National Aeronautics and Space Administration

Directed Incremental Symbolic Execution

Suzette Person
Research Computer Scientist, NASA Langley Research Center

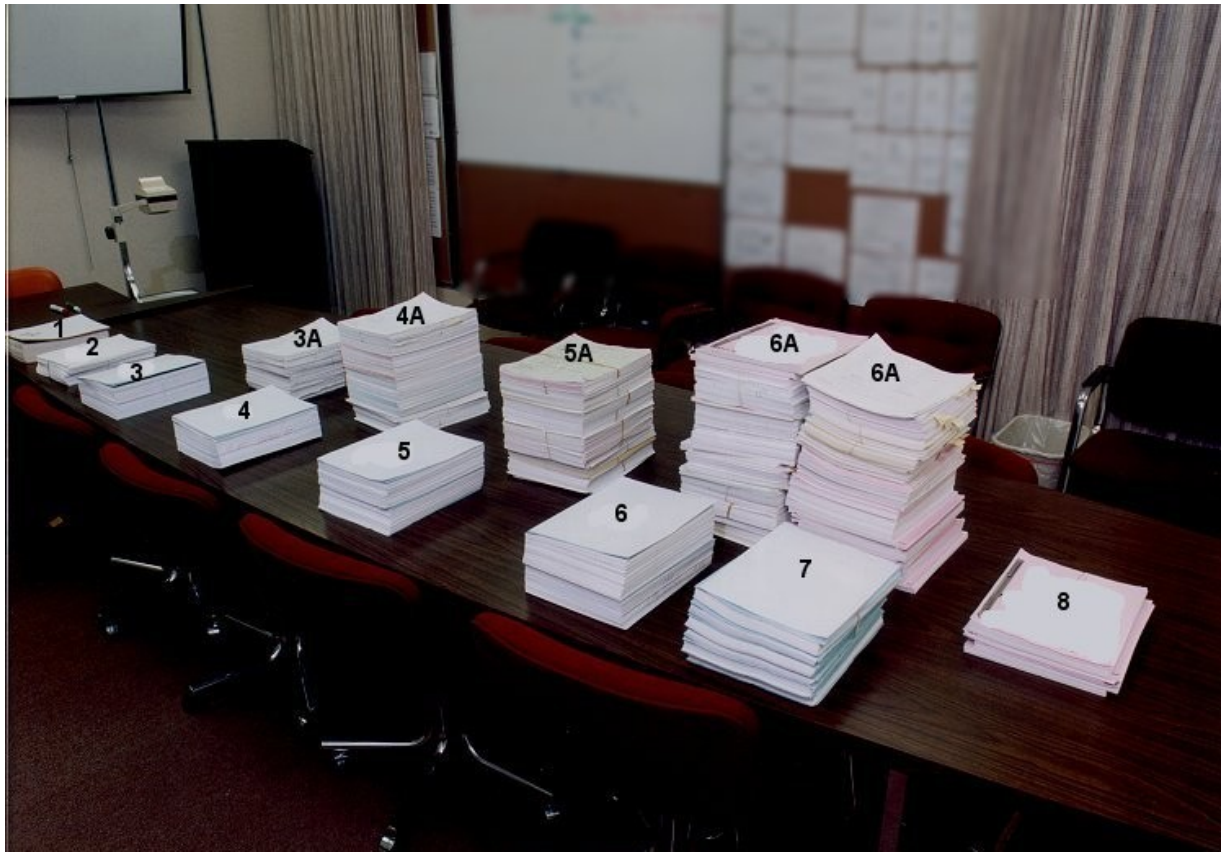
2011 Annual Technical Meeting
May 10–12, 2011
St. Louis, MO

