WCET Squeezing: On-demand Feasibility Refinement for Proven Precise WCET-Bounds

Jens Knoop\textsuperscript{1}    Laura Kovács\textsuperscript{2}    Jakob Zvirchmayr\textsuperscript{1}

\textsuperscript{1}Vienna University of Technology
Institute of Computer Languages

\textsuperscript{2}Chalmers University of Technology
Computer Science Department
Good News

...state-of-the-art approaches for WCET-analysis are

- matured and powerful.
Good News

...state-of-the-art approaches for WCET-analysis are

▶ matured and powerful.

There is

▶ strong practical evidence for this claim!
...state-of-the-art approaches for WCET-analysis are

- matured and powerful.

There is

- strong practical evidence for this claim!

This is

- WCET-analysis tools based on these approaches weren’t used routinely by industry these days if the information they compute weren’t useful.
But

Do we have

- rigorous, formal evidence for that?
But

Do we have

- rigorous, formal evidence for that?

Sadly

- No, we don’t have.
Actually

...we can not have such rigorous or formal evidence.
Actually

...we can not have such rigorous or formal evidence.

Our own experience with our tools shows

- there are corner cases, where WCET analysis results are not as tight and precise as they could and should be!

Think e.g. of the findings at the past WCET Tool Challenges.
This is

...where **WCET Squeezing** enters the scene!
This is

...where WCET Squeezing enters the scene!

**WCET Squeezing**

- a fully automatic procedure
- applicable as a post-process of any IPET/ILP-based WCET analyzer
- tightens the time bound inferred by the analyzer, if it is not, and eventually proves it precise.
Outline of the Talk

Conveying the take-away message

- WCET Squeezing is a Fully Automatic Tightening and Proof Procedure for WCET Bounds.

Guiding Questions

- How does it work?
- Why does it work?
- Does it work for me?
- Are we done?
How does it work?
Why is there a precision gap at all?

Inferred WCET bounds

- must be safe
  - to be acceptable
- shall be tight
  - to be useful
Why is there a precision gap at all?

Inferred WCET bounds

- must be safe
  - to be acceptable
- shall be tight
  - to be useful

Safety

- ensured by over-approximating the set of program
  - spurious paths enlarge the gap
Why is there a precision gap at all?

Inferred WCET bounds

- must be safe
  - to be acceptable
- shall be tight
  - to be useful

Safety

- ensured by over-approximating the set of program
  ~⇒ spurious paths enlarge the gap

Tightness

- is strived for by taming the over-approximation of the set of paths of the program model and keeping it as small as possible ~⇒ shrinkens the gap
IPET/ILP-based WCET Analyzers

...strive for reconciling safety and tightness by

- encoding the WCET optimization problem as an Integer Linear Program (ILP) derived from an annotation of the nodes/edges of the control-flow graph of with flow constraints that
  - specify limits on the maximum number of their executions, and
  - implicitly exclude infeasible paths from the timing analysis.
In Practice

- low WCET over-estimation: acceptable
- high WCET over-estimation: insufficient
- user-/context-supplied threshold
- actual WCET

 executions

 execution time
Since...
...often some infeasible paths remain
WCET Squeezing

1. Preprocess: IPET/ILP-based WCET Analysis
   1.1 Perform WCET Analysis encoded as IPET/ILP problem
   1.2 Solve ILP problem

2. Postprocess: WCET Squeezing
   2.1 Calculate/extract path(s) from ILP solution
   2.2 Symbolically execute path(s)
   2.3 Use result of symbolic execution to refine the ILP problem or stop:
      2.3.1 If feasible: stop (path not spurious and indeed WC path)
      Proof obtained!
      2.3.2 If infeasible: refine and resolve ILP, goto 2 (squeezing!)
      Time bound tightened!
Possible Outcomes of WCET Squeezing

Symbolic execution yields that

1. IPET/ILP-delivered WC path is feasible:
   ▶ Precise information: Inferred WCET is safe and tight.

2. IPET/ILP-delivered refined WC path is feasible:
   ▶ Precise information: Inferred WCET is safe and tight.

   Mission accomplished: Proof obtained!

