GenE: A Benchmark Generator for WCET Analysis

Peter Wägemann, Tobias Distler, Timo Hönig, Volkmar Sieh,
Wolfgang Schröder-Preikschat

System Software Group

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

15th International Workshop on Worst-Case Execution Time Analysis
(WCET '15)

July 7, 2015

SCHR603/9-1 (AORTA)
SCHR603/13-1 (PAX)
SFB/TRR 89 "Invasive Computing"

Motivating Example

```
void f(a, n) {
  if (a)
    g();
  else
    h();

  a = 1;

  for (i = n-1; i >= 1; i--) {
    for (j = 0; j < i; j++) {
      k();
    }
  }

  ++a;

  if (a % 2)
    f2();
}
```

- Input-dependent computation
- Nested loop
- Infeasible path

What is the WCET of function `f()`?

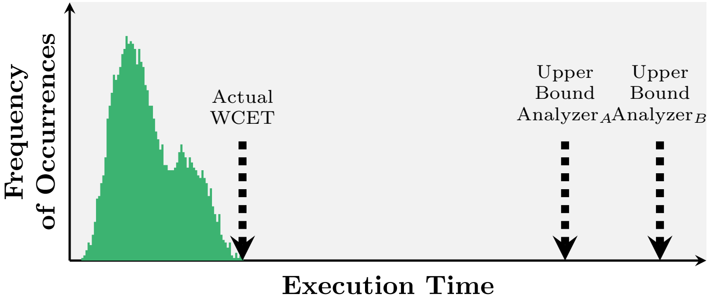
Motivation – Determining Flow Facts

What are the flow facts?

1. What are the **loop bounds**?
2. What are the **feasible paths**?
3. What are the **concrete input values** triggering the WCET?

- Having an **existing benchmark**
  - Difficult to extract all possible flow facts automatically
  - Explicit path enumeration not feasible for real-world applications
  - Manually determining flow facts labor-intensive and error-prone
- **Over-approximations** of actual WCET
  - Nested loop: $n * (n-1)/2 \mapsto n^2$
  - Include infeasible paths
  - ...

Motivation – Baseline



- WCET Tool Challenge: comparison of WCET analyzers
- Necessary to know flow facts for evaluation on **global scale**
- Create common **baseline** (actual WCET)

How to know all flow facts?

GENE: **generate a benchmark** of which the flow facts are known

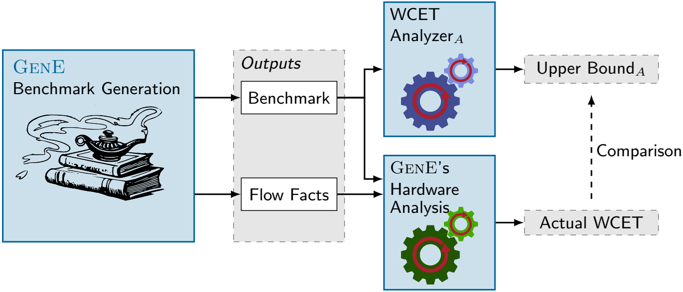
Agenda

1. Motivation

2. GenE
   2.1 Patterns and Pattern Selection
   2.2 Benchmark Weaving
   2.3 Cost Modeling & WCET Determination

3. Conclusion

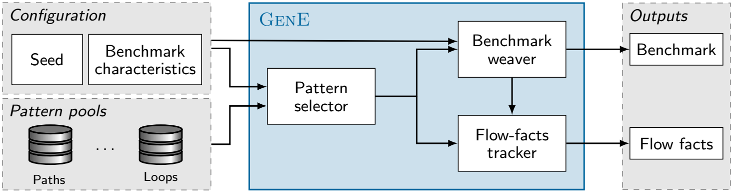
GenE – Application Scenario



- GenE's hardware analysis receives flow facts
- Comparison of **actual WCET** with upper bound of WCET analyzer
  - Evaluation: Extent of over-approximations?
  - Validation: WCET-analyzer result below actual WCET?
- **Incremental process** imaginable

