Вывод второго закона для функтора в Haskell из первого

с помощью свободной теоремы

Кайсин Илья Сергеевич

научный руководитель: Кирилл Елагин

СПб АУ НОЦНТ РАН

20 февраля 2018 г.

Введение

```
fmap :: (a -> b) -> f a -> f b
```

Законы fmap

```
1 fmap id == id
2 fmap (f . g) == fmap f . fmap g
```

Data.Functor

Functors: uniform action over a parameterized type, generalizing the `map` function on lists.

Documentation

```
class Functor f where
```

The `Functor` class is used for types that can be mapped over. Instances of `Functor` should satisfy the following laws:

```
fmap id  ==  id
fmap (f . g)  ==  fmap f . fmap g
```

The instances of `Functor` for lists, `Maybe` and `IO` satisfy these laws.

Minimal complete definition

```
fmap
```

Methods

```
fmap :: (a -> b) -> f a -> f b
```

```
(<$) :: a -> f b -> f a    infixl 4
```

Replace all locations in the input with the same value. The default definition is `fmap . const`, but this may be overridden with a more efficien

Instances

⊞ `Functor []`	# Source	*Since: 2.1*
⊞ `Functor Maybe`	# Source	*Since: 2.1*
⊞ `Functor IO`	# Source	*Since: 2.1*
⊞ `Functor Par1`	# Source	
⊞ `Functor ReadP`	# Source	*Since: 2.1*
⊞ `Functor ReadPrec`	# Source	*Since: 2.1*
⊞ `Functor Last`	# Source	

```
class Functor f where
```

The `Functor` class is used for types that can be mapped over. Instances of `Functor` should satisfy the following laws:

```
fmap id  ==  id
fmap (f . g)  ==  fmap f . fmap g
```

The instances of `Functor` for lists, `Maybe` and `IO` satisfy these laws.

```
class Functor f where
```

The `Functor` class is used for types that can be mapped over. Instances of `Functor` should satisfy the following laws:

```
fmap id  ==  id
```

The instances of `Functor` for lists, `Maybe` and `IO` satisfy these laws.

Введение

- Факт, известный в фольклоре
- Есть попытки: [1] [2]
- Но нет строгих доказательств

Цели

- Формализовать
  - В какой системе типов доказывать?
  - Что такое функтор?
  - Как понимать равенство?
- Доказать
  - Получить свободную теорему для fmap
  - Вывести второй закон из первого
- Опубликовать (PR в GHC)

Свободная теорема. Пример

Parametricity

Теорема, позволяющая делать утверждения о свойствах полиморфной функции, основываясь на её типе. Такие утверждения называются свободными теоремами. [3]

```
r : ∀X. [X] → [X]
```

Тогда для любой $a : X \to Y$

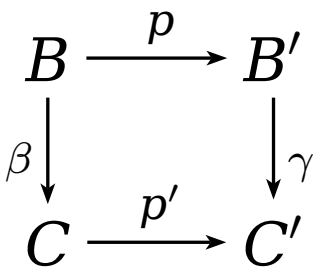
```
(map a) . r = r . (map a)
```

Формализм

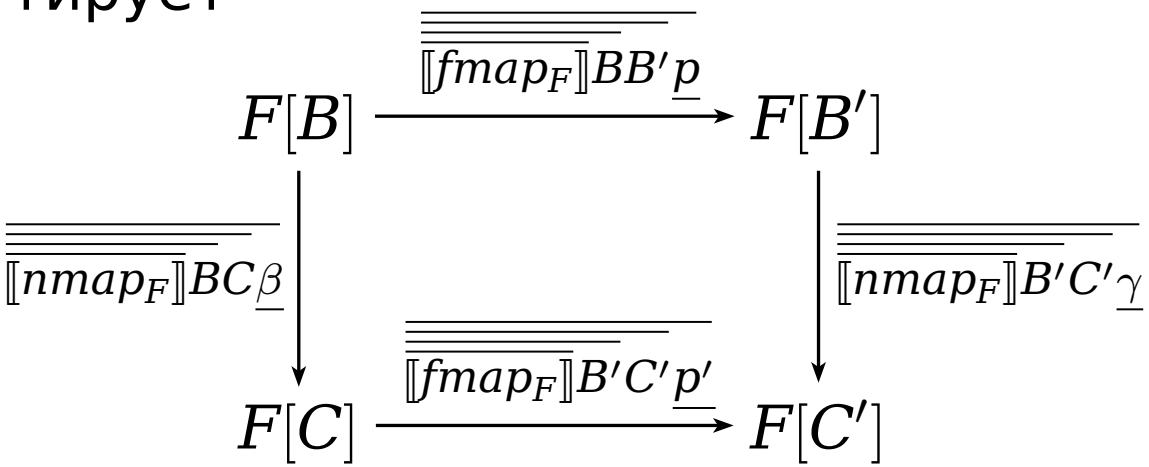
- System F
- Функтор — это тип с плейсхолдером в нужной вариантности
- Frame semantics
- Функциональные термы в конечном итоге интерпретируются как функции, поэтому равенство понимается как равенство функций.

Свободная теорема для fmap

Если коммутирует



то коммутирует



Получение второго закона

```
1 fmap f = nmap id . fmap f =
2          fmap id . nmap f = nmap f
```

```
1 fmap f . fmap g = nmap f . fmap g =
2          fmap id . nmap (f . g) = fmap (f . g)
```

Результаты

- Утверждение доказано
- Построен общий подход к формализации подобных утверждений

Дальнейшие планы

- Адаптировать решение для System FC
- Pull request в GHC
- Попробовать применить к другим операторам над типами

References

[1] Нестрогое доказательство

[2] Ещё одно нестрогое доказательство

[3] Philip Wadler, Theorems for free!