3. IPET/ILP-delivered refined WC paths are infeasible:
   ▶ More precise information. Inferred WCET is safe and tighter.

   Mission partially accomplished: Time Bound tightened!
Does this work?

Sounds naively, right?
Does it work?
Why does it work?
The r-TuBound WCET Analysis Tool Chain

- Motivation
- How?
- Why?
- For me?
- Done?
- Conclusions
The Symbolic Execution Engine SmacC

Motivation

How?

Why?

For me?

Done?

Conclusions
Experimental Data

...using the Mälardalen Benchmark Suite.

<table>
<thead>
<tr>
<th>BM</th>
<th>function</th>
<th>#iters</th>
<th>$#\pi$</th>
<th>%imp</th>
<th>%prec</th>
<th>note</th>
</tr>
</thead>
<tbody>
<tr>
<td>prime</td>
<td>prime</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>✓</td>
</tr>
<tr>
<td>compress</td>
<td>cl_block</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>✓</td>
</tr>
<tr>
<td>crc</td>
<td>icrc1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>✓</td>
</tr>
<tr>
<td>adpcm</td>
<td>logsch</td>
<td>1</td>
<td>1</td>
<td>3.24</td>
<td>3.24</td>
<td>✓ T</td>
</tr>
<tr>
<td></td>
<td>uppol2</td>
<td>2</td>
<td>2</td>
<td>9.87</td>
<td>9.87</td>
<td>✓ T</td>
</tr>
<tr>
<td>duff</td>
<td>duffcopy</td>
<td>5</td>
<td>5</td>
<td>7.5</td>
<td>7.5</td>
<td>✓ T</td>
</tr>
<tr>
<td>fibcall</td>
<td>fib</td>
<td>2</td>
<td>2</td>
<td>6.39</td>
<td>6.39</td>
<td>✓ T</td>
</tr>
<tr>
<td>expint</td>
<td>expint</td>
<td>1</td>
<td>1</td>
<td>0.94</td>
<td>93</td>
<td>T</td>
</tr>
<tr>
<td>janne-cmp</td>
<td>complex</td>
<td>1</td>
<td>1</td>
<td>5.9</td>
<td>93.3</td>
<td>T T</td>
</tr>
<tr>
<td>lcdnum</td>
<td>main</td>
<td>2</td>
<td>11</td>
<td>7.4</td>
<td>18.5</td>
<td>T T</td>
</tr>
<tr>
<td>nsichneu</td>
<td>main</td>
<td>2</td>
<td>2</td>
<td>0.28</td>
<td>-</td>
<td>T*</td>
</tr>
</tbody>
</table>

✓: precision proved, T: bound tightened
Experimental Findings

WCET Squeezing

► works!

For some benchmarks we obtained WCET tightenings of up to 90% after a few iterations.
Experimental Findings

**WCET Squeezing**

- works!

For some benchmarks we obtained WCET tightenings of up to 90% after a few iterations.

**WCET Squeezing**

- works because state-of-the-art WCET analyzers are strong!

...they just can not provide rigorous evidence and proof of that!
Does it work for me?
Good News: Yes, it does!

...the only thing we need is interoperability of WCET Tool Chains

- since WCET Squeezing is applicable as a fully automatic postprocess of any high-level IPET/ILP-WCET analyzer.
Interoperability can be achieved using a common flow facts annotation language for coupling tool chains.

One such Proposal: FFX (Flow Facts in XML)

...an open XML-based format for WCET annotations that is

▶ portable
▶ expandable
▶ easy to write, understand, and process

...joint work Armelle Bonenfant, Hugues Cassé, and Marianne de Michiel (Université de Toulouse).
FFX-Interoperability of WCET Tool Chains

