Validating static WCET analysis: a method and its application

Wei-Tsun Sun  IRT Saint Exupéry
Eric Jenn     IRT Saint Exupéry, second by Thales AVS.
Hugues Cassé  IRIT, Université de Toulouse

09/07/2019 @ WCET 2019, Stuttgart, Germany
The Project CAPHCA

An IRT team, funded with ANR
WCET (Worst-Case Execution Time)

Why do we need to care about the worst-case?
Why WCET?

- Know the worst, still schedule-able, then schedule-able for all cases.

- No bad surprises for time-critical parts.

- Safety, and to be Sure.
How to get WCET?

- Static approaches and measurement approaches.

- Add some flavours of statistics.

- Approaches are good, but we need to be Sure.

- Who performs the analysis/measurements?
  - Human
  - Program

- Who wrote the programs?
  - Human
  - Compiler, code-gen

- Who wrote the compiler......?
  - Human......
One of the problems of being human

- Human makes mistakes.

- For WCET analysis, sometimes HW info needed:
  - Looking into device drivers so you know all the details (takes a lot of time)
  - Little bird whispering next to your ears (takes too much imagination)
  - Existing models provided by vendors (takes some luck)
  - Data-sheet (mostly accessible, detailed......too detailed, 5000+ pages)

- Common pitfall and possible remedy
  - WYPMNBWYG

- Need to find a way to detect the mistake
  - Comparisons

What you program may not be what you get
Checking the WCET tool

To make sure we are on the right track
We need to make sure the estimated WCET is trust-worthy

- WCET estimation from WCET tool.
- Need to check WCET tool to make sure:
  - It is constructed correctly
  - Its implementation satisfies the theory-to-implement, e.g. abstract interpretation

- Our case:
  - WCET tool to check – OTAWA
    - From IRIT, University of Toulouse
    - Open source
    - Support ARM, RISC-V, PowerPC, Kalray MPPA, and our interested target: Infineon TriCore

- Infineon TriCore is one of our target because:
  - Three CPUs
  - Targeted for automobile
  - Powerful co-processors and peripherals
  - Good candidate for REAL case-study, not just toy examples
  - Interference analysis
  - Many other tools support it, make a good case for tool-chain integration
What does OTAWA do?

- OTAWA takes the program binary as the input
- Provides WCET (in CPU cycles)
OTAWA is a huge framework of many years work

- OTAWA is sophisticated

Binary → OTAWA → WCET

OTAWA

- Decode the binary
- Create CFGs and BBs
- Static analyses
- Generate and solve the ILP formula

Hardware info

Constraint building
ILP creation
ILP solver invocation
Flow facts

Identify Instructions
Create CFGs & BBs
Detect loops
Caller relations
Value/ address analysis
Prog/Data cache analysis
Branch prediction

© IRT AESE 2015 — All right reserved Confidential and proprietary document.
To support a new architecture by OTAWA

- To be able to decode the binary
- To have static analyses available (program cache analysis, data cache analysis, …)

OTAWA

Binary

Decide the binary

Create CFGs and BBs

Static analyses

Generate and solve the ILP formula

Hardware info

Constraint building

ILP creation

ILP solver invocation

Flow facts

Identify Instructions

Create CFGs & BBs

Detect loops

Caller relations

Value/Address analysis

Prog/Data cache analysis

Branch prediction

WCET
Analyses are platform independent and can be configured

- Analyses in OTAWA are made
  - Platform independent, so developer can focus on the analysis itself
  - Each instruction is presented by a sequence of **semantic instructions** [1]
    - e.g.
      - add r1, r2, 3
      - set t1, 3
      - add r1, r2, t1

- This is done before analyses

---

We want to check

- The binary decoding and semantic instructions
- The static analyses

Analyses are platform independent and can be configured

OTAWA

- Decode the binary
- Create CFGs and BBs
- Static analyses (sem. Instruction representation)
- Generate and solve the ILP formula
- WCET

- Identify Instructions
  - Value/address analysis
  - Prog/Data cache analysis
  - Branch prediction

Binary
Validation of the ISA model for the instruction decoding
The binary decoding

