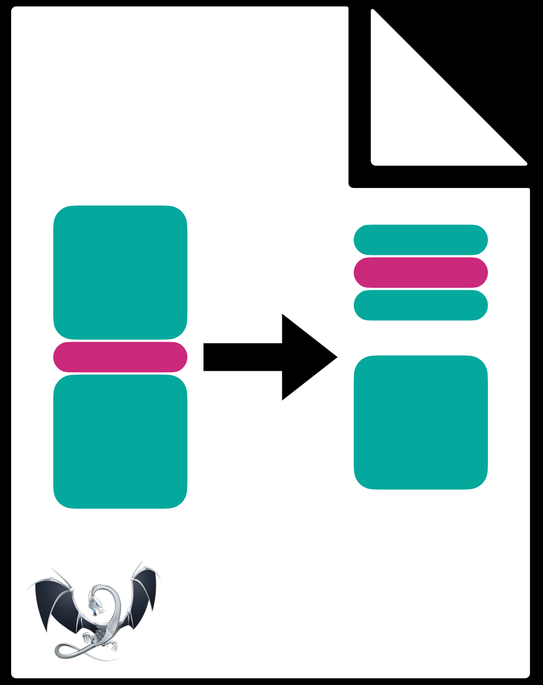
What’s New in the IR Similarity Identifier and Outliner

Andrew Litteken (Presenter), Jessica Paquette
Apple

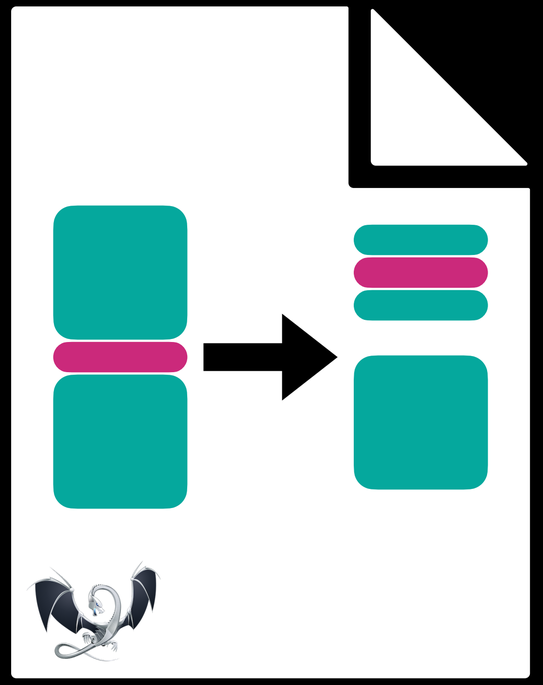
Previous Work

- 2016: Suffix Tree and Machine Outliner
  https://www.youtube.com/watch?v=yorld-WSOeU



Previous Work

- 2016: Suffix Tree and Machine Outliner
  https://www.youtube.com/watch?v=yorld-WSOeU
- 2020: IR Similarity and IR Outliner
  https://www.youtube.com/watch?v=HaN83qMyAhY



What is IR Similarity Identification and Outlining?

```
%0 = load i32, i32* %a
%add = add i32 %0, 2
%1 = load i32, i32* %b
%add1 = add i32 %1, 3

%sub = sub i32 %add, %v

%2 = load i32, i32* %c
%add2 = add i32 %2, 2
%3 = load i32, i32* %d
%add3 = add i32 %3, 3

%div = div i32 %add3, %v
```

What is IR Similarity Identification and Outlining?

```
%0 = load i32, i32* %a
%add = add i32 %0, 2
%1 = load i32, i32* %b
%add1 = add i32 %1, 3

%sub = sub i32 %add, %v

%2 = load i32, i32* %c
%add2 = add i32 %2, 2
%3 = load i32, i32* %d
%add3 = add i32 %3, 3

%div = div i32 %add3, %v
```

What is IR Similarity Identification and Outlining?

```
call void @outlined_function(i32*
  %a, i32* %b, i32* %output1, i32 0)
%add = load i32, i32* %output1

%sub = sub i32 %add, %v

call void @outlined_function(i32*
  %a, i32* %b, i32* %output2, i32 1)
%add3 = load i32, i32* %output2

%div = div i32 %add3, %v
```

```
define internal void @outlined_function(
  i32* %a, i32* %b, i32* %output, i32 %4) {
  %entry:
    %0 = load i32, i32* %a, align 4
    %add = add i32 %0, 2
    %1 = load i32, i32* %b, align 4
    %add1 = add i32 %1, 3
    switch i32 %4, label %final
        [ i32 0, label %output
          i32 1, label %output_1 ]
  %output:
    store i32 %add, i32* %output
  %output_1:
    store i32 %add, i32* %output
}
```

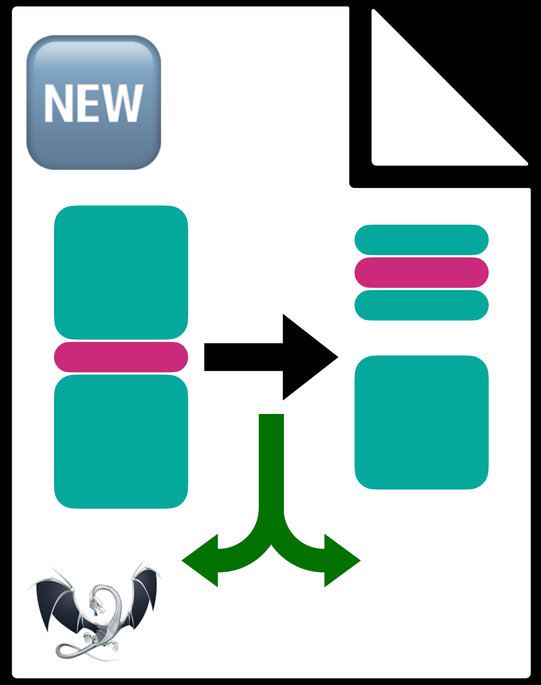
Code Size improvements on SwiftUI

swiftui.ll

Optimization	Size (Text)	Saving
-Os	14.12 MB	-
Simple IR Outliner	14.115 MB	-0.04%