...using the well-known UNCOL concept:

```
| PL 1 | | PL 2 | ... | PL n |
|------| |------|     |------|
|      | |      |     |      |
|------| |------|     |------|
|      | |      |     |      |
|------| |------|     |------|

\[/\]

\[/\]

\[/\]

| UNCOL | Computer Oriented Language
|-------|---------------------------

-------------------------------------------------------------------------------
| ML 1  | | ML 2  | ... | ML m  |
|-------| |-------|     |-------|

\[/\]

\[/\]

\[/\]

\[/\]

Are we done?
More dramatically

...will RTNS 2013 be the last and final edition of the successful and friendly RTNS conference series?
More dramatically

...will RTNS 2013 be the last and final edition of the successful and friendly RTNS conference series?

No worries!

...good news again, it won’t be.
Why?

**WCET Squeezing** works because of

- smartly combining the strengths of
  - IPET/ILP-based WCET-analysis: considers all program executions at the same time
  - **Symbolic execution**: yields precise information on the feasibility of a program path while avoiding their weaknesses
  - **SE**: does not scale to an exponentially growing number of program paths
  - **IPET/ILP**: does not take advantage of information about a program beyond flow facts

- state-of-the-art IPET/ILP-based WCET-analyzers are strong!
Nonetheless

WCET Squeezing

- can be costly requiring a lot of iterations!
Nonetheless

**WCET Squeezing**

- can be costly requiring a lot of iterations!

This, however, should not be considered reason to

- blame **WCET Squeezing** for.
Nonetheless

**WCET Squeezing**

- can be costly requiring a lot of iterations!

This, however, should not be considered reason to

- blame **WCET Squeezing** for.

Actually, this should be considered indication to look into the

- IPET/ILP-based WCET-analyzer **why it fails** to do a better job on that particular program
- and **to improve it**!
In fact

WCET Squeezing applied at its best

- does not squeeze at all
- but is simply left with verifying the feasibility of the WCET path candidate inferred by the WCET analyzer!
Moreover

**WCET Squeezing** is an “anytime” algorithm
- ...opening **new application horizons** to WCET-analysis.
Moreover

**WCET Squeezing is an “anytime” algorithm**
- ...opening new application horizons to WCET-analysis.

From the classical WCET Problem...

- **Precision-controlled (classical):**
  - Minimize the gap between the actual and the inferred WCET (Note: WCET Squeezing even closes this gap!)
Moreover

**WCET Squeezing** is an “anytime” algorithm

- ...opening new application horizons to WCET-analysis.

From the classical WCET Problem...

- **Precision-controlled (classical):**
  - Minimize the gap between the actual and the inferred WCET (Note: WCET Squeezing even closes this gap!)

...to Pragmatic WCET Problem Variants:

- **Threshold-controlled:**
  - Infer a **WCET** bound that is below the **threshold value**, if possible.
Moreover

WCET Squeezing is an “anytime” algorithm
  ▶ ...opening new application horizons to WCET-analysis.

From the classical WCET Problem...

  ▶ Precision-controlled (classical):
    ▶ Minimize the gap between the actual and the inferred WCET (Note: WCET Squeezing even closes this gap!)

...to Pragmatic WCET Problem Variants:

  ▶ Threshold-controlled:
    ▶ Infer a WCET bound that is below the threshold value, if possible.
  ▶ Cost-controlled:
    ▶ Infer an as tight as possible WCET bound within a given analysis time budget.
Illustration
Conclusions

WCET Squeezing is

- a fully automatic procedure able of tightening and proving inferred WCET bounds precise.

WCET Squeezing thus

- resolves a major short-coming of state-of-the-art WCET analysis.
Related Own Work

- RTNS 2013: WCET Squeezing
