

# Using Static Analysis to Track Inter-process Dependencies

Mike Sepowitz

Team Lead in Deployment Solutions

Bloomberg

**Bloomberg**

Engineering



# Roadmap

Introduction

Important  
Questions

Exploration

Primer

Attempt  
#1

Attempt  
#2

Takeaways

# Introduction

# Microservices

(nanoservices?)

# Documentation?



**Static analysis!**



# Outcomes

1. Dependencies of a single service
  - Highlight effects of a code change
2. Inter-service dependencies across the entire system
  - Visualize system graph
  - Detect cycles
  - Overlay traffic data on static graph

# The Plan

1. Find “interesting” function calls
2. Call “destinations” = dependencies of the service
3. Service dependencies = edges of the system graph



# Important Questions

# What are “interesting” calls?

**Network:** `net.Dial`, `net/http.Get`, `(*net/http.Client).Get`

**Higher level:** `database/sql.OpenDB`

`google.golang.org/grpc.Dial`

`github.com/go-redis/redis.NewClient`

`github.com/streadway/amqp.Dial` (RabbitMQ)

`github.com/Shopify/sarama.NewAsyncProducer` (Kafka)

**Processes:** `(*os/exec.Cmd).Run`, `os/exec.FindProcess`, `os.Pipe`

**Filesystem:** `os.Create`, `os.Open`, `os.Mkdir`

# Who are we calling?

```
http.Get("https://myservice.example.com/path/to/resource")
```

```
http.Get(myserviceURL)
```

```
url := os.Getenv("MYSERVICE_URL")
```

```
...
```

```
http.Get(url)
```

```
http.Get(app.Config.MyService.URL)
```

# Who are we calling?

```
http.Get( ... )
```

# Who are we calling?

```
http.Get( ... ) // ->myservice
```

```
// ->myservice
```

```
if resp, err := http.Get(myserviceURL); err != nil {  
    ...  
}
```

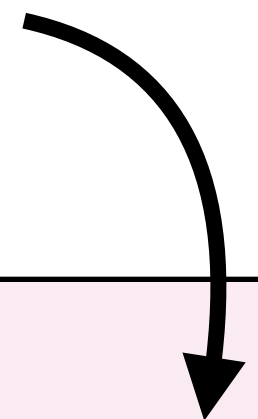
# Which call do we mark?

# Direct call

```
func main() {  
    http.Get(golangDotOrgURL)  
}
```

# Specific helper

```
func main() {  
    getGolangDotOrg()  
}
```



```
func getGolangDotOrg() (*http.Response, error) {  
    return http.Get(golangDotOrgURL)  
}
```



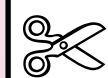
# Specific & generic helpers

```
func main() {  
    getGolangDotOrg()  
}
```

```
func getGolangDotOrg() (*http.Response, error) {  
    return retryablehttp.NewClient().Get(golangDotOrgURL)  
}
```

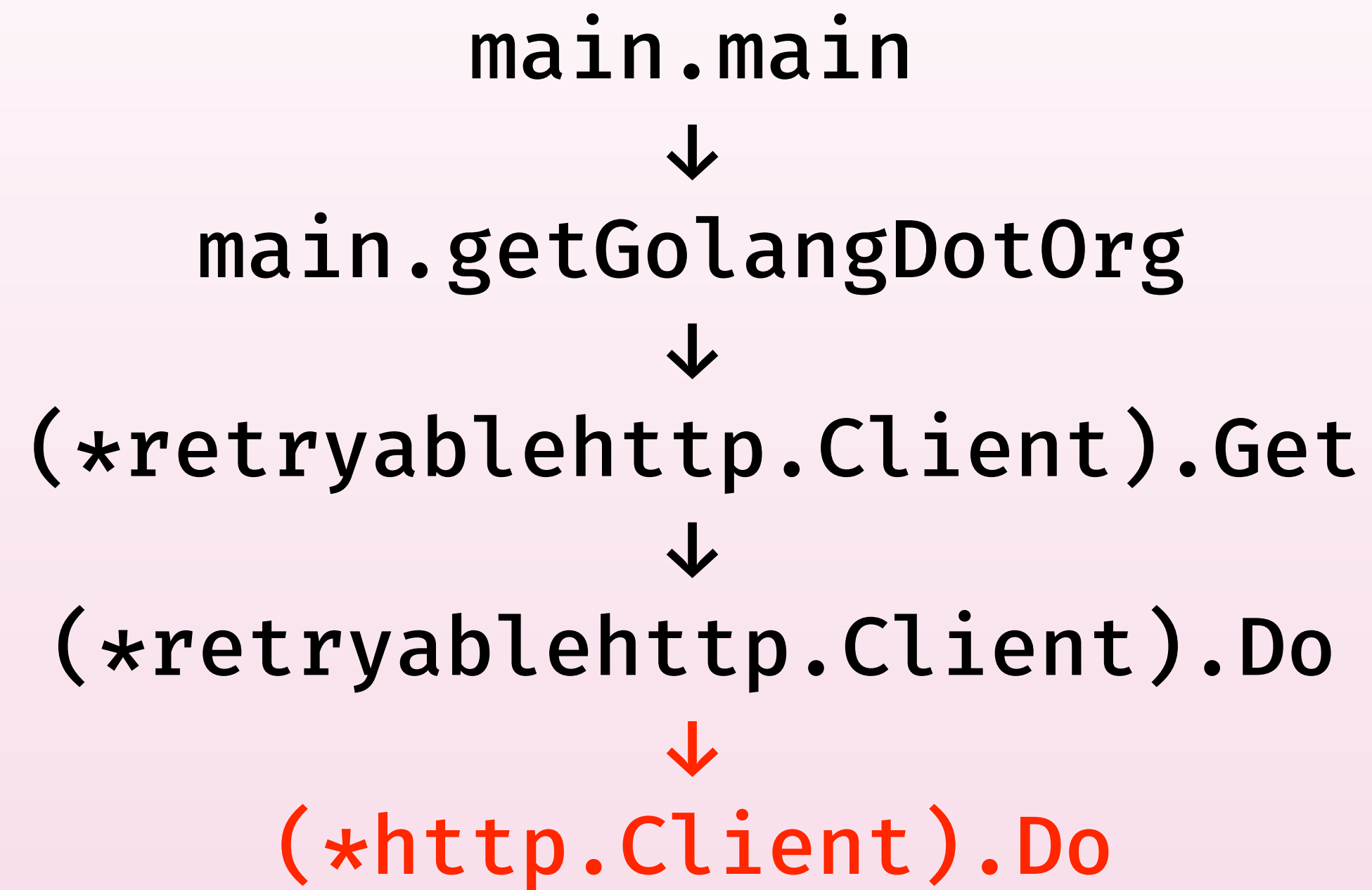
[github.com/hashicorp/go-retryablehttp](https://github.com/hashicorp/go-retryablehttp)

```
func (c *Client) Get(url string) (*http.Response, error) {  
    req, err := NewRequest("GET", url, nil)  
    ...  
    return c.Do(req)  
}
```



```
-----  
func (c *Client) Do(req *Request) (*http.Response, error) {  
    ...  
    resp, err = c.HTTPClient.Do(req.Request)  
    ...  
}
```

# Call chain



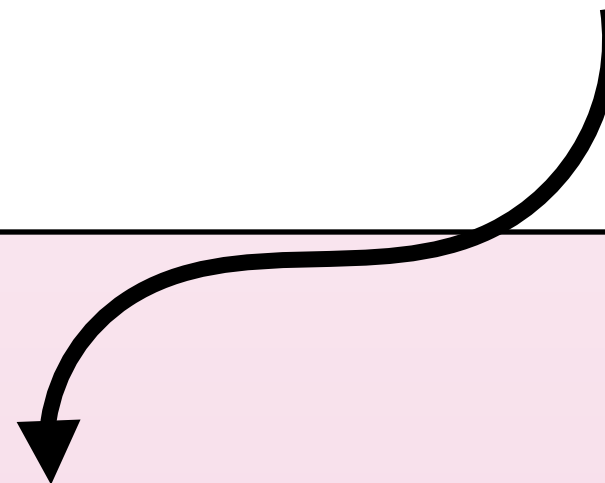
# Call chain

main.main



main.getGolangDotOrg

```
func getGolangDotOrg() (*http.Response, error) {  
    return retryablehttp.NewClient().Get(golangDotOrgURL)  
}
```



(\*retryablehttp.Client).Get



(\*retryablehttp.Client).Do



(\*http.Client).Do

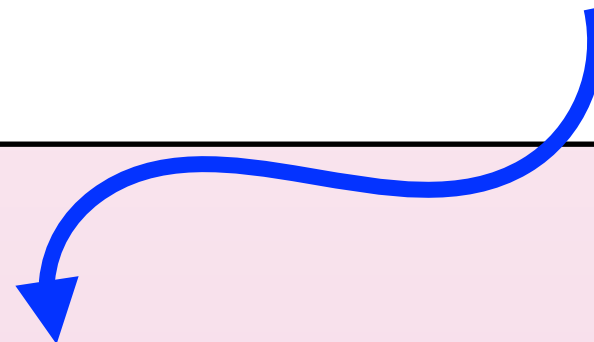
# Call chain

main.main



main.getGolangDotOrg

```
func getGolangDotOrg() (*http.Response, error) {  
    // ->golang.org  
    return retryablehttp.NewClient().Get(golangDotOrgURL)  
}
```



