Techniques to Calculate the Worst-Case Execution Time

Peter Puschner
Outline

Aspects of WCET Analysis
Flow-fact modeling
WCET-calculation strategies
Aspects of WCET Analysis

Representation Level

Flow Facts

Exec-Time Modeling
Aspects of WCET Analysis

- Representation Level
  - Matlab/Simulink
  - Stateflow
  - C source code
  - Assembly code
  - Executable binary

- Exec-Time Modeling
  - Flow Facts
Aspects of WCET Analysis

Representation Level

- Flow Facts
  - only loop bounds
  - heuristics
  - data flow analysis
  - abstract interpretation
  - symbolic analysis
  - very detailed

Exec-Time Modeling
- manual
- automatic
Aspects of WCET Analysis

Representation Level

Flow Facts

Exec-Time Modeling

- simple
- branch prediction
- caches
- pipeline
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.
Flow Facts

Loop bounds have to be known (also: recursion bounds, branch targets)

Description of further characteristics improves the quality of WCET analysis

\[
\begin{align*}
\text{for } i := 1 \text{ to } N \text{ do} & \quad \text{loop bound: } N \\
\text{for } j := 1 \text{ to } i \text{ do} & \quad \text{loop bound: } N; \text{ local: } i: 1..N \\
\text{begin} & \\
\text{if } c_1 \text{ then } & \text{A.long} \\
\text{else} & \text{B.short} \\
\text{if } c_2 \text{ then } & \text{C.short} \\
\text{else} & \text{D.long} \\
\text{end}
\end{align*}
\]

\[
\frac{(N+1)N}{2} \text{ executions}
\]
Flow Facts of Interest

Simple Architecture Model
- Information how often actions occur
- Execution-frequency bounds and relations
- Notation: marker, relations, and scopes

Complex Architecture Model
- Information about occurrence order / patterns
- Characterization of (im)possible paths
- Notation: based on regular expressions, IDL
Path Description Example

```cpp
for (i=0; i<N; i++)
{
    if (i % 3 == 0)
    {
        M1
    }
    if (i % 3 != 0)
    {
        M2
    }
}
```

path expression

\[
(M1.M2.M2)^{\lfloor N/3 \rfloor} +
(M1.M2.M2)^{\lfloor N/3 \rfloor} . M1 +
(M1.M2.M2)^{\lfloor N/3 \rfloor} . M1.M2
\]

frequency constraints

\[
f(M1) = \lfloor N/3 \rfloor,
f(M1) + f(M2) = N
\]

frequency constraint + loop context

Iteration <3*k+0..0> : f(M2) = 0
Iteration <3*k+1..2> : f(M1) = 0
f(M1) + f(M2) = N
Markers, Relations and Scopes

SCOPE
{
    for (i=0; i<N; i++)
    {
        MAX_ITERATIONS(N);
        for (j=0; j<i; j++)
        {
            MAX_ITERATIONS(N);
            MARKER(M1);
            ...
        }
    }

    REL(FREQ(M1) == N * (N+1) / 2);
}
Parameterized Path Information

Some applications require situation-specific WCET bounds:

• WCETs for different system states
• Different modes
• Use of libraries
• On-line WCET calculation
  • Instantiation WCET formulas or models at evaluation time
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
{
    ...
}

\[ T(\text{for}) = (LB+1) \cdot T(\text{test}) + LB \cdot T(\text{body}) \]

\[ T(\text{if}) = T(\text{test}) + \max( T(\text{then}), T(\text{else})) \]
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 {
            ...
        }
    }
}
```
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:

\[ x_1 = 1 \]
\[ x_1 + x_8 = x_2 \]
\[ x_2 = x_3 + x_4 \]
\[ x_3 = x_5 \]
\[ x_4 = x_6 \]
\[ x_5 + x_6 = x_7 \]
\[ x_7 = x_8 + x_9 \]
\[ x_2 \leq LB \times x_1 \]
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
Flow-fact characterization
Different WCET calculation methods as a tradeoff between strength and simplicity/performance.

• Tree-based analysis / timing schema
• Path-based analysis
• WCET analysis using IPET
http://ti.tuwien.ac.at/rts/teaching/courses/wcet

http://www.wcet.at