On Test Coverage

Jiří Patera

University of West Bohemia in Pilsen
Czech Republic

Overview

- Types of testing
- Test coverage
- Distributed environment
- Message races
- Automatic testing
- Relative debugging
- ?Possible approach?

Types of Testing

- Unit Testing:
  - Unit tests are written by developers.
  - Typically test an individual class or small group of classes.
- Regression Testing:
  - Retesting of a previously tested program.
  - After its modification to ensure that faults have not been introduced as a result of the changes made.
- Exhaustive Testing:
  - Executing the program with all possible combinations of values for program inputs/variables.
  - Feasible only for small, simple programs.

Selecting Tests

- Black-Box Testing:
  - It is based on an analysis of the specification of the component without reference to its internal workings.
- White-Box Testing:
  - Explicit knowledge of the internal workings of the item being tested are used to select the test data.
- For a complete software examination, both white box and black box tests are required.

Test Coverage (Control-Flow)

- Statement Coverage:
  - Percentage of statements in the program that have been executed at least once.
- Branch Coverage:
  - Percentage of decision points in the program whose branches have been executed at least once.
- Path Coverage:
  - This measure reports whether each of the possible paths in each function have been followed.
  - A path is a unique sequence of branches from the function entry to the exit.

Test Coverage (Data-Flow)

- This variation of path coverage considers only the sub-paths from variable definitions to its subsequent uses:
  - All definitions criterion:
    - every definition to some reachable use
  - All uses criterion:
    - some definition to every reachable use
  - All definition-use criterion:
    - every definition to every reachable use