(\*retryablehttp.Client).Get

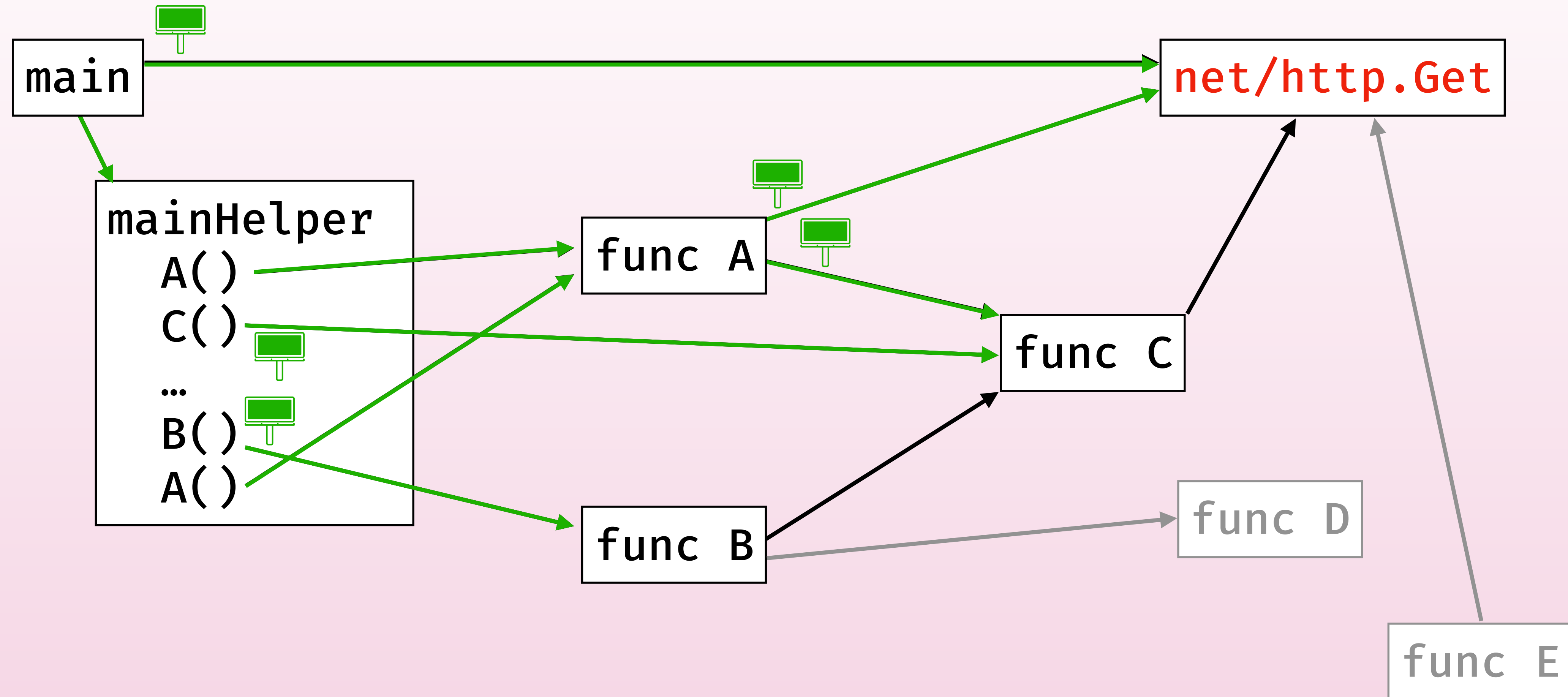


(\*retryablehttp.Client).Do



(\*http.Client).Do

# Marked paths



# Exploration

# GoDoc search: "call graph"

[golang.org/x/tools/go/callgraph](https://golang.org/x/tools/go/callgraph)

[golang.org/x/tools/go/callgraph/cha](https://golang.org/x/tools/go/callgraph/cha)

[golang.org/x/tools/go/callgraph/rta](https://golang.org/x/tools/go/callgraph/rta)

[golang.org/x/tools/go/callgraph/static](https://golang.org/x/tools/go/callgraph/static)

# Call graphs

callgraph/static

```
func CallGraph(prog *ssa.Program) *callgraph.Graph
```

callgraph/cha

```
func CallGraph(prog *ssa.Program) *callgraph.Graph
```

callgraph/rta

```
func Analyze(roots *[]ssa.Function, buildCallGraph bool) Result
```

pointer

```
func Analyze(config *Config) (result *Result, err error)
```

```
type Config struct {  
    Mains []*ssa.Package
```

```
    ...  
}
```



# golang.org/x/tools/go/...

analysis

analysis/analysistest

analysis/cmd/vet

analysis/multichecker

analysis/passes/asmdecl

analysis/passes/assign

analysis/passes/atomic

analysis/passes/atomicalign

analysis/passes/bools

analysis/passes/buildssa

analysis/passes/buildtag

analysis/passes/cgocheck

analysis/passes/cgoconv

analysis/passes/cgoescape

analysis/passes/cgofix

analysis/passes/cgoimage

analysis/passes/cgoimport

analysis/passes/cgoisys

analysis/passes/cgojson

analysis/passes/cgokey

analysis/passes/cgokeygen

analysis/passes/cgokeygen2

analysis/passes/cgokeygen3

analysis/passes/cgokeygen4

analysis/passes/cgokeygen5

analysis/passes/cgokeygen6

analysis/passes/cgokeygen7

analysis/passes/lostcancel/cmd/lostcancel

analysis/passes/nilfunc

analysis/passes/nilness

analysis/passes/nilness/cmd/nilness

analysis/passes/pkgfact

analysis/passes/printf

analysis/passes/shadow

analysis/passes/shadow/cmd/shadow

analysis/passes/shift

analysis/passes/shift/cmd/shift

analysis/passes/shift/cmd/shift2

analysis/passes/shift/cmd/shift3

analysis/passes/shift/cmd/shift4

analysis/passes/shift/cmd/shift5

analysis/passes/shift/cmd/shift6

analysis/passes/shift/cmd/shift7

analysis/passes/shift/cmd/shift8

analysis/passes/shift/cmd/shift9

analysis/passes/shift/cmd/shift10

analysis/passes/shift/cmd/shift11

analysis/passes/shift/cmd/shift12

analysis/passes/shift/cmd/shift13

analysis/passes/shift/cmd/shift14

analysis/passes/shift/cmd/shift15

analysis/passes/shift/cmd/shift16

analysis/passes/shift/cmd/shift17

analysis/passes/shift/cmd/shift18

ast/astutil

ast/inspector

buildutil

callgraph

callgraph/cha

callgraph/rta

callgraph/static

cfg

expect

gccgoexportdata

gcexportdata

loader

packages

packages/gopackages

packages/packageest

pointer

ssa

ssa/interp

ssa/ssautl

types/objectpath

types/typeutil

vcs

analysis

# golang.org/x/tools/go/analysis

"The analysis package defines the interface between a modular static analysis and an analysis driver program."

Pass: running an Analyzer on a single package

Reusable passes:

- `analysis/passes/inspect` → `*inspector.Inspector`
- `analysis/passes/ctrlflow` → `*cfg.CFG` (basically)
- `analysis/passes/buildssa` → `*ssa.Package, []*ssa.Function`

# The Plan

For each package Pass:

1. Grab the SSA result from the `buildssa` Pass.
2. Build the call graph using `rta.Analyze`.
3. Traverse the call graph, marking paths that lead to “interesting” calls.
4. Report unmarked paths as linter errors.
5. Report destination markers as dependency data.

# Primer

# Outline

go/

token

ast

parser

[golang.org/x/tools/go/](https://golang.org/x/tools/go/)

