SWE 637 Software Testing
Chapter 8.3

Applying Logic Coverage
In-class exercise

Dr. Brittany Johnson-Matthews
(Dr. B for short)

https://go.gmu.edu/SWE637

Adapted from slides by Jeff Offutt and Bob Kurtz

Exercise 8.3 #12

```
public final class GoodFastCheap {
  boolean good = false;
  boolean fast = false;
  boolean cheap = false;

  public void makeGood () {
    good = true;
    if (fast && cheap) cheap = false;
  }

  public void makeFast () {
    fast = true;
    if (good && cheap) good = false;
  }

  public void makeCheap () {
    cheap = true;
    if (good && fast) fast = false;
  }

  public void makeBad () { good = false };
  public void makeSlow () { fast = false };
  public void makeExpensive () { cheap = false; }

  public boolean isSatisfactory () {
    if ((good && fast) || (good && cheap) || (fast && cheap))
      return true;
    return false;
  }

  public boolean isSatisfactoryRefactored () {
    if (good && fast) return true;
    if (good && cheap) return true;
    if (fast && cheap) return true;
    return false;
  }
}
```

Analyze and test this predicate

Good, fast, and cheap: pick any two out of three!

isSatisfactory Predicate

```
public boolean isSatisfactory () {
  if ((good && fast) || (good && cheap) || (fast && cheap))
    return true;
  return false;
}
```

Simplify the predicate:

P =

isSatisfactory Predicate

```
public boolean isSatisfactory () {
  if ((good && fast) || (good && cheap) || (fast && cheap))
    return true;
  return false;
}
```

Simplify the predicate:

$P = g \wedge f \vee g \wedge c \vee f \wedge c$

RACC Tests for isSatisfactory

#	g	f	c	p	p_g	p_f	p_c
1	T	T	T				
2	T	T	F				
3	T	F	T				
4	T	F	F				
5	F	T	T				
6	F	T	F				
7	F	F	T				
8	F	F	F				

$$p = g \wedge f \vee g \wedge c \vee f \wedge c$$

Complete the truth table for p

RACC Tests for isSatisfactory

#	g	f	c	p	p_g	p_f	p_c
1	T	T	T	T			
2	T	T	F	T			
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			

$$p = g \wedge f \vee g \wedge c \vee f \wedge c$$

Complete the truth table for p

RACC Tests for isSatisfactory

#	g	f	c	p	p_g	p_f	p_c
1	T	T	T	T			
2	T	T	F	T			
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			

$$p = g \wedge f \vee g \wedge c \vee f \wedge c$$

Determine conditions under which ***g*** determines ***p***

$P_g = P_{g=true} \oplus P_{g=false}$

=

RACC Tests for isSatisfactory

$$p = g \wedge f \vee g \wedge c \vee f \wedge c$$

#	g	f	c	p	p_g	p_f	p_c
1	T	T	T	T			
2	T	T	F	T	1		
3	T	F	T	T	2		
4	T	F	F	F			
5	F	T	T	T			
6	F	T	F	F	1		
7	F	F	T	F	2		
8	F	F	F	F			

Determine conditions under which ***g*** determines ***p***

$P_g = P_{g=true} \oplus P_{g=false}$

$= T \wedge f \vee T \wedge c \vee f \wedge c \oplus F \wedge f \vee F \wedge c \vee f \wedge c$

$= f \vee c \vee f \wedge c \oplus f \wedge c$

$= f \vee c \oplus f \wedge c$

$= f \wedge \neg c \vee \neg f \wedge c$

RACC Tests for isSatisfactory

#	g	f	c	p	p_g	p_f	p_c
1	T	T	T	T			
2	T	T	F	T	1		
3	T	F	T	T	2		
4	T	F	F	F			
5	F	T	T	T			
6	F	T	F	F	1		
7	F	F	T	F	2		
8	F	F	F	F			

$$p = g \wedge f \vee g \wedge c \vee f \wedge c$$

Determine conditions under which *f* determines *p*

$P_f = P_{f=true} \oplus P_{f=false}$

=

RACC Tests for isSatisfactory

$$p = g \wedge f \vee g \wedge c \vee f \wedge c$$

#	g	f	c	p	p_g	p_f	p_c
1	T	T	T	T			
2	T	T	F	T	1	3	
3	T	F	T	T	2		
4	T	F	F	F		3	
5	F	T	T	T		4	
6	F	T	F	F	1		
7	F	F	T	F	2	4	
8	F	F	F	F			

