Summary

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

Table 1: Pulse Analysis Summary

Classes	Methods	States	Unsatisfiable Clauses	Unreachable States	Possible concurrent Methods	Total. no. of pairs	No. of concurrent pairs	Percentage of concurrent Methods
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 BranchSensitiveTACAnalysis

Table 2: Methods Requires Clause Satisfiability

Method	Satisfiability
BranchSensitiveTACAnalysis	$\surd$

Table 3: State Transition Matrix

	alive
alive	$\uparrow$

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	$\surd$
getSourceString	$\surd$
isUnqualifiedSuper	$\surd$
isUnqualifiedThis	$\surd$

Table 8: State Transition Matrix

	alive
alive	$\uparrow$

Table 9: Methods Concurrency Matrix

	Variable	getSourceString	isUnqualifiedSuper	isUnqualifiedThis
Variable	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
getSourceString	$\nparallel$	$\parallel$	$\parallel$	$\parallel$
isUnqualifiedSuper	$\nparallel$	$\parallel$	$\parallel$	$\parallel$
isUnqualifiedThis	$\nparallel$	$\parallel$	$\parallel$	$\parallel$

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	$\surd$
analyzeMethod	$\surd$
doAccept	$\surd$
visit	$\surd$

Table 14: State Transition Matrix

	alive
alive	$\uparrow$

Table 15: Methods Concurrency Matrix

	SimpleInstructionVisitor	analyzeMethod	doAccept	visit
SimpleInstructionVisitor	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
analyzeMethod	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$
doAccept	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$
visit	$\nparallel$	$\parallel$	$\parallel$	$\parallel$

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	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥

transferOver6	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥
transferOver4	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥
transferOver7	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥
transferOver9	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥
transferOver10	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥
transferOver8	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥
transferOver11	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥
transferOver12	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥
transferOver13	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥
transferOver14	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥
transferOver15	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥
transferOver16	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥
transferOver17	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥
transferOver18	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	
transfer	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	
getLattice	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	
getAnalysisDirection	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	
setAnalysisContext	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	
transferOver19	∦	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	∥	

7 Lattice

Table 19: Methods Requires Clause Satisfiability

Method	Satisfiability
Lattice	$\surd$

Table 20: State Transition Matrix

	alive
alive	$\uparrow$

8 TypeVariable

Table 21: Methods Requires Clause Satisfiability

Method	Satisfiability
TypeVariable	$\surd$
getType	$\surd$
hashCode	$\surd$
equals	$\surd$
toString	$\surd$
resolveType	$\surd$
dispatch	$\surd$

Table 22: State Transition Matrix

	alive
alive	$\uparrow$

Table 23: Methods Concurrency Matrix

	TypeVariable	getType	hashCode	equals	toString	resolveType	dispatch
TypeVariable	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
getType	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$
hashCode	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$
equals	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$
toString	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$
resolveType	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$
dispatch	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$

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	$\surd$
getQualifier	$\surd$
isQualified	$\surd$
setQualifier	$\surd$
toString	$\surd$

Table 31: State Transition Matrix

	alive
alive	$\uparrow$

Table 32: Methods Concurrency Matrix

	KeywordVariable	getQualifier	isQualified	setQualifier	toString
KeywordVariable	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
getQualifier	$\nparallel$	$\parallel$	$\parallel$	$\nparallel$	$\parallel$
isQualified	$\nparallel$	$\parallel$	$\parallel$	$\nparallel$	$\parallel$
setQualifier	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
toString	$\nparallel$	$\parallel$	$\parallel$	$\nparallel$	$\parallel$

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	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$
implicitThisVariable	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$
implicitThisBinding	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$
findElementDeclarationByName	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
isDefaultBinding	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$

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	$\surd$
toString	$\surd$

Table 39: State Transition Matrix

	alive
alive	$\uparrow$

Table 40: Methods Concurrency Matrix

	UnaryOperator	toString
UnaryOperator	$\nparallel$	$\parallel$
toString	$\parallel$	$\parallel$

