A Modular Methodology for Automatic Static Parametric WCET Analysis

Stefan Bygde
February 1, 2011

Abstract

In order to overcome some of the inherent pessimism of traditional static analysis, we propose a method for parametric WCET analysis expressing the Worst-Case Execution Time (WCET) of a program as an expression in terms of the input data of a program. Besides giving more information about the execution time behaviour of the program, this also has benefits in component based development where reusable components can be annotated with WCET formulae for different instantiations. Also, if efficient enough, analysis could potentially be used in dynamic real-time schedulers, as the input data for a program may be known at run-time. This thesis proposal describes the planned work for a suggested parametric method; implementation, evaluation and theoretical work with bit-precise abstract domains to achieve a precise, sound and efficient parametric analysis.

1 Introduction

The execution time of any program can vary a lot. Any program with a non-trivial task has loops and branches, and the execution path typically depends on input data to the program. This means that in most non-trivial programs, the execution time is highly dependent on input data to the program. In a real-time system, it is necessary to have a bound of the Worst-Case Execution Time (WCET) for all of its tasks, in order to schedule it properly. Naturally, the WCET would correspond to the execution of the program on the input data which causes the highest possible execution time. However, such an input is not necessarily ever used in the system. Moreover, since the execution times of a program may be highly varying, the worst-case may be too pessimistic to use in practice since the paths used in practice may have much shorter executions. The conclusion is that a single estimation of the WCET of a program is often not feasible; the final estimation should vary according to the input data.

1.1 Parametric WCET Analysis

A natural idea to solve the problem of a constant estimation of the WCET is to express the WCET as a function of the input data. The idea is not new; static parametric analyses have been suggested in [CHMW07, VHMW01, CB02, AHLW08, Lis03]. In this thesis we will continue the work presented in [Byg10], which is based on the method in [Lis03]. This analysis is, as far as we know, the most flexible and most complex of the parametric analyses proposed in literature. It is based on abstract interpretation [CC77], a well-known method for soundly abstracting program semantics. In particular, it is based on relational abstract interpretation, meaning that some of the relations among variables in the abstraction are preserved. The basic idea of the analysis is to abstract the program semantics in a relation-preserving way for each point in the program, and use the fact that a deterministic and terminating program will have to enter a new unique semantic state in each step of an execution trace. The execution count of each part of the program can then be computed parametrically by calculating the size of the
abstract states a program may have (which are guaranteed to include all concrete semantic states) with respect to some input variables considered as parameters. Since the number of abstract states are counted with respect to input variables, it is necessary for the abstraction to be aware of the relation between the parameter variables, and the other variables in the program. This is the reason why a relational abstract interpretation is necessary.

A minor evaluation of this method was presented in [Byg10]. However, the evaluation was done on very small and not very realistic benchmark programs. In order to evaluate the method on a larger scale, the following steps need to be done.

- The method has to be implemented into a tool able to analyse realistic programs, meaning representative programs that might be used in embedded systems. We are planning to get the method implemented into the static analysis tool SWEET [EG97]. SWEET can analyse standard C programs and will enable a proper evaluation of the suggested method.

- Abstract Interpretation is a key technology in the method. However, there are many practical issues that have not been solved when analysing real software, in particular, some common assumptions (such as that integer variables take values from the set of mathematical integers) can lead to unsound or imprecise results, more on this in Section 1.3. Some of these issues have to be solved in order to make a precise, sound and functioning analysis.

1.2 Parametric WCET Calculation

The parametric method for WCET analysis presented above is based on abstract interpretation and counting of abstract states. However, in order to get concrete timing estimates of any program on a specific platform, a low-level analysis is needed. A low-level analysis is an analysis of a model of the hardware which bounds the WCET of the atomic parts of a program (such as instructions, or basic blocks) while considering the effects from the hardware, such as pipelines, caches, branch-predictors etc. To get concrete time bounds on a program, static WCET analysis typically combines the low-level analysis results with the analysis results obtained from analysing the control flow of the program. This is usually referred to as the calculation phase of a WCET analysis.

