CUTTING-EDGE TIMING ANALYSIS TECHNIQUES

JAKOB ZWIRCHMAYR [1]

[1] Vienna University of Technology,
Argentinierstrasse 8,
1040 Vienna, Austria
E-mail address: `jakob@complang.tuwien.ac.at`
URL: `http://www.complang.tuwien.ac.at`

ABSTRACT. This text gives an overview about my current research in timing analysis at the Vienna University of Technology. After a short introduction to the topic follows the description of an approach relying on CLP, the *implicit path enumeration technique (IPET)*. This technique is also used in a tool developed at the institute of Computer Languages (TuBound). Current timing analysis tools suffer from a few flaws worth further investigation in order to achieve better results than current state-of-the-art timing analysis tools.

Introduction

These days *embedded software systems (ESS)* are found in many devices we rely on in our daily lives. In many cases extensive testing gives us sufficient confidence in the product and it can be sold and used the way it was intended to. On the other hand, we rely on ESS that control very crucial mechanisms and functionality in devices and machines our safety and lives depend upon every day. Examples of such ESS usually come from the avionics and automotive industry, e.g. the technologies *fly-by-wire* and *drive-by-wire*. There are no mechanical links between the control column and the steering gear of an aircraft and the steering wheel and the wheels of a car [Kov10], or the system that is responsible for the proper functioning of the airbag in a car. These systems are considered safety-critical hard real-time systems (RTS). The airbag control software is required to compute and open the airbag fast enough if sensor data matches an accident condition. In this case, correct functioning of the ESS is not a matter of comfort and convenience but a matter of life and death. It is crucial to abide to certain resource bounds, e.g. memory consumption and time consumption. Guaranteeing program execution within a certain time bound is crucial for safety-critical hard RTS, i.e. guarantee that under no circumstances the time-bound is exceeded. The *worst-case execution time (WCET)* must in all cases be below the computed bound. The WCET-bound should be as precise as possible as overestimation usually implicates higher costs or redesign of the component. As a consequence, WCET underestimation is certainly not an option.

1. Background

Precision and performance of WCET analysis tools depend on the undecidable problem of identifying and separating feasible and infeasible program paths [Kov10]. Therefore WCET analyzers often require manual user intervention, often in the form of source- or binary-code annotations. Typical code elements that require user interaction (annotations) are loop constructs (upper bound on loop iterations) and recursive procedures (upper bound on recursion depth) [Pra09b]. There are two major problems due to this fact:

- Annotating binary code is tedious and even on source code level complications can occur, e.g. annotations in external components. Therefore, a fully automated procedure that infers this information is preferred.
- Manual annotations prevent the tool from formally establishing safety and accuracy of the analysis: the tool has to rely on a *trusted annotation base*, there are no guarantees that the user provided annotations are safe [Pra09b].

2. Goals

The *Cutting-edge Timing Analysis Techniques CeTAT* project is a cooperation between the Institute of Computer Languages and the Institute of Computer Engineering, at Vienna University of Technology. Some aspects of the problems of state-of-the-art WCET tools summarized in the previous sections can be overcome: there are approaches to verify the trusted annotation base supplied by the user. For example, the tool TuBound includes a bounded model checker that can be used to verify loop bounds inferred by the tool or provided by the user. The annotations are instrumented into the program as assertions that can be verified by the model checker. Nevertheless, there are complex loop constructs where such tools cannot find an upper bound on the number of loop iterations, preventing thus accurate WCET analysis.

The WCET community would benefit greatly from a common annotation language, such that tools can share information. Most tools have their own style of storing inferred information. Moreover, for easier tool comparison, a common annotation language would be helpful. Identifying (in)feasible paths is a complex task. Nevertheless there are various approaches in the area of symbolic computation (theorem proving) and termination analysis that are able to handle complex nested loops that would require manual annotations for most WCET tools. Our research aims at combining traditional timing analysis techniques and state-of-the-art approaches that use satisfiability modulo theories (SMT) to tackle the problem of complex nested loops [Gul09] with methods from symbolic computation and theorem proving.

3. Current status

I joined the CeTAT project group in March 2010. There is a good foundation of research in various directions of WCET analysis and the techniques we want to incorporate in order to pursue research in the CeTAT project. This includes in particular:

- Beyond Loop Bounds: Comparing Formative Annotation Languages for WCET Analysis [Kir10]. This work presents a survey of state-of-the-art annotation languages considered formative for the field. According to [Kir10], the precision, generality and efficiency of WCET analysis tools depend much on the expressiveness and usability of annotation languages.
- Constraint Solving for High-Level WCET Analysis [Pra09a]. The authors present the results achieved by their tool TuBound at the WCET tool-challenge 2008. TuBound is a constraint logic based approach for loop analysis developed at Vienna University of Technology.
- ABC: Algebraic Bound Computation for Loops. Presents a software tool for automatically computing symbolic upper bounds on the number of iterations of program loops [Bla09]. The authors of [Bla09] combine static analysis of programs with symbolic summation techniques to derive loops invariant relations among program variables.

As a starting point for the project we are currently investigating WCET benchmarks that contain loops that were not handled by TuBound in the tool challenge. It will be necessary to identify techniques that yield good usability, scalability, runtime performance, and most important, WCET results. Based on the results of our experimental evaluations we will design a new tool that outperforms current state-of-the-art tools in this respect.

References

[Bla09]   Regis Blanc, Thomas Henzinger, Thibaud B. Hottelier, and Laura Kovacs. Abc: Algebraic bound computation for loops. In *LPAR-16 – 16th International Conference on Logic for Programming Artificial Intelligence and Reasoning*. 2009. Unpublished.

[Gul09]   Sumit Gulwani and Florian Zuleger. The reachability-bound problem. Tech. Rep. MSR-TR-2009-146, Microsoft Research, 2009.

[Kir10]   Raimund Kirner, Jens Knoop, Adrian Prantl, Markus Schordan, and Albrecht Kadlec. Beyond loop bounds: Comparing formative annotation languages for worst-case exeution time analysis. *Software and Systems Modeling*, 2010.

[Kov10]   Dr. Laura Ildiko Kovacs. Cutting-edge timing analysis techniques for safety-critical real-time systems (cetat), 2010. Project description.

[Pra09a] A. Prantl, J. Knoop, M. Schordan, and M. Triska. Constraint solving for high-level WCET analysis. *ArXiv e-prints*, 2009.

[Pra09b] Adrian Prantl, Jens Knoop, Raimund Kirner, Albrecht Kadlec, and Markus Schordan. From trusted annotations to verified knowledge. In Niklas Holsti (ed.), *9th Intl. Workshop on Worst-Case Execution Time (WCET) Analysis*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, Dagstuhl, Germany, 2009.