

Summary

Sink States:0(0×10^0)

Table 1: Pulse Analysis Summary

Classes	Methods	States	Unsatisfiable Clauses	Unreachable States	Possible concurrent Methods	Total. no. of pairs	No. of concurrent pairs	Percentage of concurrent Methods
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	5
2	Abbreviation	6
3	Annotated Version of Sequential Java Program generated by Sip4j	7

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	✓
getAnalysisDirection	✓
getLattice	✓
transferNode	✓

Table 9: State Transition Matrix

	alive
alive	↑

Table 10: Methods Concurrency Matrix

	BranchInsensitiveWorklist	getAnalysisDirection	getLattice	transferNode
BranchInsensitiveWorklist	⌘	⌘	⌘	⌘
getAnalysisDirection	⌘			⌘
getLattice	⌘			⌘
transferNode	⌘	⌘	⌘	⌘

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	✓
performAnalysis	✓
checkNull	✓
createAnalysisResult	✓
incomingLabel	✓

Table 14: State Transition Matrix

	alive
alive	↑

Table 15: Methods Concurrency Matrix

	WorklistTemplate	performAnalysis	checkNull	createAnalysisResult	incomingLabel
WorklistTemplate	⌘	⌘	⌘	⌘	⌘
performAnalysis	⌘	⌘			⌘
checkNull	⌘				
createAnalysisResult	⌘				
incomingLabel	⌘	⌘			

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	✓
getNormalLabel	✓

Table 20: State Transition Matrix

	alive
alive	↑

Table 21: Methods Concurrency Matrix

	NormalLabel	getNormalLabel
NormalLabel	⌘	⌘
getNormalLabel	⌘	

8 IncomingResult

Table 22: Methods Requires Clause Satisfiability

Method	Satisfiability
IncomingResult	✓
get	✓
put	✓
keySet	✓
join	✓

Table 23: State Transition Matrix

	alive
alive	↑

Table 24: Methods Concurrency Matrix

	IncomingResult	get	put	keySet	join
IncomingResult	⌘	⌘	⌘	⌘	⌘
get	⌘		⌘		⌘
put	⌘	⌘	⌘	⌘	⌘
keySet	⌘		⌘		⌘
join	⌘	⌘	⌘	⌘	⌘

9 BooleanLabel

Table 25: Methods Requires Clause Satisfiability

Method	Satisfiability
BooleanLabel	✓
getBooleanLabel	✓
getBranchValue	✓

Table 26: State Transition Matrix

	alive
alive	↑

Table 27: Methods Concurrency Matrix

	BooleanLabel	getBooleanLabel	getBranchValue
BooleanLabel	⧻	⧻	⧻
getBooleanLabel	⧻		
getBranchValue	⧻		

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	✓
analyzePreemptively	✓
getLabeledResultsAfter	✓
getLabeledResultsBefore	✓
getResultsAfter	✓
getResultsBefore	✓
addAsFakeFuture	✓
getAnalyzedMethods	✓
getEndResults	✓
getLabeledEndResult	✓
getLabeledStartResult	✓
getStartResults	✓

Table 35: State Transition Matrix

	alive
alive	↑

Table 36: Methods Concurrency Matrix

	ConcurrentFlowAnalysis	createNewFlowAnalysis	analyzePreemptively	getLabeledResultsAfter	getLabeledResultsBefore	getResultsAfter	getResultsBefore	addAsFakeFuture	getAnalyzedMethods	getEndResults	getLabeledEndResult	getLabeledStartResult	getStartResults
ConcurrentFlowAnalysis	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
createNewFlowAnalysis	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
analyzePreemptively	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
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	✓
getCrystalInstance	✓
getEnabledAnalyses	✓
enableAnalysis	✓
disableAnalysis	✓
start	✓

Table 59: State Transition Matrix

	alive
alive	↑

Table 60: Methods Concurrency Matrix

	AbstractCrystalPlugin	getCrystalInstance	getEnabledAnalyses	enableAnalysis	disableAnalysis	start
AbstractCrystalPlugin	⌘	⌘	⌘	⌘	⌘	⌘
getCrystalInstance	⌘		⌘	⌘	⌘	⌘
getEnabledAnalyses	⌘	⌘	⌘	⌘	⌘	⌘
enableAnalysis	⌘	⌘	⌘	⌘	⌘	⌘
disableAnalysis	⌘	⌘	⌘	⌘	⌘	⌘
start	⌘	⌘	⌘	⌘	⌘	⌘