packages

ssa

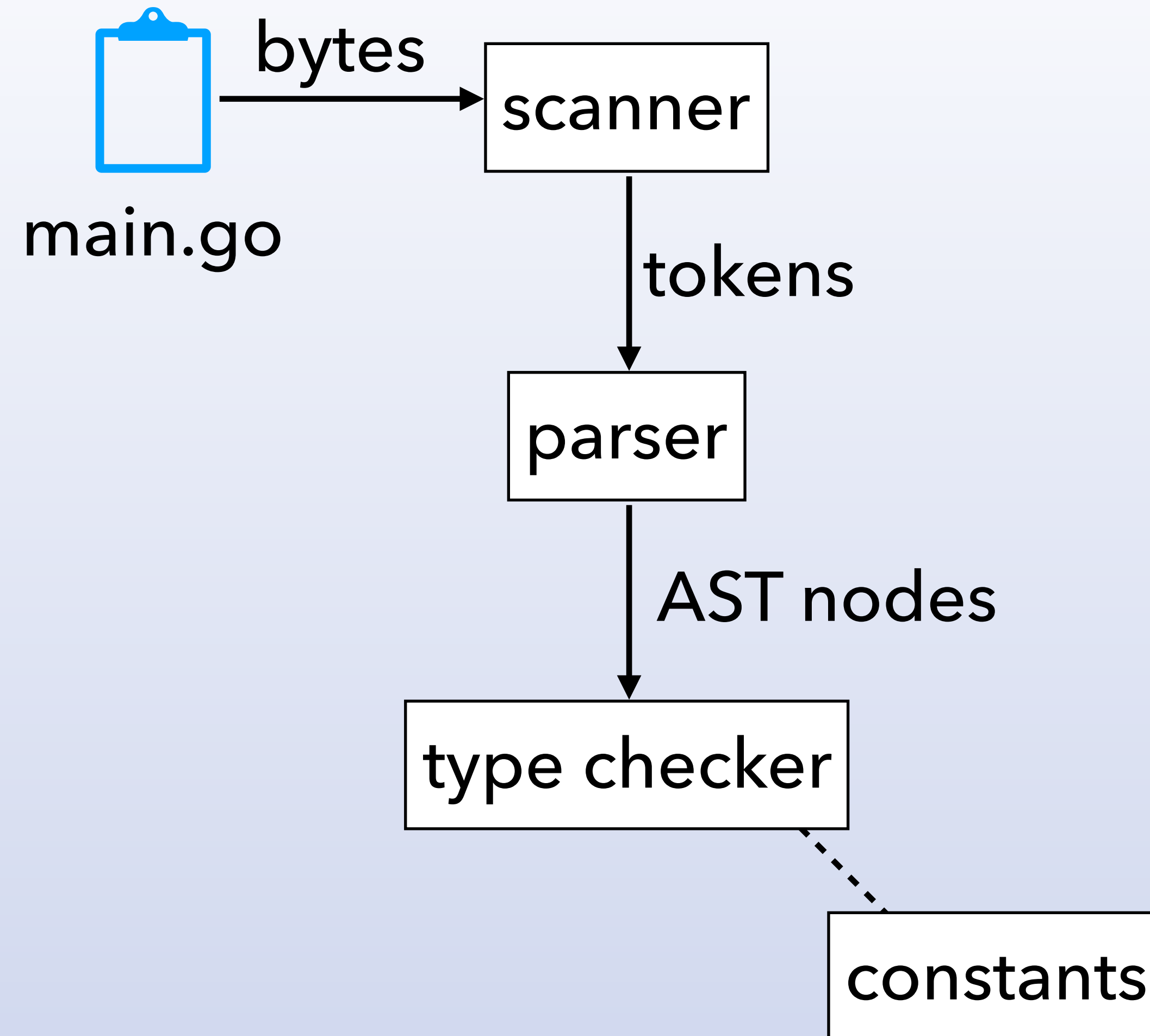
callgraph

analysis

# go/...

types  
constant  
parser  
ast  
scanner  
token

<https://golang.org/s/types-tutorial>



# go/token: FileSets

```
type Pos int
```

```
type Position struct {  
    Filename string  
    Offset   int  
    Line     int  
    Column   int  
}
```

FileSet

```
File  
name "a.go"  
base 1  
size 98
```

```
File  
name "b.go"  
base 100  
size 24
```

```
File  
name "c.go"  
base 125  
size 50
```

```
pos := node.Pos()  
fmt.Println(pass.Fset.Position(pos))
```

# ASTs and parsing

## Abstract Syntax Tree

```
func main() {  
    const source = `  
        package main  
  
        func main() {  
            answer := 42  
        }`  
  
    fset := token.NewFileSet()  
    file, _ := parser.ParseFile(fset, "", source, parser.AllErrors)  
    ast.Print(fset, file)  
}
```



# AST example

```
func main() {  
    answer := 42  
    ...  
}
```

```
FuncDecl {  
  . Name: Ident { Name: "main" }  
  . Type: FuncType { Params: FieldList }  
  . Body: BlockStmt {  
    . List: []Stmt {  
      . . 0: AssignStmt {  
        . . . Lhs: []Expr {  
          . . . . 0: Ident { Name: "answer" }  
          . . . . }  
        . . . Tok: " :="   
        . . . Rhs: []Expr {  
          . . . . 0: BasicLit {  
            . . . . . Kind: INT  
            . . . . . Value: "42"  
          }  
        }  
      }  
    }  
    ...  
  }  
}
```

# go/ast

Package  
File  
Scope  
Object

## Interfaces:

Node  
Decl  
Spec  
Stmt  
Expr

ArrayType  
AssignStmt  
BadDecl  
BadExpr  
BadStmt  
BasicLit  
BinaryExpr  
BlockStmt  
BranchStmt  
CallExpr  
CaseClause  
ChanDir  
ChanType  
CommClause  
Comment  
CommentGroup  
CommentMap  
CompositeLit  
DeclStmt  
DeferStmt  
Ellipsis

EmptyStmt  
ExprStmt  
Field  
FieldFilter  
FieldList  
ForStmt  
FuncDecl  
FuncLit  
FuncType  
GenDecl  
GoStmt  
Ident  
IfStmt  
ImportSpec  
IncDecStmt  
IndexExpr  
InterfaceType  
KeyValueExpr  
LabeledStmt  
MapType  
ParenExpr

RangeStmt  
ReturnStmt  
SelectStmt  
SelectorExpr  
SendStmt  
SliceExpr  
StarExpr  
StructType  
SwitchStmt  
TypeAssertExpr  
TypeSpec  
TypeSwitchStmt  
UnaryExpr  
ValueSpec

<https://golang.org/ref/spec>

# go/parser

```
func ParseExpr(x string) (ast.Expr, error)
```

```
func ParseFile(  
    fset      *token.FileSet,  
    filename  string,  
    src       interface{}, // string, []byte, io.Reader  
    mode      Mode,         // e.g. ParseComments  
) (*ast.File, error)
```

```
func ParseExprFrom(same as ParseFile) (ast.Expr, error)
```

```
func ParseDir(...) (map[string]*ast.Package, error)
```

# golang.org/x/tools/go/packages

Replaces `golang.org/x/tools/go/loader`.

```
type Config struct {  
    Mode LoadMode // e.g. NeedFiles | NeedSyntax  
}
```

```
func Load(cfg *Config, patterns ...string) ([]*Package, error)
```

# golang.org/x/tools/go/ssa

[https://en.wikipedia.org/wiki/Static\\_single\\_assignment\\_form](https://en.wikipedia.org/wiki/Static_single_assignment_form)

"THIS INTERFACE IS EXPERIMENTAL AND IS LIKELY TO CHANGE."

The primary interfaces of this package are:

- Member
- Value
- Instruction
- Node (a Value, an Instruction, or both)

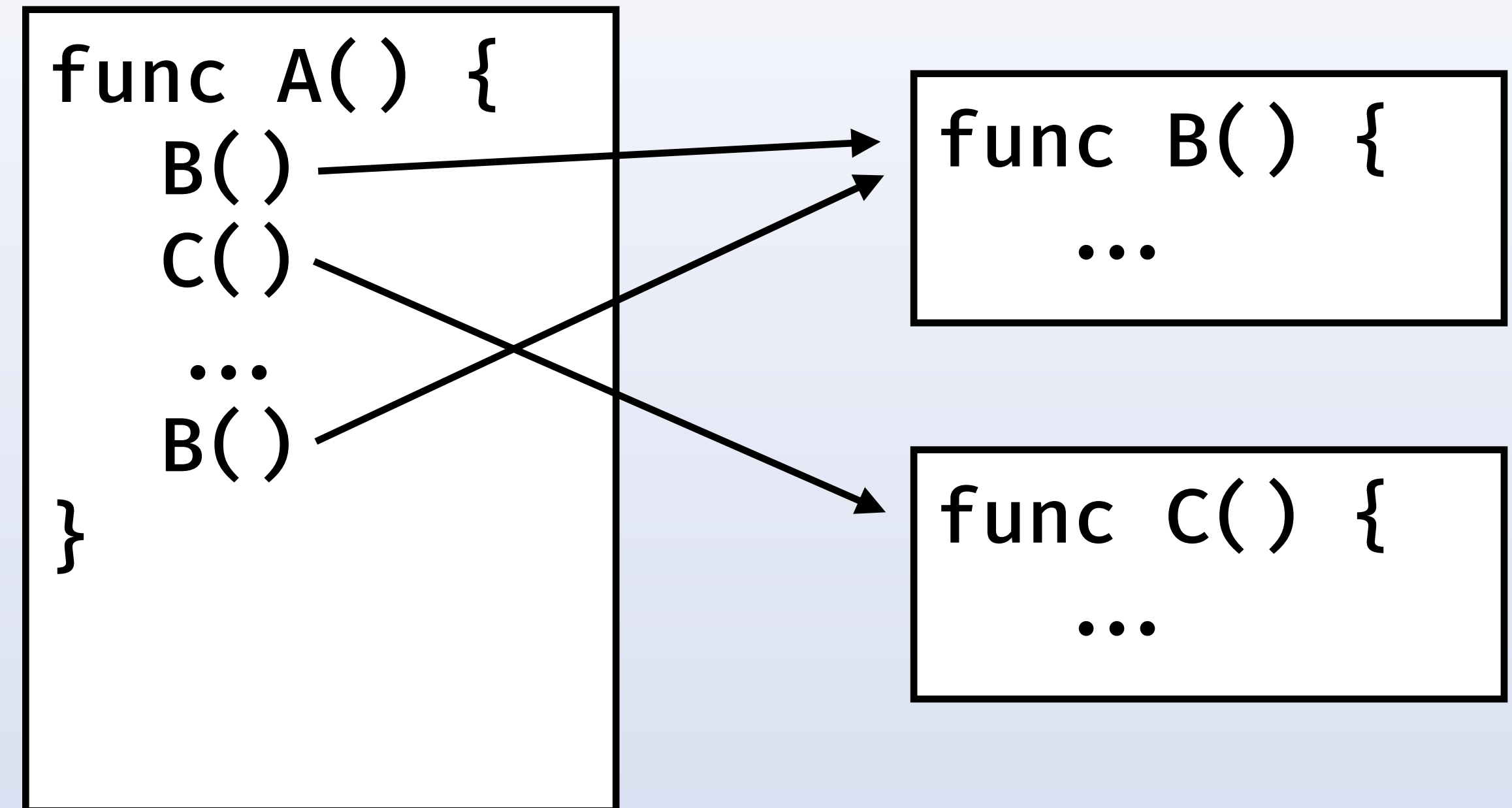
ssa/ssautil: builder and helper functions

