Reasoning about AST Changes

Reinout Stevens
resteven@vub.ac.be
@ReinoutStevens

Coen De Roover
cderoove@vub.ac.be
@oniroi

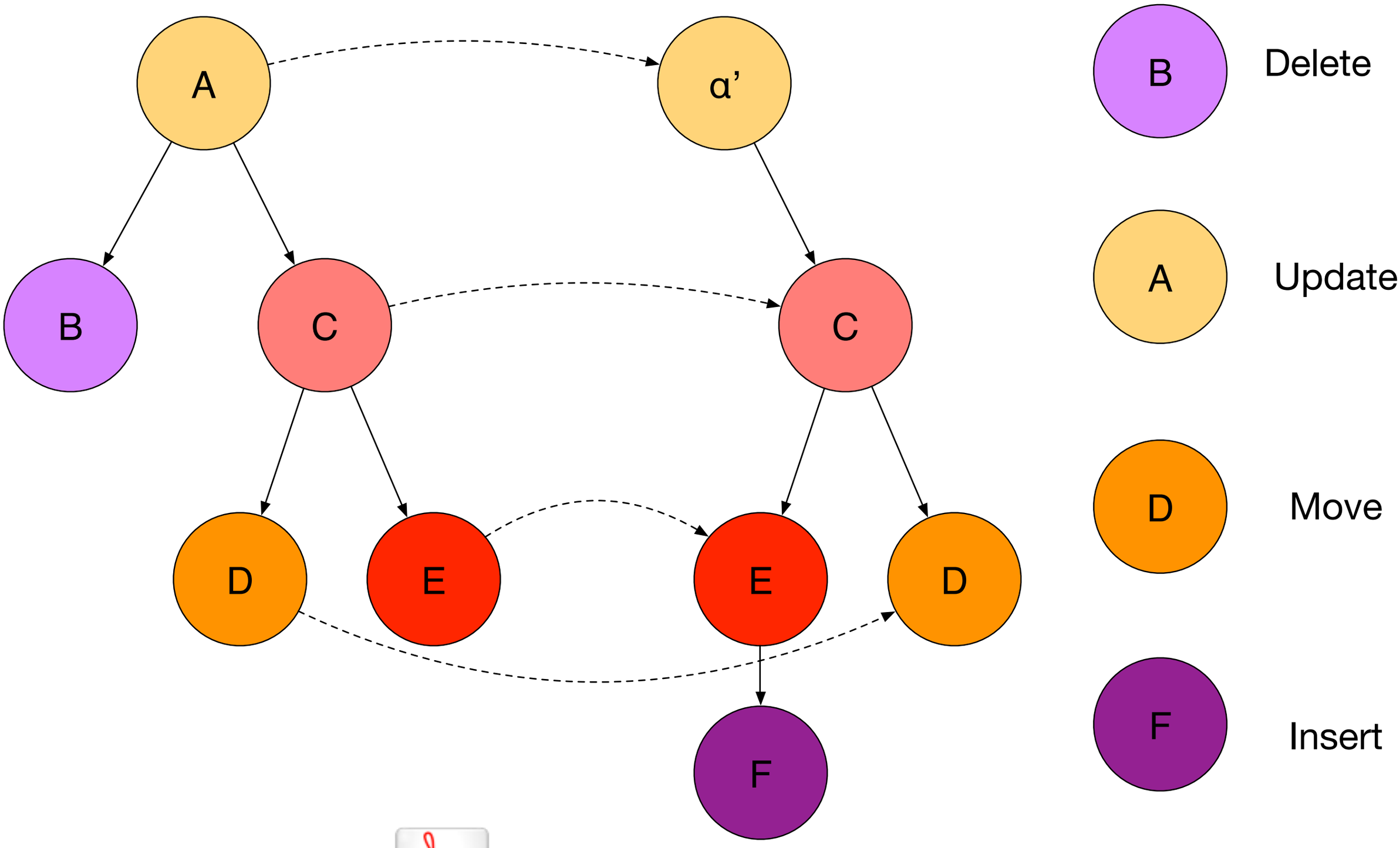
Context

```
00 public void setIncludes(String[] includes) {
01   if (includes == null) {
02     this.includes = null;
03   } else {
04     this.includes = new String[includes.length];
05     for (int i = 0; i < includes.length; i++) {
06       String pattern;
07       pattern = includes[i].replace('/', File.separatorChar).replace(
08               '\\', File.separatorChar);
09       if (pattern.endsWith(File.separator)) {
10         pattern += "**";
11       }
12       this.includes[i] = pattern;
13     }
14   }
15 }
16
17 public void setExcludes(String[] excludes) {
18   if (excludes == null) {
19     this.excludes = null;
20   } else {
21     this.excludes = new String[excludes.length];
22     for (int i = 0; i < excludes.length; i++) {
23       String pattern;
24       pattern = excludes[i].replace('/', File.separatorChar).replace(
25               '\\', File.separatorChar);
26       if (pattern.endsWith(File.separator)) {
27         pattern += "**";
28       }
29       this.excludes[i] = pattern;
30     }
31   }
32 }
```

```
00 public void setIncludes(String[] includes) {
01   if (includes == null) {
02     this.includes = null;
03   } else {
04     this.includes = new String[includes.length];
05     for (int i = 0; i < includes.length; i++) {
06       this.includes[i] = normalizePattern(includes[i]);
07     }
08   }
09 }
10
11 public void setExcludes(String[] excludes) {
12   if (excludes == null) {
13     this.excludes = null;
14   } else {
15     this.excludes = new String[excludes.length];
16     for (int i = 0; i < excludes.length; i++) {
17       this.excludes[i] = normalizePattern(excludes[i]);
18     }
19   }
20 }
21
22 private static String normalizePattern(String p) {
23   String pattern = p.replace('/', File.separatorChar)
24                     .replace('\\', File.separatorChar);
25   if (pattern.endsWith(File.separator)) {
26     pattern += "**";
27   }
28   return pattern;
29 }
```

