

# Летняя школа в университете Оксфорда 2015

Обобщённое программирование  
и программирование с вычислительными эффектами

А. М. Пеленицын  
[apel@sfedu.ru](mailto:apel@sfedu.ru)

Южный федеральный университет  
Институт математики, механики и компьютерных наук им. И. И. Воровича  
Кафедра информатики и вычислительного эксперимента



# Организаторы

Ральф Хинзе



Джереми Гиббонс



Грант EPSRC, 2012–2015:  
Unifying Theories of Generic Programming

## Публикации

- Conjugate Hylomorphisms, Or: The Mother of All Structured Recursion Schemes  
*Ralf Hinze, Nicolas Wu and Jeremy Gibbons,*  
In POPL 2015. January, 2015.
- Folding Domain–Specific Languages: Deep and Shallow Embeddings  
*Jeremy Gibbons and Nicolas Wu*, 2014.
- Effect Handlers in Scope  
*Nicolas Wu, Tom Schrijvers and Ralf Hinze*  
Haskell Symposium 2014.



# Курсы

- 1 Контейнеры для эффектов и контекстов (Т. Усталу)
- 2 Встроенные предметно-ориентированные языки в Idris (Э. Брэди)
- 3 Обобщенное программирование и программирование на уровне типов в Haskell (А. Лёх)
- 4 Эффективное в худшем случае обобщённое функциональное программирование с массивами данными (Ф. Хенглейн)
- 5 Типы данных типов данных (К. Макбрайд)
- 6 Метапрограммирование времени компиляции в информационном мире (Д. Сайм)



# Курсы

- 1 Контейнеры для эффектов и контекстов (Т. Уусталу)
- 2 Встроенные предметно-ориентированные языки в Idris (Э. Брэди)
- 3 Обобщенное программирование и программирование на уровне типов в Haskell (А. Лёх)
- 4 Эффективное в худшем случае обобщённое функциональное программирование с массивами данными (Ф. Хенглейн)
- 5 Типы данных типов данных (К. Макбрайд)
- 6 Метапрограммирование времени компиляции в информационном мире (Д. Сайм)



# Лектор

Тармо Уусталу



- *Место работы:* Институт кибернетики Талиннского университета технологии.
- *Тема диссертации:* Natural Deduction for Intuitionistic Least and Greatest Fixedpoint Logics, with an Application to Program Construction, 1998, Stockholm.

## Основные публикации

- Type-based termination of recursive definitions, 2004, coauth. Barthe, Fraise, Giménez, Pinto // *Mathematical Structures in Computer Science* 14 (01), pp. 97-141. Cit. 95.
- Iteration and coiteration schemes for higher-order and nested datatypes, 2005, coauth. Abel, Matthes // *Theoretical Computer Science* 333 (1), pp. 3-66. Cit. 53.
- Primitive (Co) Recursion and Course-of-Value (Co) Iteration, Categorically, 1999, coauth. Vene // *Informatica, Lith. Acad. Sci.* 10 (1), pp. 5-26. Cit. 78.



# Лектор

Тармо Уусталу



- *Место работы:* Институт кибернетики Талиннского университета технологии.
- *Тема диссертации:* Natural Deduction for Intuitionistic Least and Greatest Fixedpoint Logics, with an Application to Program Construction, 1998, Stockholm.

## Недавние публикации

- Certified CYK parsing of context-free languages, 2014, coauth. Firsov // *Journal of Logical and Algebraic Methods in Programming* 83 (5), pp. 459-468.
- Dependently typed programming with finite sets, 2015, coauth. Firsov // Proc. of the 11th ACM SIGPLAN Workshop on Generic Programming (WGP 2015), pp. 33–44.



## 1. Контейнеры для эффектов и контекстов (Т. Уусталу)

Oxford University Parks: А.П., Болджа Немет, Денис Фирсов, Антон Черноморд



# Содержание



# Содержание

## Лекция 1. Введение

Монады, отображения монад, дистрибутивные законы.

Примеры: исключения, Читатель, Писатель, списки в ассортименте.



# Содержание

## Лекция 1. Введение

Монады, отображения монад, дистрибутивные законы.

Примеры: исключения, Читатель, Писатель, списки в ассортименте.

## Лекция 2. Контейнеры, монадические контейнеры

- Контейнер: множество «форм»  $S$  и позиции:  $P: S \rightarrow \mathbf{Set}$ .



# Содержание

## Лекция 1. Введение

Монады, отображения монад, дистрибутивные законы.

Примеры: исключения, Читатель, Писатель, списки в ассортименте.

## Лекция 2. Контейнеры, монадические контейнеры

- Контейнер: множество «форм»  $S$  и позиции:  $P: S \rightarrow \mathbf{Set}$ .
- Пример: список,  $S = \mathbb{N}$ ,  $Ps = [0, s]_{\mathbb{N}}$ .



# Содержание

## Лекция 1. Введение

Монады, отображения монад, дистрибутивные законы.

Примеры: исключения, Читатель, Писатель, списки в ассортименте.

## Лекция 2. Контейнеры, монадические контейнеры

- Контейнер: множество «форм»  $S$  и позиции:  $P: S \rightarrow \mathbf{Set}$ .
- Пример: список,  $S = \mathbb{N}$ ,  $Ps = [0, s]_{\mathbb{N}}$ .
- Интерпретация  $\llbracket S, P \rrbracket$  контейнера это функтор  $F: \mathbf{Set} \rightarrow \mathbf{Set}$ :

$$FX = \{Ps \rightarrow X\}_{\{s \in S\}}$$



# Содержание

## Лекция 1. Введение

Монады, отображения монад, дистрибутивные законы.

