

# Software Debugging, and ... analysis methods with that view

Abhik Roychoudhury  
National University of Singapore  
(Slides from Tao Wang's guest lecture)

IISc Summer Course 2007 by  
Abhik Roychoudhury

## Outline

- Introduction
- Program Slicing
- Testing based Fault Localization
- Summary

IISc Summer Course 2007 by  
Abhik Roychoudhury

## Introduction

- Software Debugging
  - Time consuming

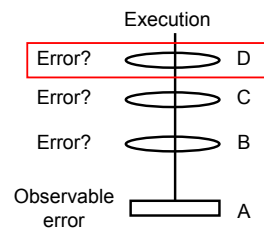
- Automatic methods for syntax errors  
**Semantic errors?**

String s1 = ...  
String s2 = ...  
...  
If (s1 == s2)  
.....

IISc Summer Course 2007 by  
Abhik Roychoudhury

## Software Debugging

- Typical Debugging Steps



1. Hypothesize the cause of the error
2. Try to confirm it

IISc Summer Course 2007 by  
Abhik Roychoudhury

## Software Debugging

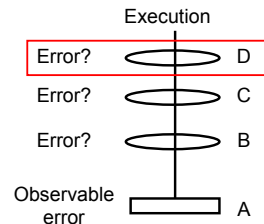
1. Hypothesize the cause of the error
  - Program analysis
  - Identify suspicious statements
2. Try to confirm it



IISc Summer Course 2007 by  
Abhik Roychoudhury

## Software Debugging

- Typical Debugging Steps



1. Hypothesize the cause of the error
2. Try to confirm it

IISc Summer Course 2007 by  
Abhik Roychoudhury

## Software Debugging

### 1. Hypothesize the cause of the error

- Program analysis
- Identify suspicious statements



### 2. Try to confirm it

- Collect information to help programmers



IISc Summer Course 2007 by  
Abhik Roychoudhury

## Program Analysis for Debugging

### ■ What to analyze?

- the execution run, Dynamic Analysis
- the source code, Static Analysis

### ■ How to analyze?

- Program Slicing
- Testing based Fault Localization

IISc Summer Course 2007 by  
Abhik Roychoudhury

## Outline

- Introduction
- Program Slicing
- Testing based Fault Localization
- Summary

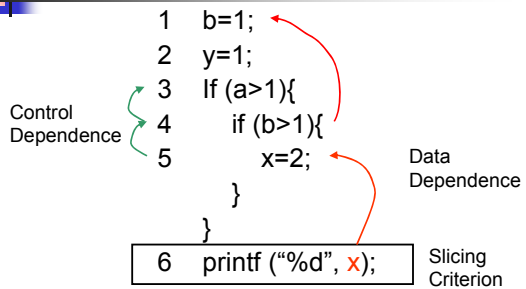
IISc Summer Course 2007 by  
Abhik Roychoudhury

## Program Slicing

```
1 b=1;
2 y=1;
3 If (a>1){
4     if (b>1){
5         x=2;
6     }
7 }
8 printf ("%d", x);
```

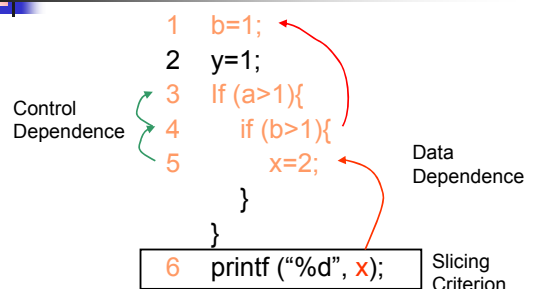
IISc Summer Course 2007 by  
Abhik Roychoudhury

