A Appendix

A.1 Add method invocation

The following section details the ADD METHOD INVOCATION transformation. Listing A.1 describes the subset of the Java language targeted, and what follows describes the transformation's behavior.

This section describes three things: (i) the transplantation point selection (i.e. What region of the code is transformed.), (ii) the transplant selection (i.e. what piece of code is inserted.), (iii) how the transformation is applied.

Transplantation point: First, this transformation target a statement inside the body of a method (and if the transplant is non void, the class defining the method).

Listing A.1 describes the grammar of the targeted class. The transplantation is any statement in such a class.

```
<class> ::= (<modifier>)*
'class' <identifier> (ref)*
'{' (<class_member>)* '}'

<class_member> ::= <attribute> |
<other> | <method>

<method> ::= (<modifier>)* (return_type)
<identifier> '('
(<type> <identifier>)* ')' '{'
(<statement>)* '}'

<statement> ::= <variable_declaration> |
<block> | <other>

<block> ::= (<block_header>) '{'
(<statement>)* '}'
```

$P = \{\text{Packages}\}$,
$C(p) = \{\text{Classes of } p \| p \in P\}$,
$M(c) = \{\text{methods in class } c\}$,
$S(m) = \{\text{statements in method } m\text{'s body}\}$,
$LV(s) = \{\text{Local variables up to } s\}$,
$Pa(m) = \{\text{Parameter of method } m\}$,
$A(c) = \{\text{Attributes of class } c\}$,
$As(c) = \{a \in A(c) \| static(a)\}$,
Let $p \in P, c \in C(p), m \in M(c), s \in S(m)$
$V(c, m, s) = LV(s) \cup Pa(m) \cup A(c) \cup \{this\}$, a set of accessible variable from s
$Vs(c, m, s) = LV(s) \cup Pa(m) \cup As(c)$, a set of accessible variable from s
$\mathcal{M}(p, c, m, s) = \{m' \in c\} \cup \{m' \in c' \| \forall c' \in p \wedge \neg private(m')\} \cup \{m' \in c' \| \forall p' \in P, \forall c' \in p' \wedge public(m')\}$
$\mathcal{M}_a(p, c, m, s) = \{m' \| static(m') \vee ClassOf(m') \in TypeOf(LV(c, m, s))\}$

Transplant:

Let $m' \in \mathcal{M}_a$

The transplant is an invocation of the method m' if it is void, and the affectation of an invocation of m' to a new field, called here "well".

Transformation:

The class targeted is transformed in the following fashion:

$$\text{Class}\,\frac{c \text{ follows } ...m...}{c \rightarrow ...Well; m...}$$

$$\text{Well}_{static \wedge \neg void}\,\frac{\text{static}(m) \quad \wedge \quad \text{TypeOf}(m') \neq void}{Well \rightarrow \text{'public static' TypeOf}(m') \text{ wellID}}$$

$$\text{Well}_{\neg static \wedge \neg void}\frac{\neg\ \text{static}(m)\ \wedge\ \text{TypeOf}(m') \neq void}{Well \rightarrow \text{'public' TypeOf}(m')\ \text{wellID}}$$

$$\text{Well}_{void}\frac{\text{TypeOf}(m') = void}{Well \rightarrow SKIP}$$

$$\text{Method}\frac{m \text{ follows } ...s...}{m \rightarrow ...\text{'try } \{\text{'}Well\ Ta\ Call;\text{'}\}\text{ catch (Exception ' eId ') }\{\}\text{'}s...}$$

$$\text{We}\frac{\text{TypeOf}(m') \neq void}{We \rightarrow wellID =}$$

$$\text{We}_{void}\frac{\text{TypeOf}(m') = void}{We \rightarrow SKIP}$$

$$\text{Target}_{static}\frac{\text{static}(m')}{Ta \rightarrow SKIP}$$

$$\text{Target}\frac{\neg\ \text{static}(m')}{Ta \rightarrow targetID.}$$

$$\text{Call}\frac{}{Call \rightarrow QN(m')(params)}$$

For the concrete implementation, see the class `fr.inria.diversify.transformation.query.AddMethodInvocationQuery`[1].

