Program Analysis in First-order Languages

Dimitris Vardoulakis

@inproceedings{kildall,
  author = {Gary A. Kildall},
  title = {A unified approach to global program optimization},
  booktitle = {Proceedings of the 1st annual ACM SIGPLAN Symposium on Principles of Programming Languages},
  pages = {194--206},
  year = {1973}
}

This paper presents a general framework for dataflow analysis of procedureless programs. The method applies to both forward and backward problems and is also parameterized over the transfer functions of a particular problem. The author provides several instances of his framework, including constant propagation and common-subexpression elimination. He proves that the algorithm terminates if the lattice of dataflow facts has no infinite chains and that, in the case of distributive transfer functions, the solution coincides with the join-over-all-paths solution.

This paper unified the various methods of the time for dataflow analysis. The framework introduced is what later became known as the Monotone Framework and is still used today for intraprocedural analysis.

@book{sharir,
  author = {Micha Sharir and Amir Pnueli},
  title = {Two approaches to interprocedural data flow analysis},
  booktitle = {Program Flow Analysis: Theory and Applications},
  editor = {Muchnick and Jones},
  publisher = {Prentice Hall International},
  year = {1981}
}

This paper presents two ways of doing interprocedural dataflow analysis: the functional approach and the call-strings approach.

In the functional approach, the authors show how to compute a transfer function for each procedure \( f \) by composing transfer functions of its basic blocks. The
new transfer function can be applied at every call site of $f$ to approximate
the effect of the call on the dataflow at the corresponding return site. The
major difficulty with the functional approach is how to represent the transfer
functions when the dataflow lattice is infinite. In this case, a table representation
is impossible and there is no general way of finding when a function admits a
symbolic representation. When the lattice is finite, the functional approach
gives a terminating algorithm that computes the optimal solution.

In the call-strings approach, the dataflow values at each program point $p$ are
tagged with a history of the pending calls at $p$. In programs with recursive
procedures, the call strings can be unboundedly long so the authors suggest two
ways of getting around this problem. One possibility is to remember a prefix-
closed, finite set of call strings $\Gamma_0$ and forget all dataflow tagged by longer call
strings. This method is unsound in general. However, when the dataflow lattice
is finite, the authors show how to statically find the length of the longest string
in $\Gamma_0$ and get a safe and optimal solution. The other possibility is to create a
map from the infinite set of call strings to a finite set. This method is sound
but merges information from distinct call sites.

This is a very significant paper. Most subsequent work on interprocedural anal-
yses is a variation of one of the two methods presented here.

@inproceedings{reps,
    author = {Thomas W. Reps and Susan Horwitz and Shmuel Sagiv},
    title = {Precise Interprocedural Dataflow Analysis via
             Graph Reachability},
    booktitle = {Proceedings of the ACM SIGPLAN Symposium
                 on Principles of Programming Languages},
    year = {1995},
    pages = {49--61}
}

This paper shows how to implement the functional approach of Sharir and Pnueli
very efficiently for a certain class of dataflow problems. Specifically, when the
dataflow facts are sets and the transfer functions are distributive, the transfer
functions admit a very compact representation and the dataflow problem can be
turned to a graph reachability problem. Their tabulation algorithm solves the
reachability problem without merging any information from distinct call sites
of the same procedure.

Initially, Sharir and Pnueli’s paper was mostly cited for call strings and the
functional approach did not take off. Reps et al. showed how the functional
approach can be made practical and thus firmly established it as a good way of
doing interprocedural analysis.