تمرین امتیازی اول درس أزمون نرم‌افزار

دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر (پلی‌تکنیک تهران)

بردیا اردکانیان
۹۸۳۱۰۷۲

استاد درس: دکتر گوهری

بهار ۱۴۰۲

Ch6.2-q7: Design an input domain model for the logic coverage web application on the book’s website. That is, model the logic coverage web application using the input domain modeling technique.

(a) List all of the input variables, including the state variables.

Input variables

1. Logical Expression: A logical expression is either true or false.
2. Test Requirements: Truth table of the logic expression to map out all combinations of the clauses.
3. GACC Test: GACC stands for "Generalized Active Clause Coverage." It is a test coverage criterion that ensures that each clause in the logical expression is evaluated as both true and false at least once during testing. This helps validate the behavior of the logical expression under different conditions.
4. CACC Test: CACC stands for "Clause-Active Clause Coverage." It is a test coverage criterion that aims to evaluate each clause in the logical expression as true at least once during testing. This coverage helps identify any potential issues or errors specific to the true evaluations of the clauses.
5. RAAC Test: RAAC stands for "Restricted Active Clause Coverage." It is a test coverage criterion that focuses on evaluating a subset of clauses within the logical expression. The specific subset is determined based on predefined rules or criteria, ensuring that critical clauses or conditions are adequately tested.
6. GICC Test: GICC stands for "Generalized Inactive Clause Coverage." It is a test coverage criterion that verifies the behavior of the logical expression when certain clauses are inactive or evaluated as false. It ensures that the logical expression can handle different combinations of inactive clauses effectively.
7. RICC Test: RICC stands for "Restricted Inactive Clause Coverage." Similar to RAAC, RICC is a test coverage criterion that focuses on evaluating a subset of inactive clauses within the logical expression. This subset is determined based on predefined rules or criteria to ensure that critical inactive clauses are appropriately tested.
8. New Expression: Redirect is a new expression that introduces a redirection mechanism, allowing the software system to redirect the flow of execution or data based on specified conditions.
9. Graph Coverage: Graph coverage is a testing technique that ensures that all possible paths in the control flow graph of a software program are executed during testing. It helps identify potential control flow issues and ensures comprehensive testing of the program's execution paths.
10. Data Flow Coverage: Data flow coverage is a testing method that aims to ensure that all possible data flows within a program are exercised during testing. It focuses on tracking the flow of data variables and ensures that each data definition and usage are covered by test cases.
11. Minimal-MMCUT Coverage: Minimal-MMCUT coverage is a testing approach that focuses on identifying the minimum number of test cases needed to cover all Multiple-Condition Single-Entry and Single-Exit (MC/DC) combinations. It helps ensure that every possible condition within a decision is tested, considering the effect of each condition on the overall decision outcome.

12. Share: Share the expression and test results that redirects the request to another website.

State variable:

1. User Authentication Status: This feature represents a required state variable that keeps track of whether a user is currently authenticated or not.
2. Session ID: This feature allows the user to track the session ID, which is a unique identifier associated with a user's session.
3. Error Messages and Warnings: State variables for error messages and warnings are useful in software development and debugging.

(b) Define characteristics of the input variables. Make sure you cover all input variables.

1. Logical Expression: A logical expression is either true or false.
2. Truth Table: Truth table variable contains a value can either be true or false.
3. GACC Test: GACC test should be applied to the logical expression. This input variable contains a value that can be either true or false.
4. CACC Test: CACC test should be applied to the logical expression. This input variable contains a value that can be either true or false.
5. RAAC Test: RACC test should be applied to the logical expression. This input variable contains a value that can be either true or false.
6. GICC Test: GICC test should be applied to the logical expression. This input variable contains a value that can be either true or false.
7. RICC Test: RICC test should be applied to the logical expression. This input variable contains a value that can be either true or false.

(c) Partition the characteristics into blocks.

Logical Expression

- Data Type: String
- Length: The length can vary according to the definition or requirement.
- Format: It follows the syntax of a logical expression, including operators, parentheses, and other relevant syntax elements.

Truth Table

- Data Type: Boolean

GACC Test

- Data Type: Boolean

CACC Test

- Data Type: Boolean

RACC Test

- Data Type: Boolean

GICC Test

- Data Type: Boolean

RICC Test

- Data Type: Boolean

(d) Designate one block in each partition as the “Base” block.

