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
JGFEulerBenchSizeA	2	1	0	0	0	3	0	0
JGFEulerBench	7	1	0	0	1	28	1	4
Tunnel	11	1	0	0	0	66	0	0
Statevector	6	1	0	0	0	21	0	0
Vector2	3	1	0	0	2	6	3	50
JGFTimer	9	1	0	0	3	45	6	13
JGFInstrumentor	13	1	0	0	12	91	12	13
Total Classes=7	51	7	0	0	18	260	22	8

Contents

1 JGFEulerBenchSizeA

Table 2: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
JGFEulerBenchSizeA	√
main	√

Table 3: State Transition Matrix

	alive
alive	↑

Table 4: Methods Concurrency Matrix

	JGFEulerBenchSizeA	main
JGFEulerBenchSizeA	∦	∦
main	∦	∦

2 JGFEulerBench

Table 5: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
JGFEulerBench	$\surd$
JGFrun	$\surd$
JGFsetsize	$\surd$
JGFinitialise	$\surd$
JGFapplication	$\surd$
JGFvalidate	$\surd$
JGFtidyup	$\surd$

Table 6: State Transition Matrix

	alive
alive	$\uparrow$

Table 7: Methods Concurrency Matrix

	JGFEulerBench	JGFrun	JGFsetsize	JGFinitialise	JGFapplication	JGFvalidate	JGFtidyup
JGFEulerBench	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
JGFrun	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
JGFsetsize	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
JGFinitialise	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
JGFapplication	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
JGFvalidate	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$
JGFtidyup	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$

3 Tunnel

Table 8: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
Tunnel	√
initialise	√
runiters	√
doIteration	√
calculateDummyCells	√
calculateDeltaT	√
calculateDamping	√
calculateF	√
calculateG	√
calculateR	√
calculateStateVar	√

Table 9: State Transition Matrix

	alive
alive	↑

Table 10: Methods Concurrency Matrix

	Tunnel	initialise	runiters	doIteration	calculateDummyCells	calculateDeltaT	calculateDamping	calculateF	calculateG	calculateR	calculateStateVar
Tunnel	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
initialise	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
runiters	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
doIteration	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
calculateDummyCells	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
calculateDeltaT	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
calculateDamping	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
calculateF	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
calculateG	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
calculateR	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
calculateStateVar	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦

4 Statevector

Table 11: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
Statevector	√
svect	√
amvect	√
avect	√
mvect	√
smvect	√

Table 12: State Transition Matrix

	alive
alive	↑

Table 13: Methods Concurrency Matrix

	Statevector	svect	amvect	avect	mvect	smvect
Statevector	∦	∦	∦	∦	∦	∦
svect	∦	∦	∦	∦	∦	∦
amvect	∦	∦	∦	∦	∦	∦
avect	∦	∦	∦	∦	∦	∦
mvect	∦	∦	∦	∦	∦	∦
smvect	∦	∦	∦	∦	∦	∦

5 Vector2

Table 14: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
Vector2	√
magnitude	√
dot	√

Table 15: State Transition Matrix

	alive
alive	↑

Table 16: Methods Concurrency Matrix

	Vector2	magnitude	dot
Vector2	∦	∦	∦
magnitude	∦	∥	∥
dot	∦	∥	∥

6 JGFTimer

Table 17: Method's Satisfiability(Code Reachabiity Analysis

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

Table 18: State Transition Matrix

	alive
alive	↑

Table 19: Methods Concurrency Matrix

	JGFTimer	reset	start	stop	addops	perf	longprint	print	printperf
JGFTimer	∦	∦	∦	∦	∦	∦	∦	∦	∦
reset	∦	∦	∦	∦	∦	∦	∦	∦	∦
start	∦	∦	∦	∦	∦	∦	∦	∦	∦
stop	∦	∦	∦	∦	∦	∦	∦	∦	∦
addops	∦	∦	∦	∦	∦	∦	∦	∦	∦
perf	∦	∦	∦	∦	∦	‖	‖	∦	‖
longprint	∦	∦	∦	∦	∦	‖	‖	∦	‖
print	∦	∦	∦	∦	∦	∦	∦	∦	∦
printperf	∦	∦	∦	∦	∦	‖	‖	∦	‖

7 JGFInstrumentor

Table 20: Method's Satisfiability(Code Reachabiity Analysis

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

Table 21: State Transition Matrix

	alive
alive	↑

Table 22: 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	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦

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.*;

4  @ClassStates({@State(name = "alive")})
5  class JGFEulerBenchSizeA {
6  @Perm(ensures="unique(this) in alive")
7  JGFEulerBenchSizeA() {   }

9  @Perm(requires="unique(this) in alive",
10 ensures="unique(this) in alive")
11   void main(String argv[]) {

13 }

15 }ENDOFCLASS

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

19 class JGFEulerBench {
20 @Perm(ensures="unique(this) in alive")
21 JGFEulerBench() {   }

23 @Perm(requires="unique(this) in alive",
24 ensures="unique(this) in alive")
25  public void JGFrun(int size) {