1. Loop bound not found by Analyzer$_A$
2. GenE: provide annotation

GenE – Overview



- Pattern pools (e.g., paths, loops)
- GenE
  - Weaving patterns
  - Tracking flow facts
- Outputs
  1. Generated benchmark
  2. Flow facts of the benchmark

Patterns

```
for(i = n-1; i >= 1; i--){
  for(j = 0; j < i; j++){
    // insertion_point_1
  }
}
// insertion_point_2
```

Loop pattern

```
if( condition ){
  // insertion_point_1
} else {
  // insertion_point_2
}
// insertion_point_3
```

Path pattern

- Patterns are inserted into code
- New **insertion points** are created
- **Flow facts known** for patterns
- Patterns store (parametric) costs
- Patterns extracted from existing benchmark suites

```
void f() {
  static int inited = 0;
  if ( !inited ){
    init();
    inited = 1;
  }
  ...
```

Path pattern

Pattern Selection

```
switch (state) {
  case GROUND:
    ...
  case ENGINE_START:
    ...
}
```

State-machine pattern

```
for (i=1; i < m; i += 2) {
  for (j=i; j <= n; j += step) {
    temp = data[j-1] - data[j];
    ...
  }
}
```

Signal-processing pattern

- Assignment of **weights** to patterns
- Weights considered during pattern selection
- ☞ **Configurable benchmark properties**

Benchmark Weaving – Grammar

1. $\mathcal{S} \mapsto FunctionBegin \cdot inp \cdot FunctionEnd$
2. $inp \mapsto \varepsilon$
3. $inp \mapsto Statement \cdot inp$
4. $Statement \mapsto Assignment$
5. $Statement \mapsto IfBegin \cdot inp \cdot ElseBegin \cdot inp \cdot EndIf$
6. $Statement \mapsto LoopHead \cdot inp \cdot LoopEnd$

$\vdots$ *(Additional production rules)*

- **Limited grammar** to reduce complexity
- Grammar considered by weaving algorithm

Benchmark Weaving – Weaving Algorithm (1)

```
function GenE(vars, cost)
 if(cost == 0)
   return vars

 switch(select_production(cost))
   case Statement · inp:
      ...
   case Assignment:
      (new_vars, operation) ← select_assignment(vars)
      emit_Assignment(operation)
   case IfBegin · inp · ElseBegin · inp · EndIf:
      ...
   case LoopHead · inp · LoopEnd:
     ...
 return new_vars // return updated variables
```

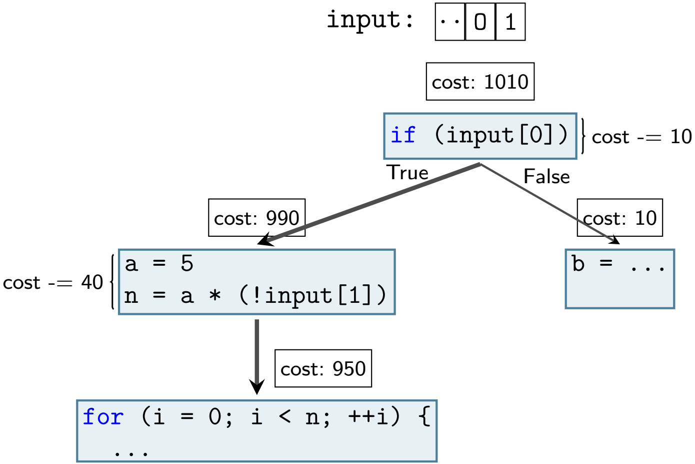
- Pattern selection based on distributable cost
- **Top-down** value tracking

Benchmark Weaving – Weaving Algorithm (2)

