CallMeMaybe: Using NLP to Automatically Generate Unit Test Cases Respecting Temporal Constraints

Arianna Blasi♦ · Alessandra Gorla♥ · Michael D. Ernst♠ · Mauro Pezzè♣♣
Automatic Test Case Generation

Input

Output
Automatic Test Case Generation

- Randomly select method calls + inputs
- Assess correctness of the outcome
Automatic Test Case Generation

Where can the generator gather knowledge about the semantics of the SUT?

How can a user check the output of thousands of automatically generated test cases?
public IteratorEnumeration() {...}

@Test
public void test001(){
    IteratorEnumeration<String> strItEn0 =
        new IteratorEnumeration<String>();

    String str1 = strItEn0.nextElement();
}

Randooop
Automatic unit test generation for Java
public IteratorEnumeration() {...}

@Test
public void test001(){
    IteratorEnumeration<String> strItEn0 =
        new IteratorEnumeration<String>();

    String str1 = strItEn0.nextElement();
}

NullPointerException
public IteratorEnumeration() {...}

@Test
public void test001(){
    IteratorEnumeration<String> strItEn0 =
            new IteratorEnumeration<String>();

    String str1 = strItEn0.nextElement();
}

NullPointerException
Your program has a bug!
public IteratorEnumeration() {...}

@Test
public void test001(){
    IteratorEnumeration<String> strItEn0 =
        new IteratorEnumeration<String>();

    String str1 = strItEn0.nextElement();
}

NullPointerException
/**
 * Constructs a new IteratorEnumeration that will not function until setIterator(Iterator) is invoked.
 */

public IteratorEnumeration() {...}

@Test
public void test001(){
    IteratorEnumeration<String> strItEn0 = new IteratorEnumeration<String>();
    String str1 = strItEn0.nextElement();
}

NullPointerException
Your program has a bug!
Alright: expected.
Improving Test Case Generators

```json
[
  {
    "operationSignature": "org.apache.commons.collections4.iterators.IteratorEnumeration",
    "isConstructor": true,
    "mustPrecede": "setIterator(java.util.Iterator<? extends E>)",
    "mustFollow": "",
  }
]
```
@Test
public void test001(){
    IteratorEnumeration<String> strItEn0 =
            new IteratorEnumeration<String>();

    String str1 = strItEn0.nextElement();
}

Bug revealing
Non-bug revealing

Improving Test Case Generators
CallMeMaybe
Translating natural language method ordering information into Java code
/**
 * This method must be invoked before the thread is started.
 */

public final void setDaemon(boolean on)
“This method must be invoked before the thread is started”

1. Subject relations
2. Adverbial relations
"This method must be invoked before the thread is started"
"This method must be invoked before the thread is started"
"This method must be **invoked before** the thread is **started**"

**Adverb**: before

<table>
<thead>
<tr>
<th>Governor</th>
<th>Adverb</th>
<th>Dependent</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;invoked&quot;</td>
<td>&quot;before&quot;</td>
<td>&quot;started&quot;</td>
</tr>
</tbody>
</table>

**subject + verb**: (this method, be invoked)

**subject + verb**: (thread, is started)
"This method must be **invoked** **before** the thread is **started**"

<table>
<thead>
<tr>
<th>Governor</th>
<th>Adverb</th>
<th>Dependent</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;invoked&quot;</td>
<td>&quot;before&quot;</td>
<td>&quot;started&quot;</td>
</tr>
</tbody>
</table>

**adverb (before):** (invoked, started)

**subject + verb:** (this method, be invoked)

**subject + verb:** (thread, is started)
“This method must be invoked **before** the thread is started”

<table>
<thead>
<tr>
<th>Governor</th>
<th>Adverb</th>
<th>Dependent</th>
</tr>
</thead>
<tbody>
<tr>
<td>“invoked”</td>
<td>“before”</td>
<td>“started”</td>
</tr>
</tbody>
</table>

**adverb (before):** (invoked, started)

**subject + verb:** (this method, be invoked)

**subject + verb:** (thread, is started)
“This method must be invoked before the thread is started”

adverb (before): (invoked, started)

subject + verb: (this method, be invoked)

subject + verb: (thread, is started)

(this method, be invoked) BEFORE (thread, is started)
(this method, be invoked) **BEFORE** (thread, is started)
(this method, be invoked) **BEFORE** (thread, is started)

Translator

Temporal Specifications
(this method, be invoked) **BEFORE** (thread, is started)

**Translator**

- **Subject Matcher**
- **Predicate Matcher**
- **Direction Chief**

**Temporal Specifications**
Efstathiou et al., *Word embeddings for the software engineering domain*, MSR 2018
(this method, be invoked)

