String Test Data Generation for Java Programs

Miaomiao Wang, Baoquan Cui, Jiwei Yan, Jun Yan, Jian Zhang

Technology Center of Software Engineering, Institute of Software, Chinese Academy of Sciences, Beijing, China
State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
University of Chinese Academy of Sciences, Beijing, China

ISSRE 2022

Overview

- Introduction
  - String Test Data Generation
  - Existing Work
- Preliminary
  - String API Usage
  - Motivating Example
- Our Approach
  - API-Regex Mapping
  - String API Invocation Sequence Extraction
  - Regex Generation
- Evaluation
  - DataSets and Experimental Results
  - Case Study
- Conclusion

1.1 String Test Data Generation

- String operations in Java programs are mostly performed by calling string APIs.
- We need to understand the semantics of the API to generate suitable test data.

```
1    // str is a method parameter
2    String[] splitArray = str.split("#");
3    String splitStr = splitArray[2];
4    String subStr = splitStr.substring(3, 8);
5    if (subStr.equals("class")) {
6       ... //True branch: omitted
7    } else { ... //False branch: omitted }
```

complex semantics for single API

multiple string API combination

1.2 Existing Work

- **SMT Solvers**([1]-[8]):
  - SMT-LIB[9] cannot support the split(...) method and the lastIndexOf(...) method
  - The automatic conversion from the string API combinations to SMT formats is not supported.

- **test case generation**
  - EvoSuite[10]
    - Seeding Strategy [11] : It supports single and simple string APIs to seed data and mutate
  - Randoop[12]
    - feedback-directed random test generation
    - It does not intentionally generate string-related data.

[1] Norn: An SMT Solver for String Constraints
[2] String constraints with concatenation and transducers solved efficiently
[3] Towards Constraint Logic Programming over Strings for Test Data Generation
[4] On Solving Word Equations Using SAT.
[5] Z3str4: A Solver for Theories over Strings
[6] cvc5: A Versatile and Industrial-Strength SMT Solver
[7] A decision procedure for path feasibility of string manipulating programs with integer data type
[8] Solving string constraints with regex-dependent functions through transducers with priorities and variables
[9] https://smtlib.cs.uiowa.edu/theories-UnicodeStrings.shtml
[10] Evosuite: automatic test suite generation for object-oriented software
[11] Seeding strategies in search-based unit test generation
[12] Randoop: feedback-directed random testing for java

Overview

- Introduction
  - String Test Data Generation
  - Existing Work
- **Preliminary**
  - String API Usage
  - Motivating Example
- Our Approach
  - API-Regex Mapping
  - String API Invocation Sequence Extraction
  - Regex Generation
- Evaluation
  - DataSets and Experimental Results
  - Case Study
- Conclusion

2.1 String API Usage

TABLE II
STRING API USAGE

API	NO.	In Path		In Branch	
		ct.(Total)	ct.(Dedup)	ct.(Total)	ct.(Dedup)
Single	1	11,536	29	8,304	25
Combination	2	**873**	**84**	**753**	**75**
	3	159	58	132	49
	4	32	7	27	6
	5	4	2	3	1
Sum		12,604	180	9,219	156

`Single`: a single string API invocation.
`Combination`: combination of multiple API invocations.
`NO`: the number of API combinations, a single API involves only one API.
`ct.(Total)`: the usage the API combination is used.
`ct.(Dedup)`: the count the API combination is used after deduplication.

- The combinations of string APIs make up 8.5% (1068/12604).
- 73.1% (9219/12604) of string API invocations or their combinations have affected the branches in the programs.

2.2 Motivating Example

Listing 1. A Motivating Example

```
1  // str is a method parameter
2  String[] splitArray = str.split("#");
3  String splitStr = splitArray[2];
4  String subStr = splitStr.substring(3, 8);
5  if (subStr.equals("class")) {
6     ... //True branch: omitted
7  } else { ... //False branch: omitted }
```

ArrayIndexOutOfBoundsException

StringIndexOutOfBoundsException

- **SMT solvers, EvoSuite and Randoop** can not generate suitable test data to cover the true branch.
- The purpose of this paper is to generate regular expressions:
  - cover the branches
  - trigger the potential exception proactively

Overview

- Introduction
  - String Test Data Generation
  - Existing Work
- Preliminary
  - String API Usage
  - Motivating Example
- **Our Approach**
  - API-Regex Mapping
  - String API Invocation Sequence Extraction
  - Regex Generation
- Evaluation
  - DataSets and Experimental Results
  - Case Study
- Conclusion

