Summary

Sink States:0(0×10^0)

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

Table 2: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFLUFactBenchSizeB	$\surd$
main	$\surd$

Table 3: State Transition Matrix

	alive
alive	$\uparrow$

Table 4: Methods Concurrency Matrix

	JGFLUFactBenchSizeB	main
JGFLUFactBenchSizeB	$\nparallel$	$\nparallel$
main	$\nparallel$	$\parallel$

2 JGFLUFactBench

Table 5: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFLUFactBench	$\surd$
JGFrun	$\surd$
JGFinitialise	$\surd$
JGFvalidate	$\surd$
JGFsetsize	$\surd$
JGFtidyup	$\surd$

Table 6: State Transition Matrix

	alive
alive	$\uparrow$

Table 7: Methods Concurrency Matrix

	JGFLUFactBench	JGFrun	JGFinitialise	JGFvalidate	JGFsetsize	JGFtidyup
JGFLUFactBench	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
JGFrun	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$
JGFinitialise	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$
JGFvalidate	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$
JGFsetsize	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$
JGFtidyup	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$

3 JGFInstrumentor

Table 8: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFInstrumentor	$\surd$
addTimer	$\surd$
printTimer	$\surd$

Table 9: State Transition Matrix

	alive
alive	$\uparrow$

Table 10: Methods Concurrency Matrix

	JGFInstrumentor	addTimer	printTimer
JGFInstrumentor	$\nparallel$	$\nparallel$	$\nparallel$
addTimer	$\nparallel$	$\nparallel$	$\nparallel$
printTimer	$\nparallel$	$\nparallel$	$\nparallel$

4 Linpack

Table 11: Methods Requires Clause Satisfiability

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

Table 12: State Transition Matrix

	alive
alive	↑

Table 13: Methods Concurrency Matrix

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

5 JGFTimer

Table 14: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFTimer	$\surd$
print	$\surd$
perf	$\surd$

Table 15: State Transition Matrix

	alive
alive	$\uparrow$

Table 16: Methods Concurrency Matrix

	JGFTimer	print	perf
JGFTimer	$\nparallel$	$\nparallel$	$\nparallel$
print	$\nparallel$	$\nparallel$	$\parallel$
perf	$\nparallel$	$\parallel$	$\parallel$

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

@Perm(requires="none(this) in alive",
ensures="unique(this) in alive")
 void main(String argv[]) {
}

}ENDOFCLASS

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

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

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

public void JGFtidyup() {
}

}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, String opname, int size) {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 void printTimer(String name) {
}

}ENDOFCLASS

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

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

@Perm(requires="full(#0) * pure(#1) * pure(#2) * full(#3) in alive",
ensures="full(#0) * pure(#1) * pure(#2) * full(#3) in alive")
final double matgen(double a[][], int lda, int n, double b[]) {
 return 0;
}
@Perm(requires="pure(#0) * full(#1) * full(#2) * pure(#3) in alive",
ensures="pure(#0) * full(#1) * full(#2) * pure(#3) in alive")
final void dmxpy(int n1, double y[], int n2, double x[], double m[][]) {
}
final double abs(double d) {
 return 0;
}
@Perm(requires="pure(#0) * pure(#1) in alive",
ensures="pure(#0) * pure(#1) in alive")
final int idamax(int n, double dx[], int dx_off, int incx) {
 return 0;
}
@Perm(requires="full(this) * full(#0) * pure(#1) * pure(#2) * full(#3) in alive",
ensures="full(this) * full(#0) * pure(#1) * pure(#2) * full(#3) in alive")
final int dgefa(double a[][], int lda, int n, int ipvt[]) {
 return 0;
}

final double epslon(double x) {
 return 0;
}
@Perm(requires="pure(#0) * full(#1) in alive",
ensures="pure(#0) * full(#1) in alive")
final void dscal(int n, double da, double dx[], int dx_off, int incx) {
}
@Perm(requires="pure(#0) * pure(#1) * full(#2) in alive",
ensures="pure(#0) * pure(#1) * full(#2) in alive")
final void daxpy(int n, double dx[], double da, int dx_off, int incx, double dy[], int dy_off, int incy)
      {
}
@Perm(requires="full(#0) * pure(#1) * pure(#2) * pure(#3) * full(#4) * full(#5) in alive",
ensures="full(#0) * pure(#1) * pure(#2) * pure(#3) * full(#4) * full(#5) in alive")
final void dgesl(double a[][], int lda, int n, int ipvt[], double b[], int job) {
}
@Perm(requires="pure(#0) * full(#1) in alive",
ensures="pure(#0) * full(#1) in alive")
final double ddot(int n, double dx[], int dx_off, int incx, double dy[], int dy_off, int incy) {
 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 print() {
}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
public double perf() {
 return 0;
}

}ENDOFCLASS
```