# golang.org/x/tools/go/callgraph

```
type Graph struct {  
    Root *Node  
    Nodes map[*ssa.Function]*Node  
}
```

```
type Node struct {  
    Func *ssa.Function  
    ID    int  
    In    []*Edge  
    Out   []*Edge  
}
```

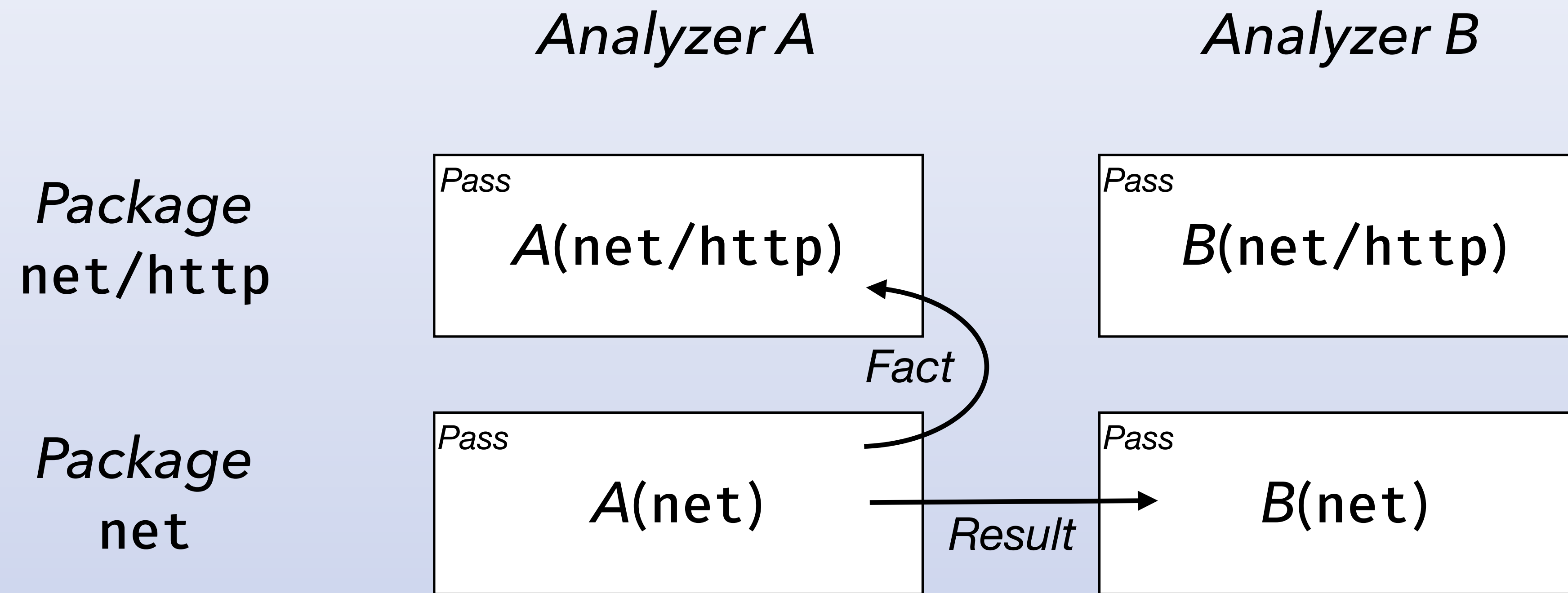
```
type Edge struct {  
    Caller *Node  
    Site   ssa.CallInstruction  
    Callee *Node  
}
```



```
go get golang.org/x/tools/cmd/callgraph
```

# golang.org/x/tools/go/analysis

"The analysis package defines the interface between a modular static analysis and an analysis driver program."



# Attempt #1



# The Plan

For each package Pass,

1. Grab the SSA result from the `buildssa` Pass.
2. Build the call graph using `rta.Analyze`.
3. Traverse the call graph, recording Facts to mark paths that lead to "interesting" calls.
4. Report unmarked paths as linter errors.
5. Report destination markers as dependency data.

# Bumps in the code

No Facts,

- no dependency-ordered tree of analyzer passes [[code](#)]
- no syntax analysis of dependent packages [code [1](#) & [2](#)]

`analysis/passes/buildssa`

- not “modular” – each pass builds a new `ssa.Program` [[code](#)]
- uses `ssa.BuilderMode(0)` [[code](#)]

# Point of no return

```
package main
```

```
import (  
    "example.org/lintandreport"  
    "golang.org/x/tools/go/analysis/singlechecker"  
)
```

```
func main() {  
    singlechecker.Main(lintandreport.Analyzer) // calls os.Exit  
    lintandreport.Analyzer.ReportResults()  
}
```

# Workarounds

1. `if pass.Pkg.Name == "main" { ...`

Multiple packages named "main"!

2. Write to `stdout` or a file as you go (locked with `sync.Mutex`)

3. Run analysis driver in child process

```
func main() {  
    if os.Getenv("singlechecker") != "" {  
        // child process  
        singlechecker.Main(analyzer)  
    }  
  
    // parent process  
    ...  
}
```

# Not working well

 Multiple passes, each with separate SSA program builds

 `ssa.BuilderMode?`

 RTA limitations?

“The resulting call graph is less precise than one produced by pointer analysis, but the algorithm is much faster. For example, running the cmd/callgraph tool on its own source takes ~2.1s for RTA and ~5.4s for points-to analysis.”

**Attempt #2**

# Rebuild from the ground up

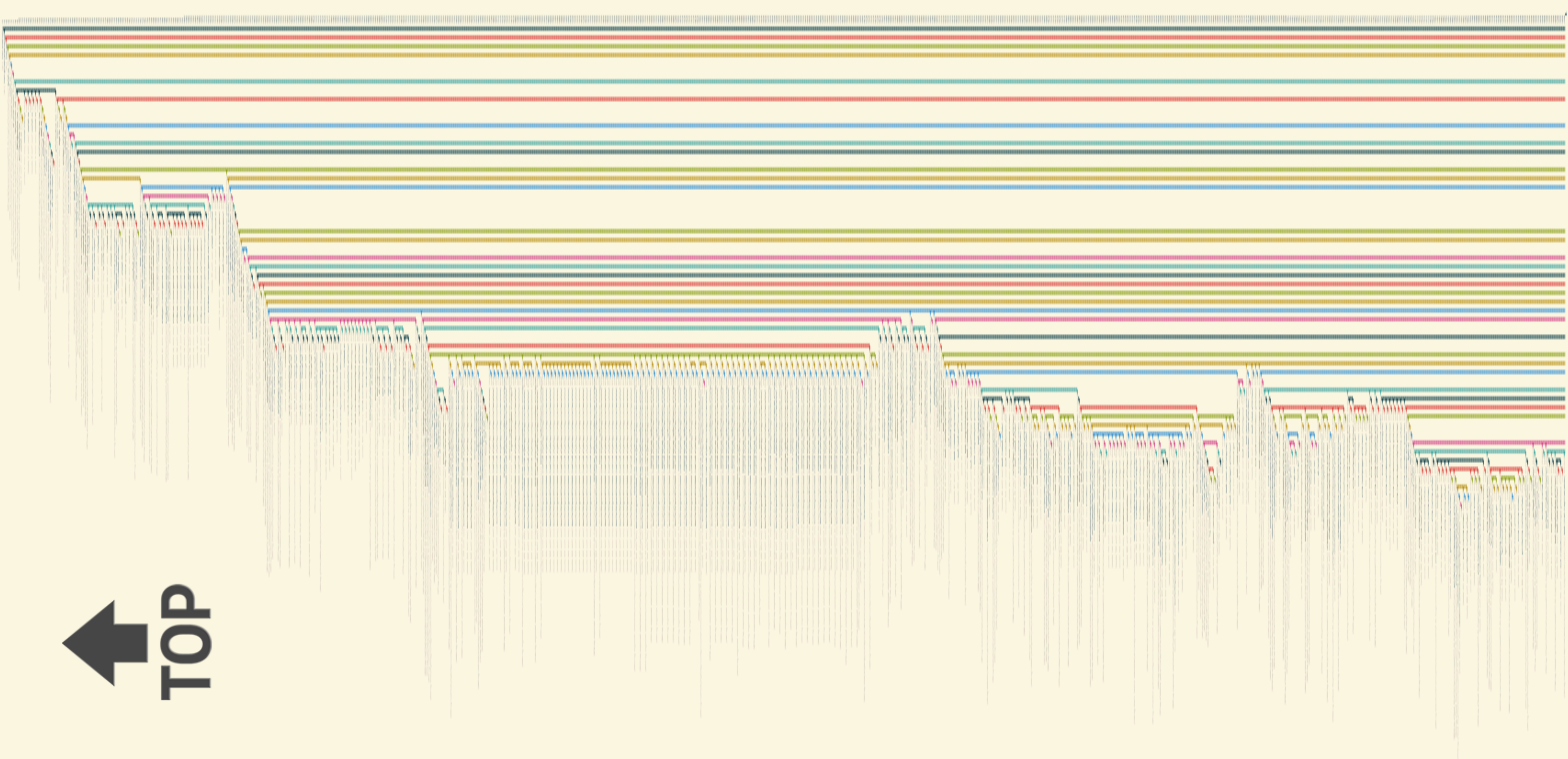
1. ~~Rapid Type Analysis (RTA)~~ → Andersen's pointer analysis (🙌)
2. Simpler driver:  
    `packages.Load`  
    `ssauti1.AllPackages and (*ssa.Program).Build`  
    `pointer.Analyze`
3. Visualize call graph

