Summary

Sink States:$0(0 \times 10^0)$

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

Table 2: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
Fibonacci	$\surd$
computeFibo	$\surd$
main	$\surd$
display	$\surd$

Table 3: State Transition Matrix

	alive
alive	$\uparrow$

Table 4: Methods Concurrency Matrix

	Fibonacci	computeFibo	main	display
Fibonacci	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
computeFibo	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$
main	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
display	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$

2 Abbreviation

Table 5: Used Abbreviation

Symbol	Meaning
$\surd$	requires clause of the method is satisfiable
$\times$	requires clause of the method is unsatisfiable
$\uparrow$	The row-state can be transitioned to the column-state
$\times$	The row-state cannot be transitioned to the column-state
$\parallel$	The row-method can be possibly executed parallel with the column-method
$\nparallel$	The row-method cannot be executed parallel with the column-method

3 Annotated version of the input program generated by Sip4J

```
package outputs;
import edu.cmu.cs.plural.annot.*;

@ClassStates({@State(name = "alive")})
class Fibonacci {
@Perm(ensures="unique(this) in alive")
Fibonacci() {   }

@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 public Integer computeFibo(Integer num) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void main(String[] args) {

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 public void display(Integer num) {

}

}ENDOFCLASS
```