Are Mutants a Valid Substitute for Real Faults in Software Testing?

René Just[*], Darioush Jalali[*], Laura Inozemtseva[†], Michael D. Ernst[*], Reid Holmes[†], Gordon Fraser[‡]

[*]University of Washington
[†]University of Waterloo
[‡]University of Sheffield

November 20, 2014

How good is my test suite?

A good test suite detects real faults

Test quality metric is necessary in many areas:

- Test generation, minimization, prioritization, ...

How good is my test suite?

A good test suite detects real faults

Test quality metric is necessary in many areas:

- Test generation, minimization, prioritization, ...

Problem: Set of real faults is unknowable

Solution: Use a **proxy metric** for test quality

- Code coverage ratio
- **Mutant detection rate**

How good is my test suite?

A good test suite detects real faults

Test quality metric is necessary in many areas:

- Test generation, minimization, prioritization, ...

Problem: Set of real faults is unknowable

Solution: Use a **proxy metric** for test quality

- Code coverage ratio
- **Mutant detection rate**

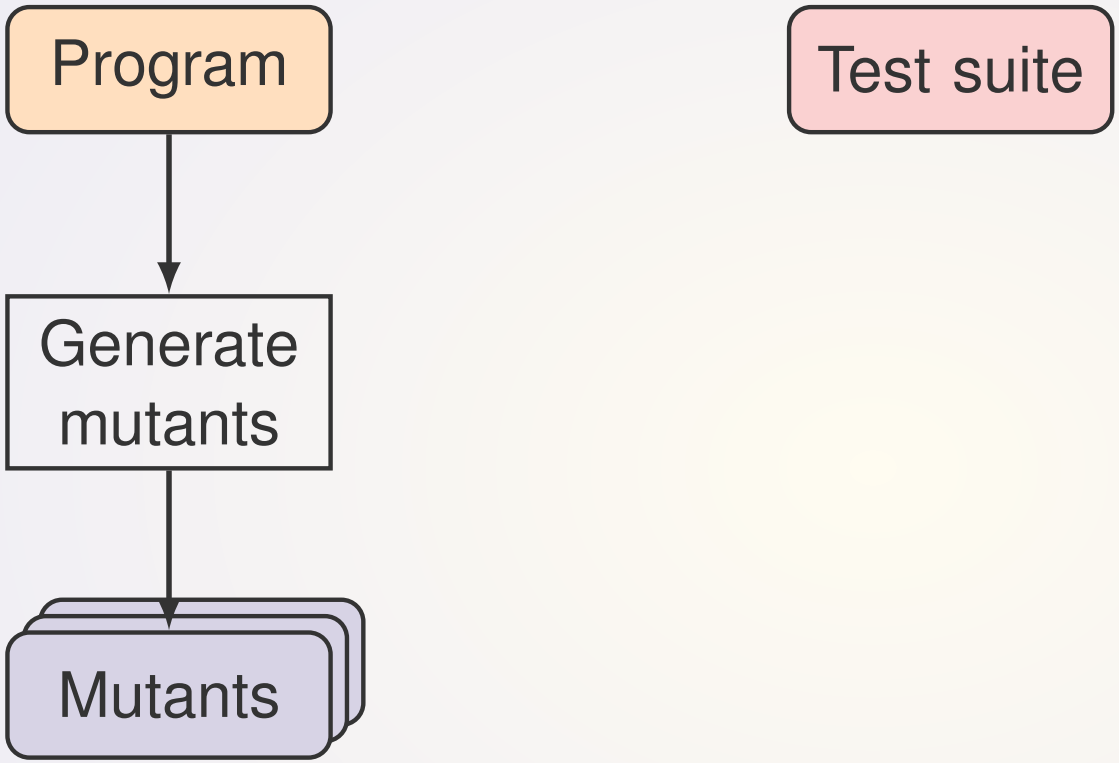
Mutant detection rate $\approx$ Real fault detection rate?

Mutation analysis: Overview

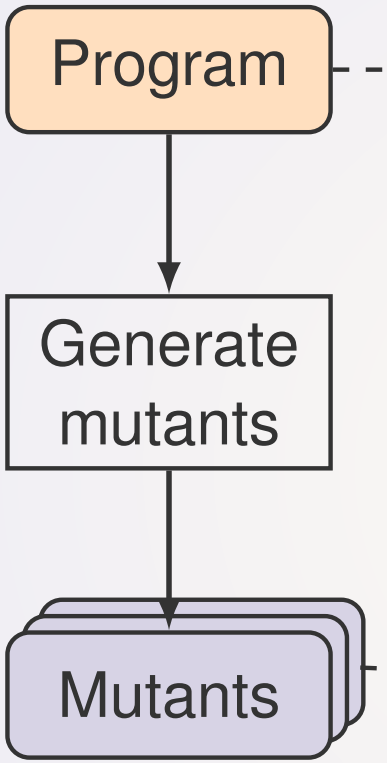
Program

Test suite

Mutation analysis: Overview



Mutation analysis: Overview

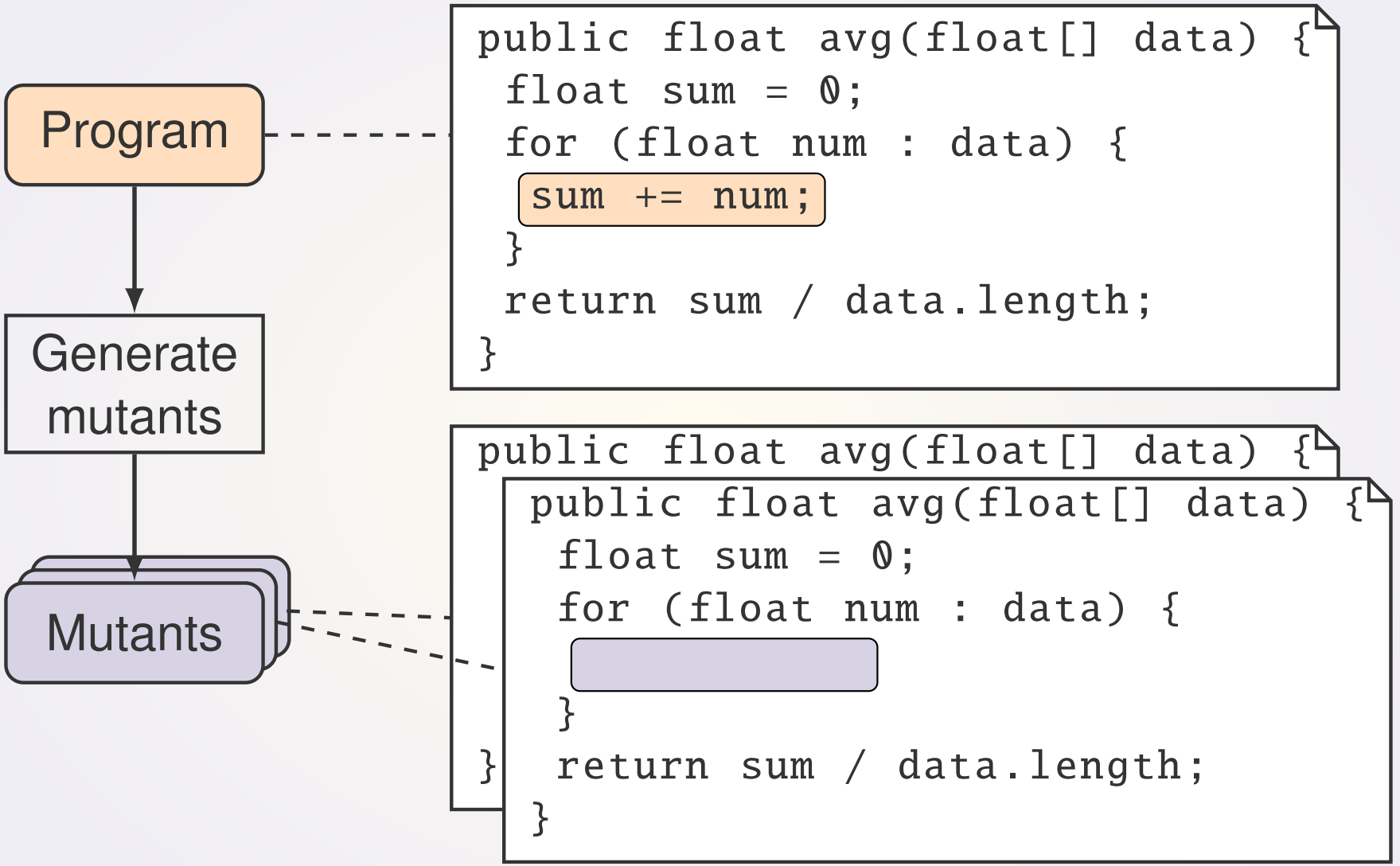



```
public float avg(float[] data) {
 float sum = 0;
 for (float num : data) {
  sum += num;
 }
 return sum / data.length;
}
```

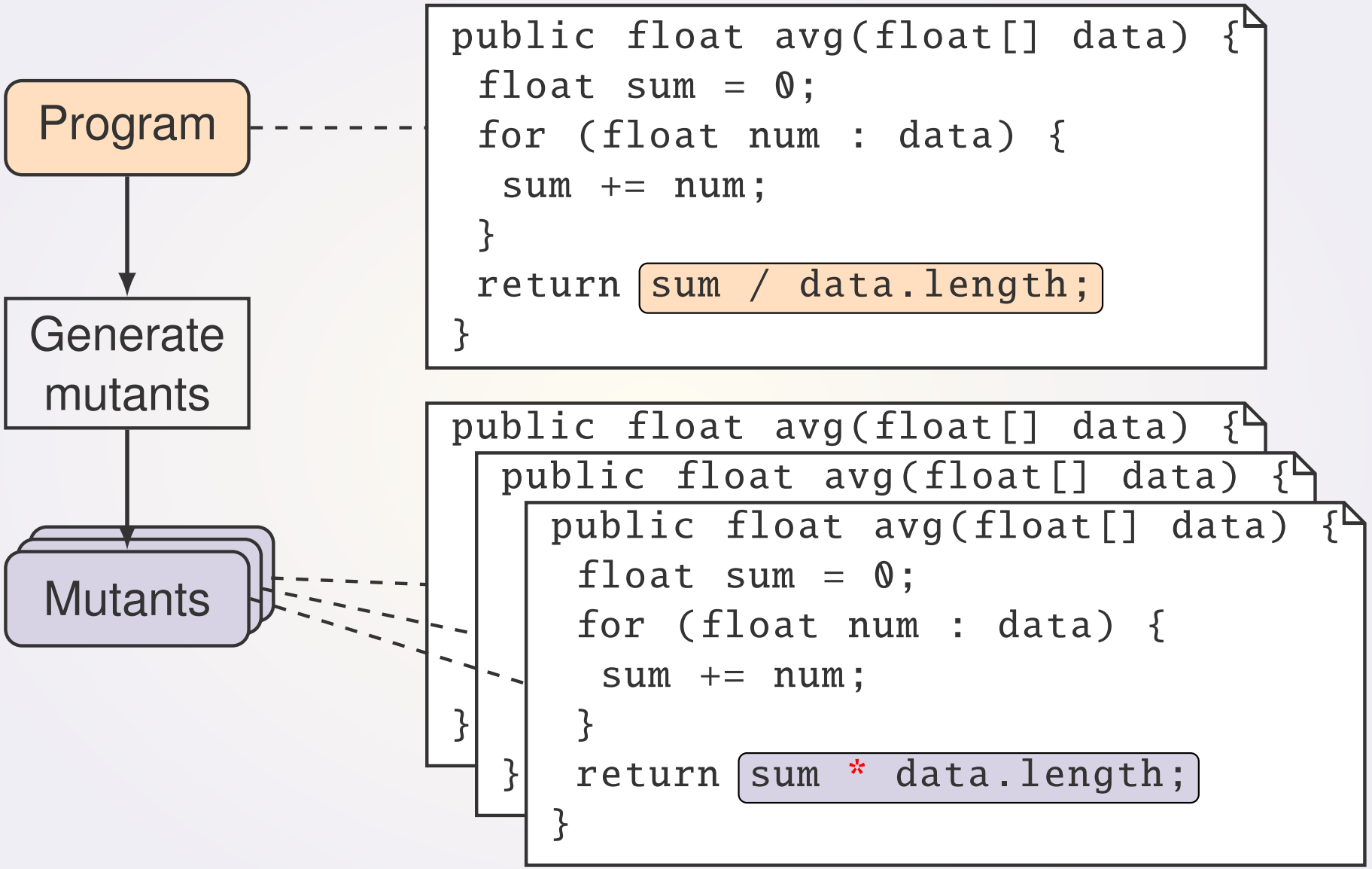
```
public float avg(float[] data) {
 float sum = 1;
 for (float num : data) {
  sum += num;
 }
 return sum / data.length;
}
```

