

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
BranchSensitiveTACAnalysis	1	1	0	0	0	1	0	0
SourceVariable	8	1	0	0	8	36	21	58
Variable	4	1	0	0	3	10	6	60
AbstractTACBranchSensitiveTransferFunction	9	1	0	0	8	45	33	73
SimpleInstructionVisitor	4	1	0	0	3	10	3	30
TransferVisitor	24	1	0	0	23	300	276	92
Lattice	1	1	0	0	0	1	0	0
TypeVariable	7	1	0	0	6	28	9	32
AbstractingTransferFunction	24	1	0	0	23	300	273	91
SuperVariable	5	1	0	0	4	15	8	53
KeywordVariable	5	1	0	0	3	15	6	40
EclipseTAC	15	1	0	0	14	120	40	33
BranchInsensitiveTACAnalysis	1	1	0	0	0	1	0	0
UnaryOperator	2	1	0	0	2	3	2	67
TempVariable	8	1	0	0	3	36	6	17
TACFlowAnalysis	13	1	0	0	2	91	3	3
BranchInsensitiveTACAnalysisDriver	3	1	0	0	1	6	1	17
BranchSensitiveTACAnalysisDriver	3	1	0	0	1	6	1	17
NewInstructionVisitor	2	1	0	0	1	3	1	33
MotherFlowAnalysis	11	1	0	0	11	66	31	47
SingleResult	1	1	0	0	0	1	0	0
ThisVariable	8	1	0	0	7	36	13	36
AbstractTACAnalysisDriver	4	1	0	0	2	10	3	30
CompilationUnitTACs	2	1	0	0	0	3	0	0
LabeledSingleResult	1	1	0	0	0	1	0	0
AbstractTransferFunction	23	1	0	0	22	276	250	91
BinaryOperator	2	1	0	0	2	3	2	67
Total Classes=27	191	27	0	0	149	1423	988	69

Contents

1	RunCrystalHandler	4
2	AbstractCrystalPlugin	5
3	Anonymous	6
4	WorkspaceUtilities	7
5	Crystal	8
6	Anonymous	9
7	ControlFlowGraph	10
8	ControlFlowNode	11
9	ControlFlowVisitor	13
10	Direction	14
11	CrystalRuntimeException	15
12	UserConsoleView	16
13	NullPrintWriter	17
14	Anonymous	18
15	ClearWarningHandler	19
16	Box	20
17	DisplayCrystalInfo	21
18	ShortFormatter	22
19	Utilities2	23
20	Option	24
21	Anonymous	25
22	AnalysisMenuPopulator	26
23	CrystalFileAction	27
24	Anonymous	28
25	Freezable	29
26	EnableAnalysisHandler	30
27	CrystalUIAction	31
28	Anonymous	32
29	MethodFindVisitor	33

30 Anonymous	34
31 BindingsCollectorVisitor	35
32 StudentRuntimeException	36
33 Abbreviation	37
34 Annotated Version of Sequential Java Program generated by Sip4j	38

1 BranchSensitiveTACAnalysis

Table 2: Methods Requires Clause Satisfiability

Method	Satisfiability
BranchSensitiveTACAnalysis	✓

Table 3: State Transition Matrix

	alive
alive	↑

2 SourceVariable

Table 4: Methods Requires Clause Satisfiability

Method	Satisfiability
SourceVariable	✓
getBinding	✓
isCapturedFromOuterScope	✓
dispatch	✓
hashCode	✓
equals	✓
toString	✓
resolveType	✓

Table 5: State Transition Matrix

	alive
alive	↑

Table 6: Methods Concurrency Matrix

	SourceVariable	getBinding	isCapturedFromOuterScope	dispatch	hashCode	equals	toString	resolveType
SourceVariable	⌈	⌈	⌋	⌈	⌈	⌈	⌋	⌈
getBinding	⌈	⌋	⌋	⌋	⌈	⌈	⌋	⌈
isCapturedFromOuterScope	⌋	⌋	⌋	⌋	⌋	⌋	⌋	⌋
dispatch	⌈	⌋	⌋	⌋	⌋	⌋	⌋	⌋
hashCode	⌈	⌈	⌋	⌋	⌈	⌈	⌋	⌈
equals	⌈	⌈	⌋	⌋	⌈	⌈	⌋	⌈
toString	⌋	⌋	⌋	⌋	⌋	⌋	⌋	⌋
resolveType	⌈	⌈	⌋	⌋	⌈	⌈	⌋	⌈

3 Variable

Table 7: Methods Requires Clause Satisfiability

Method	Satisfiability
Variable	✓
getSourceString	✓
isUnqualifiedSuper	✓
isUnqualifiedThis	✓

Table 8: State Transition Matrix

	alive
alive	↑

Table 9: Methods Concurrency Matrix

	Variable	getSourceString	isUnqualifiedSuper	isUnqualifiedThis
Variable	⌘	⌘	⌘	⌘
getSourceString	⌘			
isUnqualifiedSuper	⌘			
isUnqualifiedThis	⌘			

4 AbstractTACBranchSensitiveTransferFunction

Table 10: Methods Requires Clause Satisfiability

Method	Satisfiability
AbstractTACBranchSensitiveTransferFunction	✓
getAnalysisDirection	✓
getAnalysisContext	✓
setAnalysisContext	✓
transferOver2	✓
transferOver4	✓
transferO1	✓
transferOverload4	✓
transfer	✓

Table 11: State Transition Matrix

	alive
alive	↑

Table 12: Methods Concurrency Matrix

	AbstractTACBranchSensitiveTransferFunction	getAnalysisDirection	getAnalysisContext	setAnalysisContext	transferOver2	transferOver4	transferO1	transferOverload4	transfer
AbstractTACBranchSensitiveTransferFunction	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getAnalysisDirection	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getAnalysisContext	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
setAnalysisContext	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
transferOver2	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
transferOver4	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
transferO1	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
transferOverload4	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
transfer	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘

5 SimpleInstructionVisitor

Table 13: Methods Requires Clause Satisfiability

Method	Satisfiability
SimpleInstructionVisitor	✓
analyzeMethod	✓
doAccept	✓
visit	✓

Table 14: State Transition Matrix

	alive
alive	↑

Table 15: Methods Concurrency Matrix

	SimpleInstructionVisitor	analyzeMethod	doAccept	visit
SimpleInstructionVisitor	⌘	⌘	⌘	⌘
analyzeMethod	⌘	⌘	⌘	
doAccept	⌘	⌘	⌘	
visit	⌘			

