The following are modified versions of the publicly-available slides for Chapter 8 in the Ammann and Offutt Book, “Introduction to Software Testing” (http://www.cs.gmu.edu/~offutt/softwaretest)
Logic Expressions from Source

- Predicates are derived from decision statements
  - if, while, for, switch, do-while

- In programs, most predicates have less than four clauses
  - In fact, most have just one clause

- When a predicate only has one clause, CoC, ACC, and CC all collapse to predicate coverage (PC)
  - ACC is only useful with three or more clauses
Finding Values

- **Reachability**: Each test must reach the decision
- **Controllability**: Each test must cause the decision to have specific truth assignment
- **Internal variables**: Predicates variables that are not inputs

```java
public int checkVal(int x) {
    y = x*2;
    if (x>0)
        if ((x>10 && x<20) || y==50)
            return 1;
    else
        if ((x<-10 && x>-20) || y<-60)
            return 2;
    return 0;
}
```

- Reach: x > 0
- Control for FFT: x=25
- y is an internal variable

\[ y = \varphi \]
In-Class Exercise

• https://cs.gmu.edu/~offutt/softwaretest/java/Thermostat.java

1. Read the program together
2. Identify the predicates in the program
3. How many clauses are in each predicate?
import java.io.*;

public class Thermostat {

private int curTemp; // Current temperature reading
private int thresholdDiff; // Temp difference until heater on
private int timeSinceLastRun; // Time since heater stopped
private int minLag; // How long I need to wait
private boolean Override; // Has user overridden the program
private int overTemp; // OverridingTemp
private int runTime; // output of turnHeaterOn–how long to run
private boolean heaterOn; // output of turnHeaterOn – whether to run
private Period period; // morning, day, evening, or night
private DayType day; // week day or weekend day

// Decide whether to turn the heater on, and for how long.
public boolean turnHeaterOn (ProgrammedSettings pSet) {

}
int dTemp = pSet.getSetting(period, day);

if (((curTemp < dTemp - thresholdDiff) ||
(Override && curTemp < overTemp - thresholdDiff)) &&
(timeSinceLastRun > minLag))
{
    // Turn on the heater
    // How long? Assume 1 minute per degree (Fahrenheit)
    int timeNeeded = Math.abs(dTemp - curTemp);
    if (Override)
    {
        timeNeeded = Math.abs(overTemp - curTemp);
    }
    setTimeout(timeNeeded);
    setHeaterOn(true);
    return(true);
}
else
{
    setHeaterOn(false);
    return(false);
}
}

// End turnHeaterOn

The full class is in the book and on the book website.
Two Thermostat Predicates

28-30 : (((curTemp < dTemp - thresholdDiff) || (Override && curTemp < overTemp - thresholdDiff)) && timeSinceLastRun > minLag))

34 : (Override)

Simplify

a : curTemp < dTemp - thresholdDiff
b : Override
c : curTemp < overTemp - thresholdDiff
d : timeSinceLastRun > minLag)
28-30 : (a || (b && c)) && d
34 : b
Reachability for Thermostat Predicates

28-30 : True
34 : (a || (b && c)) && d

\[
curTemp < dTemp - thresholdDiff
\]

Need to solve for the internal variable \(dTemp\)

```java
pSet.getSetting (period, day);
```

```java
setSetting (Period.MORNING, DayType.WEEKDAY, 69);
setPeriod (Period.MORNING);
setDay (DayType.WEEKDAY);
```
Predicate Coverage (true)

\[(a \lor (b \land c)) \land d\]

