Improving Trace-Based JIT Optimisation Using Whole-Program Information
Vandercammen, Maarten; De Roover, Coen

Publication date:
2016

Link to publication

Citation for published version (APA):

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.
Improving Trace-Based JIT Optimisation using Whole-Program Information

Maarten Vandercammen, Coen De Roover

Software Languages Lab, Vrije Universiteit Brussel, Belgium

Maarten Vandercammen
Coen De Roover

@mvdcamme

maarten.vandercammen@vub.ac.be

https://github.com/mvdcamme/scala-am

1) Analysis Launch Point?
2) Extent of Scope?

---

**Program**

```python
1 function f(a) {
2   var b = a > 0 ? 1 : 2;
3   return a + b;
4 }
5
6 var n = 10;
7 var result = 0;
8 var a = user_input();
9 function loop() {
10     while (n > 1) {
11         result += f(a);
12         n -= 1;
13     }
14 }
15 loop();
16 a = 10;
17 loop();
```

**Execution Trace**

```c
loop:
  LOAD n
  CMP_CONST 1
  GUARD_TRUE
  LOAD a
  CMP_CONST 0
  GUARD_FALSE
  LOAD result
  ADD_INT
  LOAD n
  DEC
  JUMP loop
```

**Example**

<table>
<thead>
<tr>
<th>Analysis</th>
<th>Constant Variables Found</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>(a)</td>
</tr>
<tr>
<td>RT 1</td>
<td>(a, b)</td>
</tr>
<tr>
<td>RT 2</td>
<td>(a, b, c)</td>
</tr>
</tbody>
</table>

---

**Trace-based JIT Compilation**

- Precise dynamic information
- Run-time analysis
- Local analysis
- Trace optimisations

**AOT Compilation**

- Imprecise static information
- Compile-time analysis
- Whole-program optimisation

**My Approach**

- Hybrid:
  - Refine static information with observed information
  - Time: Run-time + Compile-time analysis
  - Scope: Local + whole-program analysis

**Trace Optimisations**

- Eliminated
- Extended

---

**Refinement**

- Compile time:
  - Initial Analysis
  - 1st run-time analysis
  - 2nd run-time analysis
  - Compute heavyweight initial analysis
  - Refine initial analysis with run-time information
  - Further refine run-time analysis with extra run-time information

---

**Future Work**

1) Analysis Launch Point?
2) Extent of Scope?