6 TransferVisitor

Table 16: Methods Requires Clause Satisfiability

Method	Satisfiability
TransferVisitor	✓
transferOver	✓
transferOver2	✓
transferOver3	✓
transferOver5	✓
transferOver6	✓
transferOver4	✓
transferOver7	✓
transferOver9	✓
transferOver10	✓
transferOver8	✓
transferOver11	✓
transferOver12	✓
transferOver13	✓
transferOver14	✓
transferOver15	✓
transferOver16	✓
transferOver17	✓
transferOver18	✓
transfer	✓
getLattice	✓
getAnalysisDirection	✓
setAnalysisContext	✓
transferOver19	✓

Table 17: State Transition Matrix

	alive
alive	↑

Table 18: Methods Concurrency Matrix

	TransferVisitor	transferOver	transferOver2	transferOver3	transferOver5	transferOver6	transferOver4	transferOver7	transferOver9	transferOver10	transferOver8	transferOver11	transferOver12	transferOver13	transferOver14	transferOver15	transferOver16	transferOver17	transferOver18	transfer	getLattice	getAnalysisDirection	setAnalysisContext	transferOver19
TransferVisitor	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
transferOver	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
transferOver2	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
transferOver3	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
transferOver5	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

7 Lattice

Table 19: Methods Requires Clause Satisfiability

Method	Satisfiability
Lattice	✓

Table 20: State Transition Matrix

	alive
alive	↑

8 TypeVariable

Table 21: Methods Requires Clause Satisfiability

Method	Satisfiability
TypeVariable	✓
getType	✓
hashCode	✓
equals	✓
toString	✓
resolveType	✓
dispatch	✓

Table 22: State Transition Matrix

	alive
alive	↑

Table 23: Methods Concurrency Matrix

	TypeVariable	getType	hashCode	equals	toString	resolveType	dispatch
TypeVariable	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getType	⌘		⌘	⌘	⌘		
hashCode	⌘	⌘	⌘	⌘	⌘	⌘	
equals	⌘	⌘	⌘	⌘	⌘	⌘	
toString	⌘	⌘	⌘	⌘	⌘	⌘	
resolveType	⌘		⌘	⌘	⌘		
dispatch	⌘						

9 AbstractingTransferFunction

Table 24: Methods Requires Clause Satisfiability

Method	Satisfiability
AbstractingTransferFunction	✓
getAnalysisDirection	✓
getAnalysisContext	✓
setAnalysisContext	✓
transfer	✓
transferOver	✓
transferOver2	✓
transferOver3	✓
transferOver4	✓
transferOver5	✓
transferOver6	✓
transferOver7	✓
transferOver8	✓
transferOver9	✓
transferOver10	✓
transferOver11	✓
transferOver12	✓
transferOver13	✓
transferOver14	✓
transferOver15	✓
transferOver16	✓
transferOver17	✓
transferOver18	✓
transferOver19	✓

Table 25: State Transition Matrix

	alive
alive	↑

Table 26: Methods Concurrency Matrix

	AbstractingTransferFunction	getAnalysisDirection	getAnalysisContext	setAnalysisContext	transfer	transferOver	transferOver2	transferOver3	transferOver4	transferOver5	transferOver6	transferOver7	transferOver8	transferOver9	transferOver10	transferOver11	transferOver12	transferOver13	transferOver14	transferOver15	transferOver16
AbstractingTransferFunction	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getAnalysisDirection	⌘			⌘																	

getAnalysisContext	⌘			⌘																			
setAnalysisContext	⌘	⌘	⌘	⌘																			
transfer	⌘																						
transferOver	⌘																						
transferOver2	⌘																						
transferOver3	⌘																						
transferOver4	⌘																						
transferOver5	⌘																						
transferOver6	⌘																						
transferOver7	⌘																						
transferOver8	⌘																						
transferOver9	⌘																						
transferOver10	⌘																						
transferOver11	⌘																						
transferOver12	⌘																						
transferOver13	⌘																						
transferOver14	⌘																						
transferOver15	⌘																						
transferOver16	⌘																						
transferOver17	⌘																						
transferOver18	⌘																						
transferOver19	⌘																						

10 SuperVariable

Table 27: Methods Requires Clause Satisfiability

Method	Satisfiability
SuperVariable	✓
getKeyword	✓
resolveType	✓
dispatch	✓
isUnqualifiedSuper	✓

Table 28: State Transition Matrix

	alive
alive	↑

Table 29: Methods Concurrency Matrix

	SuperVariable	getKeyword	resolveType	dispatch	isUnqualifiedSuper
SuperVariable	⌘	⌘	⌘	⌘	⌘
getKeyword	⌘				
resolveType	⌘		⌘		⌘
dispatch	⌘				
isUnqualifiedSuper	⌘		⌘		

11 KeywordVariable

Table 30: Methods Requires Clause Satisfiability

Method	Satisfiability
KeywordVariable	✓
getQualifier	✓
isQualified	✓
setQualifier	✓
toString	✓

Table 31: State Transition Matrix

	alive
alive	↑

Table 32: Methods Concurrency Matrix

	KeywordVariable	getQualifier	isQualified	setQualifier	toString
KeywordVariable	⌘	⌘	⌘	⌘	⌘
getQualifier	⌘			⌘	
isQualified	⌘			⌘	
setQualifier	⌘	⌘	⌘	⌘	⌘
toString	⌘			⌘	

12 EclipseTAC

Table 33: Methods Requires Clause Satisfiability

Method	Satisfiability
EclipseTAC	✓
resolveThisType	✓
isStaticBinding	✓
instruction	✓
createInstruction	✓
variable	✓
getVariable	✓
getThisVariable	✓
thisVariable	✓
superVariable	✓
sourceVariable	✓
implicitThisVariable	✓
implicitThisBinding	✓
findElementDeclarationByName	✓
isDefaultBinding	✓

Table 34: State Transition Matrix

	alive
alive	↑

Table 35: Methods Concurrency Matrix

	EclipseTAC	resolveThisType	isStaticBinding	instruction	createInstruction	variable	getVariable	getThisVariable	thisVariable	superVariable	sourceVariable	implicitThisVariable	implicitThisBinding	findElementDeclarationByName	isDefaultBinding
EclipseTAC	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
resolveThisType	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
isStaticBinding	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
instruction	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
createInstruction	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
variable	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getVariable	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getThisVariable	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
thisVariable	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
superVariable	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘

sourceVariable	⌘	⌘		⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘		
implicitThisVariable	⌘	⌘		⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘		
implicitThisBinding	⌘	⌘		⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘		
findElementDeclarationByName	⌘															
isDefaultBinding	⌘															