Each mutant contains one small syntactic change

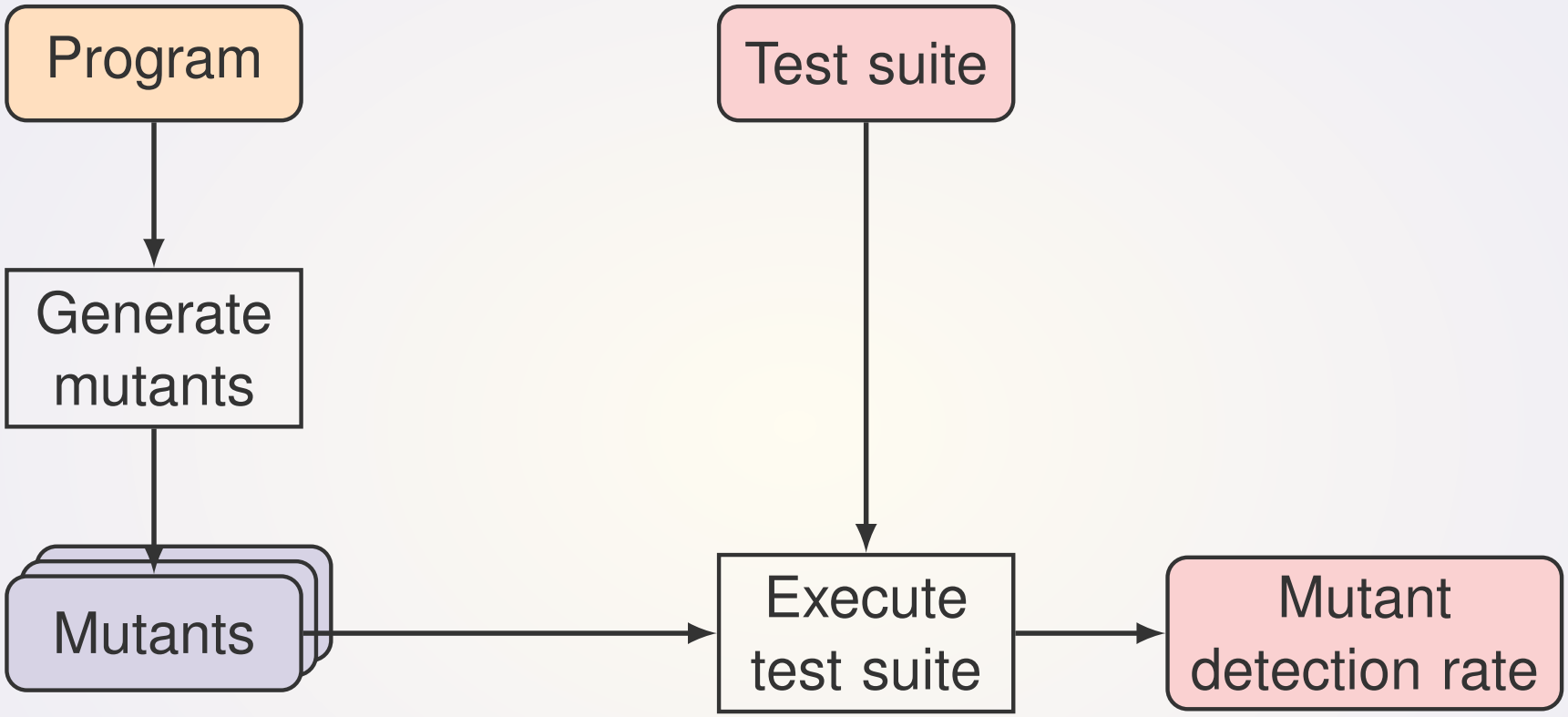
Mutation analysis: Overview



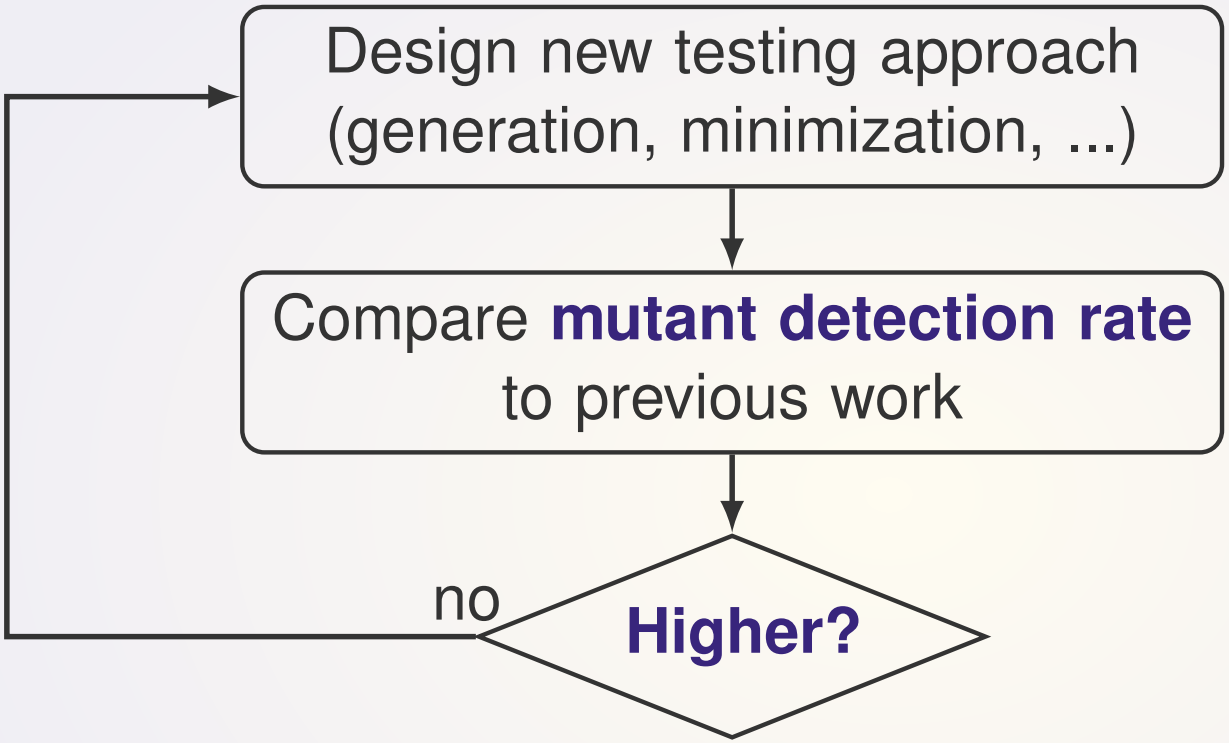
Mutation analysis: Overview



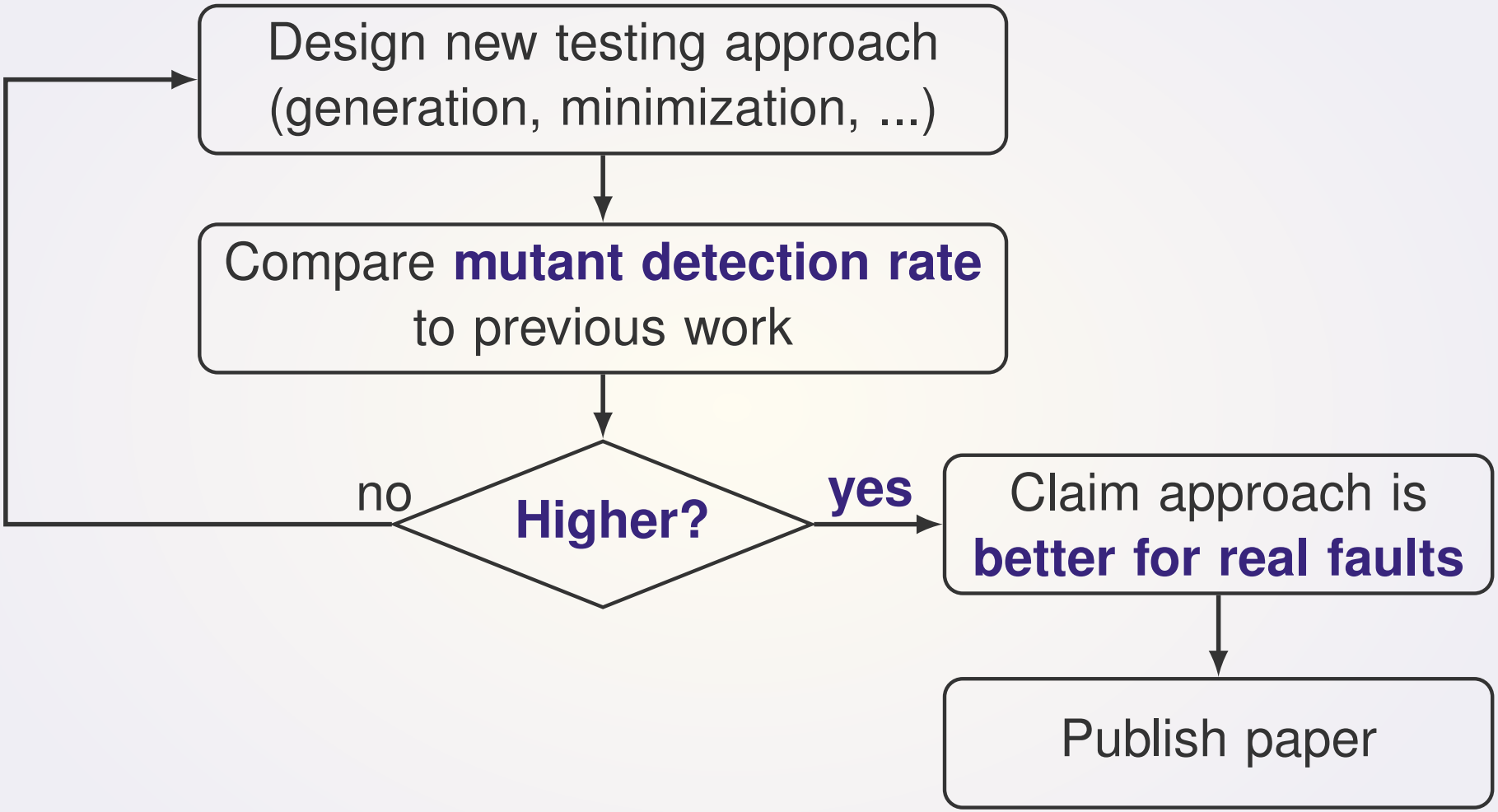
Mutation analysis: Overview



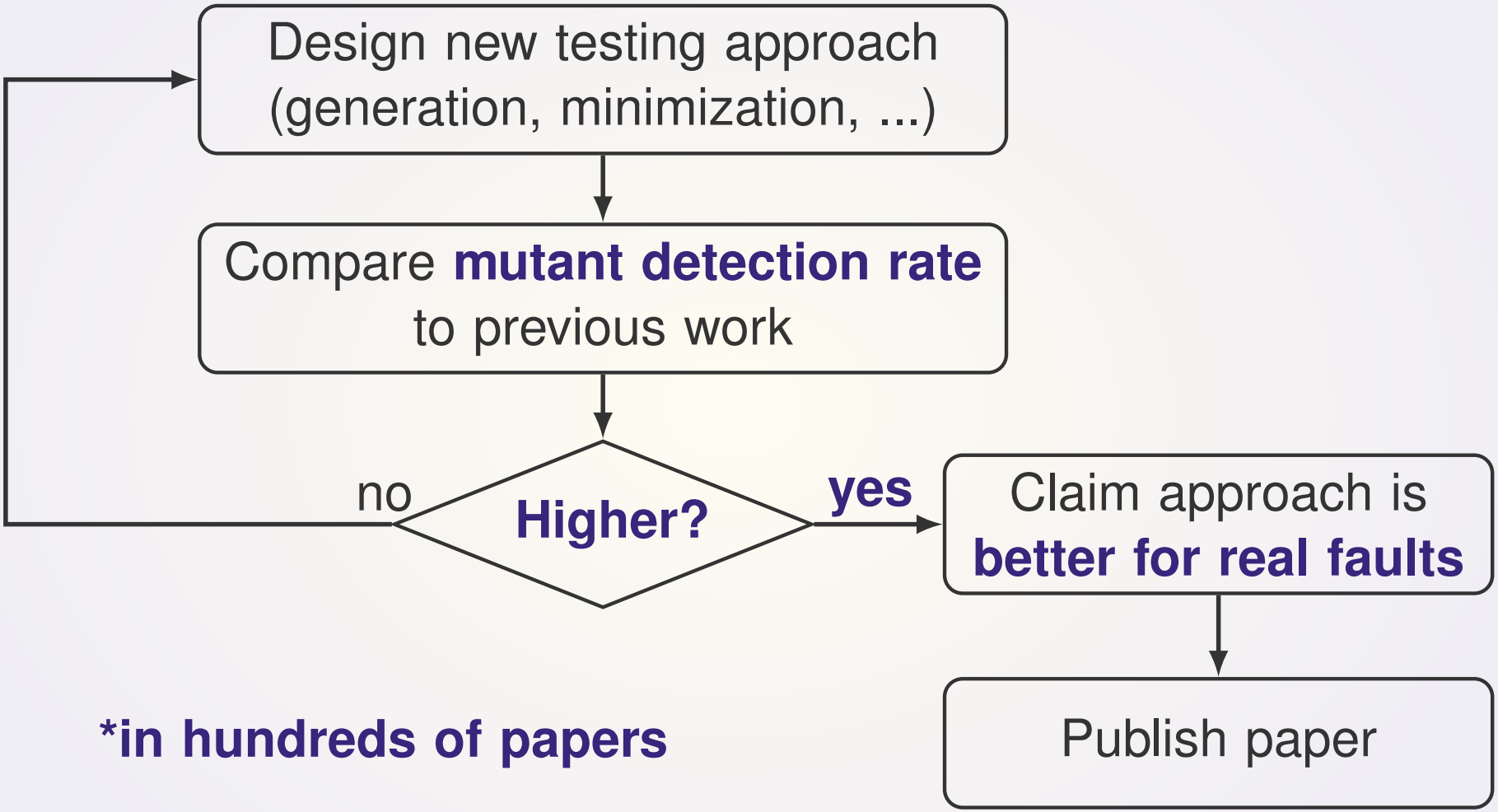
Mutation analysis: How it is used



Mutation analysis: How it is used

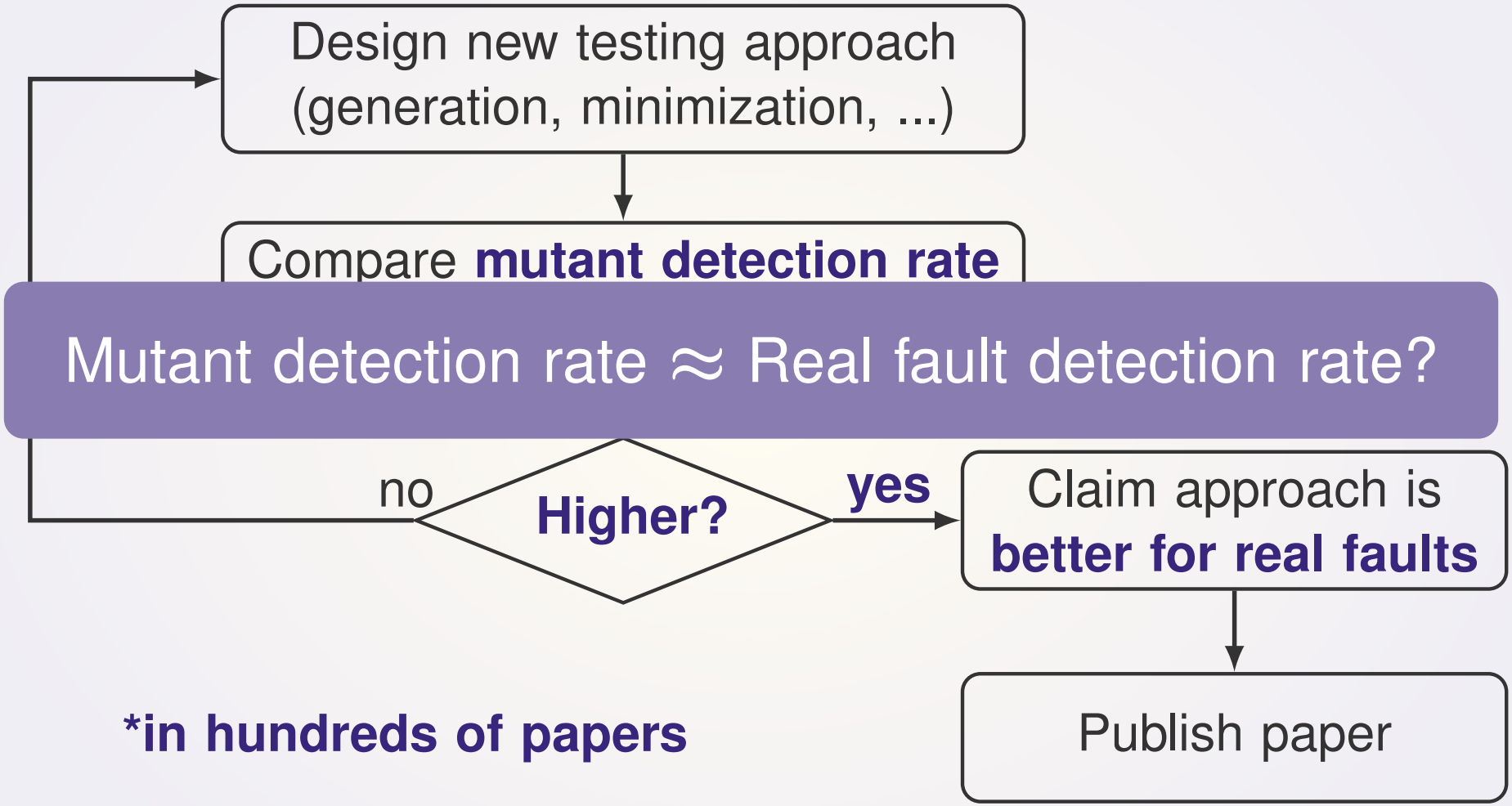


Mutation analysis: How it is used*



***in hundreds of papers**

Mutation analysis: How it is used*



***in hundreds of papers**

Related work

ISSTA'96[1] ICSE'05[2] **FSE'14**

[1] Daran and Thévenod-Fosse, *ISSTA'96*.

[2] Andrews et al., *ICSE'05*.

Related work

	ISSTA'96[1]	ICSE'05[2]	**FSE'14**
KLOC	1	6	**321**

[1] Daran and Thévenod-Fosse, *ISSTA'96*.

[2] Andrews et al., *ICSE'05*.

Related work

	ISSTA'96[1]	ICSE'05[2]	**FSE'14**
KLOC	1	6	**321**
Faults	12	38	**357**

[1] Daran and Thévenod-Fosse, *ISSTA'96*.

[2] Andrews et al., *ICSE'05*.

