Свободные би-стрелки, или Как генерировать варианты учебных заданий по программированию

Антон Марченко Мансур Зиятдинов

Казанский федеральный университет

05 апреля 2017

Содержание

1. Многовариантные задания
   - Определение и примеры
   - Проблемы генерации многовариантных заданий
2. Библиотека multivariant
   - Представление задания
   - Технические решения

Многовариантные задания

Определение

Многовариантные задания — задания, у которых много вариантов примерно одной сложности

Многовариантные задания

Определение

Многовариантные задания — задания, у которых много вариантов примерно одной сложности

Пример

1. Найти сумму элементов массива
2. Найти произведение элементов массива
3. Найти максимальный элемент массива
4. Найти минимальный элемент массива

Задача

Задача

Найти максимальный среди положительных элементов списка. Если такого нет, вернуть -1.

Как пишет решение преподаватель

```
int task(List<int> input) {
  List<int> positive = filterPos(input);
  if (positive.size() > 0)
    return findMax(input);
  else
    return -1;
}
```

Как пишет решение преподаватель

```
int task(List<int> input) {
  List<int> positive = filterPos(input);
  if (positive.size() > 0)
    return findMax(input);
  else
    return -1;
}

List<int> filterPos(List<int> input) {
  List<int> res = new LinkedList<>();
  for (int elem : input)
    if (elem > 0)
      res.add(elem);
  return res;
}
```

Как пишет решение преподаватель

```
int task(List<int> input) {
  List<int> positive = filterPos(input);
  if (positive.size() > 0)
    return findMax(input);
  else
    return -1;
}

List<int> filterPos(List<int> input) {
  List<int> res = new LinkedList<>();
  for (int elem : input)
    if (elem > 0)
      res.add(elem);
  return res;
}

List<int> findMax(List<int> input) {
  int max = input.get(0);
  for (int elem : input)
    if (elem > max)
      max = elem;
  return max;
}
```

Как пишут решение многие студенты

```
int task(List<int> input) {
  int max;
  for (int elem : input) {
    if (elem > 0 && elem > max) {
      max = elem;
    }
  }
  return max;
}
```

Противоречия

- Нельзя тестировать решение по частям

Противоречия

- Нельзя тестировать решение по частям
- Но готовить задание хочется из маленьких частей

Противоречия

- Нельзя тестировать решение по частям
- Но готовить задание хочется из маленьких частей
- Большой набор заданий необходим, чтобы вести у многих студентов

Противоречия

- Нельзя тестировать решение по частям
- Но готовить задание хочется из маленьких частей
- Большой набор заданий необходим, чтобы вести у многих студентов
- Написание большого количества задач скучно

Противоречия

- Нельзя тестировать решение по частям
- Но готовить задание хочется из маленьких частей
- Большой набор заданий необходим, чтобы вести у многих студентов
- Написание большого количества задач скучно
- Проверка решений должна быть автоматизированной

