Towards a Common WCET Annotation Language: Essential Ingredients

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

**TECHNISCHE
UNIVERSITÄT
WIEN**

TU
VIENNA

**VIENNA
UNIVERSITY OF
TECHNOLOGY**

Institute of Computer Engineering,
Institute of Computer Languages,
TU Vienna

Why a Common Annotation Language?

Specifying the problem-inherent information
- Tool-independent
- Methodology-indepent

Supporting different kinds of analysis
- Static analysis
- Measurement-based methods

Expressive enough to support annotation
- At source code level
- At object code level

➡ Enable inter-operation of tools and increase productivity.

Our Treasure Hunt

Identification of the Essential Ingredients

- Investigated timing-annotation constructs in existing languages and tools.
- Identified the essential ingredients of each timing-annotation and categorized the found ingredients.
- Described the ingredients conceptually to allow instantiation for different representation levels and tools.

Essential Ingredients

Annotation Categorization [L,C,G]

- Layer, class, group.

Addressable Units [U]

- Control-flow addressable units, loop contexts, call contexts, values of input variables, explicit enumeration of infeasible paths, goto statement.

Annotation Information [I]

- Software-Specific Annotation Information
  - Loop bounds, linear flow constraints, variable restrictions, summaries of external functions, unreachable code
  - Predicate evaluation, control-flow reconstruction.
- Hardware-Specific Annotation Information
  - Memory and memory accesses, absolute time bounds.

Layer [L] and Class [C] (Timing Invariants + Fictions)

a) Layered System View

b) System Behavior Described by Telescoping Timing Constraints
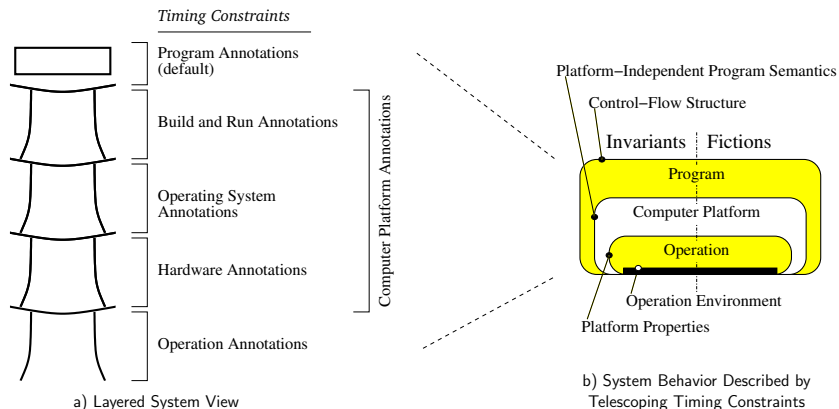
Every timing constraint is associated with one or more annotation layers

Class [C]

Timing Invariant

A timing constraint is a *timing invariant* at its associated annotation layer *L*, iff for all possible systems that use annotation layer *L*, it holds that for all possible initial system states the system execution fulfills the timing constraint *C*.

Timing Fiction

A timing constraint is a *timing fiction* if it allows to query alternative program behavior. ($\rightarrow$ it must be marked as such)

Conflict

In the case that *timing invariants* and *timing fictions* are in conflict, the semantics of *timing fictions* is to override conflicting *timing invariants*.

Example

```
1  void f(int a, char[] b)
2  {
3    int i;
4    a = a % 20;
5    for (i=0; i<a; i++)
6    {
7      //loop1
8      if (i%2 == 0)
9        b[i] = a;  //m1
10     else
11       b[i] = 0;  //m2
12   }
13 }
```

Timing Invariant:
Expressing as linear flow
constraint that the then-path
is executed at least as often
as the else-path: $m1 \geq m2$

Timing Fiction:
Specifying a lower and upper
loop bound of 40:
$LB(loop1) = 40 \ldots 40$

Putting It All Together

> **How to use the ingredients**
>
> Annotation $= (L, C, G, U, I)$

L ... Layer (e.g. program layer, operation layer, ...)
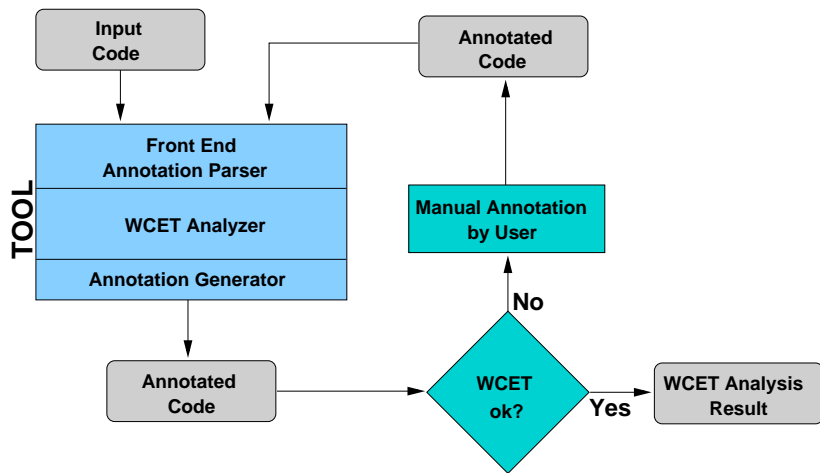
C ... Class (timing invariant or timing fiction)

G ... Group (for practical purposes)

U ... Addressable Unit (possibly referenced)

I ... Annotation Information

Ideal Work Flow

Conclusions

> **A common annotation language including all essential ingredients may allow ...**
>
> - Evaluation of tools by using the same set of annotated input sources.
> - Language designers and tool developers to focus separately on the essential ingredients of timing annotations, (L, C, G, U, I).
> - Create interchangeable building blocks within a timing analysis framework (as intended by ARTIST2).

Thank you for
your attention!

ALL-TIMES

CoSTA
Austrian Science Fund
contract P18925-N13
`costa.tuwien.ac.at`

ALL-TIMES
7th EU R&D FP
contract 215068
`all-times.org`

ARTIST2 NoE
6th EU R&D FP
contract IST-004527
`artist-embedded.org`