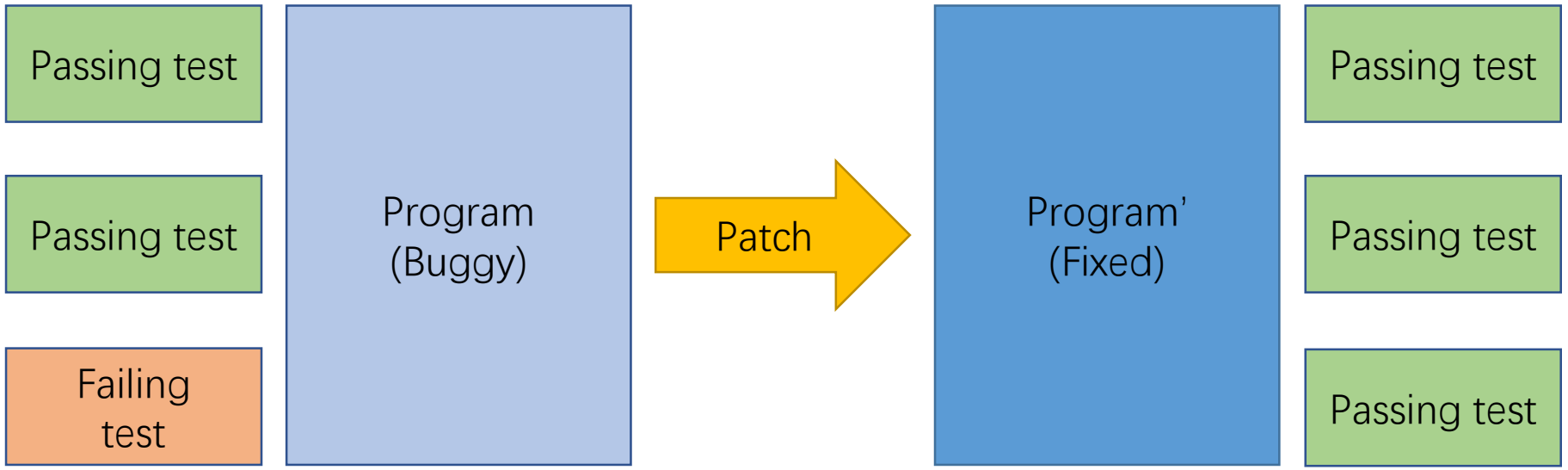
Identifying Patch Correctness in Test-based Program Repair

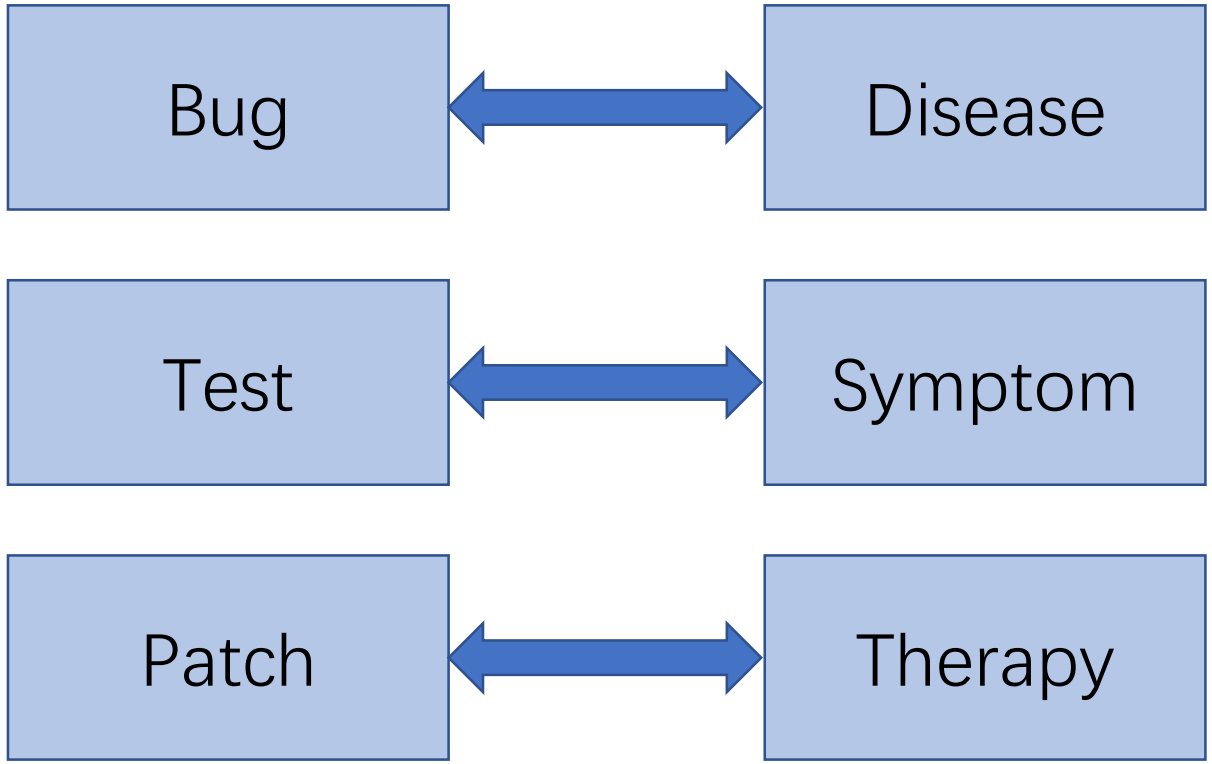
Yingfei Xiong, Xinyuan Liu, **Muhan Zeng**, Lu Zhang, Gang Huang

