Summary

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

Table 1: Pulse Analysis Summary

Classes	Methods	States	Unsatisfiable Clauses	Unreachable States	Possible concurrent Methods	Total. no. of pairs	No. of concurrent pairs	Percentage of concurrent Methods
MethodFlowAnalysis	2	1	0	0	1	3	1	33
WorklistNodeOrderComparator	4	1	0	0	0	10	0	0
BranchInsensitiveWorklist	4	1	0	0	2	10	3	30
SingleResult	1	1	0	0	0	1	0	0
WorklistTemplate	5	1	0	0	4	15	8	53
AnalysisResult	7	1	0	0	7	28	27	96
NormalLabel	2	1	0	0	1	3	1	33
IncomingResult	5	1	0	0	2	15	3	20
BooleanLabel	3	1	0	0	2	6	3	50
BranchSensitiveWorklist	5	1	0	0	2	15	3	20
WorklistFactory	4	1	0	0	2	10	3	30
ConcurrentFlowAnalysis	13	1	0	0	12	91	51	56
FlowAnalysis	1	1	0	0	0	1	0	0
Anonymous	6	1	0	0	6	21	20	95
Utilities	4	1	0	0	3	10	3	30
AbstractWorklist	5	1	0	0	2	15	3	20
EclipseNodeFirstCFG	1	1	0	0	0	1	0	0
EclipseCFG	2	1	0	0	0	3	0	0
ExceptionMap	1	1	0	0	0	1	0	0
RunCrystalHandler	7	1	0	0	6	28	20	71
AbstractCrystalPlugin	6	1	0	0	1	21	1	5
Anonymous	2	1	1	0	2	3	2	67
WorkspaceUtilities	11	1	0	0	10	66	55	83
Crystal	10	1	0	0	9	55	20	36
Anonymous	4	1	1	0	4	10	9	90
ControlFlowGraph	9	1	0	0	9	45	17	38
ControlFlowNode	27	1	0	0	5	378	15	4
ControlFlowVisitor	6	1	0	0	5	21	6	29
Direction	3	1	0	0	2	6	3	50
CrystalRuntimeException	1	1	0	0	0	1	0	0
UserConsoleView	6	1	0	0	6	21	10	48
NullPrintWriter	2	1	0	0	2	3	2	67
Anonymous	2	1	1	0	2	3	2	67
ClearWarningHandler	7	1	0	0	6	28	20	71

Box	4	1	0	0	3	10	4	40
DisplayCrystalInfo	5	1	0	0	4	15	7	47
ShortFormatter	2	1	0	0	0	3	0	0
Utilities2	6	1	0	0	5	21	15	71
Option	4	1	0	0	3	10	6	60
Anonymous	5	1	1	0	5	15	14	93
AnalysisMenuPopulator	2	1	0	0	1	3	1	33
CrystalFileAction	4	1	0	0	3	10	3	30
Anonymous	2	1	2	0	2	3	2	67
Freezable	3	1	0	0	2	6	3	50
EnableAnalysisHandler	8	1	0	0	7	36	25	69
CrystalUIAction	5	1	0	0	4	15	10	67
Anonymous	2	1	2	0	2	3	2	67
MethodFindVisitor	2	1	0	0	0	3	0	0
Anonymous	2	1	1	0	2	3	2	67
BindingsCollectorVisitor	3	1	0	0	0	6	0	0
StudentRuntimeException	1	1	0	0	0	1	0	0
Total Classes=51	238	51	9	0	158	1111	405	36

Contents

1 MethodFlowAnalysis

Table 2: Methods Requires Clause Satisfiability

Method	Satisfiability
MethodFlowAnalysis	√
analyzeMethod	√

Table 3: State Transition Matrix

	alive
alive	↑

Table 4: Methods Concurrency Matrix

	MethodFlowAnalysis	analyzeMethod
MethodFlowAnalysis	∦	∦
analyzeMethod	∦	‖

2 WorklistNodeOrderComparator

Table 5: Methods Requires Clause Satisfiability

Method	Satisfiability
WorklistNodeOrderComparator	√
createPostOrderAndPopulateNodeMap	√
registerCfgNode	√
compare	√

Table 6: State Transition Matrix

	alive
alive	↑

Table 7: Methods Concurrency Matrix

	WorklistNodeOrderComparator	createPostOrderAndPopulateNodeMap	registerCfgNode	compare
WorklistNodeOrderComparator	∦	∦	∦	∦
createPostOrderAndPopulateNodeMap	∦	∦	∦	∦
registerCfgNode	∦	∦	∦	∦
compare	∦	∦	∦	∦

3 BranchInsensitiveWorklist

Table 8: Methods Requires Clause Satisfiability

Method	Satisfiability
BranchInsensitiveWorklist	$\surd$
getAnalysisDirection	$\surd$
getLattice	$\surd$
transferNode	$\surd$

Table 9: State Transition Matrix

	alive
alive	$\uparrow$

Table 10: Methods Concurrency Matrix

	BranchInsensitiveWorklist	getAnalysisDirection	getLattice	transferNode
BranchInsensitiveWorklist	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
getAnalysisDirection	$\nparallel$	$\parallel$	$\parallel$	$\nparallel$
getLattice	$\nparallel$	$\parallel$	$\parallel$	$\nparallel$
transferNode	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$

4 SingleResult

Table 11: Methods Requires Clause Satisfiability

Method	Satisfiability
SingleResult	√

Table 12: State Transition Matrix

	alive
alive	↑

5 WorklistTemplate

Table 13: Methods Requires Clause Satisfiability

Method	Satisfiability
WorklistTemplate	$\surd$
performAnalysis	$\surd$
checkNull	$\surd$
createAnalysisResult	$\surd$
incomingLabel	$\surd$

Table 14: State Transition Matrix

	alive
alive	$\uparrow$

Table 15: Methods Concurrency Matrix

	WorklistTemplate	performAnalysis	checkNull	createAnalysisResult	incomingLabel
WorklistTemplate	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
performAnalysis	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$	$\nparallel$
checkNull	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
createAnalysisResult	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
incomingLabel	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$	$\parallel$

6 AnalysisResult

Table 16: Methods Requires Clause Satisfiability

Method	Satisfiability
AnalysisResult	√
getNodeMap	√
getLabeledResultsAfter	√
getLabeledResultsBefore	√
getLattice	√
getCfgStartNode	√
getCfgEndNode	√

Table 17: State Transition Matrix

	alive
alive	↑

Table 18: Methods Concurrency Matrix

	AnalysisResult	getNodeMap	getLabeledResultsAfter	getLabeledResultsBefore	getLattice	getCfgStartNode	getCfgEndNode
AnalysisResult	∦	‖	‖	‖	‖	‖	‖
getNodeMap	‖	‖	‖	‖	‖	‖	‖
getLabeledResultsAfter	‖	‖	‖	‖	‖	‖	‖
getLabeledResultsBefore	‖	‖	‖	‖	‖	‖	‖
getLattice	‖	‖	‖	‖	‖	‖	‖
getCfgStartNode	‖	‖	‖	‖	‖	‖	‖
getCfgEndNode	‖	‖	‖	‖	‖	‖	‖

7 NormalLabel

Table 19: Methods Requires Clause Satisfiability

Method	Satisfiability
NormalLabel	$\surd$
getNormalLabel	$\surd$

Table 20: State Transition Matrix

	alive
alive	$\uparrow$

Table 21: Methods Concurrency Matrix

	NormalLabel	getNormalLabel
NormalLabel	$\nparallel$	$\nparallel$
getNormalLabel	$\nparallel$	$\parallel$

