Exhaustive Exploration of the Failure-oblivious Computing Search Space

Thomas Durieux, Youssef Hamadi, Zhongxing Yu, Benoit Baudry and Martin Monperrus

May 2, 2018

Inria & University of Lille & KTH

Table of contents

1. Introduction
2. Failure-oblivious Computing
3. Empirical Study

Introduction

Invalid Execution

Traditional Assumption

Invalid execution → **unsafe** to continue the execution

[1] Rinard et al. *Enhancing Availability and Security Through Failure-Oblivious Computing*

Invalid Execution

Traditional Assumption

Invalid execution → **unsafe** to continue the execution

Failure-oblivious Assumption

Invalid execution → continue the execution can be a **safe** solution [1]

[1] Rinard et al. *Enhancing Availability and Security Through Failure-Oblivious Computing*

Failure-oblivious Computing

Failure-oblivious Computing

Failure-oblivious Computing

Runtime strategy that allows a program to continue the execution instead of crashing or throwing an exception.

For example:

```
+ if (array.length > i) {
    array[i] = 42;
+ }
```

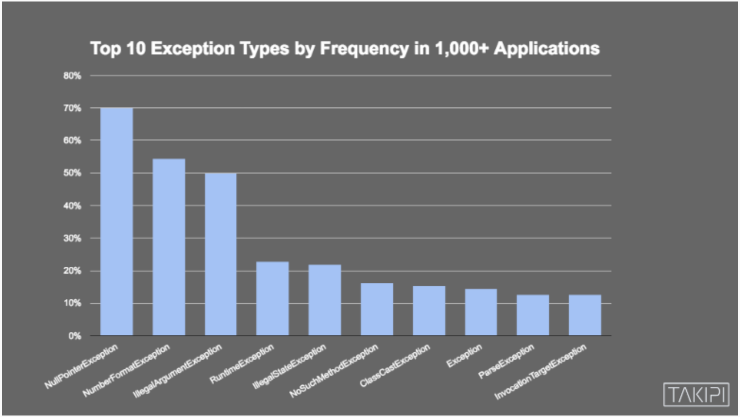
Failure-oblivious Computing

Rinard et al. Failure-oblivious Computing strategies:

- Invalid memory access: ignore and continue
- Invalid array read: read a manufactured value
- Invalid array write: discard the write operation

Failure-oblivious Computing for NPE

Null Pointer Exception (NPE) is the most common failure in production for Java



Failure-oblivious Computing of Null Pointer Exception

```
r = returnNull(); // return null
r.foo(p);
```

Instead of throwing an null pointer exception or terminating → protects the calls

```
  r = returnNull();
+ if (r != null) {
   r.foo(p);
+ }
```

Failure-oblivious Strategies for NPE

Strategies			Description
Replace	reuse		injection of an existing compatible object
	creation		injection of a new object
Skip	line		skip statement
	method	void	return a null or $\emptyset$ to caller
		creation	return a new object to caller
		reuse	return an existing compatible object to caller

Failure-Oblivious Strategies

Replace the null expression

```
+ if (var == null) {
+   anotherVar.foo(p);
+ } else {
    var.foo(p);
+ }
```

```
+ if (var == null) {
+   new Foo().foo(p);
+ } else {
    var.foo(p);
+ }
```

Failure-Oblivious Strategies

Skip the null expression

```
+ if (var == null) {
+   return anotherVar
+ }
  var.foo(p)
```

```
+ if (var == null) {
+   return new Bar();
+ }
  var.foo(p);
```

Failure-oblivious Strategies Injection

Automated injection of the six failure-oblivious strategies at all possible failure locations.

Failure Locations:

- Method call on an expression
- Foreach loop
- Unboxing primitive object to primitive type

Strategy Injection Example

```
Date getLastConnectionDate() {
 Session sess = getUserSession();
 return sess.getLastConnection();
    // NPE1
}
...
HTML.write(getLastConnectionDate()
   .toString()); // NPE2
```