Peking University

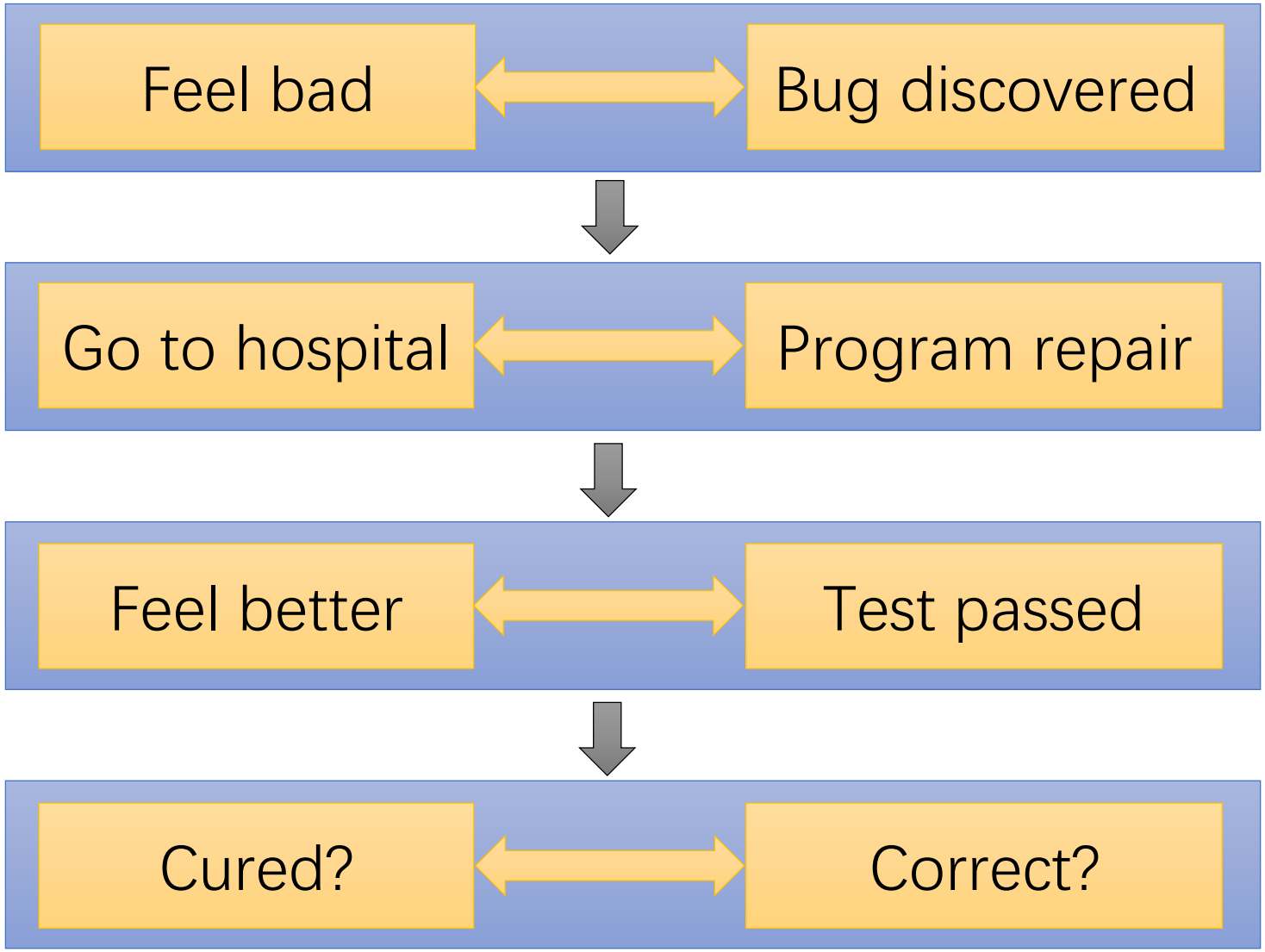
Test-based Program Repair



Program repair: The cure



Workflow : Program repair & hospital

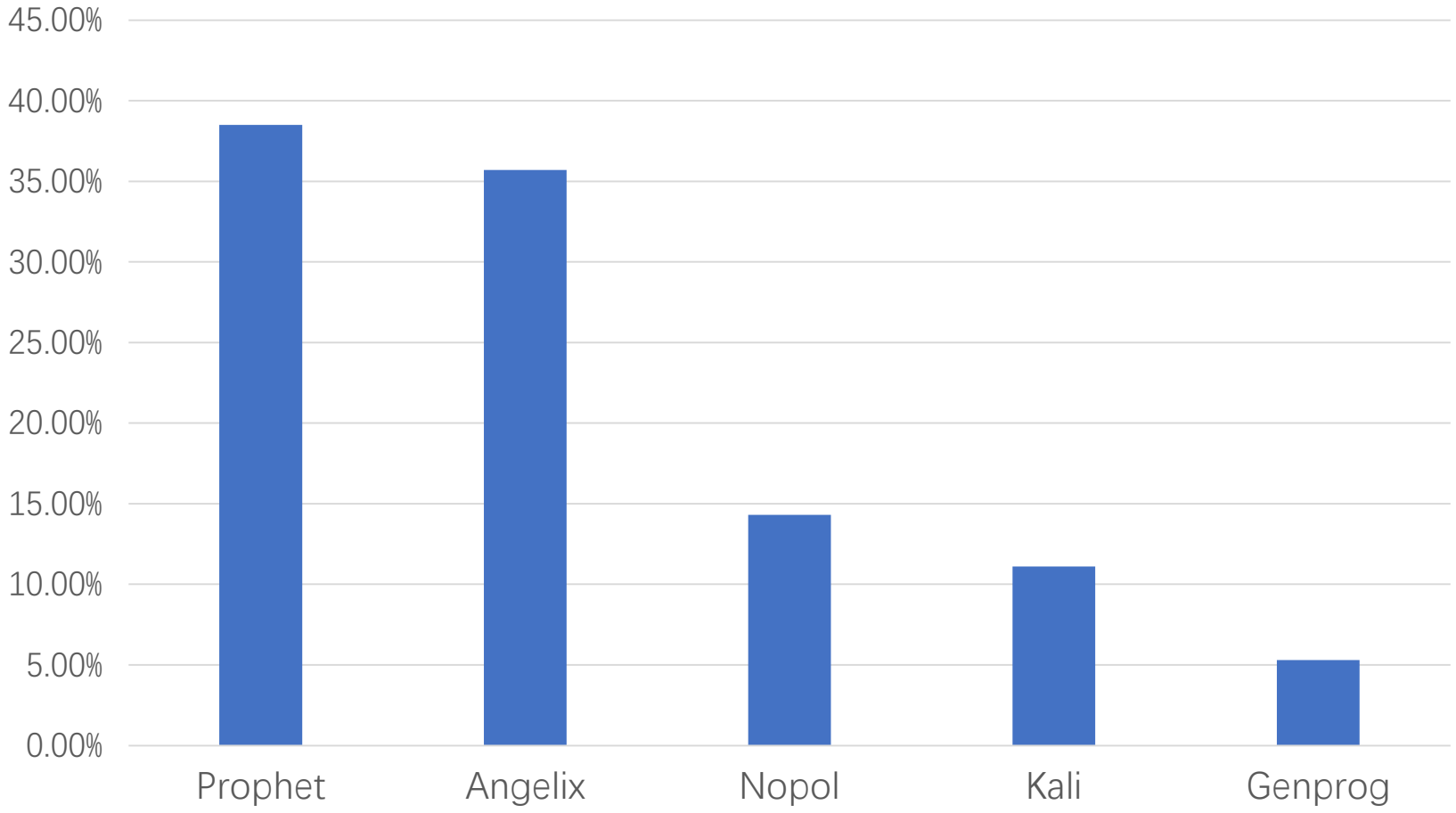


Symptoms are gone == cured?

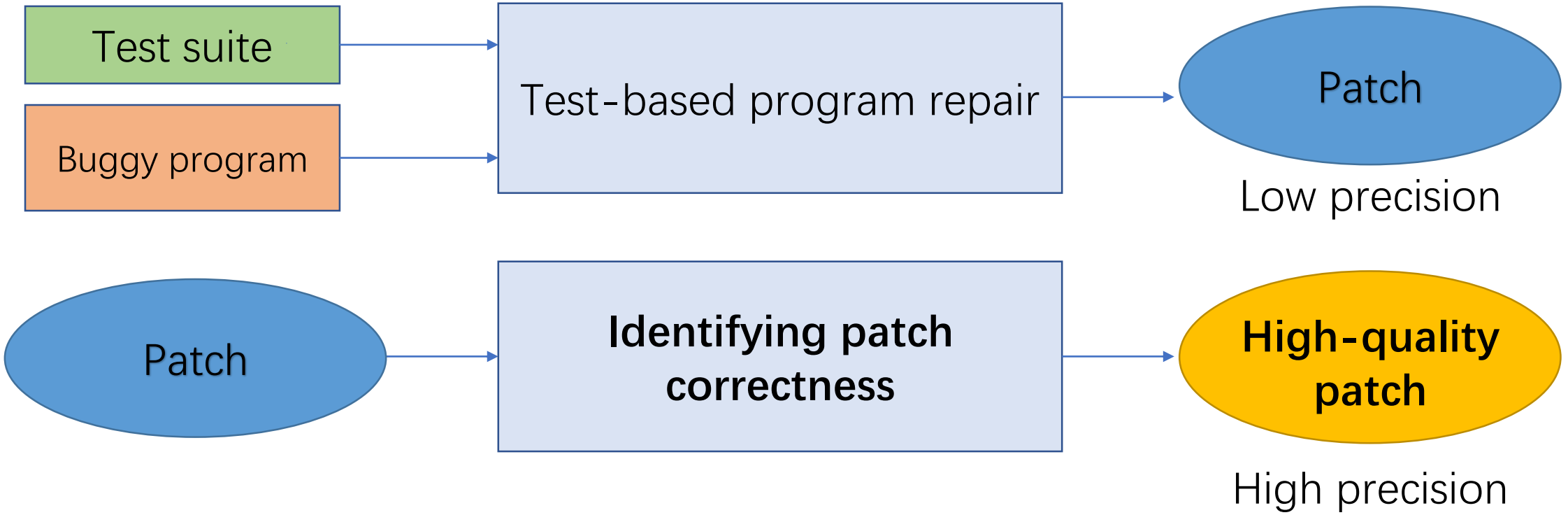
Therapy	Plausible patches
• Makes you free of pain • Disease may still be there	• Pass all the tests • Can still be incorrect (overfit)

Tools: Hospitals

- Precision: Correct / (Correct + Incorrect)



