A Differencing Algorithm for Object-Oriented Programs

Taweesup (Term) Apiwattanapong

Alessandro Orso
Mary Jean Harrold

College of Computing
Georgia Institute of Technology

National Science Foundation awards CCR-0306372, CCR-0205422, CCR-9988294, CCR-0209322, and SBE-0123532 to Georgia Tech

Example

2c2

< float n;

> double n;

6a7,9

> public void m1() {

> ...

> }

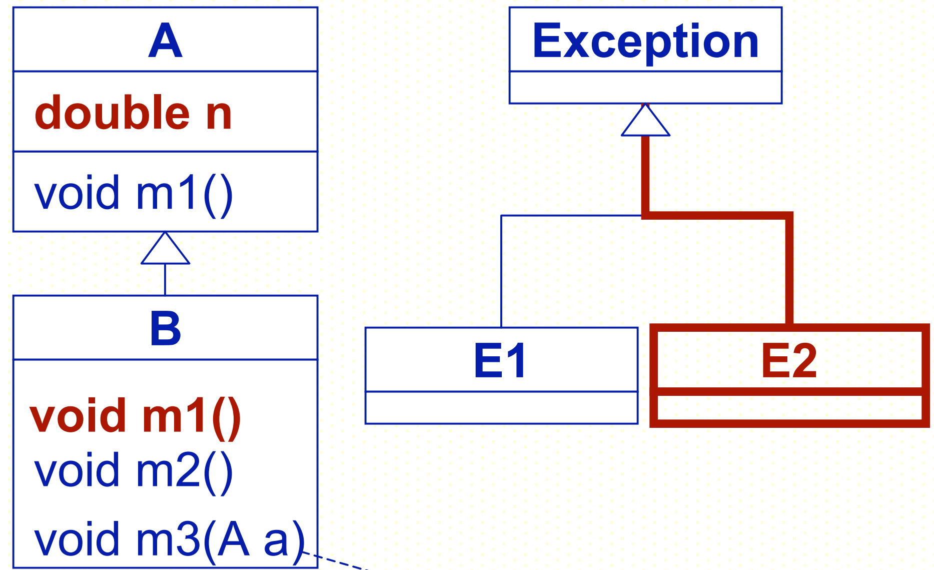
9a13

> int i=0;

21c25

< public class E2 extends E1 {

> public class E2 extends Exception {



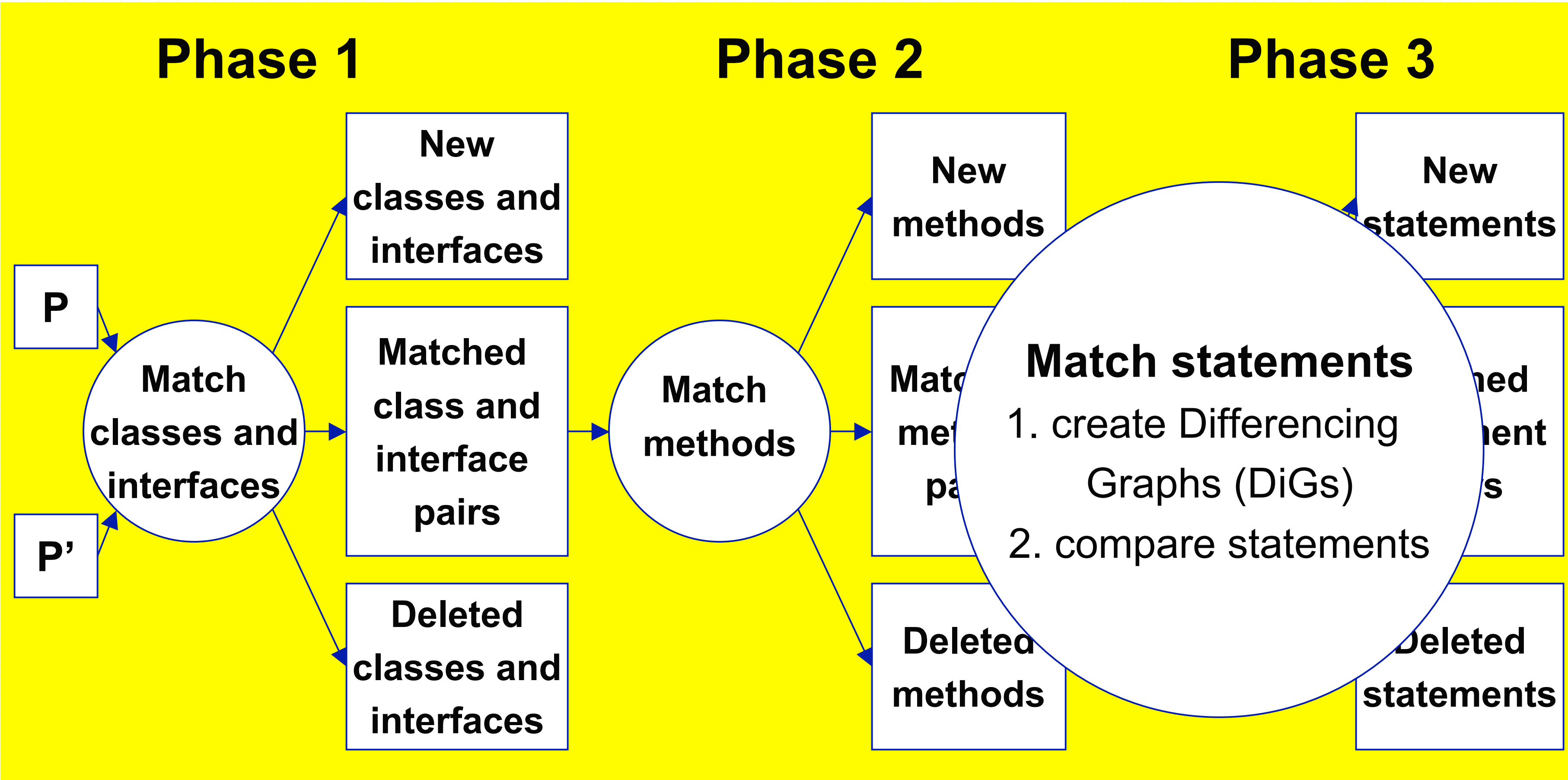
Modified Version (P')

```
n = 1/n;
int i=0;
a.m1();
try {
    …
}
catch (E1 e) { … }
catch (Exception e) { … }
```