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
✓	requires clause of the method is satisfiable
✗	requires clause of the method is unsatisfiable
↑	The row-state can be transitioned to the column-state
✕	The row-state cannot be transitioned to the column-state
	The row-method can be possibly executed parallel with the column-method
⋈	The row-method cannot be executed parallel with the column-method

53 Annotated Version of Sequential Java Program generated by Sip4j

```
1 package outputs;
2 import edu.cmu.cs.plural.annot.*;
3
4 @ClassStates({@State(name = "alive")})
5 class MethodFlowAnalysis {
6   @Perm(ensures="unique(this) in alive")
7   MethodFlowAnalysis() { }
8
9
10  public void analyzeMethod(MethodDeclaration d) {
11
12  }
13
14 }ENDOFCLASS
15
16 @ClassStates({@State(name = "alive")})
17
18 class WorklistNodeOrderComparator {
19   @Perm(ensures="unique(this) in alive")
20   WorklistNodeOrderComparator() { }
21
22   @Perm(requires="unique(this) in alive",
23   ensures="unique(this) in alive")
24   WorklistNodeOrderComparator createPostOrderAndPopulateNodeMap(final IControlFlowGraph cfg, final Map<
25     ASTNode,Set<ICFGNode>> nodeMap, final boolean isForward) {
26     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) {
31
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;
37   }
38 }
39
40 }ENDOFCLASS
41
42 @ClassStates({@State(name = "alive")})
43
44 class BranchInsensitiveWorklist {
45   @Perm(ensures="unique(this) in alive")
46   BranchInsensitiveWorklist() { }
47
48   @Perm(requires="immutable(this) in alive",
49   ensures="immutable(this) in alive")
50   protected AnalysisDirection getAnalysisDirection() {
51     return null;
52   }
53
54   @Perm(requires="immutable(this) in alive",
55   ensures="immutable(this) in alive")
56   protected Lattice<LE> getLattice() {
57     return null;
58   }
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;
64   }
65 }
66
67 }ENDOFCLASS
68
69 @ClassStates({@State(name = "alive")})
70
71 class SingleResult {
72   @Perm(ensures="unique(this) in alive")
73   SingleResult() { }
```

```

76 }ENDOFCLASS
77 @ClassStates({@State(name = "alive")})
78
79 class WorklistTemplate {
80   @Perm(ensures="unique(this) in alive")
81   WorklistTemplate() { }
82
83   @Perm(requires="unique(this) in alive",
84         ensures="unique(this) in alive")
85   public AnalysisResult<LE> performAnalysis() {
86     return null;
87   }
88 }
89
90 T checkNull(T o) {
91   return null;
92 }
93
94
95 protected AnalysisResult<LE> createAnalysisResult(Map<ICFGNode,IResult<LE>> labeledResultsBefore, Map<
96   ICFGNode,IResult<LE>> labeledResultsAfter, Map<ASTNode,Set<ICFGNode>> nodeMap, Lattice<LE> lattice,
97   ICFGNode _startNode, ICFGNode _endNode) {
98   return null;
99 }
100 @Perm(requires="immutable(this) in alive",
101       ensures="immutable(this) in alive")
102 protected ILabel incomingLabel(ILabel edgeLabel) {
103   return null;
104 }
105 }
106 }ENDOFCLASS
107
108 @ClassStates({@State(name = "alive")})
109
110 class AnalysisResult {
111   @Perm(ensures="unique(this) in alive")
112   AnalysisResult() { }
113
114   @Perm(ensures="none(this) in alive")
115   public Map<ASTNode,Set<ICFGNode>> getNodeMap() {
116     return null;
117   }
118 }
119
120 @Perm(ensures="none(this) in alive")
121 public Map<ICFGNode,IResult<LE>> getLabeledResultsAfter() {
122   return null;
123 }
124
125 @Perm(ensures="none(this) in alive")
126 public Map<ICFGNode,IResult<LE>> getLabeledResultsBefore() {
127   return null;
128 }
129 }
130 @Perm(ensures="none(this) in alive")
131 public Lattice<LE> getLattice() {
132   return null;
133 }
134
135 @Perm(ensures="none(this) in alive")
136 public ICFGNode getCfgStartNode() {
137   return null;
138 }
139 }
140 @Perm(ensures="none(this) in alive")
141 public ICFGNode getCfgEndNode() {
142   return null;
143 }
144 }
145 }ENDOFCLASS
146
147 @ClassStates({@State(name = "alive")})
148
149 class NormalLabel {
150   @Perm(ensures="unique(this) in alive")
151   NormalLabel() { }

