Summary

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

Table 1: Sip4J Analysis Summary

Classes	Methods	States	Unreachable clauses	Unreachable states	Possible concurrent methods	Total. no. of method pairs	No. of concurrent method pairs	Percentage of concurrent methods pairs
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

Table 2: Method's Satisfiability(Code Reachabiity Analysis

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: Method's Satisfiability(Code Reachabiity Analysis

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: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
JGFMolDynBenchSizeA	√
main	√

Table 9: State Transition Matrix

	alive
alive	↑

Table 10: Methods Concurrency Matrix

	JGFMolDynBenchSizeA	main
JGFMolDynBenchSizeA	∦	∦
main	∦	∦

4 JGFMolDynBench

Table 11: Method's Satisfiability(Code Reachabiity Analysis

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: Method's Satisfiability(Code Reachabiity Analysis

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: Method's Satisfiability(Code Reachabiity Analysis

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: Method's Satisfiability(Code Reachabiity Analysis

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
$\surd$	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

9 Annotated version of the input 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="share(this) in alive",
10 ensures="share(this) in alive")
11  public void reset() {
12
13 }
14 @Perm(requires="share(this) in alive",
15 ensures="share(this) in alive")
16  public void start() {
17
18 }
19 @Perm(requires="share(this) in alive",
20 ensures="share(this) in alive")
21  public void stop() {
22
23 }
24 @Perm(requires="share(this) in alive",
25 ensures="share(this) in alive")
26  public void addops(double count) {
27
28 }
29 @Perm(requires="pure(this) in alive",
30 ensures="pure(this) in alive")
31  public double perf() {
32  return 0;
33
34 }
35 @Perm(requires="pure(this) in alive",
36 ensures="pure(this) in alive")
37  public void longprint() {
38
39 }
40 @Perm(requires="share(this) in alive",
41 ensures="share(this) in alive")
42  public void print() {
43
44 }
45 @Perm(requires="pure(this) in alive",
46 ensures="pure(this) in alive")
47  public void printperf() {
48
49 }
50
51 }ENDOFCLASS
52
53 @ClassStates({@State(name = "alive")})
54
55 class JGFInstrumentor {
56 @Perm(ensures="unique(this) in alive")
57 JGFInstrumentor() {   }
58
59 @Perm(requires="share(this) in alive",
60 ensures="share(this) in alive")
61   void addTimer(String name) {
62
63 }
64 @Perm(requires="share(this) in alive",
65 ensures="share(this) in alive")
66   void addOpsToTimer(String name, double count) {
67
68 }
69 @Perm(requires="share(this) in alive",
70 ensures="share(this) in alive")
71   void startTimer(String name) {
72
73 }
74 @Perm(requires="share(this) in alive",
75 ensures="share(this) in alive")
76   void stopTimer(String name) {
77
78 }
```

```
79  @Perm(requires="share(this) in alive",
80  ensures="share(this) in alive")
81    double readTimer(String name) {
82   return 0;

84  }
85  @Perm(requires="share(this) in alive",
86  ensures="share(this) in alive")
87    void resetTimer(String name) {

89  }
90  @Perm(requires="share(this) in alive",
91  ensures="share(this) in alive")
92    void printTimer(String name) {

94  }
95  @Perm(requires="share(this) in alive",
96  ensures="share(this) in alive")
97    void printperfTimer(String name) {

99  }
100 @Perm(requires="share(this) in alive",
101 ensures="share(this) in alive")
102   void storeData(String name, Object obj) {

104 }
105 @Perm(requires="share(this) in alive",
106 ensures="share(this) in alive")
107   void retrieveData(String name, Object obj) {

109 }

111   void printHeader(int section, int size) {

113 }
114 @Perm(requires="unique(this) in alive",
115 ensures="unique(this) in alive")
116   void main(String argv[]) {

118 }

120 }ENDOFCLASS

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

124 class JGFMolDynBenchSizeA {
125 @Perm(ensures="unique(this) in alive")
126 JGFMolDynBenchSizeA() {   }

128 @Perm(requires="unique(this) in alive",
129 ensures="unique(this) in alive")
130   void main(String argv[]) {

132 }

134 }ENDOFCLASS

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

138 class JGFMolDynBench {
139 @Perm(ensures="unique(this) in alive")
140 JGFMolDynBench() {   }

142 @Perm(requires="unique(this) in alive",
143 ensures="unique(this) in alive")
144  public void JGFrun(int size) {

146 }
147 @Perm(requires="share(this) in alive",
148 ensures="share(this) in alive")
149  public void JGFsetsize(int size) {

151 }
152 @Perm(requires="unique(this) in alive",
153 ensures="unique(this) in alive")
154  public void JGFinitialise() {

156 }
157 @Perm(requires="share(this) in alive",
158 ensures="share(this) in alive")
159  public void JGFapplication() {
```

```
}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public void JGFvalidate() {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public void JGFtidyup() {

}

}ENDOFCLASS

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

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

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public void initialise() {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void runiters() {

}

}ENDOFCLASS

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

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

@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void domove(double side) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void force(double side, double rcoff, int mdsize, int x) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public double mkekin(double hsq2) {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public double velavg(double vaverh, double h) {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void dscal(double sc, int incx) {

}

}ENDOFCLASS

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

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

@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public double seed() {
 return 0;
```

```
241 }
242 @Perm(requires="share(this) in alive",
243 ensures="share(this) in alive")
244  public double update() {
245  return 0;

247 }

249 }ENDOFCLASS
```