Our Approach

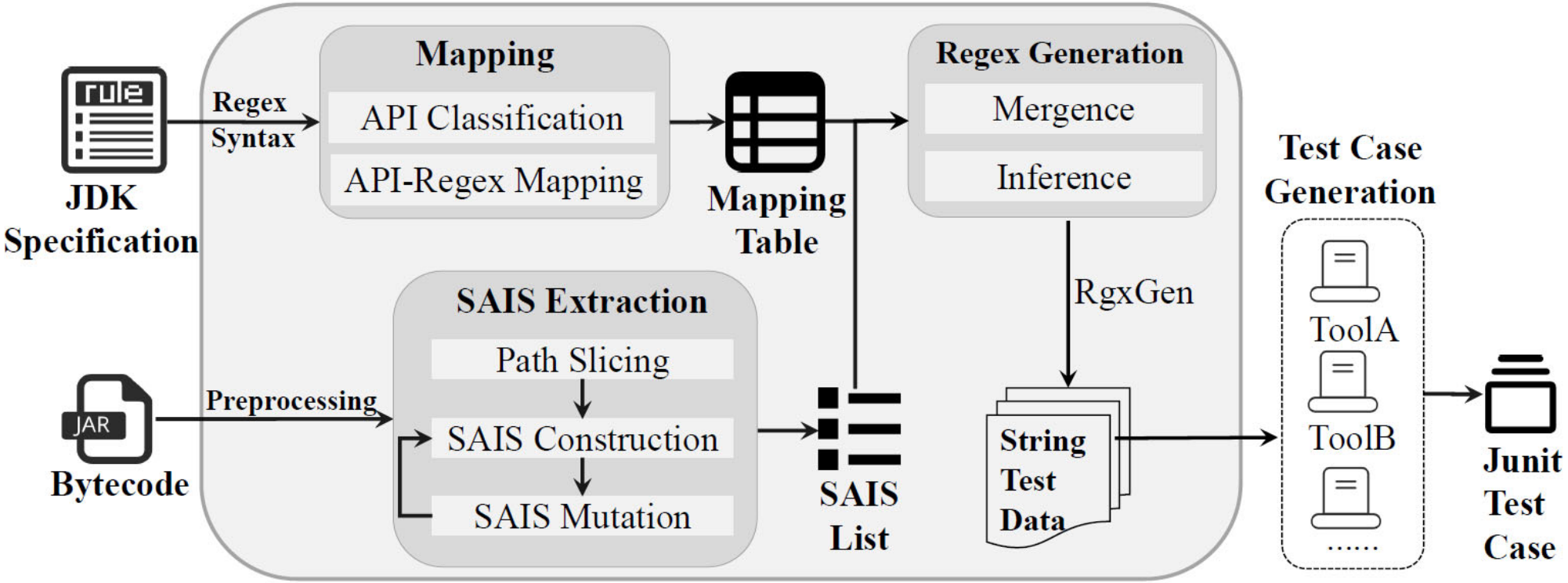


Fig. 1. Overview of Our Approach.

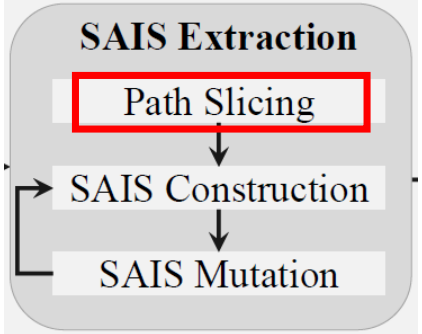
3.1 SAIS Extraction

Algorithm 1: Slice

```
Input: StrPP, condExpr
Output: snippet
containsStringParameter = False;
snippet = {};
variableSet = {};
stmt = condExpr;
snippet.insertAtFirst(stmt);
variableSet.addAll(getVariables(stmt));
while stmt != NULL do
    for each v ∈ variableSet do
        if stmt.isAssignStatementOf(v) then
            variableSet.remove(v);
            snippet.insertAtFirst(stmt);
            variables = getVariables(stmt);
            variableSet.addAll(variables);
            if isStringParaLocalization(stmt) then
                containsStringParameter = true;
            break;
    if variableSet.isEmpty() then
        break;
    stmt = StrPP.getPredecessorOf(stmt);
if containsStringParameter == False then
    snippet = NULL;
return snippet
```

Definition III-A1. (StringParaPath)

A ***StringParaPath*** is the path in a method which contains at least one string API invocation statement on the local variable of the string parameter.