13 BranchInsensitiveTACAnalysis

Table 36: Methods Requires Clause Satisfiability

Method	Satisfiability
BranchInsensitiveTACAnalysis	✓

Table 37: State Transition Matrix

	alive
alive	↑

14 UnaryOperator

Table 38: Methods Requires Clause Satisfiability

Method	Satisfiability
UnaryOperator	✓
toString	✓

Table 39: State Transition Matrix

	alive
alive	↑

Table 40: Methods Concurrency Matrix

	UnaryOperator	toString
UnaryOperator	⌈	
toString		

15 TempVariable

Table 41: Methods Requires Clause Satisfiability

Method	Satisfiability
TempVariable	✓
getNode	✓
dispatch	✓
hashCode	✓
equals	✓
toString	✓
getSourceString	✓
resolveType	✓

Table 42: State Transition Matrix

	alive
alive	↑

Table 43: Methods Concurrency Matrix

	TempVariable	getNode	dispatch	hashCode	equals	toString	getSourceString	resolveType
TempVariable	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈
getNode	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈
dispatch	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈
hashCode	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈
equals	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈
toString	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈
getSourceString	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈
resolveType	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈

16 TACFlowAnalysis

Table 44: Methods Requires Clause Satisfiability

Method	Satisfiability
TACFlowAnalysis	✓
getResultsAfter	✓
getResultsBefore	✓
getLabeledResultsAfter	✓
getLabeledResultsBefore	✓
getNode	✓
getVariable	✓
getThisVariable	✓
getSuperVariable	✓
getSourceVariable	✓
getAnalyzedMethod	✓
getImplicitThisVariable	✓
createTransferFunction	✓

Table 45: State Transition Matrix

	alive
alive	↑

Table 46: Methods Concurrency Matrix

	TACFlowAnalysis	getResultsAfter	getResultsBefore	getLabeledResultsAfter	getLabeledResultsBefore	getNode	getVariable	getThisVariable	getSuperVariable	getSourceVariable	getAnalyzedMethod	getImplicitThisVariable	createTransferFunction
TACFlowAnalysis	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getResultsAfter	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getResultsBefore	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getLabeledResultsAfter	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getLabeledResultsBefore	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getNode	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getVariable	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getThisVariable	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getSuperVariable	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getSourceVariable	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getAnalyzedMethod	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getImplicitThisVariable	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
createTransferFunction	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘

17 BranchInsensitiveTACAnalysisDriver

Table 47: Methods Requires Clause Satisfiability

Method	Satisfiability
BranchInsensitiveTACAnalysisDriver	✓
transfer	✓
deriveResult	✓

Table 48: State Transition Matrix

	alive
alive	↑

Table 49: Methods Concurrency Matrix

	BranchInsensitiveTACAnalysisDriver		
	transfer		
	deriveResult		
BranchInsensitiveTACAnalysisDriver	#	#	#
transfer	#	#	#
deriveResult	#	#	

18 BranchSensitiveTACAnalysisDriver

Table 50: Methods Requires Clause Satisfiability

Method	Satisfiability
BranchSensitiveTACAnalysisDriver	✓
transfer	✓
deriveResult	✓

Table 51: State Transition Matrix

	alive
alive	↑

Table 52: Methods Concurrency Matrix

	BranchSensitiveTACAnalysisDriver		
BranchSensitiveTACAnalysisDriver	⌘	⌘	⌘
transfer	⌘	⌘	⌘
deriveResult	⌘	⌘	

19 NewInstructionVisitor

Table 53: Methods Requires Clause Satisfiability

Method	Satisfiability
NewInstructionVisitor	✓
getResult	✓

Table 54: State Transition Matrix

	alive
alive	↑

Table 55: Methods Concurrency Matrix

	NewInstructionVisitor	getResult
NewInstructionVisitor	⌘	⌘
getResult	⌘	

20 MotherFlowAnalysis

Table 56: Methods Requires Clause Satisfiability

Method	Satisfiability
MotherFlowAnalysis	✓
getResultsAfter	✓
mergeLabeledResult	✓
checkNull	✓
getResultsBefore	✓
getLabeledResultsAfter	✓
getLabeledResultAfter	✓
mergeLabeledResults	✓
getLabeledResultsBefore	✓
getLabeledResultBefore	✓
getCurrentMethod	✓

Table 57: State Transition Matrix

	alive
alive	↑

Table 58: Methods Concurrency Matrix

	MotherFlowAnalysis	getResultsAfter	mergeLabeledResult	checkNull	getResultsBefore	getLabeledResultsAfter	getLabeledResultAfter	mergeLabeledResults	getLabeledResultsBefore	getLabeledResultBefore	getCurrentMethod
MotherFlowAnalysis	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getResultsAfter	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
mergeLabeledResult	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
checkNull	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getResultsBefore	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getLabeledResultsAfter	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getLabeledResultAfter	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
mergeLabeledResults	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getLabeledResultsBefore	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getLabeledResultBefore	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getCurrentMethod	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘

21 **SingleResult**

Table 59: Methods Requires Clause Satisfiability

Method	Satisfiability
SingleResult	✓

Table 60: State Transition Matrix

	alive
alive	↑

22 ThisVariable

Table 61: Methods Requires Clause Satisfiability

Method	Satisfiability
ThisVariable	✓
isImplicit	✓
explicitQualifier	✓
getKeyword	✓
isQualified	✓
resolveType	✓
dispatch	✓
isUnqualifiedThis	✓

Table 62: State Transition Matrix

	alive
alive	↑

Table 63: Methods Concurrency Matrix

	ThisVariable	isImplicit	explicitQualifier	getKeyword	isQualified	resolveType	dispatch	isUnqualifiedThis
ThisVariable	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
isImplicit	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
explicitQualifier	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getKeyword	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
isQualified	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
resolveType	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
dispatch	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
isUnqualifiedThis	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘

23 AbstractTACAnalysisDriver

Table 64: Methods Requires Clause Satisfiability

Method	Satisfiability
AbstractTACAnalysisDriver	✓
switchToMethod	✓
getAnalysisDirection	✓
getLattice	✓

Table 65: State Transition Matrix

	alive
alive	↑

Table 66: Methods Concurrency Matrix

	AbstractTACAnalysisDriver	switchToMethod	getAnalysisDirection	getLattice
AbstractTACAnalysisDriver	⧻	⧻	⧻	⧻
switchToMethod	⧻	⧻	⧻	⧻
getAnalysisDirection	⧻	⧻		
getLattice	⧻	⧻		

