Summary

Sink States:$0(0 \times 10^0)$

Table 1: Sip4J Analysis Summary

Classes	Methods	States	Unreachable clauses	Unreachable states	Possible concurrent methods	Total. no. of method pairs	No. of concurrent method pairs	Percentage of concurrent methods pairs
SampleAction	5	1	0	0	4	15	7	47
JMLAnnotatedJavaClass	7	1	0	0	6	28	11	39
PluralParser	41	1	0	0	41	861	83	10
EJmlSpecification	14	1	0	0	1	105	1	1
EGhost	6	1	0	0	3	21	6	29
Time	2	1	0	0	0	3	0	0
FileReader	2	1	0	0	1	3	1	33
UserSelectedClassesAnalysis	14	1	0	0	12	105	42	40
EVMDDSMCGenerator	16	1	0	0	15	136	15	11
EPackage	8	1	0	0	3	36	6	17
EClass	26	1	0	0	13	351	91	26
EMethod	22	1	0	0	6	253	21	8
ESpecification	8	1	0	0	7	36	10	28
EGeneratedPluralSpecification	3	1	0	0	0	6	0	0
ESMCModel	65	1	0	0	64	2145	74	3
EField	11	1	0	0	3	66	6	9
EDim	5	1	0	0	2	15	3	20
EParameter	9	1	0	0	4	45	10	22
EState	11	1	0	0	5	66	15	23
EInvariant	11	1	0	0	10	66	20	30
EBoolInvariant	3	1	0	0	2	6	3	50
EGrarphWriter	6	1	0	0	1	21	1	5
EOutputLatex	28	1	0	0	0	406	0	0
WorkspaceUtilities	9	1	0	0	8	45	30	67
SMCVisitor	7	1	0	0	6	28	15	54
PulseSettings	11	1	0	0	5	66	15	23
specificationStruct	1	1	0	0	0	1	0	0
Clause	1	1	0	0	0	1	0	0
Signature	1	1	0	0	0	1	0	0
MethodFindVisitor	2	1	0	0	0	3	0	0
Activator	5	1	0	0	1	15	1	7
GAPHandler	8	1	0	0	7	36	25	69

GAPIFileAction	4	1	0	0	3	10	3	30
Anonymous	2	1	0	0	0	3	0	0
Main	5	1	0	0	4	15	7	47
TypestateReturn	1	1	0	0	0	1	0	0
AtApPermissionReturn	1	1	0	0	0	1	0	0
AccesspermissionReturn	1	1	0	0	0	1	0	0
PluralLexer	63	1	0	0	62	2016	1892	94
DFA7	2	1	0	0	1	3	1	33
EAPTypeState	5	1	0	0	0	15	0	0
Total Classes=41	452	41	0	0	300	7056	2415	34

Contents

1 SampleAction

Table 2: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
SampleAction	√
run	√
selectionChanged	√
dispose	√
init	√

Table 3: State Transition Matrix

	alive
alive	↑

Table 4: Methods Concurrency Matrix

	SampleAction	run	selectionChanged	dispose	init
SampleAction	∦	∦	∦	∦	∦
run	∦	∦	∥	∥	∦
selectionChanged	∦	∥	∥	∥	∥
dispose	∦	∥	∥	∥	∥
init	∦	∦	∥	∥	∦

2 JMLAnnotatedJavaClass

Table 5: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
JMLAnnotatedJavaClass	√
translateJMLAnnotationsToPlural	√
translateClassSpecifications	√
parseAndStoreJMLAnnotation	√
translateMethodSpecification	√
getInputStream	√
readFileAsString	√

Table 6: State Transition Matrix

	alive
alive	↑

Table 7: Methods Concurrency Matrix

	JMLAnnotatedJavaClass	translateJMLAnnotationsToPlural	translateClassSpecifications	parseAndStoreJMLAnnotation	translateMethodSpecification	getInputStream	readFileAsString
JMLAnnotatedJavaClass	∦	∦	∦	∦	∦	∦	∦
translateJMLAnnotationsToPlural	∦	∦	∦	∦	∦	‖	‖
translateClassSpecifications	∦	∦	∦	∦	∦	‖	‖
parseAndStoreJMLAnnotation	∦	∦	∦	∦	∦	‖	‖
translateMethodSpecification	∦	∦	∦	∦	∦	‖	‖
getInputStream	∦	‖	‖	‖	‖	‖	‖
readFileAsString	∦	‖	‖	‖	‖	‖	‖

3 PluralParser

Table 8: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
PluralParser	√
jmlSpecifications	√
jmlClassSpecifications	√
jmlGhostDeclaration	√
jmlGhostInv	√
jmlMethodSpecification	√
jmlRequires	√
jmlReq	√
jmlOrReq	√
jmlLessThanEqualReq	√
jmlAssign	√
jmlEnsures	√
jmlEns	√
jmlOldEns	√
specifications	√
perm	√
requiresensuresClause	√
requiresClause	√
reaccesspermissionTypestates	√
accesspermission	√
typestate	√
ensuresclause	√
enaccesspermissiontypestates	√
attype	√
atappermission	√
usevalue	√
cases	√
other	√
classstates	√
startClassstates	√
state	√
invariant	√
condition	√
endclassstates	√
refine	√
states	√
dimension	√
value	√
item	√
getTokenNames	√
getGrammarFileName	√

Table 9: State Transition Matrix

	alive
alive	↑

Table 10: Methods Concurrency Mat

	PluralParser	jmlSpecifications	jmlClassSpecifications	jmlGhostDeclaration	jmlGhostInv	jmlMethodSpecification	jmlRequires	jmlReq	jmlOrReq	jmlLessThanEqualReq	jmlAssign	jmlEnsures	jmlEns	jmlOldEns	specifications	perm	requiresensuresClause	requiresClause	reaccesspermissionTypestates	accesspermission	typestate
PluralParser	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
jmlSpecifications	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
jmlClassSpecifications	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
jmlGhostDeclaration	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
jmlGhostInv	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
jmlMethodSpecification	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
jmlRequires	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
jmlReq	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
jmlOrReq	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
jmlLessThanEqualReq	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
jmlAssign	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
jmlEnsures	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
jmlEns	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
jmlOldEns	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
specifications	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
perm	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
requiresensuresClause	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
requiresClause	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
reaccesspermissionTypestates	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
accesspermission	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	
typestate	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
ensuresclause	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
enaccesspermissiontypestates	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
attype	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
atappermission	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	
usevalue	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
cases	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
other	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
classstates	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
startClassstates	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
state	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
invariant	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
condition	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
endclassstates	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
refine	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
states	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
dimension	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
value	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#
item	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#	#

getTokenNames	||	||	||	||	||	||	||	||	||	||	||	||	||	||	||	||	||	||	||	||	||
getGrammarFileName	|\	||	||	||	||	||	||	||	||	||	||	||	||	||	||	||	||	||	||	||

4 EJmlSpecification

Table 11: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
EJmlSpecification	√
setDimensionName	√
setDimensionValues	√
addRequires	√
setPerm	√
setEnsures	√
JmlClassSpec2PluralClassSpec	√
reset	√
JmlMethodSpec2PluralMethodSpec	√
moreRequires	√
getPerm	√
determineEnsures	√
oneRequires	√
noRequires	√

Table 12: State Transition Matrix

	alive
alive	↑

Table 13: Methods Concurrency Matrix

	EJmlSpecification	setDimensionName	setDimensionValues	addRequires	setPerm	setEnsures	JmlClassSpec2PluralClassSpec	reset	JmlMethodSpec2PluralMethodSpec	moreRequires	getPerm	determineEnsures	oneRequires	noRequires
EJmlSpecification	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setDimensionName	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setDimensionValues	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
addRequires	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setPerm	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setEnsures	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
JmlClassSpec2PluralClassSpec	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
reset	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
JmlMethodSpec2PluralMethodSpec	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
moreRequires	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦

getPerm	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦	∦	∦
determineEnsures	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
oneRequires	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
noRequires	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦

5 EGhost

Table 14: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
EGhost	√
setDimensionName	√
setDimensionValues	√
getDimensionName	√
getLowValueofInv	√
getHighValueofInv	√

Table 15: State Transition Matrix

	alive
alive	↑

Table 16: Methods Concurrency Matrix

	EGhost	setDimensionName	setDimensionValues	getDimensionName	getLowValueofInv	getHighValueofInv
EGhost	∦	∦	∦	∦	∦	∦
setDimensionName	∦	∦	∦	∦	∦	∦
setDimensionValues	∦	∦	∦	∦	∦	∦
getDimensionName	∦	∦	∦	‖	‖	‖
getLowValueofInv	∦	∦	∦	‖	‖	‖
getHighValueofInv	∦	∦	∦	‖	‖	‖

6 Time

Table 17: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
Time	√
toString	√

Table 18: State Transition Matrix

	alive
alive	↑

Table 19: Methods Concurrency Matrix

	Time	toString
Time	∦	∦
toString	∦	∦

7 FileReader

Table 20: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
FileReader	√
readFile	√

Table 21: State Transition Matrix

	alive
alive	↑

Table 22: Methods Concurrency Matrix

	FileReader	readFile
FileReader	∦	∦
readFile	∦	‖

8 UserSelectedClassesAnalysis

Table 23: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
UserSelectedClassesAnalysis	√
getCompilationUnit	√
analyzeFromCommandLine	√
callModelCheckerThroughCommandLine	√
printMetrics	√
printMethodMetrics	√
getTime	√
CreatePdfSummary_CommandLine	√
makePdfCommandLine	√
analyzeFromPlugin	√
getInputStream	√
callModelCheckerThroughPlugin	√
createPdfSummaryPlugin	√
makePdfPlugin	√

Table 24: State Transition Matrix

	alive
alive	↑

Table 25: Methods Concurrency Matrix

	UserSelectedClassesAnalysis	getCompilationUnit	analyzeFromCommandLine	callModelCheckerThroughCommandLine	printMetrics	printMethodMetrics	getTime	CreatePdfSummary_CommandLine	makePdfCommandLine	analyzeFromPlugin	getInputStream	callModelCheckerThroughPlugin	createPdfSummaryPlugin	makePdfPlugin
UserSelectedClassesAnalysis	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getCompilationUnit	∦	‖	‖	‖	‖	‖	‖	∦	‖	‖	‖	‖	‖	‖
analyzeFromCommandLine	∦	‖	∦	∦	∦	∦	∦	∦	‖	∦	‖	∦	∦	‖
callModelCheckerThroughCommandLine	∦	‖	∦	∦	∦	∦	∦	∦	‖	∦	‖	∦	∦	‖
printMetrics	∦	‖	∦	∦	∦	∦	∦	∦	‖	∦	‖	∦	∦	‖
printMethodMetrics	∦	‖	∦	∦	∦	∦	∦	∦	‖	∦	‖	∦	∦	‖
getTime	∦	‖	∦	∦	∦	∦	∦	∦	‖	∦	‖	∦	∦	‖
CreatePdfSummary_CommandLine	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦

makePdfCommandLine	∦	∥	∥	∥	∥	∥	∥	∦	∥	∥	∥	∥	∥	∥
analyzeFromPlugin	∦	∥	∦	∦	∦	∦	∦	∦	∥	∦	∥	∦	∦	∥
getInputStream	∦	∥	∥	∥	∥	∥	∥	∦	∥	∥	∥	∥	∥	∥
callModelCheckerThroughPlugin	∦	∥	∦	∦	∦	∦	∦	∦	∥	∦	∥	∦	∦	∥
createPdfSummaryPlugin	∦	∥	∦	∦	∦	∦	∦	∦	∥	∦	∥	∦	∦	∥
makePdfPlugin	∦	∥	∥	∥	∥	∥	∥	∦	∥	∥	∥	∥	∥	∥

9 EVMDDSMCGenerator

Table 26: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
EVMDDSMCGenerator	√
reset	√
modifyConstructorSpecifications	√
getPkgObject	√
addRequiresAPTS	√
addRequiresParamAPTS	√
addEnsuresAPTS	√
addEnsuresResultAPTS	√
addEnsuresParamAPTS	√
addCase	√
addState	√
addBoolStateInvariant	√
addStateInvariant	√
addDimension	√
addDimensionValue	√
addPkgObject	√

Table 27: State Transition Matrix

	alive
alive	↑

Table 28: Methods Concurrency Matrix

	EVMDDSMCGenerator	reset	modifyConstructorSpecifications	getPkgObject	addRequiresAPTS	addRequiresParamAPTS	addEnsuresAPTS	addEnsuresResultAPTS	addEnsuresParamAPTS	addCase	addState	addBoolStateInvariant	addStateInvariant	addDimension	addDimensionValue	addPkgObject
EVMDDSMCGenerator	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
reset	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖
modifyConstructorSpecifications	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖
getPkgObject	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖
addRequiresAPTS	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖
addRequiresParamAPTS	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖
addEnsuresAPTS	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖
addEnsuresResultAPTS	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖
addEnsuresParamAPTS	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖

addCase	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖
addState	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖
addBoolStateInvariant	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖
addStateInvariant	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖
addDimension	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖
addDimensionValue	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖
addPkgObject	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖

