

# 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

<b>1</b>	<b>BranchSensitiveTACAnalysis</b>	<b>4</b>
<b>2</b>	<b>SourceVariable</b>	<b>5</b>
<b>3</b>	<b>Variable</b>	<b>6</b>
<b>4</b>	<b>AbstractTACBranchSensitiveTransferFunction</b>	<b>7</b>
<b>5</b>	<b>SimpleInstructionVisitor</b>	<b>8</b>
<b>6</b>	<b>TransferVisitor</b>	<b>9</b>
<b>7</b>	<b>Lattice</b>	<b>11</b>
<b>8</b>	<b>TypeVariable</b>	<b>12</b>
<b>9</b>	<b>AbstractingTransferFunction</b>	<b>13</b>
<b>10</b>	<b>SuperVariable</b>	<b>15</b>
<b>11</b>	<b>KeywordVariable</b>	<b>16</b>
<b>12</b>	<b>EclipseTAC</b>	<b>17</b>
<b>13</b>	<b>BranchInsensitiveTACAnalysis</b>	<b>19</b>
<b>14</b>	<b>UnaryOperator</b>	<b>20</b>
<b>15</b>	<b>TempVariable</b>	<b>21</b>
<b>16</b>	<b>TACFlowAnalysis</b>	<b>22</b>
<b>17</b>	<b>BranchInsensitiveTACAnalysisDriver</b>	<b>23</b>
<b>18</b>	<b>BranchSensitiveTACAnalysisDriver</b>	<b>24</b>
<b>19</b>	<b>NewInstructionVisitor</b>	<b>25</b>
<b>20</b>	<b>MotherFlowAnalysis</b>	<b>26</b>
<b>21</b>	<b>SingleResult</b>	<b>27</b>
<b>22</b>	<b>ThisVariable</b>	<b>28</b>
<b>23</b>	<b>AbstractTACAnalysisDriver</b>	<b>29</b>
<b>24</b>	<b>CompilationUnitTACs</b>	<b>30</b>
<b>25</b>	<b>LabeledSingleResult</b>	<b>31</b>
<b>26</b>	<b>AbstractTransferFunction</b>	<b>32</b>
<b>27</b>	<b>BinaryOperator</b>	<b>34</b>
<b>28</b>	<b>Abbreviation</b>	<b>35</b>
<b>29</b>	<b>Annotated Version of Sequential Java Program generated by Sip4j</b>	<b>36</b>

# 1 StoreFieldInstructionImpl

Table 2: Methods Requires Clause Satisfiability

Method	Satisfiability
StoreFieldInstructionImpl	✓
getDestinationObject	✓
getAccessedObjectOperand	✓
getFieldName	✓
resolveFieldBinding	✓
isStaticFieldAccess	✓
transfer	✓
toString	✓

Table 3: State Transition Matrix

	alive
alive	↑

Table 4: Methods Concurrency Matrix

	StoreFieldInstructionImpl	getDestinationObject	getAccessedObjectOperand	getFieldName	resolveFieldBinding	isStaticFieldAccess	transfer	toString
StoreFieldInstructionImpl	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
getDestinationObject	⌘						⌘	
getAccessedObjectOperand	⌘						⌘	
getFieldName	⌘						⌘	
resolveFieldBinding	⌘						⌘	
isStaticFieldAccess	⌘						⌘	
transfer	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
toString	⌘						⌘	

## 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	✓
getNormalLabel	✓

Table 56: State Transition Matrix

	alive
alive	↑

Table 57: Methods Concurrency Matrix

	NormalLabel	getNormalLabel
NormalLabel	⌘	⌘
getNormalLabel	⌘	

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	✓
createTemp	✓
getTarget	✓
defaultVariable	✓
setTarget	✓
checkIfCopyNeeded	✓
branch	✓
assigns	✓
getResultVariable	✓

Table 79: State Transition Matrix

	alive
alive	↑

Table 80: Methods Concurrency Matrix

	AbstractAssignmentInstruction	createTemp	getTarget	defaultVariable	setTarget	checkIfCopyNeeded	branch	assigns	getResultVariable
AbstractAssignmentInstruction	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
createTemp	⌘								
getTarget	⌘		⌘	⌘	⌘				
defaultVariable	⌘		⌘		⌘				
setTarget	⌘		⌘	⌘	⌘				
checkIfCopyNeeded	⌘								
branch	⌘								
assigns	⌘								
getResultVariable	⌘								

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	✓
isSuperCall	✓
getReceiverOperand	✓
resolveBinding	✓
getArgOperands	✓
getMethodName	✓
toString	✓

Table 84: State Transition Matrix

	alive
alive	↑

Table 85: Methods Concurrency Matrix

	EclipseNormalCallInstruction	isSuperCall	getReceiverOperand	resolveBinding	getArgOperands	getMethodName	toString
EclipseNormalCallInstruction	⌘	⌘	⌘	⌘	⌘	⌘	⌘
isSuperCall	⌘						
getReceiverOperand	⌘		⌘	⌘	⌘	⌘	⌘
resolveBinding	⌘		⌘	⌘	⌘	⌘	⌘
getArgOperands	⌘		⌘	⌘	⌘	⌘	⌘
getMethodName	⌘		⌘	⌘	⌘	⌘	⌘
toString	⌘		⌘	⌘	⌘	⌘	⌘

