Summary

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

Table 1: Pulse Analysis Summary

Classes	Methods	States	Unsatisfiable Clauses	Unreachable States	Possible concurrent Methods	Total. no. of pairs	No. of concurrent pairs	Percentage of concurrent Methods
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: Methods Requires Clause Satisfiability

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: Methods Requires Clause Satisfiability

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: Methods Requires Clause Satisfiability

Method	Satisfiability
SparseMatmult	√
test	√

Table 9: State Transition Matrix

	alive
alive	↑

Table 10: Methods Concurrency Matrix

	SparseMatmult	test
SparseMatmult	∦	∦
test	∦	∦

4 JGFTimer

Table 11: Methods Requires Clause Satisfiability

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

```
76  public void JGFinitialise() {
77  }
78  @Perm(requires="full(this) in alive",
79  ensures="full(this) in alive")
80   double[] RandomVector(int N, java.util.Random R) {
81   return null;
82  }
83  @Perm(requires="full(this) in alive",
84  ensures="full(this) in alive")
85  public void JGFkernel() {
86  }
87  @Perm(requires="pure(this) in alive",
88  ensures="pure(this) in alive")
89  public void JGFvalidate() {
90  }
91  @Perm(requires="unique(this) in alive",
92  ensures="unique(this) in alive")
93  public void JGFtidyup() {
94  }
95  @Perm(requires="unique(this) in alive",
96  ensures="unique(this) in alive")
97   void main(String argv[]) {
98  }

100 }ENDOFCLASS

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

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

108 @Perm(requires="full(this) in alive",
109 ensures="full(this) in alive")
110  void test(double y[], double val[], int row[], int col[], double x[], int NUM_ITERATIONS) {
111 }

113 }ENDOFCLASS

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

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

121 @Perm(requires="full(this) in alive",
122 ensures="full(this) in alive")
123 public void start() {
124 }
125 @Perm(requires="full(this) in alive",
126 ensures="full(this) in alive")
127 public void stop() {
128 }
129 @Perm(requires="full(this) in alive",
130 ensures="full(this) in alive")
131 public void addops(double count) {
132 }
133 @Perm(requires="full(this) in alive",
134 ensures="full(this) in alive")
135 public void print() {
136 }
137 @Perm(requires="pure(this) in alive",
138 ensures="pure(this) in alive")
139 public double perf() {
140  return 0;
141 }
142 @Perm(requires="full(this) in alive",
143 ensures="full(this) in alive")
144 public void reset() {
145 }
146 @Perm(requires="pure(this) in alive",
147 ensures="pure(this) in alive")
148 public void printperf() {
149 }
150 @Perm(requires="pure(this) in alive",
151 ensures="pure(this) in alive")
152 public void longprint() {
153 }

155 }ENDOFCLASS
```