Санкт-Петербургский государственный университет

Кафедра системного программирования

Инструмент генерации юнит-тестов для GoLang

Кузиванов Сергей Юрьевич, 18.Б11-мм

Научный руководитель: доцент кафедры системного программирования, к. т. н., Ю. В. Литвинов
Консультант: разработчик ООО “Яндекс”, В. Е. Володин
Рецензент: руководитель департамента анализа программ ООО Техкомпания “Хуавей”, Д. А. Иванов

Санкт-Петербург
2022

Введение

- Различают несколько видов тестирования, один из которых – модульное тестирование
- Символьное исполнение, как один из подходов автоматической генерации модульных тестов, является перспективным, однако на текущий момент недостаточно развит
- Проект символьного движка KLEE[1] реализует символьное исполнение для LLVM биткода
- Проект компилятора языка Go Gollvm[2] позволяет транслировать исходный код Go в LLVM биткод

[1] https://klee.github.io/
[2] https://go.googlesource.com/gollvm/

Постановка задачи

Цель: создание инструмента, позволяющего для произвольной функции в коде на языке Go автоматически генерировать тестовый Go-файл с модульными тестами с использованием символьного исполнения при генерации тестов.

Задачи

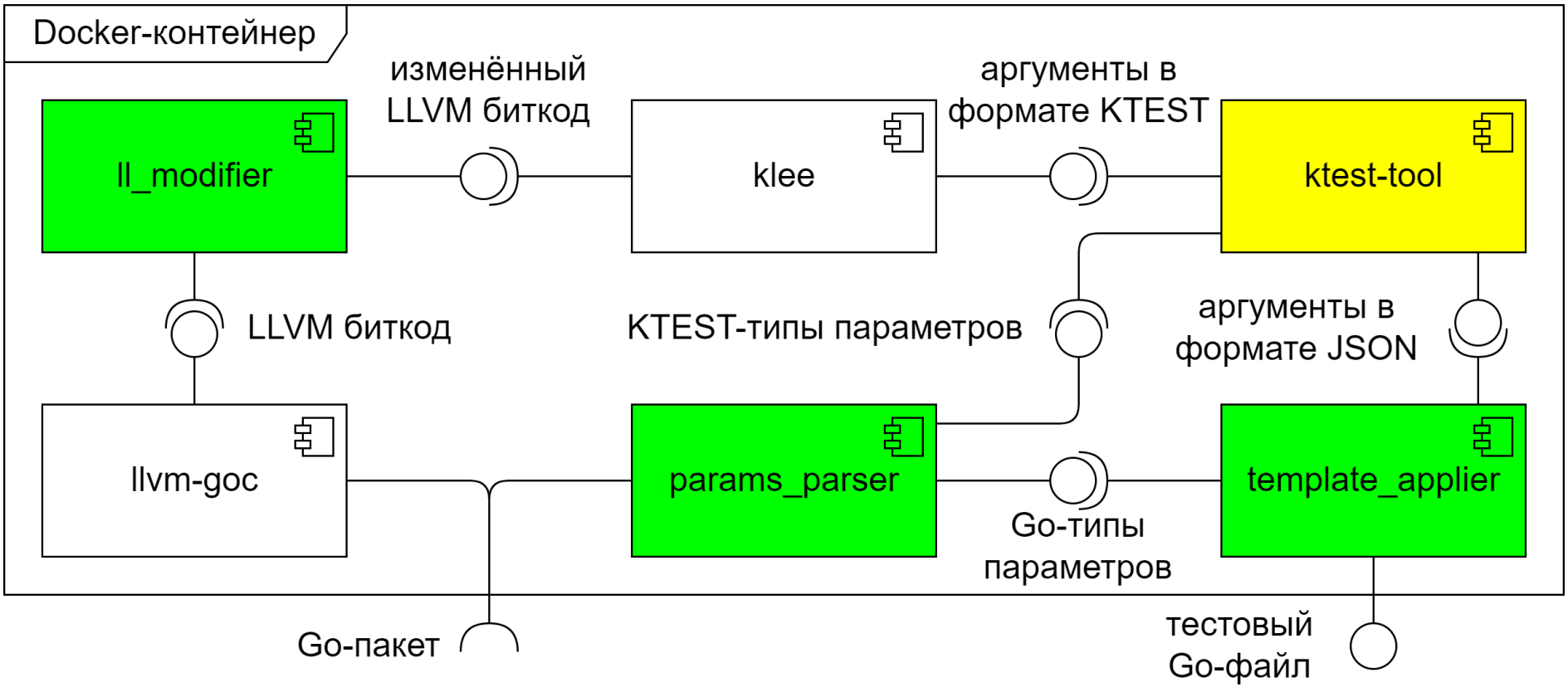
- Обзор существующих инструментов для генерации модульных тестов для кода на языке Go
- Проектирование инструмента для генерации модульных тестов для произвольной функции в виде тестового файла на Go
- Реализация спроектированного инструмента
- Проведение экспериментов с целью проверки работоспособности разработанного решения

