Metamorphic Testing, PEN Testing and Fuzzing

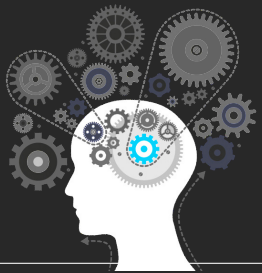
Dr. Isaac Griffith Idaho State University

Outcomes

Idaho State University | Computer Science

After today's lecture you will be able to:

- Understand the basic idea of metamorphic testing and the types of issues it addresses.
- Understand the basic idea of penetration testing and the tools that are used.
- Understand the basic idea of fuzzing and the approach used.



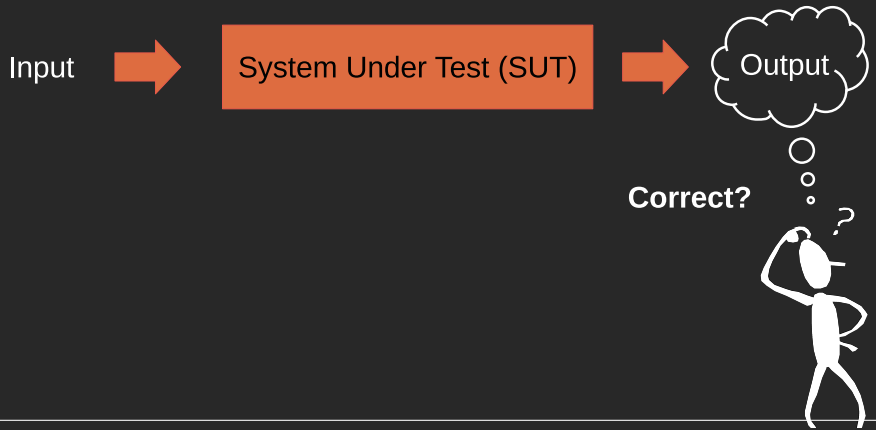
Inspiration

Metamorphic Testing

CS 3321

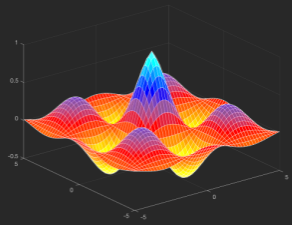
Test Oracle

Mechanism to decide whether a test output is correct or not

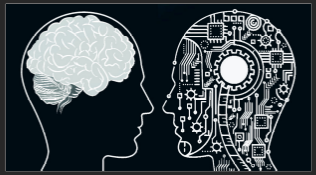


Oracle Problem

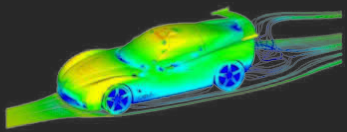
Sometimes it is not feasible to check the correctness of a test output



Scientific Calculations



Artificial Intelligence



Simulation & Modeling

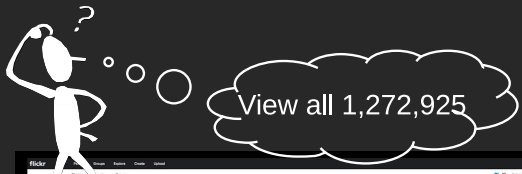
Let’s see some examples

Examples

Source = s
Destination = d

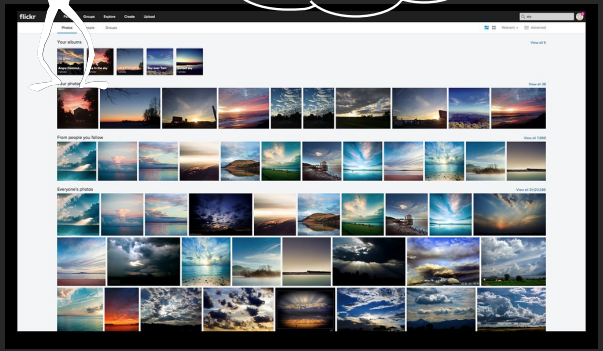
shortestPath(G, s, d)

