OTAWA, Open Tool for Adaptative WCET Analysis

Hugues Cassé, Christine Rochange

{casse, rochange}@irit.fr

IRIT - University of Toulouse – France

http://www.irit.fr/recherches/ARCHI/MARCH/

Abstract

OTAWA is a generic framework for the computation of Worst-Case Execution Time. Delivered under the LGPL licence, it provides a versatile environment with services for static analyses on executables.

1. Introduction

OTAWA[1,2] is a freeware application dedicated to the computation of the WCET[3] of programs. The WCET is usually estimated when designing critical real-time systems embedded in avionics and automotive. As it is an essential information for tasks scheduling, time constraints fulfilment checking but also for sizing the hardware resources, the computed WCET must be_both safe and tight. Safety is achieved by a proved WCET overestimation based on sound analyses. On the other hand, tightness is obtained using detailed models of the hardware allowing cycle-level accuracy of the estimated execution times.
More precisely, OTAWA concerns the numerous WCET approaches based on static analyses of the executable program. Unlike many existing usual tools, the choice has been done to avoid specialisation and to produce, instead, a generic and open framework.
After a short description of the motivations that led us to the development of OTAWA, we present the architecture of this framework, i.e. (1) how multiple architectures are supported and (2) how extensibility is achieved. Finally, the current achievement level is presented and we conclude the paper.

2. OTAWA Genesis

For many years, the compilation and architecture domains have seen the development of generic and experimental frameworks as soon as the problem model had become mature enough. For example, we can cite SUIF [1], Salto [2] and many more. Even if they let room for improvement, these frameworks have speeded up the development of new techniques by making the re-use of existing algorithms easier.
We think that WCET computation techniques have reached such a point [3] and OTAWA is attempt to provide such a generic framework. As other ones, it features some properties that allows a wide range of use: multi-architecture support, genericity, openness, re-usability, extensibility. Yet, OTAWA has been designed using the concepts and the experience provided by existing frameworks in order to avoid a maximum number of design pitfalls.

1 Open Tool for Adaptative WCET Analysis
2 OTAWA is developed in the MasCotTE project supported by the ANR-PREDIT French Research program.
3 Worst-Case Execution Time

OTAWA comes from the needs of an open and generic software usable for our research activities: (1) a tool that may be used to develop new algorithms or new analyses in the WCET estimation field and (2) a software platform allowing the implementation of an adaptive experimental approach for WCET computation. To maximize the benefits of development efforts, we have re-targeted the tool to make it as much generic as possible. This has led to the current implementation of OTAWA.

3. Architecture Abstraction

The foundations of OTAWA are its Architecture Abstraction Layer. It provides support for the multiplicity of hardware platform Instruction Set Architecture (ISA) that exists in embedded systems. It also hides the details of the actual ISA to the upper layer but exposes the hardware information useful to the WCET analyses. This allows the re-usability of the analyses whatever the actual hardware.

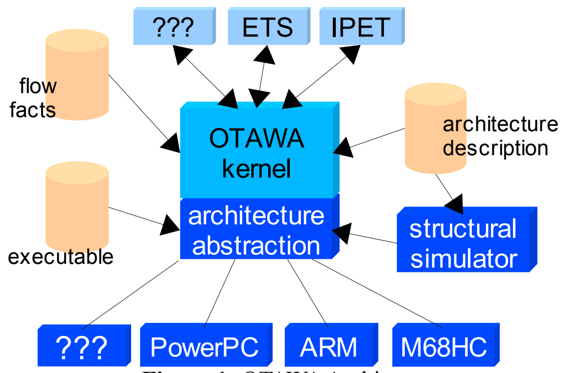

Figure 1: OTAWA Architecture

As shown in Figure 1, this layer is implemented a simple system of plug-ins that makes easier the extension and the integration in the OTAWA framework. Although it is relatively easy to develop a new plug-in for a specific architecture, a significant speed-up can be achieved using an Architecture Description Language like SimNML [4] processed by the GLISS tool suite [5] interfaced to OTAWA.
As many WCET computation methods are based on the simulation of parts of code, this layer also includes a generic processor simulator simply configured through an XML-file description and working for any ISA supported by OTAWA.

3. Properties and Analyzers

Built upon the Architecture Abstraction Layer, the concept of properties provides the main material to build analyses. Indeed, most of the WCET computation methods implement the same steps. First, the flow analysis gathers information about the possible execution paths either from the program

instructions, or from user-provided annotations. In the second step, the temporal properties of the program are computed taking the hardware model into account. Often, this step includes a global analysis phase that handles features like caches and a local analysis phase that examines the behaviour of the pipeline. Finally, the information produced by the flow and temporal analyses is merged to compute the WCET of the program.
A look to previously proposed approaches shows that they have some similarities: (1) they start from a blank program image and each phase improves the obtained information to converge to the WCET; (2) they share a lot of analyses that should be factorized in a common framework.

