Paper Review 《Shape Analysis as Generalized Path Problem》

Paper Info

《Shape Analysis as Generalized Path Problem》PEPM 1995

Thomas Reps

Main Contributions

In this paper, the author proposes a new approach to finding solutions to shape-analysis problems by formulating them as generalized graph-reachability problems. This is the core insight of his work. A path is considered to connect two vertices only if the concatenation of the labels on the edges of the path is a word in a certain CFL.

The conventional algorithm allows us to get polynomial bounds on the running time of an instance of shape analysis. Moreover, the author proposes a demand algorithm to determine shape information selectively in a bottom-up style, so particular variables at particular points in the program, rather than for every variable, are considered.

To sum up, the major contributions include

- Formulate the shape-analysis problem as a CFL reachability problem.
- Propose a demand algorithm for shape analysis in a bottom-up way

Technical Details

Familiar to other work on shape analysis, this work also takes list-reversing as a motivating example. The author firstly considers the Lisp-like imperative languages, and considers three functions: `car`, `cdr` and `cons`. Unlike the approaches of most of instances of shape analysis, the approach in the work does not abstract the memory locations. It models the heap-allocated structures in a new way and try to solve the CFL reachability problem.

In Figure 1, the author gives an example of list-reversing. The shape information of each heap-allocated structures at each program point needs to be determined. A shape information is a set of sequences in `{hd, tl}* \times {at, nil}`. Each sequence represents a possible root-to-leaf path. This is the biggest difference from the abstraction model in other work.

The author defines four kinds of CFL-reachability problems and convert the specific subproblems in shape analysis to one of these four. All of them can be solved by the generalizations of the CYK algorithm for context-free recognition.

The second important contribution is the demand algorithm. Generally speaking, the shape of all the variables at every program points needs to be considered. However, if we only care about a certain structure at some specific program points, we can decrease the time cost greatly just by a bottom-up approach. This is similar to the calculation of Fibonacci sequence.

Some Criticisms

Although the work provides a brand new way to solve this traditional problem, it does have some limitations:

- The variables in the context free grammar might be different and need to be customed for each programs. In this work, the motivating examples only contain three functions. However, if more functions are used, the semantics need to be modeled manually, just like `cons` in this work. This might be time-consuming and limits the scalability of this work.
- The Mogensen join is used in the work. However, this can cause the loss of precision. A better approach is to track the condition of each branch if possible. If the condition of the branch only contains the variables of which the information is sufficient and can be retrieved from other analyze, we can use the separate mode. Otherwise, the approach in this work is applied. This can increase the precision in a certain degree.