

Sanitizer & LLVM Pass

SWPP

2022. 3. 28

Seunghyeon Nam (slides by Juneyoung Lee)

Clang Sanitizer

- A tool that helps you detect undefined behaviors at runtime
- `clang -fsanitize=XXX a.c`
 - `undefined`: detects UBs from arithmetic operations
 - `address`: detects use-after-free, etc
 - `memory`: detects reading uninitialized memory
 - They are all undefined behaviors in C!

Running Example

```
// ubsan.c
#include<stdio.h>

int main() {
    printf("Type two positives to calculate average: ");
    int a, b;
    scanf("%d %d", &a, &b);

    int average = (a + b) / 2;
    printf("Average: %d\n", average);

    return 0;
}
```

Solution

```
// ubsan.c
#include<stdio.h>

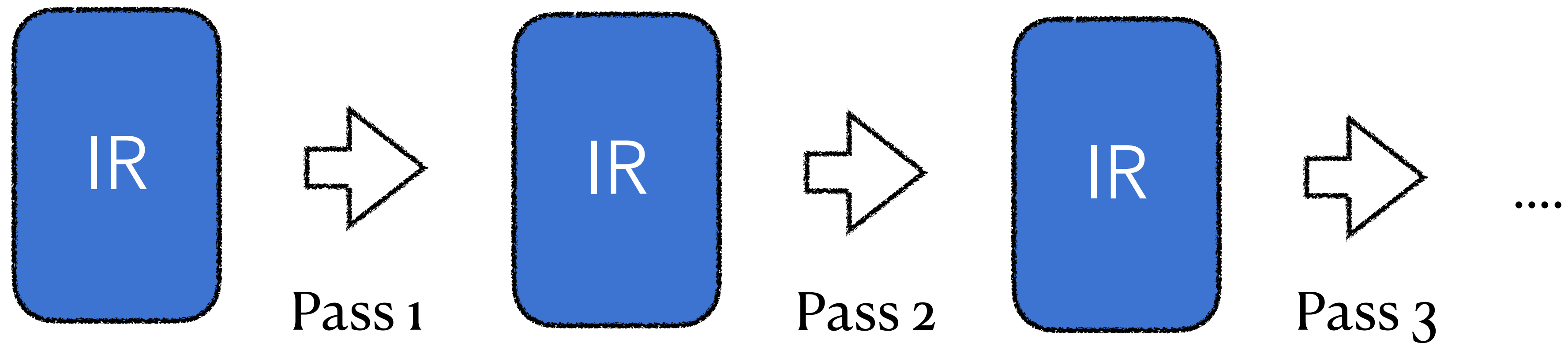
int main() {
    printf("Type two positives to calculate average: ");
    int a, b;
    scanf("%d %d", &a, &b);

    int average = a + (b - a) / 2;
    printf("Average: %d\n", average);

    return 0;
}
```

Let's see other three tests as well!

LLVM IR Pass



- Each transformation (or optimization) is called pass.
- -O1, -O2, -O3: a set of (more) passes.
- Prerequisite: LLVM

1. HelloPass

- Full code: hello.cpp

```
class HelloPass : public PassInfoMixin<HelloPass> {
public:
    PreservedAnalyses run(Function &F, FunctionAnalysisManager &FAM) {
       StringRef funcName = F.getName();
        outs() << "Hello, " << funcName << "!\n";
        return PreservedAnalyses::all();
    }
};

extern "C" ::llvm::PassPluginLibraryInfo
llvmGetPassPluginInfo() {
    // Registration of HelloPass: omitted for brevity
}
```

Hierarchy

- `llvm::Module` class
- `llvm::Function` class
- `llvm::BasicBlock` class
- `llvm::Instruction` class

To see class hierarchy & their methods..

Search Google / Read code / See Autocompletions



inherits (is-a)

`llvm::LoadInst`

`llvm::ICmpInst`

`llvm::BinaryOperator`

`llvm::StoreInst`

`llvm::BranchInst`

`llvm::PHINode...`

How To Run HelloPass?

- Compile: it is slightly complex.. Please use `./run-passes.sh build <build dir>`
 - If you like to challenge, have a look at the script! It isn't very hard
- Run: `./run-passes.sh run <build dir>` also works, but you can try:
 - `opt -disable-output -load-pass-plugin=libHello.so -passes="hello" foo.ll`

2. DumpPass

```
class DumpPass : public PassInfoMixin<DumpPass> {
public:
    PreservedAnalyses run(Function &F, FunctionAnalysisManager &FAM) {
        outs() << "<<" << F.getName() << ">>\n";

        for (BasicBlock &BB : F) {
            outs() << "BasicBlock: " << BB.getName() << "\n";

            for (Instruction &I : BB)
                outs() << "\t" << I << "\n";
        }
        return PreservedAnalyses::all();
    }
};
```

Print Successors

```
class DumpPass : public PassInfoMixin<DumpPass> {
public:
    PreservedAnalyses run(Function &F, FunctionAnalysisManager &FAM) {
        for (BasicBlock &BB : F) {
            outs() << "BasicBlock: " << BB.getName() << "\n";

            unsigned successorCnt = BB.getTerminator()->getNumSuccessors();

            for (unsigned i = 0; i < successorCnt; ++i) {
                BasicBlock *NextBB = BB.getTerminator()->getSuccessor(i);
                outs() << NextBB->getName() << "\n";
            }
        }
        return PreservedAnalyses::all();
    }
};
```