**So does it work?**



```
1: mytestdata.init (:0)
2:   mytestdata/config.init (:0)
3:   os.init (:0)
4:     111c32 os.Getwd (/usr/local/Cellar/go/1.12.7/libexec/src/os/getwd.go:26)
5:     137c14 os.Getenv (/usr/local/Cellar/go/1.12.7/libexec/src/os/env.go:101)
6:     1103c24 syscall.Getenv (/usr/local/Cellar/go/1.12.7/libexec/src/syscall/env_unix.go:71)
7:     172c12 (*sync.Once).Do (/usr/local/Cellar/go/1.12.7/libexec/src/sync/once.go:35)
8:     144c4 net.initConfVal (/usr/local/Cellar/go/1.12.7/libexec/src/net/conf.go:46)
9:     147c38 net.goDebugNetDNS (/usr/local/Cellar/go/1.12.7/libexec/src/net/conf.go:293)
10:    1294c26 net.goDebugString (/usr/local/Cellar/go/1.12.7/libexec/src/net/parse.go:363)
11:      1364c16 os.Getenv (see line 5 above)
12:      ...
13:      180c41 syscall.Getenv (see line 6 above)
14:      ...
15:      181c14 os.Getenv (see line 5 above)
16:      ...
17:      182c12 os.Getenv (see line 5 above)
18:      ...
19:      191c43 os.Getenv (see line 5 above)
20:      ...
21:      197c33 net.parseNSSConfFile (/usr/local/Cellar/go/1.12.7/libexec/src/net/nss.go:69)
22:      170c19 os.Open (/usr/local/Cellar/go/1.12.7/libexec/src/os/file.go:264)
23:      1265c17 os.OpenFile (/usr/local/Cellar/go/1.12.7/libexec/src/os/file.go:282)
24:        1284c22 os.openFileNoLog (/usr/local/Cellar/go/1.12.7/libexec/src/os/file_unix.go:190)
25:        1227c16 os.newFile (/usr/local/Cellar/go/1.12.7/libexec/src/os/file_unix.go:103)
26:        1156c22 (*internal/poll.FD).Init (/usr/local/Cellar/go/1.12.7/libexec/src/internal/poll/fd_unix.go:54)
27:          163c19 (*internal/poll.pollDesc).init (/usr/local/Cellar/go/1.12.7/libexec/src/internal/poll/fd_poll_runtime.go:37)
28:            138c15 (*sync.Once).Do (see line 7 above)
29:            ...
30:      1100c32 net.dnsReadConfig (/usr/local/Cellar/go/1.12.7/libexec/src/net/dnsconfig_unix.go:38)
31:      144c19 net.open (/usr/local/Cellar/go/1.12.7/libexec/src/net/parse.go:67)
32:        168c20 os.Open (see line 22 above)
33:        ...
34:        160c31 (*net.file).readLine (/usr/local/Cellar/go/1.12.7/libexec/src/net/parse.go:49)
35:        155c24 io.ReadFull (/usr/local/Cellar/go/1.12.7/libexec/src/io/io.go:328)
36:        1329c20 io.ReadAtLeast (/usr/local/Cellar/go/1.12.7/libexec/src/io/io.go:304)
37:          1310c19 (*crypto/rand.devReader).Read (/usr/local/Cellar/go/1.12.7/libexec/src/crypto/rand/rand_unix.go:50)
38:            163c20 os.Open (see line 22 above)
39:            ...
40:            173c17 (*bufio.Reader).Read (/usr/local/Cellar/go/1.12.7/libexec/src/bufio/bufio.go:197)
41:            1209c24 (mime/multipart.partReader).Read (/usr/local/Cellar/go/1.12.7/libexec/src/mime/multipart/multipart.go:167)
42:            1174c21 (*bufio.Reader).Peek (/usr/local/Cellar/go/1.12.7/libexec/src/bufio/bufio.go:129)
43:            1138c9 (*bufio.Reader).fill (/usr/local/Cellar/go/1.12.7/libexec/src/bufio/bufio.go:86)
44:            1100c22 (*net/http.http2gzipReader).Read (/usr/local/Cellar/go/1.12.7/libexec/src/net/http/h2_bundle.go:8833)
45:            18838c30 compress/gzip.NewReader (/usr/local/Cellar/go/1.12.7/libexec/src/compress/gzip/gunzip.go:92)
46:            194c19 (*compress/gzip.Reader).Reset (/usr/local/Cellar/go/1.12.7/libexec/src/compress/gzip/gunzip.go:103)
47:            1113c32 (*compress/gzip.Reader).readHeader (/usr/local/Cellar/go/1.12.7/libexec/src/compress/gzip/gunzip.go:174)
48:              1175c25 io.ReadFull (see line 35 above)
49:              ...
50:              1196c31 hash/crc32.ChecksumIEEE (/usr/local/Cellar/go/1.12.7/libexec/src/hash/crc32/crc32.go:251)
51:                1252c13 (*sync.Once).Do (see line 7 above)
52:                ...
53:                1199c26 io.ReadFull (see line 35 above)
54:                ...
55:                1202c26 hash/crc32.Update (/usr/local/Cellar/go/1.12.7/libexec/src/hash/crc32/crc32.go:210)
56:                  1217c14 (*sync.Once).Do (see line 7 above)
57:                  ...
58:                  1204c26 io.ReadFull (see line 35 above)
59:                  ...
60:                  1207c26 hash/crc32.Update (see line 55 above)
61:                  ...
62:                  1213c27 (*compress/gzip.Reader).readString (/usr/local/Cellar/go/1.12.7/libexec/src/compress/gzip/gunzip.go:141)
63:                    1148c31 (*bufio.Reader).ReadByte (/usr/local/Cellar/go/1.12.7/libexec/src/bufio/bufio.go:243)
64:                      1249c9 (*bufio.Reader).fill (see line 43 above)
65:                      ...
66:                      1157c27 hash/crc32.Update (see line 55 above)
67:                      ...
68:                      1220c27 (*compress/gzip.Reader).readString (see line 62 above)
69:                      ...
70:                      1227c26 io.ReadFull (see line 35 above)
71:                      ...
72:                      1238c35 compress/flate.NewReader (/usr/local/Cellar/go/1.12.7/libexec/src/compress/flate/inflate.go:796)
73:                        1797c25 compress/flate.fixedHuffmanDecoderInit (/usr/local/Cellar/go/1.12.7/libexec/src/compress/flate/inflate.go:756)
74:                          1757c14 (*sync.Once).Do (see line 7 above)
75:                          ...
76:                        18844c19 (*compress/gzip.Reader).Read (/usr/local/Cellar/go/1.12.7/libexec/src/compress/gzip/gunzip.go:246)
77:                          1251c32 (*compress/flate.decompressor).Read (/usr/local/Cellar/go/1.12.7/libexec/src/compress/flate/inflate.go:334)
78:                            1347c9 (*compress/flate.decompressor).copyData (/usr/local/Cellar/go/1.12.7/libexec/src/compress/flate/inflate.go:654)
79:                              1660c25 io.ReadFull (see line 35 above)
80:                              ...
81:                              1347c9 (*compress/flate.decompressor).nextBlock (/usr/local/Cellar/go/1.12.7/libexec/src/compress/flate/inflate.go:301)
82:                                1303c24 (*compress/flate.decompressor).moreBits (/usr/local/Cellar/go/1.12.7/libexec/src/compress/flate/inflate.go:695)
83:                                  1696c24 (*bufio.Reader).ReadByte (see line 63 above)
```





↑  
TOP

4,139 lines

# Opening files is pretty common

- `os.init`

```
Stdin  = NewFile(uintptr(syscall.Stdin), "/dev/stdin")
Stdout = NewFile(uintptr(syscall.Stdout), "/dev/stdout")
Stderr = NewFile(uintptr(syscall.Stderr), "/dev/stderr")
```