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

## 51 Annotated Version of Sequential Java Program generated by Sip4j

```
1 package outputs;
2 import edu.cmu.cs.plural.annot.*;
3
4 @ClassStates({@State(name = "alive")})
5 class StoreFieldInstructionImpl {
6   @Perm(ensures="unique(this) in alive")
7   StoreFieldInstructionImpl() { }
8
9   @Perm(requires="immutable(this) in alive",
10  ensures="immutable(this) in alive")
11   public Variable getDestinationObject() {
12     return null;
13
14   }
15   @Perm(requires="immutable(this) in alive",
16  ensures="immutable(this) in alive")
17   public Variable getAccessedObjectOperand() {
18     return null;
19
20   }
21   @Perm(requires="immutable(this) in alive",
22  ensures="immutable(this) in alive")
23   public String getFieldName() {
24     return null;
25
26   }
27   @Perm(requires="immutable(this) in alive",
28  ensures="immutable(this) in alive")
29   public IVariableBinding resolveFieldBinding() {
30     return null;
31
32   }
33   @Perm(requires="immutable(this) in alive",
34  ensures="immutable(this) in alive")
35   public boolean isStaticFieldAccess() {
36     return 0;
37
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;
43
44   }
45   @Perm(requires="immutable(this) in alive",
46  ensures="immutable(this) in alive")
47   public String toString() {
48     return null;
49
50   }
51 }
52 }ENDOFCLASS
53
54 @ClassStates({@State(name = "alive")})
55 class EclipseTACInstructionFactory {
56   @Perm(ensures="unique(this) in alive")
57   EclipseTACInstructionFactory() { }
58
59   private TACInstruction createStore(Expression node, Expression targetNode, Variable source,
60   IEclipseVariableQuery eclipseVariableQuery) {
61     return null;
62
63   }
64   @Perm(requires="unique(this) in alive",
65  ensures="unique(this) in alive")
66   public TACInstruction create(Assignment node, IEclipseVariableQuery eclipseVariableQuery) {
67     return null;
68
69   }
70   @Perm(requires="share(this) in alive",
71  ensures="share(this) in alive")
72   private boolean isLoad(Expression node) {
73     return 0;
74
75   }
76 }
```

```

76 }
77 }ENDOFCLASS
78
79 @ClassStates({@State(name = "alive")})
80
81 class NewInstructionVisitor {
82   @Perm(ensures="unique(this) in alive")
83   NewInstructionVisitor() { }
84
85   @Perm(requires="unique(this) in alive",
86     ensures="unique(this) in alive")
87   public boolean visit(ArrayAccess node) {
88     return 0;
89   }
90
91   @Perm(requires="share(this) in alive",
92     ensures="share(this) in alive")
93   private void setResult(TACInstruction result) {
94   }
95
96   @Perm(requires="pure(this) in alive",
97     ensures="pure(this) in alive")
98   public TACInstruction getResult() {
99     return null;
100   }
101
102   @Perm(requires="unique(this) in alive",
103     ensures="unique(this) in alive")
104   private void noResult() {
105   }
106 }
107
108 }ENDOFCLASS
109
110 @ClassStates({@State(name = "alive")})
111
112 class StoreArrayInstructionImpl {
113   @Perm(ensures="unique(this) in alive")
114   StoreArrayInstructionImpl() { }
115
116   @Perm(requires="immutable(this) in alive",
117     ensures="immutable(this) in alive")
118   public Variable getDestinationArray() {
119     return null;
120   }
121
122   @Perm(requires="immutable(this) in alive",
123     ensures="immutable(this) in alive")
124   protected ArrayAccess getTargetNode() {
125     return null;
126   }
127
128   @Perm(requires="immutable(this) in alive",
129     ensures="immutable(this) in alive")
130   public Variable getAccessedArrayOperand() {
131     return null;
132   }
133
134   @Perm(requires="immutable(this) in alive",
135     ensures="immutable(this) in alive")
136   public Variable getArrayIndex() {
137     return null;
138   }
139
140   @Perm(requires="unique(this) in alive",
141     ensures="unique(this) in alive")
142   public LE transfer(ITACTransferFunction<LE> tf, LE value) {
143     return null;
144   }
145
146   @Perm(requires="immutable(this) in alive",
147     ensures="immutable(this) in alive")
148   public String toString() {
149     return null;
150   }
151 }
152
153 }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 public 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 public 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() {

```

```

237     return null;
239 }
241 }ENDOFCLASS
243 @ClassStates({@State(name = "alive")})
245 class EclipseMergeHelper {
246     @Perm(ensures="unique(this) in alive")
247     EclipseMergeHelper() { }
249     @Perm(requires="unique(this) in alive",
250     ensures="unique(this) in alive")
251     public LE transfer(ITACTransferFunction<LE> tf, LE value) {
252         return null;
254     }
256 }ENDOFCLASS
258 @ClassStates({@State(name = "alive")})
260 class LabeledSingleResult {
261     @Perm(ensures="unique(this) in alive")
262     LabeledSingleResult() { }
264 }ENDOFCLASS
266 @ClassStates({@State(name = "alive")})
268 class EclipseSuperConstructorCallInstruction {
269     @Perm(ensures="unique(this) in alive")
270     EclipseSuperConstructorCallInstruction() { }
272     public KeywordVariable getConstructionObject() {
273         return null;
275     }
277 }
279     public boolean isSuperCall() {
280         return 0;
282     }
283     @Perm(requires="share(this) in alive",
284     ensures="share(this) in alive")
285     public List<Variable> getArgOperands() {
286         return null;
288     }
289     @Perm(requires="share(this) in alive",
290     ensures="share(this) in alive")
291     public Variable getEnclosingInstanceSpecifier() {
292         return null;
294     }
295     @Perm(requires="share(this) in alive",
296     ensures="share(this) in alive")
297     public boolean hasEnclosingInstanceSpecifier() {
298         return 0;
300     }
301     @Perm(requires="unique(this) in alive",
302     ensures="unique(this) in alive")
303     public IMethodBinding resolveBinding() {
304         return null;
306     }
308 }ENDOFCLASS
310 @ClassStates({@State(name = "alive")})
312 class AbstractTACInstruction {
313     @Perm(ensures="unique(this) in alive")
314     AbstractTACInstruction() { }
316     @Perm(requires="immutable(this) in alive",
317     ensures="immutable(this) in alive")

