

ラムダ計算って何だっけ？ 関数型の神髄に迫る

関数型プログラミング言語の始まり

川頭信之

@nkawagashira

概要

1. 自己紹介
2. 計算理論の始まり
3. チューリングマシン
4. ラムダ計算とは
5. ラムダ計算の関数
6. ラムダ計算の加法・乗法
7. 論理積・論理和・否定演算子のラムダ式
8. プログラミング言語の潮流
9. まとめ

自己紹介

- 川頭信之（かわがしらのぶゆき）
- データサイエンティスト
- 元リクルート
プログラミング問題出題者
- 関数型まつり運営スタッフ
- 興味：ラムダ計算、コンビネータ理論、数
理論理学、記号処理、Haskell
- 趣味：PCゲーム（Apex, Farlight 84）、ハン
ググライダー、歴史言語学

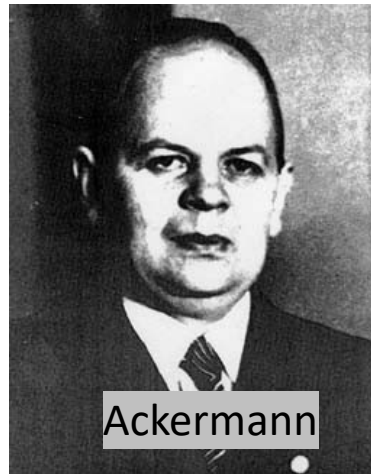
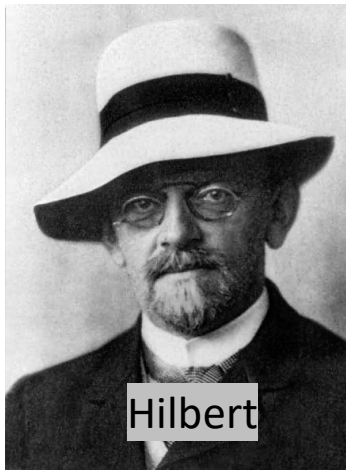


計算理論の始まり

- Entscheidungsproblem (決定問題)

By Hilbert & Ackermann (1928)

- 「任意の形式的な数学的命題について、機械的な手順 (アルゴリズム) でその真偽を常に決定できるか」



↓
だが、そのようなアルゴリズムは不可能であることが証明された！

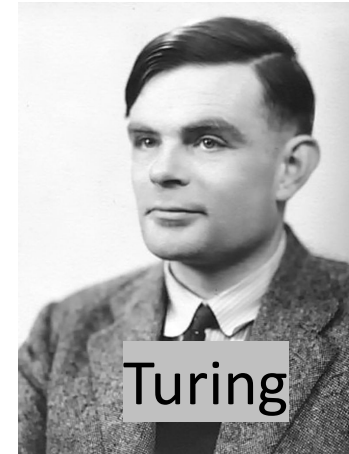
計算機の理論

計算の理論は否定的に証明された

i. チューリングマシン

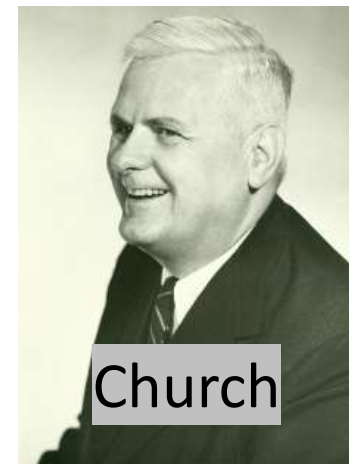
(Alan Turing, 1936/37)

- Von Neumann型、命令型プログラミング
- Fortran, Pascal, Assemblerなど



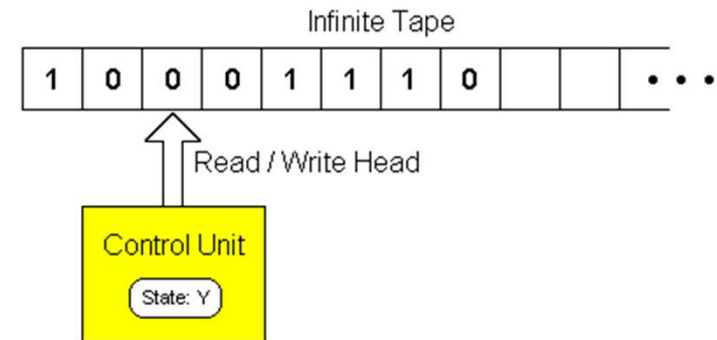
ii. λ 計算 (Alonzo Church, 1936)