Logical Expression

- Base Block 1: Logic Expression is NULL
- Base Block 2: Logic Expression is Empty
- Base Block 3: Format

Truth Table

- Base Block: DataType

GACC Test

- Base Block: DataType

CACC Test

- Base Block: DataType

RACC Test

- Base Block: DataType

GICC Test

- Base Block: DataType

RICC Test

- Base Block: DataType

(e) Define values for each block.

Logical Expression

- Base Block 1: NULL
- Base Block 2: “”
- Format: A string of characters that contains logical operators (AND, OR, NOT), Boolean values (true, false), parentheses, and variables (a, b). E.g. “a ^ b>”

Truth Table

- Base Block: DataType

GACC Test

- Base Block: DataType

CACC Test

- Base Block: DataType

RACC Test

- Base Block: DataType

GICC Test

- Base Block: DataType

RICC Test

- Base Block: DataType

(f) Define a test set that satisfies Base Choice Coverage (BCC). Write your tests with the values from the previous step. Be sure to include the test oracles.

The Truth Table, GACC, CACC, RACC, GICC, RICC characteristics are independent but have to be tested separately.

Test Set 1:

- Logic Expression: NULL
- Truth Table: True
- GACC: False
- CACC: False
- RACC: False
- GICC: False
- RICC: False

Test Set 2:

- Logic Expression: “”
- Truth Table: True
- GACC: False
- CACC: False
- RACC: False
- GICC: False
- RICC: False

Test Set 3:

- Logic Expression: “a ^ b”
- Truth Table: True
- GACC: False
- CACC: False
- RACC: False
- GICC: False
- RICC: False

Test Set 4:

- Logic Expression: “a ^ b”
- Truth Table: False
- GACC: True
- CACC: False
- RACC: False

- GICC: False
- RICC: False

Test Set 5:

- Logic Expression: “a ^ b”
- Truth Table: False
- GACC: False
- CACC: True
- RACC: False
- GICC: False
- RICC: False

Test Set 6:

- Logic Expression: “a ^ b”
- Truth Table: False
- GACC: False
- CACC: False
- RACC: True
- GICC: False
- RICC: False

Test Set 7:

- Logic Expression: “a ^ b”
- Truth Table: False
- GACC: False
- CACC: False
- RACC: False
- GICC: True
- RICC: False

Test Set 8:

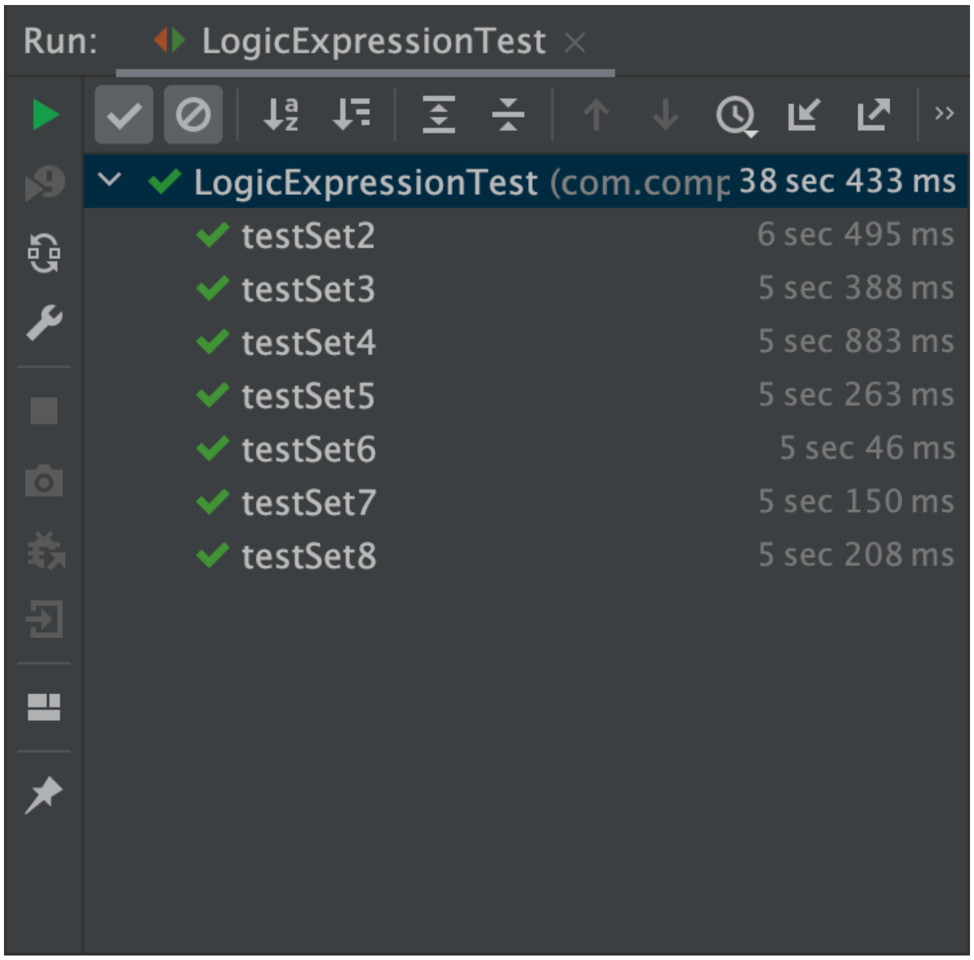
- Logic Expression: “a ^ b”
- Truth Table: False
- GACC: False
- CACC: False
- RACC: False
- GICC: False
- RICC: True