Related work

	ISSTA'96[1]	ICSE'05[2]	**FSE'14**
KLOC	1	6	**321**
Faults	12	38	**357**
Mutants	24	1,100	**230,000**

[1] Daran and Thévenod-Fosse, *ISSTA'96*.

[2] Andrews et al., *ICSE'05*.

Related work

	ISSTA'96[1]	ICSE'05[2]	**FSE'14**
KLOC	1	6	**321**
Faults	12	38	**357**
Mutants	24	1,100	**230,000**
Tests	generated	generated	**generated & developer-written**

[1] Daran and Thévenod-Fosse, *ISSTA'96*.

[2] Andrews et al., *ICSE'05*.

Related work

	ISSTA'96[1]	ICSE'05[2]	**FSE'14**
KLOC	1	6	**321**
Faults	12	38	**357**
Mutants	24	1,100	**230,000**
Tests	generated	generated	**generated & developer-written**
	—	—	**Effect of code coverage considered**
	—	—	**Qualitative study of real faults**

[1] Daran and Thévenod-Fosse, *ISSTA'96*.

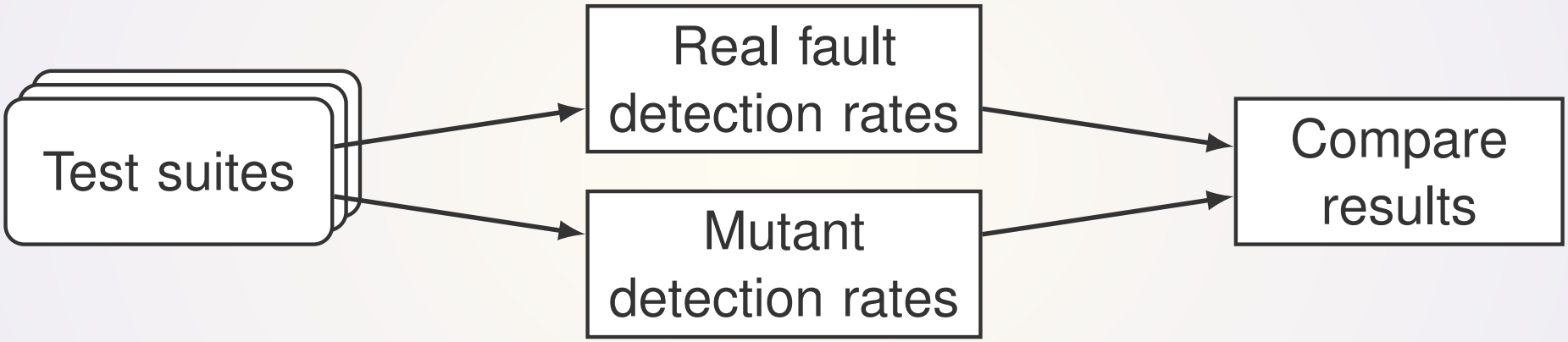
[2] Andrews et al., *ICSE'05*.

Are mutants a valid substitute for real faults?

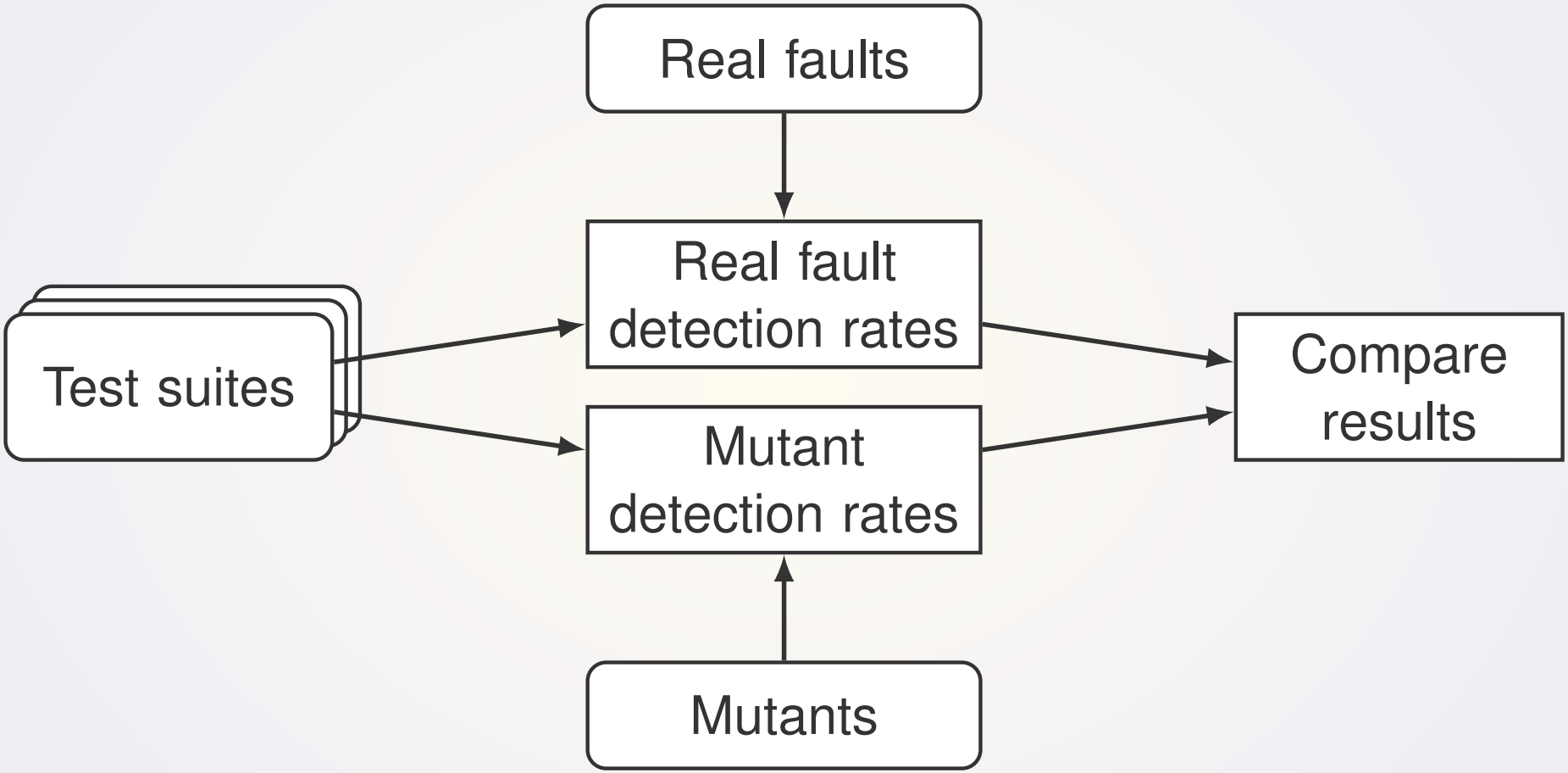
Research Questions

1. Do stronger test suites detect more mutants?
2. What types of real faults are not represented by mutants?
3. Is mutant detection correlated with fault detection?