## Program Slicing

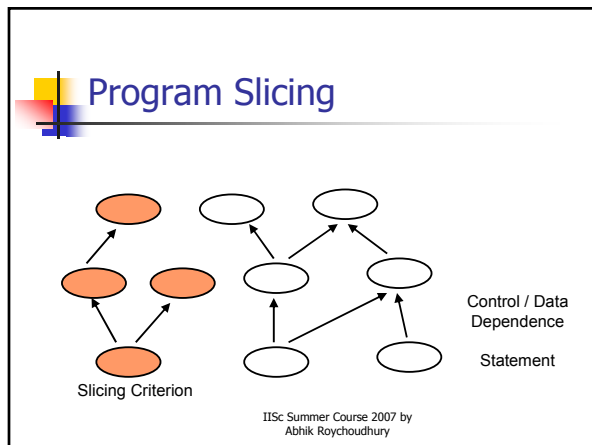
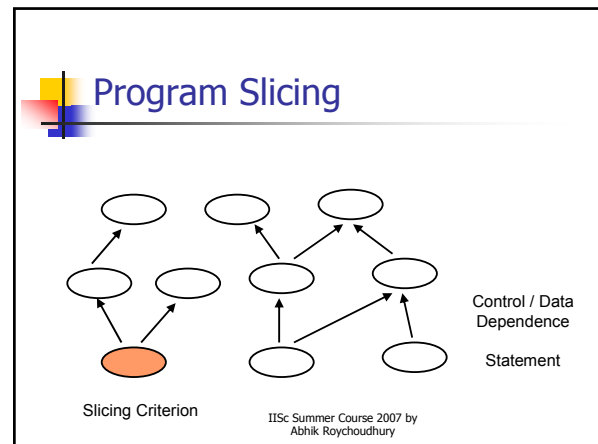
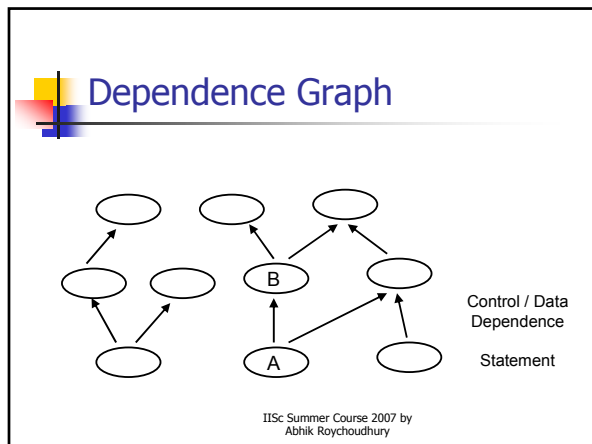


IISc Summer Course 2007 by  
Abhik Roychoudhury

## Program Slicing



IISc Summer Course 2007 by  
Abhik Roychoudhury



- ## Static vs Dynamic Slicing
- Static Slicing
    - source code
    - statement
    - static dependence
  - Dynamic Slicing
    - a particular execution
    - statement instance
    - dynamic dependence
- IISc Summer Course 2007 by Abhik Roychoudhury

## Static vs Dynamic Slicing

```

1  b=1;
2  if (a>1)
3    x=1;
4  else
5    x=2;
6  printf ("%d", x);
  
```

Slicing Criterion

IISc Summer Course 2007 by Abhik Roychoudhury

## Static vs Dynamic Slicing

```

1  p.f = 1;
2  x= q.f;
3  printf ("%d", x);
  
```

Slicing Criterion

p and q point to the same object?

- Static points-to analysis is always conservative

IISc Summer Course 2007 by Abhik Roychoudhury

## Relevant Slicing

```

1 b=10;
2 x=1;
3 If (a>1){
4   if (b>1){
5     x=2;
6   }
7 }
8 printf ("%d", x);

```

IISc Summer Course 2007 by  
Abhik Roychoudhury

## Relevant Slicing

```

1 b=1;
2 x=1;
3 If (a>1){
4   if (b>1){
5     x=2;
6   }
7 }
8 printf ("%d", x);

```

IISc Summer Course 2007 by  
Abhik Roychoudhury

## Relevant Slicing

input: a=2

Source of Failure → 1 b=1;  
Dynamic Slice → 2 x=1;  
Execution is omitted → 3 If (a>1){  
4 if (b>1){  
5 x=2;  
6 }  
7 }  
8 printf ("%d", x);

