

Delft University of Technology

Simplify Automated Refactorings with Concrete Metapatterns work in Progress

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



Motivation

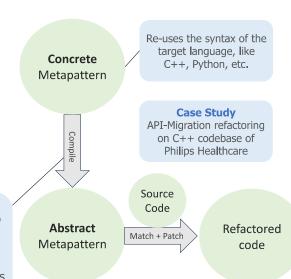
- Automated refactoring is necessary for large and outdated codebases
- Writing automated refactorings is hard; steep learning curve
- How to simplify writing refactoring implementations and improve readability?



Spoofax Language Workbench: Everything you need to design your own domain-specific programming language www.spoofax.dev

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



Concrete Metapattern Examples

Example:

matching recursive function calls

```
Decl```
int $foo() {
   Expr`$foo();
}
```

Expr Decl Stmt Syntactic type hints needed for compilation

Descendant Pattern match at any depth

Disjunctive Pattern match either left or right

Example:

matching both classes and structs

```
Decl```
(class $ | struct) $foo {
   $bodv
```

--> **Inline Patch**

Example:

refactoring test assertions

```
perform a transformation
```

Decl``` TEST(**\$f**, **\$t**) { <... Stmt` (EXPECT TRUE(\$e1 == \$e2); --> EXPECT EQ(\$e1, \$e2);) \$ (EXPECT_FALSE(\$e1 == \$e2); --> EXPECT_NE(\$e1, \$e2);)