Обзор

- gofuzz
- go-fuzz
- fzgo
- Встроенный механизм фаззинга[3]
- DUCKEE GO[4]
- gollvm + klee

[3] https://go.dev/doc/fuzz/
[4] https://github.com/DieracDelta/DuckeeGO

Архитектура инструмента



Преобразование **ll_modifier**

LLVM биткод преобразуется следующим образом:

- Все глобальные переменные, определение которых содержится в стандартной библиотеке Go, определяются нулями
- Создаются вспомогательные глобальные переменные
- Описывается сигнатура специальной функции KLEE `klee_make_symbolic`, предназначенная для того, чтобы помечать переменные как символьные
- Определяется функция `klee.main` для запуска тестируемой функции с символьными переменными

Преобразование **ll_modifier**

```
@main..types = constant { i64, [1 x i8*] } zeroinitia
lizer
@internal_1cpu..types = external externally_initializ
ed global { i64, [1 x i8*] }
@runtime..types = external externally_initialized glo
bal { i64, [1 x i8*] }
@internal_1bytealg..types = external externally_initi
alized global { i64, [1 x i8*] }
@runtime_1internal_1atomic..types = external external
ly_initialized global { i64, [1 x i8*] }
@runtime_1internal_1math..types = external externally
_initialized global { i64, [1 x i8*] }
@runtime_1internal_1sys..types = external externally_
initialized global { i64, [1 x i8*] }
@go..typelists = internal constant [7 x { i64, [1 x i
8*] }*] [{ i64, [1 x i8*] }* @internal_1cpu..types, {
 i64, [1 x i8*] }* @runtime..types, { i64, [1 x i8*]
}* @internal_1bytealg..types, { i64, [1 x i8*] }* @ru
ntime_1internal_1atomic..types, { i64, [1 x i8*] }* @
runtime_1internal_1math..types, { i64, [1 x i8*] }* @
runtime_1internal_1sys..types, { i64, [1 x i8*] }* @m
ain..types]
----------------------------------------------------
----------------------------------------------------
----------------------------------------------------

declare void @0(i8*, ...)

define void @__go_init_main(i8* nest %nest.29) #0 !db
g !29 {
<tbotgo/package/llvm_bitcode.ll [Ч] 136,1        5%
```

```
@main..types = constant { i64, [1 x i8*] } zeroinitia
lizer
@internal_1cpu..types = global { i64, [1 x i8*] } zer
oinitializer
@runtime..types = global { i64, [1 x i8*] } zeroiniti
alizer
@internal_1bytealg..types = global { i64, [1 x i8*] }
 zeroinitializer
@runtime_1internal_1atomic..types = global { i64, [1
x i8*] } zeroinitializer
@runtime_1internal_1math..types = global { i64, [1 x
i8*] } zeroinitializer
@runtime_1internal_1sys..types = global { i64, [1 x i
8*] } zeroinitializer
@go..typelists = internal constant [7 x { i64, [1 x i
8*] }*] [{ i64, [1 x i8*] }* @internal_1cpu..types, {
 i64, [1 x i8*] }* @runtime..types, { i64, [1 x i8*]
}* @internal_1bytealg..types, { i64, [1 x i8*] }* @ru
ntime_1internal_1atomic..types, { i64, [1 x i8*] }* @
runtime_1internal_1math..types, { i64, [1 x i8*] }* @
runtime_1internal_1sys..types, { i64, [1 x i8*] }* @m
ain..types]
@klee.zero = internal constant i64 0
@klee.x.str = internal constant [2 x i8] c"x\00"
@klee_modify..klee.x.str = internal constant i8 2

declare void @0(i8*, ...)

define void @__go_init_main(i8* nest %nest.29) #0 !db
g !29 {
<ctions/getSign/llvm_bitcode.ll [Ч] 136,1        5%
```

