CS409
Software Testing

TAN, Shin Hwei

陈馨慧

Southern University of Science and Technology

Slides adapted from Introduction to Software Testing, Edition 2 (Ch 8)

Logic Source Code lab

- Logic-lab in GitHub Classroom:

https://classroom.github.com/a/WbKNTVOr

- We will try logic coverage for your selected app!

Step I: Select method with 2-3 clauses from your app （My example: transistor app）

```
/* Constructor */
 public StationFetcher(Activity activity, File folder, Uri stationUri, String stationName) {
     mActivity = activity;
     mFolder = folder;
     mStationUri = stationUri;
     mStationName = stationName;
     mFolderExists = mFolder.exists();
     mStationUriScheme = stationUri.getScheme();
     if (stationUri != null && mStationUriScheme != null &&
             mStationUriScheme.startsWith("http")) {
         ...      }
 }
```

When reading the code, think about:

- What is the type of variables like mStationUriScheme?
- How to construct the inputs according to IDM?

From https://github.com/y20k/transistor/blob/83ce1f46d3b1397fa0a2073ee065dda4a4ba19dc/app/src/main/java/org/y20k/transistor/helpers/StationFetcher.java

Step 2: Simplify

stationUri != null && mStationUriScheme != null &&
mStationUriScheme.startsWith("http")

a: stationUri != null

b: mStationUriScheme != null

c: mStationUriScheme.startsWith("http")

