CS314. Функциональное программирование

Лекция 19. Тестирование и профилирование кода на Haskell

Направление «Фундаментальная информатика и информационные технологии»
Институт математики, механики и компьютерных наук имени И. И. Воровича
Южный федеральный университет

8 декабря 2017 г.

Содержание

Содержание

Модульное тестирование

- Модульное тестирование (unit testing)
- Основная цель: сокращение числа ошибок
- Разработчик и «белый ящик»
- TDD — test driven development
- testcase (тестовый случай) vs property (свойство)
- Haskell: HUnit и QuickCheck
- Объединение тестов: Tasty (Роман Чепляка)

Установка необходимых библиотек

```
$ cabal install hunit quickcheck
$ cabal install tasty tasty-hunit tasty-quickcheck
```

Содержание

Основные понятия

- Утверждение (assertion): «эта функция для параметра [1,2,3,4,5] должна возвращать пару (1, 5)».
- Тестовый случай (testcase) — это набор последовательно проверяемых утверждений (часто одно утверждение).
- Тест (test suit) — это набор тестовых случаев.
- Утверждения, тестовые случаи и тесты могут именоваться.

Пример

```
import Test.HUnit

test1 = TestCase (assertEqual "(foo 3)" (1,2) (foo 3))

test2 = TestCase (
   do
     (x,y) <- partA 3
     assertEqual "for the first result of partA," 5 x
     b <- partB y
     assertBool ("(partB " ++ show y ++ ") failed") b

tests = TestList [TestLabel "test1" test1,
                  TestLabel "test2" test2]
```

Запуск тестов

```
ghci> runTestTT tests
Cases: 2  Tried: 2  Errors: 0  Failures: 0
```

или

```
ghci> runTestTT tests
### Failure in: 0:test1
(foo 3)
expected: (1,2)
but got: (1,3)
Cases: 2  Tried: 2  Errors: 0  Failures: 1
```

Альтернативный синтаксис для задания тестов

```
tests =
  test ["test1" ~: "(foo 3)" ~: (1,2) ~=? (foo 3),
        "test2" ~:
          do (x, y) <- partA 3
             assertEqual "for the first result of partA," 5 x
             partB y @? "(partB " ++ show y ++ ") failed" ]
```

Содержание

Основные понятия

- Свойство — это предикат с универсально квантифицированными параметрами.
- Свойство — это инвариант для функций или их комбинаций.
- Проверка свойства — это проверка истинности предиката для большого числа случайно сгенерированных параметров.
- Во время проверки могут найтись (или не найтись) контрпримеры.

Пример

```haskell
import Test.QuickCheck

prop_revapp :: [Int] -> [Int] -> Bool
prop_revapp xs ys =
       reverse (xs ++ ys) == reverse xs ++ reverse ys

main = quickCheck prop_revapp
```

```
ghci> main
*** Failed! Falsifiable (after 4 tests and 7 shrinks):
[0]
[1]
```

Исправленная версия примера

```
import Test.QuickCheck

prop_revapp :: [Int] -> [Int] -> Bool
prop_revapp xs ys =
       reverse (xs ++ ys) == reverse ys ++ reverse xs

main = quickCheck prop_revapp
```

```
ghci> main
+++ OK, passed 100 tests.
```

Пример: свойства для функции сортировки (qsort)

Свойство с предусловием

```
prop_minimum' xs =
    not (null xs) ==> head (qsort xs) == minimum xs
```

Тестирование на моделях

```
prop_sort_model xs = sort xs == qsort xs
```

Содержание

Пример (1)

```
import Test.Tasty
import Test.Tasty.QuickCheck
import Test.Tasty.HUnit

import Data.List
import Data.Ord

main = defaultMain tests

tests :: TestTree
tests = testGroup "Tests" [properties, unitTests]
```

Пример (2)

```
properties = testGroup "(checked by QuickCheck)"
  [ testProperty "sort == sort . reverse" $
      \list -> sort (list :: [Int]) == sort (reverse list)
  , testProperty "Fermat's little theorem" $
      \x -> ((x :: Integer)^7 - x) `mod` 7 == 0
  -- некорректное свойство
  , testProperty "Fermat's last theorem" $
      \x y z n ->
        (n :: Integer) >= 3 ==> x^n + y^n /= (z^n :: Integer)
  ]
```

