Parametric WCET Analysis

Stefan Bygde

February 23, 2007

1 Research Area

My research area is parametric Worst Case Exectution Time (WCET) analysis. WCET analysis is about getting information about the worst-case execution time for a program on a certain platform for any possible input. Static WCET analysis is an automatic way to derive a safe upper bound for the worst-case execution time of a program by analyzing its source code and the hardware of the target platform. In many cases it is useful to have such an estimation of the WCET, especially for hard real-time systems. Normaly, the WCET is estimated by measurements of the executing program (dynamic analysis), which gives an underestimation of the actual WCET (because it is not possible to be sure that the worst case has been measured). In static analysis on the other hand, the estimation of the WCET should *always* be equal or greater than the actual WCET (ofcourse, assumed that the analysis tool is correct). In classic static WCET analysis, the upper bound of the execution time is given as an absolute value (for instance, in milliseconds) whereas parametric WCET-analysis aims to derive the WCET-bound as an expression of some undetermined variables, which later can be instantiated. The techniques used for this is quite general static analysis techniques and could possibly be used in other applications than WCET analysis.

2 Research Methodology

The first step to examine this area is to implement a fast prototype to find out how well some suggested theoretical ideas of parametric WCET analysis would work out in practise. The prototype will hopefully reveal the important issues and problems that have to be solved. These issues and problems should then be carefully examined and some theory and/or techniques should be developed. Finally, if the methods/ideas developed seem to work, the plan is to do a more serious implementation into an existing research-prototype developed at MdH.

3 Research Overview

There are in general two ways of estimation the WCET of a program; dynamic and static analysis. Dynamic analysis is done by measurements on the running software, there are many techniques and tools for doing that. Measurements takes a lot of effort to do (setting up the gear and so forth). By dynamic analysis one typically obtains an under-estimation of the "real" WCET, because it is hard (not to say impossible) to force the program to its worst case behaviour. On the other hand, we have static analysis which is an automatic proccess that derives a *safe upper bound* of the actual WCET. By a safe upper bound we mean an estimation that is *guaranteed* to be at least as high as the real WCET. However, the estimation should be as *tight* as possible, meaning that we aim to get as close as possible to the real WCET without comprimising the safety. To obtain such an estimation one has to take both software and hardware into consideration. Usually this analysis is divided into different phases called flow analysis, low-level analysis and calculation.

3.1 Flow Analysis

The most important impact on the execution time of a program is ofcourse the software. Execution typically spends most time in loops and finding (safe) upper bounds of the execution bounds of loops is essential for obtaining a good estimation of the WCET. The flow analysis phase analyses the source code (or possibly some intermediate format or even the object code), to find bounds on the execution counts for all program points. This is non-trivial, in fact, finding exact flow-information is in general uncomputable, so one typically has to introduce some abstraction to make it computable and in that ofcourse loose some precision. Research problems on this level is mainly static analysis techniques such as slicing (removing variables and parts of programs which doesnt affect the program flow), abstract interpretation (general framework for abstracting program semantics) etc.

3.2 Low-level Analysis

Finding the flow facts of a program is not enough to derive a concrete WCET estimation, programs can behave differently on different platforms, especially on modern platforms where you typically find cache memories, pipelines, branch prediction etc. These construct improves the average performance but at the same time it introduces unpredictability and it can in some cases yield very complex worst-case behaviours. The low-level analysis phase tries to find the worst-case behaviour for the atomic instructions of the hardware using a *timing model*. The timing model is a abstract model of the hardware (including caches, pipelines etc.) which focuses on timing aspects. The model has always to assume the worst possible case (e.g. assuming a cache-miss if it cannot without doubt infer a cache hit) in order to be safe. The output of the Low-level analysis is worst case behaviour of the atomic parts of a program. Research on this level includes modeling of hardware, predictability and so forth.

3.3 Calculation Phase

The calculation phase combines the results from the flow analysis and the low-level analysis to derive a concrete WCET estimation. There are different techniques to achieve this. A simple example is the Implicit Path Enumeration Technique (IPET) which is simply to take the execution bounds derived in the flow analsis, let‘s call them c_i where i is a program point, and the worst-case execution time of atomic parts of the program x_i and then maximize the sum

$$\sum_{i \in P} p_i * x_i$$

subject to some constraints obtained from the flow analysis. This sum will surely include the worst-case path through the program, however, while the path is certainly possible to take structurally, it might be impossible to take due to semantic constraints which is not taken into consideration by this method. Thus the sum may correspond to a path through the program which is not actually a valid path (i.e. it will not occur in practise) but it will certainly be at least as long as the real worst-case path and therefore we will obtain a safe over-estimation of the actual WCET. To tighten the estimation the flow analysis can provide some info on *infeasbile paths* which is paths through the program which is structually possible to take but impossible to take in practise.

3.4 Relation to my research

My research will focus on the flow-analysis part, but will also include the calculation phase. As mentioned above, yielding exact information about a program is in general uncomputable so a framework for abstracting the semantics of programs was proposed in the seventies by Cosout & Cosout [CC77]. Using abstract interpretation and a parametric IPET technique, my aim is to try and obtain the WCET estimation as a expression in some unknowns rather than as a constant value. The unknowns will in the first attempt correspond to program (input) variables, and the WCET estimate will thus be given in terms of them. The technique will mainly rely on different techniques witch has been used in other applications.