15 TempVariable

Table 41: Methods Requires Clause Satisfiability

Method	Satisfiability
TempVariable	$\surd$
getNode	$\surd$
dispatch	$\surd$
hashCode	$\surd$
equals	$\surd$
toString	$\surd$
getSourceString	$\surd$
resolveType	$\surd$

Table 42: State Transition Matrix

	alive
alive	$\uparrow$

Table 43: Methods Concurrency Matrix

	TempVariable	getNode	dispatch	hashCode	equals	toString	getSourceString	resolveType
TempVariable	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
getNode	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$
dispatch	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
hashCode	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
equals	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
toString	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$
getSourceString	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
resolveType	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$

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	$\surd$
transfer	$\surd$
deriveResult	$\surd$

Table 48: State Transition Matrix

	alive
alive	$\uparrow$

Table 49: Methods Concurrency Matrix

	BranchInsensitiveTACAnalysisDriver	transfer	deriveResult
BranchInsensitiveTACAnalysisDriver	$\nparallel$	$\nparallel$	$\nparallel$
transfer	$\nparallel$	$\nparallel$	$\nparallel$
deriveResult	$\nparallel$	$\nparallel$	$\parallel$

18 BranchSensitiveTACAnalysisDriver

Table 50: Methods Requires Clause Satisfiability

Method	Satisfiability
BranchSensitiveTACAnalysisDriver	$\surd$
transfer	$\surd$
deriveResult	$\surd$

Table 51: State Transition Matrix

	alive
alive	$\uparrow$

Table 52: Methods Concurrency Matrix

	BranchSensitiveTACAnalysisDriver	transfer	deriveResult
BranchSensitiveTACAnalysisDriver	$\nparallel$	$\nparallel$	$\nparallel$
transfer	$\nparallel$	$\nparallel$	$\nparallel$
deriveResult	$\nparallel$	$\nparallel$	$\parallel$

19 NewInstructionVisitor

Table 53: Methods Requires Clause Satisfiability

Method	Satisfiability
NewInstructionVisitor	$\surd$
getResult	$\surd$

Table 54: State Transition Matrix

	alive
alive	$\uparrow$

Table 55: Methods Concurrency Matrix

	NewInstructionVisitor	getResult
NewInstructionVisitor	$\nparallel$	$\nparallel$
getResult	$\nparallel$	$\parallel$

20 MotherFlowAnalysis

Table 56: Methods Requires Clause Satisfiability

Method	Satisfiability
MotherFlowAnalysis	$\surd$
getResultsAfter	$\surd$
mergeLabeledResult	$\surd$
checkNull	$\surd$
getResultsBefore	$\surd$
getLabeledResultsAfter	$\surd$
getLabeledResultAfter	$\surd$
mergeLabeledResults	$\surd$
getLabeledResultsBefore	$\surd$
getLabeledResultBefore	$\surd$
getCurrentMethod	$\surd$

Table 57: State Transition Matrix

	alive
alive	$\uparrow$

Table 58: Methods Concurrency Matrix

	MotherFlowAnalysis	getResultsAfter	mergeLabeledResult	checkNull	getResultsBefore	getLabeledResultsAfter	getLabeledResultAfter	mergeLabeledResults	getLabeledResultsBefore	getLabeledResultBefore	getCurrentMethod
MotherFlowAnalysis	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$
getResultsAfter	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$
mergeLabeledResult	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
checkNull	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
getResultsBefore	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$
getLabeledResultsAfter	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$
getLabeledResultAfter	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$
mergeLabeledResults	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$
getLabeledResultsBefore	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$
getLabeledResultBefore	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$
getCurrentMethod	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$

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	$\surd$
isImplicit	$\surd$
explicitQualifier	$\surd$
getKeyword	$\surd$
isQualified	$\surd$
resolveType	$\surd$
dispatch	$\surd$
isUnqualifiedThis	$\surd$