Determine conditions under which
f determines *p*

$P_f = P_{f=true} \oplus P_{f=false}$
$= g \wedge T \vee g \wedge c \vee T \wedge c \oplus g \wedge F \vee g \wedge c \vee F \wedge c$
$= g \vee g \wedge c \vee c \oplus g \wedge c$
$= g \vee c \oplus g \wedge c$
$= g \wedge \neg c \vee \neg g \wedge c$

RACC Tests for isSatisfactory

#	g	f	c	p	p_g	p_f	p_c
1	T	T	T	T			
2	T	T	F	T	1	3	
3	T	F	T	T	2		
4	T	F	F	F		3	
5	F	T	T	T		4	
6	F	T	F	F	1		
7	F	F	T	F	2	4	
8	F	F	F	F			

$$p = g \wedge f \vee g \wedge c \vee f \wedge c$$

Determine conditions under which ***c*** determines ***p***

$$P_c = P_{c=true} \oplus P_{c=false}$$

$$=$$

RACC Tests for isSatisfactory

$$p = g \wedge f \vee g \wedge c \vee f \wedge c$$

#	g	f	c	p	p_g	p_f	p_c
1	T	T	T	T			
2	T	T	F	T	1	3	
3	T	F	T	T	2		5
4	T	F	F	F		3	5
5	F	T	T	T		4	6
6	F	T	F	F	1		6
7	F	F	T	F	2	4	
8	F	F	F	F			

Determine conditions under which ***c*** determines ***p***

$$P_c = P_{c=true} \oplus P_{c=false}$$
$$= g \wedge f \vee g \wedge T \vee f \wedge T \oplus g \wedge f \vee g \wedge F \vee f \wedge F$$
$$= g \wedge f \vee g \vee f \oplus g \wedge f$$
$$= g \vee f \oplus g \wedge f$$
$$= g \wedge \neg f \vee \neg g \wedge f$$

RACC Tests for isSatisfactory

Develop a set of JUnit tests for method isSatisfactory() that satisfies the RACC criterion

DEFINITION

Restricted Active Clause Coverage (RACC) – For each ***p*** in ***P*** and each major clause c_i in C_p, choose minor clauses c_j (j!=i) such that c_i determines ***p***. ***TR*** has two requirements for c_i: c_i evaluates to true and c_i evaluates to false. The values chosen for minor clauses c_j must be the same when c_i is true as when c_i is false.

RACC Tests for isSatisfactory

#	g	f	c	p	p_g	p_f	p_c
1	T	T	T	T			
2	T	T	F	T	1	3	
3	T	F	T	T	2		5
4	T	F	F	F		3	5
5	F	T	T	T		4	6
6	F	T	F	F	1		6
7	F	F	T	F	2	4	
8	F	F	F	F			

Determine RACC tests for ***g***, ***f***, and ***c***

For g: (2,6) or (3,7)

RACC Tests for isSatisfactory

#	g	f	c	p	p_g	p_f	p_c
1	T	T	T	T			
2	T	T	F	T	1	3	
3	T	F	T	T	2		5
4	T	F	F	F		3	5
5	F	T	T	T		4	6
6	F	T	F	F	1		6
7	F	F	T	F	2	4	
8	F	F	F	F			

Determine RACC tests for ***g***, ***f***, and ***c***

For g: (2,6) or (3,7)

For f:

RACC Tests for isSatisfactory

#	g	f	c	p	p_g	p_f	p_c
1	T	T	T	T			
2	T	T	F	T	1	3	
3	T	F	T	T	2		5
4	T	F	F	F		3	5
5	F	T	T	T		4	6
6	F	T	F	F	1		6
7	F	F	T	F	2	4	
8	F	F	F	F			

Determine RACC tests for ***g***, ***f***, and ***c***

For g: (2,6) or (3,7)

For f: (2,4) or (5,7)

For c:

RACC Tests for isSatisfactory

#	g	f	c	p	p_g	p_f	p_c
1	T	T	T	T			
2	T	T	F	T	1	3	
3	T	F	T	T	2		5
4	T	F	F	F		3	5
5	F	T	T	T		4	6
6	F	T	F	F	1		6
7	F	F	T	F	2	4	
8	F	F	F	F			