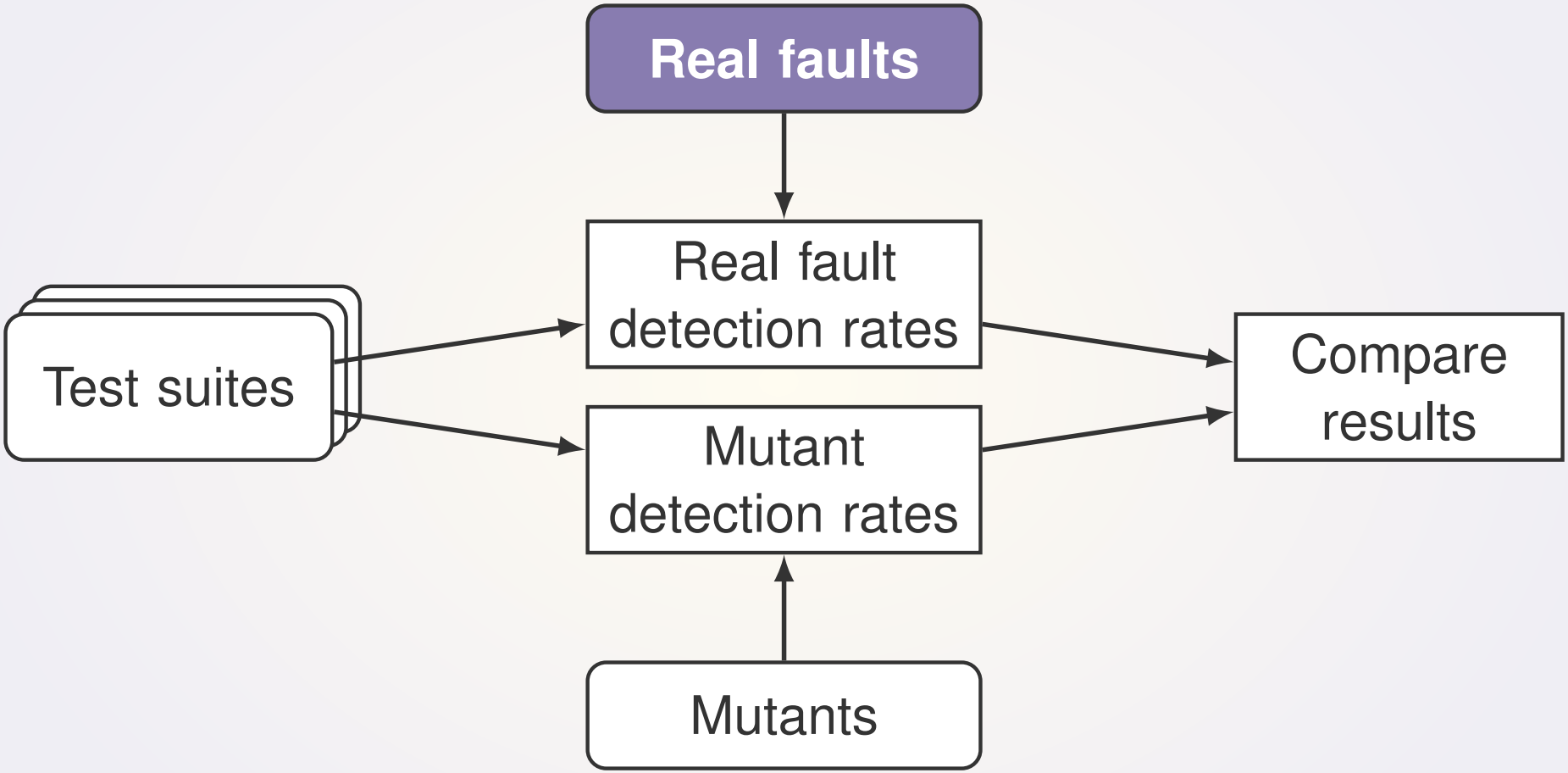
Methodology: Overview



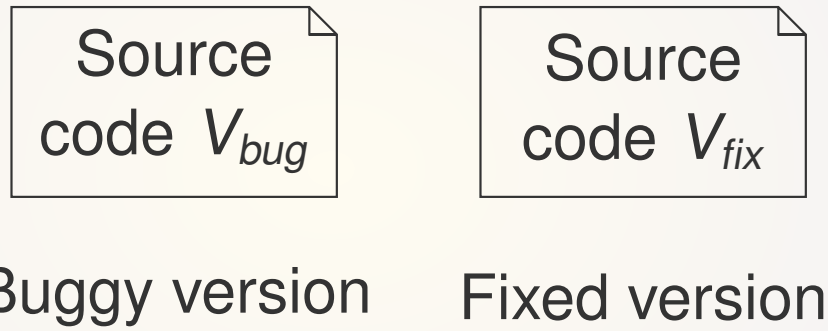
Methodology: Overview



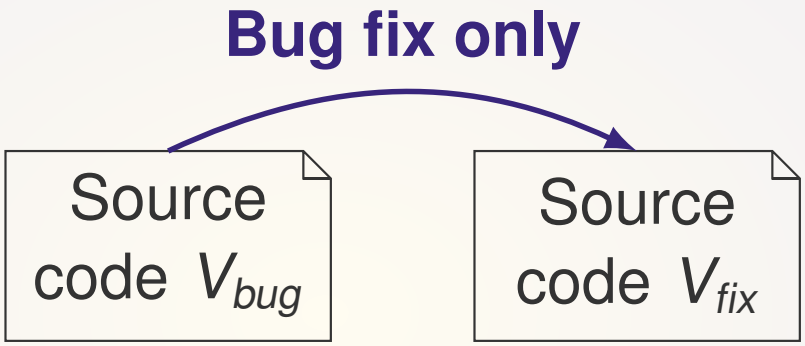
Methodology: Overview



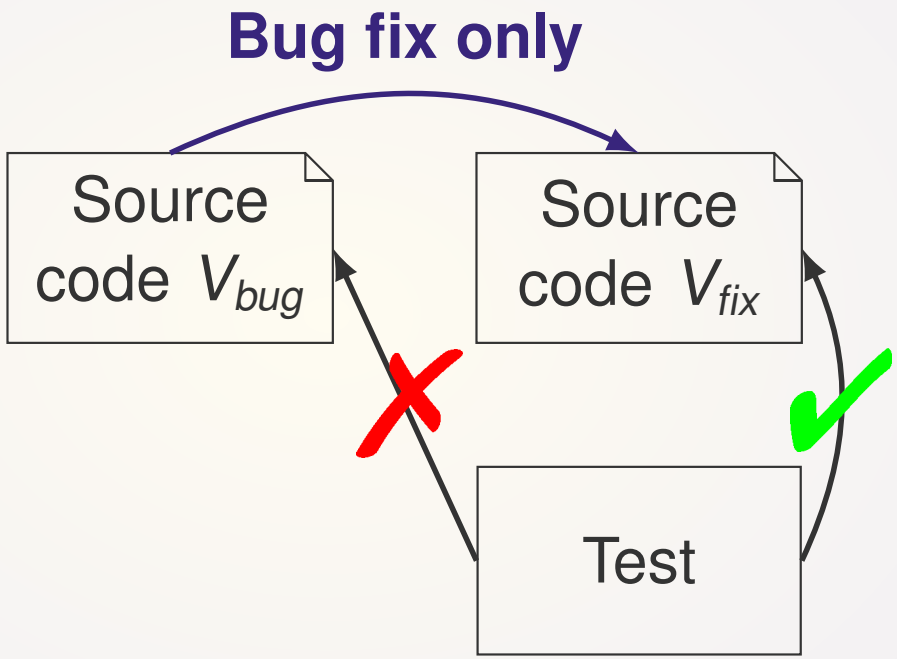
Reproducible and isolated real faults



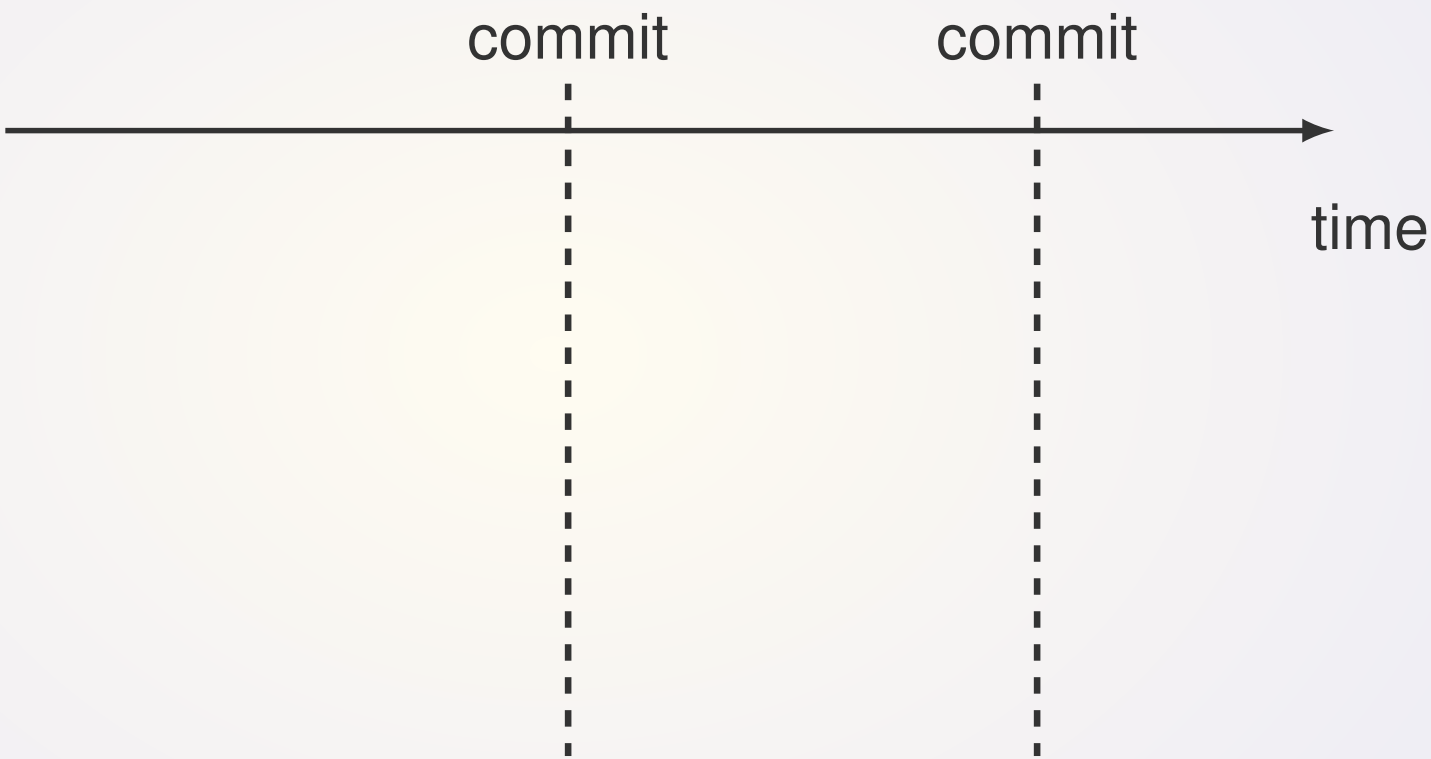
Reproducible and isolated real faults



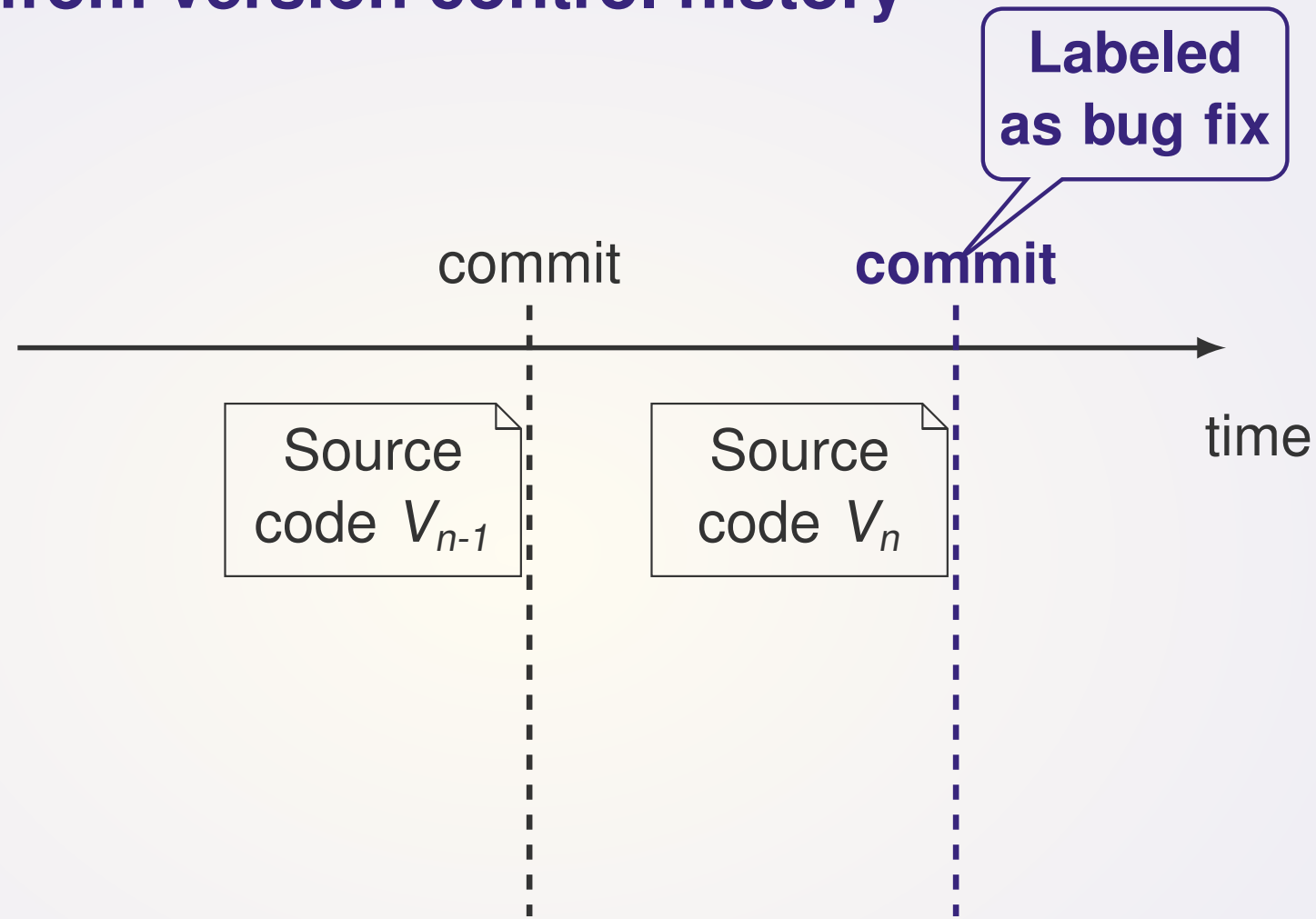
Reproducible and isolated real faults



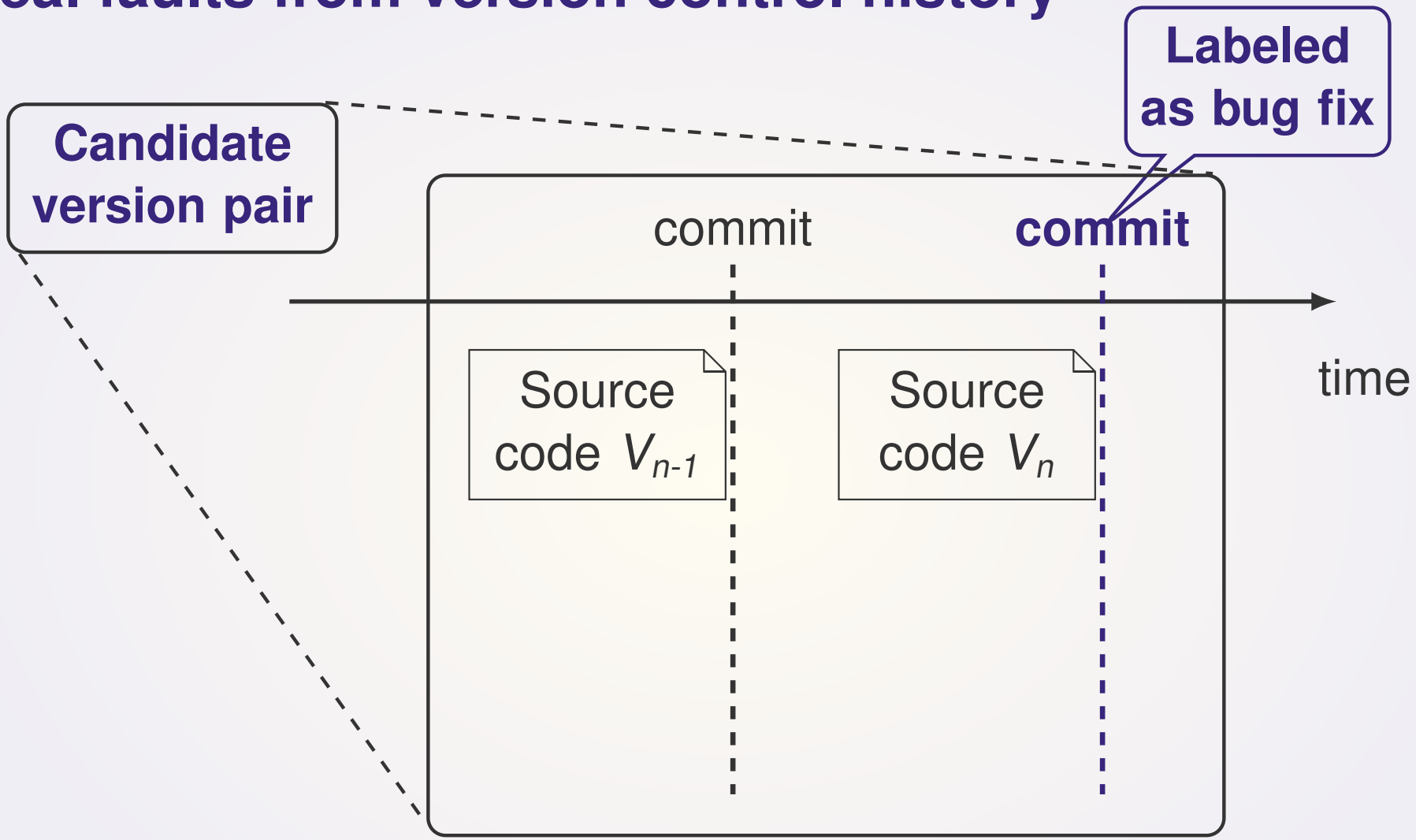
Real faults from version control history



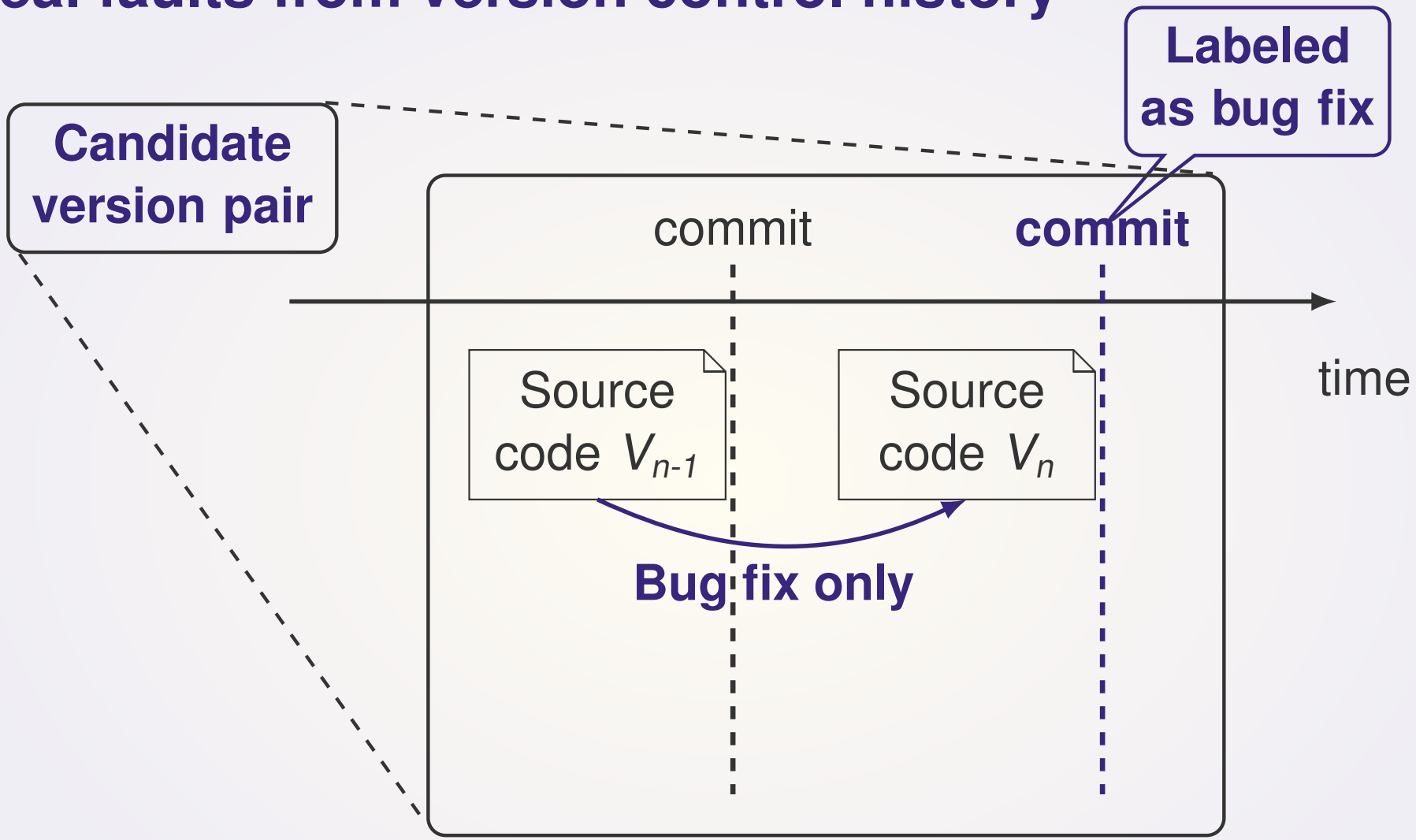
Real faults from version control history



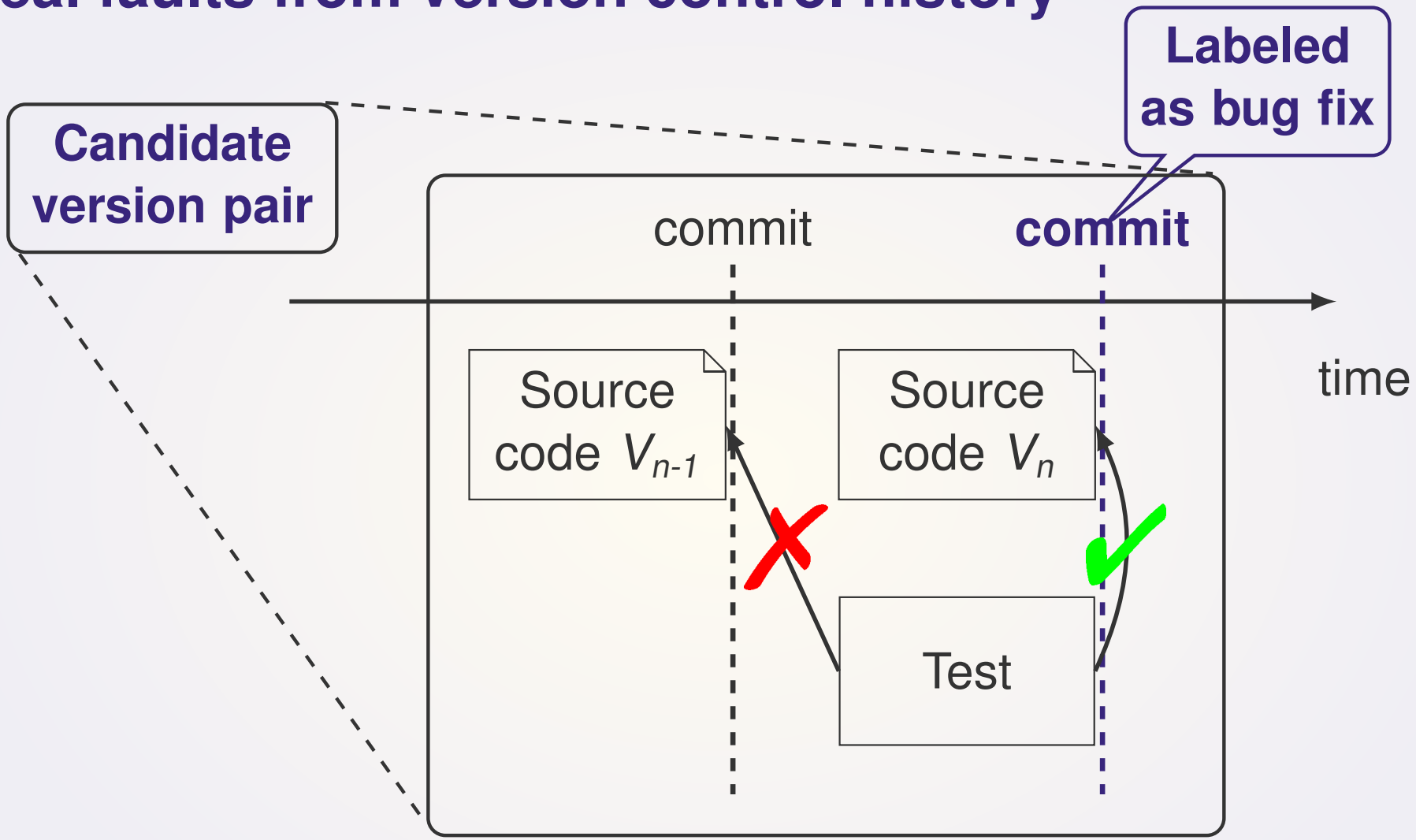
Real faults from version control history