The parametric method also has a calculation phase, but it has to be specially adapted for parametric analysis. This has been shown to be the bottleneck of our method. In our work we will investigate how the complexity of parametric calculation can be reduced. Two different parametric calculations have been suggested: the original suggestion from [Lis03] was based on parametric integer programming [Fea88], and the more efficient, but also more approximate and less flexible method called the Minimum Propagation Algorithm (MPA) [BEL10]. We plan to develop these calculation methods further, and evaluate them on a larger scale than previously.

Yet another potential approach is to model time as a variable in the abstract interpretation, very much similar to what Holsti did in [Kir08]. This would mean to add an artificial variable to the analysed program and update it in every basic block with the estimated WCET of that basic block. Doing this would eliminate the need for a calculation phase as time would explicitly be included in the abstraction. The time could then be expressed in terms of its relation to other program variables.

1.3 Bit-precise Relational Domains

Abstract interpretation is, as mentioned, a technique for soundly abstracting program semantics. It is very flexible and can abstract different characteristics via so-called abstract domains. An abstract domain describes which properties should be preserved in the abstraction. In static analysis abstract domains are usually used to abstract the program environments. Simply put, an environment is a mapping between variable names and, without too much loss of generality, integers. Abstractions of an environment can be classified into relational and non-relational domains. A non-relational domain abstracts each mapping (i.e., each variable) to a state representing a set of integers. This makes the variables independent of
each other. A relational domain, on the other hand, abstracts the full mapping into an abstract environment representing a set of mappings and thus preserves some relations among the variables. As an example, the classic interval domain [CC77], is a non-relational abstraction mapping each variable name to an interval of integers. The polyhedral domain [CH78] is a relational domain, mapping an environment to a convex polyhedron in $n$-space, where $n$ is the number of variables in the environment.

In literature, most relational integer domains are presented with mathematical integers as underlying model. In a computer, however, integers are usually represented by a fixed-size binary string. Even on an abstract level, using mathematical integers as underlying model leads to assumptions which may not be sound. An example is that $n < n + 1$ might not be true due to wrap-around effects. A list of problems with the assumption of having mathematical integers instead of fixed size binary strings follows:

- Representing integers as binary strings can cause wrap-around effects. This would make traditional analyses (which assume that $n$ is a mathematical integer) unsound. This problem may be prevalent in processors for embedded systems with short word length.

- Integers can be encoded in various ways using bit-strings. An integer can be allowed to be negative if a signed encoding (using two’s complement) is used, or it can use the most significant bit to represent larger numbers by using an unsigned encoding of integers. The semantics of a wrap-around then differs according to the used encoding. Moreover, there may be cases where the encoding is not known a priori (in for instance intermediate code), and it may be desired that the analysis derives a result which is safe for both signed and unsigned integers.

- When integers are encoded as binary strings, it is possible to use calculations involving binary operations, such as bit-shifting or logical operations. This is commonly used in low or intermediate code formats, but also sometimes in high level languages, often to optimise performance. These operations, while common, are usually not handled in a precise manner in analyses using mathematical integers as underlying model.

One of the aims of the thesis is to develop bit-precise abstract relational domains to use in abstract interpretation. These domains should solve the problems mentioned above in a manner as precise and efficient as possible. This work will be done in cooperation with a company in Finland called Tidorum [Tid11b], the developer of the tool Bound-T [Tid11a].

2 Contributions

This section lists the concrete contributions of the thesis. Note that these contributions are useful both in combination and individually. The contributions does not necessarily depend on each other.

- A functioning, flexible method for static parametric WCET analysis deriving estimations in terms of input variables.

- A set of relational bit-precise abstract domains suitable for analysis of real code on high or intermediate code level. These domains will enable analysis on low level code and allows for more precise analysis results. Bit-precise relational domains would also be useful in general for applications outside parametric WCET analysis.

- An efficient algorithm for parametric calculation (i.e., the minimum propagation algorithm) and a thorough evaluation of it in comparison with other parametric calculations.