There are various minimal combinations of tests, but we'll arbitrarily choose { 2, 3, 4, 6 }

Determine RACC tests for ***g***, ***f***, and ***c***

For g: (2,6) or (3,7)

For f: (2,4) or (5,7)

For c: (3,4) or (5,6)

Minimal combinations:

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

RACC Tests for isSatisfactory

#	g	f	c	p	p_g	p_f	p_c
1	T	T	T	T			
2	T	T	F	T	1	3	
3	T	F	T	T	2		5
4	T	F	F	F		3	5
5	F	T	T	T		4	6
6	F	T	F	F	1		6
7	F	F	T	F	2	4	
8	F	F	F	F			

RACC tests: 2,3,4,6

{ TTF, TFT, TFF, FTF }

```
GoodFastCheap gfc;

@Before public void setUp() {
   gfc = new GoodFastCheap();    // 8:  F F F
}

@Test public void test2() throws Exception {
  gfc.makeGood();        // 4: T F F
  gfc.makeFast();        // 2: T T F
  assertTrue(gfc.isSatisfactory());
}

@Test public void test3() throws Exception {
  gfc.makeGood();        // 4: T F F
  gfc.makeCheap();       // 3: T F T
  assertTrue(gfc.isSatisfactory());
}

@Test public void test4() throws Exception {
  gfc.makeGood();        // 4: T F F
  assertFalse(gfc.isSatisfactory());
}

@Test public void test6() throws Exception {
  gfc.makeFast();        // 6: F T F
  assertFalse(gfc.isSatisfactory());
}
```

END OF EXERCISE

Exercise 8.3 #12

```
public final class GoodFastCheap {
  boolean good = false;
  boolean fast = false;
  boolean cheap = false;

  public void makeGood () {
    good = true;
    if (fast && cheap) cheap = false;
  }

  public void makeFast () {
    fast = true;
    if (good && cheap) good = false;
  }

  public void makeCheap () {
    cheap = true;
    if (good && fast) fast = false;
  }

  public void makeBad () { good = false };
  public void makeSlow () { fast = false };
  public void makeExpensive () { cheap = false; }

  public boolean isSatisfactory () {
    if ((good && fast) || (good && cheap) || (fast && cheap))
      return true;
    return false;
  }

  public boolean isSatisfactoryRefactored () {
    if (good && fast) return true;
    if (good && cheap) return true;
    if (fast && cheap) return true;
    return false;
  }
}
```

Analyze and test this predicate

Good, fast, and cheap: pick any two out of three!

IsSatisfactoryRefactored Predicate

```
public boolean isSatisfactory () {
  if ((good && fast) || (good && cheap) || (fast && cheap))
    return true;
  return false;
}
```

Simplify the predicates:

P_1 =

P_2 =

P_3 =

IsSatisfactoryRefactored Predicate

```
public boolean isSatisfactory () {
  if ((good && fast) || (good && cheap) || (fast && cheap))
    return true;
  return false;
}
```

Simplify the predicates:

$P_1 = g \wedge f$

$P_2 = g \wedge c$

$P_3 = f \wedge c$

RACC for IsSatisfactoryRefactored

Compute ***p*** and determine when ***g*** and ***f*** determine p_1

p_1 =					
#	g	f	P	p_g	p_f
1	T	T	~~T~~	✓	✓
2	T	F	~~F~~		✓
3	F	T	~~F~~	✓	
4	F	F	~~F~~		

p_2 =					
#	g	c	p	p_g	p_c
5	T	T			
6	T	F			
7	F	T			
8	F	F			

p_3 =					
#	f	c	p	p_f	p_c
9	T	T			
10	T	F			
11	F	T			
12	F	F			

RACC for IsSatisfactoryRefactored

Compute RACC test pairs for ***g*** and ***f***

TRs for ***g***:

