Logic Coverage for Source Code

CS 3250 Software Testing

[Ammann and Offutt, "Introduction to Software Testing," Ch. 8]

Logic Coverage for Source Code

- Aim: to identify test requirements (and then test cases for them) according to predicates and clauses found in source code
- Clauses and predicates: identified in connection with logical operators defined by the programming language under test
- Reachability predicates: need to be derived to ensure proper evaluation of each predicate (in line with program flow)

Logic Expressions from Source

- Predicates are derived from decision statements
- In programs, most predicates have <=4 clauses
  - Keep predicates simple
- When a predicate only has one clause, CoC, ACC, ICC, and CC all collapse to PC
- Applying logic criteria to program source is hard because of reachability and controllability:
  - Reachability: must get to the predicate / statement that we are applying the criteria on
  - Controllability: must find input values that indirectly assign values to the variables in the predicates
  - Internal variables: variables in the predicates that are not inputs to the program

Example: Predicates and Clauses

```
public static int number_of_days(int m, int y)
{
    if (m <= 0 || m > 12)                                   // c1, c2
        throw new IllegalArgumentException("Months must be in range 1..12");
    if (m == 2)                                             // c3
    {
        if (y % 400 == 0 || (y % 4 == 0 && y % 100 != 0))   // c4, c5, c6
            return 29;
        else
            return 28;
    }
    if (m <= 7)                                             // c7
    {
        if (m % 2 == 1)                                     // c8
            return 31;
        return 30;
    }
    if (m % 2 == 0)                                         // c9
        return 31;
    return 30;
}
```

Predicates and clauses

- p1 = c1 || c2
- p2 = c3
- p3 = c4 || (c5 && c6)
- p4 = c7
- p5 = c8
- p6 = c9

Applying PC, CC, and CoC

```
public static int number_of_days(int m, int y)
{
        c1          c2
    if (m <= 0 || m > 12)
        throw new IllegalArgumentException("Months must be in range 1..12");
    if (m == 2) c3
    {
                    c4                  c5                  c6
        if (y % 400 == 0 || (y % 4 == 0 && y % 100 != 0))
            return 29;
        else
            return 28;
    }       c7
    if (m <= 7)
    {
                c8
        if (m % 2 == 1)
            return 31;
        return 30;
    }           c9
    if (m % 2 == 0)
        return 31;
    return 30;
}
```

Predicates and clauses

- p1 = c1 || c2
- p2 = c3
- p3 = c4 || (c5 && c6)
- p4 = c7
- p5 = c8
- p6 = c9

Applying PC

Predicates and clauses

p1 = c1 || c2
p2 = c3
p3 = c4 || (c5 && c6)
p4 = c7
p5 = c8
p6 = c9

c1 = m <= 0
c2 = m > 12
c3 = m == 2
c4 = y % 400 == 0
c5 = y % 4 == 0
c6 = y % 100 != 0
c7 = m <= 7
c8 = m % 2 == 1
c9 = m % 2 == 0

tr	p
1	p1
2	¬p1
3	p2
4	¬p2
5	p3
6	¬p3
7	p4
8	¬p4
9	p5
10	¬p5
11	p6
12	¬p6

Applying CC

Predicates and clauses

p1 = c1 || c2
p2 = c3
p3 = c4 || (c5 && c6)
p4 = c7
p5 = c8
p6 = c9

c1 = m <= 0
c2 = m > 12
c3 = m == 2
c4 = y % 400 == 0
c5 = y % 4 == 0
c6 = y % 100 != 0
c7 = m <= 7
c8 = m % 2 == 1
c9 = m % 2 == 0

tr	c
1	c1
2	¬c1
3	c2
4	¬c2
5	c3
6	¬c3
7	c4
8	¬c4
9	c5
10	¬c5
11	c6
12	¬c6
13	c7
14	¬c7
15	c8
16	¬c8
17	c9
18	¬c9

Applying CoC

Predicates and clauses

p1 = c1 || c2

p2 = c3

p3 = c4 || (c5 && c6)

p4 = c7

p5 = c8

p6 = c9

tr	c1	c2	c3	c4	c5	c6	c7	c8	c9
1	T	T							
2	T	F	p1						
3	F	T							
4	F	F							
5			T	p2					
6			F						
7				T	T	T			
8				T	T	F	p3		
9				T	F	T			
10				T	F	F			
11				F	T	T			
12				F	T	F			
13				F	F	T			
14				F	F	F			
15						p4	T		
16							F		
17							p5	T	
18								F	
19								p6	T
20									F

