Lecture 21

Regression Testing
Path Spectra

Today's Agenda (1)

- Regression Test Selection
  - Presentation by David (skeptic)
- Path Spectra
  - Presentation by Sidd (advocate)
  - Presentation by Srinivas (skeptic)

Today’s Agenda (2)

- Research problems in regression testing
  - Regression test selection
  - Regression test prioritization
  - Regression test augmentation

Today’s Agenda (3)

- Orso et al.‘s FSE 2004 on regression test selection for Java program
- Focus on Rothermel & Harrold 1997 Algorithm

What is Regression Testing?

What is Regression Testing?

- Regression testing is performed on modified software to provide confidence that
  - software behaves correctly and
  - modifications did not adversely impact software quality.

Regression Testing

- Test Case (t)
  - e.g. JUnit test
- Test suite: a set of test cases, T= {t1, t2, t3, ... tn}
- Regression testing intends to identify regression fault introduced due to changes.
- Regression test strategy?
  - The most naive one is to rerun every test case in the test suite.

Regression Test Selection

- P: old version
- P’: new version
- T is a test suite for P
- Assume that all tests in T ran on P. => Generate coverage matrix C.
- Given the delta between P and P’ and the coverage matrix C, identify a subset of T that can identify all regression faults. (Safe RTS)

Regression Test Prioritization

- P: old version
- P': new version
- T is a test suite for P
- Assume that programmers do not have enough time to select and run test cases.
- How can we order test cases so that test cases that run early can provide the most benefit when the time is limited?
- Given the delta between P and P' and C, what is an ordering of test cases in T?

Regression Test Augmentation

- P: old version
- P': new version
- T is a test suite for P
- Generate a set of test cases that effectively exercise the delta between P and P'.
- In other words, it is a test generation for evolving programs.

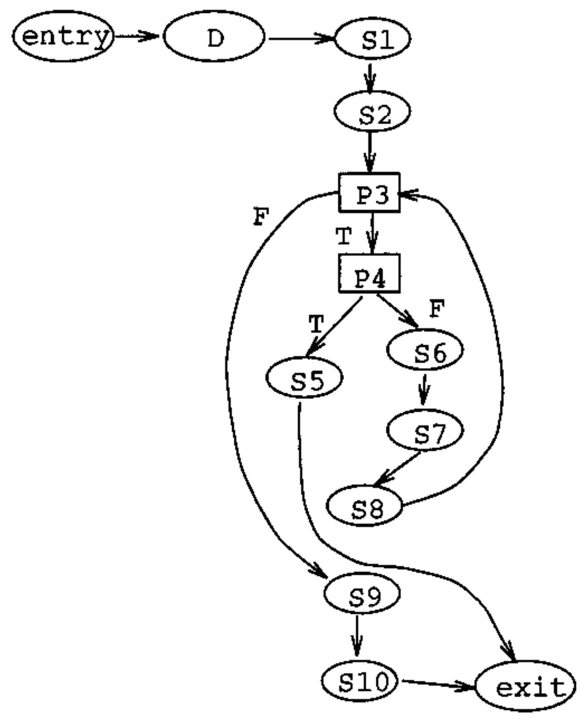
Regression Test Selection

- “Scaling Regression Testing to Large Software Systems.”
- A. Orso, N. Shi and M. J. Harrold
- FSE 2004

