INCREMENTAL COMPUTATION: A SEMANTICS-BASED SYSTEMATIC TRANSFORMATIONAL APPROACH

Yanhong Annie Liu, Ph.D.
Cornell University 1996

Incremental computation takes advantage of repeated computations on inputs that differ slightly from one another, computing each new output incrementally by making use of the previous output rather than from scratch.

This thesis concerns the theory, design, and implementation of a general approach to incremental computation. It also elucidates the essence of improving the efficiency of computations by relating it to incremental computation. Our general approach allows incremental computation to be obtained systematically from non-incremental computation and program efficiency to be systematically improved.

This research focuses on identifying the fundamentals of efficient incremental computation out of domain-specific properties and language-specific features, devising a general framework that accommodates these fundamentals, and developing a systematic approach based on the framework that exploits program semantics.

Three fundamental aspects of incremental computation are identified: avoiding repeated identical computations, caching useful intermediate results, and discovering appropriate auxiliary information. Given a program f and an operation $\oplus$, an incremental program is developed to compute $f(x \oplus y)$ efficiently by using $f(x)$, the intermediate results computed in computing $f(x)$, and auxiliary information about x that can be inexpensively maintained.

The approach in this thesis is formalized for a simple functional language, but the underlying principles also apply to other programming languages. It exploits program semantics to discover incrementality that is not directly embedded in primitive operators and takes into consideration properties of application domains as well. It is composed of step-wise program analysis and transformation modules that can, for the most part, be mechanized.

Since every non-trivial computation proceeds by iteration (or recursion), the approach is used straightforwardly for achieving efficient computation in general, by computing each iteration incrementally using an appropriate incremental program. This method is applied to problems in interactive systems, optimizing compilers, transformational program development, *etc.* The design and implementation of a prototype system, CACHET, for obtaining incremental programs is also described.