Motivation

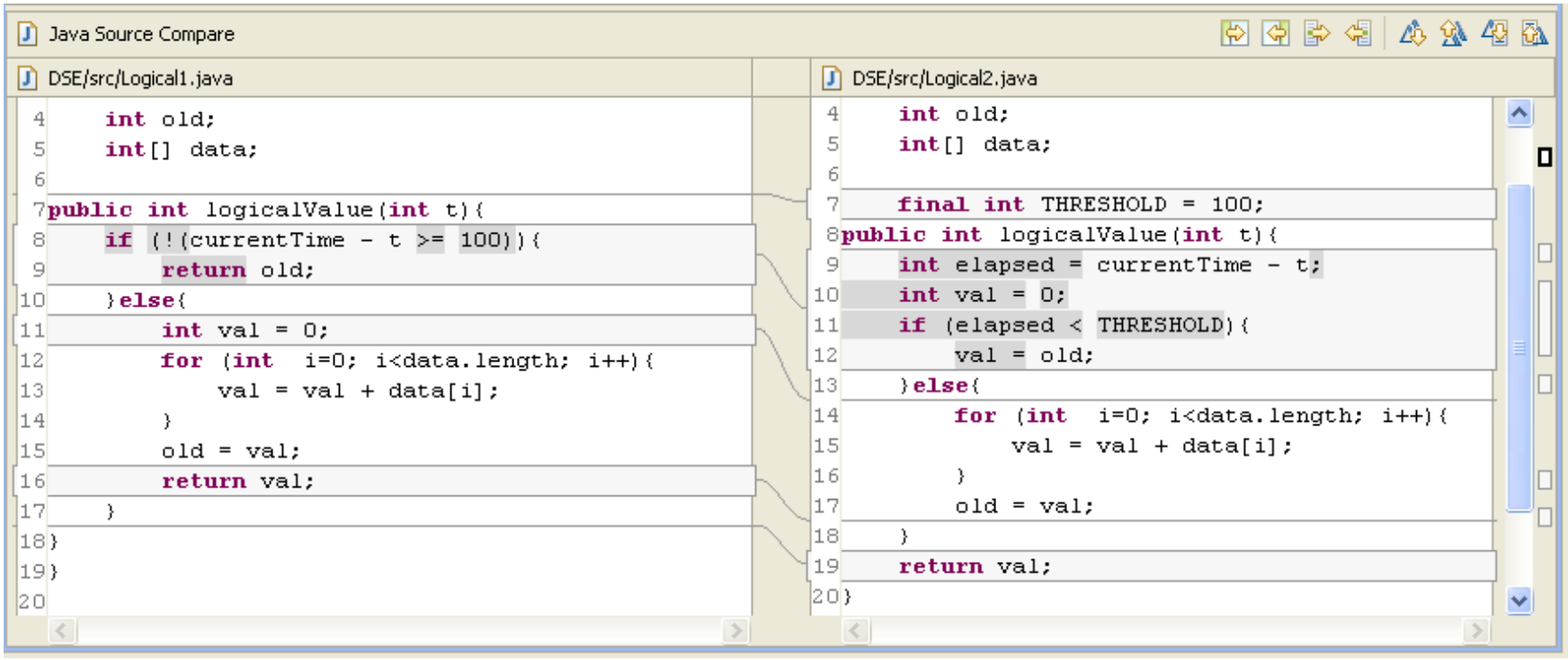
"Nothing endures but change."

Heraclitus, Greek
Philosopher
(c. 535 BC – 475 BC)

Motivation



Motivation




```
4     int old;
5     int[] data;
6
7  public int logicalValue(int t){
8     if (!(currentTime - t >= 100)){
9         return old;
10    }else{
11        int val = 0;
12        for (int  i=0; i<data.length; i++){
13            val = val + data[i];
14        }
15        old = val;
16        return val;
17    }
18 }
19 }
20
```

```
4     int old;
5     int[] data;
6
7     final int THRESHOLD = 100;
8  public int logicalValue(int t){
9     int elapsed = currentTime - t;
10    int val = 0;
11    if (elapsed < THRESHOLD){
12        val = old;
13    }else{
14        for (int  i=0; i<data.length; i++){
15            val = val + data[i];
16        }
17        old = val;
18    }
19    return val;
20 }
```

DiSE

- Regression analysis technique focused on version differences
- Leverages efficiencies of syntactic analysis results to guide semantic analysis
- Identifies and characterizes the *effects* of program changes