Table 62: State Transition Matrix

	alive
alive	$\uparrow$

Table 63: Methods Concurrency Matrix

	ThisVariable	isImplicit	explicitQualifier	getKeyword	isQualified	resolveType	dispatch	isUnqualifiedThis
ThisVariable	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
isImplicit	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\parallel$	$\nparallel$	$\nparallel$	$\parallel$
explicitQualifier	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
getKeyword	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
isQualified	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\parallel$	$\nparallel$	$\nparallel$	$\parallel$
resolveType	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
dispatch	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
isUnqualifiedThis	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\parallel$	$\nparallel$	$\nparallel$	$\parallel$

23 AbstractTACAnalysisDriver

Table 64: Methods Requires Clause Satisfiability

Method	Satisfiability
AbstractTACAnalysisDriver	$\surd$
switchToMethod	$\surd$
getAnalysisDirection	$\surd$
getLattice	$\surd$

Table 65: State Transition Matrix

	alive
alive	$\uparrow$

Table 66: Methods Concurrency Matrix

	AbstractTACAnalysisDriver	switchToMethod	getAnalysisDirection	getLattice
AbstractTACAnalysisDriver	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
switchToMethod	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
getAnalysisDirection	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$
getLattice	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$

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	$\surd$

Table 71: State Transition Matrix

	alive
alive	$\uparrow$

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	∦	∦	∦	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖

transferOver	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
transferOver2	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
transferOver3	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
transferOver4	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
transferOver5	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
transferOver6	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
transferOver7	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
transferOver8	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
transferOver9	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
transferOver10	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
transferOver11	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
transferOver12	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
transferOver13	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
transferOver14	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
transferOver15	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
transferOver16	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
transferOver17	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
transferOver18	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
transferOver19	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$

27 BinaryOperator

Table 75: Methods Requires Clause Satisfiability

Method	Satisfiability
BinaryOperator	$\surd$
toString	$\surd$

Table 76: State Transition Matrix

	alive
alive	$\uparrow$

Table 77: Methods Concurrency Matrix

	BinaryOperator	toString
BinaryOperator	$\nparallel$	$\parallel$
toString	$\parallel$	$\parallel$

28 Abbreviation

Table 78: Used Abbreviation

Symbol	Meaning
$\surd$	requires clause of the method is satisfiable
$\times$	requires clause of the method is unsatisfiable
$\uparrow$	The row-state can be transitioned to the column-state
$\times$	The row-state cannot be transitioned to the column-state
$\parallel$	The row-method can be possibly executed parallel with the column-method
$\nparallel$	The row-method cannot be executed parallel with the column-method

29 Annotated Version of Sequential Java Program generated by Sip4j

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

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


}ENDOFCLASS

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

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

@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public IVariableBinding getBinding() {
 return null;

}
@Perm(ensures="none(this) in alive")
 public boolean isCapturedFromOuterScope() {
 return 0;

}

 public T dispatch(IVariableVisitor<T> visitor) {
 return null;

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

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public boolean equals(Object obj) {
 return 0;

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

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

}

}ENDOFCLASS

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

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


 public String getSourceString() {
 return null;

}

 public boolean isUnqualifiedSuper() {
 return 0;

}
```

```
 public boolean isUnqualifiedThis() {
 return 0;

}

}ENDOFCLASS

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

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

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

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

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setAnalysisContext(ITACAnalysisContext analysisContext) {

}

 public IResult<LE> transferOver2(ArrayInitInstruction instr, List<ILabel> labels, LE value) {
 return null;

}

 public IResult<LE> transferOver4(BinaryOperation binop, List<ILabel> labels, LE value) {
 return null;

}

 public IResult<LE> transfer01(CastInstruction instr, List<ILabel> labels, LE value) {
 return null;

