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
StoreFieldInstructionImpl	8	1	0	0	6	36	21	58
EclipseTACInstructionFactory	4	1	0	0	3	10	3	30
NewInstructionVisitor	5	1	0	0	1	15	1	7
StoreArrayInstructionImpl	7	1	0	0	5	28	15	54
AbstractStoreInstruction	3	1	0	0	2	6	3	50
CopyInstructionImpl	5	1	0	0	3	15	6	40
SourceVariableReadImpl	5	1	0	0	2	15	3	20
EclipseMergeHelper	2	1	0	0	0	3	0	0
LabeledSingleResult	1	1	0	0	0	1	0	0
EclipseSuperConstructorCallInstruction	7	1	0	0	6	28	11	39
AbstractTACInstruction	10	1	0	0	9	55	37	67
AbstractConstructorCallInstruction	3	1	0	0	2	6	2	33
EclipseBinaryAssignOperation	3	1	0	0	0	6	0	0
InstanceofInstructionImpl	5	1	0	0	0	15	0	0
ArrayInitInstructionImpl	4	1	0	0	0	10	0	0
EclipseLoadDesugaredLiteralInstruction	6	1	0	0	5	21	15	71
LoadLiteralInstructionImpl	8	1	0	0	6	36	21	58
EclipseInstructionSequence	6	1	0	0	3	21	6	29
NormalLabel	2	1	0	0	1	3	1	33
ResultfulInstruction	1	1	0	0	0	1	0	0
EnhancedForConditionInstructionImpl	4	1	0	0	0	10	0	0
EclipseBinaryInfixOperation	3	1	0	0	0	6	0	0
LoadFieldInstructionImpl	8	1	0	0	6	36	21	58
AbstractMethodCallInstruction	4	1	0	0	3	10	5	50
EclipseTAC	16	1	0	0	15	136	43	32
EclipseAbstractFieldAccess	3	1	0	0	3	6	5	83
AbstractAssignmentInstruction	9	1	0	0	8	45	31	69
TempVariable	1	1	0	0	0	1	0	0
EclipseNormalCallInstruction	7	1	0	0	6	28	6	21
ReturnInstructionImpl	3	1	0	0	0	6	0	0
EclipseSuperFieldAccess	6	1	0	0	5	21	9	43
NewObjectInstructionImpl	9	1	0	0	0	45	0	0
EclipseFieldDeclaration	6	1	0	0	5	21	10	48
EclipseBinaryDesugaredOperation	3	1	0	0	3	6	5	83

CastInstructionImpl	5	1	0	0	0	15	0	0
EclipseBrokenFieldAccess	6	1	0	0	5	21	9	43
NewArrayInstructionImpl	9	1	0	0	0	45	0	0
CompilationUnitTACs	2	1	0	0	0	3	0	0
DotClassInstructionImpl	4	1	0	0	0	10	0	0
EclipseImplicitFieldAccess	6	1	0	0	5	21	12	57
EclipseThisConstructorCallInstruction	7	1	0	0	6	28	15	54
UnaryOperationImpl	5	1	0	0	1	15	1	7
SourceVariableDeclarationImpl	7	1	0	0	2	28	3	11
EclipseSuperCallInstruction	6	1	0	0	5	21	5	24
AbstractBinaryOperation	4	1	0	0	4	10	6	60
LoadArrayInstructionImpl	6	1	0	0	0	21	0	0
ThisVariable	3	1	0	0	1	6	1	17
KeywordVariable	2	1	0	0	0	3	0	0
EclipseReferenceFieldAccess	6	1	0	0	5	21	9	43
Total Classes=49	255	49	0	0	142	976	341	35

Contents

1 StoreFieldInstructionImpl

Table 2: Methods Requires Clause Satisfiability

Method	Satisfiability
StoreFieldInstructionImpl	$\surd$
getDestinationObject	$\surd$
getAccessedObjectOperand	$\surd$
getFieldName	$\surd$
resolveFieldBinding	$\surd$
isStaticFieldAccess	$\surd$
transfer	$\surd$
toString	$\surd$

Table 3: State Transition Matrix

	alive
alive	$\uparrow$

Table 4: Methods Concurrency Matrix

	StoreFieldInstructionImpl	getDestinationObject	getAccessedObjectOperand	getFieldName	resolveFieldBinding	isStaticFieldAccess	transfer	toString
StoreFieldInstructionImpl	$\not\parallel$	$\not\parallel$	$\not\parallel$	$\not\parallel$	$\not\parallel$	$\not\parallel$	$\not\parallel$	$\not\parallel$
getDestinationObject	$\not\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\not\parallel$	$\parallel$
getAccessedObjectOperand	$\not\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\not\parallel$	$\parallel$
getFieldName	$\not\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\not\parallel$	$\parallel$
resolveFieldBinding	$\not\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\not\parallel$	$\parallel$
isStaticFieldAccess	$\not\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\not\parallel$	$\parallel$
transfer	$\not\parallel$	$\not\parallel$	$\not\parallel$	$\not\parallel$	$\not\parallel$	$\not\parallel$	$\not\parallel$	$\not\parallel$
toString	$\not\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\not\parallel$	$\parallel$

2 EclipseTACInstructionFactory

Table 5: Methods Requires Clause Satisfiability

Method	Satisfiability
EclipseTACInstructionFactory	√
createStore	√
create	√
isLoad	√

Table 6: State Transition Matrix

	alive
alive	↑

Table 7: Methods Concurrency Matrix

	EclipseTACInstructionFactory	createStore	create	isLoad
EclipseTACInstructionFactory	∦	∦	∦	∦
createStore	∦	∥	∥	∥
create	∦	∥	∦	∦
isLoad	∦	∥	∦	∦

3 NewInstructionVisitor

Table 8: Methods Requires Clause Satisfiability

Method	Satisfiability
NewInstructionVisitor	√
visit	√
setResult	√
getResult	√
noResult	√

Table 9: State Transition Matrix

	alive
alive	↑

Table 10: Methods Concurrency Matrix

	NewInstructionVisitor	visit	setResult	getResult	noResult
NewInstructionVisitor	∦	∦	∦	∦	∦
visit	∦	∦	∦	∦	∦
setResult	∦	∦	∦	∦	∦
getResult	∦	∦	∦	∥	∦
noResult	∦	∦	∦	∦	∦

4 StoreArrayInstructionImpl

Table 11: Methods Requires Clause Satisfiability

Method	Satisfiability
StoreArrayInstructionImpl	√
getDestinationArray	√
getTargetNode	√
getAccessedArrayOperand	√
getArrayIndex	√
transfer	√
toString	√

Table 12: State Transition Matrix

	alive
alive	↑

Table 13: Methods Concurrency Matrix

	StoreArrayInstructionImpl	getDestinationArray	getTargetNode	getAccessedArrayOperand	getArrayIndex	transfer	toString
StoreArrayInstructionImpl	∦	∦	∦	∦	∦	∦	∦
getDestinationArray	∦	∥	∥	∥	∥	∦	∥
getTargetNode	∦	∥	∥	∥	∥	∦	∥
getAccessedArrayOperand	∦	∥	∥	∥	∥	∦	∥
getArrayIndex	∦	∥	∥	∥	∥	∦	∥
transfer	∦	∦	∦	∦	∦	∦	∦
toString	∦	∥	∥	∥	∥	∦	∥

5 AbstractStoreInstruction

Table 14: Methods Requires Clause Satisfiability

Method	Satisfiability
AbstractStoreInstruction	√
getSourceOperand	√
getResultVariable	√

Table 15: State Transition Matrix

	alive
alive	↑

Table 16: Methods Concurrency Matrix

	AbstractStoreInstruction	getSourceOperand	getResultVariable
AbstractStoreInstruction	∦	∦	∦
getSourceOperand	∦	∥	∥
getResultVariable	∦	∥	∥

6 CopyInstructionImpl

Table 17: Methods Requires Clause Satisfiability

Method	Satisfiability
CopyInstructionImpl	√
getOperand	√
getResultVariable	√
transfer	√
toString	√

Table 18: State Transition Matrix

	alive
alive	↑

Table 19: Methods Concurrency Matrix

	CopyInstructionImpl	getOperand	getResultVariable	transfer	toString
CopyInstructionImpl	∦	∦	∦	∦	∦
getOperand	∦	‖	‖	∦	‖
getResultVariable	∦	‖	‖	∦	‖
transfer	∦	∦	∦	∦	∦
toString	∦	‖	‖	∦	‖