Преобразование **ll_modifier**

```
  br i1 %trunc.43, label %then.43, label %else.43

+ +--152 строки: else.41:
declare void @runtime.registerTypeDescriptors(i8*, i6
4, i8*) #0

declare void @internal_1cpu..import(i8*) #0

declare void @runtime..import(i8*) #0

----------
----------
----------
----------
----------
----------
----------
----------
----------
----------
----------
attributes #0 = { "disable-tail-calls"="true" "frame-
pointer"="none" "null-pointer-is-valid"="true" "split
-stack" "target-cpu"="x86-64" "target-features"="+cx8
,+fxsr,+mmx,+sse,+sse2,+x87" }
attributes #1 = { argmemonly nofree nosync nounwind w
illreturn }
attributes #2 = { nofree nosync nounwind readnone spe
culatable willreturn }
<tbotgo/package/llvm_bitcode.ll [ТЧ] 1779,1        72%
```

```
  br i1 %trunc.43, label %then.43, label %else.43

+ +--152 строки: else.41:
declare void @runtime.registerTypeDescriptors(i8*, i6
4, i8*) #0

declare void @internal_1cpu..import(i8*) #0

declare void @runtime..import(i8*) #0

declare void @klee_make_symbolic(i8*, i64, i8*)

define internal void @klee.main() {
entry:
  %klee.x.addr = alloca i64, align 8
  %klee.x.bc = bitcast i64* %klee.x.addr to i8*
  call void @klee_make_symbolic(i8* %klee.x.bc, i64 8
, i8* getelementptr inbounds ([2 x i8], [2 x i8]* @kl
ee.x.str, i64 0, i64 0))
  %klee.x.ld = load i64, i64* %klee.x.addr, align 8
  %klee.call = call i64 @main.getSign(i8* undef, i64
%klee.x.ld)
  ret void
}

attributes #0 = { "disable-tail-calls"="true" "frame-
pointer"="none" "null-pointer-is-valid"="true" "split
-stack" "target-cpu"="x86-64" "target-features"="+cx8
,+fxsr,+mmx,+sse,+sse2,+x87" }
@
<ctions/getSign/llvm_bitcode.ll [ТЧ] 1782,1        72%
```