Approach overview



Plausible patches: Wrong cure

```
  public void draw(...) {
+   if (true) return ;
    ...
```

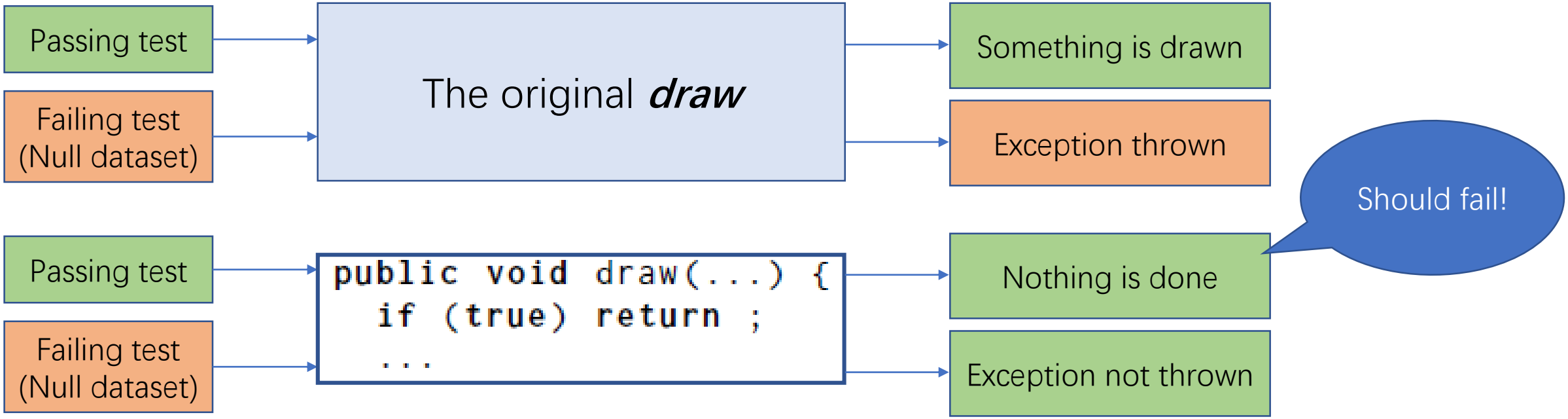
An incorrect patch produced by jKali[1]

```
public void testDrawWithNullDataset() { ...
  JFreeChart chart = ChartFactory.
    createPieChart3D("Test", null,...);
  try {...
    chart.draw(...);
    success = true; }
  catch (Exception e) {
    success = false; }
  assertTrue(success);
}
```

