

M7 - Mutation Analysis

Guillermo Polito

guillermo.polito@inria.fr
@guillep



inria



**Université
de Lille**

Goals

- Introduce Mutation Analysis as a technique to assess test quality
- The competent developer hypothesis
- Computing mutation scores

What is a good test?

“A good test is a test that catches bugs”

- me

How do we know if a test catches bugs?

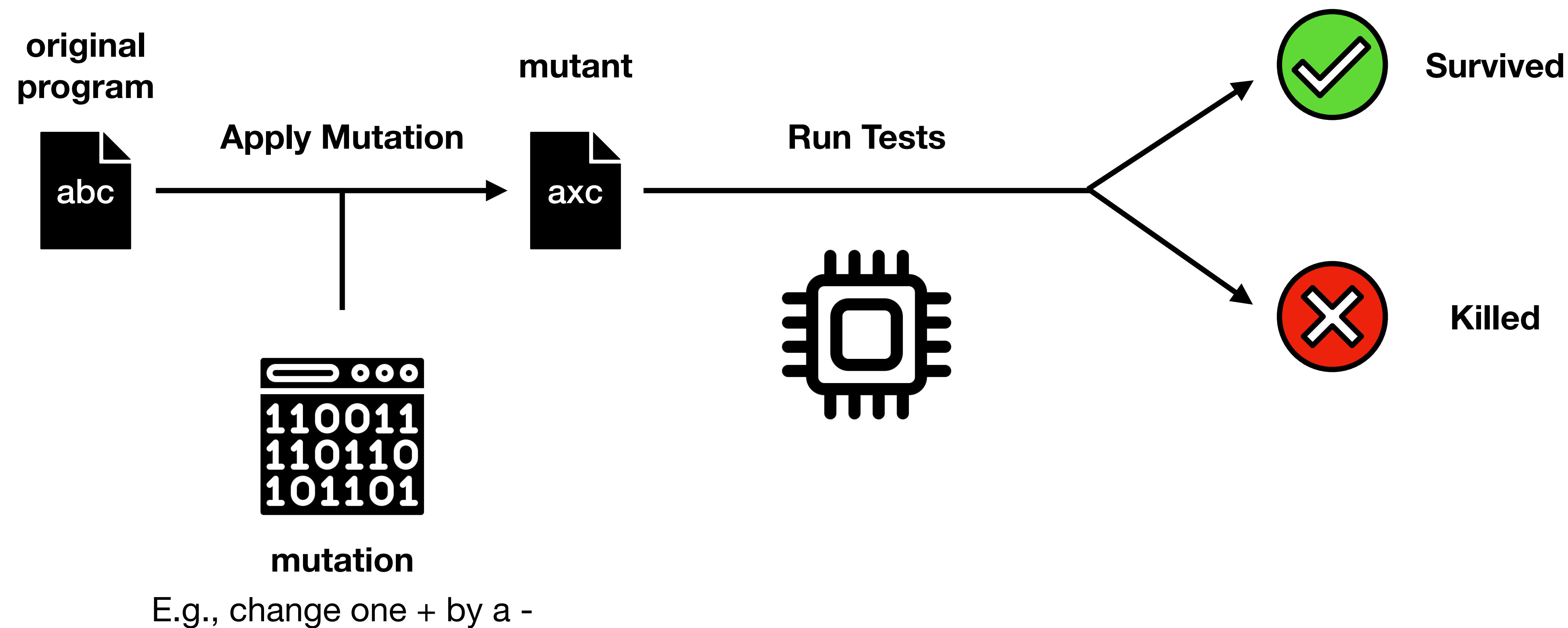
How do we know if a test catches bugs?

- Introduce artificial bugs
- See if the test suite detects them
- What kind of bugs?

Competent developer hypothesis

- Developers are capable people
- Mistakes/bugs are small details easily overseen. E.g.,
 - Missing +/- 1 in a loop
 - An inverted conditional
 - Signed/unsigned

Mutation Analysis



Mutation Score

- Run each mutation independently

- Score:

$$\frac{\text{\#Killed}}{\text{\#Mutants}}$$

The insight

- Survived mutants were either
 1. not covered
 2. had effects not asserted
 3. or were semantically **equivalent**. E.g. $A+B=A-B$ if $B=0$

1) and 2) call to ***improve our tests***

3) ***bias*** our results

Problems of Mutation Analysis

$$\text{Runtime} = \text{Time}(\text{tests}) * \# \text{Mutants}$$

Possible Extensions and Next Steps

- Select a subset of mutants to run
- Select a subset of tests
- High-order mutants
- Language/Application-specific mutations
- Use mutation testing to evaluate your fuzzer!

Takeaways

- We can automatically assess test case quality
- Mutations introduce artificial bugs and mimic developer issues
- Expensive to run
- Trivial and Equivalent mutants bias results and should be manually inspected

Material

- Mutation Testing Advances: an Analysis and Survey. Advances in Computers Journal. Papadakis et al