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
Patient	1	1	0	0	0	1	0	0
Village	10	1	0	0	9	55	9	16
Health	2	1	0	0	1	3	1	33
SeqHealth	3	1	0	0	2	6	3	50
Results	1	1	0	0	0	1	0	0
Hosp	1	1	0	0	0	1	0	0
Total Classes=6	18	6	0	0	12	67	13	19

Contents

1 Patient

Table 2: Methods Requires Clause Satisfiability

Method	Satisfiability
Patient	√

Table 3: State Transition Matrix

	alive
alive	↑

2 Village

Table 4: Methods Requires Clause Satisfiability

Method	Satisfiability
Village	√
tick	√
checkPatientsInside	√
checkPatientsAssess	√
checkPatientsWaiting	√
checkPatientsRealloc	√
putInHosp	√
checkPatientsPopulation	√
displayVillageData	√
DisplayVillagePatients	√

Table 5: State Transition Matrix

	alive
alive	↑

Table 6: Methods Concurrency Matrix

	Village	tick	checkPatientsInside	checkPatientsAssess	checkPatientsWaiting	checkPatientsRealloc	putInHosp	checkPatientsPopulation	displayVillageData	DisplayVillagePatients
Village	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
tick	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖
checkPatientsInside	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖
checkPatientsAssess	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖
checkPatientsWaiting	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖
checkPatientsRealloc	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖
putInHosp	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖
checkPatientsPopulation	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖
displayVillageData	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖
DisplayVillagePatients	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖

3 Health

Table 7: Methods Requires Clause Satisfiability

Method	Satisfiability
Health	√
allocateVillage	√

Table 8: State Transition Matrix

	alive
alive	↑

Table 9: Methods Concurrency Matrix

	Health	allocateVillage
Health	∦	∦
allocateVillage	∦	‖

4 SeqHealth

Table 10: Methods Requires Clause Satisfiability

Method	Satisfiability
SeqHealth	√
main	√
simVillage	√

Table 11: State Transition Matrix

	alive
alive	↑

Table 12: Methods Concurrency Matrix

	SeqHealth	main	simVillage
SeqHealth	⫽	⫽	⫽
main	⫽	‖	‖
simVillage	⫽	‖	‖

5 Results

Table 13: Methods Requires Clause Satisfiability

Method	Satisfiability
Results	√

Table 14: State Transition Matrix

	alive
alive	↑

6 Hosp

Table 15: Methods Requires Clause Satisfiability

Method	Satisfiability
Hosp	√

Table 16: State Transition Matrix

	alive
alive	↑

7 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

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 Patient {
6  @Perm(ensures="unique(this) in alive")
7  Patient() {   }


10 }ENDOFCLASS

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

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

18 @Perm(requires="full(this) in alive",
19 ensures="full(this) in alive")
20 public void tick() {
21 }
22 @Perm(requires="full(this) in alive",
23 ensures="full(this) in alive")
24 public void checkPatientsInside() {
25 }
26 @Perm(requires="full(this) in alive",
27 ensures="full(this) in alive")
28 public void checkPatientsAssess() {
29 }
30 @Perm(requires="full(this) in alive",
31 ensures="full(this) in alive")
32 public void checkPatientsWaiting() {
33 }
34 @Perm(requires="full(this) in alive",
35 ensures="full(this) in alive")
36 public void checkPatientsRealloc() {
37 }
38 @Perm(requires="full(this) in alive",
39 ensures="full(this) in alive")
40 public void putInHosp(Patient p) {
41 }
42 @Perm(requires="full(this) in alive",
43 ensures="full(this) in alive")
44 public void checkPatientsPopulation() {
45 }
46 @Perm(requires="full(this) in alive",
47 ensures="full(this) in alive")
48  void displayVillageData(Village v) {
49 }
50 @Perm(requires="pure(this) in alive",
51 ensures="pure(this) in alive")
52 static void DisplayVillagePatients(Village v) {
53 }

55 }ENDOFCLASS

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

59 class Health {
60 @Perm(ensures="unique(this) in alive")
61 Health() {   }

63 @Perm(requires="none(this) in alive * pure(#0) in alive",
64 ensures="unique(this) in alive * pure(#0) in alive")
65  Village allocateVillage(int level, int vid, Village back) {
66  return null;
67 }

69 }ENDOFCLASS

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

73 class SeqHealth {
74 @Perm(ensures="unique(this) in alive")
75 SeqHealth() {   }
```

```
@Perm(requires="none(this) in alive",
ensures="unique(this) in alive")
 void main(String[] args) {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 void simVillage(Village village) {
}

}ENDOFCLASS

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

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


}ENDOFCLASS

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

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


}ENDOFCLASS
```