```

```

318     protected SuperVariable superVariable(Name qualifier) {
319         return null;
320     }
321 }
322 @Perm(requires="unique(this) in alive",
323 ensures="unique(this) in alive")
324 protected List<Variable> variables(List<Expression> nodes) {
325     return null;
326 }
327 }
328 @Perm(requires="immutable(this) in alive",
329 ensures="immutable(this) in alive")
330 protected Variable variable(Expression node) {
331     return null;
332 }
333 }
334 @Perm(requires="immutable(this) in alive",
335 ensures="immutable(this) in alive")
336 protected Variable targetVariable(ASTNode node) {
337     return null;
338 }
339 }
340 @Perm(requires="immutable(this) in alive",
341 ensures="immutable(this) in alive")
342 protected TypeVariable typeVariable(ITypeBinding binding) {
343     return null;
344 }
345 }
346 @Perm(requires="immutable(this) in alive",
347 ensures="immutable(this) in alive")
348 protected ThisVariable implicitThisVariable(IBinding accessedElement) {
349     return null;
350 }
351 }
352 @Perm(requires="immutable(this) in alive",
353 ensures="immutable(this) in alive")
354 protected ThisVariable receiverVariable() {
355     return null;
356 }
357 }
358 @Perm(requires="immutable(this) in alive",
359 ensures="immutable(this) in alive")
360 public E getNode() {
361     return null;
362 }
363 }
364
365     String argsString(List<Variable> args) {
366         return null;
367     }
368 }
369
370 }ENDOFCLASS
371
372 @ClassStates({@State(name = "alive")})
373
374 class AbstractConstructorCallInstruction {
375     @Perm(ensures="unique(this) in alive")
376     AbstractConstructorCallInstruction() { }
377
378     @Perm(requires="unique(this) in alive",
379 ensures="unique(this) in alive")
380     public LE transfer(ITACTransferFunction<LE> tf, LE value) {
381         return null;
382     }
383 }
384
385     public String toString() {
386         return null;
387     }
388 }
389
390 }ENDOFCLASS
391
392 @ClassStates({@State(name = "alive")})
393
394 class EclipseBinaryAssignOperation {
395     @Perm(ensures="unique(this) in alive")
396     EclipseBinaryAssignOperation() { }
397
398     @Perm(requires="share(this) in alive",

```

```

399 ensures="share(this) in alive")
400 public Variable getOperand1() {
401     return null;
402 }
403
404 @Perm(requires="share(this) in alive",
405 ensures="share(this) in alive")
406 public Variable getOperand2() {
407     return null;
408 }
409
410 }ENDOFCLASS
411
412 @ClassStates({@State(name = "alive")})
413
414 class InstanceofInstructionImpl {
415     @Perm(ensures="unique(this) in alive")
416     InstanceofInstructionImpl() { }
417
418     @Perm(requires="unique(this) in alive",
419     ensures="unique(this) in alive")
420     public Type getTestedTypeNode() {
421         return null;
422     }
423
424     @Perm(requires="share(this) in alive",
425     ensures="share(this) in alive")
426     public Variable getOperand() {
427         return null;
428     }
429
430     @Perm(requires="unique(this) in alive",
431     ensures="unique(this) in alive")
432     public LE transfer(ITACTransferFunction<LE> tf, LE value) {
433         return null;
434     }
435
436     @Perm(requires="unique(this) in alive",
437     ensures="unique(this) in alive")
438     public String toString() {
439         return null;
440     }
441
442 }
443
444 }ENDOFCLASS
445
446 @ClassStates({@State(name = "alive")})
447
448 class ArrayInitInstructionImpl {
449     @Perm(ensures="unique(this) in alive")
450     ArrayInitInstructionImpl() { }
451
452     @Perm(requires="share(this) in alive",
453     ensures="share(this) in alive")
454     public List<Variable> getInitOperands() {
455         return null;
456     }
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;
462     }
463
464     @Perm(requires="share(this) in alive",
465     ensures="share(this) in alive")
466     public String toString() {
467         return null;
468     }
469
470 }
471
472 }ENDOFCLASS
473
474 @ClassStates({@State(name = "alive")})
475
476 class EclipseLoadDesugaredLiteralInstruction {
477     @Perm(ensures="unique(this) in alive")
478     EclipseLoadDesugaredLiteralInstruction() { }
479
480     @Perm(requires="immutable(this) in alive",