- `(*crypto/rand.devReader).Read`

```
os.Open(r.name) (which was set in init to "/dev/urandom")
```

- `net`

```
parseNSSConfFile("/etc/nsswitch.conf")
dnsReadConfig("/etc/resolv.conf")
goLookupPort → readServices → open("/etc/services")
lookupProtocol → readProtocols → open("/etc/protocols")
```

# False positive

```
package main
```

```
import (  
    "go/parser"  
    "go/token"  
)
```

```
func main() {  
    parser.ParseFile(token.NewFileSet(), "", "package main", 0)  
}
```



# False positive

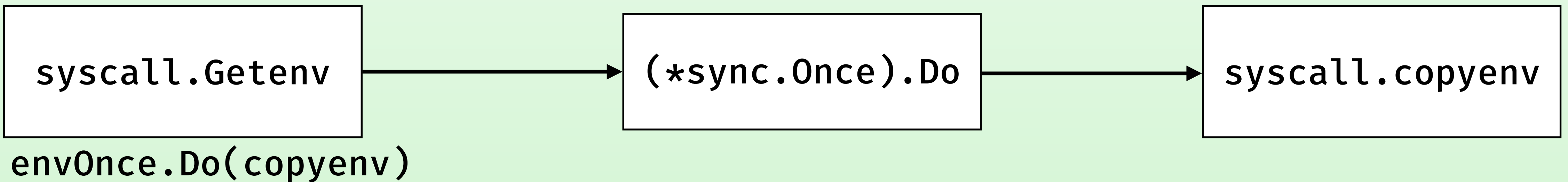
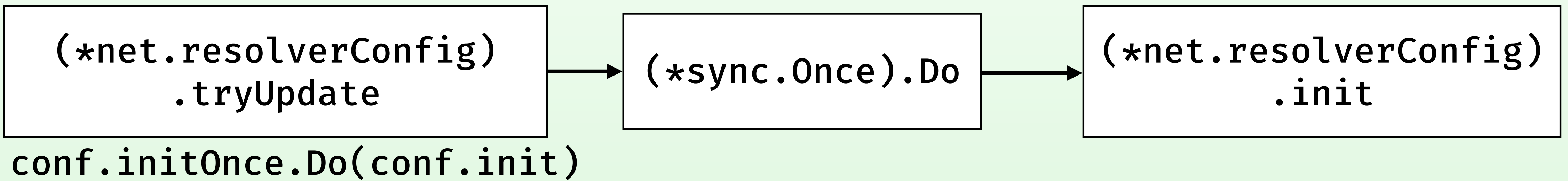
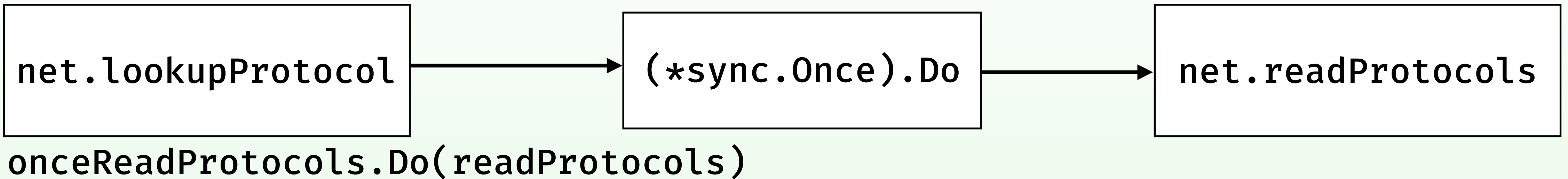
```
func ParseFile(
    fset *token.FileSet, filename string, src interface{}, mode Mode,
) (f *ast.File, err error) {
    ...
    text, err := readSource(filename, src)
    ...
}

func readSource(filename string, src interface{}) ([]byte, error) {
    if src != nil {
        switch s := src.(type) {
            // ... return a []byte
        }
    }
    return ioutil.ReadFile(filename)
}
```

The  
(*\*sync.Once*).Do  
problem

```
func (o *Once) Do(f func())
```

“Do calls the function f  
if and only if  
Do is being called for the first time  
for this instance of Once.”



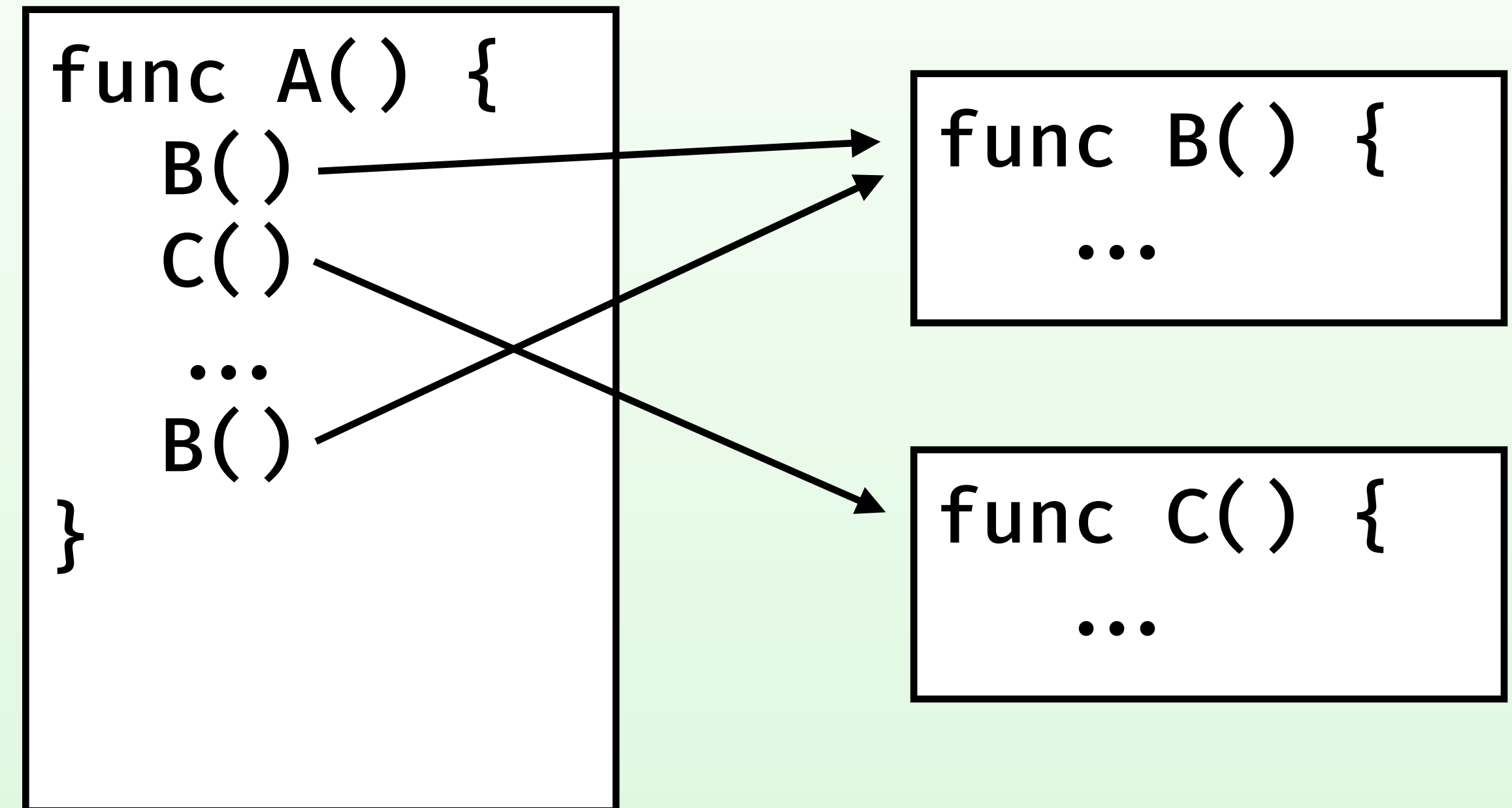


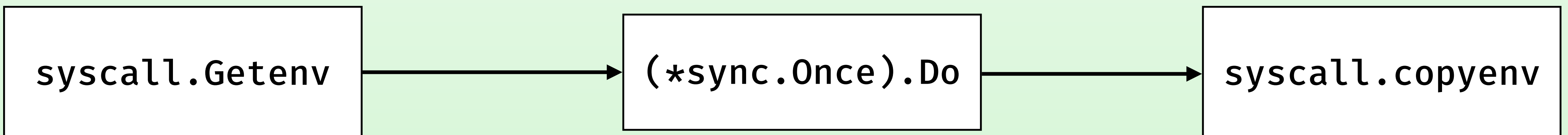
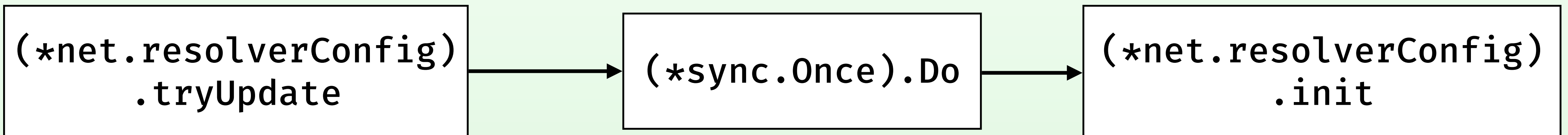
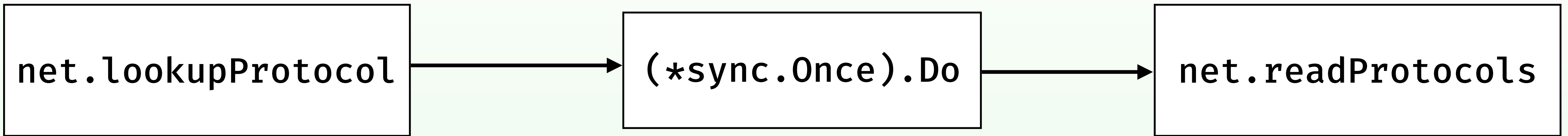
# golang.org/x/tools/go/callgraph

