### Bucketing examples from the Tiarks Benchmark of Type-3 Clones

#### Table of Contents

<table>
<thead>
<tr>
<th>Case</th>
<th>File and Lines Details</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>ConstantPool.java (lines 1645 – 1651)</td>
<td>2</td>
</tr>
<tr>
<td>23</td>
<td>BinaryExpression.java (lines 843 – 888)</td>
<td>3</td>
</tr>
<tr>
<td>25</td>
<td>BinaryExpression.java (lines 4472 – 4479)</td>
<td>5</td>
</tr>
<tr>
<td>28</td>
<td>BasicMenuitemUI.java (lines 211 – 217)</td>
<td>6</td>
</tr>
<tr>
<td>29</td>
<td>AccessibleHTML.java (lines 1443 – 1456) // Version AA</td>
<td>7</td>
</tr>
<tr>
<td>30</td>
<td>JTableHeader.java (lines 830 – 839)</td>
<td>8</td>
</tr>
<tr>
<td>32</td>
<td>BasicPopupMenuUI.java (lines 74 – 81) // Version AA</td>
<td>9</td>
</tr>
<tr>
<td>33</td>
<td>BasicTextFieldUI.java (lines 132 – 181)</td>
<td>10</td>
</tr>
<tr>
<td>36</td>
<td>ConstantPool.java (lines 1860 – 1880)</td>
<td>13</td>
</tr>
<tr>
<td>38</td>
<td>ConstantPool.java (lines 2660 – 2682)</td>
<td>15</td>
</tr>
<tr>
<td>39</td>
<td>BinaryExpression.java (lines 605 – 619)</td>
<td>16</td>
</tr>
<tr>
<td>40</td>
<td>ConstantPool.java (lines 2419 – 2425)</td>
<td>17</td>
</tr>
<tr>
<td>62</td>
<td>ConstantPool.java (lines 2665 – 2681)</td>
<td>18</td>
</tr>
<tr>
<td>63</td>
<td>JTable.java (lines 4594 – 4600)</td>
<td>19</td>
</tr>
<tr>
<td>74</td>
<td>MetalScrollButton.java (lines 79 – 89)</td>
<td>20</td>
</tr>
<tr>
<td>83</td>
<td>ZipOutputStream.java (lines 437 – 443)</td>
<td>21</td>
</tr>
<tr>
<td>86</td>
<td>OperatorExpression.java (lines 144 – 169)</td>
<td>22</td>
</tr>
<tr>
<td>90</td>
<td>CodeStream.java (lines 703 – 712)</td>
<td>23</td>
</tr>
<tr>
<td>93</td>
<td>FieldDeclarationPattern.java (lines 71 – 84)</td>
<td>24</td>
</tr>
<tr>
<td>97</td>
<td>ParamTagPanel.java (lines 72 – 115)</td>
<td>25</td>
</tr>
<tr>
<td>99</td>
<td>Project.java (lines 698 – 708)</td>
<td>27</td>
</tr>
<tr>
<td>101</td>
<td>compilers/Jikes.java (lines 111 – 126)</td>
<td>28</td>
</tr>
<tr>
<td>105</td>
<td>CodeStream.java (lines 355 – 364)</td>
<td>29</td>
</tr>
</tbody>
</table>
Preparatory modifications:
- Split 1645, to avoid side effects on the predicate; the predicate is now "if (nameAndTypeIndex == 0)" , and the assignment is out of scope for the transformation.

|Nodes| = 7
Nodes = [1645, 1646, 1647, 1648, 1649, 1650, 1651]
Data-dependence successors =
[1645:{1648}, 1646:{1647, 1648, 1649, 1650}, 1647:{1648, 1649, 1650, 1651}, 1649:{1650, 1651}, 1650:{1651}]
Control-dependence predecessors =
[1646:{1645}, 1647:{1645}, 1648:{1645}, 1649:{1645}, 1650:{1645}, 1651:{1645}]
|M| = 4
M = [1645, 1649, 1650, 1651]
|PDG data-dependence edges| = 12
|PDG control-dependence edges| = 6
|Extended constraints| = 17
Extended constraints =
[1645:{1648}, 1646:{1647, 1648, 1649, 1650, 1651}, 1647:{1648, 1649, 1650, 1651}, 1648:{1645}, 1649:{1645, 1650, 1651}, 1650:{1645, 1651}, 1651:{1645}]
Inverted extended constraints =
[1647:{1646}, 1648:{1645, 1646, 1647, 1648, 1649, 1650, 1651}, 1649:{1646, 1647, 1650, 1651}, 1650:{1646, 1647, 1649, 1650}]
|Promoted| = 0
Promoted = []
Before = [1645, 1646, 1647]
Marked = [1645, 1649, 1650, 1651]
After = [1645, 1648]
Arbitrary = []
Predicates = [1645]
public void generateOptimizedLogicalAnd(BlockScope currentScope, CodeStream codeStream, Label trueLabel, Label falseLabel, boolean valueRequired) {
    int pc = codeStream.position;
    Constant condConst;
    if ((left.implicitConversion & 0xF) == T_boolean) {
        if ((condConst = left.conditionalConstant()) != NotAConstant) {
            if (condConst.booleanValue() == true) {
                // <something equivalent to true> & x
                left.generateOptimizedBoolean(currentScope, codeStream, trueLabel, falseLabel, false);
                if ((bits & OnlyValueRequiredMASK) != 0) {
                    right.generateCode(currentScope, codeStream, valueRequired);
                } else {
                    right.generateOptimizedBoolean(currentScope, codeStream, trueLabel, falseLabel, valueRequired);
                }
            } else {
                // <something equivalent to false> & x
                left.generateOptimizedBoolean(currentScope, codeStream, trueLabel, falseLabel, false);
                right.generateOptimizedBoolean(currentScope, codeStream, trueLabel, falseLabel, false);
                if (valueRequired) {
                    if ((bits & OnlyValueRequiredMASK) != 0) {
                        codeStream.iconst_0();
                    } else {
                        if (falseLabel != null) {
                            // implicit falling through the TRUE case
                            codeStream.goto_(falseLabel);
                        }
                    }
                }
            }
            codeStream.recordPositionsFrom(pc, this.sourceStart);
            return;
        }
        if ((condConst = right.conditionalConstant()) != NotAConstant) {
            if (condConst.booleanValue() == true) {
                // x & <something equivalent to true>
if ((bits & OnlyValueRequiredMASK) != 0) {
    left.generateCode(currentScope, codeStream, valueRequired);
} else {
    left.generateOptimizedBoolean(currentScope, codeStream,
    trueLabel, falseLabel, valueRequired);
}
right.generateOptimizedBoolean(currentScope, codeStream,
trueLabel, falseLabel, false);
}
else {
// x & <something equivalent to false>
left.generateOptimizedBoolean(currentScope, codeStream,
trueLabel, falseLabel, false);
right.generateOptimizedBoolean(currentScope, codeStream,
trueLabel, falseLabel, false);
if (valueRequired) {
    if ((bits & OnlyValueRequiredMASK) != 0) {
        codeStream.iconst_0();
    } else {
        if (falseLabel != null) {
// implicit falling through the TRUE case
            codeStream.goto_(falseLabel); } } } }
    codeStream.recordPositionsFrom(pc, this.sourceStart);
    return; }
}

