MANCHESTER 1824
The University of Manchester

UNIVERSITY OF SUSSEX

diffblue
AI for Code

28th International Conference on Tools and Algorithms for the Construction and Analysis of Systems

Wit4Java

A Violation-Witness Validator for Java Verifiers

Tong Wu, Peter Schrammel and Lucas Cordeiro

Motivation

A Java verifier may produce false alarms.

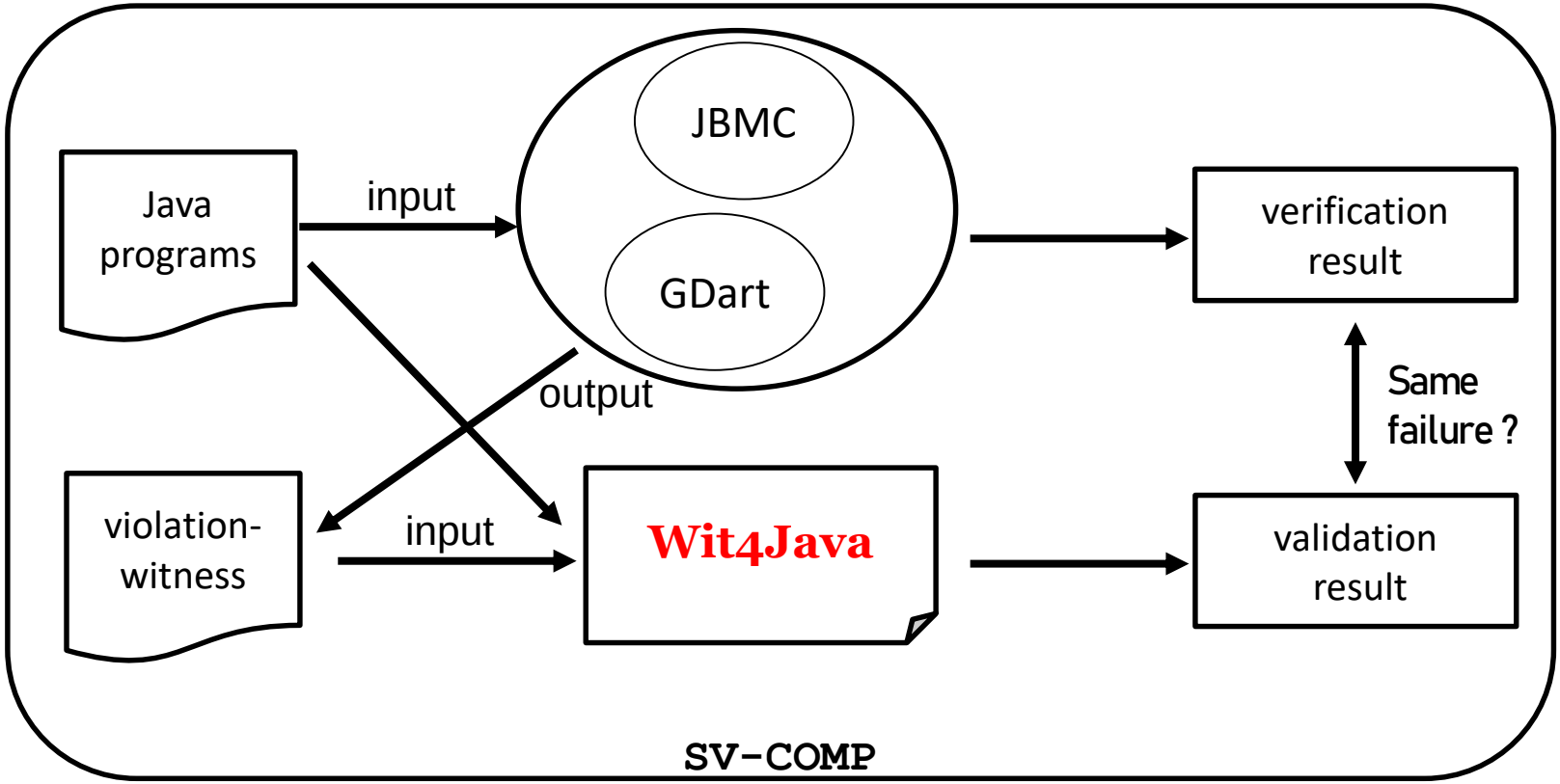
status	raw score	cpu (s)	mem (MB)	energy (J)
false	Min:Ma	Mi	Mi	Min
false	1	2.3	93	20
false	-16	2.2	84	21
false	1	2.3	86	21
false	1	350	670	5300
false	1	6.2	150	64
false	1	11	330	110

No validators for Java participated in SV-COMP.

