Which Generated Test Failures Are Fault Revealing?

Mijung Kim Shing-Chi Cheung Sunghun Kim

The Hong Kong University of Science and Technology

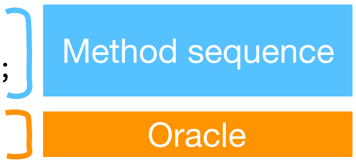
ESEC/FSE 2018

Generated Test Case

Class under test: `FinanceApp`

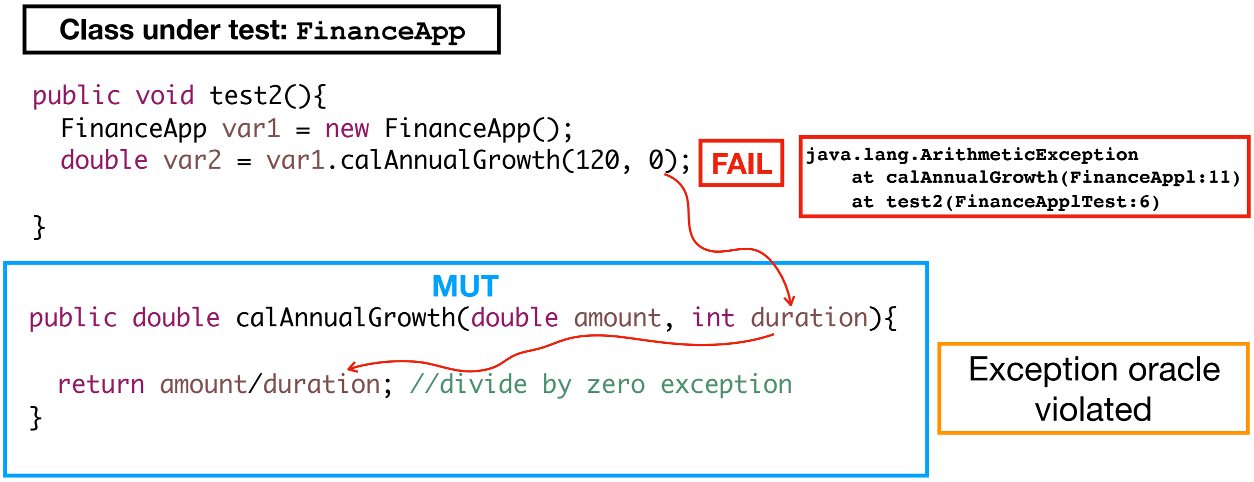
```
public void test1(){
  FinanceApp var1 = new FinanceApp();
  double var2 = var1.calAnnualGrowth(120, 12);

  assertFalse(var1.equals(null));
}
```



	Method Sequence	Automated Oracles
Randoop	Random-based	- No exception - Java contracts hold on objects `(equals, hashCode)`
EVOSUITE	Search-based	

Generated Test Failure



Class under test: `FinanceApp`

```
public void test2(){
  FinanceApp var1 = new FinanceApp();
  double var2 = var1.calAnnualGrowth(120, 0);
}
```

FAIL

```
java.lang.ArithmeticException
    at calAnnualGrowth(FinanceAppl:11)
    at test2(FinanceApplTest:6)
```

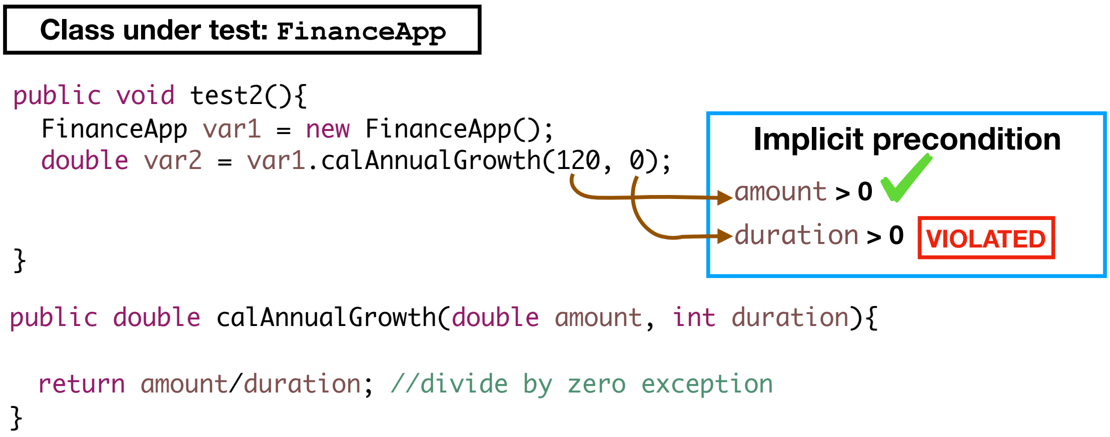
MUT

```
public double calAnnualGrowth(double amount, int duration){

  return amount/duration; //divide by zero exception
}
```

Exception oracle violated

Is the Failure Fault Revealing?