Примеры: исключения, Читатель, Писатель, списки в ассортименте.

## Лекция 2. Контейнеры, монадические контейнеры

- Контейнер: множество «форм»  $S$  и позиции:  $P: S \rightarrow \mathbf{Set}$ .
- Пример: список,  $S = \mathbb{N}$ ,  $Ps = [0, s]_{\mathbb{N}}$ .
- Интерпретация  $\llbracket S, P \rrbracket$  контейнера это функтор  $F: \mathbf{Set} \rightarrow \mathbf{Set}$ :

$$FX = \{Ps \rightarrow X\}_{\{s \in S\}}$$

- Интерпретация списка:  $FX = \text{List}X = \{[0, s) \rightarrow X\}_{\{s \in \mathbb{N}\}}$ .



# Содержание

## Лекция 1. Введение

Монады, отображения монад, дистрибутивные законы.

Примеры: исключения, Читатель, Писатель, списки в ассортименте.

## Лекция 2. Контейнеры, монадические контейнеры

- Контейнер: множество «форм»  $S$  и позиции:  $P: S \rightarrow \mathbf{Set}$ .
- Пример: список,  $S = \mathbb{N}$ ,  $Ps = [0, s]_{\mathbb{N}}$ .
- Интерпретация  $\llbracket S, P \rrbracket$  контейнера это функтор  $F: \mathbf{Set} \rightarrow \mathbf{Set}$ :

$$FX = \{Ps \rightarrow X\}_{\{s \in S\}}$$

- Интерпретация списка:  $FX = \text{List}X = \{[0, s) \rightarrow X\}_{\{s \in \mathbb{N}\}}$ .



# Содержание

## Лекция 1. Введение

Монады, отображения монад, дистрибутивные законы.

Примеры: исключения, Читатель, Писатель, списки в ассортименте.

## Лекция 2. Контейнеры, монадические контейнеры

- Контейнер: множество «форм»  $S$  и позиции:  $P: S \rightarrow \mathbf{Set}$ .
- Пример: список,  $S = \mathbb{N}$ ,  $Ps = [0, s]_{\mathbb{N}}$ .
- Интерпретация  $\llbracket S, P \rrbracket$  контейнера это функтор  $F: \mathbf{Set} \rightarrow \mathbf{Set}$ :

$$FX = \{Ps \rightarrow X\}_{\{s \in S\}}$$

- Интерпретация списка:  $FX = \text{List}X = \{[0, s) \rightarrow X\}_{\{s \in \mathbb{N}\}}$ .

## Лекция 3. Моноидальные функторы, комонады

Направленный контейнер  $\rightsquigarrow$  комонада («продвинутый» функтор).



## Ссылки

*Abbott et al. Containers. Constructing Strictly Positive Types, 2005*

*One of the main applications of containers is **generic programming**: our representation gives a convenient way to program with or reason about datatypes and polymorphic functions.*

- *Abbott et al. Containers. Constructing Strictly Positive Types, 2005*
- *Ahman et al. When is a container a comonad? 2014*



# Курсы

- 1 Контейнеры для эффектов и контекстов (Т. Уусталу)
- 2 Встроенные предметно-ориентированные языки в Idris (Э. Брэди)
- 3 Обобщенное программирование и программирование на уровне типов в Haskell (А. Лёх)
- 4 Эффективное в худшем случае обобщённое функциональное программирование с массивами данными (Ф. Хенглейн)
- 5 Типы данных типов данных (К. Макбрайд)
- 6 Метапрограммирование времени компиляции в информационном мире (Д. Сайм)



# Лектор

Эдвин Брэди



- *Место работы:*  
Университет Сэнт-Эндрюса, Шотландия.
- *Тема диссертации:* Practical Implementation of a Dependently Typed Functional Programming Language, 2005, Durham U.

## Основные публикации

- Inductive families need not store their indices, 2004, coauth. McBride, McKinna // *Types for proofs and programs*, pp. 115-129. Cit. 73.
- Idris, a general-purpose dependently typed programming language: Design and implementation, 2013 // *Journal of Functional Programming* 23 (05). Cit. 60.
- IDRIS – systems programming meets full dependent types, 2011 // ACM ws. on *Programming languages meets program verification*, pp. 43-54. Cit. 51.



# Лектор

Эдвин Брэди



- *Место работы:*  
Университет Сэнт-Эндрюса, Шотландия.
- *Тема диссертации:* Practical Implementation of a Dependently Typed Functional Programming Language, 2005, Durham U.

## Недавние публикации

- Cross-platform Compilers for Functional Languages, 2015 // Under consideration for *Trends in Functional Programming*.
- Resource-Dependent Algebraic Effects, 2015 // *Trends in Functional Programming*, pp. 18-33.



# Содержание

## Лекция 1. Знакомство с Idris

Idris is a **Pac-man Complete** functional programming language with dependent types.



# Содержание

## Лекция 1. Знакомство с Idris

Idris is a **Pac-man Complete** functional programming language with dependent types.

## Лекция 2. EDSLs на Idris

Сахар для определения DSL, сертификация, Error Reflection, Uniqueness Types.



# Содержание

## Лекция 1. Знакомство с Idris

Idris is a **Pac-man Complete** functional programming language with dependent types.

## Лекция 2. EDSLs на Idris

Сахар для определения DSL, сертификация, Error Reflection, Uniqueness Types.

## Лекция 3. EDSLs с эффектами

Проблема комбинирования эффектов и DSL Effect для её решения.