7 SourceVariableReadImpl

Table 20: Methods Requires Clause Satisfiability

Method	Satisfiability
SourceVariableReadImpl	√
getVariable	√
getResultVariable	√
transfer	√
toString	√

Table 21: State Transition Matrix

	alive
alive	↑

Table 22: Methods Concurrency Matrix

	SourceVariableReadImpl	getVariable	getResultVariable	transfer	toString
SourceVariableReadImpl	∦	∦	∦	∦	∦
getVariable	∦	‖	‖	∦	∦
getResultVariable	∦	‖	‖	∦	∦
transfer	∦	∦	∦	∦	∦
toString	∦	∦	∦	∦	∦

8 EclipseMergeHelper

Table 23: Methods Requires Clause Satisfiability

Method	Satisfiability
EclipseMergeHelper	√
transfer	√

Table 24: State Transition Matrix

	alive
alive	↑

Table 25: Methods Concurrency Matrix

	EclipseMergeHelper	transfer
EclipseMergeHelper	∦	∦
transfer	∦	∦

9 LabeledSingleResult

Table 26: Methods Requires Clause Satisfiability

Method	Satisfiability
LabeledSingleResult	√

Table 27: State Transition Matrix

	alive
alive	↑

10 EclipseSuperConstructorCallInstruction

Table 28: Methods Requires Clause Satisfiability

Method	Satisfiability
EclipseSuperConstructorCallInstruction	√
getConstructionObject	√
isSuperCall	√
getArgOperands	√
getEnclosingInstanceSpecifier	√
hasEnclosingInstanceSpecifier	√
resolveBinding	√

Table 29: State Transition Matrix

	alive
alive	↑

Table 30: Methods Concurrency Matrix

	EclipseSuperConstructorCallInstruction	getConstructionObject	isSuperCall	getArgOperands	getEnclosingInstanceSpecifier	hasEnclosingInstanceSpecifier	resolveBinding
EclipseSuperConstructorCallInstruction	∦	∦	∦	∦	∦	∦	∦
getConstructionObject	∦	‖	‖	‖	‖	‖	‖
isSuperCall	∦	‖	‖	‖	‖	‖	‖
getArgOperands	∦	‖	‖	∦	∦	∦	∦
getEnclosingInstanceSpecifier	∦	‖	‖	∦	∦	∦	∦
hasEnclosingInstanceSpecifier	∦	‖	‖	∦	∦	∦	∦
resolveBinding	∦	‖	‖	∦	∦	∦	∦

11 AbstractTACInstruction

Table 31: Methods Requires Clause Satisfiability

Method	Satisfiability
AbstractTACInstruction	√
superVariable	√
variables	√
variable	√
targetVariable	√
typeVariable	√
implicitThisVariable	√
receiverVariable	√
getNode	√
argsString	√

Table 32: State Transition Matrix

	alive
alive	↑

Table 33: Methods Concurrency Matrix

	AbstractTACInstruction	superVariable	variables	variable	targetVariable	typeVariable	implicitThisVariable	receiverVariable	getNode	argsString
AbstractTACInstruction	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
superVariable	∦	‖	∦	‖	‖	‖	‖	‖	‖	‖
variables	∦	∦	∦	∦	∦	∦	∦	∦	∦	‖
variable	∦	‖	∦	‖	‖	‖	‖	‖	‖	‖
targetVariable	∦	‖	∦	‖	‖	‖	‖	‖	‖	‖
typeVariable	∦	‖	∦	‖	‖	‖	‖	‖	‖	‖
implicitThisVariable	∦	‖	∦	‖	‖	‖	‖	‖	‖	‖
receiverVariable	∦	‖	∦	‖	‖	‖	‖	‖	‖	‖
getNode	∦	‖	∦	‖	‖	‖	‖	‖	‖	‖
argsString	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖

12 AbstractConstructorCallInstruction

Table 34: Methods Requires Clause Satisfiability

Method	Satisfiability
AbstractConstructorCallInstruction	√
transfer	√
toString	√

Table 35: State Transition Matrix

	alive
alive	↑

Table 36: Methods Concurrency Matrix

	AbstractConstructorCallInstruction	transfer	toString
AbstractConstructorCallInstruction	∦	∦	∦
transfer	∦	∦	‖
toString	∦	‖	‖

13 EclipseBinaryAssignOperation

Table 37: Methods Requires Clause Satisfiability

Method	Satisfiability
EclipseBinaryAssignOperation	√
getOperand1	√
getOperand2	√

Table 38: State Transition Matrix

	alive
alive	↑

Table 39: Methods Concurrency Matrix

	EclipseBinaryAssignOperation	getOperand1	getOperand2
EclipseBinaryAssignOperation	∦	∦	∦
getOperand1	∦	∦	∦
getOperand2	∦	∦	∦

14 InstanceofInstructionImpl

Table 40: Methods Requires Clause Satisfiability

Method	Satisfiability
InstanceofInstructionImpl	√
getTestedTypeNode	√
getOperand	√
transfer	√
toString	√

Table 41: State Transition Matrix

	alive
alive	↑

Table 42: Methods Concurrency Matrix

	InstanceofInstructionImpl	getTestedTypeNode	getOperand	transfer	toString
InstanceofInstructionImpl	∦	∦	∦	∦	∦
getTestedTypeNode	∦	∦	∦	∦	∦
getOperand	∦	∦	∦	∦	∦
transfer	∦	∦	∦	∦	∦
toString	∦	∦	∦	∦	∦

15 ArrayInitInstructionImpl

Table 43: Methods Requires Clause Satisfiability

Method	Satisfiability
ArrayInitInstructionImpl	√
getInitOperands	√
transfer	√
toString	√

Table 44: State Transition Matrix

	alive
alive	↑

Table 45: Methods Concurrency Matrix

	ArrayInitInstructionImpl	getInitOperands	transfer	toString
ArrayInitInstructionImpl	∦	∦	∦	∦
getInitOperands	∦	∦	∦	∦
transfer	∦	∦	∦	∦
toString	∦	∦	∦	∦

16 EclipseLoadDesugaredLiteralInstruction

Table 46: Methods Requires Clause Satisfiability

Method	Satisfiability
EclipseLoadDesugaredLiteralInstruction	√
getLiteral	√
isNonNullString	√
isNull	√
isNumber	√
isPrimitive	√

Table 47: State Transition Matrix

	alive
alive	↑

Table 48: Methods Concurrency Matrix

	EclipseLoadDesugaredLiteralInstruction	getLiteral	isNonNullString	isNull	isNumber	isPrimitive
EclipseLoadDesugaredLiteralInstruction	∦	∦	∦	∦	∦	∦
getLiteral	∦	‖	‖	‖	‖	‖
isNonNullString	∦	‖	‖	‖	‖	‖
isNull	∦	‖	‖	‖	‖	‖
isNumber	∦	‖	‖	‖	‖	‖
isPrimitive	∦	‖	‖	‖	‖	‖

17 LoadLiteralInstructionImpl

Table 49: Methods Requires Clause Satisfiability

Method	Satisfiability
LoadLiteralInstructionImpl	√
getLiteral	√
isPrimitive	√
isNumber	√
isNull	√
isNonNullString	√
transfer	√
toString	√

Table 50: State Transition Matrix

	alive
alive	↑

Table 51: Methods Concurrency Matrix

	LoadLiteralInstructionImpl	getLiteral	isPrimitive	isNumber	isNull	isNonNullString	transfer	toString
LoadLiteralInstructionImpl	∦	∦	∦	∦	∦	∦	∦	∦
getLiteral	∦	‖	‖	‖	‖	‖	∦	‖
isPrimitive	∦	‖	‖	‖	‖	‖	∦	‖
isNumber	∦	‖	‖	‖	‖	‖	∦	‖
isNull	∦	‖	‖	‖	‖	‖	∦	‖
isNonNullString	∦	‖	‖	‖	‖	‖	∦	‖
transfer	∦	∦	∦	∦	∦	∦	∦	∦
toString	∦	‖	‖	‖	‖	‖	∦	‖

18 EclipseInstructionSequence

Table 52: Methods Requires Clause Satisfiability

Method	Satisfiability
EclipseInstructionSequence	√
getResultVariable	√
getUseAsResult	√
getInstructions	√
transfer	√
deriveResult	√

