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
SearchGame	3	1	0	0	0	6	0	0
Game	6	1	0	0	2	21	3	14
TransGame	10	1	0	0	1	55	1	2
JGFSearchBench	7	1	0	0	0	28	0	0
JGFInstrumentor	13	1	0	0	12	91	12	13
JGFTimer	9	1	0	0	3	45	6	13
JGFSearchBenchSizeA	2	1	0	0	0	3	0	0
Total Classes=7	50	7	0	0	18	249	22	9

Contents

1 SearchGame 3

2 Game 4

3 TransGame 5

4 JGFSearchBench 6

5 JGFInstrumentor 7

6 JGFTimer 8

7 JGFSearchBenchSizeA 9

8 Abbreviation 10

9 Annotated version of the input program generated by Sip4J 11

1 SearchGame

Table 2: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
SearchGame	√
solve	√
ab	√

Table 3: State Transition Matrix

	alive
alive	↑

Table 4: Methods Concurrency Matrix

	SearchGame	solve	ab
SearchGame	∦	∦	∦
solve	∦	∦	∦
ab	∦	∦	∦

2 Game

Table 5: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
Game	√
wins	√
makemove	√
backmove	√
reset	√
toString	√

Table 6: State Transition Matrix

	alive
alive	↑

Table 7: Methods Concurrency Matrix

	Game	wins	makemove	backmove	reset	toString
Game	∦	∦	∦	∦	∦	∦
wins	∦	‖	∦	∦	∦	‖
makemove	∦	∦	∦	∦	∦	∦
backmove	∦	∦	∦	∦	∦	∦
reset	∦	∦	∦	∦	∦	∦
toString	∦	‖	∦	∦	∦	‖

3 TransGame

Table 8: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
TransGame	√
transpose	√
hash	√
transrestore	√
transput	√
transtore	√
emptyTT	√
hitRate	√
result	√
htstat	√

Table 9: State Transition Matrix

	alive
alive	↑

Table 10: Methods Concurrency Matrix

	TransGame	transpose	hash	transrestore	transput	transtore	emptyTT	hitRate	result	htstat
TransGame	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
transpose	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
hash	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
transrestore	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
transput	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
transtore	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
emptyTT	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
hitRate	∦	∦	∦	∦	∦	∦	∦	‖	∦	∦
result	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
htstat	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦

4 JGFSearchBench

Table 11: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
JGFSearchBench	√
JGFapplication	√
JGFsetsize	√
JGFrun	√
JGFinitialise	√
JGFvalidate	√
JGFtidyup	√

Table 12: State Transition Matrix

	alive
alive	↑

Table 13: Methods Concurrency Matrix

	JGFSearchBench	JGFapplication	JGFsetsize	JGFrun	JGFinitialise	JGFvalidate	JGFtidyup
JGFSearchBench	∦	∦	∦	∦	∦	∦	∦
JGFapplication	∦	∦	∦	∦	∦	∦	∦
JGFsetsize	∦	∦	∦	∦	∦	∦	∦
JGFrun	∦	∦	∦	∦	∦	∦	∦
JGFinitialise	∦	∦	∦	∦	∦	∦	∦
JGFvalidate	∦	∦	∦	∦	∦	∦	∦
JGFtidyup	∦	∦	∦	∦	∦	∦	∦

5 JGFInstrumentor

Table 14: Method's Satisfiability(Code Reachabiity Analysis

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

Table 15: State Transition Matrix

	alive
alive	↑

Table 16: Methods Concurrency Matrix

	JGFInstrumentor	addTimer	addOpsToTimer	printTimer	startTimer	stopTimer	readTimer	resetTimer	printperfTimer	storeData	retrieveData	printHeader	main
JGFInstrumentor	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
addTimer	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
addOpsToTimer	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
printTimer	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
startTimer	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
stopTimer	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
readTimer	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
resetTimer	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
printperfTimer	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
storeData	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
retrieveData	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
printHeader	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
main	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦

6 JGFTimer

Table 17: Method's Satisfiability(Code Reachabiity Analysis

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

Table 18: State Transition Matrix

	alive
alive	$\uparrow$

Table 19: Methods Concurrency Matrix

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

7 JGFSearchBenchSizeA

Table 20: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
JGFSearchBenchSizeA	√
main	√

Table 21: State Transition Matrix

	alive
alive	↑

Table 22: Methods Concurrency Matrix

	JGFSearchBenchSizeA	main
JGFSearchBenchSizeA	∦	∦
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 the input program generated by Sip4J

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

4  @ClassStates({@State(name = "alive")})
5  class SearchGame {
6  @Perm(ensures="unique(this) in alive")
7  SearchGame() {   }

