

Summary

Sink States:0(0×10^0)

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
StdRandom	25	1	0	0	1	325	1	0
MersenneTwisterFast	6	1	0	0	0	21	0	0
StdOut	5	1	0	0	0	15	0	0
Gaussian	8	1	0	0	7	36	23	64
SeqBlackScholes	5	1	0	0	1	15	1	7
BlackScholes	1	1	0	0	0	1	0	0
Total Classes=6	50	6	0	0	9	413	25	6

Contents

1	StdRandom	3
2	MersenneTwisterFast	5
3	StdOut	6
4	Gaussian	7
5	SeqBlackScholes	8
6	BlackScholes	9
7	Abbreviation	10
8	Annotated version of the input program generated by Sip4J	11

1 StdRandom









































































































































































Table 2: Method's Satisfiability(Code Reachability Analysis)

Method	Satisfiability
StdRandom	✓
setSeed	✓
getSeed	✓
uniformO1	✓
uniformO2	✓
random	✓
uniformO3	✓
uniform	✓
bernoulliO1	✓
bernoulliO2	✓
gaussianO1	✓
gaussianO2	✓
geometric	✓
poisson	✓
pareto	✓
cauchy	✓
discrete	✓
exp	✓
shuffleO1	✓
shuffleO2	✓
shuffleO3	✓
shuffleO4	✓
shuffleO5	✓
shuffleO6	✓
main	✓

Table 3: State Transition Matrix

	alive
alive	↑

Table 4: Methods Concurrency Matrix

	StdRandom	setSeed	getSeed	uniformO1	uniformO2	random	uniformO3	uniform	bernoulliO1	bernoulliO2	gaussianO1	gaussianO2	geometric	poisson	pareto	cauchy	discrete	exp	shuffleO1	shuffleO2	shuffleO3	shuffleO4	shuffleO5	shuffleO6	main
StdRandom																									
setSeed																									
getSeed																									
uniformO1																									
uniformO2																									
random																									
uniformO3																									

2 MersenneTwisterFast

Table 5: Method's Satisfiability(Code Reachability Analysis)

Method	Satisfiability
MersenneTwisterFast	✓
setSeed	✓
nextDouble	✓
nextInt	✓
nextShort	✓
nextBoolean	✓

Table 6: State Transition Matrix

	alive
alive	↑

Table 7: Methods Concurrency Matrix

	MersenneTwisterFast	setSeed	nextDouble	nextInt	nextShort	nextBoolean
MersenneTwisterFast	⌘	⌘	⌘	⌘	⌘	⌘
setSeed	⌘	⌘	⌘	⌘	⌘	⌘
nextDouble	⌘	⌘	⌘	⌘	⌘	⌘
nextInt	⌘	⌘	⌘	⌘	⌘	⌘
nextShort	⌘	⌘	⌘	⌘	⌘	⌘
nextBoolean	⌘	⌘	⌘	⌘	⌘	⌘

3 StdOut

Table 8: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
StdOut	✓
println	✓
printf	✓
print	✓
close	✓

Table 9: State Transition Matrix

	alive
alive	↑

Table 10: Methods Concurrency Matrix

	StdOut	println	printf	print	close
StdOut	⌘	⌘	⌘	⌘	⌘
println	⌘	⌘	⌘	⌘	⌘
printf	⌘	⌘	⌘	⌘	⌘
print	⌘	⌘	⌘	⌘	⌘
close	⌘	⌘	⌘	⌘	⌘

4 Gaussian

Table 11: Method's Satisfiability(Code Reachability Analysis)

Method	Satisfiability
Gaussian	✓
phi	✓
phiOverload	✓
PhiOverload	✓
Phi	✓
PhiInverse	✓
PhiInverseOverload	✓
main	✓

Table 12: State Transition Matrix

	alive
alive	↑

Table 13: Methods Concurrency Matrix

	Gaussian	phi	phiOverload	PhiOverload	Phi	PhiInverse	PhiInverseOverload	main
Gaussian	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
phi	⌘							⌘
phiOverload	⌘							⌘
PhiOverload	⌘							⌘
Phi	⌘							
PhiInverse	⌘							
PhiInverseOverload	⌘							⌘
main	⌘	⌘	⌘	⌘			⌘	⌘

5 SeqBlackScholes

Table 14: Method's Satisfiability(Code Reachability Analysis)