Harrold & Rothermel's RTS

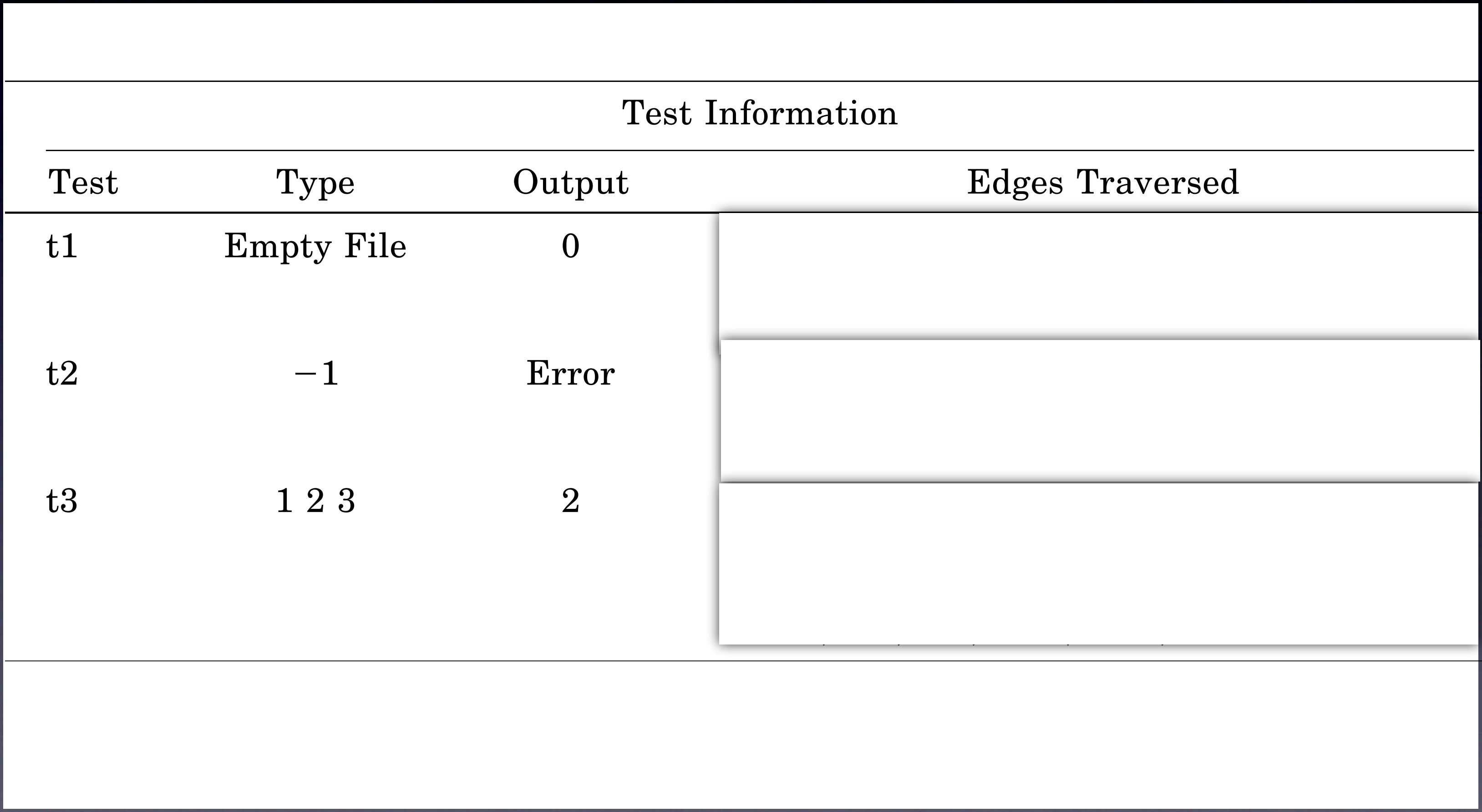
- A safe, efficient regression test selection technique
- TOSEM 1997
- RTS based on graph traversal

