静的解析

🔥 Go 2 🔥

静的解析

🔥 Go 2 🔥

(My second Go talk at Innopolis)

静的解析

github.com/go-critic

Static analysis in Go

静的解析

Even if `gometalinter --enable-all` reports no issues, gocritic can find something new

静的解析

D - Diagnostics

静的解析

Example: D nilValReturn (bad)

```
if err == nil {
  return err
}
// (A) Typo in “==”?
// (B) Wanted to return nil?
```

Example: D nilValReturn (good)

```
if err != nil { // (A)
  return err
}
if err == nil { // (B)
  return nil
}
```

Example: D caseOrder (bad)

```
switch x.(type) {
case ast.Expr:
  fmt.Println("expr")
case *ast.BasicLit:
  fmt.Println("basic lit")
}
```

Example: D caseOrder (good)

```
switch x.(type) {
case *ast.BasicLit:
  fmt.Println("expr")
case ast.Expr:
  fmt.Println("basic lit")
}
```

Example: D dupArg (bad)

```
copy(dst, dst)

reflect.Copy(x, x)

strings.Contains(s, s)
```

Example: D dupArg (good)

```
copy(dst, src)

reflect.Copy(x, y)

strings.Contains(s, s2)
```

S - Code style

静的解析

Example: S typeSwitchVar (bad)

```
switch x.(type) {
case *T1:
  return x.(*T1).v1 + x.(*T1).v2
case *T2:
  return x.(*T2).v
}
```

Example: S typeSwitchVar (good)

```
switch x := x.(type) {
case *T1:
  return x.v1 + x.v2
case *T2:
  return x.v
}
```

Example: S namedConst (bad)

```
if c.jsCtx != 0 {
  s += "_" + c.jsCtx.String()
}
if c.attr != 0 {
  s += "_" + c.attr.String()
}
```

Example: namedConst (good)

```
if c.jsCtx != jsCtxRegexp {
  s += "_" + c.jsCtx.String()
}
if c.attr != attrNone {
  s += "_" + c.attr.String()
}
```

Example: S unlambda (bad)

```
trim := strings.TrimFunc
trim(s, func(r rune) bool {
  return unicode.IsSpace(r)
})
```

Example: S unlambda (good)

```
trim := strings.TrimFunc
trim(s, unicode.IsSpace)
```

P - Performance

静的解析

Example: P rangeExprCopy (bad)

```
// Global array
builtins = [...]Func{...}

for _, fn := range builtins {
  // ...
}
```

Example: P rangeExprCopy (good)

```
// Global array
builtins = [...]Func{...}

for _, fn := range &builtins {
  // ...
}
```

56 + 54

56 checks implemented, 54 more planned/requested

https://go-critic.github.io/overview

静的解析

Useful tool: gogrep

```
$ pat="if err == nil { return err }"

$ gogrep $pat path/to/pkg

# Like grep, but for AST patterns
# https://github.com/mvdan/gogrep
```

Which one is better?

```
x := new(T)
```

```
x := &T{}
```

静的解析

Which one is better?

```
x := 0xff
```

```
x := 0xFF
```

github.com/Quasilyte/go-consistent

Answer: the one you use consistently

静的解析

Linter

Opinion

Convention

What are these?

Category 1: CI-oriented linters

- False positives > false negatives
- Confidence close to 100%
- Executed automatically
- Executed frequently (~always)

e.g.: golint, staticcheck

Category 2: Audit-oriented linters

- False negatives are important
- Confidence is 50/50 sometimes
- Executed manually
- Executed on demand

e.g.: gocritic, unparam

Category 3: Linter runners

- Integrate other linters
- Aggregated and pretty output
- Various useful integrations
- Do not implement new checks

e.g.: gometalinter, golangci-lint

Example of suspicious code

```
rows[i+0] = getRow(key0)
rows[i+0] = getRow(key1)
rows[i+2] = getRow(key2)

// Can be a copy/paste error
```