Class under test: `FinanceApp`

```
public void test2(){
  FinanceApp var1 = new FinanceApp();
  double var2 = var1.calAnnualGrowth(120, 0);


}
```

Implicit precondition

`amount` > 0 ✓

`duration` > 0 **VIOLATED**

```
public double calAnnualGrowth(double amount, int duration){

  return amount/duration; //divide by zero exception
}
```

Is the Failure Fault Revealing?

NOPE

False Alarm

Class under test: `FinanceApp`

```
public void test2(){
  FinanceApp var1 = new FinanceApp();
  double var2 = var1.calAnnualGrowth(120, 0);


}
```

FAIL

Because it violates implicit precondition

```
public double calAnnualGrowth(double amount, int duration){

  return amount/duration; //divide by zero exception
}
```

Which Failures Are Fault Revealing?

Problem

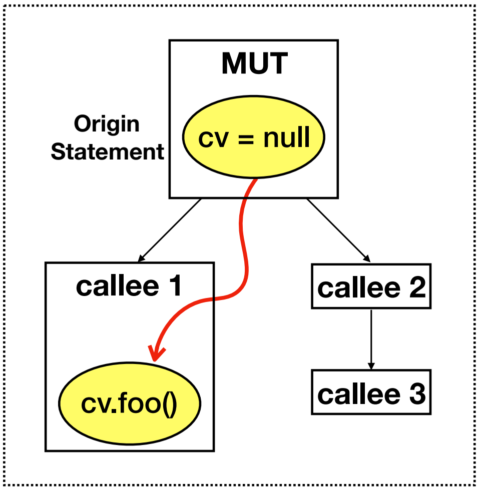
- Preconditions are **not often specified** in the code
- Preconditions can be mined using human-written tests, which heavily rely on their quality

Our work PAF

Given a set of generated failing tests throwing exceptions,

- **Infers** violations of preconditions and **partitions** failing tests to likely violation and non-violation
- **Groups** by the same cause of failures
- **Prioritizes** based on likelihood of violations

Inferring Precondition Violation



→ : Failure inducing dataflow

cv : crash variable

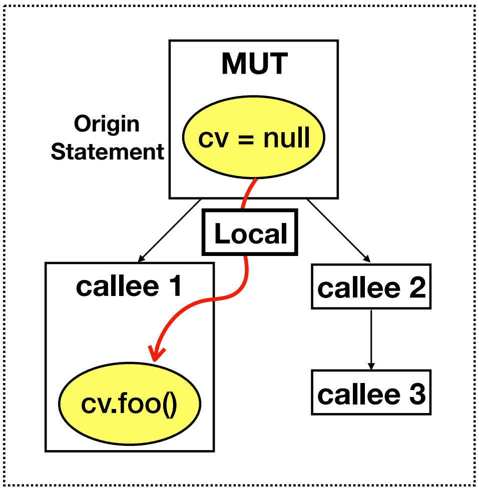
Crash variable

Variable whose value is used when the program crashes

Origin statement

Statement that assigns the incorrect value in failure inducing dataflow

Intuition Behind - Local



→ : Failure inducing dataflow

cv : crash variable

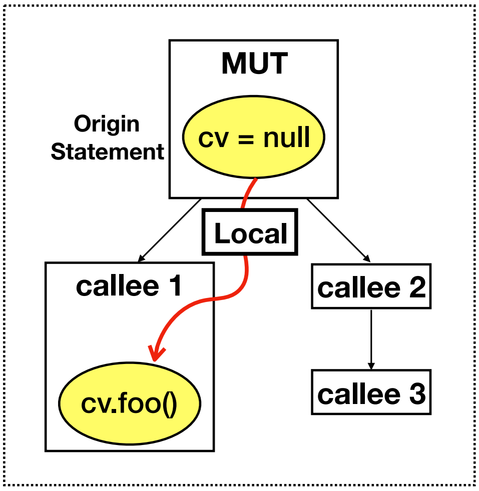
Failure-inducing dataflow is **local** to MUT's computation.