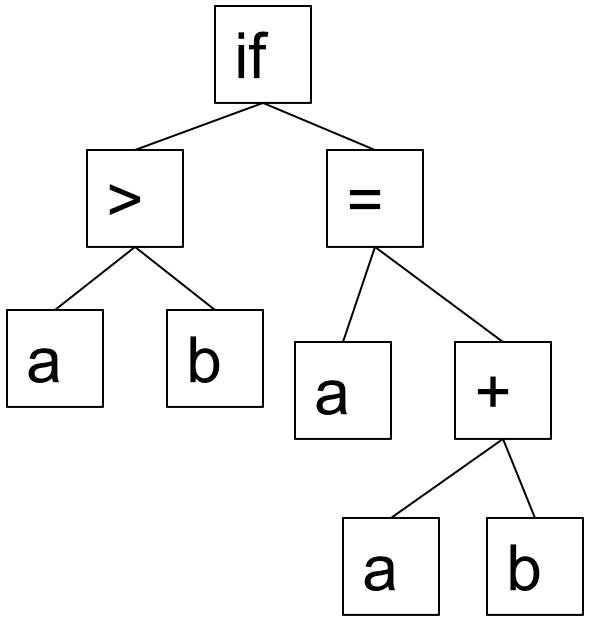
Overview of Presentation

- Motivation
- Background
- DiSE Methodology
- Evaluation
- Conclusions and Future Work