24 CompilationUnitTACs

Table 67: Methods Requires Clause Satisfiability

Method	Satisfiability
CompilationUnitTACs	✓
getMethodTAC	✓

Table 68: State Transition Matrix

	alive
alive	↑

Table 69: Methods Concurrency Matrix

	CompilationUnitTACs	getMethodTAC
CompilationUnitTACs	⌘	⌘
getMethodTAC	⌘	⌘

25 **LabeledSingleResult**

Table 70: Methods Requires Clause Satisfiability

Method	Satisfiability
LabeledSingleResult	✓

Table 71: State Transition Matrix

	alive
alive	↑

26 AbstractTransferFunction

Table 72: Methods Requires Clause Satisfiability

Method	Satisfiability
AbstractTransferFunction	✓
getAnalysisDirection	✓
getAnalysisContext	✓
setAnalysisContext	✓
transferOver	✓
transferOver2	✓
transferOver3	✓
transferOver4	✓
transferOver5	✓
transferOver6	✓
transferOver7	✓
transferOver8	✓
transferOver9	✓
transferOver10	✓
transferOver11	✓
transferOver12	✓
transferOver13	✓
transferOver14	✓
transferOver15	✓
transferOver16	✓
transferOver17	✓
transferOver18	✓
transferOver19	✓

Table 73: State Transition Matrix

	alive
alive	↑

Table 74: Methods Concurrency Matrix

	AbstractTransferFunction	getAnalysisDirection	getAnalysisContext	setAnalysisContext	transferOver	transferOver2	transferOver3	transferOver4	transferOver5	transferOver6	transferOver7	transferOver8	transferOver9	transferOver10	transferOver11	transferOver12	transferOver13	transferOver14	transferOver15	transferOver16	transferOver17	transferOver18
AbstractTransferFunction	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getAnalysisDirection	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getAnalysisContext	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
setAnalysisContext	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘

27 BinaryOperator

Table 75: Methods Requires Clause Satisfiability

Method	Satisfiability
BinaryOperator	✓
toString	✓

Table 76: State Transition Matrix

	alive
alive	↑

Table 77: Methods Concurrency Matrix

	BinaryOperator	toString
BinaryOperator	⊥	
toString		

28 Abbreviation

Table 78: 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

29 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 BranchSensitiveTACAnalysis {
6   @Perm(ensures="unique(this) in alive")
7   BranchSensitiveTACAnalysis() { }
8
9 }
10 }ENDOFCLASS
11
12 @ClassStates({@State(name = "alive")})
13
14 class SourceVariable {
15   @Perm(ensures="unique(this) in alive")
16   SourceVariable() { }
17
18   @Perm(requires="pure(this) in alive",
19   ensures="pure(this) in alive")
20   public IVariableBinding getBinding() {
21     return null;
22   }
23 }
24 @Perm(ensures="none(this) in alive")
25 public boolean isCapturedFromOuterScope() {
26   return 0;
27 }
28 }
29
30 public T dispatch(IVariableVisitor<T> visitor) {
31   return null;
32 }
33 }
34 @Perm(requires="share(this) in alive",
35 ensures="share(this) in alive")
36 public int hashCode() {
37   return 0;
38 }
39 }
40 @Perm(requires="share(this) in alive",
41 ensures="share(this) in alive")
42 public boolean equals(Object obj) {
43   return 0;
44 }
45 }
46 @Perm(ensures="none(this) in alive")
47 public String toString() {
48   return null;
49 }
50 }
51 @Perm(requires="unique(this) in alive",
52 ensures="unique(this) in alive")
53 public ITypeBinding resolveType() {
54   return null;
55 }
56 }
57 }ENDOFCLASS
58
59 @ClassStates({@State(name = "alive")})
60
61 class Variable {
62   @Perm(ensures="unique(this) in alive")
63   Variable() { }
64
65   public String getSourceString() {
66     return null;
67   }
68 }
69 }
70
71 public boolean isUnqualifiedSuper() {
72   return 0;
73 }
74 }
75 }
```

```

77 public boolean isUnqualifiedThis() {
78     return 0;
79 }
80 }
81 }ENDOFCLASS
82
83 @ClassStates({@State(name = "alive")})
84
85 class AbstractTACBranchSensitiveTransferFunction {
86     @Perm(ensures="unique(this) in alive")
87     AbstractTACBranchSensitiveTransferFunction() { }
88
89     @Perm(requires="immutable(this) in alive",
90     ensures="immutable(this) in alive")
91     public AnalysisDirection getAnalysisDirection() {
92         return null;
93     }
94
95     @Perm(requires="pure(this) in alive",
96     ensures="pure(this) in alive")
97     public ITACAnalysisContext getAnalysisContext() {
98         return null;
99     }
100 }
101
102 @Perm(requires="full(this) in alive",
103 ensures="full(this) in alive")
104 public void setAnalysisContext(ITACAnalysisContext analysisContext) {
105 }
106
107
108 public IResult<LE> transferOver2(ArrayInitInstruction instr, List<ILabel> labels, LE value) {
109     return null;
110 }
111
112
113 public IResult<LE> transferOver4(BinaryOperation binop, List<ILabel> labels, LE value) {
114     return null;
115 }
116
117
118 public IResult<LE> transfer01(CastInstruction instr, List<ILabel> labels, LE value) {
119     return null;
120 }
121
122
123 public IResult<LE> transferOverload4(DotClassInstruction instr, List<ILabel> labels, LE value) {
124     return null;
125 }
126
127
128 public IResult<LE> transfer(ConstructorCallInstruction instr, List<ILabel> labels, LE value) {
129     return null;
130 }
131 }
132 }ENDOFCLASS
133
134 @ClassStates({@State(name = "alive")})
135
136 class SimpleInstructionVisitor {
137     @Perm(ensures="unique(this) in alive")
138     SimpleInstructionVisitor() { }
139
140     @Perm(requires="unique(this) in alive",
141     ensures="unique(this) in alive")
142     public void analyzeMethod(MethodDeclaration d) {
143     }
144
145     @Perm(requires="unique(this) in alive",
146     ensures="unique(this) in alive")
147     void doAccept(MethodDeclaration d) {
148     }
149 }
150
151
152 public void visit(ArrayInitInstruction instr) {
153 }
154 }
155 }ENDOFCLASS

```

```

158 @ClassStates({@State(name = "alive")})
160 class TransferVisitor {
161   @Perm(ensures="unique(this) in alive")
162   TransferVisitor() { }

165   public SingletonLatticeElement transferOver(ArrayInitInstruction instr, SingletonLatticeElement value)
166   {
167     return null;
168   }