- The ISA is described in NMP format [2]
- Normally processor has user-manual for ISA [3]

```
MOV D[c], D[b] (RR)
```

| 31 | 28 | 27 | 20 | 19 | 18 | 17 | 16 | 15 | 12 | 11 | 8 | 7 | 0 |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| c  | 1FH| -  | b  | -  | -  | 0BH|

```
D[c] = D[b];
```

```
op mov_reg (c:reg_d, b:reg_d)
image = format("%4b %8b XXXX %4b XXXX %8b",c.image,0x1F,b.image,0x0B)
syntax = format("mov %s,%s",c.syntax,b.syntax)
action = { c = b; }
```


The binary decoding

- The ISA is described in NMP format [2]
- Normally processor has user-manual for ISA [3]

```
MOVD[c], D[b] (RR)
```

<table>
<thead>
<tr>
<th>31</th>
<th>28</th>
<th>27</th>
<th>20</th>
<th>19</th>
<th>18</th>
<th>17</th>
<th>16</th>
<th>15</th>
<th>12</th>
<th>11</th>
<th>8</th>
<th>7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td></td>
<td></td>
<td></td>
<td>1F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>b</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
D[c] = D[b];
```

```python
op = mov_reg (c:reg_d, b:reg_d)
image = format("%4b %8b XXXX %4b XXXX %8b", c.image, 0x1F, b.image, 0x0B)
syntax = format("mov %s,%s", c.syntax, b.syntax)
action = { c = b; }
```

- The description of ISA is captured manually – can be prone to human error!
- The GLISS2 can not found problems like wrong op code value
- 815 OPCODE-OPERAND combinations

Checking the instruction decoding / interpretation

- Official ISS from Infineon: TSIM
- OTAWA is able to generated ISS from NMP files
  - Describing ISA
  - Use to decode instructions in OTAWA
  - First step to support a new architecture
- Compare the “processor state” between TSIM and OTAWA-ISS
  - Register values
  - Memory accesses
Some results

- **815 combinations of Instructions + operands**
  - 203 (25%) were covered.
  - ~200,000,000 instructions were executed in 36 applications including twlRTee and dual-eMotor use cases.
  - ~20% of OTAWA’s TriCore instruction of decoding were corrected.
  - Mostly human error when writing the NMP file.
Validation of static analyses and semantic instruction mappings
Validate the static analyses

- Similarly we check for other analyses in OTAWA
- For example abstract interpretation in value analysis
Capturing the semantics of the instructions

- Extended the previous crafted ISA-NMP

```plaintext
op mov_reg (c:reg_d, b:reg_d)
image = format("%4b %8b XXXX %4b XXXX %8b", c.image, 0xF, b.image, 0x0B)
syntax = format("mov %s,%s", c.syntax, b.syntax)
action = { c = b; }
```

```plaintext
extend mov_reg
sem = { SET(D(c.i), D(b.i)); }
```
Abstract state: covers the concrete state

- Checks for two things:
  - If the value/address analysis is correctly implemented
  - If the semantic instruction is correctly implemented for TriCore

Step 1
R1 = 3

Step 2
R1 = {3}

Concretely, actual states

Abstraction

Comparison

Abstract states

Step 3
R1 = {3, 4, 5}

Static WCET Analysis

Binary + stimuli

Address and value analysis

Micro-architecture analyses

Processor model
Some results

- Correct some operations in the abstraction
  - Validation of the tool is also important to have sound results
- Some aspects were not taken into account in the previous TriCore implementation
  - So far, we are contributing significantly for the TriCore support in OTAWA
Overall validation procedures

1. NMP - kinds - targets
2. NMP - semantic instructions
3. GLISS2
4. Binary
5. ISS
6. IRG
7. GLISS2
8. OTAWA
9. Abstract state on BBs
10. Concrete state on BBs
11. Comparison

Verification of the abstract states
Verification of the processor model

 Semantic instruction translator
Address and value analysis
Other micro-architecture analyses

© IRT AESE 2015 – All right reserved Confidential and proprietary document.
Future works
Next steps

- **Finish what we did**
  - Complete support of TriCore for OTAWA

- **Linkage to other our works**
  - Already have support for RISC-V
  - Extended to support FlexPRET

- **Possible track**
  - Generate semantic instructions from the ISA descriptions
Questions?