```

```

154 @Perm(requires="immutable(this) in alive",
155 ensures="immutable(this) in alive")
156 NormalLabel getNormalLabel() {
157     return null;
158 }
159 }
160 }ENDOFCLASS
161
162 @ClassStates({@State(name = "alive")})
163
164 class IncomingResult {
165     @Perm(ensures="unique(this) in alive")
166     IncomingResult() { }
167
168     @Perm(requires="pure(this) in alive",
169     ensures="pure(this) in alive")
170     public LE get(ILabel label) {
171         return null;
172     }
173 }
174
175 @Perm(requires="share(this) in alive",
176 ensures="share(this) in alive")
177 public void put(ILabel label, LE result) {
178 }
179
180 @Perm(requires="pure(this) in alive",
181 ensures="pure(this) in alive")
182 public Set<ILabel> keySet() {
183     return null;
184 }
185
186 @Perm(requires="share(this) in alive",
187 ensures="share(this) in alive")
188 public IResult<LE> join(IResult<LE> otherResult) {
189     return null;
190 }
191 }
192 }ENDOFCLASS
193
194 @ClassStates({@State(name = "alive")})
195
196 class BooleanLabel {
197     @Perm(ensures="unique(this) in alive")
198     BooleanLabel() { }
199
200     @Perm(requires="immutable(this) in alive",
201     ensures="immutable(this) in alive")
202     BooleanLabel getBooleanLabel(boolean labelValue) {
203         return null;
204     }
205 }
206
207 @Perm(requires="immutable(this) in alive",
208 ensures="immutable(this) in alive")
209 public boolean getBranchValue() {
210     return 0;
211 }
212 }
213 }ENDOFCLASS
214
215 @ClassStates({@State(name = "alive")})
216
217 class BranchSensitiveWorklist {
218     @Perm(ensures="unique(this) in alive")
219     BranchSensitiveWorklist() { }
220
221     @Perm(requires="immutable(this) in alive",
222     ensures="immutable(this) in alive")
223     protected AnalysisDirection getAnalysisDirection() {
224         return null;
225     }
226 }
227
228 @Perm(requires="immutable(this) in alive",
229 ensures="immutable(this) in alive")
230 protected Lattice<LE> getLattice() {
231     return null;
232 }
233 }
234 @Perm(requires="unique(this) in alive",