8 IncomingResult

Table 22: Methods Requires Clause Satisfiability

Method	Satisfiability
IncomingResult	$\surd$
get	$\surd$
put	$\surd$
keySet	$\surd$
join	$\surd$

Table 23: State Transition Matrix

	alive
alive	$\uparrow$

Table 24: Methods Concurrency Matrix

	IncomingResult	get	put	keySet	join
IncomingResult	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
get	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\nparallel$
put	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
keySet	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\nparallel$
join	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$

9 BooleanLabel

Table 25: Methods Requires Clause Satisfiability

Method	Satisfiability
BooleanLabel	$\surd$
getBooleanLabel	$\surd$
getBranchValue	$\surd$

Table 26: State Transition Matrix

	alive
alive	$\uparrow$

Table 27: Methods Concurrency Matrix

	BooleanLabel	getBooleanLabel	getBranchValue
BooleanLabel	$\nparallel$	$\nparallel$	$\nparallel$
getBooleanLabel	$\nparallel$	$\parallel$	$\parallel$
getBranchValue	$\nparallel$	$\parallel$	$\parallel$

10 BranchSensitiveWorklist

Table 28: Methods Requires Clause Satisfiability

Method	Satisfiability
BranchSensitiveWorklist	√
getAnalysisDirection	√
getLattice	√
transferNode	√
getLabels	√

Table 29: State Transition Matrix

	alive
alive	↑

Table 30: Methods Concurrency Matrix

	BranchSensitiveWorklist	getAnalysisDirection	getLattice	transferNode	getLabels
BranchSensitiveWorklist	∦	∦	∦	∦	∦
getAnalysisDirection	∦	∥	∥	∦	∦
getLattice	∦	∥	∥	∦	∦
transferNode	∦	∦	∦	∦	∦
getLabels	∦	∦	∦	∦	∦

11 WorklistFactory

Table 31: Methods Requires Clause Satisfiability

Method	Satisfiability
WorklistFactory	√
setMonitor	√
createBranchInsensitiveWorklist	√
createBranchSensitiveWorklist	√

Table 32: State Transition Matrix

	alive
alive	↑

Table 33: Methods Concurrency Matrix

	WorklistFactory	setMonitor	createBranchInsensitiveWorklist	createBranchSensitiveWorklist
WorklistFactory	∦	∦	∦	∦
setMonitor	∦	∦	∦	∦
createBranchInsensitiveWorklist	∦	∦	∥	∥
createBranchSensitiveWorklist	∦	∦	∥	∥

12 ConcurrentFlowAnalysis

Table 34: Methods Requires Clause Satisfiability

Method	Satisfiability
ConcurrentFlowAnalysis	√
createNewFlowAnalysis	√
analyzePreemitively	√
getLabeledResultsAfter	√
getLabeledResultsBefore	√
getResultsAfter	√
getResultsBefore	√
addAsFakeFuture	√
getAnalyzedMethods	√
getEndResults	√
getLabeledEndResult	√
getLabeledStartResult	√
getStartResults	√

Table 35: State Transition Matrix

	alive
alive	↑

Table 36: Methods Concurrency Matrix

	ConcurrentFlowAnalysis	createNewFlowAnalysis	analyzePreemitively	getLabeledResultsAfter	getLabeledResultsBefore	getResultsAfter	getResultsBefore	addAsFakeFuture	getAnalyzedMethods	getEndResults	getLabeledEndResult	getLabeledStartResult	getStartResults
ConcurrentFlowAnalysis	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
createNewFlowAnalysis	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
analyzePreemitively	∦	‖	∦	∦	∦	∦	∦	∦	∦	‖	‖	‖	‖
getLabeledResultsAfter	∦	‖	∦	∦	∦	∦	∦	∦	∦	‖	‖	‖	‖
getLabeledResultsBefore	∦	‖	∦	∦	∦	∦	∦	∦	∦	‖	‖	‖	‖
getResultsAfter	∦	‖	∦	∦	∦	∦	∦	∦	∦	‖	‖	‖	‖
getResultsBefore	∦	‖	∦	∦	∦	∦	∦	∦	∦	‖	‖	‖	‖
addAsFakeFuture	∦	‖	∦	∦	∦	∦	∦	∦	∦	‖	‖	‖	‖
getAnalyzedMethods	∦	‖	∦	∦	∦	∦	∦	∦	‖	‖	‖	‖	‖
getEndResults	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
getLabeledEndResult	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
getLabeledStartResult	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
getStartResults	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖

13 FlowAnalysis

Table 37: Methods Requires Clause Satisfiability

Method	Satisfiability
FlowAnalysis	√

Table 38: State Transition Matrix

	alive
alive	↑

14 Anonymous

Table 39: Methods Requires Clause Satisfiability

Method	Satisfiability
Anonymous	√
call	√
cancel	√
get	√
isCancelled	√
isDone	√

Table 40: State Transition Matrix

	alive
alive	↑

Table 41: Methods Concurrency Matrix

	Anonymous	call	cancel	get	isCancelled	isDone
Anonymous	∦	‖	‖	‖	‖	‖
call	‖	‖	‖	‖	‖	‖
cancel	‖	‖	‖	‖	‖	‖
get	‖	‖	‖	‖	‖	‖
isCancelled	‖	‖	‖	‖	‖	‖
isDone	‖	‖	‖	‖	‖	‖

15 Utilities

Table 42: Methods Requires Clause Satisfiability

Method	Satisfiability
Utilities	√
getMethodDeclaration	√
main	√
methodDeclarationToString	√

Table 43: State Transition Matrix

	alive
alive	↑

Table 44: Methods Concurrency Matrix

	Utilities	getMethodDeclaration	main	methodDeclarationToString
Utilities	∦	∦	∦	∦
getMethodDeclaration	∦	∦	∦	‖
main	∦	∦	∦	‖
methodDeclarationToString	∦	‖	‖	‖

16 AbstractWorklist

Table 45: Methods Requires Clause Satisfiability

Method	Satisfiability
AbstractWorklist	√
getControlFlowGraph	√
getMethod	√
checkBreakpoint	√
checkCancel	√

Table 46: State Transition Matrix

	alive
alive	↑

Table 47: Methods Concurrency Matrix

	AbstractWorklist	getControlFlowGraph	getMethod	checkBreakpoint	checkCancel
AbstractWorklist	∦	∦	∦	∦	∦
getControlFlowGraph	∦	‖	‖	∦	∦
getMethod	∦	‖	‖	∦	∦
checkBreakpoint	∦	∦	∦	∦	∦
checkCancel	∦	∦	∦	∦	∦

17 EclipseNodeFirstCFG

Table 48: Methods Requires Clause Satisfiability

Method	Satisfiability
EclipseNodeFirstCFG	√

Table 49: State Transition Matrix

	alive
alive	↑

18 EclipseCFG

Table 50: Methods Requires Clause Satisfiability

Method	Satisfiability
EclipseCFG	√
createGraph	√

Table 51: State Transition Matrix

	alive
alive	↑

Table 52: Methods Concurrency Matrix

	EclipseCFG	createGraph
EclipseCFG	∦	∦
createGraph	∦	∦

19 ExceptionMap

Table 53: Methods Requires Clause Satisfiability

Method	Satisfiability
ExceptionMap	√

Table 54: State Transition Matrix

	alive
alive	↑

20 RunCrystalHandler

Table 55: Methods Requires Clause Satisfiability

Method	Satisfiability
RunCrystalHandler	√
execute	√
isEnabled	√
isHandled	√
addHandlerListener	√
removeHandlerListener	√
dispose	√