Reachability Predicates – r(p)

```
public static int number_of_days(int m, int y)
{
p1  if (m <= 0 || m > 12)
        throw new IllegalArgumentException("Months must be in range 1..12");
p2  if (m == 2)
    {
p3      if (y % 400 == 0 || (y % 4 == 0 && y % 100 != 0))
            return 29;
        else
            return 28;
    }
p4  if (m <= 7)
    {
p5      if (m % 2 == 1)
            return 31;
        return 30;
    }
p6  if (m % 2 == 0)
        return 31;
    return 30;
}
```

r(p1) = true (always reached)

r(p2) = r(p1) && !p1 = (m > 0 && m <= 12)

r(p3) = r(p2) && p2

= (m > 0 && m <= 12) && (m == 2)

= (m == 2)

r(p4) = r(p2) && !p2

= (m > 0 && m <= 12) && (m != 2)

r(p5) = r(p4) && p4

= ((m > 0 && m <= 12) && (m != 2)) && (m <= 7)

r(p6) = r(p4) && !p4

= ((m > 0 && m <= 12) && (m != 2)) && (m > 7)

Test Cases and Infeasible TRs

```
public static int number_of_days(int m, int y)
{
p1  if (m <= 0 || m > 12)
        throw new IllegalArgumentException("Months must be in range 1..12");
p2  if (m == 2)
    {
p3      if (y % 400 == 0 || (y % 4 == 0 && y % 100 != 0))
            return 29;
        else
            return 28;
    }
p4  if (m <= 7)
    {
p5      if (m % 2 == 1)
            return 31;
        return 30;
    }
p6  if (m % 2 == 0)
        return 31;
    return 30;
}
```

r(p1) = true (always reached)

r(p2) = r(p1) && !p1 = (m > 0 && m <= 12)

r(p3) = r(p2) && p2

= (m > 0 && m <= 12) && (m == 2)

= (m == 2)

r(p4) = r(p2) && !p2

= (m > 0 && m <= 12) && (m != 2)

r(p5) = r(p4) && p4

= ((m > 0 && m <= 12) && (m != 2)) && (m <= 7)

r(p6) = r(p4) && !p4

= ((m > 0 && m <= 12) && (m != 2)) && (m > 7)

PC (Test Inputs)

Predicates and clauses

p1 = c1 || c2
p2 = c3
p3 = c4 || (c5 && c6)
p4 = c7
p5 = c8
p6 = c9

c1 = m <= 0
c2 = m > 12
c3 = m == 2
c4 = y % 400 == 0
c5 = y % 4 == 0
c6 = y % 100 != 0
c7 = m <= 7
c8 = m % 2 == 1
c9 = m % 2 == 0

r(p1) = true
r(p2) = (m > 0 && m <= 12)
r(p3) = (m == 2)
r(p4) = (m > 0 && m <= 12) && (m != 2)
r(p5) = (m > 0 && m <= 7) && (m != 2)
r(p6) = (m > 7 && m <= 12) && (m != 2)

tr	p	Test inputs	
		m	y
1	p1	0	
2	¬p1	1	
3	p2	2	
4	¬p2	1	
5	p3	2	2000
6	¬p3	2	2017
7	p4	3	
8	¬p4	8	
9	p5	3	
10	¬p5	4	
11	p6	8	
12	¬p6	9	

CC (Test Inputs)

Predicates and clauses

p1 = c1 || c2
p2 = c3
p3 = c4 || (c5 && c6)
p4 = c7
p5 = c8
p6 = c9

c1 = m <= 0
c2 = m > 12
c3 = m == 2
c4 = y % 400 == 0
c5 = y % 4 == 0
c6 = y % 100 != 0
c7 = m <= 7
c8 = m % 2 == 1
c9 = m % 2 == 0

r(p1) = true
r(p2) = (m > 0 && m <= 12)
r(p3) = (m == 2)
r(p4) = (m > 0 && m <= 12) && (m != 2)
r(p5) = (m > 0 && m <= 7) && (m != 2)
r(p6) = (m > 7 && m <= 12) && (m != 2)

With reachability

tr	c	Test inputs	
		m	y
1	c1	0	
2	¬c1	1	
3	c2	13	
4	¬c2	1	
5	c3	2	
6	¬c3	1	
7	c4	2	2000
8	¬c4	2	2017
9	c5	2	2000
10	¬c5	2	2017
11	c6	2	2017
12	¬c6	2	2000
13	c7	3	
14	¬c7	8	
15	c8	3	
16	¬c8	4	
17	c9	8	
18	¬c9	9	

CoC (Test Inputs)