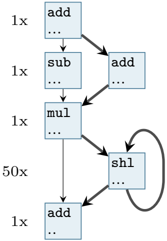
```
    ...
    case Assignment:
        ...
    case IfBegin · inp · ElseBegin · inp · EndIf:
        ...
        if_cost ← ...
        else_cost ← ...
        if_vars ← GenE(vars, if_cost)
        emit_ElseBegin()
        else_vars ← GenE(vars, else_cost)
        emit_EndIf()
        new_vars ← merge_variables(if_vars, else_vars)
    case ...
      ...
return new_vars // return updated variables
```

- **Top-down** generation and cost distribution
- Recursive application on nodes

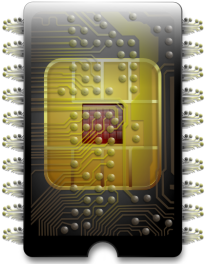
Benchmark Weaving – Example



Cost Modeling – Problem



&



- WCET analysis

1. High-level analysis: **program structure**
2. Low-level analysis: **target platform**

Flow facts must respect target platform!

Avoid generating target-dependent benchmarks

Cost Modeling – Solution

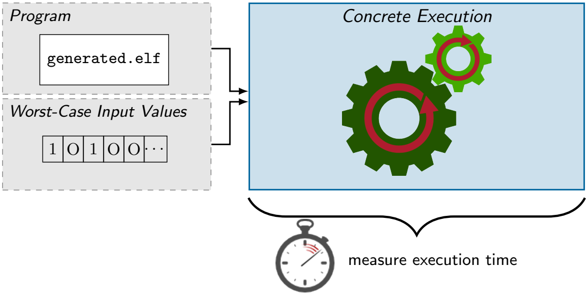
```
if( condition ){
  f(); // worst-case path      cost: 990
} else {
  g();                         cost: 10
}
```

- **Input-/path-oriented** approach
- **Relative** cost modeling
- **Overweighting** branches by large factors
- Refinement possible through target-specific knowledge
- `costof( f() )` $\ggg$ `costof( g() )`

Most important flow fact

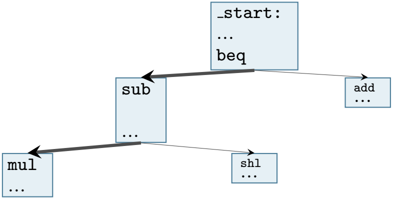
☞ **Input values** for triggering worst-case path

WCET Determination



- Determine WCET from worst-case inputs: **concrete execution**
  1. Target platform
  2. Cycle-accurate simulator
- ☞ Measured/simulated execution time yields **actual WCET**

Two Phases of Cost Modeling



- Two phases of modeling
  1. Overweighting of branches ⇝ input values
  2. Input values for measurement/simulation on hardware ⇝ actual WCET
- Hardware features (i.e., caching, pipelining) not explicitly modeled
- **Hardware implicitly modeled** through measurement/simulation

Agenda

1. Motivation
2. GenE
3. Conclusion

Conclusion

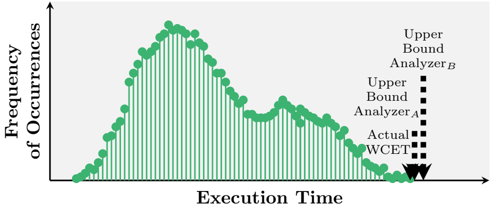
```c
void f(a, n) {
  if (a)
    g();
  else
    h();

  a = 1;

  for (i = n-1; i >= 1; i--){
    for (j = 0; j < i; j++){
      k();
    }
  }

  ++a;

  if (a % 2)
    f2();
}
```

- Problem
  - Comprehensive WCET evaluations: baseline necessary
  - Know all flow facts?
- Solution
  - GENErate flow facts!
  - **Recombining** patterns
  - Weaving benchmark top-down
  - **Relatively overweighting** branches
  - **Concrete execution** on target to determine WCET

Benchmarks with **known WCET** for WCET-analyzer evaluation ✓

Discussion

⇒



- Work in progress
- Flow-fact format?
- WTC'16?

Questions? Discussion!

Thanks for the attention!