The dataflow is **wholly induced by the MUT's implementation** programmed by its developers.

The chance of violating preconditions is **low**.

Inferring Precondition Violation

→ : Failure inducing dataflow



Origin Statement

MUT

cv = null

Local

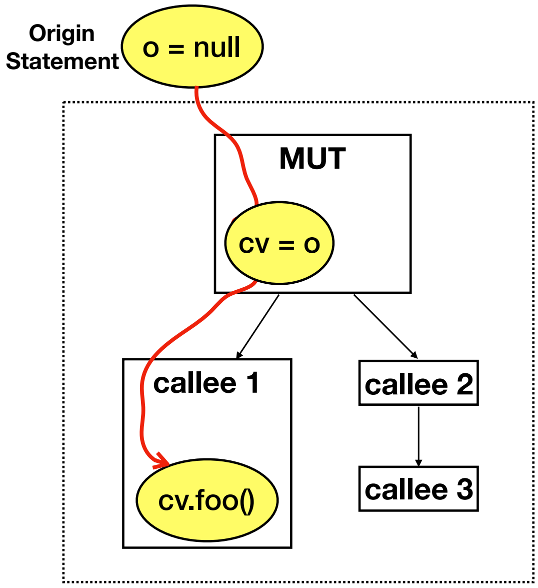
callee 1

cv.foo()

callee 2

callee 3

cv : crash variable



Origin Statement

o = null

MUT

cv = o

callee 1

cv.foo()

callee 2

callee 3

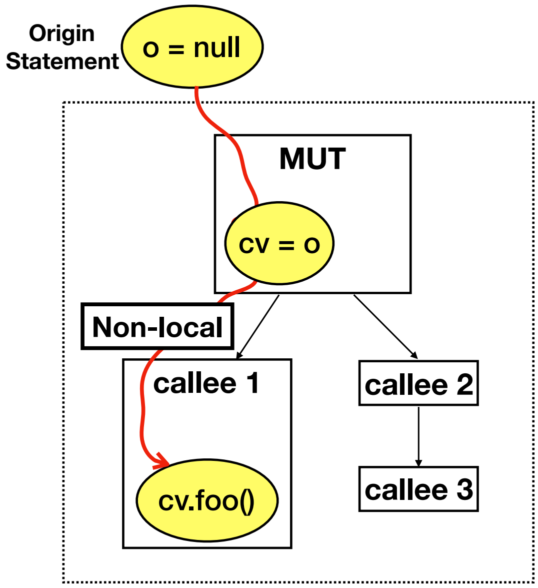
cv : crash variable

Intuition Behind - Non-local

Failure-inducing dataflow is **non-local** to MUT's computation.

Portion of the dataflow is not induced by the MUT's implementation but by the **generated test's logic**.

The chance of violating preconditions is **higher**.



Origin Statement

o = null

MUT

cv = o

Non-local

callee 1

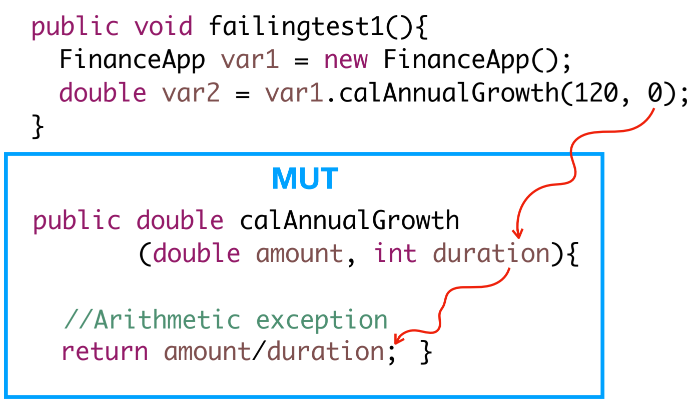
cv.foo()