```

```

235 ensures="unique(this) in alive")
236 protected IResult<LE> transferNode(ICFGNode cfgNode, LE incoming, ILabel transferLabel) {
237     return null;
238 }
239
240 @Perm(requires="unique(this) in alive",
241 ensures="unique(this) in alive")
242 private List<ILabel> getLabels(ICFGNode cfgNode) {
243     return null;
244 }
245
246 }ENDOFCLASS
247
248 @ClassStates({@State(name = "alive")})
249
250 class WorklistFactory {
251     @Perm(ensures="unique(this) in alive")
252     WorklistFactory() { }
253
254     @Perm(requires="full(this) in alive",
255 ensures="full(this) in alive")
256     public void setMonitor(IProgressMonitor monitor) {
257
258     }
259
260     @Perm(requires="pure(this) in alive",
261 ensures="pure(this) in alive")
262     public WorklistTemplate<LE> createBranchInsensitiveWorklist(MethodDeclaration method, ITransferFunction
263 <LE> transferFunction) {
264         return null;
265     }
266
267     @Perm(requires="pure(this) in alive",
268 ensures="pure(this) in alive")
269     public WorklistTemplate<LE> createBranchSensitiveWorklist(MethodDeclaration method,
270 IBranchSensitiveTransferFunction<LE> transferFunction) {
271         return null;
272     }
273 }ENDOFCLASS
274
275 @ClassStates({@State(name = "alive")})
276
277 class ConcurrentFlowAnalysis {
278     @Perm(ensures="unique(this) in alive")
279     ConcurrentFlowAnalysis() { }
280
281     private IFlowAnalysis<LE> createNewFlowAnalysis(ITransferFunction<LE> transferFunction, Crystal crystal
282 ) {
283         return null;
284     }
285
286     @Perm(requires="unique(this) in alive",
287 ensures="unique(this) in alive")
288     public void analyzePreemptively(List<MethodDeclaration> methods) {
289
290     }
291
292     @Perm(requires="share(this) in alive",
293 ensures="share(this) in alive")
294     public IResult<LE> getLabeledResultsAfter(ASTNode node) {
295         return null;
296     }
297
298     @Perm(requires="share(this) in alive",
299 ensures="share(this) in alive")
300     public IResult<LE> getLabeledResultsBefore(ASTNode node) {
301         return null;
302     }
303
304     @Perm(requires="share(this) in alive",
305 ensures="share(this) in alive")
306     public LE getResultsAfter(ASTNode node) {
307         return null;
308     }
309
310     @Perm(requires="share(this) in alive",
311 ensures="share(this) in alive")
312     public LE getResultsBefore(ASTNode node) {
313         return null;
314     }

```

```

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;
319 }
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;
325 }
326 }
327
328 public LE getEndResults(MethodDeclaration d) {
329     return null;
330 }
331 }
332
333 public IResult<LE> getLabeledEndResult(MethodDeclaration d) {
334     return null;
335 }
336 }
337
338 public IResult<LE> getLabeledStartResult(MethodDeclaration d) {
339     return null;
340 }
341 }
342
343 public LE getStartResults(MethodDeclaration d) {
344     return null;
345 }
346 }
347
348 }ENDOFCLASS
349
350 @ClassStates({@State(name = "alive")})
351
352 class FlowAnalysis {
353     @Perm(ensures="unique(this) in alive")
354     FlowAnalysis() { }
355 }
356
357 }ENDOFCLASS
358
359 @ClassStates({@State(name = "alive")})
360
361 class Anonymous {
362     @Perm(ensures="unique(this) in alive")
363     Anonymous() { }
364 }
365
366 @Perm(ensures="none(this) in alive")
367 public IFlowAnalysis<LE> call() {
368     return null;
369 }
370 }
371
372 public boolean cancel(boolean mayInterruptIfRunning) {
373     return 0;
374 }
375 }
376
377 public IFlowAnalysis<LE> get() {
378     return null;
379 }
380 }
381
382 public boolean isCancelled() {
383     return 0;
384 }
385 }
386
387 public boolean isDone() {
388     return 0;
389 }
390 }
391 }ENDOFCLASS
392
393 @ClassStates({@State(name = "alive")})

```

```

395 class Utilities {
396   @Perm(ensures="unique(this) in alive")
397   Utilities() { }

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

404   }
405   @Perm(requires="unique(this) in alive",
406         ensures="unique(this) in alive")
407   void main(String[] args) {

409   }

411   String methodDeclarationToString(MethodDeclaration md) {
412     return null;

414   }

416 }ENDOFCLASS

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

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

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

429   }
430   @Perm(requires="immutable(this) in alive",
431         ensures="immutable(this) in alive")
432   MethodDeclaration getMethod() {
433     return null;

435   }
436   @Perm(requires="unique(this) in alive",
437         ensures="unique(this) in alive")
438   boolean checkBreakpoint(ASTNode node) {
439     return 0;

441   }
442   @Perm(requires="unique(this) in alive",
443         ensures="unique(this) in alive")
444   void checkCancel() {

446   }

448 }ENDOFCLASS

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

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

456 }ENDOFCLASS

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

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

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

468   }

470 }ENDOFCLASS

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

