Logic Coverage

Lecture Objectives:

1) Define the terms predicate and clause.
2) Explain the concept of short circuit evaluation
3) Define the concept of predicate coverage and clause coverage.
4) Explain the concept of combinatorial coverage.
5) Define active clause coverage.
6) Apply active clause coverage to create test cases for an appropriate clause.
7) Explain inactive clause coverage.
8) Define the concept of infeasibility.

~~true~~

~~falset~~

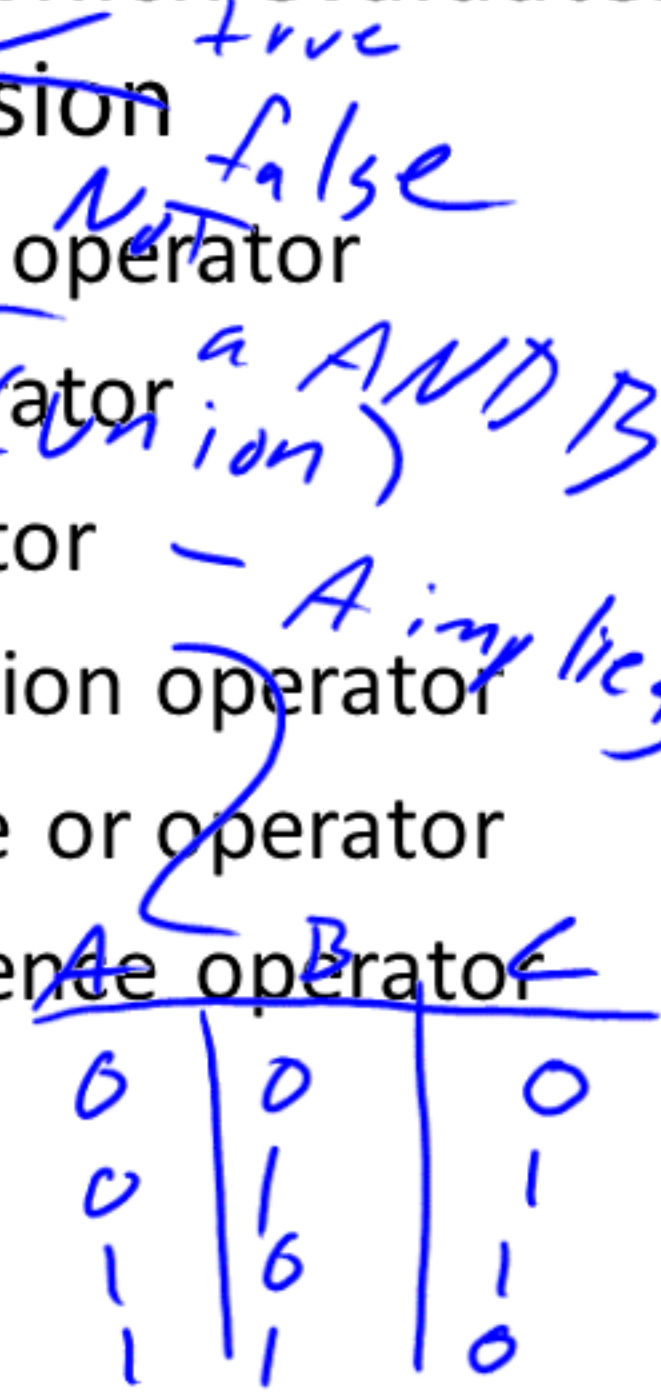
```
if( exams Graded )
{ Return exams();
}
```

Returned

Exam Graded

Predicates

- An expression which evaluates to a boolean expression
  - ¬ The negation operator
  - ^ The And operator
  - ˅ The or operator
  - → The implication operator
  - ⊕ The exclusive or operator
  - ↔ The equivalence operator



- A clause is a predicate that does not contain any logical operators

true false

Clause

$(a = b) \vee c \wedge p(x)$

3 clauses

Logical statement can be broken clauses.