Противоречия

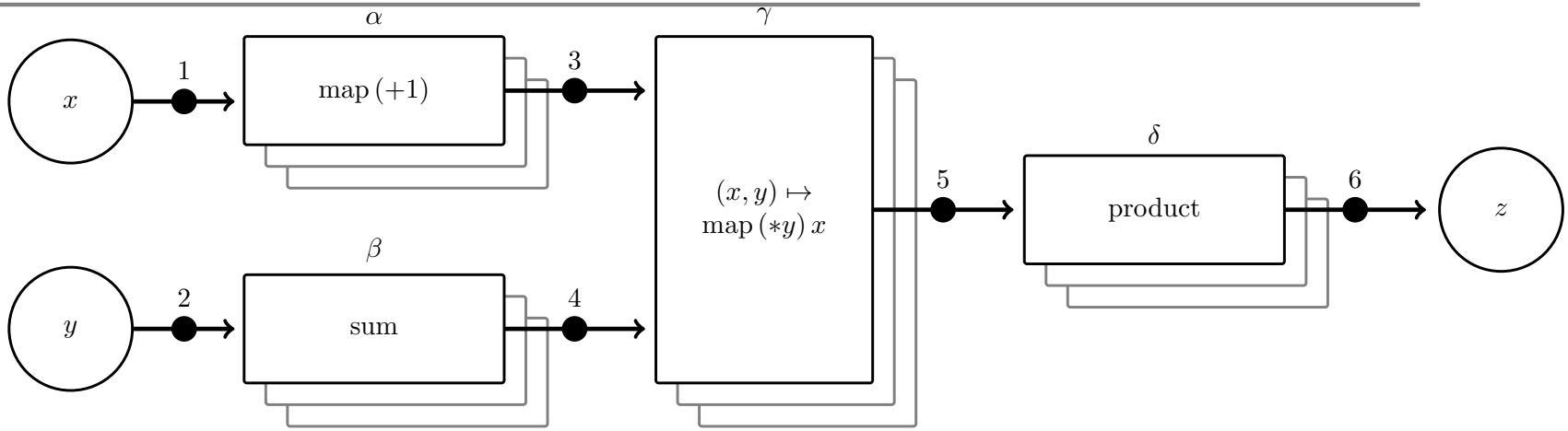
- Нельзя тестировать решение по частям
- Но готовить задание хочется из маленьких частей
- Большой набор заданий необходим, чтобы вести у многих студентов
- Написание большого количества задач скучно
- Проверка решений должна быть автоматизированной
- Тексты задач должны соответствовать тестам и решениям

Содержание

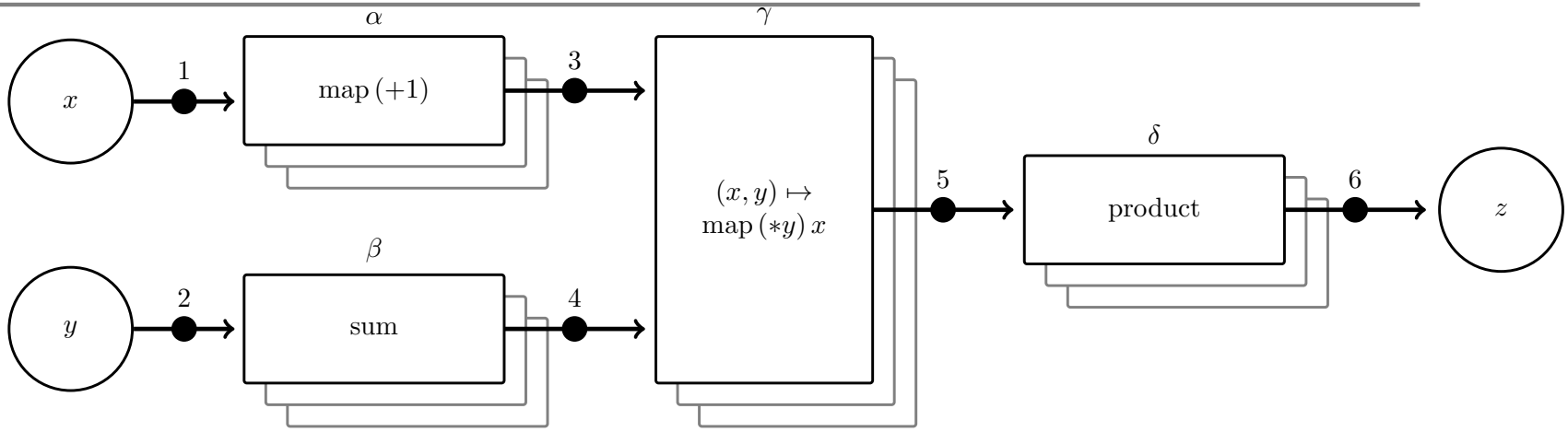
1 Многовариантные задания
- Определение и примеры
- Проблемы генерации многовариантных заданий