Утилита **template_applier**

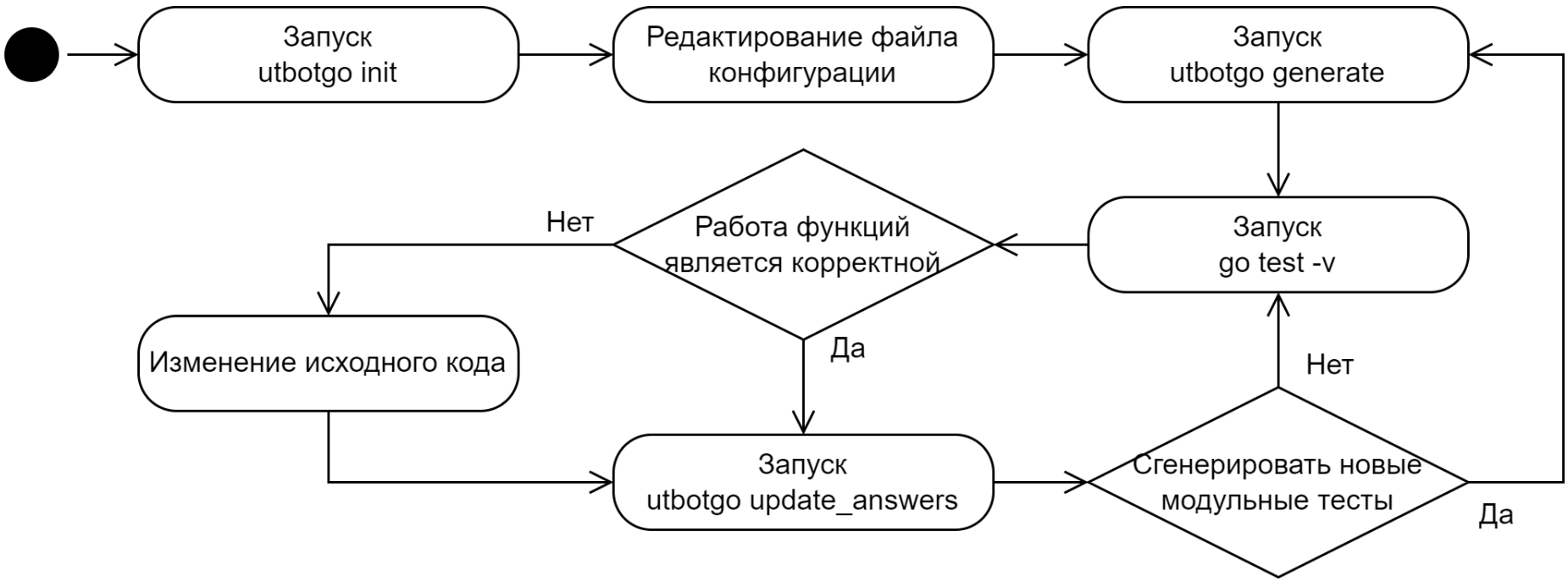
```
1  package {{package.name}}
2
3  import (
4      "encoding/json"
5      "os"
6      "strconv"
7      "testing"
8  )
9
10 type utbotgo_type_args_{{func.name}} struct {
11 {{  A{{func.args.name}} {{func.args.type}} `json:"{{f
   unc.args.name}}"`}|{
12 }}
   ----------------------------------------------------
13 }
14
15 var utbotgo_args_{{func.name}} = []utbotgo_type_args_
   {{func.name}}{
16     [[{ {{A{{func.args.name}}: [[args.{{func.args.nam
   e}}]]}|{, }} },]|[
17     ]]
   ----------------------------------------------------
   ----------------------------------------------------
18 }
19
20 type utbotgo_type_results_{{func.name}} struct {
21 {{  R{{func.results.name}} {{func.results.type}} `jso
   n:"R{{func.results.name}}"`}|{
22 }}
23 }
<otgo/utils/other_files/test.go_template 1,1          Наверху
```

```
1  package test4
2
3  import (
4      "encoding/json"
5      "os"
6      "strconv"
7      "testing"
8  )
9
10 type utbotgo_type_args_g struct {
11     Aa byte `json:"a"`
12     Ab rune `json:"b"`
13     Ac uint32 `json:"c"`
14 }
15
16 var utbotgo_args_g = []utbotgo_type_args_g{
17     { Aa: 0, Ab: 0, Ac: 0 },
18     { Aa: 252, Ab: 1, Ac: 253 },
19     { Aa: 254, Ab: 32768, Ac: 0 },
20     { Aa: 0, Ab: 2, Ac: 3 },
21 }
22
23 type utbotgo_type_results_g struct {
24     Rd int `json:"Rd"`
   ----------------------------------------------------
25 }
26
27 var utbotgo_answers_g = []utbotgo_type_resul
   ts_g{
28
utbotgo_g_test.go [ТЧ]          1,1          Наверху
```