|Nodes| = 35
Nodes = [844, 845, 846, 847, 848, 849, 851, 852, 853, 855, 858, 859, 860, 861, 862,
864, 866, 867, 868, 869, 870, 871, 873, 874, 876, 877, 880, 881, 882, 883, 884, 886,
888, 889, 890]
Data-dependence successors =
877:{889},880:{881},881:{884,888,889},884:{889},888:{889}]
Control-dependence predecessors =
|M| = 9
M = [844, 847, 851, 858, 867, 869, 871, 881, 889]
|PDG data-dependence edges| = 31
|PDG control-dependence edges| = 34
K = 4
Mk is [844, 847, 851, 858, 869]
Before = []
Marked = [851, 858, 869, 847, 844]
After = [881, 880, 855, 884, 853, 889, 859, 888, 890, 862, 866, 867, 874, 876, 877,
845]
Arbitrary = [890, 845]
Case 25. BinaryExpression.java (lines 4472 - 4479)

```java
/**
 * Sets the enabled state of the object.
 */
* @param b if true, enables this object; otherwise, disables it
*/
public void setEnabled(boolean b) {
    AccessibleContext ac = getCurrentAccessibleContext();
    if (ac instanceof AccessibleComponent) {
        ((AccessibleComponent) ac).setEnabled(b);
    } else {
        Component c = getCurrentComponent();
        if (c != null) {
            c.setEnabled(b); } } }
```

| Nodes | = 6
| Nodes = [4473, 4474, 4475, 4477, 4478, 4479] |
Data-dependence successors = [4473:{4474,4475},4477:{4478,4479}] |
Control-dependence predecessors = [4475:{4474},4477:{4474},4478:{4474},4479:{4478}] |
M = [4473, 4474, 4477] // 4478 is identical too, but it is a predicate with no marked controlled statements |
| PDG data-dependence edges| = 4 |
| PDG control-dependence edges| = 4 |
| Extended constraints| = 7 |
Extended constraints = [4473:{4474,4475,4477,4478,4479},4477:{4478,4479}] |
Inverted extended constraints = [4474:{4473},4475:{4473},4477:{4473},4478:{4473,4477},4479:{4473,4477}] |
| Promoted| = 0 |
Promoted = [] |
Before = [] |
Marked = [4473, 4474, 4477] |
After = [4474, 4475, 4478, 4479] |
Arbitrary = [] |
Predicates = [4474, 4478]
public Dimension getMinimumSize((Component c) {
    Dimension d = null;
    View v = (View) c.getClientProperty(BasicHTML.propertyKey);
    if (v != null) {
        d = getPreferredSize(c);
        d.width -= v.getPreferredSpan(View.X_AXIS) - v.getMinimumSpan(View.X_AXIS);
    }
    return d;
}

|Nodes| = 6
Nodes = [212, 213, 214, 215, 216, 217]
Data-dependence successors = [212:{215,217},213:{214,216},215:{216,217}]
Control-dependence predecessors = [215:{214},216:{214}]
|M| = 5
M = [212, 213, 214, 215, 217]
|PDG data-dependence edges| = 6
|PDG control-dependence edges| = 2
|Extended constraints| = 9
Extended constraints = [212:{215,216,217},213:{214,215,216,217},215:{216,217}]
|Promoted| = 0
Promoted = []
Before = []
Marked = [212, 213, 214, 215, 217]
After = [214, 216]
Arbitrary = []
Predicates = [214]
/**
 * Returns a boolean value indicating whether the specified row
 * is selected.
 * @param r zero-based row of the table
 * @return the boolean value true if the specified row is selected.
 * Otherwise, false.
 */

public boolean isAccessibleRowSelected(int r) {
    if (validateIfNecessary()) {
        if (r < 0 || r >= getAccessibleRowCount()) {
            return false;
        }
        int nColumns = getAccessibleColumnCount();
        TableCellElementInfo startCell = getCell(r, 0);
        if (startCell == null) {
            return false;
        }
        int start = startCell.getElement().getStartOffset();
        TableCellElementInfo endCell = getCell(r, nColumns - 1);
        if (endCell == null) {
            return false;
        }
        int end = endCell.getElement().getEndOffset();
        return start >= editor.getSelectionStart() &&
        end <= editor.getSelectionEnd();
    }
    return false;
}