```

```

480 ensures="immutable(this) in alive")
481 public Object getLiteral() {
482     return null;
483 }
484
485 @Perm(requires="immutable(this) in alive",
486 ensures="immutable(this) in alive")
487 public boolean isNonNullString() {
488     return 0;
489 }
490
491 @Perm(requires="immutable(this) in alive",
492 ensures="immutable(this) in alive")
493 public boolean isNull() {
494     return 0;
495 }
496
497 @Perm(requires="immutable(this) in alive",
498 ensures="immutable(this) in alive")
499 public boolean isNumber() {
500     return 0;
501 }
502
503 @Perm(requires="immutable(this) in alive",
504 ensures="immutable(this) in alive")
505 public boolean isPrimitive() {
506     return 0;
507 }
508 }
509
510 }ENDOFCLASS
511
512 @ClassStates({@State(name = "alive")})
513
514 class LoadLiteralInstructionImpl {
515     @Perm(ensures="unique(this) in alive")
516     LoadLiteralInstructionImpl() { }
517
518     @Perm(requires="immutable(this) in alive",
519     ensures="immutable(this) in alive")
520     public Object getLiteral() {
521         return null;
522     }
523 }
524
525 @Perm(requires="immutable(this) in alive",
526 ensures="immutable(this) in alive")
527 public boolean isPrimitive() {
528     return 0;
529 }
530
531 @Perm(requires="immutable(this) in alive",
532 ensures="immutable(this) in alive")
533 public boolean isNumber() {
534     return 0;
535 }
536
537 @Perm(requires="immutable(this) in alive",
538 ensures="immutable(this) in alive")
539 public boolean isNull() {
540     return 0;
541 }
542
543 @Perm(requires="immutable(this) in alive",
544 ensures="immutable(this) in alive")
545 public boolean isNonNullString() {
546     return 0;
547 }
548
549 @Perm(requires="unique(this) in alive",
550 ensures="unique(this) in alive")
551 public LE transfer(ITACTransferFunction<LE> tf, LE value) {
552     return null;
553 }
554
555 @Perm(requires="immutable(this) in alive",
556 ensures="immutable(this) in alive")
557 public String toString() {
558     return null;
559 }
560 }

```

```

561 }ENDOFCLASS
563 @ClassStates({@State(name = "alive")})
565 class EclipseInstructionSequence {
566   @Perm(ensures="unique(this) in alive")
567   EclipseInstructionSequence() { }
569   @Perm(requires="pure(this) in alive",
570     ensures="pure(this) in alive")
571   protected Variable getResultVariable() {
572     return null;
574   }
575   @Perm(requires="immutable(this) in alive",
576     ensures="immutable(this) in alive")
577   protected int getUseAsResult() {
578     return 0;
580   }
581   @Perm(requires="pure(this) in alive",
582     ensures="pure(this) in alive")
583   protected TACInstruction[] getInstructions() {
584     return null;
586   }
587   @Perm(requires="share(this) in alive",
588     ensures="share(this) in alive")
589   public LE transfer(ITACTransferFunction<LE> tf, LE value) {
590     return null;
592   }
593   @Perm(requires="share(this) in alive",
594     ensures="share(this) in alive")
595   public LE deriveResult(ITACTransferFunction<LE> tf, TACInstruction targetInstr, LE value, boolean
596     afterResult) {
597     return null;
598   }
600 }ENDOFCLASS
602 @ClassStates({@State(name = "alive")})
604 class NormalLabel {
605   @Perm(ensures="unique(this) in alive")
606   NormalLabel() { }
608   @Perm(requires="immutable(this) in alive",
609     ensures="immutable(this) in alive")
610   NormalLabel getNormalLabel() {
611     return null;
613   }
615 }ENDOFCLASS
617 @ClassStates({@State(name = "alive")})
619 class ResultfulInstruction {
620   @Perm(ensures="unique(this) in alive")
621   ResultfulInstruction() { }
624 }ENDOFCLASS
626 @ClassStates({@State(name = "alive")})
628 class EnhancedForConditionInstructionImpl {
629   @Perm(ensures="unique(this) in alive")
630   EnhancedForConditionInstructionImpl() { }
632   @Perm(requires="share(this) in alive",
633     ensures="share(this) in alive")
634   public Variable getIteratedOperand() {
635     return null;
637   }
638   @Perm(requires="unique(this) in alive",
639     ensures="unique(this) in alive")
640   public LE transfer(ITACTransferFunction<LE> tf, LE value) {

```

```

641     return null;
642 }
643 @Perm(requires="share(this) in alive",
644 ensures="share(this) in alive")
645 public String toString() {
646     return null;
647 }
648 }
649 }
650 }ENDOFCLASS
651
652 @ClassStates({@State(name = "alive")})
653
654 class EclipseBinaryInfixOperation {
655     @Perm(ensures="unique(this) in alive")
656     EclipseBinaryInfixOperation() { }
657
658     @Perm(requires="share(this) in alive",
659     ensures="share(this) in alive")
660     public Variable getOperand1() {
661         return null;
662     }
663
664     @Perm(requires="share(this) in alive",
665     ensures="share(this) in alive")
666     public Variable getOperand2() {
667         return null;
668     }
669 }
670 }
671 }ENDOFCLASS
672
673 @ClassStates({@State(name = "alive")})
674
675 class LoadFieldInstructionImpl {
676     @Perm(ensures="unique(this) in alive")
677     LoadFieldInstructionImpl() { }
678
679     @Perm(requires="immutable(this) in alive",
680     ensures="immutable(this) in alive")
681     public String getFieldname() {
682         return null;
683     }
684
685     @Perm(requires="immutable(this) in alive",
686     ensures="immutable(this) in alive")
687     public IVariableBinding resolveFieldBinding() {
688         return null;
689     }
690
691     @Perm(requires="immutable(this) in alive",
692     ensures="immutable(this) in alive")
693     public Variable getSourceObject() {
694         return null;
695     }
696
697     @Perm(requires="immutable(this) in alive",
698     ensures="immutable(this) in alive")
699     public Variable getAccessedObjectOperand() {
700         return null;
701     }
702
703     @Perm(requires="immutable(this) in alive",
704     ensures="immutable(this) in alive")
705     public boolean isStaticFieldAccess() {
706         return 0;
707     }
708
709     @Perm(requires="unique(this) in alive",
710     ensures="unique(this) in alive")
711     public LE transfer(ITACTransferFunction<LE> tf, LE value) {
712         return null;
713     }
714
715     @Perm(requires="immutable(this) in alive",
716     ensures="immutable(this) in alive")
717     public String toString() {
718         return null;
719     }
720 }

