Predicate Testing

- Introduction
- Basic Concepts
- Predicate Coverage
- Program-Based Predicate Testing
- Summary

Motivation

- Predicates are expressions that can be evaluated to a boolean value, i.e., true or false.
- Many decision points can be encoded as a predicate, i.e., which action should be taken under what condition?
- Predicate-based testing is about ensuring those predicates are implemented correctly.

Applications

- Program-based: Predicates can be identified from the branching points of the source code
  - e.g.: if ((a > b) || c) { … } else { … }
- Specification-based: Predicates can be identified from both formal and informal requirements as well as behavioral models such as FSM
  - "if the printer is ON and has paper then send the document for printing"
- Predicate testing is required by US FAA for safety critical avionics software in commercial aircraft

Predicate Testing

- Introduction
- Basic Concepts
- Predicate Coverage
- Program-Based Predicate Testing
- Summary

Predicate

- A predicate is an expression that evaluates to a Boolean value
- Predicates may contain:
  - Boolean variables
  - Non-boolean variables that are compared with the relational operators $\{>, <, =, \geq, \leq, \neq\}$
  - Boolean function calls
- The internal structure is created by logical operators:
  - $\neg, \wedge, \vee, \rightarrow, \oplus, \leftrightarrow$

Clause

- A clause is a predicate that does not contain any of the logical operators
- Example: $(a = b) \vee C \wedge p(x)$ has three clauses: a relational expression $(a = b)$, a boolean variable C, and a boolean function call $p(x)$

Predicate Faults

- An incorrect boolean operator is used
- An incorrect boolean variable is used
- Missing or extra boolean variables
- An incorrect relational operator is used
- Parentheses are used incorrectly

Example

- Assume that $(a < b) \vee (c > d) \wedge e$ is a correct boolean expression:
  - $(a < b) \wedge (c > d) \wedge e$
  - $(a < b) \vee (c > d) \wedge f$
  - $(a < b) \vee (c > d)$
  - $(a = b) \vee (c > d) \wedge e$
  - $(a = b) \vee (c \leq d) \wedge e$
  - $(a < b \vee c > d) \wedge e$

Predicate Testing

- Introduction
- Basic Concepts
- Predicate Coverage
- Program-Based Predicate Testing
- Summary

Predicate Coverage

- For each predicate p, TR contains two requirements: p evaluates to true, and p evaluates to false.
- Example: $((a > b) \vee C) \wedge p(x)$

	a	b	C	p(x)
1	5	4	true	true
2	5	6	false	false

Consider what happens if the predicate is written, by mistake, as $((a < b) \vee C) \wedge p(x)$

Clause Coverage

- For each clause c, TR contains two requirements: c evaluates to true, and c evaluates to false.
- Example: $((a > b) \vee C) \wedge p(x)$

	a	b	C	p(x)
1	5	4	true	true
2	5	6	false	false

Predicate vs Clause Coverage

- Does predicate coverage subsume clause coverage? Does clause coverage subsume predicate coverage?
- Example: $p = a \vee b$

	a	b	$a \vee b$
1	T	T	T
2	T	F	T
3	F	T	T
4	F	F	F

Combinatorial Coverage

- For each predicate p, TR has test requirements for the clauses in p to evaluate to each possible combination of truth values
- Example: $(a \lor b) \land c$

	a	b	c	$(a \lor b) \land c$
1	T	T	T	T
2	T	T	F	F
3	T	F	T	T
4	T	F	F	F
5	F	T	T	T
6	F	T	F	F
7	F	F	T	F
8	F	F	F	F

Active Clause

- A major clause is a clause on which we are focusing; all of the other clauses are minor clauses.
  - Each clause is treated in turn as a major clause
- Determination: Given a major clause c in predicate p, c determines p if the minor clauses have values so that changing the truth value of c changes the truth value of p.
  - Note that c and p do not have to have the same value.
- Example: $p = a \vee b$

