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
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: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFLUFactBenchSizeB	√
main	√

Table 3: State Transition Matrix

	alive
alive	↑

Table 4: Methods Concurrency Matrix

	JGFLUFactBenchSizeB	main
JGFLUFactBenchSizeB	∦	∦
main	∦	∦

2 JGFLUFactBench

Table 5: Methods Requires Clause Satisfiability

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: Methods Requires Clause Satisfiability

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: Methods Requires Clause Satisfiability

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: Methods Requires Clause Satisfiability

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 Sequential Java 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[]) {
12 }

14 }ENDOFCLASS

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

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

22 @Perm(requires="unique(this) in alive",
23 ensures="unique(this) in alive")
24 public void JGFrun(int size) {
25 }
26 @Perm(requires="full(this) in alive",
27 ensures="full(this) in alive")
28 public void JGFsetsize(int size) {
29 }
30 @Perm(requires="unique(this) in alive",
31 ensures="unique(this) in alive")
32 public void JGFinitialise() {
33 }
34 @Perm(requires="full(this) in alive",
35 ensures="full(this) in alive")
36 public void JGFkernel() {
37 }
38 @Perm(requires="full(this) in alive",
39 ensures="full(this) in alive")
40 public void JGFvalidate() {
41 }
42 @Perm(requires="unique(this) in alive",
43 ensures="unique(this) in alive")
44 public void JGFtidyup() {
45 }

47 }ENDOFCLASS

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

51 class Linpack {
52 @Perm(ensures="unique(this) in alive")
53 Linpack() {   }

55 @Perm(requires="full(this) in alive",
56 ensures="full(this) in alive")
57 final double matgen(double a[][], int lda, int n, double b[]) {
58  return 0;
59 }
60 @Perm(requires="full(this) in alive",
61 ensures="full(this) in alive")
62 final int dgefa(double a[][], int lda, int n, int ipvt[]) {
63  return 0;
64 }
65 @Perm(requires="pure(this) in alive",
66 ensures="pure(this) in alive")
67 final int idamax(double dx[], int n, int dx_off, int incx) {
68  return 0;
69 }

71 final double abs(double d) {
72  return 0;
73 }

75 final double epslon(double x) {
```

```
 return 0;
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
final void dmxpy(int n1, double y[], int n2, double x[], double m[][]) {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
final void dscal(double dx[], int n, double da, int dx_off, int incx) {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
final void daxpy(double dx[], int n, double dy[], double da, int dx_off, int incx, int dy_off, int incy)
      {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
final void dgesl(double a[][], int lda, int n, int ipvt[], double b[], int job) {
}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
final double ddot(double dx[], double dy[], int n, int dx_off, int incx, int dy_off, int incy) {
 return 0;
}

}ENDOFCLASS

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

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

@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 void addTimer(String name) {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 void startTimer(String name) {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 void stopTimer(String name) {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 void addOpsToTimer(String name, double count) {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 double readTimer(String name) {
 return 0;
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 void resetTimer(String name) {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 void printTimer(String name) {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 void printperfTimer(String name) {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 void storeData(String name, Object obj) {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 void retrieveData(String name, Object obj) {
}

 void printHeader(int section, int size) {
}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 void main(String argv[]) {
}
```

```
157 }ENDOFCLASS

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

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

165 @Perm(requires="full(this) in alive",
166 ensures="full(this) in alive")
167 public void start() {
168 }
169 @Perm(requires="full(this) in alive",
170 ensures="full(this) in alive")
171 public void stop() {
172 }
173 @Perm(requires="full(this) in alive",
174 ensures="full(this) in alive")
175 public void addops(double count) {
176 }
177 @Perm(requires="full(this) in alive",
178 ensures="full(this) in alive")
179 public void reset() {
180 }
181 @Perm(requires="full(this) in alive",
182 ensures="full(this) in alive")
183 public void print() {
184 }
185 @Perm(requires="pure(this) in alive",
186 ensures="pure(this) in alive")
187 public double perf() {
188  return 0;
189 }
190 @Perm(requires="pure(this) in alive",
191 ensures="pure(this) in alive")
192 public void printperf() {
193 }
194 @Perm(requires="pure(this) in alive",
195 ensures="pure(this) in alive")
196 public void longprint() {
197 }

199 }ENDOFCLASS
```