callee 2

callee 3

cv : crash variable

Example

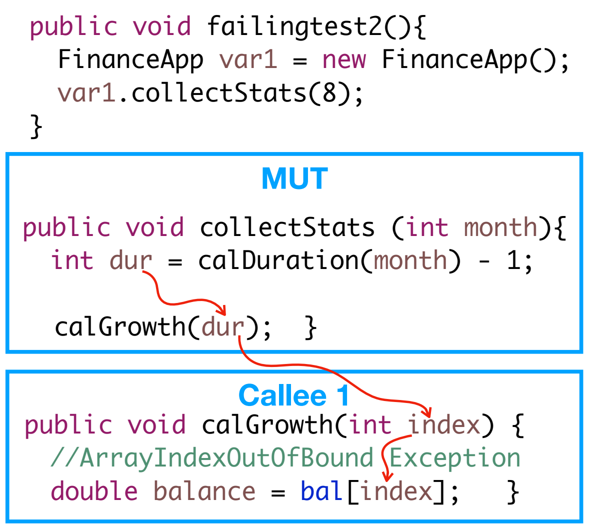


```
public void failingtest1(){
  FinanceApp var1 = new FinanceApp();
  double var2 = var1.calAnnualGrowth(120, 0);
}
```

MUT

```
public double calAnnualGrowth
       (double amount, int duration){

  //Arithmetic exception
  return amount/duration;  }
```



```
public void failingtest2(){
  FinanceApp var1 = new FinanceApp();
  var1.collectStats(8);
}
```

MUT

```
public void collectStats (int month){
  int dur = calDuration(month) - 1;

  calGrowth(dur);  }
```

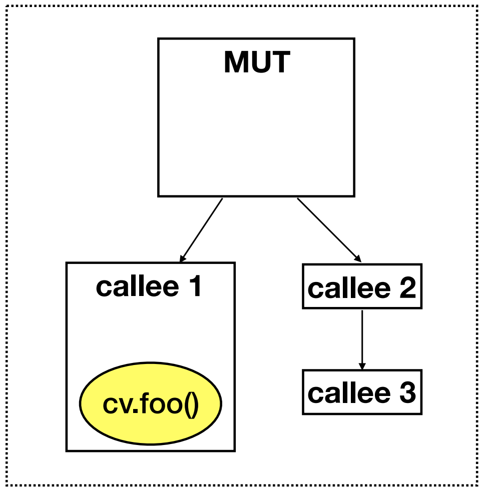
Callee 1

```
public void calGrowth(int index) {
  //ArrayIndexOutOfBound Exception
  double balance = bal[index];    }
```

Non-local

Local

PAF Analysis

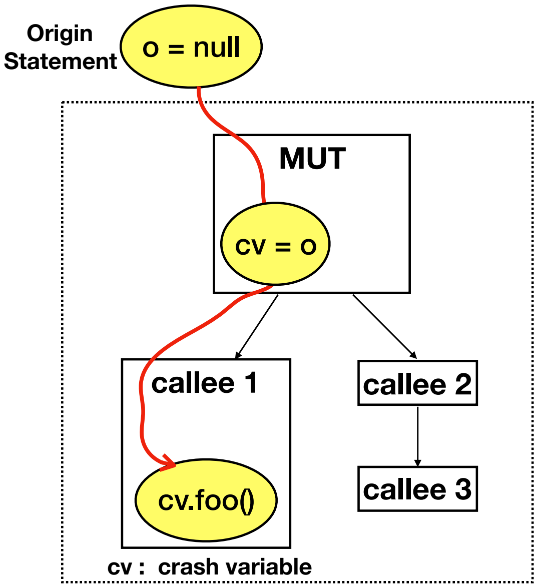


cv : crash variable

Crash variable

Instrument the program to monitor variable accessed right before crash using Soot

PAF Analysis



Crash variable

Instrument the program to monitor variable accessed right before crash using Soot

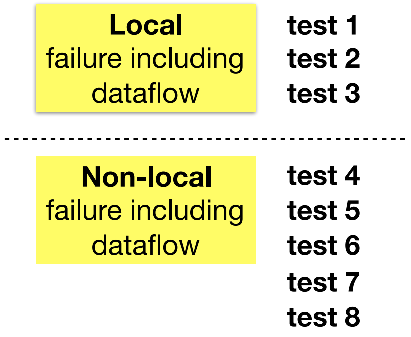
Origin statement

Perform **dynamic interprocedural** data dependance analysis using Soot

Transitively trace backward from the crash statement until reaching non-copy statement

Partitioning Failing Tests

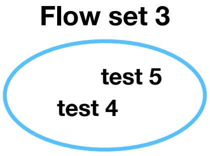
Failure Inducing Dataflow	Implicit Precondition
Local	Not violated (fault-revealing)
Non-local	Violated (non-fault-revealing)

