# 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 |
|---|---|---|---|---|---|---|---|---|
| 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 | 6 | 36 | 21 | 58 |
| 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 | 8 | 413 | 23 | 6 |

# Contents

# 1 StdRandom

Table 2: Methods Requires Clause Satisfiability

| 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 | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |

| | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| uniform | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ |
| bernoulliO1 | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ |
| bernoulliO2 | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ |
| gaussianO1 | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ |
| gaussianO2 | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ |
| geometric | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ |
| poisson | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ |
| pareto | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ |
| cauchy | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ |
| discrete | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ |
| exp | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ |
| shuffleO1 | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ |
| shuffleO2 | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ |
| shuffleO3 | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ |
| shuffleO4 | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ |
| shuffleO5 | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ |
| shuffleO6 | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ |
| main | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ | ≠ |

## 2 MersenneTwisterFast

Table 5: Methods Requires Clause Satisfiability

| 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: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| StdOut | $\surd$ |
| println | $\surd$ |
| printf | $\surd$ |
| print | $\surd$ |
| close | $\surd$ |

Table 9: State Transition Matrix

| | alive |
|---|---|
| alive | $\uparrow$ |

Table 10: Methods Concurrency Matrix

| | StdOut | println | printf | print | close |
|---|---|---|---|---|---|
| StdOut | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| println | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| printf | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| print | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |
| close | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ | $\nparallel$ |

# 4 Gaussian

Table 11: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| Gaussian | √ |
| phi | √ |
| phiOverload | √ |
| PhiOverload | √ |
| PhiInverse | √ |
| PhiInverseOverload | √ |
| main | √ |
| Phi | √ |

Table 12: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

Table 13: Methods Concurrency Matrix

| | Gaussian | phi | phiOverload | PhiOverload | PhiInverse | PhiInverseOverload | main | Phi |
|---|---|---|---|---|---|---|---|---|
| Gaussian | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| phi | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∦ | ∥ |
| phiOverload | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∦ | ∥ |
| PhiOverload | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∦ | ∥ |
| PhiInverse | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∦ | ∥ |
| PhiInverseOverload | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∦ | ∥ |
| main | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ | ∦ |
| Phi | ∦ | ∥ | ∥ | ∥ | ∥ | ∥ | ∦ | ∥ |

# 5 SeqBlackScholes

Table 14: Methods Requires Clause Satisfiability

| 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: Methods Requires Clause Satisfiability

| Method | Satisfiability |
|---|---|
| BlackScholes | √ |

Table 18: State Transition Matrix

| | alive |
|---|---|
| alive | ↑ |

# 7 Abbreviation

Table 19: 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 |

# 8 Annotated Version of Sequential Java 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 @Perm(requires="pure(this) in alive",
14 ensures="pure(this) in alive")
15  long getSeed() {
16  return 0;
17 }
18 @Perm(requires="full(this) in alive",
19 ensures="full(this) in alive")
20  double uniform01() {
21  return 0;
22 }
23 @Perm(requires="full(this) in alive",
24 ensures="full(this) in alive")
25  int uniform02(int N) {
26  return 0;
27 }
28 @Perm(requires="full(this) in alive",
29 ensures="full(this) in alive")
30  double random() {
31  return 0;
32 }
33 @Perm(requires="full(this) in alive",
34 ensures="full(this) in alive")
35  int uniform03(int a, int b) {
36  return 0;
37 }
38 @Perm(requires="full(this) in alive",
39 ensures="full(this) in alive")
40  double uniform(double a, double b) {
41  return 0;
42 }
43 @Perm(requires="full(this) in alive",
44 ensures="full(this) in alive")
45  boolean bernoulli01(double p) {
46  return 0;
47 }
48 @Perm(requires="full(this) in alive",
49 ensures="full(this) in alive")
50  boolean bernoulli02() {
51  return 0;
52 }
53 @Perm(requires="full(this) in alive",
54 ensures="full(this) in alive")
55  double gaussian01() {
56  return 0;
57 }
58 @Perm(requires="full(this) in alive",
59 ensures="full(this) in alive")
60  double gaussian02(double mean, double stddev) {
61  return 0;
62 }
63 @Perm(requires="full(this) in alive",
64 ensures="full(this) in alive")
65  int geometric(double p) {
66  return 0;
67 }
68 @Perm(requires="full(this) in alive",
69 ensures="full(this) in alive")
70  int poisson(double lambda) {
71  return 0;
72 }
73 @Perm(requires="full(this) in alive",
74 ensures="full(this) in alive")
75  double pareto(double alpha) {
```