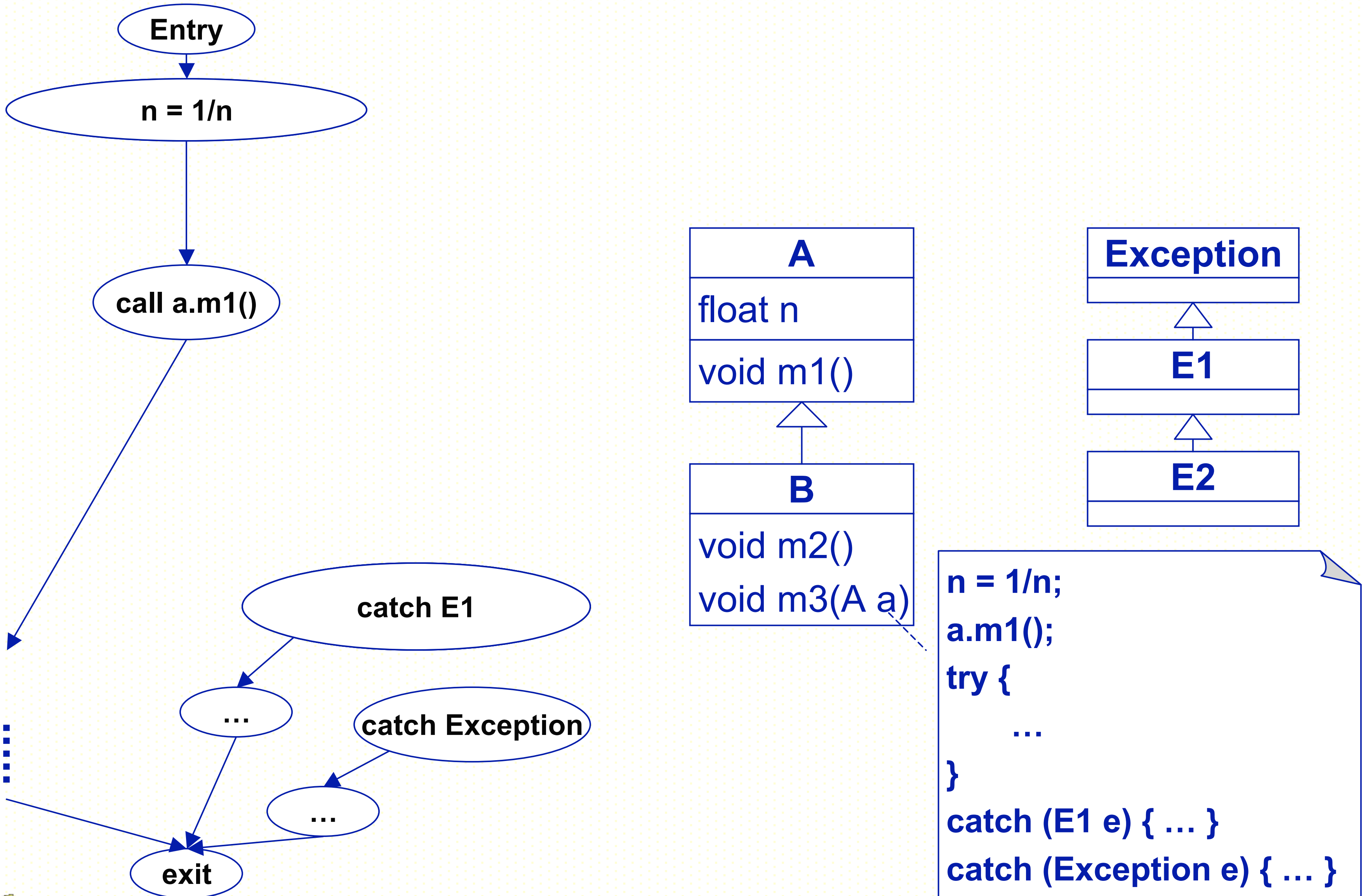
Outline

- Introduction
- Differencing Algorithm
  - Representation
  - Matching
- Empirical Studies
- Related Work
- Conclusions

