Liveness-based Sparse Points-to Analysis

Aditi Raste (SPPU)
Swati Jaiswal (VNIT, Nagpur)
Uday Khedker (IIT Bombay)
Alan Mycroft (University of Cambridge)

Overview

Pointer analysis

- Uncovers indirect reads, writes and control flow in a program by computing sets of pointer-pointee pairs.
- Major challenge is to improve the scalability of pointer analysis without compromising on precision.

Research on Pointer Analysis at IIT Bombay

- Long term goal has been to perform flow- and context-sensitive points-to analysis on millions of lines of code.

Liveness-based Pointer Analysis (LFCPA)

- Computes a set of pointers at each program point that are used later in the program.
- Computes points-to information only for such live pointers.
- Removes useless points-to information significantly.

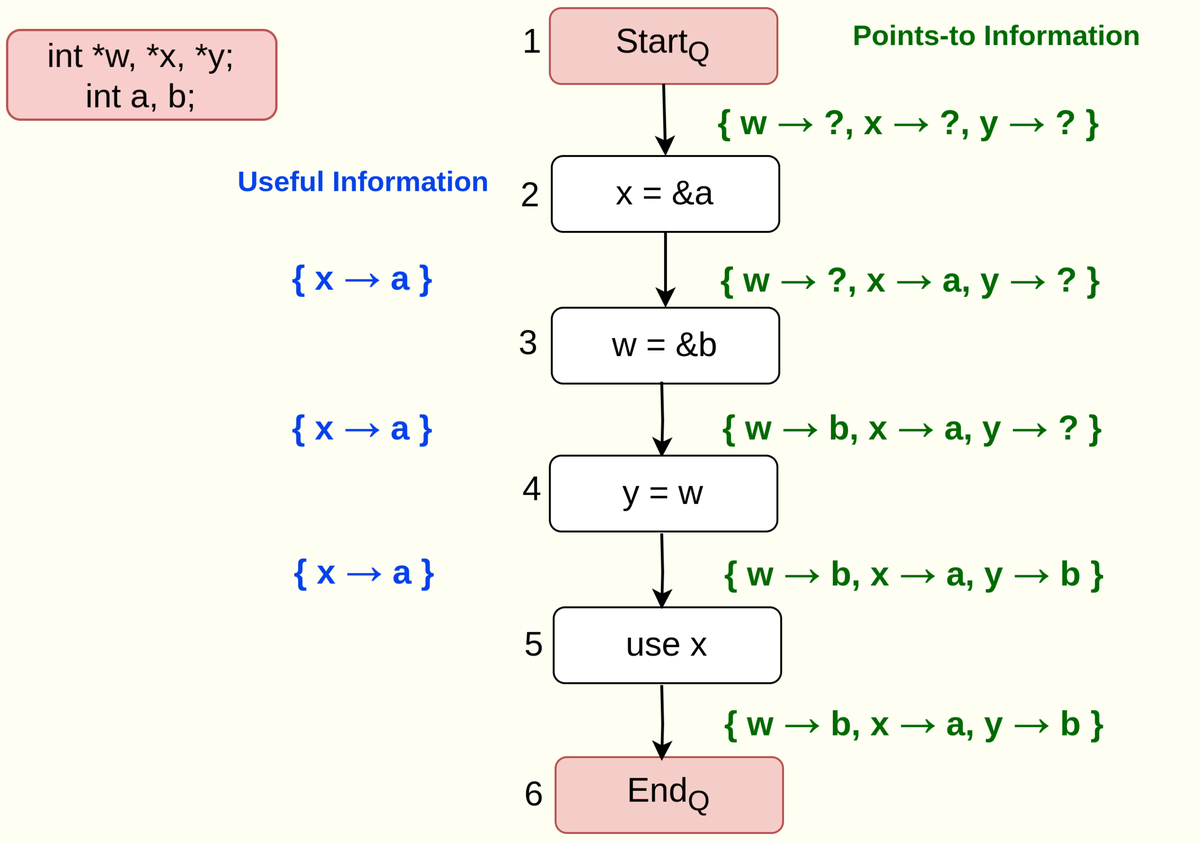
Liveness-based Sparse Pointer Analysis

- Sparse analysis computes information at relevant program points thereby improving scalability.
- Propagates points-to information from the point of generation directly to the use point computed by the liveness analysis.
- Reduces the amount of information propagated across program points significantly.