9  @Perm(requires="share(this) in alive",
10 ensures="share(this) in alive")
11   int solve() {
12  return 0;

14 }
15 @Perm(requires="share(this) in alive",
16 ensures="share(this) in alive")
17   int ab(int alpha, int beta) {
18  return 0;

20 }

22 }ENDOFCLASS

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

26 class Game {
27 @Perm(ensures="unique(this) in alive")
28 Game() {   }

30 @Perm(requires="pure(this) in alive",
31 ensures="pure(this) in alive")
32  final boolean wins(int n, int h, int sidemask) {
33  return 0;

35 }
36 @Perm(requires="share(this) in alive",
37 ensures="share(this) in alive")
38   void makemove(int n) {

40 }
41 @Perm(requires="share(this) in alive",
42 ensures="share(this) in alive")
43   void backmove() {

45 }
46 @Perm(requires="share(this) in alive",
47 ensures="share(this) in alive")
48   void reset() {

50 }
51 @Perm(requires="pure(this) in alive",
52 ensures="pure(this) in alive")
53  public String toString() {
54  return null;

56 }

58 }ENDOFCLASS

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

62 class TransGame {
63 @Perm(ensures="unique(this) in alive")
64 TransGame() {   }

66 @Perm(requires="share(this) in alive",
67 ensures="share(this) in alive")
68   int transpose() {
69  return 0;

71 }
72 @Perm(requires="share(this) in alive",
73 ensures="share(this) in alive")
74   void hash() {

76 }
77 @Perm(requires="share(this) in alive",
78 ensures="share(this) in alive")
```

```
79    void transrestore(int score, int work) {

81  }
82  @Perm(requires="share(this) in alive",
83  ensures="share(this) in alive")
84    void transput(int score, int work) {

86  }
87  @Perm(requires="share(this) in alive",
88  ensures="share(this) in alive")
89    void transtore(int score, int work) {

91  }
92  @Perm(requires="share(this) in alive",
93  ensures="share(this) in alive")
94    void emptyTT() {

96  }
97  @Perm(requires="pure(this) in alive",
98  ensures="pure(this) in alive")
99    double hitRate() {
100  return 0;

102 }
103 @Perm(requires="share(this) in alive",
104 ensures="share(this) in alive")
105   String result() {
106  return null;

108 }
109 @Perm(requires="share(this) in alive",
110 ensures="share(this) in alive")
111   String htstat() {
112  return null;

114 }

116 }ENDOFCLASS

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

120 class JGFSearchBench {
121 @Perm(ensures="unique(this) in alive")
122 JGFSearchBench() {   }

124 @Perm(requires="share(this) in alive",
125 ensures="share(this) in alive")
126  public void JGFapplication() {

128 }
129 @Perm(requires="share(this) in alive",
130 ensures="share(this) in alive")
131  public void JGFsetsize(int size) {

133 }
134 @Perm(requires="unique(this) in alive",
135 ensures="unique(this) in alive")
136  public void JGFrun(int size) {

138 }
139 @Perm(requires="share(this) in alive",
140 ensures="share(this) in alive")
141  public void JGFinitialise() {

143 }
144 @Perm(requires="share(this) in alive",
145 ensures="share(this) in alive")
146  public void JGFvalidate() {

148 }
149 @Perm(requires="unique(this) in alive",
150 ensures="unique(this) in alive")
151  public void JGFtidyup() {

153 }

155 }ENDOFCLASS

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

159 class JGFInstrumentor {
```

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

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

}
@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 printTimer(String name) {

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

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

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

```
241 }
242 @Perm(requires="pure(this) in alive",
243 ensures="pure(this) in alive")
244  public double perf() {
245  return 0;

247 }
248 @Perm(requires="share(this) in alive",
249 ensures="share(this) in alive")
250  public void reset() {

252 }
253 @Perm(requires="share(this) in alive",
254 ensures="share(this) in alive")
255  public void start() {

257 }
258 @Perm(requires="share(this) in alive",
259 ensures="share(this) in alive")
260  public void stop() {

262 }
263 @Perm(requires="pure(this) in alive",
264 ensures="pure(this) in alive")
265  public void longprint() {

267 }
268 @Perm(requires="pure(this) in alive",
269 ensures="pure(this) in alive")
270  public void printperf() {

272 }

274 }ENDOFCLASS

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

278 class JGFSearchBenchSizeA {
279 @Perm(ensures="unique(this) in alive")
280 JGFSearchBenchSizeA() {   }

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

286 }

288 }ENDOFCLASS
```