# Idris и её система типов

## Факты об Idris

- 1 Установка: cabal install idris.
- 2 Компилируемый (в C, JavaScript, ...), оптимизация.
- 3 Синтаксис похож на Хаскель: классы типов, do-блоки, idiom brackets, GADTs.
- 4 Строгая семантика, но есть тип Lazy.
- 5 Интерактивное редактирование.
- 6 Внешние вызовы.



# Idris и её система типов

## Факты об Idris

- 1 Установка: cabal install idris.
- 2 Компилируемый (в C, JavaScript, ...), оптимизация.
- 3 Синтаксис похож на Хаскель: классы типов, do-блоки, idiom brackets, GADTs.
- 4 Строгая семантика, но есть тип Lazy.
- 5 Интерактивное редактирование.
- 6 Внешние вызовы.

## Зависимые типы

- Более строгая спецификация свойств.
- Больше помощи от компилятора.
- Типы являются объектами первого класса в языке.



## Пример: интерпретатор STLC (1)



## Пример: интерпретатор STLC (1)

### Simply Typed Lambda Calculus

- типы: `Int`, `Bool`, функциональные;
- термы: целые литералы, `BinOp`, `If`, лямбда-абстракция, апплексия.



# Пример: интерпретатор STLC (1)

## Simply Typed Lambda Calculus

- типы: Int, Bool, функциональные;
- термы: целые литералы, BinOp, If, лямбда-абстракция, апплексия.

## Типы объектного языка

```
data Ty = TyInt
        | TyBool
        | TyFun Ty Ty
```



# Пример: интерпретатор STLC (1)

## Simply Typed Lambda Calculus

- типы: Int, Bool, функциональные;
- термы: целые литералы, BinOp, If, лямбда-абстракция, апплексия.

## Типы объектного языка

```
data Ty = TyInt
        | TyBool
        | TyFun Ty Ty
```

## Интерпретация типов

```
interpTy : Ty -> Type
interpTy TyInt = Int
interpTy TyBool = Bool
interpTy (TyFun s t) =
    interpTy s -> interpTy t
```



# Пример: интерпретатор STLC (1)

## Simply Typed Lambda Calculus

- типы: Int, Bool, функциональные;
- термы: целые литералы, BinOp, If, лямбда-абстракция, апплексия.

## Типы объектного языка

```
data Ty = TyInt
        | TyBool
        | TyFun Ty Ty
```

## Интерпретация типов

```
interpTy : Ty -> Type
interpTy TyInt = Int
interpTy TyBool = Bool
interpTy (TyFun s t) =
    interpTy s -> interpTy t
```

## Контекст

```
data Env : Vect n Ty -> Type where
    Nil   : Env []
    (:)  : interpTy a
                      -> Env g
                      -> Env (a :: g)
```



## Пример: интерпретатор STLC (2)

Тип для выражения на языке STLC



# Пример: интерпретатор STLC (2)

Тип для выражения на языке STLC

```
data Expr : Vect n Ty -> Ty -> Type where
  Var : HasType i g t           -> Expr g t
  Val : Int                      -> Expr g TyInt
  Lam : Expr (a :: g) t         -> Expr g (TyFun a t)
  App : Expr g (TyFun a t) -> Expr g a -> Expr g t
  Op  : (interpTy a -> interpTy b -> interpTy c) ->
        Expr g a -> Expr g b -> Expr g c
  If  : Expr g TyBool -> Expr g a -> Expr g a -> Expr g a
```



# Пример: интерпретатор STLC (2)

Тип для выражения на языке STLC

```
data Expr : Vect n Ty -> Ty -> Type where
  Var : HasType i g t           -> Expr g t
  Val : Int                      -> Expr g TyInt
  Lam : Expr (a :: g) t         -> Expr g (TyFun a t)
  App : Expr g (TyFun a t) -> Expr g a -> Expr g t
  Op  : (interpTy a -> interpTy b -> interpTy c) ->
        Expr g a -> Expr g b -> Expr g c
  If  : Expr g TyBool -> Expr g a -> Expr g a -> Expr g a
```

Интерпретация выражения на языке STLC

# Пример: интерпретатор STLC (2)

Тип для выражения на языке STLC

```
data Expr : Vect n Ty -> Ty -> Type where
  Var : HasType i g t           -> Expr g t
  Val : Int                      -> Expr g TyInt
  Lam : Expr (a :: g) t         -> Expr g (TyFun a t)
  App : Expr g (TyFun a t) -> Expr g a -> Expr g t
  Op  : (interpTy a -> interpTy b -> interpTy c) ->
        Expr g a -> Expr g b -> Expr g c
  If  : Expr g TyBool -> Expr g a -> Expr g a -> Expr g a
```

Интерпретация выражения на языке STLC

```
interp : Env g -> Expr g t -> interpTy t
interp env (Var i)      = mylookup i env
interp env (Val x)      = x
interp env (Lam sc)     = \x => interp (x :: env) sc
interp env (App f s)    = interp env f (interp env s)
interp env (Op op x y) = op (interp env x) (interp env y)
interp env (If x t e)   = if interp env x then interp env t ...
```

## Пример: интерпретатор STLC (3)

Выражение на языке STLC



## Пример: интерпретатор STLC (3)

Выражение на языке STLC

```
f : Expr g (TyFun TyInt TyInt)
f = Lam (If (Op (==)
                (Var Stop) (Val 0))
            (Val 1)
            (Op (*)
                (App f (Op (-)
                            (Var Stop)
                            (Val 1)))
                (Var Stop))))
```



