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

TECHNISCHE
UNIVERSITÄT
WIEN

VIENNA
UNIVERSITY OF
TECHNOLOGY

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

    $\leadsto$ **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

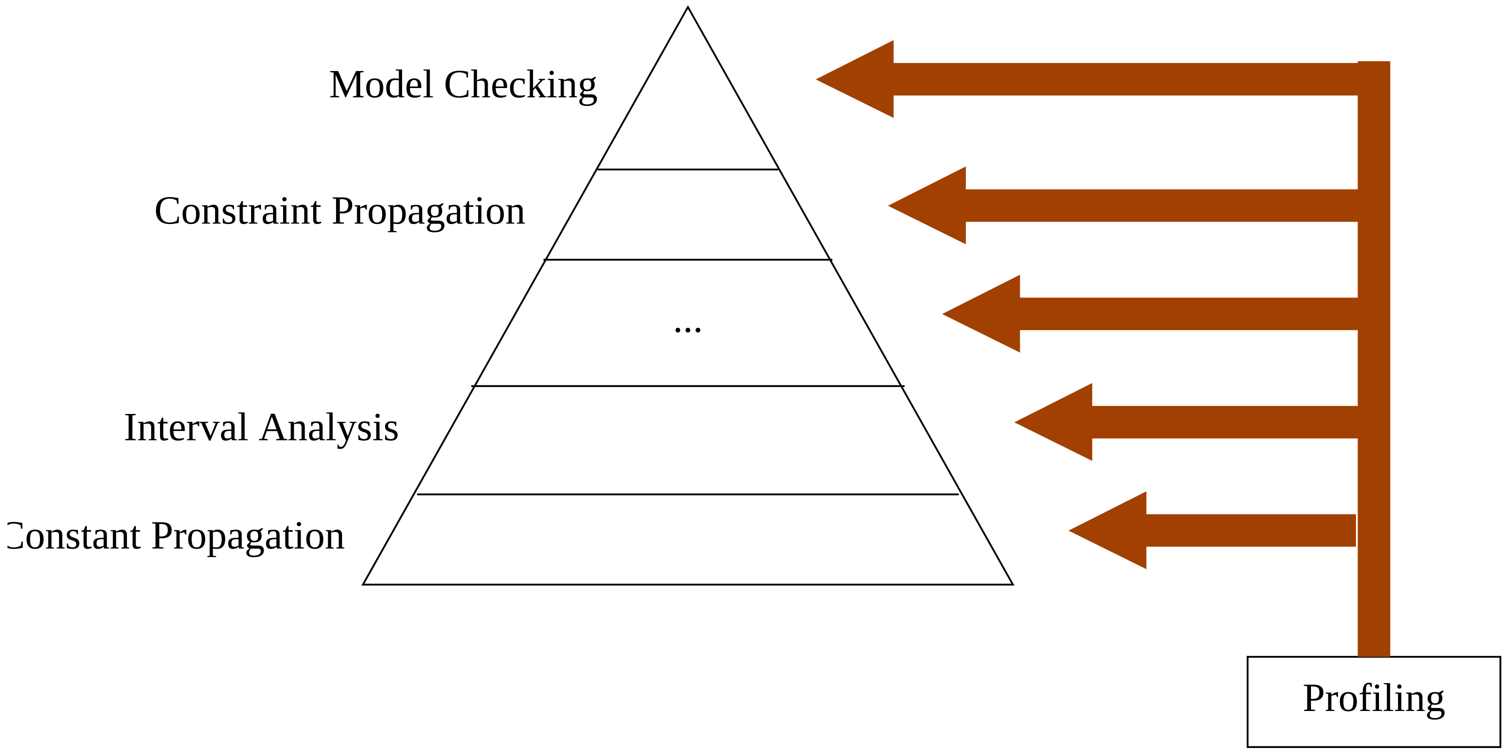
- Interval analysis
- Loop bound analysis
- Constraint analysis

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)

Benchmark	Loops	TuBound basic	with Model Checking	Runtime
recursion	0	–	–	–
bsort100	3	100%	100%	–
cnt	4	100%	100%	–
cover	3	100%	100%	–
crc	3	100%	100%	–
edn	12	100%	100%	–
expint	3	100%	100%	–
fdct	2	100%	100%	–
fibcall	1	100%	100%	–
jfdctint	3	100%	100%	–
lcdnum	1	100%	100%	–
ludcmp	11	100%	100%	–
matmult	5	100%	100%	–
ndes	12	100%	100%	–
ns	4	100%	100%	–
qurt	1	100%	100%	–
sqrt	1	100%	100%	–
st	5	100%	100%	–

Experimental Results (2)