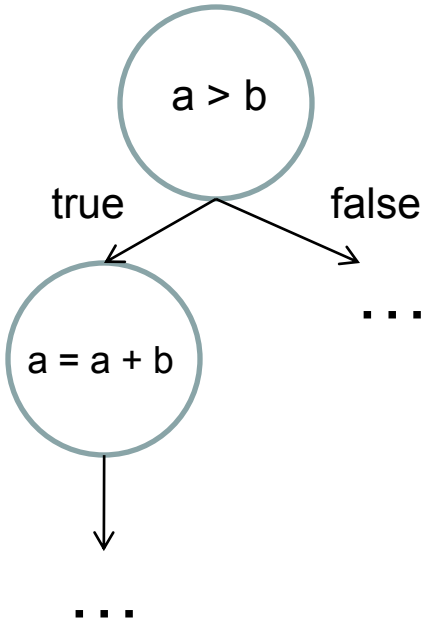
Background

- Abstract Syntax Tree

```
if (a > b)
  a = a + b;
```



- Control Flow Graph

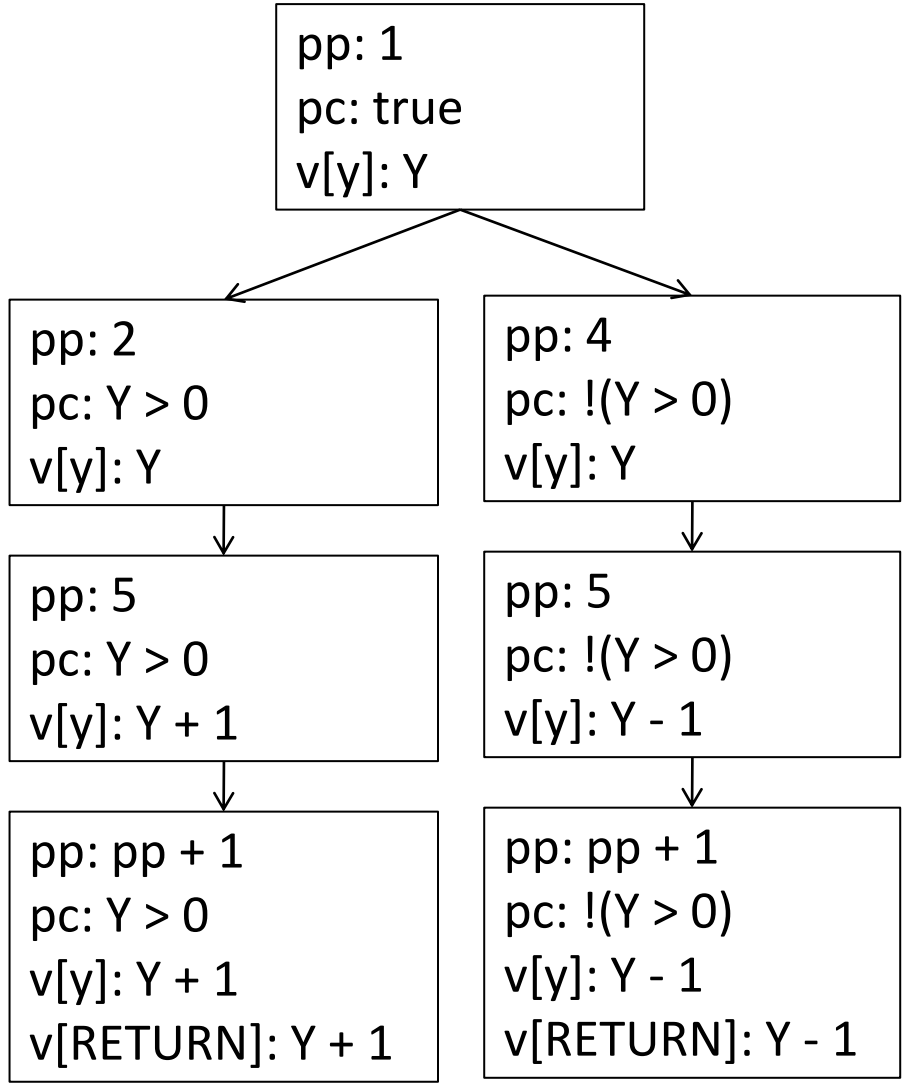


Background

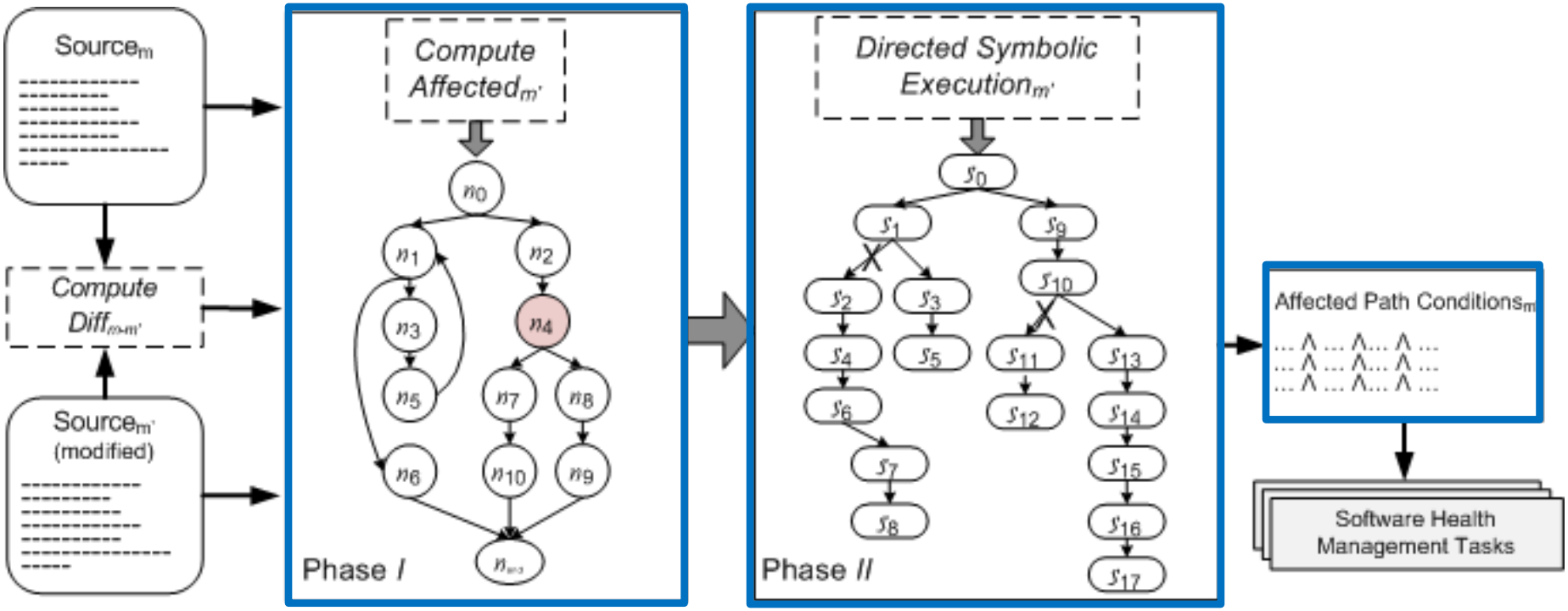
- Symbolic Execution

```
int m(int y){
1:  if (y>0)
2:    y++;
3:  else
4:    y--;
5:  return y;
}
```

