История ФП
(и немного о монадах)

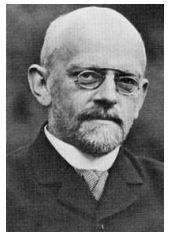
Айрат Худайгулов
Senior BI Developer
Arkadium

1. История ФП в картинках
2. Что общего между формальной логикой и программированием
3. И где это применяется
4. Немного о тех самых на букву М

Давид Гильберт
(1862-1943)

1928 – “Entscheidungsproblem”
Проблема разрешения

“Все доказуемые утверждения истинны, а все истинные утверждения доказуемы”

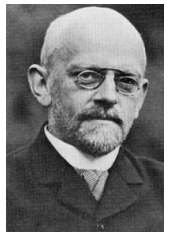


1928 – “Entscheidungsproblem”
Проблема разрешения

Курт Гёдель
(1906-1978)

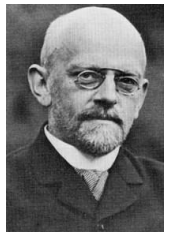
1930 – Теорема о неполноте

“Это утверждение ложно”



1928 – “Entscheidungsproblem”
Проблема разрешения

1930 – Теорема о неполноте



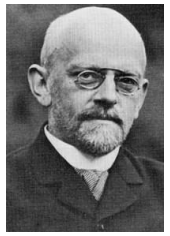
1928 – “Entscheidungsproblem”
Проблема разрешения

1930 – Теорема о неполноте

Алонзо Чёрч
(1903-1995)

1932 – λ-исчисление

$L, M, N ::= x \mid (\lambda x.N) \mid (L\ M)$

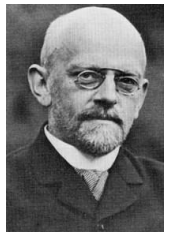


1928 – “Entscheidungsproblem”
Проблема разрешения

1932 – λ-исчисление

1930 – Теорема о неполноте

1936 – доказал что любой алгоритм написанный на λ-исчислении делает проблему разрешения неразрешимой



1928 – “Entscheidungsproblem”
Проблема разрешения

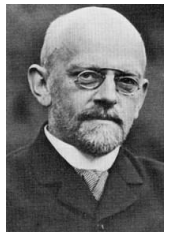
1932 – λ-исчисление
1936 – λ + проблема разрешения

1930 – Теорема о неполноте

Курт Гёдель
(1906-1978)

1936
общерекурсивные фунцкции

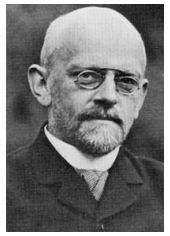
Нашёл второе доказательство неразрешимости проблемы разрешения



1928 – “Entscheidungsproblem”
Проблема разрешения

1932 – λ-исчисление
1936 – λ + проблема разрешения

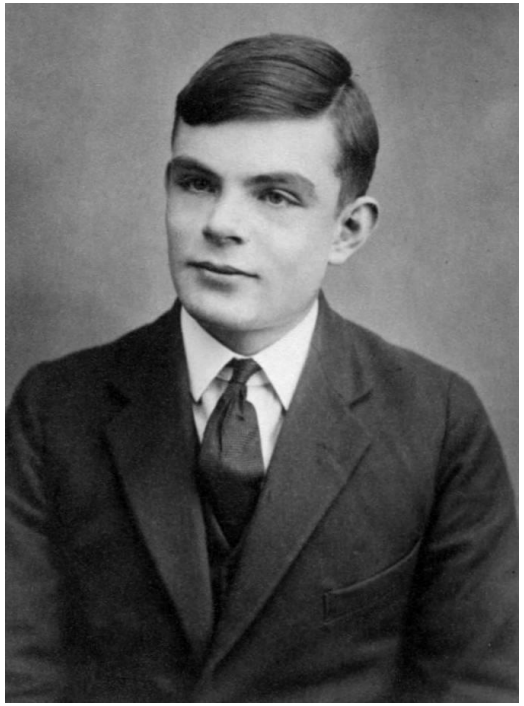
1930 – Теорема о неполноте
1936 – функции + проблема разрешения



