✅

info-inline-access-control

№ ID	SEM-17
Description	require & assert & revert conditons
Author	 현학 김
숨기기	☐
최종 편집 일시	@2024년 10월 27일 오전 11:33

💡

1. 모든 if-else 문법은 반드시 중괄호 {}를 사용한다.
2. 명시적인 revert, require, assert 활용만 검사한다.
    a. Wrapper 라이브러리는 분석할 수 없다.
    b. 일반 명령에 대한 실행 조건은 분석할 수 없다.
    c. yul 기반 접근 제어는 검사할 수 없다

1. 소스 전문
2. 전처리 (필터링)
    - str.replace
    - re.sub
3. Tokenizing (현재 별도의 클래스 없음)

> 임시로 명명

`$elf$` / `$if$"` / `$els$` / `$req$` / `$asrt$` / `$revX$` / `$rev$`

4. Parser (현재 별도의 클래스 없음)

▼ Tokenizing / Parsing 🚧

```
   _given = open("code/1.sol", "r").read()

    # TODO: <temp> focus on contract scope
    _given = re.search(r"contract\s+[\s\S]+}", _given).g
roup(0)

    # remove comments
    _given = re.sub(r"//[\S ]+", "", _given)

    # # remove library
    # _given = re.sub(r"using\s+[\s\S]+?;", "", _given)

    # all kinds of spaces to single space
    _given = re.sub(r"\s+", " ", _given)

    # tokenize
    _given = (
        _given
        .replace("else if", "$elf$")
        .replace("if", "$if$")
        .replace("else", "$els$")
        .replace("require", "$req$")
        .replace("assert", "$asrt$")
        .replace("revert()", "$revX$")
        .replace("revert", "$rev$")
        # .replace("()", "")
    )

    pattern = r"\$\w+\$\s*\([\s\S]+?\)\;|[\{\}]|\$\w+\$"
    _given = "\n".join(re.findall(pattern, _given))
    _given = re.sub(r"\$rev\$[\s\S]+?;", "$revX$;", _giv
en)

    print(_given)

    # remove all except patterns
    # 1.
```

```
    _given = (_given
              .replace("$req$", "\nrequire\n")
              .replace("$asrt$", "\nassert\n")
              .replace("$rev$", "\nrevert\n")
              .replace("$revX$", "\nrevert\n()\n")
                .replace("$if$", "\nif\n")
                .replace("$els$", "\nelse\n")
                .replace("$elf$", "\nelse if\n")
                .replace("{", "\n{\n")
                .replace("}", "}\n")
              .replace(";", " ")
              )

    _given = _given.splitlines()
    _given = list(map(str.strip, _given))

    stack = ConditionStack()
    for line in _given:
        if line != "":
            # print(line)
            stack_dump = stack.__str__()
            stack.push(line.strip())
            if stack_dump != stack.__str__():
                print(stack)
                pass
```