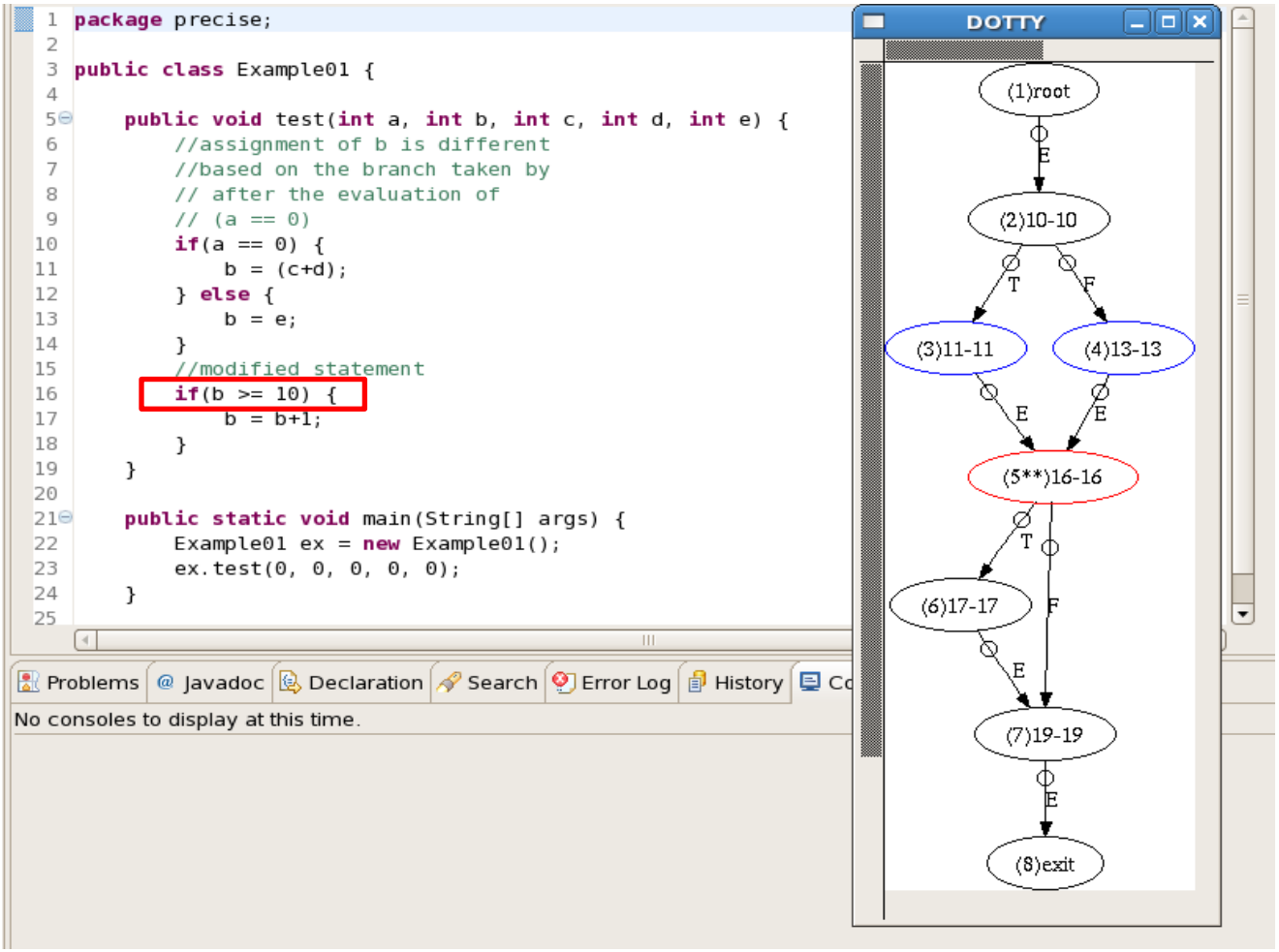
m_{sum}=
{((Y>0), RETURN=Y+1),
!(Y>0), RETURN=Y-1)}