1928 – “Entscheidungsproblem”
Проблема разрешения

1932 – λ-исчисление
1936 – λ + проблема разрешения

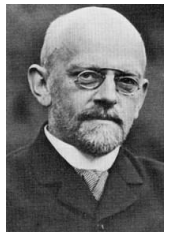
1930 – Теорема о неполноте
1936 – функции + проблема разрешения



Алан Тьюринг
(1912-1954)

1936
Машина Тьюринга

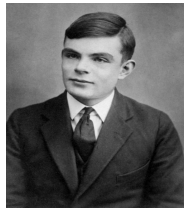
Показал что алгоритмы на машине Тьюринга так же не решают проблему разрешения



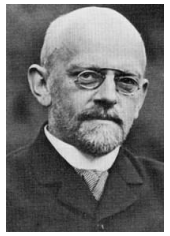
1928 – “Entscheidungsproblem”
Проблема разрешения

1932 – λ-исчисление
1936 – λ + проблема разрешения

1930 – Теорема о неполноте
1936 – функции + проблема разрешения



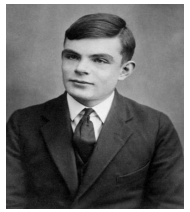
1936 – машина Тьюринга + проблема разрешения



1928 – “Entscheidungsproblem”
Проблема разрешения

1932 – λ-исчисление
1936 – λ + проблема разрешения

1930 – Теорема о неполноте
1936 – функции + проблема разрешения



1936 – машина Тьюринга + проблема разрешения

$$\frac{A \qquad B}{A \,\&\, B}\,\&\text{-I} \qquad \frac{A \,\&\, B}{A}\,\&\text{-E}_1 \qquad \frac{A \,\&\, B}{B}\,\&\text{-E}_2$$

$$\frac{\begin{array}{c}[A]^x \\ \vdots \\ B\end{array}}{A \supset B}\,\supset\text{-I}^x \qquad \frac{A \supset B \qquad A}{B}\,\supset\text{-E}$$

$$\frac{M : A \qquad N : B}{\langle M, N \rangle : A \times B}\ \times\text{-I} \qquad \frac{L : A \times B}{\pi_1\, L : A}\ \times\text{-E}_1 \qquad \frac{L : A \times B}{\pi_2\, L : B}\ \times\text{-E}_2$$

$$\frac{\begin{array}{c}[x : A]^x \\ \vdots \\ N : B\end{array}}{\lambda x.\, N : A \to B}\ \to\text{-I}^x \qquad \frac{L : A \to B \qquad M : A}{L\, M : B}\ \to\text{-E}$$

