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
JGFLUFactBenchSizeB	2	1	0	0	0	3	0	0
JGFLUFactBench	7	1	0	0	0	28	0	0
Linpack	11	1	0	0	10	66	22	33
JGFInstrumentor	13	1	0	0	12	91	12	13
JGFTimer	9	1	0	0	3	45	6	13
Total Classes=5	42	5	0	0	25	233	40	17

Contents

1 JGFLUFactBenchSizeB

Table 2: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
JGFLUFactBenchSizeB	√
main	√

Table 3: State Transition Matrix

	alive
alive	↑

Table 4: Methods Concurrency Matrix

	JGFLUFactBenchSizeB	main
JGFLUFactBenchSizeB	∦	∦
main	∦	∦

2 JGFLUFactBench

Table 5: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
JGFLUFactBench	√
JGFrun	√
JGFsetsize	√
JGFinitialise	√
JGFkernel	√
JGFvalidate	√
JGFtidyup	√

Table 6: State Transition Matrix

	alive
alive	↑

Table 7: Methods Concurrency Matrix

	JGFLUFactBench	JGFrun	JGFsetsize	JGFinitialise	JGFkernel	JGFvalidate	JGFtidyup
JGFLUFactBench	∦	∦	∦	∦	∦	∦	∦
JGFrun	∦	∦	∦	∦	∦	∦	∦
JGFsetsize	∦	∦	∦	∦	∦	∦	∦
JGFinitialise	∦	∦	∦	∦	∦	∦	∦
JGFkernel	∦	∦	∦	∦	∦	∦	∦
JGFvalidate	∦	∦	∦	∦	∦	∦	∦
JGFtidyup	∦	∦	∦	∦	∦	∦	∦

3 Linpack

Table 8: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
Linpack	√
matgen	√
dgefa	√
idamax	√
abs	√
epslon	√
dmxpy	√
dscal	√
daxpy	√
dgesl	√
ddot	√

Table 9: State Transition Matrix

	alive
alive	↑

Table 10: Methods Concurrency Matrix

	Linpack	matgen	dgefa	idamax	abs	epslon	dmxpy	dscal	daxpy	dgesl	ddot
Linpack	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
matgen	∦	∦	∦	∦	∥	∥	∦	∦	∦	∦	∦
dgefa	∦	∦	∦	∦	∥	∥	∦	∦	∦	∦	∦
idamax	∦	∦	∦	∥	∥	∥	∦	∦	∦	∦	∥
abs	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥
epslon	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥
dmxpy	∦	∦	∦	∦	∥	∥	∦	∦	∦	∦	∦
dscal	∦	∦	∦	∦	∥	∥	∦	∦	∦	∦	∦
daxpy	∦	∦	∦	∦	∥	∥	∦	∦	∦	∦	∦
dgesl	∦	∦	∦	∦	∥	∥	∦	∦	∦	∦	∦
ddot	∦	∦	∦	∥	∥	∥	∦	∦	∦	∦	∥

4 JGFInstrumentor

Table 11: Method's Satisfiability(Code Reachabiity Analysis

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

Table 12: State Transition Matrix

	alive
alive	↑

Table 13: Methods Concurrency Matrix

	JGFInstrumentor	addTimer	startTimer	stopTimer	addOpsToTimer	readTimer	resetTimer	printTimer	printperfTimer	storeData	retrieveData	printHeader	main
JGFInstrumentor	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
addTimer	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
startTimer	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
stopTimer	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
addOpsToTimer	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
readTimer	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
resetTimer	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
printTimer	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
printperfTimer	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
storeData	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
retrieveData	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
printHeader	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
main	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦

5 JGFTimer

Table 14: Method's Satisfiability(Code Reachabiity Analysis

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

Table 15: State Transition Matrix

	alive
alive	$\uparrow$

Table 16: Methods Concurrency Matrix

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

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 the input program generated by Sip4J

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

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

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 JGFLUFactBench {
20 @Perm(ensures="unique(this) in alive")
21 JGFLUFactBench() {   }

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 JGFkernel() {

42 }
43 @Perm(requires="share(this) in alive",
44 ensures="share(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 Linpack {
59 @Perm(ensures="unique(this) in alive")
60 Linpack() {   }

62 @Perm(requires="share(this) in alive",
63 ensures="share(this) in alive")
64  final double matgen(double a[][], int lda, int n, double b[]) {
65  return 0;