DiSE Methodology

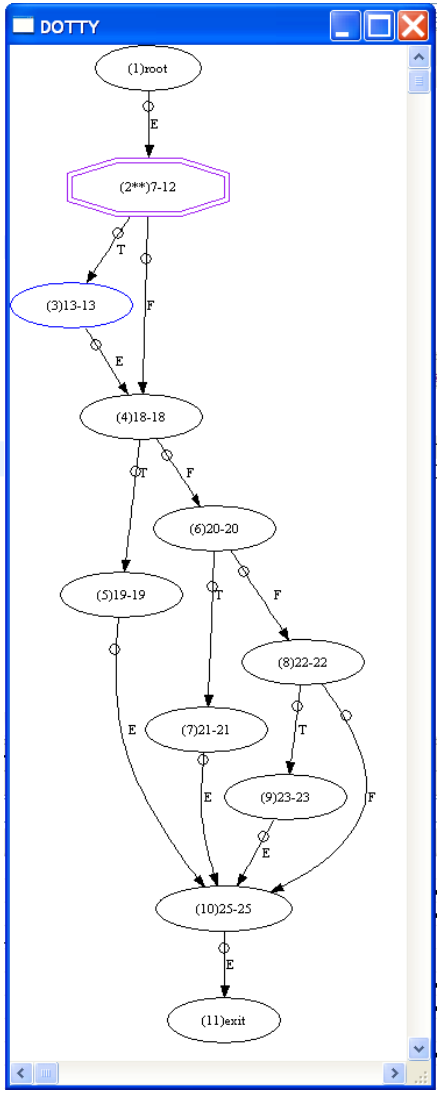


DiSE Methodology



DiSE Methodology

```
1  package precise;
2
3  public class Example05_mod{
4
5      public void test (int a, int b, int c, int d, int x) {
6          //modified statement
7          b = b - x;
8          int e = (a + b);
9          int f = (e - x);
10         // conditional branch statement
11         // affected by the change
12         if((e + f) == (c+d)) {
13             e = f;
14         }
15         // no path conditions should be generated
16         // during this set of conditional branch
17         // statements
18         if (c == d) {
19             c = d+1;
20         } else if (c < d) {
21             c = d+2;
22         } else if (c > d ) {
23             c = d+3;
24         }
25     }
26
```



2 affected path conditions

Evaluation

- Implementation
  - Custom application to compare ASTs
  - Custom data and control dependence analysis
  - Extension to NASA Ames Java PathFinder toolset
    - Leverages Symbolic PathFinder extension

Evaluation

- Artifacts
  - Three control applications written in Java
    - Wheel Brake System (WBS)
    - On-board Abort Executive (OAE)
    - Altitude Switch (ASW)
  - Manually created version history

Evaluation

- Research Questions

  RQ1 (Cost): How does the cost of applying DiSE compare with full symbolic execution of the changed method?

  RQ2 (Efficiency): How does the number of path conditions generated by DiSE compare with the number of path conditions generated by full symbolic execution?

Evaluation

- Observations
  - DiSE incurs little overhead for all three examples
  - For most version differences, DiSE computed many fewer Path Conditions
  - Factors affecting reductions
    - Number of changes
    - Location and nature of the change
    - Program structure

Evaluation

- Limitations
  - DiSE performs an intra-procedural dependence analysis
    - Does not consider the effects of the return value of a method
    - May miss affected path conditions that affect the global state

Conclusions and Future Work

- DiSE combines *efficiencies* of lightweight static analysis techniques with *precision* of symbolic execution to explore and characterize the parts of a program impacted by changes
  - General technique that does not require analysis results to be carried forward
  - Evaluation demonstrates effectiveness of DiSE relative to full symbolic execution

Conclusions and Future Work

- Extend DiSE
  - Inter-procedural analysis
  - Leverage information from previous runs
- Evaluate on wider range of programs
- Explore other applications of DiSE results

National Aeronautics and Space Administration

Directed Incremental Symbolic Execution

Suzette Person
Research Computer Scientist, NASA Langley Research Center

2011 Annual Technical Meeting
May 10–12, 2011
St. Louis, MO