Table 53: State Transition Matrix

	alive
alive	↑

Table 54: Methods Concurrency Matrix

	EclipseInstructionSequence	getResultVariable	getUseAsResult	getInstructions	transfer	deriveResult
EclipseInstructionSequence	∦	∦	∦	∦	∦	∦
getResultVariable	∦	‖	‖	‖	∦	∦
getUseAsResult	∦	‖	‖	‖	∦	∦
getInstructions	∦	‖	‖	‖	∦	∦
transfer	∦	∦	∦	∦	∦	∦
deriveResult	∦	∦	∦	∦	∦	∦

19 NormalLabel

Table 55: Methods Requires Clause Satisfiability

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

Table 56: State Transition Matrix

	alive
alive	$\uparrow$

Table 57: Methods Concurrency Matrix

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

20 ResultfulInstruction

Table 58: Methods Requires Clause Satisfiability

Method	Satisfiability
ResultfulInstruction	√

Table 59: State Transition Matrix

	alive
alive	↑

21 EnhancedForConditionInstructionImpl

Table 60: Methods Requires Clause Satisfiability

Method	Satisfiability
EnhancedForConditionInstructionImpl	√
getIteratedOperand	√
transfer	√
toString	√

Table 61: State Transition Matrix

	alive
alive	↑

Table 62: Methods Concurrency Matrix

	EnhancedForConditionInstructionImpl	getIteratedOperand	transfer	toString
EnhancedForConditionInstructionImpl	∦	∦	∦	∦
getIteratedOperand	∦	∦	∦	∦
transfer	∦	∦	∦	∦
toString	∦	∦	∦	∦

22 EclipseBinaryInfixOperation

Table 63: Methods Requires Clause Satisfiability

Method	Satisfiability
EclipseBinaryInfixOperation	√
getOperand1	√
getOperand2	√

Table 64: State Transition Matrix

	alive
alive	↑

Table 65: Methods Concurrency Matrix

	EclipseBinaryInfixOperation	getOperand1	getOperand2
EclipseBinaryInfixOperation	∦	∦	∦
getOperand1	∦	∦	∦
getOperand2	∦	∦	∦

23 LoadFieldInstructionImpl

Table 66: Methods Requires Clause Satisfiability

Method	Satisfiability
LoadFieldInstructionImpl	√
getFieldName	√
resolveFieldBinding	√
getSourceObject	√
getAccessedObjectOperand	√
isStaticFieldAccess	√
transfer	√
toString	√

Table 67: State Transition Matrix

	alive
alive	↑

Table 68: Methods Concurrency Matrix

	LoadFieldInstructionImpl	getFieldName	resolveFieldBinding	getSourceObject	getAccessedObjectOperand	isStaticFieldAccess	transfer	toString
LoadFieldInstructionImpl	∦	∦	∦	∦	∦	∦	∦	∦
getFieldName	∦	‖	‖	‖	‖	‖	∦	‖
resolveFieldBinding	∦	‖	‖	‖	‖	‖	∦	‖
getSourceObject	∦	‖	‖	‖	‖	‖	∦	‖
getAccessedObjectOperand	∦	‖	‖	‖	‖	‖	∦	‖
isStaticFieldAccess	∦	‖	‖	‖	‖	‖	∦	‖
transfer	∦	∦	∦	∦	∦	∦	∦	∦
toString	∦	‖	‖	‖	‖	‖	∦	‖

24 AbstractMethodCallInstruction

Table 69: Methods Requires Clause Satisfiability

Method	Satisfiability
AbstractMethodCallInstruction	√
isStaticMethodCall	√
transfer	√
toString	√

Table 70: State Transition Matrix

	alive
alive	↑

Table 71: Methods Concurrency Matrix

	AbstractMethodCallInstruction	isStaticMethodCall	transfer	toString
AbstractMethodCallInstruction	⫫	⫫	⫫	⫫
isStaticMethodCall	⫫	‖	‖	‖
transfer	⫫	‖	⫫	‖
toString	⫫	‖	‖	‖

25 EclipseTAC

Table 72: Methods Requires Clause Satisfiability

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

Table 73: State Transition Matrix

	alive
alive	↑

Table 74: Methods Concurrency Matrix

	EclipseTAC	isStaticBinding	implicitThisVariable	resolveThisType	implicitThisBinding	findElementDeclarationByName	isDefaultBinding	instruction	variable	getVariable	getThisVariable	createInstruction	sourceVariable	typeVariable	thisVariable	superVariable
EclipseTAC	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦	∦
isStaticBinding	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
implicitThisVariable	∦	‖	∦	∦	∦	‖	‖	∦	∦	∦	∦	∦	∦	∦	∦	∦
resolveThisType	∦	‖	∦	∦	∦	‖	‖	∦	∦	∦	∦	∦	∦	∦	∦	∦
implicitThisBinding	∦	‖	∦	∦	∦	‖	‖	∦	∦	∦	∦	∦	∦	∦	∦	∦
findElementDeclarationByName	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
isDefaultBinding	∦	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖	‖
instruction	∦	‖	∦	∦	∦	‖	‖	∦	∦	∦	∦	∦	∦	∦	∦	∦
variable	∦	‖	∦	∦	∦	‖	‖	∦	∦	∦	∦	∦	∦	∦	∦	∦

getVariable	∦	‖	∦	∦	∦	‖	‖	∦	∦	∦	∦	∦	∦	∦	∦	∦
getThisVariable	∦	‖	∦	∦	∦	‖	‖	∦	∦	∦	∦	∦	∦	∦	∦	∦
createInstruction	∦	‖	∦	∦	∦	‖	‖	∦	∦	∦	∦	‖	∦	∦	∦	∦
sourceVariable	∦	‖	∦	∦	∦	‖	‖	∦	∦	∦	∦	∦	∦	∦	∦	∦
typeVariable	∦	‖	∦	∦	∦	‖	‖	∦	∦	∦	∦	∦	∦	∦	∦	∦
thisVariable	∦	‖	∦	∦	∦	‖	‖	∦	∦	∦	∦	∦	∦	∦	∦	∦
superVariable	∦	‖	∦	∦	∦	‖	‖	∦	∦	∦	∦	∦	∦	∦	∦	∦

26 EclipseAbstractFieldAccess

Table 75: Methods Requires Clause Satisfiability

Method	Satisfiability
EclipseAbstractFieldAccess	√
isStaticFieldAccess	√
getAccessedObject	√

Table 76: State Transition Matrix

	alive
alive	↑

Table 77: Methods Concurrency Matrix

	EclipseAbstractFieldAccess	isStaticFieldAccess	getAccessedObject
EclipseAbstractFieldAccess	⊁	‖	‖
isStaticFieldAccess	‖	‖	‖
getAccessedObject	‖	‖	‖

27 AbstractAssignmentInstruction

Table 78: Methods Requires Clause Satisfiability

Method	Satisfiability
AbstractAssignmentInstruction	$\surd$
createTemp	$\surd$
getTarget	$\surd$
defaultVariable	$\surd$
setTarget	$\surd$
checkIfCopyNeeded	$\surd$
branch	$\surd$
assigns	$\surd$
getResultVariable	$\surd$

Table 79: State Transition Matrix

	alive
alive	$\uparrow$

Table 80: Methods Concurrency Matrix

	AbstractAssignmentInstruction	createTemp	getTarget	defaultVariable	setTarget	checkIfCopyNeeded	branch	assigns	getResultVariable
AbstractAssignmentInstruction	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
createTemp	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
getTarget	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
defaultVariable	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
setTarget	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
checkIfCopyNeeded	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
branch	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
assigns	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
getResultVariable	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$

28 TempVariable

Table 81: Methods Requires Clause Satisfiability

Method	Satisfiability
TempVariable	√

Table 82: State Transition Matrix

	alive
alive	↑

29 EclipseNormalCallInstruction

Table 83: Methods Requires Clause Satisfiability

Method	Satisfiability
EclipseNormalCallInstruction	$\surd$
isSuperCall	$\surd$
getReceiverOperand	$\surd$
resolveBinding	$\surd$
getArgOperands	$\surd$
getMethodName	$\surd$
toString	$\surd$