Validators

Validator	Contact	Affiliation
Validator CPAchecker	Karlheinz Friedberger, Martin Spießl	LMU Munich, Germany
Validator Ultimate Automizer	Daniel Dietsch, Matthias Heizmann	Univesity of Freiburg, Germany
Validator CPA-witness2test	Matthias Dangl, Thomas Lemberger	LMU Munich, Germany
Validator FShell-witness2test	Michael Tautschnig	Queen Mary University of London, UK
Validator MetaVal	Martin Spießl	LMU Munich, Germany
Validator NITWIT	Philipp Berger	RWTH Aachen, Germany
Validator WitnessLint	Sven Umbricht	LMU Munich, Germany

Overview


Java programs
input
JBMC
GDart
verification result
Same failure ?
output
violation-witness
input
Wit4Java
validation result
SV-COMP

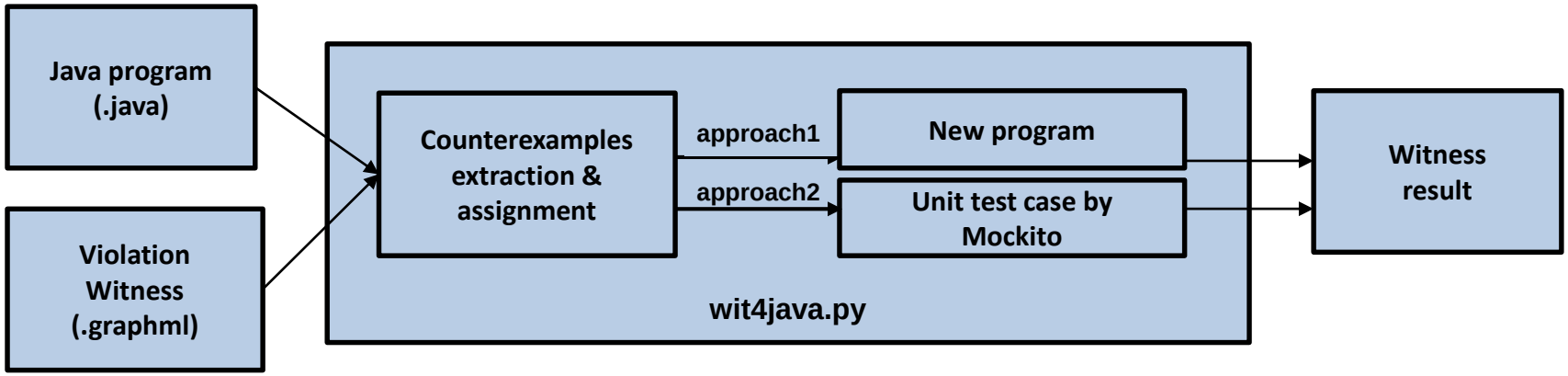
MANCHESTER 1824
The University of Manchester