A test checking for null dataset.
Test oracle: function ***draw*** returns normally (without exception)

[1]Martinez M, Durieux T, Sommerard R, et al. Automatic repair of real bugs in java: A large-scale experiment on the defects4j dataset[J]. Empirical Software Engineering, 2017, 22(4): 1936-1964.

Bad therapy: What's wrong here?



Wrong cure

- All symptoms are cured but in a bad way
  - Problems are solved but not in a satisfying way

- “My leg is wounded”
- “Cut it off so you no longer have a hurt leg”

Directly return No exception

- Weak test oracle

Weak test oracle

- No exception ≠ correct patch

```
public void testDrawWithNullDataset() { ...
  JFreeChart chart = ChartFactory.
    createPieChart3D("Test", null,...);
  try {...
    chart.draw(...);
    success = true; }
  catch (Exception e) {
    success = false; }
  assertTrue(success);
}
```

Weak test oracle

Plausible patches : Incomplete cure

```
+ if(repeat)
  for (int i = 0; i < searchList.length; i++) {
    int greater = replacementList[i].length()
      - searchList[i].length();
    if (greater > 0) increase += 3 * greater;
  }  ...
```

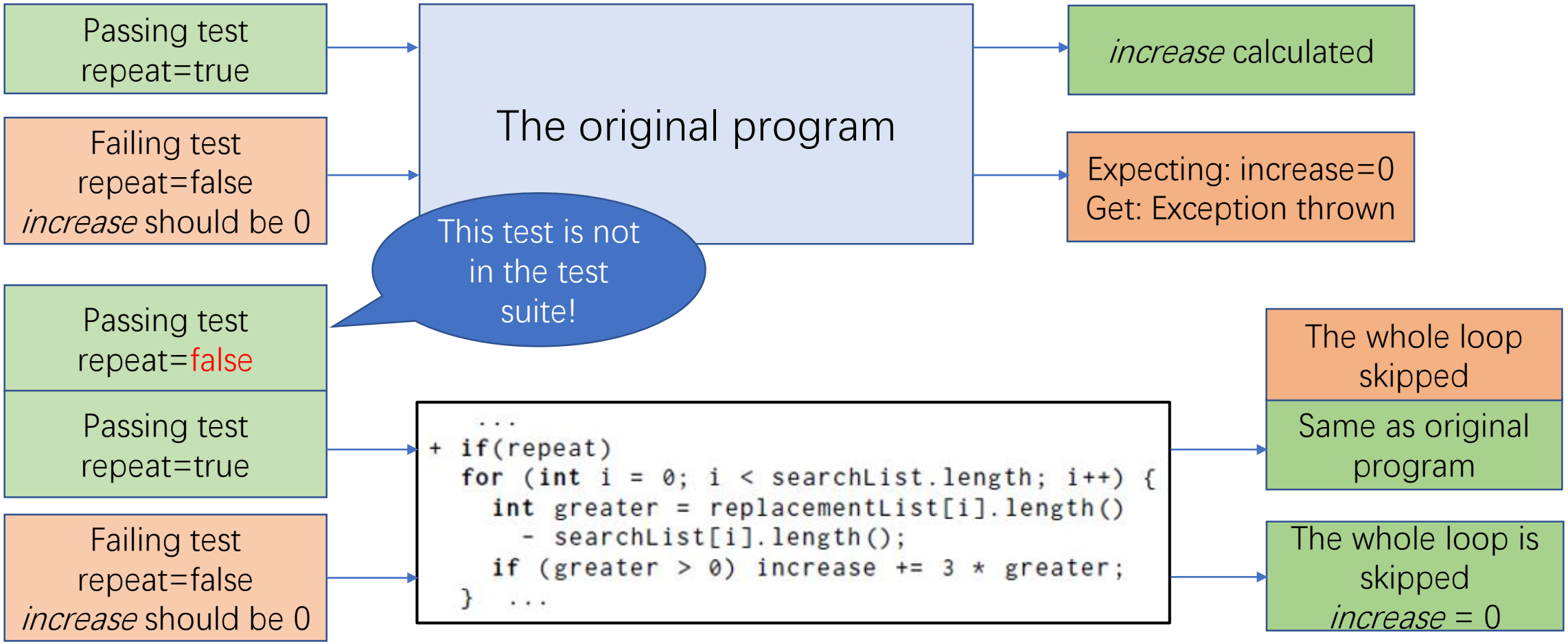
An incorrect patch with wrong condition generated by Nopol[1]

```
   ...
   for (int i = 0; i < searchList.length; i++) {
+    if (searchList[i] == null ||
+        replacementList[i] == null) {
+      continue;
+    }
     int greater = replacementList[i].length()
       - searchList[i].length();
     if (greater > 0) increase += 3 * greater;
   } ...
```

Correct developer patch with correct **null** guard

[1]Xuan J, Martinez M, Demarco F, et al. Nopol: Automatic repair of conditional statement bugs in java programs[J]. IEEE Transactions on Software Engineering, 2017, 43(1): 34-55.

Bad therapy: What's wrong here?