IISc Summer Course 2007 by  
Abhik Roychoudhury

## Potential Dependence

input: a=2

```

1 b=1;
2 x=1;
3 If (a>1){
4   if (b>1){
5     x=2;
6   }
7 }
8 printf ("%d", x);

```

IISc Summer Course 2007 by  
Abhik Roychoudhury

## Relevant Slice

input: a=2

Potential Dependence → 1 b=1;  
Dynamic Data Dependence → 2 x=1;  
3 If (a>1){  
4 if (b>1){  
5 x=2;  
6 }  
7 }  
8 printf ("%d", x);

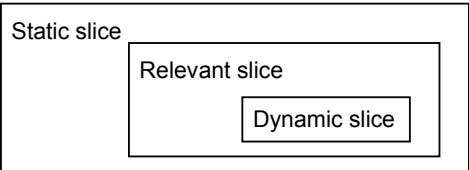
IISc Summer Course 2007 by  
Abhik Roychoudhury

## Program Slice

| Static | Dynamic | Relevant | Code                |
|--------|---------|----------|---------------------|
| 1      |         | 1        | 1 b=1; input: a=2   |
| 2      | 2       | 2        | 2 x=1;              |
| 3      |         | 3        | 3 If (a>1){         |
| 4      |         | 4        | 4 if (b>1){         |
| 5      |         | 5        | 5 x=2;              |
|        |         |          | }                   |
| 6      | 6       | 6        | 6 printf ("%d", x); |

IISc Summer Course 2007 by  
Abhik Roychoudhury

## Program Slicing



Static slice

Relevant slice

Dynamic slice

IISc Summer Course 2007 by  
Abhik Roychoudhury

## Backward Slicing

Backward Dynamic Slicing:

- Run the program for selected input.
- Store execution trace.
- Set slicing criterion.
- Traverse the trace from the slicing criterion to the beginning of the trace by going through control/data dependencies.

IISc Summer Course 2007 by  
Abhik Roychoudhury

## Forward Dynamic Slicing

- No need to store trace.
- Execute the program for selected input.
- At every step, compute and update the slice (from among statements seen so far), for different slicing criteria
  - What is that?
  - Various variables for the current line

IISc Summer Course 2007 by  
Abhik Roychoudhury

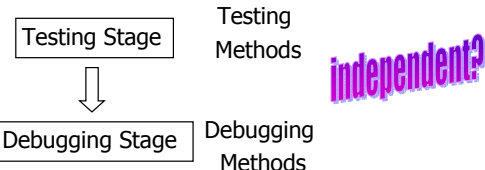
## Comparison

|   |   |
|---|---|
| <ul style="list-style-type: none"> <li>Forward           <ul style="list-style-type: none"> <li>No need to store trace</li> <li>Store slices for many slicing criteria               <ul style="list-style-type: none"> <li>Can use BDDs (paper by Zhang and Gupta in ICSE 2004)</li> </ul> </li> </ul> </li> </ul> | <ul style="list-style-type: none"> <li>Backward           <ul style="list-style-type: none"> <li>Store trace               <ul style="list-style-type: none"> <li>Can employ on-line compression (our paper in ICSE 2004)</li> </ul> </li> <li>Goal-directed, compute only the needed slice.</li> </ul> </li> </ul> |
|---|---|