Build CFG



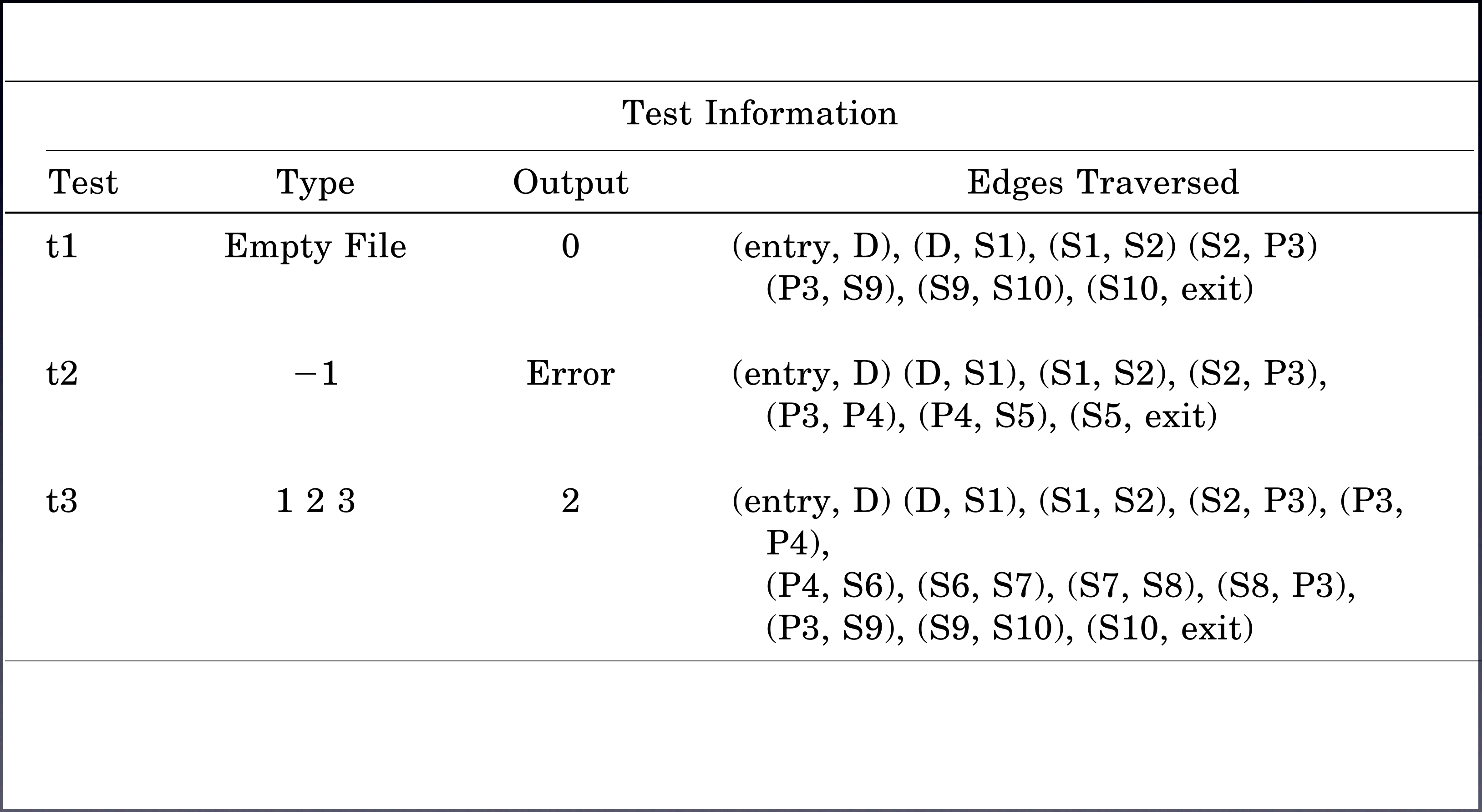
```
     Procedure avg

S1.  count = 0
S2.  fread(fileptr,n)
P3. while (not EOF) do
P4.    if (n<0)
S5.       return(error)
       else
S6.       numarray[count] = n
S7.       count++
       endif
S8.    fread(fileptr,n)
    endwhile
S9. avg = calcavg(numarray,count)
S10.return(avg)
```

Run T = {t1, t2, ...} on P



Test Information			
Test	Type	Output	Edges Traversed
t1	Empty File	0	
t2	-1	Error	
t3	1 2 3	2	