```

```

475 class ExceptionMap {
476   @Perm(ensures="unique(this) in alive")
477   ExceptionMap() { }

480 }ENDOFCLASS

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

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

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

493   }

495   public boolean isEnabled() {
496     return 0;

498   }

500   public boolean isHandled() {
501     return 0;

503   }

505   public void addHandlerListener(IHandlerListener handlerListener) {

507   }

509   public void removeHandlerListener(IHandlerListener handlerListener) {

511   }

513   public void dispose() {

515   }

517 }ENDOFCLASS

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

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

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

530   }

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

536   }

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

541   }

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

546   }

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

551   }

553 }ENDOFCLASS

555 @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>
598     compilationUnitToASTNode) {
599     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) {
638
639 }
640
641 private void runCrystalJob(ICrystalJob job) {
642
643 }
644 @Perm(requires="share(this) in alive",
645 ensures="share(this) in alive")
646 private ICrystalJob createJobFromCommand(final IRunCrystalCommand command, final IProgressMonitor
647     monitor) {
648     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;
654 }
655
656 @Perm(requires="pure(this) in alive",
657 ensures="pure(this) in alive")
658 public List<ICrystalAnalysis> getAnalyses() {
659     return null;
660 }
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
665     monitor, final IRunCrystalCommand command, List<ICrystalAnalysis> analyses_to_use) {
666
667 }
668
669 private ICrystalJob createCrystalJobFromSingleJobs(final IRunCrystalCommand command, final
670     IProgressMonitor monitor, final int num_jobs, final List<ISingleCrystalJob> jobs, final List<
671     ICrystalAnalysis> analyses_to_use) {
672     return null;
673 }
674
675 @Perm(requires="share(this) in alive",
676 ensures="share(this) in alive")
677 public void registerAnalysis(ICrystalAnalysis analysis) {
678
679 }
680
681 @Perm(requires="share(this) in alive",
682 ensures="share(this) in alive")
683 public void registerAnnotation(String annotationName, Class<? extends ICrystalAnnotation> annoClass,
684     boolean parseAsMeta) {
685
686 }
687
688 }ENDOFCLASS
689
690 @ClassStates({@State(name = "alive")})
691
692 class Anonymous {
693     @Perm(ensures="unique(this) in alive")
694     Anonymous() { }
695
696
697     public Set<String> analyses() {
698         return null;
699     }
700
701
702     public List<ICompilationUnit> compilationUnits() {
703         return null;
704     }
705
706
707     public IAnalysisReporter reporter() {
708         return null;
709     }
710
711 }
712
713 }ENDOFCLASS
714
715 @ClassStates({@State(name = "alive")})
716
717 class ControlFlowGraph {