Active Clause Coverage (ACC)

- For each predicate p and each major clause c of p, choose minor clauses so that c determines p. TR has two requirements for each c: c evaluates to true and c evaluates to false.
- Example: $p = a \vee b$

	a	b
c = a	T	f
	F	f
c = b	f	T
	f	F

General Active Clause Coverage (GACC)

- The same as ACC, and it does not require the minor clauses have the same values when the major clause evaluates to true and false.
- Does GACC subsume predicate coverage?

	a	b	a ↔ b
1	t	t	t
2	t	f	f
3	f	t	f
4	f	f	t

Correlated Active Clause Coverage (CACC)

- The same as ACC, but it requires the entire predicate to be true for one value of the major clause and false for the other.
- Example: $a \leftrightarrow b$

	a	b	$a \leftrightarrow b$
1	t	t	t
2	t	f	f
3	f	t	f
4	f	f	t

a: {1, 3}
b: {1, 2}
$a \leftrightarrow b$: {1, 2, 3}

CACC (2)

- Example: $a \wedge (b \vee c)$

	a	b	c	$a \wedge (b \vee c)$
1	T	T	T	T
2	T	T	F	T
3	T	F	T	T
5	F	T	T	F
6	F	T	F	F
7	F	F	T	F

$\{1, 2, 3\} \times \{5, 6, 7\}$

Restricted Active Clause Coverage (RACC)

- The same as ACC, but it requires the minor clauses have the same values when the major clause evaluates to true and false.
- Example: $a \wedge (b \vee c)$

	a	b	c	$a \wedge (b \vee c)$
1	T	T	T	T
5	F	T	T	F
2	T	T	F	T
6	F	T	F	F
3	T	F	T	T
7	F	F	T	F

CACC vs RACC (1)

- Consider a system with a valve that might be either open or closed, and several modes, two of which are "operational" and "standby".
- Assume the following two constraints: (1) The valve must be open in "operational" and closed in all other modes; (2) The mode cannot be both "operational" and "standby" at the same time.
- Let a = "The valve is closed", b = "The system status is operational", and c = "The system status is standby". Then, the two constraints can be formalized as (1) $\neg a \leftrightarrow b$; (2) $\neg (b \wedge c)$

CACC vs RACC (2)

- Suppose that a certain action can be taken only if the valve is closed and the system status is in Operational or Standby, i.e., $a \wedge (b \vee c)$

	a	b	c	$a \wedge (b \vee c)$	Constraints violated
1	T	T	T	T	1 & 2
2	T	T	F	T	1
3	T	F	T	T	
4	~~T~~	~~F~~	~~F~~	~~F~~	
5	F	T	T	F	2
6	F	T	F	F	
7	F	F	T	F	1
8	F	F	F	F	1

Making Active Clauses

- Let p be a predicate and c a clause. Let $p_{c=true}$ (or $p_{c=false}$) be the predicate obtained by replacing every occurrence of c with $true$ (or $false$)
- The following expression describes the exact conditions under which the value of c determines the value of p:

$$\boxed{p_c = p_{c=true} \oplus p_{c=false}}$$

Example (1)

- Consider $p = a \vee b$ and $p = a \wedge b$.

$p = a \vee b$

$p_{a=true} = T \vee b = T$

$p_{a=false} = F \vee b = b$

$p_a = p_{a=true} \oplus p_{a=false}$

$= T \oplus b$

$= \neg b$

$p = a \wedge b$

$p_{b=true} = a \wedge T = a$

$p_{b=false} = a \wedge F = F$

$p_b = p_{b=true} \oplus p_{b=false}$

$= a \oplus F$

$= a$

Example (2)

