**Relay Model**

- Selective semantic information + syntactic dependency information
  - origination of a fault
  - computational transfer of a fault
  - propagation of a fault (based on data and control flow)
- Define necessary and sufficient conditions for detecting certain classes of faults
Overview of Relay Model

- **origination**
  - introduction of potential failure at smallest (valued) subexpression containing fault

- **transfer**
  - “movement” of potential failure in program
    - Within the originating statement
      - computational transfer
    - From one statement to the next
      - data dependence transfer
      - control dependence transfer
Relay Model

```
observable
failure :=
```

```
faulttransfer := transfer
```

```
transfer
```

```
“observable”
failure :=
```

```
transfer
```
Example

Correct:

1. input B, C
2. \( A = C \times (B + 1)^2 \)
3. \( D = (A \times B) + C \)
4. \( X = B \times C \)
5. \( D < X - 5 \)
6. \( Y = (X \times 2) + C \)
7. \( Y = (2 \times X) + C \)

What test data would reveal this fault?
Example

1. input B, C
2. A: = C * (B + 1)
3. D: = (A * B) + C
4. X: = B * C
5. D < X - 5
6. Y: = (X ** 2) + C
7. Y: = (2 * X) + C

Module t. c. exp a d x d < x - 5 y output
B+C
faulty 1 1 2 2 3 1 F 2 2
B+1 correct 1 1 2 2 3 1 F 2 2

NO ORIGINATION OF POTENTIAL FAILURE AT NODE 2
Example

1. input B, C
2. \( A := C \times (B + 1) \)
3. \( D := (A \times B) + C \)
4. \( X := B \times C \)
5. \( D < X - 5 \)
6. \( Y := (X \times 2) + C \)
7. \( Y := (2 \times X) + C \)

<table>
<thead>
<tr>
<th>module</th>
<th>t. c.</th>
<th>exp</th>
<th>a</th>
<th>d</th>
<th>x</th>
<th>d &lt; x - 5</th>
<th>y</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>B+C</td>
<td>faulty</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>F</td>
<td>0</td>
</tr>
<tr>
<td>B+1</td>
<td>correct</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>F</td>
<td>0</td>
</tr>
</tbody>
</table>

ORIGINATION (NODE 2), NO COMPUTATIONAL TRANSFER AT NODE 2
Example

1. input B, C
2. A: = C * (B + 1)
3. D: = (A * B) + C
4. X: = B * C
5. D < X - 5
6. Y: = (X ** 2) + C
7. Y: = (2 * X) + C

<table>
<thead>
<tr>
<th>module</th>
<th>t. c.</th>
<th>exp</th>
<th>a</th>
<th>d</th>
<th>x</th>
<th>d &lt; x - 5</th>
<th>y</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>B+C</td>
<td>faulty</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>9</td>
<td>3</td>
<td>0</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>correct</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>F</td>
</tr>
</tbody>
</table>

ORIGINATION, COMP. TRANSFER (NODE 2)
NO DATA DEPENDENCE TRANSFER AT NODE 3
Example

1. input B, C
2. A: = C * (B + C)
3. D: = (A * B) + C
4. X: = B * C
5. D < X - 5

FALSE
6. Y: = (X ** 2) + C

TRUE
7. Y: = (2 * X) + C

output Y

<table>
<thead>
<tr>
<th>module</th>
<th>t. c.</th>
<th>exp</th>
<th>a</th>
<th>d</th>
<th>x</th>
<th>d &lt; x - 5</th>
<th>y</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>B+C</td>
<td>b</td>
<td>c</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>15</td>
<td>33</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>faulty</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B+1</td>
<td>b</td>
<td>c</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>9</td>
<td>21</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td>correct</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ORIGINATION, COMP. TRANSFER (NODE 2)
DATA DEPENDENCE TRANSFER (NODE 3)
NO DATA DEPENDENCE TRANSFER AT NODE 5
Example

1. input B, C
2. \( A = C \times (B + 1) \)
3. \( D = (A \times B) + C \)
4. \( X = B \times C \)
5. \( D < X - 5 \)
6. \( Y = (X \times 2) + C \)
7. \( Y = (2 \times X) + C \)

FALSE \quad I \quad TRUE

FAULTY

CORRECT

Module | t. c. | exp | a | d | x | d < x - 5 | y | Output
-------|------|-----|---|---|---|-----------|---|-------
B+C    | faulty | -2  | -1  | -3  | 3  | -7  | 2  | T  | 3  | 3
B+C    | correct | -2  | -1  | -1  | 1  | -3  | 2  | F  | 3  | 3

ORIGINATION, COMP. TRANSFER (NODE 2)
DATA DEPENDENCE TRANSFER (NODES 3, 5)
NO CONTROL DEPENDENCE TRANSFER AT NODE 7 and 6
Example

