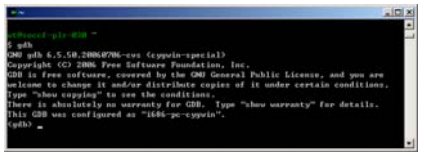
Hierarchical Dynamic Slicing

Tao Wang Abhik Roychoudhury
National University of Singapore

Preamble

Software Debugging

- Time consuming
- Traditional Source-Level Debugger
  - The key problem is automation



Slicing for Debugging

- Slicing criterion
  - (Input, Variable, Line#)
- (Dynamic) Slice
  - A fragment of the program which directly or indirectly affect the behavior of the slicing criterion
  - Compute closure of control/data dep.
  - Treat the slice as bug-report.

Why Dynamic?

- Naturally corresponds to debugging via testing.
  - Test program against test-cases.
  - Find "offending" test-cases
    - Output "unexpected".
  - Compute dynamic slice to debug the behavior in the offending test cases.

Dynamic Slicing

```
1. void setRunningVersion(boolean runningVersion){

2.   if( runningVersion ) {
3.       savedValue = value;
     }
     else{
4.       savedValue = "";
     }

5    this.runningVersion = runningVersion;

6.   System.out.println(savedValue);
   }
```

runningVersion = false

null ?

Dynamic Slice

```
1. void setRunningVersion(boolean runningVersion)
2.   if( runningVersion ) {
3.     savedValue = value;
     }
     else{
4.     savedValue = "";
     }
5    this.runningVersion = runningVersion;
6.   System.out.println(savedValue);
}
```

Slicing Criterion

So … ?

- Huge overheads
  - Backwards slicing requires trace storage.
  - Past work – Jslice tool for Java
    - Online trace compression & traversal
    - http://jslice.sourceforge.net
- Dynamic Slice is still too large …
  - … for human comprehension
  - In this paper

An example

```
1   public static void main(String[] args) {
      …….
2.    init( db );
3     operate( db );
4.    output ( db )
5.    return;
}
```

SPECJVM DB program

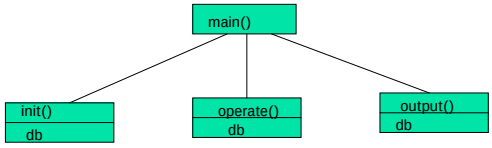
```
init( .. db) {
    db= ..
    ….
}
```

```
operate (… db ) {
    db =..
    …
}
```

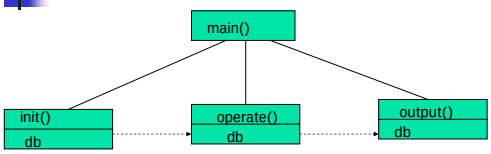
```
output (db) {
    ……
    print(db...);
}
```


Divide exec. into "phases"

```
1   public static void main(String[] args) {
      …….
2.    init( db );
3     operate( db );
4.    output ( db );
5.    return;  }
```



Report inter-phase dep.



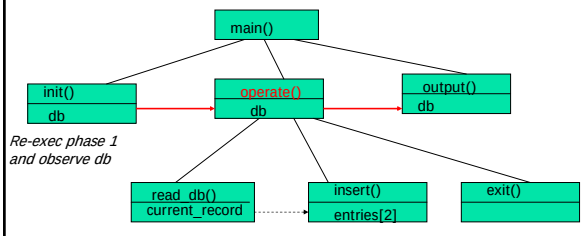
Intra-phase control and data dependencies are suppressed.

Inter-phase dep. form input-output relationships.

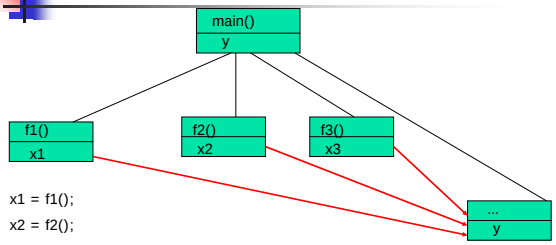
Programmer helps zoom into

… one phase by inspecting the phase outputs

-> (may/may not involve re-executing program)



Parallel dependence chains

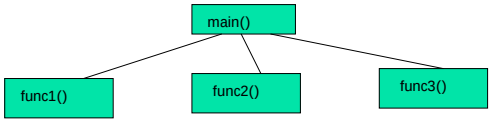



```
x1 = f1();
x2 = f2();
x3 = f3();
y = x1 + x2 + x3;
```

print y --- Criterion

Programmer intervention …

… may not be needed, for example



All dependencies inside this phase, zoom into this one

Key issues

- Compute "phases" of an exec. trace
  - Diff. from phase detection in PL community
- Augment dynamic slicing algorithm
  - Mark inter-phase dependencies
  - Compute only reachable nodes from selected inter-phase dependency.
- Programmer intervention
  - Select the first suspicious inter-phase dep.
  - Comprehension guides computation.

