Retargetable Infeasible Path Detection for WCET Analysis

Zach Hall, Kory Kraft, Thomas Mincher,
Joey Iannetta, Chris Healy

Department of Computer Science
Furman University

Problem / Goals

- Accurate, efficient WCET analysis
  - Input program usually features multiple paths
  - Longest paths might never be taken
  - Need to detect infeasible paths
- Perform analysis on assembly code
  - Instead of modifying compiler
  - Easier to retarget timing analysis tool
- Information required for WCET analysis
  - Control-flow, hierarchical structure
  - Loop iterations

Outline of Approach

- A new tool to assist WCET analysis: RALPHO
- Read assembly definition file
- Read assembly source
  - Identify functions, branch destinations, basic blocks
  - Create directed graph of basic blocks
  - Find loops, calculate # iterations
- Branch constraint analysis
- Compute paths and function instances
- Control-flow information passed to timing tool

Branch analysis

- For each pair of branches,
  - Does one influence the other?
  - 2 branches → 4 paths. Maybe some are infeasible.
  - See if they compare same register
  - Analyze the relational operator and constants
- For each basic block,
  - Does it update a register used in a comparison elsewhere?
  - If so, then branch behavior along a path may be determined

Ongoing Work

- Testing with Mälardalen benchmarks
- Consider comparisons between two registers
- Determine iteration constraints on paths
- Interprocedural analysis
  - Detecting more nested loops
- Combine with work presented at RTAS 2014: stochastic execution time