Paper Review 《Shape Analysis with Structual Invariant Checkers》

Paper Info

《Shape Analysis with Structual Invariant Checkers》SAS 2007

Bor-Yuh Evan Chang, Xavier Rival and Georage C. Necula

Main Contributions

Shape analysis is one of the most traditional research topic in the program analysis. All of the approaches in the shape analysis face the trade-off between the precision and termination or efficiency. In most of the previous work, the summary node is defined to abstract several locations in the memory, which cause the loss of the precision.

The key contribution of this work is to abstract the locations with the logical checkers instead of the summary node without useful information. In this paper, the author propose a lightweight, automatic shape analysis based on the developer-supplied structural invariant checkers, including complete and partial checkers. The partial relation and widening are defined, and the analysis is in the framework of abstract interpretation.

To sum up, the major contributions include

- Develop a shape analysis based on programmer-supplied invariant checkers
- Define partial checkers to describle the properties which only hold partially
- Develop an automatic widening strategy to make full use of iteration history of analysis, in order to guide the weakening of shape analysis

Technical Details

There are several important observations in this work.

- Checkers can be viewed as programmer-supplied summaries of heap regions bundled with a usage pattern for such regions.
- Some properties only hold partially. The information should be recorded in the abstraction of memory.
- The iteration history of the analysis can be used to guide the weakening of shape invariants, which perhaps could apply to other shape analyses.

Based on the above three observations, the authors propose a new approach of shape analysis, and the major contributions in the previous section correspond to these three observations respectively. As a whole, the shape analysis in this work is still an instance of abstract interpretaion and data flow analysis.

Similiar to other shape analyses, they use shape graph to abstract memory state. A memory state includes the points-to relation, the seperating conjunction, and the empty memory state from separation logic. Besides, a pure state with non-points-to constraints is designed to track disequalities, and inductive structure checkers are defined to indicate how the user intends to access that region of memory.

Based on the above memory abstraction, the following operations are proposed:

- Abstract transition: Just like other data flow analysis
- Checker unfolding: In many other instances of shape analysis, the multiple regions might be aggregated by a summary region(folding). It is hard to split a summary region in the next steps(unfolding), because the information of the regions represented by the summary region has been lost in the unfolding. In order to preserve this kind of information, developer-defined checkers are used to describe the properties in the shape graph.
- History-guided folding: It is apparent to find that the history of shape graphs in the iteration provides useful information. To assure the termination of the iteration, the widening operation is defined based on the approximation test. In the approximation test, two shape graphs are compared to check whether they have the paritial relation. Figure 2 and 3 illustrate the process of approximation test and widening respectively.

All necessary components in abstract interpretation are well-defined in this work, and it is quite easy to perform the analysis in the framework of DFA. The experimental results demonstrate the effectiveness of shape analysis in the structures in regular sharing patterns, including skip lists, circular list, doubly-linked list and trees with parent pointers.

Some Criticisms

Although the combination of shape graph and seperation logic is a creative idea and more powerful than traditional shape graph with summary nodes, there are some pain points in this work.

- If the code tranverses the structures in a different direction than the checkers, the shape analysis is difficult to perform. The effectiveness of the approach is not the same for all code using these structures.
- In their meomory abstraction, the partial checkers edges go from one hole. This formulation can only handle code that use cursors along a path through the structure, and the code that uses multiple cursors along different branches of a structure can not be well-analyzed in the work.
- The checkers are developer-defined, and this makes the approach more scalable and parametric. However, some general checkers should be defined as core checkers to describe the general patterns in some certain kinds of structures. Some ideas on core predicates in the 《A Relational Approach to Interprocedural Shape Analysis》 can be referred.