Generating Heap Updates for Dynamic Software Updating

Suriya Subramanian[1] Michael Hicks[2]
Kathryn S. McKinley[1]

[1]Department of Computer Science
The University of Texas at Austin

[2]Department of Computer Science
University of Maryland, College Park

January 14, 2010
Perpetually Available Software Systems
Research Meeting

What we have now

- Jvolve - A Dynamic Software Updating system for Java
- Demonstrated on 3 open source Java applications
  - Jetty webserver, JavaEmailServer, CrossFTP server
- Run a version of an application, bring in a new version, Jvolve system will examine differences, update to the new one

Not quite there yet

- Understands nothing about semantics
- Guarantees type safety, but nothing else
- Generates nothing but the simplest state transformers

Need to relieve programmer of the burden of safety as much as possible.

Can we automate state-transformer generation?

- Start with simple bugfixes, memory leaks, null pointer exceptions
- The patch is usually one line of code
    - Set a field to null
    - Remove an entry from some collection
    - Don't set a field to null
    - Create a new object
- Dynamically applying the patch is easy
- Can we repair application state at update time?
    - Free already leaked objects
    - Create new objects and prevent a future crash
    - ...

Example

```
 1      class Foo {
 2        void bar() {
 3          ...
 4          if (...) {
 5            ...
 6  +         this.field = null;
 7            ...
 8          }
 9        }
10      }
```

- Applying this patch is easy
- Can we free already leaked objects?
- What is the condition under which an object got to line 6?
- Can we recover this condition from heap state?

Example

```
 1      class Foo {
 2        void bar() {
 3          ...
 4          if (...) {
 5            ...
 6  +         this.field = null;
 7            ...
 8          }
 9        }
10      }
```

- Applying this patch is easy
- Can we free already leaked objects?
- What is the condition under which an object got to line 6?
- Can we recover this condition from heap state?

Conditions such as

For all leaky objects, o is leaky if and only if

- o.some_field == false
- o.p == null
- o is/is not pointed by object of type Bar
- ...

Eclipse leak

```
 class NavigationHistory {
   private ArrayList<History> history;
   private ArrayList<Editor> editors;

   // Each object in editors stores a refcount
   // of the number of history objects that point to
   // it. In one instance an object is removed from
   // history, but the refcount not updated.

   private void updateNavigationHistory() {
     for (History h : history) {
       Editor editor = h.editorInfo;
       if (editor ...) {
+        h.editorInfo.refCount--;
+        if (h.editorInfo.refCount == 0)
+          editors.remove(h.editorInfo);
         h.dispose();
       }
     }
   }
 }
```

How can we discover these conditions?

- Static analysis ???
- Dynamic analysis
  - Instrument patched function and mark leaky objects
  - Periodically dump all objects
  - Mine differences between marked and unmarked objects