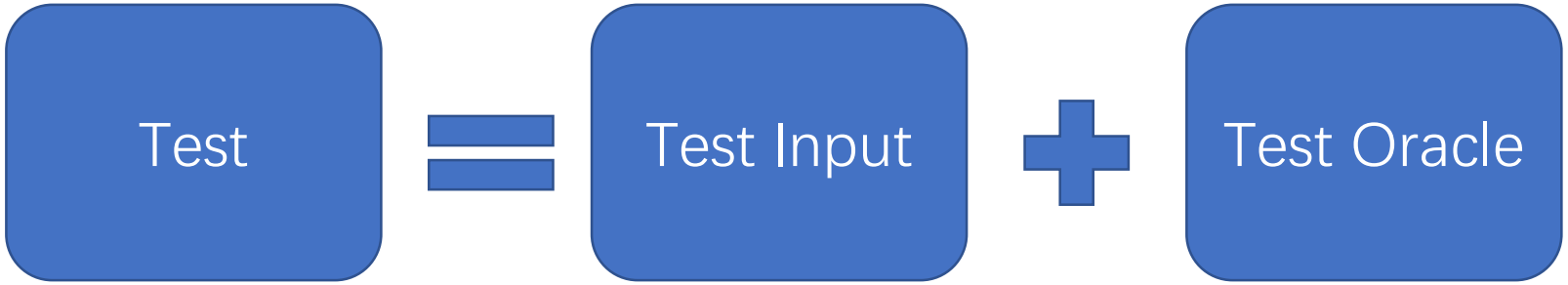
Incomplete cure

- Incomplete cure: concerned symptoms are cured, but some other symptoms are not.
  - Bugs that covered by tests is fixed while others not

- "We cured your left leg and cut off your right leg" Wrong condition
- "So what about my right leg?" Missing test inputs
- "Well, we only care about your left leg" Existing test inputs

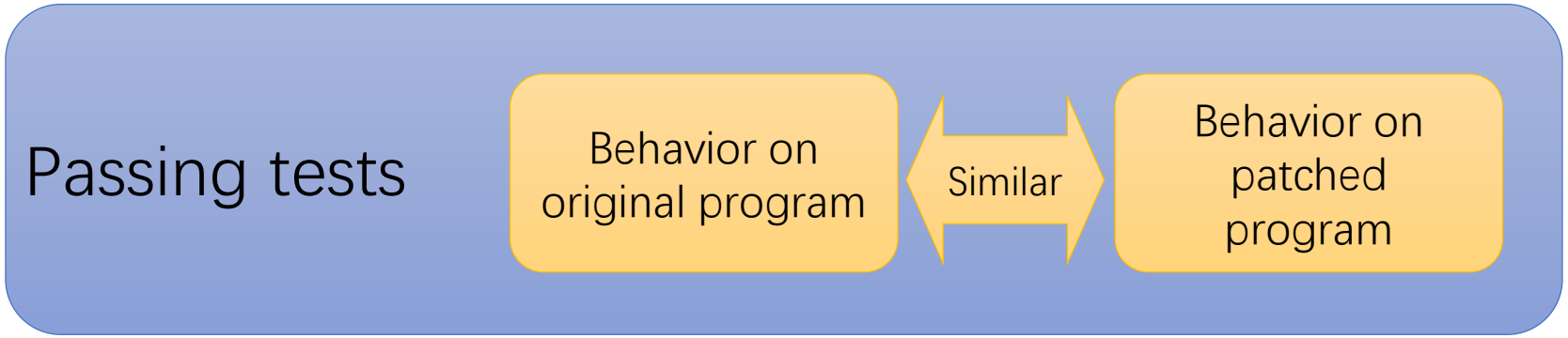
- Weak test input

Test suites and heuristics

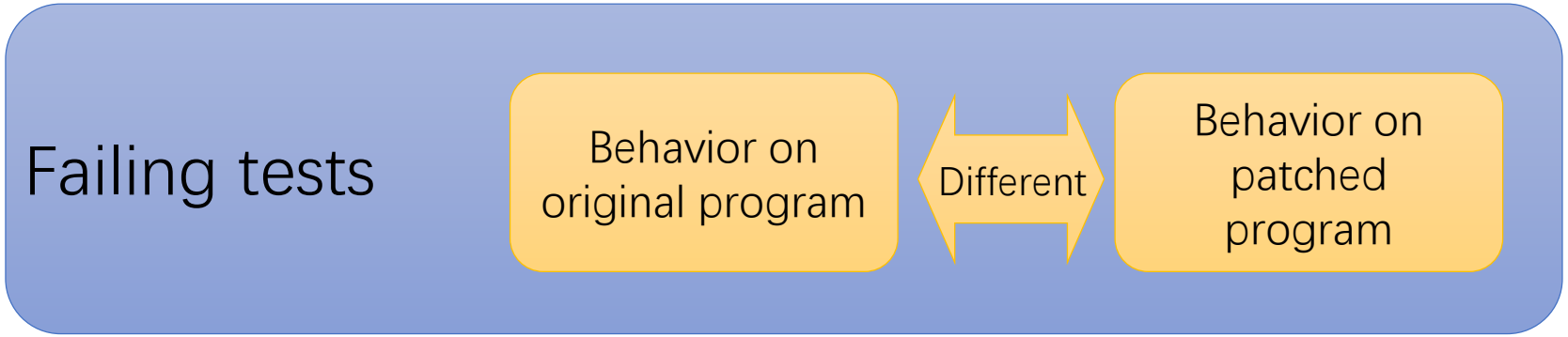


- Test suites are weak on both input and oracle.
- Two heuristics to save weak test suites:
  - PATCH-SIM: compensate for weak test oracle
  - TEST-SIM: compensate for weak test input