Objectives

Implement a violation-witness validator for Java verifiers

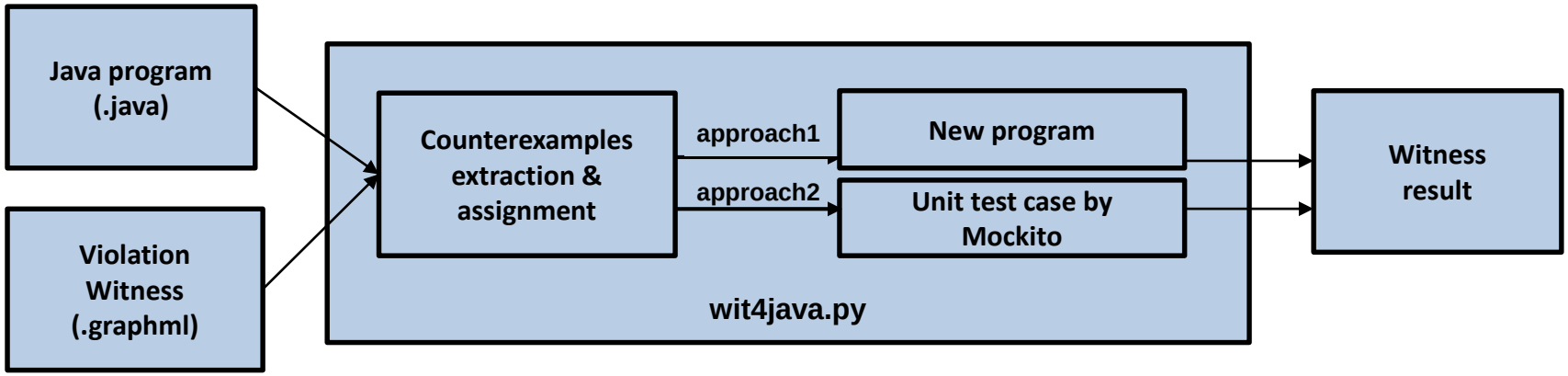
- Integrate the validation tool into BenchExec ecosystem to have precise resource limits and measurement
- Evaluate the performance of the witness validator in SV-COMP

Architecture



Architecture

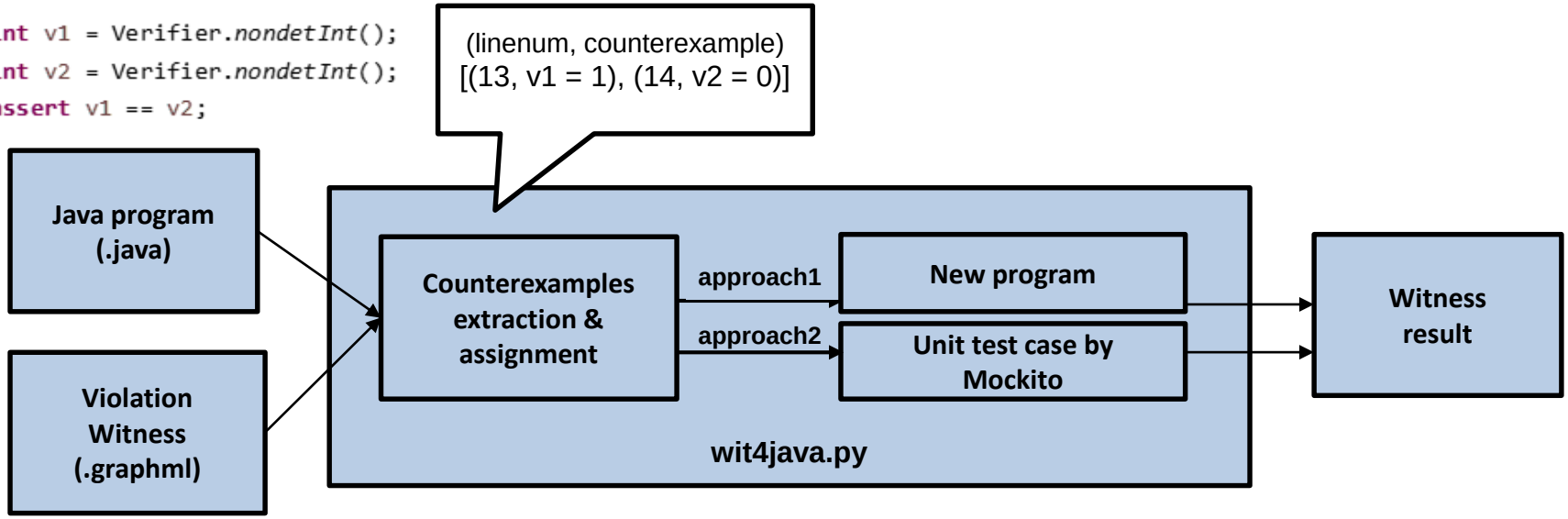
```
int v1 = Verifier.nondetInt();
int v2 = Verifier.nondetInt();
assert v1 == v2;
```




```
<edge source="203.167" target="207.186">
      <data key="originfile">Main.java</data>
      <data key="startline">13</data>
      <data key="assumption">v1 = 1;</data>
</edge>
 <edge source="207.186" target="252.201">
      <data key="originfile">Main.java</data>
      <data key="startline">14</data>
      <data key="assumption">v2 = 0;</data>
 </edge>
```