- Consider $p = a \wedge b \vee a \wedge \neg b = a \wedge (b \vee \neg b) = a$

$P_{a=true} = T \wedge b \vee T \wedge \neg b = b \vee \neg b = T$

$P_{a=false} = F \wedge b \vee F \wedge \neg b = F \vee F = F$

$P_a = P_{a=true} \oplus P_{a=false}$

$= T \oplus F$

$= T$

$P_{b=true} = a \wedge T \vee a \wedge F = a \vee F = a$

$P_{b=false} = a \wedge F \vee a \wedge T = F \vee a = a$

$P_b = P_{b=true} \oplus P_{b=false}$

$= a \oplus a$

$= F$

Example (3)

- Consider $p = a \wedge (b \vee c)$.

$$p_{c=true} = a \wedge (b \vee T)$$
$$= a \wedge T = a$$

$$p_{c=false} = a \wedge (b \vee F)$$
$$= a \wedge b$$

$$x \oplus y \equiv (x \wedge \neg y) \vee (\neg x \wedge y)$$

$$p_c = p_{c=true} \oplus p_{c=false}$$
$$= \underbrace{\boxed{a}}_{x} \oplus \underbrace{\boxed{(a \wedge b)}}_{y}$$
$$= (a \wedge \neg(a \wedge b)) \vee (\neg a \wedge (a \wedge b))$$
$$= (a \wedge (\neg a \vee \neg b)) \vee (\underline{\neg a \wedge a} \wedge b)$$
$$= (a \wedge \neg a) \vee (a \wedge \neg b) \vee (F \wedge b)$$
$$= F \vee (a \wedge \neg b) \vee F$$
$$= a \wedge \neg b$$

Finding Satisfying Values

- How to choose values that satisfy a given coverage goal?

Example (1)

- Consider $p = (a \lor b) \land c$:

a	x < y
b	done
c	List.contains(str)

How to choose values to satisfy predicate coverage?

Example (2)

	a	b	c	p
1	t	t	t	t
2	t	t	f	f
3	t	f	t	t
4	t	f	f	f
5	f	t	t	t
6	f	t	f	f
7	f	f	t	f
8	f	f	f	f

$\{1, 3, 5\} \times \{2, 4, 6, 7, 8\}$

Example (3)

- Suppose we choose {1, 2}.

a	b	c
x = 3 y = 5	done = true	List = [“Rat”, “cat”, “dog”] str = “cat”
x = 0, y = 7	done = true	List = [“Red”, “White”] str = “Blue”

Example (4)

- What about clause coverage?

	a	b	c	p
1	t	t	t	t
2	t	t	f	f
3	t	f	t	t
4	t	f	f	f
5	f	t	t	t
6	f	t	f	f
7	f	f	t	f
8	f	f	f	f

{{1, 8}, {2, 7}, {3, 6}, {4, 5}}

Example (5)

- What about GACC, CACC, and RACC?

$P_a = \neg b \wedge c \quad P_b = \neg a \wedge c \quad P_c = a \vee b$

	a	b	c	p	p_a	p_b	p_c
1	t	t	t	t			t
2	t	t	f	f			t
3	t	f	t	t	t		t
4	t	f	f	f			t
5	f	t	t	t		t	t
6	f	t	f	f			t
7	f	f	t	f	t	t	
8	f	f	f	f			

Consider clause a:
GACC: {3, 7}
CACC: {3, 7}
RACC: {3, 7}

Consider clause b:
GACC: {5, 7}
CACC: {5, 7}
RACC: {5, 7}

Consider clause c:
GACC: {1, 3, 5} × {2, 4, 6}
CACC: {1, 3, 5} × {2, 4, 6}
RACC: {{1, 2}, {3, 4}, {5, 6}}

Predicate Testing

- Introduction
- Basic Concepts
- Predicate Coverage
- Program-Based Predicate Testing
- Summary

Reachability & Controllability

- Reachability: A test must be able to reach the predicate being tested.
- Controllability: Internal variables must be rewritten in terms of external input variables
- In general, finding values to satisfy reachability and controllability is undecidable.