PATCH-SIM: heuristic for test oracle

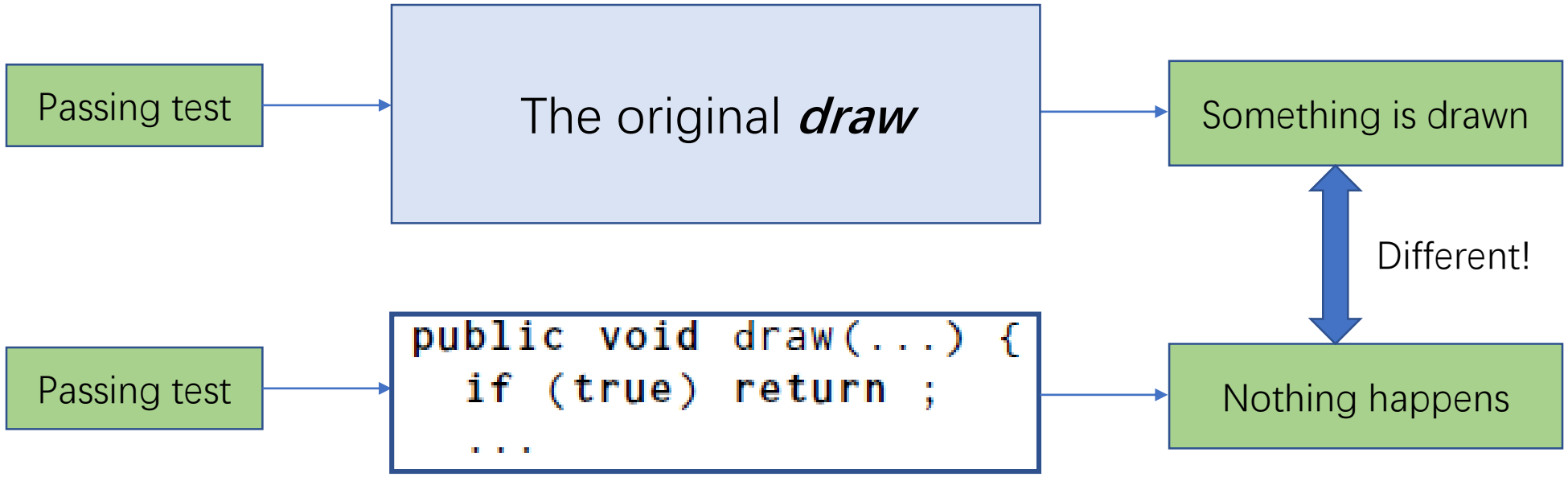


“Well, you should keep my legs (which were good) as good as before”



“What’s more, the wound (which was bad) should be cured”

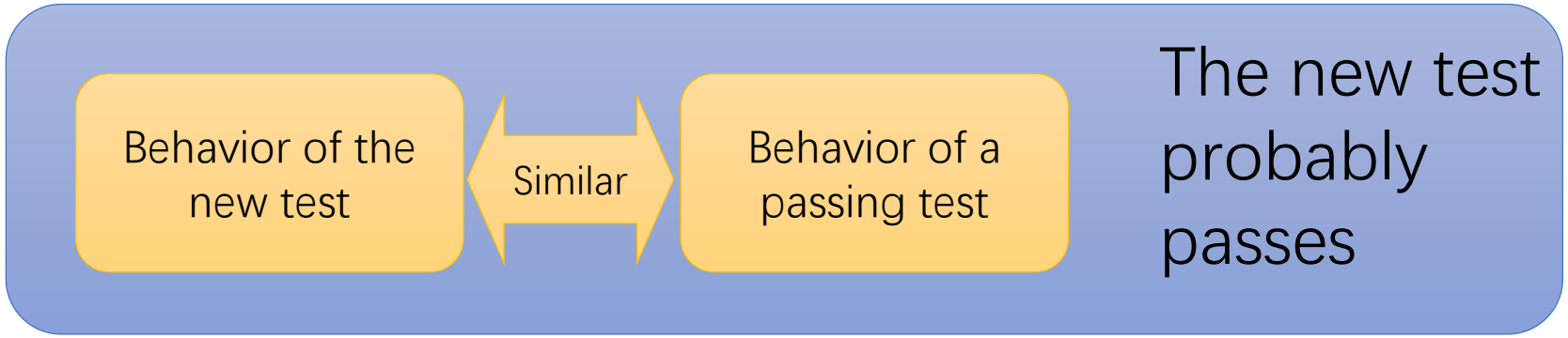
Bad cure identified!



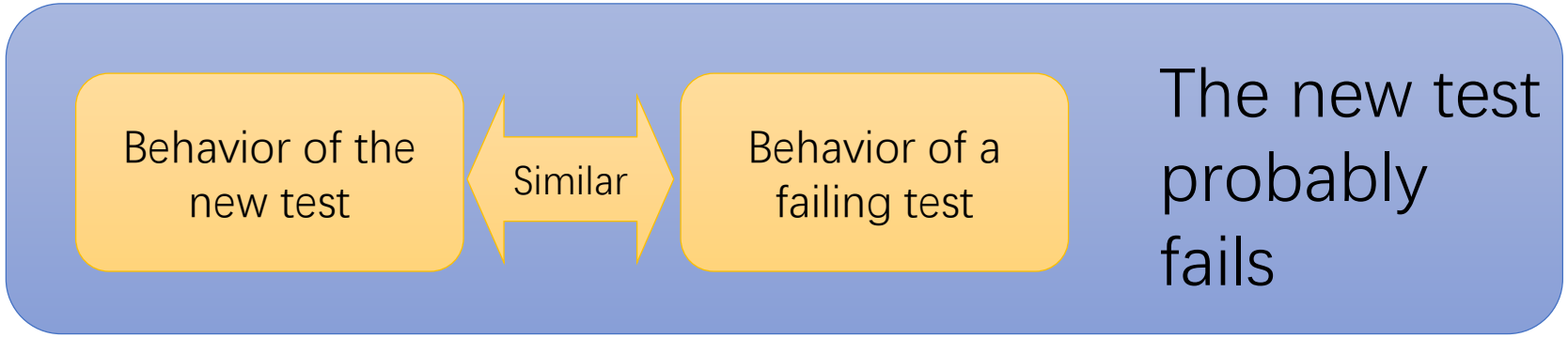
"Well, you should keep my legs (which were good) as good as before"

TEST-SIM: heuristic for test input

- PATCH-SIM on newly generated tests: pass or fail?