Examples

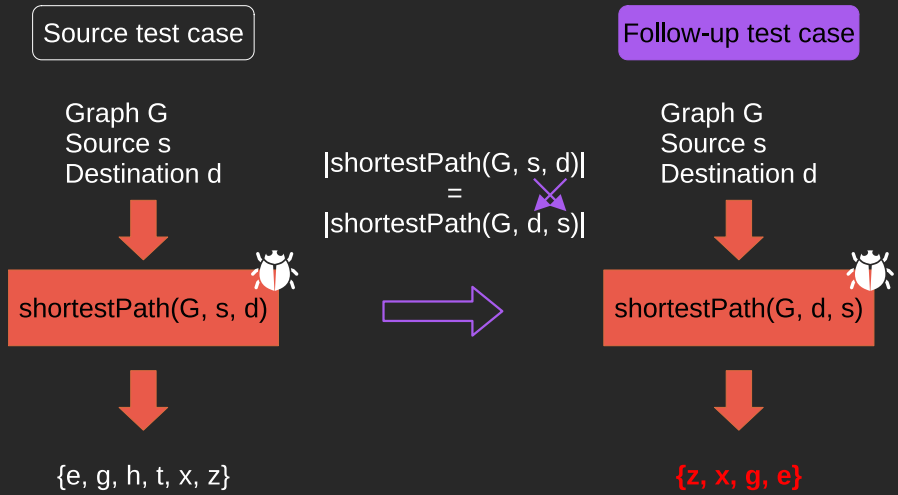


flickr

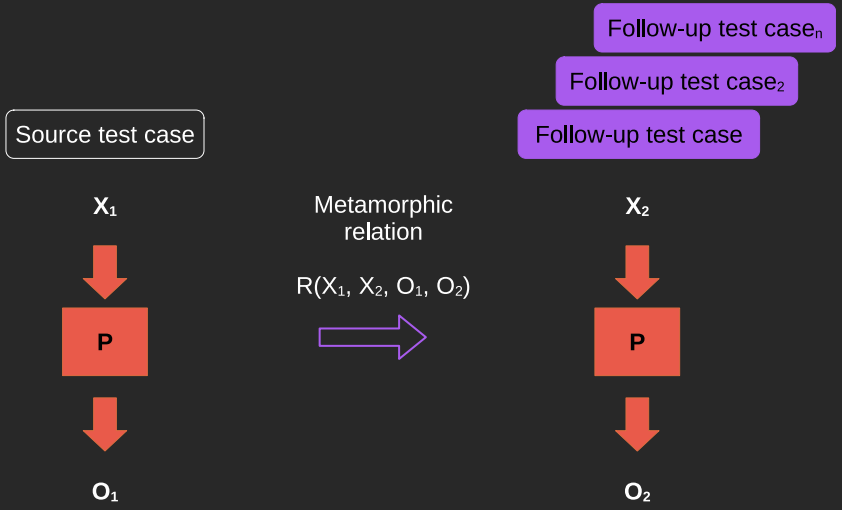
Software



Examples


Source test case
Follow-up test case
Graph G
Source s
Destination d
|shortestPath(G, s, d)|
=
|shortestPath(G, d, s)|
Graph G
Source s
Destination d
shortestPath(G, s, d)
shortestPath(G, d, s)
{e, g, h, t, x, z}
{z, x, g, e}

Examples


Follow-up test case$_n$
Follow-up test case$_2$
Follow-up test case
Source test case
X_1
Metamorphic relation
$R(X_1, X_2, O_1, O_2)$
X_2
P
P
O_1
O_2

Metamorphic Testing Process

1. Identification of metamorphic relations.
2. Generation/Selection of source test cases.
3. Generation of follow-up test cases.
4. Checking of metamorphic relations

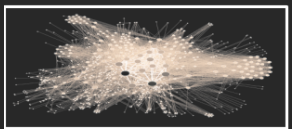
Oh, I get it. This is about alleviating the oracle problem. Is that it?