// in this example the after is beyond the return; would be correct with a goto

|Nodes| = 15
|Nodes| = [1, 1443, 1444, 1445, 1446, 1447, 1448, 1449, 1450, 1451, 1452, 1453, 1454, 1455, 1457]

Data-dependence successors =
[1:{1443}, 1446:{1451}, 1447:{1448, 1450}, 1450:{1455}, 1451:{1452, 1454}, 1454:{1455}]

Control-dependence predecessors =

|M| = 4
|M| = [1, 1450, 1454, 1455]

|PDG data-dependence edges| = 8
|PDG control-dependence edges| = 21

|Extended constraints| = 38
|Extended constraints| =
[1:{1443, 1444, 1445, 1446, 1447, 1448, 1449, 1450, 1451, 1452, 1453, 1454, 1455, 1457}, 1446:{1451, 1452, 1453, 1454, 1455, 1457}, 1447:{1448, 1449, 1450, 1451, 1452, 1453, 1454, 1455, 1457}, 1450:{1455, 1457}, 1451:{1452, 1453, 1454, 1455, 1457}, 1454:{1455, 1457}]

Inverted extended constraints =
[1443:{1}, 1444:{1}, 1445:{1}, 1446:{1}, 1447:{1}, 1448:{1, 1447}, 1449:{1, 1447}, 1450:{1, 1447}, 1451:{1, 1446, 1447, 1451}, 1452:{1, 1446, 1447, 1451}, 1453:{1, 1446, 1447, 1451}, 1454:{1, 1446, 1447, 1451}, 1455:{1, 1446, 1447, 1450, 1451, 1454}, 1457:{1, 1446, 1447, 1450, 1451, 1454}]

|Promoted| = 9
|Promoted| = [1443, 1444, 1445, 1446, 1447, 1448, 1449, 1451, 1452]

Before = []
|Before| = [1, 1443, 1444, 1445, 1446, 1447, 1448, 1449, 1450, 1451, 1452, 1454, 1455]

After = [1443, 1444, 1445, 1448, 1449, 1452, 1453, 1455, 1457]

Arbitrary = []
|Arbitrary| = [1443, 1444, 1445, 1448, 1449, 1452, 1455]
Case 30. JTableHeader.java (lines 830 – 839)

```java
public Color getBackground() {
    AccessibleContext ac = getCurrentAccessibleContext();
    if (ac instanceof AccessibleComponent) {
        return ((AccessibleComponent) ac).getBackground();
    } else {
        Component c = getCurrentComponent();
        if (c != null) {
            return c.getBackground();
        } else {
            return null; }
    }
}
```

|Nodes| = 7
Nodes = [831, 832, 833, 835, 836, 837, 839]
Data-dependence successors = [831:{832,833},835:{836,837}]
Control-dependence predecessors =
[833:{832},835:{832},836:{832},837:{836},839:{836}]
|M| = 2
M = [831, 835]
|PDG data-dependence edges| = 4
|PDG control-dependence edges| = 5
|Extended constraints| = 9
Extended constraints = [831:{832,833,835,836,837,839},835:{836,837,839}]
Inverted extended constraints =
[832:{831},833:{831},835:{831},836:{831,835},837:{831,835},839:{831,835}]
|Promoted| = 1
Promoted = [832]
Before = []
Marked = [831, 832, 835]
After = [832, 833, 836, 837, 839]
Arbitrary = []
Predicates = [832, 836]
static ActionMap getActionMap() {
    ActionMap map = (ActionMap)UIManager.get("PopupMenu.actionMap");
    if (map == null) {
        map = createActionMap();
        if (map != null) {
            UIManager.getLookAndFeelDefaults().put("PopupMenu.actionMap", map); }
    }
    return map; }

|Nodes| = 6
Nodes = [75, 76, 77, 78, 79, 81]
Data-dependence successors = [75:{76,77,81},76:{77},77:{78,79,81}]
Control-dependence predecessors = [77:{76},78:{76},79:{78}]
|M| = 3
M = [76, 77, 81]
|PDG data-dependence edges| = 7
|PDG control-dependence edges| = 3
|Extended constraints| = 21
Inverted extended constraints = [76:{75},77:{75,76,77,78,79},78:{75,76,77,78,79},79:{75,76,77,78,79},81:{75,76,77,78,79}]
|Promoted| = 2
Promoted = [78, 79]
|Before| = [75]
Marked = [76, 77, 78, 79, 81]
After = []
|Arbitrary| = []
Predicates = [76, 78]
Case 33. BasicTextFieldUI.java (lines 132 – 181)

The method signature is on line 132 and the return null is on line 181:

```java
Shape adjustAllocation(Shape a) {
    if (a != null) {
        Rectangle bounds = a.getBounds();
        int vspan = (int) getPreferredSpan(Y_AXIS);
        int hspan = (int) getPreferredSpan(X_AXIS);
        if (bounds.height != vspan) {
            int slop = bounds.height - vspan;
            bounds.y += slop / 2;
            bounds.height -= slop;
        }
        // horizontal adjustments
        Component c = getContainer();
        if (c instanceof JTextField) {
            JTextField field = (JTextField) c;
            BoundedRangeModel vis = field.getHorizontalVisibility();
            int max = Math.max(hspan, bounds.width);
            int value = vis.getValue();
            int extent = Math.min(max, bounds.width - 1);
            if ((value + extent) > max) {
                value = max - extent;
            }
            vis.setRangeProperties(value, extent,
                max, false);
            if (hspan < bounds.width) {
                // horizontally align the interior
                int slop = bounds.width - 1 - hspan;
                int align = ((JTextField)c).getHorizontalAlignment();
                if(isLeftToRight(c)) {
                    if(align==LEADING) {
                        align = LEFT;
                    } else if(align==TRAILING) {
                        align = RIGHT;
                    }
                } else {
                    if(align==LEADING) {
                        align = RIGHT;
                    } else if(align==TRAILING) {
                        align = LEFT;
                    }
                }
                switch (align) {
                    case SwingConstants.CENTER:
                        bounds.x += slop / 2;
                        break;
                    case SwingConstants.LEADING:
                        break;
                    case SwingConstants.TRAILING:
                        break;
                    default:
                        break;
                }
            }
```
bounds.width -= slop;
break;

case SwingConstants.RIGHT:
    bounds.x += slop;
    bounds.width -= slop;
    break;
}

