JS-WALA
(and related topics)

Manu Sridharan
Samsung Research America
http://manu.sridharan.net

Background

- Growing interest in static and dynamic analysis for JavaScript

- WALA project (among others) has useful infrastructure for doing such analysis

- Don’t re-invent the wheel!

Overview

- **jsdelta:** delta debugger for analyses
  https://github.com/wala/jsdelta

- **JS_WALA:** code normalizer
  https://github.com/wala/JS_WALA

- **wala.util constraint solver**

jsdelta: motivation

- Building a JavaScript analysis
- Works great on unit tests
- But, crashes on jQuery!
- What went wrong? Need a **minimized input**
- jsdelta does **automatic input minimization**

jsdelta: example

- **Tool:** Eclipse Orion type inference
- **Bug:** stack overflow on huge input
- **jsdelta output (with variable renaming):**

```
var f = function () {}
var new_f = new f();
f.prototype = new_f;
var c = new f();
```

Using jsdelta

- Easy: write a shell script that prints some message when error occurs
- Also works on JSON inputs
- See http://manu.sridharan.net/blog/2014/09/05/JS-Delta-Walkthrough/ for a walkthrough
- WALA Delta (https://github.com/wala/WALADelta) has utility scripts for debugging WALA's JavaScript analysis using jsdelta (finding timeouts, errors, etc.)

Tips for jsdelta and dynamic analysis

- Check for runtime errors before instrumenting
- Always run with a timeout (e.g., via `timeout` command)
- For browser code, generate wrapper HTML and use PhantomJS / Selenium
- Desired feature: delta-debug multiple scripts in one invocation (any volunteers? :-))

Scaling static analysis using jsdelta

- Idea: use jsdelta to find a small input that causes excessive analysis runtime
- Essential for developing techniques in three recent papers
  - correlation tracking [ECOOP12]
  - dynamic determinacy analysis [PLDI13]
  - static determinacy analysis [OOPSLA14]
- Important: use synthetic measure of time (e.g., # of iterations), not wall-clock time

Idea: applying jsdelta to bytecodes

- Problem: painful to delta debug JVM/Dalvik bytecode analyses
  - multiple files, packaged in jars
  - rewriting without generating too many invalid inputs
- Idea: delta debug a JSON *inclusions file* (opposite of exclusions file)
  - only analyze jars/classes/methods listed in file
  - initially, generate a file that includes everything
  - use standard jsdelta to reduce file

JS_WALA

```
var x = { y: 1 };
console.log(x.y);
```

```
(function (__global) {
    var tmp0, tmp1, tmp2, tmp3,
tmp4, tmp5, tmp6, tmp7, tmp8, tmp9,
tmp10;
    // var x = { y: 1 };
    tmp2 = 1;
    tmp1 = { y: tmp2 };
    tmp0 = 'x';
    __global[tmp0] = tmp1;
    // console.log(x.y);
    tmp5 = 'console';
    tmp3 = __global[tmp5];
    tmp4 = 'log';
    tmp9 = 'x';
    tmp7 = __global[tmp9];
    tmp8 = 'y';
    tmp6 = tmp7[tmp8];
    tmp10 = tmp3[tmp4](tmp6);
}(typeof global === 'undefined' ?
this : global));
```

JS_WALA transformations

- three-address form via additional temporaries
- transform all loops to `while` or `for-in`
- single type of property access
- full grammar in `doc/normalization.md`

JS_WALA uses

- **Key benefit:** fewer cases to consider
- Caveats
  - May complicate some analyses (e.g., static analysis, analysis of globals)
  - Bloats code size
  - No handling of node.js module-scope vars

Jalangi(2)

- JS dynamic analysis framework
  - https://github.com/Samsung/jalangi2
- Analyses implement callbacks for program events
- Jalangi2 provides:
  - code instrumentation to invoke callbacks
  - source location tracking
  - on-the-fly instrumentation for web (via proxy), node.js

Jalangi2 vs JS_WALA

- JS_WALA allows for selective instrumentation, possibly less overhead
  - limited support in Jalangi2
- Less effort to write Jalangi2 analysis

wala.util constraint solver

- general solver for directional (subset / subtype) constraints
- highly optimized (data structures, worklist orderings)
- uses within WALA: bytecode type inference, pointer analysis / call graph construction
- At Samsung, using for JavaScript type inference
- Some docs at https://github.com/wala/WALA/wiki/wala.util-overview#fixed-point-solvers

Questions?