Schedulability and modular analysis: how to fit timing model?

Hugues Casse[†], Claire Maiza[*], Pascal Raymond[*]
and Catherine Vigouroux[*]

In real-time systems, program execution must fulfill timing constraints and respect deadlines. Schedulability analysis checks the feasibility of a schedule according to those timing constraints. In such analysis, a task i is abstracted by a *timing model*, typically (at least) made of a period T_i, a deadline D_i and a bound on the execution time C_i also named worst-case execution time (WCET).

In real-time systems (and more generally in software engineering), the trend is to reuse code: a function is designed to be used in different contexts. For instance, a function that depends on the velocity of a car may accept as parameter a range from 0 to 350 km/h. This kind of information (range) may influence the execution time and thus is taken into account by the WCET analysis. However, the constructor knows that the actual range depends on the vehicle: for instance, from 0 to 135 km/h for a smart ForTwo or from 0 to 315 km/h for a Nissan GTR. In this case, the estimated WCET may be largely overestimated if the velocity variable has a large influence on the executed path. For example, a large part of code may become dead code if the velocity never exceeds 200km (for instance, a condition "if ($vel > 200$) then XXX" where XXX is a very heavy block of instruction due to some calls to mathematical libraries).

In this context of dynamically-dead code, there is a need of modular and contextual analysis:

- analysis function by function (vs. for the whole application),
- analysis that takes into account a refined execution context (vs. one large range for each parameter).

However, as far as we know, there is no work related to this topic. Some modular analysis have been introduced, but only to refine the hardware analysis [3, 1, 2, 4]. Some infeasible path analyses exist, but only to refine the general execution path and does not take contextual information into account [7, 5, 6].

This problem could be solved by combining modular analysis, infeasible path analysis and a contextual analysis. The main difficulties of such an analysis are:

- express the context to be used in WCET analysis: what kind of context (range, one value, dynamic context)?

[*]Université Grenoble-Alpes, first.name@imag.fr
[†]Université Toulouse, hugues.casse@irit.fr

- transfer the context to the analysis level: WCET analysis is derived on the binary level, but the developer knows C variables,
- analyse each function (or module) depending on the context,
- combine all modular analysis to compute the whole "context-aware" WCET,
- would the schedulability analysis and scheduling be able to integrate such bounds?

References

[1] Clément Ballabriga and Hugues Cassé. Improving the WCET computation time by IPET using control flow graph partitioning. In *International Workshop on Worst-Case Execution Time Analysis (WCET), Prague, 01/07/2008-01/07/2008*, pages 19–27, http://www.ocg.at, juillet 2008. Austrian Computer society.

[2] Clément Ballabriga, Hugues Cassé, and Marianne de Michiel. A Generic Framework for Blackbox Components in WCET Computation (regular paper). In *Workshop on Worst-Case Execution Time Analysis, Dublin, 30/06/2009*, volume 252, pages 118–129, http://www.ocg.at, octobre 2009. Austrian Computer society.

[3] Clément Ballabriga, Hugues Cassé, and Pascal Sainrat. An improved approach for set-associative instruction cache partial analysis. In *Annual ACM Symposium on Applied Computing (SAC), Fortaleza (Brazil), 16/03/2008-20/03/2008*, pages 360–368, http://www.acm.org/, mars 2008. ACM. TA = 10/33.

[4] Marianne de Michiel, Armelle Bonenfant, Clément Ballabriga, and Hugues Cassé. Partial Flow Analysis with oRange (short paper). In *ISoLA Symposium On Leveraging Applications of Formal Methods, Verification and Validation, Heraklion, 18/10/2010-20/10/2010*, number 6416 in LNCS, pages 479–482, http://www.springerlink.com, octobre 2010. Springer.

[5] Jan Gustafsson, Andreas Ermedahl, Christer Sandberg, and Björn Lisper. Automatic derivation of loop bounds and infeasible paths for WCET analysis using abstract execution. In *RTSS*, 2006.

[6] Niklas Holsti. Computing time as a program variable: a way around infeasible paths. In *WCET*, 2008.

[7] Ingmar Stein and Florian Martin. Analysis of path exclusion at the machine code level. In *WCET*, 2007.