- An evaluation of the method on realistic benchmarks. The following evaluations are planned:
  - Is it feasible to analyse realistic systems using the method?
  - How is the precision using a parametric analysis in comparison to a traditional one?
  - How much precision do we gain by using bit-precise domains compared to traditional ones?
How do different trade-offs (using different abstract domains/calculation methods) compared in precision/efficiency?

3 Related Work

3.1 Parametric WCET Analysis

There are some previous work on parametric static WCET analysis. A number of analyses parameterised in loop bounds has been suggested [VHMW01, Cha94, CHMW07]. In [CB02], an analysis is proposed which is parameterised in some function parameters. However, manual annotations has to be provided to support this analysis. A parametric WCET analysis suggested in [AHLW08] automatically computes from a given program suitable variables to be used as parameters. The analysis could be classified as weakly relational, but it does not fully preserve relations among all variables. As far as we know, the method originally presented in [Lis03] is the most complex and general proposed method for parametric analysis. It handles arbitrary program flow, is based on relational abstract interpretation and can potentially obtain very detailed formulae binding the WCET. The method have been implemented and further investigated in [BL08, BEL10, Byg10] and is the method on which we base our research.

3.2 Bit-Precise Domains

The problems listed in Section 1.3 has been recognised in the research community. Some work has been done to solve this for non-relational domains, most commonly on the classical interval domain [CC77], or the congruence domain [Gra89] or the reduced product of the two\textsuperscript{1}. In [SS07], an abstract domain based on Circular Linear Progressions has been suggested. The abstraction is similar to the reduced product of intervals and congruences, but it is modelled specifically to account for fixed size bit-strings. In [MOS07], a version of the congruence domain over the set of congruence classes modulus $n$ is presented, also with the purpose to account for finite size representations of integers. In [Byg07, BEL10], a modification of the congruence domain was suggested. This modification makes the domain suitable for finite-sized bit-strings and is safe for signed and unsigned integers at the same time. In [GEL05], suggestions on how to use the interval domain with fixed size integer representation is presented, and with a dual representation to handle the problem of signed and unsigned integers. To the best of our knowledge, no serious attempt has been made to handle the mentioned problems for relational domains such as the polyhedral domain.

4 Publications

This section presents planned and published papers related to the thesis. Section 4.1 and 4.2 present the publications that will form the basis of the thesis and Section 4.3 presents papers which are related to the thesis. Although the thesis will be based on the publications listed in this section, it is planned to be written as a monograph.

4.1 Planned Publications

**Paper A** This paper will be about bit-precise relational domains. The paper will describe our approach to solve the problems mentioned in Section 1.3. It will define the bit-precise relational domains, which should be designed to precisely and efficiently model program states where integers are represented by fixed size binary strings. The paper should also contain some sort of verification or evaluation of the suggested domains.

**Paper B** This paper should evaluate the parametric method on a set of realistic benchmark programs. This evaluation should evaluate different approaches used in the method; the Minimum Propagation Algorithm and bit-precise relational domains. The aim is to publish this in a journal.

(Paper C) This is a possible collaboration paper. If written, it will be about a significant improvement to the parametric calculation based
on parametric integer programming. This will be evaluated and compared to the minimum propagation algorithm. The outcome of the paper is dependent on an external research group and will be included if appropriate.

4.2 Previous Publications

The following has already been published and the contents of the work in the following publications are going to be included in the thesis.

**Paper D Towards an Automatic Parametric WCET Analysis** [BL08]. This paper presents the initial work with the implementation of the parametric method along with a small evaluation of its feasibility.

**Paper E An Efficient Algorithm for Parametric WCET Calculation** [BEL10]. In this paper, the Minimum Propagation Algorithm is presented and evaluated.

**Licentiate Thesis** The current work is a continuation of the work done in the licentiate thesis [Byg10], and most parts of the work done in the licentiate thesis will be included in the PhD thesis. The licentiate thesis was based on paper D and E as well as two other publications (see Section 4.3) which will not be included in the PhD thesis.

4.3 Related Publications

In this section we present previous publications which are related to the thesis, but the content of these papers will not be included in the thesis since it is a bit out of scope for the focus of the thesis.