a : true  b : true  c : true  d : true

\[a: \text{curTemp} < \text{dTemp} - \text{thresholdDiff} : \text{true}\]
\[b: \text{Override} : \text{true}\]
\[c: \text{curTemp} < \text{overTemp} - \text{thresholdDiff} : \text{true}\]
\[d: \text{timeSinceLastRun} > (\text{minLag}) : \text{true}\]

```java
thermo = new Thermostat(); // Needed object
settings = new ProgrammedSettings(); // Needed object
settings.setSetting(Period.MORNING, DayType.WEEKDAY, 69); // dTemp
thermo.setPeriod(Period.MORNING); // dTemp
thermo.setDay(DayType.WEEKDAY); // dTemp
thermo.setCurrentTemp(63); // clause a
thermo.setThresholdDiff(5); // clause a
thermo.setOverride(true); // clause b
thermo.setOverTemp(70); // clause c
thermo.setMinLag(10); // clause d
thermo.setTimeSinceLastRun(12); // clause d
assertTrue(thermo.turnHeaterOn(settings)); // Run test
```
Correlated Active Clause Coverage

Solve for $P_a$: $((a \lor (b \land\land c)) \land\land d)$

$P_a = ((T \lor (b \land\land c)) \land\land d) \oplus ((F \lor (b \land\land c)) \land\land d)$

$(T \land\land d) \oplus ((b \land\land c) \land\land d)$

$d \oplus ((b \land\land c) \land\land d)$

Identity: $(X \oplus y \land\land X \equiv \lnot y \land\land X)$

$(\lnot (b \land\land c)) \land\land d$
Solve for $P_b$: $((a \ || \ (b \ && \ c)) \ && \ d)$

Solve for $P_c$: $((a \ || \ (b \ && \ c)) \ && \ d)$

Solve for $P_d$: $((a \ || \ (b \ && \ c)) \ && \ d)$
### Six tests needed for CACC on Thermostat

\[(a \lor (b \land c)) \land d\]

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_a:</td>
<td>T</td>
<td>t</td>
<td>f</td>
</tr>
<tr>
<td>P_b:</td>
<td>f</td>
<td>T</td>
<td>t</td>
</tr>
<tr>
<td>P_c:</td>
<td>f</td>
<td>t</td>
<td>T</td>
</tr>
<tr>
<td>P_d:</td>
<td>t</td>
<td>t</td>
<td>t</td>
</tr>
</tbody>
</table>

Duplicates:

- P_a
- P_c

© Ammann & Offutt
CACC Values for Clauses

<table>
<thead>
<tr>
<th>Condition</th>
<th>curTemp</th>
<th>dTemp</th>
<th>thresholdDiff</th>
</tr>
</thead>
<tbody>
<tr>
<td>a=t : curTemp &lt; dTemp - thresholdDiff</td>
<td>63</td>
<td>69</td>
<td>5</td>
</tr>
<tr>
<td>a=f : !(curTemp &lt; dTemp - thresholdDiff)</td>
<td>66</td>
<td>69</td>
<td>5</td>
</tr>
</tbody>
</table>

dT emp:
- settings.setSettings (Period.MORNING, DayType.WEEKDAY, 69)
- thermo.setPeriod (Period.MORNING);
- thermo.setDay (Daytype.WEEKDAY);

Override
- b=t : Override T
- b=f : !Override F

<table>
<thead>
<tr>
<th>Condition</th>
<th>curTemp</th>
<th>overTemp</th>
<th>thresholdDiff</th>
</tr>
</thead>
<tbody>
<tr>
<td>c=t : curTemp &lt; overTemp - thresholdDiff</td>
<td>63</td>
<td>72</td>
<td>5</td>
</tr>
<tr>
<td>c=f : !(curTemp &lt; overTemp - thresholdDiff)</td>
<td>66</td>
<td>67</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Condition</th>
<th>timeSinceLastRun</th>
<th>minLag</th>
</tr>
</thead>
<tbody>
<tr>
<td>d=t : timeSinceLastRun &gt; minLag</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>d=f : !(timeSinceLastRun &gt; minLag)</td>
<td>8</td>
<td>10</td>
</tr>
</tbody>
</table>

These values need to be placed into calls to turnHeaterOn() to satisfy the 6 tests for CACC.
dTemp = 69 (period = MORNING, daytype = WEEKDAY)