→ **StringParaPath**

3.1 SAIS Extraction

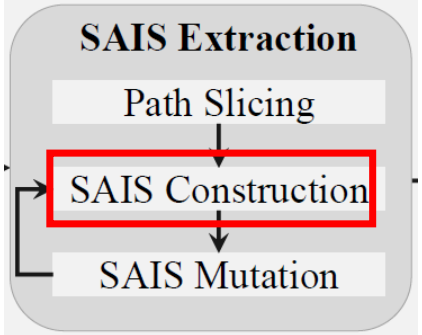
Definition III-A2. (String API Invocation Sequence (SAIS))
A ***SAIS*** is a 2-tuple

SAIS = <CondExpr, Trajectory>

where the ***CondExpr*** is a conditional expression in the ***StringParaPath***, and the Trajectory is a k-length list of ordered pairs

$<(stmt_1, map_1), (stmt_2, map_2), ..., (stmt_k, map_k)>$

where the $stmt_i$ in each pair is a string API invocation statement related to the conditional expression and the map_i maps the variables in the $stmt_i$ to their values.




```
1  // str is a method parameter
2  String[] splitArray = str.split("#");
3  String splitStr = splitArray[2];
4  String subStr = splitStr.substring(3, 8);
5  if (subStr.equals("class")) {
6     ... //True branch: omitted
7  } else { ... //False branch: omitted }
```

SAIS	CondExpr:	eq==True	
	Trajectory:	$splitStr$=str.split(v_4)[i_1]	{v_4:"#", i_1:2}
		$substr$=$splitStr$.substring(v_2,v_3)	{v_2:3,v_3:8}
		eq=$subStr$.equals(v_1),	{v_1:"class"}

3.1 SAIS Extraction

```
1    // str is a method parameter
2    String[] splitArray = str.split("#");
3    String splitStr = splitArray[2];
4    String subStr = splitStr.substring(3, 8);
5    if (subStr.equals("class")) {
6       ... //True branch: omitted
7    } else { ... //False branch: omitted }
```

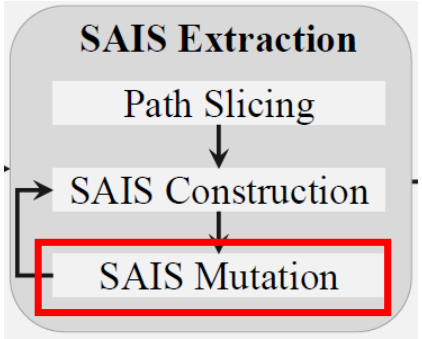


TABLE III
A SAIS AND ITS VARIANT

SAIS	CondExpr:	eq==True	
	Trajectory:	$splitStr$=str.split(v_4)[i_1]	{v_4:"#", i_1:2}
		$substr$=$splitStr$.substring(v_2,v_3)	{v_2:3,v_3:8}
		eq=$subStr$.equals(v_1),	{v_1:"class"}
SAIS$_N$	+CondExpr:	$length$<3	
	Trajectory:	$splitStr$=str.split(v_4)[i_1]	{v_4:"#", i_1:2}
		+ $length$=str.split(v_4).length	{v_4:"#"}

3.2 API-Regex Mapping

- **Terminated API.**
  - The API whose return type is char, int, boolean, char[] or byte[], that is, primitive type or its array type, is considered as a terminated API (Total: 15). In general, its return value is used directly in the conditional expression

- **Non-terminated API.**
  - The API whose return type is String, CharSequence or String[] is considered as a non-terminated API (Total: 33). Their return values cannot appear directly in conditional expressions.

3.2 API-Regex Mapping

Definition (RegexWrapper)

A ***RegexWrapper*** is a 4-tuple

- R is the regular expression according to the string API and the expression on its return value;
- L is the length of the string that the R allows to merge and –1 indicates that there is no limit for the length;
- S is the a special regular expression with which to replace during inference if the any character regular expression is in the suffix of the R;
- L_{min} is the minimum length of a string corresponding to the R;

Definition (API-Regex Mapping)

A ***API-Regex Mapping*** is a 2-tuple

$$\mathcal{M} = \langle APIPair, RegexWrapper \rangle$$

where ***APIPair*** is a pair including the string API invocation and its related conditional expression, and ***RegexWrapper*** is defined before.

