CS638: Program Analysis

Uday Khedker

December 12, 2006

1. ***Overall Plan***.

   The course begins with the mathematical foundations of program analysis and then concentrates on advanced applications of data flow analysis. The advances are in terms of data flow properties which cannot be represented using bit vectors which is the usual model discussed even in advanced courses. The thrust of the course is on Non-Separable Data Flow Frameworks.

2. ***Course Contents***.

   (a) Mathematical Foundations of Program Analysis. Lattices: partial orders and approximations, bounded lattices, complete lattices.
   Function spaces: monotonicity, distributivity, separability, boundedness.
   Fixed Point Computations: least fixed point and induction, greatest fixed point and coinduction.
   Abstract interpretation: concretisation and abstraction functions, Galois connection.

   (b) Non-Separable Data Flow Frameworks. Modelling non-separable frameworks. Defining information flow paths. Characterizing complexity of non-separable data flows.

   (c) Heap Reference Analysis. Representations of heap properties: shape graphs, alias graphs, k-limiting, compact representations, access graphs. Data flow analyses for discovering aliasing, liveness, availability, and anticipability. Safety of NULL assignment insertion.

   (d) Interprocedural data flow analysis
   Context sensitive/insensitive analysis. Functional approach. Call strings approach. Call strings graphs. Call strings approach for non-separable problems.

3. ***Reading Material***.

   There is no particular book which can be used as a text book. The course is based on research papers and selected book chapters listed in the reference list.

4. ***Prerequisites***. First course in compiler construction and some exposure of data flow analysis.

References

[1] Michael Hind, Michael Burke, Paul Carini, and Jong-Deok Choi. Interprocedural pointer alias analysis. *ACM Transactions on Programming Languages and Systems*, 21(4):848–894, 1999.

[2] U. P. Khedker. Data flow analysis. In Y. N. Srikant and Priti Shankar, editors, *The Compiler Design Handbook: Optimizations & Machine Code Generation*. CRC Press, USA, 2002.

[3] U. P. Khedker, D. M. Dhamdhere, and A. Mycroft. Bidirectional data flow analysis for type inferencing. *Computer Languages, Systems and Structures*, 29(1-2):15–44, 2003.

[4] U. P. Khedker, A. Sanyal, and A Karkare. Heap reference analysis using access graphs. 2006. http://arxiv.org/abs/cs.PL/0608104.

[5] U. P. Khedker and B. Sathe. Complexity of intraprocedural and interprocedural analysis of non-separable data flow frameworks. 2006. (Under Preparation).

[6] F. Nielson, H. R. Nielson, and C. Hankin. *Principles of Program Analysis*. Springer-Verlag, 1998.

[7] Ran Shaham, Eran Yahav, Elliot K. Kolodnere, and Shmuel Sagiv. Establishing local temporal heap safety properties with applications to compile-time memory management. In *SAS '03: Proceedings of the 10th International Symposium on Static Analysis*, pages 483–503, London, UK, 2003. Springer-Verlag.

[8] M. Sharir and A. Pnueli. Two approaches to interprocedural data flow analysis. In S. S. Muchnick and N. D. Jones, editors, *Program Flow Analysis : Theory and Applications*. Prentice-Hall Inc., 1981.