Order of operations Challenges

- What is wrong with the following code?
  - Assuming normal stack conventions
    - void push(int x);
    - int pop();
- Push(pop()-pop())

Size ensure items on the stack

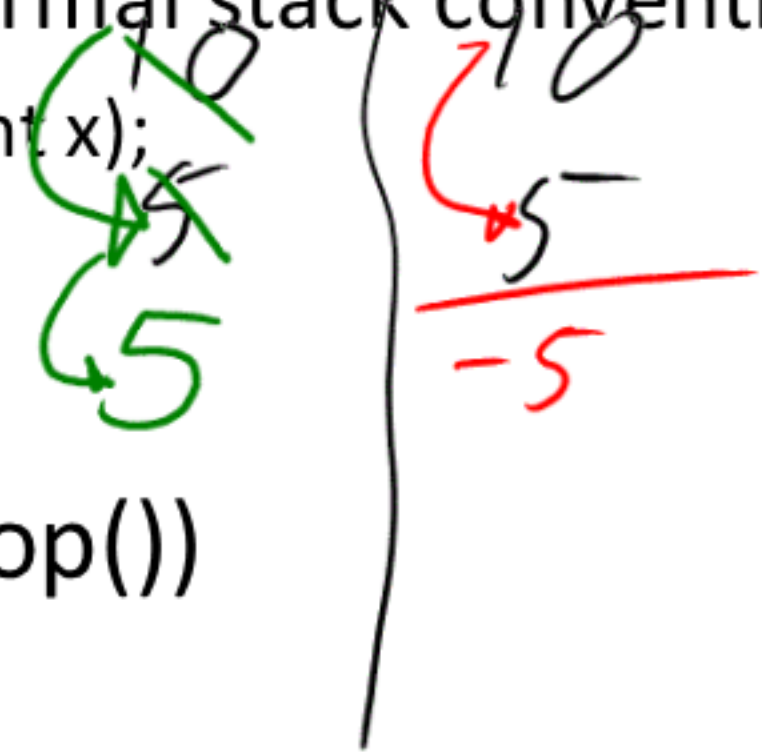
p²

Loop off by one.

return types

Order of operations Challenges

- What is wrong with the following code?
  - Assuming normal stack conventions
    - void push(int x);
    - int pop();



- Push(pop()-pop())

Order of operations

Challenges

- What is wrong with the following code?
  - Assuming normal stack conventions
    - void push(int x);
    - int pop();

- Push(pop()-pop());

```
int a = pop();
int b = pop();
push(a-b);
```

Short Circuit Evaluation (McCartyhy evaluation)

```
int denom = 0;
if (denom && num/denom) { ... //
ensures that calculating num/denom
never results in divide-by-zero error
}
```

May have a divide by zero.

```
if((ptr != NULL) && (*ptr == 5))
{
    ...
}
```

First ..:
NULL ptr
dereference.

Short circuit evaluation

```
if (ishigh && (x==f(x)))
{
.
.
.
}
```

```
bool ishigh;
static int count;
int f(int y)
{
return count++;
}
```

May or may not be executed if ishigh is true.

Predicate Coverage

- Predicate Coverage: For each p in P, TR contains two requirements: p evaluates to true and p evaluates to false

- $((a > b) \cup C) \cap p(x)$

Clause Coverage

- Predicate Coverage: For each p in P, TR contains two requirements: p evaluates to true and p evaluates to false

clause to evaluate to true and false

$$Predicate: ((a > b) \cup C) \cap p(x)$$

A>b=true, false
C=true and false
P(x)=true, false

How many test case? - one Logical branch covered

f	f	t
t	t	f

Problems with PC and CC

- PC does not fully exercise all the clauses, especially in the presence of short circuit evaluation

- CC does not always ensure PC

- That is, we can satisfy CC without causing the predicate to be both true and false

- This is definitely not what we want !