Predicates and clauses

p1 = c1 || c2
p2 = c3
p3 = c4 || (c5 && c6)
p4 = c7
p5 = c8
p6 = c9

c1 = m <= 0
c2 = m > 12
c3 = m == 2
c4 = y % 400 == 0
c5 = y % 4 == 0
c6 = y % 100 != 0
c7 = m <= 7
c8 = m % 2 == 1
c9 = m % 2 == 0

r(p1) = true
r(p2) = (m > 0 && m <= 12)
r(p3) = (m == 2)
r(p4) = (m > 0 && m <= 12) && (m != 2)
r(p5) = (m > 0 && m <= 7) && (m != 2)
r(p6) = (m > 7 && m <= 12) && (m != 2)

tr	c1	c2	c3	c4	c5	c6	c7	c8	c9
1	T	T		Infeasible					
2	T	F	p1	m=0					
3	F	T		m=13					
4	F	F		m= 1					
5			T	p2	m=2				
6			F		m=1				
7	infeasible			T	T	T			
8	m=2, y=2000			T	T	F			
9	Infeasible			T	F	T			
10	infeasible			T	F	F	p3		
11	m=2, y=2016			F	T	T			
12	m=2, y=2100			F	T	F			
13	m=2, y=2017			F	F	T			
14	infeasible			F	F	F			
15			m=3	p4			T		
16			m=8				F		
17				m=3			p5	T	
18				m=4				F	
19					m=8			p6	T
20					m=9				F

Applying CACC

Predicates and clauses

p1 = c1 || c2
p2 = c3
p3 = c4 || (c5 && c6)
p4 = c7
p5 = c8
p6 = c9

c1 = m <= 0
c2 = m > 12
c3 = m == 2
c4 = y % 400 == 0
c5 = y % 4 == 0
c6 = y % 100 != 0
c7 = m <= 7
c8 = m % 2 == 1
c9 = m % 2 == 0

row	c1	c2	p1	p_c1	p_c2
1	T	T	T		
2	T		T	T	
3		T	T		T
4				T	T

Major clause	Set of possible tests
c1	(2,4)
c2	(3,4)

row	c3	p2	p_c3
1	T	T	T
2			T

Major clause	Set of possible tests
c3	(1,2)

Applying CACC

Predicates and clauses

p1 = c1 || c2
p2 = c3
p3 = c4 || (c5 && c6)
p4 = c7
p5 = c8
p6 = c9

c1 = m <= 0
c2 = m > 12
c3 = m == 2
c4 = y % 400 == 0
c5 = y % 4 == 0
c6 = y % 100 != 0
c7 = m <= 7
c8 = m % 2 == 1
c9 = m % 2 == 0

row	c4	c5	c6	p3	p_c4	p_c5	p_c6
1	T	T	T	T			
2	T	T		T	T		
3	T		T	T	T		
4	T			T	T		
5		T	T	T		T	T
6		T			T		T
7			T		T	T	
8					T		

Major clause	Set of possible tests
c4	(2,6), (2,7), (2,8), (3,6), (3,7), (3,8), (4,6), (4,7), (4,8)
c5	(5,7)
c6	(5,6)

Applying CACC

Predicates and clauses

p1 = c1 || c2
p2 = c3
p3 = c4 || (c5 && c6)
p4 = c7
p5 = c8
p6 = c9

c1 = m <= 0
c2 = m > 12
c3 = m == 2
c4 = y % 400 == 0
c5 = y % 4 == 0
c6 = y % 100 != 0
c7 = m <= 7
c8 = m % 2 == 1
c9 = m % 2 == 0

row	c7	p4	p_c7
1	T	T	T
2			T

Major clause	Set of possible tests
c7	(1,2)

row	c8	p5	p_c8
1	T	T	T
2			T

Major clause	Set of possible tests
c8	(1,2)

row	c9	p6	p_c9
1	T	T	T
2			T

Major clause	Set of possible tests
c9	(1,2)

CACC (Test Inputs)

Predicates and clauses

p1 = c1 || c2
p2 = c3
p3 = c4 || (c5 && c6)
p4 = c7
p5 = c8
p6 = c9

c1 = m <= 0
c2 = m > 12
c3 = m == 2
c4 = y % 400 == 0
c5 = y % 4 == 0
c6 = y % 100 != 0
c7 = m <= 7
c8 = m % 2 == 1
c9 = m % 2 == 0

r(p1) = true
r(p2) = (m > 0 && m <= 12)
r(p3) = (m == 2)
r(p4) = (m > 0 && m <= 12) && (m != 2)
r(p5) = (m > 0 && m <= 7) && (m != 2)
r(p6) = (m > 7 && m <= 12) && (m != 2)

row	c1	c2	p1	p_c1	p_c2
1	T	T	T		
2	T		T	T	
3		T	T		T
4				T	T

Major clause	Set of possible tests	Test inputs
c1	(2,4)	m=0, m=1
c2	(3,4)	m=13, m=1

row	c3	p2	p_c3
1	T	T	T
2			T

Major clause	Set of possible tests	Test inputs
c3	(1,2)	m=2, m=1

CACC (Test Inputs)

