Summary

Sink States:$0(0 \times 10^0)$

Table 1: Sip4J Analysis Summary

Classes	Methods	States	Unreachable clauses	Unreachable states	Possible concurrent methods	Total. no. of method pairs	No. of concurrent method pairs	Percentage of concurrent methods pairs
JGFTimer	9	1	0	0	3	45	6	13
JGFInstrumentor	13	1	0	0	12	91	12	13
SOR	2	1	0	0	0	3	0	0
JGFSORBenchSizeB	2	1	0	0	0	3	0	0
JGFSORBench	8	1	0	0	1	36	1	3
Total Classes=5	34	5	0	0	16	178	19	11

Contents

1 JGFTimer

Table 2: Method's Satisfiability(Code Reachabiity Analysis

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

Table 3: State Transition Matrix

	alive
alive	$\uparrow$

Table 4: 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	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$	$\nparallel$	$\parallel$

2 JGFInstrumentor

Table 5: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
JGFInstrumentor	$\surd$
addTimer	$\surd$
addOpsToTimer	$\surd$
startTimer	$\surd$
stopTimer	$\surd$
readTimer	$\surd$
resetTimer	$\surd$
printTimer	$\surd$
printperfTimer	$\surd$
storeData	$\surd$
retrieveData	$\surd$
printHeader	$\surd$
main	$\surd$

Table 6: State Transition Matrix

	alive
alive	$\uparrow$

Table 7: Methods Concurrency Matrix

	JGFInstrumentor	addTimer	addOpsToTimer	startTimer	stopTimer	readTimer	resetTimer	printTimer	printperfTimer	storeData	retrieveData	printHeader	main
JGFInstrumentor	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
addTimer	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$
addOpsToTimer	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$
startTimer	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$
stopTimer	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$
readTimer	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$
resetTimer	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$
printTimer	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$
printperfTimer	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$
storeData	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$
retrieveData	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$
printHeader	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
main	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$

3 SOR

Table 8: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
SOR	$\surd$
SORrun	$\surd$

Table 9: State Transition Matrix

	alive
alive	$\uparrow$

Table 10: Methods Concurrency Matrix

	SOR	SORrun
SOR	$\nparallel$	$\nparallel$
SORrun	$\nparallel$	$\nparallel$

4 JGFSORBenchSizeB

Table 11: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
JGFSORBenchSizeB	$\surd$
main	$\surd$

Table 12: State Transition Matrix

	alive
alive	$\uparrow$

Table 13: Methods Concurrency Matrix

	JGFSORBenchSizeB	main
JGFSORBenchSizeB	$\nparallel$	$\nparallel$
main	$\nparallel$	$\nparallel$

5 JGFSORBench

Table 14: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
JGFSORBench	√
JGFrun	√
JGFsetsize	√
JGFinitialise	√
JGFkernel	√
RandomMatrix	√
JGFvalidate	√
JGFtidyup	√

Table 15: State Transition Matrix

	alive
alive	↑

Table 16: Methods Concurrency Matrix

	JGFSORBench	JGFrun	JGFsetsize	JGFinitialise	JGFkernel	RandomMatrix	JGFvalidate	JGFtidyup
JGFSORBench	∦	∦	∦	∦	∦	∦	∦	∦
JGFrun	∦	∦	∦	∦	∦	∦	∦	∦
JGFsetsize	∦	∦	∦	∦	∦	∦	∦	∦
JGFinitialise	∦	∦	∦	∦	∦	∦	∦	∦
JGFkernel	∦	∦	∦	∦	∦	∦	∦	∦
RandomMatrix	∦	∦	∦	∦	∦	∦	∦	∦
JGFvalidate	∦	∦	∦	∦	∦	∦	∥	∦
JGFtidyup	∦	∦	∦	∦	∦	∦	∦	∦

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 the input program generated by Sip4J

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

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

@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void reset() {

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

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

}
@Perm(requires="share(this) in alive",
ensures="share(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="share(this) in alive",
ensures="share(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="share(this) in alive",
ensures="share(this) in alive")
  void addTimer(String name) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void addOpsToTimer(String name, double count) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void startTimer(String name) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void stopTimer(String name) {

}
```

```
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  double readTimer(String name) {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void resetTimer(String name) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void printTimer(String name) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void printperfTimer(String name) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void storeData(String name, Object obj) {

}
@Perm(requires="share(this) in alive",
ensures="share(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 SOR {
@Perm(ensures="unique(this) in alive")
SOR() {   }

@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void SORrun(int num_iterations, double G[][], double omega) {

}

}ENDOFCLASS

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

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

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

}

}ENDOFCLASS

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

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

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

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

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void JGFkernel() {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  double[][] RandomMatrix(int M, int N, java.util.Random R) {
 return null;

}
@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
```