(g) Automate your tests using the web test automation framework HttpUnit. Demonstrate success by submitting the HttpUnit tests and a screen dump or output showing the result of execution.
(Note to instructors: HttpUnit is based on JUnit and is quite similar. The tests must include a URL and the framework issues the appropriate HTTP request. We usually use this question as a non-required bonus, allowing students to choose whether to learn HttpUnit on their own.)

```
package com.company;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;

import static org.junit.Assert.assertTrue;

public class LogicExpressionTest {
   private WebDriver driver;

   @Before
   public void setUp() {
       // Set up the WebDriver instance
       System.setProperty("webdriver.chrome.driver", "../../lib/geckodriver");
       driver = new FirefoxDriver();
   }

   @Test
   public void testSet2() {
       // Navigate to the login page
       driver.get("https://cs.gmu.edu:8443/offutt/coverage/LogicCoverage");

       // Find the login form and enter the expression
       WebElement form = driver.findElement(By.tagName("form"));
       form.findElement(By.name("expression")).sendKeys("");

       /// Click on the truth table button
       form.findElement(By.cssSelector("input[value='Truth
Table'][name='action']")).click();

       // Check the results
       boolean EmptyExpression = driver.getPageSource().contains("The expression
is empty");
       System.out.println("Truth Table Displayed: " + EmptyExpression);

       // Verify that the truth table was displayed
       assertTrue("Expression was not empty", EmptyExpression);
   }
```

```
    @Test
    public void testSet3() {
        // Navigate to the login page
        driver.get("https://cs.gmu.edu:8443/offutt/coverage/LogicCoverage");

        // Find the login form and enter the expression
        WebElement form = driver.findElement(By.tagName("form"));
        form.findElement(By.name("expression")).sendKeys("a ^ b");

        /// Click on the truth table button
        form.findElement(By.cssSelector("input[value='Truth
Table'][name='action']")).click();

        // Check the results
        boolean truthTableDisplayed = driver.getPageSource().contains("Truth
Table:");
        System.out.println("Truth Table Displayed: " + truthTableDisplayed);

        // Verify that the truth table was displayed
        assertTrue("Truth table not displayed", truthTableDisplayed);
    }

    @Test
    public void testSet4() {
        // Navigate to the login page
        driver.get("https://cs.gmu.edu:8443/offutt/coverage/LogicCoverage");

        // Find the login form and enter the expression
        WebElement form = driver.findElement(By.tagName("form"));
        form.findElement(By.name("expression")).sendKeys("a ^ b");

        /// Click on the truth table button
        form.findElement(By.cssSelector("input[value='GACC'][name='action']")).cli
ck();

        // Check the results
        boolean testTableDisplayed = driver.getPageSource().contains("Truth
Table:");
        System.out.println("GACC Test Displayed: " + testTableDisplayed);

        // Verify that the truth table was displayed
        assertTrue("GACC not displayed", testTableDisplayed);

        // Check the results
        boolean truthTableDisplayed = driver.getPageSource().contains("Truth
Table:");
        System.out.println("Truth Table Displayed: " + truthTableDisplayed);
        // Verify that the truth table was displayed
        assertTrue("Truth table not displayed", truthTableDisplayed);
    }

    @Test
    public void testSet5() {
        // Navigate to the login page
        driver.get("https://cs.gmu.edu:8443/offutt/coverage/LogicCoverage");

        // Find the login form and enter the expression
        WebElement form = driver.findElement(By.tagName("form"));
        form.findElement(By.name("expression")).sendKeys("a ^ b");

        /// Click on the truth table button
        form.findElement(By.cssSelector("input[value='CACC'][name='action']")).cli
ck();

        // Check the results
        boolean testTableDisplayed = driver.getPageSource().contains("Truth
Table:");
        System.out.println("CACC Test Displayed: " + testTableDisplayed);

