From Trusted Annotations to Verified Knowledge

or

Squeeze your Trusted Annotation Base!

Albrecht Kadlec, Raimund Kirner,
Adrian Prantl, Jens Knoop and Markus Schordan

Vienna University of Technology, Austria
Fact: Compile-Time Analysis is Limited

- Compiler Optimization
  - Less severely hit
    * strikes power/optimality, not usefulness
    * **but** opens the avenue to take advantage of the trade-off between precision and performance

- Worst-case Execution Time Analysis (WCET)
  - Severely hit
    * strikes **tightness**, not just power/optimality:
      No loop bound, no WCET bound!
    - unclear, how to take advantage of the trade-off between precision and performance

  **hence**: rest on **user-assistance**!
Motivation

Resting on user-assistance, however, means introducing a...

**Trusted Annotation Base (TAB)**

into WCET Analysis

Unintended but unavoidable consequences...

- **Soundness**: Computed time bounds hold only up to the correctness of the TAB - are we still *safe*?

- **Optimality**: Same for tightness
Replacing Trust by Proof

- From Trust to Proof
  - Squeeze the TAB: Compress it!
    - Soundness: Replacing trust by proof
      - Getting rid of user-assistance
    - Optimality: Sharpening the time bounds

- From Proof to Power
  - Squeeze the TAB/VAB: Wring it!
    - Taking advantage of the TAB/VAB
    - Taking advantage of the trade-off between power and performance of analysis algorithms
Replacing Trust by Proof – Model-checking

- If something can’t be bounded automatically, ask for user-assistance
- Apply model-checking to prove/disprove the user-provided bound

... controlled and guided by

- **Binary Tightening**
  sharpen the current bound

- **Binary Widening**
  find start value if the user annotation is missing or faulty
The TuBound Tool

So far, TuBound featured

- Interval analysis
- Loop bound analysis
- Constraint analysis

...to reduce the need for user-provided annotations to a minimum.

Recently, TuBound has been extended to feature

- Model-checking
  - BLAST
  - CBMC (turned out to be superior for our setting!)
Taking Advantage of TAB/VAB and Trade-Off...

...using a pool of complementary analysis techniques of different power and computational complexity.
## Experimental Results (1)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Loops</th>
<th>TuBound basic</th>
<th>with Model Checking</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>recursion</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>bsort100</td>
<td>3</td>
<td>100%</td>
<td>100%</td>
<td>–</td>
</tr>
<tr>
<td>cnt</td>
<td>4</td>
<td>100%</td>
<td>100%</td>
<td>–</td>
</tr>
<tr>
<td>cover</td>
<td>3</td>
<td>100%</td>
<td>100%</td>
<td>–</td>
</tr>
<tr>
<td>crc</td>
<td>3</td>
<td>100%</td>
<td>100%</td>
<td>–</td>
</tr>
<tr>
<td>edn</td>
<td>12</td>
<td>100%</td>
<td>100%</td>
<td>–</td>
</tr>
<tr>
<td>expint</td>
<td>3</td>
<td>100%</td>
<td>100%</td>
<td>–</td>
</tr>
<tr>
<td>fdct</td>
<td>2</td>
<td>100%</td>
<td>100%</td>
<td>–</td>
</tr>
<tr>
<td>fibcall</td>
<td>1</td>
<td>100%</td>
<td>100%</td>
<td>–</td>
</tr>
<tr>
<td>jfdctint</td>
<td>3</td>
<td>100%</td>
<td>100%</td>
<td>–</td>
</tr>
<tr>
<td>lcdnum</td>
<td>1</td>
<td>100%</td>
<td>100%</td>
<td>–</td>
</tr>
<tr>
<td>ludcmp</td>
<td>11</td>
<td>100%</td>
<td>100%</td>
<td>–</td>
</tr>
<tr>
<td>matmult</td>
<td>5</td>
<td>100%</td>
<td>100%</td>
<td>–</td>
</tr>
<tr>
<td>ndes</td>
<td>12</td>
<td>100%</td>
<td>100%</td>
<td>–</td>
</tr>
<tr>
<td>ns</td>
<td>4</td>
<td>100%</td>
<td>100%</td>
<td>–</td>
</tr>
<tr>
<td>qurt</td>
<td>1</td>
<td>100%</td>
<td>100%</td>
<td>–</td>
</tr>
<tr>
<td>sqrt</td>
<td>1</td>
<td>100%</td>
<td>100%</td>
<td>–</td>
</tr>
<tr>
<td>st</td>
<td>5</td>
<td>100%</td>
<td>100%</td>
<td>–</td>
</tr>
</tbody>
</table>
## Experimental Results (2)