- WCET 2013: Symbolic analysis in timing analysis
- ATVA 2013: SmacC, the symbolic analyzer
- RTNS 2012: ffx, interoperability of WCET tool chains
void main ()
    int i;
    bool exec = false;
    if (*)
        exec = true;    // t1
    for (i = 0; i < 5; i++)
        if (exec == false)
            expensive();    // t2
        exec = false;
    else
        exec = true;    // f2
/* assumption:
   cost(t1)=1, cost(t2)=10, cost(other)=0 */
Initial WCET Analysis of the Example

- IPET maximizes costs
  - selects each true block
    (in the absence of “sophisticated” high-level program analysis)

- Executing t1 restricts the execution of t2
  - * denotes nondeterministic choice

- BUT IPET will “choose” to execute t1
  - AND to execute t2 in every iteration of the loop
    (because this maximizes the costs)
Initial ILP for the Running Example

- **Input to WCET Squeezing, (initial) ILP problem:**

  
  \[
  \begin{align*}
  n &\leq 1; \quad // \text{enter once} \\
  c_1 &\leq n; \quad // \text{what goes in must come out} \\
  t_1 + f_1 &\leq c_1; \quad // \text{same, for conditional} \\
  \text{loopHead} &\leq 1; \quad // \text{simplified for loop} \\
  \text{loopBody} &\leq \text{loopHead} \times 5; \\
  \text{loopBody} &\leq t_2 + f_2; \\
  \text{loopExit} &\leq \text{loopHead}; \\
  x &\leq \text{loopExit};
  \end{align*}
  \]

- **Its solution yields execution frequencies for} t_1, t_2, f_2 \text{ that imply a WCET-bound of 51:}

  
  \[
  \begin{align*}
  \text{freq}(t_1) &= 1, \quad \text{freq}(t_2) = 5, \\
  \text{freq}(f_2) &= 0 \\
  \text{i.e.} \quad 1 \times 1 + 5 \times 10 &= 51
  \end{align*}
  \]
Extracting the Abstract Candidate

- **Requires:** source, ILP solution (+mapping solution-source)

- **ILP solution:** freq(t1) = 1
  - implies: condition1 evaluates to true
    (otherwise freq(f2) would be 1)

- **ILP solution:** freq(t2) = 5, freq(f2) = 0
  - implies: condition2 evaluates to true 5 times
    (otherwise freq(f2) would be > 0)

- **Abstract candidate specified as “branching behaviour”**
  - i.e. evaluation of the conditions implied by the solution

- **Here:** t(ttttt) is implied
  - (omitting evaluation of the loop condition)
  - for easier readability “( )” marks decisions inside loop
Computing Concrete Candidates

- A single abstract cand might yield more than one Concrete cand
  - when the freq of the edges is not all-and-nothing
- i.e. if both $freq(t) > 0$ and $freq(f) > 0$
  - then the abstract cand implies $p$ concrete candidates
- where $p = \frac{(freq(t_{Ci}) + freq(f_{Ci}))!}{(freq(t_{Ci})!) \times (freq(f_{Ci})!)}$ concrete candidates.
  - (i.e. the number of permutations of $f$ and $t$)
- these concrete candidates need to be checked for feasibility
  - if one of them is feasible, the ILP computed the WCET bound for a concrete execution
- Example: $t(ttttt)$ implies a single concrete candidate: $t(ttttt)$
  - this candidate is infeasible by symbolic execution
Computing Concrete Candidates (Cont’d)

- $\Rightarrow$ Exclude the infeasible trace/candidate
  - s.t. it is not considered in further WCET bound computations
  - i.e. “the model of WCET computation” $=$ the ILP
Encoding the Counterexample

- **Variant 1:** explicitly exclude trace ("syntactic")
  - by making decisions explicit and removing the trace
  - i.e. a graph or program transformation

- **Variant 2:** implicitly exclude trace ("semantic")
  - by restricting execution frequencies
  - i.e. a logical restriction (formula)
Pros and Cons of the Encoding Variants

- **Variant 1** increases the number of ILP variables
  - but makes branching decisions explicit (less permutations)
  - i.e. less concrete candidates from abstract candidate

- **Variant 2** increases the number of candidates
  - implicit branching decisions (more permutations)