Real faults from version control history



Real faults from version control history



Subject programs

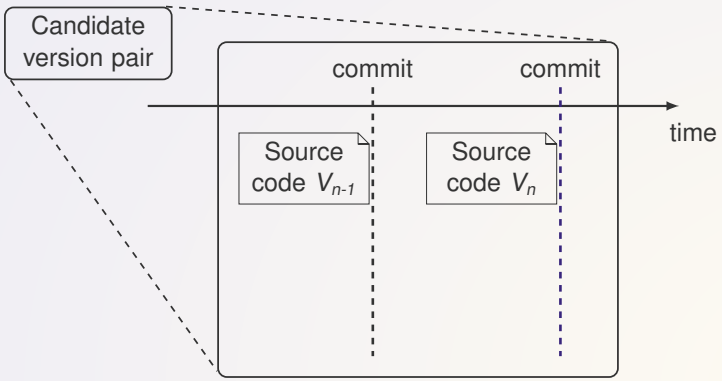
5 open source Java programs

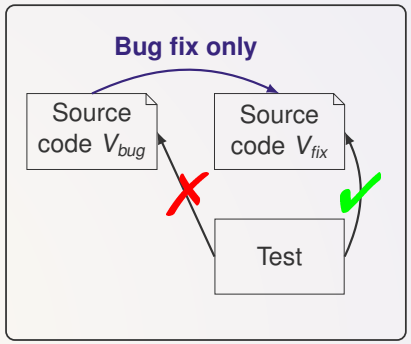
- Different application domains
- Version control and bug tracking systems
- Comprehensive test suites

	KLOC	Test KLOC	Tests
JFreeChart	96	50	2,205
Closure Compiler	90	83	7,927
Commons Math	85	19	3,602
Joda Time	28	53	4,130
Commons Lang	22	6	2,245
Total	**321**	**211**	**20,109**

Real faults

357 reproducible and isolated real faults

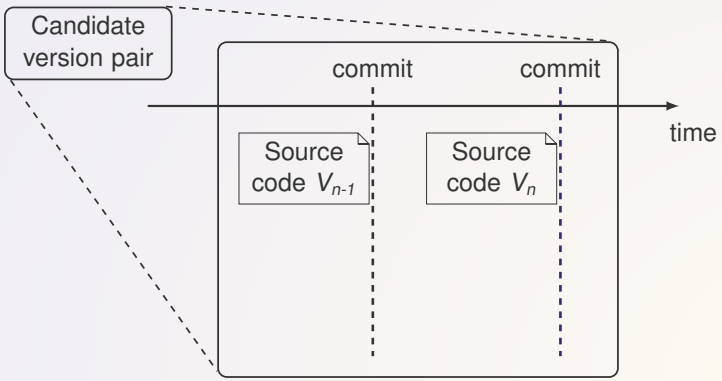


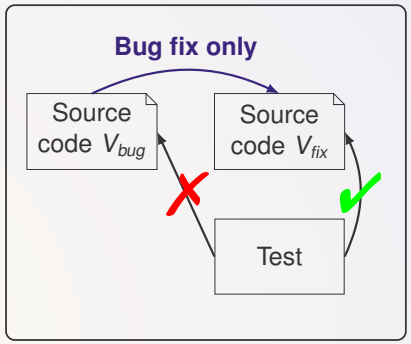


	Candidates	Compilable	Reproducible	Isolated
JFreeChart	80	62	28	26
Closure Compiler	316	227	179	133
Commons Math	435	304	132	106
Joda Time	75	57	29	27
Commons Lang	273	186	69	65
Total	**1,179**	**836**	**437**	**357**

Real faults

357 reproducible and isolated real faults

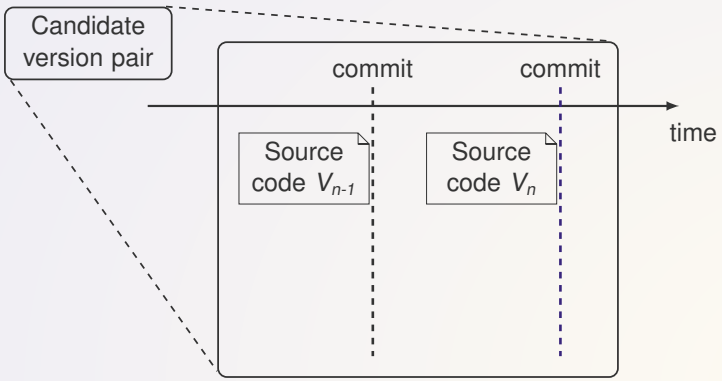


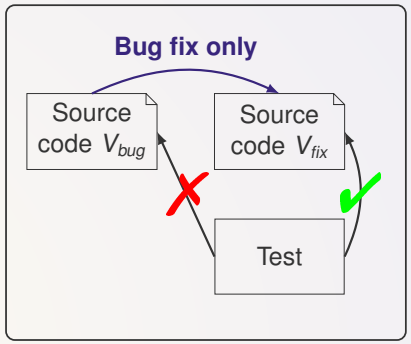


	Candidates	Compilable	Reproducible	Isolated
JFreeChart	80	62	28	26
Closure Compiler	316	227	179	133
Commons Math	435	304	132	106
Joda Time	75	57	29	27
Commons Lang	273	186	69	65
Total	**1,179**	**836**	**437**	**357**

Real faults

357 reproducible and isolated real faults

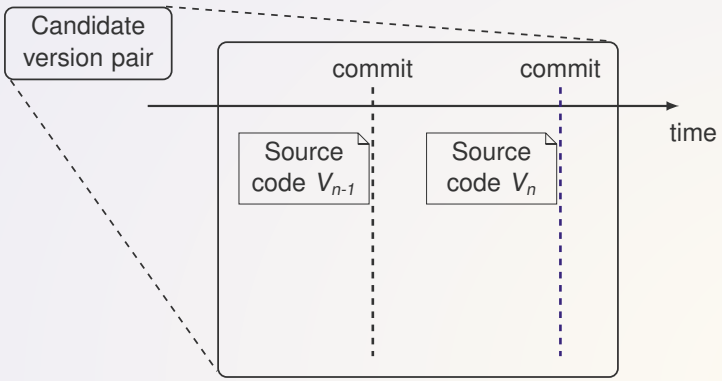


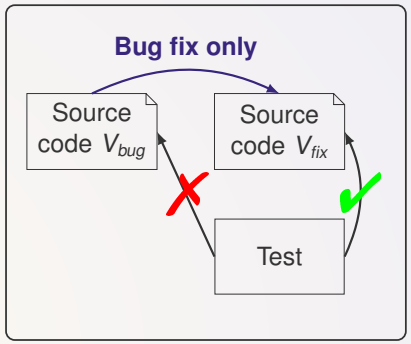


	Candidates	Compilable	Reproducible	Isolated
JFreeChart	80	62	28	26
Closure Compiler	316	227	179	133
Commons Math	435	304	132	106
Joda Time	75	57	29	27
Commons Lang	273	186	69	65
Total	**1,179**	**836**	**437**	**357**

Real faults

357 reproducible and isolated real faults

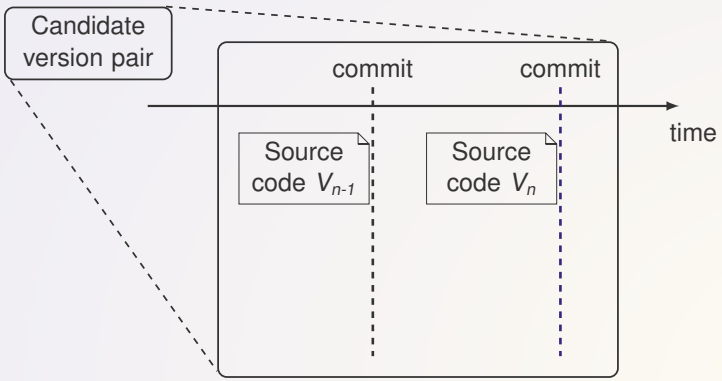




	Candidates	Compilable	Reproducible	Isolated
JFreeChart	80	62	28	26
Closure Compiler	316	227	179	133
Commons Math	435	304	132	106
Joda Time	75	57	29	27
Commons Lang	273	186	69	65
Total	**1,179**	**836**	**437**	**357**

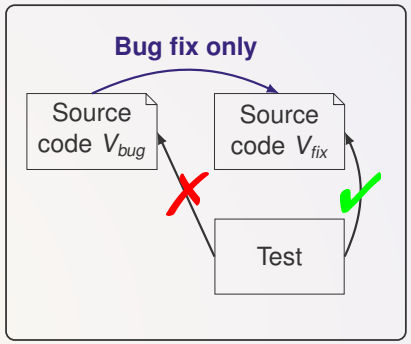
Real faults

357 reproducible and isolated real faults



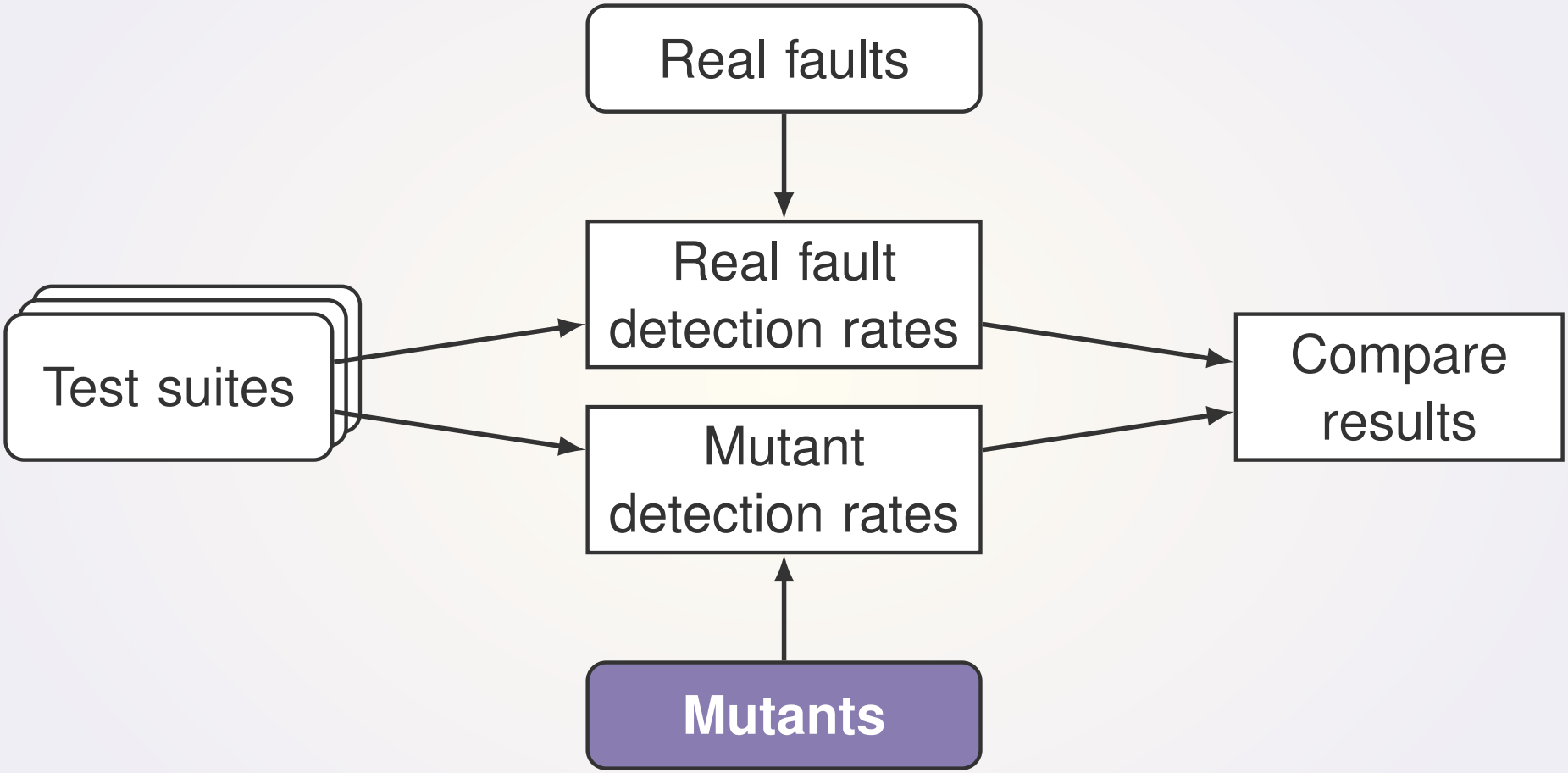
~1 person year

no false positives



	Candidates	Compilable	Reproducible	Isolated
JFreeChart	80	62	28	26
Closure Compiler	316	227	179	133
Commons Math	435	304	132	106
Joda Time	75	57	29	27
Commons Lang	273	186	69	65
Total	**1,179**	**836**	**437**	**357**

Methodology: Overview



Mutants

230,000 mutants generated by Major mutation framework

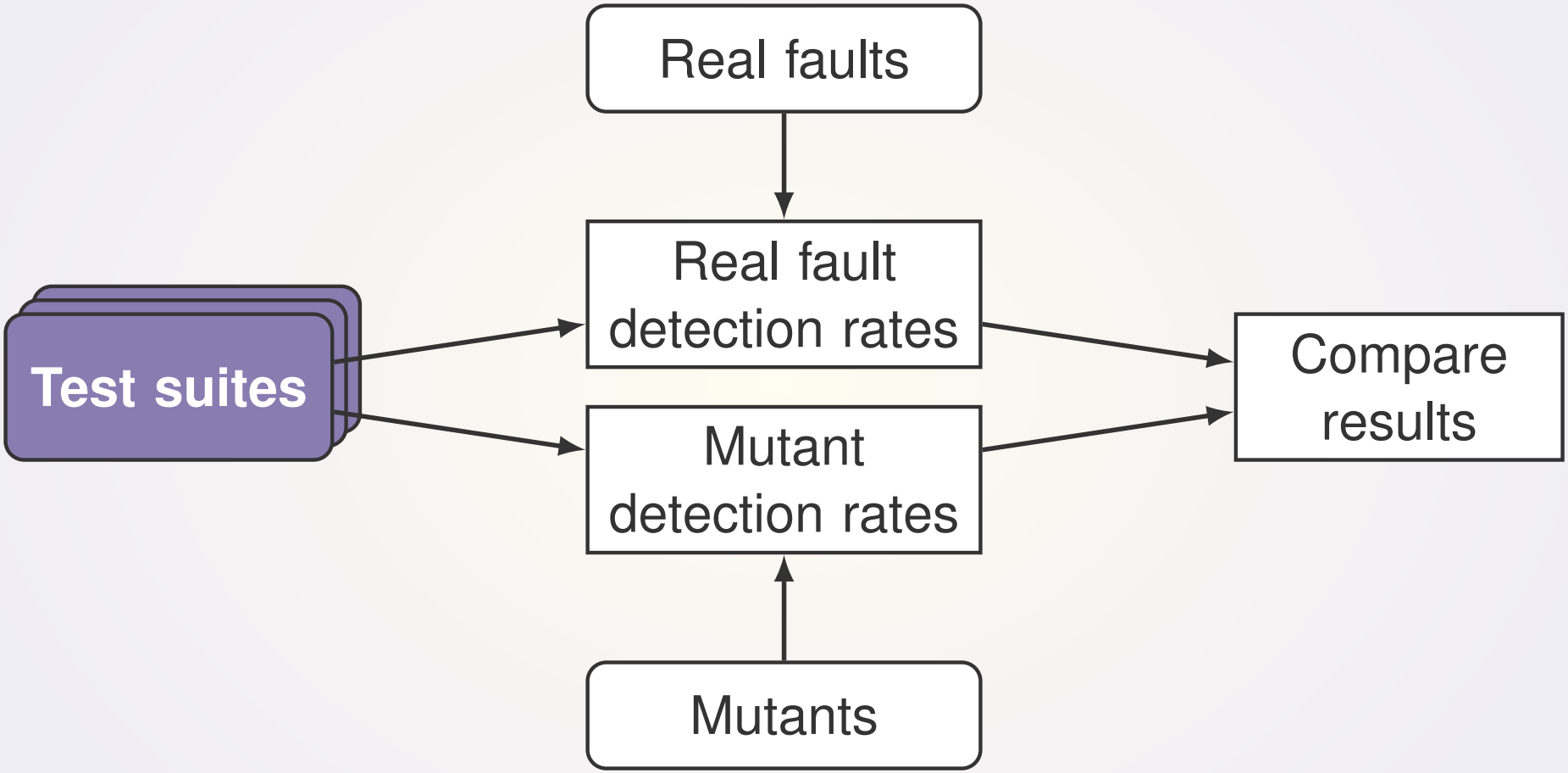
Mutation operators[1,2]

- Replace operators
- Replace literals
- Delete statements
- Modify branch conditions

[1] Namin et al., *ICSE'08*.