```

```

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

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

719 }
720 @Perm(requires="share(this) in alive",
721 ensures="share(this) in alive")
722 void removeControlFlowNode(ASTNode astNode) {

724 }
725 @Perm(requires="unique(this) in alive",
726 ensures="unique(this) in alive")
727 ControlFlowNode getControlFlowNode(ASTNode inNode) {
728 return null;

730 }
731 @Perm(requires="immutable(this) in alive",
732 ensures="immutable(this) in alive")
733 public ControlFlowNode getStartNode() {
734 return null;

736 }
737 @Perm(ensures="none(this) in alive")
738 public ControlFlowNode getEndNode() {
739 return null;

741 }

743 public String toString() {
744 return null;

746 }
747 @Perm(requires="unique(this) in alive",
748 ensures="unique(this) in alive")
749 public Set<ControlFlowNode> getNodeSet() {
750 return null;

752 }
753 @Perm(requires="unique(this) in alive",
754 ensures="unique(this) in alive")
755 private void buildNodeList(ControlFlowNode cfn, Set<ControlFlowNode> set) {

757 }

759 }ENDOFCLASS

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

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

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

772 }
773 @Perm(requires="unique(this) in alive",
774 ensures="unique(this) in alive")
775 public void moveEdges(Direction direction, ControlFlowNode node) {

777 }
778 @Perm(requires="unique(this) in alive",
779 ensures="unique(this) in alive")
780 public Iterator<ControlFlowNode> getIterator(Direction direction) {
781 return null;

783 }
784 @Perm(requires="share(this) in alive",
785 ensures="share(this) in alive")
786 private void removeNode(Direction direction, ControlFlowNode cfn) {

788 }
789 @Perm(requires="share(this) in alive",
790 ensures="share(this) in alive")
791 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() {
797
798 }
799 @Perm(requires="unique(this) in alive",
800 ensures="unique(this) in alive")
801 public void insertNode(Direction direction, ControlFlowNode insertNode) {
802
803 }
804 @Perm(requires="unique(this) in alive",
805 ensures="unique(this) in alive")
806 public void addEdge(Direction direction, ControlFlowNode node) {
807
808 }
809 @Perm(requires="unique(this) in alive",
810 ensures="unique(this) in alive")
811 private void addNode(Direction direction, ControlFlowNode cfn) {
812
813 }
814 @Perm(requires="unique(this) in alive",
815 ensures="unique(this) in alive")
816 protected void removeEdges(Direction direction) {
817
818 }
819 @Perm(requires="share(this) in alive",
820 ensures="share(this) in alive")
821 public String toString() {
822     return null;
823 }
824
825 @Perm(requires="pure(this) in alive",
826 ensures="pure(this) in alive")
827 public void evaluate() {
828
829 }
830 @Perm(requires="pure(this) in alive",
831 ensures="pure(this) in alive")
832 public ASTNode getASTNode() {
833     return null;
834 }
835
836 @Perm(requires="pure(this) in alive",
837 ensures="pure(this) in alive")
838 public boolean isDummy() {
839     return 0;
840 }
841
842 @Perm(requires="share(this) in alive",
843 ensures="share(this) in alive")
844 public void setLoopPaths(ControlFlowNode enter, ControlFlowNode exit) {
845
846 }
847 @Perm(requires="unique(this) in alive",
848 ensures="unique(this) in alive")
849 public void setFirstChild(ControlFlowNode child) {
850
851 }
852 @Perm(requires="immutable(this) in alive",
853 ensures="immutable(this) in alive")
854 public ControlFlowGraph getControlFlowGraph() {
855     return null;
856 }
857
858 @Perm(requires="share(this) in alive",
859 ensures="share(this) in alive")
860 public ControlFlowNode getNode(Direction direction) {
861     return null;
862 }
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;
868 }
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;
880 }
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;
886 }
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;
892 }
893 }
894 @Perm(requires="pure(this) in alive",
895 ensures="pure(this) in alive")
896 private List<ControlFlowNode> getEdges(Direction direction) {
897     return null;
898 }
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;
904 }
905 }
906 @Perm(requires="share(this) in alive",
907 ensures="share(this) in alive")
908 public String toStringGraphOverload(Direction direction) {
909     return null;
910 }
911 }
912 }ENDOFCLASS
913
914 @ClassStates({@State(name = "alive")})
915
916 class ControlFlowVisitor {
917     @Perm(ensures="unique(this) in alive")
918     ControlFlowVisitor() { }
919
920     @Perm(requires="unique(this) in alive",
921     ensures="unique(this) in alive")
922     public boolean visit(ArrayAccess node) {
923         return 0;
924     }
925 }
926 }
927 @Perm(requires="pure(this) in alive",
928 ensures="pure(this) in alive")
929 public void performVisit() {
930 }
931 }
932 @Perm(requires="unique(this) in alive",
933 ensures="unique(this) in alive")
934 protected List<ControlFlowNode> createCFNListFromASTNodeList(List nodes) {
935     return null;
936 }
937 }
938
939 public void preVisit(ASTNode node) {
940 }
941 }
942 @Perm(requires="share(this) in alive",
943 ensures="share(this) in alive")
944 protected void evaluate(List<ControlFlowNode> list) {
945 }
946 }
947 }ENDOFCLASS
948
949 @ClassStates({@State(name = "alive")})
950
951 class Direction {
952     @Perm(ensures="unique(this) in alive")
953     Direction() { }
954 }

