Program Semantics in Model-Based WCET Analysis

--- State of the Art Perspective ---

Mihail Asavoae  Claire Maiza  Pascal Raymond

WCET: SEmantics, Precision and Traceability

WCET 2013 - PARIS
Key Point! - Advocate for WCET analysis based on the program semantics

(1) Model-Based Design

+ 

(2) WCET Analysis
“This spirit is in full accordance with modern thinking about system design, sometimes called model-based design, which advocates the use of high-level models with “ideal”, implementation-independent semantics, and the separation of concerns between design and implementation.”

“Synchronous Programming” - P. Caspi et al. in Handbook of Real-Time and Embedded Systems - 2008
Synchronous Programming

- Generic programming paradigm - designed to generate code

  - deterministic concurrent programming style
  - in data-flow (e.g. Lustre), control-flow (e.g. Esterel) flavors, or both (e.g. Scade 6 - industrial version of Lustre + finite state machines)

- Lustre

  - semantics - functions over flows
  - style - network of operating nodes (similar to sequential circuits or block diagrams)

  - Scade - in critical systems: transportation, energy
A Typical Control System

**Design**

- overall subsystems (A, B)
- subsystem inter-dependent data flows
- dependencies direct and logical (delays, i.e. Reg)
- clocks (clkA and clkB)

**Elements**
Typical Control System

**Design**

- instantaneous execution

**Elements**

→ “… the separation of concerns between design and implementation.”

**Design:**
- instantaneous execution

**Implementation:**
- bounded execution (with a **guaranteed upper bound on time**)

"the separation of concerns between design and implementation."
Typical Compilation

**Design**

- scheduling → \( \text{A} \) before \( \text{B} \)

- code →

**C Code**

```c
global Reg;

if clkA then \text{A}_\text{Step}();
if clkB then {
    \text{B}_\text{Step}();
    Reg = \text{B}.out;
}
```

- Deterministic, traceable C code (heap-free, no recursion, bounded loops)
MBDs through Synch.

- Typical Synchronous Program + Compilation

Program Semantics Levels

HL → C → bin

Transfer Semantics Levels

- From the WCET analysis perspective: both global and local interest
- Generated code from model-based designs:
  → subsystems
  → inter-dependent
  → **clocks (Lustre)**

- Assume a relation on the nodes: at most one of \( A, B, C \) is not executed.

[Diagram showing relationship between nodes A, B, C with clocks clkA, clkB, clkC and a scheduling analysis arrow towards an 'X'.]
- Two levels of interest:

- **HL to C**: it features a higher degree of control over the model semantics transformation

- **C to bin**: compilation with optimizations
- identify relations between different representations

```c
.C FILE
if clkA then A_Step()
if clkB then B_Step()

A_Step() {
    d = ...
}

B_step() {
    e = ...
}
```
Optimizations

- transfer of semantics through optimizations

if $clkA$ then $A\_Step()$
if $clkB$ then $B\_Step()$

$A\_Step()$ {
    $d = ...$
}

$B\_step()$ {
    $e = ...$
}

- obfuscation of the control structure

- no clear solution, in general optimizations are handled in a case by case manner

- specialized annotation language

- transfer functions
Global MBD + WCET Analysis

- Integration methods

HL → C → bin

Information Flow

Timing Analyzer

- seamless integration vs modified MBD infrastructure
Path Analysis

- the path analysis determines the worst-case execution time based on previously computed (or annotated) artifacts

- represented outside MBD

- IPET

- alternatives: on the program syntax, on model checking

Is IPET the best approach to represent the semantics?

- separation of concerns at the level of WCET analysis workflow
- IPET - path analysis through ILP constraint solving

- at most one of \( A, B, C \) is not executed

\[
X_A + X_B + X_C \leq 2
\]

and

ILP representation of the control structure

- IPET enhancements are possible
Program Semantics Levels

HL → C → bin

Transfer Semantics Levels

SEMANTICS INFORMATION

WCET Analyzer
Conclusions

- The paper presents a state-of-the-art overview of approaches for both local and global w.r.t. MBD workflow, from the WCET analysis perspective.

- Particular MBD: synchronous programs as models and C code generation capabilities.

- The presentation (complements the paper)

  - Follows several aspects of the synchronous programming.

  - Advocates for a separation of concerns at the level of WCET analysis.
The WCET analysis of a particular program is performed at the binary level and with knowledge about the underlying architecture. As it is summarized in Figure 1 and shown in Figure 1(A), a typical workflow for a WCET analysis proceeds with the CFG extraction, a number of program flow- and processor-behavior analyses, and finally, the bound computation. The WCET analysis should provide safe and tight estimations of the actual WCET of a program. To address these, the WCET analysis workflow relies on a number of specific analyses, spawn from both the flow analysis (i.e. detection of loop bounds and infeasible paths) and the architecture analysis (i.e. cache and pipeline behavior prediction).

In this paper we present a survey study on how the WCET analysis workflow is projected on the MBD workflow, from a particular point of view - the separation of concerns at the level of program semantics manipulation. Due to the generality of the MBD framework and the multitude of contributions in the WCET analysis, as well as the current space limit, we restrict our presentation under a setting defined by the following constraints. First, we consider MBDs where the model is compiled into C code. Second, we consider the architecture-related analyses to be orthogonal to our investigation on the program semantics and thus are left out, as shown in Figure 1(A). This second restriction activates other intended omissions, from our survey: analyses for CFG extraction and for classification of load/store instructions. Third, we discuss the path-analysis problem from the popular implicit path-enumeration technique (IPET) point of view, classifying the approaches as enhancements or alternatives to it.

The works in [37, 56] survey, from certain angles, the state-of-the-art approaches in the WCET analysis field of research. The authors of [37] rely on the notion of the flow fact and classify then-existing WCET analysis approaches w.r.t. this notion. As a consequence, this allows comparisons between various approaches at the confluence of axis for the representation levels and the execution-time modeling. The survey in [56] is ampler and newer than [37], covering both the methods and the existing tools in the WCET analysis field of research. The methods are classified into static and measurement-based ones, with the information...