170   public SingletonLatticeElement transferOver2(BinaryOperation binop, SingletonLatticeElement value) {
171     return null;
172   }

173   }

175   public SingletonLatticeElement transferOver3(CastInstruction instr, SingletonLatticeElement value) {
176     return null;
177   }

178   }

180   public SingletonLatticeElement transferOver5(ConstructorCallInstruction instr, SingletonLatticeElement
181     value) {
182     return null;
183   }

185   public SingletonLatticeElement transferOver6(CopyInstruction instr, SingletonLatticeElement value) {
186     return null;
187   }

188   }

190   public SingletonLatticeElement transferOver4(DotClassInstruction instr, SingletonLatticeElement value)
191   {
192     return null;
193   }

195   public SingletonLatticeElement transferOver7(InstanceOfInstruction instr, SingletonLatticeElement value
196     ) {
197     return null;
198   }

200   public SingletonLatticeElement transferOver9(LoadArrayInstruction instr, SingletonLatticeElement value)
201   {
202     return null;
203   }

205   public SingletonLatticeElement transferOver10(LoadFieldInstruction instr, SingletonLatticeElement value
206     ) {
207     return null;
208   }

210   public SingletonLatticeElement transferOver8(LoadLiteralInstruction instr, SingletonLatticeElement
211     value) {
212     return null;
213   }

215   public SingletonLatticeElement transferOver11(MethodCallInstruction instr, SingletonLatticeElement
216     value) {
217     return null;
218   }

220   public SingletonLatticeElement transferOver12(NewArrayInstruction instr, SingletonLatticeElement value)
221   {
222     return null;
223   }

225   public SingletonLatticeElement transferOver13(NewObjectInstruction instr, SingletonLatticeElement value
226     ) {
227     return null;

```

```

228 }

230 public SingletonLatticeElement transferOver14(ReturnInstruction instr, SingletonLatticeElement value) {
231     return null;
232 }

233 }

235 public SingletonLatticeElement transferOver15(StoreArrayInstruction instr, SingletonLatticeElement
236     value) {
237     return null;
238 }

240 public SingletonLatticeElement transferOver16(StoreFieldInstruction instr, SingletonLatticeElement
241     value) {
242     return null;
243 }

245 public SingletonLatticeElement transferOver17(SourceVariableDeclaration instr, SingletonLatticeElement
246     value) {
247     return null;
248 }

250 public SingletonLatticeElement transferOver18(SourceVariableRead instr, SingletonLatticeElement value)
251     {
252     return null;
253 }

255 public SingletonLatticeElement transfer(UnaryOperation unop, SingletonLatticeElement value) {
256     return null;
257 }

258 }
259 @Perm(requires="immutable(this) in alive",
260     ensures="immutable(this) in alive")
261 public Lattice<SingletonLatticeElement> getLattice(MethodDeclaration d) {
262     return null;
263 }

264 }
265 @Perm(requires="immutable(this) in alive",
266     ensures="immutable(this) in alive")
267 public AnalysisDirection getAnalysisDirection() {
268     return null;
269 }

270 }

272 public void setAnalysisContext(ITACAnalysisContext analysisContext) {
273 }

274 }

276 public SingletonLatticeElement transferOver19(UnaryOperation unop, SingletonLatticeElement value) {
277     return null;
278 }

279 }

281 }ENDOFCLASS

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

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

290 }ENDOFCLASS

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

294 class TypeVariable {
295     @Perm(ensures="unique(this) in alive")
296     TypeVariable() { }
297 }

298 @Perm(requires="pure(this) in alive",
299     ensures="pure(this) in alive")
300 public ITypeBinding getType() {
301     return null;
302 }

303 }
304 @Perm(requires="share(this) in alive",

```

```

305 ensures="share(this) in alive")
306 public int hashCode() {
307     return 0;
308 }
309 }
310 @Perm(requires="share(this) in alive",
311 ensures="share(this) in alive")
312 public boolean equals(Object obj) {
313     return 0;
314 }
315 }
316 @Perm(requires="unique(this) in alive",
317 ensures="unique(this) in alive")
318 public String toString() {
319     return null;
320 }
321 }
322 @Perm(requires="pure(this) in alive",
323 ensures="pure(this) in alive")
324 public ITypeBinding resolveType() {
325     return null;
326 }
327 }
328
329 public T dispatch(IVariableVisitor<T> visitor) {
330     return null;
331 }
332 }
333 }ENDOFCLASS
334
335 @ClassStates({@State(name = "alive")})
336
337 class AbstractingTransferFunction {
338     @Perm(ensures="unique(this) in alive")
339     AbstractingTransferFunction() { }
340
341     @Perm(requires="immutable(this) in alive",
342     ensures="immutable(this) in alive")
343     public AnalysisDirection getAnalysisDirection() {
344         return null;
345     }
346 }
347 }
348 @Perm(requires="pure(this) in alive",
349 ensures="pure(this) in alive")
350 public ITACAnalysisContext getAnalysisContext() {
351     return null;
352 }
353 }
354 @Perm(requires="full(this) in alive",
355 ensures="full(this) in alive")
356 public void setAnalysisContext(ITACAnalysisContext analysisContext) {
357 }
358 }
359
360 public LE transfer(TACInstruction instr, LE value) {
361     return null;
362 }
363 }
364
365 public LE transferOver(ArrayInitInstruction instr, LE value) {
366     return null;
367 }
368 }
369
370 public LE transferOver2(BinaryOperation binop, LE value) {
371     return null;
372 }
373 }
374
375 public LE transferOver3(CastInstruction instr, LE value) {
376     return null;
377 }
378 }
379
380 public LE transferOver4(DotClassInstruction instr, LE value) {
381     return null;
382 }
383 }
384
385 public LE transferOver5(ConstructorCallInstruction instr, LE value) {

```