# Пример: интерпретатор STLC (3)

## Выражение на языке STLC

```
f : Expr g (TyFun TyInt TyInt)
f = Lam (If (Op (==)
                  (Var Stop) (Val 0))
            (Val 1)
            (Op (*)
                  (App f (Op (-)
                                (Var Stop)
                                (Val 1)))
                  (Var Stop))))
```

## Нотация dsl

```
dsl expr
lambda = Lam
variable = Var
index_first = Stop
index_next = Pop
```



# Пример: интерпретатор STLC (3)

## Выражение на языке STLC

```
f : Expr g (TyFun TyInt TyInt)
f = Lam (If (Op (==)
                 (Var Stop) (Val 0))
           (Val 1)
           (Op (*)
                 (App f (Op (-)
                               (Var Stop)
                               (Val 1)))
                 (Var Stop))))
```

## Нотация dsl

```
dsl expr
lambda = Lam
variable = Var
index_first = Stop
index_next = Pop
```

## DSL-выражение на языке STLC

```
fact : Expr g (TyFun TyInt TyInt)
fact = expr (\x => If (Op (==) x (Val 0))
              (Val 1)
              (Op (*) (App fact (Op (-) x (Val 1))) x))
```



## Пример: интерпретатор STLC (4)

Неявный instance Applicative

```
(<$>) : (f : Expr g (TyFun a t)) -> Expr g a -> Expr g t
(<$>) = App
```

```
pure : Expr g a -> Expr g a
pure = id
```

Idiom brackets

```
fact = expr (\x => If (Op (==) x (Val 0))
              (Val 1)
              (Op (*) [I fact (Op (-) x (Val 1)) I] x))
```



## Ссылки

- 1 ГитХаб-аккаунт Э. Брэди.
- 2 Конспект лекций.
- 3 Эрланг-бэкенд для Idris, бакалаврская работа.
- 4 Type-Driven Development with Idris, изд-во Manning.



# Курсы

- 1 Контейнеры для эффектов и контекстов (Т. Уусталу)
- 2 Встроенные предметно-ориентированные языки в Idris (Э. Брэди)
- 3 Обобщенное программирование и программирование на уровне типов в Haskell (А. Лёх)
- 4 Эффективное в худшем случае обобщённое функциональное программирование с массивами данными (Ф. Хенглейн)
- 5 Типы данных типов данных (К. Макбрайд)
- 6 Метапрограммирование времени компиляции в информационном мире (Д. Сайм)



# Лектор

Андрес Лёх



- *Место работы:* компания Well-Typed.
- *Тема диссертации:* Exploring Generic Haskell. 2004, Utrecht U.

## Основные публикации

- Type-indexed data types, 2002, coauth. Hinze, Jeuring // *Mathematics of Program Construction*, pp. 148-174. Cit. 118.
- “Scrap Your Boilerplate” Reloaded, 2006, coauth. Hinze, Oliveira // *Functional and Logic Programming*, pp. 13-29. Cit. 77.
- Generic Haskell, Specifically, 2003, coauth. Clarke // *Generic Programming*, pp. 21-47. Cit. 76.



# Лектор

Андрес Лёх



- *Место работы:* компания Well-Typed.
- *Тема диссертации:* Exploring Generic Haskell. 2004, Utrecht U.

## Недавние публикации

- The semantics of version control, 2014, coauth. Swierstra // Proceedings of the 2014 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software, pp. 43-54.
- Type-level web APIs with servant: an exercise in domain-specific generic programming, 2015, coauth... // Workshop on Generic Programming, pp. 1–12.
- Generic generic programming, 2014, coauth. Magalhaes // Practical Aspects of Declarative Languages, pp. 216-231.



# Содержание

Лекция 1. Обобщённое программирование.  $N$ -арные произведения



# Содержание

## Лекция 1. Обобщённое программирование. N-арные произведения

```
class Generic a where
    type Rep a
    from :: a -> Rep a
    to :: Rep a -> a
```



# Содержание

## Лекция 1. Обобщённое программирование. N-арные произведения

```
class Generic a where
    type Rep a
    from :: a -> Rep a
    to :: Rep a -> a
```

Терм:

(I 1 :\* I "Hi!" :\* Nil)  
типа NP I '[Integer, [Char]] .



# Содержание

## Лекция 1. Обобщённое программирование. N-арные произведения

```
class Generic a where
    type Rep a
    from :: a -> Rep a
    to :: Rep a -> a
```

Терм:

$(I\ 1\ :*\ I\ "Hi!"\ :*\ Nil)$   
типа  $NP\ I\ '[Integer,\ [Char]]$ .

## Лекция 2. Ограниченнное обобщённое программирование. N-арные суммы

Тип  $type\ ExampleChoice = NS\ I\ '[Char,\ Bool,\ Int]$ , термы:

$c_0, c_1, c_2 :: ExampleChoice$

$c_0 = Z\ (I\ 'x')$

$c_1 = S\ (Z\ (I\ True))$

$c_2 = S\ (S\ (Z\ (I\ 3)))$



# Содержание

## Лекция 1. Обобщённое программирование. N-арные произведения

```
class Generic a where
    type Rep a
    from :: a -> Rep a
    to :: Rep a -> a
```

Терм:

$(I\ 1\ :*\ I\ "Hi!"\ :*\ Nil)$   
типа  $NP\ I\ '[Integer,\ [Char]]$ .