The cloned code is extracted in a new method, and each clone instance is replaced by a method invocation to the newly introduced method.
Additionally, we want to detect whether and what additional operations are performed by the developer.

Changes



Beat Fluri and Harald C. Gall. Classifying Change Types for Qualifying Change Couplings. In Proceedings of the 14th International Conference on Program Comprehension, 2006.

Changes

```
00 public void setIncludes(String[] includes) {
01   if (includes == null) {
02     this.includes = null;
03   } else {
04     this.includes = new String[includes.length];
05     for (int i = 0; i < includes.length; i++) {
06       String pattern;
07       pattern = includes[i].replace('/', File.separatorChar).replace(
08               '\\', File.separatorChar);
09       if (pattern.endsWith(File.separator)) {
10         pattern += "**";
11       }
12       this.includes[i] = pattern;
13     }
14   }
15 }
16
17 public void setExcludes(String[] excludes) {
18   if (excludes == null) {
19     this.excludes = null;
20   } else {
21     this.excludes = new String[excludes.length];
22     for (int i = 0; i < excludes.length; i++) {
23       String pattern;
24       pattern = excludes[i].replace('/', File.separatorChar).replace(
25               '\\', File.separatorChar);
26       if (pattern.endsWith(File.separator)) {
27         pattern += "**";
28       }
29       this.excludes[i] = pattern;
30     }
31   }
32 }
```

```
00 public void setIncludes(String[] includes) {
01   if (includes == null) {
02     this.includes = null;
03   } else {
04     this.includes = new String[includes.length];
05     for (int i = 0; i < includes.length; i++) {
06       this.includes[i] = normalizePattern(includes[i]);
07     }
08   }
09 }
10
11 public void setExcludes(String[] excludes) {
12   if (excludes == null) {
13     this.excludes = null;
14   } else {
15     this.excludes = new String[excludes.length];
16     for (int i = 0; i < excludes.length; i++) {
17       this.excludes[i] = normalizePattern(excludes[i]);
18     }
19   }
20 }
21
22 private static String normalizePattern(String p) {
23   String pattern = p.replace('/', File.separatorChar)
24                     .replace('\\', File.separatorChar);
25   if (pattern.endsWith(File.separator)) {
26     pattern += "**";
27   }
28   return pattern;
29 }
```