Yes! but MT can also support test data generation!

Test Data Generation

|shortestPath(G, s, d)|
=
|shortestPath(G, d, s)|

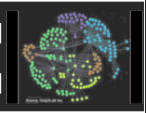
Metamorphic relation



Graph database

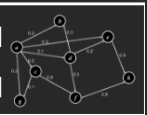
MT 1

|shortestPath(G, k, t)|
=
|shortestPath(G, t, k)|



MT 2

|shortestPath(P, c, a)|
=
|shortestPath(P, c, a)|



MT 3

|shortestPath(S, 2, 41)|
=
|shortestPath(S, 41, 2)|

Test Data Generation

If Q_2 == Q_1 AND size = large
then Count(Q_2) <= Count (Q_1)

Metamorphic relation

Random word generator

MT 1

Q_1 = “dog”
Q_2 = “dog”, size = large
Count(Q_2) <= Count(Q_1)

MT 2

Q_1 = “wind”
Q_2 = “wind”, size = large
Count(Q_2) <= Count(Q_1)

MT 3

Q_1 = “thing”
Q_2 = “thing”, size = large
Count(Q_2) <= Count(Q_1)

MT 4

Q_1 = “money”
Q_2 = “money”, size = large
Count(Q_2) <= Count(Q_1)

State of the art

CS 3321

Domains

Idaho State University | Computer Science

From a survey of 84 Case Studies

- Numerical programs ~5%
- Variability and decision support ~5%
- Compilers ~4%
- Components ~3%
- Autonomous Vehicles ~2%
- Bioinformatics ~8%
- Machine Learning ~8%
- Simulation and Modeling ~8%
- Embedded Systems ~8%
- Computer Graphics ~11%
- Web Services/Apps ~14%
- Other (Adobe, NASA, CyberSec) ~24%

Lesson Learned

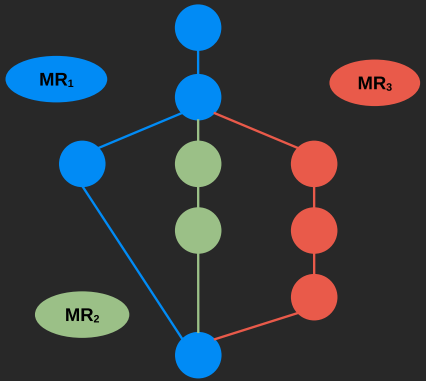
Metamorphic testing requires good knowledge of the problem domain

Lesson Learned

Different metamorphic relations can have different fault-detection capability

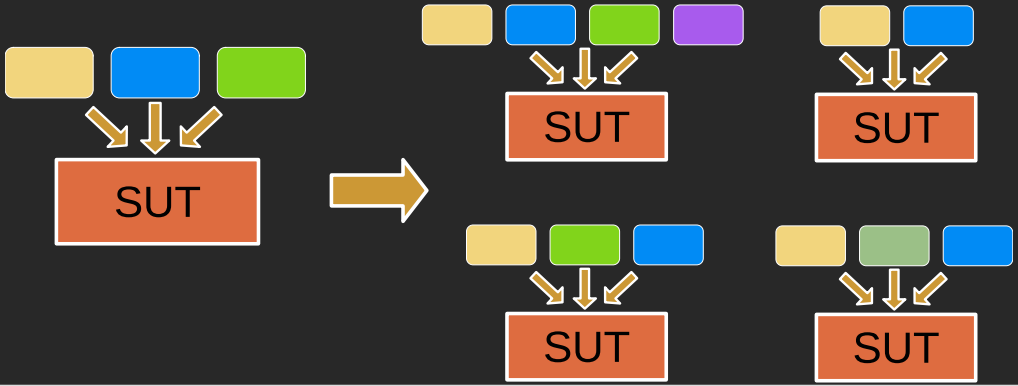
Lesson Learned

Metamorphic relations should be diverse so they exercise different parts of the program.