## Лекция 2. Ограниченнное обобщённое программирование. N-арные суммы

Тип  $type\ ExampleChoice = NS\ I\ '[Char,\ Bool,\ Int]$ , термы:

```
c0, c1, c2 :: ExampleChoice
c0 = Z (I 'x')
c1 = S (Z (I True))
c2 = S (S (Z (I 3)))
```

## Лекция 3. Суммы произведений. Примеры. Метаинформация



# Пример: тип арифметического выражения

АТД арифметического выражения

```
data Expr =  
    | NumL Int  
    | BoolL Bool  
    | Add Expr Expr  
    | If Expr Expr Expr
```



# Пример: тип арифметического выражения

АТД арифметического выражения

```
data Expr =
| NumL Int
| BoolL Bool
| Add Expr Expr
| If Expr Expr Expr
```

Его обобщённый вид

```
type RepExpr = NS (NP I) ('
[',[Int]
,[Bool]
,[Expr, Expr]
,[Expr, Expr, Expr]
])
```



# Пример: тип арифметического выражения

АТД арифметического выражения

```
data Expr =
| NumL Int
| BoolL Bool
| Add Expr Expr
| If Expr Expr Expr
```

Его обобщённый вид

```
type RepExpr = NS (NP I) (,
[',[Int]
,[Bool]
,[Expr, Expr]
,[Expr, Expr, Expr]
])
```

Бонус: определение NP

```
data NP (f :: k -> *) (xs :: [k]) where
Nil :: NP f []
(:*) :: f x -> NP f xs -> NP f (x ': xs)
```



# Ссылки

## Применение

- генерация тестовых данных (`Generics.SOP.Arbitrary` из [basic-sop](#)),
- претти-принтинг ([pretty-sop](#)),
- генерация JSON/XML-представлений ([json-sop](#)).



# Ссылки

## Применение

- генерация тестовых данных (`Generics.SOP.Arbitrary` из [basic-sop](#)),
  - претти-принтинг ([pretty-sop](#)),
  - генерация JSON/XML-представлений ([json-sop](#)).
- 
- [WGP 2014: Andres Löh – True Sums of Products](#) – получасовой доклад, в котором излагается весь материал курса.
  - Отличный конспект лекций, выданный лектором.



# Ссылки

## Применение

- генерация тестовых данных (`Generics.SOP.Arbitrary` из [basic-sop](#)),
  - претти-принтинг ([pretty-sop](#)),
  - генерация JSON/XML-представлений ([json-sop](#)).
- 
- [WGP 2014: Andres Löh – True Sums of Products](#) – получасовой доклад, в котором излагается весь материал курса.
  - [Отличный конспект лекций, выданный лектором.](#)

## PS

- Generic generic programming;



# Ссылки

## Применение

- генерация тестовых данных (`Generics.SOP.Arbitrary` из [basic-sop](#)),
  - претти-принтинг ([pretty-sop](#)),
  - генерация JSON/XML-представлений ([json-sop](#)).
- 
- [WGP 2014: Andres Löh – True Sums of Products](#) – получасовой доклад, в котором излагается весь материал курса.
  - Отличный конспект лекций, выданный лектором.

## PS

- Generic generic programming;
- наблюдение: никто на планете не умеет generalized datatype generic programming.



# Курсы

- 1 Контейнеры для эффектов и контекстов (Т. Уусталу)
- 2 Встроенные предметно-ориентированные языки в Idris (Э. Брэди)
- 3 Обобщенное программирование и программирование на уровне типов в Haskell (А. Лёх)
- 4 Эффективное в худшем случае обобщённое функциональное программирование с массивами данными (Ф. Хенглейн)
- 5 Типы данных типов данных (К. Макбрайд)
- 6 Метапрограммирование времени компиляции в информационном мире (Д. Сайм)



# Лектор

Фриц Хенглейн



- *Место работы:* Университет Копенгагена, Дания.
- *Тема диссертации:* Polymorphic Type Inference and Semi-Unification, 1989, Rutgers U.

## Основные публикации

- Type inference with polymorphic recursion, 1993 // *ACM Transactions on Programming Languages and Systems* 15 (2), pp. 253-289. Cit. 288.
- Dynamic typing: Syntax and proof theory, 1994 // *Science of Computer Programming* 22 (3), pp. 197-230. Cit. 168.
- Efficient type inference for higher-order binding-time analysis, 1991 // *Functional Programming Languages and Computer Architecture*, pp. 448-472. Cit. 153.



# Лектор

Фриц Хенглейн



- *Место работы:* Университет Копенгагена, Дания.
- *Тема диссертации:* Polymorphic Type Inference and Semi-Unification, 1989, Rutgers U.

## Недавние публикации

- Domain-specific languages for enterprise systems, 2014 // *Leveraging Applications of Formal Methods, Verification and Validation*, pp. 73-95.
- Optimally Streaming Greedy Regular Expression Parsing, 2014, coauth. Grathwohl, Rasmussen // *Theoretical Aspects of Computing*, pp. 224-240.



# Университет Южной Дании (Экс-Одерсе)



UNIVERSITY OF SOUTHERN DENMARK

Contact  
Directory  
Login

Vacant positions  
Sitemap  
 Dansk

PROGRAMMES

LIFELONG LEARNING

RESEARCH

NEWS

COOPERATION

LIBRARY

ABOUT



*Are you tomorrow's researcher?*



You can still make it ...  
We have master programmes with  
available study places



# Содержание

## Лекция 1. Обобщённое дискриминирование, композиционные порядки

Сортировка	Основана на сравнениях	Дистрибутивная
Порядок фиксирован	Quicksort, Mergesort etc.	Bucketsort, Radixsort, Counting sort
Порядок – параметр	Quicksort, Mergesort etc. с компаратором	

Аналогично – для разбиения на классы эквивалентности.