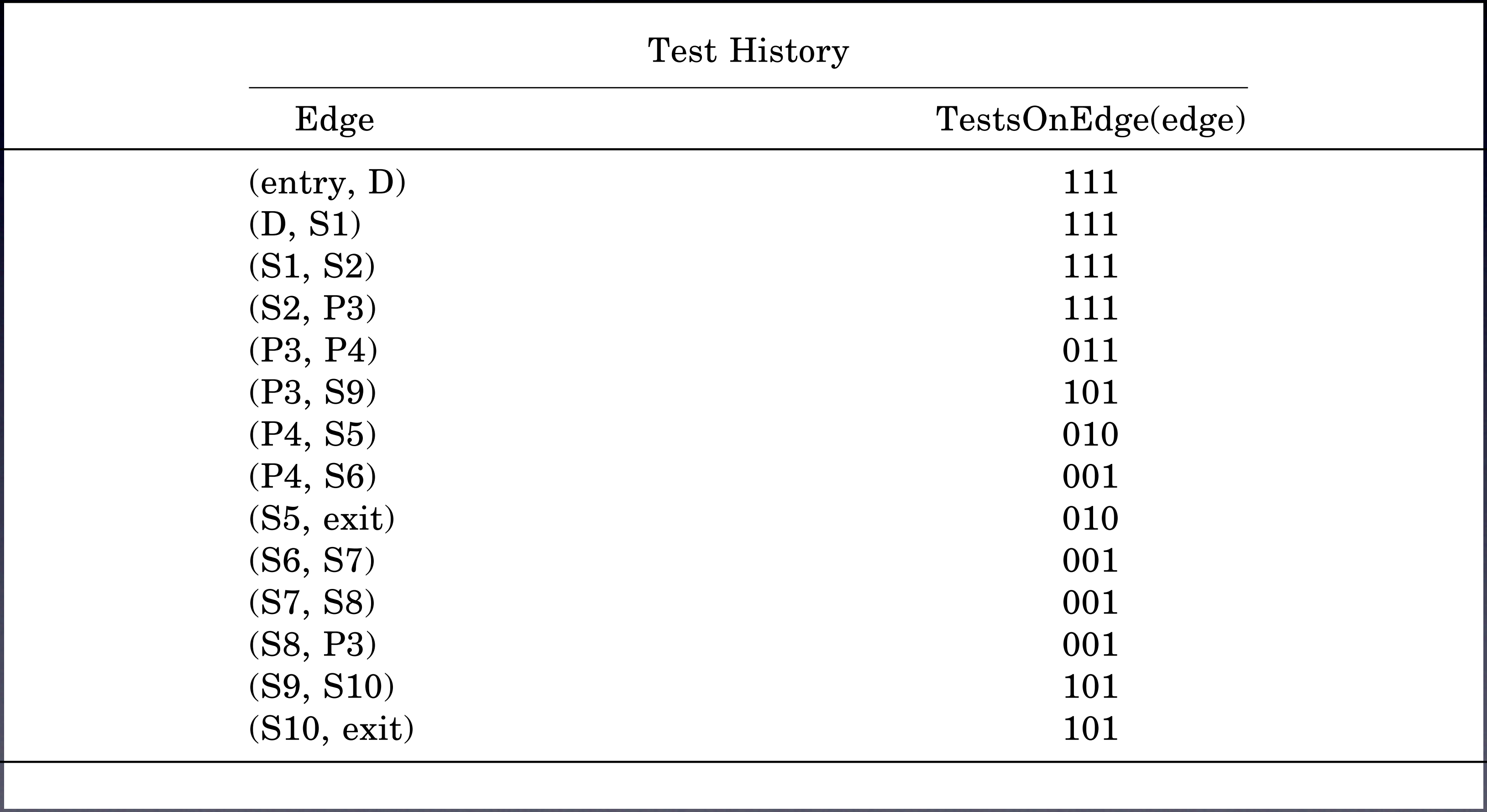
Run T = {t1, t2, ...} on P



Test Information

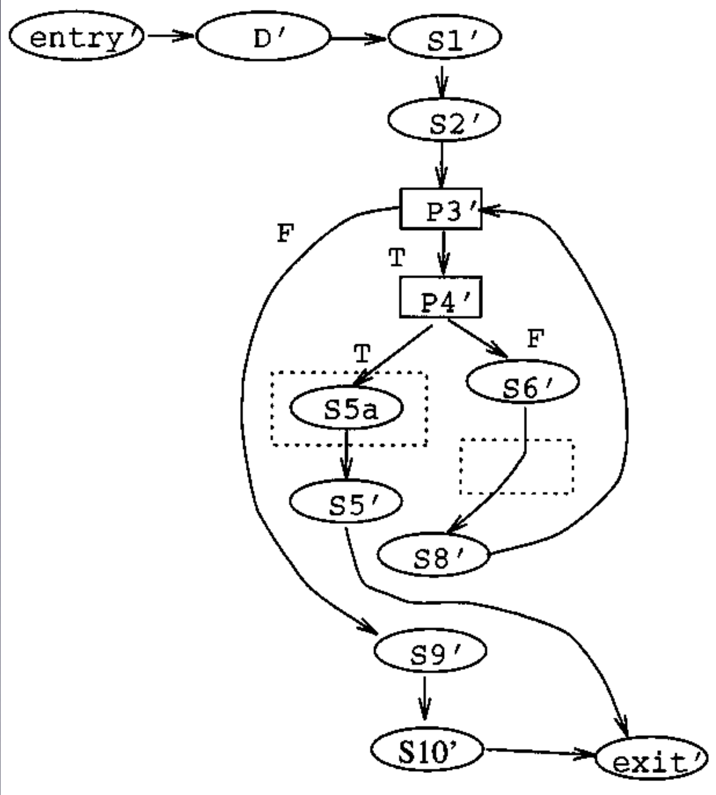
Test	Type	Output	Edges Traversed
t1	Empty File	0	(entry, D), (D, S1), (S1, S2) (S2, P3) (P3, S9), (S9, S10), (S10, exit)
t2	−1	Error	(entry, D) (D, S1), (S1, S2), (S2, P3), (P3, P4), (P4, S5), (S5, exit)
t3	1 2 3	2	(entry, D) (D, S1), (S1, S2), (S2, P3), (P3, P4), (P4, S6), (S6, S7), (S7, S8), (S8, P3), (P3, S9), (S9, S10), (S10, exit)

