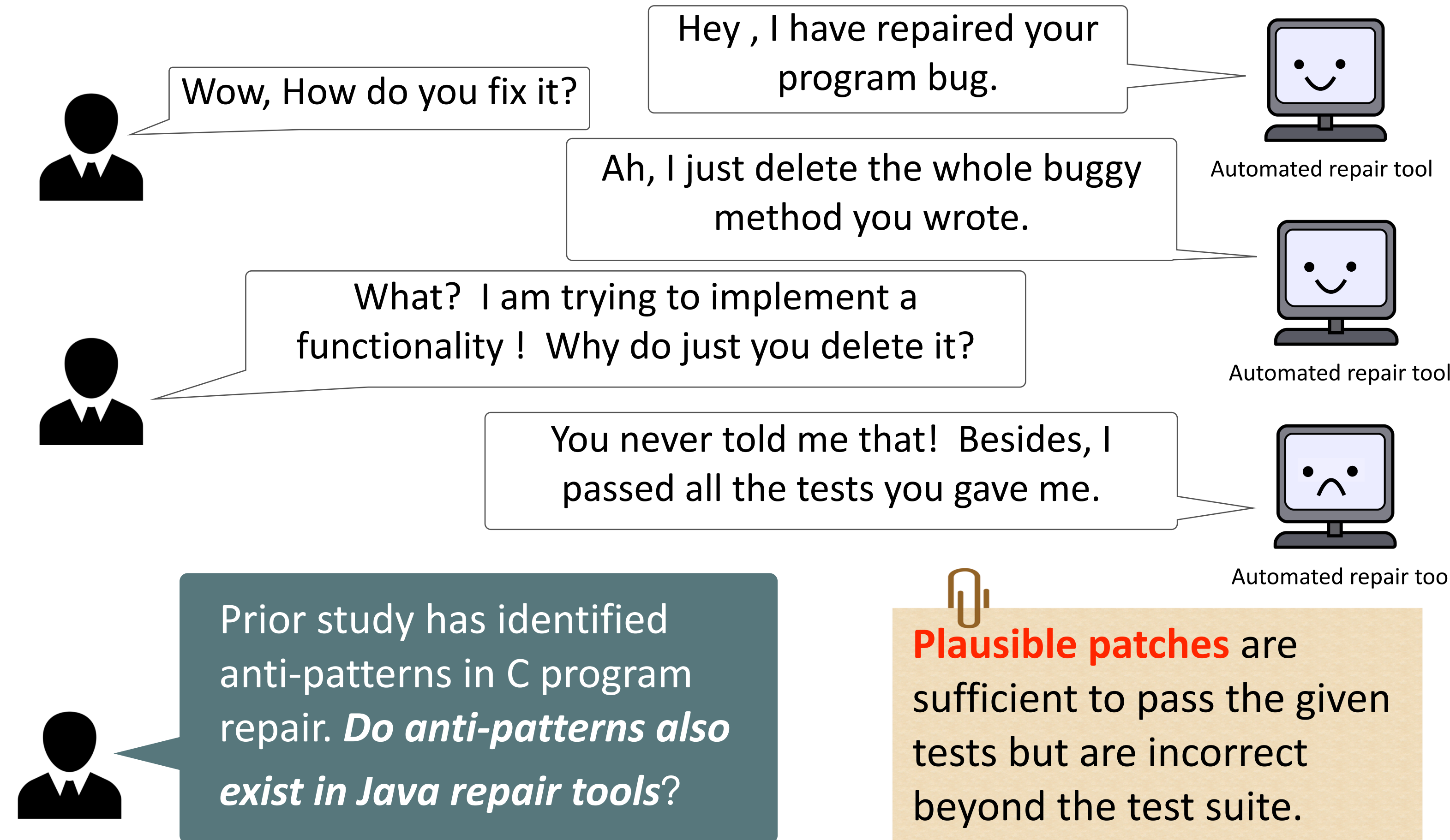


Motivation



- Use test suites to specify the intended behavior of programs.
 - Insufficient test cases cause Incomplete specification.
 - Automated repair tools generate **plausible patches**.
- ↑ solution : specify what not to do.

Anti-pattern — a set of forbidden transformations on programs

Anti-pattern Category

Anti-append Trivial Condition

SimFix for Defects4J Math80

```
- for (int k = 0; k < 4; k += step) {
+ for (int k = 0; k < 0; k += step) {
    final double tmp=work[i+k];
    ...
}
```

If condition is always false

Anti-delete Control Statement

LSRepair for Defects4J Lang51

```
- if (str == "true") {return true; }
- if (str == null) {return false;}
- switch (str.length()) { ... }
+ return !(arg0.startsWith("(") && arg0.endsWith(")"));
```

Delete if and switch statements

Anti-append Early Exit

Observed running behavior of jGenProg2 for Defects4J Math73

```
double yInitial = f.value(initial);
+ return result;
if (...){ ... }
```

Insert early return

Prevalence of Anti-patterns

- Manually analyzed plausible patches generated by SimFix, CapGen, and LSRepair, shown in Table1 below.
- Difference of anti-pattern distribution is due to the **distinct repair strategies** of the Java repair tools