TriType

```
// Jeff Offut – Java version Feb 2003
// The old standby: classify triangles
import java.io.*;

class trityp
{
   private static String[] triTypes = { "",    // Ignore 0.
          "scalene", "isosceles", "equilateral", "not a valid
             triangle"};
   private static String instructions = "This is the ancient
  TriTyp program.\nEnter three integers that represent the lengths
   of the sides of a triangle.\nThe triangle will be categorized as
   either scalene, isosceles, equilateral\nor invalid.\n";

public static void main (String[] argv)
{  // Driver program for trityp
   int A, B, C;
   int T;
   System.out.println (instructions);
   System.out.println ("Enter side 1: ");
   A = getN();
   System.out.println ("Enter side 2: ");
   B = getN();
   System.out.println ("Enter side 3: ");
   C = getN();
   T = Triang (A, B, C);

   System.out.println ("Result is: " + triTypes [T]);
}

// ====================================

// The main triangle classification method
private static int Triang (int Side1, int Side2, int Side3)
{
   int tri_out;

   // tri_out is output from the routine:
   //    Triang = 1 if triangle is scalene
   //    Triang = 2 if triangle is isosceles
   //    Triang = 3 if triangle is equilateral
   //    Triang = 4 if not a triangle


   // After a quick confirmation that it's a legal
   // triangle, detect any sides of equal length
   if (Side1 <= 0 || Side2 <= 0 || Side3 <= 0)
   {
     tri_out = 4;
     return (tri_out);
   }
   tri_out = 0;
   if (Side1 == Side2)
     tri_out = tri_out + 1;
   if (Side1 == Side3)
     tri_out = tri_out + 2;
   if (Side2 == Side3)
     tri_out = tri_out + 3;
   if (tri_out == 0)
   {  // Confirm it's a legal triangle before declaring
     // it to be scalene

     if (Side1+Side2 <= Side3 || Side2+Side3 <= Side1 ||
        Side1+Side3 <= Side2)
        tri_out = 4;
     else
        tri_out = 1;
     return (tri_out);
   }
   /* Confirm it's a legal triangle before declaring  */
   /* it to be isosceles or equilateral  */

   if (tri_out > 3)
     tri_out = 3;
   else if (tri_out == 1 && Side1+Side2 > Side3)
    tri_out = 2;
   else if (tri_out == 2 && Side1+Side3 > Side2)
    tri_out = 2;
   else if (tri_out == 3 && Side2+Side3 > Side1)
    tri_out = 2;
   else
    tri_out = 4;
   return (tri_out);
} // end Triang
```

Predicates

```
42: (Side1 <= 0 || Side2 <= 0 || Side3 <= 0)
49: (Side1 == Side2)
51: (Side1 == Side3)
53: (Side2 == Side3)
55: (triOut == 0)
59: (Side1+Side2 <= Side3 || Side2+Side3 <= Side1 ||
      Side1+Side3 <= Side2)
70: (triOut > 3)
72: (triOut == 1 && Side1+Side2 > Side3)
74: (triOut == 2 && Side1+Side3 > Side2)
76: (triOut == 3 && Side2+Side3 > Side1)
```

Reachability

42: True
49: P1 = s1>0 && s2>0 && s3>0
51: P1
53: P1
55: P1
59: P1 && triOut = 0
62: P1 && triOut = 0
&& (s1+s2 > s3) && (s2+s3 > s1) && (s1+s3 > s2)
70: P1 && triOut != 0
72: P1 && triOut != 0 && triOut <= 3
74: P1 && triOut != 0 && triOut <= 3 && (triOut !=1 || s1+s2<=s3)
76: P1 && triOut != 0 && triOut <= 3 && (triOut !=1 || s1+s2<=s3)
&& (triOut !=2 || s1+s3<=s2)
78: P1 && triOut != 0 && triOut <= 3 && (triOut !=1 || s1+s2<=s3)
&& (triOut !=2 || s1+s3 <= s2) && (triOut !=3 || s2+s3 <= s1)