2 Библиотека multivariant
- Представление задания
- Технические решения

Схема

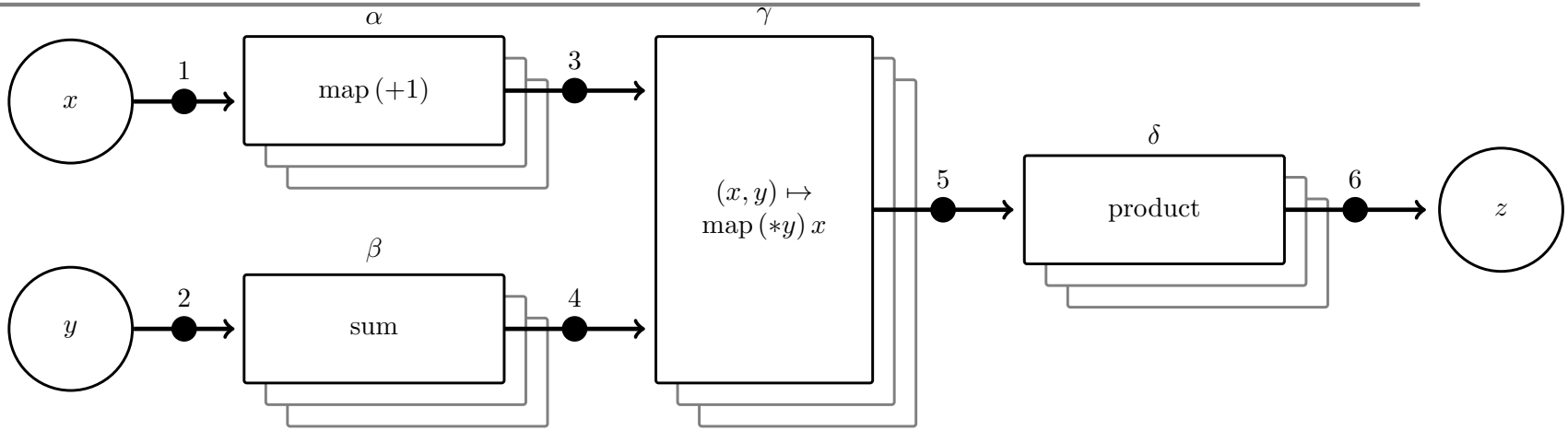

α
γ
x
1
map $(+1)$
3
$(x, y) \mapsto$
map $(*y)\, x$
δ
5
product
6
z
β
y
2
sum
4

Схема



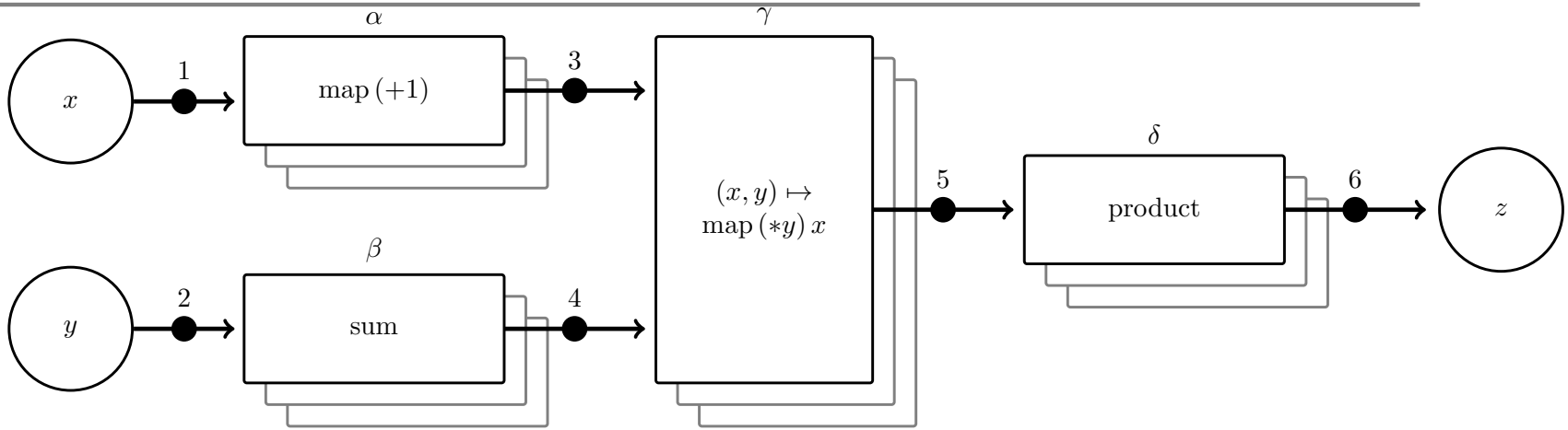