Method	Satisfiability
SeqBlackScholes	✓
callPrice	✓
call	✓
call2	✓
main	✓

Table 15: State Transition Matrix

	alive
alive	↑

Table 16: Methods Concurrency Matrix

	SeqBlackScholes	callPrice	call	call2	main
SeqBlackScholes	⌘	⌘	⌘	⌘	⌘
callPrice	⌘	⌘	⌘	⌘	⌘
call	⌘	⌘	⌘	⌘	⌘
call2	⌘	⌘	⌘	⌘	⌘
main	⌘	⌘	⌘	⌘	⌘

6 BlackScholes

Table 17: Method’s Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
BlackScholes	✓

Table 18: State Transition Matrix

	alive
alive	↑

7 Abbreviation

Table 19: Used Abbreviation

Symbol	Meaning
✓	requires clause of the method is satisfiable
✗	requires clause of the method is unsatisfiable
↑	The row-state can be transitioned to the column-state
✕	The row-state cannot be transitioned to the column-state
	The row-method can be possibly executed parallel with the column-method
⋈	The row-method cannot be executed parallel with the column-method

8 Annotated version of the input program generated by Sip4J

```
1 package outputs;
2 import edu.cmu.cs.plural.annot.*;
3
4 @ClassStates({@State(name = "alive")})
5 class StdRandom {
6   @Perm(ensures="unique(this) in alive")
7   StdRandom() { }
8
9   @Perm(requires="unique(this) in alive",
10  ensures="unique(this) in alive")
11   void setSeed(long s) {
12
13   }
14   @Perm(requires="pure(this) in alive",
15  ensures="pure(this) in alive")
16   long getSeed() {
17     return 0;
18
19   }
20   @Perm(requires="share(this) in alive",
21  ensures="share(this) in alive")
22   double uniform01() {
23     return 0;
24
25   }
26   @Perm(requires="share(this) in alive",
27  ensures="share(this) in alive")
28   int uniform02(int N) {
29     return 0;
30
31   }
32   @Perm(requires="share(this) in alive",
33  ensures="share(this) in alive")
34   double random() {
35     return 0;
36
37   }
38   @Perm(requires="share(this) in alive",
39  ensures="share(this) in alive")
40   int uniform03(int a, int b) {
41     return 0;
42
43   }
44   @Perm(requires="share(this) in alive",
45  ensures="share(this) in alive")
46   double uniform(double a, double b) {
47     return 0;
48
49   }
50   @Perm(requires="share(this) in alive",
51  ensures="share(this) in alive")
52   boolean bernoulli01(double p) {
53     return 0;
54
55   }
56   @Perm(requires="share(this) in alive",
57  ensures="share(this) in alive")
58   boolean bernoulli02() {
59     return 0;
60
61   }
62   @Perm(requires="share(this) in alive",
63  ensures="share(this) in alive")
64   double gaussian01() {
65     return 0;
66
67   }
68   @Perm(requires="share(this) in alive",
69  ensures="share(this) in alive")
70   double gaussian02(double mean, double stddev) {
71     return 0;
72
73   }
74   @Perm(requires="share(this) in alive",
75  ensures="share(this) in alive")
76   int geometric(double p) {
77     return 0;
```

```

79 }
80 @Perm(requires="share(this) in alive",
81 ensures="share(this) in alive")
82 int poisson(double lambda) {
83     return 0;
84 }
85 }
86 @Perm(requires="share(this) in alive",
87 ensures="share(this) in alive")
88 double pareto(double alpha) {
89     return 0;
90 }
91 }
92 @Perm(requires="share(this) in alive",
93 ensures="share(this) in alive")
94 double cauchy() {
95     return 0;
96 }
97 }
98 @Perm(requires="share(this) in alive",
99 ensures="share(this) in alive")
100 int discrete(double[] a) {
101     return 0;
102 }
103 }
104 @Perm(requires="share(this) in alive",
105 ensures="share(this) in alive")
106 double exp(double lambda) {
107     return 0;
108 }
109 }
110 @Perm(requires="share(this) in alive",
111 ensures="share(this) in alive")
112 void shuffle01(Object[] a) {
113 }
114 }
115 @Perm(requires="share(this) in alive",
116 ensures="share(this) in alive")
117 void shuffle02(double[] a) {
118 }
119 }
120 @Perm(requires="share(this) in alive",
121 ensures="share(this) in alive")
122 void shuffle03(int[] a) {
123 }
124 }
125 @Perm(requires="share(this) in alive",
126 ensures="share(this) in alive")
127 void shuffle04(Object[] a, int lo, int hi) {
128 }
129 }
130 @Perm(requires="share(this) in alive",
131 ensures="share(this) in alive")
132 void shuffle05(double[] a, int lo, int hi) {
133 }
134 }
135 @Perm(requires="share(this) in alive",
136 ensures="share(this) in alive")
137 void shuffle06(int[] a, int lo, int hi) {
138 }
139 }
140 @Perm(requires="unique(this) in alive",
141 ensures="unique(this) in alive")
142 void main(String[] args) {
143 }
144 }
145 }
146 }ENDOFCLASS
147
148 @ClassStates({@State(name = "alive")})
149
150 class MersenneTwisterFast {
151     @Perm(ensures="unique(this) in alive")
152     MersenneTwisterFast() { }
153
154     @Perm(requires="unique(this) in alive",
155 ensures="unique(this) in alive")
156     void setSeed(final long seed) {
157 }
158 }
159 @Perm(requires="share(this) in alive",