```

386     return null;
387 }
388
389 public LE transferOver6(CopyInstruction instr, LE value) {
390     return null;
391 }
392
393
394 public LE transferOver7(InstanceOfInstruction instr, LE value) {
395     return null;
396 }
397
398
399
400 public LE transferOver8(LoadLiteralInstruction instr, LE value) {
401     return null;
402 }
403
404
405 public LE transferOver9(LoadArrayInstruction instr, LE value) {
406     return null;
407 }
408
409
410 public LE transferOver10(LoadFieldInstruction instr, LE value) {
411     return null;
412 }
413
414
415 public LE transferOver11(MethodCallInstruction instr, LE value) {
416     return null;
417 }
418
419
420 public LE transferOver12(NewArrayInstruction instr, LE value) {
421     return null;
422 }
423
424
425 public LE transferOver13(NewObjectInstruction instr, LE value) {
426     return null;
427 }
428
429
430 public LE transferOver14(ReturnInstruction instr, LE value) {
431     return null;
432 }
433
434
435 public LE transferOver15(StoreArrayInstruction instr, LE value) {
436     return null;
437 }
438
439
440 public LE transferOver16(StoreFieldInstruction instr, LE value) {
441     return null;
442 }
443
444
445 public LE transferOver17(SourceVariableDeclaration instr, LE value) {
446     return null;
447 }
448
449
450 public LE transferOver18(SourceVariableRead instr, LE value) {
451     return null;
452 }
453
454
455 public LE transferOver19(UnaryOperation unop, LE value) {
456     return null;
457 }
458
459 }
460 }ENDOFCLASS
461
462 @ClassStates({@State(name = "alive")})
463
464 class SuperVariable {
465     @Perm(ensures="unique(this) in alive")
466     SuperVariable() { }

```

```

469 public String getKeyword() {
470     return null;
471 }
472 }
473 @Perm(requires="unique(this) in alive",
474 ensures="unique(this) in alive")
475 public ITypeBinding resolveType() {
476     return null;
477 }
478 }
479
480 public T dispatch(IVariableVisitor<T> visitor) {
481     return null;
482 }
483 }
484 @Perm(requires="pure(this) in alive",
485 ensures="pure(this) in alive")
486 public boolean isUnqualifiedSuper() {
487     return 0;
488 }
489 }
490 }ENDOFCLASS
491
492 @ClassStates({@State(name = "alive")})
493
494 class KeywordVariable {
495     @Perm(ensures="unique(this) in alive")
496     KeywordVariable() { }
497 }
498
499 @Perm(requires="pure(this) in alive",
500 ensures="pure(this) in alive")
501 public Name getQualifier() {
502     return null;
503 }
504 }
505 @Perm(requires="pure(this) in alive",
506 ensures="pure(this) in alive")
507 public boolean isQualified() {
508     return 0;
509 }
510 }
511 @Perm(requires="share(this) in alive",
512 ensures="share(this) in alive")
513 protected void setQualifier(Name qualifier) {
514 }
515 }
516 @Perm(requires="pure(this) in alive",
517 ensures="pure(this) in alive")
518 public String toString() {
519     return null;
520 }
521 }
522 }ENDOFCLASS
523
524 @ClassStates({@State(name = "alive")})
525
526 class EclipseTAC {
527     @Perm(ensures="unique(this) in alive")
528     EclipseTAC() { }
529 }
530
531 @Perm(requires="unique(this) in alive",
532 ensures="unique(this) in alive")
533 public ITypeBinding resolveThisType() {
534     return null;
535 }
536 }
537
538 boolean isStaticBinding(IBinding binding) {
539     return 0;
540 }
541 }
542 @Perm(requires="unique(this) in alive",
543 ensures="unique(this) in alive")
544 public TACInstruction instruction(ASTNode astNode) {
545     return null;
546 }
547 }

```

```

548 @Perm(requires="immutable(this) in alive",
549 ensures="immutable(this) in alive")
550 private TACInstruction createInstruction(ASTNode astNode) {
551     return null;
552 }
553 }
554 @Perm(requires="unique(this) in alive",
555 ensures="unique(this) in alive")
556 public Variable variable(ASTNode astNode) {
557     return null;
558 }
559 }
560 @Perm(requires="share(this) in alive",
561 ensures="share(this) in alive")
562 private Variable getVariable(IBinding binding) {
563     return null;
564 }
565 }
566 @Perm(requires="unique(this) in alive",
567 ensures="unique(this) in alive")
568 private ThisVariable getThisVariable(ThisExpression node) {
569     return null;
570 }
571 }
572 @Perm(requires="unique(this) in alive",
573 ensures="unique(this) in alive")
574 public ThisVariable thisVariable() {
575     return null;
576 }
577 }
578 @Perm(requires="unique(this) in alive",
579 ensures="unique(this) in alive")
580 public SuperVariable superVariable(Name qualifier) {
581     return null;
582 }
583 }
584 @Perm(requires="share(this) in alive",
585 ensures="share(this) in alive")
586 public SourceVariable sourceVariable(IVariableBinding binding) {
587     return null;
588 }
589 }
590 @Perm(requires="share(this) in alive",
591 ensures="share(this) in alive")
592 public ThisVariable implicitThisVariable(IBinding accessedElement) {
593     return null;
594 }
595 }
596 @Perm(requires="share(this) in alive",
597 ensures="share(this) in alive")
598 private ITypeBinding implicitThisBinding(IBinding accessedElement) {
599     return null;
600 }
601 }
602 }
603 private ITypeBinding findElementDeclarationByName(IBinding genericAccessedElement, boolean isMethod,
604 ITypeBinding type, boolean skipPrivate, boolean skipPackagePrivate) {
605     return null;
606 }
607 }
608 boolean isDefaultBinding(IBinding binding) {
609     return 0;
610 }
611 }
612 }
613 }ENDOFCLASS
614 @ClassStates({@State(name = "alive")})
615 }
616 class BranchInsensitiveTACAnalysis {
617 @Perm(ensures="unique(this) in alive")
618 BranchInsensitiveTACAnalysis() { }
619 }
620 }ENDOFCLASS
621 }
622 @ClassStates({@State(name = "alive")})
623 }
624 class UnaryOperator {
625 @Perm(ensures="unique(this) in alive")

```