- Свободные — как в *свободной группе*

Схема



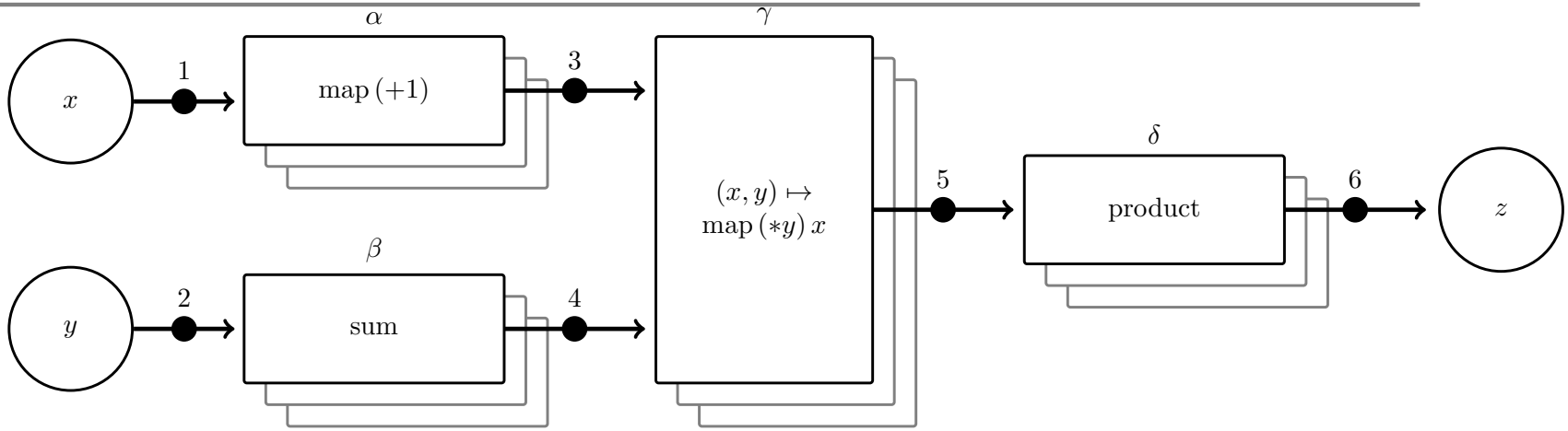
- Свободные — как в *свободной группе*
- *Би*-стрелки — тесты для компонентов нужно преобразовывать в тесты для всей программы

