

# Lambda calculus

Functional models of computation

Ivan Trepakov

NSU Sys.Pro

## History

- 1928 - Hilbert's *Entscheidungsproblem* <sup>1</sup>
  - Is there an *algorithm* for deciding whether a proposition in first-order logic is true or false?
- Replacement for set theory as foundation of mathematics
  - 1930 - Combinatory logic (*Curry, Schönfinkel*)
  - 1932 -  $\lambda$ -calculus (*Church*)
  - 1935 - Kleene-Rosser paradox
- Effective computability
  - 1935 - Untyped  $\lambda$ -calculus (*Church, Kleene, Rosser*)
  - 1936 - Turing machine
  - 1936 - Church-Turing thesis
- 1936 - Undecidability of first-order logic
  - Halting problem of Turing machine
  - Equivalence of  $\lambda$ -terms

---

<sup>1</sup>German for “decision problem”

## History

- 1928 - Hilbert's *Entscheidungsproblem*<sup>1</sup>
  - Is there an *algorithm* for deciding whether a proposition in first-order logic is true or false?
- Replacement for set theory as foundation of mathematics
  - 1930 - Combinatory logic (*Curry, Schönfinkel*)
  - 1932 -  $\lambda$ -calculus (*Church*)
  - 1935 - Kleene-Rosser paradox
- Effective computability
  - 1935 - Untyped  $\lambda$ -calculus (*Church, Kleene, Rosser*)
  - 1936 - Turing machine
  - 1936 - Church-Turing thesis
- 1936 - Undecidability of first-order logic
  - Halting problem of Turing machine
  - Equivalence of  $\lambda$ -terms

## David Hilbert

- [Haskell Curry](#)
- Wilhelm Ackermann
- John von Neumann
- Ernst Zermelo
- ...

## Alonzo Church

- [Stephen Cole Kleene](#)
- [J. Barkley Rosser](#)
- Alan Turing
- Dana Scott
- Michael O. Rabin
- ...

---

<sup>1</sup>German for "decision problem"

## Grammar

$$term ::= \underbrace{var}_{\text{Variable}} \mid \underbrace{(term\ term)}_{\text{Application}} \mid \underbrace{(\lambda var. term)}_{\text{Abstraction}}$$

## Examples

$$\lambda x. x \quad (\lambda x. xx)(\lambda y. yy) \quad \lambda f. \lambda x. f(fx)$$

## Conventions

- Application is left associative  
 $abc = (ab)c$
- Abstraction is right associative  
 $\lambda x. \lambda y. x = \lambda x. (\lambda y. x)$
- Consecutive abstractions can be combined  
 $\lambda x. \lambda y. x = \lambda xy. x$

## Grammar

$$term ::= \underbrace{var}_{\text{Variable}} \mid \underbrace{(term\ term)}_{\text{Application}} \mid \underbrace{(\lambda var. term)}_{\text{Abstraction}}$$

## Examples

$\lambda x. x$      $(\lambda x. xx)(\lambda y. yy)$      $\lambda f. \lambda x. f(fx)$

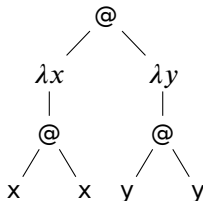
## Conventions

- Application is left associative  
 $abc = (ab)c$
- Abstraction is right associative  
 $\lambda x. \lambda y. x = \lambda x. (\lambda y. x)$
- Consecutive abstractions can be combined  
 $\lambda x. \lambda y. x = \lambda xy. x$

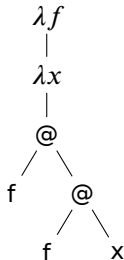
## Tree representation

$\lambda x. x$   
 $\lambda x$   
|  
 $x$

$(\lambda x. xx)(\lambda y. yy)$



$\lambda f. \lambda x. f(fx)$



Free and bound variables

Substitution

$\alpha$ -equivalence

# $\beta$ -conversion

$\beta$ -reduction

$\beta$ -abstraction







# Normal order reduction

First Church-Rosser theorem

Second Church-Rosser theorem

Normal order reduction

Fixed-point combinator

Curry's  $Y$ -combinator

$$Y = \lambda f. (\lambda x. f(xx)) (\lambda x. f(xx))$$

Turing's  $\Theta$ -combinator

$$\Theta = (\lambda xy. x(xxy)) (\lambda xy. x(xxy))$$

Undecidability

Church numerals

Relation to folds

Algebraic data types

Predecessor

Q&A