

Implementing Optimization

2023.03.30

SWPP Practice Session

Seunghyeon Nam (with lots of derived works)

Optimization

- Rewriting the code so that it does the same job faster
- In LLVM: use **optimization pass**
- Optimization pass may do one of the following:
 - Refactor the code into **simpler form**
 - Rewrite simplified code to **make them run faster**

Example – Constant Folding

- Reduce simple constant arithmetic into a single constant
- Code and examples are at class repository!

```
define i32 @constant_fold() {  
    %a = add i32 1, 2  
    %b = sub i32 %a, 1  
    ret i32 %b  
}
```



```
define i32 @constant_fold() {  
    ret i32 2  
}
```

Example – Use to Undef

- Replace values used in undef_zone to undef
- Code and examples are at class repository!

```
define i32 @f(i1 %cond, i32 %arg) {  
    %inst = add i32 1, 1  
    br i1 %cond, label %undef_zone, label %normal_zone  
undef_zone:  
    %x1 = add i32 %arg, 0 ; %x1 = add i32 undef, 0  
    ret i32 %inst          ; ret i32 undef  
normal_zone:  
    %x2 = add i32 %arg, 0  
    ret i32 %inst  
}
```

Example – Instruction Matching

- Matches values yielded from certain instruction sequence
- Code and examples are at class repository!

Testing the Optimization

- How do we know if our optimization works as intended?
- Testing is an easier and most widely used method
- LLVM offers `FileCheck` for IR-based testing

FileCheck

- Syntactic check (regex match)
- Write regexes to match against as comments in IR

```
define i32 @constant_fold() {  
; CHECK-LABEL: @constant_fold(  
; CHECK-NEXT:      ret i32 2  
  %a = add i32 1, 2  
  %b = sub i32 %a, 1  
  ret i32 %b  
}
```

```
define i32 @f(i32 %x, i32 %y) {  
; CHECK-LABEL: @f(  
; CHECK-NEXT: [[Z:%.*]] = add i32 %x, %y  
; CHECK-NEXT: ret i32 [[Z]]  
  %z = add i32 %x, %y  
  %z2 = add i32 %z, 0  
  ret i32 %z2  
}
```

FileCheck Test Directives

- **Test directives**: Instruct how to match output texts
 - **CHECK** : Simply check for existence of match
 - **CHECK-NEXT** : Check that the very next line matches the pattern
 - **CHECK-NOT** : Check that such pattern does not exist
 - **CHECK-LABEL** : Only match the code inside the specified block
- Check the [official documentation page](#) for more details

How to Use FileCheck

- `~/llvm-swpp/bin/opt -passes="my-opt" src.ll -S -o tgt.ll`
- `~/llvm-swpp/bin/FileCheck src.ll < tgt.ll`

Verifying the Optimization

- Optimization is a behavioral refinement
- Does our optimization actually ‘refines’?
 - Reasoning about the refinement is difficult
 - ... Which makes verifying the optimization hard as well
- Solution: let program to do the reasoning for us!

Alive2

- Automatically checks refinement between pair of IRs
- Take a look at `install-alive2.sh` to install Alive2
- `/<alive2-build-dir>/alive-tv src.ll tgt.ll`
- Or use the [interactive webpage](#)
- We may require you to use Alive2 in the project!