CITS5501 Software Testing and Quality Assurance
Semester 1, 2022
Workshop 5 (week 6) – logic-based testing

Reading

It is strongly suggested you complete the recommended readings for weeks 1-5 *before* attempting this lab/workshop.

0. Notation

When writing logic expressions, we will normally use mathematical notation for "and", "or", and "not":

- $\wedge$ – "and"
- $\vee$ – "or"
- $\neg$ – "not"

If writing actual Java code, however, we use the normal Java logical operators:

- `&&` – "and"
- `||` – "or"
- `!` – "not"

(We won't be using any Python in this unit – but for reference, in Python, the logic operators are all spelled out: "and", "or" and "not".)

1. Terminology – clauses and predicates

If you need to, review the lecture material and recommended readings that explain what *predicates* and *clauses* are.

What are the *clauses* in the predicates below?

a. $((f \leqslant g) \wedge (x > 0)) \vee (M \wedge (e < d + c))$

b. $G \vee ((m > a) \vee (s \leqslant o + n)) \wedge U$

2. Making clauses active

To make a particular clause c in some predicate *active* means to assign values to variables so that the truth-value of the whole predicate depends on c.

When coming up with test values which make clauses active, the easiest way of showing your test values is in a table.

E.g. Suppose we have a predicate $s \wedge (m \vee w)$, where

- m = "the moon is full"
- s = "the sky is clear"
- w = "the wind is calm"

If asked to come up with test inputs which make each clause active in turn, and achieve Restricted Active Clause Coverage, we could show them like this:

Test description	**Inputs**	**Predicate value**
Make s active, and		
s = true	s = true, m = true, w = false	true
s = false	s = false, m = true, w = false	false
Make m active, and		
m = true	s = true, m = true, w = false	true
m = false	s = true, m = false, w = false	false
Make w active, and		
w = true	s = true, m = false, w = true	true
w = false	s = true, m = false, w = false	false

(Here, we aren't told what the expected outcome is if the predicate comes out true or false; if we were, we could add a column "Expected outcome" which listed this.)

If you aren't told the exact *types* of variables or methods used in a predicate, that means you should be able to work them out from context. For example, for the predicate

$$(x > 0) \vee (M \wedge (e < d + c))$$

you can assume that M is a boolean, and that x, c, d and e are some integral type (such as `int`).

For each of the clauses in the predicates below, identify test inputs which will make the clause *active* (that is: state what values need to be assigned to the variables in the predicate), and vary that clause so it takes on both true and false values. (In other words: write test values that achieve Restricted Active Clause Coverage.) Explain your reasoning.

a. $A \vee (B \wedge \neg C)$
b. $x > 0 \ \vee (M \wedge (e < d + c))$
c. $G \vee (m \geqslant a) \vee H \wedge U$

3. Scenario – trap-doors

Suppose a component under test has the following requirements:

> If the lever is pulled and the chair is occupied, open the trap-door.
>
> If the button is pressed, open the trap-door.

Represent the component as a set of logic expressions. You should explain what each variable in your expressions means. (For an example, look in section 2 at the way we gave definitions for the variables in the predicate $s \wedge (m \vee w)$.)

(Hint: if you're stuck, try writing out what the component does as one or more "`if`" statements, in pseudocode. Then recall that the set of all predicates in a system means the set of all logical expressions found in things like "`if`" statements.)

4. Scenario – login page

If you don't finish the previous sections of this worksheet in class, then attempt this in your own time.

Suppose you are part of a team developing a website called "RateMyVeterinarian", where people can log in and provide anonymous reviews of the veterinarian services they use.

Requirements for the site are currently being finalised, and one requirement is stated as follows:

> When a user enters a user ID and password into the login page and hits the "log in" button, then if that user ID is listed in the "users" database, and the password matches against the password in the record for that user, and the user record does not state that the account has been disabled, a "Welcome" page should be displayed.

a. How easy to understand do you think this requirement is? If you think it could be made easier to understand, suggest how.

b. One of your colleagues suggests that because correctly authenticating users (and keeping their details secure) is an important feature, this requirement should be thoroughly tested – so you should design a test suite that meets RAC (Restricted Active Clause) levels of coverage. Do you agree? Why or why not?