Research Interests

Research Interests: General

- Program analysis and transformation.

Research Interests: Specific

- Study of fundamental properties of different code transformations and associated program analyses to identify common features and to seek possibilities for improved algorithms.

Research Experience

- Compilers: Code Transformations(Aug 1994 - July 1999), Department of Computer Science and Automation, Indian Institute of Science, Bangalore.
- Formal Methods in Concurrent Computation(Jan 1989 - Aug 1990), Leningrad Electro Technical Institute(LETI), St.Petersburg, Russia(USSR Govt. Scholarship). Got introduced to formal models of concurrent computations such as Communicating Sequential Processes and Petri Nets.

Publications

- Nabizath Saleena and Vineeth Paleri. Global Value Numbering for Redundancy Detection: A Simple and Efficient Algorithm. In Proceedings of the 29th Annual ACM Symposium on Applied Computing, pages 1609-1611. ACM 2014.
- Saleena Nabeezath and Vineeth Paleri. A Simple Algorithm for Global Value Numbering, arXiv:1303.1880v1 [cs.PL], 2013.
- Vineeth Paleri, Y.N.Srikant, and Priti Shankar. Partial Redundancy Elimination: A Simple, Pragmatic, and Provably Correct Algorithm. *Science of Computer Programming* 48, 1(2003), 1-20.
- Vineeth Paleri. Automatic Generation of Code Optimizers from Formal Specifications. In Y.N.Srikant and Priti Shankar, Editors. *The Compiler Design Handbook: Optimizations and Machine Code Generation.* CRC Press, 2002.
- Vineeth Paleri, Y.N.Srikant, and Priti Shankar. A Simple Algorithm for Partial Redundancy Elimination. *ACM SIGPLAN Notices* 33, 12(1998), 35-43.

Abstracts of Important Research Contributions

- Nabizath Saleena and Vineeth Paleri. Global Value Numbering for Redundancy Detection: A Simple and Efficient Algorithm. In Proceedings of the 29th Annual ACM Symposium on Applied Computing, pages 1609-1611. ACM 2014.

  Abstract: Global Value Numbering (GVN) is a method for detecting equivalence among program expressions. Here we consider the problem of GVN in the context of redundancy detection and present a simple, polynomial time algorithm for the same. The basic idea is to use the concept of value expression - an abstraction of a set of expressions - enabling a representation of the equivalence information which is compact and simple to manipulate. The algorithm detects expression equivalences that are required for identifying value based redundancies. In addition, it achieves completeness in detecting equivalence among variables.

- Vineeth Paleri, Y.N.Srikant, and Priti Shankar. A Simple Algorithm for Partial Redundancy Elimination. *ACM SIGPLAN Notices*, 33, 12(1998), pp. 35-43.

  Abstract: The paper proposes a new algorithm for partial redundancy elimination based on the new concepts of *safe partial availability* and *safe partial anticipability.* These new concepts are derived by the integration of the notion of *safety* into the definitions of partial availability and partial anticipability. It is both computationally and lifetime optimal and requires four unidirectional analyses. The most important feature of the algorithm is its simplicity; the algorithm evolves naturally from the new concept of safe partial availability.

- Vineeth Paleri. Automatic Generation of Code Optimizers from Formal Specifications. In Y.N.Srikant and Priti Shankar, Editors. The Compiler Design Handbook: Optimizations and Machine Code Generation. CRC Press, 2002.

  Abstract: Code optimization or code transformation is a complex function of a compiler involving analyses and modifications with the entire program as its scope. In spite of its complexity, hardly any tools exist to support this function of the compiler. This article presents the development of a code transformation system, specifically for scalar transformations, which can be used either as a tool to assist the generation of code transformers or as an environment for experimentation with code transformations. The system is unique of its kind, providing a complete environment in which one can specify a transformation using dependence relations - in the specification language we have designed, generate code for a transformer from its specification, and experiment with the generated transformers on real-world programs.