Program Analysis
Path Profiling (Part 1)
Outline

1. Motivation and Challenges
2. Ball-Larus algorithm for DAGs
3. Generalization and Applications

Mostly based on this paper:

- *Efficient path profiling*, Ball and Larus, MICRO 1996

Other reading material:

- *Whole program paths*, Larus, PLDI 1999
- *HOLMES: Effective statistical debugging via efficient path profiling*, Chilimbi et al., ICSE 2009
Path Profiling

- **Goal:** Count how often a path through a function is executed

- **Interesting for various applications**
  - Profile-directed compiler optimizations
  - Performance tuning: Which paths are worth optimizing?
  - Test coverage: Which paths are not yet tested?
Challenges

■ Runtime **overhead**
  □ Limit slowdown of program

■ **Accuracy**
  □ Ideally: *Precise profiles* (no heuristics, no approximations)

■ **Infinitely many paths**
  □ Cycles in control flow graph
### Running Example

<table>
<thead>
<tr>
<th>#</th>
<th>Path</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ACDF</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>ACDEFE</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>AB CDF</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ABCDEF</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ABDF</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>ABDEF</td>
<td></td>
</tr>
</tbody>
</table>
Edge Profiling

Naive approach: Edge profiling

- Instrument each branching point
- Count how often each CFG edge is executed
- Estimate most frequent path: Always follow most frequent edge
Example: Edge Profiling

Frequency of execution

Most frequent path?

ACDE

Really? Two possible path profiles:

Path | Profile 1 | Profile 2
--- | --- | ---
ACDF | 90 | 110
ACDE | 60 | 40
ABCD | 0 | 0
ABCDEF | 100 | 100
ABDF | 20 | 0
ABDEF | 0 | 20
Edge Profiling

Naive approach: Edge profiling

■ Instrument each branching point
■ Count how often each CFG edge is executed
■ Estimate most frequent path: Always follow most frequent edge

Fails to uniquely identify most frequent path