        // Verify that the truth table was displayed
        assertTrue("CACC not displayed", testTableDisplayed);

        // Check the results
        boolean truthTableDisplayed = driver.getPageSource().contains("Truth
Table:");
        System.out.println("Truth Table Displayed: " + truthTableDisplayed);

        // Verify that the truth table was displayed
        assertTrue("Truth table not displayed", truthTableDisplayed);
    }

    @Test
    public void testSet6() {
        // Navigate to the login page
        driver.get("https://cs.gmu.edu:8443/offutt/coverage/LogicCoverage");

        // Find the login form and enter the expression
        WebElement form = driver.findElement(By.tagName("form"));
        form.findElement(By.name("expression")).sendKeys("a ^ b");

        /// Click on the truth table button
        form.findElement(By.cssSelector("input[value='GACC'][name='action']")).cli
ck();

        // Check the results
        boolean testTableDisplayed = driver.getPageSource().contains("Truth
Table:");
        System.out.println("RACC Test Displayed: " + testTableDisplayed);

        // Verify that the truth table was displayed
        assertTrue("RACC not displayed", testTableDisplayed);

        // Check the results
        boolean truthTableDisplayed = driver.getPageSource().contains("Truth
Table:");
        System.out.println("Truth Table Displayed: " + truthTableDisplayed);

        // Verify that the truth table was displayed
        assertTrue("Truth table not displayed", truthTableDisplayed);
    }

    @Test
    public void testSet7() {
        // Navigate to the login page
        driver.get("https://cs.gmu.edu:8443/offutt/coverage/LogicCoverage");

        // Find the login form and enter the expression
        WebElement form = driver.findElement(By.tagName("form"));
        form.findElement(By.name("expression")).sendKeys("a ^ b");

        /// Click on the truth table button
        form.findElement(By.cssSelector("input[value='GACC'][name='action']")).cli
ck();

        // Check the results
        boolean testTableDisplayed = driver.getPageSource().contains("Truth
Table:");
        System.out.println("GICC Test Displayed: " + testTableDisplayed);

        // Verify that the truth table was displayed
        assertTrue("GICC not displayed", testTableDisplayed);

        // Check the results
        boolean truthTableDisplayed = driver.getPageSource().contains("Truth
Table:");
        System.out.println("Truth Table Displayed: " + truthTableDisplayed);

        // Verify that the truth table was displayed
        assertTrue("Truth table not displayed", truthTableDisplayed);
    }

    @Test
    public void testSet8() {
        // Navigate to the login page
        driver.get("https://cs.gmu.edu:8443/offutt/coverage/LogicCoverage");
        // Find the login form and enter the expression
        WebElement form = driver.findElement(By.tagName("form"));
        form.findElement(By.name("expression")).sendKeys("a ^ b");

        /// Click on the truth table button
        form.findElement(By.cssSelector("input[value='RICC'][name='action']")).cli
ck();

        // Check the results
        boolean testTableDisplayed = driver.getPageSource().contains("Truth
Table:");
        System.out.println("RIC Test Displayed: " + testTableDisplayed);

        // Verify that the truth table was displayed
        assertTrue("RICC not displayed", testTableDisplayed);

        // Check the results
        boolean truthTableDisplayed = driver.getPageSource().contains("Truth
Table:");
        System.out.println("Truth Table Displayed: " + truthTableDisplayed);

        // Verify that the truth table was displayed
        assertTrue("Truth table not displayed", truthTableDisplayed);
    }

    @After
    public void tearDown() {
        // Close the WebDriver instance
        driver.quit();
    }
}
```

خروجی



```
Run:  LogicExpressionTest
LogicExpressionTest (com.comp 38 sec 433 ms
    testSet2    6 sec 495 ms
    testSet3    5 sec 388 ms
    testSet4    5 sec 883 ms
    testSet5    5 sec 263 ms
    testSet6     5 sec 46 ms
    testSet7    5 sec 150 ms
    testSet8    5 sec 208 ms
```