- 関数型プログラミング
- Miranda, ML



チューリングマシン

- 1/0の情報を格納することのできる無限に長い記憶セルが存在する。
- 一つの制御ユニットがある。
- 記憶セルのテープ上を移動できる。
- 0あるいは1のマスの状態に従って、0あるいは1に書き換えることができる。



ラムダ計算とは

計算の概念を抽象化したものの一つ（他にはTuring機械）
最小セットのプログラミング言語と言われている。

特徴

- i. 通常の数学的関数の記述の曖昧性を排除できる。
関数の定義と関数の値を区別できる。
- ii. 多変数関数は、単一変数の関数に変換できる。（カリー化）
- iii. 関数と変数（値）の区別がない。関数の引数として関数を取ることができる。

ラムダ計算の関数について

ラムダ式 $\lambda < \text{引数の列} > . < \text{式} >$

$$f(x, y) = x - y + c$$
$$f = (\lambda xy. x - y + c)$$

$$f\ 5\ 3 = (\lambda xy. x - y + c)\ 5\ 3$$
$$= 5 - 3 + c = 2 + c$$

$x = 5, y = 3$ とすると
 $f(5, 3) = 2 + c$ となる。
 c は自由変数なので外部
的に決定される。

x, y : 束縛変数 (ローカル変数)

c : 自由変数 (外部変数)

λ 適用 (λ -applicative)

f は 1 変数の関数

f

$\lambda x. f(x)$

fa

$f(a)$

f は 3 変数の関数

f

$\lambda xyz. f(x, y, z)$

$\lambda x. (\lambda y. (yz. f(x, y, z)))$

fa

$\lambda yz. f(a, y, z)$

fab

$\lambda z. f(a, b, z)$

$fab c$

$f(a, b, c)$

$f(ga)b$

$\lambda z. f(g(a), b, z)$

$fab c = ((fa)b)c$

加法と乗法のラムダ式

$A \equiv \lambda xy. x + y$ 加法

$M \equiv \lambda xy. x \cdot y$ 乗法

$N \equiv \lambda x. (-x)$ マイナス化

$A1$ $\lambda x. 1 + x$

Aab $a + b$

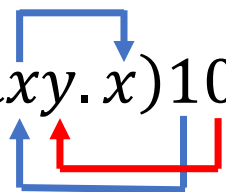
$A(Mab)(N(Mac))$ $ab + (-ac)$

論理値のラムダ式

論理値は関数として表す！

$T \equiv \lambda xy. x$ true 一つ目の変数を取り出す

$F \equiv \lambda xy. y$ false 二つ目の変数を取り出す

$$T10 = (\lambda xy. x)10 \rightarrow 1$$


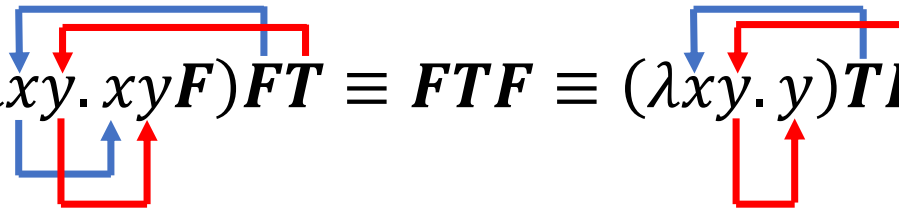
$$F10 = (\lambda xy. y)10 \rightarrow 0$$