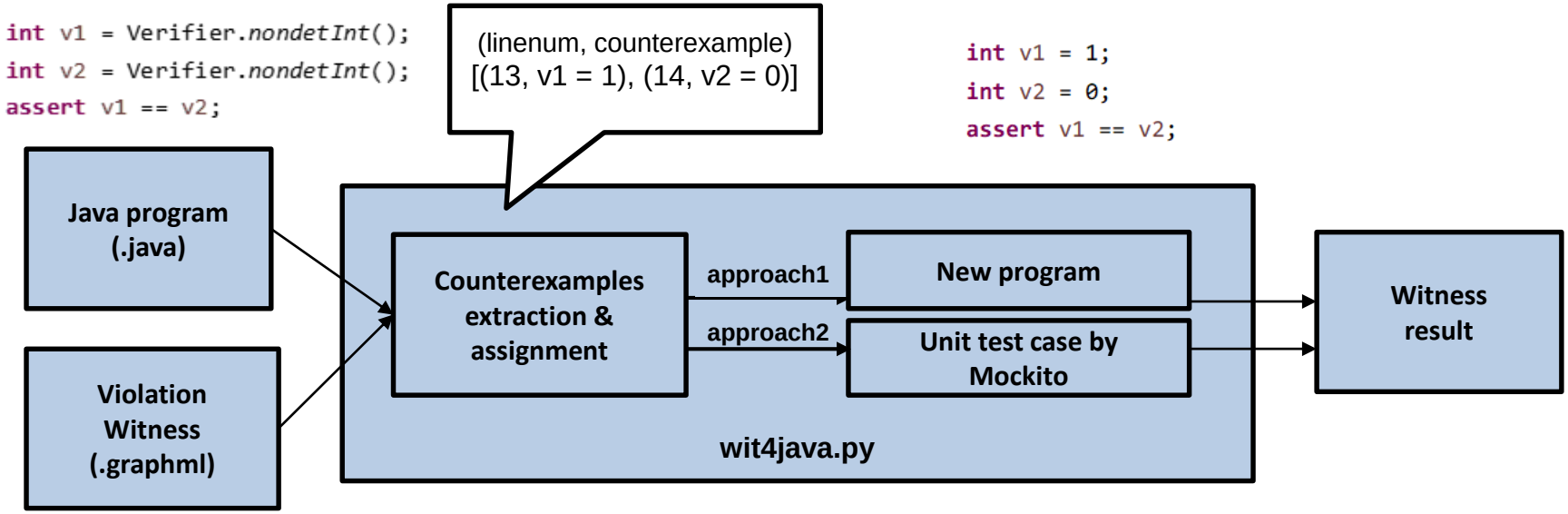
Architecture

```
int v1 = Verifier.nondetInt();
int v2 = Verifier.nondetInt();
assert v1 == v2;
```




```
<edge source="203.167" target="207.186">
      <data key="originfile">Main.java</data>
      <data key="startline">13</data>
      <data key="assumption">v1 = 1;</data>
</edge>
 <edge source="207.186" target="252.201">
      <data key="originfile">Main.java</data>
      <data key="startline">14</data>
      <data key="assumption">v2 = 0;</data>
 </edge>
```