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Loops</th>
<th>TuBound basic</th>
<th>with Model Checking</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>qsort-exam</td>
<td>6</td>
<td>0%</td>
<td>66.6%</td>
<td>0.02s</td>
</tr>
<tr>
<td>bs</td>
<td>1</td>
<td>0%</td>
<td>100%</td>
<td>0.03s%</td>
</tr>
<tr>
<td>nsichneu</td>
<td>1</td>
<td>0%</td>
<td>100%</td>
<td>5.59s</td>
</tr>
<tr>
<td>statemate</td>
<td>1</td>
<td>0%</td>
<td>100%</td>
<td>0.06s</td>
</tr>
<tr>
<td>janne_complex</td>
<td>2</td>
<td>0%</td>
<td>100%</td>
<td>0.18s</td>
</tr>
<tr>
<td>minver</td>
<td>17</td>
<td>94.1%</td>
<td>100%</td>
<td>0.06s</td>
</tr>
<tr>
<td>fft1</td>
<td>11</td>
<td>54.5%</td>
<td>81.8%</td>
<td>0.43s</td>
</tr>
<tr>
<td>duff</td>
<td>2</td>
<td>50%</td>
<td>50%</td>
<td>0s</td>
</tr>
<tr>
<td>whet</td>
<td>11</td>
<td>90.9%</td>
<td>90.9%</td>
<td>–</td>
</tr>
<tr>
<td>adpcm</td>
<td>18</td>
<td>83.3%</td>
<td>83.3%</td>
<td>timeout</td>
</tr>
<tr>
<td>compress</td>
<td>8</td>
<td>25.0%</td>
<td>25.0%</td>
<td>timeout</td>
</tr>
<tr>
<td>fir</td>
<td>2</td>
<td>50%</td>
<td>50%</td>
<td>timeout</td>
</tr>
<tr>
<td>lms</td>
<td>10</td>
<td>60%</td>
<td>60%</td>
<td>timeout</td>
</tr>
<tr>
<td>insertsort</td>
<td>2</td>
<td>0%</td>
<td>0%</td>
<td>timeout</td>
</tr>
<tr>
<td>select</td>
<td>4</td>
<td>0%</td>
<td>0%</td>
<td>timeout</td>
</tr>
<tr>
<td><strong>Total Percentage</strong></td>
<td><strong>77.0%</strong></td>
<td><strong>84.7%</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Conclusions

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Soundness</th>
<th>WCET Anal.</th>
<th>Characterization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class. Automatic PAs</td>
<td>Sound</td>
<td>May Fail</td>
<td>Insufficient</td>
</tr>
<tr>
<td>w/out user-assistance</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class. Automatic PAs</td>
<td>Sound rel.</td>
<td>Always</td>
<td>State-of-the-Art</td>
</tr>
<tr>
<td>w/ user-annotations</td>
<td>to TAB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Class. Automatic PAs</td>
<td>Sound rel.</td>
<td>Always</td>
<td>Contribution</td>
</tr>
<tr>
<td>w/ user-ann. checking</td>
<td>to red. TAB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Soph. Automatic PAs</td>
<td>Sound</td>
<td>Always</td>
<td>The Holy Grail</td>
</tr>
<tr>
<td>w/out user-assistance</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Thank you!

Questions?

This work has been supported by the 7th EU R&D Framework Programme under contract No 215068, “Integrating European Timing Analysis Technology (ALL-TIMES)”, and by the Austrian Science Fund (Fonds zur Förderung der wissenschaftlichen Forschung) under contract P18925-N13, “Compiler-Support for Timing Analysis (CoSTA)”. 
Experience: The TuBound Tool

Architecture overview...
Verifying Loop Bounds using CBMC

```c
int binary_search(int x) {
    int fvalue, mid, low = 0, up = 14;
    fvalue = (-1); /* all data are positive */
    { => unsigned int __bound = 0;
        while(low <= up){=>
            mid = low + up >> 1;
            if (data[mid].key == x) { /* found */
                up = low - 1;
                fvalue = data[mid].value;
            } => else if (data[mid].key > x)
                up = mid - 1;
            else low = mid + 1;
            => _bound += 1;
        }
    => assert(_bound <= 7);
    => return fvalue;
}
```
int complex(int a, int b) {
    while (a < 30) {
        #pragma wcet_trusted_loopbound(16) =>
        while (b < a) {
            #pragma wcet_trusted_loopbound(16) ==
            if (b > 5) {
                b = b * 3;
            } else {
                ++__bound;
                b = b + 2;
            }
            if (b >= 10 && b <= 12) {
                a = a + 10;
            } else {
                a = a + 1;
            }
            a = a + 2;
            b = b - 10;
        }
    }
    return 1;
}
...

int complex(int a, int b) {
    while (a < 30) {
        #pragma wcet_loopbound(16)
        {
            unsigned int __bound = 0U;
            while (b < a) {
                ++__bound;
                if (b > 5) {
                    b = b * 3;
                } else {
                    b = b + 2;
                }
                if (b >= 10 && b <= 12) {
                    a = a + 10;
                } else {
                    a = a + 1;
                }
                a = a + 2;
                b = b - 10;
            }
        }
    }
    return 1;
}
...