Table 84: State Transition Matrix

	alive
alive	$\uparrow$

Table 85: Methods Concurrency Matrix

	EclipseNormalCallInstruction	isSuperCall	getReceiverOperand	resolveBinding	getArgOperands	getMethodName	toString
EclipseNormalCallInstruction	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
isSuperCall	$\nparallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$	$\parallel$
getReceiverOperand	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
resolveBinding	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
getArgOperands	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
getMethodName	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$
toString	$\nparallel$	$\parallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$	$\nparallel$

30 ReturnInstructionImpl

Table 86: Methods Requires Clause Satisfiability

Method	Satisfiability
ReturnInstructionImpl	√
getReturnedVariable	√
transfer	√

Table 87: State Transition Matrix

	alive
alive	↑

Table 88: Methods Concurrency Matrix

	ReturnInstructionImpl	getReturnedVariable	transfer
ReturnInstructionImpl	∦	∦	∦
getReturnedVariable	∦	∦	∦
transfer	∦	∦	∦

31 EclipseSuperFieldAccess

Table 89: Methods Requires Clause Satisfiability

Method	Satisfiability
EclipseSuperFieldAccess	√
getFieldName	√
resolveFieldBinding	√
isImplicitThisAccess	√
isExplicitSuperAccess	√
getAccessedInstanceInternal	√

Table 90: State Transition Matrix

	alive
alive	↑

Table 91: Methods Concurrency Matrix

	EclipseSuperFieldAccess	getFieldName	resolveFieldBinding	isImplicitThisAccess	isExplicitSuperAccess	getAccessedInstanceInternal
EclipseSuperFieldAccess	⫮	⫮	⫮	⫮	⫮	⫮
getFieldName	⫮	⫮	⫮	‖	‖	⫮
resolveFieldBinding	⫮	⫮	⫮	‖	‖	⫮
isImplicitThisAccess	⫮	‖	‖	‖	‖	‖
isExplicitSuperAccess	⫮	‖	‖	‖	‖	‖
getAccessedInstanceInternal	⫮	⫮	⫮	‖	‖	⫮

32 NewObjectInstructionImpl

Table 92: Methods Requires Clause Satisfiability

Method	Satisfiability
NewObjectInstructionImpl	√
resolveBinding	√
isAnonClassType	√
getArgOperands	√
resolveInstantiatedType	√
hasOuterObjectSpecifier	√
getOuterObjectSpecifierOperand	√
transfer	√
toString	√

Table 93: State Transition Matrix

	alive
alive	↑

Table 94: Methods Concurrency Matrix

	NewObjectInstructionImpl	resolveBinding	isAnonClassType	getArgOperands	resolveInstantiatedType	hasOuterObjectSpecifier	getOuterObjectSpecifierOperand	transfer	toString
NewObjectInstructionImpl	∦	∦	∦	∦	∦	∦	∦	∦	∦
resolveBinding	∦	∦	∦	∦	∦	∦	∦	∦	∦
isAnonClassType	∦	∦	∦	∦	∦	∦	∦	∦	∦
getArgOperands	∦	∦	∦	∦	∦	∦	∦	∦	∦
resolveInstantiatedType	∦	∦	∦	∦	∦	∦	∦	∦	∦
hasOuterObjectSpecifier	∦	∦	∦	∦	∦	∦	∦	∦	∦
getOuterObjectSpecifierOperand	∦	∦	∦	∦	∦	∦	∦	∦	∦
transfer	∦	∦	∦	∦	∦	∦	∦	∦	∦
toString	∦	∦	∦	∦	∦	∦	∦	∦	∦

33 EclipseFieldDeclaration

Table 95: Methods Requires Clause Satisfiability

Method	Satisfiability
EclipseFieldDeclaration	√
getFieldName	√
isExplicitSuperAccess	√
isImplicitThisAccess	√
resolveFieldBinding	√
getAccessedInstanceInternal	√

Table 96: State Transition Matrix

	alive
alive	↑

Table 97: Methods Concurrency Matrix

	EclipseFieldDeclaration	getFieldName	isExplicitSuperAccess	isImplicitThisAccess	resolveFieldBinding	getAccessedInstanceInternal
EclipseFieldDeclaration	∦	∦	∦	∦	∦	∦
getFieldName	∦	∦	‖	‖	∦	∦
isExplicitSuperAccess	∦	‖	‖	‖	‖	‖
isImplicitThisAccess	∦	‖	‖	‖	‖	‖
resolveFieldBinding	∦	∦	‖	‖	∦	∦
getAccessedInstanceInternal	∦	∦	‖	‖	∦	‖

34 EclipseBinaryDesugaredOperation

Table 98: Methods Requires Clause Satisfiability

Method	Satisfiability
EclipseBinaryDesugaredOperation	√
getOperand1	√
getOperand2	√

Table 99: State Transition Matrix

	alive
alive	↑

Table 100: Methods Concurrency Matrix

	EclipseBinaryDesugaredOperation	getOperand1	getOperand2
EclipseBinaryDesugaredOperation	#	||	||
getOperand1	||	||	||
getOperand2	||	||	||

35 CastInstructionImpl

Table 101: Methods Requires Clause Satisfiability

Method	Satisfiability
CastInstructionImpl	√
getCastToTypeNode	√
getOperand	√
transfer	√
toString	√

Table 102: State Transition Matrix

	alive
alive	↑

Table 103: Methods Concurrency Matrix

	CastInstructionImpl	getCastToTypeNode	getOperand	transfer	toString
CastInstructionImpl	∦	∦	∦	∦	∦
getCastToTypeNode	∦	∦	∦	∦	∦
getOperand	∦	∦	∦	∦	∦
transfer	∦	∦	∦	∦	∦
toString	∦	∦	∦	∦	∦

36 EclipseBrokenFieldAccess

Table 104: Methods Requires Clause Satisfiability

Method	Satisfiability
EclipseBrokenFieldAccess	√
getFieldName	√
resolveFieldBinding	√
isImplicitThisAccess	√
isExplicitSuperAccess	√
getAccessedInstanceInternal	√

Table 105: State Transition Matrix

	alive
alive	↑

Table 106: Methods Concurrency Matrix

	EclipseBrokenFieldAccess	getFieldName	resolveFieldBinding	isImplicitThisAccess	isExplicitSuperAccess	getAccessedInstanceInternal
EclipseBrokenFieldAccess	⫫	⫫	⫫	⫫	⫫	⫫
getFieldName	⫫	⫫	⫫	‖	‖	⫫
resolveFieldBinding	⫫	⫫	⫫	‖	‖	⫫
isImplicitThisAccess	⫫	‖	‖	‖	‖	‖
isExplicitSuperAccess	⫫	‖	‖	‖	‖	‖
getAccessedInstanceInternal	⫫	⫫	⫫	‖	‖	⫫

37 NewArrayInstructionImpl

Table 107: Methods Requires Clause Satisfiability

Method	Satisfiability
NewArrayInstructionImpl	√
getArrayType	√
getDimensionOperands	√
getUnallocated	√
isInitialized	√
getDimensions	√
getInitOperand	√
transfer	√
toString	√

Table 108: State Transition Matrix

	alive
alive	↑

Table 109: Methods Concurrency Matrix

	NewArrayInstructionImpl	getArrayType	getDimensionOperands	getUnallocated	isInitialized	getDimensions	getInitOperand	transfer	toString
NewArrayInstructionImpl	∦	∦	∦	∦	∦	∦	∦	∦	∦
getArrayType	∦	∦	∦	∦	∦	∦	∦	∦	∦
getDimensionOperands	∦	∦	∦	∦	∦	∦	∦	∦	∦
getUnallocated	∦	∦	∦	∦	∦	∦	∦	∦	∦
isInitialized	∦	∦	∦	∦	∦	∦	∦	∦	∦
getDimensions	∦	∦	∦	∦	∦	∦	∦	∦	∦
getInitOperand	∦	∦	∦	∦	∦	∦	∦	∦	∦
transfer	∦	∦	∦	∦	∦	∦	∦	∦	∦
toString	∦	∦	∦	∦	∦	∦	∦	∦	∦

38 CompilationUnitTACs

Table 110: Methods Requires Clause Satisfiability

Method	Satisfiability
CompilationUnitTACs	√
getMethodTAC	√

Table 111: State Transition Matrix

	alive
alive	↑

Table 112: Methods Concurrency Matrix

	CompilationUnitTACs	getMethodTAC
CompilationUnitTACs	∦	∦
getMethodTAC	∦	∦

39 DotClassInstructionImpl

Table 113: Methods Requires Clause Satisfiability

Method	Satisfiability
DotClassInstructionImpl	√
getTypeNode	√
transfer	√
toString	√