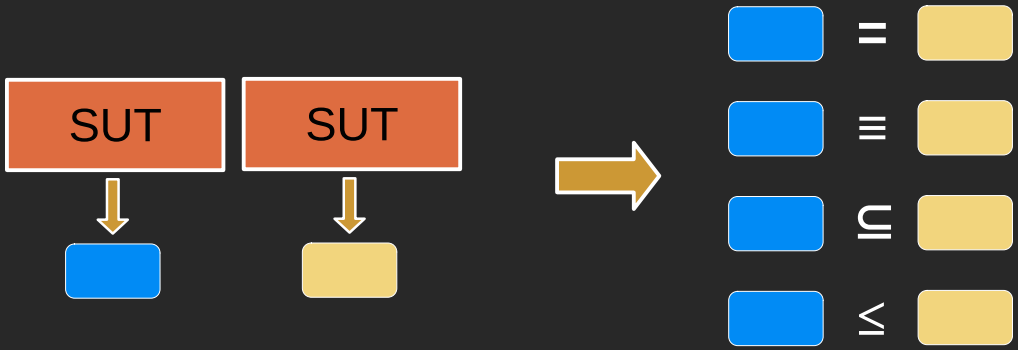
Lesson Learned

Two common approaches for the construction of metamorphic relations: **input-driven** vs. output-driven



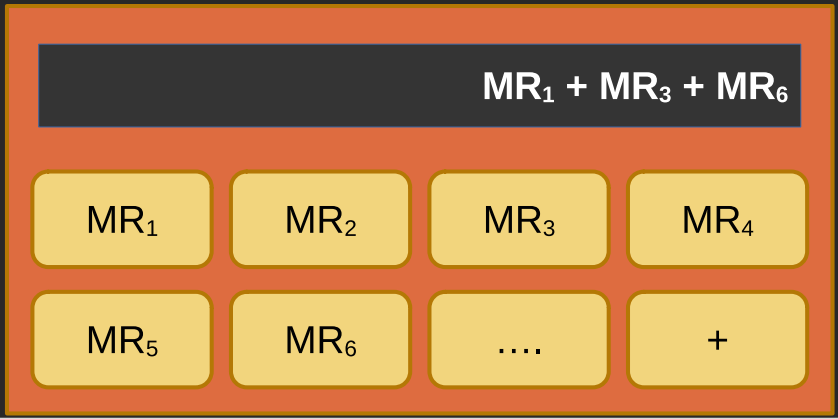
Lesson Learned

Two common approaches for the construction of metamorphic relations: input-driven vs. **output-driven**



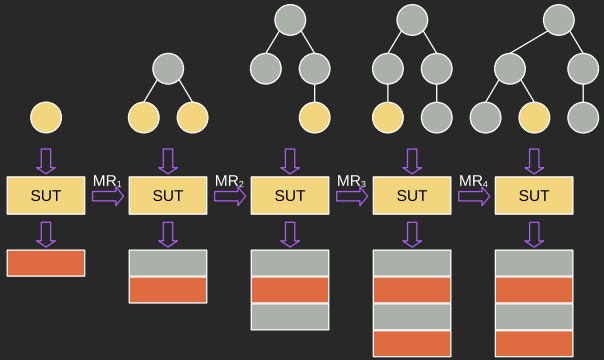
Lesson Learned

Metamorphic relations can be combined



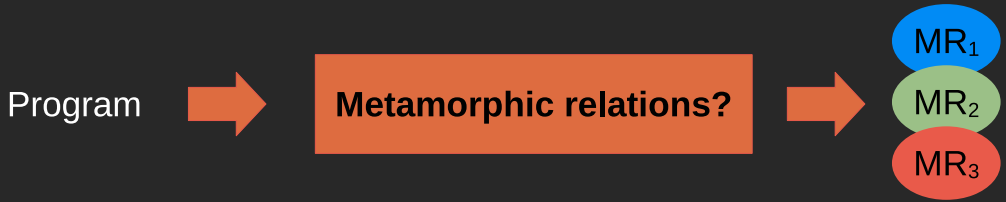
Lesson Learned

Metamorphic relations can be combined



Lesson Learned

The automated discovery of metamorphic relations seems feasible in certain domains