IISc Summer Course 2007 by  
Abhik Roychoudhury

## Testing and Debugging

- Two very close software development activities



Testing Stage

Testing Methods

Debugging Stage

Debugging Methods

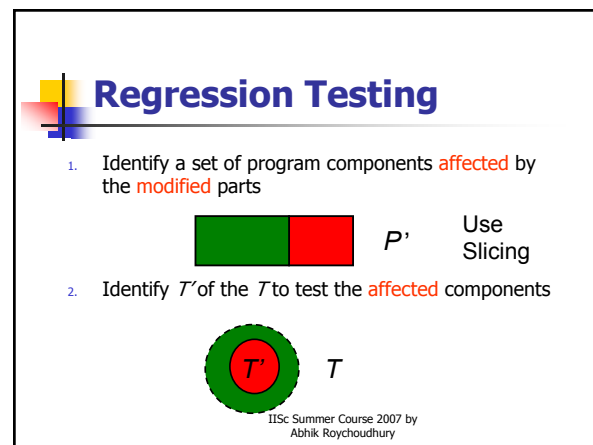
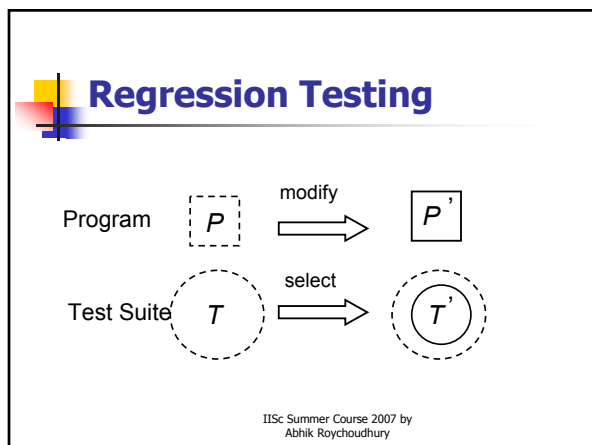
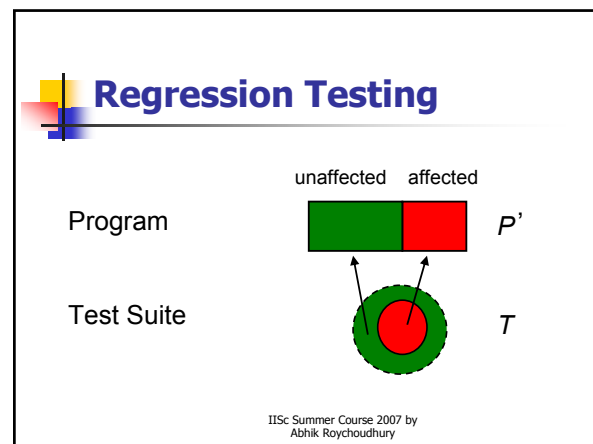
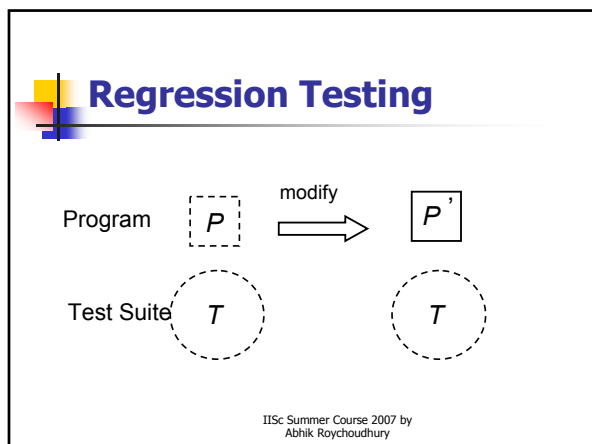
*independent?*

IISc Summer Course 2007 by  
Abhik Roychoudhury

## Testing and Debugging

- Testing and Debugging techniques can help each other
- Improve testing
  - Forward Slicing for Regression Testing

IISc Summer Course 2007 by  
Abhik Roychoudhury



## Forward Slicing for Regression Testing

|                      | Test 1, $b=0$ | Test 2, $b=5$ |
|----------------------|---------------|---------------|
| 1 $a=1;$             | 1             | 1             |
| 2 $\text{if } (b>1)$ | 2             | 2             |
| 3 $y=x+10;$          |               | 3             |
| 4 $b=2;$             | 4             | 4             |

$P$

IISc Summer Course 2007 by  
Abhik Roychoudhury

## Forward Slicing for Regression Testing

|                      | Test 1, $b=0$ | Test 2, $b=5$ |
|----------------------|---------------|---------------|
| 1 $a=1;$             | 1             | 1             |
| 2 $x=1;$             |               |               |
| 3 $\text{if } (b>1)$ | 2             | 2             |
| 4 $y=x+10;$          |               | 3             |
| 5 $b=2;$             | 4             | 4             |

