Solving range constraints using Interval analysis

Student name: Shlomit Arian
Supervisors: Prof. Amir Ben-Amram & Dr. Iris Rosenblum
Presented  May 2012

Our goal: Given a program in an imperative language, we want to bound the possible values of every variable in the program.

Motivations:
- Detecting variable overflow.
- Checking array bounds
- Detecting unreachable areas in the code.
- Detecting uninitialized variables.
- Bounding the running time of a program.

The means: Converting the program to a set of range constraints that over-approximate the range of possible values for each variable, and solve them using an efficient algorithm.

How?
- For each program variable $x$ we define a set of range variables $X_i$, such that $X_i$ denotes the variable instance at location $i$ in the program.
- We convert the code to a set of range constraints (we only deal with a specific type of linear range constraints). For each range variable $X$, we are interested in its smallest range in any possible solution for the constraint set, called the least solution for $X$.
- We represent the set of constraints using a directed graph called the Constraint graph, where each vertex stands for a range variable and each arc represents a constraint.
- In order to solve the constraint set and get its least solution, we propose a new Range constraint algorithm based on Bellman-Ford Shortest-path algorithm, whose running time is $O(nm^2)$ for $n$ variables and $m$ constraints. The algorithm is a constraint relaxation algorithm and is based in part on a paper of Su and Wagner, who attempted to solve to the same problem in a similar manner.