~ the documented method should be invoked

receiverObjectID.setDaemon(args[0])
Subject Matcher → Predicate Matcher → Direction Chief

\[(this \text{ method}, \text{ be invoked})\]

\sim \text{ the documented method should be invoked}

\text{receiverObjectID.setDaemon(args[0])}
(thread, is started)

**Code Candidates**

- Formal Parameters
- Class Name
- Methods
- Fields

<table>
<thead>
<tr>
<th>Candidate</th>
<th>Edit Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>on</td>
<td>8</td>
</tr>
<tr>
<td>Thread</td>
<td>0</td>
</tr>
<tr>
<td>start</td>
<td>8</td>
</tr>
<tr>
<td>sleep</td>
<td>8</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
(thread, is started)

**Code Candidates**
- Formal Parameters
- Class Name
- Methods
- Fields

<table>
<thead>
<tr>
<th>Candidate</th>
<th>Edit Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>on</td>
<td>8</td>
</tr>
<tr>
<td>Thread</td>
<td>0</td>
</tr>
<tr>
<td>start</td>
<td>8</td>
</tr>
<tr>
<td>sleep</td>
<td>8</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Subject Matcher -> Predicate Matcher -> Direction Chief

(thread, is **started**)
Subject Matcher \rightarrow \text{Predicate Matcher} \rightarrow \text{Direction Chief}

(thread, \textbf{is started})

**Code Candidates**

- Formal Parameters
- Class Name
- Methods
- Fields

<table>
<thead>
<tr>
<th>Candidate</th>
<th>Edit Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>on</td>
<td>8</td>
</tr>
<tr>
<td>Thread</td>
<td>7</td>
</tr>
<tr>
<td>start</td>
<td>2</td>
</tr>
<tr>
<td>sleep</td>
<td>8</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
(thread, is started)

**Code Candidates**
- Candidate: on, Edit Distance: 8
- Candidate: Thread, Edit Distance: 7
- Candidate: start, Edit Distance: 2
- Candidate: sleep, Edit Distance: 8

```
receiverObjectID.start()
```
receiverObjectID.setDaemon(args[0])

BEFORE

receiverObjectID.start()
receiverObjectID.setDaemon(args[0])

BEFORE

receiverObjectID.start()

Sentence tense?

Previous observed examples?
receiverObjectID.setDaemon(args[0])

BEFORE

receiverObjectID.start()
receiverObjectID.setDaemon(args[0])

**BEFORE**

receiverObjectID.start()
setDaemon(args[0])

```json
[
  {
    "operationSignature": "setDaemon",
    "isConstructor": false,
    "mustPrecede": "receiverObjectID.start()",
    "mustFollow": "",
  }
]
```
/**
 * This method must be invoked before the thread is started.
 */

public final void setDaemon(boolean on)
Evaluating CallMeMaybe

1. Can it accurately translate natural language to temporal specification?

2. Can it improve automatic test case generation?
Experimental setup

Popular Java systems
Experimental setup

Manually-written Java translations
Translation accuracy

83%  70%

Precision  Recall
Improving Randoop

CallMeMaybe improves Random output:

Reveals **False Alarms** in error test cases

Gives reasons for **Expected Exceptions** in regression test cases
@Test
public void test001(){
    IteratorChain<java.io.Serializable> serializableIt0 = new
            IteratorChain<java.io.Serializable>(. . .);
    // The following exception was thrown during execution
    try {
        serializableIt0.remove();
        org.junit.Assert.fail("Expected exception
            java.lang.IllegalStateException; message: 
            Iterator contains no elements");
    } catch (java.lang.IllegalStateException e) {
        // Expected exception.
        /* Violated CMM Constraint confirms this:
            "You will normally use addIterator(Iterator)
            to add some iterators after using this constructor." */
    }
}
False Alarms
in error-revealing test cases

@Test
public void test001(){
    IteratorEnumeration<String> strItEn0 =
        new IteratorEnumeration<String>();

    /* during test generation this statement threw an exception of type java.lang.NullPointerException in error
     
     But CMM Constraint was violated:
     "Constructs a new IteratorEnumeration that will not function
      until setIterator(Iterator) is invoked." */

    String str1 = strItEn0.nextElement();
}
Conclusions

- CallMeMaybe translates natural language temporal information into machine-readable specifications.
- Its translations are accurate (83% precision; 70% recall).
- Translations improve automatic test case generation by revealing false alarms and explaining exceptions.

https://github.com/ariannab/callmemaybe