Использованные языки и технологии

- **C++** – изменение LLVM биткода посредством LLVM API
- **Make** – сборка и взаимодействие утилит инструмента
- **CMake** – генерация LLVM биткода стандартной библиотеки Go
- **Python** – работа с форматами JSON и YAML
- **Go** – обработка и изменение текста и парсинг кода на Go
- **Docker** – создание Docker-образа
- **Shell** – написание простых утилит

Использование утилиты **utbotgo**

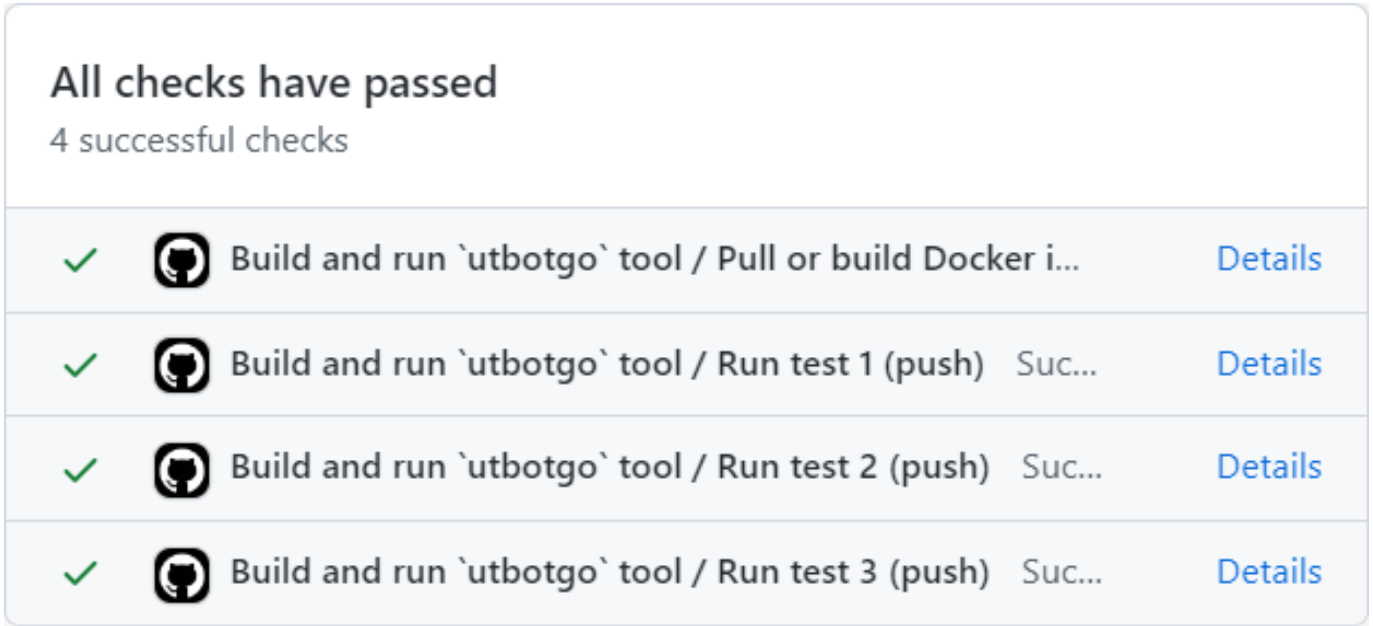


Ограничения на функции

- Результат функции не должен зависеть от глобальных переменных и не должен вызывать функцию, результат которой зависит от глобальных переменных
- Функция не должна вызывать функцию из другого пакета; при необходимости рекомендуется перенести весь используемый код из других пакетов в текущий, при этом код не должен содержать вставок кода на языке C/C++ или на ассемблере
- Все параметры функции должны быть целочисленными; этому ограничению соответствуют следующие типы языка Go: `rune`, `int`, `int8`, `int16`, `int32`, `int64`, `byte`, `uint`, `uint8`, `uint16`, `uint32` и `uint64`.
- Функция должна возвращать ровно одно значение
- Тип возвращаемого значения должен быть простым базовым Go-типом, сложные типы, такие как массив, карта и структура, не допускаются