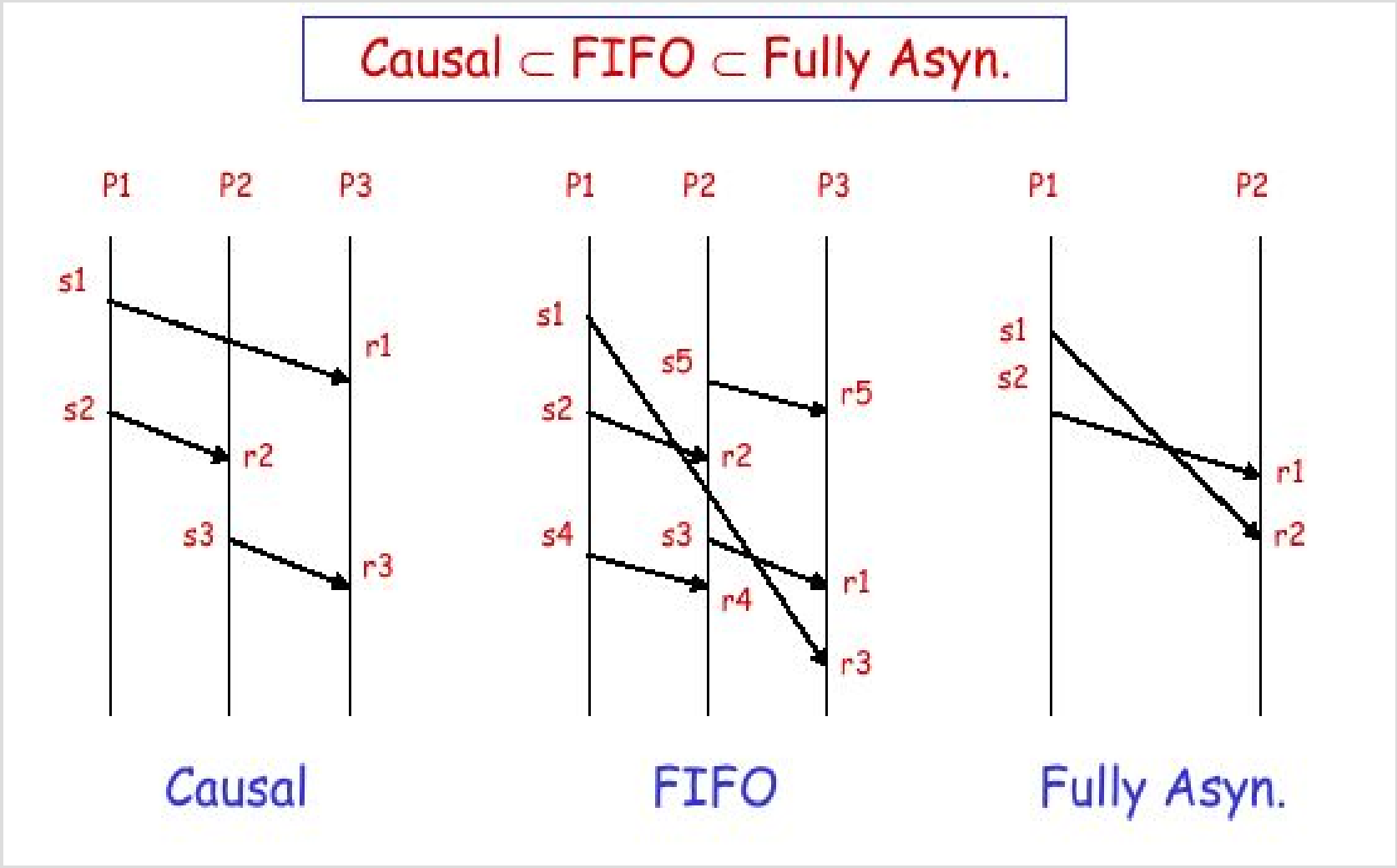
Distributed Environment

- Programs are non-deterministic
  - i.e. each execution with the same input data can lead to a different output
- Concurrency and race conditions make revealing of errors hard for a programmer.
- Reproducing faults is very hard.
- Exhaustive testing is not feasible.

Async vs. Sync

- In an asynchronous message-passing program, each process uses non-blocking send and blocking receive operations.
- This is in contrast with synchronous message-passing, where blocking send and blocking receive operations are used.

Communication Subsystem



Message Race

- There is a message race between two events if the messages sent by the two send events may be received by the same receive event.
- SR-sequence is a triple (S, R, Y) where S is a set of send events, R a set of receive events, and Y a set of synchronizations.
- Given an SR-sequence, race analysis can be performed and the set of send events for every receive event can be determined.

Message Race Detection

- Race analysis of the SR-sequence is to determine the race set for every receive event in the SR-sequence.
- Let ***r*** be the receive event of P_i. A send event ***s*** of P_j is in race if:
  - the message sent by ***s*** is destined to P_i,
  - neither ***s*** happens before ***r***, nor ***r*** happens before ***s*** (i.e. ***s*** and ***r*** are concurrent events).

Automatic Testing in Java

- Usually, there are no ***failures*** and extreme ***network delays*** in the environment where the application is being developped.
- A way to test the application is a simulation of network delays so that hidden errors appear.
- Can be done by inserting auxiliary code to the tested program and re-executing it.

Replaying a Distributed Execution

- Two possible ways of record/replay:
  - Deterministic replay
    - useful for short computations
    - records all events from the beginning of the computation
  - Incremental replay
    - useful for long-running computations
    - records gradually last known consistent state and all events that happened after it

Inserting Code

- Tester's code can be inserted to the program at the bytecode level.
- Redirection of the calls to the Java Datagram API so that they were served by a class which inserts random delays, duplicates/throws away some messages, etc. and, afterwards, calls the JVM API.
- Advantages: No re-compilation is needed and no change of the application's source code is needed. Reference [1][3].

Probe Effect

- It can affect any observation.
- Modifying a distributed program in any way (adding/removing a piece of code) may alter the timing in the system.
- In order to observe a system we put a probe (auxiliary code) into the internals of the system.

Relative Debugging

- Does not force the programmer to understand the expected state and internal operation of a program.
- Based on the comparison of data structures between programs at run time.
- Finds when and where differences between the old and new codes occured.
- Reference [2].

? Possible Approach ?

- Execute the developped application, record trace data, and make sure it ended up as requested by the specification.
- Testing Phase: Simulate network behaviour in such a way that the most of the states from the state lattice were covered (heuristics). Record trace data.
- Compare the data sets with most attention given to places where message races occured.

Thank You

Thank you for your attention...

Now it is time for your questions.

References

- [1] E. Farchi, Y. Krasny, Y. Nir: Automatic Simulation of Network Problems in UDP-Based Java Programs, Proceedings of IPDPS'04, IBM Research Labs, 2004.
- [2] Watson, G., Abramson, D. “Parallel Relative Debugging for Distributed Memory Applications: A Case Study”, http://www.guardsoft.com, International Conference on Parallel and Distributed Processing Techniques and Applications June, USA, 2001.
- [3] O. Edelstein, E. Farchi, Y. Nir, G. Ratsaby, S. Ur: Multithreaded Java Program Test Generation, IBM Systems Journal, Vol. 41 (No. 1) , 2002.