a && b && c

Step 2: Get Predicate Coverage

p = a && b && c

1. Make p=true
2. Make p=false

Step 2: Predicate Coverage

p = a && b && c

Make p=true

- a=true
- b=true
- c = true

Make p=false

- a=false
- b=false
- c=false

Step 3: Get Clause Coverage

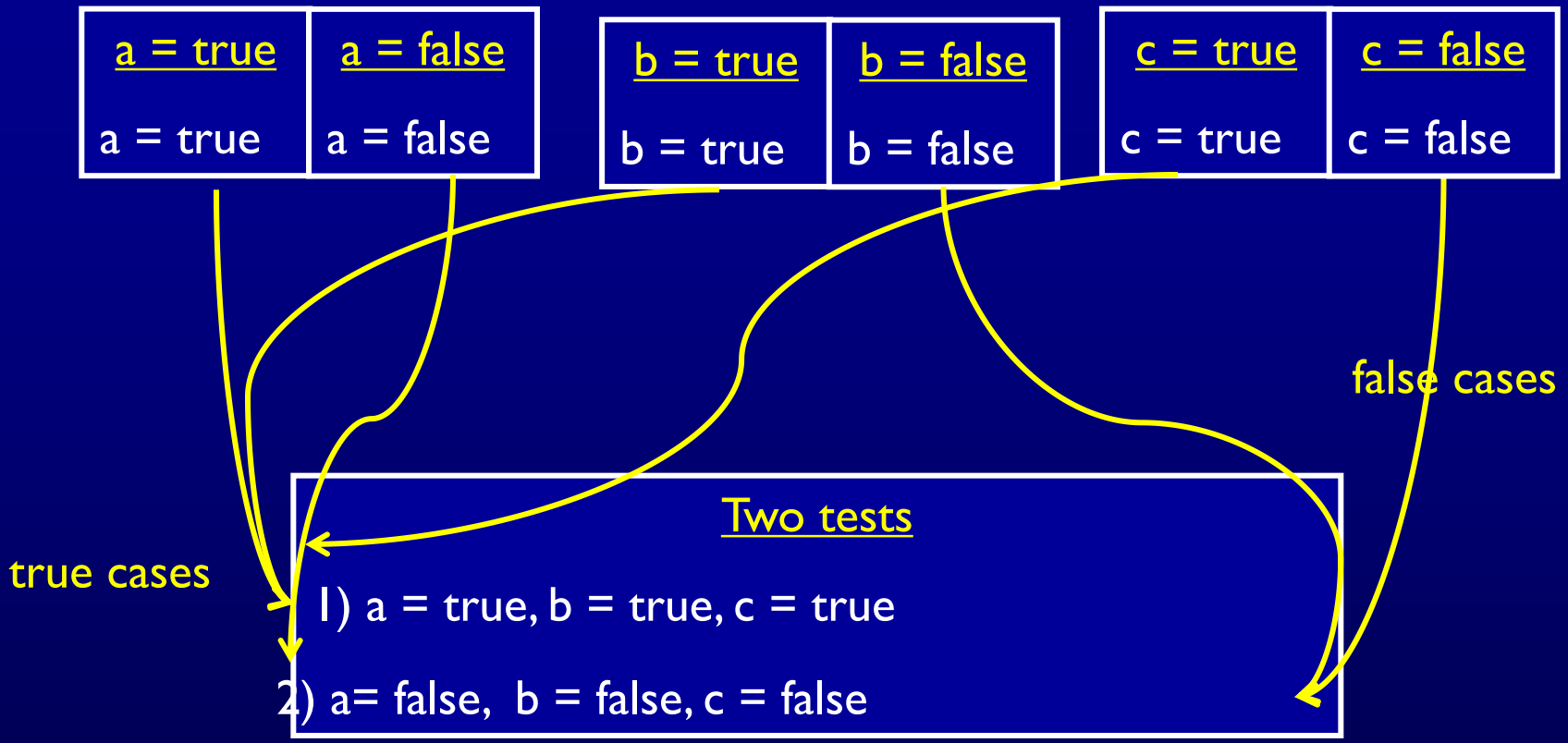
p = a && b && c

Make each clause true & false

Step 3: Get Clause Coverage

p = a && b && c

Make each clause true & false



Step 4: Get Combinatorial Coverage

p = a && b && c

Combinatorial Coverage (CoC) : For each p in P, TR has test requirements for the clauses in Cp to evaluate to each possible combination of truth values.

How many clauses? 3

How many tests is need? 2^3

work …

Step 4: Get Combinatorial Coverage

p = a && b && c

	a	b	c	a && b && c
1	T	T	T	T
2	T	T	F	F
3	T	F	T	F
4	T	F	F	F
5	F	T	T	F
6	F	T	F	F
7	F	F	T	F
8	F	F	F	F

Step 5: Get Correlated Active Clause Coverage

p = a && b && c

Consider a as active clause

Correlated Active Clause Coverage (CACC) : TR has two requirements for each c_i : c_i evaluates to true and c_i evaluates to false. The values chosen for the minor clauses c_j must cause p to be true for one value of the major clause c_i and false for the other, that is, it is required that $p(c_i = true) != p(c_i = false)$.

work …

Step 6: Get Restricted Active Clause Coverage

p = a && b && c

Consider a as active clause

Restricted Active Clause Coverage (RACC) :TR has two requirements for each c_i: c_i evaluates to true and c_i evaluates to false. The values chosen for the minor clauses c_j must be the same when c_i is true as when c_i is false, that is, it is required that $c_j(c_i = true) = c_j(c_i = false)$ for all c_j.

work …

CACC and RACC

	a	b	c	a && b && c
1	T	T	T	T
2	T	T	F	F
3	T	F	T	F
4	T	F	F	F
5	F	T	T	F
6	F	T	F	F
7	F	F	T	F
8	F	F	F	F

major clause

P_a : b=true or c = true

CACC can be satisfied by choosing any of rows 1 AND any of rows 5, 6, 7, 8 – a total of 4 pairs

	a	b	c	a && b && c
1	T	T	T	T
2	T	T	F	F
3	T	F	T	F
4	T	F	F	F
5	F	T	T	F
6	F	T	F	F
7	F	F	T	F
8	F	F	F	F

RACC can only be satisfied by row pairs (1, 5)

Only three pairs

Step 7: Find p_c that determine p

p = a && b && c

Consider a as active clause

$p_a = p_{a=true} \oplus p_{a=false}$

work …

Step 7: Find p_c that determine p

p = a&&(b && c)

$P_a = P_{a=true} \oplus P_{a=false}$

= (true && b && c) $\oplus$ (false && b && c)

= (b && c) $\oplus$ (false)

= b && c

Step 8: Tried it on Tool

- Tried to use the logic coverage tool at:
- https://cs.gmu.edu:8443/offutt/coverage/LogicCoverage

Step 9: Write test

- Write JUnit test for achieving:
  - Predicate Coverage
  - Clause Coverage
  - CACC
  - RACC

From unit test to GUI test

ANDROID SPECIFIC

Unit Level: How to print?

```
import android.util.Log;
 /* Constructor */
    public StationFetcher(Activity activity, File folder, Uri stationUri, String stationName) {
        ...
        Log.w("myApp", "statementUri:"+mStationUriScheme);
        if (stationUri != null && mStationUriScheme != null &&
                    mStationUriScheme.startsWith("http")) {
            ...        }
  }
```

API	Log levels
Log.e("ApiUrl = ", "MyApiUrl")	Error
Log.w("ApiUrl = ", "MyApiUrl")	Warning
Log.i("ApiUrl = ", "MyApiUrl")	information
Log.d("ApiUrl = ", "MyApiUrl")	debug
Log.v("ApiUrl = ", "MyApiUrl")	verbose

GUI Level:View the output via Logcat

- View logs while replaying test scripts
  - Command line:
    - adb logcat
  - Android Studio:
    - Click View > Tool Windows > Logcat

References:

https://appuals.com/install-adb-windows-7-8-10/

https://developer.android.com/studio/debug/am-logcat

To display the log messages for an app:

1. Build and run your app on a device.
2. Click **View > Tool Windows > Logcat** (or click **Logcat** in the tool window bar).