}

 public IResult<LE> transferOverload4(DotClassInstruction instr, List<ILabel> labels, LE value) {
 return null;

}

 public IResult<LE> transfer(ConstructorCallInstruction instr, List<ILabel> labels, LE value) {
 return null;

}

}ENDOFCLASS

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

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

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

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

}

 public void visit(ArrayInitInstruction instr) {

}

}ENDOFCLASS
```

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

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


 public SingletonLatticeElement transferOver(ArrayInitInstruction instr, SingletonLatticeElement value)
      {
 return null;

}

 public SingletonLatticeElement transferOver2(BinaryOperation binop, SingletonLatticeElement value) {
 return null;

}

 public SingletonLatticeElement transferOver3(CastInstruction instr, SingletonLatticeElement value) {
 return null;

}

 public SingletonLatticeElement transferOver5(ConstructorCallInstruction instr, SingletonLatticeElement
      value) {
 return null;

}

 public SingletonLatticeElement transferOver6(CopyInstruction instr, SingletonLatticeElement value) {
 return null;

}

 public SingletonLatticeElement transferOver4(DotClassInstruction instr, SingletonLatticeElement value)
      {
 return null;

}

 public SingletonLatticeElement transferOver7(InstanceofInstruction instr, SingletonLatticeElement value
      ) {
 return null;

}

 public SingletonLatticeElement transferOver9(LoadArrayInstruction instr, SingletonLatticeElement value)
       {
 return null;

}

 public SingletonLatticeElement transferOver10(LoadFieldInstruction instr, SingletonLatticeElement value
      ) {
 return null;

}

 public SingletonLatticeElement transferOver8(LoadLiteralInstruction instr, SingletonLatticeElement
      value) {
 return null;

}

 public SingletonLatticeElement transferOver11(MethodCallInstruction instr, SingletonLatticeElement
      value) {
 return null;

}

 public SingletonLatticeElement transferOver12(NewArrayInstruction instr, SingletonLatticeElement value)
       {
 return null;

}

 public SingletonLatticeElement transferOver13(NewObjectInstruction instr, SingletonLatticeElement value
      ) {
 return null;
```

```
}

 public SingletonLatticeElement transferOver14(ReturnInstruction instr, SingletonLatticeElement value) {
 return null;

}

 public SingletonLatticeElement transferOver15(StoreArrayInstruction instr, SingletonLatticeElement
      value) {
 return null;

}

 public SingletonLatticeElement transferOver16(StoreFieldInstruction instr, SingletonLatticeElement
      value) {
 return null;

}

 public SingletonLatticeElement transferOver17(SourceVariableDeclaration instr, SingletonLatticeElement
      value) {
 return null;

}

 public SingletonLatticeElement transferOver18(SourceVariableRead instr, SingletonLatticeElement value)
      {
 return null;

}

 public SingletonLatticeElement transfer(UnaryOperation unop, SingletonLatticeElement value) {
 return null;

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

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

}

 public void setAnalysisContext(ITACAnalysisContext analysisContext) {

}

 public SingletonLatticeElement transferOver19(UnaryOperation unop, SingletonLatticeElement value) {
 return null;

}

}ENDOFCLASS

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

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


}ENDOFCLASS

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

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

@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public ITypeBinding getType() {
 return null;

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

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public boolean equals(Object obj) {
 return 0;

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

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

}

 public T dispatch(IVariableVisitor<T> visitor) {
 return null;

}

}ENDOFCLASS

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

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

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

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

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setAnalysisContext(ITACAnalysisContext analysisContext) {

}

 public LE transfer(TACInstruction instr, LE value) {
 return null;

}

 public LE transferOver(ArrayInitInstruction instr, LE value) {
 return null;

}

 public LE transferOver2(BinaryOperation binop, LE value) {
 return null;

}

 public LE transferOver3(CastInstruction instr, LE value) {
 return null;

}

 public LE transferOver4(DotClassInstruction instr, LE value) {
 return null;

}

 public LE transferOver5(ConstructorCallInstruction instr, LE value) {
```

```
 return null;

}

 public LE transferOver6(CopyInstruction instr, LE value) {
 return null;

}

 public LE transferOver7(InstanceofInstruction instr, LE value) {
 return null;

}

 public LE transferOver8(LoadLiteralInstruction instr, LE value) {
 return null;

}

 public LE transferOver9(LoadArrayInstruction instr, LE value) {
 return null;

}

 public LE transferOver10(LoadFieldInstruction instr, LE value) {
 return null;

}

 public LE transferOver11(MethodCallInstruction instr, LE value) {
 return null;

}

 public LE transferOver12(NewArrayInstruction instr, LE value) {
 return null;

}

 public LE transferOver13(NewObjectInstruction instr, LE value) {
 return null;

}

 public LE transferOver14(ReturnInstruction instr, LE value) {
 return null;

}

 public LE transferOver15(StoreArrayInstruction instr, LE value) {
 return null;

}

 public LE transferOver16(StoreFieldInstruction instr, LE value) {
 return null;

}

 public LE transferOver17(SourceVariableDeclaration instr, LE value) {
 return null;

}

 public LE transferOver18(SourceVariableRead instr, LE value) {
 return null;

}

 public LE transferOver19(UnaryOperation unop, LE value) {
 return null;

}

}ENDOFCLASS

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

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