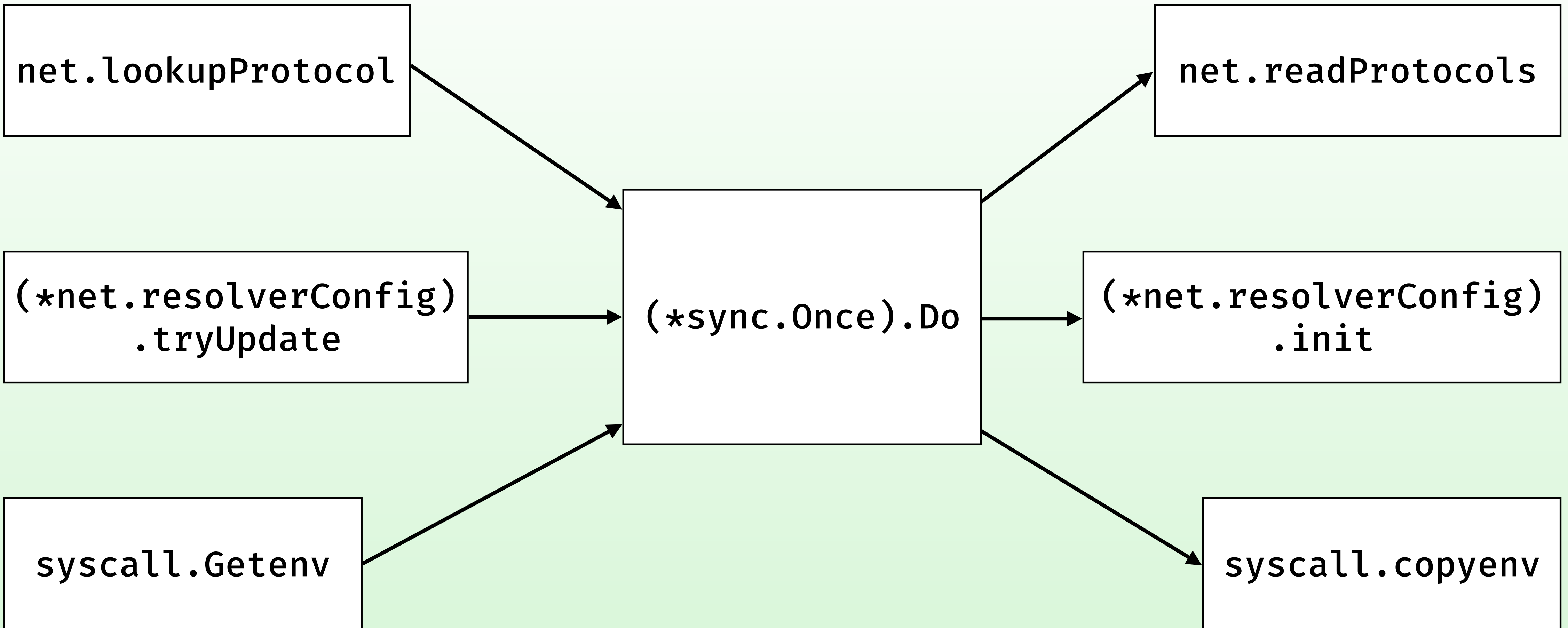
```
type Graph struct {  
    Root *Node  
    Nodes map[*ssa.Function]*Node  
}
```

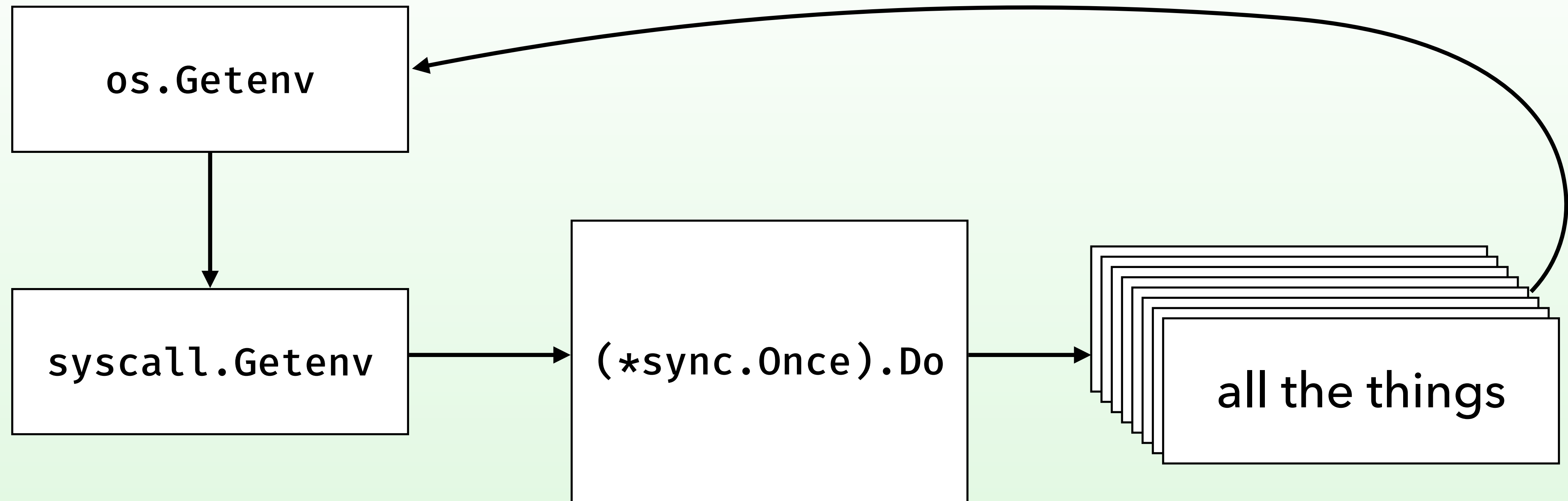
```
type Node struct {  
    Func *ssa.Function  
    ID    int  
    In    []*Edge  
    Out   []*Edge  
}
```

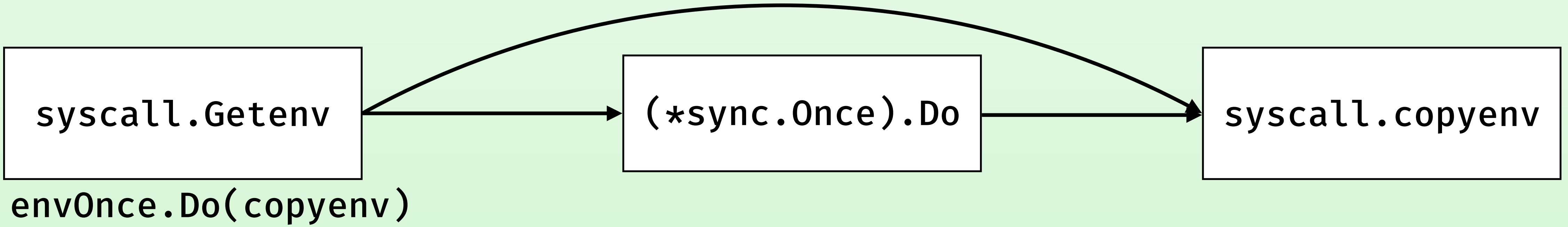
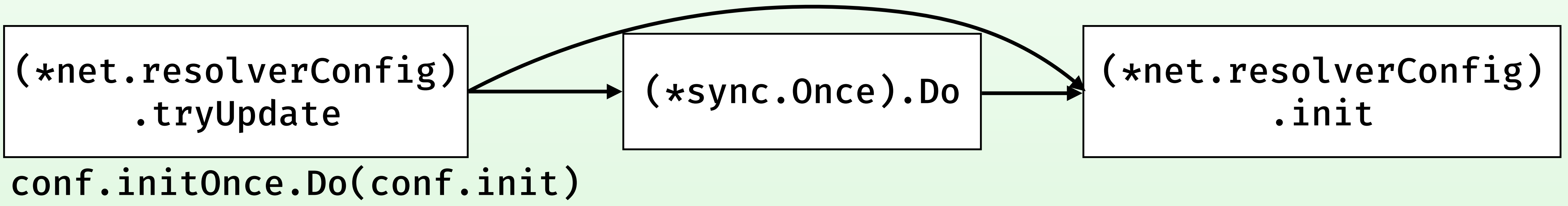
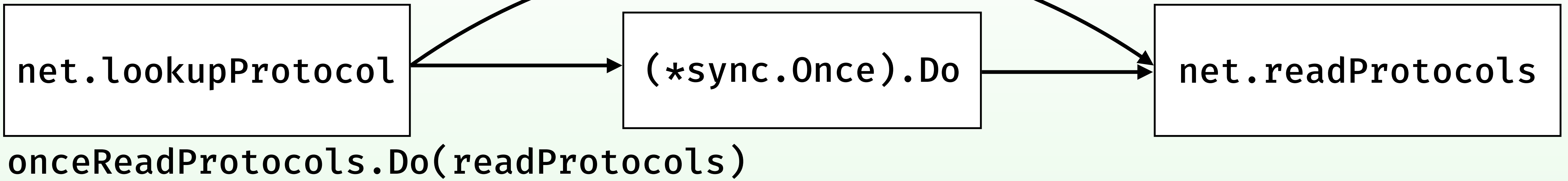
```
type Edge struct {  
    Caller *Node  
    Site   ssa.CallInstruction  
    Callee *Node  
}
```











