The WCET Tool Challenge 2006

Jan Gustafsson
Dept. of Computer Science and Electronics, Mälardalen University, Västerås, Sweden
jan.gustafsson@mdh.se

***Abstract*— The purpose of the WCET Tool Challenge is to be able to study, compare and discuss the properties of different WCET tools and approaches, to define common metrics, and to enhance the existing benchmarks. The WCET Tool Challenge has been designed to find a good balance between openness for a wide range of analysis approaches, and specific participation guidelines to provide a level playing field. This should make results transparent and facilitate friendly competition among the participants.**

I. Introduction

The WCET Tool Challenge will be performed during the autumn of 2006. It will concentrate on three aspects of WCET analysis:

1) flow analysis,
2) required user interaction,
3) performance.

Companies as well as research groups are welcome to participate. The actual work with the tools is made by a PhD student and/or the development teams. We will target the evaluation on a set of benchmark programs.

The report to be presented at ISoLA 2006 will be based on the reports from the developers and the student. The report will be compiled by the working group.

For more details, consult the Challenge web page `http://www.idt.mdh.se/personal/jgn/challenge/`.

A. Goals

The goals of the WCET Tool Challenge are the following:

1) To exhibit the wide range of timing analysis tools available today:

- using static program analysis, or
- combining analysis and measurements,
- for various target processors,
- in various application domains,
- supporting various programming languages and design tools,
- academic, commercial; free or at a charge.

2) To illuminate the features, abilities and intended uses of each tool:

- in finding the feasible execution paths in the SW,
- in modelling complex processor and system HW,
- in deriving useful WCET bounds or estimates,
- in usability, scalability and adaptability,
- in the range of supported targets (processors, compilers, ..)

3) To collect and maintain a growing set of community standard benchmark programs and related test suites that:

- contain typical (both easy and hard) programming constructs,
- can be analyzed by several tools with comparable results,
- test enough of the actual behaviour of each benchmark to satisfy measurement-based tools and to validate results from static-analysis tools, and ideally, have known exact answers (paths and WCETs).

Note that the test suites for the benchmark programs have two roles: firstly, they enable the participation of the measurement-based tools like SymTA/P and RapiTime; secondly, they provide a validation of the static analysis results.

B. Aspects of WCET analysis

1) Area 1 - Flow analysis: The purpose of the flow analysis phase is to extract the dynamic behaviour of the program. This includes information on which functions get called, loops bounds, if there are dependencies between if-statements, etc. We propose the following flow analysis metrics to be measured:

- number of automatically found loop bounds (including context-depending bounds, like triangular loop limits and loops in functions called from several sites)
- tightness of these (compared to real loop bounds, assuming they are known)
- the number of automatically found infeasible paths
- their reduction effects on the WCET estimates
- the number of automatically found correct memory accesses
- the number of automatically found resolved call targets (function pointers)

2) Area 2 - Required user interaction: This area of evaluation is concerned with the amount of work that is involved with setting up a WCET calculation to receive a result. One important metric for this area is number of program-specific manual annotations. Necessary annotations (like CPU type, frequency etc) can be excluded from the number.

3) Area 3 - Performance: This area is about the bottom line: the final WCET value and the performance of the tool. We propose the following metrics:

- estimated WCET value (in clock tics and μs)
- tightness of the estimated WCET value (assuming the real WCET is known)
- limits of program sizes to be handled

- analysis time and memory requirements for the analysis (give the CPU time, memory usage (Mb), and describe the execution platform)

C. *WCET tool rounds*

Aspects may be non-orthogonal and influence each other. For example, much preparation work may give a better (tighter) WCET. This is expected and normal, and shows the signs of a flexible WCET tool. The same tool can be used for different aspects with different setups. Therefore we suggest that each tool is used in three rounds for each target processor:

1) One initial round with no manual annotations for loop bounds etc. Necessary annotations (like CPU type, frequency etc) are however allowed. This run may not give a WCET bound at all for some tools and benchmarks.
2) A basic round with the smallest set of manual annotations possible to get a WCET bound.
3) An optimal round with the largest set of annotations to get as tight WCET bound as possible.

The required user interaction is of course growing for each round. For each round, the metrics are measured for the three aspects. For each metric, the complete setup is described.

D. *Carrying the evaluation out*

There will be two possibilities:

1) The evaluation is carried out by a PhD student Lili Tan `<lili.tan@icb.uni-due.de>`, who makes this job as part of her PhD studies. This will have the advantage of letting an external newcomer try out the tools, which can give feedback of the usability of the tools. The student will make a report on the evaluation and give to the working team.
2) The evaluation is carried out by the development team. Results are expected to be sent in to be included in the report to ISoLA.

There will be a choice for the participants to use one or both of these approaches.

E. *Selection of benchmark programs, processors and compilers.*

The benchmarks will represent different types of codes, for example code with different types of loops, infeasible paths, automatically generated code, hardware specialized code, and also large real world programs. A mix of single-path programs and multi-path programs will be included.

We will use open source benchmark programs from the MŁlardalen WCET benchmark and PapaBench. These benchmarks are all available on the web.

We suggest that each participant selects up to three processors for which to do the analyses; for example one simple (e.g., Renesas H8), one medium complex (e.g., ARM7/9, C167NEC, V850E) and one very complex (e.g., PowerPC), if possible.

As there is no overview over which compilers are supported by which tools, we let the participants decide on one or two compiler(s).

II. Conclusion

The WCET Tool Challenge will be performed during the autumn of 2006. A report of the work will be presented at ISoLA 2006. In the report, the participating tools and their results will be presented in detail.

Acknowledgment

This work is being supported by ARTIST2 European Network of Excellence.

We would also like to thank the other members of the WCET Challenge Working Team:

Reinhard Wilhelm `<wilhelm@cs.uni-sb.de>`, Reinhard v. Hanxleden `< rvh@informatik.uni-kiel.de>`, Steffen Goerzig `<steffen.goerzig@daimlerchrysler.com>`, and Paul Levi `<Paul.Levi@informatik.uni-stuttgart.de>`.