```
 public String getKeyword() {
 return null;

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

}

 public T dispatch(IVariableVisitor<T> visitor) {
 return null;

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

}

}ENDOFCLASS

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

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

@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public Name getQualifier() {
 return null;

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

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 protected void setQualifier(Name qualifier) {

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

}

}ENDOFCLASS

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

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

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

}

  boolean isStaticBinding(IBinding binding) {
 return 0;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public TACInstruction instruction(ASTNode astNode) {
 return null;

}
```

```
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 private TACInstruction createInstruction(ASTNode astNode) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Variable variable(ASTNode astNode) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 private Variable getVariable(IBinding binding) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 private ThisVariable getThisVariable(ThisExpression node) {
 return null;

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

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public SuperVariable superVariable(Name qualifier) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public SourceVariable sourceVariable(IVariableBinding binding) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public ThisVariable implicitThisVariable(IBinding accessedElement) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 private ITypeBinding implicitThisBinding(IBinding accessedElement) {
 return null;

}

 private ITypeBinding findElementDeclarationByName(IBinding genericAccessedElement, boolean isMethod,
      ITypeBinding type, boolean skipPrivate, boolean skipPackagePrivate) {
 return null;

}

  boolean isDefaultBinding(IBinding binding) {
 return 0;

}

}ENDOFCLASS

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

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


}ENDOFCLASS

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

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

```
UnaryOperator() {   }

@Perm(ensures="none(this) in alive")
 public String toString() {
 return null;

}

}ENDOFCLASS

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

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

@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public ASTNode getNode() {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public T dispatch(IVariableVisitor<T> visitor) {
 return null;

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

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public boolean equals(Object obj) {
 return 0;

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

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

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

}

}ENDOFCLASS

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

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

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public LE getResultsAfter(TACInstruction instr) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public LE getResultsBefore(TACInstruction instr) {
 return null;

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

```
 public IResult<LE> getLabeledResultsAfter(TACInstruction instr) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public IResult<LE> getLabeledResultsBefore(TACInstruction instr) {
 return null;

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public ASTNode getNode(Variable x, TACInstruction instruction) {
 return null;

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

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public ThisVariable getThisVariable(MethodDeclaration methodDecl) {
 return null;

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

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public SourceVariable getSourceVariable(IVariableBinding varBinding) {
 return null;

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

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public ThisVariable getImplicitThisVariable(IBinding accessedElement) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 protected IFlowAnalysisDefinition<LE> createTransferFunction(MethodDeclaration method) {
 return null;

