Département d'Informatique de l'Ecole Normale Supérieure

Accurate and Efficient Disjunctions of Abstract Properties of Memory States

Internship location : École Normale Supérieure ; 45, rue d'Ulm ; 75 230, PARIS.

Team : Équipe Sémantique et Interprétation Abstraite / Équipe-Projet "ANTIQUE".

Advisor & Contact : Xavier RIVAL (*e-mail* : `rival@di.ens.fr`, tél : 01 44 32 21 50, fax : 01 44 32 21 51)

Internship topic :

Shape analyses such as [1] aimed at inferring structural invariants for complex data-structures often need to rely on case splits, which effectively amounts to introducing some form of disjunctions in the analysis. For instance, if the analysis needs to reason about an update into a region that is known to store a *list structure* represented by a *summary* (abstract predicate describing lists of possibly unbounded size), it is common that a case analysis has to be performed, where the null pointer / empty list has to be treated separately from a non null pointer / non empty list :

$$\mathbf{list}(x) \quad \equiv \quad \begin{cases} & \mathbf{emp} \wedge x = 0 \\ \vee & \\ & x.n \mapsto x' * x.d \mapsto x'' * \mathbf{list}(x') \wedge x \neq 0 \end{cases}$$

In this case, the analyzer will compute symbolic disjuncts of several shape graphs that express very different properties, so that using their join (into a single shape graph) would cause an inacceptable loss in precision.

Such disjuncts are necessary, but also represent a major challenge for scalability. Thus static analyzer should try to eliminate disjuncts that are "similar enough" so that an abstract join will not deteriorate precision too much. Unfortunately, determining which abstract heaps are similar enough to be merged is very costly as there is currently no other solution than testing equality of graphs, or attempting to compute their union and giving up when the precision loss is too high.

The most common techniques to handle such case splits consist in disjunctive completion [2] or some simplistic implementation of trace partitioning [3]. Disjunctive completion simply describes finite sets of disjuncts without any structure, whereas trace partitioning keeps track of the history of each disjunct. None of these approaches is fully satisfactory in shape analysis though : even trace partitioning approaches do not know how to merge efficiently disjuncts that are similar enough.

The purpose of this internship is to explore a solution to this problem, so as to speed up significantly shape analyses, while retaining their ability to compute very precise data-structure invariants. The proposed approach consist in defining light-weight abstractions of the shape graphs in order to decide when shape graphs are similar enough that their join should not lose too much precision. Possible abstractions include numbers of basic predicates (such as points-to edges), local graph patterns (relations between pointer values), or recency abstraction (when have graph areas been updated last by the analysis). This amounts to doing a static analysis of abstract states, and using the results of this analysis so as to guide the main analysis.

Internship tasks :

The purpose of the internship is to explore a possible solution to the problem of disjunctions of heap properties, both in the theoretical and in the practical points of view. The expected tasks are the following :

1. **Define a framework for the abstraction of local properties of abstract states**. In this phase, a general model for abstractions of abstract heaps will be developped, and used in order to formalize a few basic abstractions. Those will be chosen by observing how the shape graph join operators loses precision. Ideally, this work will result in a characterization of which abstractions can identify acceptable or inacceptable precision losses
2. **Design algorithms to compute the abstractions of local properties**. This task aims at making the analysis automatic and efficient. Two main approaches can be considered : the most obvious one consists in computing the approach from the shape graphs, but it may turn out costly, so a second approach would make this process incremental.

3. **Implementation and empirical evaluation**. The last part consist in implementing the techniques formalized earlier in the internship and in assessing their effectiveness. Implementation will be done in the **MemCAD** static analyzer. Experimentation may be carried out on libraries for basic data-structures or on medium size open source projects like the Minix micro-kernel.

Moreover, a funding to pursue this work as part of a PhD is available as part of the **MemCAD** ERC project.

Pré-requis :

For this internship, it would be preferable that the student is familiar enough with abstract interpretation techniques as taught in lecture "2–6 Abstract Interpretation : Application to Verification and Static Analysis".

Références

[1] Bor-Yuh Evan Chang and Xavier Rival. Relational inductive shape analysis. In POPL'08, pages 247–260, 2008.

[2] Patrick Cousot, Radhia Cousot. Abstract Interpretation and application to pogic programs. In Journal of Logic Programming, pages 103–179, 1992.

[3] Xavier Rival and Laurent Mauborgne. The trace partitioning abstract domain. ACM TOPLAS 2007.