Improvements

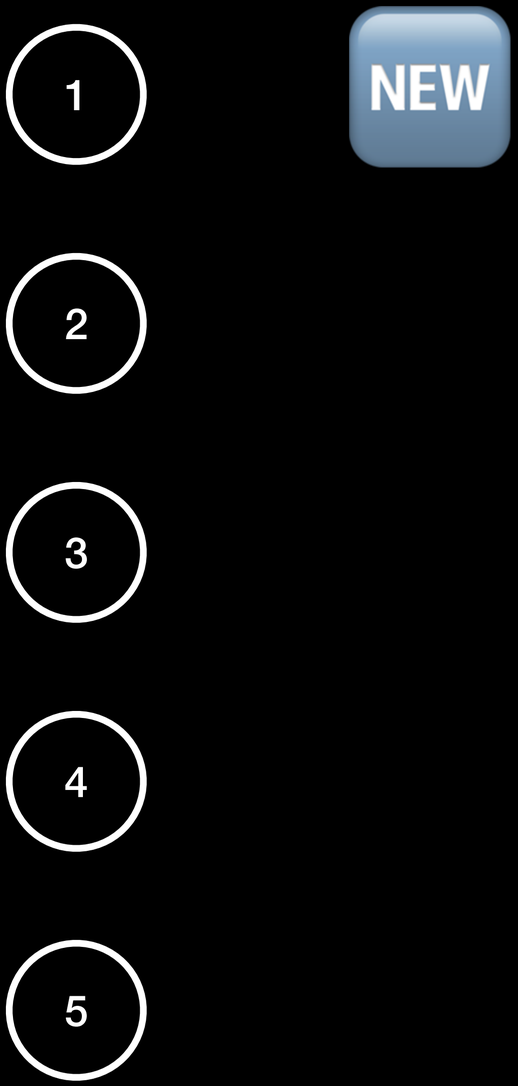
- Branch instructions
- Phi nodes



Branch Structure Identification


```
define i1 @outlined_func({ i64, %type1* } %0, i64 %1, i64 %2) {
1:
  . . .
  br i1 %5, label %5, label %2

2:
  . . .
  br i1 %or.cond, label %3, label %4

3:
  call void @release(%swift.bridge* %4)
  call void @release(%swift.bridge* %7)
  br label %outside_1

4:
  %13 = call i1 @str_cmp(i64 %2, %swift.bridge* %7,
                         i64 %3, %swift.bridge* %4, i8 0)
  call void @release(%swift.bridge* %4)
  call void @release(%swift.bridge* %7)
  br i1 %13, label %outside_1, label %outside_2

5:
  tail call void @release(%swift.bridge* %4)
  br label %outside_2
}
```

Branch Structure Identification

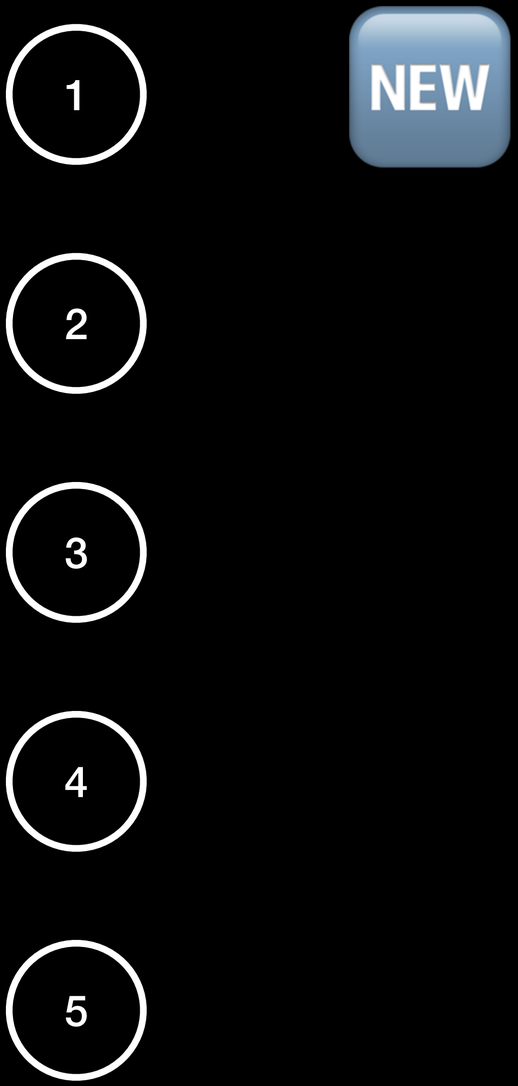



```
define i1 @outlined_func({ i64, %type1* } %0, i64 %1, i64 %2) {
1:
  . . .
  br i1 %5, label %5, label %2

2:
  . . .
  br i1 %or.cond, label %3, label %4

3:
  call void @release(%swift.bridge* %4)
  call void @release(%swift.bridge* %7)
  br label %outside_1

4:
  %13 = call i1 @str_cmp(i64 %2, %swift.bridge* %7,
                         i64 %3, %swift.bridge* %4, i8 0)
  call void @release(%swift.bridge* %4)
  call void @release(%swift.bridge* %7)
  br i1 %13, label %outside_1, label %outside_2

5:
  tail call void @release(%swift.bridge* %4)
  br label %outside_2
}
```

