abstract

COMPUTER SCIENCE/DISCRETE MATH I
Topic:

Speaker:

Affiliation:

Date:

Time/Room:

Program checking, program self-correcting and program self-testing were pioneered by [Blum and Kannan] and [Blum, Luby and Rubinfeld] in the mid eighties as a new way to gain confidence in software, by considering program correctness on an input by input basis rather than full program verification.

In this work we prove a new composition theorem in the field of program checking, which essentially shows how to build program checkers that delegate their work to the (potentially faulty) program that is being checked, thus improving the checker's efficiency. This composition theorem enables us to address several central issues in the area of program checking and to prove a multitude of results. We highlight two of our main contributions:

- We show checkers (testers and correctors) for a large class of functions, that are {\em provably} more efficient, in terms of circuit depth, than the optimal program for the function. This is done by relating (for the first time) the complexity of checking to the complexity of computing.

- Blum et.al. considered a non-standard (and weaker) notion of program checkers (testers and correctors) for functions that belong to a {\em library} of several related functions. They showed checkers, testers and correctors in this notion, for a library of matrix functions containing multiplication, inversion, rank and determinant. We show how to eliminate the need for this library, and instead we show {\em standard } checkers, testers and correctors for all of the above mentioned matrix functions. By abstracting these ideas, we get another general methodology to design program checkers (testers and correctors).

abstract

Work by Shafi Goldwasser, Dan Gutfreund, Alex Healy and Tali Kaufman and Guy Rothblum.