A Source Code Checker Using Declarative Patterns to Represent Rule Violations

Tomoharu Ugawa(*1) Seiji Umatani(*2) Shinya Nakamura(*1)
*1 Kochi University of Technology *2 Kyoto University

ASTgrep: a `lint` Tool for C/C++

- ✓ Search for potentially incorrect constructs, or *semantic errors*
  violations of library's conventions, insecure coding
- ✓ Support errors that can be determined by syntax or types
- ✓ Flexible and easy to use

Existing Tools

Rosecheckers ROSE compiler

```
bool JNIRule(const SgNode *node) {
  if (!isSgAssignmentOp(node))
    return false;
  const SgNode* lhs =
    node->get_lhs_operand();
  const SgVarRefExp* ref =
    isSgVarRefExp(lhs);
  …
}
```

☹ operational style

ASTMatchers Clang compiler

```
binaryOperator(
  hasOperatorName("="),
  hasLHS(ignoringParenImpCasts(
    declRefExpr(
      hasStaticStorageDuration()))),
  hasRHS(ignoringParenImpCasts(
    hasType(typedefType(
      …
```

☹ bunch of keyowrds

Flexible and Easy to Use

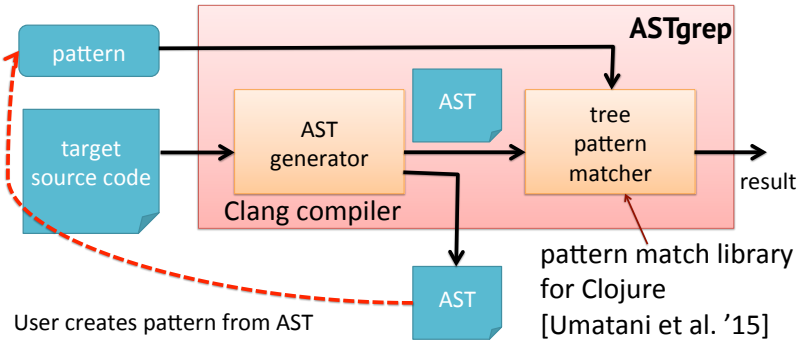
Idea the `grep` command

- ✓ We provide patterns
- ✓ We write patterns in a declarative style
- ✓ We often write patterns from an example

```
> grep "[a-zA-Z0-9_]+\s+==\s+NULL" xyz.c
```

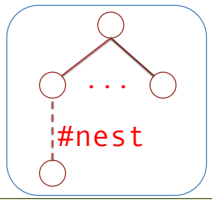
Design

- ✓ **ASTgrep** generates AST from the target source code
- ✓ **ASTgrep** matches tree pattern against the AST
- ✓ User creates pattern from the AST of source code that contains the error



bcmacro's pattern languaes

- ✓ Similar to AST
- ✓ Wildcards (variables, `...`, `#nest`)



Example: JNI Rule Violation

```
int f (jobject obj) {          xyz.c
  static jobject x;
  if (x == NULL)
    x = obj;
  /* use x */
}
```

potential bug

Java Native Interface does not guarantee value of `jobject` type is valid after returning

Pattern Development

ASTgrep

```
{:kind "BinOp"                          AST
 :op "="
 :loc [xyz.c 4]
 :lhs {:kind "VarRef" :name "x"
       :scope "local" :static "true"
       :type [{:typedef "jobject"}]}
 :rhs {:kind "VarRef" :name "obj"
       :scope "local"
       :type [{:typedef "jobject"}]}}
```

user

```
(defrule jni-rule-assign-static          pattern
 @{:kind "BinOp" :op "="
   :lhs {:static "true"}
   :rhs {:type […{:typedef "jobject"}]}}
 "assigning jobject to static var")
```

ASTgrep

```
                                         result
xyz.c:4:assigning jobject to static var
```

Case Study

- ✓ We detected JNI bugs in Android 2.3.7 that we had detected in our previous work using SEAN [Nishiwaki et al. '12]
  - ✓ Assignment of jobject values to non-local variables
  - ✓ Comparison between jobject values using == or !=

```
(defrule jni-rule-comp-eq
 @{:kind "BinOp" :op "=="
   :lhs {:type […{:typedef "jobject"}]}}
 "comparing jobject with ==")
```

  - ✓ Cast from jobject to other types

```
(defrule jni-rule-cast
 @{:type[{:typedef "jobject"} … x]}
 "cast from jobject to another type")
```