Predicates and clauses

p1 = c1 || c2
p2 = c3
p3 = c4 || (c5 && c6)
p4 = c7
p5 = c8
p6 = c9

c1 = m <= 0
c2 = m > 12
c3 = m == 2
c4 = y % 400 == 0
c5 = y % 4 == 0
c6 = y % 100 != 0
c7 = m <= 7
c8 = m % 2 == 1
c9 = m % 2 == 0

r(p1) = true
r(p2) = (m > 0 && m <= 12)
r(p3) = (m == 2)
r(p4) = (m > 0 && m <= 12) && (m != 2)
r(p5) = (m > 0 && m <= 7) && (m != 2)
r(p6) = (m > 7 && m <= 12) && (m != 2)

row	c4	c5	c6	p3	p_c4	p_c5	p_c6
1	T	T	T	T			
2	T	T		T	T		
3	T		T	T	T		
4	T			T	T		
5		T	T	T		T	T
6		T			T		T
7			T		T	T	
8					T		

Major clause	Set of possible tests	Test inputs
c4	(2,6), (2,7), (2,8), (3,6), (3,7), (3,8), (4,6), (4,7), (4,8)	Assume: we choose (2,6) m=2, y=2000 m=2, y=2100
c5	(5,7)	m=2, y=2016 m=2, y=2017
c6	(5,6)	m=2, y=2016 m=2, y=2100

CACC (Test Inputs)

Predicates and clauses

p1 = c1 || c2
p2 = c3
p3 = c4 || (c5 && c6)
p4 = c7
p5 = c8
p6 = c9

c1 = m <= 0
c2 = m > 12
c3 = m == 2
c4 = y % 400 == 0
c5 = y % 4 == 0
c6 = y % 100 != 0
c7 = m <= 7
c8 = m % 2 == 1
c9 = m % 2 == 0

r(p1) = true
r(p2) = (m > 0 && m <= 12)
r(p3) = (m == 2)
r(p4) = (m > 0 && m <= 12) && (m != 2)
r(p5) = (m > 0 && m <= 7) && (m != 2)
r(p6) = (m > 7 && m <= 12) && (m != 2)

row	c7	p4	p_c7
1	T	T	T
2			T

Major clause	Set of possible tests	Test inputs
c7	(1,2)	m=3, m=8

row	c8	p5	p_c8
1	T	T	T
2			T

Major clause	Set of possible tests	Test inputs
c8	(1,2)	m=3, m=4

row	c9	p6	p_c9
1	T	T	T
2			T

Major clause	Set of possible tests	Test inputs
c9	(1,2)	m=8, m=9

Note: Side Effects in Predicates

- If a predicate contains the same clause twice, and a cause in between has a side effect that can change the value of the clause that appears twice, the test values get much harder to create

p = A && (B || A)

  - Check A, then check B
  - If B is false, then A is checked again
  - Suppose B is a method call that has a side effect of changing the value of A
  - No clear answer – how to write the test to control two different values of A in the same predicate

Ammann and Offutt suggest a social solution
"Go ask the programmer"

Summary

- Predicates appear in decision statements (if, while, for, etc.)
- Most predicates have <= 3 clauses, but some program have a few predicates with many clauses
- The hard part of applying logic criteria to source is usually resolving the internal variables
  - Sometimes setting variables requires calling other methods
- Non-local variables (class, global, etc.) are also input variables if they are used
- If an input variable is changed within a method, it is treated as an internal variable
- Avoid transformations that hide predicate structure

Extra practice

```
public class TriangleType {
    /** @param s1, s2, s3:  sides of the putative triangle
      * @return enum describing type of triangle */
    public Triangle triangle (int s1, int s2, int s3)
    {
        // Reject non-positive sides
        if (s1 <= 0 || s2 <= 0 || s3 <= 0)
            return (Triangle.INVALID);

        // Check triangle inequality
        if (s1+s2 <= s3 || s2+s3 <= s1 || s1+s3 <= s2)
            return (Triangle.INVALID);

        // Identify equilateral triangles
        if ((s1 == s2) && (s2 == s3))
            return Triangle.EQUILATERAL;

        // Identify isosceles triangles
        if ((s1 == s2) || (s2 == s3) || (s1 == s3))
            return Triangle.ISOSCELES;

        return (Triangle.SCALENE);
    }
}
```

```
public enum Triangle {
    SCALENE, ISOSCELES, EQUILATERAL, INVALID
}
```

Exercise

Identify

- Reachability predicates
- TRs and test cases that satisfy PC
- TRs and test cases that satisfy CC
- Determination predicates (compute and simplify)
- TRs and test cases that satisfy CACC (or RACC)
- Infeasible requirements