Colloquium de Patrick Cousot

29 Septembre 2016

`http://www.lip6.fr/colloquium/`

Programme

Rencontres avec les chercheurs du LIP6 et de l'IRILL de 9:30 à 12:30

***Master classes* de 14:30 à 17:00 (Salle Noguez 26-00/101)**

14:30–14:55	Thomas Blanc (OCamlPRO) *Implémentation d'une analyse tout-programme sur OCaml*
14:55–15:20	Thibault Suzanne (LIP6, APR) *Interprétation abstraite modulaire dans les modèles mémoire faibles*
15:20–15:45	Ghiles Ziat (LIP6, APR) *Réduire l'approximation en programmation par contraintes en utilisant les produits de l'interprétation abstraite*
15:45–16:10	pause
16:10–16:35	Huisong Li (ÉNS, Antique) *Dynamic clumping of shape abstractions*
16:35–17:00	Jiangchao Liu (ÉNS, Antique) *Inference of structural invariants on arrays storing dynamic structures*

Cocktail à 17:15 (à côté de l'Amphi 15)

Colloquium de Patrick Cousot à 18:00 (Amphi 15)

Master classes – Résumés

- **Implémentation d'une analyse tout-programme sur OCaml**

  Thomas Blanc (OCamlPRO)

  Si le typage permet d'obtenir de nombreuses garanties de sûreté sur les programmes, certains bugs échappent toujours à la vigilance du codeur et du compilateur. Dans le cas de programmes impératifs, l'interprétation abstraite permet de trouver de nombreuses erreurs. Cependant, il est difficile de mettre en place une interprétation abstraite sur un langage fonctionnel, notamment à cause des fonctions d'ordre supérieur qui rendent le graphe de flux de contrôle difficile à évaluer avant l'analyse de valeur. Nous avons donc mis en place un graphe de flux de contrôle modifié dynamiquement lors de l'analyse qui permet non seulement de suivre les appels de fonctions mais aussi de caractériser effectivement et complètement le graphe de flux une fois l'analyse achevée.

- **Interprétation abstraite modulaire dans les modèles mémoire faibles**

  Thibault Suzanne (LIP6, APR)

  La méthode naïve d'interprétation abstraite de programmes concurrents, qui consiste à travailler sur le produit des graphes de flot de contrôle des processus, ne passe pas à l'échelle. Nous cherchons donc à concevoir des méthodes modulaires, qui analysent chaque processus individuellement en prenant en compte les effets possibles de chaque exécutions sur celle des autres processus. Si l'abstraction de ces interférences est importante pour l'analyse de programmes séquentiellement cohérents, elle est cruciale en présence de mémoire faible en raison de l'augmentation significative de la taille des états concrets. Nous présenterons nos travaux en cours sur l'analyse modulaire de ces programmes, ainsi que les pistes de recherche pour permettre un meilleur passage à l'échelle.

- **Réduire l'approximation en programmation par contraintes en utilisant les produits de l'interprétation abstraite**

  Ghiles Ziat (LIP6, APR)

  Nous nous intéressons à des sur-approximations de l'ensemble des solutions d'un problème de satisfaction de contraintes. Nous utilisons une méthode "classique" de résolution alternant propagation et exploration en adaptant ces notions aux domaines abstraits de l'interprétation abstraite. Nous nous intéressons en particulier aux domaines relationnels de l'interprétation abstraite et nous expliquons en quoi leur utilisation, mixée à celle de domaines non relationnels, nous permet d'obtenir une méthode de résolution efficace. Nous verrons ensuite une nouvelle stratégie d'exploration, bien adaptée aux domaines abstraits que nous utilisons, avant de parler de quelques résultats expérimentaux.

- **Dynamic clumping of shape abstractions**

  Huisong Li (ÉNS, Antique)

  One of the key challenges in shape analysis is to manage the number of disjuncts, each

of which describes a particular shape of memory states. In order to keep low disjunct number for scalability, disjuncts describing similar memory states should be partitioned into a group and joining only those within each group. Therefore, we propose to abstract shape abstractions by their silhouettes, which keep only the information about reachability among program variables content. These silhouettes are grouped into equivalence classes, which in turn are used to decide which disjuncts of the shape analysis should be joined together: disjuncts whose silhouettes belong to the same equivalence class get joined to a single disjunct.

- **Inference of structural invariants on arrays storing dynamic structures**

  Jiangchao Liu (ÉNS, Antique)

  Low-level code commonly use dynamic structures (e.g. lists) nested into arrays, for instance to store lists of processes in operating system services, and to avoid relying on dynamic allocation in embedded software. The verification of such programs is very challenging as it requires reasoning both about array structures and about complex structural invariants. We propose an automatic static analysis framework for the verification of this programming pattern. Our analysis non-contiguously partitions an array into groups of cells with similar properties. The framework attaches around each partitioned group inductive predicates to abstract dynamic structures, and numeric/set relations to capture array-specific properties in order for precise automatic inference. It successfully verifies several components of operating system kernels, including the sortness of a priority task list stored in an array.