Схема



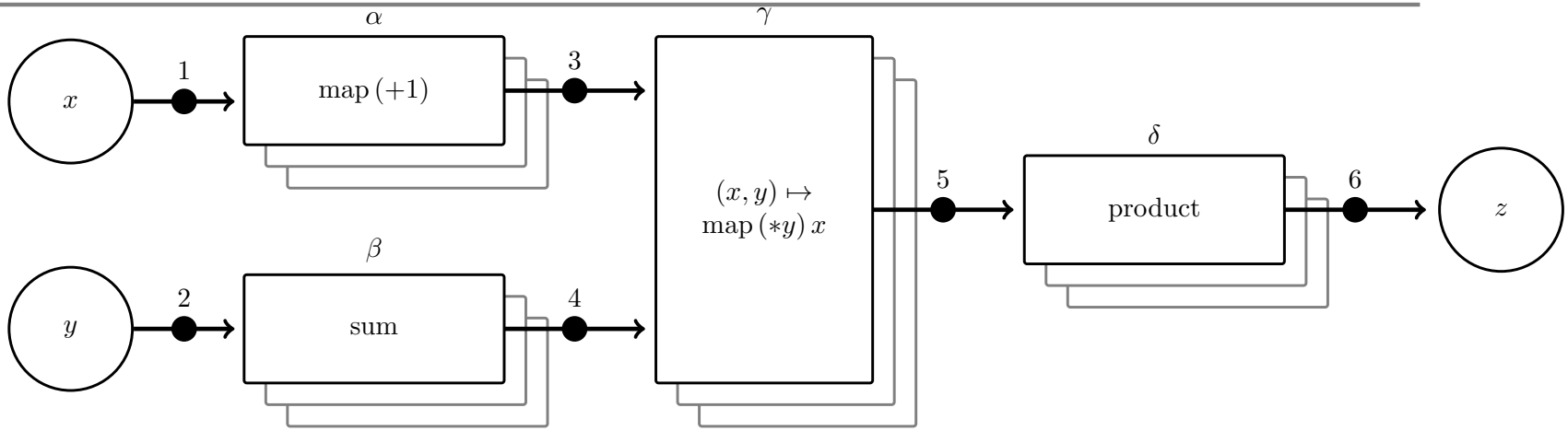
- Свободные — как в *свободной группе*
- *Би*-стрелки — тесты для компонентов нужно преобразовывать в тесты для всей программы
- Би-*стрелки* — необходимо иметь доступ и к типу левой части функции, поэтому монады недостаточно

Схема



- Свободные — как в *свободной группе*
- *Би*-стрелки — тесты для компонентов нужно преобразовывать в тесты для всей программы
- Би-*стрелки* — необходимо иметь доступ и к типу левой части функции, поэтому монады недостаточно
- Бестэговое финальное кодирование — чтобы добавлять команды и интерпретаторы

Схема



- Свободные — как в *свободной группе*
- *Би*-стрелки — тесты для компонентов нужно преобразовывать в тесты для всей программы
- Би-*стрелки* — необходимо иметь доступ и к типу левой части функции, поэтому монады недостаточно
- Бестэговое финальное кодирование — чтобы добавлять команды и интерпретаторы
- Множество функций из Prelude переведены для обратимых вычислений в пакете invertible

Код

```
type P prog a b = (WithDescription prog, WithCornerCases prog)
                => prog a b

alpha :: P prog [Integer] [Integer]
alpha = step (map $ (\x -> x+5) :<->: (\x -> x-5))
        `withCornerCases` ([[],[-1,5],[5,4]],
                           [[0]])
        `withDescription` "Add 5 to each element of the list"

-- ...

delta = delta1 <+++> delta2

task = (alpha <***> beta) ~> gamma ~> delta
```

Обратная функция

$$f \cdot f^{-1} = \mathrm{id}$$

Обратная функция

$$f \cdot f^{-1} = \mathrm{id}$$

```
beta2 = step (P.product :<->: (\p -> [p,1]))
        `withDescription` "Compute␣product␣of␣elements␣of␣list"
        `withCornerCases` ([[],[0]],
                           [1])
```

Спасибо за внимание!

Библиотека multivariant

`https://hackage.haskell.org/package/multivariant`

`https://bitbucket.org/gltronred/multivariant`