Лекция 11. Игра COUNTDOWN

Пережогин А.С.

11 октября 2012 г.

Игра COUNTDOWN

Условия игры: заданы несколько чисел, арифметические операции и фиксированное число. Необходимо, используя числа и операции, сгенерировать математическое выражение равно фиксированному числу.

Правила

- Все промежуточные результаты вычислений – это натуральные числа
- Каждое заданное число используется один раз

Пример.

```
1, 3, 7, 10, 25, 50 - числа

+ - * / - арифметические операции

785 - фиксированное число
```

Выигрышная комбинация: (25 - 10)*(50+1) = 765

Число выигрышных комбинаций выражений = 780.

Решение задачи

Решим данную задачу с помощью языка программирования Haskell
Тип данных Op

```
data Op = Add | Sub | Mul | Div
```

Функция apply

```
apply           :: Op -> Int -> Int -> Int
apply Add x y  = x + y
apply Sub x y  = x - y
apply Mul x y  = x * y
apply Div x y  = x `div` y
```

Функция составления максимального числа списков из элементов списка.

```
choices :: [a] -> [[a]]
```

```
values              :: Expr -> [Int]
values (Val n)       = [n]
values (App _ l r) = values l ++ values r
```

Решение задачи

```
solution          :: Expr -> [Int] -> Int -> Bool
solution e ns n = elem (values e) (choices ns)
                    && eval e == [n]
```

Разделение списка элементов на максимальное число двухместных кортежей :

```
split :: [a] -> [([a],[a])]
```

Применения операторов к 2 положительным числам

```
valid           :: Op -> Int -> Int -> Bool
valid Add _ _    = True
valid Sub x y    = x > y
valid Mul _ _    = True
valid Div x y    = x `mod` y == 0
```

Определим тип данных Expr:

```
data Expr = Val Int | App Op Expr Expr
```

eval – завершается успешно и возвращает единственное значение или с ошибкой и возвращает пустой список:

Решение задачи

```
eval             :: Expr -> [Int]
eval (Val n)      = [n | n > 0]
eval (App o l r) = [apply o x y | x <- eval l
                                , y <- eval r
                                , valid o x y]
```

Общее число выражений, которые можно составить из списка чисел:

```
exprs    :: [Int] -> [Expr]
exprs []  = []
exprs [n] = [Val n]
exprs ns  = [e | (ls,rs) <- split ns
               , l       <- exprs ls
               , r       <- exprs rs
               , e       <- combine l r]
```

Комбинация двух выражений с использование всех операций

```
combine    :: Expr -> Expr -> [Expr]
combine l r =
   [App o l r | o <- [Add,Sub,Mul,Div]]
```

Решение задачи

```
solutions      :: [Int] -> Int -> [Expr]
solutions ns n = [e | ns' <- choices ns
                    , e   <- exprs ns'
                    , eval e == [n]]
```

Контрольные задания

- Замерить скорость работы алгоритма для условий из примера