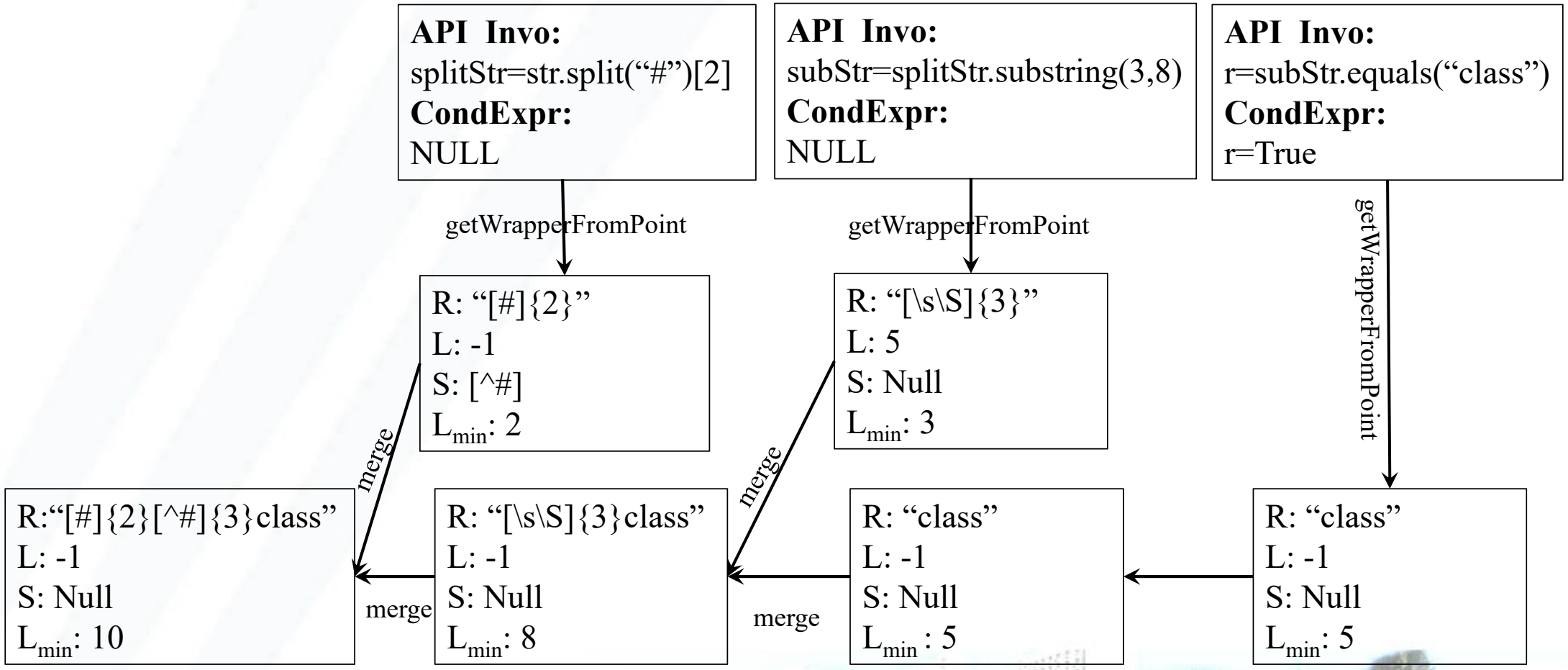
3.2 API-Regex Mapping

TABLE IV
PARTIAL API-REGEX MAPPING

APIPair		RegexWrapper			
API Invo. (caller: *str* | *arr*)	CondExpr	R	L	S	L_{min}
r=str.startswith(v)	r==T	v[\s\S]*	–1	NULL	Len(v)
r=str.length()	r==l	[\s\S]{l}	–1	NULL	l
r=str.substring(i_1,i_2)	NULL	[\s\S]{i_1}	i_2-i_1	NULL	i_1
r=str.split(v)[i]	NULL	[v]{i}	–1	[^v]	i
r=arr.length	r >i	[\s\S]{i+1,}	–1	NULL	i+1

https://github.com/suoyi123wang/JustinStr

3.3 Regex Generation



3.3 Regex Generation

TABLE I
COMPARISON ON JAVA TOOLS GENERATING STRING DATA

Tool	E-prone Regular Expression			
	[\s\S]{1,4}	[^a]{1,3}	(?i)AbC	[\Q*\E]{1,3}
xeger [4]	✗	✓	✗	✗
Generex [39]	✓	✓	✗	E
MutRex [41]	✓	✓	✗	✗
bfgex [22]	✗	✗	E	✗
RgxGen [19]	✓	✓	✗	✓
random-string [9]	E	E	✗	E

✓: the generated string matches the regex; ✗: not matching; *E*: an exception occurs during generation.