**Does it work now?**



```
1: x mytestdata.init -> :0
2:   x mytestdata/config.init -> :0
3:     x mytestdata/config.init#1 -> /Users/mseplowitz/mytestdata/config/config.go:26
4:       x os.Open config.go:28:30 -> /usr/local/Cellar/go/1.12.7/libexec/src/os/file.go:264
5:     x mytestdata/config.init#2 -> /Users/mseplowitz/mytestdata/config/config.go:34
6:       x os.Open config.go:36:30 -> (see line 4 above)
7:     x os.Open config.go:12:28 -> (see line 4 above)
8:     x os.Open config.go:14:40 -> (see line 4 above)
9: x mytestdata.main -> /Users/mseplowitz/mytestdata/main.go:14
10:   x mytestdata.httpFunctionCalls main.go:18:19 -> /Users/mseplowitz/mytestdata/main.go:22
11:     x mytestdata/golangdotorg.HTTPGet main.go:23:35 -> /Users/mseplowitz/mytestdata/golangdotorg/http.go:11
12:       x net/http.Get http.go:12:17 -> /usr/local/Cellar/go/1.12.7/libexec/src/net/http/client.go:369
13:     x mytestdata/golangdotorg.AnonFuncWrappedHTTPGet main.go:26:49 -> /Users/mseplowitz/mytestdata/golangdotorg/http.go:15
14:       x mytestdata/golangdotorg.AnonFuncWrappedHTTPGet$1 http.go:16:60 -> /Users/mseplowitz/mytestdata/golangdotorg/http.go:16
15:         x mytestdata/golangdotorg.HTTPGet http.go:16:56 -> (see line 11 above)
16:         ...
17:     x mytestdata/golangdotorg.HTTPGet main.go:29:40 -> (see line 11 above)
18:     ...
19:     x mytestdata/golangdotorg.HTTPGet main.go:33:16 -> (see line 11 above)
20:     ...
21:     x mytestdata/golangdotorg.ReturnAnonFuncWrappingHTTPGet$1 main.go:37:16 -> /Users/mseplowitz/mytestdata/golangdotorg/http.go:31
22:       x mytestdata/golangdotorg.HTTPGet http.go:31:56 -> (see line 11 above)
23:       ...
24:     x mytestdata/golangdotorg.HTTPGet main.go:41:16 -> (see line 11 above)
25:     ...
26:     x mytestdata/golangdotorg.HTTPClientGet main.go:44:40 -> /Users/mseplowitz/mytestdata/golangdotorg/http.go:47
27:       x (*net/http.Client).Get http.go:49:15 -> /usr/local/Cellar/go/1.12.7/libexec/src/net/http/client.go:393
28:     x mytestdata/golangdotorg.HTTPClientDoGet main.go:47:42 -> /Users/mseplowitz/mytestdata/golangdotorg/http.go:52
29:       x (*net/http.Client).Do http.go:65:14 -> /usr/local/Cellar/go/1.12.7/libexec/src/net/http/client.go:508
30:   x mytestdata.httpMethodCalls main.go:19:17 -> /Users/mseplowitz/mytestdata/main.go:51
31:     x (*mytestdata/golangdotorg.Client).Get main.go:54:21 -> /Users/mseplowitz/mytestdata/golangdotorg/http.go:74
32:       x mytestdata/golangdotorg.HTTPGet http.go:75:16 -> (see line 11 above)
33:       ...
```



```
1: x mytestdata.init -> :0
2: └─ x mytestdata/config.init -> :0
3:   └─ x mytestdata/config.init#1 -> /Users/mseplowitz/mytestdata/config/config.go:26
4:     └─ x os.Open config.go:28:30 -> /usr/local/Cellar/go/1.12.7/libexec/src/os/file.go:264
5:   └─ x mytestdata/config.init#2 -> /Users/mseplowitz/mytestdata/config/config.go:34
6:     └─ x os.Open config.go:36:30 -> (see line 4 above)
7:   └─ x os.Open config.go:12:28 -> (see line 4 above)
8:   └─ x os.Open config.go:14:40 -> (see line 4 above)
9: ✓ mytestdata.main -> /Users/mseplowitz/mytestdata/main.go:14
10: └─ ✓ mytestdata.httpFunctionCalls main.go:18:19 -> /Users/mseplowitz/mytestdata/main.go:22
11:   └─ ✓ mytestdata/golangdotorg.HTTPGet main.go:23:35 -> /Users/mseplowitz/mytestdata/golangdotorg/http.go:11
12:     └─ ✓ net/http.Get [golang.org] http.go:12:17 -> /usr/local/Cellar/go/1.12.7/libexec/src/net/http/client.go:369
13:   └─ ✓ mytestdata/golangdotorg.AnonFuncWrappedHTTPGet main.go:26:49 -> /Users/mseplowitz/mytestdata/golangdotorg/http.go:15
14:     └─ ✓ mytestdata/golangdotorg.AnonFuncWrappedHTTPGet$1 http.go:16:60 -> /Users/mseplowitz/mytestdata/golangdotorg/http.go:16
15:       └─ ✓ mytestdata/golangdotorg.HTTPGet http.go:16:56 -> (see line 11 above)
16:         └─ ...
17:   └─ ✓ mytestdata/golangdotorg.HTTPGet main.go:29:40 -> (see line 11 above)
18:     └─ ...
19:   └─ ✓ mytestdata/golangdotorg.HTTPGet main.go:33:16 -> (see line 11 above)
20:     └─ ...
21:   └─ ✓ mytestdata/golangdotorg.ReturnAnonFuncWrappingHTTPGet$1 main.go:37:16 -> /Users/mseplowitz/mytestdata/golangdotorg/http.go:31
22:     └─ ✓ mytestdata/golangdotorg.HTTPGet http.go:31:56 -> (see line 11 above)
23:       └─ ...
24:   └─ ✓ mytestdata/golangdotorg.HTTPGet main.go:41:16 -> (see line 11 above)
25:     └─ ...
26:   └─ ✓ mytestdata/golangdotorg.HTTPClientGet main.go:44:40 -> /Users/mseplowitz/mytestdata/golangdotorg/http.go:47
27:     └─ ✓ (*net/http.Client).Get [golang.org] http.go:49:15 -> /usr/local/Cellar/go/1.12.7/libexec/src/net/http/client.go:393
28:   └─ ✓ mytestdata/golangdotorg.HTTPClientDoGet main.go:47:42 -> /Users/mseplowitz/mytestdata/golangdotorg/http.go:52
29:     └─ ✓ (*net/http.Client).Do [golang.org] http.go:66:14 -> /usr/local/Cellar/go/1.12.7/libexec/src/net/http/client.go:508
30: └─ ✓ mytestdata.httpMethodCalls main.go:19:17 -> /Users/mseplowitz/mytestdata/main.go:51
31:   └─ ✓ (*mytestdata/golangdotorg.Client).Get main.go:54:21 -> /Users/mseplowitz/mytestdata/golangdotorg/http.go:75
32:     └─ ✓ mytestdata/golangdotorg.HTTPGet http.go:76:16 -> (see line 11 above)
33:       └─ ...
```



**Run it on real code!**

# Reflection 🙈

- Team-owned microservice framework 😎
- Command-line parser: [github.com/jessevdk/go-flags](https://github.com/jessevdk/go-flags)

## Rapid Type Analysis

```
type Result struct {  
    // CallGraph is the discovered callgraph.  
    // It does not include edges for calls made via reflection.
```

## Pointer analysis

“Most but not all reflection operations are supported. In particular, addressable `reflect.Values` are not yet implemented, so operations such as `(reflect.Value).Set` have no analytic effect.”

# Takeaways

# Takeaways

- Call graphs can be easy or tricky!
- Some documentation is great!
- Start simple!
- Talk to the community!

# Thanks!

Mike Seplowitz

 @mikesep

 @mikesep

 <https://mikesep.dev>

Bloomberg

Engineering

 @bloomberg

 @TechAtBloomberg

 <https://TechAtBloomberg.com>