# Содержание

## Лекция 1. Обобщённое дискриминирование, композиционные порядки

Сортировка	Основана на сравнениях	Дистрибутивная
Порядок фиксирован	Quicksort, Mergesort etc.	Bucketsort, Radixsort, Counting sort
Порядок – параметр	Quicksort, Mergesort etc. с компаратором	

Аналогично – для разбиения на классы эквивалентности.

## Лекция 2. Обобщённое программирование с мульти множествами

Символические декартовы произведения и INNER JOINs на основе обобщённого дискриминирования.



# Содержание

## Лекция 1. Обобщённое дискриминирование, композиционные порядки

Сортировка	Основана на сравнениях	Дистрибутивная
Порядок фиксирован	Quicksort, Mergesort etc.	Bucketsort, Radixsort, Counting sort
Порядок – параметр	Quicksort, Mergesort etc. с компаратором	

Аналогично – для разбиения на классы эквивалентности.

## Лекция 2. Обобщённое программирование с мульти множествами

Символические декартовы произведения и INNER JOINs на основе обобщённого дискриминирования.

## Лекция 3. Обобщённое программирование линейной алгебры

От функций  $a \rightarrow [0, \infty)_\mathbb{N}$  (мультимножества) к  $a \rightarrow \text{Floating}$ .



# Основные результаты лекции 1

1 disc бьёт nub:

length	(reps eqInt32 [1..1000000])	1.5 сек
length	(nub [1..1000000])	1.5 часа

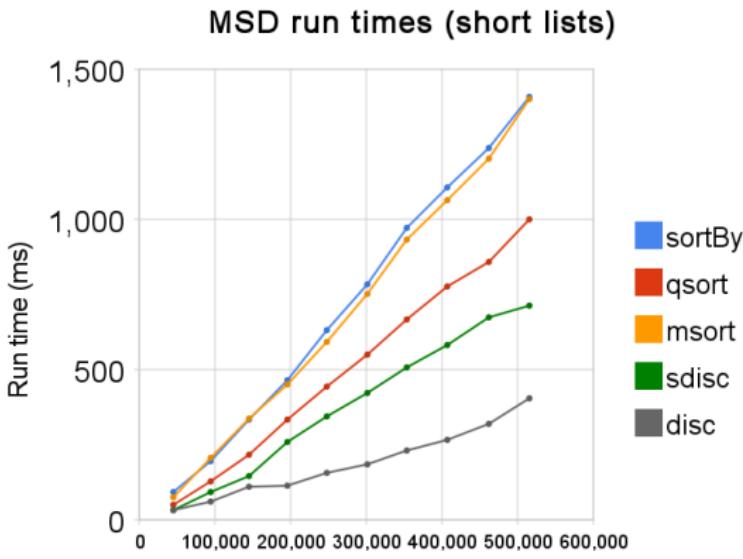


# Основные результаты лекции 1

1 disc бьёт nub:

length (reps eqInt32 [1..1000000])	1.5 сек
length (nub [1..1000000])	1.5 часа

2 disc и sdisc сравнимы с общими алгоритмами сортировки:



# Примеры (1)

Композиционное определение порядка

```
data Order t where
  Nat0 :: Int → Order Int
  Triv0 :: Order t
  SumL :: Order t1 → Order t2 → Order (Either t1 t2)
  ProdL :: Order t1 → Order t2 → Order (t1, t2)
  Map0 :: (t1 → t2) → Order t2 → Order t1
  ListL :: Order t → Order [t]
  Bag0 :: Order t → Order [t]
  Set0 :: Order t → Order [t]
  Inv :: Order t → Order t
```

Аналогично определяется эквивалентность.



# Примеры (2)

## Дискриминирующая функция

```

type Disc k = forall v. [(k, v)] → [[v]]
sdisc :: Order k → Disc k
sdisc [] = []
sdisc [_ , v] = [[v]]
sdisc (Nat0 n) xs = sdiscNat n xs           -- Bucket Sort
sdisc (Triv0) xs = [[ v | (_ , v) ← xs ]]
sdisc (SumL r1 r2) xs = sdisc r1 [ (k, v) | (Left k, v) ← xs ]
                     ++ sdisc r2 [ (k, v) | (Right k, v) ← xs ]
sdisc (ProdL r1 r2) xs = [ vs |
                           ys ← sdisc r1 [ (k1, (k2, v)) | ((k1, k2), v) ← xs ],
                           vs ← sdisc r2 ys ]
sdisc (Map0 f r) xs = sdisc r [ (f k, v) | (k, v) ← xs ]
...

```



## Бонус: личный опыт

Проблема Эйлера 75



# Бонус: личный опыт

## Проблема Эйлера 75

```
-- Naive impl, 2.2 sec      -- IntMap, 2.4 sec
length                      IM.size
  . filter ((== 1) . length)
  . group
  . sort
                                . IM.filter (== 1)
                                . IM.fromListWith (+)
                                . flip zip (repeat 1)

-- Generic                     -- Array, 2.6 sec
-- Discrimination, 5.6 sec    length
length                      . filter (\(n, ari) -> ari == 1)
  . filter ((== 1) . length)
  . Disc.group
                                . assocs
                                . arities -- (*)
```

## (\*) arities

```
arities xs = accumArray (+) 0
                    (0, maximum xs)
                    (zip xs (repeat 1))
```



## Ссылки

- Generic Top-down Discrimination for Sorting and Partitioning in Linear Time
- Generic Multiset Programming with Discrimination-based Joins and Symbolic Cartesian Products
- Библиотека [discrimination](#), автор: Эдвард Кметт. В описании явно указано, что теоретическую основу составляют работы проф. Хенглейна.