```

```

160 ensures="share(this) in alive")
161     double nextDouble() {
162         return 0;
163     }
164 }
165 @Perm(requires="share(this) in alive",
166 ensures="share(this) in alive")
167     int nextInt(final int n) {
168         return 0;
169     }
170 }
171 @Perm(requires="share(this) in alive",
172 ensures="share(this) in alive")
173     short nextShort() {
174         return 0;
175     }
176 }
177 @Perm(requires="share(this) in alive",
178 ensures="share(this) in alive")
179     boolean nextBoolean() {
180         return 0;
181     }
182 }
183 }ENDOFCLASS
184
185 @ClassStates({@State(name = "alive")})
186
187 class StdOut {
188     @Perm(ensures="unique(this) in alive")
189     StdOut() { }
190 }
191
192 @Perm(requires="share(this) in alive",
193 ensures="share(this) in alive")
194     void println(Object x) {
195 }
196
197 @Perm(requires="share(this) in alive",
198 ensures="share(this) in alive")
199     void printf(String format, Object... args) {
200 }
201
202 @Perm(requires="share(this) in alive",
203 ensures="share(this) in alive")
204     void print(Object x) {
205 }
206
207 @Perm(requires="share(this) in alive",
208 ensures="share(this) in alive")
209     void close() {
210 }
211 }
212 }ENDOFCLASS
213
214 @ClassStates({@State(name = "alive")})
215
216 class Gaussian {
217     @Perm(ensures="unique(this) in alive")
218     Gaussian() { }
219 }
220
221 @Perm(requires="pure(this) in alive",
222 ensures="pure(this) in alive")
223     double phi(double x) {
224         return 0;
225     }
226 }
227 @Perm(requires="pure(this) in alive",
228 ensures="pure(this) in alive")
229     double phiOverload(double x, double mu, double sigma) {
230         return 0;
231     }
232 }
233 @Perm(requires="pure(this) in alive",
234 ensures="pure(this) in alive")
235     double PhiOverload(double z) {
236         return 0;
237     }
238 }
239
240     double Phi(double z, double mu, double sigma) {

```

```

241     return 0;
242 }
243
244 double PhiInverse(double y, double delta, double lo, double hi) {
245     return 0;
246 }
247
248 @Perm(requires="pure(this) in alive",
249 ensures="pure(this) in alive")
250 double PhiInverseOverload(double y) {
251     return 0;
252 }
253
254 @Perm(requires="unique(this) in alive",
255 ensures="unique(this) in alive")
256 void main(String[] args) {
257 }
258
259 }
260
261 }ENDOFCLASS
262
263 @ClassStates({@State(name = "alive")})
264
265 class SeqBlackScholes {
266     @Perm(ensures="unique(this) in alive")
267     SeqBlackScholes() { }
268
269     @Perm(requires="immutable(this) in alive",
270 ensures="immutable(this) in alive")
271     double callPrice(double S, double X, double r, double sigma, double T) {
272         return 0;
273     }
274
275     @Perm(requires="share(this) in alive",
276 ensures="share(this) in alive")
277     double call(double S, double X, double r, double sigma, double T, long N) {
278         return 0;
279     }
280
281     @Perm(requires="share(this) in alive",
282 ensures="share(this) in alive")
283     double call2(double S, double X, double r, double sigma, double T, long N) {
284         return 0;
285     }
286
287     @Perm(requires="unique(this) in alive",
288 ensures="unique(this) in alive")
289     void main(String[] args) {
290 }
291
292 }ENDOFCLASS
293
294 @ClassStates({@State(name = "alive")})
295
296 class BlackScholes {
297     @Perm(ensures="unique(this) in alive")
298     BlackScholes() { }
299 }
300
301 }ENDOFCLASS

```