

Summary

Sink States:0(0×10^0)

Table 1: Sip4J Analysis Summary

Classes	Methods	States	Unreachable clauses	Unreachable states	Possible concurrent methods	Total. no. of method pairs	No. of concurrent method pairs	Percentage of concurrent methods pairs
Fibonacci	4	1	0	0	2	10	3	30
Total Classes=1	4	1	0	0	2	10	3	30

Contents

1	Fibonacci	3
2	Abbreviation	4
3	Annotated version of the input program generated by Sip4J	5

1 Fibonacci

Table 2: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
Fibonacci	✓
computeFibo	✓
main	✓
display	✓

Table 3: State Transition Matrix

	alive
alive	↑

Table 4: Methods Concurrency Matrix

	Fibonacci	computeFibo	main	display
Fibonacci	⌘	⌘	⌘	⌘
computeFibo	⌘		⌘	
main	⌘	⌘	⌘	⌘
display	⌘		⌘	

2 Abbreviation

Table 5: Used Abbreviation

Symbol	Meaning
✓	requires clause of the method is satisfiable
✗	requires clause of the method is unsatisfiable
↑	The row-state can be transitioned to the column-state
✕	The row-state cannot be transitioned to the column-state
	The row-method can be possibly executed parallel with the column-method
⋈	The row-method cannot be executed parallel with the column-method

3 Annotated version of the input program generated by Sip4J

```
1 package outputs;
2 import edu.cmu.cs.plural.annot.*;
3
4 @ClassStates({@State(name = "alive")})
5 class Fibonacci {
6   @Perm(ensures="unique(this) in alive")
7   Fibonacci() { }
8
9   @Perm(requires="immutable(this) in alive",
10  ensures="immutable(this) in alive")
11   public Integer computeFibo(Integer num) {
12     return null;
13   }
14
15   @Perm(requires="unique(this) in alive",
16  ensures="unique(this) in alive")
17   void main(String[] args) {
18
19   }
20   @Perm(requires="immutable(this) in alive",
21  ensures="immutable(this) in alive")
22   public void display(Integer num) {
23
24   }
25 }
26 }ENDOFCLASS
```