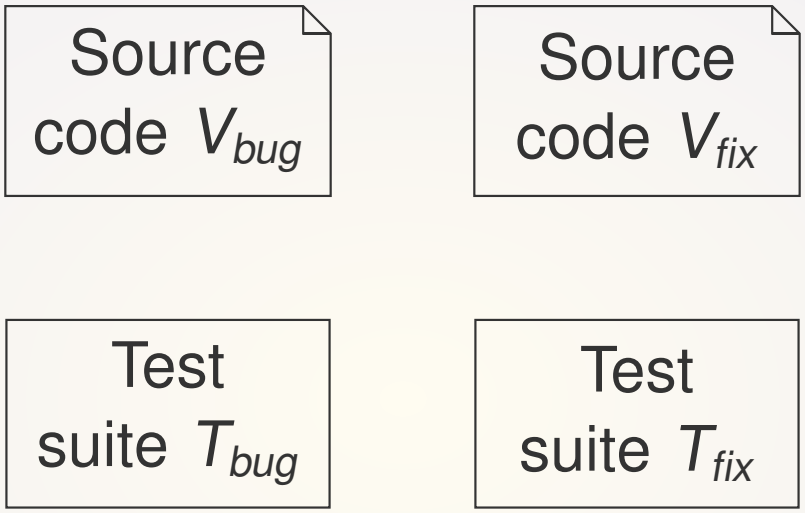
[2] Jia and Harman, *TSE'11*.

Methodology: Overview



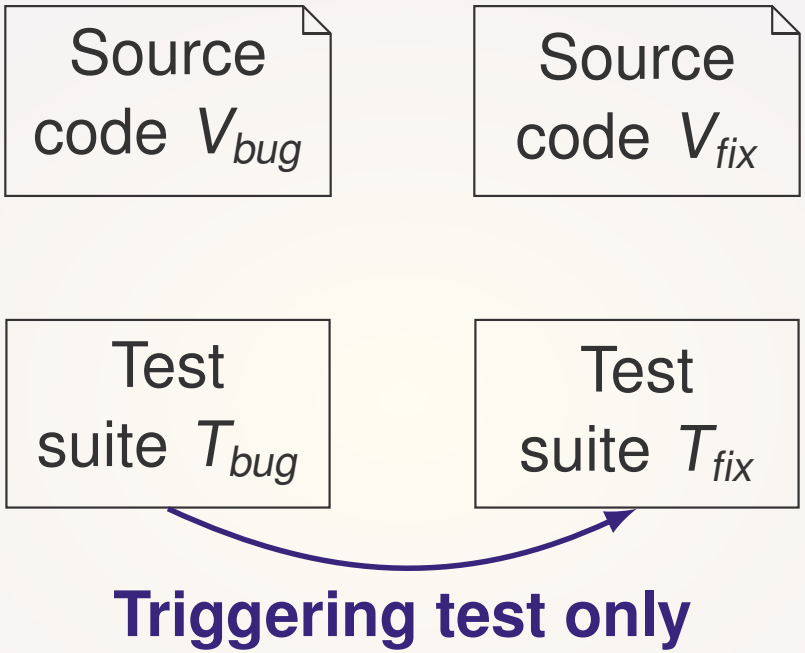
Developer-written test suites

Obtaining related test suites T_{bug} and T_{fix}



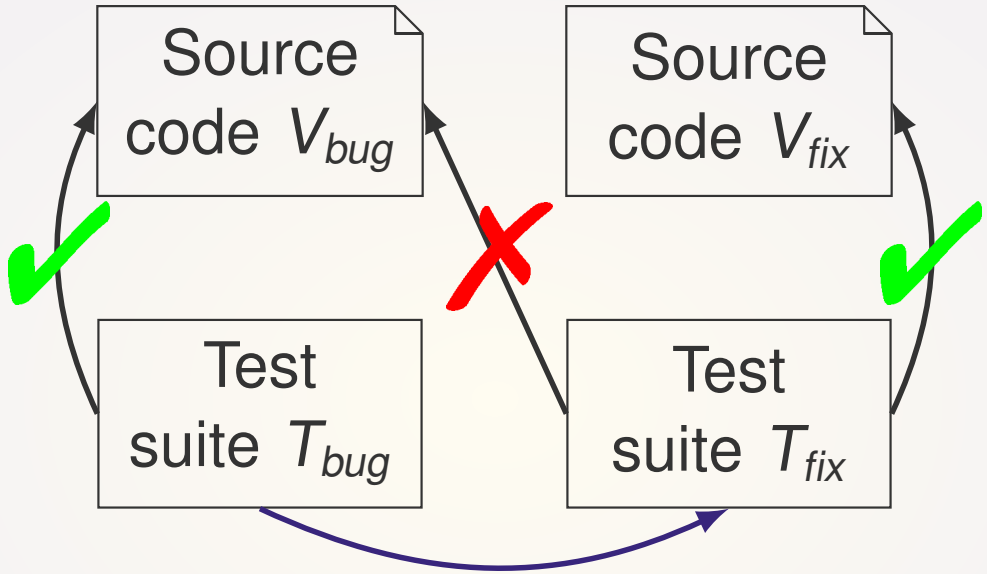
Developer-written test suites

Obtaining related test suites T_{bug} and T_{fix}



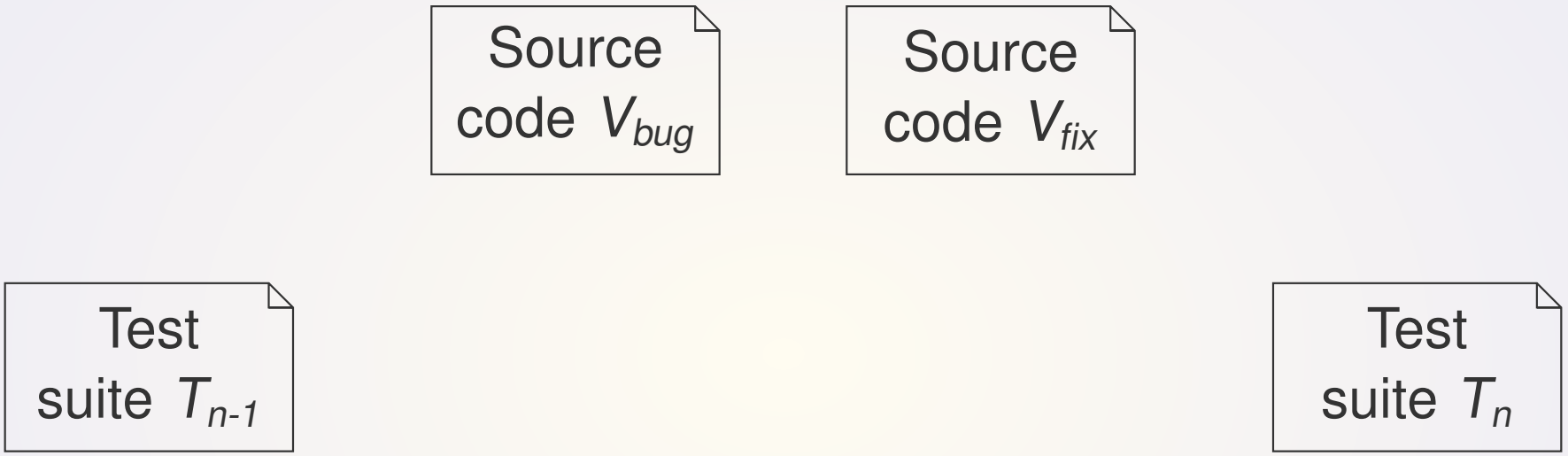
Developer-written test suites

Obtaining related test suites T_{bug} and T_{fix}



Developer-written test suites

Obtaining related test suites T_{bug} and T_{fix}

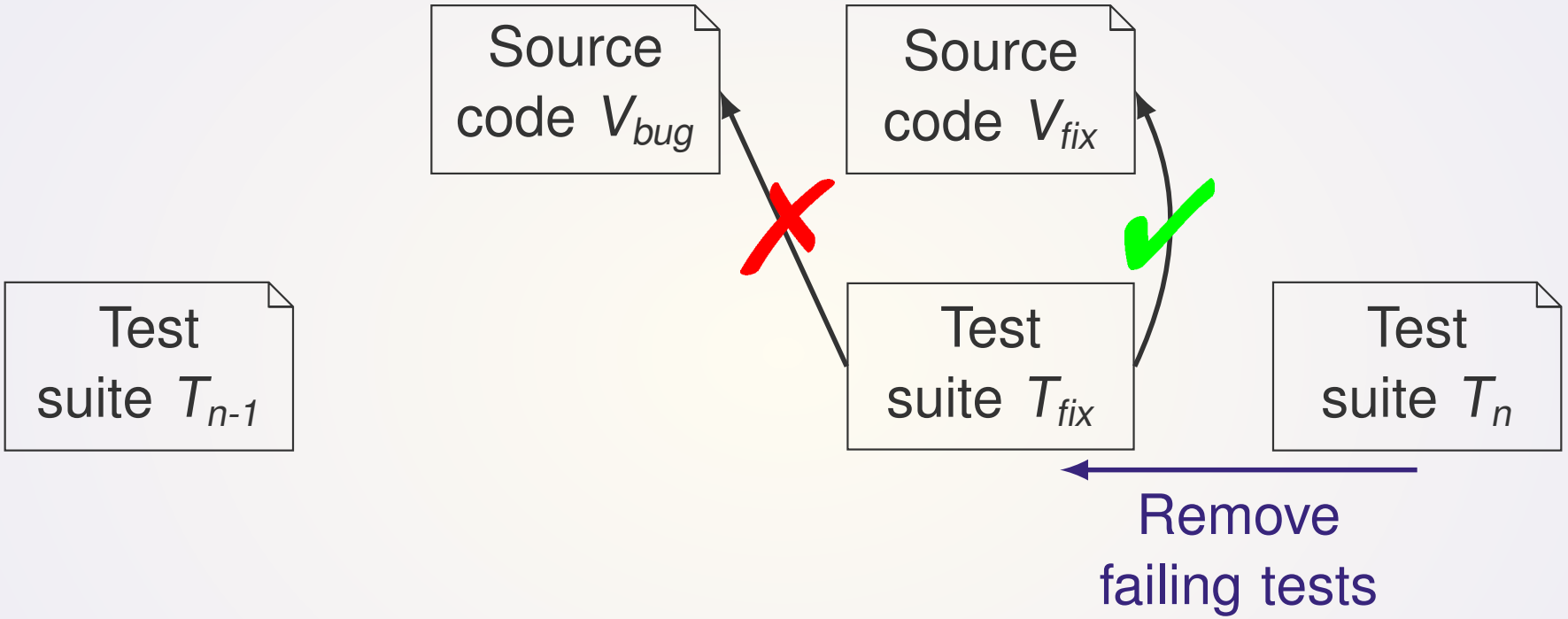


We cannot directly use T_{n-1} and T_n from version control

- T_{n-1} and T_n might include failing tests
- T_n might include additional tests (unrelated to the fault)

Developer-written test suites

Obtaining related test suites T_{bug} and T_{fix}

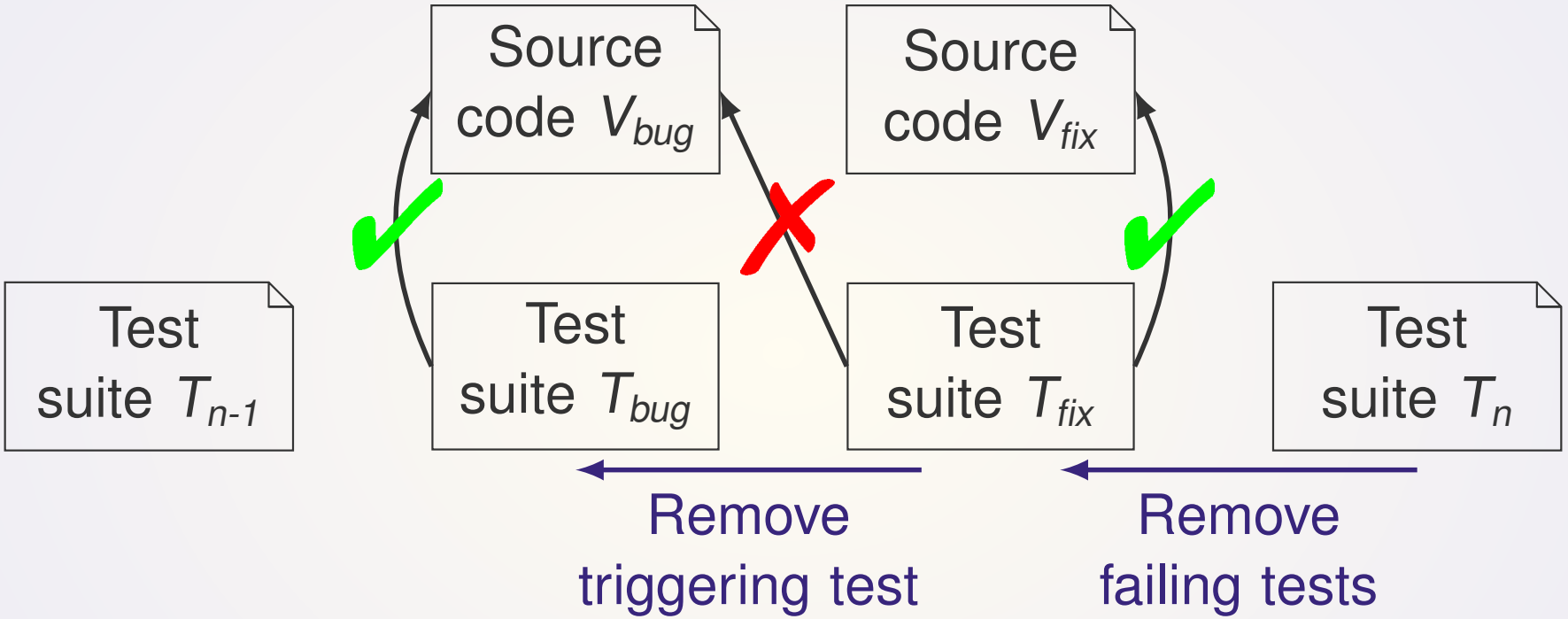


We cannot directly use T_{n-1} and T_n from version control

- T_{n-1} and T_n might include failing tests
- T_n might include additional tests (unrelated to the fault)

