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

А.А. Марченко* М.Т. Зиятдинов*

Аннотация

Многовариантные задания могли бы более широко использоваться в учебном процессе, однако процесс их подготовки трудоёмкий, монотонный и рутинный. Мы предлагаем решение для автоматизации этого процесса — язык для описания многовариантных заданий с возможностью генерации постановок задач в текстовом виде, тестовых сценариев для проверки корректности решений и эталонного решения для каждого из вариантов.

Ключевые слова: свободные би-стрелки, бестэговое терминальное кодирование, многовариантные задания.

При обучении программированию часто применяются многовариантные учебные задания одинакового уровня сложности для проведения контрольных работ и отработки у студентов навыков решения задач по определенной теме. Несмотря на все плюсы использования такого рода заданий, их подготовка и последующая проверка решений являются очень трудоемкими, что может сильно ограничивать их применение. Решением проблемы является использование инструментов автоматической генерации многовариантных заданий. Некоторые подходы к генерации заданий рассматривались в работах [5; 6]. Мы предлагаем язык описания учебных заданий на основе свободных би-стрелок и использование интерпретаторов для генерации вариантов.

Стрелки являются моделью вычисления — обобщением монад. Они были введены в статье Hughes [1]. Для более удобной работы со стрелками Paterson [3] предложил расширение синтаксиса языка Haskell. Alimarine и др. [4] ввели понятие би-стрелок для описания обратимых вычислений.

Мы представляем многовариантное задание в виде схемы (см. рис. 1), в котором каждый элемент является набором преобразований. Фиксируя одно преобразование в каждом наборе, мы получаем один вариант задания.

Это позволяет при помощи небольшого количества преобразований получить большое количество вариантов. Текст варианта можно получить,

*Казанский Федеральный университет

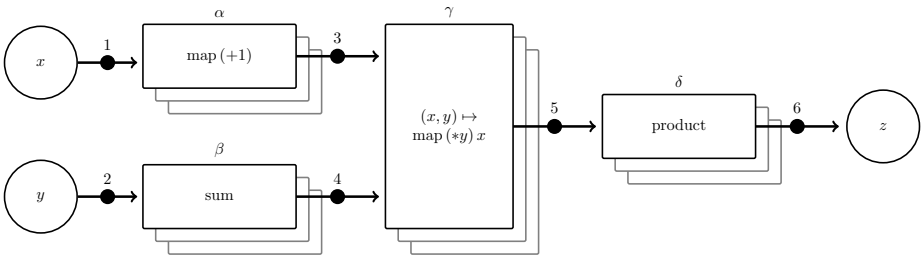

Рис. 1: Задание как схема из наборов преобразований

соединяя тексты отдельных преобразований в топологическом порядке обхода схемы. Аналогично можно получить эталонное решение варианта.

При генерации тестовых сценариев необходимо, чтобы граничные условия отдельных преобразований были учтены в тестовых сценариях для всего варианта в целом, поэтому для каждого элемента задания граничные условия нужно передавать предшествующим и последующим преобразованиям, чтобы получить соответствующие входные и выходные данные. Для этого мы представляем каждое преобразование в виде изоморфизма входных и выходных данных вместе с набором граничных условий и текстом.

Представив схему в виде бестэгового терминального кодирования [2] свободной би-стрелки, мы описываем три её однотипных интерпретатора.

Разработанный пакет программ доступен в репозитории пакетов языка Haskell по ссылке `http://hackage.haskell.org/package/multivariant`.

Список литературы

1. *Hughes J.* Generalizing monads to arrows // Science of Computer Programming. — 2000. — Т. 37, № 1. — С. 67—111. — DOI: `10.1016/S0167-6423(99)00023-4`. — arXiv: `arXiv:1011.1669v3`.
2. *Kiselyov O.* Typed tagless final interpreters // Generic and Indexed Programming. — Springer, 2012. — С. 130—174.
3. *Paterson R.* Arrows and computation // The Fun of Programming. — 2003. — С. 201—222.
4. There and back again / A. Alimarine [и др.] // Proc. of the 2005 ACM SIGPLAN workshop on Haskell. Т. 66. — ACM Press, 2005. — С. 86—97.
5. *Зорин Ю.* Интерпретатор языка построения генераторов тестовых заданий на основе деревьев и/или // Докл. Томского гос. ун-та систем управления и радиоэлектроники. — 2013. — Т. 1(27). — С. 75—79.
6. *Посов И.* Обзор генераторов и методов генерации учебных заданий // Образовательные технологии и общество. — 2014. — Т. 17, № 4.