```

628 UnaryOperator() { }
630 @Perm(ensures="none(this) in alive")
631 public String toString() {
632     return null;
633 }
634 }
635 }ENDOFCLASS
636 @ClassStates({@State(name = "alive")})
637
638 class TempVariable {
639     @Perm(ensures="unique(this) in alive")
640     TempVariable() { }
641
642     @Perm(requires="pure(this) in alive",
643           ensures="pure(this) in alive")
644     public ASTNode getNode() {
645         return null;
646     }
647 }
648
649 @Perm(requires="unique(this) in alive",
650       ensures="unique(this) in alive")
651 public T dispatch(IVariableVisitor<T> visitor) {
652     return null;
653 }
654
655 @Perm(requires="share(this) in alive",
656       ensures="share(this) in alive")
657 public int hashCode() {
658     return 0;
659 }
660
661 @Perm(requires="share(this) in alive",
662       ensures="share(this) in alive")
663 public boolean equals(Object obj) {
664     return 0;
665 }
666
667 @Perm(requires="pure(this) in alive",
668       ensures="pure(this) in alive")
669 public String toString() {
670     return null;
671 }
672
673 @Perm(requires="unique(this) in alive",
674       ensures="unique(this) in alive")
675 public String getSourceString() {
676     return null;
677 }
678
679 @Perm(requires="pure(this) in alive",
680       ensures="pure(this) in alive")
681 public ITypeBinding resolveType() {
682     return null;
683 }
684 }
685 }ENDOFCLASS
686
687 @ClassStates({@State(name = "alive")})
688
689 class TACFlowAnalysis {
690     @Perm(ensures="unique(this) in alive")
691     TACFlowAnalysis() { }
692
693     @Perm(requires="unique(this) in alive",
694           ensures="unique(this) in alive")
695     public LE getResultsAfter(TACInstruction instr) {
696         return null;
697     }
698 }
699
700 @Perm(requires="unique(this) in alive",
701       ensures="unique(this) in alive")
702 public LE getResultsBefore(TACInstruction instr) {
703     return null;
704 }
705
706 @Perm(requires="unique(this) in alive",
707       ensures="unique(this) in alive")
708

```

```

709 public IResult<LE> getLabeledResultsAfter(TACInstruction instr) {
710     return null;
711 }
712
713 @Perm(requires="unique(this) in alive",
714     ensures="unique(this) in alive")
715 public IResult<LE> getLabeledResultsBefore(TACInstruction instr) {
716     return null;
717 }
718
719 @Perm(requires="pure(this) in alive",
720     ensures="pure(this) in alive")
721 public ASTNode getNode(Variable x, TACInstruction instruction) {
722     return null;
723 }
724
725 @Perm(requires="unique(this) in alive",
726     ensures="unique(this) in alive")
727 public Variable getVariable(ASTNode node) {
728     return null;
729 }
730
731 @Perm(requires="unique(this) in alive",
732     ensures="unique(this) in alive")
733 public ThisVariable getThisVariable(MethodDeclaration methodDecl) {
734     return null;
735 }
736
737 @Perm(requires="unique(this) in alive",
738     ensures="unique(this) in alive")
739 public SuperVariable getSuperVariable() {
740     return null;
741 }
742
743 @Perm(requires="share(this) in alive",
744     ensures="share(this) in alive")
745 public SourceVariable getSourceVariable(IVariableBinding varBinding) {
746     return null;
747 }
748
749 @Perm(requires="immutable(this) in alive",
750     ensures="immutable(this) in alive")
751 public MethodDeclaration getAnalyzedMethod() {
752     return null;
753 }
754
755 @Perm(requires="share(this) in alive",
756     ensures="share(this) in alive")
757 public ThisVariable getImplicitThisVariable(IBinding accessedElement) {
758     return null;
759 }
760
761 @Perm(requires="unique(this) in alive",
762     ensures="unique(this) in alive")
763 protected IFlowAnalysisDefinition<LE> createTransferFunction(MethodDeclaration method) {
764     return null;
765 }
766 }
767
768 }ENDOFCLASS
769
770 @ClassStates({@State(name = "alive")})
771
772 class BranchInsensitiveTACAnalysisDriver {
773     @Perm(ensures="unique(this) in alive")
774     BranchInsensitiveTACAnalysisDriver() { }
775
776     @Perm(requires="unique(this) in alive",
777         ensures="unique(this) in alive")
778     public LE transfer(ASTNode astNode, LE incoming) {
779         return null;
780     }
781
782     @Perm(requires="immutable(this) in alive",
783         ensures="immutable(this) in alive")
784     public IResult<LE> deriveResult(EclipseInstructionSequence seq, LE incoming, TACInstruction
785         targetInstruction, boolean afterResult) {
786         return null;
787     }
788 }

```

```

789 }ENDOFCLASS
791 @ClassStates({@State(name = "alive")})
793 class BranchSensitiveTACAnalysisDriver {
794   @Perm(ensures="unique(this) in alive")
795   BranchSensitiveTACAnalysisDriver() { }
797   @Perm(requires="unique(this) in alive",
798     ensures="unique(this) in alive")
799   public IResult<LE> transfer(ASTNode astNode, List<ILabel> labels, LE value) {
800     return null;
802   }
803   @Perm(requires="immutable(this) in alive",
804     ensures="immutable(this) in alive")
805   public IResult<LE> deriveResult(EclipseInstructionSequence seq, LE incoming, TACInstruction
      targetInstruction, boolean afterResult) {
806     return null;
808   }
810 }ENDOFCLASS
812 @ClassStates({@State(name = "alive")})
814 class NewInstructionVisitor {
815   @Perm(ensures="unique(this) in alive")
816   NewInstructionVisitor() { }
818   @Perm(requires="immutable(this) in alive",
819     ensures="immutable(this) in alive")
820   public TACInstruction getResult() {
821     return null;
823   }
825 }ENDOFCLASS
827 @ClassStates({@State(name = "alive")})
829 class MotherFlowAnalysis {
830   @Perm(ensures="unique(this) in alive")
831   MotherFlowAnalysis() { }
833   @Perm(requires="immutable(this) in alive",
834     ensures="immutable(this) in alive")
835   public LE getResultsAfter(ASTNode node) {
836     return null;
838   }
840   protected LE mergeLabeledResult(IResult<LE> labeledResult, ASTNode node) {
841     return null;
843   }
845   T checkNull(T o) {
846     return null;
848   }
849   @Perm(requires="immutable(this) in alive",
850     ensures="immutable(this) in alive")
851   public LE getResultsBefore(ASTNode node) {
852     return null;
854   }
855   @Perm(requires="unique(this) in alive",
856     ensures="unique(this) in alive")
857   public IResult<LE> getLabeledResultsAfter(ASTNode node) {
858     return null;
860   }
861   @Perm(requires="unique(this) in alive",
862     ensures="unique(this) in alive")
863   protected IResult<LE> getLabeledResultAfter(ICFGNode node) {
864     return null;
866   }
867   @Perm(requires="unique(this) in alive",
868     ensures="unique(this) in alive")

```

