

Proveravači za Clang

Svetlana Bićanin, Lucija Miličić, Aleksandra Pešić

Matematički fakultet

Verifikacija softvera

Sadržaj

- 1 Uvod
- 2 Provera dodele u uslovima grananja
 - Primeri upotrebe
 - Implementacija
- 3 Provera beskonačnih petlji
 - Primeri upotrebe
 - Implementacija
- 4 Problemi i ograničenja

Sadržaj

- 1 Uvod
- 2 Provera dodele u uslovima grananja
 - Primeri upotrebe
 - Implementacija
- 3 Provera beskonačnih petlji
 - Primeri upotrebe
 - Implementacija
- 4 Problemi i ograničenja

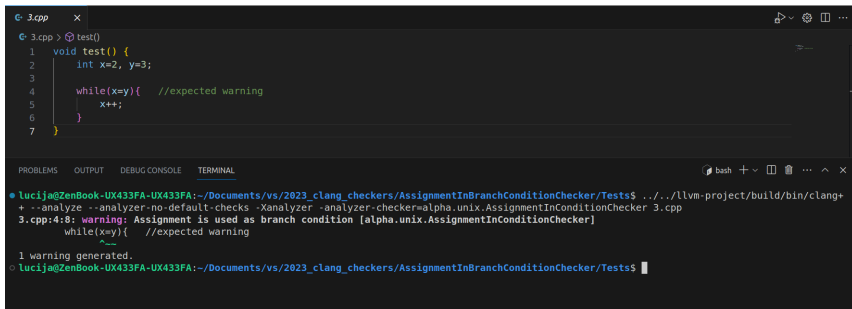
Uvod

- Clang statički analizator
- Dostupni proveravači za bezbednosne provere, korišćenje API funkcija, traženje mrtvog koda, logičkih grešaka...
- Implementirani novi proveravači:
 - Provera dodele u uslovima grananja
 - Provera beskonačnih petlji

Sadržaj

- 1 Uvod
- 2 Provera dodele u uslovima grananja
 - Primeri upotrebe
 - Implementacija
- 3 Provera beskonačnih petlji
 - Primeri upotrebe
 - Implementacija
- 4 Problemi i ograničenja

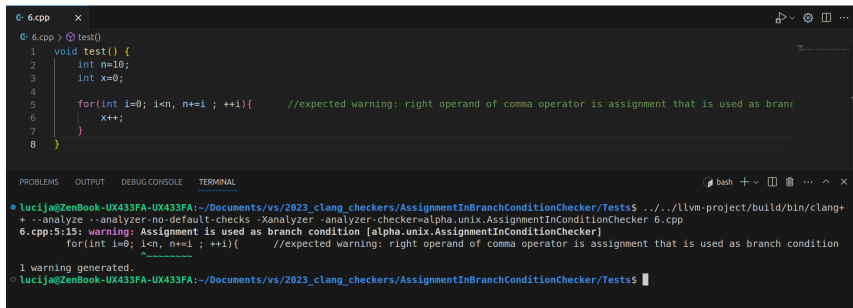
AssignmentInConditionChecker



```
3.cpp x
3.cpp > test()
1 void test() {
2     int x=2, y=3;
3
4     while(x=y){ //expected warning
5         x++;
6     }
7 }
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
lucija@ZenBook-UX433FA-UX433FA:~/Documents/vs/2023_clang_checkers/AssignmentInBranchConditionChecker/Tests$ ../../llvm-project/build/bin/clang+
+ --analyze --analyzer-no-default-checks -Xanalyzer -analyzer-checker=alpha.unix.AssignmentInConditionChecker 3.cpp
3.cpp:4:8: warning: Assignment is used as branch condition [alpha.unix.AssignmentInConditionChecker]
    while(x=y){ //expected warning
          ^~~
1 warning generated.
lucija@ZenBook-UX433FA-UX433FA:~/Documents/vs/2023_clang_checkers/AssignmentInBranchConditionChecker/Tests$
```