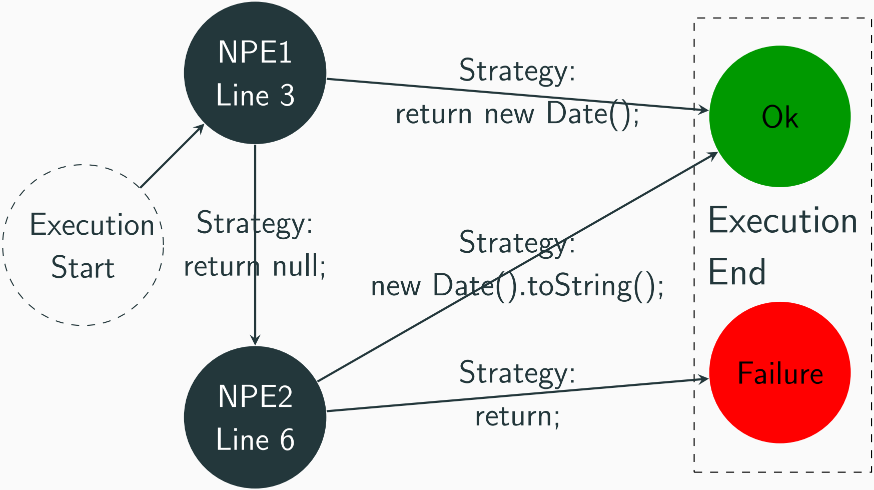
Strategy Injection Example

```
 Date getLastConnectionDate() {
+ try {
   Session sess = getUserSession();
   return
+    failureOblivious(sess)
      .getLastConnection(); // NPE1
+ } catch (SkipMethod e){
+  if (returnVar) return getVar()
+  if (returnNew)
+   return createDate()
+ }
}
```

Example Failure-oblivious Search-space

```
Date getLastConnectionDate() {
 Session sess = getUserSession();
 return sess.getLastConnection();
    // NPE1
}
...
HTML.write(getLastConnectionDate()
   .toString()); // NPE2
```

Example Failure-oblivious Search-space



Search-space of Failure-oblivious Computing

Path of Failure-oblivious Computing

A path in the failure-oblivious search-space graph is the sequence of the **failure-oblivious decisions** that has been taken at each **failure-point**.

Search-space of Failure-oblivious Computing

The search-space of failure-oblivious computing is composed of all the possible paths .

Empirical Study

Research Question 1

What is the size of the failure-oblivious computing search space in real applications?

Benchmark

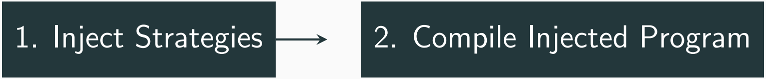
#	Bug ID	LOC	#	Bug ID	LOC
1	Collections-360	21 650	9	Math-1115	90 782
2	Felix-4960	33 057	10	Math-1117	90 794
3	Lang-304	17 277	11	Math-290	38 265
4	Lang-587	17 317	12	Math-305	38 893
5	Lang-703	19 047	13	Math-369	41 082
6	PDFBox-2812	67 294	14	Math-988A	82 442
7	PDFBox-2965	64 375	15	Math-988B	82 443
8	PDFBox-2995	64 821	16	Sling-4982	1 182

16 production null pointer failures.

Exploration of the Failure-oblivious Search Space Protocol

1. Inject Strategies

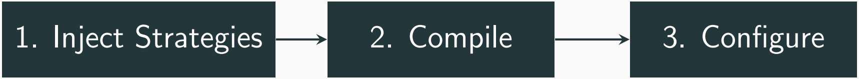
Exploration of the Failure-oblivious Search Space Protocol