10 EPackage

Table 29: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
EPackage	√
getClasses	√
getTotalStates	√
getTotalReachableStates	√
getSinkStates	√
setSinkStates	√
setName	√
getName	√

Table 30: State Transition Matrix

	alive
alive	↑

Table 31: Methods Concurrency Matrix

	EPackage	getClasses	getTotalStates	getTotalReachableStates	getSinkStates	setSinkStates	setName	getName
EPackage	∦	∦	∦	∦	∦	∦	∦	∦
getClasses	∦	‖	∦	‖	‖	∦	∦	∦
getTotalStates	∦	∦	∦	∦	∦	∦	∦	∦
getTotalReachableStates	∦	‖	∦	‖	‖	∦	∦	∦
getSinkStates	∦	‖	∦	‖	‖	∦	∦	∦
setSinkStates	∦	∦	∦	∦	∦	∦	∦	∦
setName	∦	∦	∦	∦	∦	∦	∦	∦
getName	∦	∦	∦	∦	∦	∦	∦	∦

11 EClass

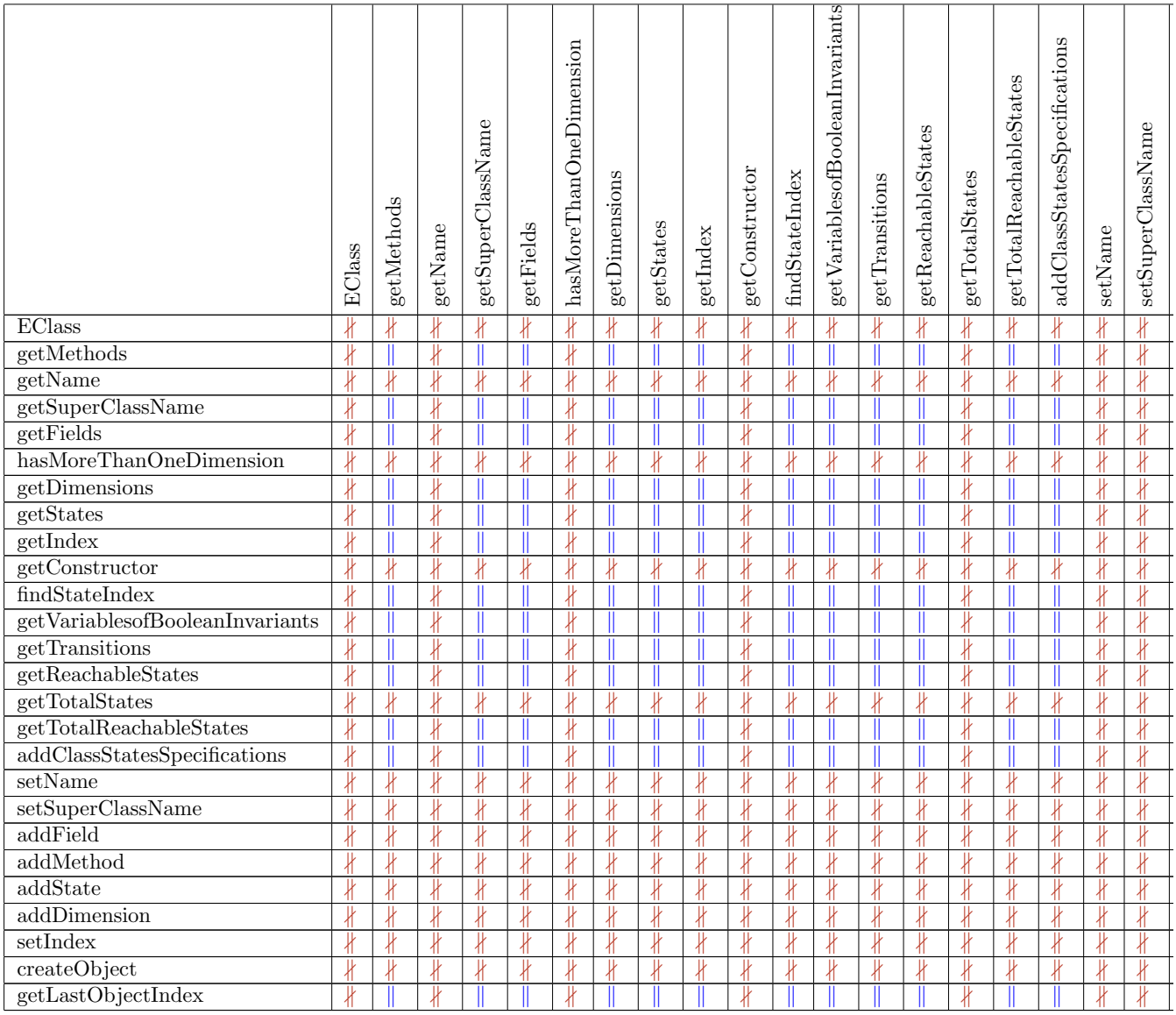
Table 32: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
EClass	√
getMethods	√
getName	√
getSuperClassName	√
getFields	√
hasMoreThanOneDimension	√
getDimensions	√
getStates	√
getIndex	√
getConstructor	√
findStateIndex	√
getVariablesofBooleanInvariants	√
getTransitions	√
getReachableStates	√
getTotalStates	√
getTotalReachableStates	√
addClassStatesSpecifications	√
setName	√
setSuperClassName	√
addField	√
addMethod	√
addState	√
addDimension	√
setIndex	√
createObject	√
getLastObjectIndex	√

Table 33: State Transition Matrix

	alive
alive	↑

Table 34: Methods Concurrency Matrix


EClass
getMethods
getName
getSuperClassName
getFields
hasMoreThanOneDimension
getDimensions
getStates
getIndex
getConstructor
findStateIndex
getVariablesofBooleanInvariants
getTransitions
getReachableStates
getTotalStates
getTotalReachableStates
addClassStatesSpecifications
setName
setSuperClassName
EClass
getMethods
getName
getSuperClassName
getFields
hasMoreThanOneDimension
getDimensions
getStates
getIndex
getConstructor
findStateIndex
getVariablesofBooleanInvariants
getTransitions
getReachableStates
getTotalStates
getTotalReachableStates
addClassStatesSpecifications
setName
setSuperClassName
addField
addMethod
addState
addDimension
setIndex
createObject
getLastObjectIndex

12 EMethod

Table 35: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
EMethod	√
getName	√
getRequiresAPTS	√
getEnsuresAPTS	√
getIdentifier	√
getParameters	√
getIndex	√
getRequiresClauseSatisfiability	√
isConcurrentMethod	√
setRequiresClauseSatisfiability	√
setConcurrentMethod	√
addSpecifications	√
setName	√
setReturnType	√
setIdentifier	√
addParameter	√
getReturnType	√
setCaseNumber	√
getCaseNumber	√
setIndex	√
setJMLPermission	√
getJMLPermission	√

Table 36: State Transition Matrix

	alive
alive	↑

Table 37: Methods Concurrency Matrix

	EMethod	getName	getRequiresAPTS	getEnsuresAPTS	getIdentifier	getParameters	getIndex	getRequiresClauseSatisfiability	isConcurrentMethod	setRequiresClauseSatisfiability	setConcurrentMethod	addSpecifications	setName	setReturnType	setIdentifier	addParameter	getReturnType	setCaseNumber	getCaseNumber	setIndex
EMethod	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getName	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getRequiresAPTS	∦	∦	‖	‖	∦	‖	‖	‖	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
getEnsuresAPTS	∦	∦	‖	‖	∦	‖	‖	‖	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦

getIdentifier	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getParameters	∦	∦	‖	‖	∦	‖	‖	‖	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
getIndex	∦	∦	‖	‖	∦	‖	‖	‖	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
getRequiresClauseSatisfiability	∦	∦	‖	‖	∦	‖	‖	‖	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
isConcurrentMethod	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setRequiresClauseSatisfiability	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setConcurrentMethod	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
addSpecifications	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setName	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setReturnType	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setIdentifier	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
addParameter	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getReturnType	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setCaseNumber	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getCaseNumber	∦	∦	‖	‖	∦	‖	‖	‖	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
setIndex	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setJMLPermission	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getJMLPermission	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦

13 ESpecification

Table 38: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
ESpecification	√
setAP	√
getParentClass	√
getFieldName	√
getTS	√
getAP	√
setAPTS	√
clone	√

Table 39: State Transition Matrix

	alive
alive	↑

Table 40: Methods Concurrency Matrix

	ESpecification	setAP	getParentClass	getFieldName	getTS	getAP	setAPTS	clone
ESpecification	∦	∦	∦	∦	∦	∦	∦	∦
setAP	∦	∦	∦	∦	∦	∦	∦	‖
getParentClass	∦	∦	‖	‖	∦	∦	∦	‖
getFieldName	∦	∦	‖	‖	∦	∦	∦	‖
getTS	∦	∦	∦	∦	∦	∦	∦	‖
getAP	∦	∦	∦	∦	∦	∦	∦	‖
setAPTS	∦	∦	∦	∦	∦	∦	∦	‖
clone	∦	‖	‖	‖	‖	‖	‖	‖

14 EGeneratedPluralSpecification

Table 41: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
EGeneratedPluralSpecification	√
createFromCommandLine	√
createFromPlugin	√

Table 42: State Transition Matrix

	alive
alive	↑

Table 43: Methods Concurrency Matrix

	EGeneratedPluralSpecification	createFromCommandLine	createFromPlugin
EGeneratedPluralSpecification	∦	∦	∦
createFromCommandLine	∦	∦	∦
createFromPlugin	∦	∦	∦

15 ESMCModel

Table 44: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
ESMCModel	√
setK	√
generateSMCmodelCommandLine	√
Transitions	√
comment	√
declarationsAndinitilizations	√
initialize	√
modelAlias	√
isClassExist	√
createInstanceInModel	√
modelPrimePCandMethod	√
startMethod	√
modelPCConstructor	√
modelAPs	√
getClass	√
getObjectIndex	√
modelPCMethod	√
getFieldClass	√
getDimensionIndex	√
startAPTS	√
startAPTSPARAM	√
error	√
startPrimeTSPARAM	√
starPrimeAP	√
modelPrimeConstructor	√
modelInheritance	√
modelPrimeAPStateInvariants	√
getClassIndex	√
modelPrimeAP	√
getAPId	√
modelEndPCMethod	√
endMethod	√
modelEndPCConstructor	√
modelPrimePCConstructor	√
modelPrimePC	√
endPrimeAPTS	√
modelendConstructor	√
updateBoolStateInvariants	√
updateStateInvariants	√
updateState	√
modelState	√
modelStateInvariants	√
modelBoolStateInvariants	√
methodsReachability	√
modelAP	√
updateTokens	√
endPrimeAPTSPARAM	√

initilizeVariables	√
initilizeKVariables	√
defineVariables	√
defineKVariables	√
isPrivateAndIndexEqualToZero	√
generateSMCmodelPlugin	√
createAlias	√
addIndexes	√
createDimensionsObject	√
createDimensionAsField	√
createParentObject	√
createParentAsField	√
addInvariantStateIndex	√
setInvariantVariableType	√
Spec	√
statesAdjancyMatrix	√
concurrentMethods	√
sinkStates	√

Table 45: State Transition Matrix

	alive
alive	↑

	ESMCModel	setK	generateSMCmodelCommandLine	Transitions	comment	declarationsAndinitilizations	initialize	modelAlias	isClassExist	createInstanceInModel	modelPrimePCandMethod	startMethod	modelPCConstructor	modelAPs	getClass	getObjectIndex	modelPCMethod	getFieldClass	getDimensionIndex	startAPTS
ESMCModel	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setK	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
generateSMCmodelCommandLine	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
Transitions	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
comment	∦	∦	∦	∦	‖	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
declarationsAndinitilizations	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
initialize	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
modelAlias	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
isClassExist	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
createInstanceInModel	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
modelPrimePCandMethod	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦

startMethod	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
modelPCConstructor	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
modelAPs	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getClass	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getObjectIndex	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
modelPCMethod	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getFieldClass	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getDimensionIndex	∦	∦	∦	∦	‖	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
startAPTS	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
startAPTSPARAM	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
error	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
startPrimeTSPARAM	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
starPrimeAP	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
modelPrimeConstructor	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
modelInheritance	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
modelPrimeAPStateInvariants	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getClassIndex	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
modelPrimeAP	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getAPId	∦	∦	∦	∦	‖	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
modelEndPCMethod	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
endMethod	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
modelEndPCConstructor	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
modelPrimePCConstructor	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
modelPrimePC	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
endPrimeAPTS	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
modelendConstructor	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
updateBoolStateInvariants	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
updateStateInvariants	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
updateState	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
modelState	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
modelStateInvariants	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
modelBoolStateInvariants	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
methodsReachability	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
modelAP	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
updateTokens	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
endPrimeAPTSPARAM	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
initilizeVariables	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
initilizeKVariables	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
defineVariables	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
defineKVariables	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
isPrivateAndIndexEqualToZero	∦	∦	∦	∦	‖	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖	∦
generateSMCmodelPlugin	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
createAlias	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
addIndexes	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
createDimensionsObject	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
createDimensionAsField	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
createParentObject	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
createParentAsField	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
addInvariantStateIndex	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setInvariantVariableType	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
Spec	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦

statesAdjancyMatrix	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫
concurrentMethods	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫
sinkStates	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫	⫫