Table 56: State Transition Matrix

	alive
alive	↑

Table 57: Methods Concurrency Matrix

	RunCrystalHandler	execute	isEnabled	isHandled	addHandlerListener	removeHandlerListener	dispose
RunCrystalHandler	∦	∦	∦	∦	∦	∦	∦
execute	∦	∦	‖	‖	‖	‖	‖
isEnabled	∦	‖	‖	‖	‖	‖	‖
isHandled	∦	‖	‖	‖	‖	‖	‖
addHandlerListener	∦	‖	‖	‖	‖	‖	‖
removeHandlerListener	∦	‖	‖	‖	‖	‖	‖
dispose	∦	‖	‖	‖	‖	‖	‖

21 AbstractCrystalPlugin

Table 58: Methods Requires Clause Satisfiability

Method	Satisfiability
AbstractCrystalPlugin	$\surd$
getCrystalInstance	$\surd$
getEnabledAnalyses	$\surd$
enableAnalysis	$\surd$
disableAnalysis	$\surd$
start	$\surd$

Table 59: State Transition Matrix

	alive
alive	$\uparrow$

Table 60: Methods Concurrency Matrix

	AbstractCrystalPlugin	getCrystalInstance	getEnabledAnalyses	enableAnalysis	disableAnalysis	start
AbstractCrystalPlugin	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
getCrystalInstance	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
getEnabledAnalyses	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
enableAnalysis	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
disableAnalysis	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
start	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$

22 Anonymous

Table 61: Methods Requires Clause Satisfiability

Method	Satisfiability
Anonymous	×
run	√

Table 62: State Transition Matrix

	alive
alive	↑

Table 63: Methods Concurrency Matrix

	Anonymous	run
Anonymous	#	‖
run	‖	‖

23 WorkspaceUtilities

Table 64: Methods Requires Clause Satisfiability

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

Table 65: State Transition Matrix

	alive
alive	↑

Table 66: Methods Concurrency Matrix

	WorkspaceUtilities	scanForCompilationUnits	collectCompilationUnits	getASTNodeFromCompilationUnit	parseCompilationUnits	scanForMethodDeclarations	scanForMethodDeclarationsFromAST	scanForBindings	findCompilationUnits	getWorkspaceRelativeName	getDeclNodeFromType
WorkspaceUtilities	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
scanForCompilationUnits	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥
collectCompilationUnits	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥
getASTNodeFromCompilationUnit	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥
parseCompilationUnits	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥
scanForMethodDeclarations	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥
scanForMethodDeclarationsFromAST	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥
scanForBindings	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥
findCompilationUnits	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥
getWorkspaceRelativeName	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥
getDeclNodeFromType	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥

24 Crystal

Table 67: Methods Requires Clause Satisfiability

Method	Satisfiability
Crystal	√
runAnalyses	√
runCrystalJob	√
createJobFromCommand	√
findAnalysisWithName	√
getAnalyses	√
run	√
createCrystalJobFromSingleJobs	√
registerAnalysis	√
registerAnnotation	√

Table 68: State Transition Matrix

	alive
alive	↑

Table 69: Methods Concurrency Matrix

	Crystal	runAnalyses	runCrystalJob	createJobFromCommand	findAnalysisWithName	getAnalyses	run	createCrystalJobFromSingleJobs	registerAnalysis	registerAnnotation
Crystal	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
runAnalyses	∦	∦	‖	∦	∦	∦	∦	‖	∦	∦
runCrystalJob	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖
createJobFromCommand	∦	∦	‖	∦	∦	∦	∦	‖	∦	∦
findAnalysisWithName	∦	∦	‖	∦	‖	‖	∦	‖	∦	∦
getAnalyses	∦	∦	‖	∦	‖	‖	∦	‖	∦	∦
run	∦	∦	‖	∦	∦	∦	∦	‖	∦	∦
createCrystalJobFromSingleJobs	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖
registerAnalysis	∦	∦	‖	∦	∦	∦	∦	‖	∦	∦
registerAnnotation	∦	∦	‖	∦	∦	∦	∦	‖	∦	∦

25 Anonymous

Table 70: Methods Requires Clause Satisfiability

Method	Satisfiability
Anonymous	×
analyses	√
compilationUnits	√
reporter	√

Table 71: State Transition Matrix

	alive
alive	↑

Table 72: Methods Concurrency Matrix

	Anonymous	analyses	compilationUnits	reporter
Anonymous	∦	‖	‖	‖
analyses	‖	‖	‖	‖
compilationUnits	‖	‖	‖	‖
reporter	‖	‖	‖	‖

26 ControlFlowGraph

Table 73: Methods Requires Clause Satisfiability

Method	Satisfiability
ControlFlowGraph	√
addControlFlowNode	√
removeControlFlowNode	√
getControlFlowNode	√
getStartNode	√
getEndNode	√
toString	√
getNodeSet	√
buildNodeList	√

Table 74: State Transition Matrix

	alive
alive	↑

Table 75: Methods Concurrency Matrix

	ControlFlowGraph	addControlFlowNode	removeControlFlowNode	getControlFlowNode	getStartNode	getEndNode	toString	getNodeSet	buildNodeList
ControlFlowGraph	∦	∦	∦	∦	∦	‖	∦	∦	∦
addControlFlowNode	∦	∦	∦	∦	∦	‖	‖	∦	∦
removeControlFlowNode	∦	∦	∦	∦	∦	‖	‖	∦	∦
getControlFlowNode	∦	∦	∦	∦	∦	‖	‖	∦	∦
getStartNode	∦	∦	∦	∦	‖	‖	‖	∦	∦
getEndNode	‖	‖	‖	‖	‖	‖	‖	‖	‖
toString	∦	‖	‖	‖	‖	‖	‖	‖	‖
getNodeSet	∦	∦	∦	∦	∦	‖	‖	∦	∦
buildNodeList	∦	∦	∦	∦	∦	‖	‖	∦	∦