A.2 SwapSubtype

The following section details the behavior of the SWAP SUBTYPE transformation, the subset of java targeted A.2, and how it modified it.

```
<affectation> ::= <ls> '=' <rs>
<ls> ::=  (<interface> ('<' <type> '>')?)? <identifier>
<rs> ::= 'new' <concrete_class_constructor> ('<' <type> '>')? '('
    <param_list> ')'
<param_list> ::= <> | <param> | <param> ',' <param_list>
\label{lst:col-assign}
```

$I = \{\text{Interfaces}\}$
$T = \{\text{Types}\}$
$C(t) = \{\text{Constructor of } t\}$
$T(i) = \{t \in T \| t \text{ implements } i\}$
Let $i \in I, t_1, t_2 \in T(i)^2$ such as $t_1 \neq t_2$

$$\text{Affectation}\frac{\text{Aff}(I, id, t_1, params) \wedge \exists c \in C(t_2)}{\text{Aff}(I, id, t_1, params) \rightarrow Iid =' new' t_2'('params')'}$$

The following sections list the different interfaces targeted by our implementation of the transformation, and for each interface the different classes implementing these interfaces used interchangeably.

A.2.1 java.util.SortedSet

- java.util.concurrent.ConcurrentSkipListSet
- java.util.TreeSet

A.2.2 java.util.concurrent.BlockingDeque

- java.util.concurrent.LinkedBlockingDeque

[1] `https://github.com/DIVERSIFY-project/sosiefier/releases/tag/2.0.0`

A.2.3 java.util.Collection

- java.util.concurrent.LinkedTransferQueue
- java.util.concurrent.SynchronousQueue
- java.util.PriorityQueue
- java.util.concurrent.CopyOnWriteArraySet
- java.util.concurrent.LinkedBlockingQueue
- java.util.TreeSet
- java.util.concurrent.ConcurrentLinkedDeque
- java.util.Stack
- java.util.concurrent.PriorityBlockingQueue
- java.util.ArrayList
- java.util.HashSet
- java.util.concurrent.ArrayBlockingQueue
- java.util.Vector
- java.util.concurrent.ConcurrentSkipListSet
- java.util.concurrent.LinkedBlockingDeque
- java.util.concurrent.DelayQueue
- java.util.ArrayDeque
- java.util.LinkedList
- java.util.LinkedHashSet
- java.util.concurrent.ConcurrentLinkedQueue
- java.util.concurrent.CopyOnWriteArrayList

A.2.4 java.util.concurrent.ConcurrentNavigableMap

- java.util.concurrent.ConcurrentSkipListMap

A.2.5 java.util.Set

- java.util.HashSet
- gnu.trove.set.hash.THashSet
- java.util.concurrent.ConcurrentSkipListSet
- org.apache.commons.collections4.set.ListOrderedSet
- java.util.concurrent.CopyOnWriteArraySet
- java.util.TreeSet
- java.util.LinkedHashSet
- gnu.trove.set.hash.TCustomHashSet

A.2.6 java.util.concurrent.BlockingQueue

- java.util.concurrent.ArrayBlockingQueue
- java.util.concurrent.LinkedTransferQueue
- java.util.concurrent.SynchronousQueue
- java.util.concurrent.LinkedBlockingDeque
- java.util.concurrent.DelayQueue
- java.util.concurrent.LinkedBlockingQueue
- java.util.concurrent.PriorityBlockingQueue

A.2.7 java.util.NavigableSet

- java.util.concurrent.ConcurrentSkipListSet
- java.util.TreeSet

A.2.8 java.util.Deque

- java.util.concurrent.LinkedBlockingDeque
- java.util.ArrayDeque
- java.util.LinkedList
- java.util.concurrent.ConcurrentLinkedDeque

A.2.9 java.util.concurrent.TransferQueue

- java.util.concurrent.LinkedTransferQueue

A.2.10 java.util.NavigableMap

- java.util.concurrent.ConcurrentSkipListMap
- java.util.TreeMap

A.2.11 java.util.concurrent.ConcurrentMap

- java.util.concurrent.ConcurrentSkipListMap
- java.util.concurrent.ConcurrentHashMap

A.2.12 java.util.List

- org.apache.commons.collections4.list.TreeList
- java.util.Vector
- org.apache.commons.collections4.list.NodeCachingLinkedList
- org.apache.commons.collections4.list.CursorableLinkedList
- java.util.LinkedList
- org.apache.commons.collections4.list.GrowthList
- java.util.Stack
- java.util.ArrayList
- java.util.concurrent.CopyOnWriteArrayList
- org.apache.commons.collections4.ArrayStack

A.2.13 java.util.Map

- org.apache.commons.collections4.map.SingletonMap
- org.apache.commons.collections4.map.Flat3Map
- org.apache.commons.collections4.map.LinkedMap
- java.util.concurrent.ConcurrentHashMap
- org.apache.commons.collections4.map.LRUMap
- org.apache.commons.collections4.map.ListOrderedMap
- java.util.HashMap
- org.apache.commons.collections4.map.HashedMap
- org.apache.commons.collections4.map.ReferenceMap
- org.apache.commons.collections4.map.CaseInsensitiveMap
- gnu.trove.map.hash.TCustomHashMap
- java.util.LinkedHashMap
- org.apache.commons.collections4.map.PassiveExpiringMap
- java.util.concurrent.ConcurrentSkipListMap
- org.apache.commons.collections4.map.StaticBucketMap
- java.util.TreeMap
- gnu.trove.map.hash.THashMap
- java.util.Hashtable
- java.util.WeakHashMap
- org.apache.commons.collections4.map.ReferenceIdentityMap