outside_1

outside_2

Branch Structure Identification



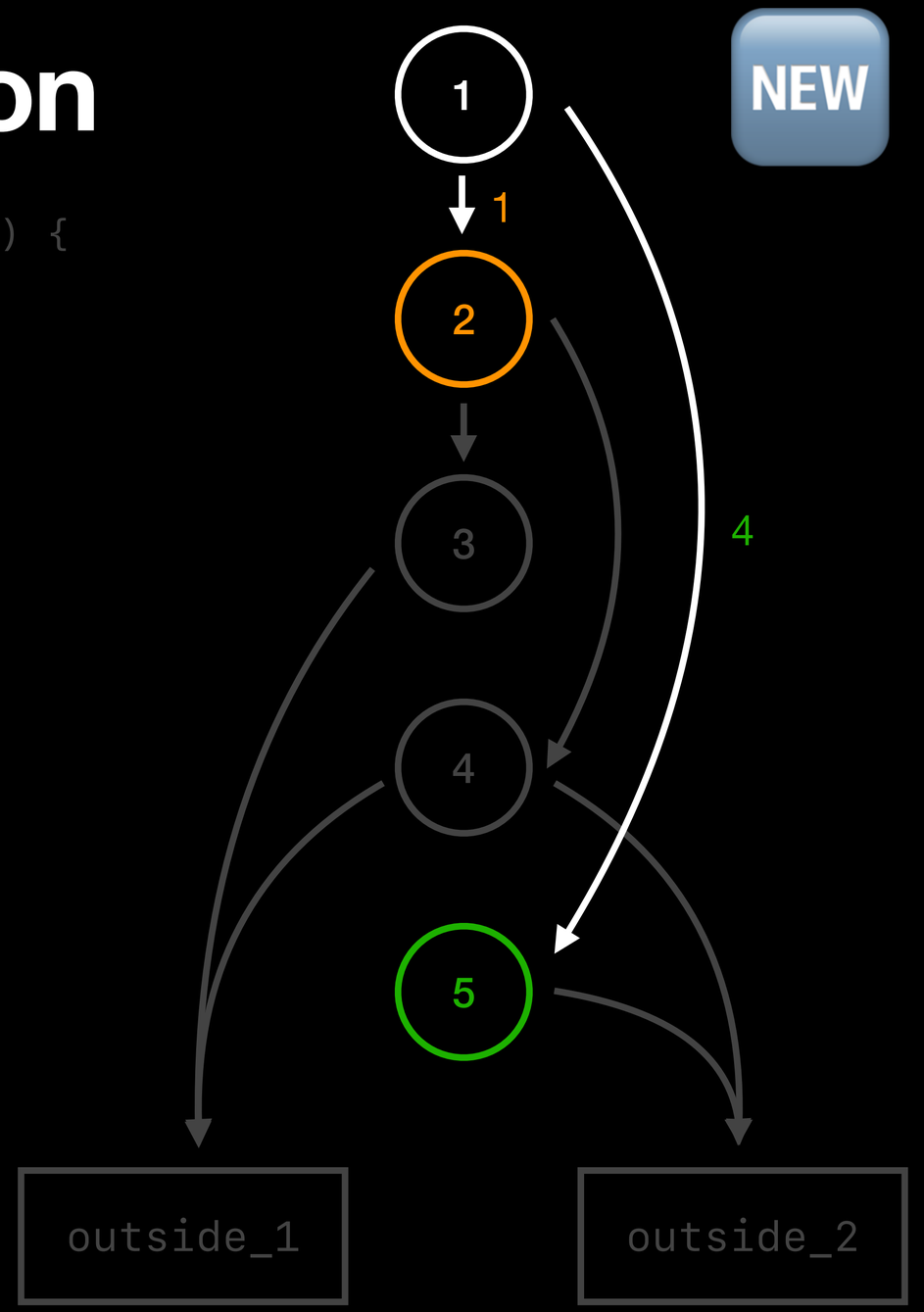

```
define i1 @outlined_func({ i64, %type1* } %0, i64 %1, i64 %2) {
1:
  . . .
  br i1 %5, label %5, label %2
2:
  . . .
  br i1 %or.cond, label %3, label %4
3:
  call void @release(%swift.bridge* %4)
  call void @release(%swift.bridge* %7)
  br label %outside_1
4:
  %13 = call i1 @str_cmp(i64 %2, %swift.bridge* %7,
                         i64 %3, %swift.bridge* %4, i8 0)
  call void @release(%swift.bridge* %4)
  call void @release(%swift.bridge* %7)
  br i1 %13, label %outside_1, label %outside_2
5:
  tail call void @release(%swift.bridge* %4)
  br label %outside_2
}
```

