要 旨

木構造を用いたパタンマッチによる
意味的誤り検出ツールの設計と実装

中村 真也

プログラムに含まれる誤りには，一般的なコンパイラで検出できる構文上の誤りと検出が困難である構文や型は正しい誤りがある．ここでは後者の誤りを意味的誤りと呼ぶ．意味的誤りは，コンパイラで検出できないだけではなく，プログラムを注意して見ても発見することは困難である．さらに，意味的誤りを含むプログラムは実行することができるため，予期しない動作をする可能性があり，しばしばセキュリティ問題の原因となる．したがって，これまでに意味的誤りを検出する研究が多くなされてきた．しかしながら，既存のツールは特定の規則に特化しているか，誤り検出モジュールを記述できる一般的なツールであってもモジュールの記述は簡単ではない．

本研究では，意味的誤り検出ツール ASTgrep を開発した．ASTgrep は，入力プログラムを解析しその抽象構文木（AST）を作成する AST 生成器と，その AST に対して意味的誤りのパタンマッチを行うパタンマッチ部の二つのモジュールから構成される．ASTgrep の利用者は，意味的誤りのパタンを AST 生成器で作られた AST と同様の形式で与える．これにより，利用者は生成された AST から簡単にパタンを作成することができる．

ケーススタディとして，JNI 規則違反のパタンを作成し，誤りが検出されることを検証した．この検証で，生成した AST からパタンを記述できるため，他のツールに比べてパタンの作成が簡単であることが確認できた．さらに，ASTgrep がコーディング規約違反を検出できるかを検討した．その結果，ASTgrep は，構文や型を基に検出できる意味的誤りについて有効であることが分かった．

キーワード　パタンマッチ，誤り検出，セキュリティ

Abstract

Design and Implementation of Semantic Error Detector Based on Pattern Matching Using Tree Patterns

Shinya Nakamura

While some bugs of computer programs, such as syntax errors, can be detected by general compilers, bugs that are correct on syntax and types are difficult to be detected. We call the latter "semantic errors". Not only by compilers, semantic errors are difficult to be discovered even by reading the program carefully. To make matters worse, programs that have semantic errors can run. Hence, they may show unexpected behavior, which often lead to security problems. Therefore, much work has been done to detect semantic errors so far. However, the existing tools are specialized in specific rules or, even for generalized tools where we can define error detection modules, making modules is not an easy task.

In this study, we developed a semantic error detecting tool, ASTgrep. ASTgrep comprises two modules; an abstract syntax tree (AST) generator, which parses the input program to generate its AST, and a pattern matcher, which matches patterns of semantic errors against the AST. The users gives patterns of semantic errors, which have similar forms of the AST that the AST genarator module generates. This allows the users to develop patterns easily from the AST.

For a case study, we developed patterns of JNI coding rule violations and ensured that violations are successfully detected. From this case study, we found that it was easier than other tools to develop rules, because we could develop patterns from the

generated AST. Furthermore, we examined whether ASTgrep is effective for detecting violations of a cording standard. As a result, we found that ASTgrep is effective for detecting semantic errors that can be determined based on syntax and types.

key words pattren match, bug detection, security