# Курсы

- 1 Контейнеры для эффектов и контекстов (Т. Усталу)
- 2 Встроенные предметно-ориентированные языки в Idris (Э. Брэди)
- 3 Обобщенное программирование и программирование на уровне типов в Haskell (А. Лёх)
- 4 Эффективное в худшем случае обобщённое функциональное программирование с массивами данными (Ф. Хенглейн)
- 5 Типы данных типов данных (К. Макбрайд)
- 6 Метапрограммирование времени компиляции в информационном мире (Д. Сайм)



# Лектор

Конор Макбрайд



- *Место работы:* Университет Стрэтклайда, Шотландия.
- *Тема диссертации:* Dependently Typed Functional Programs and their Proofs, 1999, U. of Edinburg.

## Основные публикации

- The view from the left, 2004, coauth. McKinna // *Journal of Functional Programming* 14 (1), pp. 69-111. Cit. 335.
- Applicative programming with effects, 2008, coauth. Paterson // *Journal of Functional Programming* 18 (01), pp. 1-13. Cit. 263.
- Faking it Simulating dependent types in Haskell, 2002 // *Journal of Functional Programming* 12 (4-5), pp. 375-392. Cit. 119.



# Лектор

Конор Макбрайд



- *Место работы:* Университет Стрэтклайда, Шотландия.
- *Тема диссертации:* Dependently Typed Functional Programs and their Proofs, 1999, U. of Edinburg.

## Недавние публикации

- Indexed containers, 2015, coauth. Altenkirch, Ghani, Hancock, Morris // *Journal of Functional Programming* 25, e5.
- Hasochism: the pleasure and pain of dependently typed Haskell programming, 2014, coauth. Lindley // *ACM SIGPLAN Notices* 48 (12), pp. 81-92.



# Содержание

## Лекция 1. Нормальные функторы

## Лекция 2. Контейнеры

```

record Cont : Set1 where
  constructor _<|_
    field
      Sh : Set           -- data view
      Po : Sh -> Set   -- shapes
                           -- positions
    effect view
    commands
    responses
open Cont

[_]C          : Cont -> Set -> Set
[ S <| P ]C X = Sg S  s -> P s -> X

```

## Лекция 3. Индексированные контейнеры, левитация



## Ссылки

- [Agda-листинг](#), который использовался в качестве слайдов на лекциях.
- [Конспект](#), выданный до лекций – имеет ряд расхождений с лекциями (можно понять, если сравнивать с Agda-листингом из первой ссылки).



# Курсы

- 1 Контейнеры для эффектов и контекстов (Т. Уусталу)
- 2 Встроенные предметно-ориентированные языки в Idris (Э. Брэди)
- 3 Обобщенное программирование и программирование на уровне типов в Haskell (А. Лёх)
- 4 Эффективное в худшем случае обобщённое функциональное программирование с массивами данными (Ф. Хенглейн)
- 5 Типы данных типов данных (К. Макбрайд)
- 6 Метапрограммирование времени компиляции в информационном мире  
(Д. Сайм)



# Лектор

Дон Сайм



- *Место работы:* Microsoft Research в Кэмбридже, Великобритания.
- *Тема диссертации:* ???, PhD studies in Cambridge U.

## Основные публикации

- Design and implementation of generics for the .NET common language runtime, 2001, coauth. Kennedy // *ACM SIGPLAN Notices* 36 (5), pp. 1-12. Cit. 263.
- Proving Java type soundness, 1999 // *Formal Syntax and Semantics of Java*, pp. 83-118. Cit. 161.
- Typing a multi-language intermediate code, 2001, coauth. Gordon // *ACM SIGPLAN Notices* 36 (3), pp. 248-260. Cit. 121.



# Лектор

Дон Сайм



- *Место работы:* Microsoft Research в Кэмбридже, Великобритания.
- *Тема диссертации:* ???, PhD studies in Cambridge U.

## Недавние публикации

- F# Data: Making structured data first class citizens, 2015, coauth. Petricek, Guerra // Unpublished draft.
- Expert F# 3.0, 2012, coauth. Granicz, A Cisternino // Apress.



# Содержание

*Поставщик типа* (type provider) состоит из

- библиотеки,
- адаптера между источником данных и системой типов .NET,
- плагин к компилятору/IDE.

В качестве IDE демонстрировалась Visual Studio, хотя утверждается, что в других окружениях тоже должно работать.



# Содержание

*Поставщик типа* (type provider) состоит из

- библиотеки,
- адаптера между источником данных и системой типов .NET,
- плагин к компилятору/IDE.

В качестве IDE демонстрировалась Visual Studio, хотя утверждается, что в других окружениях тоже должно работать.

## Ссылки

- Пособие по созданию поставщика типов.
- Поставщик типов для регулярного выражения.
- **Доклад Сергея Тихона на fuso.nf.**



# Конференция *fpconf*

Сайт: [fpconf.ru](http://fpconf.ru)

# *fpconf*

*Москва, 15 августа 2015*

Цена билета - 7000Р

[Купить билет](#)



Участники: молодые разработчики.

Приглашённый докладчик: Эдвард Кметт.



# Бонус: научная аттестация по информатике в России и ИСП РАН



# «Научные специальности» в России

Технические науки. Информатика, вычислительная техника и управление  
(05.13.00)

- 05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей,
- 05.13.17 – Теоретические основы информатики.



# «Научные специальности» в России

Технические науки. Информатика, вычислительная техника и управление  
(05.13.00)

- 05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей,
- 05.13.17 – Теоретические основы информатики.

