

Summary

Sink States:0(0×10^0)

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
JGFMolDynBenchSizeA	2	1	0	0	0	3	0	0
JGFMolDynBench	7	1	0	0	1	28	1	4
md	3	1	0	0	0	6	0	0
particle	6	1	0	0	0	21	0	0
random	3	1	0	0	0	6	0	0
Total Classes=7	43	7	0	0	16	200	19	10

Contents

1	JGFTimer	3
2	JGFInstrumentor	4
3	JGFMolDynBenchSizeA	5
4	JGFMolDynBench	6
5	md	7
6	particle	8
7	random	9
8	Abbreviation	10
9	Annotated Version of Sequential Java Program generated by Sip4j	11

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 JGFMolDynBenchSizeA

Table 8: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFMolDynBenchSizeA	✓
main	✓

Table 9: State Transition Matrix

	alive
alive	↑

Table 10: Methods Concurrency Matrix

	JGFMolDynBenchSizeA	main
JGFMolDynBenchSizeA	⌘	⌘
main	⌘	⌘

4 JGFMolDynBench

Table 11: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFMolDynBench	✓
JGFrun	✓
JGFsetsize	✓
JGFinitialise	✓
JGFapplication	✓
JGFvalidate	✓
JGFtidyup	✓

Table 12: State Transition Matrix

	alive
alive	↑

Table 13: Methods Concurrency Matrix

	JGFMolDynBench	JGFrun	JGFsetsize	JGFinitialise	JGFapplication	JGFvalidate	JGFtidyup
JGFMolDynBench	⌘	⌘	⌘	⌘	⌘	⌘	⌘
JGFrun	⌘	⌘	⌘	⌘	⌘	⌘	⌘
JGFsetsize	⌘	⌘	⌘	⌘	⌘	⌘	⌘
JGFinitialise	⌘	⌘	⌘	⌘	⌘	⌘	⌘
JGFapplication	⌘	⌘	⌘	⌘	⌘	⌘	⌘
JGFvalidate	⌘	⌘	⌘	⌘	⌘	⌘	⌘
JGFtidyup	⌘	⌘	⌘	⌘	⌘	⌘	⌘

5 **md**

Table 14: Methods Requires Clause Satisfiability

Method	Satisfiability
md	✓
initialise	✓
runiters	✓

Table 15: State Transition Matrix

	alive
alive	↑

Table 16: Methods Concurrency Matrix

	md	initialise	runiters
md	⧻	⧻	⧻
initialise	⧻	⧻	⧻
runiters	⧻	⧻	⧻

6 particle

Table 17: Methods Requires Clause Satisfiability

Method	Satisfiability
particle	✓
domove	✓
force	✓
mkekin	✓
velavg	✓
dscal	✓

Table 18: State Transition Matrix

	alive
alive	↑

Table 19: Methods Concurrency Matrix

	particle	domove	force	mkekin	velavg	dscal
particle	⌘	⌘	⌘	⌘	⌘	⌘
domove	⌘	⌘	⌘	⌘	⌘	⌘
force	⌘	⌘	⌘	⌘	⌘	⌘
mkekin	⌘	⌘	⌘	⌘	⌘	⌘
velavg	⌘	⌘	⌘	⌘	⌘	⌘
dscal	⌘	⌘	⌘	⌘	⌘	⌘

7 random

Table 20: Methods Requires Clause Satisfiability

Method	Satisfiability
random	✓
seed	✓
update	✓

Table 21: State Transition Matrix

	alive
alive	↑

Table 22: Methods Concurrency Matrix

	random	seed	update
random	⌘	⌘	⌘
seed	⌘	⌘	⌘
update	⌘	⌘	⌘

8 Abbreviation

Table 23: 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

9 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 JGFMolDynBenchSizeA {
105 @Perm(ensures="unique(this) in alive")
106 JGFMolDynBenchSizeA() { }
107
108 @Perm(requires="unique(this) in alive",
109 ensures="unique(this) in alive")
110 void main(String argv[]) {
111 }
112
113 }ENDOFCLASS
114
115 @ClassStates({@State(name = "alive")})
116
117 class JGFMolDynBench {
118 @Perm(ensures="unique(this) in alive")
119 JGFMolDynBench() { }
120
121 @Perm(requires="unique(this) in alive",
122 ensures="unique(this) in alive")
123 public void JGFrUn(int size) {
124 }
125 @Perm(requires="full(this) in alive",
126 ensures="full(this) in alive")
127 public void JGFsetsize(int size) {
128 }
129 @Perm(requires="unique(this) in alive",
130 ensures="unique(this) in alive")
131 public void JGFinitialise() {
132 }
133 @Perm(requires="full(this) in alive",
134 ensures="full(this) in alive")
135 public void JGFapplication() {
136 }
137 @Perm(requires="pure(this) in alive",
138 ensures="pure(this) in alive")
139 public void JGFvalidate() {
140 }
141 @Perm(requires="unique(this) in alive",
142 ensures="unique(this) in alive")
143 public void JGFtidyup() {
144 }
145
146 }ENDOFCLASS
147
148 @ClassStates({@State(name = "alive")})
149
150 class md {
151 @Perm(ensures="unique(this) in alive")
152 md() { }
153
154 @Perm(requires="unique(this) in alive",
155 ensures="unique(this) in alive")
156 public void initialise() {

```

```

157 }
158 @Perm(requires="full(this) in alive",
159 ensures="full(this) in alive")
160 public void runiters() {
161 }
162
163 }ENDOFCLASS
164
165 @ClassStates({@State(name = "alive")})
166
167 class particle {
168 @Perm(ensures="unique(this) in alive")
169 particle() { }
170
171 @Perm(requires="full(this) in alive",
172 ensures="full(this) in alive")
173 public void domove(double side) {
174 }
175 @Perm(requires="full(this) in alive",
176 ensures="full(this) in alive")
177 public void force(double side, double rcoeff, int mdsz, int x) {
178 }
179 @Perm(requires="full(this) in alive",
180 ensures="full(this) in alive")
181 public double mkekin(double hsq2) {
182     return 0;
183 }
184 @Perm(requires="full(this) in alive",
185 ensures="full(this) in alive")
186 public double velavg(double vaverh, double h) {
187     return 0;
188 }
189 @Perm(requires="full(this) in alive",
190 ensures="full(this) in alive")
191 public void dscal(double sc, int incx) {
192 }
193
194 }ENDOFCLASS
195
196 @ClassStates({@State(name = "alive")})
197
198 class random {
199 @Perm(ensures="unique(this) in alive")
200 random() { }
201
202 @Perm(requires="full(this) in alive",
203 ensures="full(this) in alive")
204 public double seed() {
205     return 0;
206 }
207 @Perm(requires="full(this) in alive",
208 ensures="full(this) in alive")
209 public double update() {
210     return 0;
211 }
212
213 }ENDOFCLASS

```