Build Edge Coverage Matrix



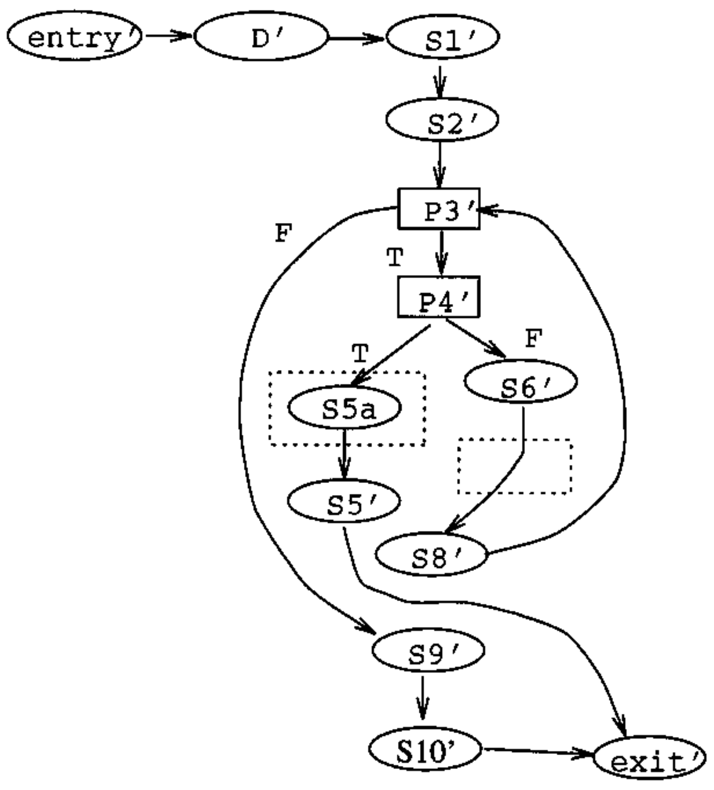
Test History	
Edge	TestsOnEdge(edge)
(entry, D)	111
(D, S1)	111
(S1, S2)	111
(S2, P3)	111
(P3, P4)	011
(P3, S9)	101
(P4, S5)	010
(P4, S6)	001
(S5, exit)	010
(S6, S7)	001
(S7, S8)	001
(S8, P3)	001
(S9, S10)	101
(S10, exit)	101

Traverse two CFGs in parallel



```
Procedure avg2

S1'. count = 0
S2'. fread(fileptr,n)
P3'. while (not EOF) do
P4'.    if (n<0)
S5a.        print("bad input")
S5'.        return(error)
        else
S6'.        numarray[count] = n
        endif
S8'.    fread(fileptr,n)
      endwhile
S9'. avg = calcavg(numarray,count)
S10'.return(avg)
```