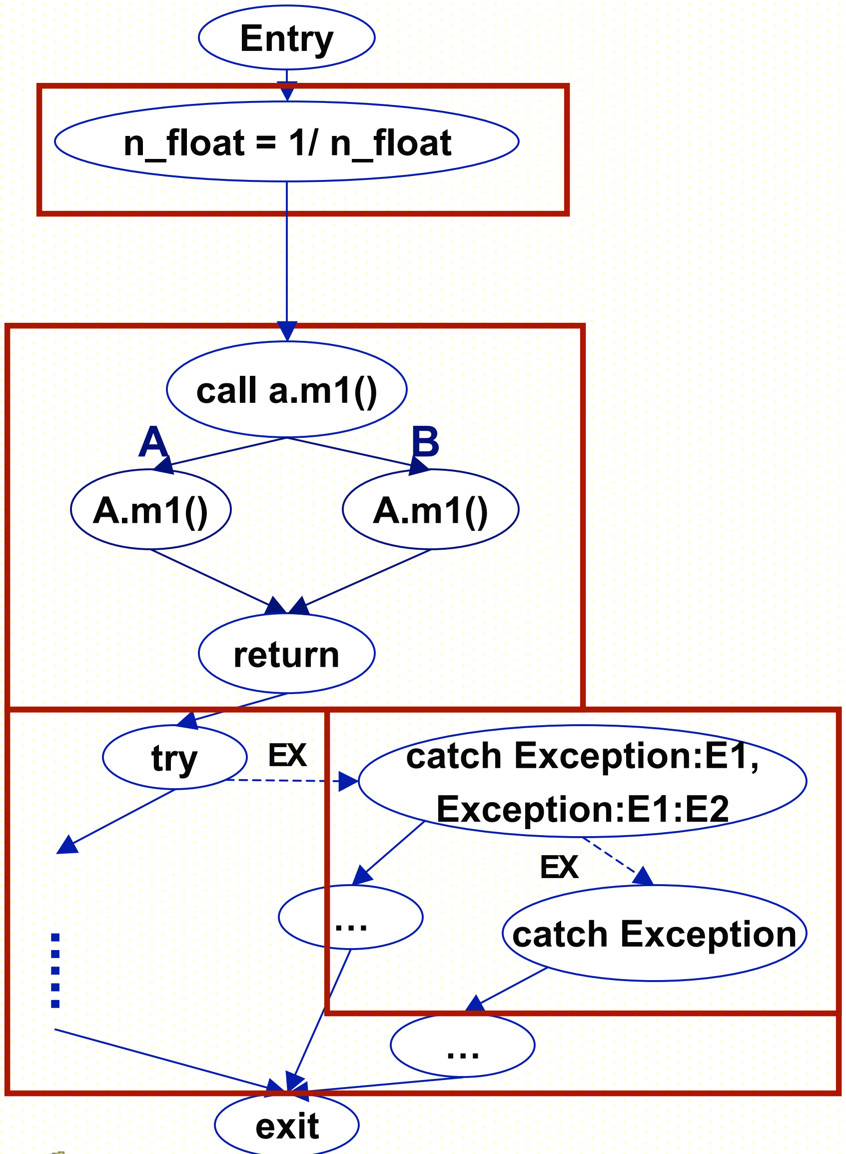
Overview of Differencing Algorithm



DiG: Extend CFG



DiG: Extend CFG

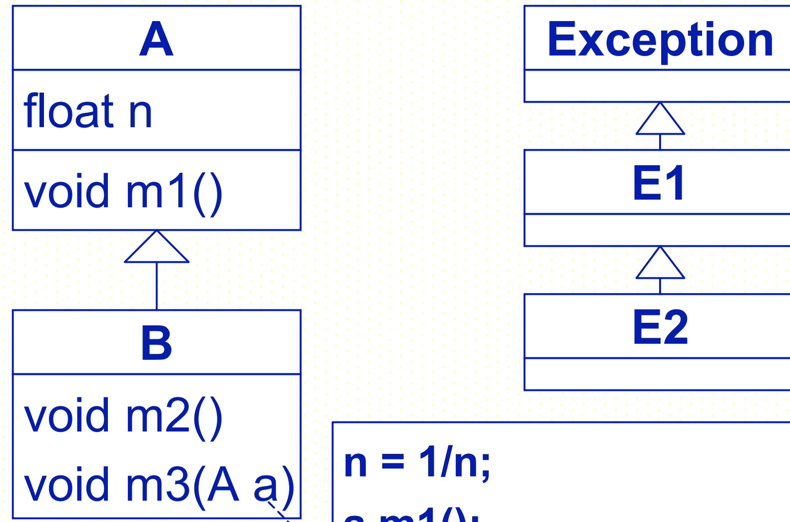


Type in scalar variables' names