Solving Internal Vars

At line 55, triOut has a value in the range (0 .. 6)

triOut = 0	**s1!=s2**	**&&**	**s1!=s3**	**&&**	**s2!=s3**
1	**s1=s2**	**&&**	**s1!=s3**	**&&**	**s2!=s3**
2	**s1!=s2**	**&&**	**s1=s3**	**&&**	**s2!=s3**
3	**s1!=s2**	**&&**	**s1!=s3**	**&&**	**s2=s3**
4	**s1=s2**	**&&**	**s1!=s3**	**&&**	**s2=s3**
5	**s1!=s2**	**&&**	**s1=s3**	**&&**	**s2=s3**
6	**s1=s2**	**&&**	**s1=s3**	**&&**	**s2=s3**

Reduced Reachability

42: True
49: P1 = s1>0 && s2>0 && s3>0
51: P1
53: P1
55: P1
59: P1 && s1 != s2 && s2 != s3 && s2 != s3 (triOut = 0)
62: P1 && s1 != s2 && s2 != s3 && s2 != s3 (triOut = 0)
&& (s1+s2 > s3) && (s2+s3 > s1) && (s1+s3 > s2)
70: P1 && P2 = (s1=s2 || s1=s3 || s2=s3) (triOut != 0)
72: P1 && P2 && P3 = (s1!=s2 || s1!=s3 || s2!=s3) (triOut <= 3)
74: P1 && P2 && P3 && (s1 != s2 || s1+s2<=s3)
76: P1 && P2 && P3 && (s1 != s2 || s1+s2<=s3)
&& (s1 != s3 || s1+s3<=s2)
78: P1 && P2 && P3 && (s1 != s2 || s1+s2<=s3)
&& (s1 != s3 || s1+s3<=s2) && (s2 != s3 || s2+s3<=s1)

Predicate Coverage

	Predicate	True				False			
		A	B	C	EO	A	B	C	EO
P42:	(Side1 <= 0 || Side2 <= 0 || Side3 <= 0)	0	0	0	4	1	1	1	3
P49:	(Side1 == Side2)	1	1	1	3	1	2	2	3
P51:	(Side1 == Side3)	1	1	1	3	1	2	2	2
P53:	(Side2 == Side3)	1	1	1	3	2	1	2	2
P55:	(triOut == 0)	1	2	3	4	1	1	1	3
P59:	(Side1+Side2 <= Side3 || Side2+Side3 <= Side1 || Side1+Side3 <= Side2)	1	2	3	4	2	3	4	1
P70:	(triOut > 3)	1	1	1	3	2	2	3	2
P72:	(triOut == 1 && Side1+Side2 > Side3)	2	2	3	2	2	2	4	4
P74:	(triOut == 2 && Side1+Side3 > Side2)	2	3	2	2	2	4	2	4
P76:	(triOut == 3 && Side2+Side3 > Side1)	3	2	2	2	4	2	2	4

Clause Coverage

		True				False			
Predicate		A	B	C	EO	A	B	C	EO
P42:	(Side1 <= 0)	0	1	1	4	1	1	1	3
	(Side2 <= 0)	1	0	1	4	1	1	1	3
	(Side3 <= 0)	1	1	0	4	1	1	1	3
P59:	(Side1+Side2 <= Side3)	2	3	6	4	2	3	4	1
	(Side2+Side3 <= Side1)	6	2	3	4	2	3	4	1
	(Side1+Side3 <= Side2)	2	6	3	4	2	3	4	1
P72:	(triOut == 1)	2	2	3	2	2	3	2	2
	(Side1+Side2 > Side3)	2	2	3	2	2	2	5	4
P74:	(triOut == 2)	2	3	2	2	2	4	2	4
	(Side1+Side3 > Side2)	2	3	2	2	2	5	2	4
P76:	(triOut == 3)	3	2	2	2	1	2	1	4
	(Side2+Side3 > Side1)	3	2	2	2	5	2	2	4