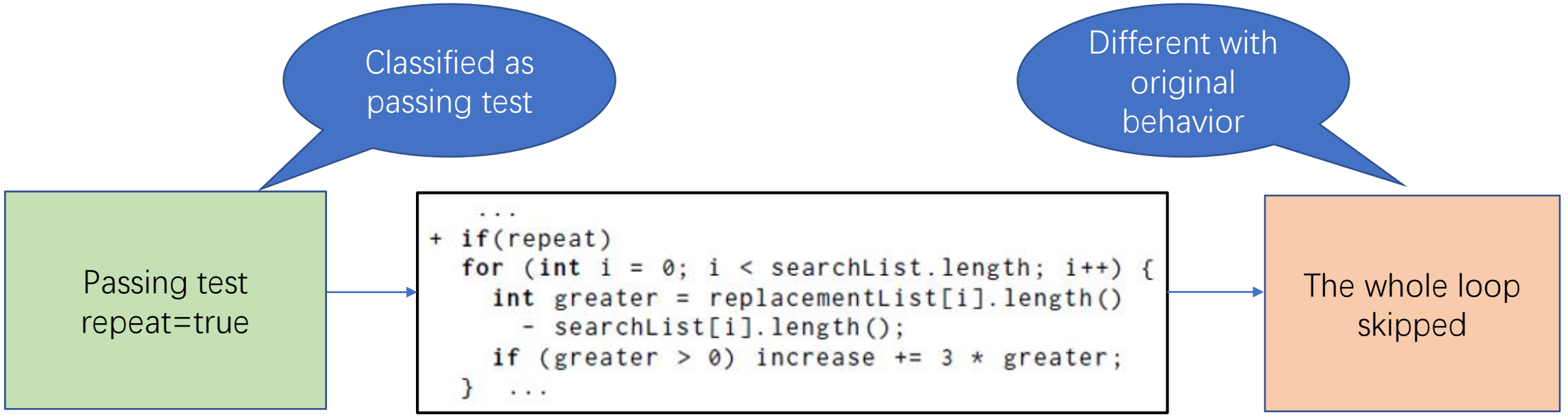


"My left leg is just like my right leg.
My right leg is good, so my left leg is also good"



- Classification result can be used by PATCH-SIM

Bad cure identified!



“Check my left leg, it’s good and I want it as good as before”

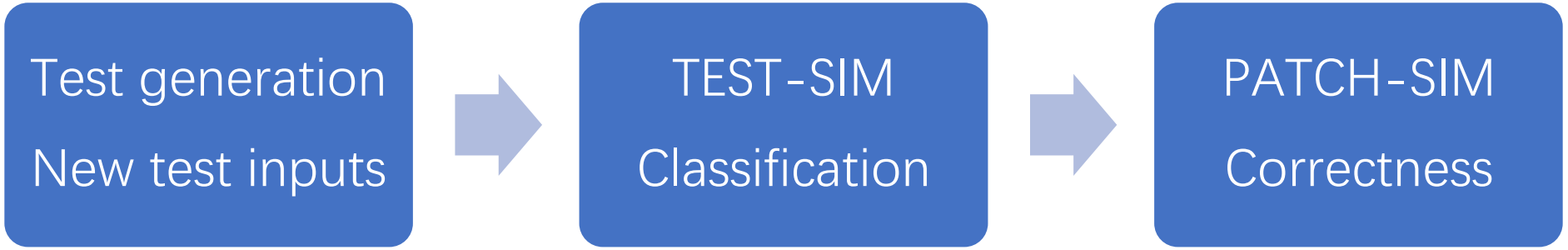
Workflow

- “Check my left leg, it’s good and I want it as good as before”

Test generation

Classification by TEST-SIM

Oracle of PATCH-SIM



Similar? Different?

- Test oracle: output

- Result is not all: the process is also important
- Runtime information: Behavior similarity

Details for ‘Behavior similarity’

- Complete-path spectrum[1]: the sequence of executed statements

```
1  x=0;
2  for(int i=0;i<3;i++)
3      x++;
4  return x;
```