1. Inject Strategies

2. Compile Injected Program

Exploration of the Failure-oblivious Search Space Protocol

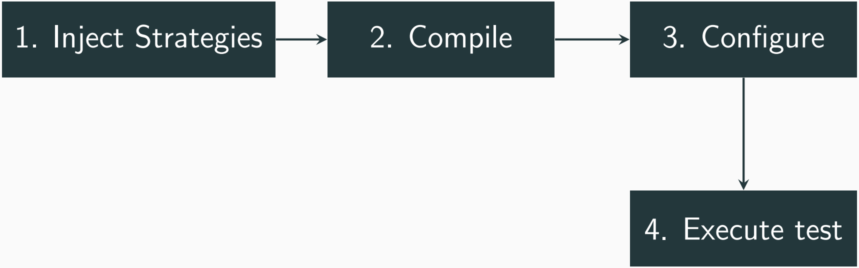


1. Inject Strategies

2. Compile

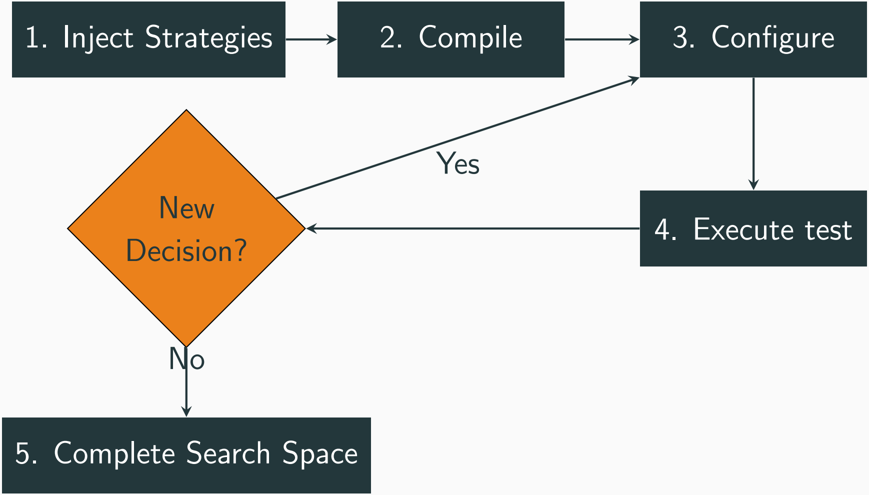
3. Configure

Exploration of the Failure-oblivious Search Space Protocol



1. Inject Strategies
2. Compile
3. Configure
4. Execute test

Exploration of the Failure-oblivious Search Space Protocol



1. Inject Strategies

2. Compile

3. Configure

4. Execute test

New Decision?