**Loop Bound Analysis** The paper [ESG+07] is about using a technique similar to the one used in the parametric method (i.e. counting abstract states) to automatically obtain tight loop bounds. While an important part of WCET analysis, the paper is not related enough to be included in the thesis.

**The Congruence Domain** In [Byg07] an enhancement of a non-relational abstract domain, the congruence domain, was presented. The enhancement bases the domain on bit-strings instead of mathematical integers. While related to the work in the PhD, this is a non-relational domain, and as such, it cannot be used directly in the parametric method and will not be included in the thesis.

5 Time Plan

The time plan of the remaining PhD work and future work is shown in Figure 1. During January-February a first version of a the parametric method should be implemented into SWEET. This is for two reasons; first to be able to later evaluate the method on realistic real-time programs and secondly to be able to experiment on bit-precise relational abstract domains. From March to May, work will continue in cooperation with Tidorum in Finland. During this period focus will be on defining new bit-precise relational domains as described in Section 1.3. The planned result of this stay is a technical report of bit-precise relational domain and eventually a publication (Paper A in Section 4). June to September is devoted to implementation of the suggested domains as well as evaluation. This evaluation should be both on the bit-precise domains in particular and to the parametric method in general, and is planned to result in a publication (Paper B). September-December is then devoted to writing the thesis and take remaining courses. The defence is planned for early 2012.

6 Future Work

In early 2012, after the defence, and eight month forward, another stay in Finland in cooperation with Tidorum, this time as post-doc, is planned for continued work on bit-precise relational domains. This work is to be specified in more detail once the research has started.
<table>
<thead>
<tr>
<th>Time Block</th>
<th>Activities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan-Feb 2011</td>
<td>Implementation SWEET</td>
</tr>
<tr>
<td></td>
<td>PhD Proposal</td>
</tr>
<tr>
<td></td>
<td>Stay in Finland</td>
</tr>
<tr>
<td></td>
<td>Bit-precise Datatational Domains (BPRD)</td>
</tr>
<tr>
<td></td>
<td>Tech. Report</td>
</tr>
<tr>
<td></td>
<td>Paper A</td>
</tr>
<tr>
<td>Mar-May 2011</td>
<td>Implementation BPRD Evaluation</td>
</tr>
<tr>
<td></td>
<td>Paper B</td>
</tr>
<tr>
<td></td>
<td>Courses</td>
</tr>
<tr>
<td>Jun-Sep 2011</td>
<td>Thesis writing</td>
</tr>
<tr>
<td></td>
<td>Paper B</td>
</tr>
<tr>
<td></td>
<td>Courses</td>
</tr>
<tr>
<td>Sep-Dec 2011</td>
<td>PhD Defence</td>
</tr>
<tr>
<td></td>
<td>Stay in Finland Post-doc</td>
</tr>
<tr>
<td></td>
<td>BPRD</td>
</tr>
<tr>
<td>Jan-Aug 2012</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: Timeplan for 2011-2012
7 Courses

7.1 Completed Courses

- Program Language Semantics (7.5 credits)
- Research Methodology (7.5 credits)
- Program Analysis (7.5 credits)
- Real-Time and Embedded Systems (4.5 credits)
- Research Planning (4.5 credits)
- Logic II (3.5 credits)
- PhD School (4.5 credits)
- Advanced Compiler Construction (6 credits)
- PROGRESS reading course (7.5 credits)
- Complexity theory (7.5 credits)
- Advanced computer architecture (not finished) (7.5 credits)

Total: 61.5 credits (40.5 old credits).

7.2 Courses to Take

This means that around 14 credits (9.5 old credits) are missing for a PhD degree. Two courses are planned

- Advanced computer architecture (nearly finished) (7.5 credits)
- Program Analysis for low-level code (7.5 credits)

The work of the first course has already started and needs only to be finished up. The second course is a planned reading course about program analysis and the special challenges when analysing code on a low-level (such as intermediate or object code formats). This will yield another 15 credits (10 old credits) and will be sufficient for a PhD degree.

8 References

References


