Analyse de bornes de boucles dans le code machine

Contexte

L'analyse statique du pire temps d'exécution (WCET) d'un programme requiert la connaissance de limites sur le nombre d'itérations de chacune des **boucles** du programme. Cette analyse WCET s'effectue sur le **code machine** du programme, pour une architecture donnée.

```
     ; ARM assembly code
8d14 movs r0, r0, lsl #1
8d18 bne 8d14
     ; How many 8d18 → 8d14?
```

Les informations de bornes de boucles peuvent être connues au niveau du code **source**, mais leur validité est facilement **remise en cause** par le processus de compilation. La détermination **manuelle** des bornes de boucles sur l'assembleur étant **fastidieuse** et sujette à erreur, la question d'une analyse automatique de bornes de boucles sur le code machine est tout à fait pertinente (et non triviale).

Des travaux ont été réalisés visant à appliquer le moteur de calcul de bornes proposé par [1] à tout programme qui soit analysable à l'aide de la bibliothèque de domaines abstraits MemCAD [2]. Un outil a été développé mais il ne traite pour l'instant que le langage C.

Objectifs du stage

L'objectif du stage est de réaliser les développements théoriques et pratiques permettant d'ajouter à l'outil d'analyse de bornes de boucles un *front-end* pour le code machine.

L'achèvement de cet objectif requerra probablement la réalisation des points suivants :

(a) Développement d'un moteur d'analyse, fondé sur MemCAD, pour l'assembleur générique proposé dans [3] ; implémentation en OCaml.

(b) Proposition d'un domaine abstrait MemCAD dont la précision soit adaptée aux besoins de l'analyse de bornes de boucles.

(c) Amélioration des performances de l'analyse de bornes de boucles ; développement fondé sur la mise sous forme normale de Jordan offert par la bibliothèque Python SymPy.

Références

[1] Jeannet, Schrammel, Sankaranarayanan : Abstract Acceleration of General Linear Loops

[2] http://www.di.ens.fr/~rival/memcad.html

[3] Cassé, Birée, Sainrat : Multi-architecture Value Analysis for Machine Code

Qualités et compétences attendues

Goût pour la programmation, pour la compilation, pour la sémantique des langages de programmation, rigueur, maîtrise des mathématiques discrètes et de l'algèbre linéaire, connaissance des langages OCaml et Python, compréhension fine des langages assembleurs.

Contact et localisation

Pascal Sotin – pascal.sotin@irit.fr

Le stage s'effectuera dans les locaux de l'équipe TRACES à l'IRIT (Toulouse).