A.2.14 java.util.Iterable

- java.util.concurrent.LinkedTransferQueue
- java.util.concurrent.SynchronousQueue
- java.util.PriorityQueue
- java.util.concurrent.CopyOnWriteArraySet
- java.util.concurrent.LinkedBlockingQueue
- java.util.TreeSet
- java.util.concurrent.ConcurrentLinkedDeque
- java.util.Stack
- java.util.concurrent.PriorityBlockingQueue
- java.util.ArrayList
- java.util.HashSet
- java.util.concurrent.ArrayBlockingQueue
- java.util.Vector
- java.util.concurrent.ConcurrentSkipListSet
- java.util.concurrent.LinkedBlockingDeque
- java.util.concurrent.DelayQueue
- java.util.ArrayDeque
- java.util.LinkedList
- java.util.LinkedHashSet
- java.util.concurrent.ConcurrentLinkedQueue
- java.util.concurrent.CopyOnWriteArrayList

A.2.15 java.util.Queue

- java.util.concurrent.LinkedTransferQueue
- java.util.concurrent.SynchronousQueue
- java.util.PriorityQueue
- java.util.concurrent.LinkedBlockingQueue
- java.util.concurrent.ConcurrentLinkedDeque
- java.util.concurrent.PriorityBlockingQueue
- java.util.concurrent.ArrayBlockingQueue
- org.apache.commons.collections4.queue.CircularFifoQueue
- java.util.concurrent.LinkedBlockingDeque
- java.util.concurrent.DelayQueue
- java.util.ArrayDeque
- java.util.LinkedList
- java.util.concurrent.ConcurrentLinkedQueue

A.2.16 java.util.SortedMap

- java.util.concurrent.ConcurrentSkipListMap
- java.util.TreeMap

For the concrete implementation, see the class `fr.inria.diversify.transformation.query.SwapSubTypeQuery`[2].

[2] `https://github.com/DIVERSIFY-project/sosiefier/releases/tag/2.0.0`

A.3 Loopflip

Transplantation point: LOOP FLIP targets loops describes by Listing A.3

```
<loop> ::= 'for(' <initialization> ';'
<condition> ';' <update> ')' '{'
(<statement>)* '}'

<initialization> ::= <identifier>
'=' <expression>

<condition> ::= <identifier>
<binary_operator> <expression>

<binary_operator> ::= '<' |
'<=' | '>' | '>='

<update> ::= <identifier> '='
<identifier> <operator> <expression>

<operator> ::= '+' | '-'
```

We extends update statements such as `i++` into `i = i + 1` and `i -= 2` into `i = i - 2`

Transplant: LOOP FLIP replaces the targeted `loop` with a new one constructed as following.

$\text{comp} \in \{<, >, \geq, \leq\}, \text{op} \in \{+, -\}$

$$\overline{a}, \forall a \in \{<, >, \geq, \leq\} \cup \{+, -\} \begin{cases} >, \geq \mapsto \leq \\ <, \leq \mapsto \geq \\ + \mapsto - \\ - \mapsto + \end{cases}$$

$$\text{For}_L \frac{\text{comp} \in \{\geq, \leq\} \ \wedge \ |i_{end} - i_0| \equiv 0 \pmod p}{(\text{For}_L(i = i_0 i \text{ comp } i_{end} i = i \text{ op } p) \rightarrow (\text{For}_L(i = i_{end} i \ \overline{\text{comp}} \ i_0 i = i \ \overline{\text{op}} \ p)}$$

$$\text{For}_L \frac{\text{comp} \notin \{\geq, \leq\} \ \vee \ \neg(|i_{end} - i_0| \equiv 0 \pmod p)}{(\text{For}_L(i = i_0 i \text{ comp } i_{end} i = i \text{ op } p) \rightarrow (\text{For}_L(i = i_{end} \ \overline{\text{op}} \ (|i_{end} - i_0| \pmod p)) i \ \overline{\text{comp}} \ i_0 i = i \ \overline{\text{op}} \ p)}$$

For the concrete implementation, see the class `fr.inria.diversify.transformation.query.LoopFlipQuery`[3].

[3] https://github.com/DIVERSIFY-project/sosiefier/releases/tag/2.0.0