

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
JGFTimer	9	1	0	0	3	45	6	13
JGFInstrumentor	13	1	0	0	12	91	12	13
SOR	2	1	0	0	0	3	0	0
JGFSORBenchSizeB	2	1	0	0	0	3	0	0
JGFSORBench	8	1	0	0	1	36	1	3
Total Classes=5	34	5	0	0	16	178	19	11

Contents

1	JGFTimer	3
2	JGFInstrumentor	4
3	SOR	5
4	JGFSORBenchSizeB	6
5	JGFSORBench	7
6	Abbreviation	8
7	Annotated Version of Sequential Java Program generated by Sip4j	9

1 JGFTimer

Table 2: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFTimer	✓
reset	✓
start	✓
stop	✓
addops	✓
perf	✓
longprint	✓
print	✓
printperf	✓

Table 3: State Transition Matrix

	alive
alive	↑

Table 4: Methods Concurrency Matrix

	JGFTimer	reset	start	stop	addops	perf	longprint	print	printperf
JGFTimer	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
reset	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
start	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
stop	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
addops	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
perf	⌘	⌘	⌘	⌘	⌘			⌘	
longprint	⌘	⌘	⌘	⌘	⌘			⌘	
print	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
printperf	⌘	⌘	⌘	⌘	⌘			⌘	

2 JGFInstrumentor

Table 5: Methods Requires Clause Satisfiability

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

Table 6: State Transition Matrix

	alive
alive	↑

Table 7: Methods Concurrency Matrix

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

3 SOR

Table 8: Methods Requires Clause Satisfiability

Method	Satisfiability
SOR	✓
SORrun	✓

Table 9: State Transition Matrix

	alive
alive	↑

Table 10: Methods Concurrency Matrix

	SOR	SORrun
SOR	⧻	⧻
SORrun	⧻	⧻

4 JGFSORBenchSizeB

Table 11: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFSORBenchSizeB	✓
main	✓

Table 12: State Transition Matrix

	alive
alive	↑

Table 13: Methods Concurrency Matrix

	JGFSORBenchSizeB	main
JGFSORBenchSizeB	⌘	⌘
main	⌘	⌘

5 JGFSORBench

Table 14: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFSORBench	✓
JGFrUn	✓
JGFsetsize	✓
JGFinitialise	✓
JGFkernel	✓
RandomMatrix	✓
JGFvalidate	✓
JGFtidyup	✓

Table 15: State Transition Matrix

	alive
alive	↑

Table 16: Methods Concurrency Matrix

	JGFSORBench	JGFrUn	JGFsetsize	JGFinitialise	JGFkernel	RandomMatrix	JGFvalidate	JGFtidyup
JGFSORBench	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
JGFrUn	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
JGFsetsize	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
JGFinitialise	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
JGFkernel	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
RandomMatrix	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
JGFvalidate	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
JGFtidyup	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘

6 Abbreviation

Table 17: 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

7 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 JGFTimer {
6   @Perm(ensures="unique(this) in alive")
7   JGFTimer() { }
8
9   @Perm(requires="full(this) in alive",
10  ensures="full(this) in alive")
11   public void reset() {
12   }
13   @Perm(requires="full(this) in alive",
14  ensures="full(this) in alive")
15   public void start() {
16   }
17   @Perm(requires="full(this) in alive",
18  ensures="full(this) in alive")
19   public void stop() {
20   }
21   @Perm(requires="full(this) in alive",
22  ensures="full(this) in alive")
23   public void addops(double count) {
24   }
25   @Perm(requires="pure(this) in alive",
26  ensures="pure(this) in alive")
27   public double perf() {
28     return 0;
29   }
30   @Perm(requires="pure(this) in alive",
31  ensures="pure(this) in alive")
32   public void longprint() {
33   }
34   @Perm(requires="full(this) in alive",
35  ensures="full(this) in alive")
36   public void print() {
37   }
38   @Perm(requires="pure(this) in alive",
39  ensures="pure(this) in alive")
40   public void printperf() {
41   }
42
43 }ENDOFCLASS
44
45 @ClassStates({@State(name = "alive")})
46
47 class JGFInstrumentor {
48   @Perm(ensures="unique(this) in alive")
49   JGFInstrumentor() { }
50
51   @Perm(requires="full(this) in alive",
52  ensures="full(this) in alive")
53   void addTimer(String name) {
54   }
55   @Perm(requires="full(this) in alive",
56  ensures="full(this) in alive")
57   void addOpsToTimer(String name, double count) {
58   }
59   @Perm(requires="full(this) in alive",
60  ensures="full(this) in alive")
61   void startTimer(String name) {
62   }
63   @Perm(requires="full(this) in alive",
64  ensures="full(this) in alive")
65   void stopTimer(String name) {
66   }
67   @Perm(requires="full(this) in alive",
68  ensures="full(this) in alive")
69   double readTimer(String name) {
70     return 0;
71   }
72   @Perm(requires="full(this) in alive",
73  ensures="full(this) in alive")
74   void resetTimer(String name) {
75   }
76 }
```

```

76 @Perm(requires="full(this) in alive",
77 ensures="full(this) in alive")
78 void printTimer(String name) {
79 }
80 @Perm(requires="full(this) in alive",
81 ensures="full(this) in alive")
82 void printperfTimer(String name) {
83 }
84 @Perm(requires="full(this) in alive",
85 ensures="full(this) in alive")
86 void storeData(String name, Object obj) {
87 }
88 @Perm(requires="full(this) in alive",
89 ensures="full(this) in alive")
90 void retrieveData(String name, Object obj) {
91 }
92
93 void printHeader(int section, int size) {
94 }
95 @Perm(requires="unique(this) in alive",
96 ensures="unique(this) in alive")
97 void main(String argv[]) {
98 }
99
100 }ENDOFCLASS
101
102 @ClassStates({@State(name = "alive")})
103
104 class SOR {
105 @Perm(ensures="unique(this) in alive")
106 SOR() { }
107
108 @Perm(requires="full(this) in alive",
109 ensures="full(this) in alive")
110 void SORrun(int num_iterations, double G[][], double omega) {
111 }
112
113 }ENDOFCLASS
114
115 @ClassStates({@State(name = "alive")})
116
117 class JGFSORBenchSizeB {
118 @Perm(ensures="unique(this) in alive")
119 JGFSORBenchSizeB() { }
120
121 @Perm(requires="unique(this) in alive",
122 ensures="unique(this) in alive")
123 void main(String argv[]) {
124 }
125
126 }ENDOFCLASS
127
128 @ClassStates({@State(name = "alive")})
129
130 class JGFSORBench {
131 @Perm(ensures="unique(this) in alive")
132 JGFSORBench() { }
133
134 @Perm(requires="unique(this) in alive",
135 ensures="unique(this) in alive")
136 public void JGFrun(int size) {
137 }
138 @Perm(requires="full(this) in alive",
139 ensures="full(this) in alive")
140 public void JGFsetsize(int size) {
141 }
142 @Perm(requires="unique(this) in alive",
143 ensures="unique(this) in alive")
144 public void JGFinitialise() {
145 }
146 @Perm(requires="full(this) in alive",
147 ensures="full(this) in alive")
148 public void JGFkernel() {
149 }
150 @Perm(requires="full(this) in alive",
151 ensures="full(this) in alive")
152 double[][] RandomMatrix(int M, int N, java.util.Random R) {
153 return null;
154 }
155 @Perm(requires="pure(this) in alive",
156 ensures="pure(this) in alive")

```

```
157 public void JGFvalidate() {  
158 }  
159 @Perm(requires="unique(this) in alive",  
160 ensures="unique(this) in alive")  
161 public void JGFtidyup() {  
162 }  
164 }ENDOFCLASS
```