} else {
    // adjust the allocation to match the bounded range.

    bounds.width = hspan;
    bounds.x -= vis.getValue();
}

return bounds;  
return null;

//Since all lines are identical except one that does the same as the clone, we will mark it all and put this under "not interesting"
//This example contains a switch statement.
//The switch will be represented as one line containing all dependences of the block

|Nodes| = 33


Data-dependence successors =

Control-dependence predecessors =

|M| = 32


|PDG data-dependence edges| = 68
|PDG control-dependence edges| = 33

|Extended constraints| = 165

Extended constraints =

|Inverted extended constraints| = 33

Inverted extended constraints =
| Promoted | = 1 |
| Promoted = [157] |
| Before = [] |
| After = [] |
| Arbitrary = [] |
| Predicates = [133, 137, 143, 149, 153, 157, 158, 160, 163, 165] |
public int literalIndexForJavaLangShortYPE() {
  int index;
  if ((index = wellKnownFields[TYPE_SHORT_FIELD]) == 0) {
    int nameAndTypeIndex;
    int classIndex;
    // The entry doesn't exit yet
    classIndex = literalIndexForJavaLangShort();
    if ((nameAndTypeIndex = wellKnownFieldNameAndTypes[TYPE_JAVALANGCLASS_NAME_AND_TYPE]) == 0) {
      int nameIndex = literalIndex(QualifiedNamesConstants.TYPE);
      int typeIndex = literalIndex(QualifiedNamesConstants.JavaLangClassSignature);
      nameAndTypeIndex = wellKnownFieldNameAndTypes[TYPE_JAVALANGCLASS_NAME_AND_TYPE] = currentIndex++;
      writeU1(NameAndTypeTag);
      writeU2(nameIndex);
      writeU2(typeIndex); }
    index = wellKnownFields[TYPE_SHORT_FIELD] = currentIndex++;
    if (index > 0xFFFF){
      this.classFile.referenceBinding.scope.problemReporter().noMoreAvailableSpaceInConstantPool(this.classFile);
    }
    writeU1(FieldRefTag);
    writeU2(classIndex);
  } return index; }

Preparatory modifications:
- Added two nodes "1" and "2" to avoid side effects in the predicate:
  - "1" before 1862: "index = wellKnownFields[TYPE_SHORT_FIELD])"
  - "2" before 1867: "nameAndTypeIndex = wellKnownFieldNameAndTypes[TYPE_JAVALANGCLASS_NAME_AND_TYPE]"
  - 1862: "if (index==0)"
  - 1867: "if (nameAndTypeIndex==0)"

|Nodes| = 16
Nodes = [1862, 1866, 2, 1867, 1868, 1869, 1870, 1871, 1872, 1873, 1874, 1875, 1876, 1877, 1878, 1879]
Data-dependence successors =
[2:1867,1870,1879],1862:1874,1866:1868,1874,1876,1877),1867:1870),1868:1869,1872,1873,1874,1875,1877,1878,1879]
Control-dependence predecessors =
$M$ = [1862, 1867, 1871, 1872, 1873, 1875, 1876, 1878, 1879]

Extended constraints =

$\{2,\{1867,1868,1869,1870,1871,1872,1873,1874,1875,1876,1877,1878,1879\}\}$
$\{1862,1865,1875\}$

Inverted extended constraints =

$\{2,\{1867,1868,1869,1870,1871,1872,1873,1874,1875,1876,1877,1878,1879\}\}$
$\{1862,1865\}$

Promoted = [1870, 1874, 1877]
Before = [2, 1862, 1865, 1867, 1868, 1869]
Marked = [1862, 1865, 1867, 1868, 1869]
After = []
Predicates = [1862, 1865, 1875]
|Nodes| = 15
|Nodes = [2665, 2666, 2, 2667, 2668, 2669, 2670, 2671, 2672, 2673, 2674, 2677, 2679, 2681, 2682]

Data-dependence successors =

Control-dependence predecessors =

|M| = 9
M = [2665, 2666, 2667, 2668, 2669, 2670, 2671, 2672, 2673, 2677, 2679, 2681, 2682]

|PDG data-dependence edges| = 22

|PDG control-dependence edges| = 13

|Extended constraints| = 84

Extended constraints =

Inverted extended constraints =

|Promoted| = 2
Promoted = [2670, 2674]

Before = [2, 2665, 2666, 2667, 2668, 2669]

Marked = [2665, 2667, 2670, 2671, 2672, 2673, 2674, 2677, 2679, 2681, 2682]

After = []

Arbitrary = []

