


WCET Analysis: The Annotation Language Challenge¹

Raimund Kirner, Jens Knoop, Adrian Prantl,
Markus Schordan and Ingomar Wenzel

Institute of Computer Languages
Vienna University of Technology

July 3, 2007

¹This work has been partially supported by the Austrian Science Fund (Fonds zur Förderung der wissenschaftlichen Forschung) within the research project "Compiler-Support for Timing Analysis" (CoSTA) under contract P18925-N13. This work has been funded in part by the ARTIST2 Network of Excellence. 

A closely related challenge

The WCET Tool Challenge (Gustafsson et al.)

- ▶ presented at ISoLA 2006
- ▶ surveyed the diverse strengths of WCET *tools*

But what about *Annotation Languages*?

Surveyed Languages

We considered languages that

- ▶ showcase unique features
- ▶ employ distinct methods

... almost 20 years of annotation language history!

| Language | Type | Authors | Year |
|----------------------|-------------|----------------|------|
| TAL | Tree-based | Mok et al. | 1989 |
| PL/IDL | Path-based | Park & Shaw | 1991 |
| SPARK-Ada | Tree-based | Chapman et al. | 1994 |
| Symbolic Annotations | Tree-based | Blieberger | 1994 |
| IPET | Graph-based | Li & Malik | 1995 |
| Bound-T | Graph-based | Holsti et al. | 2000 |

Assessment Criteria

Language Design

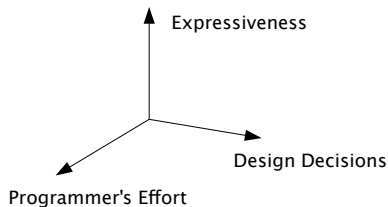
- ▶ Expressiveness
- ▶ Annotation Placement
- ▶ Programming Language

Usability

- ▶ Programmer's Effort

Work Flow

- ▶ Tool Support



Expressiveness

- ▶ 1. Explicit Execution Frequency

...possible with all surveyed languages!

Expressiveness

- ▶ 1. Explicit Execution Frequency
- ▶ 2. Explicit Execution Order

Example from PL/IDL (Park and Shaw 1991)

```
1 void cond(int a[], int b[])
2 {
3     int i=0, j=0;
4     while (i < 100) {
5         if (a[i] < 10)
6             j++;
7         else
8             a[i]=10;
9             i++;
10        if (b[i] < 10)
11            j++;
12    }
13 }
```

C-Source

Path Language:

Loop-bound:

$\neg(*L5*) + (.L2(.L5)^{100}) * .$

$(*L6*) \cap (*L11*) + \neg(*L6*) \cap \neg(*L11*)$

IDL Statements:

Loop-bound: loop L4 100 times

samepath(L6, L11)

Path Annotations

...only one language supports this!

Expressiveness

- ▶ 1. Explicit Execution Frequency
- ▶ 2. Explicit Execution Order
- ▶ 3. Context-Sensitive Flow Information

Example from SPARK Ada (Chapman et al. 1994)

```
1  —\# proof function pow(FLOAT,INTEGER) return FLOAT;  
2  function POWER(BASE: in FLOAT; EXPONENT: in INTEGER) return FLOAT  
3  —\# pre true;  
4  —\# mode A (EXPONENT >=0);  
5  —\# mode B (EXPONENT < 0);  
6  —\# post (POWER = pow(BASE,EXPONENT));  
7  is ONE: constant FLOAT := 1.0;  
8     EXCHANGE: BOOLEAN; L.RES: FLOAT; L.EXP: INTEGER; RESULT: FLOAT;  
9  begin  
10     L.RES := ONE;  
11     if EXPONENT >= 0 then  
12     ...
```

Call-Context-sensitive: \approx 50% of the languages.

Expressiveness

- ▶ 1. Explicit Execution Frequency
- ▶ 2. Explicit Execution Order
- ▶ 3. Context-Sensitive Flow Information
 - ▶ Call Context
 - ▶ Loop Context

Treat first and subsequent iteration in a loop differently

- ▶ One approach to integrate this with Linear Flow Constraints was presented by Engblom and Ermedahl 2000

Annotation Placement Different Approaches:

- ▶ Integrated into the source language
- ▶ Inside the source file
- ▶ As separate file

Example: Discrete Loops (Blieberger 1994)

```
1 k:= ...;  
2 discrete h := k in 1..N/2  
3     new h := 2*h | 2*h+1 loop  
4 <loop body>  
5 end loop
```

Example from Bound-T (Holsti et al. 2000)

```
1 subprogram "compute_sum"  
2     loop that contains (loop)  
3     repeats N_MAX times;  
4     end loop  
5     loop that is in (loop)  
6     repeats N_MAX-1 times;  
7     end loop  
8 end "compute_sum"
```

Comparison

| Criteria \ Method | TAL | PL and IDL | Lin. Flow Constr. | Bound-T | SPARK Ada | Symbolic Annot. |
|-------------------------------|---|--------------------------------|--------------------------------|---------------------------------|----------------------|----------------------------------|
| Expressiveness | Timing schema | Regular expressions | Constraint-based | Constraint-based | Loop-annotations | - |
| Loop-bounds | yes | yes | yes | yes | yes | yes |
| Triangle-loops | yes | no | yes | some | no | yes |
| Call contexts | yes | no | possible | implicit | modes | no |
| Loop contexts | no | no | possible | no | no | no |
| Modes | no | no | no | no | yes | no |
| Execution order | no | yes | no | no | no | no |
| Annotation placement | External TAL-script | Ideally inside the source code | Ideally inside the source code | External file | Source code comments | Integral part of the source code |
| Object code annotation | yes | no | no | yes | no | no |
| Host language | Assembler, some support from a C-compiler | C, any structured language | Any structured language | C, Ada, any structured language | Ada83 | Any structured language |
| Programmer's effort | high | mid-high | mid-high | mid | low | mid |
| Tool support | yes | no | yes | commercial | yes | prototype |

Why a new Challenge?

Precision and performance of WCET analysis depends on

- ▶ expressiveness
- ▶ usability

of the **annotation language**.

We believe that

- ▶ mastering the WCET Annotation Language Challenge is essential for consolidating and advancing the state-of-the-art

Join the Challenge

Join the challenge to develop...

- ▶ Annotation languages enjoying the strengths of the languages while avoiding their limitations!

Contribute to establish...

- ▶ a unified benchmark suite for all different types of flow information.

We offer to host such a benchmark library for annotation languages!

Thank You!