27 }
28 @Perm(requires="share(this) in alive",
29 ensures="share(this) in alive")
30  public void JGFsetsize(int size) {

32 }
33 @Perm(requires="unique(this) in alive",
34 ensures="unique(this) in alive")
35  public void JGFinitialise() {

37 }
38 @Perm(requires="share(this) in alive",
39 ensures="share(this) in alive")
40  public void JGFapplication() {

42 }
43 @Perm(requires="pure(this) in alive",
44 ensures="pure(this) in alive")
45  public void JGFvalidate() {

47 }
48 @Perm(requires="unique(this) in alive",
49 ensures="unique(this) in alive")
50  public void JGFtidyup() {

52 }

54 }ENDOFCLASS

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

58 class Tunnel {
59 @Perm(ensures="unique(this) in alive")
60 Tunnel() {   }

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

66 }
67 @Perm(requires="share(this) in alive",
68 ensures="share(this) in alive")
69  public void runiters() {

71 }
72 @Perm(requires="share(this) in alive",
73 ensures="share(this) in alive")
74   void doIteration() {

76 }
77 @Perm(requires="share(this) in alive",
78 ensures="share(this) in alive")
```

```
 private void calculateDummyCells(double localpg[][], double localtg[][], Statevector localug[][]) {

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

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 private void calculateDamping(double localpg[][], Statevector localug[][]) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 private void calculateF(double localpg[][], double localtg[][], Statevector localug[][]) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 private void calculateG(double localpg[][], double localtg[][], Statevector localug[][]) {

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

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 private void calculateStateVar(double localpg[][], double localtg[][], Statevector localug[][]) {

}

}ENDOFCLASS

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

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

@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public Statevector svect(Statevector that) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public Statevector amvect(double m, Statevector that) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public Statevector avect(Statevector that) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public Statevector mvect(double m) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public Statevector smvect(double m, Statevector that) {
 return null;

}

}ENDOFCLASS

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

class Vector2 {
@Perm(ensures="unique(this) in alive")
Vector2() {   }
```

```
160 @Perm(requires="pure(this) in alive",
161 ensures="pure(this) in alive")
162  public double magnitude() {
163  return 0;

165 }
166 @Perm(requires="pure(this) in alive",
167 ensures="pure(this) in alive")
168  public double dot(Vector2 that) {
169  return 0;

171 }

173 }ENDOFCLASS

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

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

181 @Perm(requires="share(this) in alive",
182 ensures="share(this) in alive")
183  public void reset() {

185 }
186 @Perm(requires="share(this) in alive",
187 ensures="share(this) in alive")
188  public void start() {

190 }
191 @Perm(requires="share(this) in alive",
192 ensures="share(this) in alive")
193  public void stop() {

195 }
196 @Perm(requires="share(this) in alive",
197 ensures="share(this) in alive")
198  public void addops(double count) {

200 }
201 @Perm(requires="pure(this) in alive",
202 ensures="pure(this) in alive")
203  public double perf() {
204  return 0;

206 }
207 @Perm(requires="pure(this) in alive",
208 ensures="pure(this) in alive")
209  public void longprint() {

211 }
212 @Perm(requires="share(this) in alive",
213 ensures="share(this) in alive")
214  public void print() {

216 }
217 @Perm(requires="pure(this) in alive",
218 ensures="pure(this) in alive")
219  public void printperf() {

221 }

223 }ENDOFCLASS

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

227 class JGFInstrumentor {
228 @Perm(ensures="unique(this) in alive")
229 JGFInstrumentor() {   }

231 @Perm(requires="share(this) in alive",
232 ensures="share(this) in alive")
233   void addTimer(String name) {

235 }
236 @Perm(requires="share(this) in alive",
237 ensures="share(this) in alive")
238   void addOpsToTimer(String name, double count) {

240 }
```

```
241 @Perm(requires="share(this) in alive",
242 ensures="share(this) in alive")
243   void startTimer(String name) {

245 }
246 @Perm(requires="share(this) in alive",
247 ensures="share(this) in alive")
248   void stopTimer(String name) {

250 }
251 @Perm(requires="share(this) in alive",
252 ensures="share(this) in alive")
253   double readTimer(String name) {
254  return 0;

256 }
257 @Perm(requires="share(this) in alive",
258 ensures="share(this) in alive")
259   void resetTimer(String name) {

261 }
262 @Perm(requires="share(this) in alive",
263 ensures="share(this) in alive")
264   void printTimer(String name) {

266 }
267 @Perm(requires="share(this) in alive",
268 ensures="share(this) in alive")
269   void printperfTimer(String name) {

271 }
272 @Perm(requires="share(this) in alive",
273 ensures="share(this) in alive")
274   void storeData(String name, Object obj) {

276 }
277 @Perm(requires="share(this) in alive",
278 ensures="share(this) in alive")
279   void retrieveData(String name, Object obj) {

281 }

283   void printHeader(int section, int size) {

285 }
286 @Perm(requires="unique(this) in alive",
287 ensures="unique(this) in alive")
288   void main(String argv[]) {

290 }

292 }ENDOFCLASS
```