

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	1	3	1	33
JGFLUFactBench	6	1	0	0	5	21	5	24
JGFInstrumentor	3	1	0	0	0	6	0	0
Linpack	11	1	0	0	10	66	54	82
JGFTimer	3	1	0	0	2	6	2	33
Total Classes=5	25	5	0	0	18	102	62	61

Contents

1	JGFLUFactBenchSizeB	3
2	JGFLUFactBench	4
3	JGFInstrumentor	5
4	Linpack	6
5	JGFTimer	7
6	Abbreviation	8
7	Annotated Version of Sequential Java Program generated by Sip4j	9

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	✓
JGFinialise	✓
JGFvalidate	✓
JGFsetsize	✓
JGFtidyup	✓

Table 6: State Transition Matrix

	alive
alive	↑

Table 7: Methods Concurrency Matrix

	JGFLUFactBench	JGFRun	JGFinialise	JGFvalidate	JGFsetsize	JGFtidyup
JGFLUFactBench	⌈	⌈	⌈	⌈	⌈	⌈
JGFRun	⌈	⌈	⌈	⌈	⌈	⌈
JGFinialise	⌈	⌈	⌈	⌈	⌈	⌈
JGFvalidate	⌈	⌈	⌈	⌈	⌈	⌈
JGFsetsize	⌈	⌈	⌈	⌈	⌈	⌈
JGFtidyup	⌈	⌈	⌈	⌈	⌈	⌈

3 JGFInstrumentor

Table 8: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFInstrumentor	✓
addTimer	✓
printTimer	✓

Table 9: State Transition Matrix

	alive
alive	↑

Table 10: Methods Concurrency Matrix

	JGFInstrumentor	addTimer	printTimer
JGFInstrumentor	⌘	⌘	⌘
addTimer	⌘	⌘	⌘
printTimer	⌘	⌘	⌘

4 Linpack

Table 11: Methods Requires Clause Satisfiability

Method	Satisfiability
Linpack	✓
matgen	✓
dmxpy	✓
abs	✓
idamax	✓
dgefa	✓
epsilon	✓
dscal	✓
daxpy	✓
dgesl	✓
ddot	✓

Table 12: State Transition Matrix

	alive
alive	↑

Table 13: Methods Concurrency Matrix

	Linpack	matgen	dmxpy	abs	idamax	dgefa	epsilon	dscal	daxpy	dgesl	ddot
Linpack	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
matgen	⌘										
dmxpy	⌘										
abs	⌘										
idamax	⌘										
dgefa	⌘					⌘					
epsilon	⌘										
dscal	⌘										
daxpy	⌘										
dgesl	⌘										
ddot	⌘										

5 JGFTimer

Table 14: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFTimer	✓
print	✓
perf	✓

Table 15: State Transition Matrix

	alive
alive	↑

Table 16: Methods Concurrency Matrix

	JGFTimer	print	perf
JGFTimer	⌈	⌈	⌈
print	⌈	⌈	
perf	⌈		

6 Abbreviation

Table 17: Used Abbreviation

Symbol	Meaning
✓	requires clause of the method is satisfiable
✗	requires clause of the method is unsatisfiable
↑	The row-state can be transitioned to the column-state
✕	The row-state cannot be transitioned to the column-state
	The row-method can be possibly executed parallel with the column-method
⋈	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 JGFLUFactBenchSizeB {
6   @Perm(ensures="unique(this) in alive")
7   JGFLUFactBenchSizeB() { }
8
9   @Perm(requires="none(this) in alive",
10  ensures="unique(this) in alive")
11   void main(String argv[]) {
12   }
13
14 }ENDOFCLASS
15
16 @ClassStates({@State(name = "alive")})
17
18 class JGFLUFactBench {
19   @Perm(ensures="unique(this) in alive")
20   JGFLUFactBench() { }
21
22   @Perm(requires="full(this) in alive",
23  ensures="full(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 JGFinitialise() {
29   }
30   @Perm(requires="full(this) in alive",
31  ensures="full(this) in alive")
32   public void JGFvalidate() {
33   }
34   @Perm(requires="full(this) in alive",
35  ensures="full(this) in alive")
36   public void JGFsetsize(int size) {
37   }
38
39   public void JGFtidyup() {
40   }
41
42 }ENDOFCLASS
43
44 @ClassStates({@State(name = "alive")})
45
46 class JGFInstrumentor {
47   @Perm(ensures="unique(this) in alive")
48   JGFInstrumentor() { }
49
50   @Perm(requires="full(this) in alive",
51  ensures="full(this) in alive")
52   void addTimer(String name, String opname, 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 Linpack {
64   @Perm(ensures="unique(this) in alive")
65   Linpack() { }
66
67   @Perm(requires="full(#0) * pure(#1) * pure(#2) * full(#3) in alive",
68  ensures="full(#0) * pure(#1) * pure(#2) * full(#3) in alive")
69   final double matgen(double a[][], int lda, int n, double b[]) {
70   return 0;
71   }
72   @Perm(requires="pure(#0) * full(#1) * full(#2) * pure(#3) in alive",
73  ensures="pure(#0) * full(#1) * full(#2) * pure(#3) in alive")
74   final void dmcpy(int n1, double y[], int n2, double x[], double m[][]) {
75   }
```

```

77 final double abs(double d) {
78     return 0;
79 }
80 @Perm(requires="pure(#0) * pure(#1) in alive",
81 ensures="pure(#0) * pure(#1) in alive")
82 final int idamax(int n, double dx[], int dx_off, int incx) {
83     return 0;
84 }
85 @Perm(requires="full(this) * full(#0) * pure(#1) * pure(#2) * full(#3) in alive",
86 ensures="full(this) * full(#0) * pure(#1) * pure(#2) * full(#3) in alive")
87 final int dgefa(double a[][], int lda, int n, int ipvt[]) {
88     return 0;
89 }

91 final double epslon(double x) {
92     return 0;
93 }
94 @Perm(requires="pure(#0) * full(#1) in alive",
95 ensures="pure(#0) * full(#1) in alive")
96 final void dscal(int n, double da, double dx[], int dx_off, int incx) {
97 }
98 @Perm(requires="pure(#0) * pure(#1) * full(#2) in alive",
99 ensures="pure(#0) * pure(#1) * full(#2) in alive")
100 final void daxpy(int n, double dx[], double da, int dx_off, int incx, double dy[], int dy_off, int incy)
    {
101 }
102 @Perm(requires="full(#0) * pure(#1) * pure(#2) * pure(#3) * full(#4) * full(#5) in alive",
103 ensures="full(#0) * pure(#1) * pure(#2) * pure(#3) * full(#4) * full(#5) in alive")
104 final void dgesl(double a[][], int lda, int n, int ipvt[], double b[], int job) {
105 }
106 @Perm(requires="pure(#0) * full(#1) in alive",
107 ensures="pure(#0) * full(#1) in alive")
108 final double ddot(int n, double dx[], int dx_off, int incx, double dy[], int dy_off, int incy) {
109     return 0;
110 }

112 }ENDOFCLASS

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

116 class JGFTimer {
117     @Perm(ensures="unique(this) in alive")
118     JGFTimer() { }

120     @Perm(requires="full(this) in alive",
121 ensures="full(this) in alive")
122     public void print() {
123     }
124     @Perm(requires="pure(this) in alive",
125 ensures="pure(this) in alive")
126     public double perf() {
127         return 0;
128     }
129 }

130 }ENDOFCLASS

```