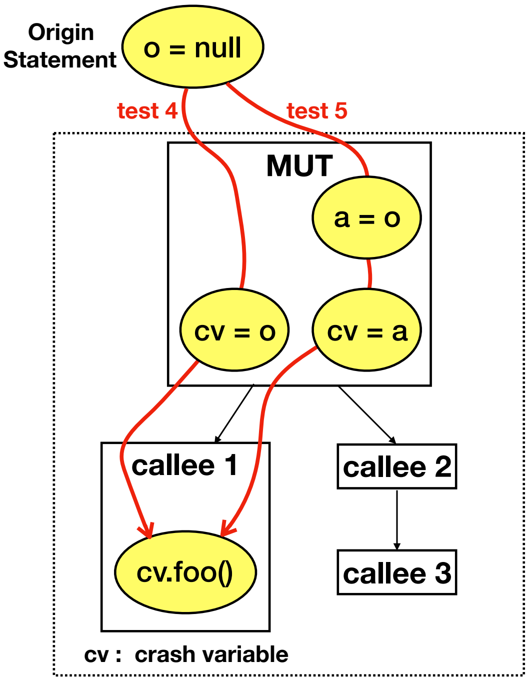


Grouping Failing Tests with Same Failure Cause

Group failing tests into ***flow sets*** sharing the **same**

(origin statement, crash statement, crash variable)

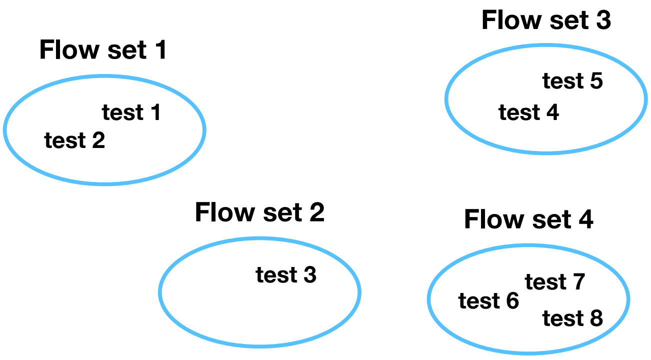




Grouping Failing Tests with Same Failure Cause

Group failing tests into ***flow sets*** sharing the **same**

(origin statement, crash statement, crash variable)



Local failure including dataflow	test 1 test 2	Flow set 1
	test 3	Flow set 2
Non-local failure including dataflow	test 4 test 5	Flow set 3
	test 6 test 7 test 8	Flow set 4

Evaluation

- Integrated PAF into Randoop (v3.1.0)
- Used DUA-Forencis *[Santelices et al. SOAP'13]* to compute interprocedural def-use associations
- Research questions
  - RQ1: Accuracy of partitioning based on locality of failure inducing dataflow
  - RQ2: Accuracy of grouping based on same (origin statement, crash statement, crash variable)
  - RQ3: Effectiveness of prioritization based on precondition violation likelihood

Subjects

FR: Fault-revealing

Subject	Version	Label	Randoop		PAF	
			Failing Test	FR Test	Failing Flow-sets	FR Flow-sets
Ant	1.6.5	Ant1	1086	548	74	6
	1.8.1	Ant2	1462	77	124	7
Collections	2.0	Coll1	402	38	34	1
	2.1	Coll2	256	17	33	1
Ivy	2.2.0	Ivy1	360	1	65	1
	2.4.0	Ivy2	565	0	64	0
Math	2.2	Math1	45	8	22	3
Rhino	1.7.R2	Rhino1	258	41	46	1
	1.7.R3	Rhino2	676	37	152	1
	1.7.R5	Rhino3	660	44	151	1
Average			641	90	82	2.4

Ground Truth of Fault-revealing (FR) Tests

- A failing test is **fault-revealing**
  if it passes in a subsequent version

- A failing test is **non-fault-revealing**
  if it still fails in the latest code, which is at least 18 months old.

  - Rationale behind: “if a bug is introduced to code, the bug will be detected and fixed within few months” *[Ray et al. MSR’15]*

RQ1: Accuracy of Locality Based Partitioning

- Measured **precision and recall** of partitioning fault-revealing tests

- Compared results with JCrasher and Daikon’s false alarm filtering heuristics

  - JCrasher considers exception type and method modifiers

  - Daikon considers dynamic invariants mined from passing executions

    - If invariants relevant to crash variable exist at the entry of MUT, the corresponding failure violates precondition