```

```

723 }ENDOFCLASS
725 @ClassStates({@State(name = "alive")})
727 class AbstractMethodCallInstruction {
728   @Perm(ensures="unique(this) in alive")
729   AbstractMethodCallInstruction() { }
731
732   public boolean isStaticMethodCall() {
733     return 0;
734   }
735
736   @Perm(requires="unique(this) in alive",
737     ensures="unique(this) in alive")
738   public LE transfer(ITACTransferFunction<LE> tf, LE value) {
739     return null;
740   }
741 }
743   public String toString() {
744     return null;
745   }
746 }
748 }ENDOFCLASS
750 @ClassStates({@State(name = "alive")})
752 class EclipseTAC {
753   @Perm(ensures="unique(this) in alive")
754   EclipseTAC() { }
755
756   boolean isStaticBinding(IBinding binding) {
757     return 0;
758   }
759
760   @Perm(requires="share(this) in alive",
761     ensures="share(this) in alive")
762   public ThisVariable implicitThisVariable(IBinding accessedElement) {
763     return null;
764   }
765
766   @Perm(requires="unique(this) in alive",
767     ensures="unique(this) in alive")
768   public ITypeBinding resolveThisType() {
769     return null;
770   }
771
772   @Perm(requires="share(this) in alive",
773     ensures="share(this) in alive")
774   private ITypeBinding implicitThisBinding(IBinding accessedElement) {
775     return null;
776   }
777 }
778
779   private ITypeBinding findElementDeclarationByName(IBinding genericAccessedElement, boolean isMethod,
780     ITypeBinding type, boolean skipPrivate, boolean skipPackagePrivate) {
781     return null;
782   }
783 }
785   boolean isDefaultBinding(IBinding binding) {
786     return 0;
787   }
788
789   @Perm(requires="unique(this) in alive",
790     ensures="unique(this) in alive")
791   public TACInstruction instruction(ASTNode astNode) {
792     return null;
793   }
794
795   @Perm(requires="unique(this) in alive",
796     ensures="unique(this) in alive")
797   public Variable variable(ASTNode astNode) {
798     return null;
799   }
800 }
801 @Perm(requires="share(this) in alive",

```



```

802 ensures="share(this) in alive")
803 private Variable getVariable(IBinding binding) {
804     return null;
805 }
806 }
807 @Perm(requires="unique(this) in alive",
808 ensures="unique(this) in alive")
809 private ThisVariable getThisVariable(ThisExpression node) {
810     return null;
811 }
812 }
813 @Perm(requires="pure(this) in alive",
814 ensures="pure(this) in alive")
815 private TACInstruction createInstruction(ASTNode astNode) {
816     return null;
817 }
818 }
819 @Perm(requires="share(this) in alive",
820 ensures="share(this) in alive")
821 public SourceVariable sourceVariable(IVariableBinding binding) {
822     return null;
823 }
824 }
825 @Perm(requires="share(this) in alive",
826 ensures="share(this) in alive")
827 public TypeVariable typeVariable(ITypeBinding binding) {
828     return null;
829 }
830 }
831 @Perm(requires="unique(this) in alive",
832 ensures="unique(this) in alive")
833 public ThisVariable thisVariable() {
834     return null;
835 }
836 }
837 @Perm(requires="unique(this) in alive",
838 ensures="unique(this) in alive")
839 public SuperVariable superVariable(Name qualifier) {
840     return null;
841 }
842 }
843 }
844 }ENDOFCLASS
845
846 @ClassStates({@State(name = "alive")})
847
848 class EclipseAbstractFieldAccess {
849     @Perm(ensures="unique(this) in alive")
850     EclipseAbstractFieldAccess() { }
851
852     @Perm(ensures="none(this) in alive")
853     boolean isStaticFieldAccess() {
854         return 0;
855     }
856 }
857 @Perm(ensures="none(this) in alive")
858 Variable getAccessedObject() {
859     return null;
860 }
861 }
862 }ENDOFCLASS
863
864 @ClassStates({@State(name = "alive")})
865
866 class AbstractAssignmentInstruction {
867     @Perm(ensures="unique(this) in alive")
868     AbstractAssignmentInstruction() { }
869
870     protected TempVariable createTemp(ASTNode node) {
871         return null;
872     }
873 }
874 }
875 }
876 @Perm(requires="share(this) in alive",
877 ensures="share(this) in alive")
878 public Variable getTarget() {
879     return null;
880 }
881 }
882 @Perm(requires="immutable(this) in alive",

