM Press, 2001.

[2] Paul Feautrier and Laure Gonnord. Accelerated Invariant Generation for C Programs with Aspic and C2fsm. In *Workshop on Tools for Automatic Program AnalysiS, TAPAS'10*, Perpignan, France, September 2010.

[3] Björn Franke and Michael O'Boyle. Array recovery and high-level transformations for dsp applications. *ACM Trans. Embed. Comput. Syst.*, 2(2) :132–162, May 2003.