```

869     protected IResult<LE> mergeLabeledResults(HashMap<ICFGNode, IResult<LE>> results) {
870         return null;
871     }
872 }
873 @Perm(requires="unique(this) in alive",
874 ensures="unique(this) in alive")
875 public IResult<LE> getLabeledResultsBefore(ASTNode node) {
876     return null;
877 }
878 }
879 @Perm(requires="unique(this) in alive",
880 ensures="unique(this) in alive")
881 protected IResult<LE> getLabeledResultBefore(ICFGNode node) {
882     return null;
883 }
884 }
885 @Perm(ensures="none(this) in alive")
886 MethodDeclaration getCurrentMethod() {
887     return null;
888 }
889 }
890 }ENDOFCLASS
891
892 @ClassStates({@State(name = "alive")})
893
894 class SingleResult {
895     @Perm(ensures="unique(this) in alive")
896     SingleResult() { }
897 }
898 }ENDOFCLASS
899
900 @ClassStates({@State(name = "alive")})
901
902 class ThisVariable {
903     @Perm(ensures="unique(this) in alive")
904     ThisVariable() { }
905 }
906
907 @Perm(requires="pure(this) in alive",
908 ensures="pure(this) in alive")
909 public boolean isImplicit() {
910     return 0;
911 }
912 }
913
914 @Perm(requires="unique(this) in alive",
915 ensures="unique(this) in alive")
916 public void explicitQualifier(Name qualifier) {
917 }
918 }
919
920 public String getKeyword() {
921     return null;
922 }
923 }
924 @Perm(requires="pure(this) in alive",
925 ensures="pure(this) in alive")
926 public boolean isQualified() {
927     return 0;
928 }
929 }
930 @Perm(requires="unique(this) in alive",
931 ensures="unique(this) in alive")
932 public ITypeBinding resolveType() {
933     return null;
934 }
935 }
936 @Perm(requires="unique(this) in alive",
937 ensures="unique(this) in alive")
938 public T dispatch(IVariableVisitor<T> visitor) {
939     return null;
940 }
941 }
942 @Perm(requires="pure(this) in alive",
943 ensures="pure(this) in alive")
944 public boolean isUnqualifiedThis() {
945     return 0;
946 }
947 }
948 }ENDOFCLASS

```

```

951 @ClassStates({@State(name = "alive")})
952
953 class AbstractTACAnalysisDriver {
954   @Perm(ensures="unique(this) in alive")
955   AbstractTACAnalysisDriver() { }
956
957   @Perm(requires="unique(this) in alive",
958     ensures="unique(this) in alive")
959   public void switchToMethod(MethodDeclaration methodDecl) {
960
961   }
962   @Perm(requires="immutable(this) in alive",
963     ensures="immutable(this) in alive")
964   public AnalysisDirection getAnalysisDirection() {
965     return null;
966   }
967
968   @Perm(requires="immutable(this) in alive",
969     ensures="immutable(this) in alive")
970   public Lattice<LE> getLattice(MethodDeclaration methodDeclaration) {
971     return null;
972   }
973 }
974
975 }ENDOFCLASS
976
977 @ClassStates({@State(name = "alive")})
978
979 class CompilationUnitTACs {
980   @Perm(ensures="unique(this) in alive")
981   CompilationUnitTACs() { }
982
983   @Perm(requires="share(this) in alive",
984     ensures="share(this) in alive")
985   EclipseTAC getMethodTAC(MethodDeclaration methodDecl) {
986     return null;
987   }
988 }
989
990 }ENDOFCLASS
991
992 @ClassStates({@State(name = "alive")})
993
994 class LabeledSingleResult {
995   @Perm(ensures="unique(this) in alive")
996   LabeledSingleResult() { }
997
998 }
999
1000 }ENDOFCLASS
1001
1002 @ClassStates({@State(name = "alive")})
1003
1004 class AbstractTransferFunction {
1005   @Perm(ensures="unique(this) in alive")
1006   AbstractTransferFunction() { }
1007
1008   @Perm(requires="immutable(this) in alive",
1009     ensures="immutable(this) in alive")
1010   public AnalysisDirection getAnalysisDirection() {
1011     return null;
1012   }
1013
1014   @Perm(requires="pure(this) in alive",
1015     ensures="pure(this) in alive")
1016   public ITACAnalysisContext getAnalysisContext() {
1017     return null;
1018   }
1019
1020   @Perm(requires="full(this) in alive",
1021     ensures="full(this) in alive")
1022   public void setAnalysisContext(ITACAnalysisContext analysisContext) {
1023
1024   }
1025
1026   public LE transferOver(ArrayInitInstruction instr, LE value) {
1027     return null;
1028   }
1029
1030   public LE transferOver2(BinaryOperation binop, LE value) {

```



```

1031     return null;
1032 }
1033
1034 public LE transferOver3(CastInstruction instr, LE value) {
1035     return null;
1036 }
1037
1038 public LE transferOver4(DotClassInstruction instr, LE value) {
1039     return null;
1040 }
1041
1042 public LE transferOver5(ConstructorCallInstruction instr, LE value) {
1043     return null;
1044 }
1045
1046 public LE transferOver6(CopyInstruction instr, LE value) {
1047     return null;
1048 }
1049
1050 public LE transferOver7(InstanceOfInstruction instr, LE value) {
1051     return null;
1052 }
1053
1054 public LE transferOver8(LoadLiteralInstruction instr, LE value) {
1055     return null;
1056 }
1057
1058 public LE transferOver9(LoadArrayInstruction instr, LE value) {
1059     return null;
1060 }
1061
1062 public LE transferOver10(LoadFieldInstruction instr, LE value) {
1063     return null;
1064 }
1065
1066 public LE transferOver11(MethodCallInstruction instr, LE value) {
1067     return null;
1068 }
1069
1070 public LE transferOver12(NewArrayInstruction instr, LE value) {
1071     return null;
1072 }
1073
1074 public LE transferOver13(NewObjectInstruction instr, LE value) {
1075     return null;
1076 }
1077
1078 public LE transferOver14(ReturnInstruction instr, LE value) {
1079     return null;
1080 }
1081
1082 public LE transferOver15(StoreArrayInstruction instr, LE value) {
1083     return null;
1084 }
1085
1086 public LE transferOver16(StoreFieldInstruction instr, LE value) {
1087     return null;
1088 }
1089
1090 public LE transferOver17(SourceVariableDeclaration instr, LE value) {
1091     return null;
1092 }
1093
1094 public LE transferOver18(SourceVariableRead instr, LE value) {
1095     return null;
1096 }

```

```
1113 }  
  
1115 public LE transferOver19(UnaryOperation unop, LE value) {  
1116     return null;  
  
1118 }  
  
1120 }ENDOFCLASS  
  
1122 @ClassStates({@State(name = "alive")})  
  
1124 class BinaryOperator {  
1125     @Perm(ensures="unique(this) in alive")  
1126     BinaryOperator() { }  
  
1128     @Perm(ensures="none(this) in alive")  
1129     public String toString() {  
1130         return null;  
  
1132     }  
  
1134 }ENDOFCLASS
```