Socio-Technical Solution to Large-Scale Formal Specification Mining

May1620

Members: Alex Dana & Cody Hanika
Client/Advisor: Professor Hridesh Rajan

Project Website: http://may1620.github.io/

What are formal specifications?

- Allow a programmer to describe the behavior of a piece of code in a programmatic and verifiable manner.
- Consist of a precondition and postcondition pair that specify what must be true about the program state
- Allow programmers to more easily write correct code
- Research has been conducted to create formal specification languages (JML) that can be applied to many programming languages

Informal vs. Formal Specifications

```java
/*
 * This function returns a divided by b. An exception will be thrown if
 * b is 0.
 */
public static int divide1(int a, int b) { return a / b; }

/*@   public normal_behavior
  @      requires b != 0;
  @      ensures \result == a / b;
  @ also
  @   public exceptional_behavior
  @      requires b == 0;
  @      signals_only ArithmeticException
 */
public static int divide2(int a, int b) { return a / b; }
```

The Problem

- Formal specifications are not widely available to programmers
- Many programmers do not know how to use formal specifications
- It can be difficult to write new formal specifications without a lot of experience using them
- It is difficult to automatically generate formal specifications by analyzing pieces of code

Initial Approach

- Use the success of the popular StackOverflow site to crowdsource the specifications
- A system will automatically post questions asking for specifications, then automatically pull down the answers
- Store these specifications in Boa, a software repository mining infrastructure, to make them available to users
- Eventually link our project with a custom Q&A site based on StackOverflow developed by another group

(Big) Risks

- As formal specifications are not well understood by the general programmer, there may be trouble getting quality responses from users
- StackOverflow may restrict us to posting very rarely