Developer-written test suites

Obtaining related test suites T_{bug} and T_{fix}

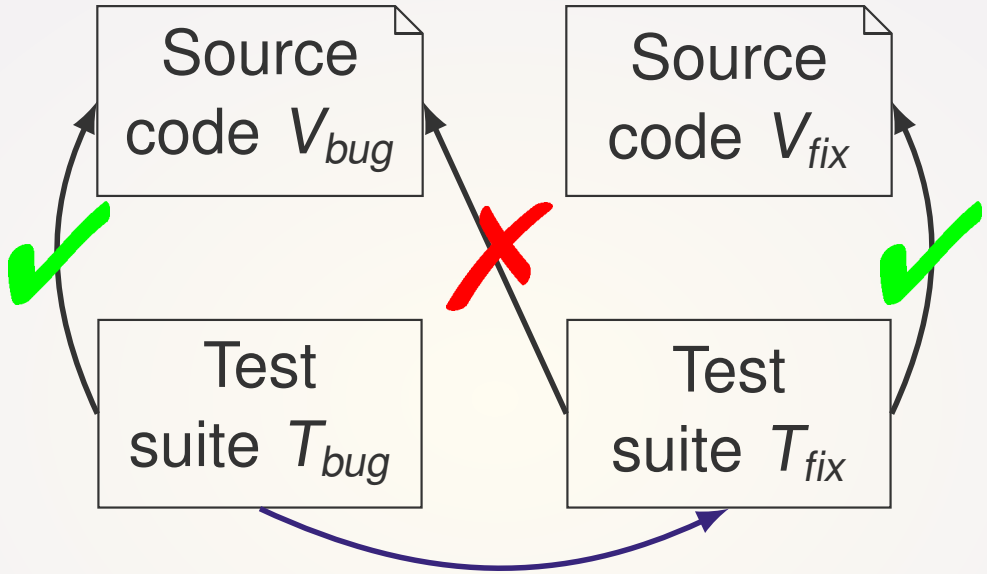


We cannot directly use T_{n-1} and T_n from version control

- T_{n-1} and T_n might include failing tests
- T_n might include additional tests (unrelated to the fault)

Developer-written test suites

Obtaining related test suites T_{bug} and T_{fix}



Automatically-generated test suites

EvoSuite, Randoop, and JCrasher

- Multiple configurations and test objectives

Workflow

1. Generate tests for fixed program version
2. Automatically remove failing tests

Test suites: Summary

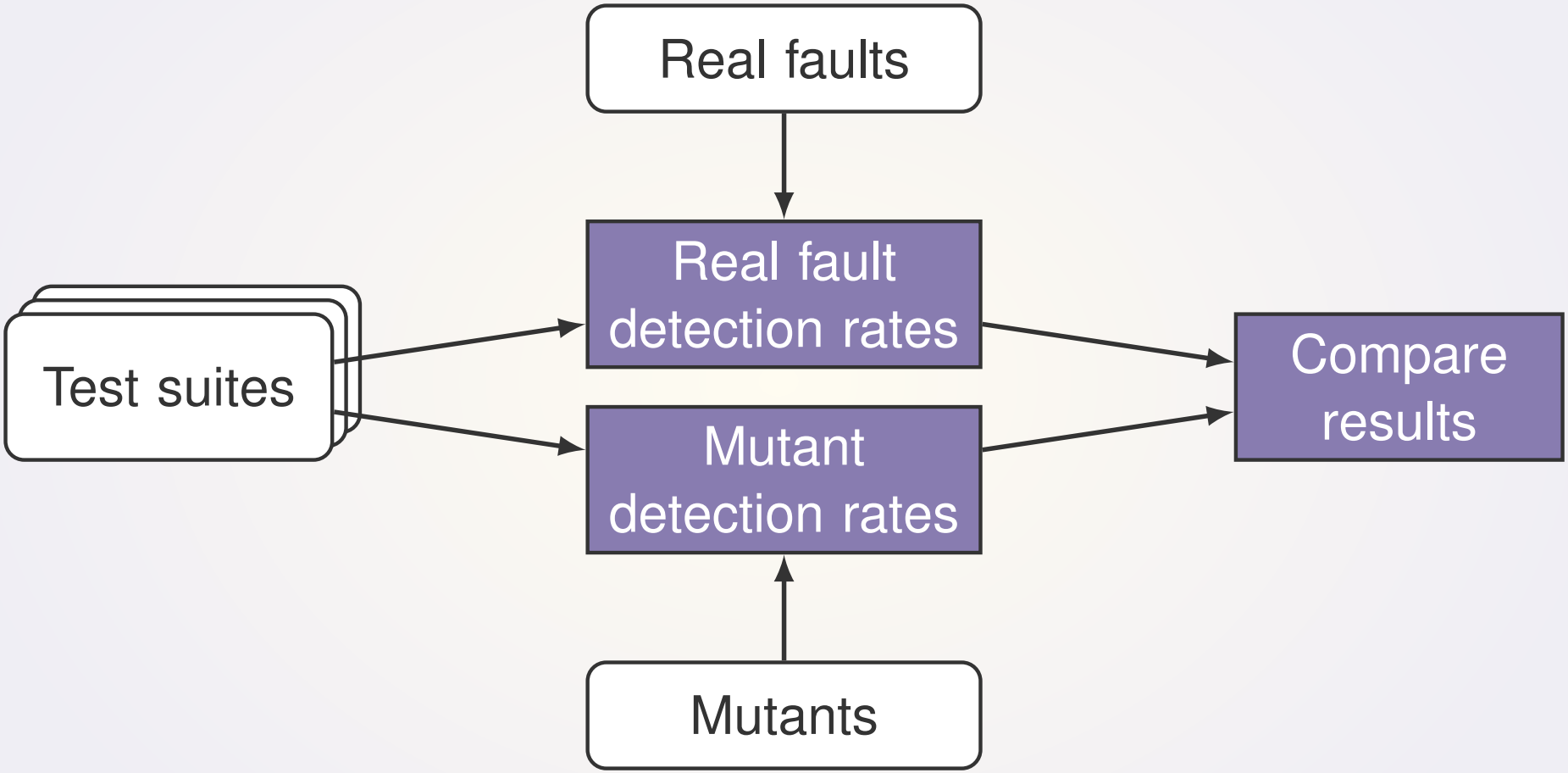
Developer-written test suites

- Related test suite pairs T_{bug} and T_{fix}
- Average **statement coverage** of T_{bug}: **90%**

Automatically-generated test suites

- 35,141 test suites
- Average **statement coverage**: **55%**

Methodology: Overview



Evaluation: Overview

Research Questions

1. Do stronger test suites detect more mutants?
2. What types of real faults are not represented by mutants?
3. Is mutant detection correlated with fault detection?

RQ1: Do stronger test suites detect more mutants?

Setup

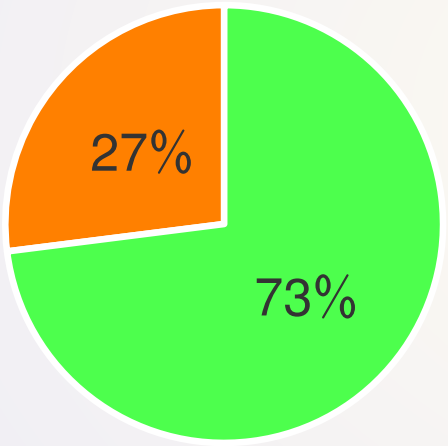
- Developer-written test suite pairs T_{bug} and T_{fix}
- Does T_{fix} have a higher mutant detection rate than T_{bug}?

RQ1: Do stronger test suites detect more mutants?

Setup

- Developer-written test suite pairs T_{bug} and T_{fix}
- Does T_{fix} have a higher mutant detection rate than T_{bug}?

Results

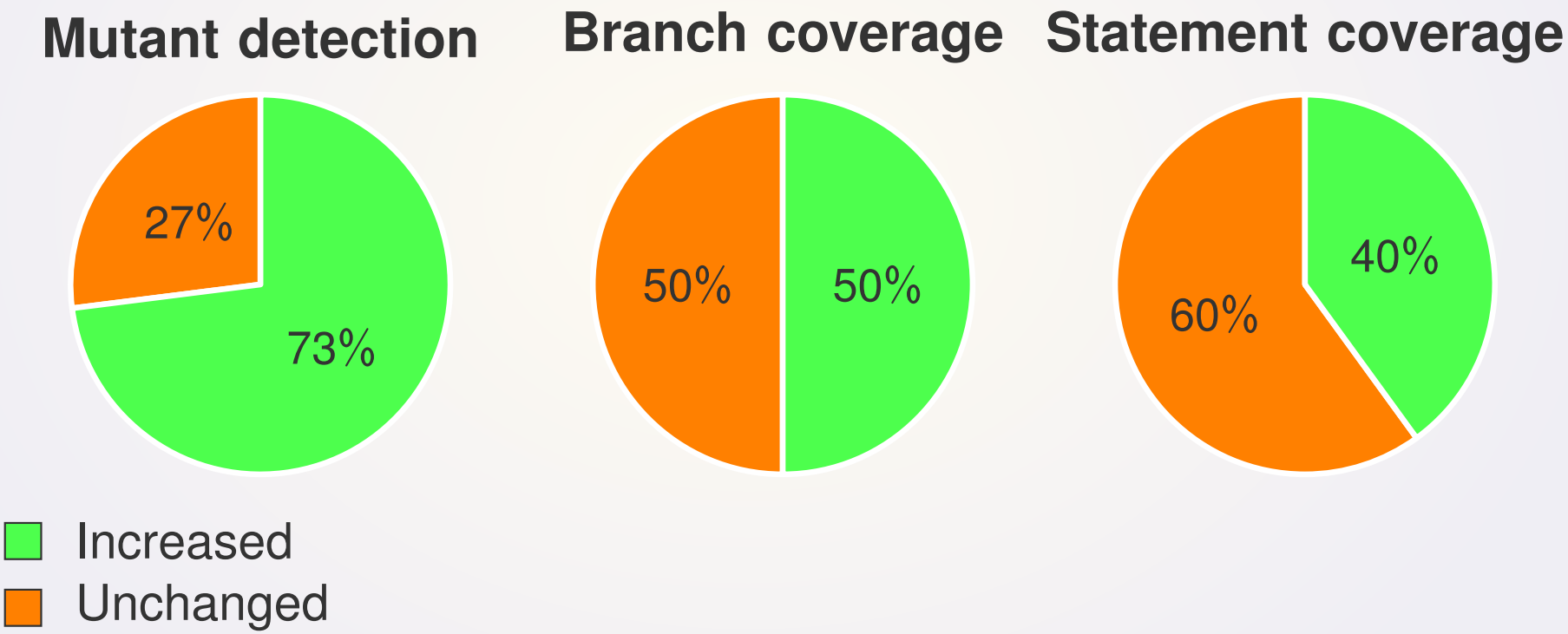


Mutant detection rate
increased for **73%** of faults

Mutant detection rate increased
Mutant detection rate unchanged

RQ1: Do stronger test suites detect more mutants?

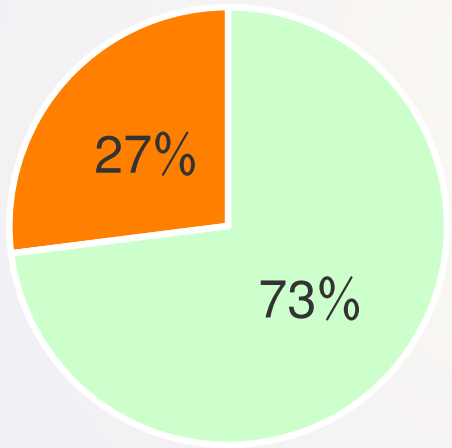
Comparison to code coverage



RQ2: What types of faults are not represented by mutants?

Setup

- Qualitative study for 27% of faults
- Weakness or general limitation?

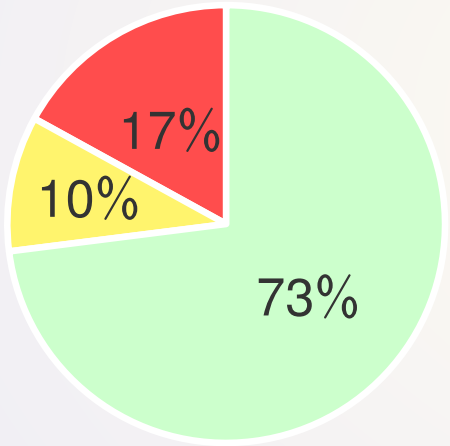


RQ2: What types of faults are not represented by mutants?

Setup

- Qualitative study for 27% of faults
- Weakness or general limitation?

Results



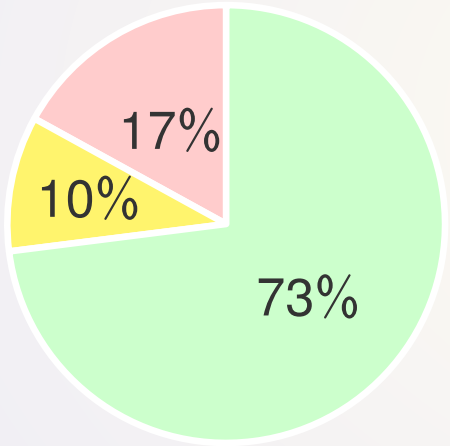
- Mutant detection rate increased
- Weak or missing mutation operator
- No such mutation operator

RQ2: What types of faults are not represented by mutants?

Setup

- Qualitative study for 27% of faults
- Weakness or general limitation?

Results



- Mutant detection rate increased
- Weak or missing mutation operator
- No such mutation operator

Buggy version

```
switch (x) {
 case 1:
    ...
    [ ]
 case 2:
...
```

Fixed version

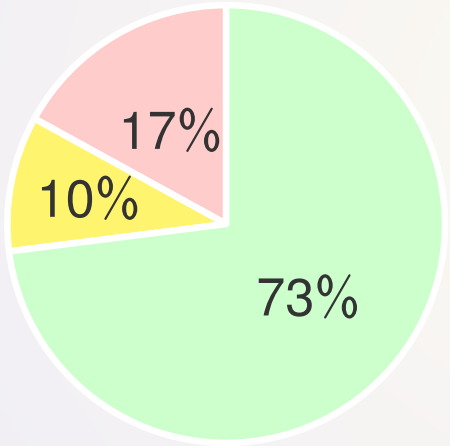
```
switch (x) {
 case 1:
    ...
    return false;
 case 2:
...
```

RQ2: What types of faults are not represented by mutants?

Setup

- Qualitative study for 27% of faults
- Weakness or general limitation?

Results



- Mutant detection rate increased
- Weak or missing mutation operator
- No such mutation operator

Mutation operator:
Delete all returns

Buggy version

```
switch (x) {
 case 1:
   ...
   [          ]
 case 2:
...
```

Fixed version

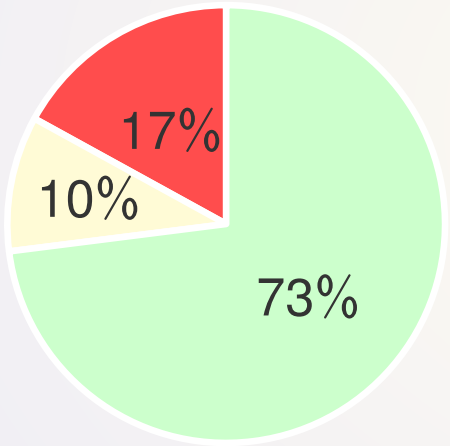
```
switch (x) {
 case 1:
   ...
   return false;
 case 2:
...
```

RQ2: What types of faults are not represented by mutants?