1. input B, C
2. A: = C * (B + 1)
3. D: = (A * B) + C
4. X: = B * C
5. D < X - 5

FALSE

6. Y: = (X ** 2) + C

TRUE

7. Y: = (2 * X) + C

output Y

<table>
<thead>
<tr>
<th>module</th>
<th>t. c.</th>
<th>exp</th>
<th>a</th>
<th>d</th>
<th>x</th>
<th>d &lt; x - 5</th>
<th>y</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>B+C</td>
<td>faulty</td>
<td>1</td>
<td>-3</td>
<td>-2</td>
<td>6</td>
<td>3</td>
<td>F</td>
<td>6</td>
</tr>
<tr>
<td>B+1</td>
<td>correct</td>
<td>1</td>
<td>-3</td>
<td>2</td>
<td>-6</td>
<td>-9</td>
<td>T</td>
<td>-9</td>
</tr>
</tbody>
</table>

ORIGINATION, COMP. TRANSFER (NODE 2)
DATA DEPENDENCE TRANSFER (NODES 3, 5)
CONTROL DEPENDENCE TRANSFER (NODE 6)
FAILURE AT NODE 8
To Guarantee Detection

Step 1: guarantee introduction of potential failure at statement containing hypothetical fault
  • origination condition
  • computational transfer conditions at statement
  • called original state potential failure condition

Step 2: guarantee transfer of potential failure along information flow to some output
  • called transfer set condition
Step 1a

• origination condition
  • guarantees introduction of potential failure in smallest subexpression
  • \( \exp \neq \exp^* \)
  • defined for fault
  • suppose \( c^*(b+1) \) instead of \( c^*(b+c) \)
    \[ \Rightarrow c \neq 1 \]
Step 1b

- computational transfer condition for a statement
  - $\text{exp1} \ <\text{op}> \ \text{exp2} \neq \text{exp1}' \ <\text{op}> \ \text{exp2}$
  - defined for operator and fault
    - e.g., $(b + c) \neq (b + 1) \Rightarrow c \neq 1$
  - many are fault independent
    - $c \ast (\text{exp}) \neq c \ast (\text{exp}')$
      - $\Rightarrow c \neq 0$
Step 2

- Information flow transfer
  - combines data dependence and control dependence transfer
  - occurs along information flow chains
  - to guarantee transfer from (hypothetically) faulty node to output must guarantee transfer along transfer set
    - collection of information flow chains that can be executed together
**Simpler Example**

1. **input** $X, Y, Z$

2. $A := X + Y$

3. $B := A \times X$

4. $C := A \times Y$

5. $D := B \times C$

6. **output** $D$

条件:
- $Y \neq Z$
- $X \neq 0$
- $C \neq 0 \Rightarrow A \times Y \neq 0 \Rightarrow Y \neq 0 \land A \neq 0$
- $Y \neq 0 \land X + Y \neq 0 \Rightarrow Y \neq 0 \land X \neq -Y$
- $B \neq 0 \Rightarrow A \times X \neq 0 \Rightarrow$
Necessary but not sufficient?

1. input X, Y, Z
2. A := X + Y
3. B := A * X
4. C := A * Y
5. D := B * C
6. output D

Y ≠ Z
X ≠ 0
C ≠ 0 ⇒
A * Y ≠ 0 ⇒
Y ≠ 0 ∧ A ≠ 0 ⇒
Y ≠ 0 ∧ X + Y ≠ 0 ⇒
Y ≠ 0 ∧ X ≠ −Y
B ≠ 0 ⇒
A * X ≠ 0 ⇒
X ≠ 0 ∧ A ≠ 0 ⇒
X ≠ 0 ∧ X + Y ≠ 0 ⇒
X ≠ 0 ∧ X ≠ −Y

<table>
<thead>
<tr>
<th>module</th>
<th>test case</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>x+y</td>
<td>faulty</td>
<td>1</td>
<td>-3</td>
<td>1</td>
<td>-2</td>
<td>-2</td>
</tr>
<tr>
<td>x+z</td>
<td>correct</td>
<td>1</td>
<td>-3</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

[Image of the diagram]
input X, Y, Z

2* A: = X + Z

Y ≠ Z

module test case | a | b | c | d | output
--- | --- | --- | --- | --- | ---
x+y | x | y | z | | |
1 | faulty | 1 | -3 | 1 | -2 | -2 | 6 | -12 | -12
2 | correct | 1 | -3 | 1 | 2 | 2 | -6 | -12 | -12
x+z | | | | | | | | |
x+y | | | | | | | | |
2 | faulty | 1 | -1 | 1 | 0 | 0 | 0 | 0 | 0
x+z | correct | 1 | -1 | 1 | 2 | 2 | -2 | -4 | -4
Transfer Condition

- condition that guarantees transfer
  - must know points of interaction
    - places where two or more potential failures come together

- Transfer set defines locations of potential interaction
  - Notation: \((U_n, V_m)\) means faulty value for variable \(U\) at node \(n\) transfers to variable \(V\) at node \(m\)

- Transfer route defines chains of transfer set elements that can be combined to form a path
Example

- Transfer Set = 
  \{(A_2, B_3), (B_3, D_5), (D_5, \text{out}_6), (A_2, C_4) (C_4, D_5) \}