AssignmentInConditionChecker

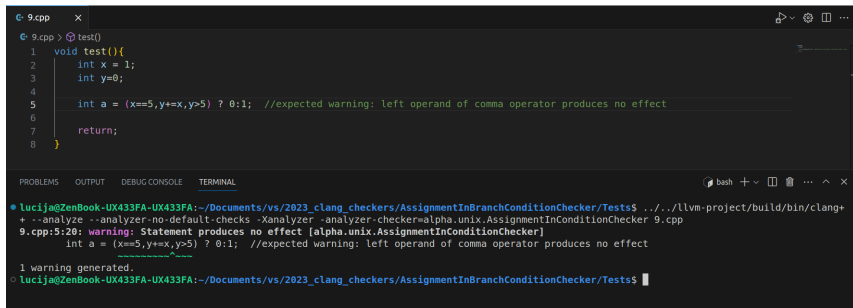


```
6.cpp x
6.cpp > test()
1 void test() {
2     int n=10;
3     int x=0;
4
5     for(int i=0; i<n, n+=i ; ++i){ //expected warning: right operand of comma operator is assignment that is used as branch
6         x++;
7     }
8 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
• lucija@ZenBook-UX433FA-UX433FA:~/Documents/vs/2023_clang_checkers/AssignmentInBranchConditionChecker/Tests$ ../../llvm-project/build/bin/clang+
+ --analyze --analyzer-no-default-checks -Xanalyzer -analyzer-checker=alpha.unix.AssignmentInConditionChecker 6.cpp
6.cpp:5:15: warning: Assignment is used as branch condition [alpha.unix.AssignmentInConditionChecker]
    for(int i=0; i<n, n+=i ; ++i){ //expected warning: right operand of comma operator is assignment that is used as branch condition
                   ^~~~~~
1 warning generated.
o lucija@ZenBook-UX433FA-UX433FA:~/Documents/vs/2023_clang_checkers/AssignmentInBranchConditionChecker/Tests$
```

AssignmentInConditionChecker



```
9.cpp x
9.cpp > test()
1 void test(){
2     int x = 1;
3     int y=0;
4
5     int a = (x==5,y+=x,y>5) ? 0:1; //expected warning: left operand of comma operator produces no effect
6
7     return;
8 }
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
lucija@ZenBook-UX433FA-UX433FA:~/Documents/vs/2023_clang_checkers/AssignmentInBranchConditionChecker/Tests$ ../../llvm-project/build/bin/clang+
+ --analyze --analyzer-no-default-checks -Xanalyzer -analyzer-checker=alpha.unix.AssignmentInConditionChecker 9.cpp
9.cpp:5:20: warning: Statement produces no effect [alpha.unix.AssignmentInConditionChecker]
    int a = (x==5,y+=x,y>5) ? 0:1; //expected warning: left operand of comma operator produces no effect
               ^~~~~~
1 warning generated.
lucija@ZenBook-UX433FA-UX433FA:~/Documents/vs/2023_clang_checkers/AssignmentInBranchConditionChecker/Tests$
```

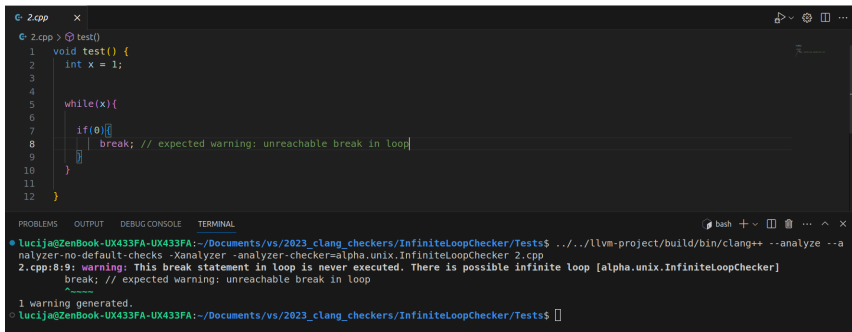

AssignmentInConditionChecker

- `Checker<check::BranchCondition>`
- `void checkBranchCondition(const Stmt *Condition, CheckerContext &Ctx) const;`
- `ImplicitCastExpr, BinaryOperator, ParenExpr`
- `isAssignment, checkCommaOp`

Sadržaj

- 1 Uvod
- 2 Provera dodele u uslovima grananja
 - Primeri upotrebe
 - Implementacija
- 3 Provera beskonačnih petlji
 - Primeri upotrebe
 - Implementacija
- 4 Problemi i ograničenja

InfiniteLoopChecker

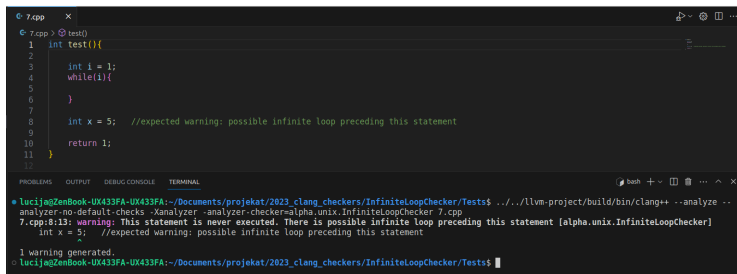


```
2.cpp x
2.cpp > test()
1 void test() {
2   int x = 1;
3
4
5   while(x){
6
7     if(0){
8       break; // expected warning: unreachable break in loop
9     }
10  }
11
12 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
bash +v [] ... ^ x
• lucija@ZenBook-UX433FA-UX433FA:~/Documents/vs/2023_clang_checkers/InfiniteLoopChecker/Tests$ ../../llvm-project/build/bin/clang++ --analyze --a
nalyzer-no-default-checks -Xanalyzer -analyzer-checker=alpha.unix.InfiniteLoopChecker 2.cpp
2.cpp:8:9: warning: This break statement in loop is never executed. There is possible infinite loop [alpha.unix.InfiniteLoopChecker]
      break; // expected warning: unreachable break in loop
      ^~~~~~
1 warning generated.
o lucija@ZenBook-UX433FA-UX433FA:~/Documents/vs/2023_clang_checkers/InfiniteLoopChecker/Tests$
```