RQ1: Accuracy of Locality Based Partitioning

Subject	Flow-sets				Individual Tests									
	FR	PAF			FR	PAF			JCrasher			Daikon		
		Actual / Partition	Prec.	Rec.		Actual / Partition	Prec.	Rec.	Actual / Partition	Prec.	Rec.	Actual / Partition	Prec.	Rec.
Ant1	6	2/2	100	33.3	548	20/20	100	3.6	334/392	85.2	60.9	21/174	12.1	3.8
Ant2	7	1/1	100	14.3	77	10/10	100	13.0	13/50	26	16.9	75/330	22.7	97.4
Coll1	1	1/1	100	100	38	38/38	100	100	16/87	18.4	42.1	38/90	42.2	100
Coll2	1	1/1	100	100	17	17/17	100	100	8/77	10.4	47.1	17/34	50	100
Ivy1	1	1/8	12.5	100	1	1/36	2.8	100	0/169	0	0	1/238	0.4	100
Math1	3	3/3	100	100	8	8/8	100	100	5/25	20	62.5	8/45	17.8	100
Rhino1	1	1/2	50	100	41	41/44	93.2	100	0/48	0	0	41/166	24.7	100
Rhino2	1	1/2	50	100	37	37/40	92.5	100	0/138	0	0	37/424	8.7	100
Rhino3	1	1/2	50	100	44	44/46	95.7	100	0/259	0	0	44/495	8.9	100
Total	22	12/24	50	54.5	811	78.8	78.8	26.6	376/15510	24.9	46.4	282/2143	13.2	34.8

RQ1: Accuracy of Locality Based Partitioning

Subject	Flow-sets				Individual Tests									
	FR	PAF			FR	PAF			JCrasher			Daikon		
		Actual / Partition	Prec.	Rec.		Actual / Partition	Prec.	Rec.	Actual / Partition	Prec.	Rec.	Actual / Partition	Prec.	Rec.
Ant1	6	2/2	100	33.3	548	20/20	100	3.6	334/392	85.2	60.9	21/174	12.1	3.8
Ant2	7	1/1	100	14.3	77	10/10	100	13.0	13/50	26	16.9	75/330	22.7	97.4
Coll1	1	1/1	100	100	38	38/38	100	100	16/87	18.4	42.1	38/90	42.2	100
Coll2	1	1/1	100	100	17	17/17	100	100	8/77	10.4	47.1	17/34	50	100
Ivy1	1	1/8	12.5	100	1	1/36	2.8	100	0/169	0	0	1/238	0.4	100
Math1	3	3/3	100	100	8	8/8	100	100	5/25	20	62.5	8/45	17.8	100
Rhino1	1	1/2	50	100	41	41/44	93.2	100	0/48	0	0	41/166	24.7	100
Rhino2	1	1/2	50	100	37	37/40	92.5	100	0/138	0	0	37/424	8.7	100
Rhino3	1	1/2	50	100	44	44/46	95.7	100	0/259	0	0	44/495	8.9	100
Total	22	12/24	50	54.5	811	78.8	78.8	26.6	376/15510	24.9	46.4	282/2143	13.2	34.8

RQ1: Accuracy of Locality Based Partitioning

Subject	Flow-sets				Individual Tests									
	FR	PAF			FR	PAF			JCrasher			Daikon		
		Actual / Partition	Prec.	Rec.		Actual / Partition	Prec.	Rec.	Actual / Partition	Prec.	Rec.	Actual / Partition	Prec.	Rec.
Ant1	6	2/2	100	33.3	548	20/20	100	3.6	334/392	85.2	60.9	21/174	12.1	3.8
Ant2	7	1/1	100	14.3	77	10/10	100	13.0	13/50	26	16.9	75/330	22.7	97.4
Coll1	1	1/1	100	100	38	38/38	100	100	16/87	18.4	42.1	38/90	42.2	100
Coll2	1	1/1	100	100	17	17/17	100	100	8/77	10.4	47.1	17/34	50	100
Ivy1	1	1/8	12.5	100	1	1/36	2.8	100	0/169	0	0	1/238	0.4	100
Math1	3	3/3	100	100	8	8/8	100	100	5/25	20	62.5	8/45	17.8	100
Rhino1	1	1/2	50	100	41	41/44	93.2	100	0/48	0	0	41/166	24.7	100
Rhino2	1	1/2	50	100	37	37/40	92.5	100	0/138	0	0	37/424	8.7	100
Rhino3	1	1/2	50	100	44	44/46	95.7	100	0/259	0	0	44/495	8.9	100
Total	22	12/24	50	54.5	811	78.8	78.8	26.6	376/15510	24.9	46.4	282/2143	13.2	34.8