67 }
68 @Perm(requires="share(this) in alive",
69 ensures="share(this) in alive")
70  final int dgefa(double a[][], int lda, int n, int ipvt[]) {
71  return 0;

73 }
74 @Perm(requires="pure(this) in alive",
75 ensures="pure(this) in alive")
76  final int idamax(double dx[], int n, int dx_off, int incx) {
77  return 0;
```

```
79  }
80
81   final double abs(double d) {
82   return 0;
83
84  }
85
86   final double epslon(double x) {
87   return 0;
88
89  }
90  @Perm(requires="share(this) in alive",
91  ensures="share(this) in alive")
92   final void dmxpy(int n1, double y[], int n2, double x[], double m[][]) {
93
94  }
95  @Perm(requires="full(this) in alive",
96  ensures="full(this) in alive")
97   final void dscal(double dx[], int n, double da, int dx_off, int incx) {
98
99  }
100 @Perm(requires="share(this) in alive",
101 ensures="share(this) in alive")
102  final void daxpy(double dx[], int n, double dy[], double da, int dx_off, int incx, int dy_off, int incy
         ) {
103
104 }
105 @Perm(requires="share(this) in alive",
106 ensures="share(this) in alive")
107  final void dgesl(double a[][], int lda, int n, int ipvt[], double b[], int job) {
108
109 }
110 @Perm(requires="pure(this) in alive",
111 ensures="pure(this) in alive")
112  final double ddot(double dx[], double dy[], int n, int dx_off, int incx, int dy_off, int incy) {
113  return 0;
114
115 }
116
117 }ENDOFCLASS
118
119 @ClassStates({@State(name = "alive")})
120
121 class JGFInstrumentor {
122 @Perm(ensures="unique(this) in alive")
123 JGFInstrumentor() {   }
124
125 @Perm(requires="share(this) in alive",
126 ensures="share(this) in alive")
127   void addTimer(String name) {
128
129 }
130 @Perm(requires="share(this) in alive",
131 ensures="share(this) in alive")
132   void startTimer(String name) {
133
134 }
135 @Perm(requires="share(this) in alive",
136 ensures="share(this) in alive")
137   void stopTimer(String name) {
138
139 }
140 @Perm(requires="share(this) in alive",
141 ensures="share(this) in alive")
142   void addOpsToTimer(String name, double count) {
143
144 }
145 @Perm(requires="share(this) in alive",
146 ensures="share(this) in alive")
147   double readTimer(String name) {
148  return 0;
149
150 }
151 @Perm(requires="share(this) in alive",
152 ensures="share(this) in alive")
153   void resetTimer(String name) {
154
155 }
156 @Perm(requires="share(this) in alive",
157 ensures="share(this) in alive")
158   void printTimer(String name) {
```

```
160  }
161  @Perm(requires="share(this) in alive",
162  ensures="share(this) in alive")
163    void printperfTimer(String name) {

165  }
166  @Perm(requires="share(this) in alive",
167  ensures="share(this) in alive")
168    void storeData(String name, Object obj) {

170  }
171  @Perm(requires="share(this) in alive",
172  ensures="share(this) in alive")
173    void retrieveData(String name, Object obj) {

175  }

177    void printHeader(int section, int size) {

179  }
180  @Perm(requires="unique(this) in alive",
181  ensures="unique(this) in alive")
182    void main(String argv[]) {

184  }

186  }ENDOFCLASS

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

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

194  @Perm(requires="share(this) in alive",
195  ensures="share(this) in alive")
196   public void start() {

198  }
199  @Perm(requires="share(this) in alive",
200  ensures="share(this) in alive")
201   public void stop() {

203  }
204  @Perm(requires="share(this) in alive",
205  ensures="share(this) in alive")
206   public void addops(double count) {

208  }
209  @Perm(requires="share(this) in alive",
210  ensures="share(this) in alive")
211   public void reset() {

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

218  }
219  @Perm(requires="pure(this) in alive",
220  ensures="pure(this) in alive")
221   public double perf() {
222   return 0;

224  }
225  @Perm(requires="pure(this) in alive",
226  ensures="pure(this) in alive")
227   public void printperf() {

229  }
230  @Perm(requires="pure(this) in alive",
231  ensures="pure(this) in alive")
232   public void longprint() {

234  }

236  }ENDOFCLASS
```