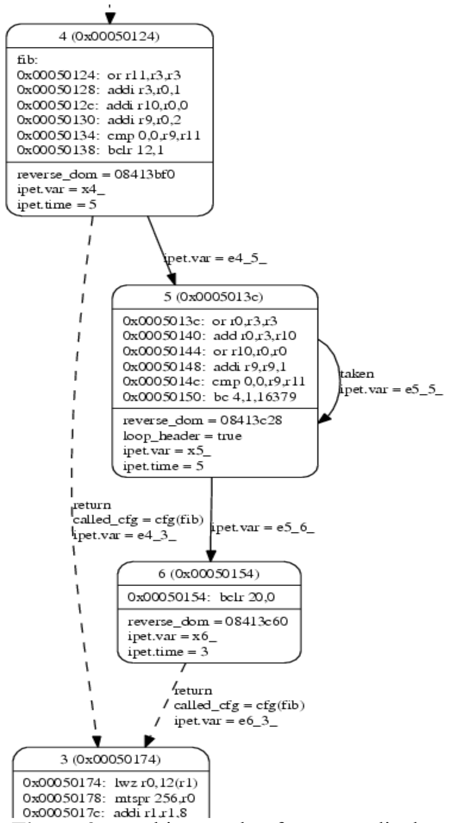

Figure 2: graphic sample of property display.

Based on these two observations, the OTAWA property system is built on two main concepts: properties and analysers. The properties are pieces of information defined by an identifier, a type and a value. They are used to annotate the execution image of the program with information useful to derive the WCET. The analysers use available properties hooked to the program and produce new properties that express their results.
This simple scheme makes easier the interactions between the analyses because they have to share the same properties to be composed. Moreover, it is also very easy to substitute one analysis by one or several other ones, or to get the required information from an external source.
Another useful service coming from the OTAWA property system comes from its standard interface to retrieve the information hooked to the code. This interface allows developing general purpose facilities on properties: textual or graphics display, serialization to disk, property filtering or clean-up, etc. Figure 2 shows a sample produced by a library dedicated to the output of graphs with the attached properties.

4. Achievement Level

Today, the first version OTAWA is about to be distributed with support for the PowerPC and ARM ISAs.
Two main approaches for WCET computation are provided: (1) the Implicit Path Enumeration Technique (IPET) [6] is fully functional and handles several pipeline and cache analyses while (2) only basics of the Extended Timing Schema [7] are available. This last scheme has been developed to prove the versatility of the framework but it will be completed soon.
Generic facilities for the development of new analyses are also provided: Control Flow Graphs, Abstract Syntactic Trees, iterative Data Flow Analysis engine, abstract interpretation engine, Integer Linear Programming solver based on `lp_solve`, flow fact loader, etc. We are also developing a plug-in to integrate OTAWA in the Eclipse environment.

5. Conclusion

OTAWA is a generic framework dedicated to the implementation of static analyses used to compute WCETs. Its original architecture is aimed at speeding up the implementation of new analyses and at promoting the re-use and the interaction among different analyses. One of the goal of OTAWA was to fulfil the lack of open-source tool in the WCET domain and we hope that its extensibility and openness features might also stimulate a community of developers looking to share their programs.
In the future, we plan to extend OTAWA with more ISA supports (like M68HC processors), more output facilities and analyses for new hardware features (like data cache and branch prediction).

6. References

[1] R.P. Wilson, R.S. French, C.S. Wilson, S.P. Amarasinghe, J.M. Anderson, S.W.K. Tijang, S.-W. Liao, C.-W. Tseng, M.W. Hall, M.S. Lam, J.L. Hennessy. *SUIF: An Infrastructure for Research on Parallelizing and Optimizing Compilers*. ACM SIGPLAN Notices, V29, N12, 1994.
[2] R. Rohou, F. Bodin, A. Seznec, G. Le Fol, F. Charot, F. Raimbault. *Salto : System for Assembly-Language Transformation and Optimization*. Technical Report RR-2980, INRIA, 1996.
[3] J. Engblom, A. Ermedahl, M. Sjodin, J. Gustafsson, and H. Hansson. *Towards industry-strength worst case execution time analysis*. Technical Report ASTEC 99/02, April 1999.
[4] M. Freericks, *The nML Machine Description Formalism*, TU Berlin Computer Science Technical Report, 1993.
[5] *GLISS*, www.irit.fr/recherches/ARCHI/MARCH/
[6] Y.-T. S. Li, S. Malik, *Performance Analysis of Embedded Software using Implicit Path Enumeration*, Workshop on Languages, Compilers, and Tools for Real-time Systems, 1995.
[7] R.C. Shaw, *Reasoning about time in higher-level language software.* IEEE Transactions On Software Engineering, 15(7):875.889, July 1989.