$P'$

IISc Summer Course 2007 by  
Abhik Roychoudhury

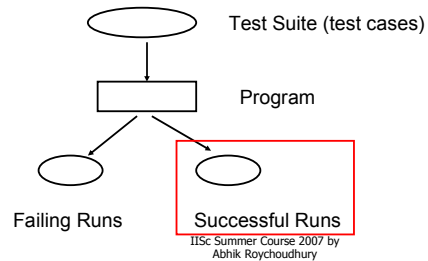
## Outline

- Introduction
- Program Slicing
- **Testing based Fault Localization**
- Summary

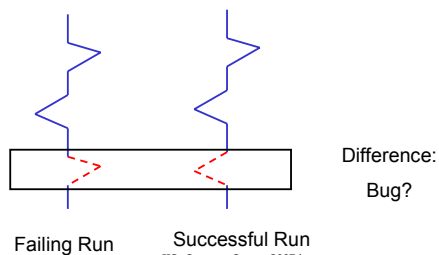
IISc Summer Course 2007 by  
Abhik Roychoudhury

## Testing Based Fault Localization

- Use testing to help debugging



## Testing Based Fault Localization



IISc Summer Course 2007 by  
Abhik Roychoudhury

## Testing Based Fault Localization

- What to Compare
  - choice of the Execution Run

IISc Summer Course 2007 by  
Abhik Roychoudhury

## Testing Based Fault Localization

- What to Compare
  - choice of the Execution Run
- How to Compare

IISc Summer Course 2007 by  
Abhik Roychoudhury

## Choice of the Execution Run

- Failing run
  - Select one failing run for comparison
  - Different failing runs may correspond to different error causes
- Successful Run
  - A **Single** successful run, VS
  - A **Set** of successful runs

IISc Summer Course 2007 by  
Abhik Roychoudhury

## Choice of a Single Successful Run

- The difference can be related to:
  - different inputs
  - the error
- The successful run and the failing run should be as similar as possible
  - Choose a "good" successful run for comparison

IISc Summer Course 2007 by  
Abhik Roychoudhury

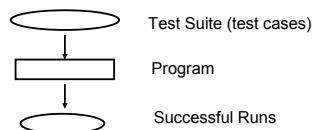
## Testing Based Fault Localization

- What to Compare
  - choice of the Execution Run
- How to Compare
  - statement / basic block
  - predicates / branch statement
  - potential invariants
  - variable values

IISc Summer Course 2007 by  
Abhik Roychoudhury

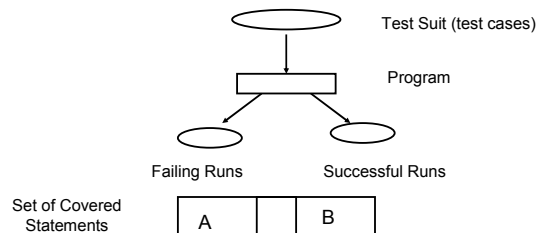
## Fault Localization - statement

- Statement Coverage for Testing
  - every statement should be executed at least once with test cases in the Test Suite
  - intuition for this Coverage Criteria



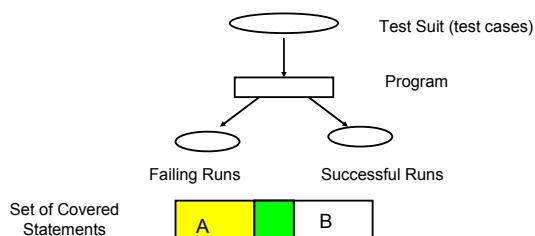
IISc Summer Course 2007 by  
Abhik Roychoudhury

## Fault Localization - statement



IISc Summer Course 2007 by  
Abhik Roychoudhury

## Fault Localization - statement



IISc Summer Course 2007 by  
Abhik Roychoudhury

## Fault Localization - branches

```

1. v=0;
2. if (x>0) → if (x>=0)
3. u=5;
4. else
5. u=v;
6. printf("%d",u);
    
```