System Overview
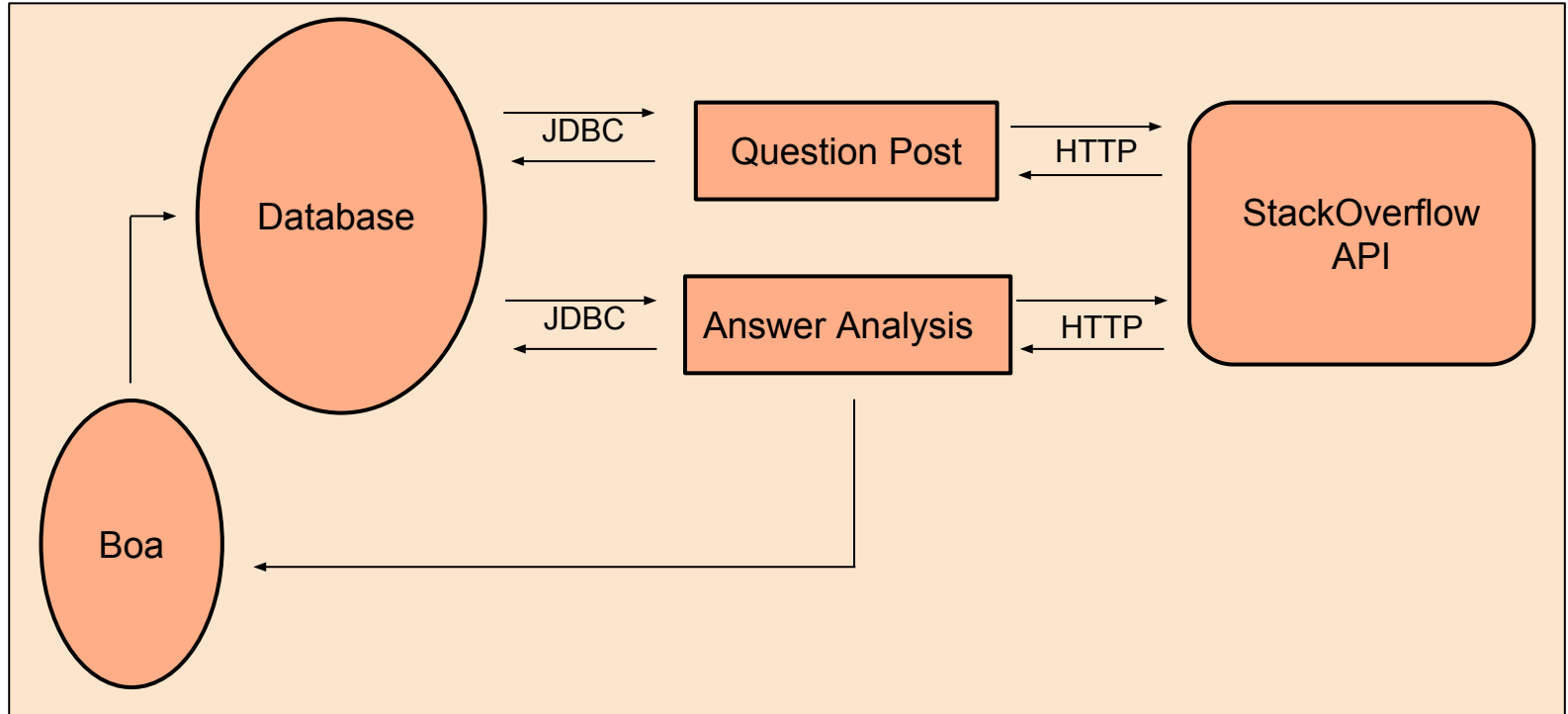
Posting Expanded

- Database grabs a function from Boa that does not have a spec created
- Question is generated based on the function
- Posting requires OAuth 2.0 authentication
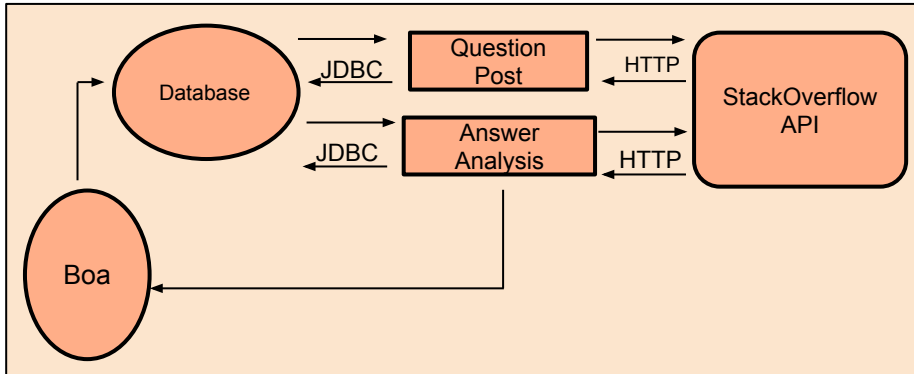
Analysis Expanded

- Grabs question ID of functions that currently have questions posted for them (that are not complete)
- We grab those questions via the API and then extract JML from them
  - Use upvotes and accepted answers to determine quality
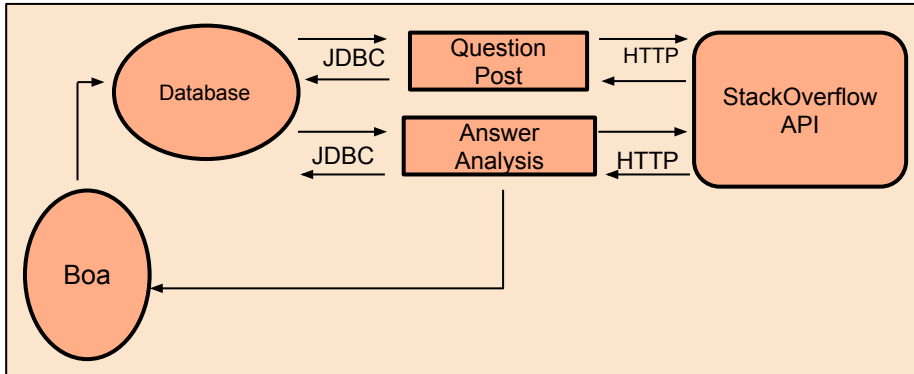
What is the proper JML for the java Scanner.hasNextInt(int) method?