Dynamic Dispatch

Exception Handling

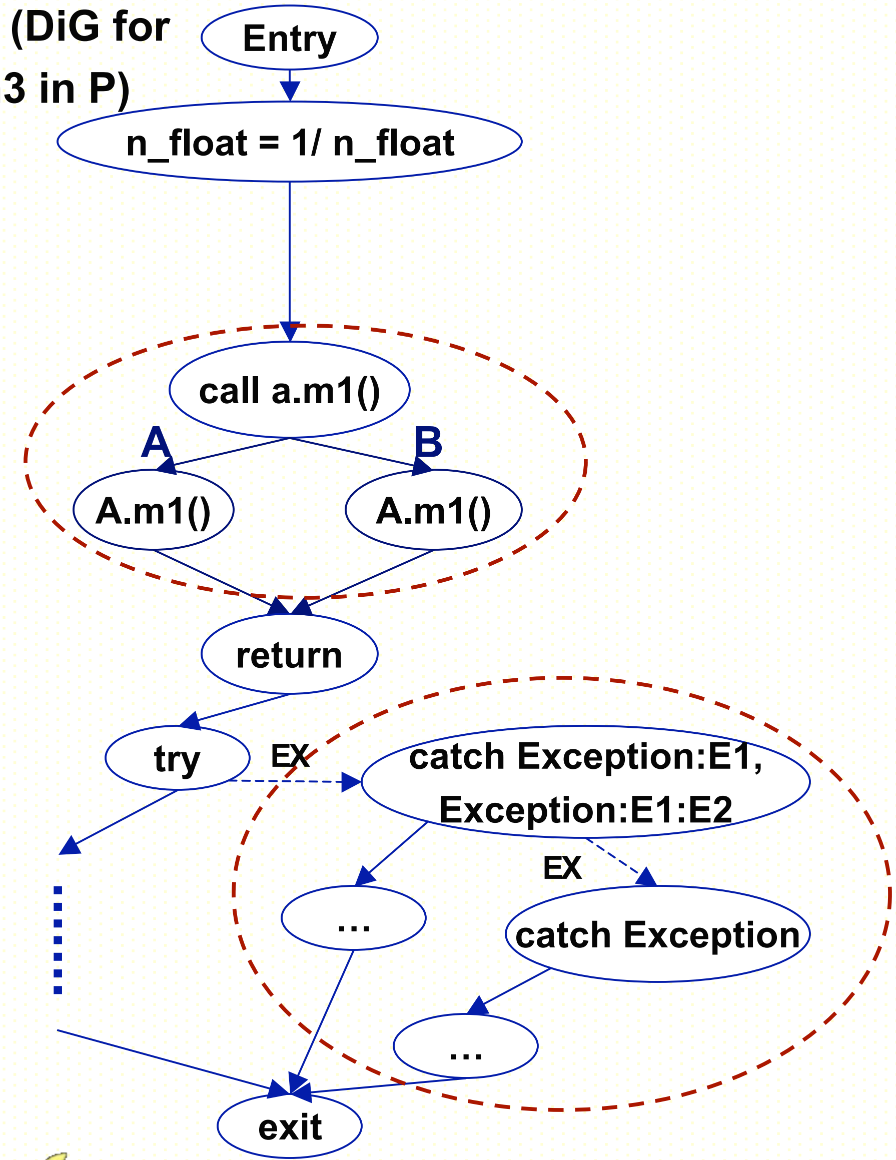
Globally-qualified names




```
n = 1/n;
a.m1();
try {
      …
}
catch (E1 e) { … }
catch (Exception e) { … }
```