}

}ENDOFCLASS

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

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

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public LE transfer(ASTNode astNode, LE incoming) {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 public IResult<LE> deriveResult(EclipseInstructionSequence seq, LE incoming, TACInstruction
      targetInstruction, boolean afterResult) {
 return null;

}
```

```
}ENDOFCLASS

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

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

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public IResult<LE> transfer(ASTNode astNode, List<ILabel> labels, LE value) {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 public IResult<LE> deriveResult(EclipseInstructionSequence seq, LE incoming, TACInstruction
      targetInstruction, boolean afterResult) {
 return null;

}

}ENDOFCLASS

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

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

@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 public TACInstruction getResult() {
 return null;

}

}ENDOFCLASS

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

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

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

}

 protected LE mergeLabeledResult(IResult<LE> labeledResult, ASTNode node) {
 return null;

}

  T checkNull(T o) {
 return null;

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

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

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 protected IResult<LE> getLabeledResultAfter(ICFGNode node) {
 return null;

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

```
 protected IResult<LE> mergeLabeledResults(HashMap<ICFGNode,IResult<LE>> results) {
 return null;

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

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 protected IResult<LE> getLabeledResultBefore(ICFGNode node) {
 return null;

}
@Perm(ensures="none(this) in alive")
  MethodDeclaration getCurrentMethod() {
 return null;

}

}ENDOFCLASS

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

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


}ENDOFCLASS

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

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

@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 public boolean isImplicit() {
 return 0;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public void explicitQualifier(Name qualifier) {

}

 public String getKeyword() {
 return null;

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

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

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public T dispatch(IVariableVisitor<T> visitor) {
 return null;

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

}

}ENDOFCLASS
```

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

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

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

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

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

}

}ENDOFCLASS

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

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

@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
  EclipseTAC getMethodTAC(MethodDeclaration methodDecl) {
 return null;

}

}ENDOFCLASS

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

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


}ENDOFCLASS

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

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

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

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

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public void setAnalysisContext(ITACAnalysisContext analysisContext) {

}

 public LE transferOver(ArrayInitInstruction instr, LE value) {
 return null;

}

 public LE transferOver2(BinaryOperation binop, LE value) {
  return null;

}

  public LE transferOver3(CastInstruction instr, LE value) {
  return null;

}

  public LE transferOver4(DotClassInstruction instr, LE value) {
  return null;

}

  public LE transferOver5(ConstructorCallInstruction instr, LE value) {
  return null;

}

  public LE transferOver6(CopyInstruction instr, LE value) {
  return null;

}

  public LE transferOver7(InstanceofInstruction instr, LE value) {
  return null;

}

  public LE transferOver8(LoadLiteralInstruction instr, LE value) {
  return null;

}

  public LE transferOver9(LoadArrayInstruction instr, LE value) {
  return null;

}

  public LE transferOver10(LoadFieldInstruction instr, LE value) {
  return null;

}

  public LE transferOver11(MethodCallInstruction instr, LE value) {
  return null;

}

  public LE transferOver12(NewArrayInstruction instr, LE value) {
  return null;

}

  public LE transferOver13(NewObjectInstruction instr, LE value) {
  return null;

}

  public LE transferOver14(ReturnInstruction instr, LE value) {
  return null;

}

  public LE transferOver15(StoreArrayInstruction instr, LE value) {
  return null;

}

  public LE transferOver16(StoreFieldInstruction instr, LE value) {
  return null;

}

  public LE transferOver17(SourceVariableDeclaration instr, LE value) {
  return null;

}

  public LE transferOver18(SourceVariableRead instr, LE value) {
  return null;
```

```
}

 public LE transferOver19(UnaryOperation unop, LE value) {
 return null;

}

}ENDOFCLASS

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

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

@Perm(ensures="none(this) in alive")
 public String toString() {
 return null;

}

}ENDOFCLASS
```