Heap Abstraction for Visualization

Raoul Veroy
Understanding software

- Lines of code
Understanding software

- Lines of code
- Reusable software components
Understanding software

- Lines of code
- Reusable software components
- Legacy code
Understanding software

- Lines of code
- Reusable software components
- Legacy code

Challenging!
Related Work

- Unabstracred graph representation
Pheng and Verbugge (2006)
Related Work

- Unabtracted graph representation
- Data analysis represented by line graphs/ bar graphs
FIGURE 1. Sample memory visualization.

Reiss (2009)
Related Work

- Unabstracted graph representation
- Data analysis represented by line graphs/bar graphs
- Representations to identify memory problems
FIGURE 1. Sample memory visualization.

Reiss (2009)
Related Work

- Unabstracted graph representation
- Data analysis represented by line graphs/bar graphs
- Representations to identify memory problems
- Heap graph visualization using an abstraction framework
Heapviz


- Heapviz: Interactive Heap Visualization for Program Understanding and Debugging.
  - In *Proceedings of the 5th ACM Symposium on Software Visualization (SOFTVIS)*, Salt Lake City, Utah, October 2010.
  - http://www.cs.tufts.edu/research/redline/heapviz/
Heapviz

• **Interactive** tool and framework for analyzing and visualizing heap snapshots from running Java programs
Heapviz

- Interactive framework for analyzing and visualizing heap snapshots from running Java programs
- Represent heap snapshots as directed graphs
- Uses a graph summarization algorithm to reduce the size and complexity of the graph
Heapviz

- Interactive framework for analyzing and visualizing heap snapshots from running Java programs
- Represent heap snapshots as directed graphs
- Uses a graph summarization algorithm to reduce the size and complexity of the graph
- Existing graph summarization algorithm fails on certain real-world benchmarks
  - SPEC JBB
  - Problem of scale
Heapviz - Spec JBB

Aftandilian et al, 2010
## Heapviz Results

<table>
<thead>
<tr>
<th></th>
<th>vertices</th>
<th>edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original heap</td>
<td>119,099</td>
<td>174,909</td>
</tr>
<tr>
<td>Heapvis 2010</td>
<td>7,583</td>
<td>12,735</td>
</tr>
</tbody>
</table>
Heapviz pipeline
Focus on summarization
Proposed Abstraction

Type graph

All objects of type $t$ is represented by a single vertex in the type graph.

Let $v_1$ be of type $t_1$

$v_2$ be of type $t_2$

s.t. there is an edge from $v_1$ to $v_2$.

Then there is an edge from $T_1$ to $T_2$. 
Type Graph

In the type graph:

- Multiple objects of same type map to a single vertex.
- Multiple edges can also map to a single edge.
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>Vertices</th>
<th>Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original heap</td>
<td>119,099</td>
<td>174,909</td>
</tr>
<tr>
<td>Heapvis 2010</td>
<td>7,583</td>
<td>12,735</td>
</tr>
<tr>
<td>Type Graph</td>
<td>251</td>
<td>503</td>
</tr>
</tbody>
</table>
Problem with Type Graph
Original

Type Graph
Object Signature

Definition:

- The **signature** of an object is the set of types of objects that the object is linked to.
Object Signature

Definition:
- The **signature** of an object is the set of types of objects that the object is linked to.
- The **out signature** is the set of types of objects that the object points to.
- The **in signature** is the set of types of objects that points to the object.
Type Signature

Definition:

- The **signature** of a type is the set of all the signatures of the concrete objects of that type.
- The **out signature** of a type is the **set** of all the out signatures of the concrete objects of that type.
- The **in signature** of a type is the **set** of all the in signatures of the concrete objects of that type.
spec.jbb.Stock

spec.jbb.Company

char[]

java.lang.String

Stock Out Signature
Algorithm Option 1

- Convert a heap graph to a type graph.
Algorithm Option 1

- Convert a heap graph to a type graph.
- Determine the type signature for a given unsummarized heap graph.
Algorithm Option 1

- Convert a heap graph to a type graph.
- Determine the type signature for a given unsummarized heap graph.
- Fix the problems of the type graph using the signatures.
Algorithm Option 1

- Convert a heap graph to a type graph.
- Determine the type signature for a given unsummarized heap graph.
- Fix the problems of the type graph using the signatures.
- Work in progress.
Algorithm Option 2

• Determine the type signature for a given unsummarized heap graph.
Algorithm Option 2

- Determine the type signature for a given unsummarized heap graph.
- Reconstruct an abstract graph using the signatures.
Spec JBB

- No such configuration in spec JBB using out signatures
- Work to do using in signatures
- ...

Spec JBB Type Graph
Questions

• Is it possible to reconstruct a 'good' abstract graph?
Valid Abstraction

- Retains structure
- B and C are not grouped together
Nested Model
Nested Model

- Chang (Comp 150VAN - 2010)
- Data abstraction
- Does it solve the problem?
- Is the algorithm fast enough?
Abstraction Goals

- Validity
- Tractability as measured by number of graph elements
<table>
<thead>
<tr>
<th></th>
<th>vertices</th>
<th>edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original heap</td>
<td>119,099</td>
<td>174,909</td>
</tr>
<tr>
<td>Heapvis 2010</td>
<td>7,583</td>
<td>12,735</td>
</tr>
<tr>
<td>Type Graph</td>
<td>251</td>
<td>503</td>
</tr>
<tr>
<td><strong>Algorithm 1/2</strong></td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Open Research Questions

- Is it possible to reconstruct a 'good' abstract graph?
- Is it possible to reconstruct a minimal abstract graph?
- Complexity of algorithms
- Use of type signatures to create other visualizations (Wong et al, 2006)
Thank you!

- Slides available online at: http://bit.ly/vBASP8