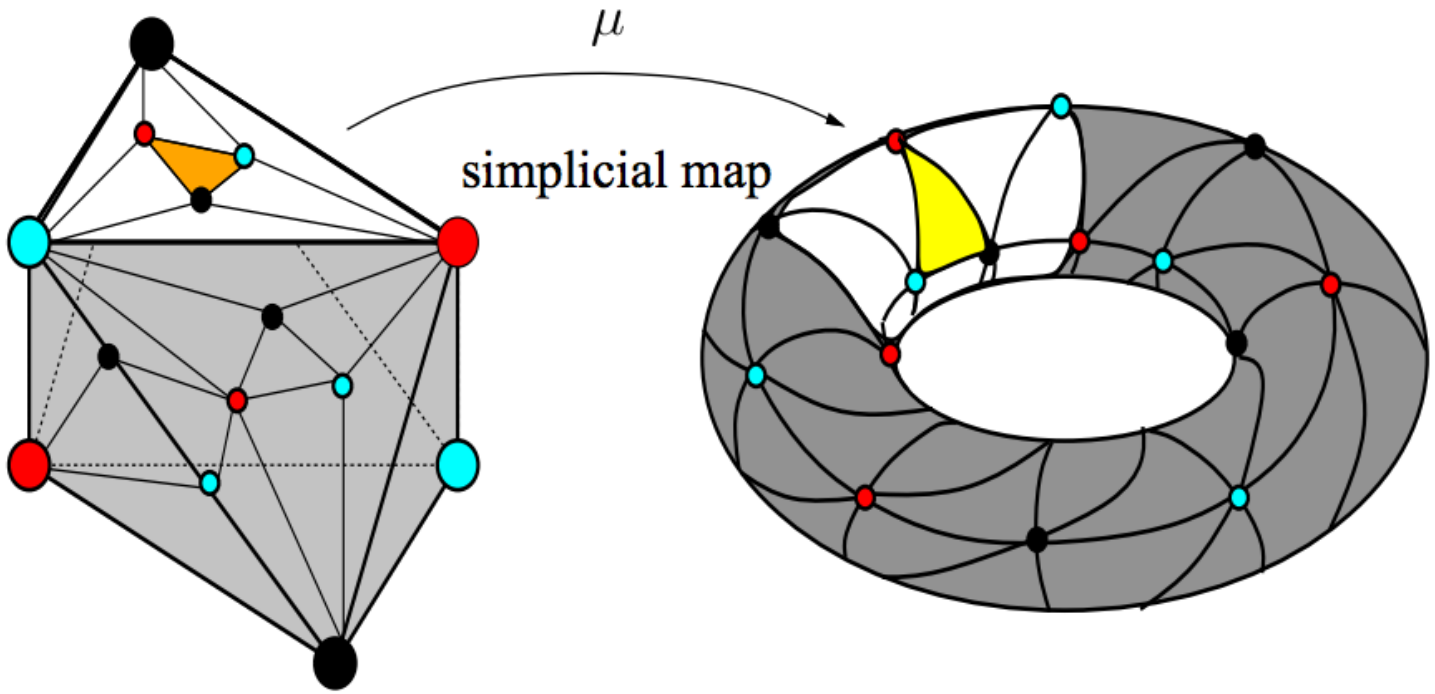
$$\dfrac{\dfrac{\dfrac{[z : B \times A]^{z}}{\pi_2\, z : A}\times\text{-E}_2 \qquad \dfrac{[z : B \times A]^{z}}{\pi_1\, z : B}\times\text{-E}_1}{\langle \pi_2\, z, \pi_1\, z\rangle : A \times B}\times\text{-I}}{\lambda z.\, \langle \pi_2\, z, \pi_1\, z\rangle : (B \times A) \to (A \times B)}\to\text{-I}^{z}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{[z : B \times A]^z}{\pi_2\ z : A}\times\text{-E}_2 \quad \dfrac{[z : B \times A]^z}{\pi_1\ z : B}\times\text{-E}_1}{\langle \pi_2\ z, \pi_1\ z\rangle : A \times B}\times\text{-I}}{\lambda z.\ \langle \pi_2\ z, \pi_1\ z\rangle : (B \times A) \to (A \times B)}\to\text{-I}^z \quad \dfrac{y : B \qquad x : A}{\langle y, x\rangle : B \times A}\times\text{-I}}{(\lambda z.\ \langle \pi_2\ z, \pi_1\ z\rangle)\ \langle y, x\rangle : A \times B}\to\text{-E}$$

Логика	**Теория типов**	**Теория категорий**
Истина	Unit	Терминальный объект
Ложь	Void	Начальный объект
Пропозиция	Тип данных	Объект
Доказательство	Вычисление программы	Морфизм
Импликация →	Функция: f : A → B	Множество морфизмов
Конъюнкция ∧	Тип-произведение A*B	Произведение объектов
Дизъюнкция ∨	Тип-сумма A+B	Сумма объектов