27 ControlFlowNode

Table 76: Methods Requires Clause Satisfiability

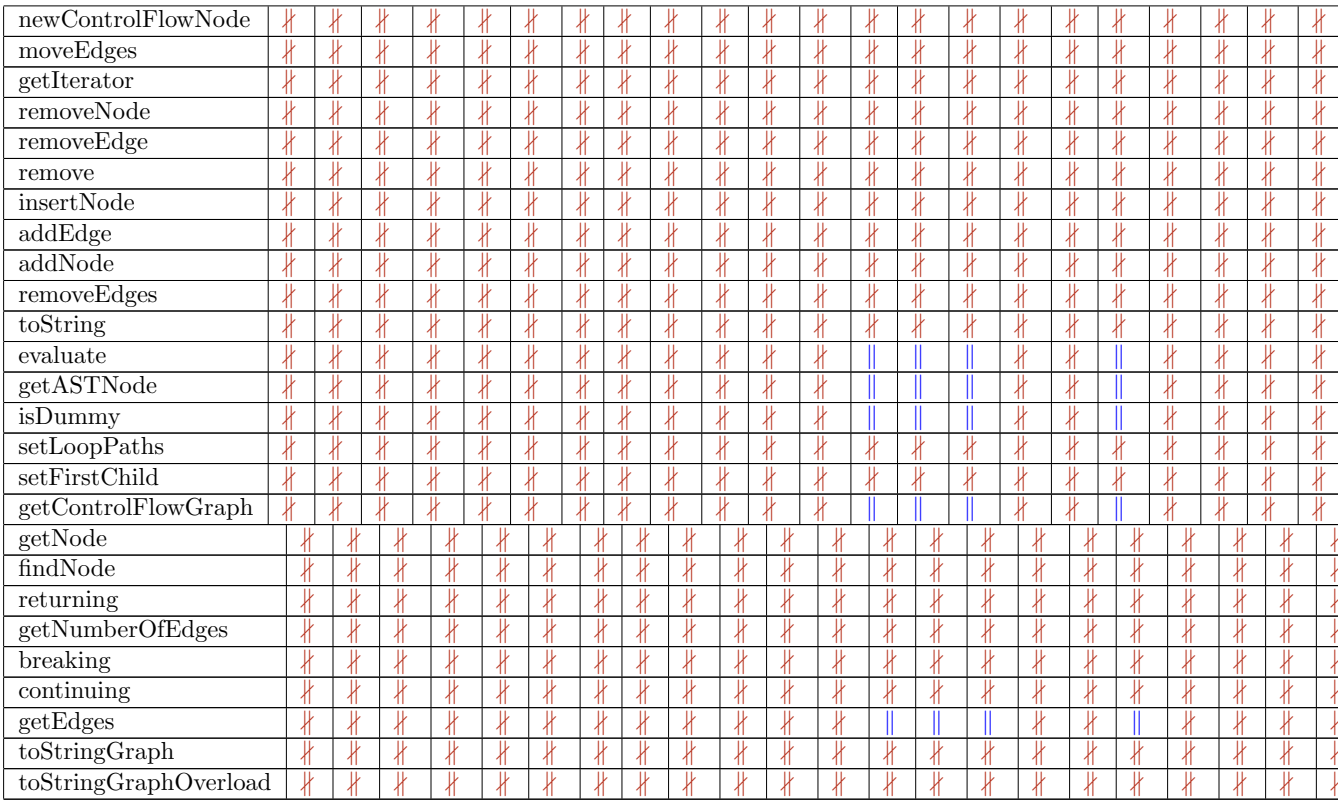
Method	Satisfiability
ControlFlowNode	√
newControlFlowNode	√
moveEdges	√
getIterator	√
removeNode	√
removeEdge	√
remove	√
insertNode	√
addEdge	√
addNode	√
removeEdges	√
toString	√
evaluate	√
getASTNode	√
isDummy	√
setLoopPaths	√
setFirstChild	√
getControlFlowGraph	√
getNode	√
findNode	√
returning	√
getNumberOfEdges	√
breaking	√
continuing	√
getEdges	√
toStringGraph	√
toStringGraphOverload	√

Table 77: State Transition Matrix

	alive
alive	↑

Table 78: Methods Concurrency Matrix

	ControlFlowNode	newControlFlowNode	moveEdges	getIterator	removeNode	removeEdge	remove	insertNode	addEdge	addNode	removeEdges	toString	evaluate	getASTNode	isDummy	setLoopPaths	setFirstChild	getControlFlowGraph	getNode	findNode	returning	getNumberOfEdges
ControlFlowNode	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦


newControlFlowNode
moveEdges
getIterator
removeNode
removeEdge
remove
insertNode
addEdge
addNode
removeEdges
toString
evaluate
getASTNode
isDummy
setLoopPaths
setFirstChild
getControlFlowGraph
getNode
findNode
returning
getNumberOfEdges
breaking
continuing
getEdges
toStringGraph
toStringGraphOverload

28 ControlFlowVisitor

Table 79: Methods Requires Clause Satisfiability

Method	Satisfiability
ControlFlowVisitor	√
visit	√
performVisit	√
createCFNListFromASTNodeList	√
preVisit	√
evaluate	√

Table 80: State Transition Matrix

	alive
alive	↑

Table 81: Methods Concurrency Matrix

	ControlFlowVisitor	visit	performVisit	createCFNListFromASTNodeList	preVisit	evaluate
ControlFlowVisitor	∦	∦	∦	∦	∦	∦
visit	∦	∦	∦	∦	‖	∦
performVisit	∦	∦	‖	∦	‖	∦
createCFNListFromASTNodeList	∦	∦	∦	∦	‖	∦
preVisit	∦	‖	‖	‖	‖	‖
evaluate	∦	∦	∦	∦	‖	∦

29 Direction

Table 82: Methods Requires Clause Satisfiability

Method	Satisfiability
Direction	√
changeDirection	√
toString	√

Table 83: State Transition Matrix

	alive
alive	↑

Table 84: Methods Concurrency Matrix

	Direction	changeDirection	toString
Direction	∦	∦	∦
changeDirection	∦	‖	‖
toString	∦	‖	‖

30 CrystalRuntimeException

Table 85: Methods Requires Clause Satisfiability

Method	Satisfiability
CrystalRuntimeException	√

Table 86: State Transition Matrix

	alive
alive	↑

31 UserConsoleView

Table 87: Methods Requires Clause Satisfiability

Method	Satisfiability
UserConsoleView	√
getInstance	√
createPartControl	√
getPrintWriter	√
clearConsole	√
setFocus	√

Table 88: State Transition Matrix

	alive
alive	↑

Table 89: Methods Concurrency Matrix

	UserConsoleView	getInstance	createPartControl	getPrintWriter	clearConsole	setFocus
UserConsoleView	∦	‖	∦	∦	∦	∦
getInstance	‖	‖	‖	‖	‖	‖
createPartControl	∦	‖	∦	∦	‖	∦
getPrintWriter	∦	‖	∦	∦	‖	∦
clearConsole	∦	‖	‖	‖	‖	‖
setFocus	∦	‖	∦	∦	‖	∦

32 NullPrintWriter

Table 90: Methods Requires Clause Satisfiability

Method	Satisfiability
NullPrintWriter	√
instance	√

Table 91: State Transition Matrix

	alive
alive	↑

Table 92: Methods Concurrency Matrix

	NullPrintWriter	instance
NullPrintWriter	∦	‖
instance	‖	‖

33 Anonymous

Table 93: Methods Requires Clause Satisfiability

Method	Satisfiability
Anonymous	×
write	√

Table 94: State Transition Matrix

	alive
alive	↑

Table 95: Methods Concurrency Matrix

	Anonymous	write
Anonymous	#	‖
write	‖	‖

34 ClearWarningHandler

Table 96: Methods Requires Clause Satisfiability

Method	Satisfiability
ClearWarningHandler	√
execute	√
isEnabled	√
isHandled	√
removeHandlerListener	√
addHandlerListener	√
dispose	√

Table 97: State Transition Matrix

	alive
alive	↑

Table 98: Methods Concurrency Matrix

	ClearWarningHandler	execute	isEnabled	isHandled	removeHandlerListener	addHandlerListener	dispose
ClearWarningHandler	∦	∦	∦	∦	∦	∦	∦
execute	∦	∦	‖	‖	‖	‖	‖
isEnabled	∦	‖	‖	‖	‖	‖	‖
isHandled	∦	‖	‖	‖	‖	‖	‖
removeHandlerListener	∦	‖	‖	‖	‖	‖	‖
addHandlerListener	∦	‖	‖	‖	‖	‖	‖
dispose	∦	‖	‖	‖	‖	‖	‖

