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
Item	1	1	0	0	0	1	0	0
SeqGA	2	1	0	0	1	3	1	33
Knapsack	7	1	0	0	6	28	14	50
MersenneTwisterFast	6	1	0	0	0	21	0	0
Indiv	3	1	0	0	2	6	2	33
ComparatorOnFitness	2	1	0	0	1	3	1	33
Total Classes=6	21	6	0	0	10	62	18	29

Contents

1 Item

Table 2: Methods Requires Clause Satisfiability

Method	Satisfiability
Item	√

Table 3: State Transition Matrix

	alive
alive	↑

2 SeqGA

Table 4: Methods Requires Clause Satisfiability

Method	Satisfiability
SeqGA	√
main	√

Table 5: State Transition Matrix

	alive
alive	↑

Table 6: Methods Concurrency Matrix

	SeqGA	main
SeqGA	∦	∦
main	∦	‖

3 Knapsack

Table 7: Methods Requires Clause Satisfiability

Method	Satisfiability
Knapsack	$\surd$
resetSeed	$\surd$
createRandomIndiv	$\surd$
recombine	$\surd$
phenotype	$\surd$
evaluate	$\surd$
mutate	$\surd$

Table 8: State Transition Matrix

	alive
alive	$\uparrow$

Table 9: Methods Concurrency Matrix

	Knapsack	resetSeed	createRandomIndiv	recombine	phenotype	evaluate	mutate
Knapsack	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
resetSeed	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
createRandomIndiv	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$	$\nparallel$
recombine	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$	$\nparallel$
phenotype	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
evaluate	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
mutate	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$	$\nparallel$

4 MersenneTwisterFast

Table 10: Methods Requires Clause Satisfiability

Method	Satisfiability
MersenneTwisterFast	√
nextDouble	√
nextInt	√
setSeed	√
nextBoolean	√
nextFloat	√

Table 11: State Transition Matrix

	alive
alive	↑

Table 12: Methods Concurrency Matrix

	MersenneTwisterFast	nextDouble	nextInt	setSeed	nextBoolean	nextFloat
MersenneTwisterFast	∦	∦	∦	∦	∦	∦
nextDouble	∦	∦	∦	∦	∦	∦
nextInt	∦	∦	∦	∦	∦	∦
setSeed	∦	∦	∦	∦	∦	∦
nextBoolean	∦	∦	∦	∦	∦	∦
nextFloat	∦	∦	∦	∦	∦	∦

5 Indiv

Table 13: Methods Requires Clause Satisfiability

Method	Satisfiability
Indiv	√
set	√
compareTo	√

Table 14: State Transition Matrix

	alive
alive	↑

Table 15: Methods Concurrency Matrix

	Indiv	set	compareTo
Indiv	∦	∦	∦
set	∦	∦	‖
compareTo	∦	‖	‖

6 ComparatorOnFitness

Table 16: Methods Requires Clause Satisfiability

Method	Satisfiability
ComparatorOnFitness	√
compare	√

Table 17: State Transition Matrix

	alive
alive	↑

Table 18: Methods Concurrency Matrix

	ComparatorOnFitness	compare
ComparatorOnFitness	∦	∦
compare	∦	‖

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.*;

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


10 }ENDOFCLASS

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

14 class SeqGA {
15 @Perm(ensures="unique(this) in alive")
16 SeqGA() {   }

18 @Perm(requires="none(this) in alive",
19 ensures="unique(this) in alive")
20  void main(String[] args) {
21 }

23 }ENDOFCLASS

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

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

31 @Perm(requires="full(#0) in alive",
32 ensures="full(#0) in alive")
33  void resetSeed(MersenneTwisterFast prand) {
34 }
35 @Perm(requires="full(this) in alive",
36 ensures="full(this) in alive")
37  Indiv createRandomIndiv(Indiv ind) {
38  return null;
39 }
40 @Perm(requires="full(this) in alive",
41 ensures="full(this) in alive")
42  Indiv recombine(Indiv ind, Indiv p1, Indiv p2) {
43  return null;
44 }
45 @Perm(requires="pure(this) in alive",
46 ensures="pure(this) in alive")
47  int[] phenotype(Indiv indiv) {
48  return null;
49 }
50 @Perm(requires="pure(this) in alive",
51 ensures="pure(this) in alive")
52  void evaluate(Indiv indiv) {
53 }
54 @Perm(requires="full(this) in alive",
55 ensures="full(this) in alive")
56  void mutate(Indiv indiv) {
57 }

59 }ENDOFCLASS

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

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

67 @Perm(requires="full(this) in alive",
68 ensures="full(this) in alive")
69  double nextDouble() {
70  return 0;
71 }
72 @Perm(requires="full(this) * pure(#0) in alive",
73 ensures="full(this) * pure(#0) in alive")
74  int nextInt(final int n) {
75  return 0;
```

```
76  }
77  @Perm(requires="unique(this) in alive",
78  ensures="unique(this) in alive")
79   void setSeed(final long seed) {
80  }
81  @Perm(requires="full(this) in alive",
82  ensures="full(this) in alive")
83   boolean nextBoolean() {
84   return 0;
85  }
86  @Perm(requires="full(this) in alive",
87  ensures="full(this) in alive")
88   float nextFloat() {
89   return 0;
90  }

92  }ENDOFCLASS

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

96  class Indiv {
97  @Perm(ensures="unique(this) in alive")
98  Indiv() {   }

100 @Perm(requires="full(this) in alive",
101 ensures="full(this) in alive")
102 public void set(int w, boolean h) {
103 }
104 @Perm(requires="pure(this) in alive",
105 ensures="pure(this) in alive")
106 public int compareTo(Indiv other) {
107  return 0;
108 }

110 }ENDOFCLASS

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

114 class ComparatorOnFitness {
115 @Perm(ensures="unique(this) in alive")
116 ComparatorOnFitness() {   }

118 @Perm(requires="pure(this) in alive",
119 ensures="pure(this) in alive")
120 public int compare(Integer a, Integer b) {
121  return 0;
122 }

124 }ENDOFCLASS
```