Notation: \((U_n, V_m)\) means faulty value for variable \(U\) at node \(n\) transfers to variable \(V\) at node \(m\)
Construction of Transfer Route

- different ways to transfer along same set, depending on which portions of chains transfer and which do not

- a transfer route is a subset of the nodes in a transfer set where transfer does and does not occur

- a transfer route defines where actual interactions occur
Example

• Transfer Set = 
  \{ (A_2, B_3), (B_3, D_5), (D_5, \text{out}_6), (A_2, C_4), (C_4, D_5) \}

• Transfer Routes
  1. (A transfers to B at 3) and  
     (A does not transfer to C at 4)  
     and (B transfers to D at 5)  
  2. (A does not transfer to B at 3)  
     and (A transfers to C at 4)  
     and (C transfers to D at 5)  
  3. (A transfers to B at 3) and  
     (A transfers to C at 4) and  
     (B and C transfer to D at 5)
Transfer Condition

1. Path Condition
   - guarantees execution of a particular transfer route
     - must guarantee execution of nodes in chain as well as def-clear paths between nodes

2. Transfer Route Condition
   - guarantees transfer for particular transfer route
     - computational transfer conditions at nodes in transfer route where transfer does occur
     - complement of computational transfer conditions at nodes where transfer does not occur
Transfer Routes for Example

1. (A transfers to B at 3) and
   (A does not transfer to C at 4) and
   (B transfers to D at 5)
2. (A does not transfer to B at 3) and
   (A transfers to C at 4) and
   (C transfers to D at 5)
3. (A transfers to B at 3) and
   (A transfers to C at 4) and
   (B and C transfer to D at 5)
Condition for First Transfer Route

(A transfers to B at 3) and
(A does not transfer to C at 4) and
(B transfers to D at 5)

- **Transfer Route Conditions:**
  \[ x \neq 0 \land y = 0 \land c \neq 0 \Rightarrow \]
  \[ x \neq 0 \land y = 0 \land a \cdot y \neq 0 \Rightarrow \text{false} \]

```
2* A := X + Z
```

```
1 input X, Y, Z
2 A := X + Y
3 B := A * X
4 C := A * Y
5 D := B * C
output D
```
Condition for Second Transfer Route

(A does not transfer to B at 3) and
(A transfers to C at 4) and
(C transfers to D at 5)

• Transfer Route Conditions:
  \( x = 0 \land y \neq 0 \land b \neq 0 \Rightarrow \)
  \( x = 0 \land y \neq 0 \land a \times x \neq 0 \Rightarrow \text{false} \)
Condition for Third Transfer Route

(A transfers to B at 3) and
(A transfers to C at 4) and
(B and C transfer to D at 5)

• Transfer Route Conditions:

\[ x \neq 0 \land y \neq 0 \land b \ast c \neq b' \ast c' \Rightarrow \]
\[ x \neq 0 \land y \neq 0 \land (a \ast x)(a \ast y) \neq (a' \ast x)(a' \ast y) \Rightarrow \]
\[ x \neq 0 \land y \neq 0 \land (x+y)x(x+y)y \neq (x+z)x(x+z)y \Rightarrow \]
\[ x \neq 0 \land y \neq 0 \land (x+y)^2 \neq (x+z)^2 \Rightarrow \]
\[ x \neq 0 \land y \neq 0 \land y \neq z \]

test case: \(x=1, y=1, z=2\) satisfies the conditions and causes the fault to be revealed
Example

1. input X, Y, Z

2. A := X + Y

3. B := A * X

4. C := A * Y

5. D := B * C

6. output D

2* A: = X + Z

A= 2
B= 2
C= 4
D= 8

A= 3
B= 3
C= 3
D= 9

test case: x=1, y= 1, z=2 satisfies the conditions x \neq 0 \land y \neq 0 \land y \neq z
and causes the fault to be revealed
Failure condition

failure condition =

original state potential failure condition and transfer condition

if test data satisfies failure condition (fc) and failure $\rightarrow$ fault
if test data satisfies fc and no failure $\rightarrow$ no fault
if can't satisfy fc $\rightarrow$ try another transfer set
if can't satisfy fc for all transfer sets $\rightarrow$ no fault
Relay Fault Based Approach

- recognizes what is needed to transfer to output
- other fault based techniques:
  - do not deal with how to select test data that transfers
  - may recognize need to transfer but provide no guidance in test data selection (assume transfer “usually” occurs)
  - do not consider control dependence
  - none discuss interactions for a single fault/multiple faults -- they assume that there is a single fault or if there is more than one that there is no interaction
Relay Fault Based Approach

- defines what is needed to reveal a fault at a statement
  - a general procedure that could be applied to any "atomic" fault
- defines what is needed to propagate erroneous values to output
  - a very negative result!
  - if interaction is not accounted for, then the constraints are neither necessary nor sufficient
  - assumptions about single faults are now very questionable
  - can not assume constraints are necessary