InfiniteLoopChecker



```
7.cpp
1 int test(){
2
3     int i = 1;
4     while(i){
5
6     }
7
8     int x = 5; //expected warning: possible infinite loop preceding this statement
9
10    return 1;
11 }
12

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
lucija@ZenBook-UX433FA-UX433FA:~/Documents/projekat/2023_clang_checkers/InfiniteLoopChecker/Tests$ ../../llvm-project/build/bin/clang++ --analyze --analyzer-no-default-checks -Xanalyzer -analyzer-checker=alpha.unix.InfiniteLoopChecker 7.cpp
7.cpp:8:13: warning: This statement is never executed. There is possible infinite loop preceding this statement [alpha.unix.InfiniteLoopChecker]
    int x = 5; //expected warning: possible infinite loop preceding this statement
    ^
1 warning generated.
lucija@ZenBook-UX433FA-UX433FA:~/Documents/projekat/2023_clang_checkers/InfiniteLoopChecker/Tests$
```


InfiniteLoopChecker

```
1.cpp > ...
1 void test() {
2
3 while(1){ //expected warning: infinite loop without break or return
4
5 }
6
7 }
8
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
bash + v [ ] ... ^ x
lucija@ZenBook-UX433FA-UX433FA:~/Documents/vs/2023_clang_checkers/InfiniteLoopChecker/Tests$ ../../llvm-project/build/bin/clang++ --analyze --analyzer-no-default-checks -Xanalyzer -analyzer-checker=alpha.unix.InfiniteLoopChecker 1.cpp
1.cpp:3:3: warning: This loop is infinite, possibly missing break or return statement. [alpha.unix.InfiniteLoopChecker]
while(1){ //expected warning: infinite loop without break or return
^
-----
1 warning generated.
lucija@ZenBook-UX433FA-UX433FA:~/Documents/vs/2023_clang_checkers/InfiniteLoopChecker/Tests$
```


InfiniteLoopChecker



```
1 void test() {
2     int x = 1;
3
4     while(x){
5
6         if(x){
7             break;
8         }else{
9             while(1){ // expected warning: infinite loop
10
11             }
12         }
13     }
14 }
15 }
```

```
bash + -
1 warning generated.
lucija@ZenBook-UX433FA-UX433FA:~/Documents/vs/2023_clang_checkers/InfiniteLoopChecker/Tests$
```

InfiniteLoopChecker



The screenshot shows a C++ IDE with a file named 13.cpp. The code defines a function test() containing a while(1) loop. Inside this loop, there is an int x = 0; declaration and another while(1) loop. The inner loop has an if(x == 0) condition that triggers a break; statement. The IDE's terminal window displays the output of a clang++ command, showing a warning from the alpha.unix.InfiniteLoopChecker that the loop is infinite because the break statement is unreachable.

```

13.cpp > test()
1  void test(){
2
3      while(1){
4          int x = 0;
5
6          while(1){
7              if(x == 0){
8                  break;
9              }
10         }
11     }
12 }
13 }

```

```

bash +  [ ] [ ] ... ^ x
lucija@ZenBook-UX433FA-UX433FA:~/Documents/projekat/2023_clang_checkers/InfiniteLoopChecker/Tests$ ../../llvm-project/build/bin/clang++ -analyze --analyzer-no-default-checks -Xanalyzer -analyzer-checker=alpha.unix.InfiniteLoopChecker 13.cpp
13.cpp:3:5: warning: This loop is infinite, possibly missing break or return statement. [alpha.unix.InfiniteLoopChecker]
    while(1){
    ^~~~~~
1 warning generated.
lucija@ZenBook-UX433FA-UX433FA:~/Documents/projekat/2023_clang_checkers/InfiniteLoopChecker/Tests$

```

InfiniteLoopChecker

- `Checker<check::EndAnalysis>`
- `void checkEndAnalysis(ExplodedGraph &G, BugReporter &B, ExprEngine &Eng) const;`
- `static inline const Stmt *getUnreachableStmt(const CFGBlock *CB);`
- `isInLoop, isReturnStmt, hasBreakStmt`
- Dva pristupa rešavanju problema
 - analiza nedostižnog koda
 - analiza konstantnih uslova petlje

Sadržaj

- 1 Uvod
- 2 Provera dodele u uslovima grananja
 - Primeri upotrebe
 - Implementacija
- 3 Provera beskonačnih petlji
 - Primeri upotrebe
 - Implementacija
- 4 Problemi i ograničenja

Problemi i ograničenja

- UnixAlpha - eksperimentalni proveravači
- AssignmentInConditionChecker: `while(x- = 1)`
- InfiniteLoopChecker: konstantni uslovi, mogućnost za proširenje