16 EField

Table 47: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
EField	√
getName	√
getObjectIndex	√
getType	√
getClassIndex	√
getModifier	√
setName	√
setType	√
setModifier	√
setClassIndex	√
setObjectIndex	√

Table 48: State Transition Matrix

	alive
alive	↑

Table 49: Methods Concurrency Matrix

	EField	getName	getObjectIndex	getType	getClassIndex	getModifier	setName	setType	setModifier	setClassIndex	setObjectIndex
EField	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getName	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getObjectIndex	∦	∦	‖	∦	‖	‖	∦	∦	∦	∦	∦
getType	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getClassIndex	∦	∦	‖	∦	‖	‖	∦	∦	∦	∦	∦
getModifier	∦	∦	‖	∦	‖	‖	∦	∦	∦	∦	∦
setName	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setType	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setModifier	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setClassIndex	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setObjectIndex	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦

17 EDim

Table 50: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
EDim	$\surd$
getValues	$\surd$
addValue	$\surd$
setName	$\surd$
getName	$\surd$

Table 51: State Transition Matrix

	alive
alive	$\uparrow$

Table 52: Methods Concurrency Matrix

	EDim	getValues	addValue	setName	getName
EDim	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
getValues	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\parallel$
addValue	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
setName	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
getName	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\parallel$

18 EParameter

Table 53: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
EParameter	√
getRequiresAPTS	√
getType	√
getEnsuresAPTS	√
getName	√
getNumber	√
setNumber	√
setName	√
setType	√

Table 54: State Transition Matrix

	alive
alive	↑

Table 55: Methods Concurrency Matrix

	EParameter	getRequiresAPTS	getType	getEnsuresAPTS	getName	getNumber	setNumber	setName	setType
EParameter	∦	∦	∦	∦	∦	∦	∦	∦	∦
getRequiresAPTS	∦	‖	‖	‖	∦	‖	∦	∦	∦
getType	∦	‖	‖	‖	∦	‖	∦	∦	∦
getEnsuresAPTS	∦	‖	‖	‖	∦	‖	∦	∦	∦
getName	∦	∦	∦	∦	∦	∦	∦	∦	∦
getNumber	∦	‖	‖	‖	∦	‖	∦	∦	∦
setNumber	∦	∦	∦	∦	∦	∦	∦	∦	∦
setName	∦	∦	∦	∦	∦	∦	∦	∦	∦
setType	∦	∦	∦	∦	∦	∦	∦	∦	∦

19 EState

Table 56: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
EState	√
getName	√
getInvariants	√
getBoolInvariants	√
getStateIndex	√
isReachable	√
setReachability	√
isReachableState	√
addBoolInvariant	√
addInvariant	√
setIndex	√

Table 57: State Transition Matrix

	alive
alive	↑

Table 58: Methods Concurrency Matrix

	EState	getName	getInvariants	getBoolInvariants	getStateIndex	isReachable	setReachability	isReachableState	addBoolInvariant	addInvariant	setIndex
EState	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getName	∦	‖	‖	‖	‖	‖	∦	∦	∦	∦	∦
getInvariants	∦	‖	‖	‖	‖	‖	∦	∦	∦	∦	∦
getBoolInvariants	∦	‖	‖	‖	‖	‖	∦	∦	∦	∦	∦
getStateIndex	∦	‖	‖	‖	‖	‖	∦	∦	∦	∦	∦
isReachable	∦	‖	‖	‖	‖	‖	∦	∦	∦	∦	∦
setReachability	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
isReachableState	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
addBoolInvariant	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
addInvariant	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setIndex	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦

20 EInvariant

Table 59: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
EInvariant	√
getAP	√
getVariableType	√
getVariable	√
getStateName	√
getStateInvariants	√
setVariableType	√
setStateIndex	√
setAP	√
setVariable	√
setState	√

Table 60: State Transition Matrix

	alive
alive	↑

Table 61: Methods Concurrency Matrix

	EInvariant	getAP	getVariableType	getVariable	getStateName	getStateInvariants	setVariableType	setStateIndex	setAP	setVariable	setState
EInvariant	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getAP	∦	‖	‖	‖	‖	∦	∦	‖	∦	∦	∦
getVariableType	∦	‖	‖	‖	‖	∦	∦	‖	∦	∦	∦
getVariable	∦	‖	‖	‖	‖	∦	∦	‖	∦	∦	∦
getStateName	∦	‖	‖	‖	‖	∦	∦	‖	∦	∦	∦
getStateInvariants	∦	∦	∦	∦	∦	∦	∦	‖	∦	∦	∦
setVariableType	∦	∦	∦	∦	∦	∦	∦	‖	∦	∦	∦
setStateIndex	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
setAP	∦	∦	∦	∦	∦	∦	∦	‖	∦	∦	∦
setVariable	∦	∦	∦	∦	∦	∦	∦	‖	∦	∦	∦
setState	∦	∦	∦	∦	∦	∦	∦	‖	∦	∦	∦

21 EBoolInvariant

Table 62: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
EBoolInvariant	√
getVariable	√
getValue	√

Table 63: State Transition Matrix

	alive
alive	↑

Table 64: Methods Concurrency Matrix

	EBoolInvariant	getVariable	getValue
EBoolInvariant	∦	∦	∦
getVariable	∦	‖	‖
getValue	∦	‖	‖

22 EGrarphWriter

Table 65: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
EGrarphWriter	√
addTrnsitions	√
parseMethodReachability	√
createGraph	√
getNumberofUnReachableMethods	√
setNumberofUnReachableMethods	√

Table 66: State Transition Matrix

	alive
alive	↑

Table 67: Methods Concurrency Matrix

	EGrarphWriter	addTrnsitions	parseMethodReachability	createGraph	getNumberofUnReachableMethods	setNumberofUnReachableMethods
EGrarphWriter	∦	∦	∦	∦	∦	∦
addTrnsitions	∦	∦	∦	∦	∦	∦
parseMethodReachability	∦	∦	∦	∦	∦	∦
createGraph	∦	∦	∦	∦	∦	∦
getNumberofUnReachableMethods	∦	∦	∦	∦	‖	∦
setNumberofUnReachableMethods	∦	∦	∦	∦	∦	∦

23 EOutputLatex

Table 68: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
EOutputLatex	√
create_CommandLine	√
addUsePackages	√
writeToLatex	√
WriteSummary	√
addSummaryTableColumns	√
addSummaryTableHeaders	√
addSummaryTableRows	√
writeRequiresClauseSatisfiabilty	√
writeStateTransitionMatrix	√
addSTMNumberofColumns	√
addSTMColumnsHeaders	√
addSTMRows	√
getStateReachabilityValue	√
writeMethodConcurrencyMatrix	√
addConcurrencyMatrixColumns	√
addConcurrencyMatrixHeaders	√
addConcurrencyMatrixRows	√
getConcurrencyValue	√
writeAbbervations	√
reset	√
setText	√
parseRequires	√
getMethod	√
parseTransitions	√
parseConcurrentMethods	√
parseSinkStates	√
create_Plugin	√

Table 69: State Transition Matrix

	alive
alive	↑

Table 70: Methods Concurrency Matrix

	EOutputLatex	create_CommandLine	addUsePackages	writeToLatex	WriteSummary	addSummaryTableColumns	addSummaryTableHeaders	addSummaryTableRows	writeRequiresClauseSatisfiabilty	writeStateTransitionMatrix	addSTMNumberofColumns	addSTMColumnsHeaders	addSTMRows	getStateReachabilityValue	writeMethodConcurrencyMatrix	addConcurrencyMatrixColumns	addConcurrencyMatrixHeaders	addConcurrencyMatrixRows	getConcurrencyValue
EOutputLatex	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
create_CommandLine	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
addUsePackages	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
writeToLatex	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
WriteSummary	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
addSummaryTableColumns	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
addSummaryTableHeaders	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
addSummaryTableRows	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
writeRequiresClauseSatisfiabilty	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
writeStateTransitionMatrix	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
addSTMNumberofColumns	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
addSTMColumnsHeaders	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
addSTMRows	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
getStateReachabilityValue	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
writeMethodConcurrencyMatrix	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
addConcurrencyMatrixColumns	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
addConcurrencyMatrixHeaders	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
addConcurrencyMatrixRows	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
getConcurrencyValue	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
writeAbbervations	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
reset	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
setText	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
parseRequires	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
getMethod	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
parseTransitions	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
parseConcurrentMethods	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
parseSinkStates	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠
create_Plugin	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠	≠

24 WorkspaceUtilities

Table 71: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
WorkspaceUtilities	√
getASTNodeFromCompilationUnit	√
scanForCompilationUnits	√
collectCompilationUnits	√
findCompilationUnits	√
getWorkspaceRelativeName	√
parseCompilationUnits	√
scanForMethodDeclarations	√
scanForMethodDeclarationsFromAST	√

Table 72: State Transition Matrix

	alive
alive	↑

Table 73: Methods Concurrency Matrix

	WorkspaceUtilities	getASTNodeFromCompilationUnit	scanForCompilationUnits	collectCompilationUnits	findCompilationUnits	getWorkspaceRelativeName	parseCompilationUnits	scanForMethodDeclarations	scanForMethodDeclarationsFromAST
WorkspaceUtilities	∦	∦	∦	∦	∦	∦	∦	∦	∦
getASTNodeFromCompilationUnit	∦	‖	‖	‖	‖	‖	‖	‖	‖
scanForCompilationUnits	∦	‖	∦	∦	∦	‖	‖	‖	‖
collectCompilationUnits	∦	‖	∦	∦	∦	‖	‖	‖	‖
findCompilationUnits	∦	‖	∦	∦	∦	‖	‖	‖	‖
getWorkspaceRelativeName	∦	‖	‖	‖	‖	‖	‖	‖	‖
parseCompilationUnits	∦	‖	‖	‖	‖	‖	‖	‖	‖
scanForMethodDeclarations	∦	‖	‖	‖	‖	‖	‖	‖	‖
scanForMethodDeclarationsFromAST	∦	‖	‖	‖	‖	‖	‖	‖	‖

25 SMCVisitor

Table 74: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
SMCVisitor	√
addUnparsedSpecifications	√
preVisit	√
postVisit	√
visit	√
endVisit	√
callParser	√

Table 75: State Transition Matrix

	alive
alive	↑

Table 76: Methods Concurrency Matrix

	SMCVisitor	addUnparsedSpecifications	preVisit	postVisit	visit	endVisit	callParser
SMCVisitor	∦	∦	∦	∦	∦	∦	∦
addUnparsedSpecifications	∦	∦	‖	‖	∦	‖	∦
preVisit	∦	‖	‖	‖	‖	‖	‖
postVisit	∦	‖	‖	‖	‖	‖	‖
visit	∦	∦	‖	‖	∦	‖	∦
endVisit	∦	‖	‖	‖	‖	‖	‖
callParser	∦	∦	‖	‖	∦	‖	∦

26 PulseSettings

Table 77: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
PulseSettings	√
getInheritance	√
getFullModel	√
getInvariants	√
getDimensions	√
setInvariants	√
setAliasPerObject	√
setFullModel	√
setDimensions	√
setInheritance	√
getAliasPerObject	√

Table 78: State Transition Matrix

	alive
alive	↑

Table 79: Methods Concurrency Matrix

	PulseSettings	getInheritance	getFullModel	getInvariants	getDimensions	setInvariants	setAliasPerObject	setFullModel	setDimensions	setInheritance	getAliasPerObject
PulseSettings	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getInheritance	∦	‖	‖	‖	‖	∦	∦	∦	∦	∦	‖
getFullModel	∦	‖	‖	‖	‖	∦	∦	∦	∦	∦	‖
getInvariants	∦	‖	‖	‖	‖	∦	∦	∦	∦	∦	‖
getDimensions	∦	‖	‖	‖	‖	∦	∦	∦	∦	∦	‖
setInvariants	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setAliasPerObject	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setFullModel	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setDimensions	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
setInheritance	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getAliasPerObject	∦	‖	‖	‖	‖	∦	∦	∦	∦	∦	‖

27 specificationStruct

Table 80: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
specificationStruct	√

Table 81: State Transition Matrix

	alive
alive	↑

28 Clause

Table 82: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
Clause	√

Table 83: State Transition Matrix

	alive
alive	↑

29 Signature

Table 84: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
Signature	√

Table 85: State Transition Matrix

	alive
alive	↑

30 MethodFindVisitor

Table 86: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
MethodFindVisitor	√
visit	√