RQ1: Accuracy of Locality Based Partitioning

Subject	Flow-sets				Individual Tests									
	FR	PAF			FR	PAF			JCrasher			Daikon		
		Actual / Partition	Prec.	Rec.		Actual / Partition	Prec.	Rec.	Actual / Partition	Prec.	Rec.	Actual / Partition	Prec.	Rec.
Ant1	6	2/2	100	33.3	548	20/20	100	3.6	334/392	85.2	60.9	21/174	12.1	3.8
Ant2	7	1/1	100	14.3	77	10/10	100	13.0	13/50	26	16.9	75/330	22.7	97.4
Coll1	1	1/1	100	100	38	38/38	100	100	16/87	18.4	42.1	38/90	42.2	100
Coll2	1	1/1	100	100	17	17/17	100	100	8/77	10.4	47.1	17/34	50	100
Ivy1	1	1/8	12.5	100	1	1/36	2.8	100	0/169	0	0	1/238	0.4	100
Math1	3	3/3	100	100	8	8/8	100	100	5/25	20	62.5	8/45	17.8	100
Rhino1	1	1/2	50	100	41	41/44	93.2	100	0/48	0	0	41/166	24.7	100
Rhino2	1	1/2	50	100	37	37/40	92.5	100	0/138	0	0	37/424	8.7	100
Rhino3	1	1/2	50	100	44	44/46	95.7	100	0/259	0	0	44/495	8.9	100
Total	22	12/24	50	54.5	811	78.8	78.8	26.6	376/15510	24.9	46.4	282/2143	13.2	34.8

RQ1: Accuracy of Locality Based Partitioning

Subject	Flow-sets				Individual Tests									
	FR	PAF			FR	PAF			JCrasher			Daikon		
		Actual / Partition	Prec.	Rec.		Actual / Partition	Prec.	Rec.	Actual / Partition	Prec.	Rec.	Actual / Partition	Prec.	Rec.
Ant1	6	2/2	100	33.3	548	20/20	100	3.6	334/392	85.2	60.9	21/174	12.1	3.8
Ant2	7	1/1	100	14.3	77	10/10	100	13.0	13/50	26	16.9	75/330	22.7	97.4
Coll1	1	1/1	100	100	38	38/38	100	100	16/87	18.4	42.1	38/90	42.2	100
Coll2	1	1/1	100	100	17	17/17	100	100	8/77	10.4	47.1	17/34	50	100
Ivy1	1	1/8	12.5	100	1	1/36	2.8	100	0/169	0	0	1/238	0.4	100
Math1	3	3/3	100	100	8	8/8	100	100	5/25	20	62.5	8/45	17.8	100
Rhino1	1	1/2	50	100	41	41/44	93.2	100	0/48	0	0	41/166	24.7	100
Rhino2	1	1/2	50	100	37	37/40	92.5	100	0/138	0	0	37/424	8.7	100
Rhino3	1	1/2	50	100	44	44/46	95.7	100	0/259	0	0	44/495	8.9	100
Total	22	12/24	50	54.5	811	78.8	78.8	26.6	376/15510	24.9	46.4	282/2143	13.2	34.8

RQ1: Accuracy of Locality Based Partitioning

Subject	Flow-sets				Individual Tests									
	FR	PAF			FR	PAF			JCrasher			Daikon		
		Actual / Partition	Prec.	Rec.		Actual / Partition	Prec.	Rec.	Actual / Partition	Prec.	Rec.	Actual / Partition	Prec.	Rec.
Ant1	6	2/2	100	33.3	548	20/20	100	3.6	334/392	85.2	60.9	21/174	12.1	3.8
Ant2	7	1/1	100	14.3	77	10/10	100	13.0	13/50	26	16.9	75/330	22.7	97.4
Coll1	1	1/1	100	100	38	38/38	100	100	16/87	18.4	42.1	38/90	42.2	100
Coll2	1	1/1	100	100	17	17/17	100	100	8/77	10.4	47.1	17/34	50	100
Ivy1	1	1/8	12.5	100	1	1/36	2.8	100	0/169	0	0	1/238	0.4	100
Math1	3	3/3	100	100	8	8/8	100	100	5/25	20	62.5	8/45	17.8	100
Rhino1	1	1/2	50	100	41	41/44	93.2	100	0/48	0	0	41/166	24.7	100
Rhino2	1	1/2	50	100	37	37/40	92.5	100	0/138	0	0	37/424	8.7	100
Rhino3	1	1/2	50	100	44	44/46	95.7	100	0/259	0	0	44/495	8.9	100
Total	22	12/24	50	54.5	811	216/274	78.8	26.6	376/15510	24.9	46.4	282/2143	13.2	34.8

