Summary

Sink States:0(0×10^0)

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
MTTS	5	1	0	0	4	15	7	47
TaskData	2	1	0	0	1	3	1	33
Total Classes=2	7	2	0	0	5	18	8	44

Contents

1 MTTS

Table 2: Methods Requires Clause Satisfiability

Method	Satisfiability
MTTS	$\surd$
setData	$\surd$
getData	$\surd$
execute	$\surd$
getTaskStatus	$\surd$

Table 3: State Transition Matrix

	alive
alive	$\uparrow$

Table 4: Methods Concurrency Matrix

	MTTS	setData	getData	execute	getTaskStatus
MTTS	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
setData	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$
getData	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
execute	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$
getTaskStatus	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$

2 TaskData

Table 5: Methods Requires Clause Satisfiability

Method	Satisfiability
TaskData	$\surd$
main	$\surd$

Table 6: State Transition Matrix

	alive
alive	$\uparrow$

Table 7: Methods Concurrency Matrix

	TaskData	main
TaskData	$\nparallel$	$\nparallel$
main	$\nparallel$	$\parallel$

3 Abbreviation

Table 8: 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

4 Annotated Version of Sequential Java Program generated by Sip4j

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

@ClassStates({@State(name = "alive")})
class MTTS {
@Perm(ensures="unique(this) in alive")
MTTS() {   }

@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
public void setData(TaskData d) {
}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
public TaskData getData() {
 return null;
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
public void execute(TaskData d) {
}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
public TaskData getTaskStatus() {
 return null;
}

}ENDOFCLASS

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

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

@Perm(requires="none(this) in alive",
ensures="unique(this) in alive")
 void main(String args) {
}

}ENDOFCLASS
```