outside_1

outside_2

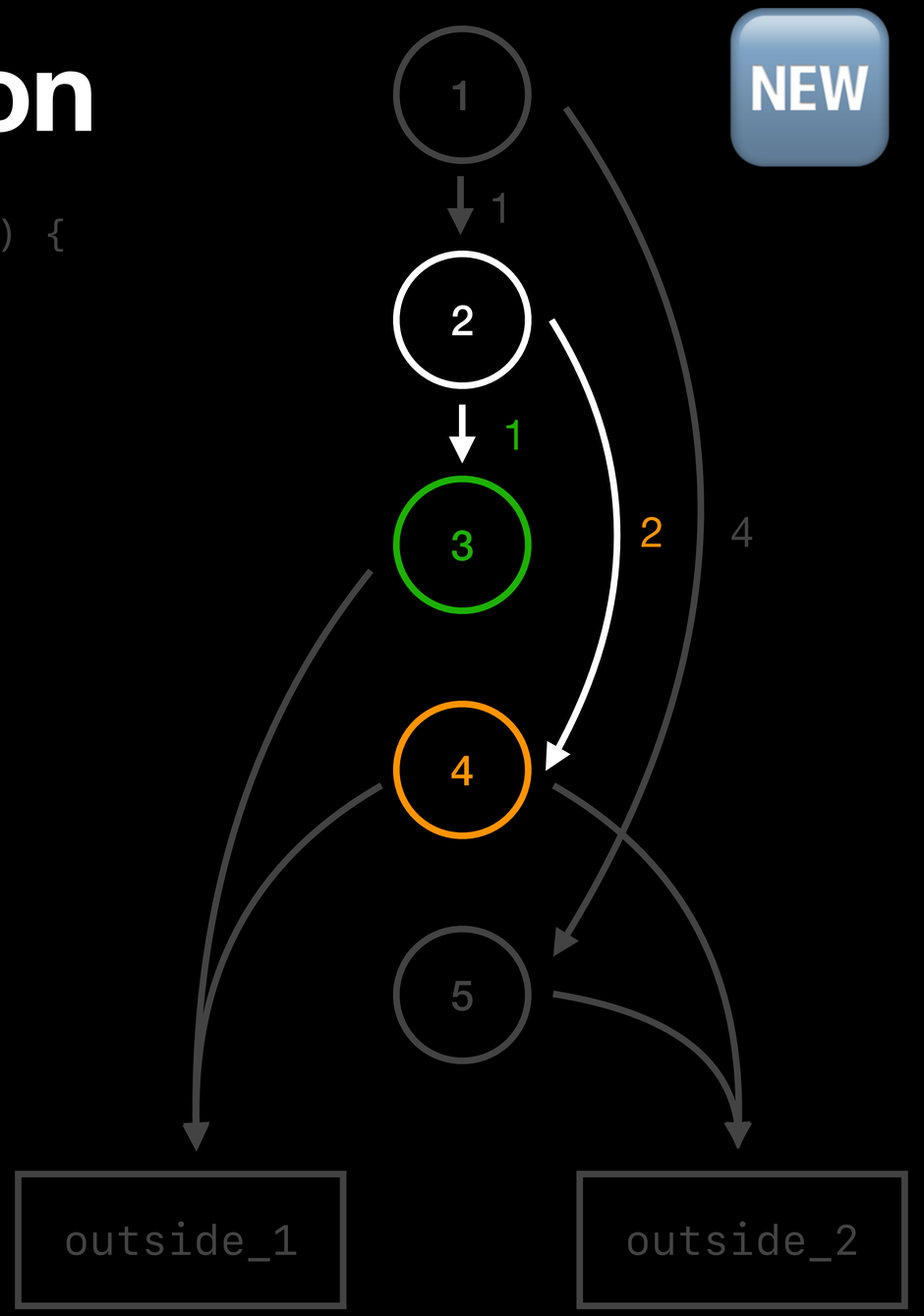
Branch Structure Identification

```
define i1 @outlined_func({ i64, %type1* } %0, i64 %1, i64 %2) {
1:
  . . .
  br i1 %5, label %5, label %2
2:
  . . .
  br i1 %or.cond, label %3, label %4
3:
  call void @release(%swift.bridge* %4)
  call void @release(%swift.bridge* %7)
  br label %outside_1
4:
  %13 = call i1 @str_cmp(i64 %2, %swift.bridge* %7,
                         i64 %3, %swift.bridge* %4, i8 0)
  call void @release(%swift.bridge* %4)
  call void @release(%swift.bridge* %7)
  br i1 %13, label %outside_1, label %outside_2
5:
  tail call void @release(%swift.bridge* %4)
  br label %outside_2
}
```



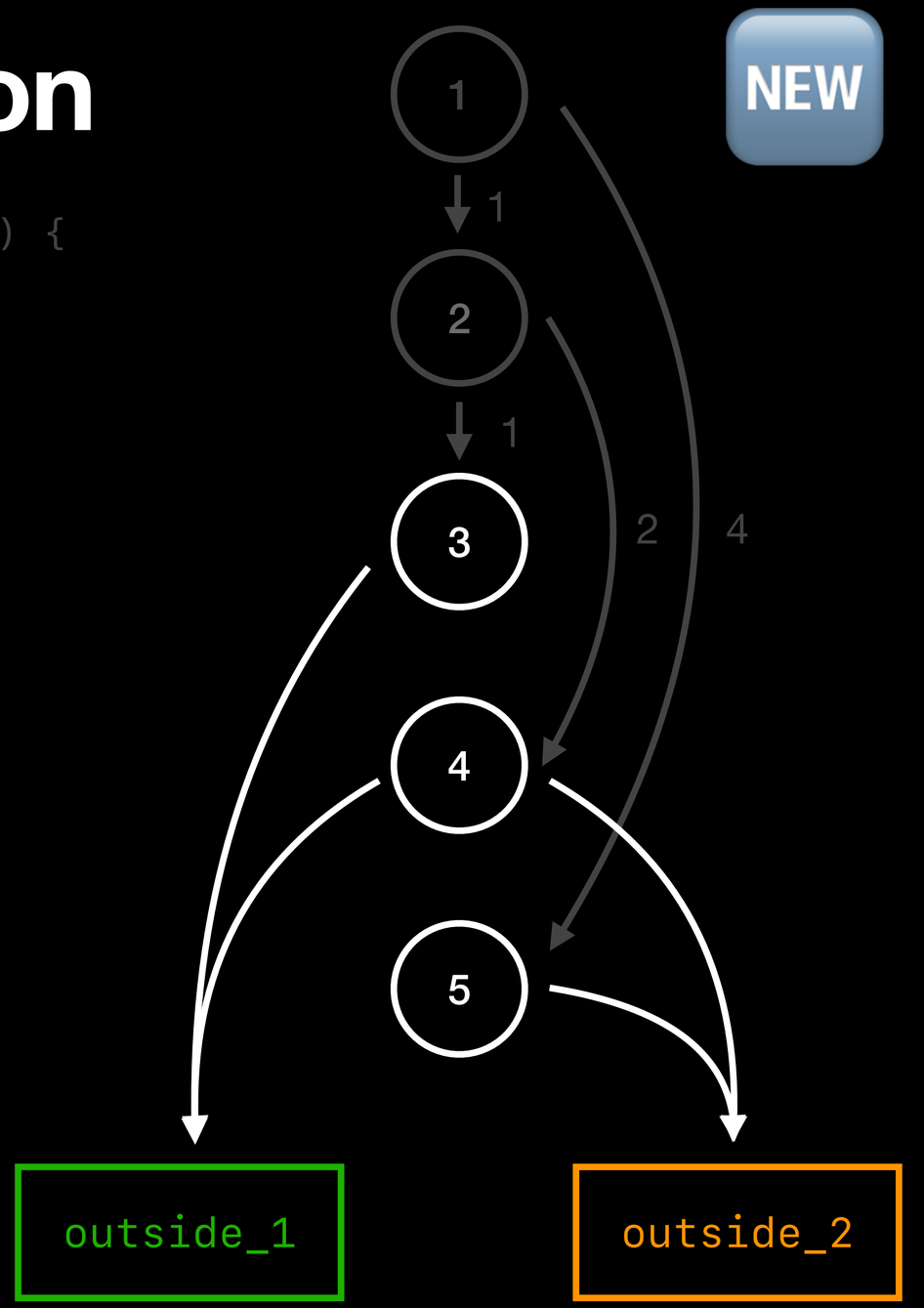
Branch Structure Identification

```
define i1 @outlined_func({ i64, %type1* } %0, i64 %1, i64 %2) {
1:
  . . .
  br i1 %5, label %5, label %2
2:
  . . .
  br i1 %or.cond, label %3, label %4
3:
  call void @release(%swift.bridge* %4)
  call void @release(%swift.bridge* %7)
  br label %outside_1
4:
  %13 = call i1 @str_cmp(i64 %2, %swift.bridge* %7,
                         i64 %3, %swift.bridge* %4, i8 0)
  call void @release(%swift.bridge* %4)
  call void @release(%swift.bridge* %7)
  br i1 %13, label %outside_1, label %outside_2
5:
  tail call void @release(%swift.bridge* %4)
  br label %outside_2
}
```