5. Lexer (Logic, LogicNode)
    a. Lexer의 AND 또는 OR 연산은 각각의 피연산자 또한 소괄호로 wrapping 되어 Python의 and / or 연산으로 연결된다.

테스트 1✅

테스트 1: 중첩된 if-else문

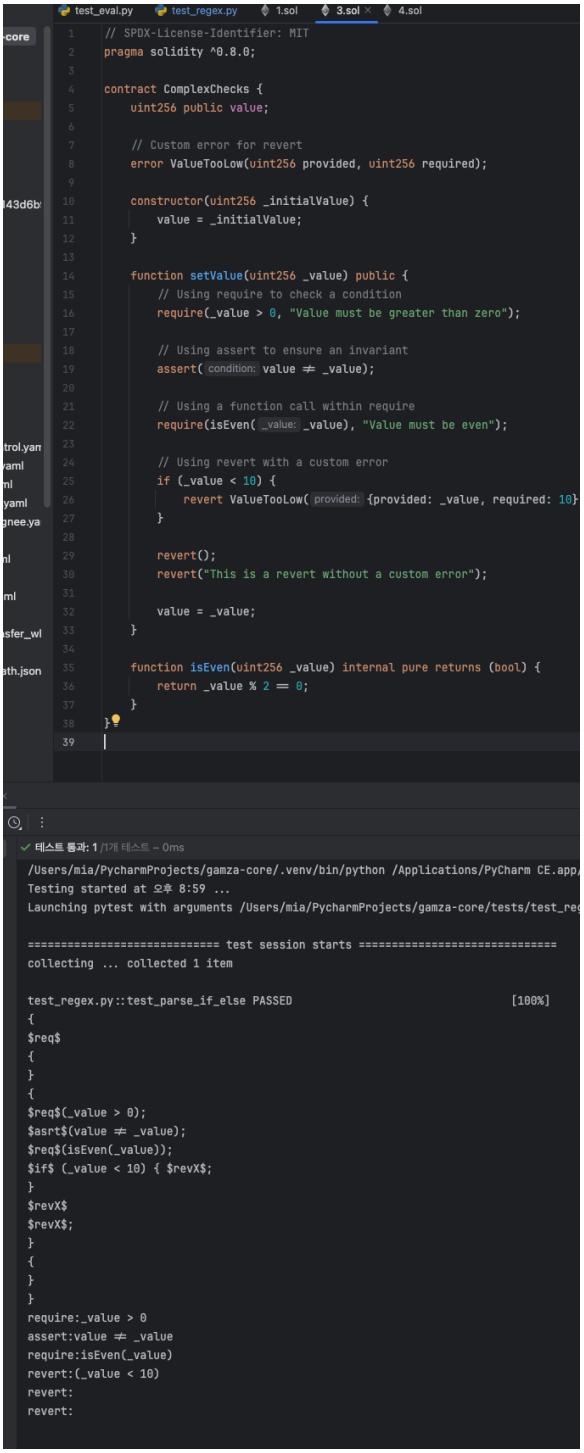
▼ 테스트 소스

```
    contract Test {
        {
            if (x > 0) {
                if (y > 0) {
                // do nothing
                }
            }
        }
        if (x > 0) {
            revert(); // ✅ #1 revert:(x > 0)
            for (int i = 0; i < 10; i++) {
                if (i == 0) {
                    revert(); // ✅ #2 revert:(#1 and (i
== 0))
                } else if (i == 1) {
                    assert(y > 0); // ✅ #3 assert:((#1
and ((i == 1) and not(#2) and (y > 0))) and y > 0)
                }
            }
        x > 0;
        if (y > 0) {
            revert(); // ✅ #4 revert:(y > 0)
        } else if (z > 0) {
            assert(k > 0); // ✅ #5 assert:((z > 0) and
not(#4)) and k > 0)
```

```
        } else {
            require(y > 0 or z > 0); // ✅ #6 require:(n
ot(((z > 0) and not((y > 0)))) and y > 0 or z > 0)
        }
    }
    }
```

테스트 2 ✅

> 💡 평문을 parsing하여 동작시켰으나, 범용성이 없어서 tokenizer의 도입을 결정했다.



테스트 2: 메세지를 포함하는 접근 제어문

▼ 테스트 소스

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract ComplexChecks {
```

```
    uint256 public value;

    // Custom error for revert
    error ValueTooLow(uint256 provided, uint256 require
d);

    constructor(uint256 _initialValue) {
        value = _initialValue;
    }

    function setValue(uint256 _value) public {
        // Using require to check a condition
        require(_value > 0, "Value must be greater than
zero");

        // Using assert to ensure an invariant
        assert(value != _value);

        // Using a function call within require
        require(isEven(_value), "Value must be even");

        // Using revert with a custom error
        if (_value < 10) {
            revert ValueTooLow({provided: _value, requir
ed: 10});
        }

        revert();
        revert("This is a revert without a custom erro
r");

        value = _value;
    }

    function isEven(uint256 _value) internal pure return
s (bool) {
        return _value % 2 == 0;
```

```
    }
}
```

테스트 3 🚧

> 고도화가 필요하다면 Slither 활용 가능성 고려

▼ legacy

중첩된 if에 대한 처리

and not ({operation2})

`# if (5 > 3)`

`# 5 <= 3 or 2 >= 4`

not(op1 or op2)

각 수식은 수정 못 하도록 보존

💡 현재 top-level과 revert 한정 1개의 if 스코프에 대해서만 검사를 진행

```
rules:
  - id: info-access-control
    languages:
      - solidity
    severity: ERROR
    message: $CONTRACT | $SIG | $METHOD | $COND
    pattern-either:
      - patterns:
          - pattern-inside: contract $CONTRACT { ... ...
```

```
... }
          - pattern-inside: function $SIG ( ... ) { ...
... ... }
          - pattern-not-inside: if ( ... ) { ... ... ...
}
          - pattern-either:
              - pattern-regex: |-
                  (?P<METHOD>assert)\s*\((?P<COND>[\S ]
+?)\);
              - pattern-regex: |-
                  (?<METHOD>require)\s*\((?<COND>[\S\s]
+?)[,\);]
              - pattern-regex: |-
                  (?P<METHOD>revert)\s*\((?:[\S ]+?)\);
      - patterns:
          - pattern-inside: contract $CONTRACT { ... ...
... }
          - pattern-inside: function $SIG ( ... ) { ...
... ... }
          - pattern-inside: if ( ... ) { ... ... ... }
          - pattern-regex: |-
              if\s+\((?<COND>[\S\s]+?)\)\s*\{[\s\S]+?\}
          - pattern-regex: |-
              (?P<METHOD>revert)\(
    metadata:
      category: maintainability
      technology:
        - solidity
```