35 Box

Table 99: Methods Requires Clause Satisfiability

Method	Satisfiability
Box	√
box	√
getValue	√
setValue	√

Table 100: State Transition Matrix

	alive
alive	↑

Table 101: Methods Concurrency Matrix

	Box	box	getValue	setValue
Box	⫽	⫽	⫽	⫽
box	⫽	‖	‖	‖
getValue	⫽	‖	‖	⫽
setValue	⫽	‖	⫽	⫽

36 DisplayCrystalInfo

Table 102: Methods Requires Clause Satisfiability

Method	Satisfiability
DisplayCrystalInfo	√
run	√
selectionChanged	√
dispose	√
init	√

Table 103: State Transition Matrix

	alive
alive	↑

Table 104: Methods Concurrency Matrix

	DisplayCrystalInfo	run	selectionChanged	dispose	init
DisplayCrystalInfo	∦	∦	∦	∦	∦
run	∦	∦	‖	‖	∦
selectionChanged	∦	‖	‖	‖	‖
dispose	∦	‖	‖	‖	‖
init	∦	∦	‖	‖	∦

37 ShortFormatter

Table 105: Methods Requires Clause Satisfiability

Method	Satisfiability
ShortFormatter	√
format	√

Table 106: State Transition Matrix

	alive
alive	↑

Table 107: Methods Concurrency Matrix

	ShortFormatter	format
ShortFormatter	∦	∦
format	∦	∦

38 Utilities2

Table 108: Methods Requires Clause Satisfiability

Method	Satisfiability
Utilities2	√
ASTNodeToString	√
ModifierToString	√
getMethodDeclaration	√
methodDeclarationToString	√
nyi	√

Table 109: State Transition Matrix

	alive
alive	↑

Table 110: Methods Concurrency Matrix

	Utilities2	ASTNodeToString	ModifierToString	getMethodDeclaration	methodDeclarationToString	nyi
Utilities2	∦	∦	∦	∦	∦	∦
ASTNodeToString	∦	∥	∥	∥	∥	∥
ModifierToString	∦	∥	∥	∥	∥	∥
getMethodDeclaration	∦	∥	∥	∥	∥	∥
methodDeclarationToString	∦	∥	∥	∥	∥	∥
nyi	∦	∥	∥	∥	∥	∥

39 Option

Table 111: Methods Requires Clause Satisfiability

Method	Satisfiability
Option	√
none	√
some	√
wrap	√

Table 112: State Transition Matrix

	alive
alive	↑

Table 113: Methods Concurrency Matrix

	Option	none	some	wrap
Option	∦	∦	∦	∦
none	∦	∥	∥	∥
some	∦	∥	∥	∥
wrap	∦	∥	∥	∥

40 Anonymous

Table 114: Methods Requires Clause Satisfiability

Method	Satisfiability
Anonymous	×
isNone	√
isSome	√
unwrap	√
toString	√

Table 115: State Transition Matrix

	alive
alive	↑

Table 116: Methods Concurrency Matrix

	Anonymous	isNone	isSome	unwrap	toString
Anonymous	∦	‖	‖	‖	‖
isNone	‖	‖	‖	‖	‖
isSome	‖	‖	‖	‖	‖
unwrap	‖	‖	‖	‖	‖
toString	‖	‖	‖	‖	‖

41 AnalysisMenuPopulator

Table 117: Methods Requires Clause Satisfiability

Method	Satisfiability
AnalysisMenuPopulator	√
getContributionItems	√

Table 118: State Transition Matrix

	alive
alive	↑

Table 119: Methods Concurrency Matrix

	AnalysisMenuPopulator	getContributionItems
AnalysisMenuPopulator	∦	∦
getContributionItems	∦	‖

42 CrystalFileAction

Table 120: Methods Requires Clause Satisfiability

Method	Satisfiability
CrystalFileAction	√
setActivePart	√
run	√
selectionChanged	√

Table 121: State Transition Matrix

	alive
alive	↑

Table 122: Methods Concurrency Matrix

	CrystalFileAction	setActivePart	run	selectionChanged
CrystalFileAction	∦	∦	∦	∦
setActivePart	∦	‖	‖	‖
run	∦	‖	∦	∦
selectionChanged	∦	‖	∦	∦

43 Anonymous

Table 123: Methods Requires Clause Satisfiability

Method	Satisfiability
Anonymous	×
run	×

Table 124: State Transition Matrix

	alive
alive	↑

Table 125: Methods Concurrency Matrix

	Anonymous	run
Anonymous	∦	‖
run	‖	‖

44 Freezable

Table 126: Methods Requires Clause Satisfiability

Method	Satisfiability
Freezable	√
mutableCopy	√
freeze	√

Table 127: State Transition Matrix

	alive
alive	↑

Table 128: Methods Concurrency Matrix

	Freezable	mutableCopy	freeze
Freezable	∦	∦	∦
mutableCopy	∦	‖	‖
freeze	∦	‖	‖

45 EnableAnalysisHandler

Table 129: Methods Requires Clause Satisfiability

Method	Satisfiability
EnableAnalysisHandler	√
addHandlerListener	√
dispose	√
execute	√
isEnabled	√
isHandled	√
removeHandlerListener	√
updateElement	√

Table 130: State Transition Matrix

	alive
alive	↑

Table 131: Methods Concurrency Matrix

	EnableAnalysisHandler	addHandlerListener	dispose	execute	isEnabled	isHandled	removeHandlerListener	updateElement
EnableAnalysisHandler	∦	∦	∦	∦	∦	∦	∦	∦
addHandlerListener	∦	‖	‖	‖	‖	‖	‖	‖
dispose	∦	‖	‖	‖	‖	‖	‖	‖
execute	∦	‖	‖	∦	‖	‖	‖	∦
isEnabled	∦	‖	‖	‖	‖	‖	‖	‖
isHandled	∦	‖	‖	‖	‖	‖	‖	‖
removeHandlerListener	∦	‖	‖	‖	‖	‖	‖	‖
updateElement	∦	‖	‖	∦	‖	‖	‖	∦

46 CrystalUIAction

Table 132: Methods Requires Clause Satisfiability

Method	Satisfiability
CrystalUIAction	√
run	√
selectionChanged	√
dispose	√
init	√

Table 133: State Transition Matrix

	alive
alive	↑

Table 134: Methods Concurrency Matrix

	CrystalUIAction	run	selectionChanged	dispose	init
CrystalUIAction	⫽	⫽	⫽	⫽	⫽
run	⫽	‖	‖	‖	‖
selectionChanged	⫽	‖	‖	‖	‖
dispose	⫽	‖	‖	‖	‖
init	⫽	‖	‖	‖	‖

47 Anonymous

Table 135: Methods Requires Clause Satisfiability

Method	Satisfiability
Anonymous	×
run	×

Table 136: State Transition Matrix

	alive
alive	↑

Table 137: Methods Concurrency Matrix

	Anonymous	run
Anonymous	∦	‖
run	‖	‖

48 MethodFindVisitor

Table 138: Methods Requires Clause Satisfiability

Method	Satisfiability
MethodFindVisitor	√
visit	√

Table 139: State Transition Matrix

	alive
alive	↑

Table 140: Methods Concurrency Matrix

	MethodFindVisitor	visit
MethodFindVisitor	∦	∦
visit	∦	∦

49 Anonymous

Table 141: Methods Requires Clause Satisfiability

Method	Satisfiability
Anonymous	×
endVisit	√

Table 142: State Transition Matrix

	alive
alive	↑

Table 143: Methods Concurrency Matrix

	Anonymous	endVisit
Anonymous	#	‖
endVisit	‖	‖

50 BindingsCollectorVisitor

Table 144: Methods Requires Clause Satisfiability

Method	Satisfiability
BindingsCollectorVisitor	√
addNewBinding	√
visit	√

Table 145: State Transition Matrix

	alive
alive	↑

Table 146: Methods Concurrency Matrix

	BindingsCollectorVisitor	addNewBinding	visit
BindingsCollectorVisitor	∦	∦	∦
addNewBinding	∦	∦	∦
visit	∦	∦	∦