Table 114: State Transition Matrix

	alive
alive	↑

Table 115: Methods Concurrency Matrix

	DotClassInstructionImpl	getTypeNode	transfer	toString
DotClassInstructionImpl	∦	∦	∦	∦
getTypeNode	∦	∦	∦	∦
transfer	∦	∦	∦	∦
toString	∦	∦	∦	∦

40 EclipseImplicitFieldAccess

Table 116: Methods Requires Clause Satisfiability

Method	Satisfiability
EclipseImplicitFieldAccess	√
getFieldName	√
resolveFieldBinding	√
isImplicitThisAccess	√
isExplicitSuperAccess	√
getAccessedInstanceInternal	√

Table 117: State Transition Matrix

	alive
alive	↑

Table 118: Methods Concurrency Matrix

	EclipseImplicitFieldAccess	getFieldName	resolveFieldBinding	isImplicitThisAccess	isExplicitSuperAccess	getAccessedInstanceInternal
EclipseImplicitFieldAccess	∦	∦	∦	∦	∦	∦
getFieldName	∦	‖	∦	‖	‖	‖
resolveFieldBinding	∦	∦	∦	‖	‖	∦
isImplicitThisAccess	∦	‖	‖	‖	‖	‖
isExplicitSuperAccess	∦	‖	‖	‖	‖	‖
getAccessedInstanceInternal	∦	‖	∦	‖	‖	‖

41 EclipseThisConstructorCallInstruction

Table 119: Methods Requires Clause Satisfiability

Method	Satisfiability
EclipseThisConstructorCallInstruction	√
getConstructionObject	√
resolveBinding	√
isSuperCall	√
getArgOperands	√
hasEnclosingInstanceSpecifier	√
getEnclosingInstanceSpecifier	√

Table 120: State Transition Matrix

	alive
alive	↑

Table 121: Methods Concurrency Matrix

	EclipseThisConstructorCallInstruction	getConstructionObject	resolveBinding	isSuperCall	getArgOperands	hasEnclosingInstanceSpecifier	getEnclosingInstanceSpecifier
EclipseThisConstructorCallInstruction	∦	∦	∦	∦	∦	∦	∦
getConstructionObject	∦	∦	∦	‖	∦	‖	‖
resolveBinding	∦	∦	∦	‖	∦	‖	‖
isSuperCall	∦	‖	‖	‖	‖	‖	‖
getArgOperands	∦	∦	∦	‖	∦	‖	‖
hasEnclosingInstanceSpecifier	∦	‖	‖	‖	‖	‖	‖
getEnclosingInstanceSpecifier	∦	‖	‖	‖	‖	‖	‖

42 UnaryOperationImpl

Table 122: Methods Requires Clause Satisfiability

Method	Satisfiability
UnaryOperationImpl	√
getOperand	√
getOperator	√
transfer	√
toString	√

Table 123: State Transition Matrix

	alive
alive	↑

Table 124: Methods Concurrency Matrix

	UnaryOperationImpl	getOperand	getOperator	transfer	toString
UnaryOperationImpl	∦	∦	∦	∦	∦
getOperand	∦	∦	∦	∦	∦
getOperator	∦	∦	∥	∦	∦
transfer	∦	∦	∦	∦	∦
toString	∦	∦	∦	∦	∦

43 SourceVariableDeclarationImpl

Table 125: Methods Requires Clause Satisfiability

Method	Satisfiability
SourceVariableDeclarationImpl	√
getDeclaredVariable	√
resolveBinding	√
isCaughtVariable	√
isFormalParameter	√
transfer	√
toString	√

Table 126: State Transition Matrix

	alive
alive	↑

Table 127: Methods Concurrency Matrix

	SourceVariableDeclarationImpl	getDeclaredVariable	resolveBinding	isCaughtVariable	isFormalParameter	transfer	toString
SourceVariableDeclarationImpl	∦	∦	∦	∦	∦	∦	∦
getDeclaredVariable	∦	‖	∦	∦	∦	∦	‖
resolveBinding	∦	∦	∦	∦	∦	∦	∦
isCaughtVariable	∦	∦	∦	∦	∦	∦	∦
isFormalParameter	∦	∦	∦	∦	∦	∦	∦
transfer	∦	∦	∦	∦	∦	∦	∦
toString	∦	‖	∦	∦	∦	∦	‖

44 EclipseSuperCallInstruction

Table 128: Methods Requires Clause Satisfiability

Method	Satisfiability
EclipseSuperCallInstruction	√
getMethodName	√
getReceiverOperand	√
resolveBinding	√
isSuperCall	√
getArgOperands	√

Table 129: State Transition Matrix

	alive
alive	↑

Table 130: Methods Concurrency Matrix

	EclipseSuperCallInstruction	getMethodName	getReceiverOperand	resolveBinding	isSuperCall	getArgOperands
EclipseSuperCallInstruction	∦	∦	∦	∦	∦	∦
getMethodName	∦	∦	∦	∦	‖	∦
getReceiverOperand	∦	∦	∦	∦	‖	∦
resolveBinding	∦	∦	∦	∦	‖	∦
isSuperCall	∦	‖	‖	‖	‖	‖
getArgOperands	∦	∦	∦	∦	‖	∦

45 AbstractBinaryOperation

Table 131: Methods Requires Clause Satisfiability

Method	Satisfiability
AbstractBinaryOperation	√
getOperator	√
transfer	√
toString	√

Table 132: State Transition Matrix

	alive
alive	↑

Table 133: Methods Concurrency Matrix

	AbstractBinaryOperation	getOperator	transfer	toString
AbstractBinaryOperation	⫽	‖	⫽	⫽
getOperator	‖	‖	‖	‖
transfer	⫽	‖	⫽	‖
toString	⫽	‖	‖	‖

46 LoadArrayInstructionImpl

Table 134: Methods Requires Clause Satisfiability

Method	Satisfiability
LoadArrayInstructionImpl	√
getSourceArray	√
getAccessedArrayOperand	√
getArrayIndex	√
transfer	√
toString	√

Table 135: State Transition Matrix

	alive
alive	↑

Table 136: Methods Concurrency Matrix

	LoadArrayInstructionImpl	getSourceArray	getAccessedArrayOperand	getArrayIndex	transfer	toString
LoadArrayInstructionImpl	∦	∦	∦	∦	∦	∦
getSourceArray	∦	∦	∦	∦	∦	∦
getAccessedArrayOperand	∦	∦	∦	∦	∦	∦
getArrayIndex	∦	∦	∦	∦	∦	∦
transfer	∦	∦	∦	∦	∦	∦
toString	∦	∦	∦	∦	∦	∦

47 ThisVariable

Table 137: Methods Requires Clause Satisfiability

Method	Satisfiability
ThisVariable	√
isImplicit	√
explicitQualifier	√

Table 138: State Transition Matrix

	alive
alive	↑

Table 139: Methods Concurrency Matrix

	ThisVariable	isImplicit	explicitQualifier
ThisVariable	∦	∦	∦
isImplicit	∦	‖	∦
explicitQualifier	∦	∦	∦

48 KeywordVariable

Table 140: Methods Requires Clause Satisfiability

Method	Satisfiability
KeywordVariable	√
setQualifier	√

Table 141: State Transition Matrix

	alive
alive	↑

Table 142: Methods Concurrency Matrix

	KeywordVariable	setQualifier
KeywordVariable	∦	∦
setQualifier	∦	∦

49 EclipseReferenceFieldAccess

Table 143: Methods Requires Clause Satisfiability

Method	Satisfiability
EclipseReferenceFieldAccess	√
getFieldName	√
resolveFieldBinding	√
isImplicitThisAccess	√
isExplicitSuperAccess	√
getAccessedInstanceInternal	√

Table 144: State Transition Matrix

	alive
alive	↑

Table 145: Methods Concurrency Matrix

	EclipseReferenceFieldAccess	getFieldName	resolveFieldBinding	isImplicitThisAccess	isExplicitSuperAccess	getAccessedInstanceInternal
EclipseReferenceFieldAccess	∦	∦	∦	∦	∦	∦
getFieldName	∦	∦	∦	‖	‖	∦
resolveFieldBinding	∦	∦	∦	‖	‖	∦
isImplicitThisAccess	∦	‖	‖	‖	‖	‖
isExplicitSuperAccess	∦	‖	‖	‖	‖	‖
getAccessedInstanceInternal	∦	∦	∦	‖	‖	∦