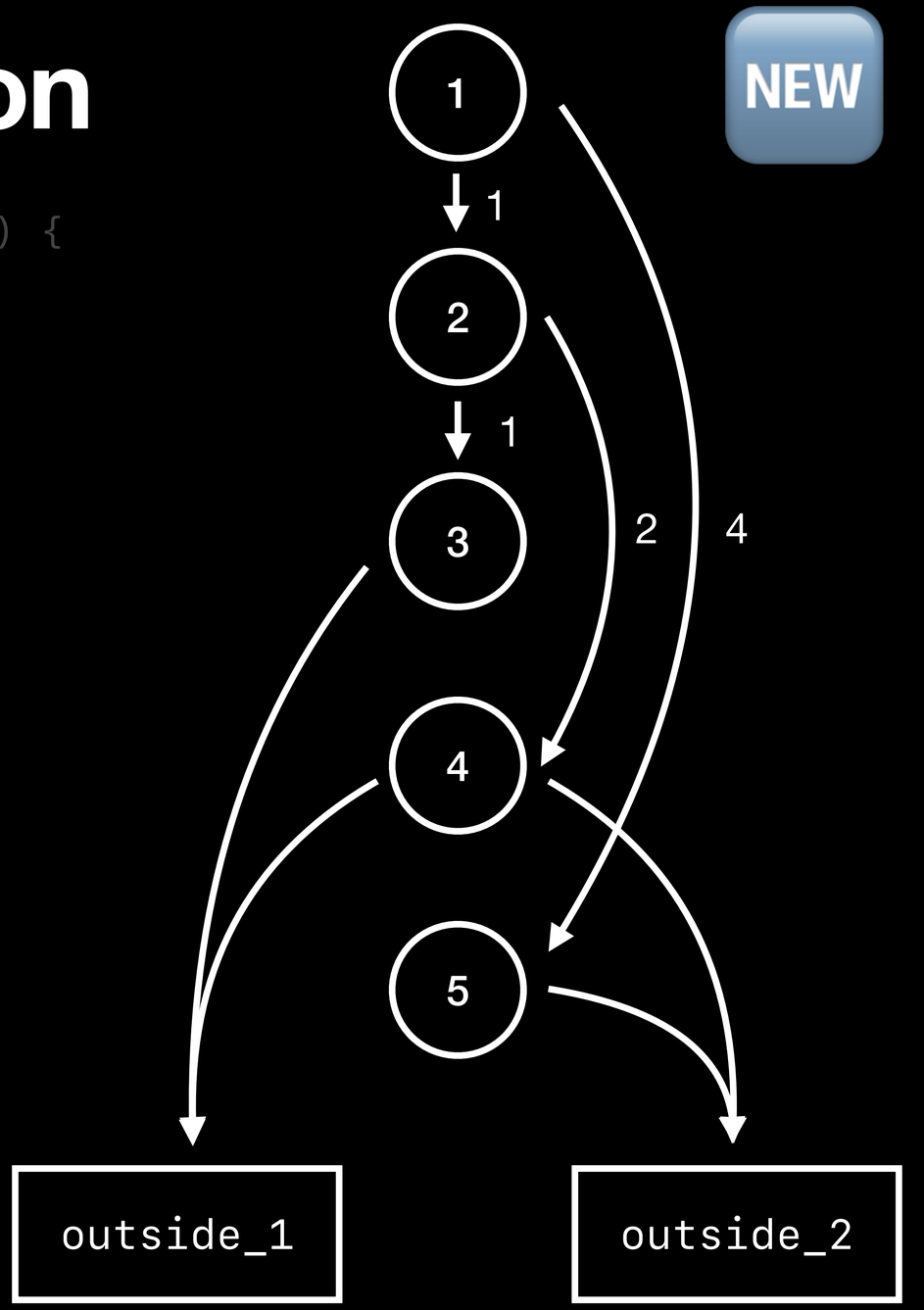
Branch Structure Identification

```
define i1 @outlined_func({ i64, %type1* } %0, i64 %1, i64 %2) {
1:
  . . .
  br i1 %5, label %5, label %2
2:
  . . .
  br i1 %or.cond, label %3, label %4
3:
  call void @release(%swift.bridge* %4)
  call void @release(%swift.bridge* %7)
  br label %outside_1
4:
  %13 = call i1 @str_cmp(i64 %2, %swift.bridge* %7,
                         i64 %3, %swift.bridge* %4, i8 0)
  call void @release(%swift.bridge* %4)
  call void @release(%swift.bridge* %7)
  br i1 %13, label %outside_1, label %outside_2
5:
  tail call void @release(%swift.bridge* %4)
  br label %outside_2
}
```