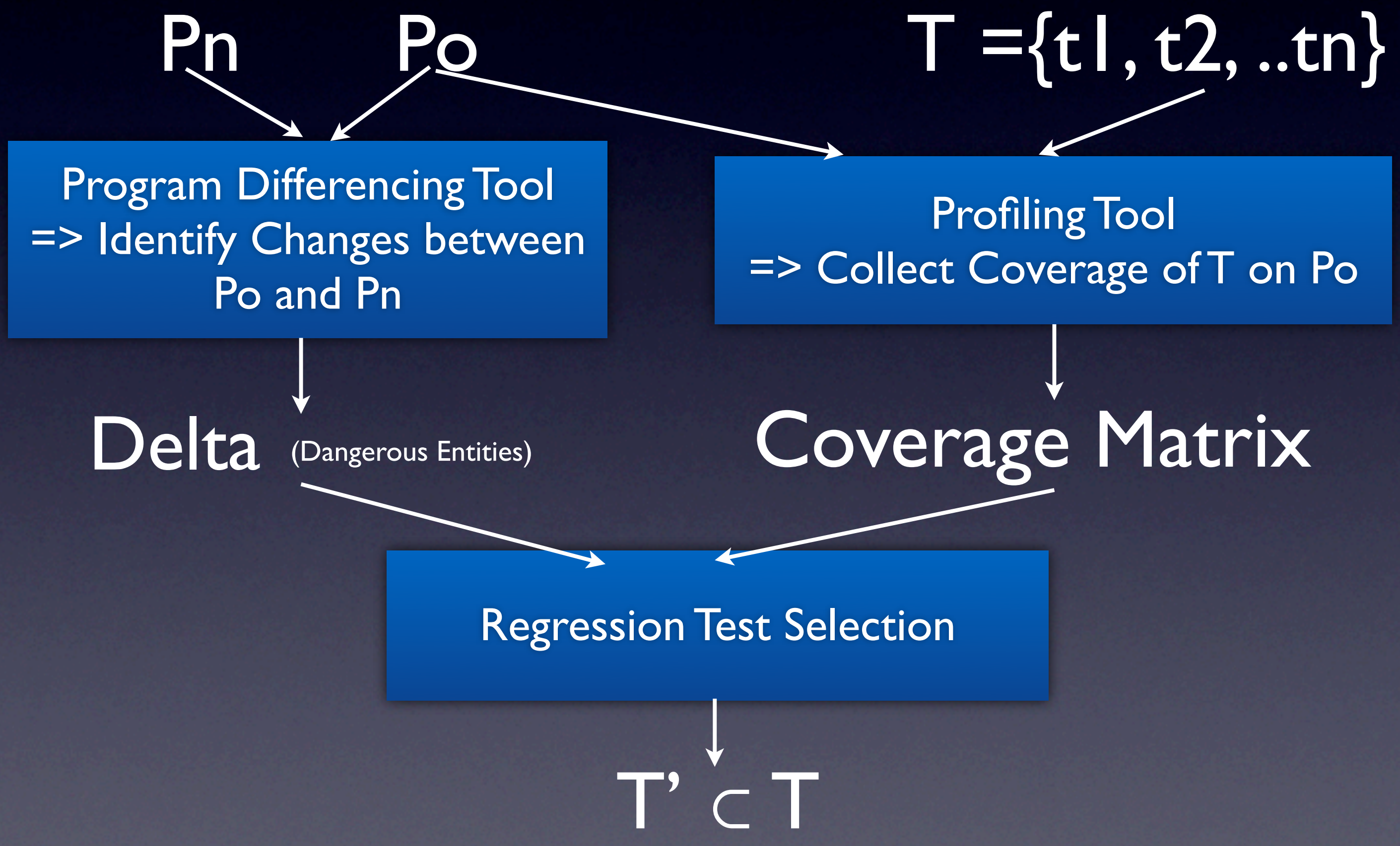
Traverse two CFGs in parallel



```
Procedure avg2

S1'. count = 0
S2'. fread(fileptr,n)
P3'. while (not EOF) do
P4'.    if (n<0)
S5a.         print("bad input")
S5'.         return(error)
        else
S6'.         numarray[count] = n
        endif
S8'.    fread(fileptr,n)
     endwhile
S9'. avg = calcavg(numarray,count)
S10'.return(avg)
```

Select all test cases that visited (P4, P5) and (S6, S7)

Recap: RTS Framework


Pn
Po
T ={t1, t2, ..tn}
Program Differencing Tool => Identify Changes between Po and Pn
Profiling Tool => Collect Coverage of T on Po
Delta
(Dangerous Entities)
Coverage Matrix
Regression Test Selection
T' ⊂ T

Harrold et al. RTS for Java

- Regression Test Selection for Java Software
- OOPSLA 2001
- What are main challenges for making RTS work in Java?
- How did Harrold et al. address challenges for Java software?
- What are differences between this work and Harrold et al.'s RTS for procedural languages?