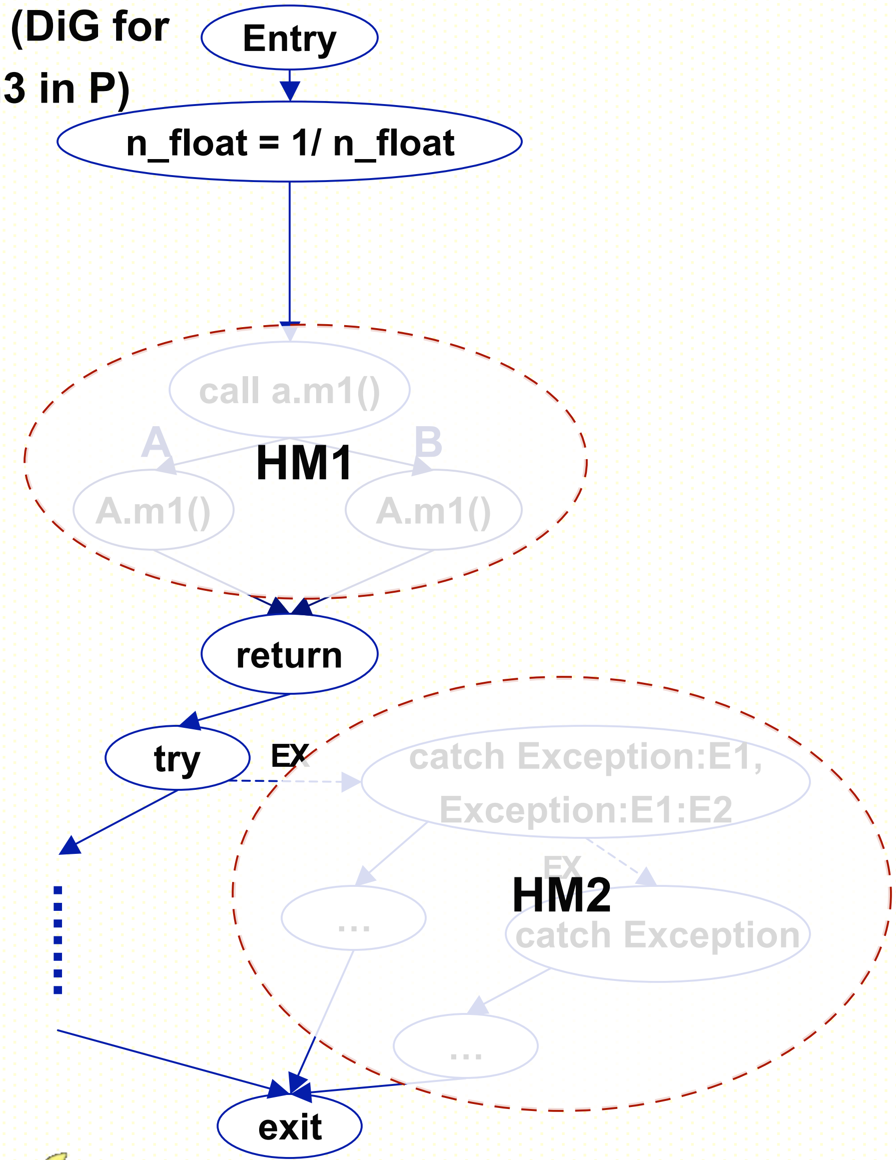
DiG: Simplify the extended CFG

G (DiG for m3 in P)

