Techniques to Calculate the Worst-Case Execution Time

Peter Puschner

slides: P. Puschner, R. Kirner, B. Huber
VU 2.0 182.101 SS 2015
Static WCET Analysis

Goal: calculation of safe upper bounds of the worst-case execution time

Quality of Analysis:
- considered control flow paths
- modeled instruction timing

Selection of WCET calculation method:
- tradeoff between quality/strength and performance/simplicity
Aspects of WCET Analysis

Representation Level

Flow Facts

Exec-Time Modeling
Aspects of WCET Analysis

Representation Level

Matlab/Simulink
C source code
Executable binary

Stateflow
Assembly code

Flow Facts
Exec-Time Modeling
Aspects of WCET Analysis

Representation Level

- Exec-Time Modeling
  - manual
    - only loop bounds
    - heuristics
    - data flow analysis
    - abstract interpretation
    - symbolic analysis
    - very detailed
  - automatic

Flow Facts
Aspects of WCET Analysis

Representation Level

Flow Facts

- simple
- branch prediction
- caches
- pipeline

Exec-Time Modeling
Flow Facts

In general, automation is impossible (intractable due to the large state space of real systems)

Some information can be extracted automatically
  • abstract interpretation
  • simulation

➤ Program constructs, annotations, interactive input of path constraints
  ➤ to derive flow facts for the specific WCET calculation method.
Representation Level

Different representation level between program development and WCET analysis:

- precise **timing analysis** has to be done after all program transformations, based on **executable**.
- **automatic extraction** of flow facts and **manual flow facts annotations** preferred at source-code level.

Flow facts have to be mapped between different program representation levels.

**Formal correctness** of this mapping has to be shown for structure-changing program transformations.
Representation Level

Mapping of flow facts may require support by the code transformer:
→ ambiguous mapping between original and transformed code.

Support by code transformer

• emit execution history
  (flow facts update by an external tool)

• perform flow facts transformation in parallel to code transformation.

Flow facts transformation should be extensible.
Representation Level

WCET analysis of Matlab/Simulink models:
Exec-Time Modeling

Assigning an execution time to a sequence of code.

This assignment is typically context-sensitive (depends on call context, execution history)

Execution time of instructions can vary with the value of their parameters.
→ solution: use maximal execution time

May also depend on internal state of processor (execution history)
Generic WCET Analysis Framework

source code → Compilation → Extraction of Flow Facts → Transformation of Flow Facts → Exec-Time Modeling → Calculation of Execution Scenarios → WCET

object code → Compilation → Extraction of Flow Facts → Transformation of Flow Facts → Exec-Time Modeling → Calculation of Execution Scenarios → WCET
Tree-Based WCET Calculation

Also called “timing schema”

Bottom-up traversal of syntax tree.

Using rules to compute timing of compound program statements.
Tree-Based WCET Calculation

for (i=0; i<N; i++)
{
    ...
}

if (a==5)
{
    ...
}
else
{
    ...
}

T(for) = (LB+1)\cdot T(test) + LB \cdot T(body)

T(if) = T(test) + \max( T(then), T(else))
Tree-Based WCET Calculation

Advantages:
• Simple method with low computation effort.
• Scales very good with program size.

Drawbacks:
• Does not allow to consider generic flow facts in a direct way.
Path-Based WCET Calculation

Calculate times for different paths in a program.

**Path**: Instructions that can be executed during a single iteration of a loop.

Different paths produced by conditional statements.

**Flow facts** can be used to prune set of valid paths.
(by limiting exec. frequency of particular paths)
Path-Based WCET Calculation

```c
int a;
...
for (i=0; i<N; i++)
{
...
for (j=0; j<i; j++)
{
  ... 
  if (a==5) {
    ... 
  } else {
    ... 
  }
}
}
```
Path-Based WCET Calculation

```c
int a;
...
for (i=0; i<N; i++)
{
...
for (j=0; j<i; j++)
{
    if (a==5) {
...
    }
else {
...
}
}
```
Path-Based WCET Calculation

```c
int a;
...
for (i=0; i<N; i++)
{
    ...
    for (j=0; j<i; j++)
    {
        if (a==5) {
            ...
        }
        else {
            ...
        }
    }
}
```
Path-Based WCET Calculation

```c
int a;
...
for (i=0; i<N; i++) {
    ...
    for (j=0; j<i; j++) {
        ...
        if (a==5) {
            ...
        }
        else {
            ...
        }
    }
}
```
Path-Based WCET Calculation

Advantages:

• Scales relatively good with program size.
• Allows simple integration of pipeline modeling.

Drawbacks:

• Exponential complexity with depth of conditional statements.
• Allows only to consider flow facts relative to surrounding loop in a direct way.
WCET Calculation using IPET

IPET...Implicit Path Enumeration Technique

Program flow graph is mapped into a set of graph flow constraints.

Uses methods like integer linear programming (ILP) or constraint-solving to calculate the WCET.

WCET: optimization/maximization problem

- Maximize goal function describing execution time under
- a set of constraints describing possible paths (characterize graph structure, semantics, and context)
WCET IPET: goal function

Program

WCET: maximize $\sum x_i \cdot t_i$

- $x_i$ … execution frequency of CFG edge $e_i$
- $t_i$ … execution time of CFG edge $e_i$
WCET IPET: constraints

**Program**

**Graph flow constraints:**

\[
\begin{align*}
x_1 &= 1 \\
x_1 + x_8 &= x_2 \\
x_2 &= x_3 + x_4 \\
x_3 &= x_5 \\
x_4 &= x_6 \\
x_5 + x_6 &= x_7 \\
x_7 &= x_8 + x_9 \\
x_2 &\leq LB \times x_1
\end{align*}
\]
WCET Calculation using IPET

Result: WCET bound plus a variable setting for $x_i$

Advantages:
• Description of complex flow facts is possible.
• Generation of constraints is quite simple.
• Constraints can be solved by existing tools.

Drawbacks:
• Solving ILP is in general NP hard.
• Flow facts that describe execution order are difficult to integrate.
Alternatives to Static WCET Analysis

Static WCET analysis methods have their limitations on:

- generic control flow
- complex hardware

Dynamic WCET analysis by measurements:

“The processor is the best hardware model”

Test data generation instead of path analysis.

Use of problem-specific hardware/software architectures
Summary

Aspects of WCET Analysis

Different WCET calculation methods as a tradeoff between strength and simplicity/performance.

Flow facts are required for static analysis.

Limitations of static WCET analysis.
http://ti.tuwien.ac.at/rts/teaching/courses/wcet

http://www.wcet.at