0

I am using the java.util.Scanner class. I would like to have the JML for the nextInt(int) function. I am mainly concerned with the requires, ensures, assignable, and the signals_only clauses (although more is always better). So what is the JML for this method?

`java`   `java.util.scanner`   `specifications`   `jml`

share edit flag

add a comment

start a bounty

Our initial ideal question that would be answered

What assumptions can we safely make about the Java Collections shuffle(List<?>, Random) method?

So I am looking into the collections shuffle method and trying to come up with a list of what is and is not ensured when we run it. There are some obvious cases I've come up which are the following:

1. The list given will contain the same elements after shuffling as before
2. The list may or may not be the same after running the method (you could end up with the same order of elements)
3. The method will run in linear time (I think that this is true but am not 100% positive).

Does this list sum it up or am I missing some possible cases?

`java` `shuffle`

edited Jan 11 at 22:19

asked Jan 11 at 21:36
Alex1620
165 ● 15

asked   3 months ago

viewed   67 times

active   3 months ago

Successful question

The official documentation of `Collections.shuffle` has a lot to say about what will happen. The list will be shuffled using what seems to be the Fisher-Yates shuffle algorithm, which (assuming that random access is available in O(1)) runs in time O(n) and space O(1). The implementation will use space O(n) if random access isn't available. Assuming that the underlying random source is totally unbiased, the probability of any particular ordering occurring is equal (that is, you get a uniformly-random distribution over possible permutations).

So, to answer your questions:

1. The list will contain the same elements.

2. They're probably in a different order, but there's a 1 / n! chance than they'll be in the same order.

3. The runtime is O(n), and the space usage is either O(1) or O(n) depending on whether your list has random access support.

share edit flag

edited Jan 11 at 22:33

answered Jan 11 at 21:46

templatetypedef
183k ● 41 ● 461 ● 703

Unstructured answer

What can be safely assumed about the java Collections.sort(List<T>) method after it runs? [on hold]

So if we use the sort method what can we safely expect (assuming it ran properly)? So far using the javadoc I have:

- Our list will be the same size as it was before
- Equal elements won't be rearranged
- Our list will contain the same elements as before

`java` `sorting` `collections`

share edit delete flag

-8

asked today
viewed 33 times

asked 1 hour ago
Alex1620
149 • 15

put on hold as unclear what you're asking by Sean Bright, Cássio Mazzochi Molin, Hovercraft Full Of Eels, trrw, Trilarion 52 mins ago

Please clarify your specific problem or add additional details to highlight exactly what you need. As it's currently written, it's hard to tell exactly what you're asking. See the How to Ask page for help clarifying this question.

If this question can be reworded to fit the rules in the help center, please edit your question.

6 You can assume it will do everything it says in the Javadoc. Apart from just copying and pasting that, what sort of answer are you looking for? – resueman 1 hour ago

Typical comment/answer

Issues

- The risks we were initially concerned with became real problems
- The lack of JML knowledge forced us to ask more informal questions which were not typically well received
- When questions were answered they typically involved telling us to look at the documentation
- Our StackOverflow account was eventually banned

New Approach

- Use currently existing javadoc to gather informal specifications
  - Javadoc solves the structuring problem
- Generate JML utilizing a natural language processor
- Attempt to handle the most commonly occurring specifications first in order to maximize coverage

Goal for the New Approach

Throws:

   `NullPointerException` - if the specified collection is null

```
/*
 @ ...
 @ requires c == null;
 @ signals_only NullPointerException;
 */
public boolean addAll(Collection<? extends E> c)
```

New Risks

- This is a really difficult problem
- Limited time
- Our group had no prior knowledge of natural language processors
- Javadoc is not written in a consistent manner (helpful for parsing)

StanfordNLP

- Powerful many-featured natural language processor built with Java
- Breaks sentences into "tokens" (words) and analyzes based on things such as part of speech
- The analysis results in a parse tree