Setup

- Qualitative study for 27% of faults
- Weakness or general limitation?

Results



- Mutant detection rate increased
- Weak or missing mutation operator
- No such mutation operator

Buggy version

```
...
if (isNumZero) {
  return INF;
}
return NaN;
...
```

Fixed version

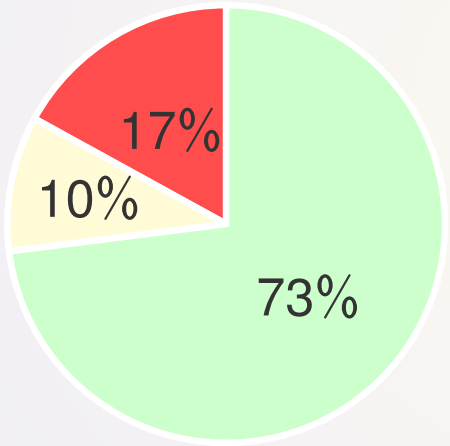
```
...

return NaN;
...
```

RQ2: What types of faults are not represented by mutants?

Setup

- Qualitative study for 27% of faults
- Weakness or general limitation?

Results



- Mutant detection rate increased
- Weak or missing mutation operator
- No such mutation operator

Mutation operator:
Insert ???

Buggy version

```
...
if (isNumZero) {
  return INF;
}
return NaN;
...
```

Fixed version

```
...
return NaN;
...
```

RQ3: Is mutant detection correlated with fault detection?

Setup

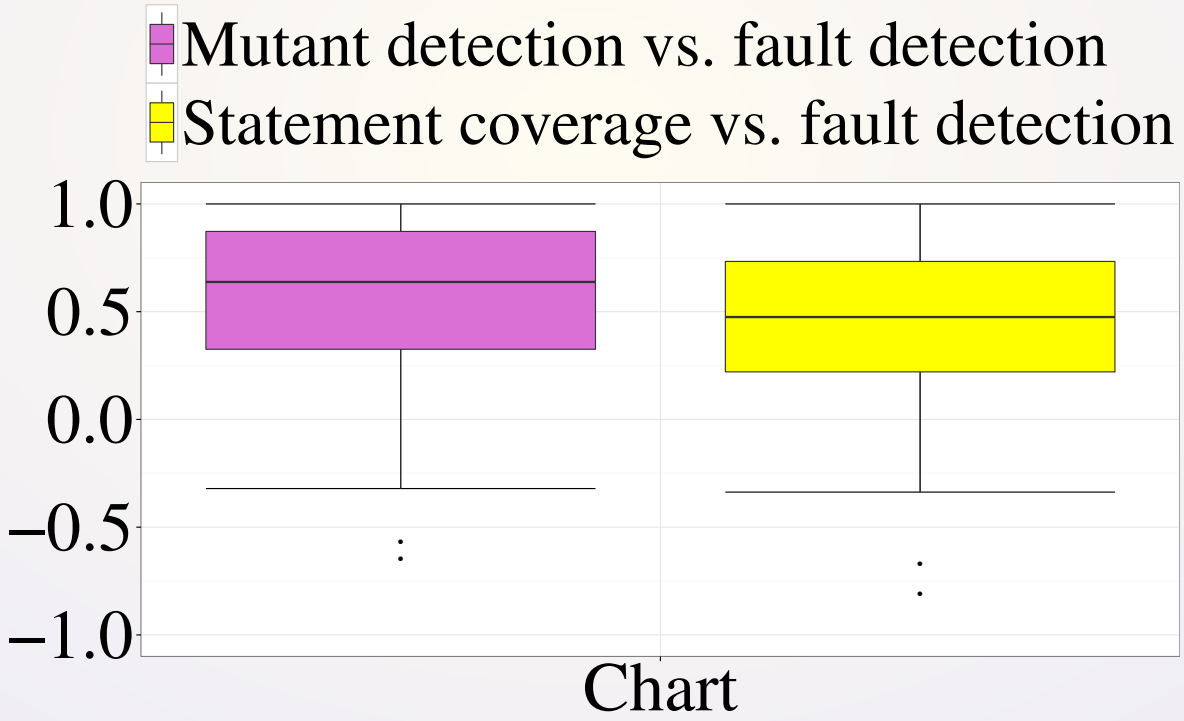
- 35,141 automatically-generated test suites
- How well does mutant detection predict fault detection?

RQ3: Is mutant detection correlated with fault detection?

Setup

- 35,141 automatically-generated test suites
- How well does mutant detection predict fault detection?

Results

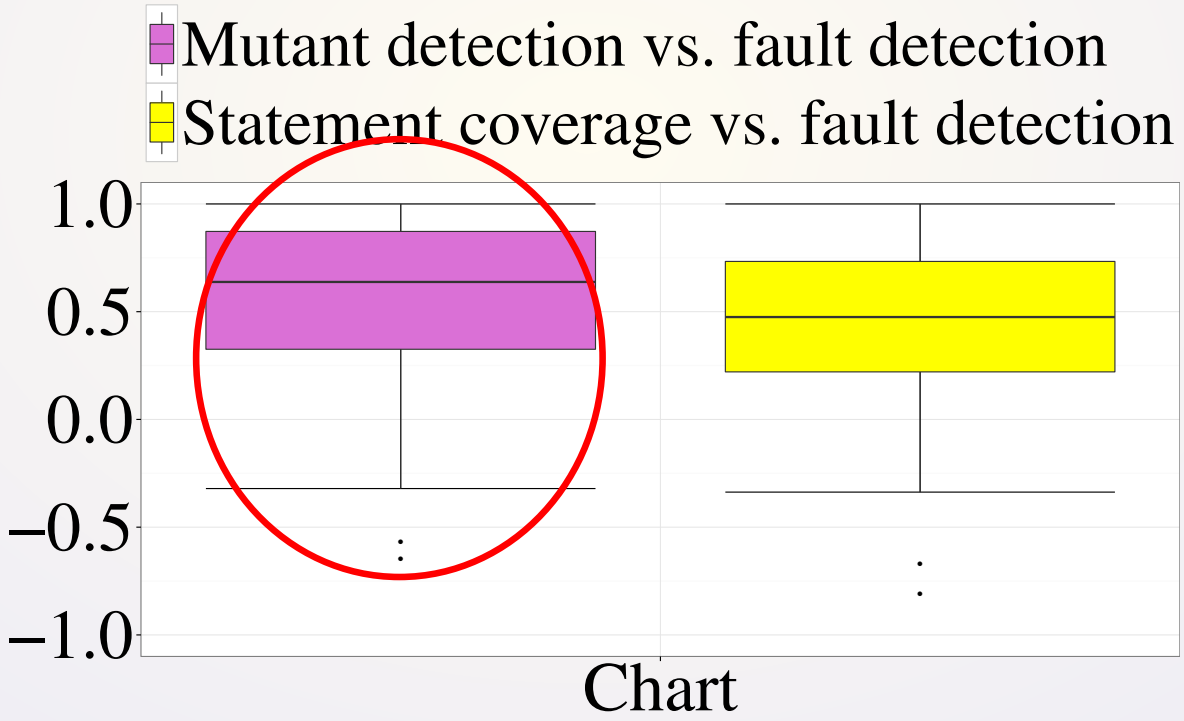


RQ3: Is mutant detection correlated with fault detection?

Setup

- 35,141 automatically-generated test suites
- How well does mutant detection predict fault detection?

Results

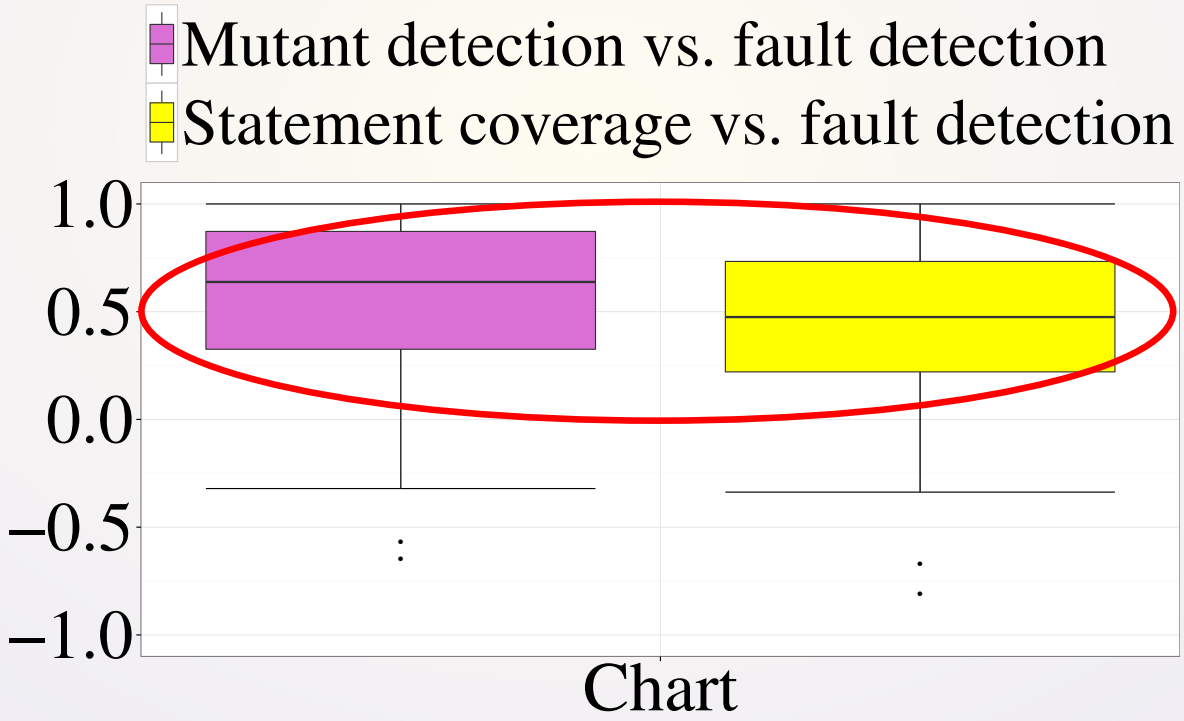


RQ3: Is mutant detection correlated with fault detection?

Setup

- 35,141 automatically-generated test suites
- How well does mutant detection predict fault detection?

Results

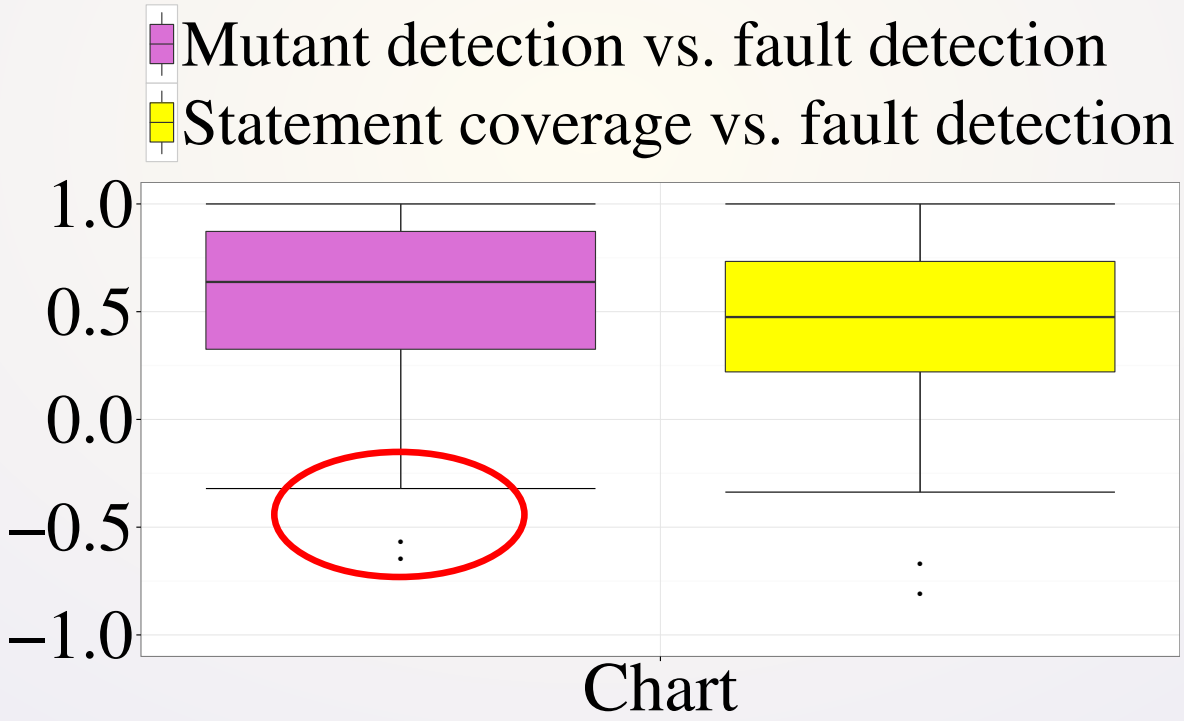


RQ3: Is mutant detection correlated with fault detection?

Setup

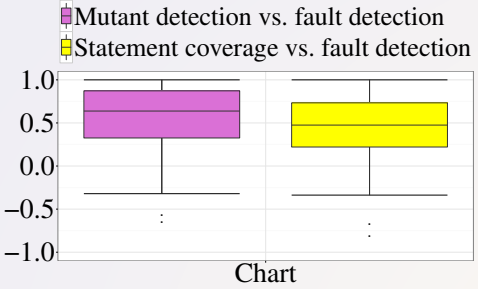
- 35,141 automatically-generated test suites
- How well does mutant detection predict fault detection?

Results



Mutants are a valid substitute for most real faults

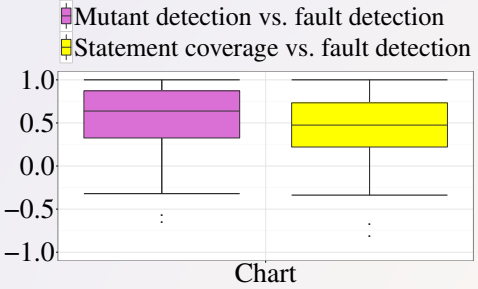
Mutant detection is positively correlated with fault detection



Mutation-based test generation is promising

Mutants are a valid substitute for most real faults

Mutant detection is positively correlated with fault detection



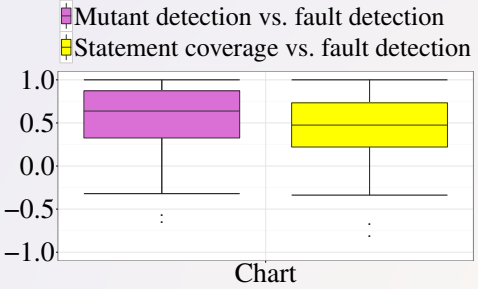
Mutation-based test generation is promising

Mutant detection is more sensitive to faults than coverage

Don’t use code coverage for test suite minimization:
You might miss up to 60% of real faults!

Mutants are a valid substitute for most real faults

Mutant detection is positively correlated with fault detection

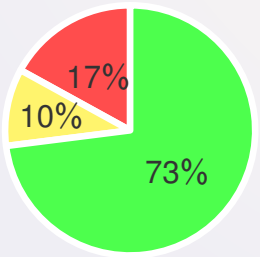


Mutation-based test generation is promising

Mutant detection is more sensitive to faults than coverage

Don't use code coverage for test suite minimization:
You might miss up to 60% of real faults!

17% of faults cannot be represented by any mutants



Mutation results do not generalize to those faults

http://defects4j.org

http://mutation-testing.org