Эксперименты

Тестируемая функциональность	`test1`	`test2`	`test3`
Многофайловый Go-пакет	✓		
Исполняемые Go-пакет	✓		
Несколько тестовых функций		✓	✓
Арифметические выражения		✓	
Оператор `if`	✓		✓
Оператор `for`			✓



Результаты

В ходе выполнения работы были достигнуты следующие результаты:

- Проведён обзор существующих решений для генерации модульных тестов для языка Go
- Разработана архитектура инструмента в виде набора взаимодействующих друг с другом утилит и работающих в Docker-контейнере
- Реализован инструмент с использованием различных языков и технологий, таких как C++, Go, Python, Make, Docker
- Проведено тестирование работы реализованного инструмента на различных тестовых Go-пакетах, находящихся в папке `examples` репозитория

Репозиторий:
https://github.com/Software-Analysis-Team/UTBotGo

Использование утилиты **utbotgo**

```
# cd examples/test1
# ls
getSign.go  go.mod  main.go
#
```

Использование утилиты **utbotgo**

```
package main

func main() {
    var a int
    getSign(a)
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
main.go                                1,1           Весь
```

```
package main

func getSign(x int) int {
    if x == 0 {
        return 0
    }
    if x > 0 {
        return 1
    } else {
        return -1
    }
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
getSign.go                             1,1           Весь
```

```
"getSign.go" 12L, 150C
```

Использование утилиты **utbotgo**

```
# cd examples/test1
# ls
getSign.go  go.mod  main.go
# vim -O main.go getSign.go
Файлов для редактирования: 2
# utbotgo init
Initialization continued successfully.
Configuration defined in file `utbotgo/config.yml`.
# ls
getSign.go  go.mod  main.go  utbotgo
#
```

Использование утилиты **utbotgo**

```
tested-functions:
  - getSign
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"utbotgo/config.yml" 2L, 30C записано                         1,1          Весь
```

Использование утилиты **utbotgo**

```
# ls
getSign.go  go.mod  main.go  utbotgo
# vim utbotgo/config.yml
# utbotgo generate > /dev/null
KLEE: output directory is "/workspace/utbotgo/functions/getSign/klee-out-0"
KLEE: Using Z3 solver backend
KLEE: WARNING: undefined reference to function: internal_1cpu..import
KLEE: WARNING: undefined reference to function: runtime..import
KLEE: WARNING: undefined reference to function: runtime.gList.empty
KLEE: WARNING: undefined reference to function: runtime.gList.pop
KLEE: WARNING: undefined reference to function: runtime.gList.push
KLEE: WARNING: undefined reference to function: runtime.gList.pushAll
KLEE: WARNING: undefined reference to function: runtime.goPanicIndex
KLEE: WARNING: undefined reference to function: runtime.memequal
KLEE: WARNING: undefined reference to function: runtime.panicmem
KLEE: WARNING: undefined reference to function: runtime.registerTypeDescriptors
KLEE: WARNING: executable has module level assembly (ignoring)

KLEE: done: total instructions = 43
KLEE: done: completed paths = 3
KLEE: done: partially completed paths = 0
KLEE: done: generated tests = 3
# ls
getSign.go  go.mod  main.go  utbotgo  utbotgo_getSign_test.go
#
```

Использование утилиты **utbotgo**

```
KLEE: done: partially completed paths = 0
KLEE: done: generated tests = 3
# ls
getSign.go  go.mod  main.go  utbotgo  utbotgo_getSign_test.go
# go test
--- FAIL: Test_utbotgo_getSign (0.00s)
    --- FAIL: Test_utbotgo_getSign/test_#1 (0.00s)
        utbotgo_getSign_test.go:71:
            arguments: {-92233720368547758O8}
            expected is unknown
            actual: {-1}
    --- FAIL: Test_utbotgo_getSign/test_#2 (0.00s)
        utbotgo_getSign_test.go:71:
            arguments: {0}
            expected is unknown
            actual: {0}
    --- FAIL: Test_utbotgo_getSign/test_#3 (0.00s)
        utbotgo_getSign_test.go:71:
            arguments: {255}
            expected is unknown
            actual: {1}
FAIL
exit status 1
FAIL    simpleTest3     0.002s
#
```