The Logcat window shows the log messages for the selected app, as selected from the dropdown lists at the top of the window, as shown in figure 1.

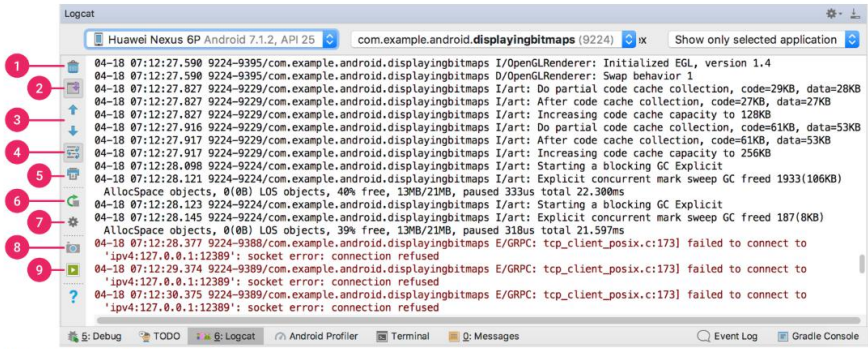


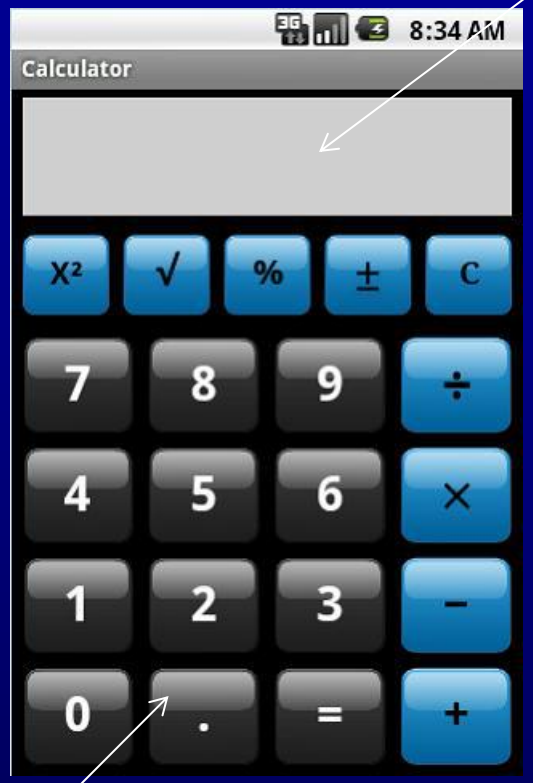
Figure 1. Logcat window

Input from Unit level to GUI level

- Unit Level
  - From the current methods, does the values for variables like mStationUriScheme comes from inputs in buttons, text fields?
- GUI Level:
  - Could we enter input from the unit level at the GUI level?
    - e.g., in the unit level, we could give "a" as input for calculator but couldn't at the GUI level

Expected Output (Assertions) from Unit level to GUI level

- Unit Level
  - assert(2, add(1,1))
- GUI Level:
  - How will the output be display?
    - New Screen?
    - Dialog box?
    - Option Menu?
    - Crash? (If the input could lead to crash in GUI level, then you may have found a bug!)



What to submit

- For your selected app,
  1. Select method with 2-3 clauses from your app
  2. Write JUnit tests for
     1. Predicate Coverage
     2. Clause Coverage
     3. CACC
     4. RACC
  3. Record **GUI tests for satisfying predicate coverage using** MonkeyRunner/Espresso
  4. Prepare a README.md file
     1. Write your name and student id
     2. Answer the following question:
        1. Can you translate the tests from unit level to GUI level? If you cannot get the input from unit level to GUI level, explain why.
        2. If yes, can the GUI tests above find any new bugs? If you find a new bug, file a bug report following the example at https://github.com/orgs/cs409-software-testing2020/teams/allstudents/discussions/2 (Don't forget to include the link to the app repository to get the bonus!)

Administrative Info

- MP2 due November 13, 23:59pm (Late submission will get zero score)
- There are two parts of MP2:
  - Two invitation links:
    - IDM for Joda Time
    - IDM for your selected app
    - Q: Where should we write the answers?
    - All answers (except for JUnit tests and bonus question) should be written in README.md (can overwrite the default README.md)

  Q: Should we commit the code for the selected app?

    - No, only the source code for the selected method needs to be committed
    - However, you should try adding and running your JUnit tests in your selected app to check if you have successfully find a bug for the bonus question!
    - **Start early!**