```

```

883 ensures="immutable(this) in alive")
884 private Variable defaultVariable() {
885     return null;
886 }
887 }
888 @Perm(requires="share(this) in alive",
889 ensures="share(this) in alive")
890 protected void setTarget(Variable newTarget) {
891 }
892 }
893
894 ASTNode checkIfCopyNeeded(ASTNode n) {
895     return null;
896 }
897 }
898
899 boolean branch(ASTNode p, ASTNode n) {
900     return 0;
901 }
902 }
903
904 boolean assigns(ASTNode p, ASTNode n) {
905     return 0;
906 }
907 }
908
909 protected Variable getResultVariable() {
910     return null;
911 }
912 }
913 }ENDOFCLASS
914
915 @ClassStates({@State(name = "alive")})
916
917 class TempVariable {
918     @Perm(ensures="unique(this) in alive")
919     TempVariable() { }
920 }
921 }ENDOFCLASS
922
923 @ClassStates({@State(name = "alive")})
924
925 class EclipseNormalCallInstruction {
926     @Perm(ensures="unique(this) in alive")
927     EclipseNormalCallInstruction() { }
928 }
929
930 public boolean isSuperCall() {
931     return 0;
932 }
933
934 }
935 @Perm(requires="unique(this) in alive",
936 ensures="unique(this) in alive")
937 public Variable getReceiverOperand() {
938     return null;
939 }
940
941 }
942 @Perm(requires="unique(this) in alive",
943 ensures="unique(this) in alive")
944 public IMethodBinding resolveBinding() {
945     return null;
946 }
947
948 }
949 @Perm(requires="share(this) in alive",
950 ensures="share(this) in alive")
951 public List<Variable> getArgOperands() {
952     return null;
953 }
954
955 }
956 @Perm(requires="share(this) in alive",
957 ensures="share(this) in alive")
958 public String getMethodName() {
959     return null;
960 }
961
962 }
963 @Perm(requires="unique(this) in alive",
964 ensures="unique(this) in alive")
965 public String toString() {
966     return null;
967 }

```

```

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   }

```

```

1045 @Perm(requires="share(this) in alive",
1046 ensures="share(this) in alive")
1047 public List<Variable> getArgOperands() {
1048     return null;
1049 }
1050 }
1051 @Perm(requires="unique(this) in alive",
1052 ensures="unique(this) in alive")
1053 public ITypeBinding resolveInstantiatedType() {
1054     return null;
1055 }
1056 }
1057 @Perm(requires="share(this) in alive",
1058 ensures="share(this) in alive")
1059 public boolean hasOuterObjectSpecifier() {
1060     return 0;
1061 }
1062 }
1063 @Perm(requires="share(this) in alive",
1064 ensures="share(this) in alive")
1065 public Variable getOuterObjectSpecifierOperand() {
1066     return null;
1067 }
1068 }
1069 @Perm(requires="unique(this) in alive",
1070 ensures="unique(this) in alive")
1071 public LE transfer(ITACTransferFunction<LE> tf, LE value) {
1072     return null;
1073 }
1074 }
1075 @Perm(requires="unique(this) in alive",
1076 ensures="unique(this) in alive")
1077 public String toString() {
1078     return null;
1079 }
1080 }
1081 }ENDOFCLASS
1082
1083 @ClassStates({@State(name = "alive")})
1084
1085 class EclipseFieldDeclaration {
1086     @Perm(ensures="unique(this) in alive")
1087     EclipseFieldDeclaration() { }
1088 }
1089
1090 @Perm(requires="unique(this) in alive",
1091 ensures="unique(this) in alive")
1092 public SimpleName getFieldName() {
1093     return null;
1094 }
1095 }
1096
1097 public boolean isExplicitSuperAccess() {
1098     return 0;
1099 }
1100 }
1101
1102 public boolean isImplicitThisAccess() {
1103     return 0;
1104 }
1105 }
1106 @Perm(requires="unique(this) in alive",
1107 ensures="unique(this) in alive")
1108 public IVariableBinding resolveFieldBinding() {
1109     return null;
1110 }
1111 }
1112 @Perm(requires="pure(this) in alive",
1113 ensures="pure(this) in alive")
1114 protected Variable getAccessedInstanceInternal(IVariableBinding field) {
1115     return null;
1116 }
1117 }
1118 }ENDOFCLASS
1119
1120 @ClassStates({@State(name = "alive")})
1121
1122 class EclipseBinaryDesugaredOperation {
1123     @Perm(ensures="unique(this) in alive")
1124     EclipseBinaryDesugaredOperation() { }
1125 }

```

```

1127 @Perm(ensures="none(this) in alive")
1128 public Variable getOperand1() {
1129     return null;
1131 }
1132 @Perm(ensures="none(this) in alive")
1133 public Variable getOperand2() {
1134     return null;
1136 }
1138 }ENDOFCLASS
1140 @ClassStates({@State(name = "alive")})
1142 class CastInstructionImpl {
1143     @Perm(ensures="unique(this) in alive")
1144     CastInstructionImpl() { }
1146     @Perm(requires="unique(this) in alive",
1147     ensures="unique(this) in alive")
1148     public Type getCastToTypeNode() {
1149         return null;
1151     }
1152     @Perm(requires="share(this) in alive",
1153     ensures="share(this) in alive")
1154     public Variable getOperand() {
1155         return null;
1157     }
1158     @Perm(requires="unique(this) in alive",
1159     ensures="unique(this) in alive")
1160     public LE transfer(ITACTransferFunction<LE> tf, LE value) {
1161         return null;
1163     }
1164     @Perm(requires="unique(this) in alive",
1165     ensures="unique(this) in alive")
1166     public String toString() {
1167         return null;
1169     }
1171 }ENDOFCLASS
1173 @ClassStates({@State(name = "alive")})
1175 class EclipseBrokenFieldAccess {
1176     @Perm(ensures="unique(this) in alive")
1177     EclipseBrokenFieldAccess() { }
1179     @Perm(requires="unique(this) in alive",
1180     ensures="unique(this) in alive")
1181     public SimpleName getFieldName() {
1182         return null;
1184     }
1185     @Perm(requires="share(this) in alive",
1186     ensures="share(this) in alive")
1187     public IVariableBinding resolveFieldBinding() {
1188         return null;
1190     }
1192     public boolean isImplicitThisAccess() {
1193         return 0;
1195     }
1197     public boolean isExplicitSuperAccess() {
1198         return 0;
1200     }
1201     @Perm(requires="share(this) in alive",
1202     ensures="share(this) in alive")
1203     protected Variable getAccessedInstanceInternal(IVariableBinding field) {
1204         return null;
1206     }