Example Parse Tree

"The Collection is null"

Sentence

Noun Phrase

Verb Phrase

The

Collection

is

null

POS: Def. Article

POS: Noun
NER: Parameter

POS: Verb

POS: Noun
NER: Java Keyword

New System Overview

Java Source File →

Parse Javadoc from JDK source	Use contextual information	Find common phrases	Match input against these phrases
JavadocParse →	**RegexNER** →	**TreeMatching** →	**TokensRegex**
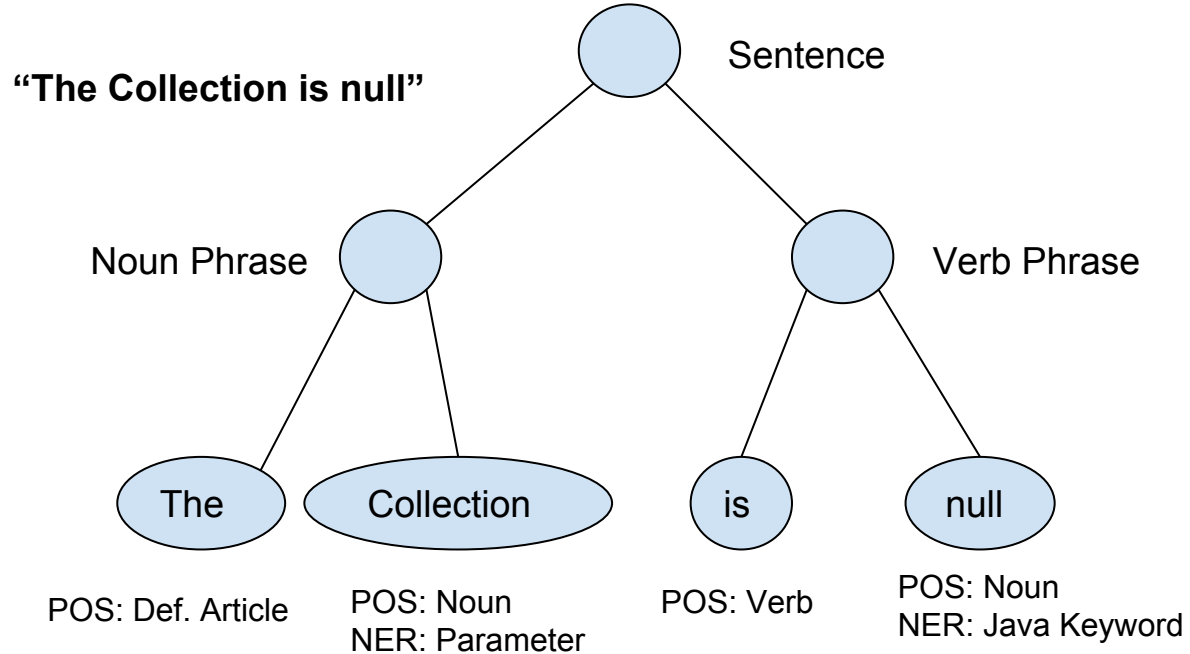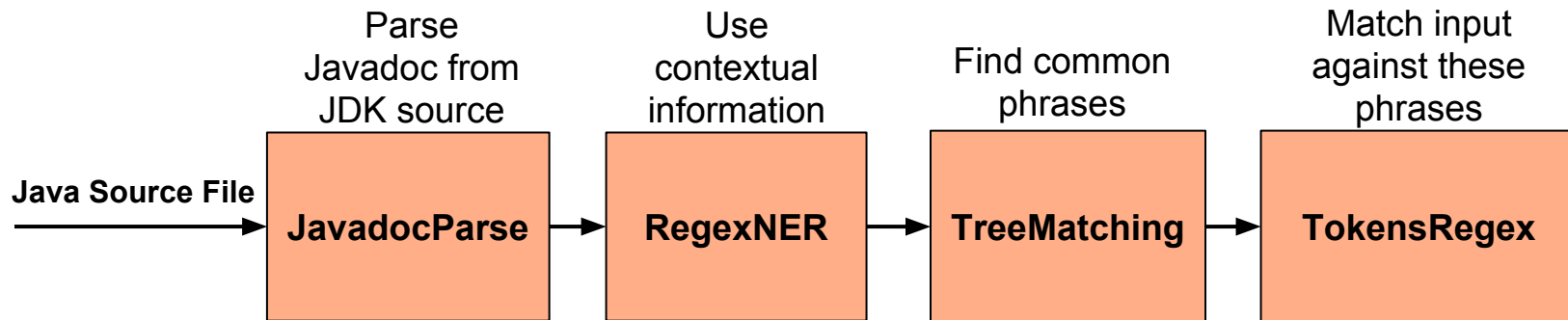
RegexNER Stage