Insert Move Delete

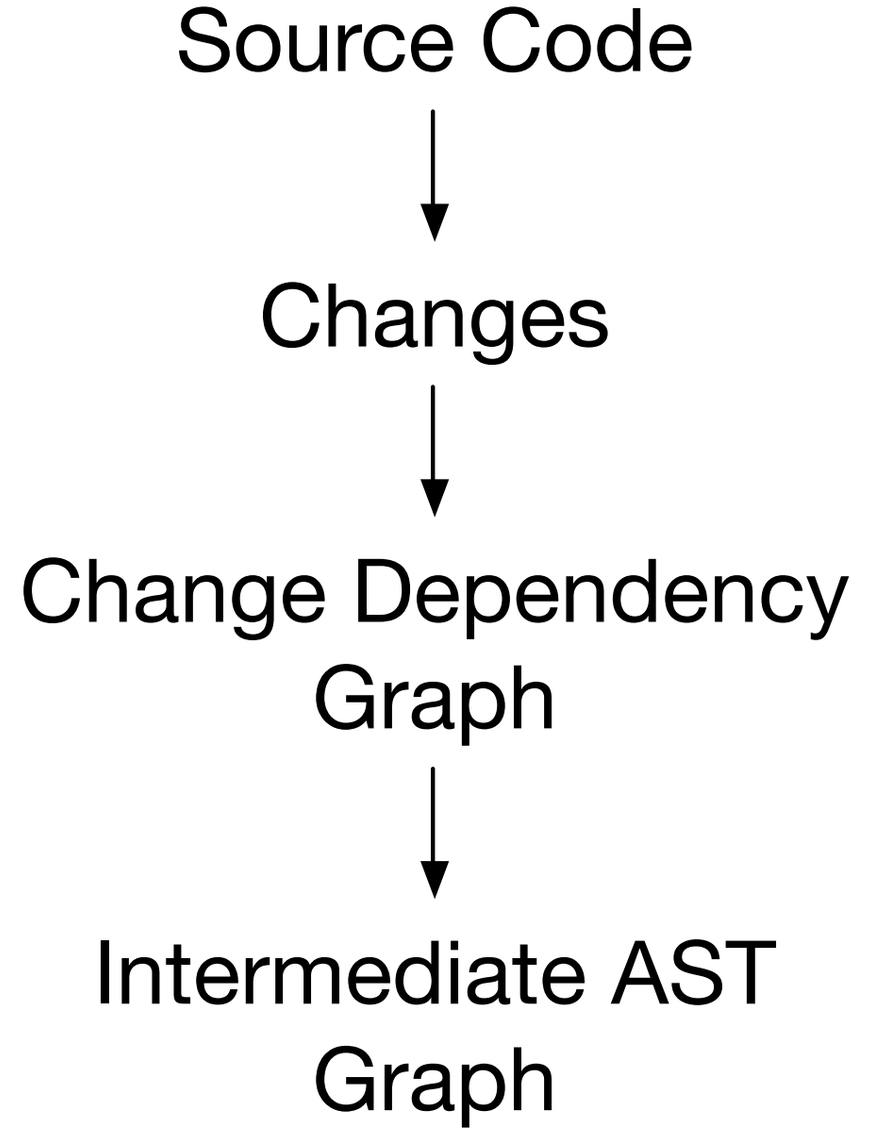
Problems

- There is no unique way to describe a code evolution query using change operations
- There are implicit dependencies between changes

Solution

We propose a novel way to query changes, in which a user navigates a graph of intermediate ASTs and defines source code characteristics that have to hold in these ASTs, hereby describing the effect of one or more changes in a change-agnostic manner.