51 StudentRuntimeException

Table 147: Methods Requires Clause Satisfiability

Method	Satisfiability
StudentRuntimeException	√

Table 148: State Transition Matrix

	alive
alive	↑

52 Abbreviation

Table 149: 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

53 Annotated Version of Sequential Java Program generated by Sip4j

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

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


10  public void analyzeMethod(MethodDeclaration d) {

12 }

14 }ENDOFCLASS

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

18 class WorklistNodeOrderComparator {
19 @Perm(ensures="unique(this) in alive")
20 WorklistNodeOrderComparator() {   }

22 @Perm(requires="unique(this) in alive",
23 ensures="unique(this) in alive")
24   WorklistNodeOrderComparator createPostOrderAndPopulateNodeMap(final IControlFlowGraph cfg, final Map<
       ASTNode,Set<ICFGNode>> nodeMap, final boolean isForward) {
25  return null;

27 }
28 @Perm(requires="unique(this) in alive",
29 ensures="unique(this) in alive")
30   void registerCfgNode(Map<ASTNode,Set<ICFGNode>> nodeMap, ICFGNode cfgNode) {

32 }
33 @Perm(requires="unique(this) in alive",
34 ensures="unique(this) in alive")
35  public int compare(ICFGNode node1, ICFGNode node2) {
36  return 0;

38 }

40 }ENDOFCLASS

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

44 class BranchInsensitiveWorklist {
45 @Perm(ensures="unique(this) in alive")
46 BranchInsensitiveWorklist() {   }

48 @Perm(requires="immutable(this) in alive",
49 ensures="immutable(this) in alive")
50  protected AnalysisDirection getAnalysisDirection() {
51  return null;

53 }
54 @Perm(requires="immutable(this) in alive",
55 ensures="immutable(this) in alive")
56  protected Lattice<LE> getLattice() {
57  return null;

59 }
60 @Perm(requires="unique(this) in alive",
61 ensures="unique(this) in alive")
62  protected IResult<LE> transferNode(ICFGNode cfgNode, LE incoming, ILabel transferLabel) {
63  return null;

65 }

67 }ENDOFCLASS

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

71 class SingleResult {
72 @Perm(ensures="unique(this) in alive")
73 SingleResult() {   }
```

```
}ENDOFCLASS

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

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

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public AnalysisResult<LE> performAnalysis() {
 return null;

}

  T checkNull(T o) {
 return null;

}

 protected AnalysisResult<LE> createAnalysisResult(Map<ICFGNode,IResult<LE>> labeledResultsBefore, Map<
      ICFGNode,IResult<LE>> labeledResultsAfter, Map<ASTNode,Set<ICFGNode>> nodeMap, Lattice<LE> lattice,
       ICFGNode _startNode, ICFGNode _endNode) {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 protected ILabel incomingLabel(ILabel edgeLabel) {
 return null;

}

}ENDOFCLASS

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

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

@Perm(ensures="none(this) in alive")
 public Map<ASTNode,Set<ICFGNode>> getNodeMap() {
 return null;

}
@Perm(ensures="none(this) in alive")
 public Map<ICFGNode,IResult<LE>> getLabeledResultsAfter() {
 return null;

}
@Perm(ensures="none(this) in alive")
 public Map<ICFGNode,IResult<LE>> getLabeledResultsBefore() {
 return null;

}
@Perm(ensures="none(this) in alive")
 public Lattice<LE> getLattice() {
 return null;

}
@Perm(ensures="none(this) in alive")
 public ICFGNode getCfgStartNode() {
 return null;

}
@Perm(ensures="none(this) in alive")
 public ICFGNode getCfgEndNode() {
 return null;

}

}ENDOFCLASS

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

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

```
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
  NormalLabel getNormalLabel() {
 return null;

}

}ENDOFCLASS

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

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

@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public LE get(ILabel label) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void put(ILabel label, LE result) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public Set<ILabel> keySet() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public IResult<LE> join(IResult<LE> otherResult) {
 return null;

}

}ENDOFCLASS

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

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

@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
  BooleanLabel getBooleanLabel(boolean labelValue) {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 public boolean getBranchValue() {
 return 0;

}

}ENDOFCLASS

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

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

@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 protected AnalysisDirection getAnalysisDirection() {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 protected Lattice<LE> getLattice() {
 return null;

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

```
ensures="unique(this) in alive")
 protected IResult<LE> transferNode(ICFGNode cfgNode, LE incoming, ILabel transferLabel) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 private List<ILabel> getLabels(ICFGNode cfgNode) {
 return null;

}

}ENDOFCLASS

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

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

@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setMonitor(IProgressMonitor monitor) {

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public WorklistTemplate<LE> createBranchInsensitiveWorklist(MethodDeclaration method, ITransferFunction
     <LE> transferFunction) {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public WorklistTemplate<LE> createBranchSensitiveWorklist(MethodDeclaration method,
     IBranchSensitiveTransferFunction<LE> transferFunction) {
 return null;

}

}ENDOFCLASS

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

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


 private IFlowAnalysis<LE> createNewFlowAnalysis(ITransferFunction<LE> transferFunction, Crystal crystal
     ) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public void analyzePreemitively(List<MethodDeclaration> methods) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public IResult<LE> getLabeledResultsAfter(ASTNode node) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public IResult<LE> getLabeledResultsBefore(ASTNode node) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public LE getResultsAfter(ASTNode node) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public LE getResultsBefore(ASTNode node) {
 return null;
```

```
314 }
315 @Perm(requires="share(this) in alive",
316 ensures="share(this) in alive")
317  private IFlowAnalysis<LE> addAsFakeFuture(MethodDeclaration decl, final IFlowAnalysis<LE> fa) {
318  return null;

320 }
321 @Perm(requires="pure(this) in alive",
322 ensures="pure(this) in alive")
323  protected Map<MethodDeclaration,Future<IFlowAnalysis<LE>>> getAnalyzedMethods() {
324  return null;

326 }

328  public LE getEndResults(MethodDeclaration d) {
329  return null;

331 }

333  public IResult<LE> getLabeledEndResult(MethodDeclaration d) {
334  return null;

336 }

338  public IResult<LE> getLabeledStartResult(MethodDeclaration d) {
339  return null;

341 }

343  public LE getStartResults(MethodDeclaration d) {
344  return null;

346 }

348 }ENDOFCLASS

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

352 class FlowAnalysis {
353 @Perm(ensures="unique(this) in alive")
354 FlowAnalysis() {   }


357 }ENDOFCLASS

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

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

365 @Perm(ensures="none(this) in alive")
366  public IFlowAnalysis<LE> call() {
367  return null;

369 }

371  public boolean cancel(boolean mayInterruptIfRunning) {
372  return 0;

374 }

376  public IFlowAnalysis<LE> get() {
377  return null;

379 }

381  public boolean isCancelled() {
382  return 0;

384 }

386  public boolean isDone() {
387  return 0;