Branch Structure Identification

```
define i1 @outlined_func({ i64, %type1* } %0, i64 %1, i64 %2) {
1:
  . . .
  br i1 %5, label %5, label %2
2:
  . . .
  br i1 %or.cond, label %3, label %4
3:
  call void @release(%swift.bridge* %4)
  call void @release(%swift.bridge* %7)
  br label %outside_1
4:
  %13 = call i1 @str_cmp(i64 %2, %swift.bridge* %7,
                         i64 %3, %swift.bridge* %4, i8 0)
  call void @release(%swift.bridge* %4)
  call void @release(%swift.bridge* %7)
  br i1 %13, label %outside_1, label %outside_2
5:
  tail call void @release(%swift.bridge* %4)
  br label %outside_2
}
```



Examples of Newly Outlined Code

Range Checks

```
if case .willInsert? = info.items[i].phase {
  info.items[i].phase = .normal
  changed = true
}
```

```
switch info.items[index].phase
```

Range Checks

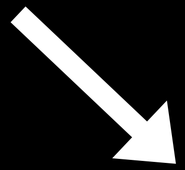
```
if case .willInsert? = info.items[i].phase {
  info.items[i].phase = .normal
  changed = true
}
```

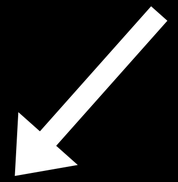
```
switch info.items[index].phase
```

Range Checks

```
if case .willInsert? = info.items[i].phase {
  info.items[i].phase = .normal
  changed = true
}
```

```
switch info.items[index].phase
```





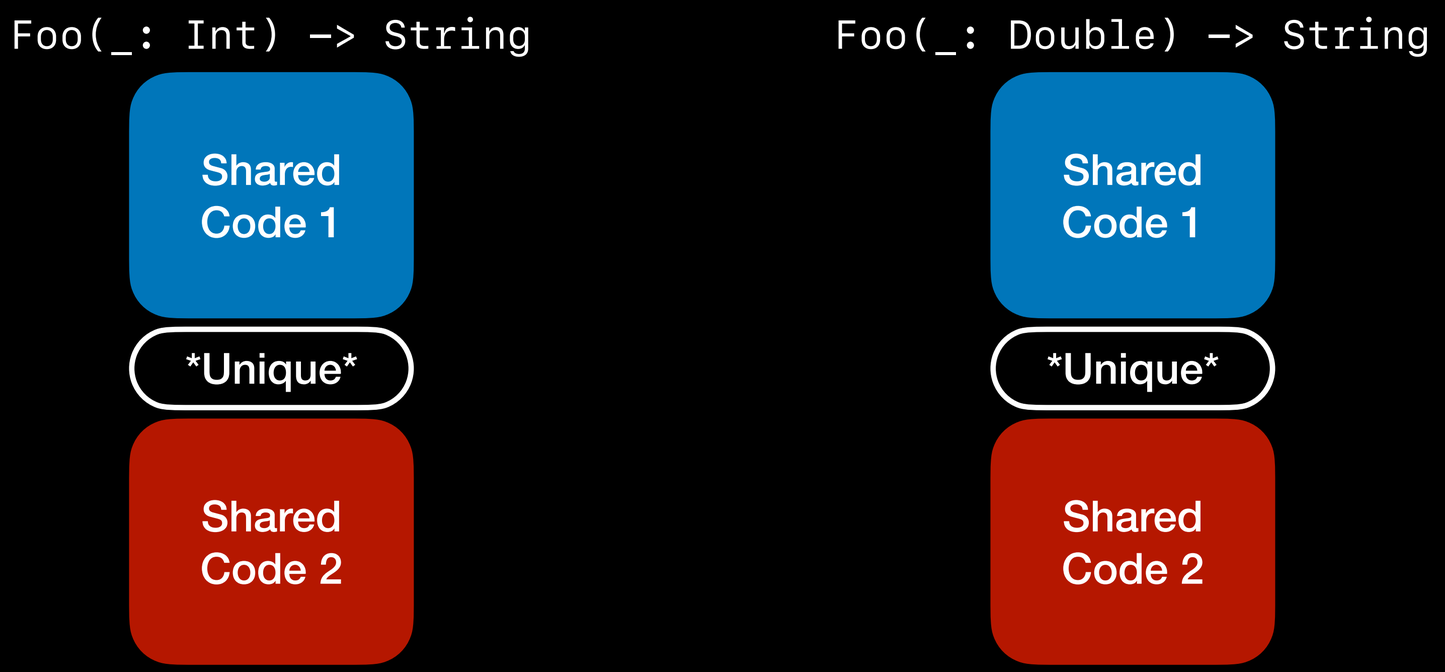
```
  tail call void @“func1”(/*parameters*/)
  br i1 %1, label %8, label %13

8:
  %and1 = and i64 %3, 4611686018427387904
  %cmp1 = icmp ne i64 %9and1, 0
  %cmp2 = icmp slt %swift.bridge* %2, null
  %or1 = or i1 %cmp1, %cmp2
  br i1 %pr1, label %16, label %.exitStub

13:
  %output1 = tail call %objc_object* @“func2”(/*parameters*/)
  %cast1 = bitcast %objc_object* %14 to %some_ty*
  br label %exit

16:
  tail call swiftcc void @“$fatalErrorMessage”(/*parameters*/)
```