Predicates = [2665, 2667]
Case 39. BinaryExpression.java (lines 605 – 619)

```java
604 if ((right.constant != NotAConstant)
605     && (right.constant.intValue() == 0)) {
606     left.generateCode(currentScope, codeStream, valueRequired);
607     if (valueRequired){
608         if (falseLabel == null){
609             if (trueLabel != null){
610                 // implicitly falling through the FALSE case
611                 codeStream.ifge(trueLabel); }
612             } else {
613                 if (trueLabel == null){
614                     // implicitly falling through the TRUE case
615                     codeStream.iflt(falseLabel);
616                 } else {
617                     codeStream.recordPositionsFrom(pc, this.sourceStart);
618                     return; }
619                 }

|Nodes| = 8
Nodes = [606, 607, 608, 609, 611, 613, 615, 618]
Data-dependence successors = [606:{611,615,618},611:{618},615:{618}]
Control-dependence predecessors =
[608:{607},609:{608},611:{609},613:{608},615:{613}]
|M| = 2
M = [606, 618] // Nodes 607,608,609,613 are predicates with no marked controlled statements
|PDG data-dependence edges| = 5
|PDG control-dependence edges| = 5
|Extended constraints| = 5
Extended constraints = [606:{611,615,618},611:{618},615:{618}]
Inverted extended constraints = [611:{606},615:{606},618:{606,611,615}]
|Promoted| = 6
Promoted = [607, 608, 609, 611, 613, 615]
Before = []
Marked = [606, 607, 608, 609, 611, 613, 615, 618]
After = []
Arbitrary = []
Predicates = [607, 608, 609, 613]
```
Case 40. ConstantPool.java (lines 2419 – 2425)

```java
if ((nameAndTypeIndex = wellKnownMethodNameAndTypes[EXIT_METHOD_NAME_AND_TYPE]) == 0) {
    int nameIndex = literalIndex(QualifiedNamesConstants.Exit);
    int typeIndex = literalIndex(QualifiedNamesConstants.ExitIntSignature);
    nameAndTypeIndex = wellKnownMethodNameAndTypes[EXIT_METHOD_NAME_AND_TYPE] = currentIndex++;
    writeU1(nameAndTypeTag);
    writeU2(nameIndex);
    writeU2(typeIndex);
}
```

Preparatory modifications:
- Added 2418 before the clone to avoid side effects on the predicate.

|Nodes| = 7
Nodes = [2419, 2420, 2421, 2422, 2423, 2424, 2425]
Data-dependence successors =
[2419:{2422},2420:{2421,2424},2421:{2422,2423,2425},2423:{2424},2424:{2425}]
Control-dependence predecessors =
[2420:{2419},2421:{2419},2422:{2419},2423:{2419},2424:{2419},2425:{2419}]
|M| = 4
M = [2419, 2423, 2424, 2425]
|PDG data-dependence edges| = 8
|PDG control-dependence edges| = 6
|Extended constraints| = 17
Extended constraints =
[2419:{2422},2420:{2421,2422,2423,2424,2425},2421:{2422,2423,2424,2425},2422:{2422},
2423:{2422,2424,2425},2424:{2422,2425},2425:{2422}]
Inverted extended constraints =
[2421:{2420},2422:{2419,2420,2421,2422,2423,2424,2425},2423:{2420,2421},2424:{2420,2421,2423,2424,2425}]
|Promoted| = 0
Promoted = []
Before = [2419, 2420, 2421]
Marked = [2419, 2423, 2424, 2425]
After = [2419, 2422]
Arbitrary = []
Predicates = [2419]
Preparatory modifications:
  • Added nodes 1,2 before the predicates to avoid side effects

|Nodes| = 14
Nodes = [2665, 2666, 2, 2667, 2668, 2669, 2670, 2671, 2672, 2673, 2674, 2677, 2679, 2681]
Data-dependence successors =
Control-dependence predecessors =
|M| = 8
M = [2665, 2667, 2671, 2672, 2673, 2677, 2679, 2681]
|PDG data-dependence edges| = 21
|PDG control-dependence edges| = 13
|Extended constraints| = 70
Extended constraints =
Inverted extended constraints =
|Promoted| = 1
Promoted = [2670]
Before = [2, 2665, 2666, 2667, 2668, 2669]
Marked = [2665, 2667, 2670, 2671, 2672, 2673, 2677, 2679, 2681]
After = [2665, 2674]
Arbitrary = []
Predicates = [2665, 2667]
Case 63. JTable.java (lines 4594 – 4600)

```java
public void setBounds(Rectangle r) {
    AccessibleContext ac = getCurrentAccessibleContext();
    if (ac instanceof AccessibleComponent) {
        ((AccessibleComponent) ac).setBounds(r);
    } else {
        Component c = getCurrentComponent();
        if (c != null) {
            c.setBounds(r);
        }
    }
}
```

|Nodes| = 7  
Nodes = [4594, 4595, 4596, 4598, 4599, 4600]  
Data-dependence successors = [4594:{4595,4596},4598:{4599,4600}]  
Control-dependence predecessors = [4596:{4595},4598:{4595},4599:{4595},4600:{4599}]  
|M| = 2  
M = [4594, 4598] // Nodes 4595,4599 are predicates with no marked controlled statements  
|PDG data-dependence edges| = 4  
|PDG control-dependence edges| = 4  
|Extended constraints| = 7  
Extended constraints = [4594:{4595,4596,4598,4599,4600},4598:{4599,4600}]  
Inverted extended constraints = [4595:{4594},4596:{4594},4598:{4594},4599:{4594,4598},4600:{4594,4598}]  
|Promoted| = 1  
Promoted = [4595]  
Before = []  
Marked = [4594, 4595, 4598]  
After = [4595, 4596, 4599, 4600]  
Arbitrary = []  
Predicates = [4595, 4599]
```java
if (isEnabled) {
    g.setColor( highlightColor );
    if ( !isPressed ) {
        g.drawLine( 1, 1, width - 3, 1 );
        g.drawLine( 1, 1, 1, height - 1 );
    }
    g.drawLine( width - 1, 1, width - 1, height - 1 );
    g.setColor( shadowColor );
    g.drawLine( 0, 0, width - 2, 0 );
    g.drawLine( 0, 0, 0, height - 1 );
    g.drawLine( width - 2, 2, width - 2, height - 1 );
} else {
```

