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
SeriesTest	6	1	0	0	5	21	9	43
JGFTimer	9	1	0	0	3	45	6	13
JGFInstrumentor	13	1	0	0	12	91	12	13
JGFSeriesBenchSizeB	2	1	0	0	0	3	0	0
JGFSeriesBench	7	1	0	0	1	28	1	4
Total Classes=5	37	5	0	0	21	188	28	15

Contents

1 SeriesTest

Table 2: Methods Requires Clause Satisfiability

Method	Satisfiability
SeriesTest	√
buildTestData	√
Do	√
TrapezoidIntegrate	√
thefunction	√
freeTestData	√

Table 3: State Transition Matrix

	alive
alive	↑

Table 4: Methods Concurrency Matrix

	SeriesTest	buildTestData	Do	TrapezoidIntegrate	thefunction	freeTestData
SeriesTest	∦	∦	∦	∦	∦	∦
buildTestData	∦	∦	∦	‖	‖	∦
Do	∦	∦	∦	‖	‖	∦
TrapezoidIntegrate	∦	‖	‖	‖	‖	‖
thefunction	∦	‖	‖	‖	‖	‖
freeTestData	∦	∦	∦	‖	‖	∦

2 JGFTimer

Table 5: Methods Requires Clause Satisfiability

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

Table 6: State Transition Matrix

	alive
alive	$\uparrow$

Table 7: Methods Concurrency Matrix

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

3 JGFInstrumentor

Table 8: Methods Requires Clause Satisfiability

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

Table 9: State Transition Matrix

	alive
alive	↑

Table 10: 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	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦

4 JGFSeriesBenchSizeB

Table 11: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFSeriesBenchSizeB	√
main	√

Table 12: State Transition Matrix

	alive
alive	↑

Table 13: Methods Concurrency Matrix

	JGFSeriesBenchSizeB	main
JGFSeriesBenchSizeB	∦	∦
main	∦	∦

5 JGFSeriesBench

Table 14: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFSeriesBench	$\surd$
JGFrun	$\surd$
JGFsetsize	$\surd$
JGFinitialise	$\surd$
JGFkernel	$\surd$
JGFvalidate	$\surd$
JGFtidyup	$\surd$

Table 15: State Transition Matrix

	alive
alive	$\uparrow$

Table 16: Methods Concurrency Matrix

	JGFSeriesBench	JGFrun	JGFsetsize	JGFinitialise	JGFkernel	JGFvalidate	JGFtidyup
JGFSeriesBench	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
JGFrun	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
JGFsetsize	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
JGFinitialise	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
JGFkernel	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
JGFvalidate	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$
JGFtidyup	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$

6 Abbreviation

Table 17: 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

7 Annotated Version of Sequential Java Program generated by Sip4j

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

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

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 void buildTestData() {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 void Do() {
}

private double TrapezoidIntegrate(double x0, double x1, int nsteps, double omegan, int select) {
 return 0;
}

private double thefunction(double x, double omegan, int select) {
 return 0;
}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 void freeTestData() {
}

}ENDOFCLASS

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

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

@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
public void reset() {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
public void start() {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
public void stop() {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
public void addops(double count) {
}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
public double perf() {
 return 0;
}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
public void longprint() {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
public void print() {
}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
public void printperf() {
}

}ENDOFCLASS

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

```
76 class JGFInstrumentor {
77 @Perm(ensures="unique(this) in alive")
78 JGFInstrumentor() {   }

80 @Perm(requires="full(this) in alive",
81 ensures="full(this) in alive")
82  void addTimer(String name) {
83 }
84 @Perm(requires="full(this) in alive",
85 ensures="full(this) in alive")
86  void addOpsToTimer(String name, double count) {
87 }
88 @Perm(requires="full(this) in alive",
89 ensures="full(this) in alive")
90  void startTimer(String name) {
91 }
92 @Perm(requires="full(this) in alive",
93 ensures="full(this) in alive")
94  void stopTimer(String name) {
95 }
96 @Perm(requires="full(this) in alive",
97 ensures="full(this) in alive")
98  double readTimer(String name) {
99  return 0;
100 }
101 @Perm(requires="full(this) in alive",
102 ensures="full(this) in alive")
103  void resetTimer(String name) {
104 }
105 @Perm(requires="full(this) in alive",
106 ensures="full(this) in alive")
107  void printTimer(String name) {
108 }
109 @Perm(requires="full(this) in alive",
110 ensures="full(this) in alive")
111  void printperfTimer(String name) {
112 }
113 @Perm(requires="full(this) in alive",
114 ensures="full(this) in alive")
115  void storeData(String name, Object obj) {
116 }
117 @Perm(requires="full(this) in alive",
118 ensures="full(this) in alive")
119  void retrieveData(String name, Object obj) {
120 }

122  void printHeader(int section, int size) {
123 }
124 @Perm(requires="unique(this) in alive",
125 ensures="unique(this) in alive")
126  void main(String argv[]) {
127 }

129 }ENDOFCLASS

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

133 class JGFSeriesBenchSizeB {
134 @Perm(ensures="unique(this) in alive")
135 JGFSeriesBenchSizeB() {   }

137 @Perm(requires="unique(this) in alive",
138 ensures="unique(this) in alive")
139  void main(String argv[]) {
140 }

142 }ENDOFCLASS

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

146 class JGFSeriesBench {
147 @Perm(ensures="unique(this) in alive")
148 JGFSeriesBench() {   }

150 @Perm(requires="unique(this) in alive",
151 ensures="unique(this) in alive")
152 public void JGFrun(int size) {
153 }
154 @Perm(requires="full(this) in alive",
155 ensures="full(this) in alive")
156 public void JGFsetsize(int size) {
```

```
157 }
158 @Perm(requires="unique(this) in alive",
159 ensures="unique(this) in alive")
160 public void JGFinitialise() {
161 }
162 @Perm(requires="full(this) in alive",
163 ensures="full(this) in alive")
164 public void JGFkernel() {
165 }
166 @Perm(requires="pure(this) in alive",
167 ensures="pure(this) in alive")
168 public void JGFvalidate() {
169 }
170 @Perm(requires="unique(this) in alive",
171 ensures="unique(this) in alive")
172 public void JGFtidyup() {
173 }

175 }ENDOFCLASS
```