TRs for ***f***:

p_1 =					
#	g	f	P	p_g	p_f
1	T	T	~~T~~	✓	✓
2	T	F	~~F~~		✓
3	F	T	~~F~~	✓	
4	F	F	~~F~~		

p_2 =					
#	g	c	p	p_g	p_c
5	T	T			
6	T	F			
7	F	T			
8	F	F			

p_3 =					
#	f	c	p	p_f	p_c
9	T	T			
10	T	F			
11	F	T			
12	F	F			

RACC for IsSatisfactoryRefactored

Compute RACC test pairs for ***g*** and ***f***

TRs for ***g***: (1,3)
TRs for ***f***: (1,2)

p_1 =					
#	g	f	P	p_g	p_f
1	T	T	~~T~~	✓	✓
2	T	F	~~F~~		✓
3	F	T	~~F~~	✓	
4	F	F	~~F~~		

p_2 =					
#	g	c	p	p_g	p_c
5	T	T			
6	T	F			
7	F	T			
8	F	F			

p_3 =					
#	f	c	p	p_f	p_c
9	T	T			
10	T	F			
11	F	T			
12	F	F			

RACC for IsSatisfactoryRefactored

Compute ***p*** and determine when ***g*** and ***c*** determine p_2

TRs for ***g***: (1,3)
TRs for ***f***: (1,2)

p_1 =					
#	g	f	P	p_g	p_f
1	T	T	~~T~~	✓	✓
2	T	F	~~F~~		✓
3	F	T	~~F~~	✓	
4	F	F	~~F~~		

p_2 =					
#	g	c	p	p_g	p_c
5	T	T	~~T~~	✓	✓
6	T	F	~~F~~		✓
7	F	T	~~F~~	✓	
8	F	F	~~F~~		

p_3 =					
#	f	c	p	p_f	p_c
9	T	T			
10	T	F			
11	F	T			
12	F	F			

RACC for IsSatisfactoryRefactored

Compute ***p*** and determine when ***g*** and ***c*** determine p_2

TRs for ***g***: (1,3)
TRs for ***f***: (1,2)

Compute RACC test pairs for ***g*** and ***c***

TRs for ***g***:

TRs for ***c***:

p_1 =					
#	g	f	P	p_g	p_f
1	T	T	~~T~~	✓	✓
2	T	F	~~F~~		✓
3	F	T	~~F~~	✓	
4	F	F	~~F~~		

#	g		p	p_g	p_c
5	T		~~T~~	✓	✓
6	T		~~F~~		✓
7	F		~~F~~	✓	
8	F		~~F~~		

p_3 =					
#	f	c	p	p_f	p_c
9	T	T			
10	T	F			
11	F	T			
12	F	F			

RACC for IsSatisfactoryRefactored

Compute ***p*** and determine when ***g*** and ***c*** determine p_2

TRs for ***g***: (1,3)
TRs for ***f***: (1,2)

Compute RACC test pairs for ***g*** and ***c***
TRs for ***g***: (5,7)
TRs for ***c***: (5,6)

p_1 =					
#	g	f	P	p_g	p_f
1	T	T	T	✓	✓
2	T	F	F		✓
3	F	T	F	✓	
4	F	F	F		

#	g		p	p_g	p_c
5	T		T	✓	✓
6	T		F		✓
7	F		F	✓	
8	F		F		

p_3 =					
#	f	c	p	p_f	p_c
9	T	T			
10	T	F			
11	F	T			
12	F	F			

RACC for IsSatisfactoryRefactored

Compute ***p*** and determine when ***g*** and ***f*** determine p_3

TRs for ***g***: (1,3)
TRs for ***f***: (1,2)

TRs for ***g***: (5,7)
TRs for ***c***: (5,6)

p_1 =

#	g	f	P	p_g	p_f
1	T	T	T	✓	✓
2	T	F	F		✓
3	F	T	F	✓	
4	F	F	F		

#	g		p	p_g	p_c
5	T		T	✓	✓
6	T		F		✓
7	F		F	✓	
8	F		F		

p_3 =

#	f	c	p	p_f	p_c
9	T	T	T	✓	✓
10	T	F	F		✓
11	F	T	F	✓	
12	F	F	F		

RACC for IsSatisfactoryRefactored

Compute ***p*** and determine when ***g*** and ***f*** determine p_3

TRs for ***g***: (1,3)
TRs for ***f***: (1,2)

p_1 =					
#	g	f	P	p_g	p_f
1	T	T	~~T~~	✓	✓
2	T	F	~~F~~		✓
3	F	T	~~F~~	✓	
4	F	F	~~F~~		

TRs for ***g***: (5,7)
TRs for ***c***: (5,6)