50 Abbreviation

Table 146: 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

51 Annotated Version of Sequential Java Program generated by Sip4j

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

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

9  @Perm(requires="immutable(this) in alive",
10 ensures="immutable(this) in alive")
11  public Variable getDestinationObject() {
12  return null;

14 }
15 @Perm(requires="immutable(this) in alive",
16 ensures="immutable(this) in alive")
17  public Variable getAccessedObjectOperand() {
18  return null;

20 }
21 @Perm(requires="immutable(this) in alive",
22 ensures="immutable(this) in alive")
23  public String getFieldName() {
24  return null;

26 }
27 @Perm(requires="immutable(this) in alive",
28 ensures="immutable(this) in alive")
29  public IVariableBinding resolveFieldBinding() {
30  return null;

32 }
33 @Perm(requires="immutable(this) in alive",
34 ensures="immutable(this) in alive")
35  public boolean isStaticFieldAccess() {
36  return 0;

38 }
39 @Perm(requires="unique(this) in alive",
40 ensures="unique(this) in alive")
41  public LE transfer(ITACTransferFunction<LE> tf, LE value) {
42  return null;

44 }
45 @Perm(requires="immutable(this) in alive",
46 ensures="immutable(this) in alive")
47  public String toString() {
48  return null;

50 }

52 }ENDOFCLASS

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

56 class EclipseTACInstructionFactory {
57 @Perm(ensures="unique(this) in alive")
58 EclipseTACInstructionFactory() {   }


61  private TACInstruction createStore(Expression node, Expression targetNode, Variable source,
        IEclipseVariableQuery eclipseVariableQuery) {
62  return null;

64 }
65 @Perm(requires="unique(this) in alive",
66 ensures="unique(this) in alive")
67  public TACInstruction create(Assignment node, IEclipseVariableQuery eclipseVariableQuery) {
68  return null;

70 }
71 @Perm(requires="share(this) in alive",
72 ensures="share(this) in alive")
73  private boolean isLoad(Expression node) {
74  return 0;
```

```
76 }

78 }ENDOFCLASS

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

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

86 @Perm(requires="unique(this) in alive",
87 ensures="unique(this) in alive")
88  public boolean visit(ArrayAccess node) {
89  return 0;

91 }
92 @Perm(requires="share(this) in alive",
93 ensures="share(this) in alive")
94  private void setResult(TACInstruction result) {

96 }
97 @Perm(requires="pure(this) in alive",
98 ensures="pure(this) in alive")
99  public TACInstruction getResult() {
100  return null;

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

107 }

109 }ENDOFCLASS

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

113 class StoreArrayInstructionImpl {
114 @Perm(ensures="unique(this) in alive")
115 StoreArrayInstructionImpl() {   }

117 @Perm(requires="immutable(this) in alive",
118 ensures="immutable(this) in alive")
119  public Variable getDestinationArray() {
120  return null;

122 }
123 @Perm(requires="immutable(this) in alive",
124 ensures="immutable(this) in alive")
125  protected ArrayAccess getTargetNode() {
126  return null;

128 }
129 @Perm(requires="immutable(this) in alive",
130 ensures="immutable(this) in alive")
131  public Variable getAccessedArrayOperand() {
132  return null;

134 }
135 @Perm(requires="immutable(this) in alive",
136 ensures="immutable(this) in alive")
137  public Variable getArrayIndex() {
138  return null;

140 }
141 @Perm(requires="unique(this) in alive",
142 ensures="unique(this) in alive")
143  public LE transfer(ITACTransferFunction<LE> tf, LE value) {
144  return null;

146 }
147 @Perm(requires="immutable(this) in alive",
148 ensures="immutable(this) in alive")
149  public String toString() {
150  return null;

152 }

154 }ENDOFCLASS
```

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

158 class AbstractStoreInstruction {
159 @Perm(ensures="unique(this) in alive")
160 AbstractStoreInstruction() {   }

162 @Perm(requires="immutable(this) in alive",
163 ensures="immutable(this) in alive")
164  public Variable getSourceOperand() {
165  return null;

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

173 }

175 }ENDOFCLASS

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

179 class CopyInstructionImpl {
180 @Perm(ensures="unique(this) in alive")
181 CopyInstructionImpl() {   }

183 @Perm(requires="immutable(this) in alive",
184 ensures="immutable(this) in alive")
185  public Variable getOperand() {
186  return null;

188 }
189 @Perm(requires="immutable(this) in alive",
190 ensures="immutable(this) in alive")
191  public Variable getResultVariable() {
192  return null;

194 }
195 @Perm(requires="unique(this) in alive",
196 ensures="unique(this) in alive")
197  public LE transfer(ITACTransferFunction<LE> tf, LE value) {
198  return null;

200 }
201 @Perm(requires="immutable(this) in alive",
202 ensures="immutable(this) in alive")
203  public String toString() {
204  return null;

206 }

208 }ENDOFCLASS

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

212 class SourceVariableReadImpl {
213 @Perm(ensures="unique(this) in alive")
214 SourceVariableReadImpl() {   }

216 @Perm(requires="pure(this) in alive",
217 ensures="pure(this) in alive")
218  public Variable getVariable() {
219  return null;

221 }
222 @Perm(requires="pure(this) in alive",
223 ensures="pure(this) in alive")
224   Variable getResultVariable() {
225  return null;

227 }
228 @Perm(requires="unique(this) in alive",
229 ensures="unique(this) in alive")
230  public LE transfer(ITACTransferFunction<LE> tf, LE value) {
231  return null;

233 }
234 @Perm(requires="unique(this) in alive",
235 ensures="unique(this) in alive")
236  public String toString() {
```

```
 return null;

}

}ENDOFCLASS

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

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

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public LE transfer(ITACTransferFunction<LE> tf, LE value) {
 return null;

}

}ENDOFCLASS

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

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


}ENDOFCLASS

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

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


 public KeywordVariable getConstructionObject() {
 return null;

}

 public boolean isSuperCall() {
 return 0;

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

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

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

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

}

}ENDOFCLASS

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

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

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

```
 protected SuperVariable superVariable(Name qualifier) {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 protected List<Variable> variables(List<Expression> nodes) {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 protected Variable variable(Expression node) {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 protected Variable targetVariable(ASTNode node) {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 protected TypeVariable typeVariable(ITypeBinding binding) {
 return null;

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

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 protected ThisVariable receiverVariable() {
 return null;

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

}

  String argsString(List<Variable> args) {
 return null;

}

}ENDOFCLASS

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

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

@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public LE transfer(ITACTransferFunction<LE> tf, LE value) {
 return null;

}

 public String toString() {
 return null;