|Nodes| = 11
Nodes = [79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 90]
Data-dependence successors =
[80:{82,84,85},82:{83,85},83:{84,85},84:{85,86},85:{86,87,88},86:{87},87:{88}]
Control-dependence predecessors =
[80:{79},81:{79},82:{81},83:{81},84:{79},85:{79},86:{79},87:{79},88:{79},90:{79}]
|M| = 3
M = [79, 80, 85] // Node 81 is a predicate with no marked controlled statements
|PDG data-dependence edges| = 14
|PDG control-dependence edges| = 10
|Extended constraints| = 28
Extended constraints =
[80:{82,83,84,85,86,87,88},82:{83,84,85,86,87,88},83:{84,85,86,87,88},84:{85,86,87,88}]
Inverted extended constraints =
[82:{80},83:{80,82},84:{80,82,83},85:{80,82,83,84},86:{80,82,83,84,85},87:{80,82,83,84,85,86},88:{80,82,83,84,85,86,87}]
|Promoted| = 4
Promoted = [81, 82, 83, 84]
Before = []
Marked = [79, 80, 81, 82, 83, 84, 85]
After = [79, 86, 87, 88, 90]
Arbitrary = [90]
Predicates = [79, 81]
Case 83. ZipOutputStream.java (lines 437 – 443)

437        written += 4;
438        // compression method
439        out.write((new ZipShort(ze.getMethod())).getBytes());
440        written += 2;
441        // last mod. time and date
442        out.write(toDosTime(new Date(ze.getTime())).getBytes());
443        written += 4;

// the group extraction would fail (out of order)

[Nodes] = 5
Nodes = [437, 439, 440, 442, 443]
Data-dependence successors = [437:{440}, 439:{442}, 440:{443}]
Control-dependence predecessors = []
|M| = 3
M = [437, 440, 443]
|PDG data-dependence edges| = 3
|PDG control-dependence edges| = 0
|Extended constraints| = 4
Extended constraints = [437:{440,443}, 439:{442}, 440:{443}]
Inverted extended constraints = [440:{437}, 442:{439}, 443:{437,440}]
|Promoted| = 0
Promoted = []
Before = []
Marked = [437, 440, 443]
After = [439, 442]
Arbitrary = [439, 442]
Predicates = []
Case 86. OperatorExpression.java (lines 144 - 169)

```java
table[(T_byte<<4)+T_byte] = (Byte2Int<<12) + (Byte2Int<<4) + T_int;
table[(T_byte<<4)+T_long] = (Byte2Long<<12) + (Long2Long<<4) + T_long;
table[(T_byte<<4)+T_short] = (Byte2Int<<12) + (Short2Int<<4) + T_int;
table[(T_byte<<4)+T_char] = (Byte2Int<<12) + (Char2Int<<4) + T_int;
table[(T_long<<4)+T_byte] = (Long2Long<<12) + (Byte2Long<<4) + T_long;
table[(T_long<<4)+T_long] = (Long2Long<<12) + (Long2Long<<4) + T_long;
table[(T_long<<4)+T_short] = (Long2Long<<12) + (Short2Long<<4) + T_long);
table[(T_long<<4)+T_char] = (Long2Long<<12) + (Char2Long<<4) + T_long;
table[(T_long<<4)+T_int] = (Long2Long<<12) + (Int2Long<<4) + T_long;
table[(T_short<<4)+T_byte] = (Short2Int<<12) + (Byte2Int<<4) + T_int;
table[(T_short<<4)+T_long] = (Short2Long<<12) + (Long2Long<<4) + T_long;
table[(T_short<<4)+T_short] = (Short2Int<<12) + (Short2Int<<4) + T_int;
table[(T_short<<4)+T_char] = (Short2Int<<12) + (Char2Int<<4) + T_int;
table[(T_short<<4)+T_int] = (Short2Int<<12) + (Int2Int<<4) + T_int;
table[(T_boolean<<4)+T_boolean] = (Boolean2Boolean << 12) + (Boolean2Boolean << 4) + T_boolean;
table[(T_char<<4)+T_byte] = (Char2Int<<12) + (Byte2Int<<4) + T_int;
table[(T_char<<4)+T_long] = (Char2Long<<12) + (Long2Long<<4) + T_long;
table[(T_char<<4)+T_short] = (Char2Int<<12) + (Short2Int<<4) + T_int;
table[(T_char<<4)+T_char] = (Char2Int<<12) + (Char2Int<<4) + T_int;
table[(T_char<<4)+T_int] = (Char2Int<<12) + (Int2Int<<4) + T_int;
table[(T_int<<4)+T_byte] = (Int2Int<<12) + (Byte2Int<<4) + T_int;
table[(T_int<<4)+T_long] = (Int2Long<<12) + (Long2Long<<4) + T_long;
table[(T_int<<4)+T_short] = (Int2Int<<12) + (Short2Int<<4) + T_int;
table[(T_int<<4)+T_char] = (Int2Int<<12) + (Char2Int<<4) + T_int;
table[(T_int<<4)+T_int] = (Int2Int<<12) + (Int2Int<<4) + T_int;
table[(T_boolean<<4)+T_boolean] = (Boolean2Boolean << 12) + (Int2Int<<4) + T_int;
```

//This example belongs to the non-interesting promotions category: the hammock contains lines: 160-166 and all lines are marked.
//HOWEVER, the right clone is interesting since the lines that are marked there are not-contiguous (depending on the accuracy of the dependence analysis).