#	g		p	p_g	p_c
5	T		~~T~~	✓	✓
6	T		~~F~~		✓
7	F		~~F~~	✓	
8	F		~~F~~		

Compute RACC test pairs for ***f*** and ***c***
TRs for ***f***:
TRs for ***c***:

p_3 =					
#	f	c	p	p_f	p_c
9	T	T	~~T~~	✓	✓
10	T	F	~~F~~		✓
11	F	T	~~F~~	✓	
12	F	F	~~F~~		

RACC for IsSatisfactoryRefactored

Compute ***p*** and determine when ***g*** and ***f*** determine p_3

TRs for ***g***: (1,3)
TRs for ***f***: (1,2)

p_1 =					
#	g	f	P	p_g	p_f
1	T	T	T	✓	✓
2	T	F	F		✓
3	F	T	F	✓	
4	F	F	F		

TRs for ***g***: (5,7)
TRs for ***c***: (5,6)

#	g		p	p_g	p_c
5	T		T	✓	✓
6	T		F		✓
7	F		F	✓	
8	F		F		

Compute RACC test pairs for ***f*** and ***c***
TRs for ***f***: (9,11)
TRs for ***c***: (9,10)

p_3 =					
#	f	c	p	p_f	p_c
9	T	T	T	✓	✓
10	T	F	F		✓
11	F	T	F	✓	
12	F	F	F		

RACC for IsSatisfactoryRefactored

TRs for **g**: (1,3) and (5,7)
TRs for **f**: (1,2) and (9,11)
TRs for **c**: (5,6) and (9,10)

For f∧c:

Note that to reach g∧c, g∧f must be false. To reach f∧c, g∧c and g∧f must be false. That suggests we should work from the bottom up!

p_1 =					
#	g	f	P	p_g	p_f
1	T	T	~~T~~	✓	✓
2	T	F	~~F~~		✓
3	F	T	~~F~~	✓	
4	F	F	~~F~~		

#	g		p	p_g	p_c
5	T		~~T~~	✓	✓
6	T		~~F~~		✓
7	F		~~F~~	✓	
8	F		~~F~~		

p_3 =					
#	f	c	p	p_f	p_c
9	T	T	~~T~~	✓	✓
10	T	F	~~F~~		✓
11	F	T	~~F~~	✓	
12	F	F	~~F~~		

RACC for IsSatisfactoryRefactored

TRs for **g**: (1,3) and (5,7)
TRs for **f**: (1,2) and (9,11)
TRs for **c**: (5,6) and (9,10)

For f∧c: tests 9, 10, 11
{*TT, *TF, *FT}

Which solutions are controllable and reachable?

Note that to reach g∧c, g∧f must be false. To reach f∧c, g∧c and g∧f must be false. That suggests we should work from the bottom up!

p_1 =					
#	g	f	P	p_g	p_f
1	T	T	~~T~~	✓	✓
2	T	F	~~F~~		✓
3	F	T	~~F~~	✓	
4	F	F	~~F~~		

#	g		p	p_g	p_c
5	T		~~T~~	✓	✓
6	T		~~F~~		✓
7	F		~~F~~	✓	
8	F		~~F~~		

p_3 =					
#	f	c	p	p_f	p_c
9	T	T	~~T~~	✓	✓
10	T	F	~~F~~		✓
11	F	T	~~F~~	✓	
12	F	F	~~F~~		

RACC for IsSatisfactoryRefactored

TRs for **g**: (1,3) and (5,7)
TRs for **f**: (1,2) and (9,11)
TRs for **c**: (5,6) and (9,10)

For f∧c: tests 9, 10, 11

{ *TT, *TF, *FT }

Which solutions are controllable and reachable?