1. T t f t
   thermo.setCurrentTemp (63);
   thermo.setThresholdDiff (5);
   thermo.setOverride (true);
   thermo.setOverTemp (67); // c is false
   thermo.setMinLag (10);
   thermo.setTimeSinceLastRun (12);

2. F t f t
   thermo.setCurrentTemp (66); // a is false
   thermo.setThresholdDiff (5);
   thermo.setOverride (true);
   thermo.setOverTemp (67); // c is false
   thermo.setMinLag (10);
   thermo.setTimeSinceLastRun (12);
CACC Tests 3 & 4

dTemp = 69 (period = MORNING, daytype = WEEKDAY)

3. f T t t
   thermo.setCurrentTemp (66); // a is false
   thermo.setThresholdDiff (5);
   thermo.setOverride (true);
   thermo.setOverTemp (72); // to make c true
   thermo.setMinLag (10);
   thermo.setTimeSinceLastRun (12);

4. F f T t
   thermo.setCurrentTemp (66); // a is false
   thermo.setThresholdDiff (5);
   thermo.setOverride (false); // b is false
   thermo.setOverTemp (72);
   thermo.setMinLag (10);
   thermo.setTimeSinceLastRun (12);
dTemp = 69 (period = MORNING, daytype = WEEKDAY)

5. t t t T
   thermo.setCurrentTemp (63);
   thermo.setThresholdDiff (5);
   thermo.setOverride (true);
   thermo.setOverTemp (72);
   thermo.setMinLag (10);
   thermo.setTimeSinceLastRun (12);

6. t t t F
   thermo.setCurrentTemp (63);
   thermo.setThresholdDiff (5);
   thermo.setOverride (true);
   thermo.setOverTemp (72);
   thermo.setMinLag (10);
   thermo.setTimeSinceLastRun (8); // d is false
Program Transformation Issues

```c
if ((a && b) || c)
{
    S1;
}
else
{
    S2;
}
```

```c
if (a) {
    if (b)
        S1;
    else {
        if (c)
            S1;
        else
            S2;
    }
} 
else {
    if (c)
        S1;
    else
        S2;
}
```
Problems With Transformation 1

- We trade one problem for two problems:
  - Maintenance becomes harder
  - Reachability becomes harder

- Consider coverage:
  - CACC on the original requires four rows marked in the table
  - PC on the transformed version requires five different rows

- PC on the transformed version has two problems:
  1. It does not satisfy CACC on the original
  2. It is more expensive (more tests)
Program Transformation Issue 2

```c
if ((a && b) || c)
{
    S1;
}
else
{
    S2;
}
```

Transform (2) ?

```c
d = a && b;
e = d || c;
if (e)
{
    S1;
}
else
{
    S2;
}
```
Problems With Transformation 2

• We move **complexity** into computations
  – Logic criteria are not effective at testing computations

• Consider **coverage**:
  – **CACC** on the original requires four rows marked in the table
  – **PC** on the transformed version requires only two

• **PC** on the transformed version becomes equivalent to clause coverage on the original
  – Not an effective testing technique

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>(ab)c</th>
<th>CACC</th>
<th>PC_T</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Logic coverage criteria exist to help us develop better software

Circumventing the criteria is unsafe
Side Effects in Predicates

- Side effects occur when a value is changed while evaluating a predicate
  - A clause appears twice in the same predicate
  - A clause in between changes the value of the clause that appears twice

- Example:
  - Evaluation: Runtime system checks A, then B, if B is false, check A again
  - But now A has a different value!
  - How do we write a test that has two different values for the same predicate?

- No clear answers to this controllability problem

We suggest a social solution: Go ask the programmer
Summary

- **Predicates** from decision statements (if, while, for, etc.)
- Most predicates have less than **four clauses**
  - But some programs have a few predicates with many clauses
- The challenge is resolving **internal** variables
- Don’t forget **non-local** variables
- If an input variable is changed within a method, it is treated as an **internal variable** thereafter
- Avoid transformations that hide predicate structure
Next

• Syntax-based testing
• Course project