

Software Testing, Quality Assurance & Maintenance (ECE453/CS447/SE465): Assignment 2

Patrick Lam

Due: March 9, 2009

You may discuss the assignment with others, but I expect each of you to do the assignment independently. I will follow UW's Policy 71 if I discover any cases of plagiarism.

I expect this assignment to be shorter than Assignment 1.

Question 1 (10 points)

You will find a description of the Roomba Red's diagnostic mode at the following page:

`http://www.patricklam.ca/stqam/roomba.shtml`

Create a finite state machine describing this diagnostic mode. Note that your finite state machine will contain self-loops with both input conditions and output conditions; even though we have not seen output conditions, they should be fairly intuitive.

Create test sets which achieve Node Coverage and Edge Coverage for your finite state machine. How useful is each test set for repairing a broken Roomba? Discuss whether any other coverage criteria would help or not.

Question 2 (12 points)

(based on Section 3.2, Q1) For the following predicates:

1. $p = a \wedge (b \leftrightarrow c)$
2. $p = a \vee (b \wedge c) \vee d$
3. $p = a \oplus b$
4. $p = a \rightarrow (b \rightarrow c)$
5. $p = a \leftrightarrow (b \vee c)$

(a, 0 points) Identify the clauses that go with each p .

(b, 2 points) Compute (and simplify) the conditions under which each of the clauses determines predicate p .

(c, 0 points) Write the complete truth table for all clauses. Label your rows. Use the format from the table underneath combinatorial coverage from Section 3.2. Include columns for the truth value of the predicate and for the conditions under which each clause determines the predicate.

(d, 2 points) Identify all pairs of rows from your table that satisfy general active clause coverage (GACC) with respect to each clause.

(e, 2 points) Identify all pairs of rows from your table that satisfy correlated active clause coverage (CACC) with respect to each clause.

(f, 2 points) Identify all pairs of rows from your table that satisfy restricted active clause coverage (RACC) with respect to each clause.

(g, 2 points) Identify all 4-tuples of rows from your table that satisfy general inactive clause coverage (GICC) with respect to each clause. Identify any infeasible GICC test requirements.

(h, 2 points) Identify all 4-tuples of rows from your table that satisfy restricted inactive clause coverage (RICC) with respect to each clause. Identify any infeasible GICC test requirements.

Question 3 (13 points)

For the following predicates and test sets:

1. $p = a \wedge (c \rightarrow b)$
2. $p = a \vee (b \wedge c)$
3. $p = a \oplus (b \wedge \neg c)$
4. $p = a \leftrightarrow (b \rightarrow c)$
5. $p = a \rightarrow (b \wedge c)$

$$T_1 = \{TTF, TFT, FTT, FTF, FFF\}$$

$$T_2 = \{TTT, TFF, FTT, FTF, FFT\}$$

$$T_3 = \{TTT, TTF, TFT, TFF, FTT, FTF, FFT, FFF\}$$

Which coverage criteria do each of the test sets satisfy for each of the predicates?

Question 4 (30 points)

(15) Write a program to generate CACC test sets for predicates. (15) Test your program (same deal as Q3 from assignment 1). (You could also use this program to do questions 2 and 3. However, that won't prepare you very well for any exam questions about logic coverage. It's up to you!)

Please accept predicates in MONA form, as in

<http://www.brics.dk/mona/>

Of course, you don't have to handle all MONA predicates, only those that we've seen in this class. You also don't need to parse exclusive-or. If you want, you can use my Java parser, found at:

<http://www.patricklam.ca/software/monaviewer-1.0.tar.gz>

Question 5 (20 points)

(Section 3.3, Question 4) For the `TestPat` program (changed from `TriTyp`, which seems to be worked out already) from the textbook, complete the test sets for the following coverage criteria by identifying predicates, filling in all values, ensuring reachability, and deriving the expected output. Download the program, compile it, and run it with your resulting test cases to verify correct outputs.

- Predicate coverage (PC)
- Clause coverage (CC)
- Combinatorial coverage (CoC)
- Correlated active clause coverage (CACC)

Question 6 (15 points)

(Section 3.4, modified) Consider the `remove()` method from the Java `Iterator` interface. This method has a complex precondition on the state of the `Iterator`, and the interface implementor may choose to detect violations of the precondition and report them with `IllegalStateExceptions`.

- (5) Use JML to formalize the precondition. (Please do not look at the sample `remove()` precondition on the Internet.)
- Find or write an implementation of an `Iterator`. (5) Test your formalized precondition by running the annotated version of your `remove()` implementation, as compiled by `jmlc`.
- (5) Develop and run CACC tests on the implementation.