}

}ENDOFCLASS

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

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

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

```
399 ensures="share(this) in alive")
400  public Variable getOperand1() {
401  return null;

403 }
404 @Perm(requires="share(this) in alive",
405 ensures="share(this) in alive")
406  public Variable getOperand2() {
407  return null;

409 }

411 }ENDOFCLASS

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

415 class InstanceofInstructionImpl {
416 @Perm(ensures="unique(this) in alive")
417 InstanceofInstructionImpl() {   }

419 @Perm(requires="unique(this) in alive",
420 ensures="unique(this) in alive")
421  public Type getTestedTypeNode() {
422  return null;

424 }
425 @Perm(requires="share(this) in alive",
426 ensures="share(this) in alive")
427  public Variable getOperand() {
428  return null;

430 }
431 @Perm(requires="unique(this) in alive",
432 ensures="unique(this) in alive")
433  public LE transfer(ITACTransferFunction<LE> tf, LE value) {
434  return null;

436 }
437 @Perm(requires="unique(this) in alive",
438 ensures="unique(this) in alive")
439  public String toString() {
440  return null;

442 }

444 }ENDOFCLASS

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

448 class ArrayInitInstructionImpl {
449 @Perm(ensures="unique(this) in alive")
450 ArrayInitInstructionImpl() {   }

452 @Perm(requires="share(this) in alive",
453 ensures="share(this) in alive")
454  public List<Variable> getInitOperands() {
455  return null;

457 }
458 @Perm(requires="unique(this) in alive",
459 ensures="unique(this) in alive")
460  public LE transfer(ITACTransferFunction<LE> tf, LE value) {
461  return null;

463 }
464 @Perm(requires="share(this) in alive",
465 ensures="share(this) in alive")
466  public String toString() {
467  return null;

469 }

471 }ENDOFCLASS

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

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

479 @Perm(requires="immutable(this) in alive",
```

```
480 ensures="immutable(this) in alive")
481  public Object getLiteral() {
482  return null;

484 }
485 @Perm(requires="immutable(this) in alive",
486 ensures="immutable(this) in alive")
487  public boolean isNonNullString() {
488  return 0;

490 }
491 @Perm(requires="immutable(this) in alive",
492 ensures="immutable(this) in alive")
493  public boolean isNull() {
494  return 0;

496 }
497 @Perm(requires="immutable(this) in alive",
498 ensures="immutable(this) in alive")
499  public boolean isNumber() {
500  return 0;

502 }
503 @Perm(requires="immutable(this) in alive",
504 ensures="immutable(this) in alive")
505  public boolean isPrimitive() {
506  return 0;

508 }

510 }ENDOFCLASS

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

514 class LoadLiteralInstructionImpl {
515 @Perm(ensures="unique(this) in alive")
516 LoadLiteralInstructionImpl() {   }

518 @Perm(requires="immutable(this) in alive",
519 ensures="immutable(this) in alive")
520  public Object getLiteral() {
521  return null;

523 }
524 @Perm(requires="immutable(this) in alive",
525 ensures="immutable(this) in alive")
526  public boolean isPrimitive() {
527  return 0;

529 }
530 @Perm(requires="immutable(this) in alive",
531 ensures="immutable(this) in alive")
532  public boolean isNumber() {
533  return 0;

535 }
536 @Perm(requires="immutable(this) in alive",
537 ensures="immutable(this) in alive")
538  public boolean isNull() {
539  return 0;

541 }
542 @Perm(requires="immutable(this) in alive",
543 ensures="immutable(this) in alive")
544  public boolean isNonNullString() {
545  return 0;

547 }
548 @Perm(requires="unique(this) in alive",
549 ensures="unique(this) in alive")
550  public LE transfer(ITACTransferFunction<LE> tf, LE value) {
551  return null;

553 }
554 @Perm(requires="immutable(this) in alive",
555 ensures="immutable(this) in alive")
556  public String toString() {
557  return null;

559 }
```

```
}ENDOFCLASS

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

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

@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 protected Variable getResultVariable() {
 return null;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 protected int getUseAsResult() {
 return 0;

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

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public LE transfer(ITACTransferFunction<LE> tf, LE value) {
 return null;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public LE deriveResult(ITACTransferFunction<LE> tf, TACInstruction targetInstr, LE value, boolean
      afterResult) {
 return null;

}

}ENDOFCLASS

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

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

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

}

}ENDOFCLASS

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

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


}ENDOFCLASS

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

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

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

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public LE transfer(ITACTransferFunction<LE> tf, LE value) {
```

```
641   return null;

643  }
644  @Perm(requires="share(this) in alive",
645  ensures="share(this) in alive")
646   public String toString() {
647   return null;

649  }

651  }ENDOFCLASS

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

655  class EclipseBinaryInfixOperation {
656  @Perm(ensures="unique(this) in alive")
657  EclipseBinaryInfixOperation() {   }

659  @Perm(requires="share(this) in alive",
660  ensures="share(this) in alive")
661   public Variable getOperand1() {
662   return null;

664  }
665  @Perm(requires="share(this) in alive",
666  ensures="share(this) in alive")
667   public Variable getOperand2() {
668   return null;

670  }

672  }ENDOFCLASS

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

676  class LoadFieldInstructionImpl {
677  @Perm(ensures="unique(this) in alive")
678  LoadFieldInstructionImpl() {   }

680  @Perm(requires="immutable(this) in alive",
681  ensures="immutable(this) in alive")
682   public String getFieldName() {
683   return null;

685  }
686  @Perm(requires="immutable(this) in alive",
687  ensures="immutable(this) in alive")
688   public IVariableBinding resolveFieldBinding() {
689   return null;

691  }
692  @Perm(requires="immutable(this) in alive",
693  ensures="immutable(this) in alive")
694   public Variable getSourceObject() {
695   return null;

697  }
698  @Perm(requires="immutable(this) in alive",
699  ensures="immutable(this) in alive")
700   public Variable getAccessedObjectOperand() {
701   return null;

703  }
704  @Perm(requires="immutable(this) in alive",
705  ensures="immutable(this) in alive")
706   public boolean isStaticFieldAccess() {
707   return 0;

709  }
710  @Perm(requires="unique(this) in alive",
711  ensures="unique(this) in alive")
712   public LE transfer(ITACTransferFunction<LE> tf, LE value) {
713   return null;

715  }
716  @Perm(requires="immutable(this) in alive",
717  ensures="immutable(this) in alive")
718   public String toString() {
719   return null;

721  }
```

```
}ENDOFCLASS

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

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


 public boolean isStaticMethodCall() {
 return 0;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public LE transfer(ITACTransferFunction<LE> tf, LE value) {
 return null;

}

 public String toString() {
 return null;

}

}ENDOFCLASS

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

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


  boolean isStaticBinding(IBinding binding) {
 return 0;

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

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

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public TACInstruction instruction(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="pure(this) in alive",
ensures="pure(this) in alive")
 private TACInstruction createInstruction(ASTNode astNode) {
 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 TypeVariable typeVariable(ITypeBinding binding) {
 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;

}

}ENDOFCLASS

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

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

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

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

}

}ENDOFCLASS

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

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


 protected TempVariable createTemp(ASTNode node) {
 return null;

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

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

```
883 ensures="immutable(this) in alive")
884  private Variable defaultVariable() {
885  return null;

887 }
888 @Perm(requires="share(this) in alive",
889 ensures="share(this) in alive")
890  protected void setTarget(Variable newTarget) {

892 }

894   ASTNode checkIfCopyNeeded(ASTNode n) {
895  return null;

897 }

899   boolean branch(ASTNode p, ASTNode n) {
900  return 0;

902 }

904   boolean assigns(ASTNode p, ASTNode n) {
905  return 0;

907 }

909  protected Variable getResultVariable() {
910  return null;

912 }

914 }ENDOFCLASS

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

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


923 }ENDOFCLASS

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

927 class EclipseNormalCallInstruction {
928 @Perm(ensures="unique(this) in alive")
929 EclipseNormalCallInstruction() {   }


932  public boolean isSuperCall() {
933  return 0;

935 }
936 @Perm(requires="unique(this) in alive",
937 ensures="unique(this) in alive")
938  public Variable getReceiverOperand() {
939  return null;

941 }
942 @Perm(requires="unique(this) in alive",
943 ensures="unique(this) in alive")
944  public IMethodBinding resolveBinding() {
945  return null;

947 }
948 @Perm(requires="share(this) in alive",
949 ensures="share(this) in alive")
950  public List<Variable> getArgOperands() {
951  return null;

953 }
954 @Perm(requires="share(this) in alive",
955 ensures="share(this) in alive")
956  public String getMethodName() {
957  return null;

959 }
960 @Perm(requires="unique(this) in alive",
961 ensures="unique(this) in alive")
962  public String toString() {
963  return null;
```

```
965 }

967 }ENDOFCLASS

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

971 class ReturnInstructionImpl {
972 @Perm(ensures="unique(this) in alive")
973 ReturnInstructionImpl() {   }

975 @Perm(requires="unique(this) in alive",
976 ensures="unique(this) in alive")
977  public Variable getReturnedVariable() {
978  return null;

980 }
981 @Perm(requires="unique(this) in alive",
982 ensures="unique(this) in alive")
983  public LE transfer(ITACTransferFunction<LE> tf, LE value) {
984  return null;

986 }

988 }ENDOFCLASS

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

992 class EclipseSuperFieldAccess {
993 @Perm(ensures="unique(this) in alive")
994 EclipseSuperFieldAccess() {   }

996 @Perm(requires="unique(this) in alive",
997 ensures="unique(this) in alive")
998  public SimpleName getFieldName() {
999  return null;

1001 }
1002 @Perm(requires="unique(this) in alive",
1003 ensures="unique(this) in alive")
1004  public IVariableBinding resolveFieldBinding() {
1005  return null;