```

```

956 @Perm(requires="immutable(this) in alive",
957 ensures="immutable(this) in alive")
958 public Direction changeDirection() {
959     return null;
960 }
961 }
962 @Perm(requires="immutable(this) in alive",
963 ensures="immutable(this) in alive")
964 public String toString() {
965     return null;
966 }
967 }
968 }ENDOFCLASS
969
970 @ClassStates({@State(name = "alive")})
971
972 class CrystalRuntimeException {
973 @Perm(ensures="unique(this) in alive")
974 CrystalRuntimeException() { }
975 }
976 }ENDOFCLASS
977
978 @ClassStates({@State(name = "alive")})
979
980 class UserConsoleView {
981 @Perm(ensures="unique(this) in alive")
982 UserConsoleView() { }
983
984 @Perm(ensures="none(this) in alive")
985 UserConsoleView getInstance() {
986     return null;
987 }
988 }
989 @Perm(requires="unique(this) in alive",
990 ensures="unique(this) in alive")
991 public void createPartControl(Composite parent) {
992 }
993 }
994 @Perm(requires="share(this) in alive",
995 ensures="share(this) in alive")
996 public PrintWriter getPrintWriter() {
997     return null;
998 }
999 }
1000 }
1001
1002 public void clearConsole() {
1003 }
1004 }
1005 @Perm(requires="share(this) in alive",
1006 ensures="share(this) in alive")
1007 public void setFocus() {
1008 }
1009 }
1010 }ENDOFCLASS
1011
1012 @ClassStates({@State(name = "alive")})
1013
1014 class NullPrintWriter {
1015 @Perm(ensures="unique(this) in alive")
1016 NullPrintWriter() { }
1017
1018 @Perm(ensures="none(this) in alive")
1019 NullPrintWriter instance() {
1020     return null;
1021 }
1022 }
1023 }ENDOFCLASS
1024
1025 @ClassStates({@State(name = "alive")})
1026
1027 class Anonymous {
1028 @Perm(ensures="unique(this) in alive")
1029 Anonymous() { }
1030
1031 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() { }
1151 String ASTNodeToString(ASTNode node) {
1152 return null;
1153 }
1155 String ModifierToString(int modifier) {
1156 return null;
1157 }
1159 MethodDeclaration getMethodDeclaration(ASTNode node) {
1160 return null;
1161 }
1163 String methodDeclarationToString(MethodDeclaration md) {
1164 return null;
1165 }
1167 T nyi() {
1168 return null;
1169 }
1171 }ENDOFCLASS
1173 @ClassStates({@State(name = "alive")})
1175 class Option {
1176 @Perm(ensures="unique(this) in alive")
1177 Option() { }
1179 @Perm(requires="immutable(this) in alive",
1180 ensures="immutable(this) in alive")
1181 Option<T> none() {
1182 return null;
1183 }
1185 Option<T> some(final T t) {
1186 return null;
1187 }
1189 Option<T> wrap(final T t) {

```

```

1198     return null;
1200 }
1202 }ENDOFCLASS
1204 @ClassStates({@State(name = "alive")})
1206 class Anonymous {
1207     @Perm(ensures="unique(this) in alive")
1208     Anonymous() { }
1209
1211     public boolean isNone() {
1212         return 0;
1213     }
1214 }
1216     public boolean isSome() {
1217         return 0;
1218     }
1219 }
1221     public Object unwrap() {
1222         return null;
1223     }
1224 }
1225 @Perm(requires="immutable(this) in alive",
1226     ensures="immutable(this) in alive")
1227     public String toString() {
1228         return null;
1229     }
1230 }
1232 }ENDOFCLASS
1234 @ClassStates({@State(name = "alive")})
1236 class AnalysisMenuPopulator {
1237     @Perm(ensures="unique(this) in alive")
1238     AnalysisMenuPopulator() { }
1239
1240     @Perm(requires="pure(this) in alive",
1241         ensures="pure(this) in alive")
1242     protected IContributionItem[] getContributionItems() {
1243         return null;
1244     }
1245 }
1247 }ENDOFCLASS
1249 @ClassStates({@State(name = "alive")})
1251 class CrystalFileAction {
1252     @Perm(ensures="unique(this) in alive")
1253     CrystalFileAction() { }
1254
1256     public void setActivePart(IAction action, IWorkbenchPart targetPart) {
1257     }
1258 }
1259 @Perm(requires="share(this) in alive",
1260     ensures="share(this) in alive")
1261     public void run(IAction action) {
1262     }
1263 }
1264 @Perm(requires="share(this) in alive",
1265     ensures="share(this) in alive")
1266     public void selectionChanged(IAction action, ISelection selection) {
1267     }
1268 }
1270 }ENDOFCLASS
1272 @ClassStates({@State(name = "alive")})
1274 class Anonymous {
1275     @Perm(ensures="unique(this) in alive")
1276     Anonymous() { }
1277
1278     @Perm(requires="unique(this) in alive",