  - i.e. more concrete candidates from abstract candidate

- **Variant 3**: combine variants 1 and 2 ("hybrid")
  - make some branching explicit
  - introduce some new ILP variables

- **Heuristics could be** (not yet implemented):
  - ILP solving fast ⇒ use syntactic encoding
  - Symbolic executions fast ⇒ use semantic encoding
Running Example: Pure Constraints

- Evaluation of conditional to `t` in first iteration infeasible (`t2`)
  - (after executing the true-branch of the conditional before the loop (`t1`))

- Purely syntactic:
  - duplicate loop and move it into conditional branches
  - make explicit first iteration
  - remove infeasible edge in peel `t1`
  - “blows up” ILP

- Purely semantic:
  - restrict (decrement) combined execution frequency of `t1` and `t2`
  - “blows up” symbolic execution
Running Example: Hybrid Constraint

(peels’ loop cond not drawn)

- peel loop (syntactic), restrict execution frequency (semantic)
  - peel-off first loop iteration (make first iteration explicit)
  - restrict (decrement) combined execution frequency of t1 and t2

- Thus: \( t1 + t1_P < 2 \)
  - coming to peeled iteration via t1 leaves “no more executions” for t2_P
  - coming to peeled iteration via f1 allows executing t2_P

- “blow up” for combined constraint:
  - introduces vars for peel (but does not duplicate the whole loop)
  - could introduce additional symbolic executions (but not over all loop iterations)
Example: Terminating Iteration of Squeezing

- New ILP vars for peel: \( t_1 \_P(eel) \), \( t_2 \_P(eel) \), \( \text{cond} \_P(eel) \)
  - and ILP constraints modeling “flow” for the peel

- New constraint: \( t_1 + t_2 \_P < 2 \)
  - decrements combined execution frequency (before: < 3)

- Re-solve: \( \text{freq}(t_1) = 0 \), \( \text{freq}(f_1) = 1 \), \( \text{freq}(t_1 \_P) = 1 \), \( \text{freq}(f_1 \_P) = 0 \), \( \text{freq}(t_2) = 4 \), \( \text{freq}(f_2) = 0 \)
  - i.e. \( 1 \times 0 + 1 \times 10 + 4 \times 10 = 50 \)
Discussion & Pragmatics

- WCET bound for an actual execution of the program
  “perfect” feasible path analysis

- Pragmatics: timeout, threshold
  “until good enough”, “can we get below”

⇒ Pragmatic WCET Problem vs. classic WCET problem
Evaluation of the Proof-Procedure

<table>
<thead>
<tr>
<th>BM</th>
<th>function</th>
<th>#iters</th>
<th>#π</th>
<th>%imp</th>
<th>%prec</th>
<th>note</th>
</tr>
</thead>
<tbody>
<tr>
<td>prime</td>
<td>prime</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>✓</td>
</tr>
<tr>
<td>compress</td>
<td>cl_block</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>✓</td>
</tr>
<tr>
<td>crc</td>
<td>icrc1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>✓</td>
</tr>
<tr>
<td>adpcm</td>
<td>logsch</td>
<td>1</td>
<td>1</td>
<td>3.24</td>
<td>3.24</td>
<td>✓ T</td>
</tr>
<tr>
<td></td>
<td>uppol2</td>
<td>2</td>
<td>2</td>
<td>9.87</td>
<td>9.87</td>
<td>✓ T</td>
</tr>
<tr>
<td>duff</td>
<td>duffcopy</td>
<td>5</td>
<td>5</td>
<td>7.5</td>
<td>7.5</td>
<td>✓ T</td>
</tr>
<tr>
<td>fibcall</td>
<td>fib</td>
<td>2</td>
<td>2</td>
<td>6.39</td>
<td>6.39</td>
<td>✓ T</td>
</tr>
<tr>
<td>expint</td>
<td>expint</td>
<td>1</td>
<td>1</td>
<td>0.94</td>
<td>93</td>
<td>T</td>
</tr>
<tr>
<td>janne-cmp</td>
<td>complex</td>
<td>1</td>
<td>1</td>
<td>5.9</td>
<td>93.3</td>
<td>T</td>
</tr>
<tr>
<td>lcdnum</td>
<td>main</td>
<td>2</td>
<td>11</td>
<td>7.4</td>
<td>18.5</td>
<td>T</td>
</tr>
<tr>
<td>nsichneu</td>
<td>main</td>
<td>2</td>
<td>2</td>
<td>0.28</td>
<td>-</td>
<td>T*</td>
</tr>
</tbody>
</table>

- WCET Squeezing works!

✓: precision proved, T: bound tightened
Future Work

- Test-case generation for WCET path
- Frequency counting
- Full WCET analysis after WCET Squeezing
  - (of ALL feasible concrete candidates)
- WCET Squeezing on binary