{ FTT, FTF, FFT }

All others won't reach

Note that to reach g∧c, g∧f must be false. To reach f∧c, g∧c and g∧f must be false. That suggests we should work from the bottom up!

p_1 =					
#	g	f	P	p_g	p_f
1	T	T	~~T~~	✓	✓
2	T	F	~~F~~		✓
3	F	T	~~F~~	✓	
4	F	F	~~F~~		

#	g		p	p_g	p_c
5	T		~~T~~	✓	✓
6	T		~~F~~		✓
7	F		~~F~~	✓	
8	F		~~F~~		

p_3 =					
#	f	c	p	p_f	p_c
9	T	T	~~T~~	✓	✓
10	T	F	~~F~~		✓
11	F	T	~~F~~	✓	
12	F	F	~~F~~		

RACC for IsSatisfactoryRefactored

TRs for ***g***: (1,3) and (5,7)
TRs for ***f***: (1,2) and (9,11)

TRs for ***c***: (5,6) and (9,10)

f∧c: { FTT, FTF, FFT }

For g∧c:

p_1 =					
#	g	f	P	p_g	p_f
1	T	T	T	✓	✓
2	T	F	F		✓
3	F	T	F	✓	
4	F	F	F		

#	g		p	p_g	p_c
5	T		T	✓	✓
6	T		F		✓
7	F		F	✓	
8	F		F		

p_3 =					
#	f	c	p	p_f	p_c
9	T	T	T	✓	✓
10	T	F	F		✓
11	F	T	F	✓	
12	F	F	F		

RACC for IsSatisfactoryRefactored

TRs for ***g***: (1,3) and (5,7)
TRs for ***f***: (1,2) and (9,11)
TRs for ***c***: (5,6) and (9,10)

f∧c: { FTT, FTF, FFT }

For g∧c: tests 5, 6, 7
{ T*T, T*F, F*T }

Which solutions are controllable and reachable?

p_1 =					
#	g	f	P	p_g	p_f
1	T	T	T	✓	✓
2	T	F	F		✓
3	F	T	F	✓	
4	F	F	F		

#	g		p	p_g	p_c
5	T		T	✓	✓
6	T		F		✓
7	F		F	✓	
8	F		F		

p_3 =					
#	f	c	p	p_f	p_c
9	T	T	T	✓	✓
10	T	F	F		✓
11	F	T	F	✓	
12	F	F	F		

RACC for IsSatisfactoryRefactored

TRs for ***g***: (1,3) and (5,7)
TRs for ***f***: (1,2) and (9,11)
TRs for ***c***: (5,6) and (9,10)

f∧c: { FTT, FTF, FFT }

For g∧c: tests 5, 6, 7
{ T*T, T*F, F*T }

Which solutions are controllable and reachable?
{ TFT, TFF, F*T }

All others won't reach f∧c

p_1 =					
#	g	f	P	p_g	p_f
1	T	T	~~T~~	✓	✓
2	T	F	~~F~~		✓
3	F	T	~~F~~	✓	
4	F	F	~~F~~		

#	g		p	p_g	p_c
5	T		~~T~~	✓	✓
6	T		~~F~~		✓
7	F		~~F~~	✓	
8	F		~~F~~		

p_3 =					
#	f	c	p	p_f	p_c
9	T	T	~~T~~	✓	✓
10	T	F	~~F~~		✓
11	F	T	~~F~~	✓	
12	F	F	~~F~~		

RACC for IsSatisfactoryRefactored

TRs for ***g***: (1,3) and (5,7)
TRs for ***f***: (1,2) and (9,11)
TRs for ***c***: (5,6) and (9,10)

f∧c: { FTT, FTF, FFT }

g∧c: { TFT, TFF, F*T }

For g∧f:

p_1 =					
#	g	f	P	p_g	p_f
1	T	T	T	✓	✓
2	T	F	F		✓
3	F	T	F	✓	
4	F	F	F		

#	g		p	p_g	p_c
5	T		T	✓	✓
6	T		F		✓
7	F		F	✓	
8	F		F		

p_3 =					
#	f	c	p	p_f	p_c
9	T	T	T	✓	✓
10	T	F	F		✓
11	F	T	F	✓	
12	F	F	F		

RACC for IsSatisfactoryRefactored

TRs for **g**: (1,3) and (5,7)
TRs for **f**: (1,2) and (9,11)
TRs for **c**: (5,6) and (9,10)

f∧c: { FTT, FTF, FFT }

g∧c: { TFT, TFF, F*T }

For g∧f: tests 1, 2, 3

{ TT*, TF*, FT* }

Which solutions are controllable and reachable?

p_1 =					
#	g	f	P	p_g	p_f
1	T	T	~~T~~	✓	✓
2	T	F	~~F~~		✓
3	F	T	~~F~~	✓	
4	F	F	~~F~~		