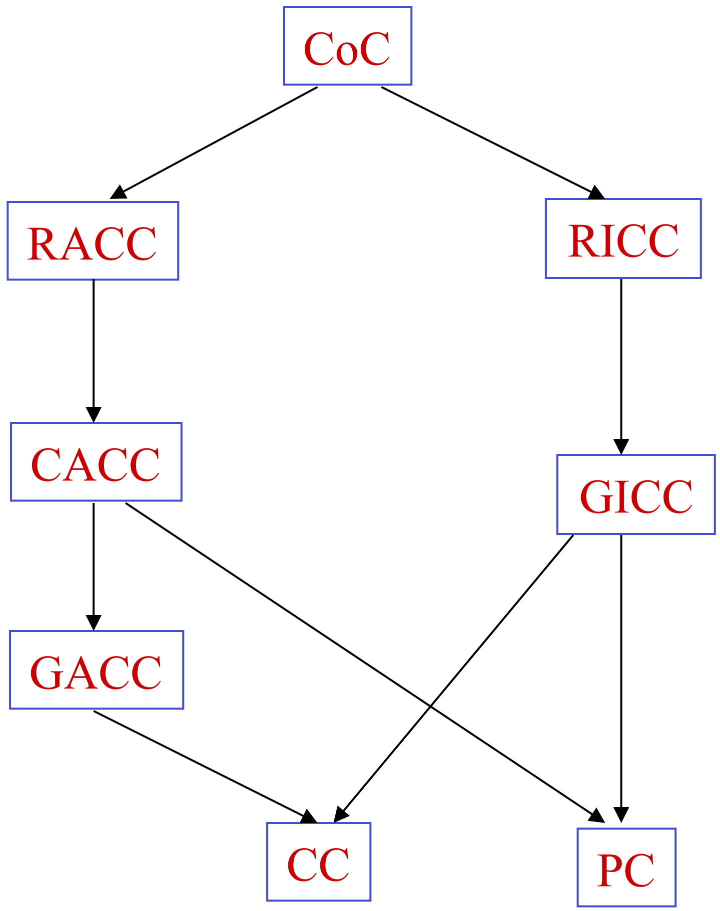
CACC Coverage

	Predicate	Clauses			A	B	C	EO
P42:	(Side1 <= 0)	T	f	f	0	1	1	4
	(Side2 <= 0)	F	f	f	1	1	1	3
	(Side3 <= 0)	f	T	f	1	0	1	4
		f	f	T	1	1	0	4
P59:	(Side1+Side2 <= Side3)	T	f	f	2	3	6	4
	(Side2+Side3 <= Side1)	F	f	f	2	3	4	1
	(Side1+Side3 <= Side2)	f	T	f	6	2	3	4
		f	f	T	2	6	3	4
P72:	(triOut == 1)	T	t	-	2	2	3	2
	(Side1+Side2 > Side3)	F	t	-	2	3	3	2
		t	F	-	2	2	5	4
P74:	(triOut == 2)	T	t	-	2	3	2	2
	(Side1+Side3 > Side2)	F	t	-	2	3	3	2
		t	F	-	2	5	2	4
P76:	(triOut == 3)	T	t	-	3	2	2	2
	(Side2+Side3 > Side1)	F	t	-	3	6	3	4
		t	F	-	5	2	2	4

Predicate Testing

- Introduction
- Basic Concepts
- Predicate Coverage
- Program-Based Predicate Testing
- Summary

Subsumption



Recap

- Predicate testing is about ensuring that each decision point is implemented correctly.
- If we flip the value of an active clause, we will change the value of the entire predicate.
- Different active clause criteria are defined to clarify the requirements on the values of the minor clauses.
- Reachability and controllability are two practical challenges that have to be met when we apply predicate testing to programs.