Использование утилиты **utbotgo**

```
# go test
--- FAIL: Test_utbotgo_getSign (0.00s)
    --- FAIL: Test_utbotgo_getSign/test_#1 (0.00s)
        utbotgo_getSign_test.go:71:
            arguments: {-92233720368547758O8}
            expected is unknown
            actual: {-1}
    --- FAIL: Test_utbotgo_getSign/test_#2 (0.00s)
        utbotgo_getSign_test.go:71:
            arguments: {0}
            expected is unknown
            actual: {0}
    --- FAIL: Test_utbotgo_getSign/test_#3 (0.00s)
        utbotgo_getSign_test.go:71:
            arguments: {255}
            expected is unknown
            actual: {1}
FAIL
exit status 1
FAIL    simpleTest3     0.002s
# utbotgo update_answers > /dev/null
# go test
PASS
ok      simpleTest3     0.003s
#
```

Использование утилиты **utbotgo**

```
package main

func main() {
    var a int
    getSign(a)
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
main.go                                1,1           Весь
```

```
package main

func getSign(x int) int {
    if x == 0 {
        return -2
    }
    if x > 0 {
        return 1
    } else {
        return -1
    }
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
getSign.go                             1,1           Весь
```

```
"getSign.go" 12L, 151C записано
```

Использование утилиты **utbotgo**

```
    --- FAIL: Test_utbotgo_getSign/test_#3 (0.00s)
        utbotgo_getSign_test.go:46:
            arguments: {255}
            expected is unknown
            actual: {1}
FAIL
exit status 1
FAIL    simpleTest3     0.003s
# utbotgo update_answers > /dev/null
# go test
PASS
ok      simpleTest3     0.003s
# vim -O main.go getSign.go
Файлов для редактирования: 2
# go test
--- FAIL: Test_utbotgo_getSign (0.00s)
    --- FAIL: Test_utbotgo_getSign/test_#2 (0.00s)
        utbotgo_getSign_test.go:51:
            arguments: {0}
            expected: {0}
            actual: {-2}
FAIL
exit status 1
FAIL    simpleTest3     0.002s
#
```

Использование утилиты **utbotgo**

```
)

type utbotgo_type_args_getSign struct {
        Ax int `json:"x"`
}

var utbotgo_args_getSign = []utbotgo_type_args_getSign{
        { Ax: -9223372036854775808 },
        { Ax: 0 },
        { Ax: 255 },
}

type utbotgo_type_results_getSign struct {
        R int `json:"R"`
}

var utbotgo_answers_getSign = []utbotgo_type_results_getSign{
        { R: -1 },
        { R: -2 },
        { R: 1 },
}

func utbotgo_writeResults_getSign(results []utbotgo_type_results_getSign) (err error) {
        var data []byte
"utbotgo_getSign_test.go" 61L, 1341C записано                    26,1-8        18%
```

Использование утилиты **utbotgo**

```
            actual: {1}
FAIL
exit status 1
FAIL    simpleTest3     0.003s
# utbotgo update_answers > /dev/null
# go test
PASS
ok      simpleTest3     0.003s
# vim -O main.go getSign.go
Файлов для редактирования: 2
# go test
--- FAIL: Test_utbotgo_getSign (0.00s)
    --- FAIL: Test_utbotgo_getSign/test_#2 (0.00s)
        utbotgo_getSign_test.go:51:
            arguments: {0}
            expected: {0}
            actual: {-2}
FAIL
exit status 1
FAIL    simpleTest3     0.002s
# vim utbotgo_getSign_test.go
# go test
PASS
ok      simpleTest3     0.002s
#
```