IISc Summer Course 2007 by  
Abhik Roychoudhury



## Fault Localization - branches

**Failing run,  $x=0$**

1. `v=0;`
2. `if (x>0)`
3. `u=5;`
4. `else`
5. `u=v;`
6. `printf("%d",u);`

**Successful run,  $x=1$**

1. `v=0;`
2. `if (x>0)`
3. `u=5;`
4. `else`
5. `u=v;`
6. `printf("%d",u);`

IISc Summer Course 2007 by Abhik Roychoudhury

## Compare Corresponding Statement Instances

1. `while (a){`
2. `if (b)`
3. `i++;`
4. `}`

*Program*

IISc Summer Course 2007 by Abhik Roychoudhury

## Compare Corresponding Statement Instances

1. `while (a){`
2. `if (b)`
3. `i++;`
4. `}`

Execution run  $\pi$

1. `while (a){`
2. `if (b)`
3. `i++;`
4. `}`

Execution run  $\pi'$

IISc Summer Course 2007 by Abhik Roychoudhury

## Compare Corresponding Statement Instances

1. `while (a){`
2. `if (b)`
3. `i++;`
4. `}`

Execution run  $\pi$

1. `while (a){`
2. `if (b)`
3. `i++;`
4. `}`

Execution run  $\pi'$

IISc Summer Course 2007 by Abhik Roychoudhury

## Fault Localization - branches

1. `while (a){`
2. `if (b)`
3. `i++;`
4. `}`

Execution run  $\pi$

1. `while (a){`
2. `if (b)`
3. `i++;`
4. `}`

Execution run  $\pi'$

IISc Summer Course 2007 by Abhik Roychoudhury

## Choose a Successful Run

Comparison of Differences Select one failing run for comparison

Failing  $\pi$     Successful  $\pi'$

Failing  $\pi$     Successful  $\pi'$

IISc Summer Course 2007 by Abhik Roychoudhury

## Choose a Successful Run

- Comparison of Differences Select one failing run for comparison

IISc Summer Course 2007 by Abhik Roychoudhury

## Location of Branches

TCAS program

```

1. int main(int argc, char **argv)
2. if (argc < 3){
3.     printf("parameter error\n");
4.     return 0;
5. }
6.
7. if (m == -1)
8.     ....
9. }

```

check the input

Favor branches near to the observable error

IISc Summer Course 2007 by Abhik Roychoudhury

## Choose a Successful Run

- Comparison of Differences Select one failing run for comparison

IISc Summer Course 2007 by Abhik Roychoudhury

## Choose a Successful Run

- Comparison of Differences Select one failing run for comparison

IISc Summer Course 2007 by Abhik Roychoudhury

## Fault Localization - invariants

- Collect potential invariants from a set of successful runs
  - such as:  $x > 0$  at the beginning of the method `foo()`
- Invariant vs Specification
  - informal vs formal
  - Invariant detector --- Daikon
    - <http://pag.csail.mit.edu/daikon/>

IISc Summer Course 2007 by Abhik Roychoudhury

## Fault Localization - variable values

IISc Summer Course 2007 by Abhik Roychoudhury

## Fault Localization - variable values

- Locate an actual error in GCC
- Problems:
  - Mixed state, feasible or infeasible?
  - Map variables, Pointer, Array

IISc Summer Course 2007 by  
Abhik Roychoudhury

## Summary

- Computer-Aided Debugging
  - Automatically identify suspicious statements
- Future Trends
  - More accurate results
  - Novel way to use results of existing methods

IISc Summer Course 2007 by  
Abhik Roychoudhury

## Tools

- Testing
  - Numerous, Junit for unit testing
- Slicing
  - Several tools for static slicing
    - Codesurfer, Indus
  - Dynamic slicing – Jslice  
<http://jslice.sourceforge.net>
    - More naturally corresponds to debugging, than static slicing
- Fault Localization
  - Andreas Zeller's Askigor system

IISc Summer Course 2007 by  
Abhik Roychoudhury