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
JGFInstrumentor	13	1	0	0	12	91	12	13
JGFTimer	9	1	0	0	3	45	6	13
JGFCryptBenchSizeA	2	1	0	0	0	3	0	0
JGFCryptBench	7	1	0	0	2	28	3	11
IDEATest	9	1	0	0	8	45	16	36
Total Classes=5	40	5	0	0	25	212	37	17

Contents

1 JGFInstrumentor

Table 2: Methods Requires Clause Satisfiability

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

Table 3: State Transition Matrix

	alive
alive	↑

Table 4: 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	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦

2 JGFTimer

Table 5: 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 6: State Transition Matrix

	alive
alive	$\uparrow$

Table 7: 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	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$	$\parallel$

3 JGFCryptBenchSizeA

Table 8: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFCryptBenchSizeA	√
main	√

Table 9: State Transition Matrix

	alive
alive	↑

Table 10: Methods Concurrency Matrix

	JGFCryptBenchSizeA	main
JGFCryptBenchSizeA	∦	∦
main	∦	∦

4 JGFCryptBench

Table 11: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFCryptBench	$\surd$
JGFrun	$\surd$
JGFsetsize	$\surd$
JGFinitialise	$\surd$
JGFkernel	$\surd$
JGFvalidate	$\surd$
JGFtidyup	$\surd$

Table 12: State Transition Matrix

	alive
alive	$\uparrow$

Table 13: Methods Concurrency Matrix

	JGFCryptBench	JGFrun	JGFsetsize	JGFinitialise	JGFkernel	JGFvalidate	JGFtidyup
JGFCryptBench	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
JGFrun	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
JGFsetsize	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
JGFinitialise	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
JGFkernel	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$	$\nparallel$
JGFvalidate	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$	$\nparallel$
JGFtidyup	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$

5 IDEATest

Table 14: Methods Requires Clause Satisfiability

Method	Satisfiability
IDEATest	$\surd$
buildTestData	$\surd$
calcEncryptKey	$\surd$
calcDecryptKey	$\surd$
inv	$\surd$
Do	$\surd$
cipheridea	$\surd$
freeTestData	$\surd$
mul	$\surd$

Table 15: State Transition Matrix

	alive
alive	$\uparrow$

Table 16: Methods Concurrency Matrix

	IDEATest	buildTestData	calcEncryptKey	calcDecryptKey	inv	Do	cipheridea	freeTestData	mul
IDEATest	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
buildTestData	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$
calcEncryptKey	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$
calcDecryptKey	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$
inv	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
Do	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$	$\nparallel$	$\nparallel$	$\parallel$
cipheridea	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$
freeTestData	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$
mul	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$

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

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

@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[]) {
}

}ENDOFCLASS

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

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

@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
public void start() {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
public void stop() {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
```

```
76  public void addops(double count) {
77  }
78  @Perm(requires="full(this) in alive",
79  ensures="full(this) in alive")
80  public void reset() {
81  }
82  @Perm(requires="full(this) in alive",
83  ensures="full(this) in alive")
84  public void print() {
85  }
86  @Perm(requires="pure(this) in alive",
87  ensures="pure(this) in alive")
88  public double perf() {
89   return 0;
90  }
91  @Perm(requires="pure(this) in alive",
92  ensures="pure(this) in alive")
93  public void printperf() {
94  }
95  @Perm(requires="pure(this) in alive",
96  ensures="pure(this) in alive")
97  public void longprint() {
98  }

100 }ENDOFCLASS

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

104 class JGFCryptBenchSizeA {
105 @Perm(ensures="unique(this) in alive")
106 JGFCryptBenchSizeA() {   }

108 @Perm(requires="unique(this) in alive",
109 ensures="unique(this) in alive")
110  void main(String argv[]) {
111 }

113 }ENDOFCLASS

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

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

121 @Perm(requires="unique(this) in alive",
122 ensures="unique(this) in alive")
123 public void JGFrun(int size) {
124 }
125 @Perm(requires="full(this) in alive",
126 ensures="full(this) in alive")
127 public void JGFsetsize(int size) {
128 }
129 @Perm(requires="unique(this) in alive",
130 ensures="unique(this) in alive")
131 public void JGFinitialise() {
132 }
133 @Perm(requires="pure(this) in alive",
134 ensures="pure(this) in alive")
135 public void JGFkernel() {
136 }
137 @Perm(requires="pure(this) in alive",
138 ensures="pure(this) in alive")
139 public void JGFvalidate() {
140 }
141 @Perm(requires="unique(this) in alive",
142 ensures="unique(this) in alive")
143 public void JGFtidyup() {
144 }

146 }ENDOFCLASS

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

150 class IDEATest {
151 @Perm(ensures="unique(this) in alive")
152 IDEATest() {   }

154 @Perm(requires="unique(this) in alive",
155 ensures="unique(this) in alive")
156  void buildTestData() {
```

```
157 }
158 @Perm(requires="full(this) in alive",
159 ensures="full(this) in alive")
160 private void calcEncryptKey() {
161 }
162 @Perm(requires="full(this) in alive",
163 ensures="full(this) in alive")
164 private void calcDecryptKey() {
165 }

167 private int inv(int x) {
168  return 0;
169 }
170 @Perm(requires="pure(this) in alive",
171 ensures="pure(this) in alive")
172 public void Do() {
173 }
174 @Perm(requires="full(this) in alive",
175 ensures="full(this) in alive")
176 private void cipheridea(byte[] text1, byte[] text2, int[] key) {
177 }
178 @Perm(requires="unique(this) in alive",
179 ensures="unique(this) in alive")
180  void freeTestData() {
181 }

183 private int mul(int a, int b) {
184  return 0;
185 }

187 }ENDOFCLASS
```