389 }

391 }ENDOFCLASS

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

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

@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  MethodDeclaration getMethodDeclaration(ASTNode node) {
 return null;

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

}

  String methodDeclarationToString(MethodDeclaration md) {
 return null;

}

}ENDOFCLASS

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

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

@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 protected IControlFlowGraph getControlFlowGraph() {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
  MethodDeclaration getMethod() {
 return null;

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

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

}

}ENDOFCLASS

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

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


}ENDOFCLASS

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

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

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

}

}ENDOFCLASS

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

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


}ENDOFCLASS

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

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

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

}

 public boolean isEnabled() {
 return 0;

}

 public boolean isHandled() {
 return 0;

}

 public void addHandlerListener(IHandlerListener handlerListener) {

}

 public void removeHandlerListener(IHandlerListener handlerListener) {

}

 public void dispose() {

}

}ENDOFCLASS

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

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

@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
  Crystal getCrystalInstance() {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  Set<String> getEnabledAnalyses() {
 return null;

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

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

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

}

}ENDOFCLASS

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

```
557 class Anonymous {
558 @Perm(ensures="unique(this) in alive")
559 Anonymous() {   }

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

566 }

568 }ENDOFCLASS

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

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


577   List<ICompilationUnit> scanForCompilationUnits() {
578  return null;

580 }

582   List<ICompilationUnit> collectCompilationUnits(IJavaElement javaElement) {
583  return null;

585 }

587   ASTNode getASTNodeFromCompilationUnit(ICompilationUnit compUnit) {
588  return null;

590 }

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

595 }

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

600 }

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

605 }

607   Map<String,ASTNode> scanForBindings(Map<ICompilationUnit,ASTNode> compilationUnitToASTNode) {
608  return null;

610 }

612   List<ICompilationUnit> findCompilationUnits(List<String> files) {
613  return null;

615 }

617   String getWorkspaceRelativeName(IJavaElement element) {
618  return null;

620 }

622   ITypeBinding getDeclNodeFromType(final IType type) {
623  return null;

625 }

627 }ENDOFCLASS

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

631 class Crystal {
632 @Perm(ensures="unique(this) in alive")
633 Crystal() {   }

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

```
636 ensures="share(this) in alive")
637  public void runAnalyses(IRunCrystalCommand command, IProgressMonitor monitor) {

639 }

641  private void runCrystalJob(ICrystalJob job) {

643 }
644 @Perm(requires="share(this) in alive",
645 ensures="share(this) in alive")
646  private ICrystalJob createJobFromCommand(final IRunCrystalCommand command, final IProgressMonitor
         monitor) {
647  return null;

649 }
650 @Perm(requires="pure(this) in alive",
651 ensures="pure(this) in alive")
652  private Option<ICrystalAnalysis> findAnalysisWithName(String analysis_name) {
653  return null;

655 }
656 @Perm(requires="pure(this) in alive",
657 ensures="pure(this) in alive")
658  public List<ICrystalAnalysis> getAnalyses() {
659  return null;

661 }
662 @Perm(requires="share(this) in alive",
663 ensures="share(this) in alive")
664  public void run(final ICompilationUnit cu, final AnnotationDatabase annoDB, final IProgressMonitor
         monitor, final IRunCrystalCommand command, List<ICrystalAnalysis> analyses_to_use) {

666 }

668  private ICrystalJob createCrystalJobFromSingleJobs(final IRunCrystalCommand command, final
         IProgressMonitor monitor, final int num_jobs, final List<ISingleCrystalJob> jobs, final List<
         ICrystalAnalysis> analyses_to_use) {
669  return null;

671 }
672 @Perm(requires="share(this) in alive",
673 ensures="share(this) in alive")
674  public void registerAnalysis(ICrystalAnalysis analysis) {

676 }
677 @Perm(requires="share(this) in alive",
678 ensures="share(this) in alive")
679  public void registerAnnotation(String annotationName, Class<? extends ICrystalAnnotation> annoClass,
         boolean parseAsMeta) {

681 }

683 }ENDOFCLASS

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

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


692  public Set<String> analyses() {
693  return null;

695 }

697  public List<ICompilationUnit> compilationUnits() {
698  return null;

700 }

702  public IAnalysisReporter reporter() {
703  return null;