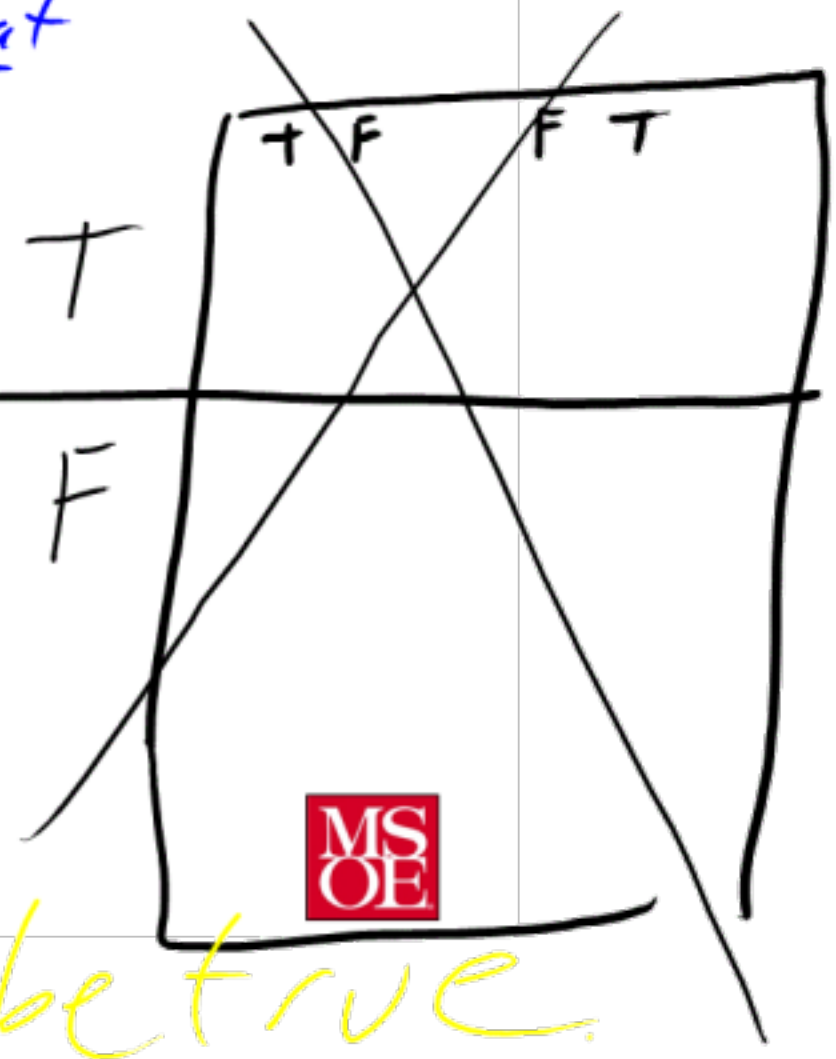
BAD

Combinatorial Coverage

- *Predicate*: $(a \cup b) \cap c$

2 clause / or 2 possibilities

a	b	c	Predicate
F	F	F	F
F	F	T	F
F	T	F	F
F	T	T	T
T	F	F	F
T	F	T	T
T	T	F	F
T	T	T	T



C must be true.

Karnaugh Map

AB / C	TT	TF	FF	FT
T	T	T	F	T
F	F	F	F	F

A C V B C'

Active Clause

- *Which clause makes a bigger difference?*
- *Predicate:* $(a \cup b) \cap c$

C has a larger impact than A or B.

Active Clause

- *Which clause makes a bigger difference?*
- *Predicate*: $(a \cup b) \cap c$

- ***To really test the results of a clause, the clause should be the determining factor in the value of the predicate***

Predicates

P = A ∨ B

if *B = true*, *p* is always true.

so if *B = false*, *A* determines *p*.

if *A = false*, *B* determines *p*.

P = A ∧ B

if *B = false*, *p* is always false.

so if *B = true*, *A* determines *p*.

if *A = true*, *B* determines *p*.

Determining

- Goal : Find tests for each clause when the clause determines the value of the predicate
- This is formalized in several criteria that have subtle, but very important, differences

Active Clause Coverage

- Active Clause Coverage (ACC) : For each *p* in *P* and ~~each~~ major clause c_i in *Cp*, choose minor clauses c_j, j != i, so that c_i determines *p*. TR has two requirements for each c_i : c_i evaluates to true and c_i evaluates to false.

- P=aUb

Active clause coverage

	A	B	P
A	T	T	T
T	T	F	T
F	F	T	T
B	F	F	F
T			
F			

Example

Doesn't buy us anything here.

Active Clause Continued

- *Predicate*: $(a \cup b) \cap c$

Active Clause Coverage

- This is a form of MCDC, which is required by the FAA for safety critical software