Architecture



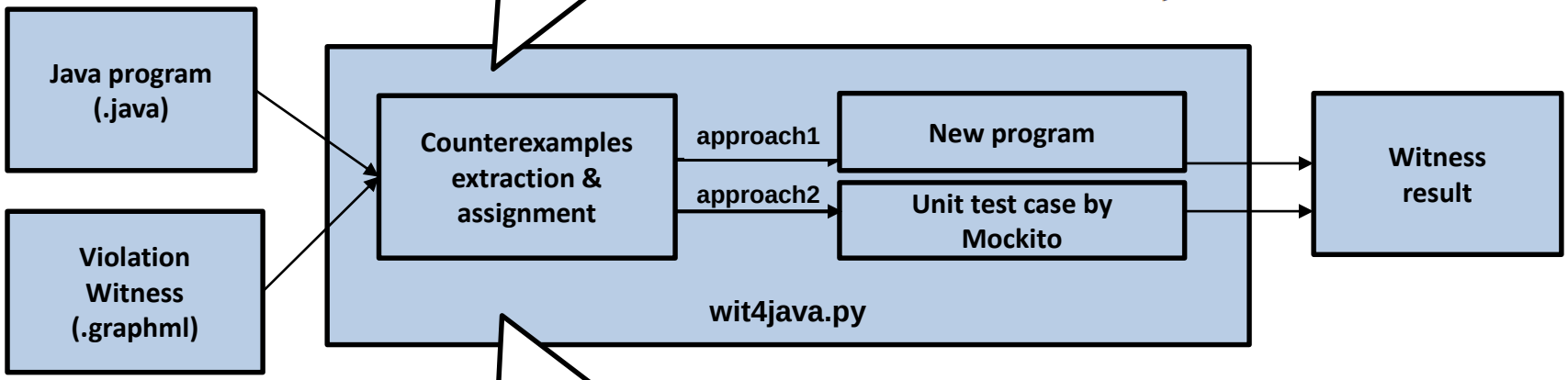

```
<edge source="203.167" target="207.186">
      <data key="originfile">Main.java</data>
      <data key="startline">13</data>
      <data key="assumption">v1 = 1;</data>
</edge>
 <edge source="207.186" target="252.201">
      <data key="originfile">Main.java</data>
      <data key="startline">14</data>
      <data key="assumption">v2 = 0;</data>
 </edge>
```

Architecture

```
int v1 = Verifier.nondetInt();
int v2 = Verifier.nondetInt();
assert v1 == v2;
```

(linenum, counterexample)
[(13, v1 = 1), (14, v2 = 0)]

```
int v1 = 1;
int v2 = 0;
assert v1 == v2;
```



List_type = [int, int]
List_value = [1, 0]

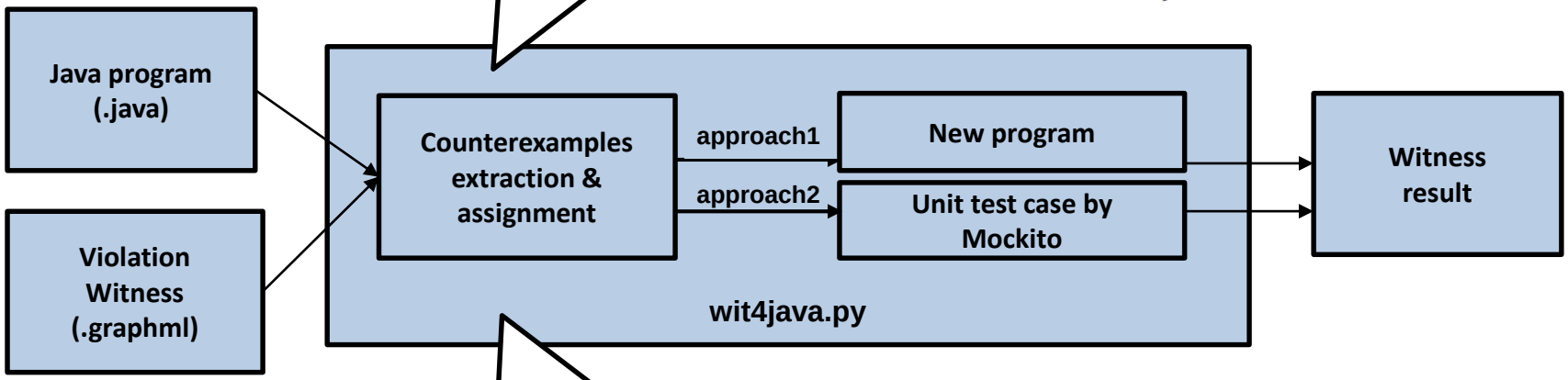
```
<edge source="203.167" target="207.186">
      <data key="originfile">Main.java</data>
      <data key="startline">13</data>
      <data key="assumption">v1 = 1;</data>
</edge>
 <edge source="207.186" target="252.201">
      <data key="originfile">Main.java</data>
      <data key="startline">14</data>
      <data key="assumption">v2 = 0;</data>
 </edge>
```