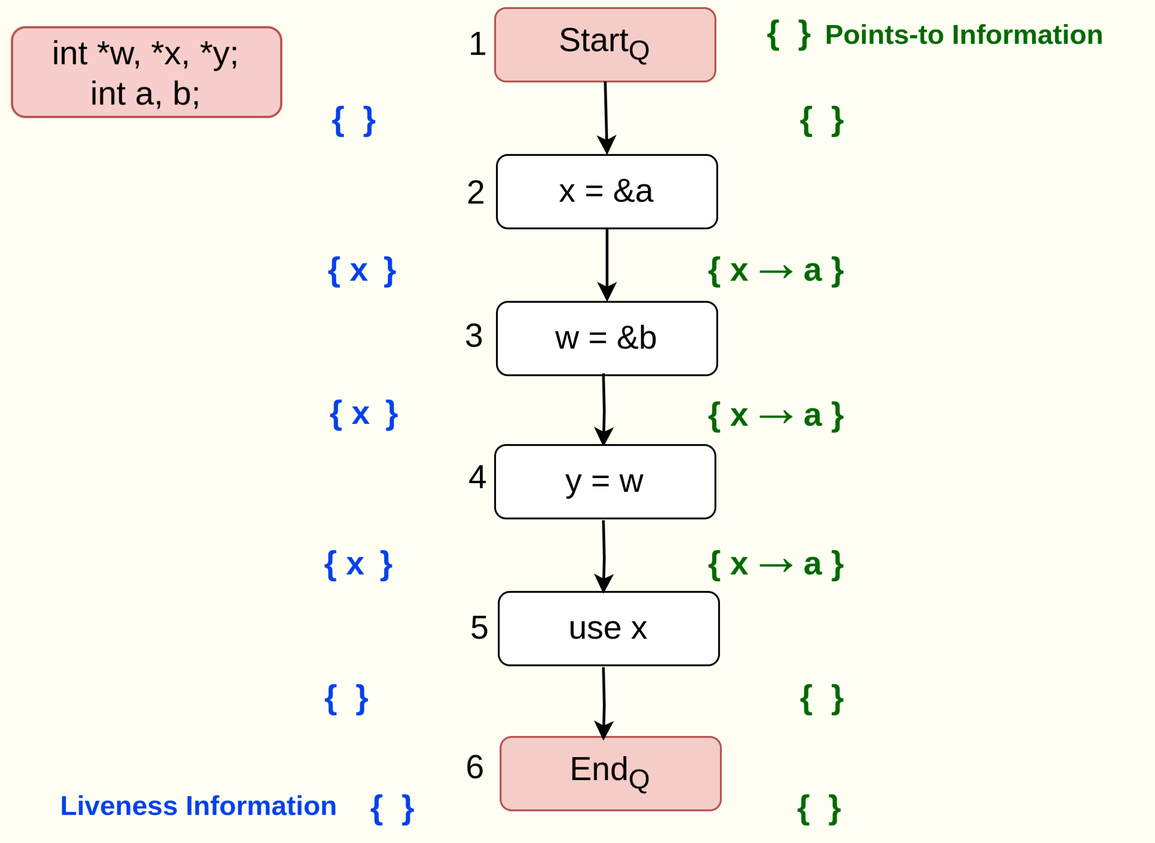
Implementation Details

- Analysis is being implemented in LLVM (14.0.0) on Ubuntu.
- Programs in C from SPEC CPU 2017 benchmarks are used for measuring the efficiency and scalability gains of the analysis.

Intraprocedural Flow-Sensitive Points-to Analysis



Intraprocedural Liveness-based Points-to Analysis



Intraprocedural Liveness-based Sparse Points-to Analysis

Interprocedural Liveness-based Sparse Points-to Analysis

The liveness of *w* reaches the call node and it is live in callee *Q*. However, the points-to information {w→b} bypasses the call to *Q as w* is used after the call to *Q* and not in procedure *Q*.