- We need a way to use contextual information
  - Method name
  - Method parameter types/names
  - Java keywords
- This stage looks at each word and adds an NER annotation that marks words that relate to the context

```
public String substring(int beginIndex, int endIndex)
```

TreeMatching Stage

- As seen earlier SNLP creates parse trees
- Processing the entire JDK Javadoc results in similar parse trees (or similar subtrees of parse trees)
- These subtrees represent similar structure of informal specifications
- We find the most common identical subtrees to determine common informal spec patterns

TokensRegex Stage

- TokensRegex takes regular expression "rule" files
- These files dictate how informal specifications will be transformed

```
ENV.defaults["stage"] = 4
{
  ruleType: "tokens",
  pattern: ( [{ner:PARAMETER}] /is/ [{ner:NULL}]),
  result: Format("@requires %s == null", $0[0].value)
}
```

Result: 'Throws NullPointerException if collection is null' -> 'collection is null' ->
'PARAMETER is NULL' -> '@requires c == null;   @signals_only NullPointerException;'

Results

Java package	Match Rate
util	50%
io	11%
lang	25%
security	21%

Closing Remarks

May1620

Project Website: http://may1620.github.io/

Java Source File → JavadocParse → RegexNER → TreeMatching → TokensRegex

Parse Javadoc from JDK source	Use contextual information	Find common phrases	Match input against these phrases
JavadocParse	**RegexNER**	**TreeMatching**	**TokensRegex**

In Action
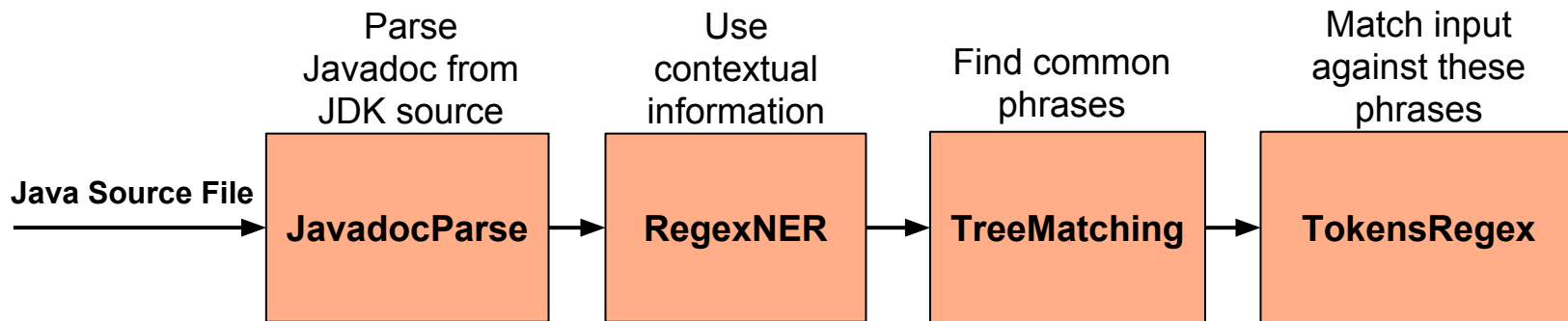
Analysis run on the Java util package