Table 87: State Transition Matrix

	alive
alive	↑

Table 88: Methods Concurrency Matrix

	MethodFindVisitor	visit
MethodFindVisitor	∦	∦
visit	∦	∦

31 Activator

Table 89: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
Activator	√
start	√
stop	√
getDefault	√
getImageDescriptor	√

Table 90: State Transition Matrix

	alive
alive	↑

Table 91: Methods Concurrency Matrix

	Activator	start	stop	getDefault	getImageDescriptor
Activator	∦	∦	∦	∦	∦
start	∦	∦	∦	∦	∦
stop	∦	∦	∦	∦	∦
getDefault	∦	∦	∦	‖	∦
getImageDescriptor	∦	∦	∦	∦	∦

32 GAPHandler

Table 92: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
GAPHandler	√
addHandlerListener	√
dispose	√
execute	√
extractSettings	√
isEnabled	√
isHandled	√
removeHandlerListener	√

Table 93: State Transition Matrix

	alive
alive	↑

Table 94: Methods Concurrency Matrix

	GAPHandler	addHandlerListener	dispose	execute	extractSettings	isEnabled	isHandled	removeHandlerListener
GAPHandler	∦	∦	∦	∦	∦	∦	∦	∦
addHandlerListener	∦	‖	‖	‖	‖	‖	‖	‖
dispose	∦	‖	‖	‖	‖	‖	‖	‖
execute	∦	‖	‖	∦	∦	‖	‖	‖
extractSettings	∦	‖	‖	∦	∦	‖	‖	‖
isEnabled	∦	‖	‖	‖	‖	‖	‖	‖
isHandled	∦	‖	‖	‖	‖	‖	‖	‖
removeHandlerListener	∦	‖	‖	‖	‖	‖	‖	‖

33 GAPIFileAction

Table 95: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
GAPIFileAction	√
selectionChanged	√
setActivePart	√
run	√

Table 96: State Transition Matrix

	alive
alive	↑

Table 97: Methods Concurrency Matrix

	GAPIFileAction	selectionChanged	setActivePart	run
GAPIFileAction	∦	∦	∦	∦
selectionChanged	∦	∦	∥	∦
setActivePart	∦	∥	∥	∥
run	∦	∦	∥	∦

34 Anonymous

Table 98: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
Anonymous	√
run	√

Table 99: State Transition Matrix

	alive
alive	↑

Table 100: Methods Concurrency Matrix

	Anonymous	run
Anonymous	∦	∦
run	∦	∦

35 Main

Table 101: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
Main	√
main	√
testRead	√
seprateJavaFile	√
anTest	√

Table 102: State Transition Matrix

	alive
alive	↑

Table 103: Methods Concurrency Matrix

	Main	main	testRead	seprateJavaFile	anTest
Main	∦	∦	∦	∦	∦
main	∦	∦	‖	∦	‖
testRead	∦	‖	‖	‖	‖
seprateJavaFile	∦	∦	‖	∦	‖
anTest	∦	‖	‖	‖	‖

36 TypestateReturn

Table 104: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
TypestateReturn	√

Table 105: State Transition Matrix

	alive
alive	↑

37 AtApPermissionReturn

Table 106: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
AtApPermissionReturn	√

Table 107: State Transition Matrix

	alive
alive	↑

38 AccesspermissionReturn

Table 108: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
AccesspermissionReturn	√

Table 109: State Transition Matrix

	alive
alive	↑

39 PluralLexer

Table 110: Method's Satisfiability(Code Reachabiity Analysis

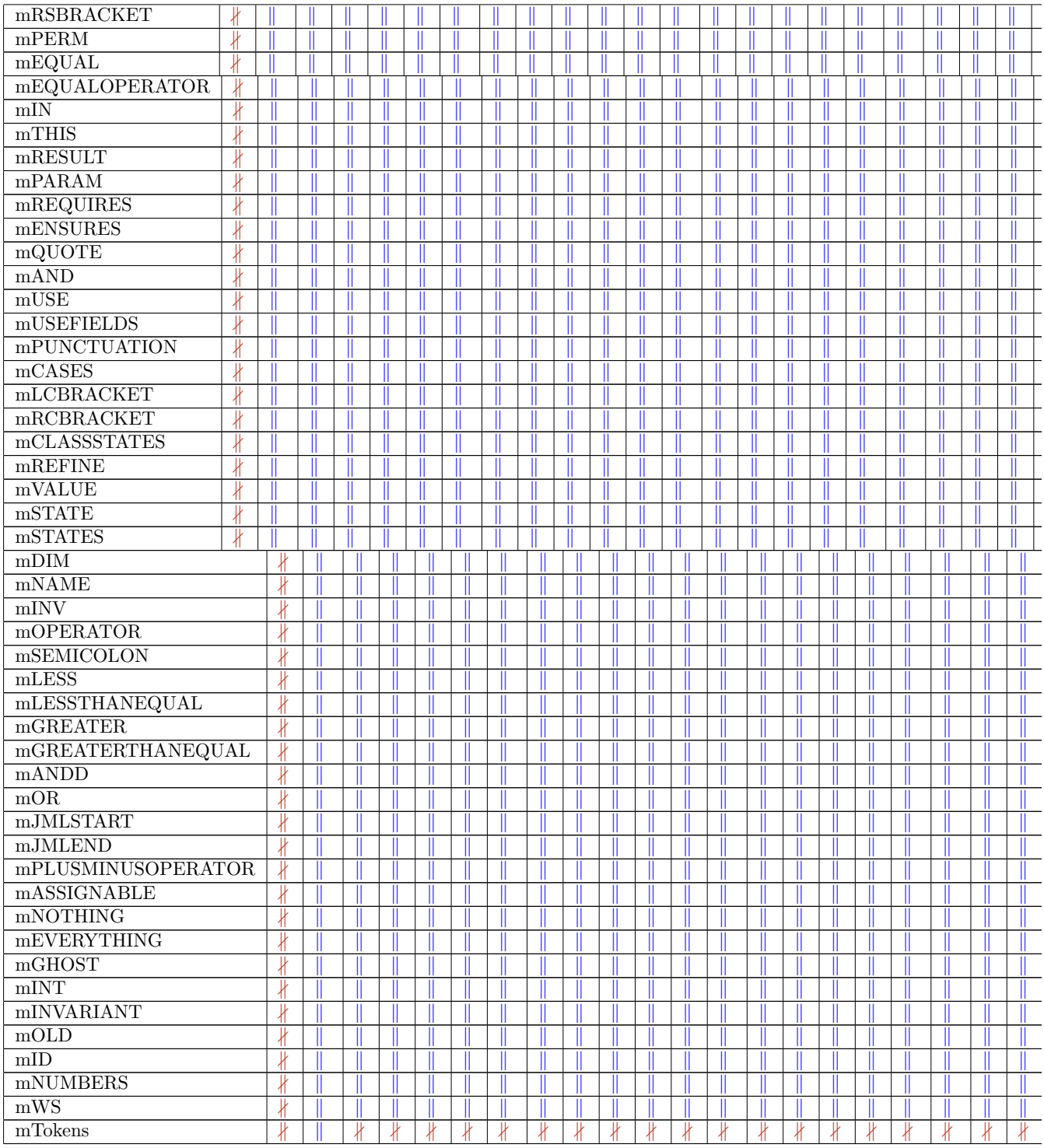
Method	Satisfiability
PluralLexer	√
getGrammarFileName	√
mATFULL	√
mATPURE	√
mATIMMUTABLE	√
mATSHARE	√
mATUNIQUE	√
mPUBLICBEHAVIOR	√
mFULL	√
mPURE	√
mIMMUTABLE	√
mSHARE	√
mUNIQUE	√
mNONE	√
mLSBRACKET	√
mRSBRACKET	√
mPERM	√
mEQUAL	√
mEQUALOPERATOR	√
mIN	√
mTHIS	√
mRESULT	√
mPARAM	√
mREQUIRES	√
mENSURES	√
mQUOTE	√
mAND	√
mUSE	√
mUSEFIELDS	√
mPUNCTUATION	√
mCASES	√
mLCBRACKET	√
mRCBRACKET	√
mCLASSSTATES	√
mREFINE	√
mVALUE	√
mSTATE	√
mSTATES	√
mDIM	√
mNAME	√
mINV	√
mOPERATOR	√
mSEMICOLON	√
mLESS	√
mLESSTHANEQUAL	√
mGREATER	√
mGREATERTHANEQUAL	√

mANDD	√
mOR	√
mJMLSTART	√
mJMLEND	√
mPLUSMINUSOPERATOR	√
mASSIGNABLE	√
mNOTHING	√
mEVERYTHING	√
mGHOST	√
mINT	√
mINVARIANT	√
mOLD	√
mID	√
mNUMBERS	√
mWS	√
mTokens	√

Table 111: State Transition Matrix

	alive
alive	↑

	PluralLexer	getGrammarFileName	mATFULL	mATPURE	mATIMMUTABLE	mATSHARE	mATUNIQUE	mPUBLICBEHAVIOR	mFULL	mPURE	mIMMUTABLE	mSHARE	mUNIQUE	mNONE	mLSBRACKET	mRSBRACKET	mPERM	mEQUAL	mEQUALOPERATOR	mIN	mTHIS	mRESULT
PluralLexer	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
getGrammarFileName	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
mATFULL	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
mATPURE	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
mATIMMUTABLE	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
mATSHARE	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
mATUNIQUE	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
mPUBLICBEHAVIOR	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
mFULL	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
mPURE	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
mIMMUTABLE	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
mSHARE	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
mUNIQUE	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
mNONE	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
mLSBRACKET	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖


mRSBRACKET
mPERM
mEQUAL
mEQUALOPERATOR
mIN
mTHIS
mRESULT
mPARAM
mREQUIRES
mENSURES
mQUOTE
mAND
mUSE
mUSEFIELDS
mPUNCTUATION
mCASES
mLCBRACKET
mRCBRACKET
mCLASSSTATES
mREFINE
mVALUE
mSTATE
mSTATES
mDIM
mNAME
mINV
mOPERATOR
mSEMICOLON
mLESS
mLESSTHANEQUAL
mGREATER
mGREATERTHANEQUAL
mANDD
mOR
mJMLSTART
mJMLEND
mPLUSMINUSOPERATOR
mASSIGNABLE
mNOTHING
mEVERYTHING
mGHOST
mINT
mINVARIANT
mOLD
mID
mNUMBERS
mWS
mTokens

40 DFA7

Table 113: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
DFA7	√
getDescription	√

Table 114: State Transition Matrix

	alive
alive	↑

Table 115: Methods Concurrency Matrix

	DFA7	getDescription
DFA7	∦	∦
getDescription	∦	‖

41 EAPTypeState

Table 116: Method's Satisfiability(Code Reachabiity Analysis

Method	Satisfiability
EAPTypeState	√
setAP	√
getAP	√
setTS	√
getTS	√

Table 117: State Transition Matrix

	alive
alive	↑

Table 118: Methods Concurrency Matrix

	EAPTypeState	setAP	getAP	setTS	getTS
EAPTypeState	∦	∦	∦	∦	∦
setAP	∦	∦	∦	∦	∦
getAP	∦	∦	∦	∦	∦
setTS	∦	∦	∦	∦	∦
getTS	∦	∦	∦	∦	∦

42 Abbreviation

Table 119: 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

43 Annotated version of the input program generated by Sip4J

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

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

9  @Perm(requires="share(this) in alive",
10 ensures="share(this) in alive")
11  public void run(IAction action) {

13 }

15  public void selectionChanged(IAction action, ISelection selection) {

17 }

19  public void dispose() {

21 }
22 @Perm(requires="share(this) in alive",
23 ensures="share(this) in alive")
24  public void init(IWorkbenchWindow window) {

26 }

28 }ENDOFCLASS

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

32 class JMLAnnotatedJavaClass {
33 @Perm(ensures="unique(this) in alive")
34 JMLAnnotatedJavaClass() {   }

36 @Perm(requires="unique(this) in alive",
37 ensures="unique(this) in alive")
38  public String translateJMLAnnotationsToPlural(String JProgram) {
39  return null;

41 }
42 @Perm(requires="unique(this) in alive",
43 ensures="unique(this) in alive")
44  private String translateClassSpecifications(String JProgram) {
45  return null;

47 }
48 @Perm(requires="unique(this) in alive",
49 ensures="unique(this) in alive")
50  private void parseAndStoreJMLAnnotation(String JMLAnnotation) {

52 }
53 @Perm(requires="unique(this) in alive",
54 ensures="unique(this) in alive")
55  private String translateMethodSpecification(String JProgram) {
56  return null;

58 }

60  public String getInputStream(ICompilationUnit unit) {
61  return null;

63 }

65  public String readFileAsString(String filePath) {
66  return null;

68 }

70 }ENDOFCLASS

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

74 class PluralParser {
75 @Perm(ensures="unique(this) in alive")
76 PluralParser() {   }

