Context-sensitive Pointer Analysis with Cycle Elimination and Hash-consing

Woongsik Choi
KAIST

ROSAEC 3rd Workshop
2010. 1. 7
Cycle Elimination

- $x_1 \supseteq x_2 \supseteq \cdots \supseteq x_n \supseteq x_1$
  - All $x_1 \sim x_n$ have same solution
  - They can be merged into single representative
  - No precision is lost

- Pointer analysis
  - New edges are added as more pointer values get known
  - Dynamic transitive closure

- Online cycle elimination for pointer analysis
  - Many approaches for context-insensitive pointer analysis
  - No attempt for context-sensitive pointer analysis
Cloning-based Context-sensitivity (1/3)

- Get analysis result same as fully inlined program
- Infinite for recursive program
  - No inlining for recursive calls
  - Exponential but finite
- Fully Binary Decision Diagram (BDD) based approach
  - “Cloning-Based Context-sensitive Pointer Alias Analysis using BDD” by Whaley and Lam
  - Redundancy sharing through BDD
  - Encode contexts and constraints with BDD
  - Not ideal for cycle elimination
Cloning-based Context-sensitivity (2/3)

- Fully BDD-based approach
  \[
  \frac{X \supseteq_c Y \quad Y \supseteq_c a}{X \supseteq_c a}
  \]
  - Separate constraint for each context
  - Exponential constraints are represented by BDD

- Restrict sharing technique only for context
  \[
  \frac{X \supseteq_{\rho_1} Y \quad Y \supseteq_{\rho_2} a}{X \supseteq_{\rho_1 \cap \rho_2} a} \quad \rho_1 \cap \rho_2 \neq \bot
  \]
  - Explicit set of context
  - BDD-encoded context is not a must
Cloning-based Context-sensitivity (3/3)

- Context-sensitivity only for address-not-taken locals
  - Exclusively belong to a function
  - Inter-procedural effect only through parameter, return

\[
\frac{\mathcal{X} \supseteq_{\rho_1} \mathcal{Y} \quad \mathcal{Y} \supseteq_{\rho_2} a}{\mathcal{X} \supseteq a} \quad \neg cs(\mathcal{X}) \land \rho_1 \cap \rho_2 \neq \bot
\]
Invocation Graph

Program

```c
void m() {
    f()^{1};
    f()^{2};
}

void f() {
    g()^{3};
    g()^{4};
}

void g() { }
```

Call graph

```
    m
   / \      
  1   2
```

Invocation graph

```
    m
   / \      
  1   2
```

Reversed graph for f

```
    f
   / \      
  3   4
```

Reversed graph for g

```
    g
   / \      
  3   4
```

Full set of context for f

```
    f
   / \      
  m   m
```

Full set of context for g

```
    g
   / \      
  m   m
```


Context Representation

- First child/next sibling binary tree
  - Encodes arbitrary forest

- Sort sibling on node label
  - Siblings represent union of contexts

- Explicit $\top$ for full set of context
  - $\top$ is polymorphic over arbitrary function
Hash-consing

- Sharing technique for recursive data-structure
  - Every constructed term is stored in *identity* hash table
  - Every construction of term goes through hash table
    - Reuse existing term whenever found
    - Structurally equal terms are physically shared

- Effective caching of operations
Cycle Elimination in Context-sensitive Analysis

- \( x_1 \supseteq_{\rho_1} x_2 \supseteq_{\rho_2} \cdots \supseteq x_n \supseteq_{\rho_n} x_1 \)
  - Form cycle only for \( \rho_1 \cap \rho_2 \cap \cdots \cap \rho_n \)

- Eliminate only \( T \)-cycles
  - \( x_1 \supseteq_T x_2 \supseteq_T \cdots \supseteq x_n \supseteq_T x_1 \)

- Analysis is tuned to find \( x \supseteq_T y \) as much as possible
  - Especially with context-insensitive (CI) variables

- If one of \( x_i \) is CI, all \( x_i \) become CI
  - More opportunities for cycle elimination
## Experimental Results

<table>
<thead>
<tr>
<th>Program</th>
<th>Line</th>
<th>HC-CE</th>
<th>HC-NoCE</th>
<th>BDD-CE</th>
<th>BDD-NoCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>make</td>
<td>25K</td>
<td>0.2</td>
<td>0.8</td>
<td>0.5</td>
<td>1.3</td>
</tr>
<tr>
<td>m4</td>
<td>35K</td>
<td>1.6</td>
<td>16.4</td>
<td>1.7</td>
<td>16.7</td>
</tr>
<tr>
<td>gawk</td>
<td>50K</td>
<td>1.0</td>
<td>3.9</td>
<td>1.2</td>
<td>4.3</td>
</tr>
<tr>
<td>tar</td>
<td>56K</td>
<td>1.8</td>
<td>17.2</td>
<td>2.3</td>
<td>18.0</td>
</tr>
<tr>
<td>vortex</td>
<td>63K</td>
<td>0.7</td>
<td>1.5</td>
<td>9.8</td>
<td>30.6</td>
</tr>
<tr>
<td>sqlite</td>
<td>65K</td>
<td>36.5</td>
<td>156.9</td>
<td>116.4</td>
<td>302.1</td>
</tr>
<tr>
<td>gap</td>
<td>78K</td>
<td>1.9</td>
<td>10.6</td>
<td>2.2</td>
<td>4.6</td>
</tr>
<tr>
<td>povray</td>
<td>82K</td>
<td>11.3</td>
<td>52.0</td>
<td>37.3</td>
<td>104.0</td>
</tr>
<tr>
<td>perlbmk</td>
<td>90K</td>
<td>4.9</td>
<td>40.2</td>
<td>5.3</td>
<td>41.3</td>
</tr>
<tr>
<td>bash</td>
<td>127K</td>
<td>12.3</td>
<td>52.3</td>
<td>75.6</td>
<td>183.5</td>
</tr>
<tr>
<td>cfengine</td>
<td>129K</td>
<td>24.0</td>
<td>58.0</td>
<td>52.7</td>
<td>188.0</td>
</tr>
<tr>
<td>sshd</td>
<td>137K</td>
<td>959.0</td>
<td>11884.0</td>
<td>1090.0</td>
<td>13400.1</td>
</tr>
<tr>
<td>python</td>
<td>164K</td>
<td>413.5</td>
<td>1458.6</td>
<td>2555.0</td>
<td>4207.2</td>
</tr>
<tr>
<td>vim</td>
<td>224K</td>
<td>20.6</td>
<td>102.0</td>
<td>24.8</td>
<td>111.4</td>
</tr>
<tr>
<td>gcc</td>
<td>228K</td>
<td>3.1</td>
<td>4.1</td>
<td>112.9</td>
<td>68.1</td>
</tr>
</tbody>
</table>