Synthesis of JML-Contracts via Large Language Models

Topic for "Praxis der Forschung"

Samuel Teuber

Large Language Models

- Significant progress in the past years
- Interesting capabilities with in-context learning
- Large Language Models already help with writing code (e.g. GitHub Copilot [1])

JML-Contracts

- Can be used for verification of Java Software
- Requires annotation of source-code
- Not only top-level specification, but also:
  - JML-Contracts for submethods
  - JML-Invariants for loops
- Annotation is a tedious process.

Code Input:

```java
public static char[] makePalindrome(char[] str) {
    if (isPalindrome(str)) {
        return str;
    }
    char[] result = new char[str.length * 2];
    copyInto(str, result, 0);
    copyInto(reverse(str), result, str.length);
    return result;
}
```

Question

What is the contract for `copyInto`?
Assume the following signature:

```java
void copyInto(char[] src, char[] dest, int pos)
```

Idea: Use LLMs to synthesize JML

- There already exist various methods for synthesis of specifications (see e.g. survey by Lathouwers et al. [2]).
- Preliminary experiments: Naively generating JML-Contracts does not work well.
- Generated invariants/sub-contracts are usually not helpful
- A working approach requires (at least):
  - Prompt-Engineering
  - Approaches to recover from insufficient contract proposals (maybe the work by Laurent et al. [3] could help)

How can we harness LLMs for the synthesis of useful JML-Contracts?