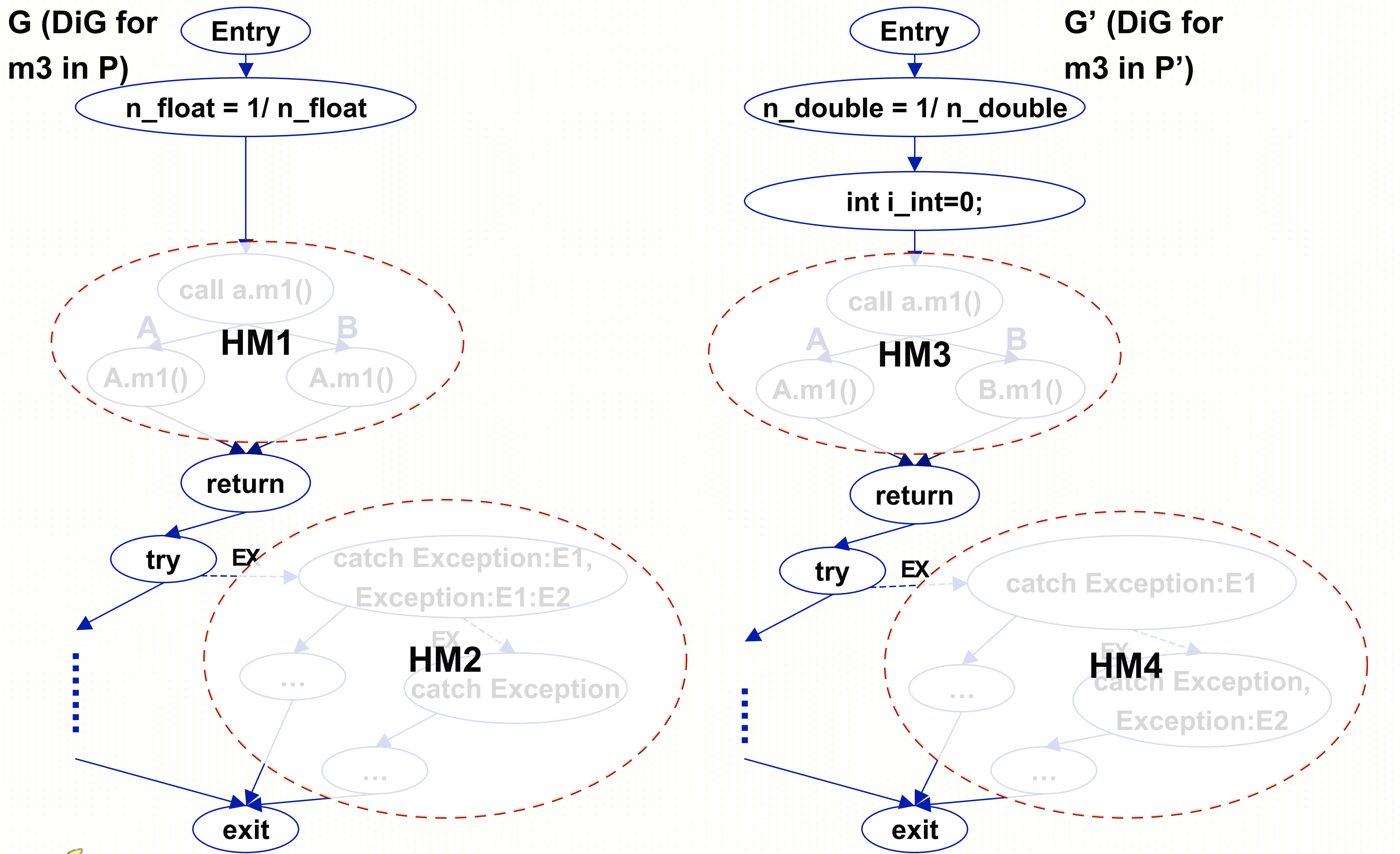


DiG: Simplify the extended CFG

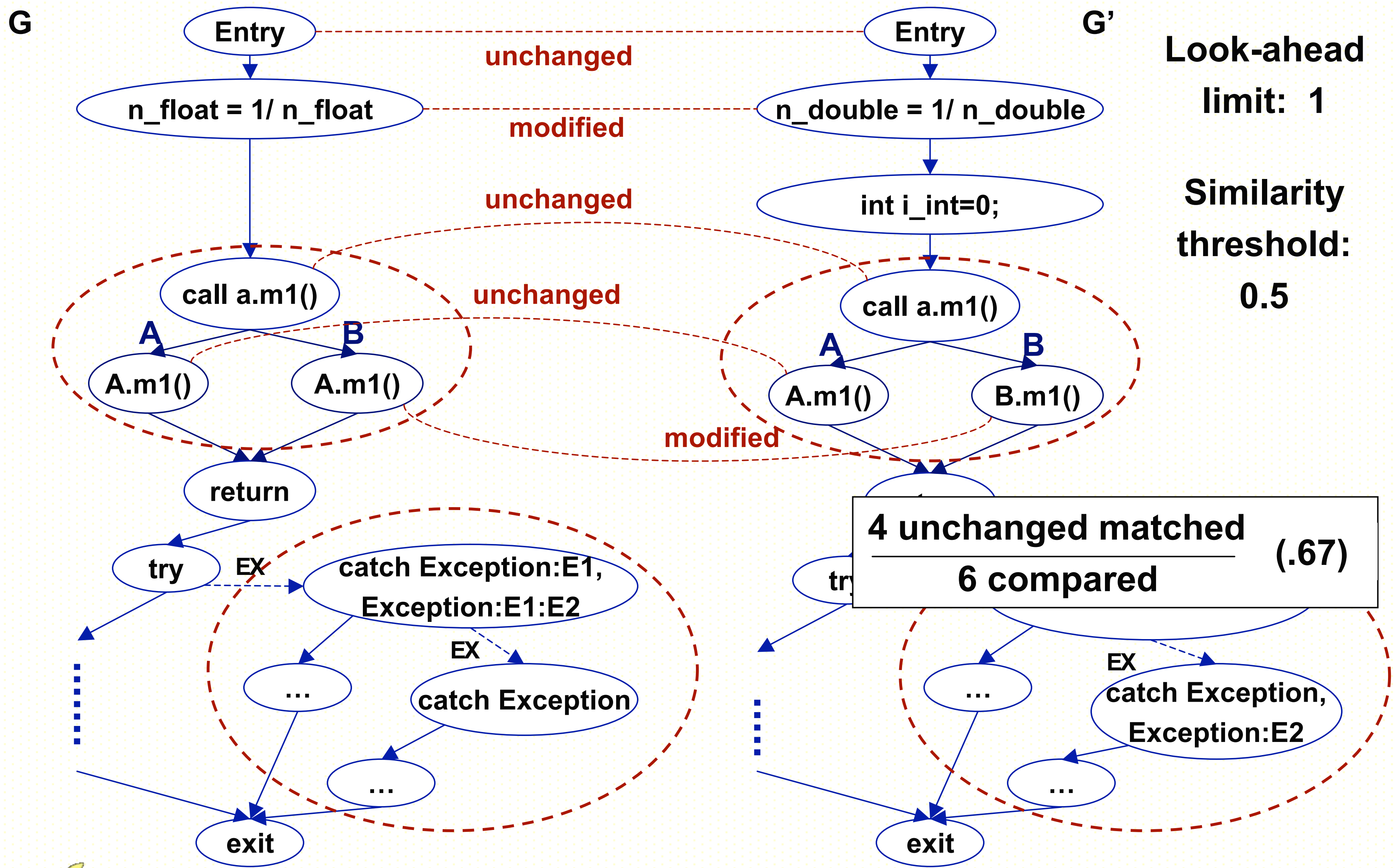
G (DiG for m3 in P)