Main Challenges for making RTS work in Java

- Java language features: in particular, (1) polymorphism, (2) dynamic binding, and (3) exception handling
- Why is polymorphism & dynamic binding difficult to handle in RTS?
    -

Main Challenges for making RTS work in Java

- Java language features: in particular, (1) polymorphism, (2) dynamic binding, and (3) exception handling
- Why is polymorphism & dynamic binding difficult to handle in RTS?
  - The target of method calls depends on the dynamic type of a receiver object.

```
class B extends A {
};
class C extends B {
 public void m(){...};
};
void bar(A p) {
  A.foo();
  p.m();
}
```

```
class B extends A {
};
class C extends B {
 public void m(){...};
};
void bar(A p) {
  A.foo();
  p.m();
}
```

A few other enhancements

- Eternal libraries and components
    - Why is it important to model interaction between the main code and its libraries?
    -

A few other enhancements

- Eternal libraries and components
  - Why is it important to model interaction between the main code and its libraries?
  - External library code can invoke internal methods if the internal methods override external methods.

```
class B extends A {
  public void foo() {...};
}
class C extends B {
 public void bar() {...};
};
```

```
class B extends A {
 public void foo() {...};
 public void bar() {...};
}
class C extends B {       (

};
```

Orso et. al.'s Scalable RTS

- Scalable Regression Test Selection for Java
- FSE 2004
- What are main limitations for Harrold et al.'s OOPLSA 2001 techniques?
- How did they address these limitations?
    -

Orso et. al.'s Scalable RTS

- Scalable Regression Test Selection for Java, FSE 2004
- What are main limitations for Harrold et al.'s OOPLSA 2001 techniques?
  - low-level analysis for all classes while the scope of classes that are affected by modification can be partitioned using a class hierarchy analysis
- How did they address these limitations?
  - For each type with modification, identify its superclasses and subclasses as well as classes that have direct dependence on them through explicit references.

Evaluation of Orso et. al.'s RTS

- What are main research questions raised by Orso et al.?
  - RQ1: Cost comparison with edge-level selection (that does not use partitioning analysis)
  - RQ2: Cost comparison with high-level selection (without CFG edge level analysis)
  - RQ3: Cost comparison (test selection + running selected tests) vs. re-running all tests

Path Spectra [Reps et. al.1997]

- The use of program profiling for software maintenance with applications to the Y2K problem
- ESEC/FSE 1997

What is Program Profiling?

- Recording behavior of a program during execution
- What can you measure about a program?
  -

Program Profiling

- Memory usage; e.g., heap size over time. # of times a garbage collector was called.
- The depth of a stack, etc.
- Coverage
  - Function coverage: Has each function been executed?
  - Statement coverage: Has each statement been executed?
  - Branch coverage: Has each control structure evaluated both true and false?
  - Path coverage: Has every possible route been executed?

Motivation of Reps et al.

- Y2K problem
  - Would my program have erroneous behavior when run on input year = 2001?
  - => Would my program exercise a different path during program execution in comparison to input year= {1900, 1901, 1902, ... 1999}?
  - => How can we concisely represent path profiles for a set of inputs (in order to do this profile comparison)?

Research Problem addressed by Reps et al.

- Given two different sets of inputs for the same program, how can we reason about path-profile differences (divergences?
- What is an appropriate representation for reasoning about program path profiles for a set of inputs?
- What is an efficient numbering scheme for loop-free paths?

Recap (1)

- Software evolution may introduce regression faults.
- Regression testing intends to check preservation of desirable program behavior and to prevent undesirable program behavior (regression faults) through testing.
- Given a test suite T, two program versions, RTS selects a subset of T that have a potential to reveal regression faults.
- RTS needs three building tools: (1) program differencing tool, (2) coverage gathering tool, and (3) test selection algorithm.

Recap (2)

- Regression testing is an exciting research area with practical impact on software evolution.
    - Test Selection
    - Test Prioritization
    - Test Minimization
    - Test Generation & Augmentation

Future Direction: Behavior Differencing

- I am *personally* excited about this problem.
- Given a test suite T, and two program versions Po and Pn
  - What is an appropriate representation of behavioral differences caused by source code change between Po and Pn?
  - How can we effectively identify behavioral differences with respect to such representation?
  - Can we use similarities (systematicness) among individual differences to concisely represent the differences? If so, can inconsistencies be used for identifying potential bugs introduced by code modifications?

Preview for Next Week

- Change Impact Analysis by Ren et al. OOPSLA 2004
- We will move on to a new topic, reverse engineering and knowledge discovery => software metrics & visualization
  - Murphy et al. Software Reflexion Model (Wed, 4/15)
  - Lanza et al. Polymetric Views (Mon, 4/20)
- Literature Survey and Project Final Report Draft is due on Apr 21 Tuesday. Less than 2 weeks from now.
- I will publish grading guidelines for the literature survey & project final report.