Natural Deduction Gentzen (1935)	⇔	Typed Lambda Calculus Church (1940)
Type Schemes Hindley (1969)	⇔	ML Type System Milner (1975)
System F Girard (1972)	⇔	Polymorphic Lambda Calculus Reynolds (1974)
Modal Logic Lewis (1910)	⇔	Monads (state, exceptions) Kleisli (1965), Moggi (1987)
Classical-Intuitionistic Embedding Gödel (1933)	⇔	Continuation Passing Style Reynolds (1972)
Intuitionistic logic Heyting (1930)	⇔	Intuitionistic type theory Martin-Löf
Linear logic	???	Concurrency


Idris



“In this post I describe the Asynchronous Computability Theorem, which uses tools from Algebraic Topology to show whether a task is solvable in a distributed system…” (c) Lev Gorodinski

Прекрасный мир математики.

Чистые функции:
$A \to B$
$B \to C$

Композируются как:
$A \to C$

Жестокая реальность.

Функции?..
unit → *unit* (берём *A* из глобального состояния и в этом же состоянии задаём *B*)
$B \to C$ (иногда падаем с NRE)

Вообще не композируются.

```
GetCustomerById: int -> Customer
GetCart: Customer -> Cart
AddItem: Item -> Cart -> Cart
```

```
GetCustomerById 5
|> GetCart
|> AddItem someItem
|> AddItem
anotherItem
```

```
GetCustomerById(5)
    .GetCart()
    .AddItem(someItem)
    .AddItem(anotherItem)
```

```
GetCustomerById: int -> Customer option
GetCart: Customer -> Cart option
AddItem: Item -> Cart -> Cart


GetCustomerById 5
|> GetCart //error
|> AddItem someItem
|> AddItem
anotherItem
```

```
let customerOpt = GetCustomerById 5
match customerOpt with
| None -> None
| Some customer ->
  let cartOpt = GetCart customer
  match cartOpt with
  | None -> None
  | Some cart ->
    cart
    |> AddItem someItem
    |> AddItem anotherItem
    |> Some
```

```
match opt with
| None -> None
| Some s ->
//continuation on value s
```

```
let bindO  (f: 'a -> Option<'b>)
           (opt: Option<'a>)
           : Option<'b> =
   match opt with
   | None -> None
   | Some s -> f s
```

```
let mapO   (f: 'a -> 'b)
           (opt: Option<'a>)
           : Option<'b> =
      match opt with
      | None -> None
      | Some s -> Some (f s)
```

```
let customerOpt = GetCustomerById 5
match customerOpt with
| None -> None
| Some customer ->
  let cartOpt = GetCart customer
  match cartOpt with
  | None -> None
  | Some cart ->
    cart
    |> AddItem someItem
    |> AddItem anotherItem
    |> Some
```

```
GetCustomerById 5
|> bindO GetCart
|> mapO (AddItem someItem)
|> mapO (AddItem someItem)
```

```
GetUserFromDataBase: int -> (User -> unit) -> unit
GetUserFriendList: User -> (User list -> unit) -> unit
SendGreetings: string -> User list -> unit


 GetUserFromDataBase 1 (fun user ->
    GetUserFriendList user (fun friendList ->
      SendGreetings “Hello" friendList
    )
 )
```

```
continuation:
   fun value -> //something
returns unit
```

```
type Async<'a> = ('a -> unit) -> unit
```

```
let bindA
   (f: 'a -> (('b -> unit) -> unit))
   (asyncA: ('a -> unit) -> unit)
   : ('b -> unit) -> unit =

   fun (cont:'b -> unit) ->
      asyncA (fun (a:'a) -> f a cont)
```

```
let mapA
   (f: 'a -> 'b)
   (asyncA: ('a -> unit) -> unit)
   : ('b -> unit) -> unit =

   fun (cont:'b -> unit) ->
   asyncA (fun (a:'a) -> cont (f a))
```

```
continuation:
   fun value -> //something
returns unit
```

```
type Async<'a> = ('a -> unit) -> unit
```

```
let bindA
   (f: 'a -> Async<'b>)
   (asyncA: Async<'a>)
   : Async<'b> =

   fun cont ->
      asyncA (fun a -> f a cont)
```

```
let mapA
   (f: 'a -> 'b)
   (asyncA: Async<'a>)
   : Async<'b> =

   fun cont ->
      asyncA (f >> cont)
```

```
GetUserFromDataBase 1 (fun user ->
   GetUserFriendList user (fun friendList ->
      SendGreetings “Hello" friendList
   )
)
```

```
GetUserFromDataBaseAsync 1
|> bindA GetUserFriendListAsync
|> mapA (SendGreetings "NewName")
```

```
GetAllManagers: unit -> List<Manager>
GetDirectEmployees: Manager -> List<Employee>
RaizeSalary: Employee -> unit

let managers = GetAllManagers()
for manager in managers do
   let directEmployees = GetDirectEmployees manager
   for employee in directEmployees do
      RaizeSalary employee
```

```
for value in values
  //continuation on value
```

```
let bindL
      (f: 'a -> List<'b>)
      (list: List<'a>)
      : List<'b> =
   let output = List()
   for i=0 to list.Count-1 do
      let values = f list.[i]
      output.AddRange values
   output
```

```
let mapL
      (f: 'a -> 'b)
      (list: List<'a>)
      : List<'b> =
   let output = List()
   for i=0 to list.Count-1 do
      let value = f list.[i]
      output.Add value
   output
```

```
let managers = GetAllManagers()
for manager in managers do
  let directEmployees = GetDirectEmployees manager
  for employee in directEmployees do
    RaizeSalary employee
```

```
GetAllManagers()
|> bindL GetDirectEmployees
|> mapL RaizeSalary
```

```
GetAllManagers()
  .SelectMany(GetDirectEmployees)
  .Select(RaizeSalary)
  .ToList()
```