#	g		p	p_g	p_c
5	T		~~T~~	✓	✓
6	T		~~F~~		✓
7	F		~~F~~	✓	
8	F		~~F~~		

p_3 =					
#	f	c	p	p_f	p_c
9	T	T	~~T~~	✓	✓
10	T	F	~~F~~		✓
11	F	T	~~F~~	✓	
12	F	F	~~F~~		

RACC for IsSatisfactoryRefactored

TRs for **g**: (1,3) and (5,7)
TRs for **f**: (1,2) and (9,11)
TRs for **c**: (5,6) and (9,10)

f∧c: { FTT, FTF, FFT }

g∧c: { TFT, TFF, F*T }

For g∧f: tests 1, 2, 3

{ TT*, TF*, FT* }

Which solutions are controllable and reachable?

{ TTF, TF*, FT* }

p_1 =					
#	g	f	P	p_g	p_f
1	T	T	T	✓	✓
2	T	F	F		✓
3	F	T	F	✓	
4	F	F	F		

#	g		p	p_g	p_c
5	T		T	✓	✓
6	T		F		✓
7	F		F	✓	
8	F		F		

p_3 =					
#	f	c	p	p_f	p_c
9	T	T	T	✓	✓
10	T	F	F		✓
11	F	T	F	✓	
12	F	F	F		

RACC for IsSatisfactoryRefactored

TRs for **g**: (1,3) and (5,7)
TRs for **f**: (1,2) and (9,11)
TRs for **c**: (5,6) and (9,10)

f∧c: { FTT, FTF, FFT }
g∧c: { TFT, TFF, F*T }
g∧f: { TTF, TF*, FT* }

Fill in the *s for a minimal set of test inputs:

p_1 =					
#	g	f	P	p_g	p_f
1	T	T	T	✓	✓
2	T	F	F		✓
3	F	T	F	✓	
4	F	F	F		

#	g		p	p_g	p_c
5	T		T	✓	✓
6	T		F		✓
7	F		F	✓	
8	F		F		

p_3 =					
#	f	c	p	p_f	p_c
9	T	T	T	✓	✓
10	T	F	F		✓
11	F	T	F	✓	
12	F	F	F		

RACC for IsSatisfactoryRefactored

TRs for **g**: (1,3) and (5,7)
TRs for **f**: (1,2) and (9,11)
TRs for **c**: (5,6) and (9,10)

f∧c: { FTT, FTF, FFT }

g∧c: { TFT, TFF, F*T }

g∧f: { TTF, TF*, FT* }

Fill in the *s for a minimal set of test inputs:

{ FTT, FTF, FFT, TFT, TFF, TTF }

Compare to isSatisfactory:

{ TTF, TFT, TFF, FTF

p_1 =					
#	g	f	P	p_g	p_f
1	T	T	T	✓	✓
2	T	F	F		✓
3	F	T	F	✓	
4	F	F	F		

#	g		p	p_g	p_c
5	T		T	✓	✓
6	T		F		✓
7	F		F	✓	
8	F		F		

p_3 =					
#	f	c	p	p_f	p_c
9	T	T	T	✓	✓
10	T	F	F		✓
11	F	T	F	✓	
12	F	F	F		

RACC Tests for IsSatisfactory

```
GoodFastCheap gfc;

@Before public void setUp() {
  gfc = new GoodFastCheap();    // 8:  F F F
}

@Test public void test1() throws Exception {
  gfc.makeGood();          // 4: T F F
  gfc.makeFast();          // 2: T T F
  assertTrue(gfc.isSatisfactory());
}

@Test public void test2() throws Exception {
  gfc.makeGood();          // 4: T F F
  assertFalse(gfc.isSatisfactory());
}

@Test public void test3() throws Exception {
  gfc.makeFast();          // 6: F T F
  assertFalse(gfc.isSatisfactory());
}

@Test public void test4() throws Exception {
  gfc.makeGood();          // 4: T F F
  gfc.makeCheap();         // 3: T F T
  assertTrue(gfc.isSatisfactory());
}
```

```
@Test public void test5() throws Exception
{
  gfc.makeCheap();         // 7: F F T
  assertFalse(gfc.isSatisfactory());
}

@Test public void test6() throws Exception
{
  gfc.makeFast();          // 6: F T F
  gfc.makeCheap();         // 5: F T T
  assertTrue(gfc.isSatisfactory());
}
```