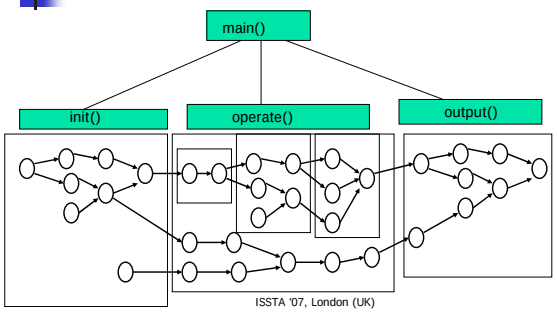
Phase detection

- Divide an exec. trace at boundaries of
  - Loops
  - Method calls
  - Loop iterations
  - …
    - and recursively again at these control structure boundaries.

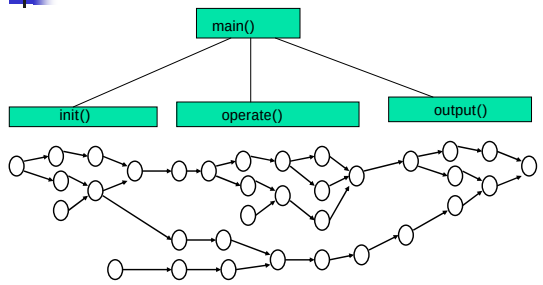
Differs from work in PL

- Phase detection to guide program opt.
  - Divide a phase into fixed-length intervals.
  - Compute metric (e.g. Basic block vectors) for each interval.
  - Cluster adjacent intervals with similar metric.
- No notion of control flow context.

Programmer intervention



Beyond Visualization



Subject Programs

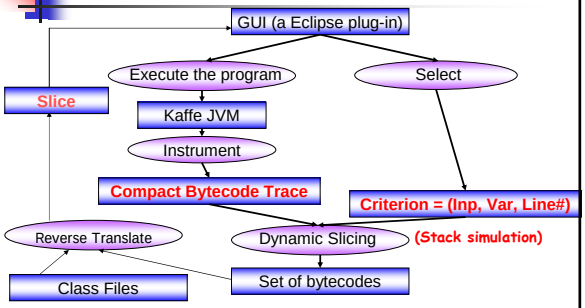
Subject Pgm.	Description	Size
NanoXML	XML parser for Java	7646 LOC 24 classes
JTopas	Java library for parsing text	5400 LOC 50 classes
Apache JMeter	Performance testing tool	43400 LOC 389 classes

SIR subjects --- for each used 3 buggy versions with seeded faults.

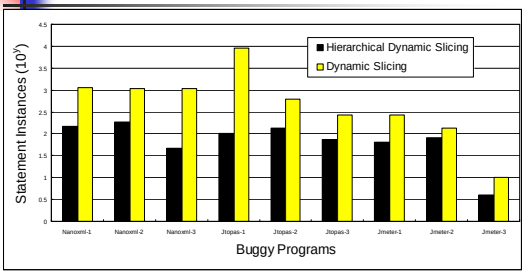
Slicing Infrastructure - JSlice

- Dynamic Slicing tool for Java programs
  - The first one
- Open source, bytecode level slicing tool
  - http://jslice.sourceforge.net
  - Distributed with the Kaffe VM
- 50+ registered users.
  - released 8 months ago
  - Adopted for teaching in few universities

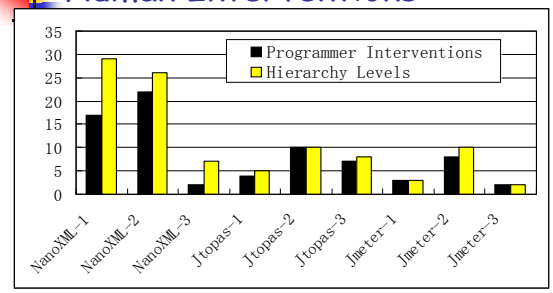
Slicing Infrastructure JSlice



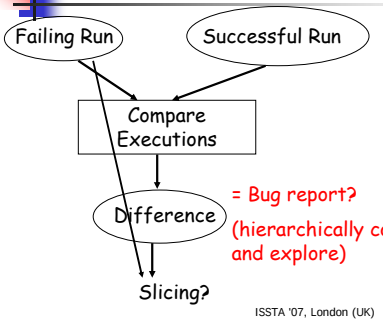
Statement Inst. Examined (Log scale)



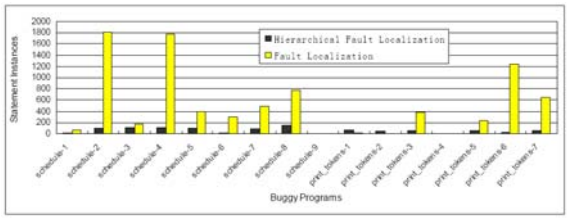
Human Interventions



Fault Localization



Experiments



Summary

- Hierarchically Explore Bug-report
  - Dynamic vs. Static
    - Thin Slicing – explore static slices
  - On-line vs. post-mortem
    - DDgraph – explore dynamic dependence graph hierarchically.
  - Inter-phase vs. intra-phase
    - Work on parallel pgm. debugging – present intra-phase dependencies

Summary

- Hierarchically construct and explore the dynamic slice for understanding.
  - Comprehension/computation interleaved.
  - Structures programmer feedback
    - Only examine inter-phase dependencies
    - Not just view a programmer chosen phase.
  - Detailed expt. on SIR benchmarks
- Efficacy on human subjects
  - To be distributed in future releases of Jslice.
  - http://jslice.sourceforge.net/