```



```

1279 ensures="unique(this) in alive")
1280 protected IStatus run(IProgressMonitor monitor) {
1281     return null;
1282 }
1283 }
1284 }ENDOFCLASS
1285
1286 @ClassStates({@State(name = "alive")})
1287
1288 class Freezable {
1289     @Perm(ensures="unique(this) in alive")
1290     Freezable() { }
1291
1292     public void mutableCopy() {
1293     }
1294
1295     public T freeze() {
1296         return null;
1297     }
1298 }
1299 }ENDOFCLASS
1300
1301 @ClassStates({@State(name = "alive")})
1302
1303 class EnableAnalysisHandler {
1304     @Perm(ensures="unique(this) in alive")
1305     EnableAnalysisHandler() { }
1306
1307     public void addHandlerListener(IHandlerListener handlerListener) {
1308     }
1309
1310     public void dispose() {
1311     }
1312
1313     @Perm(requires="share(this) in alive",
1314           ensures="share(this) in alive")
1315     public Object execute(ExecutionEvent event) {
1316         return null;
1317     }
1318
1319     public boolean isEnabled() {
1320         return 0;
1321     }
1322
1323     public boolean isHandled() {
1324         return 0;
1325     }
1326
1327     public void removeHandlerListener(IHandlerListener handlerListener) {
1328     }
1329
1330     @Perm(requires="share(this) in alive",
1331           ensures="share(this) in alive")
1332     public void updateElement(UIElement element, Map parameters) {
1333     }
1334 }ENDOFCLASS
1335
1336 @ClassStates({@State(name = "alive")})
1337
1338 class CrystalUIAction {
1339     @Perm(ensures="unique(this) in alive")
1340     CrystalUIAction() { }
1341
1342     @Perm(requires="pure(this) in alive",
1343           ensures="pure(this) in alive")
1344     public void run(IAction action) {
1345     }
1346
1347     public void selectionChanged(IAction action, ISelection selection) {

```

```

1361 }
1363 public void dispose() {
1365 }
1367 public void init(IWorkbenchWindow window) {
1369 }
1371 }ENDOFCLASS
1373 @ClassStates({@State(name = "alive")})
1375 class Anonymous {
1376 @Perm(ensures="unique(this) in alive")
1377 Anonymous() { }
1379 @Perm(requires="unique(this) in alive",
1380 ensures="unique(this) in alive")
1381 protected IStatus run(IProgressMonitor monitor) {
1382 return null;
1384 }
1386 }ENDOFCLASS
1388 @ClassStates({@State(name = "alive")})
1390 class MethodFindVisitor {
1391 @Perm(ensures="unique(this) in alive")
1392 MethodFindVisitor() { }
1394 @Perm(requires="unique(this) in alive",
1395 ensures="unique(this) in alive")
1396 public boolean visit(MethodDeclaration methodDeclaration) {
1397 return 0;
1399 }
1401 }ENDOFCLASS
1403 @ClassStates({@State(name = "alive")})
1405 class Anonymous {
1406 @Perm(ensures="unique(this) in alive")
1407 Anonymous() { }
1410 public void endVisit(TypeDeclaration node) {
1412 }
1414 }ENDOFCLASS
1416 @ClassStates({@State(name = "alive")})
1418 class BindingsCollectorVisitor {
1419 @Perm(ensures="unique(this) in alive")
1420 BindingsCollectorVisitor() { }
1422 @Perm(requires="share(this) in alive",
1423 ensures="share(this) in alive")
1424 protected void addNewBinding(IBinding binding, ASTNode node) {
1426 }
1427 @Perm(requires="share(this) in alive",
1428 ensures="share(this) in alive")
1429 public boolean visit(AnonymousClassDeclaration node) {
1430 return 0;
1432 }
1434 }ENDOFCLASS
1436 @ClassStates({@State(name = "alive")})
1438 class StudentRuntimeException {
1439 @Perm(ensures="unique(this) in alive")
1440 StudentRuntimeException() { }

```

```
1443 }ENDOFCLASS
```