Физико-математические науки. Математика (01.01.00)

- 01.01.06 – Математическая логика, алгебра и теория чисел,
- 01.01.09 – Дискретная математика и математическая кибернетика.



# «Научные специальности» в России

Технические науки. Информатика, вычислительная техника и управление  
(05.13.00)

- 05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей,
- 05.13.17 – Теоретические основы информатики.

Физико-математические науки. Математика (01.01.00)

- 01.01.06 – Математическая логика, алгебра и теория чисел,
- 01.01.09 – Дискретная математика и математическая кибернетика.



Edward A. Hirsch

I am a *theoretical computer scientist* (that is, mathematician) interested in computational complexity and proof theory.

*News:* 1. Semester Program on [Proof and Computational Complexity](#) will be held in April-June 2016 in St.Petersburg. The deadlines are **October 31** and **November 30, 2015**.

2. CSR-2016 will be held [in June in St.Petersburg](#). It will feature the opening lecture by Christos Papadimitriou. The deadline is **December 11**.



# Паспорт специальности 05.13.11

## «Формула специальности»

Специальность, включающая задачи развития теории программирования, создания и сопровождения программных средств различного назначения.

Научное и народнохозяйственное значение решения проблем данной специальности состоит в повышении **эффективности и надежности** процессов обработки и передачи данных и знаний в вычислительных машинах, комплексах и компьютерных сетях.



# Паспорт специальности 05.13.11

## Области исследования

- 1** модели, методы и алгоритмы («ММА») проектирования и анализа программ и программных систем, их эквивалентных преобразований, верификации и тестирования;
- 2** языки программирования и системы программирования, семантика программ;
- 3** ММА, языки и программные инструменты для организации взаимодействия программ и программных систем;
- 4** системы управления базами данных и знаний;
- 5** программные системы символьных вычислений;
- 6** операционные системы;
- 7** человеко-машинные интерфейсы; ММА и программные средства машинной графики, визуализации, обработки изображений;
- 8** языки, инструменты параллельной и распределенной обработки данных;
- 9** оценка качества, стандартизация и сопровождение программных систем.

# Отрасль наук: физ.-мат. vs технические



# Отрасль наук: физ.-мат. vs технические

Старый паспорт 05.13.11 ([копия на mmcs.sfedu.ru/~ulysses](http://mmcs.sfedu.ru/~ulysses))

- технические** за исследования, содержащие результаты, дающие существенный технический эффект их использования, и при **внедрении** результатов,
- физ.-мат.** при получении результатов в виде новых математических **методов** и доказанных свойств языков или систем программирования, квалифицируемых как вклад в развитие математической теории программирования и систем обработки данных и знаний.



# Отрасль наук: физ.-мат. vs технические

Старый паспорт 05.13.11 ([копия на mmcs.sfedu.ru/~ulysses](http://mmcs.sfedu.ru/~ulysses))

- технические** за исследования, содержащие результаты, дающие существенный технический эффект их использования, и при **внедрении** результатов,
- физ.-мат.** при получении результатов в виде новых математических **методов** и доказанных свойств языков или систем программирования, квалифицируемых как вклад в развитие математической теории программирования и систем обработки данных и знаний.

Новый паспорт 05.13.11 (на сайте ВАК)

технические

физ.-мат.

(ПУСТО)



# Формула диссертационного исследования от ИСП РАН (05.13.11)



# Формула диссертационного исследования от ИСП РАН (05.13.11)

- 1 Программная система («работа должна быть видна»).



# Формула диссертационного исследования от ИСП РАН (05.13.11)

- 1 Программная система («работа должна быть видна»).
- 2 В ней должен быть воплощён **метод**  
( «= повторно используемая идея»).



# Формула диссертационного исследования от ИСП РАН (05.13.11)

- 1 Программная система («работа должна быть видна»).
- 2 В ней должен быть воплощён **метод**  
( «= повторно используемая идея»).
- 3 Должна быть измерена «полезность» метода («найти подходящих попугаев и в них измерить»).



# Формула диссертационного исследования от ИСП РАН (05.13.11)

- 1 Программная система («работа должна быть видна»).
- 2 В ней должен быть воплощён **метод**  
( «= повторно используемая идея»).
- 3 Должна быть измерена «полезность» метода («найти подходящих попугаев и в них измерить»).
- 4 Желательно, чтобы этой ПС кто-то пользовался («пара актов о внедрении не повредит... хотя бы в учебный процесс»).



Конец доклада



# Паспорт специальности 05.13.17

## «Формула специальности»

- 1** исследование процессов создания, накопления и обработки информации;
- 2** исследование методов преобразования информации в данные и знания;
- 3** создание и исследование информационных моделей, моделей данных и знаний, методов работы со знаниями, методов машинного обучения и обнаружения новых знаний;
- 4** исследование принципов создания и функционирования аппаратных и программных средств автоматизации указанных процессов.



# Паспорт специальности 05.13.17

## Области исследования

- 1 исследование информационных процессов;
- 2 разработка и анализ моделей информационных процессов и структур;
- 3 методы и средства кодирования информации в виде данных;
- 4 исследование и разработка средств представления знаний;
- 5 модели и алгоритмы анализа данных (текст, устная речь и изображения);
- 6 методы, языки и модели человекомашинного общения;
- 7 разработка новых интернет-технологий, включая средства поиска, анализа и фильтрации информации;
- 8 разработка основ математической теории языков и грамматик, теории конечных автоматов и теории графов;
- 9 высоконадежная обработка информации, помехоустойчивость;
- 10 разработка теоретических основ создания программных систем для новых информационных технологий;
- 11 разработка требований к программно-техническим средствам...