[#]{2}[^#]{3}class

RgxGen: "##abcclass"

Cover the true branch (line 6)

```
1    // str is a method parameter
2    String[] splitArray = str.split("#");
3    String splitStr = splitArray[2];
4    String subStr = splitStr.substring(3, 8);
5    if (subStr.equals("class")) {
6      ... //True branch: omitted
7    } else { ... //False branch: omitted }
```

Overview

- Introduction
  - String Test Data Generation
  - Existing Work
- Preliminary
  - String API Usage
  - Motivating Example
- Our Approach
  - API-Regex Mapping
  - String API Invocation Sequence Extraction
  - Regex Generation
- **Evaluation**
  - DataSets and Experimental Results
  - Case Study
- Conclusion

4.1 DataSets

TABLE V
STATISTICS OF THREE DATASETS

Program	#Cls	#LOC	M_1	M_2	#RE
23 Assginments	406	9667	45	42	98
289 Solutions	289	4529	217	183	677
closure-compiler-20220202	8,162	218,939	194	153	475
commons-lang3-3.12.0	505	9,600	73	39	211
commons-io-2.11.0	240	5,544	21	17	81
mysql-connector-java-8.0.29	1,345	42,726	56	44	163
poi-ooxml-5.2.2	1,529	47,176	45	31	84
OpenJDK-8u292	25,574	697,752	1,513	1,250	6,697

4.2 Experimental Results

- **RQ1: How effectively does Justin-Str characterize the input string parameters?**

Length	Branch	Single (%)	Combination (%)			
			2	3	4	5
L=10,20 (30,40,50)	True	100.00	100.00	100.00	100.00	100.00
L=10	False	99.77	99.41	99.33	98.09	100.00
L=20	False	99.55	99.14	99.02	99.11	100.00
L=30	False	99.35	98.84	98.72	99.35	99.99
L=40	False	99.14	98.47	98.50	99.49	99.99
L=50	False	98.94	98.20	98.18	99.65	99.98

- **85x5 = 425 methods,**
  - **Each one ends with a branch (T/F)**

4.2 Experimental Results

- **RQ2: Improvement of the branch coverage**

Program	Seeds	$M_{Cov100\%}$	$M_{Cov}(\%)$	Max_{Cov}	Ave_{Cov}
closure-compiler	972	58	+18(19%)	+50%	+21%
commons-lang3	441	10	+10(34%)	+57%	+11%
commons-io	128	4	+5(38%)	+43%	+17%
mysql-connector-java	301	16	+5(18%)	+34%	+11%
poi-ooxml	125	10	+2(10%)	+50%	+28%

- **+22%(40/186) of the method**
- **up to +57%, and +17% on average**

4.2 Experimental Results

- **RQ3: Bug Finding**

TABLE VIII
COMPARISON OF BUGS FOUND BY JUSTINSTR AND EVOSUITE

Program	JustinStr	Common	Δ	Time (s)	
				JustinStr	EvoSuite
Assignments	38	27	11	339	1627
Solutions	72	60	12	14	12,453
closure-compiler	33	21	12	72	2,394
commons-io	11	1	10	16	80
commons-lang3	15	3	12	15	709
mysql-connector-java	7	3	4	23	306
poi-ooxml	3	0	3	40	-
OpenJDK	17	0	17	10,228	-
Total	196	115	81	-	-

TABLE IX
BUGS CONFIRMED OR FIXED BY THE JDK DEVELOPERS

Type:	ArrayIndexOutOfBoundsException (2)
Issue ID:	I4MWI1 (Commit ID), 8279422
Type:	StringIndexOutOfBoundsException (14)
Issue ID:	8278186, 8279128, 8279129, 8279198, 8279218, 8279336, 8279341, 8279342, 8279362, 8279423, **21212bd18(Commit ID), 8279424, **411a404a9(Commit ID), **8baba7d11(Commit ID)
Type:	Infinite Loop (1), **Issue ID**: 8278993