705 }

707 }ENDOFCLASS

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

711 class ControlFlowGraph {
```

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

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  void addControlFlowNode(ASTNode astNode, ControlFlowNode cfn) {

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

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
  ControlFlowNode getControlFlowNode(ASTNode inNode) {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 public ControlFlowNode getStartNode() {
 return null;

}
@Perm(ensures="none(this) in alive")
 public ControlFlowNode getEndNode() {
 return null;

}

 public String toString() {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Set<ControlFlowNode> getNodeSet() {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 private void buildNodeList(ControlFlowNode cfn, Set<ControlFlowNode> set) {

}

}ENDOFCLASS

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

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

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public ControlFlowNode newControlFlowNode(ASTNode node) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public void moveEdges(Direction direction, ControlFlowNode node) {

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Iterator<ControlFlowNode> getIterator(Direction direction) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 private void removeNode(Direction direction, ControlFlowNode cfn) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 protected void removeEdge(Direction direction, ControlFlowNode node) {
```

```
793  }
794  @Perm(requires="unique(this) in alive",
795  ensures="unique(this) in alive")
796   public void remove() {

798  }
799  @Perm(requires="unique(this) in alive",
800  ensures="unique(this) in alive")
801   public void insertNode(Direction direction, ControlFlowNode insertNode) {

803  }
804  @Perm(requires="unique(this) in alive",
805  ensures="unique(this) in alive")
806   public void addEdge(Direction direction, ControlFlowNode node) {

808  }
809  @Perm(requires="unique(this) in alive",
810  ensures="unique(this) in alive")
811   private void addNode(Direction direction, ControlFlowNode cfn) {

813  }
814  @Perm(requires="unique(this) in alive",
815  ensures="unique(this) in alive")
816   protected void removeEdges(Direction direction) {

818  }
819  @Perm(requires="share(this) in alive",
820  ensures="share(this) in alive")
821   public String toString() {
822   return null;

824  }
825  @Perm(requires="pure(this) in alive",
826  ensures="pure(this) in alive")
827   public void evaluate() {

829  }
830  @Perm(requires="pure(this) in alive",
831  ensures="pure(this) in alive")
832   public ASTNode getASTNode() {
833   return null;

835  }
836  @Perm(requires="pure(this) in alive",
837  ensures="pure(this) in alive")
838   public boolean isDummy() {
839   return 0;

841  }
842  @Perm(requires="share(this) in alive",
843  ensures="share(this) in alive")
844   public void setLoopPaths(ControlFlowNode enter, ControlFlowNode exit) {

846  }
847  @Perm(requires="unique(this) in alive",
848  ensures="unique(this) in alive")
849   public void setFirstChild(ControlFlowNode child) {

851  }
852  @Perm(requires="immutable(this) in alive",
853  ensures="immutable(this) in alive")
854   public ControlFlowGraph getControlFlowGraph() {
855   return null;

857  }
858  @Perm(requires="share(this) in alive",
859  ensures="share(this) in alive")
860   public ControlFlowNode getNode(Direction direction) {
861   return null;

863  }
864  @Perm(requires="share(this) in alive",
865  ensures="share(this) in alive")
866   public ControlFlowNode findNode(Direction direction, int astNodeType) {
867   return null;

869  }
870  @Perm(requires="unique(this) in alive",
871  ensures="unique(this) in alive")
872   public ControlFlowNode returning() {
873   return null;
```

```
875 }
876 @Perm(requires="unique(this) in alive",
877 ensures="unique(this) in alive")
878  public int getNumberOfEdges(Direction direction) {
879  return 0;

881 }
882 @Perm(requires="unique(this) in alive",
883 ensures="unique(this) in alive")
884  public ControlFlowNode breaking(String label, boolean keepRemovingNodes) {
885  return null;

887 }
888 @Perm(requires="unique(this) in alive",
889 ensures="unique(this) in alive")
890  public ControlFlowNode continuing(String label, boolean keepRemovingNodes) {
891  return null;

893 }
894 @Perm(requires="pure(this) in alive",
895 ensures="pure(this) in alive")
896  private List<ControlFlowNode> getEdges(Direction direction) {
897  return null;

899 }
900 @Perm(requires="share(this) in alive",
901 ensures="share(this) in alive")
902   String toStringGraph(ControlFlowNode cfn, int depth, Set<ControlFlowNode> seen, Direction direction) {
903  return null;

905 }
906 @Perm(requires="share(this) in alive",
907 ensures="share(this) in alive")
908  public String toStringGraphOverload(Direction direction) {
909  return null;

911 }

913 }ENDOFCLASS

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

917 class ControlFlowVisitor {
918 @Perm(ensures="unique(this) in alive")
919 ControlFlowVisitor() {   }

921 @Perm(requires="unique(this) in alive",
922 ensures="unique(this) in alive")
923  public boolean visit(ArrayAccess node) {
924  return 0;

926 }
927 @Perm(requires="pure(this) in alive",
928 ensures="pure(this) in alive")
929  public void performVisit() {

931 }
932 @Perm(requires="unique(this) in alive",
933 ensures="unique(this) in alive")
934  protected List<ControlFlowNode> createCFNListFromASTNodeList(List nodes) {
935  return null;

937 }

939  public void preVisit(ASTNode node) {

941 }
942 @Perm(requires="share(this) in alive",
943 ensures="share(this) in alive")
944  protected void evaluate(List<ControlFlowNode> list) {

946 }

948 }ENDOFCLASS

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

952 class Direction {
953 @Perm(ensures="unique(this) in alive")
954 Direction() {   }
```

```
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 public Direction changeDirection() {
 return null;

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

}

}ENDOFCLASS

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

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


}ENDOFCLASS

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

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

@Perm(ensures="none(this) in alive")
  UserConsoleView getInstance() {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public void createPartControl(Composite parent) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public PrintWriter getPrintWriter() {
 return null;

}

 public void clearConsole() {

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

}

}ENDOFCLASS

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

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

@Perm(ensures="none(this) in alive")
  NullPrintWriter instance() {
 return null;

}

}ENDOFCLASS

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

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


 public void write(int b) {
```

```
1037 }

1039 }ENDOFCLASS

1041 @ClassStates({@State(name = "alive")})

1043 class ClearWarningHandler {
1044 @Perm(ensures="unique(this) in alive")
1045 ClearWarningHandler() {   }

1047 @Perm(requires="unique(this) in alive",
1048 ensures="unique(this) in alive")
1049  public Object execute(ExecutionEvent event) {
1050  return null;

1052 }

1054  public boolean isEnabled() {
1055  return 0;

1057 }

1059  public boolean isHandled() {
1060  return 0;

1062 }

1064  public void removeHandlerListener(IHandlerListener handlerListener) {

1066 }

1068  public void addHandlerListener(IHandlerListener handlerListener) {

1070 }

1072  public void dispose() {

1074 }

1076 }ENDOFCLASS

1078 @ClassStates({@State(name = "alive")})

1080 class Box {
1081 @Perm(ensures="unique(this) in alive")
1082 Box() {   }


1085   Box<T> box(T t) {
1086  return null;

1088 }
1089 @Perm(requires="pure(this) in alive",
1090 ensures="pure(this) in alive")
1091  public T getValue() {
1092  return null;

1094 }
1095 @Perm(requires="full(this) in alive",
1096 ensures="full(this) in alive")
1097  public void setValue(T t) {

1099 }

1101 }ENDOFCLASS

1103 @ClassStates({@State(name = "alive")})

1105 class DisplayCrystalInfo {
1106 @Perm(ensures="unique(this) in alive")
1107 DisplayCrystalInfo() {   }

1109 @Perm(requires="share(this) in alive",
1110 ensures="share(this) in alive")
1111  public void run(IAction action) {

1113 }

1115  public void selectionChanged(IAction action, ISelection selection) {
```

```
1117 }

1119  public void dispose() {

1121 }
1122 @Perm(requires="share(this) in alive",
1123 ensures="share(this) in alive")
1124  public void init(IWorkbenchWindow window) {

1126 }

1128 }ENDOFCLASS

1130 @ClassStates({@State(name = "alive")})

1132 class ShortFormatter {
1133 @Perm(ensures="unique(this) in alive")
1134 ShortFormatter() {   }

1136 @Perm(requires="unique(this) in alive",
1137 ensures="unique(this) in alive")
1138  public String format(LogRecord record) {
1139  return null;

1141 }

1143 }ENDOFCLASS

1145 @ClassStates({@State(name = "alive")})

1147 class Utilities2 {
1148 @Perm(ensures="unique(this) in alive")
1149 Utilities2() {   }


1152   String ASTNodeToString(ASTNode node) {
1153  return null;

1155 }

1157   String ModifierToString(int modifier) {
1158  return null;

1160 }

1162   MethodDeclaration getMethodDeclaration(ASTNode node) {
1163  return null;

1165 }

1167   String methodDeclarationToString(MethodDeclaration md) {
1168  return null;

1170 }

1172   T nyi() {
1173  return null;

1175 }

1177 }ENDOFCLASS

1179 @ClassStates({@State(name = "alive")})

1181 class Option {
1182 @Perm(ensures="unique(this) in alive")
1183 Option() {   }

1185 @Perm(requires="immutable(this) in alive",
1186 ensures="immutable(this) in alive")
1187   Option<T> none() {
1188  return null;

1190 }

1192   Option<T> some(final T t) {
1193  return null;

1195 }

1197   Option<T> wrap(final T t) {
```

```
 return null;

}

}ENDOFCLASS

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

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


 public boolean isNone() {
 return 0;

}

 public boolean isSome() {
 return 0;

}

 public Object unwrap() {
 return null;

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

}

}ENDOFCLASS

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

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

@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 protected IContributionItem[] getContributionItems() {
 return null;

}

}ENDOFCLASS

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

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


 public void setActivePart(IAction action, IWorkbenchPart targetPart) {

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

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

}

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


 public void mutableCopy() {

}

 public T freeze() {
 return null;

}

}ENDOFCLASS

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

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


 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;

}

 public boolean isEnabled() {
 return 0;

}

 public boolean isHandled() {
 return 0;

}

 public void removeHandlerListener(IHandlerListener handlerListener) {

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public void updateElement(UIElement element, Map parameters) {

}

}ENDOFCLASS

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

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

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

}

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

```
}

 public void dispose() {

}

 public void init(IWorkbenchWindow window) {

}

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


 public void endVisit(TypeDeclaration node) {

}

}ENDOFCLASS

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

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

@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 protected void addNewBinding(IBinding binding, ASTNode node) {

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

}

}ENDOFCLASS

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

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

```
1443 }ENDOFCLASS
```