Architecture

```
int v1 = Verifier.nondetInt();
int v2 = Verifier.nondetInt();
assert v1 == v2;
```

(linenum, counterexample)
[(13, v1 = 1), (14, v2 = 0)]

```
int v1 = 1;
int v2 = 0;
assert v1 == v2;
```




```
<edge source="203.167" target="207.186">
      <data key="originfile">Main.java</data>
      <data key="startline">13</data>
      <data key="assumption">v1 = 1;</data>
</edge>
 <edge source="207.186" target="252.201">
      <data key="originfile">Main.java</data>
      <data key="startline">14</data>
      <data key="assumption">v2 = 0;</data>
 </edge>
```

List_type = [int, int]
List_value = [1, 0]

```
Mockito.when(Verifier.nondetInt()).
thenReturn(1).thenReturn(0);
```

MANCHESTER 1824
The University of Manchester

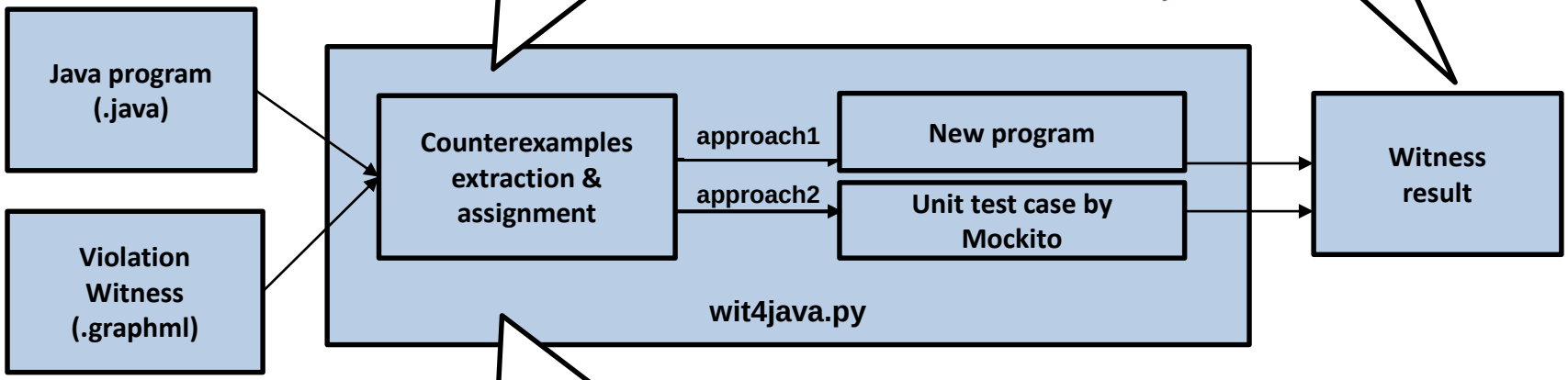
Architecture

```
wit4java version: 1.0
witness:  ../../results-verified/jbmc.2021-12-07_10-30-2
benchmark:  ['../../sv-benchmarks/java/common/org/sosy_l
Exception in thread "main" java.lang.AssertionError
        at Main.main(Main.java:15)
```

```
int v1 = Verifier.nondetInt();
int v2 = Verifier.nondetInt();
assert v1 == v2;
```