Approach


Source Code
Changes
Change Dependency Graph
Intermediate AST Graph

Example Query

```
1 (query-changes iag ?end-state [?method ?name]
2   change->*
3   (in-current-ias [node ast]
4     (child+ ast ?method)
5     (ast :MethodDeclaration ?method)
6     (has :name ?method ?name)
7     (name|simple-string ?name "normalizePattern")))
```

Changes

Source AST

```
public class Example {
}
```

Target AST

```
public class Example {
   int x = 0;
   int y = 1;
}
```

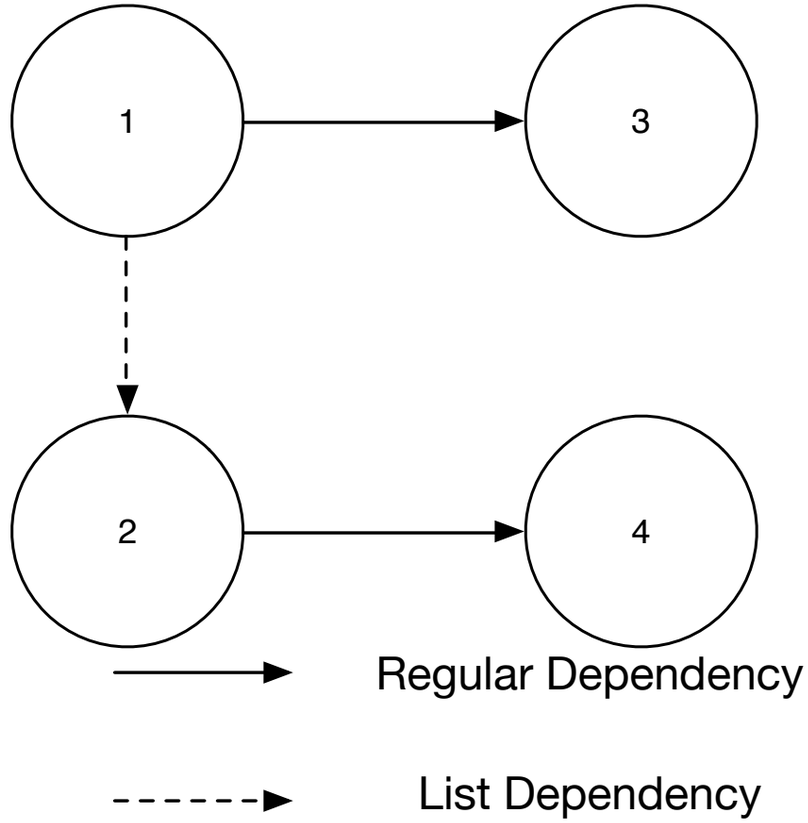
```
1.insert(int x, Example, Example, :BodyDeclarations, 0)
2.insert(int y, Example, Example, :BodyDeclarations, 1)
3.insert(0, nil, int x, :Initializer, nil)
4.insert(1, nil, int y, :Initializer, nil)
```

Dependency Graph

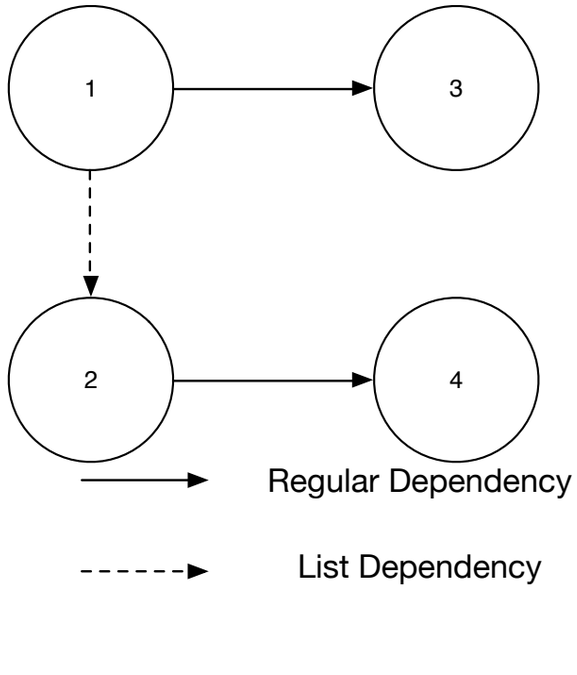
```
1.insert(int x, Example, Example, :BodyDeclarations, 0)
2.insert(int y, Example, Example, :BodyDeclarations, 1)
3.insert(0, nil, int x, :Initializer, nil)
4.insert(1, nil, int y, :Initializer, nil)
```