4 Conferences

This project is somewhat overlapping with different communities. While the context of the project is to obtain a parametric WCET formula for a piece of code in embedded real-time systems, the techniques and research is quite general and could be used in many other applications as well. Therefore conferences in static analysis in general are the most interesting conferences and those of embedded systems and real-time system are of less interest.

SAS

One of the most important conferences in program analysis in general is the Static Analysis Symposium (SAS), topics in this conference is limited to static analysis including topics as: abstract interpretation, control flow analysis, model checking etc. In this conference there are mainly theoretical and general approaches presented.

TACAS

Another conference related to WCET analysis is the Tools and Algorithms for the Construction and Analysis of Systems (TACAS) which is more focused (as the name suggests) on tools and algorithms. The conference is quite general and is meant to find similarities between different communities. Topics include but are not limited to for instance: formal methods, software and hardware verification, static analysis, programming languages and real-time systems.

SCAM

Another conference which focuses on static analysis which is possibly more applied than SAS is the conference on Source Code Analysis and Manipulation (SCAM) conference. Some topics: program transformation, abstract interpretation, program slicing, source level software metrics and decompilation.

WCET

There is also a very focused international workshop on WCET under the Euromicro Conference on Real- Time Systems (ECRTS) conference. The topics in this conference is limited to timing analysis and accepts submissions that deals with, for example: Flow analysis for WCET, Low-level timing analysis, modelling and analysis of processor features, Calculation methods for WCET, Strategies to reduce the complexity of WCET analysis, Timing analysis through measurement, Probabilistic analysis techniques and others.

5 Literature

Abstract Interpretation

The topic of static analysis, abstract interpretation and WCET analysis is quite big, there are several conferences with related material and so forth. This section will be focusing on central literature in my particular work. Since my work will be centralized around abstract interpretation, there is one important seminal paper, that is the introduction paper by Cousots in 1977. It has the not-so-short title "abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints" [CC77]. This has generally been accepted as a subject and as could be seen above; many conferences has this as a topic. One important topic in abstract interpretation is choosing a suitable abstraction for the semantics, such an abstraction is commonly refered to as an *abstract domain*. Important such domains has been proposed in [CC77], [CH78], [Gra89].

Program Analysis

When it comes to static program analysis in general there is a very important book which attempts to unify most static analyses and showing similarities between them. It is an esential source of knowledge for beginners and researchers in static analysis. The title is "Principles of Program Analysis" by Nielson, Nielson and Hankin [NNH05].

Parametric WCET Analysis

There has not to our knowledge been much research on this specific subject. My research will be heavily based on Björn Lispers paper "Fully automatic, Parametric Worst-Case Execution Time Analysis" [Lis03]. The ideas in this paper will be further investigated, implemented and evaluated. Inspired by Lispers paper are two master thesises developed at Saarland University [Hum06][Alt06]. These thesises used many ideas from [Lis03] but used a little bit different approach. This literature will be very central to this work.

6 Research Groups

There are some research groups that actively works with static WCET analysis.

The compiler design lab of Saarland university

Parts of this lab has timing research as interest and some interesting publications has been written by this group.

Patrick Cousots group on abstract interpretation

Patrick Cousot, one of the founders of abstract interpretation is still active in the research of abstract interpretation. His research group has a very informative web page with alot of pointers to publications involving abstract interpretation.

AbsInt

This is a company rather than a research group. As the name of their title hints, they do static analysis based on abstract interpretation. They have developed a tool called aiT, which is a static analyzer. The tool focuses on hardware timing and performs abstract interpretation over the states of caches and pipeline.

References

[Alt06] S. Altmeyer. Parametric wcet analysis, parametric framework and parametric path analysis. Master's thesis, Saarland University, Department of Computer Science, Oct 2006.

[CC77] Patrick Cousot and Radhia Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proc. 4th ACM Symposium on Principles of Programming Languages*, pages 238–252, Los Angeles, January 1977.

[CH78] Patrick Cousot and Nicholas Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Proc. 5th ACM Symposium on Principles of Programming Languages*, pages 84–97, 1978.

[Gra89] Philippe Granger. Static Analysis of Arithmetical Congruences. *International Journal of Computer Mathematics*, pages 165–199, 1989.

[Hum06] C. Humbert. Parametric wcet analysis, parameter analysis and parametric loop analysis. Master's thesis, Saarland University, Department of Computer Science, Oct 2006.

[Lis03] Björn Lisper. Fully automatic, parametric worst-case execution time analysis. MRTC report, Dept. of Computer Science and Engineering, Mälardalen University, April 2003. MRTC Report ISSN 1404-3041 ISRN MDH-MRTC-97/2003-1-SE.

[NNH05] F. Nielson, H. R. Nielson, and C. Hankin. *Principles of Program Analysis, 2^{nd} edition*. Springer, 2005. ISBN 3-540-65410-0.