```
 return 0;
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 double cauchy() {
 return 0;
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 int discrete(double[] a) {
 return 0;
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 double exp(double lambda) {
 return 0;
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 void shuffle01(Object[] a) {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 void shuffle02(double[] a) {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 void shuffle03(int[] a) {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 void shuffle04(Object[] a, int lo, int hi) {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 void shuffle05(double[] a, int lo, int hi) {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 void shuffle06(int[] a, int lo, int hi) {
}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 void main(String[] args) {
}

}ENDOFCLASS

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

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

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 void setSeed(final long seed) {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 double nextDouble() {
 return 0;
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 int nextInt(final int n) {
 return 0;
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 short nextShort() {
 return 0;
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 boolean nextBoolean() {
 return 0;
}

}ENDOFCLASS
```

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

159 class StdOut {
160 @Perm(ensures="unique(this) in alive")
161 StdOut() {   }

163 @Perm(requires="full(this) in alive",
164 ensures="full(this) in alive")
165  void println(Object x) {
166 }
167 @Perm(requires="full(this) in alive",
168 ensures="full(this) in alive")
169  void printf(String format, Object... args) {
170 }
171 @Perm(requires="full(this) in alive",
172 ensures="full(this) in alive")
173  void print(Object x) {
174 }
175 @Perm(requires="full(this) in alive",
176 ensures="full(this) in alive")
177  void close() {
178 }

180 }ENDOFCLASS

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

184 class Gaussian {
185 @Perm(ensures="unique(this) in alive")
186 Gaussian() {   }

188 @Perm(requires="pure(this) in alive",
189 ensures="pure(this) in alive")
190  double phi(double x) {
191  return 0;
192 }
193 @Perm(requires="pure(this) in alive",
194 ensures="pure(this) in alive")
195  double phiOverload(double x, double mu, double sigma) {
196  return 0;
197 }
198 @Perm(requires="pure(this) in alive",
199 ensures="pure(this) in alive")
200  double PhiOverload(double z) {
201  return 0;
202 }
203 @Perm(requires="pure(this) in alive",
204 ensures="pure(this) in alive")
205  double PhiInverse(double y, double delta, double lo, double hi) {
206  return 0;
207 }
208 @Perm(requires="pure(this) in alive",
209 ensures="pure(this) in alive")
210  double PhiInverseOverload(double y) {
211  return 0;
212 }
213 @Perm(requires="unique(this) in alive",
214 ensures="unique(this) in alive")
215  void main(String[] args) {
216 }
217 @Perm(requires="pure(this) in alive",
218 ensures="pure(this) in alive")
219  double Phi(double z, double mu, double sigma) {
220  return 0;
221 }

223 }ENDOFCLASS

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

227 class SeqBlackScholes {
228 @Perm(ensures="unique(this) in alive")
229 SeqBlackScholes() {   }

231 @Perm(requires="pure(this) in alive",
232 ensures="pure(this) in alive")
233  double callPrice(double S, double X, double r, double sigma, double T) {
234  return 0;
235 }
236 @Perm(requires="full(this) in alive",
237 ensures="full(this) in alive")
```

```
238   double call(double S, double X, double r, double sigma, double T, long N) {
239   return 0;
240  }
241  @Perm(requires="full(this) in alive",
242  ensures="full(this) in alive")
243   double call2(double S, double X, double r, double sigma, double T, long N) {
244   return 0;
245  }
246  @Perm(requires="unique(this) in alive",
247  ensures="unique(this) in alive")
248   void main(String[] args) {
249  }

251  }ENDOFCLASS

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

255  class BlackScholes {
256  @Perm(ensures="unique(this) in alive")
257  BlackScholes() {   }


260  }ENDOFCLASS
```