

Simplify Automated Refactorings with Concrete Metapatterns

Work in Progress

A MasCot Software Restructuring Project (17933)
Luka Miljak, Msc.

in collaboration with:



Motivation

- Automated refactoring is necessary for **large and outdated codebases**
- Writing automated refactorings is hard; **steep learning curve**
- How to simplify writing refactoring implementations and **improve readability**?

Concrete Metapattern

Re-uses the syntax of the target language, like C++, Python, etc.

Case Study

API-Migration refactoring on C++ codebase of Philips Healthcare

Compile

Abstract Metapattern

Source Code

Match + Patch

Refactored code

Black-Box Parser required of the target language (e.g., C++ parser to support C++ patterns)

- Advantage:** Easy to interface with any new language
- Disadvantage:** Compilation requires some syntactic type hints



Spoofox Language Workbench;
Everything you need to design your own domain-specific programming language
www.spoofox.dev

Concrete Metapattern Examples

Example:
matching recursive function calls

```
Decl`
  int $foo() {
    <...
    Expr`$foo();`
    ...>
  }
```

<... ...>
Descendant Pattern
match at any depth

Expr Decl Stmt
Syntactic type hints needed for compilation

\$|
Disjunctive Pattern
match either left or right

Example:
matching both classes and structs

```
Decl`
  (class $| struct) $foo {
    $body
  }
```

-->
Inline Patch
perform a transformation

Example:
refactoring test assertions

```
Decl`
  TEST($f, $t) {
    <... Stmt`
      (EXPECT_TRUE($e1 == $e2); --> EXPECT_EQ($e1, $e2);)
    $| (EXPECT_FALSE($e1 == $e2); --> EXPECT_NE($e1, $e2);)
    ...>
  }
```