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
JGFInstrumentor	13	1	0	0	12	91	12	13
JGFSparseMatmultBench	9	1	0	0	1	45	1	2
SparseMatmult	2	1	0	0	0	3	0	0
JGFTimer	9	1	0	0	3	45	6	13
Total Classes=4	33	4	0	0	16	184	19	10

Contents

1 JGFInstrumentor

Table 2: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
JGFInstrumentor	√
addTimer	√
startTimer	√
stopTimer	√
addOpsToTimer	√
printTimer	√
readTimer	√
resetTimer	√
printperfTimer	√
storeData	√
retrieveData	√
printHeader	√
main	√

Table 3: State Transition Matrix

	alive
alive	↑

Table 4: Methods Concurrency Matrix

	JGFInstrumentor	addTimer	startTimer	stopTimer	addOpsToTimer	printTimer	readTimer	resetTimer	printperfTimer	storeData	retrieveData	printHeader	main
JGFInstrumentor	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
addTimer	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
startTimer	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
stopTimer	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
addOpsToTimer	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
printTimer	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
readTimer	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
resetTimer	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
printperfTimer	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
storeData	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
retrieveData	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
printHeader	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
main	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦

2 JGFSparseMatmultBench

Table 5: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
JGFSparseMatmultBench	√
JGFrun	√
JGFsetsize	√
JGFinitialise	√
RandomVector	√
JGFkernel	√
JGFvalidate	√
JGFtidyup	√
main	√

Table 6: State Transition Matrix

	alive
alive	↑

Table 7: Methods Concurrency Matrix

	JGFSparseMatmultBench	JGFrun	JGFsetsize	JGFinitialise	RandomVector	JGFkernel	JGFvalidate	JGFtidyup	main
JGFSparseMatmultBench	∦	∦	∦	∦	∦	∦	∦	∦	∦
JGFrun	∦	∦	∦	∦	∦	∦	∦	∦	∦
JGFsetsize	∦	∦	∦	∦	∦	∦	∦	∦	∦
JGFinitialise	∦	∦	∦	∦	∦	∦	∦	∦	∦
RandomVector	∦	∦	∦	∦	∦	∦	∦	∦	∦
JGFkernel	∦	∦	∦	∦	∦	∦	∦	∦	∦
JGFvalidate	∦	∦	∦	∦	∦	∦	∥	∦	∦
JGFtidyup	∦	∦	∦	∦	∦	∦	∦	∦	∦
main	∦	∦	∦	∦	∦	∦	∦	∦	∦

3 SparseMatmult

Table 8: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
SparseMatmult	√
test	√

Table 9: State Transition Matrix

	alive
alive	↑

Table 10: Methods Concurrency Matrix

	SparseMatmult	test
SparseMatmult	∦	∦
test	∦	∦

4 JGFTimer

Table 11: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
JGFTimer	$\surd$
start	$\surd$
stop	$\surd$
addops	$\surd$
print	$\surd$
perf	$\surd$
reset	$\surd$
printperf	$\surd$
longprint	$\surd$

Table 12: State Transition Matrix

	alive
alive	$\uparrow$

Table 13: Methods Concurrency Matrix

	JGFTimer	start	stop	addops	print	perf	reset	printperf	longprint
JGFTimer	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
start	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
stop	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
addops	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
print	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
perf	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\parallel$
reset	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
printperf	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\parallel$
longprint	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\parallel$

5 Abbreviation

Table 14: 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

6 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 JGFInstrumentor {
6  @Perm(ensures="unique(this) in alive")
7  JGFInstrumentor() {   }
8
9  @Perm(requires="share(this) in alive",
10 ensures="share(this) in alive")
11   void addTimer(String name) {
12
13 }
14 @Perm(requires="share(this) in alive",
15 ensures="share(this) in alive")
16   void startTimer(String name) {
17
18 }
19 @Perm(requires="share(this) in alive",
20 ensures="share(this) in alive")
21   void stopTimer(String name) {
22
23 }
24 @Perm(requires="share(this) in alive",
25 ensures="share(this) in alive")
26   void addOpsToTimer(String name, double count) {
27
28 }
29 @Perm(requires="share(this) in alive",
30 ensures="share(this) in alive")
31   void printTimer(String name) {
32
33 }
34 @Perm(requires="share(this) in alive",
35 ensures="share(this) in alive")
36   double readTimer(String name) {
37  return 0;
38
39 }
40 @Perm(requires="share(this) in alive",
41 ensures="share(this) in alive")
42   void resetTimer(String name) {
43
44 }
45 @Perm(requires="share(this) in alive",
46 ensures="share(this) in alive")
47   void printperfTimer(String name) {
48
49 }
50 @Perm(requires="share(this) in alive",
51 ensures="share(this) in alive")
52   void storeData(String name, Object obj) {
53
54 }
55 @Perm(requires="share(this) in alive",
56 ensures="share(this) in alive")
57   void retrieveData(String name, Object obj) {
58
59 }
60
61   void printHeader(int section, int size) {
62
63 }
64 @Perm(requires="unique(this) in alive",
65 ensures="unique(this) in alive")
66   void main(String argv[]) {
67
68 }
69
70 }ENDOFCLASS
71
72 @ClassStates({@State(name = "alive")})
73
74 class JGFSparseMatmultBench {
75 @Perm(ensures="unique(this) in alive")
76 JGFSparseMatmultBench() {   }
77
78 @Perm(requires="unique(this) in alive",
```

```
ensures="unique(this) in alive")
 public void JGFrun(int size) {

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void JGFsetsize(int size) {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public void JGFinitialise() {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  double[] RandomVector(int N, java.util.Random R) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void JGFkernel() {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public void JGFvalidate() {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public void JGFtidyup() {

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

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class SparseMatmult {
@Perm(ensures="unique(this) in alive")
SparseMatmult() {   }

@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void test(double y[], double val[], int row[], int col[], double x[], int NUM_ITERATIONS) {

}

}ENDOFCLASS

@ClassStates({@State(name = "alive")})

class JGFTimer {
@Perm(ensures="unique(this) in alive")
JGFTimer() {   }

@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void start() {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void stop() {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void addops(double count) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void print() {
```

```
161  }
162  @Perm(requires="pure(this) in alive",
163  ensures="pure(this) in alive")
164   public double perf() {
165   return 0;

167  }
168  @Perm(requires="share(this) in alive",
169  ensures="share(this) in alive")
170   public void reset() {

172  }
173  @Perm(requires="pure(this) in alive",
174  ensures="pure(this) in alive")
175   public void printperf() {

177  }
178  @Perm(requires="pure(this) in alive",
179  ensures="pure(this) in alive")
180   public void longprint() {

182  }

184  }ENDOFCLASS
```