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
JGFMolDynBenchSizeA	2	1	0	0	1	3	1	33
JGFInstrumentor	4	1	0	0	3	10	3	30
JGFMolDynBench	5	1	0	0	0	15	0	0
JGFTimer	2	1	0	0	0	3	0	0
md	3	1	0	0	0	6	0	0
particle	6	1	0	0	5	21	5	24
random	3	1	0	0	0	6	0	0
Total Classes=7	25	7	0	0	9	64	9	14

Contents

1 JGFMolDynBenchSizeA

Table 2: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFMolDynBenchSizeA	$\surd$
main	$\surd$

Table 3: State Transition Matrix

	alive
alive	$\uparrow$

Table 4: Methods Concurrency Matrix

	JGFMolDynBenchSizeA	main
JGFMolDynBenchSizeA	$\nparallel$	$\nparallel$
main	$\nparallel$	$\parallel$

2 JGFInstrumentor

Table 5: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFInstrumentor	$\surd$
printHeader	$\surd$
stopTimer	$\surd$
addOpsToTimer	$\surd$

Table 6: State Transition Matrix

	alive
alive	$\uparrow$

Table 7: Methods Concurrency Matrix

	JGFInstrumentor	printHeader	stopTimer	addOpsToTimer
JGFInstrumentor	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
printHeader	$\nparallel$	$\parallel$	$\parallel$	$\parallel$
stopTimer	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$
addOpsToTimer	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$

3 JGFMolDynBench

Table 8: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFMolDynBench	$\surd$
JGFrun	$\surd$
JGFapplication	$\surd$
JGFtidyup	$\surd$
JGFinitialise	$\surd$

Table 9: State Transition Matrix

	alive
alive	$\uparrow$

Table 10: Methods Concurrency Matrix

	JGFMolDynBench	JGFrun	JGFapplication	JGFtidyup	JGFinitialise
JGFMolDynBench	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
JGFrun	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
JGFapplication	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
JGFtidyup	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
JGFinitialise	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$

4 JGFTimer

Table 11: Methods Requires Clause Satisfiability

Method	Satisfiability
JGFTimer	$\surd$
stop	$\surd$

Table 12: State Transition Matrix

	alive
alive	$\uparrow$

Table 13: Methods Concurrency Matrix

	JGFTimer	stop
JGFTimer	$\nparallel$	$\nparallel$
stop	$\nparallel$	$\nparallel$

5 md

Table 14: Methods Requires Clause Satisfiability

Method	Satisfiability
md	$\surd$
initialise	$\surd$
runiters	$\surd$

Table 15: State Transition Matrix

	alive
alive	$\uparrow$

Table 16: Methods Concurrency Matrix

	md	initialise	runiters
md	$\nparallel$	$\nparallel$	$\nparallel$
initialise	$\nparallel$	$\nparallel$	$\nparallel$
runiters	$\nparallel$	$\nparallel$	$\nparallel$

6 particle

Table 17: Methods Requires Clause Satisfiability

Method	Satisfiability
particle	$\surd$
force	$\surd$
mkekin	$\surd$
velavg	$\surd$
dscal	$\surd$
domove	$\surd$

Table 18: State Transition Matrix

	alive
alive	$\uparrow$

Table 19: Methods Concurrency Matrix

	particle	force	mkekin	velavg	dscal	domove
particle	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
force	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$
mkekin	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$
velavg	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
dscal	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$
domove	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$

7 random

Table 20: Methods Requires Clause Satisfiability

Method	Satisfiability
random	$\surd$
seed	$\surd$
update	$\surd$

Table 21: State Transition Matrix

	alive
alive	$\uparrow$

Table 22: Methods Concurrency Matrix

	random	seed	update
random	$\nparallel$	$\nparallel$	$\nparallel$
seed	$\nparallel$	$\nparallel$	$\nparallel$
update	$\nparallel$	$\nparallel$	$\nparallel$

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

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

}ENDOFCLASS

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

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


 void printHeader(int section, int size) {
}
@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) {
}

}ENDOFCLASS

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

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

@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 JGFapplication() {
}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
public void JGFtidyup() {
}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
public void JGFinitialise() {
}

}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 stop() {
}

}ENDOFCLASS

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

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

@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() {
}

}ENDOFCLASS

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

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

@Perm(requires="full(this) in alive * pure(#0) in alive * pure(#1) in alive * pure(#2) in alive * pure
    (#3) in alive",
ensures="full(this) in alive * pure(#0) in alive * pure(#1) in alive * pure(#2) in alive * pure(#3) in
    alive")
public void force(double side, double rcoff, int mdsize, int x) {
}
@Perm(requires="full(this) in alive * pure(#0) in alive",
ensures="full(this) in alive * pure(#0) in alive")
public double mkekin(double hsq2) {
 return 0;
}
@Perm(requires="pure(this) in alive * pure(#0) in alive * pure(#1) in alive",
ensures="pure(this) in alive * pure(#0) in alive * pure(#1) in alive")
public double velavg(double vaverh, double h) {
 return 0;
}
@Perm(requires="full(this) in alive * pure(#0) in alive",
ensures="full(this) in alive * pure(#0) in alive")
public void dscal(double sc, int incx) {
}
@Perm(requires="full(this) in alive * pure(#0) in alive",
ensures="full(this) in alive * pure(#0) in alive")
public void domove(double side) {
}

}ENDOFCLASS

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

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

@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
public double seed() {
 return 0;
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
public double update() {
 return 0;
}

}ENDOFCLASS
```