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
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: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFEulerBenchSizeA	√
main	√

Table 3: State Transition Matrix

	alive
alive	↑

Table 4: Methods Concurrency Matrix

	JGFEulerBenchSizeA	main
JGFEulerBenchSizeA	∦	∦
main	∦	∦

2 JGFEulerBench

Table 5: Methods Requires Clause Satisfiability

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: Methods Requires Clause Satisfiability

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: Methods Requires Clause Satisfiability

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: Methods Requires Clause Satisfiability

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: Methods Requires Clause Satisfiability

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

Table 18: State Transition Matrix

	alive
alive	$\uparrow$

Table 19: Methods Concurrency Matrix

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

7 JGFInstrumentor

Table 20: Methods Requires Clause Satisfiability

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 Sequential Java Program generated by Sip4j

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

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

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

}ENDOFCLASS

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

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

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
public void JGFrun(int size) {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
public void JGFsetsize(int size) {
}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
public void JGFinitialise() {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
public void JGFapplication() {
}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
public void JGFvalidate() {
}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
public void JGFtidyup() {
}

}ENDOFCLASS

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

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

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
public void initialise() {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
public void runiters() {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 void doIteration() {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
private void calculateDummyCells(double localpg[][], double localtg[][], Statevector localug[][]) {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
private void calculateDeltaT() {
}
@Perm(requires="full(this) in alive",
```

```
ensures="full(this) in alive")
private void calculateDamping(double localpg[][], Statevector localug[][]) {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
private void calculateF(double localpg[][], double localtg[][], Statevector localug[][]) {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
private void calculateG(double localpg[][], double localtg[][], Statevector localug[][]) {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
private void calculateR() {
}
@Perm(requires="full(this) in alive",
ensures="full(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="full(this) in alive",
ensures="full(this) in alive")
public Statevector svect(Statevector that) {
 return null;
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
public Statevector amvect(double m, Statevector that) {
 return null;
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
public Statevector avect(Statevector that) {
 return null;
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
public Statevector mvect(double m) {
 return null;
}
@Perm(requires="full(this) in alive",
ensures="full(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() {   }

@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
public double magnitude() {
 return 0;
}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
public double dot(Vector2 that) {
 return 0;
}

}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 reset() {
}
@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")
public void addops(double count) {
}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
public double perf() {
 return 0;
}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
public void longprint() {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
public void print() {
}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
public void printperf() {
}

}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 addOpsToTimer(String name, double count) {
}
@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")
 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")
```

```
238   void retrieveData(String name, Object obj) {
239  }

241   void printHeader(int section, int size) {
242  }
243  @Perm(requires="unique(this) in alive",
244  ensures="unique(this) in alive")
245   void main(String argv[]) {
246  }

248  }ENDOFCLASS
```