|Nodes| = 7
Nodes = [160, 161, 162, 163, 164, 165, 166]
Data-dependence successors = [160:{161,162,163,164,165,166}]
Control-dependence predecessors = []
|M| = 7
M = [160, 161, 162, 163, 164, 165, 166]
|PDG data-dependence edges| = 6
|PDG control-dependence edges| = 0
|Extended constraints| = 21
Extended constraints = [160:{161,162,163,164,165,166},161:{160,161,162,163,164,165,166},162:{160,161,162,163,164,165,166}]
Inverted extended constraints = [161:{160,162,163,164,165,166},162:{160,161,162,163,164,165,166},163:{160,161,162,163,164,165,166}]
|Promoted| = 0
Promoted = []
Before = []
Marked = [160, 161, 162, 163, 164, 165, 166]
After = []
Arbitrary = []
Predicates = []
Case 90. CodeStream.java (lines 703 – 712)

```java
703  final public void dstore_2() {
704      countLabels = 0;
705      stackDepth -= 2;
706      if (maxLocals < 4) {
707          maxLocals = 4;
708      }
709      try {
710          position++;
711          bCodeStream[classFileOffset++] = OPC_dstore_2;
712      } catch (IndexOutOfBoundsException e) {
713          resizeByteArray(OPC_dstore_2);
714      }
```
Case 93. FieldDeclarationPattern.java (lines 71 – 84)

```java
public boolean matchesBinary(Object binaryInfo, Object enclosingBinaryInfo) {
    if (!(binaryInfo instanceof IBinaryField)) return false;
    IBinaryField field = (IBinaryField)binaryInfo;
    // field name
    if (!this.matchesName(this.name, field.getName()))
        return false;
    // declaring type
    IBinaryType declaringType = (IBinaryType)enclosingBinaryInfo;
    if (declaringType != null) {
        char[] declaringTypeName = (char[])declaringType.getName().clone();
        charOperation.replace(declaringTypeName, '/', '.'));
        if (this.matchesType(this.declaringSimpleName, this.declaringQualification, declaringTypeName)) {
            return false; }
    }
    // field type
    String fieldTypeSignature = new String(field.getTypeName()).replace('/', '.');
```

|Nodes| = 10
Nodes = [77, 78, 79, 80, 81, 82, 84, 85, 86, 87]
Data-dependence successors = [77:{78,79},79:{80,81}]
Control-dependence predecessors = [79:{78},80:{78},81:{78},82:{81},84:{82},85:{82},86:{85},87:{86}]
|M| = 5
M = [77, 78, 79, 80, 81]
|PDG data-dependence edges| = 4
|PDG control-dependence edges| = 8
|Extended constraints| = 16
Extended constraints = [77:{78,79,80,81,82,84,85,86,87},79:{80,81,82,84,85,86,87}]
Inverted extended constraints = [78:{77},79:{77,80:{77,79},81:{77,79},82:{77,79},84:{77,79},85:{77,79},86:{77,79},87:{77,79}]
|Promoted| = 0
Promoted = []
Before = []
Marked = [77, 78, 79, 80, 81]
After = [78, 81, 82, 84, 85, 86, 87]
Arbitrary = []
Predicates = [78, 81, 82, 85, 86]
Case 97. ParamTagPanel.java (lines 72 – 115)

```java
jLabel2.setText(org.openide.util.NbBundle.getBundle(ParamTagPanel.class).getString("CTL_ParamTagPanel.jLabel2.text"));
jLabel2.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints ancor = java.awt.GridBagConstraints.WEST;
gridBagConstraints insets = new java.awt.Insets(2, 2, 1);
add(jLabel2, gridBagConstraints);
parameterComboBox.setMaximumRowCount(4);
parameterComboBox.setToolTipText(org.openide.util.NbBundle.getBundle(ParamTagPanel.class).getString("ACS_ParamTagPanel.parameterComboBox.textA11yDesc"));
parameterComboBox.setEditable(true);
parameterComboBox.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        parameterComboBoxActionPerformed(evt); }
});
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
gridBagConstraints.fill = java.awt.GridBagConstraints.HORIZONTAL;
gridBagConstraints. ancor = java.awt.GridBagConstraints.WEST;
gridBagConstraints.insets = new java.awt.Insets(2, 1, 2, 2);
add(parameterComboBox, gridBagConstraints);

jLabel1.setText(org.openide.util.NbBundle.getBundle(ParamTagPanel.class).getString("CTL_ParamTagPanel.jLabel1.text"));
jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints. ancor = java.awt.GridBagConstraints.NORTHWEST;
gridBagConstraints.insets = new java.awt.Insets(2, 2, 1);
add(jLabel1, gridBagConstraints);
descriptionTextArea.setContentType("text/html");
descriptionTextArea.addFocusListener(new java.awt.event.FocusAdapter() {
    public void focusGained(java.awt.event.FocusEvent evt) {
        descriptionTextAreaFocusGained(evt); }
    public void focusLost(java.awt.event.FocusEvent evt) {
        descriptionTextAreaFocusLost(evt); }
});
descriptionScrollPane.setViewportView(descriptionTextArea);
gridBagConstraints = new java.awt.GridBagConstraints();
gridBagConstraints.gridwidth = java.awt.GridBagConstraints.REMAINDER;
gridBagConstraints.gridheight = java.awt.GridBagConstraints.REMAINDER;
gridBagConstraints.fill = java.awt.GridBagConstraints.BOTH;
gridBagConstraints.weightx = 1.0;
gridBagConstraints.weighty = 1.0;
gridBagConstraints.insets = new java.awt.Insets(2, 1, 2, 2);
add(descriptionScrollPane, gridBagConstraints);

// this code was automatically generated

|Nodes| = 36
Nodes = [72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 107, 108, 109, 110, 111, 112, 113, 114, 115]
Data-dependence successors =
[75:{76,77,78,86},76:{86},77:{86},78:{86,91},86:{87,88,89,90,91,95,87:{95}},88:{95},89,
Control-dependence predecessors = []

\[ M = [72, 75, 76, 77, 82, 84, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 100, 101, 107, 108, 109, 110, 111, 112, 113, 114, 115] \]

\[ |\text{PDG data-dependence edges}| = 35 \]
\[ |\text{PDG control-dependence edges}| = 0 \]
\[ |\text{Extended constraints}| = 194 \]

Extended constraints =

Inverted extended constraints =

Promoted = 1

Promoted = [78]

Before = []

Marked = [72, 75, 76, 77, 78, 82, 84, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 100, 101, 107, 108, 109, 110, 111, 112, 113, 114, 115]

After = [73, 74, 79, 80, 81, 99]

Arbitrary = [73, 74, 79, 80, 81, 99]

Predicates = []
Case 99. Project.java (lines 698 – 708)

protected void fireBuildFinished(Throwable exception) {
    BuildEvent event = new BuildEvent(this);
    event.setException(exception);
    for (int i = 0; i < listeners.size(); i++) {
        BuildListener listener = (BuildListener) listeners.elementAt(i);
        listener.buildFinished(event); }
}

protected void fireTargetStarted(Target target) {
    BuildEvent event = new BuildEvent(target);
    for (int i = 0; i < listeners.size(); i++) {
        BuildListener listener = (BuildListener) listeners.elementAt(i);
        listener.targetStarted(event); }
}

Let’s examine each method separately:

//this is the first part of the clone 99 (lines 698-703)

[Nodes] = 3
Nodes = [701, 702, 703]
Data-dependence successors = [701:{701,702},702:{703},703:{702}]
Control-dependence predecessors = [702:{701},703:{701}]
|M| = 2
M = [701, 702]
|PDG data-dependence edges| = 4
|PDG control-dependence edges| = 2
|Extended constraints| = 9
Extended constraints = [701:{701,702,703},702:{701,702,703},703:{701,702,703}]
Inverted extended constraints = [701:{701,702,703},702:{701,702,703},703:{701,702,703}]
|Promoted| = 1
Promoted = [703]
Before = []
Marked = [701, 702, 703]
After = []
Arbitrary = []
Predicates = [701]

//this is the second part of the clone 99 (lines 704-708)

[Nodes] = 3
Nodes = [706, 707, 708]
Data-dependence successors = [706:{706,707},707:{708},708:{708}]
Control-dependence predecessors = [707:{706},708:{706}]
|M| = 2
M = [706, 707]
|PDG data-dependence edges| = 4
|PDG control-dependence edges| = 2
|Extended constraints| = 9
Extended constraints = [706:{706,707,708},707:{706,707,708},708:{706,707,708}]
Inverted extended constraints = [706:{706,707,708},707:{706,707,708},708:{706,707,708}]
|Promoted| = 1
Promoted = [708]
Before = []
Marked = [706, 707, 708]
After = []
Arbitrary = []
Predicates = [706]
Case 101. compilers/Jikes.java (lines 111 – 126)