78 @Perm(requires="unique(this) in alive",
```

```
79  ensures="unique(this) in alive")
80    void jmlSpecifications() {

82  }
83  @Perm(requires="unique(this) in alive",
84  ensures="unique(this) in alive")
85    void jmlClassSpecifications() {

87  }
88  @Perm(requires="unique(this) in alive",
89  ensures="unique(this) in alive")
90    void jmlGhostDeclaration() {

92  }
93  @Perm(requires="unique(this) in alive",
94  ensures="unique(this) in alive")
95    void jmlGhostInv() {

97  }
98  @Perm(requires="unique(this) in alive",
99  ensures="unique(this) in alive")
100   void jmlMethodSpecification() {

102 }
103 @Perm(requires="unique(this) in alive",
104 ensures="unique(this) in alive")
105   void jmlRequires() {

107 }
108 @Perm(requires="unique(this) in alive",
109 ensures="unique(this) in alive")
110   void jmlReq() {

112 }
113 @Perm(requires="unique(this) in alive",
114 ensures="unique(this) in alive")
115   void jmlOrReq() {

117 }
118 @Perm(requires="unique(this) in alive",
119 ensures="unique(this) in alive")
120   void jmlLessThanEqualReq() {

122 }
123 @Perm(requires="unique(this) in alive",
124 ensures="unique(this) in alive")
125   void jmlAssign() {

127 }
128 @Perm(requires="unique(this) in alive",
129 ensures="unique(this) in alive")
130   void jmlEnsures() {

132 }
133 @Perm(requires="unique(this) in alive",
134 ensures="unique(this) in alive")
135   void jmlEns() {

137 }
138 @Perm(requires="unique(this) in alive",
139 ensures="unique(this) in alive")
140   void jmlOldEns() {

142 }
143 @Perm(requires="unique(this) in alive",
144 ensures="unique(this) in alive")
145   void specifications() {

147 }
148 @Perm(requires="unique(this) in alive",
149 ensures="unique(this) in alive")
150   void perm() {

152 }
153 @Perm(requires="unique(this) in alive",
154 ensures="unique(this) in alive")
155   void requiresensuresClause() {

157 }
158 @Perm(requires="unique(this) in alive",
159 ensures="unique(this) in alive")
```

```
160   void requiresClause() {

162 }
163 @Perm(requires="unique(this) in alive",
164 ensures="unique(this) in alive")
165   void reaccesspermissionTypestates() {

167 }
168 @Perm(requires="immutable(this) in alive",
169 ensures="immutable(this) in alive")
170   AccesspermissionReturn accesspermission() {
171  return null;

173 }
174 @Perm(requires="unique(this) in alive",
175 ensures="unique(this) in alive")
176   TypestateReturn typestate() {
177  return null;

179 }
180 @Perm(requires="unique(this) in alive",
181 ensures="unique(this) in alive")
182   void ensuresclause() {

184 }
185 @Perm(requires="unique(this) in alive",
186 ensures="unique(this) in alive")
187   void enaccesspermissiontypestates() {

189 }
190 @Perm(requires="unique(this) in alive",
191 ensures="unique(this) in alive")
192   void attype() {

194 }
195 @Perm(requires="immutable(this) in alive",
196 ensures="immutable(this) in alive")
197   AtApPermissionReturn atappermission() {
198  return null;

200 }
201 @Perm(requires="unique(this) in alive",
202 ensures="unique(this) in alive")
203   void usevalue() {

205 }
206 @Perm(requires="unique(this) in alive",
207 ensures="unique(this) in alive")
208   void cases() {

210 }
211 @Perm(requires="unique(this) in alive",
212 ensures="unique(this) in alive")
213   void other() {

215 }
216 @Perm(requires="unique(this) in alive",
217 ensures="unique(this) in alive")
218   void classstates() {

220 }
221 @Perm(requires="unique(this) in alive",
222 ensures="unique(this) in alive")
223   void startClassstates() {

225 }
226 @Perm(requires="unique(this) in alive",
227 ensures="unique(this) in alive")
228   void state() {

230 }
231 @Perm(requires="unique(this) in alive",
232 ensures="unique(this) in alive")
233   void invariant() {

235 }
236 @Perm(requires="unique(this) in alive",
237 ensures="unique(this) in alive")
238   void condition() {

240 }
```

```
241 @Perm(requires="unique(this) in alive",
242 ensures="unique(this) in alive")
243   void endclassstates() {

245 }
246 @Perm(requires="unique(this) in alive",
247 ensures="unique(this) in alive")
248   void refine() {

250 }
251 @Perm(requires="unique(this) in alive",
252 ensures="unique(this) in alive")
253   void states() {

255 }
256 @Perm(requires="unique(this) in alive",
257 ensures="unique(this) in alive")
258   void dimension() {

260 }
261 @Perm(requires="unique(this) in alive",
262 ensures="unique(this) in alive")
263   void value() {

265 }
266 @Perm(requires="unique(this) in alive",
267 ensures="unique(this) in alive")
268   void item() {

270 }
271 @Perm(ensures="none(this) in alive")
272  public String[] getTokenNames() {
273  return null;

275 }

277  public String getGrammarFileName() {
278  return null;

280 }

282 }ENDOFCLASS

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

286 class EJmlSpecification {
287 @Perm(ensures="unique(this) in alive")
288 EJmlSpecification() {   }

290 @Perm(requires="share(this) in alive",
291 ensures="share(this) in alive")
292   void setDimensionName(String str) {

294 }
295 @Perm(requires="share(this) in alive",
296 ensures="share(this) in alive")
297   void setDimensionValues(int low, int high) {

299 }
300 @Perm(requires="share(this) in alive",
301 ensures="share(this) in alive")
302   void addRequires(String str) {

304 }
305 @Perm(requires="share(this) in alive",
306 ensures="share(this) in alive")
307   void setPerm(String str) {

309 }
310 @Perm(requires="share(this) in alive",
311 ensures="share(this) in alive")
312   void setEnsures(String str) {

314 }
315 @Perm(requires="share(this) in alive",
316 ensures="share(this) in alive")
317   String JmlClassSpec2PluralClassSpec() {
318  return null;

320 }
321 @Perm(requires="unique(this) in alive",
```

```
322 ensures="unique(this) in alive")
323   void reset() {

325 }
326 @Perm(requires="share(this) in alive",
327 ensures="share(this) in alive")
328   String JmlMethodSpec2PluralMethodSpec() {
329  return null;

331 }
332 @Perm(requires="share(this) in alive",
333 ensures="share(this) in alive")
334   String moreRequires() {
335  return null;

337 }
338 @Perm(requires="pure(this) in alive",
339 ensures="pure(this) in alive")
340   String getPerm() {
341  return null;

343 }
344 @Perm(requires="share(this) in alive",
345 ensures="share(this) in alive")
346   String determineEnsures(String req) {
347  return null;

349 }
350 @Perm(requires="share(this) in alive",
351 ensures="share(this) in alive")
352   String oneRequires() {
353  return null;

355 }
356 @Perm(requires="share(this) in alive",
357 ensures="share(this) in alive")
358   String noRequires() {
359  return null;

361 }

363 }ENDOFCLASS

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

367 class EGhost {
368 @Perm(ensures="unique(this) in alive")
369 EGhost() {   }

371 @Perm(requires="share(this) in alive",
372 ensures="share(this) in alive")
373  public void setDimensionName(String str) {

375 }
376 @Perm(requires="share(this) in alive",
377 ensures="share(this) in alive")
378  public void setDimensionValues(int low, int high) {

380 }
381 @Perm(requires="pure(this) in alive",
382 ensures="pure(this) in alive")
383  public String getDimensionName() {
384  return null;

386 }
387 @Perm(requires="pure(this) in alive",
388 ensures="pure(this) in alive")
389  public int getLowValueofInv() {
390  return 0;

392 }
393 @Perm(requires="pure(this) in alive",
394 ensures="pure(this) in alive")
395  public int getHighValueofInv() {
396  return 0;

398 }

400 }ENDOFCLASS

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

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

@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public String toString() {
 return null;

}

}ENDOFCLASS

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

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


  String readFile(String pathname) {
 return null;

}

}ENDOFCLASS

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

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


 private CompilationUnit getCompilationUnit(String prog) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void analyzeFromCommandLine(LinkedList<String> inputFiles, String strType, String strK) {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void callModelCheckerThroughCommandLine() {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void printMetrics() {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void printMethodMetrics() {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  Time getTime() {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void CreatePdfSummary_CommandLine() {

}

  void makePdfCommandLine() {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public void analyzeFromPlugin(List<ICompilationUnit> compilationUnitList, int test) {

}

 public String getInputStream(ICompilationUnit unit) {
```

```
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void callModelCheckerThroughPlugin() {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void createPdfSummaryPlugin() {

}

  void makePdfPlugin() {

}

}ENDOFCLASS

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

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

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void reset() {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  String modifyConstructorSpecifications(String prog) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  EPackage getPkgObject() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void addRequiresAPTS(String ap, String ts) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void addRequiresParamAPTS(String ap, String ts, String argumentNumber) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void addEnsuresAPTS(String ap, String ts) {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void addEnsuresResultAPTS(String ap, String ts) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void addEnsuresParamAPTS(String ap, String ts, String argumentNumber) {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void addCase() {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void addState(String stateName) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void addBoolStateInvariant(String variable, String operator, String value) {
```

```
566 }
567 @Perm(requires="share(this) in alive",
568 ensures="share(this) in alive")
569   void addStateInvariant(String accessPermission, String variable, String state) {

571 }
572 @Perm(requires="share(this) in alive",
573 ensures="share(this) in alive")
574   void addDimension(String name) {

576 }
577 @Perm(requires="share(this) in alive",
578 ensures="share(this) in alive")
579   void addDimensionValue(String value) {

581 }

583   void addPkgObject(EPackage _pkg) {

585 }

587 }ENDOFCLASS

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

591 class EPackage {
592 @Perm(ensures="unique(this) in alive")
593 EPackage() {   }

595 @Perm(requires="pure(this) in alive",
596 ensures="pure(this) in alive")
597  public LinkedList<EClass> getClasses() {
598  return null;

600 }
601 @Perm(requires="unique(this) in alive",
602 ensures="unique(this) in alive")
603  public int getTotalStates() {
604  return 0;

606 }
607 @Perm(requires="pure(this) in alive",
608 ensures="pure(this) in alive")
609  public int getTotalReachableStates() {
610  return 0;

612 }
613 @Perm(requires="pure(this) in alive",
614 ensures="pure(this) in alive")
615  public String getSinkStates() {
616  return null;

618 }
619 @Perm(requires="share(this) in alive",
620 ensures="share(this) in alive")
621  public void setSinkStates(String sinkStates) {

623 }
624 @Perm(requires="share(this) in alive",
625 ensures="share(this) in alive")
626  public void setName(String str) {

628 }
629 @Perm(requires="unique(this) in alive",
630 ensures="unique(this) in alive")
631  public String getName() {
632  return null;

634 }

636 }ENDOFCLASS

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

640 class EClass {
641 @Perm(ensures="unique(this) in alive")
642 EClass() {   }

644 @Perm(requires="pure(this) in alive",
645 ensures="pure(this) in alive")
```

```
646   public LinkedList<EMethod> getMethods() {
647   return null;

649  }
650  @Perm(requires="unique(this) in alive",
651  ensures="unique(this) in alive")
652   public String getName() {
653   return null;

655  }
656  @Perm(requires="pure(this) in alive",
657  ensures="pure(this) in alive")
658   public String getSuperClassName() {
659   return null;

661  }
662  @Perm(requires="pure(this) in alive",
663  ensures="pure(this) in alive")
664   public LinkedList<EField> getFields() {
665   return null;

667  }
668  @Perm(requires="share(this) in alive",
669  ensures="share(this) in alive")
670   public boolean hasMoreThanOneDimension() {
671   return 0;

673  }
674  @Perm(requires="pure(this) in alive",
675  ensures="pure(this) in alive")
676   public LinkedList<EDim> getDimensions() {
677   return null;

679  }
680  @Perm(requires="pure(this) in alive",
681  ensures="pure(this) in alive")
682   public LinkedList<EState> getStates() {
683   return null;

685  }
686  @Perm(requires="pure(this) in alive",
687  ensures="pure(this) in alive")
688   public int getIndex() {
689   return 0;

691  }
692  @Perm(requires="share(this) in alive",
693  ensures="share(this) in alive")
694   public EMethod getConstructor() {
695   return null;

697  }
698  @Perm(requires="pure(this) in alive",
699  ensures="pure(this) in alive")
700   public int findStateIndex(String st) {
701   return 0;

703  }
704  @Perm(requires="immutable(this) in alive",
705  ensures="immutable(this) in alive")
706   public ArrayList<String> getVariablesofBooleanInvariants() {
707   return null;

709  }
710  @Perm(requires="immutable(this) in alive",
711  ensures="immutable(this) in alive")
712   public LinkedList<String> getTransitions() {
713   return null;