```

```

1208 }ENDOFCLASS
1210 @ClassStates({@State(name = "alive")})
1212 class NewArrayInstructionImpl {
1213   @Perm(ensures="unique(this) in alive")
1214   NewArrayInstructionImpl() { }
1216   @Perm(requires="unique(this) in alive",
1217     ensures="unique(this) in alive")
1218   public ArrayType getArrayType() {
1219     return null;
1221   }
1222   @Perm(requires="share(this) in alive",
1223     ensures="share(this) in alive")
1224   public List<Variable> getDimensionOperands() {
1225     return null;
1227   }
1228   @Perm(requires="unique(this) in alive",
1229     ensures="unique(this) in alive")
1230   public int getUnallocated() {
1231     return 0;
1233   }
1234   @Perm(requires="share(this) in alive",
1235     ensures="share(this) in alive")
1236   public boolean isInitialized() {
1237     return 0;
1239   }
1240   @Perm(requires="unique(this) in alive",
1241     ensures="unique(this) in alive")
1242   public int getDimensions() {
1243     return 0;
1245   }
1246   @Perm(requires="share(this) in alive",
1247     ensures="share(this) in alive")
1248   public Variable getInitOperand() {
1249     return null;
1251   }
1252   @Perm(requires="unique(this) in alive",
1253     ensures="unique(this) in alive")
1254   public LE transfer(ITACTransferFunction<LE> tf, LE value) {
1255     return null;
1257   }
1258   @Perm(requires="unique(this) in alive",
1259     ensures="unique(this) in alive")
1260   public String toString() {
1261     return null;
1263   }
1265 }ENDOFCLASS
1267 @ClassStates({@State(name = "alive")})
1269 class CompilationUnitTACs {
1270   @Perm(ensures="unique(this) in alive")
1271   CompilationUnitTACs() { }
1273   @Perm(requires="unique(this) in alive",
1274     ensures="unique(this) in alive")
1275   EclipseTAC getMethodTAC(MethodDeclaration methodDecl) {
1276     return null;
1278   }
1280 }ENDOFCLASS
1282 @ClassStates({@State(name = "alive")})
1284 class DotClassInstructionImpl {
1285   @Perm(ensures="unique(this) in alive")
1286   DotClassInstructionImpl() { }

```

```

1288 @Perm(requires="unique(this) in alive",
1289 ensures="unique(this) in alive")
1290 public Type getTypeNode() {
1291     return null;
1292 }
1293
1294 @Perm(requires="unique(this) in alive",
1295 ensures="unique(this) in alive")
1296 public LE transfer(ITACTransferFunction<LE> tf, LE value) {
1297     return null;
1298 }
1299
1300 @Perm(requires="unique(this) in alive",
1301 ensures="unique(this) in alive")
1302 public String toString() {
1303     return null;
1304 }
1305 }
1306
1307 }ENDOFCLASS
1308
1309 @ClassStates({@State(name = "alive")})
1310
1311 class EclipseImplicitFieldAccess {
1312     @Perm(ensures="unique(this) in alive")
1313     EclipseImplicitFieldAccess() { }
1314
1315     @Perm(requires="pure(this) in alive",
1316     ensures="pure(this) in alive")
1317     public SimpleName getFieldName() {
1318         return null;
1319     }
1320 }
1321
1322 @Perm(requires="full(this) in alive",
1323 ensures="full(this) in alive")
1324 public IVariableBinding resolveFieldBinding() {
1325     return null;
1326 }
1327
1328 public boolean isImplicitThisAccess() {
1329     return 0;
1330 }
1331
1332 public boolean isExplicitSuperAccess() {
1333     return 0;
1334 }
1335
1336 }
1337
1338 @Perm(requires="immutable(this) in alive",
1339 ensures="immutable(this) in alive")
1340 protected Variable getAccessedInstanceInternal(IVariableBinding field) {
1341     return null;
1342 }
1343
1344 }ENDOFCLASS
1345
1346 @ClassStates({@State(name = "alive")})
1347
1348 class EclipseThisConstructorCallInstruction {
1349     @Perm(ensures="unique(this) in alive")
1350     EclipseThisConstructorCallInstruction() { }
1351
1352     @Perm(requires="unique(this) in alive",
1353     ensures="unique(this) in alive")
1354     public KeywordVariable getConstructionObject() {
1355         return null;
1356     }
1357 }
1358
1359 @Perm(requires="unique(this) in alive",
1360 ensures="unique(this) in alive")
1361 public IMethodBinding resolveBinding() {
1362     return null;
1363 }
1364
1365 public boolean isSuperCall() {
1366     return 0;
1367 }
1368 }