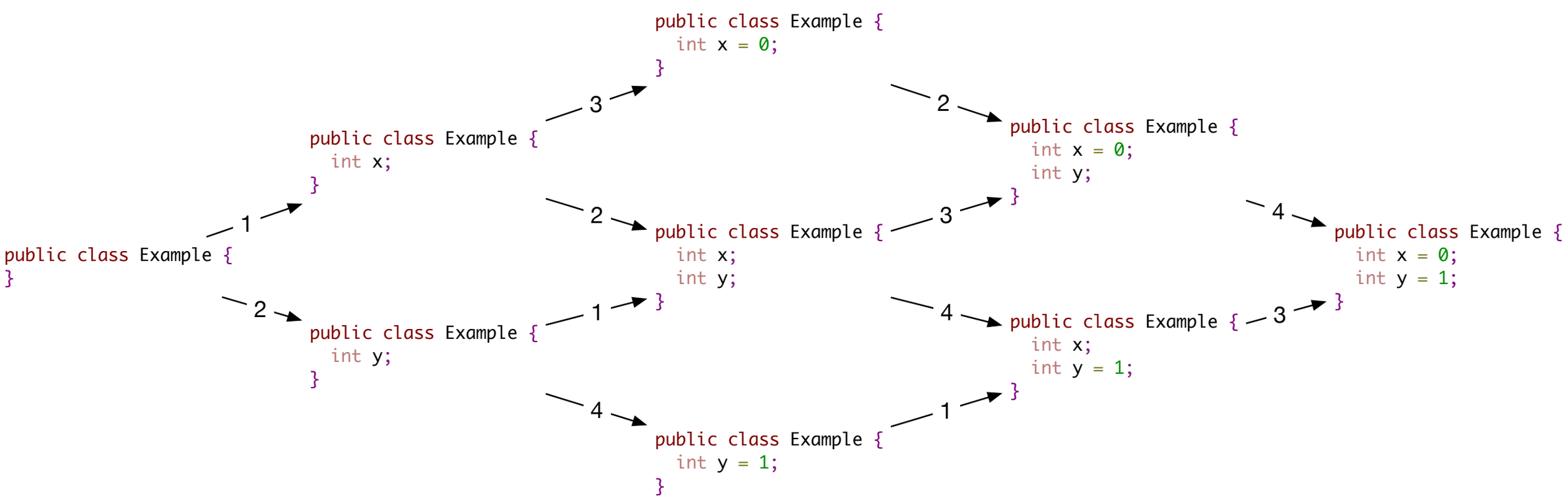
Target AST

```
public class Example {
   int x = 0;
   int y = 1;
}
```



Intermediate AST Graph





Query

```
(defn refactoring-query [iag nameA nameB extracted]
    (query-changes iag ?end
      [?cloneA ?cloneB ?extracted ?aInvoc ?bInvoc ?aCurr ?bCurr]
      (in-current-ias [curr ast]
        (child+ ast ?cloneA)
        (child+ ast ?cloneB)
        (method-string|named ?cloneA nameA)
        (method-string|named ?cloneB nameB))
      change->+
      (in-current-ias [curr ast]
        (ast :MethodDeclaration ?extracted)
        (ast-ast|same ?extracted extracted)
        (ias-node-node|tracked curr ?cloneA ?aCurr)
        (ias-node-node|tracked curr ?cloneB ?bCurr)
        (child+ ?aCurr ?aInvoc)
        (child+ ?bCurr ?bInvoc)
        (methodinvoc-method|invokes ?aInvoc ?extracted)
        (methodinvoc-method|invokes ?bInvoc ?extracted))))
```