Optimistic merging

rfc.zeromq.org/spec:42/C4

“People before code: build the right community and it will build the right code”

gocritic objectively

Vague, goals are not clearly defined from the start

静的解析

gocritic of mine

What I wanted gocritic to be

静的解析

gocritic of others

What others want gocritc to become

静的解析

Project-local conventions

e.g.: use X instead of Y

https://github.com/ewgRa/gocsfixer

静的解析

Some gocritic checks future

dominikh commented on Jul 30 • edited

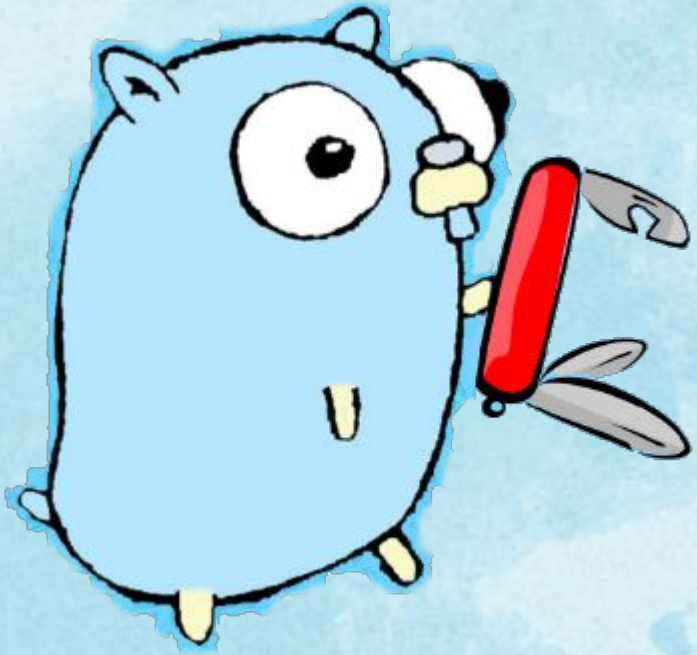
Owner

- [ ] ~~builtinShadow – non-default stylecheck~~ – not going to add this, too likely to be misused
- [ ] caseOrder – staticcheck
- [ ] deadCodeAfterLogFatal – gosimple, dup of #111
- [x] defaultCaseOrder – stylecheck
- [ ] dupBranchBody – staticcheck, dup of #174
- [ ] dupCase – gosimple
- [ ] elseif – stylecheck or gosimple
- [ ] evalOrder – staticcheck, dup of #258
- [ ] flagDeref – staticcheck
- [ ] importShadow – non-default stylecheck
- [ ] indexOnlyLoop – gosimple
- [ ] initClause – stylecheck or gosimple
- [ ] regexpMust – stylecheck
- [ ] singleCaseSwitch – stylecheck or gosimple
- [ ] sloppyLen – staticcheck
- [ ] typeSwitchVar – gosimple, dup of #126
- [x] yodaStyleExpr – stylecheck

Good or bad?

Depends on the project goals and perceiving side

静的解析



Built with
go-toolsmith

github.com/go-toolsmith

astfmt	Print ast.Node with %s
strparse	Parse string to AST
astcopy	Deep copy for AST nodes
astequal	Deep equal for AST nodes
astp	Predicates for AST nodes

github.com/go-toolsmith

Checking Go toolchain with gocritic

RU: https://habr.com/post/416903/

EN: https://bit.ly/2MhQoN5

静的解析

Getting involved

- Suggest Go projects for audit
- Share ideas
- Provide feedback
- Use go-toolsmith in your tools
- Contribute your code

静的解析

Networking: compilers, Go tools

- E-mail: quasilyte@gmail.com
- Go community slacks (@quasilyte):
  - RU: golang-ru.slack.com
  - EN: gophers.slack.com
- gocritic Telegram group:
  - https://t.me/go_critic_ru

Questions time

- gocritic linter
- go-consistent linter
- Optimistic merging
- go-toolsmith libs
- Go projects audit
- Contributing