	Option	**Async**	**List**	**Linq**
map	(‘a -> ‘b) -> Option<‘a> -> Option<‘b>	(‘a -> ‘b) -> Async<‘a> -> Async<‘b>	(‘a -> ‘b) -> List<‘a> -> List<‘b>	(TI -> TO) -> IEnumerable<TI> -> IEnumerable<TO>
bind	(‘a -> Option<‘b>) -> Option<‘a> -> Option<‘b>	(‘a -> Async<‘b>) -> Async<‘a> -> Async<‘b>	(‘a -> List<‘b>) -> List<‘a> -> List<‘b>	(TI -> IEnumerable<TO>) -> IEnumerable<TI> -> IEnumerable<TO>

Что я хотел сказать

- Функциональное программирование это не только React и LINQ
- Теоретическая база фундаментальна как сама природа
- Развитие ФП === развитие CS
- Оно и правда помогает решать проблемы
- Поэтому его полезно знать

Спасибо за внимание!

ВОПРОСЫ

Айрат Худайгулов

Полезные ссылки

Proposition as types. Philip Wadler - https://www.youtube.com/watch?v=dgrucfgv2Tw

Truth about Types. Bartosz Milewski - https://www.youtube.com/watch?v=dgrucfgv2Tw

The Asynchronous Computability Theorem. Lev Gorodinski - https://medium.com/@eulerfx/the-asynchronous-computability-theorem-171e9d7b9423

Monads explained in C# (again). Mikhail Shilkov - https://mikhail.io/2018/07/monads-explained-in-csharp-again/

Understanding map and apply. Scott Wlaschin - https://fsharpforfunandprofit.com/posts/elevated-world/

Project Everest (verified TSL) - https://project-everest.github.io/

Curry-Howard Correspondences for Concurrency - http://www.mat.unb.br/ayala/EVENTS/JorgeAPerezCurryHoward.pdf