Conclusion

Source AST

```
public class Example {
}
```

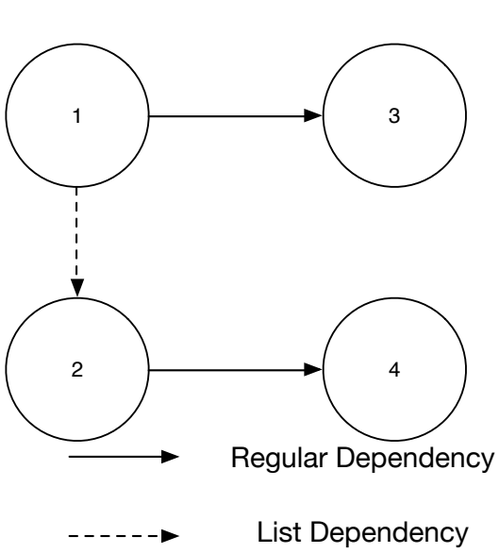
Target AST

```
public class Example {
  int x = 0;
  int y = 1;
}
```

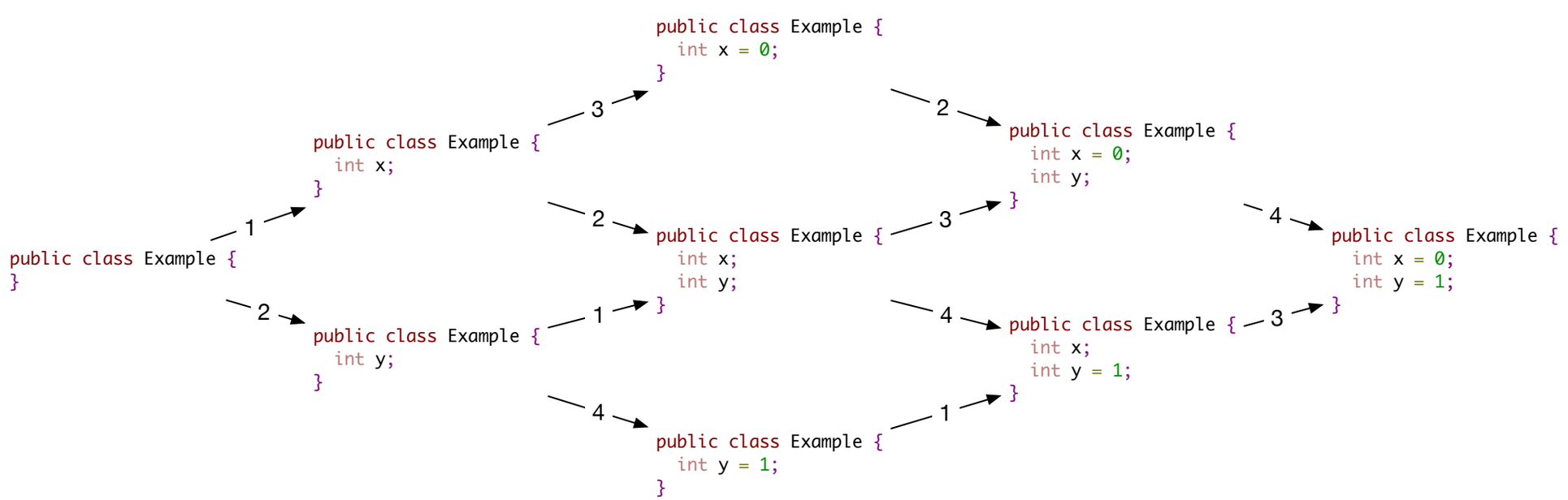
Distilled Change Sequence

```
1. insert(int x, Example, Example, nil, :BodyDeclarations, 0)
2. insert(int y, Example, Example, nil, :BodyDeclarations, 1)
3. insert(0, nil, int x, nil, :Initializer, nil)
4. insert(1, nil, int y, nil, :Initializer, nil)
```

Change Dependency Graph



Intermediate AST Graph



https://github.com/ReinoutStevens/ChangeNodes
https://github.com/ReinoutStevens/damp.qwalkeko