DiG: Simplify the extended CFG

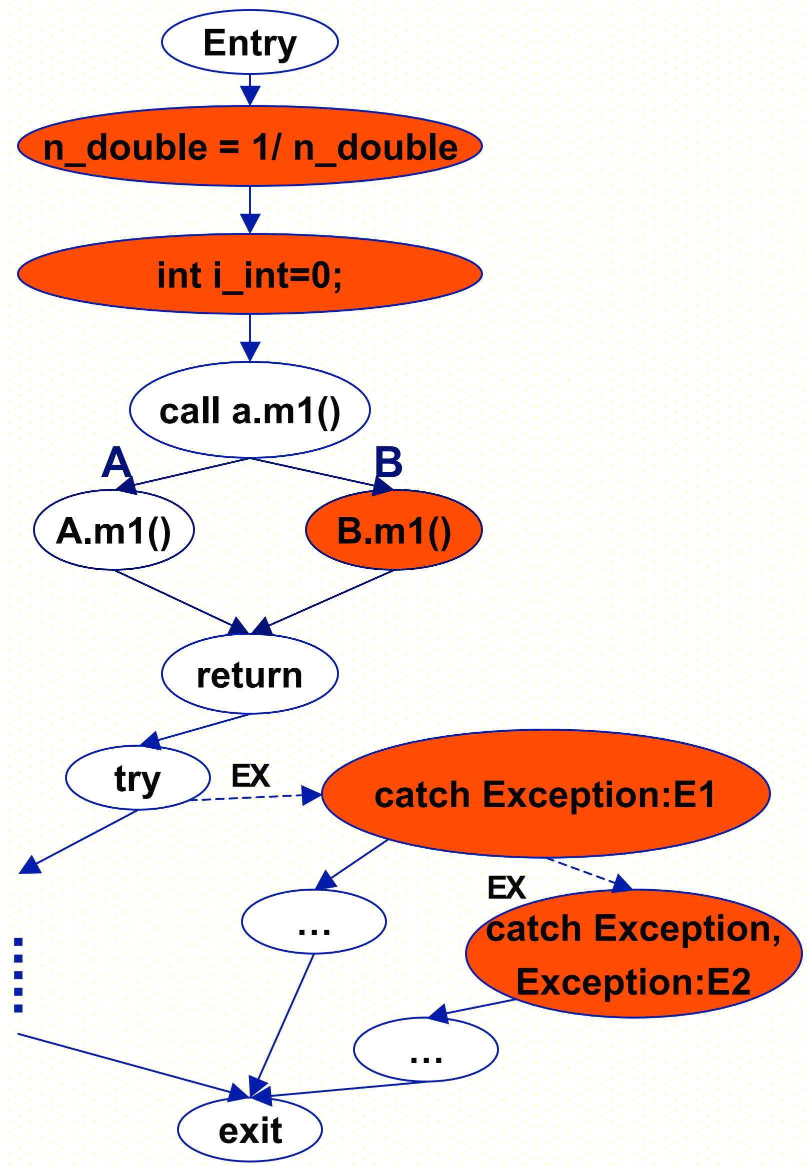


Matching


G
Entry
n_float = 1/ n_float
call a.m1()
A
B
A.m1()
A.m1()
return
try
EX
catch Exception:E1,
Exception:E1:E2
EX
…
catch Exception
…
exit
unchanged
modified
unchanged
unchanged
modified
G'
Entry
n_double = 1/ n_double
int i_int=0;
call a.m1()
A
B
A.m1()
B.m1()
tr
EX
…
catch Exception,
Exception:E2
…
exit
4 unchanged matched
6 compared
(.67)
Look-ahead
limit: 1
Similarity
threshold:
0.5

Matching

```
public class A {
   double n;
public void m1() { ... }
}
public class B extends A {
   public void m1() {
      …
   }
public void m2() { ... }
   public void m3() {
      n = 1/n;
      int i = 0;
      a.m1();
      try {
         ...
      }
      catch (E1 e) { ... }
      catch (Exception e) { ... }
}
public class E1 extends Exception {
   …
}
public class E2 extends Exception {
…
}
```



Outline

- Introduction
- Differencing Algorithm
  - Representation
  - Matching
- Empirical Studies
- Related Work
- Conclusions

Empirical Studies

Experimental Setup

- JDiff: A Java implementation of our technique
- Subject : Jaba
  - A Java bytecode analysis tool
  - 60KLOC (550 classes, 2800 methods)
  - 2 sets of 4 consecutive versions
    - Low activity: v1, …, v4 (3-20 changes)
    - High activity: va, …, vd (15-150 changes)

Studies

1. Efficiency of our algorithm
2. Effectiveness of our algorithm in matching
3. Effectiveness of our algorithm for a maintenance task