(linenum, counterexample)
[(13, v1 = 1), (14, v2 = 0)]

```
int v1 = 1;
int v2 = 0;
assert v1 == v2;
```




```
<edge source="203.167" target="207.186">
      <data key="originfile">Main.java</data>
      <data key="startline">13</data>
      <data key="assumption">v1 = 1;</data>
</edge>
 <edge source="207.186" target="252.201">
      <data key="originfile">Main.java</data>
      <data key="startline">14</data>
      <data key="assumption">v2 = 0;</data>
 </edge>
```

List_type = [int, int]
List_value = [1, 0]

```
Mockito.when(Verifier.nondetInt()).
thenReturn(1).thenReturn(0);
```

MANCHESTER 1824
The University of Manchester

Approach 1

```
import org.sosy_lab.sv_benchmarks.Verifier;

class Main {
  public static void main(String[] args) {
    int v1 = Verifier.nondetInt() 1;
    int v2 = Verifier.nondetInt() 0;
    assert v1 == v2;
  }
}
```

Replace with counterexample

MANCHESTER 1824
The University of Manchester

Approach 2

Tasty mocking framework for unit tests in Java

mockito

```java
public class test {
    public static void main(String[] args) {
        Mockito.mockStatic(Verifier.class);
        String[] types = {"int","int"};
        String[] assumptions = {"1","0"};

        int n = types.length;
        OngoingStubbing <Integer> stubbing_int = Mockito.when(Verifier.nondetInt());
        for (int i = 0; i < n; i++) {
            if ("int".equals(types[i])) {
                stubbing_int = stubbing_int.thenReturn(Integer.parseInt(assumptions[i]));
            }
        }
        try {
            Main.main(new String[0]);
            System.out.println("OK ");
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

Mockito test case

Experimental evaluation

- **Goals**
  - To have a good performance on the benchmarks and participate in the SV-COMP as one of the first violation-witness validators for Java verifiers
- **Benchmarks**
  - All benchmarks are based from the SV-COMP 2022
  - https://gitlab.com/sosy-lab/benchmarking/sv-benchmarks/-/tree/main/java
  - The benchmarks contain 344 Java programs and their corresponding 302 violation-witnesses (Generated by GDart).

Experimental evaluation

- **Setup**
  - **Command:**
  - /wit4java.py -witness <path-to-sv-witnesses>/witness.graphml <path-to-sv-benchmarks>/java/jbmc-regression/return2
  - **Benchmark setup:**
  - The benchexec tool info module - **wit4java.py**.
  - https://github.com/sosy-lab/benchexec/blob/main/benchexec/tools/wit4java.py
  - The benchmark definition file - **wit4java-validate-violation-witnesses.xml**.
  - https://gitlab.com/sosy-lab/sv-comp/bench-defs/-/blob/main/benchmark-defs/wit4java-validate-violation-witnesses.xml

Benchmark results

Strengths and weaknesses

Strengths

It can validate benchmarks with multiple variables of 8 basic datatypes:

```
-byte        Verifier.nondetByte()
-short       Verifier.nondetShort()
-int         Verifier.nondetInt()
-long        Verifier.nondetLong()
-float       Verifier.nondetFloat()
-double      Verifier.nondetDouble()
-char        Verifier.nondetChar()
-boolean     Verifier.nondetBoolean()
```

The tool is sound and produces no false results based on SV-COMP:

```
Wit4Java(Mockito) 56% correct, 44% unknown, 0% wrong
GWIT              50% correct, 49% unknown, 1% wrong
Wit4Java 1.0      46% correct, 54% unknown, 0% wrong
```

Strengths and weaknesses

Weaknesses

Validation for strings is not supported:

```
String s1 = new String(Verifier.nondetString());
```

**Unable to test string counterexamples produced by JBMC.*
**JBMC should be improved to better support string manipulation.*

Rely on concrete counterexamples of nondeterministic variables:

```
try {
    Object x = new Integer(0);
    String y = (String) x;
  } catch (ClassCastException exc) {
    assert false;
  }
```

Not verifiable

MANCHESTER 1824
The University of Manchester

Thank you!