Yes

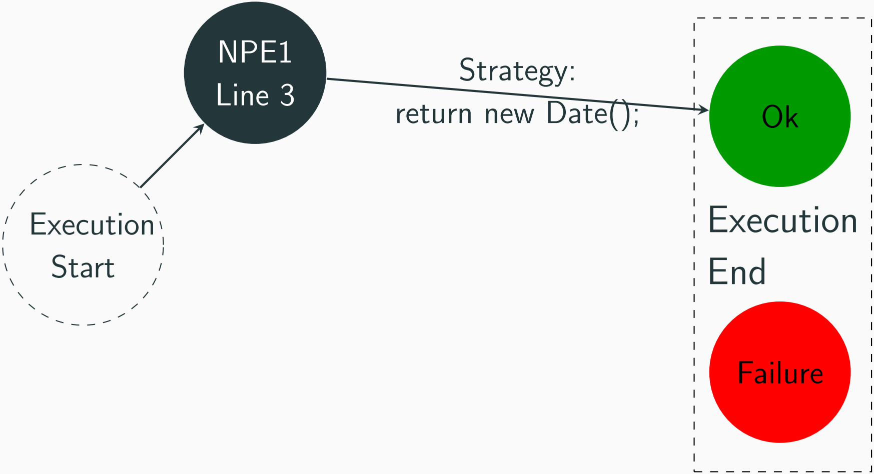
No

5. Complete Search Space

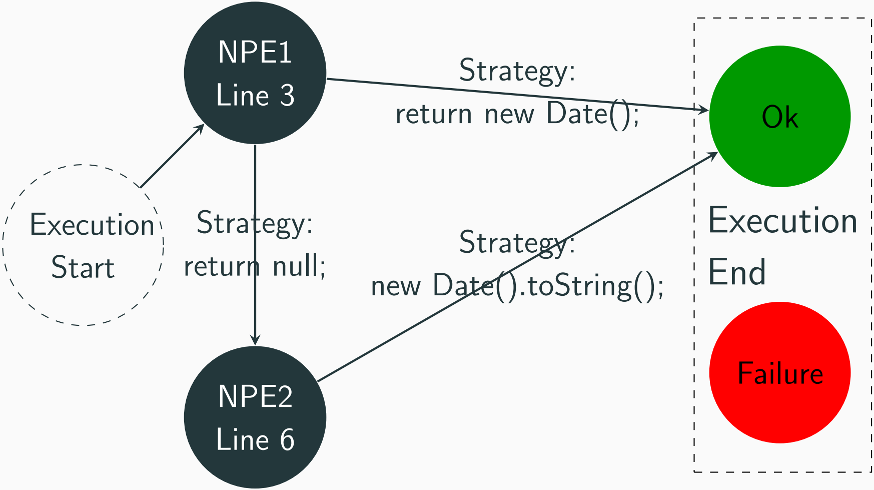
Example Failure-oblivious Search-space

Execution Start
Ok
Execution End
Failure

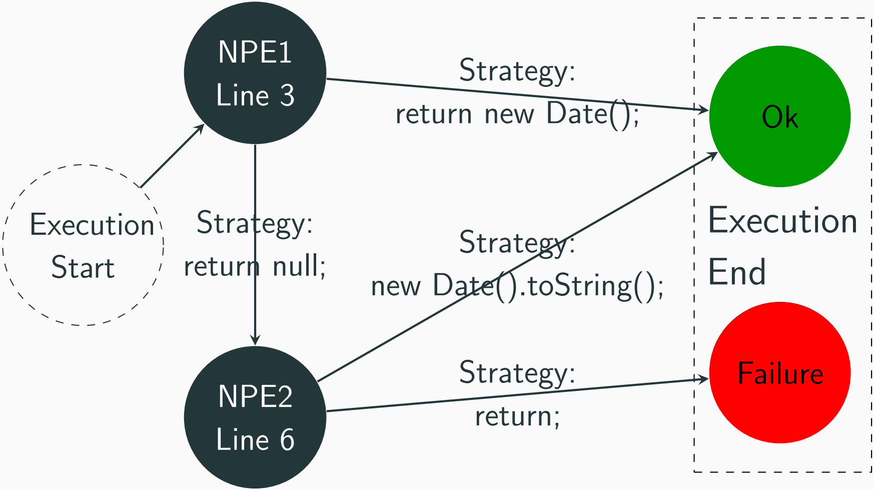
Example Failure-oblivious Search-space



Example Failure-oblivious Search-space



Example Failure-oblivious Search-space

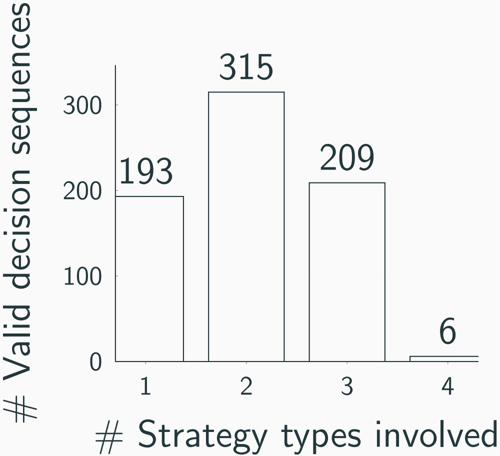


The Size of the Search Space

Bug Id	$\|FP\|$	# Decision	Bug Id	$\|FP\|$	# Decision
Collections-360	2	45	Math-1115	1	5
Felix-4960	1	10	Math-1117	21	51 785
Lang-304	1	7	Math-290	1	14
Lang-587	1	28	Math-305	1	4
Lang-703	4	459	Math-369	2	14
Pdfbox-2812	8	294	Math-988A	3	576
Pdfbox-2965	1	4	Math-988B	1	32
Pdfbox-2995	1	5	Sling-4982	2	16

Failure-oblivious computing involves sequences of failure-oblivious decisions (8/16 no cherry picking).

# of Strategies by Decision Sequence



Support multi strategy is required

Research Question 2

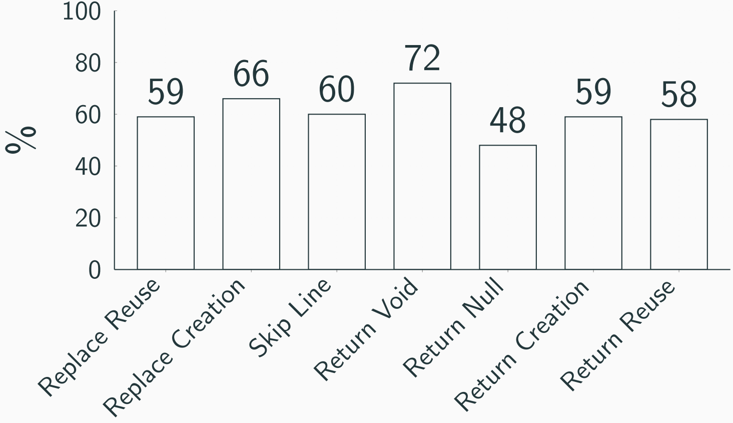
What is the proportion of valid failure-oblivious decision sequences?

Number of Valid Decision Sequence

Bug Id	# Valid	%	Bug Id	# Valid	%
Collections-360	16	35,5%	Math-1115	5	100%
Felix-4960	4	40%	Math-1117	7 708	14,9%
Lang-304	6	35,5%	Math-290	4	28,6%
Lang-587	1	3,0%	Math-305	3	75%
Lang-703	130	28,3%	Math-369	0	0%
Pdfbox-2812	168	57,1%	Math-988A	383	66,5%
Pdfbox-2965	3	75%	Math-988B	17	53,1%
Pdfbox-2995	1	20%	Sling-4982	11	68,7%

There are much more than one valid failure-oblivious sequence.

Success Rate Failure-oblivious Strategies



There is no obvious better strategy

Future Work

Study the impact of the context in the selection of the strategy.

New failure-oblivious tool that has heuristic to select the best strategy depending on the context. Example: if we are in a loop maybe skip line is better than stop the execution of the method.

Conclusion

Take Away

Failure-oblivious computing involves sequences of failure-oblivious decisions.

There exists a failure-oblivious computing search-space which is largely unexplored.

Open-science:

`https://github.com/Spirals-Team/runtime-repair-experiments`