```

```

1369 @Perm(requires="share(this) in alive",
1370 ensures="share(this) in alive")
1371 public List<Variable> getArgOperands() {
1372     return null;
1373 }
1374
1376 public boolean hasEnclosingInstanceSpecifier() {
1377     return 0;
1378 }
1379
1381 public Variable getEnclosingInstanceSpecifier() {
1382     return null;
1383 }
1384
1386 }ENDOFCLASS
1387
1388 @ClassStates({@State(name = "alive")})
1389
1390 class UnaryOperationImpl {
1391     @Perm(ensures="unique(this) in alive")
1392     UnaryOperationImpl() { }
1393
1394     @Perm(requires="share(this) in alive",
1395     ensures="share(this) in alive")
1396     public Variable getOperand() {
1397         return null;
1398     }
1399 }
1400 @Perm(requires="immutable(this) in alive",
1401 ensures="immutable(this) in alive")
1402 public UnaryOperator getOperator() {
1403     return null;
1404 }
1405
1406 @Perm(requires="unique(this) in alive",
1407 ensures="unique(this) in alive")
1408 public LE transfer(ITACTransferFunction<LE> tf, LE value) {
1409     return null;
1410 }
1411
1412 @Perm(requires="share(this) in alive",
1413 ensures="share(this) in alive")
1414 public String toString() {
1415     return null;
1416 }
1417
1419 }ENDOFCLASS
1420
1421 @ClassStates({@State(name = "alive")})
1422
1423 class SourceVariableDeclarationImpl {
1424     @Perm(ensures="unique(this) in alive")
1425     SourceVariableDeclarationImpl() { }
1426
1427     @Perm(requires="pure(this) in alive",
1428     ensures="pure(this) in alive")
1429     public SourceVariable getDeclaredVariable() {
1430         return null;
1431     }
1432 }
1433 @Perm(requires="unique(this) in alive",
1434 ensures="unique(this) in alive")
1435 public IVariableBinding resolveBinding() {
1436     return null;
1437 }
1438
1439 @Perm(requires="share(this) in alive",
1440 ensures="share(this) in alive")
1441 public boolean isCaughtVariable() {
1442     return 0;
1443 }
1444
1445 @Perm(requires="share(this) in alive",
1446 ensures="share(this) in alive")
1447 public boolean isFormalParameter() {
1448     return 0;
1449 }

```



```

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;
1455 }
1456 }
1457 @Perm(requires="pure(this) in alive",
1458 ensures="pure(this) in alive")
1459 public String toString() {
1460     return null;
1461 }
1462 }
1463 }ENDOFCLASS
1464 @ClassStates({@State(name = "alive")})
1465
1466 class EclipseSuperCallInstruction {
1467     @Perm(ensures="unique(this) in alive")
1468     EclipseSuperCallInstruction() { }
1469
1470     @Perm(requires="share(this) in alive",
1471     ensures="share(this) in alive")
1472     public String getMethodName() {
1473         return null;
1474     }
1475 }
1476 @Perm(requires="unique(this) in alive",
1477 ensures="unique(this) in alive")
1478 public Variable getReceiverOperand() {
1479     return null;
1480 }
1481 }
1482 @Perm(requires="unique(this) in alive",
1483 ensures="unique(this) in alive")
1484 public IMethodBinding resolveBinding() {
1485     return null;
1486 }
1487 }
1488 }
1489
1490 public boolean isSuperCall() {
1491     return 0;
1492 }
1493 }
1494 @Perm(requires="share(this) in alive",
1495 ensures="share(this) in alive")
1496 public List<Variable> getArgOperands() {
1497     return null;
1498 }
1499 }
1500 }
1501 }ENDOFCLASS
1502 @ClassStates({@State(name = "alive")})
1503
1504 class AbstractBinaryOperation {
1505     @Perm(ensures="unique(this) in alive")
1506     AbstractBinaryOperation() { }
1507
1508     @Perm(ensures="none(this) in alive")
1509     public BinaryOperator getOperator() {
1510         return null;
1511     }
1512 }
1513 @Perm(requires="unique(this) in alive",
1514 ensures="unique(this) in alive")
1515 public LE transfer(ITACTransferFunction<LE> tf, LE value) {
1516     return null;
1517 }
1518 }
1519 }
1520
1521 public String toString() {
1522     return null;
1523 }
1524 }
1525 }ENDOFCLASS
1526 @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;
1539   }

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

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

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

1558   @Perm(requires="share(this) in alive",
1559         ensures="share(this) in alive")
1560   public String toString() {
1561     return null;
1562   }
1563 }
1564 }ENDOFCLASS

1566 @ClassStates({@State(name = "alive")})

1568 class ThisVariable {
1569   @Perm(ensures="unique(this) in alive")
1570   ThisVariable() { }

1572   @Perm(requires="pure(this) in alive",
1573         ensures="pure(this) in alive")
1574   public boolean isImplicit() {
1575     return 0;
1576   }
1577 }

1579 @Perm(requires="unique(this) in alive",
1580       ensures="unique(this) in alive")
1581 public void explicitQualifier(Name qualifier) {
1582 }
1583 }
1584 }ENDOFCLASS

1586 @ClassStates({@State(name = "alive")})

1588 class KeywordVariable {
1589   @Perm(ensures="unique(this) in alive")
1590   KeywordVariable() { }

1592   @Perm(requires="share(this) in alive",
1593         ensures="share(this) in alive")
1594   protected void setQualifier(Name qualifier) {
1595 }
1596 }
1597 }ENDOFCLASS

1600 @ClassStates({@State(name = "alive")})

1602 class EclipseReferenceFieldAccess {
1603   @Perm(ensures="unique(this) in alive")
1604   EclipseReferenceFieldAccess() { }

1606   @Perm(requires="unique(this) in alive",
1607         ensures="unique(this) in alive")
1608   public SimpleName getFieldName() {
1609     return null;
1610   }
1611 }

```

```

1613 }
1614 @Perm(requires="unique(this) in alive",
1615 ensures="unique(this) in alive")
1616 public IVariableBinding resolveFieldBinding() {
1617     return null;
1618 }
1619 }
1620
1621 public boolean isImplicitThisAccess() {
1622     return 0;
1623 }
1624 }
1625
1626 public boolean isExplicitSuperAccess() {
1627     return 0;
1628 }
1629 }
1630 @Perm(requires="share(this) in alive",
1631 ensures="share(this) in alive")
1632 protected Variable getAccessedInstanceInternal(IVariableBinding field) {
1633     return null;
1634 }
1635 }
1636 }
1637 }ENDOFCLASS

```