4.2 Experimental Results

- **RQ3: Bug Finding (Order)**

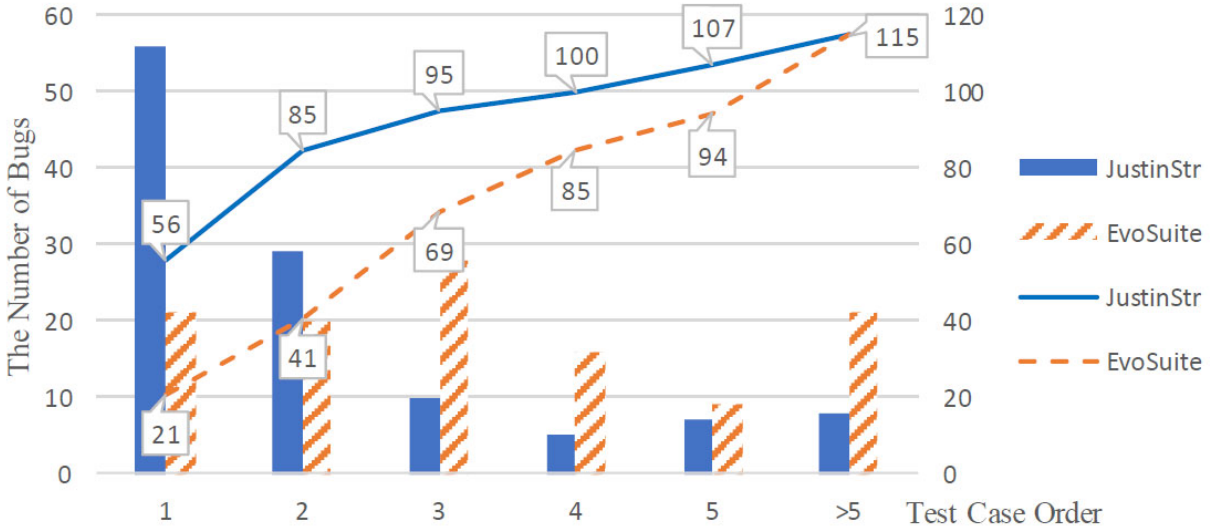


Fig. 3. Comparison on Number of Bugs Found by `JustinStr` and `EvoSuite` under Test Case Order. Bars represent the number of bugs triggered by the test case at current order and poly-lines represent the number of accumulated bugs triggered by the test cases before and within the order.

- The first two test cases in JustinStr can trigger 74% of the bugs, while EvoSuite only triggers 36%.

4.3 Case Study

Listing 2. Case Study in JDK

```
1  // Case 1: StringIndexOutOfBoundsException
2  public static String parseIdFromSameDocumentURI(String
       uri) {
3      if (uri.length() == 0) {
4          return null;
5      }
6      String id = uri.substring(1);
7      if (id != null && id.startsWith("xpointer(id(")) {
8          int i1 = id.indexOf('\'');
9          int i2 = id.indexOf('\'', i1+1);
10         id = id.substring(i1+1, i2);
11     }
12     return id;
13 }
14 //Case 2: Infinit Loop
15 static public String sansArrayInfo (String name) {
16     int index = name.indexOf ('[');
17     if (index >= 0) {
18         String array = name.substring (index);
19         name = name.substring (0, index);
20         while (!array.equals ("")) {
21             name = name + "[]";
22             array = array.substring(array.indexOf(']') + 1);
23         }
24     }
25     return name;
26 }
```

StringIndexOutOfBoundsException
Regex: "[\s\S]{1}xpointer\(id[\s\S]*
String: 0xpointer(id(G6

Infinite loop:
Regex: [[\s\S]*
String: [

https://bugs.java.com/bugdatabase/view_bug.do?bug_id=JDK-8278186
https://bugs.java.com/bugdatabase/view_bug.do?bug_id=JDK-8278993

Overview

- Introduction
  - String Test Data Generation
  - Existing Work
- Preliminary
  - String API Usage
  - Motivating Example
- Our Approach
  - API-Regex Mapping
  - String API Invocation Sequence Extraction
  - Regex Generation
- Evaluation
  - DataSets and Experimental Results
  - Case Study
- **Conclusion**

Conclusion

- **API-Regex Mapping**. We build a mapping from 48 string APIs to regular expressions based on their semantics, which are equivalent or approximate.
- **Inference Algorithm**: We design an inference algorithm that can characterize the string parameter with regular expressions under the combination of its string API invocations.
- **String Test Data Generation Tool**. We developed an automatic tool **JustinStr** to output the string test data for test cases generation.

Future work:

- Convert the string API with the symbolic value into regular expressions.
- Support for converting more string APIs to regular expressions.

Q & A