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
Webserver	8	1	0	0	7	36	22	61
Server	2	1	0	0	0	3	0	0
Client	2	1	0	0	1	3	1	33
Total Classes=3	12	3	0	0	8	42	23	55

Contents

1 Webserver

Table 2: Methods Requires Clause Satisfiability

Method	Satisfiability
Webserver	√
main	√
webserver	√
serveClient	√
LOG	√
transfer	√
transferHeader	√
transferData	√

Table 3: State Transition Matrix

	alive
alive	↑

Table 4: Methods Concurrency Matrix

	Webserver	main	webserver	serveClient	LOG	transfer	transferHeader	transferData
Webserver	∦	∦	∦	∦	∦	∦	∦	∦
main	∦	∦	∦	∦	∥	∥	∥	∥
webserver	∦	∦	∦	∦	∥	∥	∥	∥
serveClient	∦	∦	∦	∦	∥	∥	∥	∥
LOG	∦	∥	∥	∥	∥	∥	∥	∥
transfer	∦	∥	∥	∥	∥	∥	∥	∥
transferHeader	∦	∥	∥	∥	∥	∥	∥	∥
transferData	∦	∥	∥	∥	∥	∥	∥	∥

2 Server

Table 5: Methods Requires Clause Satisfiability

Method	Satisfiability
Server	✓
main	✓

Table 6: State Transition Matrix

	alive
alive	↑

Table 7: Methods Concurrency Matrix

	Server	main
Server	∦	∦
main	∦	∦

3   Client

Table 8: Methods Requires Clause Satisfiability

Method	Satisfiability
Client	√
main	√

Table 9: State Transition Matrix

	alive
alive	↑

Table 10: Methods Concurrency Matrix

	Client	main
Client	∦	∦
main	∦	∥

4 Abbreviation

Table 11: 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

5 Annotated Version of Sequential Java Program generated by Sip4j

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

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

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 void main(String[] args) {
}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 void webserver() {
}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 void serveClient(Socket socketClient) {
}

 void LOG(String msg, Object... args) {
}

 void transfer(OutputStream outStream, File file) {
}

 void transferHeader(Writer writer, File file) {
}

 void transferData(Writer outWriter, Reader inReader, long count) {
}

}ENDOFCLASS

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

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

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 void main(String argv[]) {
}

}ENDOFCLASS

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

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


 void main(String argv[]) {
}

}ENDOFCLASS
```