```java
if (destDir != null) {
    cmd.createArgument().setValue("-d");
    cmd.createArgument().setFile(destDir); }
cmd.createArgument().setValue("-classpath");
cmd.createArgument().setPath(classpath);
if (encoding != null) {
    cmd.createArgument().setValue("-encoding");
    cmd.createArgument().setValue(encoding); }
if (debug) {
    cmd.createArgument().setValue("-g"); }
if (optimize) {
    cmd.createArgument().setValue("-O"); }
if (verbose) {
    cmd.createArgument().setValue("-verbose"); }
if (depend) {
    cmd.createArgument().setValue("-depend"); }
```

// the hammock contains lines 116-126

|Nodes| = 11
Nodes = [116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126]
Data-dependence successors = [117:{118},118:{120},120:{122},122:{124},124:{126}]
Control-dependence predecessors =
[117:{116},118:{116},120:{119},122:{121},124:{123},126:{125}]
|M| = 9
M = [116, 117, 118, 119, 120, 123, 124, 125, 126]
|PDG data-dependence edges| = 5
|PDG control-dependence edges| = 6
|Extended constraints| = 15
Extended constraints =
[117:{118,120,122,124,126},118:{120,122,124,126},120:{122,124,126},122:{124,126},124 :{126}]
Inverted extended constraints =
[118:{117},120:{117,118},122:{117,118,120},124:{117,118,120,122},126:{117,118,120,12 2,124}]
|Promoted| = 2
Promoted = [121, 122]
Before = []
Marked = [116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126]
After = []
Arbitrary = []
Predicates = [116, 119, 121, 123, 125]
Case 105. CodeStream.java (lines 355 – 364)

```java
final public void astore_1() {
    countLabels = 0;
    stackDepth--;  
    if (maxLocals <= 1) {
        maxLocals = 2;  
    try {
        position++;  
        bCodeStream[classFileOffset++] = OPC astore_1;  
    } catch (IndexOutOfBoundsException e) {
        resizeByteArray(OPC astore_1);  
    }
}
```

// Node 361 is considered separately from the try

| Nodes | 5
Nodes = [356, 357, 358, 359, 361]
Data-dependence successors = [358:{359}]
Control-dependence predecessors = [359:{358}]
| M | 2
M = [356, 361]
| PDG data-dependence edges| = 1
| PDG control-dependence edges| = 1
| Extended constraints| = 2
Extended constraints = [358:{359},359:{359}]
Inverted extended constraints = [359:{358,359}]
| Promoted| = 0
Promoted = []
Before = []
Marked = [356, 361]
After = [357, 358, 359]
Arbitrary = [357, 359]
Predicates = [358]