Specialized Code Outlining

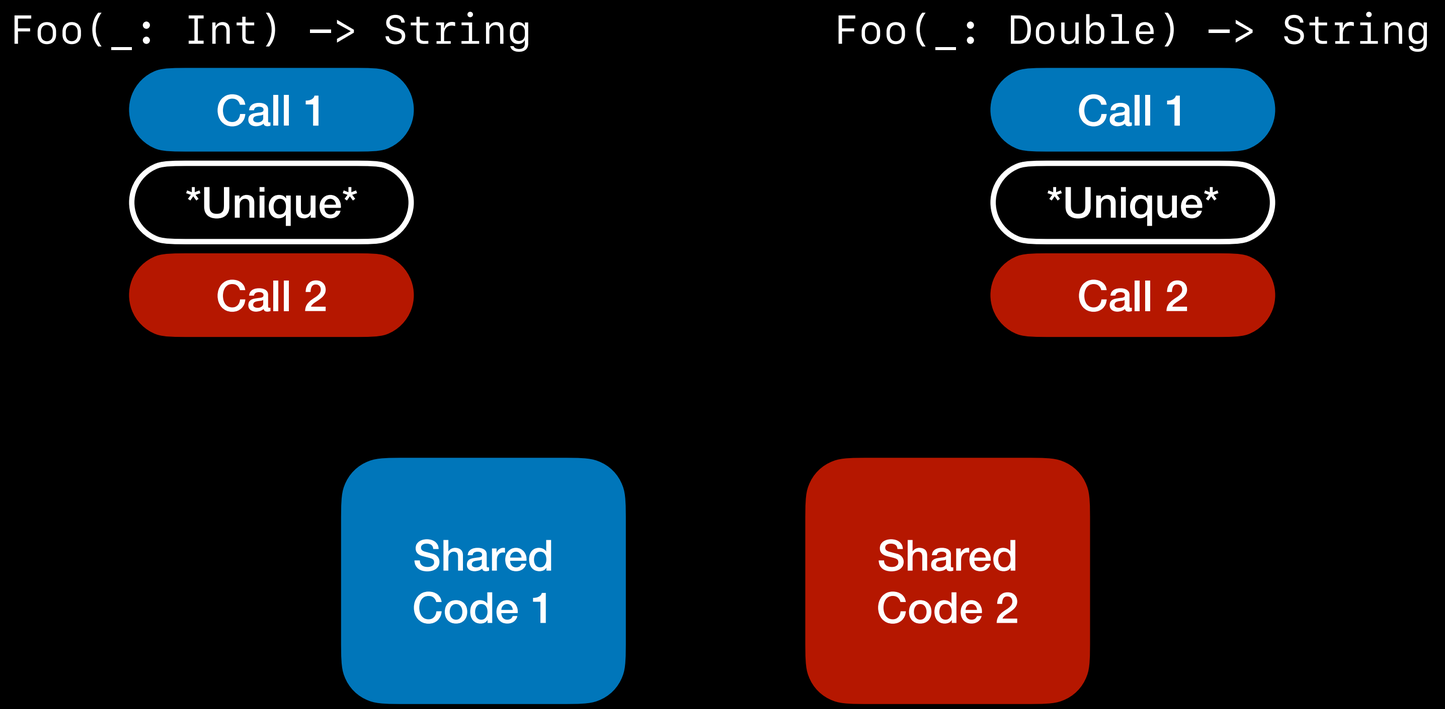
```
Foo(_: Int) -> String
```

```
Foo(_: Double) -> String
```

Specialized Code Outlining



Specialized Code Outlining



IROutliner can avoid unique code that blocks deduplication

Allow General Matching of Calls

- Direct and indirect calls by type signature
- Intrinsic calls by name

Code Size improvements on SwiftUI

swiftui.ll

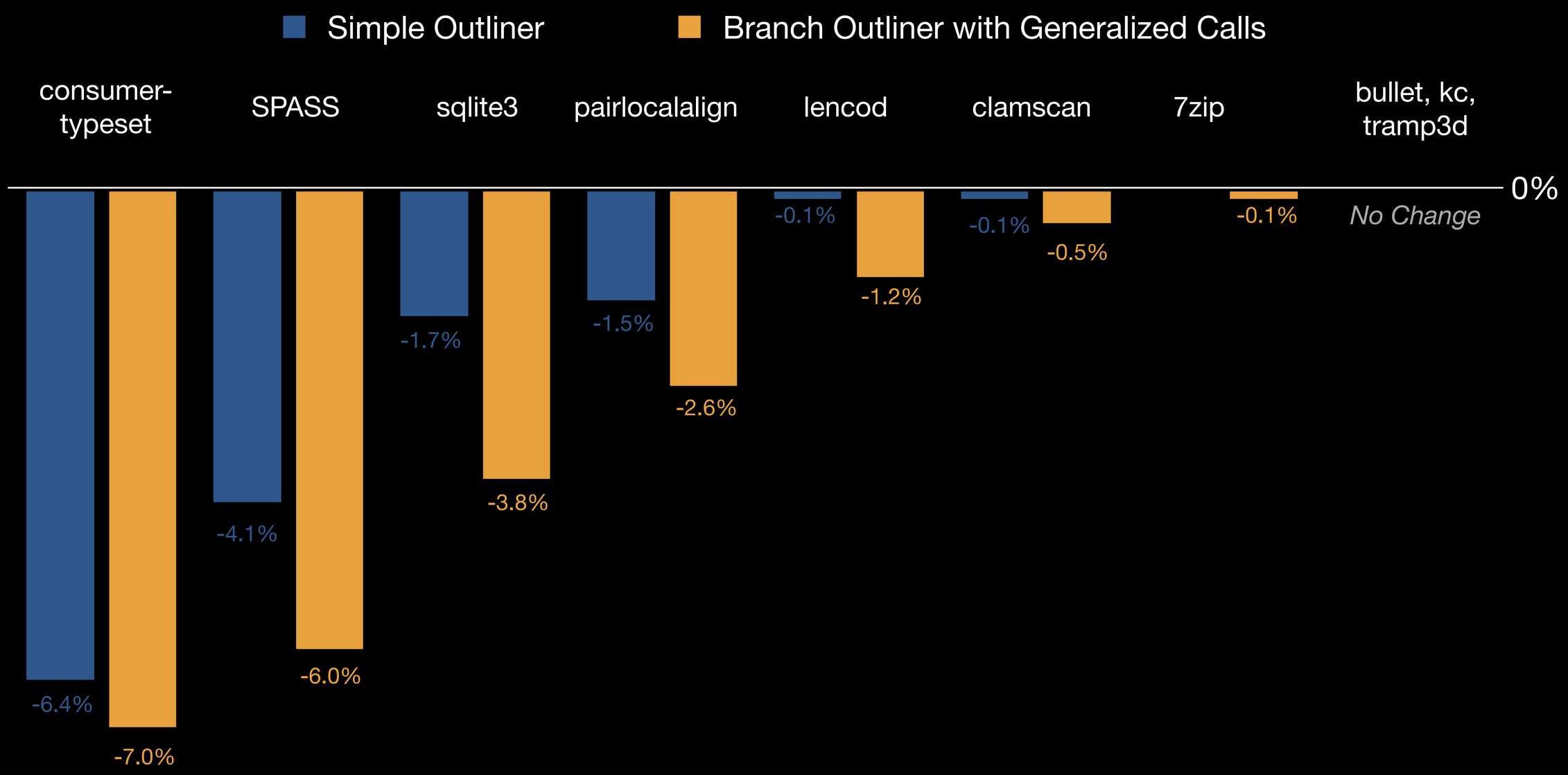
Optimization	Size (Text)	Saving
-Os	14.12 MB	-
Simple IR Outliner	14.11 MB	-0.04%
Branch IR Outliner	13.94 MB	-1.27%