715  }
716  @Perm(requires="pure(this) in alive",
717  ensures="pure(this) in alive")
718   public LinkedList<EState> getReachableStates() {
719   return null;

721  }
722  @Perm(requires="unique(this) in alive",
723  ensures="unique(this) in alive")
724   public int getTotalStates() {
725   return 0;
```

```
727  }
728  @Perm(requires="pure(this) in alive",
729  ensures="pure(this) in alive")
730   public int getTotalReachableStates() {
731   return 0;

733  }
734  @Perm(requires="immutable(this) in alive",
735  ensures="immutable(this) in alive")
736   public void addClassStatesSpecifications(String annotation) {

738  }
739  @Perm(requires="share(this) in alive",
740  ensures="share(this) in alive")
741   public void setName(String str) {

743  }
744  @Perm(requires="share(this) in alive",
745  ensures="share(this) in alive")
746   public void setSuperClassName(String str) {

748  }
749  @Perm(requires="share(this) in alive",
750  ensures="share(this) in alive")
751   public void addField(EField field) {

753  }
754  @Perm(requires="share(this) in alive",
755  ensures="share(this) in alive")
756   public void addMethod(EMethod method) {

758  }
759  @Perm(requires="share(this) in alive",
760  ensures="share(this) in alive")
761   public void addState(EState state) {

763  }
764  @Perm(requires="share(this) in alive",
765  ensures="share(this) in alive")
766   public void addDimension(EDim dim) {

768  }
769  @Perm(requires="share(this) in alive",
770  ensures="share(this) in alive")
771   public void setIndex(int classIndex) {

773  }
774  @Perm(requires="share(this) in alive",
775  ensures="share(this) in alive")
776   public void createObject() {

778  }
779  @Perm(requires="pure(this) in alive",
780  ensures="pure(this) in alive")
781   public int getLastObjectIndex() {
782   return 0;

784  }

786  }ENDOFCLASS

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

790  class EMethod {
791  @Perm(ensures="unique(this) in alive")
792  EMethod() {   }

794  @Perm(requires="unique(this) in alive",
795  ensures="unique(this) in alive")
796   public String getName() {
797   return null;

799  }
800  @Perm(requires="immutable(this) in alive",
801  ensures="immutable(this) in alive")
802   public LinkedList<ESpecification> getRequiresAPTS() {
803   return null;

805  }
806  @Perm(requires="immutable(this) in alive",
807  ensures="immutable(this) in alive")
```

```
808   public LinkedList<ESpecification> getEnsuresAPTS() {
809   return null;

811  }
812  @Perm(requires="unique(this) in alive",
813  ensures="unique(this) in alive")
814   public String getIdentifier() {
815   return null;

817  }
818  @Perm(requires="pure(this) in alive",
819  ensures="pure(this) in alive")
820   public LinkedList<EParameter> getParameters() {
821   return null;

823  }
824  @Perm(requires="pure(this) in alive",
825  ensures="pure(this) in alive")
826   public int getIndex() {
827   return 0;

829  }
830  @Perm(requires="pure(this) in alive",
831  ensures="pure(this) in alive")
832   public boolean getRequiresClauseSatisfiability() {
833   return 0;

835  }
836  @Perm(requires="share(this) in alive",
837  ensures="share(this) in alive")
838   public Boolean isConcurrentMethod() {
839   return null;

841  }
842  @Perm(requires="share(this) in alive",
843  ensures="share(this) in alive")
844   public void setRequiresClauseSatisfiability(Boolean flag) {

846  }
847  @Perm(requires="share(this) in alive",
848  ensures="share(this) in alive")
849   public void setConcurrentMethod(String toMethod) {

851  }
852  @Perm(requires="share(this) in alive",
853  ensures="share(this) in alive")
854   public void addSpecifications(String annotation) {

856  }
857  @Perm(requires="share(this) in alive",
858  ensures="share(this) in alive")
859   public void setName(String str) {

861  }
862  @Perm(requires="share(this) in alive",
863  ensures="share(this) in alive")
864   public void setReturnType(String str) {

866  }
867  @Perm(requires="share(this) in alive",
868  ensures="share(this) in alive")
869   public void setIdentifier(String str) {

871  }
872  @Perm(requires="share(this) in alive",
873  ensures="share(this) in alive")
874   public void addParameter(EParameter parameter) {

876  }
877  @Perm(requires="unique(this) in alive",
878  ensures="unique(this) in alive")
879   public String getReturnType() {
880   return null;

882  }
883  @Perm(requires="share(this) in alive",
884  ensures="share(this) in alive")
885   public void setCaseNumber(int x) {

887  }
888  @Perm(requires="pure(this) in alive",
```

```
ensures="pure(this) in alive")
 public int getCaseNumber() {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setIndex(int methodIndex) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setJMLPermission(String Permission) {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public String getJMLPermission() {
 return null;

}

}ENDOFCLASS

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

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

@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setAP(String ap) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public EClass getParentClass() {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public String getFieldName() {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public String getTS() {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public String getAP() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setAPTS(String ap, String ts) {

}

 public Object clone() {
 return null;

}

}ENDOFCLASS

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

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

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void createFromCommandLine(String prog, String className) {
```

```
971  }
972  @Perm(requires="unique(this) in alive",
973  ensures="unique(this) in alive")
974    void createFromPlugin(String prog, String className) {

976  }

978  }ENDOFCLASS

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

982  class ESMCModel {
983  @Perm(ensures="unique(this) in alive")
984  ESMCModel() {   }

986  @Perm(requires="share(this) in alive",
987  ensures="share(this) in alive")
988    void setK(int k) {

990  }
991  @Perm(requires="unique(this) in alive",
992  ensures="unique(this) in alive")
993    void generateSMCmodelCommandLine(int testType) {

995  }
996  @Perm(requires="unique(this) in alive",
997  ensures="unique(this) in alive")
998    void Transitions() {

1000 }
1001 @Perm(requires="pure(this) in alive",
1002 ensures="pure(this) in alive")
1003   String comment(String str) {
1004  return null;

1006 }
1007 @Perm(requires="unique(this) in alive",
1008 ensures="unique(this) in alive")
1009   void declarationsAndinitilizations() {

1011 }
1012 @Perm(requires="unique(this) in alive",
1013 ensures="unique(this) in alive")
1014   void initialize(LinkedList<EClass> _listClasses) {

1016 }
1017 @Perm(requires="unique(this) in alive",
1018 ensures="unique(this) in alive")
1019   void modelAlias(String className, Integer objectIndex, Integer refIndex) {

1021 }
1022 @Perm(requires="unique(this) in alive",
1023 ensures="unique(this) in alive")
1024   boolean isClassExist(String className) {
1025  return 0;

1027 }
1028 @Perm(requires="unique(this) in alive",
1029 ensures="unique(this) in alive")
1030   void createInstanceInModel(EClass _class, String name, int objectIndex, int J) {

1032 }
1033 @Perm(requires="unique(this) in alive",
1034 ensures="unique(this) in alive")
1035   void modelPrimePCandMethod(EClass _class, EMethod _method, Integer objectIndex, Integer refIndex) {

1037 }
1038 @Perm(requires="unique(this) in alive",
1039 ensures="unique(this) in alive")
1040   void startMethod(EClass _class, EMethod _method, Integer objectIndex, Integer refIndex) {

1042 }
1043 @Perm(requires="unique(this) in alive",
1044 ensures="unique(this) in alive")
1045   void modelPCConstructor(EClass _class, Integer objectIndex, Integer refIndex, EClass _currentClass) {

1047 }
1048 @Perm(requires="unique(this) in alive",
1049 ensures="unique(this) in alive")
1050   void modelAPs(EClass _class, Integer objectIndex, Integer refIndex) {
```

```
1052 }
1053 @Perm(requires="unique(this) in alive",
1054 ensures="unique(this) in alive")
1055   EClass getClass(String className) {
1056  return null;

1058 }
1059 @Perm(requires="unique(this) in alive",
1060 ensures="unique(this) in alive")
1061   int getObjectIndex(EClass _class, String variable) {
1062  return 0;

1064 }
1065 @Perm(requires="unique(this) in alive",
1066 ensures="unique(this) in alive")
1067   void modelPCMethod(EClass _class, Integer objectIndex, Integer refIndex) {

1069 }
1070 @Perm(requires="unique(this) in alive",
1071 ensures="unique(this) in alive")
1072   EClass getFieldClass(EClass _class, String fieldName) {
1073  return null;

1075 }
1076 @Perm(requires="pure(this) in alive",
1077 ensures="pure(this) in alive")
1078   int getDimensionIndex(EClass _class, String ts) {
1079  return 0;

1081 }
1082 @Perm(requires="unique(this) in alive",
1083 ensures="unique(this) in alive")
1084   void startAPTS(EClass _class, EMethod _method, String ap, String stateName, Integer objectIndex,
        Integer refIndex) {

1086 }
1087 @Perm(requires="unique(this) in alive",
1088 ensures="unique(this) in alive")
1089   void startAPTSPARAM(EMethod _method, Integer J) {

1091 }

1093   void error(String state, String method) {

1095 }
1096 @Perm(requires="unique(this) in alive",
1097 ensures="unique(this) in alive")
1098   void startPrimeTSPARAM(EMethod method, Integer refIndex) {

1100 }
1101 @Perm(requires="unique(this) in alive",
1102 ensures="unique(this) in alive")
1103   void starPrimeAP(String ap, EClass _class, Integer objectIndex, Integer refIndex, String stateName) {

1105 }
1106 @Perm(requires="unique(this) in alive",
1107 ensures="unique(this) in alive")
1108   void modelPrimeConstructor(EClass _class, Integer objectIndex, Integer refIndex) {

1110 }
1111 @Perm(requires="unique(this) in alive",
1112 ensures="unique(this) in alive")
1113   void modelInheritance(EClass _class, Integer objectIndex, Integer refIndex) {

1115 }
1116 @Perm(requires="unique(this) in alive",
1117 ensures="unique(this) in alive")
1118   void modelPrimeAPStateInvariants(EClass _class, Integer refIndex, String stateName) {

1120 }
1121 @Perm(requires="unique(this) in alive",
1122 ensures="unique(this) in alive")
1123   int getClassIndex(String name) {
1124  return 0;

1126 }
1127 @Perm(requires="share(this) in alive",
1128 ensures="share(this) in alive")
1129   void modelPrimeAP(String ap, String className, Integer objectIndex, Integer refIndex) {
```

```
1131  }
1132  @Perm(requires="pure(this) in alive",
1133  ensures="pure(this) in alive")
1134    int getAPId(String ap) {
1135   return 0;

1137  }
1138  @Perm(requires="unique(this) in alive",
1139  ensures="unique(this) in alive")
1140    void modelEndPCMethod(EClass _class, EMethod _method, Integer objectIndex, Integer refIndex) {

1142  }
1143  @Perm(requires="unique(this) in alive",
1144  ensures="unique(this) in alive")
1145    void endMethod(EClass _class, EMethod _method, Integer objectIndex, Integer refIndex) {

1147  }
1148  @Perm(requires="unique(this) in alive",
1149  ensures="unique(this) in alive")
1150    void modelEndPCConstructor(EClass _class, EMethod _method, Integer objectIndex, Integer refIndex,
          EClass _currentClass) {

1152  }
1153  @Perm(requires="unique(this) in alive",
1154  ensures="unique(this) in alive")
1155    void modelPrimePCConstructor(EClass _class, Integer objectIndex, Integer refIndex, EClass
          _currentClass) {

1157  }
1158  @Perm(requires="unique(this) in alive",
1159  ensures="unique(this) in alive")
1160    void modelPrimePC(EClass _class, Integer objectIndex, Integer refIndex) {

1162  }
1163  @Perm(requires="unique(this) in alive",
1164  ensures="unique(this) in alive")
1165    void endPrimeAPTS(EClass _class, String methodName, String ap, String stateName, Integer objectIndex,
          Integer refIndex) {

1167  }
1168  @Perm(requires="unique(this) in alive",
1169  ensures="unique(this) in alive")
1170    void modelendConstructor(EClass _class, EMethod _method, Integer refIndex) {

1172  }
1173  @Perm(requires="unique(this) in alive",
1174  ensures="unique(this) in alive")
1175    void updateBoolStateInvariants(EClass _class, String methodName, String stateName, Integer objectIndex
          ) {

1177  }
1178  @Perm(requires="unique(this) in alive",
1179  ensures="unique(this) in alive")
1180    void updateStateInvariants(EClass _class, String methodName, String stateName, Integer refIndex) {

1182  }
1183  @Perm(requires="unique(this) in alive",
1184  ensures="unique(this) in alive")
1185    void updateState(String methodName, String state, EClass _class, Integer objectIndex) {

1187  }
1188  @Perm(requires="unique(this) in alive",
1189  ensures="unique(this) in alive")
1190    void modelState(EClass _class, Integer objectIndex, EMethod _method, String stateName) {

