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
JGFSeriesBenchSizeB	2	1	0	0	1	3	1	33
JGFInstrumentor	3	1	0	0	2	6	2	33
JGFSeriesBench	6	1	0	0	2	21	2	10
JGFTimer	3	1	0	0	2	6	2	33
Total Classes=5	20	5	0	0	12	57	16	28

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 JGFSeriesBenchSizeB

Table 5: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFSeriesBenchSizeB	√
main	√

Table 6: State Transition Matrix

	alive
alive	↑

Table 7: Methods Concurrency Matrix

	JGFSeriesBenchSizeB	main
JGFSeriesBenchSizeB	∦	∦
main	∦	∥

3 JGFInstrumentor

Table 8: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFInstrumentor	√
printHeader	√
printTimer	√

Table 9: State Transition Matrix

	alive
alive	↑

Table 10: Methods Concurrency Matrix

	JGFInstrumentor	printHeader	printTimer
JGFInstrumentor	∦	∦	∦
printHeader	∦	∥	∥
printTimer	∦	∥	∦

4 JGFSeriesBench

Table 11: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFSeriesBench	√
JGFrun	√
JGFinitialise	√
JGFvalidate	√
JGFtidyup	√
JGFsetsize	√

Table 12: State Transition Matrix

	alive
alive	↑

Table 13: Methods Concurrency Matrix

	JGFSeriesBench	JGFrun	JGFinitialise	JGFvalidate	JGFtidyup	JGFsetsize
JGFSeriesBench	∦	∦	∦	∦	∦	∦
JGFrun	∦	∦	∦	∦	∦	∦
JGFinitialise	∦	∦	∦	∦	∦	∦
JGFvalidate	∦	∦	∦	‖	∦	‖
JGFtidyup	∦	∦	∦	∦	∦	∦
JGFsetsize	∦	∦	∦	‖	∦	∦

5 JGFTimer

Table 14: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFTimer	$\surd$
print	$\surd$
perf	$\surd$

Table 15: State Transition Matrix

	alive
alive	$\uparrow$

Table 16: Methods Concurrency Matrix

	JGFTimer	print	perf
JGFTimer	$\nparallel$	$\nparallel$	$\nparallel$
print	$\nparallel$	$\nparallel$	$\parallel$
perf	$\nparallel$	$\parallel$	$\parallel$

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

```
1  package outputs;
2  import edu.cmu.cs.plural.annot.*;
3
4  @ClassStates({@State(name = "alive")})
5  class SeriesTest {
6  @Perm(ensures="unique(this) in alive")
7  SeriesTest() {   }
8
9  @Perm(requires="unique(this) in alive",
10 ensures="unique(this) in alive")
11  void buildTestData() {
12 }
13 @Perm(requires="full(this) in alive",
14 ensures="full(this) in alive")
15  void Do() {
16 }
17
18 private double TrapezoidIntegrate(double x0, double x1, int nsteps, double omegan, int select) {
19  return 0;
20 }
21
22 private double thefunction(double x, double omegan, int select) {
23  return 0;
24 }
25 @Perm(requires="unique(this) in alive",
26 ensures="unique(this) in alive")
27  void freeTestData() {
28 }
29
30 }ENDOFCLASS
31
32 @ClassStates({@State(name = "alive")})
33
34 class JGFSeriesBenchSizeB {
35 @Perm(ensures="unique(this) in alive")
36 JGFSeriesBenchSizeB() {   }
37
38 @Perm(requires="none(this) in alive",
39 ensures="unique(this) in alive")
40  void main(String argv[]) {
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
52  void printHeader(int section, int size) {
53 }
54 @Perm(requires="full(this) in alive",
55 ensures="full(this) in alive")
56  void printTimer(String name) {
57 }
58
59 }ENDOFCLASS
60
61 @ClassStates({@State(name = "alive")})
62
63 class JGFSeriesBench {
64 @Perm(ensures="unique(this) in alive")
65 JGFSeriesBench() {   }
66
67 @Perm(requires="unique(this) in alive",
68 ensures="unique(this) in alive")
69 public void JGFrun(int size) {
70 }
71 @Perm(requires="unique(this) in alive",
72 ensures="unique(this) in alive")
73 public void JGFinitialise() {
74 }
75 @Perm(requires="pure(this) in alive",
```

```
76  ensures="pure(this) in alive")
77  public void JGFvalidate() {
78  }
79  @Perm(requires="unique(this) in alive",
80  ensures="unique(this) in alive")
81  public void JGFtidyup() {
82  }
83  @Perm(requires="full(this) in alive",
84  ensures="full(this) in alive")
85  public void JGFsetsize(int size) {
86  }

88  }ENDOFCLASS

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

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

96  @Perm(requires="full(this) in alive",
97  ensures="full(this) in alive")
98  public void print() {
99  }
100 @Perm(requires="pure(this) in alive",
101 ensures="pure(this) in alive")
102 public double perf() {
103  return 0;
104 }

106 }ENDOFCLASS
```