Compositional Decompilation

Robin Eklind

University of Portsmouth

Problem

Recover high-level control flow primitives (e.g. if-statements, for-loop, ...) from low-level LLVM IR (platform-independent assembly).

Method

Identify control flow primitives using subgraph isomorphism search algorithms [1].

```
if A {
    B
}
C
```

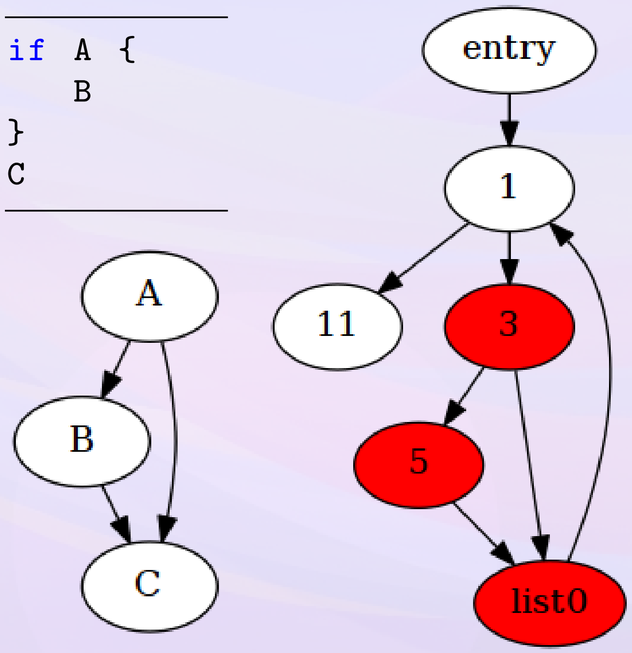

Figure 1: Pseudo-code and graph representation of an if-statement (left) and the control flow graph of `main` (right).

Results

```
define i32 @main(i32 %argc, i8** %argv) {
  br label %1
; <label>:1
  %i.0 = phi i32 [ 0, %0 ], [ %10, %9 ]
  %x.0 = phi i32 [ 0, %0 ], [ %x.1, %9 ]
  %2 = icmp slt i32 %i.0, 10
  br i1 %2, label %3, label %11
; <label>:3
  %4 = icmp slt i32 %x.0, 100
  br i1 %4, label %5, label %8
; <label>:5
  %6 = mul nsw i32 3, %i.0
  %7 = add nsw i32 %x.0, %6
  br label %8
; <label>:8
  %x.1 = phi i32 [ %7, %5 ], [ %x.0, %3 ]
  br label %9
; <label>:9
  %10 = add nsw i32 %i.0, 1
  br label %1
; <label>:11
  ret i32 %x.0
}
```

```
package main

import "os"

func main() {
  x := 0
  for i := 0; i < 10; i++ {
    if x < 100 {
      x += 3 * i
    }
  }
  os.Exit(x)
}
```

Figure 2: LLVM IR input (left) and Go output (right).

Source Code

`https://github.com/decomp`

References

[1] S. Moll, *Decompilation of LLVM IR*. BSc thesis, Saarland University, 2011.