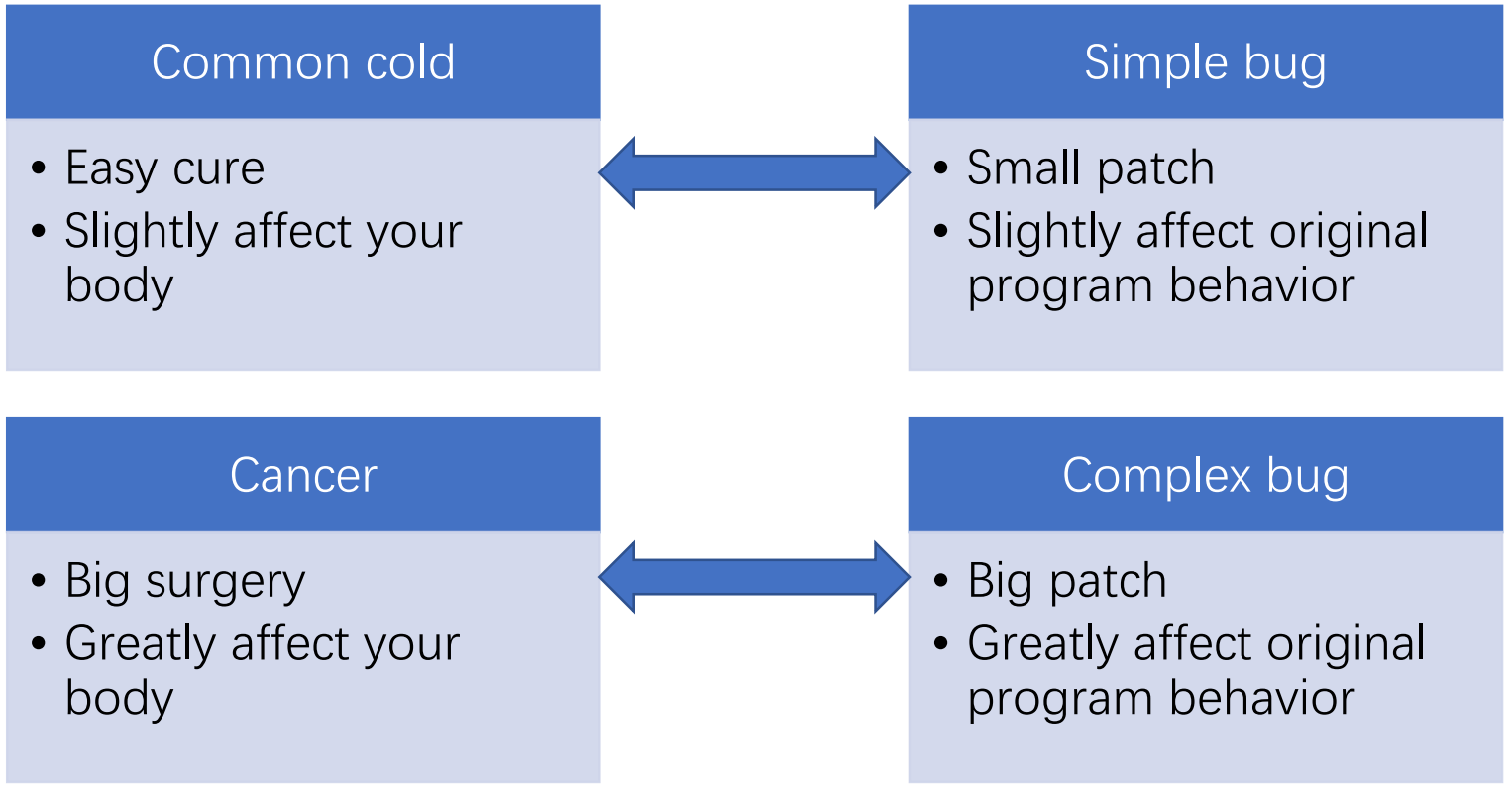
{1,2,3,2,3,2,3,2,4}

- Distance and similarity:

$$distance(a,b) = 1 - \frac{|LCS(a,b)|}{max(|a|,|b|)}$$

[1]Harrold M J, Rothermel G, Wu R, et al. An empirical investigation of program spectra, Acm Sigplan Notices. ACM, 1998, 33(7): 83-90

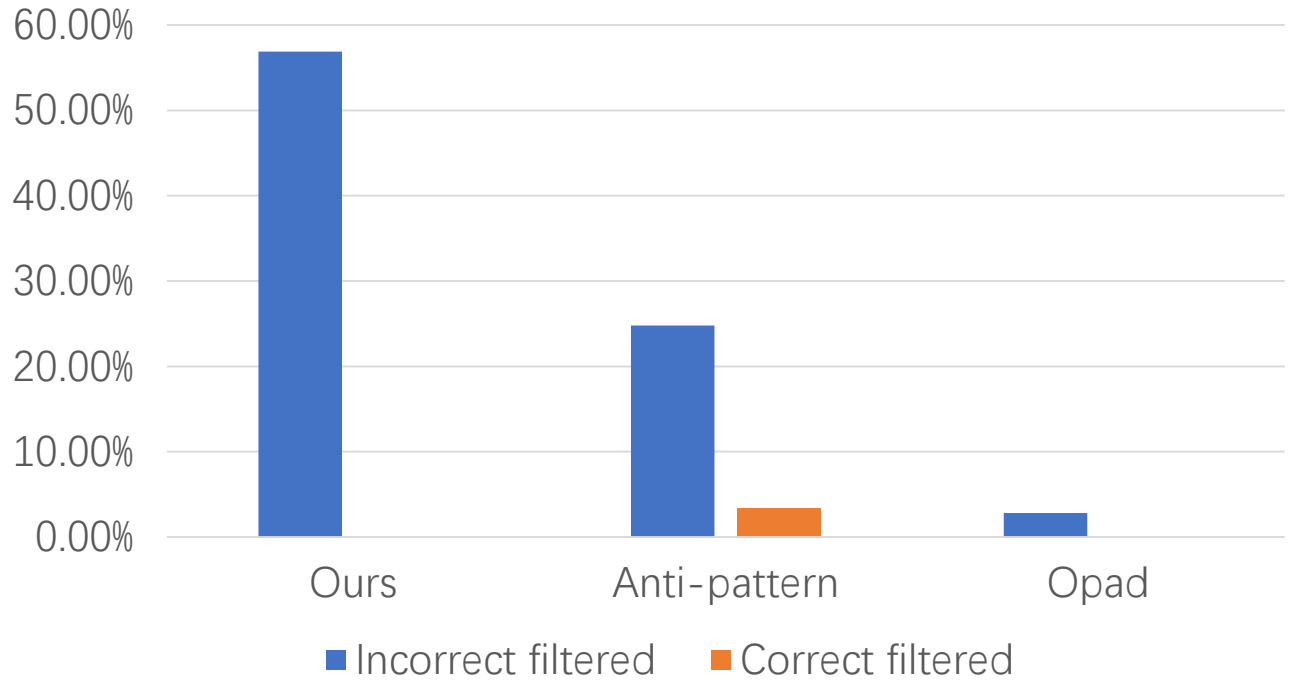
‘Similar’ is relative, not absolute



- Behaviors on passing tests should be more similar

Effectiveness

- Dataset: 139 Patches from jGenProg, Nopol, jKali, ACS and HDRepair
  - Defects4J benchmark
- 56.3% of incorrect patches filtered out without losing any of the correct patches.



Anti-pattern: pre-defined patterns

Opad: patches shouldn't introduce crash or memory safety problem (designed for C)

Summary

- Many program repair tools have low precision
- Patch correctness can be identified based on behavior similarity
  - 2 heuristics: PATCH-SIM and TEST-SIM
- 56.3% incorrect patches filtered, 0 loss on correct patches

Discussion: complicate patches

- Patches from APR are simple (for now).

- Will our approach still be effective in the future?
  - E.g. on more complicate patches

Developer patches

- 194 correct patches from Defects4J benchmark
- 178(91.75%) still classified as correct
- Reason for misclassification:
  - Significant behavior change
  - Calling a different method with the same functionality