Code Size improvements on SwiftUI

swiftui.ll

Optimization	Size (Text)	Saving
-Os	14.12 MB	-
Simple IR Outliner	14.11 MB	-0.04%
Branch IR Outliner	13.94 MB	-1.27%
Branch+Calls IR Outliner	13.87 MB	-1.90%

CTMark Code Size Reduction (AArch64, -Os)



Future Work

- Placement of Outliner/On by default
- Improve LLVM instruction estimation for code size
- Create a canonical order of operations for unordered instructions

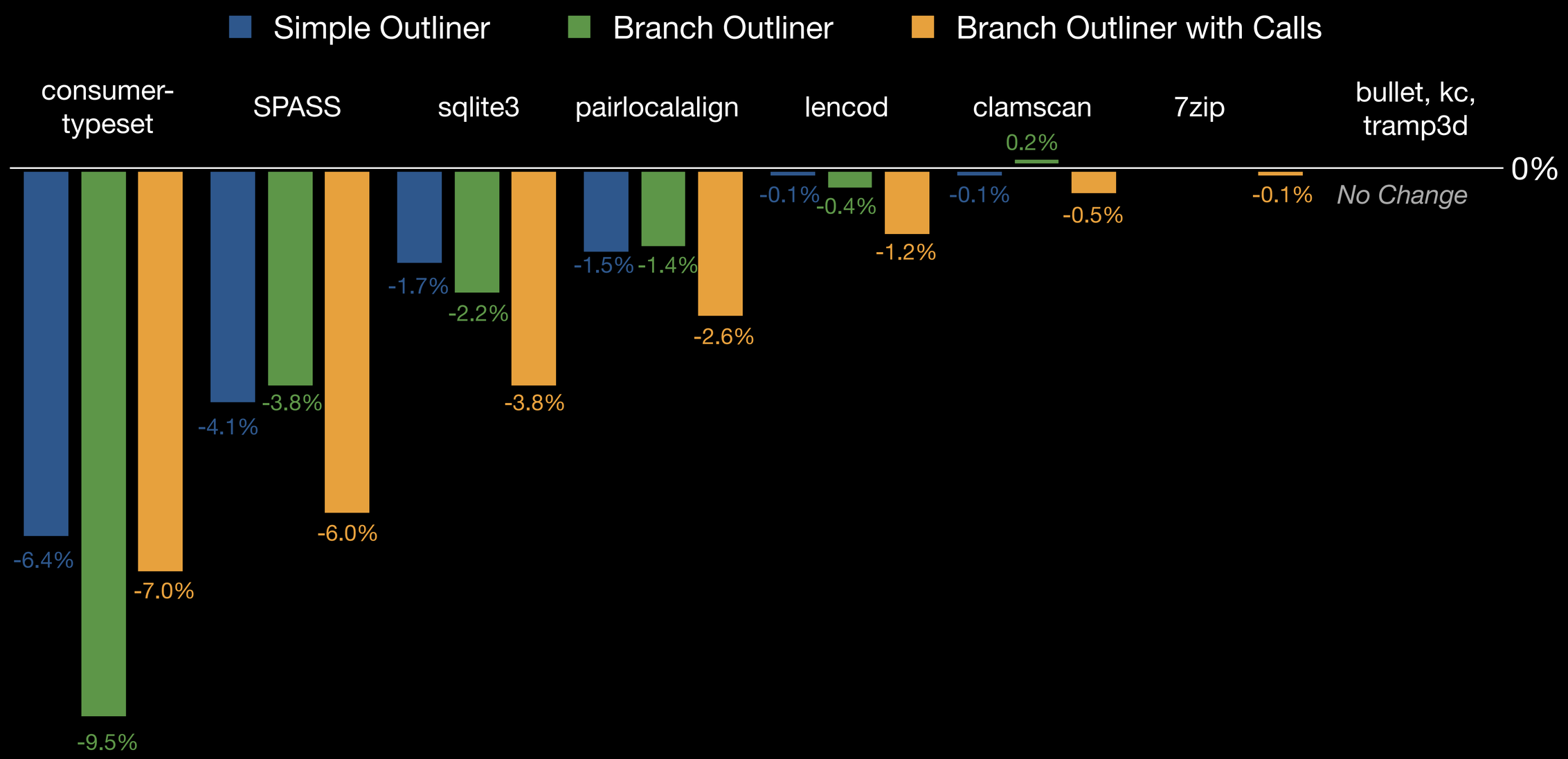
Thank You!

Andrew: andrew.litteken@gmail.com

Jessica: jpaquette@apple.com

Appendix

CTMark Code Size Reduction (AArch64, -Os)



Runtime Changes with Code Size Decreases

arm64, -Os, Initial sizes greater than 100 KB

	Avg Text Size Change	Avg Exec Time Change
LLVM Test Suite SingleSource, MultiSource	-1.14% [-3.55%, 0.51%]	0.39% [-1.00%, 3.63%]
SPECCFP (2000, 2006, 2007rate/speed) SPECCINT (95, 2000, 2006, 2007rate/speed)	-0.62% [-2.84%, 2.46%]	3.2% [-1.24%, 10.33%]