Efficient Whole Program Path Tracing

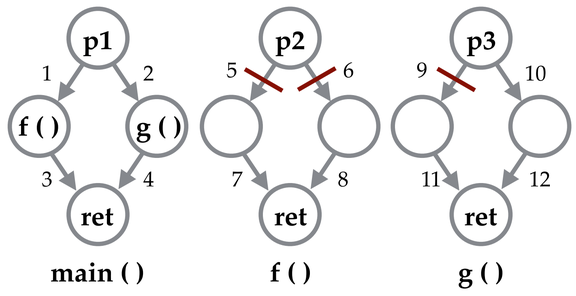
Sridhar Gopinath, Murali Krishna Ramanathan — Indian Institute of Science
Suresh Jagannathan — Purdue University

Problem Statement

Finding minimum number of instrumentation points in the program to derive precise the Whole Program Path (WPP).

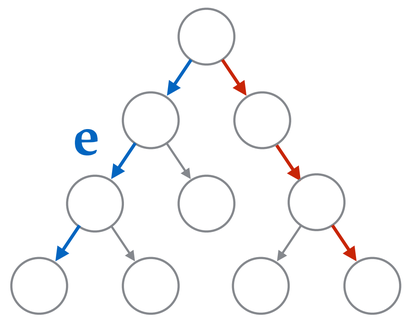
Motivation

- Existing approaches perform redundant instrumentations.
- Whole program structure can be utilized to reduce it.
- Results in lower runtime overhead.



4 paths are identified by the edges **5, 6, 9** and ***empty-log***.

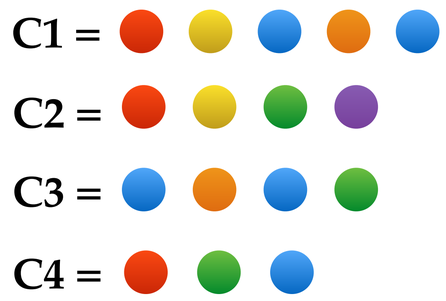
Overview



Conflict set, C =
{ **Blue edges** }
∪
{ **Red edges** }.

Instrumenting any **e** ∈ **C** can distinguish between **red** and **blue** path.

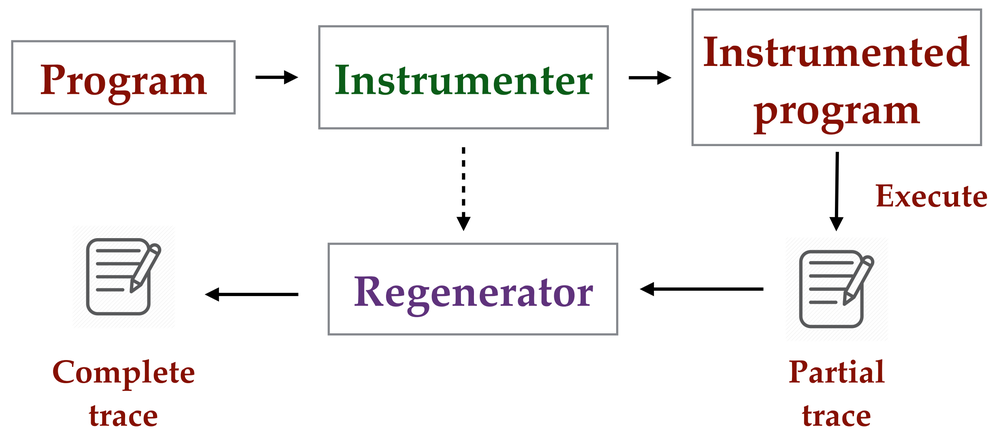
C1 = **C2 =** **C3 =** **C4 =**



Minimum hitting set is,
H =

- Conflict sets are created for the entire program.
- The **minimum hitting set** of the conflict sets is the set of instrumentation points.

Workflow



Instrumenter

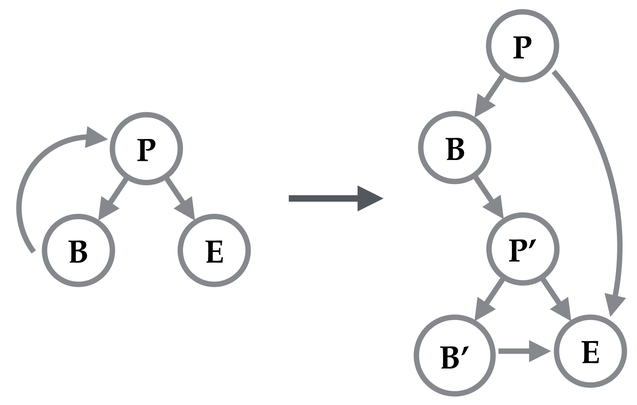
- Identify edges to instrument
- Instrumentation to emit edge identifier
- Save information for regeneration

Regenerator

- Generate complete trace using partial trace
- Use information from instrumenter for regeneration

Challenges

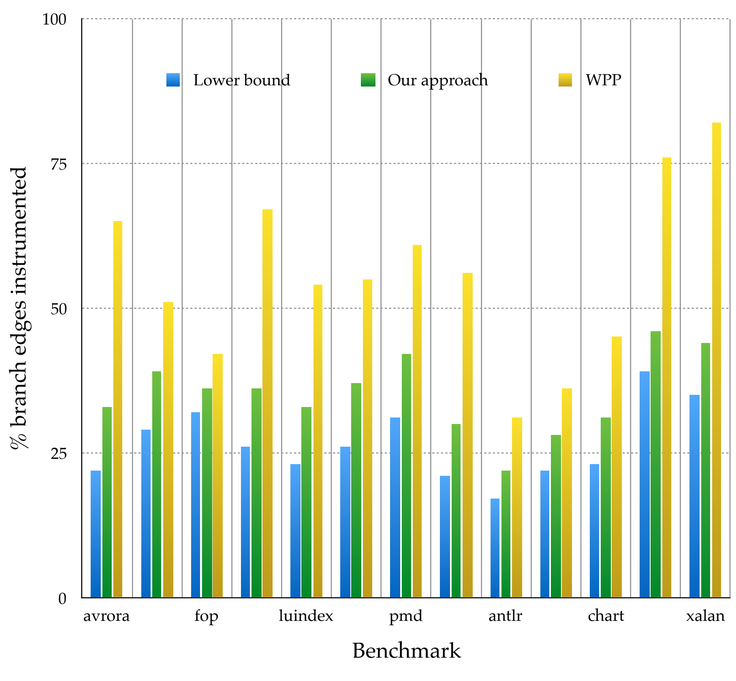
- ***Handling intersecting paths:*** We define merge points which are used to create conflict sets.
- ***Handling loops in CFGs:*** Loop transformation to convert cyclic CFG to acyclic CFG.
- ***Handling path enumeration:*** Approximations are used to reduce the number of conflict sets generated.



Experimental Results

- Instruments **9%** of total edges on average
- Incurs **97%** runtime overhead on average
  - Any optimal instrumentation approach incurs at least **71%** overhead
  - State of the art incurs **278%** overhead

Instrumentation points



Runtime overhead