Пример (3)

```
unitTests = testGroup "Unit tests"
  [ testCase "List comparison (different length)" $
      [1, 2, 3] `compare` [1,2] @?= GT

  -- некорректный тест
  , testCase "List comparison (same length)" $
      [1, 2, 3] `compare` [1,2,2] @?= LT
  ]
```

Запуск тестов

```
$ ghc tasty.hs
[1 of 1] Compiling Main             ( tasty.hs, tasty.o )
Linking tasty ...
$ ./tasty
Tests
  (checked by QuickCheck)
    sort == sort . reverse:               OK
      +++ OK, passed 100 tests.
    Fermat's little theorem:              OK
      +++ OK, passed 100 tests.
    Fermat's last theorem:                FAIL
      *** Failed! Falsifiable (after 3 tests):
      3
      0
      3
      3
  Unit tests
    List comparison (different length): OK
    List comparison (same length):      FAIL
      expected: LT
       but got: GT

2 out of 5 tests failed
```

Содержание

Компиляция и запуск с профилированием

```
$ ghc program.hs -rtsopts
...
$ ./program +RTS -sstderr
5000000.5
1,689,133,824 bytes allocated in the heap
697,882,192 bytes copied during GC (scavenged)
465,051,008 bytes copied during GC (not scavenged)
382,705,664 bytes maximum residency (10 sample(s))

       3222 collections in generation 0 (  0.91s)
         10 collections in generation 1 ( 18.69s)

        742 Mb total memory in use

  INIT  time    0.00s  (  0.00s elapsed)
  MUT   time    0.63s  (  0.71s elapsed)
  GC    time   19.60s  ( 20.73s elapsed)
  EXIT  time    0.00s  (  0.00s elapsed)
  Total time   20.23s  ( 21.44s elapsed)

  %GC time      96.9%  (96.7% elapsed)
```

Сохранение результатов профилирования

```
$ ghc program.hs -prof
...
$ ./program +RTS -p
...
$ cat program.prof
...
                                          individual      inherited
COST CENTRE MODULE   no.     entries  %time %alloc   %time %alloc

MAIN        MAIN      1           0   0.0    0.0     100.0  100.0
 main       Main    166           2   0.0    0.0       0.0    0.0
  mean      Main    168           1   0.0    0.0       0.0    0.0
 CAF:sum    Main    160           1  78.6   25.0      78.6   25.0
...
```

Библиотека Criterion: пример программы

```
import Criterion.Main

fib :: Int -> Int
fib 0 = 0
fib 1 = 1
fib n = fib (n-1) + fib (n-2)

main = defaultMain [
        bench "fib 10" $ nf fib 10
      , bench "fib 30" $ nf fib 30
      , bench "fib 35" $ nf fib 35
      ]
```

Результаты испытаний (1)

```
warming up
estimating clock resolution...
mean is 1.065197 us (640001 iterations)
found 1113950 outliers among 639999 samples (174.1%)
  483877 (75.6%) low severe
  630073 (98.4%) high severe
estimating cost of a clock call...
mean is 28.33667 ns (6 iterations)

benchmarking fib 10
mean: 606.9125 ns, lb 604.3461 ns, ub 610.1427 ns, ci 0.950
std dev: 14.68682 ns, lb 11.99081 ns, ub 19.44081 ns, ci 0.950
found 2 outliers among 100 samples (2.0%)
  2 (2.0%) high mild
variance introduced by outliers: 18.037%
variance is moderately inflated by outliers
```

Результаты испытаний (2)

```
benchmarking fib 30
mean: 9.288316 ms, lb 9.258108 ms, ub 9.327903 ms, ci 0.950
std dev: 176.7920 us, lb 142.8436 us, ub 258.2864 us, ci 0.950
found 2 outliers among 100 samples (2.0%)
  1 (1.0%) high severe
variance introduced by outliers: 12.271%
variance is moderately inflated by outliers

benchmarking fib 35
collecting 100 samples, 1 iterations each, in estimated 10.21161 s
mean: 102.5937 ms, lb 102.4473 ms, ub 102.7769 ms, ci 0.950
std dev: 835.2956 us, lb 688.1433 us, ub 1.106325 ms, ci 0.950
```