Challenges

- Systematic guidelines for the construction of good metamorphic relations
- Generation of likely metamorphic relations
- Non-functional metamorphic testing
- Provide tools to foster the use of the technique

PEN Testing

CS 3321

Penetration Testing

Idaho State University | Computer Science

Definition from DOI:

> *Penetration testing is a controlled attack simulation that helps identify susceptibility to application, network, and operating system breaches.*

- Also known as **PEN testing** or **ethical hacking**

Penetration Testing

Idaho State University | Computer Science

- PEN testing is about finding vulnerabilities in software systems
- Thinking in the perspective of an attacker or hacker

- Software systems are more connected (and vulnerable) than ever in the age of the internet.
  - Web Apps
  - E-commerce
  - Public APIs
  - Internal Enterprise Applications
  - Etc.

PEN Testing Strategies

Consider different PEN testing strategies when planning a PEN test

- **External PEN testing:**
  - Performing the attack outside the organization's boundary using the internet
- **Internal PEN testing:**
  - Performing the attack from inside the organization's network.
  - This would simulate a disgruntled employee

PEN Testing Strategies

Idaho State University | Computer Science

- **Blind PEN testing:**
  - The testing team performing the attack is given no or little information about the organization.
  - This simulates a real-lief hacking attempt

- **Double Blind PEN testing:**
  - An extended version of a blind PEN test where the organization's IT staff and security team are not aware of the test

Types of PEN Testing

- **Black Box**
  - PEN testers have no knowledge of the target system
- **White Box**
  - PEN testers are provided all information about a target system; source code, operating system details, IP addresses, etc.
- **Grey Box**
  - PEN testers are given some knowledge about a system (e.g., OS details and IP addresses but no source code)

General Approach to PEN Testing

Idaho State University | Computer Science

1. Define the Scope
2. Reconnaissance (passive)
3. Scanning
4. Exploit Vulnerability
5. Report & Cleanup

In the literature, the full approach is often called Vulnerability Assessment and Penetration Testing (VAPT)

Common Vulnerabilities

Idaho State University | Computer Science

- Injection
- Broken Authentication
- Sensitive Data Exposure
- XML External Entities (XXE)
- Security Mis-configuration
- Cross-Site Scripting (XSS)
- Using components with identified vulnerabilities
- Insufficient Logging & Monitoring

Use Open Web Application Security Project (OWASP) for up-to-date top 10 list

Common Tools

Idaho State University | Computer Science

Most of these tools aid in scanning/reconnaissance

- Nmap
- Nessus
- Wireshark
- Metasploit
- The harvester
- Zed Attack Proxy (ZAP)
- Browser Exploitation Framework (Beef)
- SQLMAP

Fuzzing

CS 3321

Fuzzing (or Fuzz Testing)

Idaho State University | Computer Science

Definition:

Fuzzing or fuzz testing is an automated software testing technique that involves providing invalid, unexpected, or random data as inputs to a computer program

- Developed by Barton Miller at the University of Wisconsin in 1989

Fuzzing Process (automated)

- Enter random and/or unexpected inputs
- If the program hangs or crashes, the test failed

Fuzzing

- Good for finding unknown vulnerabilities
- Black box technique
- Simple
  - The criteria for passing the test is if the program didn't crash or hang
- Easily automated

For Next Time

- Review the Reading
- Review this Lecture
- Come to Class

Are there any questions?