Table 1: Proportion of Anti-patterns

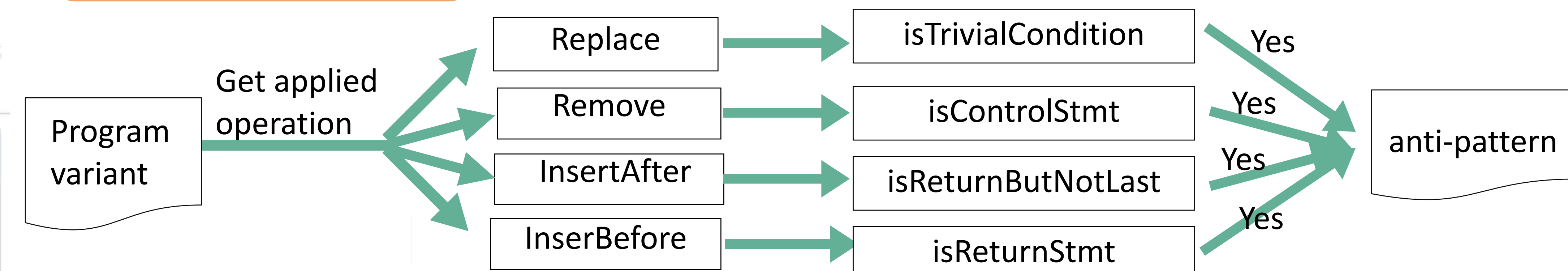
	Total number of plausible patches inspected	Anti-append trivial condition		Anti-delete Control Statement		Anti-append early exit
		Insert contradiction	Insert tautology	Delete if-statement	Delete loop	Insert early return/exit
SimFix	22	8.33%	0	0	0	0
CapGen	219	10.0%	1.83%	0	0	0
LSRepair	484	0	0	64.87%	19.63%	2.48%

*SimFix has only 22 plausible patches available on GitHub)

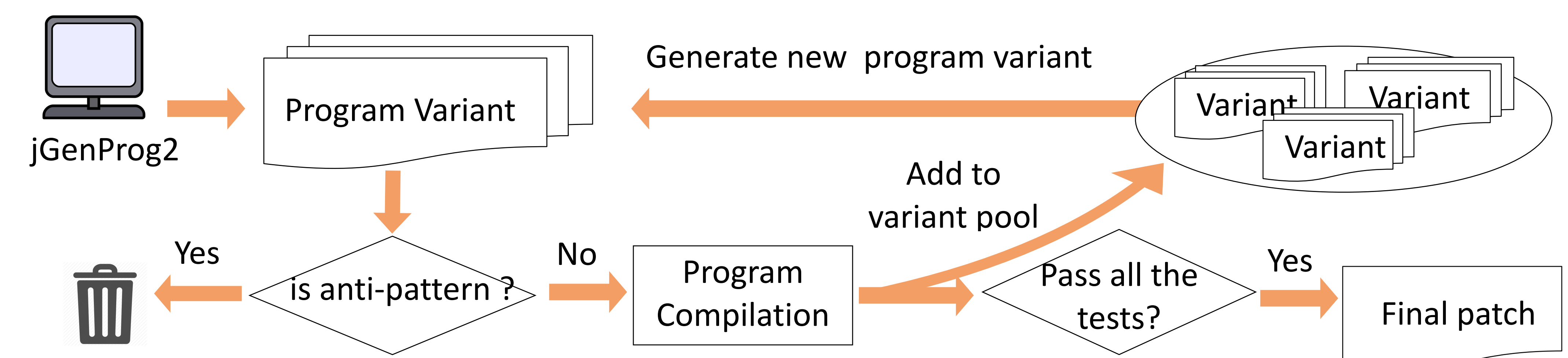
Anti-pattern exists despite advancement of technique in Java repair tools.

Integrating Anti-patterns

Checking for anti-pattern



Anti-pattern in evolution



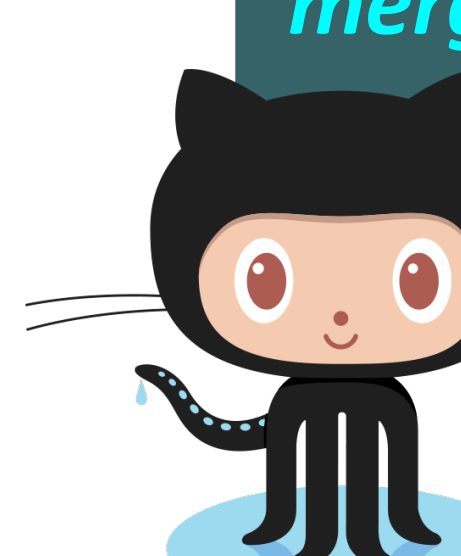
Integrated in **jGenProg2(Astor)** to check if anti-patterns generalize across Java repair tools.

[\[Feature\] Add anti-pattern to improv...](#) #245
WuYff opened this issue on 16 Sep · 3 comments

[martinezmarias](#) commented on 16 Sep

Dear @WuYff
Thanks a lot for sharing the Java implementation of Anti-patterns with us and with the community. We appreciate that.
For curiosity, are the results from your experiment (e.g. Table you put in this issue) published somewhere?
Thanks
Matias

We opened a **pull request** in Astor and the implementation has been **merged** by the developers. [Click to see!](#)



GitHub

Experiment

- Focus on the 29 Defect4J bugs where original jGenProg2 generates patches.
- Run jGenProg2 on each bug up to 20 times with different seeds.
- Recorded the first run jGenProg2 produced any patches.

Results

- Successfully generated patches for 14 bugs with original jGenProg2
- For the 14 bugs:
 - The average repair time is reduced by **22.6 %**.
 - The number of generated plausible patches is reduced from **67** to **29** in total.

Potential New Anti-patterns

Control loop condition by irrelevant variable

```
for (int i = 0; i < j; i += 4) {
- for (int k = 0; k < 4; k +=
+ for (int k = 0; n < 4; k +=
```