論理積のラムダ式

論理積(AND)

$$\wedge \equiv \lambda xy. xyF$$

$$\wedge FF \equiv (\lambda xy. xyF)FF \equiv FFF \equiv (\lambda xy. y)FF \rightarrow F$$

$$\wedge FT \equiv (\lambda xy. xyF)FT \equiv FTF \equiv (\lambda xy. y)TF \rightarrow F$$


$$\wedge TF \equiv ?$$

$$\wedge TT \equiv (\lambda xy. xyF)TT \equiv TTF \equiv (\lambda xy. x)TF \rightarrow T$$

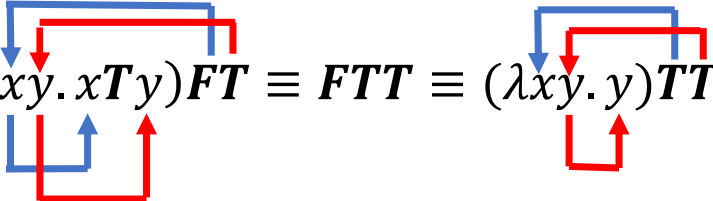
a	b	a ∧ b
F	F	F
F	T	F
T	F	F
T	T	T

論理和 \vee のラムダ式

論理和(OR)

$$\vee \equiv \lambda xy. xTy$$

$$\vee FF \equiv (\lambda xy. xTy)FF \equiv FTF \equiv (\lambda xy. y)TF \rightarrow F$$

$$\vee FT \equiv (\lambda xy. xTy)FT \equiv FTT \equiv (\lambda xy. y)TT \rightarrow T$$


$$\vee TF \equiv ?$$

$$\vee TT \equiv (\lambda xy. xTy)TT \equiv TTT \equiv (\lambda xy. x)TT \rightarrow T$$

a	b	$a \vee b$
F	F	F
F	T	T
T	F	T
T	T	T

否定演算子 \neg

否定演算子(NOT)

$$\neg \equiv \lambda x. xFT$$

$$\neg T \equiv (\lambda x. xFT)T \equiv TFT \equiv (\lambda xy. x)FT \rightarrow F$$

$$\neg F \equiv (\lambda x. xFT)F \equiv FFT \equiv (\lambda xy. y)FT \rightarrow T$$

a	b
T	F
F	T

演習問題 演習問題 $\lambda xy. (xyx)FT$ はどんなラムダ関数になるか？

チャーチ数と後者関数(successor)

$$\bar{0} \equiv \lambda sz. z$$

$$\bar{1} \equiv \lambda sz. s(z)$$

$$\bar{2} \equiv \lambda sz. s(s(z))$$

$$\bar{3} \equiv \lambda sz. s(s(s(z)))$$

$$\bar{3}fa \rightarrow (\lambda sz. s(s(s(z))))fa \rightarrow f(f(f(a)))$$

$$S \equiv \lambda nab. a(nab) \quad \text{後者関数}$$

$$S\bar{1} \equiv (\lambda nab. a(nab))\bar{1} \equiv (\lambda n. (\lambda ab. a(nab)))\bar{1} \rightarrow \lambda ab. a(\bar{1}ab)$$