1192  }
1193  @Perm(requires="unique(this) in alive",
1194  ensures="unique(this) in alive")
1195    void modelStateInvariants(EClass _class, int refIndex, EMethod _method, String stateName) {

1197  }
1198  @Perm(requires="unique(this) in alive",
1199  ensures="unique(this) in alive")
1200    void modelBoolStateInvariants(EClass _class, Integer objectIndex, String stateName) {

1202  }
1203  @Perm(requires="unique(this) in alive",
1204  ensures="unique(this) in alive")
1205    void methodsReachability(EClass _class, Integer objectIndex, Integer refIndex) {

1207  }
```

```
1208 @Perm(requires="unique(this) in alive",
1209 ensures="unique(this) in alive")
1210   void modelAP(EClass _class, Integer objectIndex, Integer refIndex, String ap) {

1212 }
1213 @Perm(requires="share(this) in alive",
1214 ensures="share(this) in alive")
1215   void updateTokens(String ap, String className, Integer objectIndex, Integer refIndex) {

1217 }
1218 @Perm(requires="unique(this) in alive",
1219 ensures="unique(this) in alive")
1220   void endPrimeAPTSPARAM(EMethod method, Integer refIndex) {

1222 }
1223 @Perm(requires="share(this) in alive",
1224 ensures="share(this) in alive")
1225   void initilizeVariables(String className, int objectIndex, EClass _class, int modifier) {

1227 }
1228 @Perm(requires="share(this) in alive",
1229 ensures="share(this) in alive")
1230   void initilizeKVariables(String className, int objectIndex, int K) {

1232 }
1233 @Perm(requires="share(this) in alive",
1234 ensures="share(this) in alive")
1235   void defineVariables(String className, int objectIndex, EClass _class, int modifier) {

1237 }
1238 @Perm(requires="share(this) in alive",
1239 ensures="share(this) in alive")
1240   void defineKVariables(String className, int objectIndex, EClass _class, int K) {

1242 }
1243 @Perm(requires="pure(this) in alive",
1244 ensures="pure(this) in alive")
1245   boolean isPrivateAndIndexEqualToZero(int refIndex, EField _field) {
1246  return 0;

1248 }
1249 @Perm(requires="share(this) in alive",
1250 ensures="share(this) in alive")
1251   void generateSMCmodelPlugin(EPackage _pkg, int testType) {

1253 }
1254 @Perm(requires="unique(this) in alive",
1255 ensures="unique(this) in alive")
1256   void createAlias() {

1258 }
1259 @Perm(requires="unique(this) in alive",
1260 ensures="unique(this) in alive")
1261   void addIndexes() {

1263 }
1264 @Perm(requires="unique(this) in alive",
1265 ensures="unique(this) in alive")
1266   void createDimensionsObject(EClass _class) {

1268 }
1269 @Perm(requires="unique(this) in alive",
1270 ensures="unique(this) in alive")
1271   void createDimensionAsField(EClass _class, EDim _dim, int count) {

1273 }
1274 @Perm(requires="unique(this) in alive",
1275 ensures="unique(this) in alive")
1276   void createParentObject(EClass _class) {

1278 }
1279 @Perm(requires="unique(this) in alive",
1280 ensures="unique(this) in alive")
1281   void createParentAsField(EClass _class, EClass _currentClass) {

1283 }
1284 @Perm(requires="unique(this) in alive",
1285 ensures="unique(this) in alive")
1286   void addInvariantStateIndex(EClass _class) {

1288 }
```

```
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void setInvariantVariableType(EClass _class, EInvariant inv) {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void Spec() {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void statesAdjancyMatrix(EClass _class, Integer objectIndex) {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void concurrentMethods(EClass _class, Integer objectIndex, Integer refIndex) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void sinkStates() {

}

}ENDOFCLASS

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

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

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public String getName() {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public int getObjectIndex() {
 return 0;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public String getType() {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public int getClassIndex() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public int getModifier() {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setName(String str) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setType(String str) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setModifier(int mod) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
```

```
1370  public void setClassIndex(int classIndex) {

1372 }
1373 @Perm(requires="share(this) in alive",
1374 ensures="share(this) in alive")
1375  public void setObjectIndex(int objectIndex) {

1377 }

1379 }ENDOFCLASS

1381 @ClassStates({@State(name = "alive")})

1383 class EDim {
1384 @Perm(ensures="unique(this) in alive")
1385 EDim() {   }

1387 @Perm(requires="pure(this) in alive",
1388 ensures="pure(this) in alive")
1389  public ArrayList<String> getValues() {
1390  return null;

1392 }
1393 @Perm(requires="share(this) in alive",
1394 ensures="share(this) in alive")
1395  public void addValue(String str) {

1397 }
1398 @Perm(requires="full(this) in alive",
1399 ensures="full(this) in alive")
1400  public void setName(String str) {

1402 }
1403 @Perm(requires="pure(this) in alive",
1404 ensures="pure(this) in alive")
1405  public String getName(String str) {
1406  return null;

1408 }

1410 }ENDOFCLASS

1412 @ClassStates({@State(name = "alive")})

1414 class EParameter {
1415 @Perm(ensures="unique(this) in alive")
1416 EParameter() {   }

1418 @Perm(requires="immutable(this) in alive",
1419 ensures="immutable(this) in alive")
1420  public LinkedList<ESpecification> getRequiresAPTS() {
1421  return null;

1423 }
1424 @Perm(requires="pure(this) in alive",
1425 ensures="pure(this) in alive")
1426  public String getType() {
1427  return null;

1429 }
1430 @Perm(requires="immutable(this) in alive",
1431 ensures="immutable(this) in alive")
1432  public LinkedList<ESpecification> getEnsuresAPTS() {
1433  return null;

1435 }
1436 @Perm(requires="unique(this) in alive",
1437 ensures="unique(this) in alive")
1438  public String getName() {
1439  return null;

1441 }
1442 @Perm(requires="pure(this) in alive",
1443 ensures="pure(this) in alive")
1444  public int getNumber() {
1445  return 0;

1447 }
1448 @Perm(requires="full(this) in alive",
1449 ensures="full(this) in alive")
1450  public void setNumber(int n) {
```

```
1452 }
1453 @Perm(requires="share(this) in alive",
1454 ensures="share(this) in alive")
1455  public void setName(String str) {

1457 }
1458 @Perm(requires="full(this) in alive",
1459 ensures="full(this) in alive")
1460  public void setType(String str) {

1462 }

1464 }ENDOFCLASS

1466 @ClassStates({@State(name = "alive")})

1468 class EState {
1469 @Perm(ensures="unique(this) in alive")
1470 EState() {   }

1472 @Perm(requires="pure(this) in alive",
1473 ensures="pure(this) in alive")
1474  public String getName() {
1475  return null;

1477 }
1478 @Perm(requires="pure(this) in alive",
1479 ensures="pure(this) in alive")
1480  public LinkedList<EInvariant> getInvariants() {
1481  return null;

1483 }
1484 @Perm(requires="pure(this) in alive",
1485 ensures="pure(this) in alive")
1486  public LinkedList<EBoolInvariant> getBoolInvariants() {
1487  return null;

1489 }
1490 @Perm(requires="pure(this) in alive",
1491 ensures="pure(this) in alive")
1492  public int getStateIndex() {
1493  return 0;

1495 }
1496 @Perm(requires="pure(this) in alive",
1497 ensures="pure(this) in alive")
1498  public int isReachable() {
1499  return 0;

1501 }
1502 @Perm(requires="share(this) in alive",
1503 ensures="share(this) in alive")
1504  public void setReachability(int value) {

1506 }
1507 @Perm(requires="share(this) in alive",
1508 ensures="share(this) in alive")
1509  public Boolean isReachableState() {
1510  return null;

1512 }
1513 @Perm(requires="share(this) in alive",
1514 ensures="share(this) in alive")
1515  public void addBoolInvariant(EBoolInvariant inv) {

1517 }
1518 @Perm(requires="share(this) in alive",
1519 ensures="share(this) in alive")
1520  public void addInvariant(EInvariant inv) {

1522 }
1523 @Perm(requires="share(this) in alive",
1524 ensures="share(this) in alive")
1525  public void setIndex(int stateIndex) {

1527 }

1529 }ENDOFCLASS

1531 @ClassStates({@State(name = "alive")})
```

```
1533 class EInvariant {
1534 @Perm(ensures="unique(this) in alive")
1535 EInvariant() {   }

1537 @Perm(requires="pure(this) in alive",
1538 ensures="pure(this) in alive")
1539  public String getAP() {
1540  return null;

1542 }
1543 @Perm(requires="pure(this) in alive",
1544 ensures="pure(this) in alive")
1545  public String getVariableType() {
1546  return null;

1548 }
1549 @Perm(requires="pure(this) in alive",
1550 ensures="pure(this) in alive")
1551  public String getVariable() {
1552  return null;

1554 }
1555 @Perm(requires="pure(this) in alive",
1556 ensures="pure(this) in alive")
1557  public String getStateName() {
1558  return null;

1560 }
1561 @Perm(requires="unique(this) in alive",
1562 ensures="unique(this) in alive")
1563  public LinkedList<EInvariant> getStateInvariants(EPackage _pkg) {
1564  return null;

1566 }
1567 @Perm(requires="share(this) in alive",
1568 ensures="share(this) in alive")
1569  public void setVariableType(String type) {

1571 }

1573  public void setStateIndex(int stateIndex) {

1575 }
1576 @Perm(requires="full(this) in alive",
1577 ensures="full(this) in alive")
1578  public void setAP(String str) {

1580 }
1581 @Perm(requires="full(this) in alive",
1582 ensures="full(this) in alive")
1583  public void setVariable(String str) {

1585 }
1586 @Perm(requires="full(this) in alive",
1587 ensures="full(this) in alive")
1588  public void setState(String str) {

1590 }

1592 }ENDOFCLASS

1594 @ClassStates({@State(name = "alive")})

1596 class EBoolInvariant {
1597 @Perm(ensures="unique(this) in alive")
1598 EBoolInvariant() {   }

1600 @Perm(requires="immutable(this) in alive",
1601 ensures="immutable(this) in alive")
1602  public String getVariable() {
1603  return null;

1605 }
1606 @Perm(requires="immutable(this) in alive",
1607 ensures="immutable(this) in alive")
1608  public String getValue() {
1609  return null;

1611 }
```

```
}ENDOFCLASS

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

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

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void addTrnsitions(String str) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void parseMethodReachability(String str) {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void createGraph() {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
  int getNumberofUnReachableMethods() {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void setNumberofUnReachableMethods() {

}

}ENDOFCLASS

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

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

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void create_CommandLine() {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void addUsePackages() {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void writeToLatex() {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void WriteSummary() {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void addSummaryTableColumns() {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void addSummaryTableHeaders() {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void addSummaryTableRows() {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void writeRequiresClauseSatisfiabilty(EClass _class) {
```

```
1695  }
1696  @Perm(requires="share(this) in alive",
1697  ensures="share(this) in alive")
1698    void writeStateTransitionMatrix(EClass _class) {

1700  }
1701  @Perm(requires="share(this) in alive",
1702  ensures="share(this) in alive")
1703    void addSTMNumberofColumns(EClass _class) {

1705  }
1706  @Perm(requires="share(this) in alive",
1707  ensures="share(this) in alive")
1708    void addSTMColumnsHeaders(EClass _class) {

1710  }
1711  @Perm(requires="share(this) in alive",
1712  ensures="share(this) in alive")
1713    void addSTMRows(EClass _class) {

1715  }
1716  @Perm(requires="share(this) in alive",
1717  ensures="share(this) in alive")
1718    String getStateReachabilityValue(EState _state, EState __state) {
1719   return null;

1721  }
1722  @Perm(requires="unique(this) in alive",
1723  ensures="unique(this) in alive")
1724    void writeMethodConcurrencyMatrix(EClass _class) {

1726  }
1727  @Perm(requires="share(this) in alive",
1728  ensures="share(this) in alive")
1729    void addConcurrencyMatrixColumns(EClass _class) {

1731  }
1732  @Perm(requires="unique(this) in alive",
1733  ensures="unique(this) in alive")
1734    void addConcurrencyMatrixHeaders(EClass _class) {

1736  }
1737  @Perm(requires="unique(this) in alive",
1738  ensures="unique(this) in alive")
1739    void addConcurrencyMatrixRows(EClass _class) {

1741  }
1742  @Perm(requires="unique(this) in alive",
1743  ensures="unique(this) in alive")
1744    String getConcurrencyValue(EMethod _method, EMethod __method) {
1745   return null;

1747  }
1748  @Perm(requires="share(this) in alive",
1749  ensures="share(this) in alive")
1750    void writeAbbervations() {

1752  }
1753  @Perm(requires="share(this) in alive",
1754  ensures="share(this) in alive")
1755    void reset() {

1757  }
1758  @Perm(requires="unique(this) in alive",
1759  ensures="unique(this) in alive")
1760    void setText(String str) {