1007 }

1009  public boolean isImplicitThisAccess() {
1010  return 0;

1012 }

1014  public boolean isExplicitSuperAccess() {
1015  return 0;

1017 }
1018 @Perm(requires="share(this) in alive",
1019 ensures="share(this) in alive")
1020  protected Variable getAccessedInstanceInternal(IVariableBinding field) {
1021  return null;

1023 }

1025 }ENDOFCLASS

1027 @ClassStates({@State(name = "alive")})

1029 class NewObjectInstructionImpl {
1030 @Perm(ensures="unique(this) in alive")
1031 NewObjectInstructionImpl() {   }

1033 @Perm(requires="unique(this) in alive",
1034 ensures="unique(this) in alive")
1035  public IMethodBinding resolveBinding() {
1036  return null;

1038 }
1039 @Perm(requires="share(this) in alive",
1040 ensures="share(this) in alive")
1041  public boolean isAnonClassType() {
1042  return 0;

1044 }
```

```
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public List<Variable> getArgOperands() {
 return null;

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

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

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

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public LE transfer(ITACTransferFunction<LE> tf, LE value) {
 return null;

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

}

}ENDOFCLASS

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

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

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

}

 public boolean isExplicitSuperAccess() {
 return 0;

}

 public boolean isImplicitThisAccess() {
 return 0;

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

}
@Perm(requires="pure(this) in alive",
ensures="pure(this) in alive")
 protected Variable getAccessedInstanceInternal(IVariableBinding field) {
 return null;

}

}ENDOFCLASS

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

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

```
@Perm(ensures="none(this) in alive")
 public Variable getOperand1() {
 return null;

}
@Perm(ensures="none(this) in alive")
 public Variable getOperand2() {
 return null;

}

}ENDOFCLASS

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

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

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

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

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public LE transfer(ITACTransferFunction<LE> tf, LE value) {
 return null;

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

}

}ENDOFCLASS

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

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

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

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

}

 public boolean isImplicitThisAccess() {
 return 0;

}

 public boolean isExplicitSuperAccess() {
 return 0;

}
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 protected Variable getAccessedInstanceInternal(IVariableBinding field) {
 return null;

}
```

```
}ENDOFCLASS

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

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

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

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

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

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

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

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

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public LE transfer(ITACTransferFunction<LE> tf, LE value) {
 return null;

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

}

}ENDOFCLASS

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

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

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

}

}ENDOFCLASS

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

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

```
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public Type getTypeNode() {
 return null;

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public LE transfer(ITACTransferFunction<LE> tf, LE value) {
 return null;

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

}

}ENDOFCLASS

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

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

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

}
@Perm(requires="full(this) in alive",
ensures="full(this) in alive")
 public IVariableBinding resolveFieldBinding() {
 return null;

}

 public boolean isImplicitThisAccess() {
 return 0;

}

 public boolean isExplicitSuperAccess() {
 return 0;

}
@Perm(requires="immutable(this) in alive",
ensures="immutable(this) in alive")
 protected Variable getAccessedInstanceInternal(IVariableBinding field) {
 return null;

}

}ENDOFCLASS

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

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

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

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

}

 public boolean isSuperCall() {
 return 0;

}
```

```
@Perm(requires="share(this) in alive",
ensures="share(this) in alive")
 public List<Variable> getArgOperands() {
 return null;

}

 public boolean hasEnclosingInstanceSpecifier() {
 return 0;

}

 public Variable getEnclosingInstanceSpecifier() {
 return null;

}

}ENDOFCLASS

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

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

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

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

}
@Perm(requires="unique(this) in alive",
ensures="unique(this) in alive")
 public LE transfer(ITACTransferFunction<LE> tf, LE value) {
 return null;

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

}

}ENDOFCLASS

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

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

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

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

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

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

```
1450 }
1451 @Perm(requires="unique(this) in alive",
1452 ensures="unique(this) in alive")
1453  public LE transfer(ITACTransferFunction<LE> tf, LE value) {
1454  return null;

1456 }
1457 @Perm(requires="pure(this) in alive",
1458 ensures="pure(this) in alive")
1459  public String toString() {
1460  return null;

1462 }

1464 }ENDOFCLASS

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

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

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

1477 }
1478 @Perm(requires="unique(this) in alive",
1479 ensures="unique(this) in alive")
1480  public Variable getReceiverOperand() {
1481  return null;

1483 }
1484 @Perm(requires="unique(this) in alive",
1485 ensures="unique(this) in alive")
1486  public IMethodBinding resolveBinding() {
1487  return null;

1489 }

1491  public boolean isSuperCall() {
1492  return 0;

1494 }
1495 @Perm(requires="share(this) in alive",
1496 ensures="share(this) in alive")
1497  public List<Variable> getArgOperands() {
1498  return null;

1500 }

1502 }ENDOFCLASS

1504 @ClassStates({@State(name = "alive")})

1506 class AbstractBinaryOperation {
1507 @Perm(ensures="unique(this) in alive")
1508 AbstractBinaryOperation() {   }

1510 @Perm(ensures="none(this) in alive")
1511  public BinaryOperator getOperator() {
1512  return null;

1514 }
1515 @Perm(requires="unique(this) in alive",
1516 ensures="unique(this) in alive")
1517  public LE transfer(ITACTransferFunction<LE> tf, LE value) {
1518  return null;

1520 }

1522  public String toString() {
1523  return null;

1525 }

1527 }ENDOFCLASS

1529 @ClassStates({@State(name = "alive")})
```

```
1531 class LoadArrayInstructionImpl {
1532 @Perm(ensures="unique(this) in alive")
1533 LoadArrayInstructionImpl() {   }

1535 @Perm(requires="share(this) in alive",
1536 ensures="share(this) in alive")
1537  public Variable getSourceArray() {
1538  return null;

1540 }
1541 @Perm(requires="share(this) in alive",
1542 ensures="share(this) in alive")
1543  public Variable getAccessedArrayOperand() {
1544  return null;

1546 }
1547 @Perm(requires="share(this) in alive",
1548 ensures="share(this) in alive")
1549  public Variable getArrayIndex() {
1550  return null;

1552 }
1553 @Perm(requires="unique(this) in alive",
1554 ensures="unique(this) in alive")
1555  public LE transfer(ITACTransferFunction<LE> tf, LE value) {
1556  return null;

1558 }
1559 @Perm(requires="share(this) in alive",
1560 ensures="share(this) in alive")
1561  public String toString() {
1562  return null;

1564 }

1566 }ENDOFCLASS

1568 @ClassStates({@State(name = "alive")})

1570 class ThisVariable {
1571 @Perm(ensures="unique(this) in alive")
1572 ThisVariable() {   }

1574 @Perm(requires="pure(this) in alive",
1575 ensures="pure(this) in alive")
1576  public boolean isImplicit() {
1577  return 0;

1579 }
1580 @Perm(requires="unique(this) in alive",
1581 ensures="unique(this) in alive")
1582  public void explicitQualifier(Name qualifier) {

1584 }

1586 }ENDOFCLASS

1588 @ClassStates({@State(name = "alive")})

1590 class KeywordVariable {
1591 @Perm(ensures="unique(this) in alive")
1592 KeywordVariable() {   }

1594 @Perm(requires="share(this) in alive",
1595 ensures="share(this) in alive")
1596  protected void setQualifier(Name qualifier) {

1598 }

1600 }ENDOFCLASS

1602 @ClassStates({@State(name = "alive")})

1604 class EclipseReferenceFieldAccess {
1605 @Perm(ensures="unique(this) in alive")
1606 EclipseReferenceFieldAccess() {   }

1608 @Perm(requires="unique(this) in alive",
1609 ensures="unique(this) in alive")
1610  public SimpleName getFieldName() {
1611  return null;
```

```
1613  }
1614  @Perm(requires="unique(this) in alive",
1615  ensures="unique(this) in alive")
1616   public IVariableBinding resolveFieldBinding() {
1617   return null;

1619  }

1621   public boolean isImplicitThisAccess() {
1622   return 0;

1624  }

1626   public boolean isExplicitSuperAccess() {
1627   return 0;

1629  }
1630  @Perm(requires="share(this) in alive",
1631  ensures="share(this) in alive")
1632   protected Variable getAccessedInstanceInternal(IVariableBinding field) {
1633   return null;

1635  }

1637  }ENDOFCLASS
```