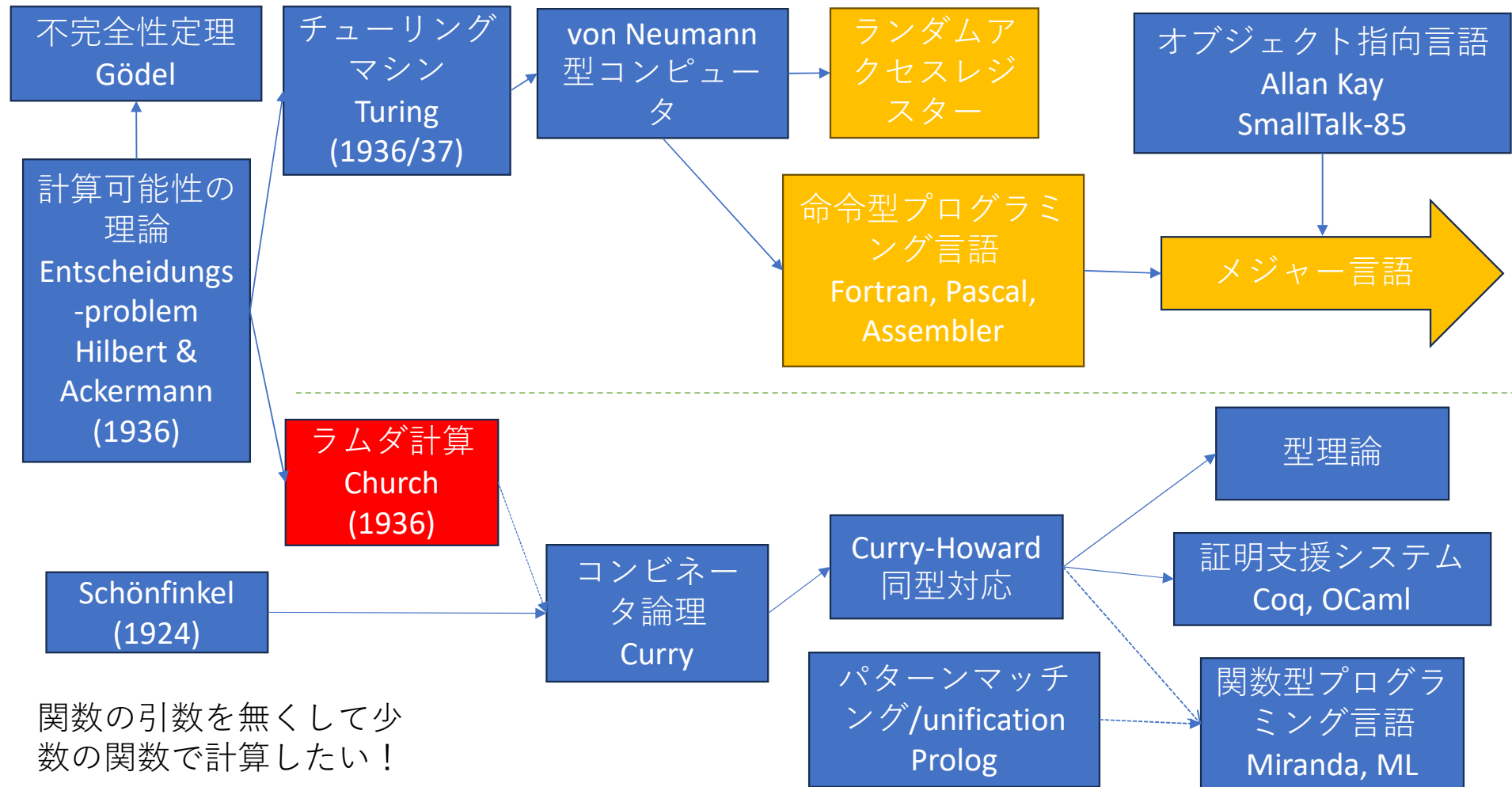
$$\equiv \lambda ab. a((\lambda sz. s(z))ab) \rightarrow \lambda ab. a(a(b)) \equiv \bar{2}$$

$$\text{従って } S\bar{1} \rightarrow \bar{2}$$

$$\bar{2}S\bar{3} \rightarrow \overline{2+3} \equiv \bar{5} \quad S \text{ で加算も定義できる}$$

演習問題 $S\bar{0} \rightarrow \bar{1}$ を確認せよ

プログラミング言語の潮流



まとめ

- 計算の理論には、チューリングマシンとラムダ計算の2つの流れがある。両者は数学的に同一。
- ラムダ計算は、関数の引数に関数を取ることができる（高階関数）。
- ラムダ計算からコンビネータ論理へ、更にCurry-Howard同型対応へと繋がり、関数型プログラミング言語が作られてきた。

参考文献

- J. Roger Hindley & Jonathan P. Seldin (2008)
[Lambda-Calculus and Combinators: An Introduction](#)
- Greg Michaelson (1988)
[An Introduction to Functional Programming through Lambda Calculus](#)
- Henk Barendregt & Erik Barendsen (2000)
[Introduction to Lambda Calculus](#)
- Haskell B. Curry & Robert Feys (1958)
[Combinatory Logic, Volume I](#)
- 関数型プログラミングの推薦図書：理論編（川頭）
<https://qiita.com/Trubetzkoy/items/6593edb4bd3d0c69fbde>