1762  }
1763  @Perm(requires="unique(this) in alive",
1764  ensures="unique(this) in alive")
1765    void parseRequires(String str) {

1767  }
1768  @Perm(requires="unique(this) in alive",
1769  ensures="unique(this) in alive")
1770    EMethod getMethod(String className, String methodName) {
1771   return null;

1773  }
1774  @Perm(requires="unique(this) in alive",
```

```
ensures="unique(this) in alive")
  void parseTransitions(String str) {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void parseConcurrentMethods(String str) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void parseSinkStates(String str) {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void create_Plugin() {

}

}ENDOFCLASS

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

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


  ASTNode getASTNodeFromCompilationUnit(ICompilationUnit compUnit) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  List<ICompilationUnit> scanForCompilationUnits() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  List<ICompilationUnit> collectCompilationUnits(IJavaElement javaElement) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  List<ICompilationUnit> findCompilationUnits(List<String> files) {
 return null;

}

  String getWorkspaceRelativeName(IJavaElement element) {
 return null;

}

  Map<ICompilationUnit,ASTNode> parseCompilationUnits(List<ICompilationUnit> compilationUnits) {
 return null;

}

  List<MethodDeclaration> scanForMethodDeclarations(Map<ICompilationUnit,ASTNode>
      compilationUnitToASTNode) {
 return null;

}

  List<MethodDeclaration> scanForMethodDeclarationsFromAST(ASTNode node) {
 return null;

}

}ENDOFCLASS

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

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

```
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 private void addUnparsedSpecifications(String annotation) {

}

 public void preVisit(ASTNode node) {

}

 public void postVisit(ASTNode node) {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public boolean visit(PackageDeclaration node) {
 return 0;

}

 public void endVisit(PackageDeclaration node) {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 private void callParser(String annotation) {

}

}ENDOFCLASS

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

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

@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
  int getInheritance() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
  int getFullModel() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
  int getInvariants() {
 return 0;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
  int getDimensions() {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void setInvariants(int x) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void setAliasPerObject(int x) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void setFullModel(int x) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void setDimensions(int x) {

}
```

```
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void setInheritance(int x) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
  int getAliasPerObject() {
 return 0;

}

}ENDOFCLASS

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

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


}ENDOFCLASS

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

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


}ENDOFCLASS

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

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


}ENDOFCLASS

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

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

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public boolean visit(MethodDeclaration methodDeclaration) {
 return 0;

}

}ENDOFCLASS

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

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

@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void start(BundleContext context) {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public void stop(BundleContext context) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
  Activator getDefault() {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  ImageDescriptor getImageDescriptor(String path) {
```

```
 return null;

}

}ENDOFCLASS

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

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


 public void addHandlerListener(IHandlerListener handlerListener) {

}

 public void dispose() {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public Object execute(ExecutionEvent event) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 private void extractSettings(ExecutionEvent event) {

}

 public boolean isEnabled() {
 return 0;

}

 public boolean isHandled() {
 return 0;

}

 public void removeHandlerListener(IHandlerListener handlerListener) {

}

}ENDOFCLASS

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

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

@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void selectionChanged(IAction action, ISelection selection) {

}

 public void setActivePart(IAction action, IWorkbenchPart targetPart) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void run(IAction action) {

}

}ENDOFCLASS

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

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

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 protected IStatus run(IProgressMonitor monitor) {
 return null;
```

```
}

}ENDOFCLASS

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

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

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

}

  String testRead(String file) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  void seprateJavaFile(String str) {

}

  void anTest() {

}

}ENDOFCLASS

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

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


}ENDOFCLASS

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

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


}ENDOFCLASS

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

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


}ENDOFCLASS

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

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


 public String getGrammarFileName() {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
  void mATFULL() {

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
  void mATPURE() {

}
```

```
2179  @Perm(requires="immutable(this) in alive",
2180  ensures="immutable(this) in alive")
2181    void mATIMMUTABLE() {

2183  }
2184  @Perm(requires="immutable(this) in alive",
2185  ensures="immutable(this) in alive")
2186    void mATSHARE() {

2188  }
2189  @Perm(requires="immutable(this) in alive",
2190  ensures="immutable(this) in alive")
2191    void mATUNIQUE() {

2193  }
2194  @Perm(requires="immutable(this) in alive",
2195  ensures="immutable(this) in alive")
2196    void mPUBLICBEHAVIOR() {

2198  }
2199  @Perm(requires="immutable(this) in alive",
2200  ensures="immutable(this) in alive")
2201    void mFULL() {

2203  }
2204  @Perm(requires="immutable(this) in alive",
2205  ensures="immutable(this) in alive")
2206    void mPURE() {

2208  }
2209  @Perm(requires="immutable(this) in alive",
2210  ensures="immutable(this) in alive")
2211    void mIMMUTABLE() {

2213  }
2214  @Perm(requires="immutable(this) in alive",
2215  ensures="immutable(this) in alive")
2216    void mSHARE() {

2218  }
2219  @Perm(requires="immutable(this) in alive",
2220  ensures="immutable(this) in alive")
2221    void mUNIQUE() {

2223  }
2224  @Perm(requires="immutable(this) in alive",
2225  ensures="immutable(this) in alive")
2226    void mNONE() {

2228  }
2229  @Perm(requires="immutable(this) in alive",
2230  ensures="immutable(this) in alive")
2231    void mLSBRACKET() {

2233  }
2234  @Perm(requires="immutable(this) in alive",
2235  ensures="immutable(this) in alive")
2236    void mRSBRACKET() {

2238  }
2239  @Perm(requires="immutable(this) in alive",
2240  ensures="immutable(this) in alive")
2241    void mPERM() {

2243  }
2244  @Perm(requires="immutable(this) in alive",
2245  ensures="immutable(this) in alive")
2246    void mEQUAL() {

2248  }
2249  @Perm(requires="immutable(this) in alive",
2250  ensures="immutable(this) in alive")
2251    void mEQUALOPERATOR() {

2253  }
2254  @Perm(requires="immutable(this) in alive",
2255  ensures="immutable(this) in alive")
2256    void mIN() {

2258  }
2259  @Perm(requires="immutable(this) in alive",
```

```
2260  ensures="immutable(this) in alive")
2261    void mTHIS() {

2263  }
2264  @Perm(requires="immutable(this) in alive",
2265  ensures="immutable(this) in alive")
2266    void mRESULT() {

2268  }
2269  @Perm(requires="immutable(this) in alive",
2270  ensures="immutable(this) in alive")
2271    void mPARAM() {

2273  }
2274  @Perm(requires="immutable(this) in alive",
2275  ensures="immutable(this) in alive")
2276    void mREQUIRES() {

2278  }
2279  @Perm(requires="immutable(this) in alive",
2280  ensures="immutable(this) in alive")
2281    void mENSURES() {

2283  }
2284  @Perm(requires="immutable(this) in alive",
2285  ensures="immutable(this) in alive")
2286    void mQUOTE() {

2288  }
2289  @Perm(requires="immutable(this) in alive",
2290  ensures="immutable(this) in alive")
2291    void mAND() {

2293  }
2294  @Perm(requires="immutable(this) in alive",
2295  ensures="immutable(this) in alive")
2296    void mUSE() {

2298  }
2299  @Perm(requires="immutable(this) in alive",
2300  ensures="immutable(this) in alive")
2301    void mUSEFIELDS() {

2303  }
2304  @Perm(requires="immutable(this) in alive",
2305  ensures="immutable(this) in alive")
2306    void mPUNCTUATION() {

2308  }
2309  @Perm(requires="immutable(this) in alive",
2310  ensures="immutable(this) in alive")
2311    void mCASES() {

2313  }
2314  @Perm(requires="immutable(this) in alive",
2315  ensures="immutable(this) in alive")
2316    void mLCBRACKET() {

2318  }
2319  @Perm(requires="immutable(this) in alive",
2320  ensures="immutable(this) in alive")
2321    void mRCBRACKET() {

2323  }
2324  @Perm(requires="immutable(this) in alive",
2325  ensures="immutable(this) in alive")
2326    void mCLASSSTATES() {

2328  }
2329  @Perm(requires="immutable(this) in alive",
2330  ensures="immutable(this) in alive")
2331    void mREFINE() {

2333  }
2334  @Perm(requires="immutable(this) in alive",
2335  ensures="immutable(this) in alive")
2336    void mVALUE() {

2338  }
2339  @Perm(requires="immutable(this) in alive",
2340  ensures="immutable(this) in alive")
```

```
2341    void mSTATE() {

2343  }
2344  @Perm(requires="immutable(this) in alive",
2345  ensures="immutable(this) in alive")
2346    void mSTATES() {

2348  }
2349  @Perm(requires="immutable(this) in alive",
2350  ensures="immutable(this) in alive")
2351    void mDIM() {

2353  }
2354  @Perm(requires="immutable(this) in alive",
2355  ensures="immutable(this) in alive")
2356    void mNAME() {

2358  }
2359  @Perm(requires="immutable(this) in alive",
2360  ensures="immutable(this) in alive")
2361    void mINV() {

2363  }
2364  @Perm(requires="immutable(this) in alive",
2365  ensures="immutable(this) in alive")
2366    void mOPERATOR() {

2368  }
2369  @Perm(requires="immutable(this) in alive",
2370  ensures="immutable(this) in alive")
2371    void mSEMICOLON() {

2373  }
2374  @Perm(requires="immutable(this) in alive",
2375  ensures="immutable(this) in alive")
2376    void mLESS() {

2378  }
2379  @Perm(requires="immutable(this) in alive",
2380  ensures="immutable(this) in alive")
2381    void mLESSTHANEQUAL() {

2383  }
2384  @Perm(requires="immutable(this) in alive",
2385  ensures="immutable(this) in alive")
2386    void mGREATER() {

2388  }
2389  @Perm(requires="immutable(this) in alive",
2390  ensures="immutable(this) in alive")
2391    void mGREATERTHANEQUAL() {

2393  }
2394  @Perm(requires="immutable(this) in alive",
2395  ensures="immutable(this) in alive")
2396    void mANDD() {

2398  }
2399  @Perm(requires="immutable(this) in alive",
2400  ensures="immutable(this) in alive")
2401    void mOR() {

2403  }
2404  @Perm(requires="immutable(this) in alive",
2405  ensures="immutable(this) in alive")
2406    void mJMLSTART() {

2408  }
2409  @Perm(requires="immutable(this) in alive",
2410  ensures="immutable(this) in alive")
2411    void mJMLEND() {

2413  }
2414  @Perm(requires="immutable(this) in alive",
2415  ensures="immutable(this) in alive")
2416    void mPLUSMINUSOPERATOR() {

2418  }
2419  @Perm(requires="immutable(this) in alive",
2420  ensures="immutable(this) in alive")
2421    void mASSIGNABLE() {
```

```
2423  }
2424  @Perm(requires="immutable(this) in alive",
2425  ensures="immutable(this) in alive")
2426    void mNOTHING() {

2428  }
2429  @Perm(requires="immutable(this) in alive",
2430  ensures="immutable(this) in alive")
2431    void mEVERYTHING() {

2433  }
2434  @Perm(requires="immutable(this) in alive",
2435  ensures="immutable(this) in alive")
2436    void mGHOST() {

2438  }
2439  @Perm(requires="immutable(this) in alive",
2440  ensures="immutable(this) in alive")
2441    void mINT() {

2443  }
2444  @Perm(requires="immutable(this) in alive",
2445  ensures="immutable(this) in alive")
2446    void mINVARIANT() {

2448  }
2449  @Perm(requires="immutable(this) in alive",
2450  ensures="immutable(this) in alive")
2451    void mOLD() {

2453  }
2454  @Perm(requires="immutable(this) in alive",
2455  ensures="immutable(this) in alive")
2456    void mID() {

2458  }
2459  @Perm(requires="immutable(this) in alive",
2460  ensures="immutable(this) in alive")
2461    void mNUMBERS() {

2463  }
2464  @Perm(requires="immutable(this) in alive",
2465  ensures="immutable(this) in alive")
2466    void mWS() {

2468  }
2469  @Perm(requires="unique(this) in alive",
2470  ensures="unique(this) in alive")
2471   public void mTokens() {

2473  }

2475  }ENDOFCLASS

2477  @ClassStates({@State(name = "alive")})

2479  class DFA7 {
2480  @Perm(ensures="unique(this) in alive")
2481  DFA7() {   }


2484   public String getDescription() {
2485   return null;

2487  }

2489  }ENDOFCLASS

2491  @ClassStates({@State(name = "alive")})

2493  class EAPTypeState {
2494  @Perm(ensures="unique(this) in alive")
2495  EAPTypeState() {   }

2497  @Perm(requires="share(this) in alive",
2498  ensures="share(this) in alive")
2499   public void setAP(String str) {

2501  }
2502  @Perm(requires="unique(this) in alive",
```

```
ensures="unique(this) in alive")
 public String getAP(String str) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void setTS(String str) {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public String getTS() {
 return null;

}

}ENDOFCLASS
```