RQ1: Accuracy of Locality Based Partitioning

Subject	Flow-sets				Individual Tests									
	FR	PAF			FR	PAF			JCrasher			Daikon		
		Actual / Partition	Prec.	Rec.		Actual / Partition	Prec.	Rec.	Actual / Partition	Prec.	Rec.	Actual / Partition	Prec.	Rec.
Ant1	6	2/2	100	33.3	548	20/20	100	3.6	334/392	85.2	60.9	21/174	12.1	3.8
Ant2	7	1/1	100	14.3	77	10/10	100	13.0	13/50	26	16.9	75/330	22.7	97.4
Coll1	1	1/1	100	100	38	38/38	100	100	16/87	18.4	42.1	38/90	42.2	100
Coll2	1	1/1	100	100	17	17/17	100	100	8/77	10.4	47.1	17/34	50	100
Ivy1	1	1/8	12.5	100	1	1/36	2.8	100	0/169	0	0	1/238	0.4	100
Math1	3	3/3	100	100	8	8/8	100	100	5/25	20	62.5	8/45	17.8	100
Rhino1	1	1/2	50	100	41	41/44	93.2	100	0/48	0	0	41/166	24.7	100
Rhino2	1	1/2	50	100	37	37/40	92.5	100	0/138	0	0	37/424	8.7	100
Rhino3	1	1/2	50	100	44	44/46	95.7	100	0/259	0	0	44/495	8.9	100
Total	22	12/24	50	54.5	811	216/274	78.8	26.6	376/1510	24.9	46.4	282/2143	13.2	34.8

RQ2: Accuracy of Grouping by Same Failure Cause

- Evaluated with only fault-revealing tests
- Ground truth of optimal groups
  - Multiple failing tests are induced by the same fault
    if they failed before a patch and passed after the patch
- Compared with
  ReBucket (stack trace based) and MSeer (execution trace based)

RQ2: Accuracy of Grouping by Same Failure Cause

Subject	F-Measure			# of Groups			
	PAF	ReBucket	MSeer	PAF	ReBucket	MSeer	Ground Truth
Ant1	1	0.708	0.708	5	14	7	5
Ant2	0.987	0.979	0.711	7	7	4	6
Coll1	1	0.593	0.733	1	23	2	1
Coll2	1	0.64	0.786	1	10	2	1
Ivy1	1	1	1	1	1	1	1
Math1	0.925	1	0.583	3	2	3	2
Rhino1	1	0.048	0.988	1	41	2	1
Rhino2	1	0.053	0.857	1	37	3	1
Rhino3	1	0.044	0.842	1	4	2	1

RQ2: Accuracy of Grouping by Same Failure Cause

Subject	F-Measure			# of Groups			
	PAF	ReBucket	MSeer	PAF	ReBucket	MSeer	Ground Truth
Ant1	1	0.708	0.708	5	14	7	5
Ant2	0.987	0.979	0.711	7	7	4	6
Coll1	1	0.593	0.733	1	23	2	1
Coll2	1	0.64	0.786	1	10	2	1
Ivy1	1	1	1	1	1	1	1
Math1	0.925	1	0.583	3	2	3	2
Rhino1	1	0.048	0.988	1	41	2	1
Rhino2	1	0.053	0.857	1	37	3	1
Rhino3	1	0.044	0.842	1	4	2	1

RQ2: Accuracy of Grouping by Same Failure Cause

Subject	F-Measure			# of Groups			
	PAF	ReBucket	MSeer	PAF	ReBucket	MSeer	Ground Truth
Ant1	1	0.708	0.708	5	14	7	5
Ant2	0.987	0.979	0.711	7	7	4	6
Coll1	1	0.593	0.733	1	23	2	1
Coll2	1	0.64	0.786	1	10	2	1
Ivy1	1	1	1	1	1	1	1
Math1	0.925	1	0.583	3	2	3	2
Rhino1	1	0.048	0.988	1	41	2	1
Rhino2	1	0.053	0.857	1	37	3	1
Rhino3	1	0.044	0.842	1	4	2	1

Conclusion

- Infers violations of preconditions and prioritizes based on likelihood of violations
- Groups failures with the same failure cause
- Achieves high accuracy on inferring and grouping
- Enhances usability of test generation tools and saves debugging cost