Benchmark	Loops	TuBound basic	with Model Checking	Runtime
qsort-exam	6	0%	66.6%	$0.02s$
bs	1	0%	100%	$0.03s\%$
nsichneu	1	0%	100%	$5.59s$
statemate	1	0%	100%	$0.06s$
janne_complex	2	0%	100%	$0.18s$
minver	17	94.1%	100%	$0.06s$
fft1	11	54.5%	81.8%	$0.43s$
duff	2	50%	50%	$0s$
whet	11	90.9%	90.9%	$-$
adpcm	18	83.3%	83.3%	timeout
compress	8	25.0%	25.0%	timeout
fir	2	50%	50%	timeout
lms	10	60%	60%	timeout
insertsort	2	0%	0%	timeout
select	4	0%	0%	timeout
Total Percentage		77.0%	84.7%	

Conclusions

Configuration	Soundness	WCET Anal. Succeeds	Characterization
Class. Automatic PAs w/out user-assistance	Sound	May Fail	Insufficient
Class. Automatic PAs w/ user-annotations	Sound rel. to TAB	Always	State-of-the-Art
Class. Automatic PAs w/ user-ann. checking	Sound rel. to red. TAB	Always	Contribution
Soph. Automatic PAs w/out user-assistance	Sound	Always	**The Holy Grail**

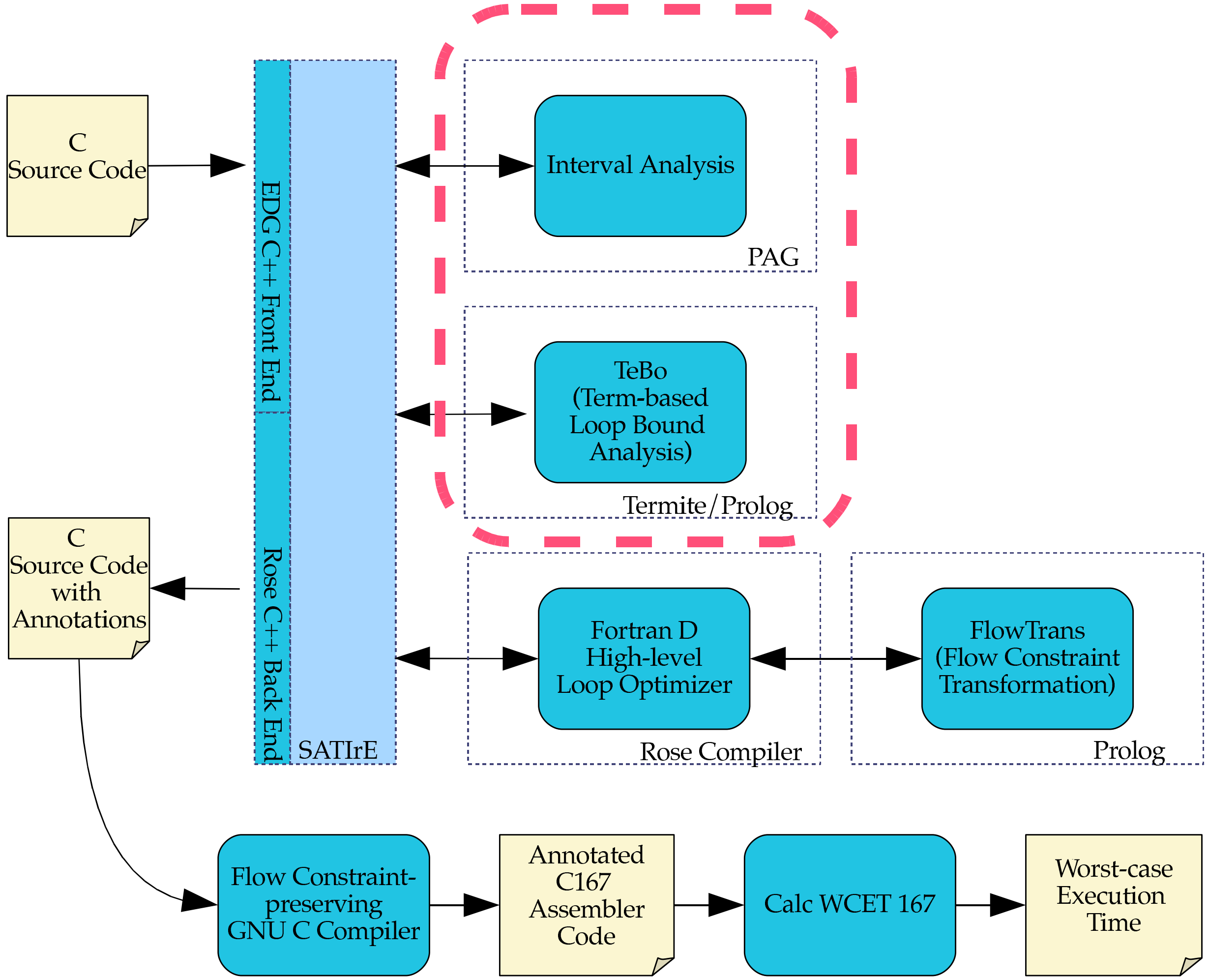
Thank you!

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

```
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;
}
```

CBMC at Work: Replacing Trust by Proof

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