Study 1: Efficiency of Our Algorithm

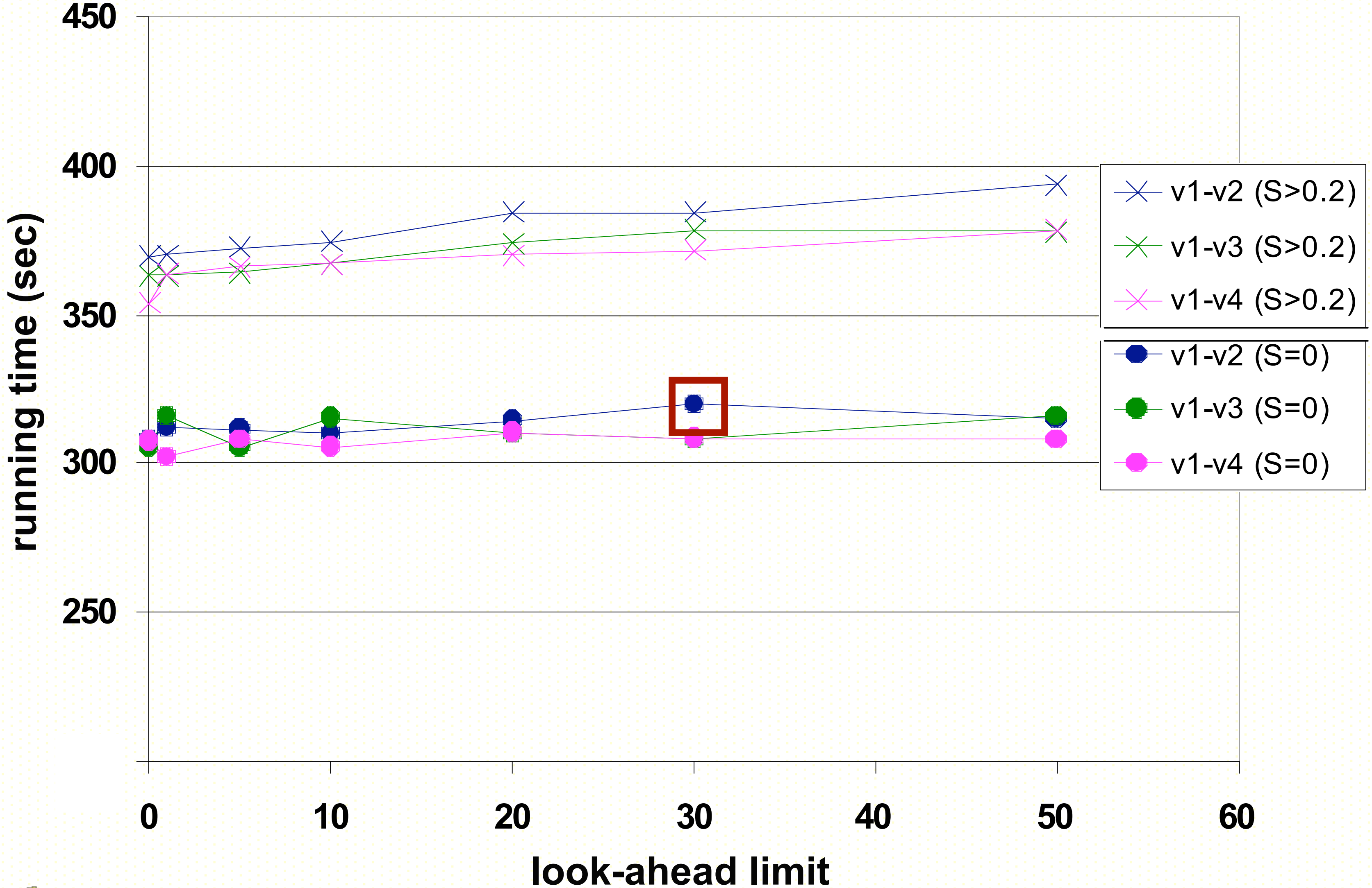
Goal:

Measure the efficiency of our algorithm for various look-ahead limits and hammock similarity thresholds

Method:

1. Run JDiff
   - Low-activity versions (v1-v2, v1-v3, and v1-v4)
   - Various look-ahead limits (0-50)
   - Various similarity thresholds (0-1)
2. Collect the running times.

Study 1: Efficiency of Our Algorithm



Study 3: Effectiveness for a Maintenance Task

Goal:

Assess the effectiveness of our algorithm for a maintenance task (coverage estimation)

Method:

- Jaba's regression test suite (~60% coverage)
- Both low- and high-activity versions (v1-v2, v1-v3, v1-v4, va-vb, va-vc, and va-vd)
- For each pair (vi-vj),
  1. Collect coverage for vi
  2. Run JDiff on vi-vj to get mappings
  3. Get estimated coverage of vj based on mappings
  4. Collect actual coverage for vj
  5. Compare actual and estimated coverage of vj

Study 3: Effectiveness for a Maintenance Task

Pair vi,vj	Avg. Correctly estimated for vj (%)
v1,v2	98.57
v1,v3	98.46
v1,v4	98.03
va,vb	96.25
va,vc	86.08
va,vd	84.70

Low-activity period

High-activity period

Outline

- Introduction
- Differencing Algorithm
  - Representation
  - Matching
- Empirical Studies
- Related Work
- Conclusions

Related Work

Textual
- E.W. Myers. *Algorithmica* 1986 (UNIX diff)

Control-flow graph based
- J. Laski and W. Szermer. *ICSM 1992*
- Z. Wang, K. Pierce, and S. McFarling. *JILP 2000* (BMAT)

Dependence graph based
- S. Horwitz. *PLDI 1990*
- D. Binkley. *ICSM 1992*

Abstract syntax tree based
- Raghavan et al. *ICSM 2004* (Dex)
- Ren et al. *Technical Report 2004* (Chianti)

Input-output dependence based
- D. Jackson. *ICSM 1994* (Semantic diff)

Conclusions

Contributions

- A differencing algorithm that
  - Based on a new graph representation which models object-oriented features
  - Uses several strategies to increase matching capability
- A tool that implements our technique (JDiff)
- A set of studies that show the efficiency and effectiveness of the approach

Future Directions

- To improve matching results
  - Investigate additional heuristics
  - Use common change patterns
- Test-suite augmentation
  - Create new test cases based on changes in the program

Questions?