情報数学 まとめ

萩野　達也
hagino@sfc.keio.ac.jp

Slides URL

https://vu5.sfc.keio.ac.jp/slide/

授業概要

- コンピュータプログラムは入力をもらって出力を出すという意味では数学の関数とみなすことができるが，この授業ではプログラムに対応する関数の性質につて取り上げる
- まず，プログラムにより計算可能な関数を理解するために，プログラムの3つのモデル，帰納的関数，チューリング機械，ラムダ計算を紹介する．これら3つのモデルは同等である．
- 次に，ラムダ計算およびプログラムのモデルを理解するために，完全半順序について取り上げる．
- 最後に，プログラムのデータ型を理解するために，関数を抽象的に取り扱う圏理論について紹介する．

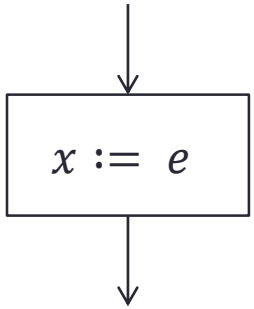
授業予定

1. Whileプログラム
2. 原始帰納的関数
3. 帰納的関数
4. チューリング機械
5. チューリング機械と計算可能性
6. ラムダ計算
7. ラムダ計算と計算可能性
8. 完全半順序
9. 完全半順序とデータ型
10. 連続関数
11. 表示的意味論
12. 圏論入門
13. 極限および随伴関手
14. まとめ
15. 期末試験

WHILEプログラム

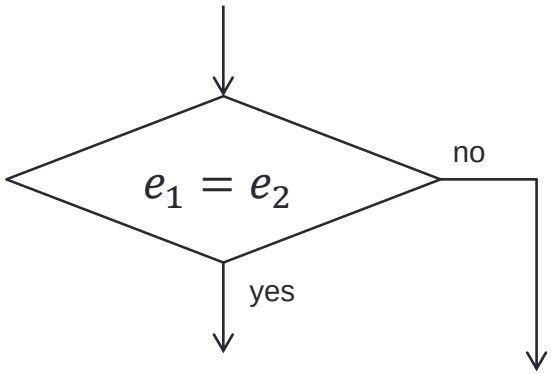
フローチャート

- 代入



e は変数と自然数を含む四則演算の式

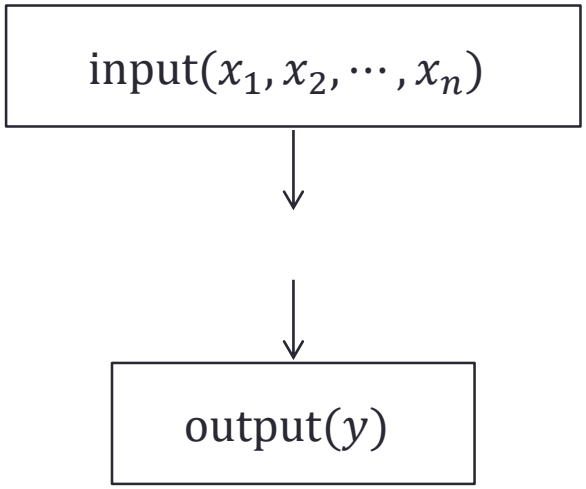
- 条件分岐



e_1 と e_2 は変数と自然数を含む四則演算の式

入出力

- 入力

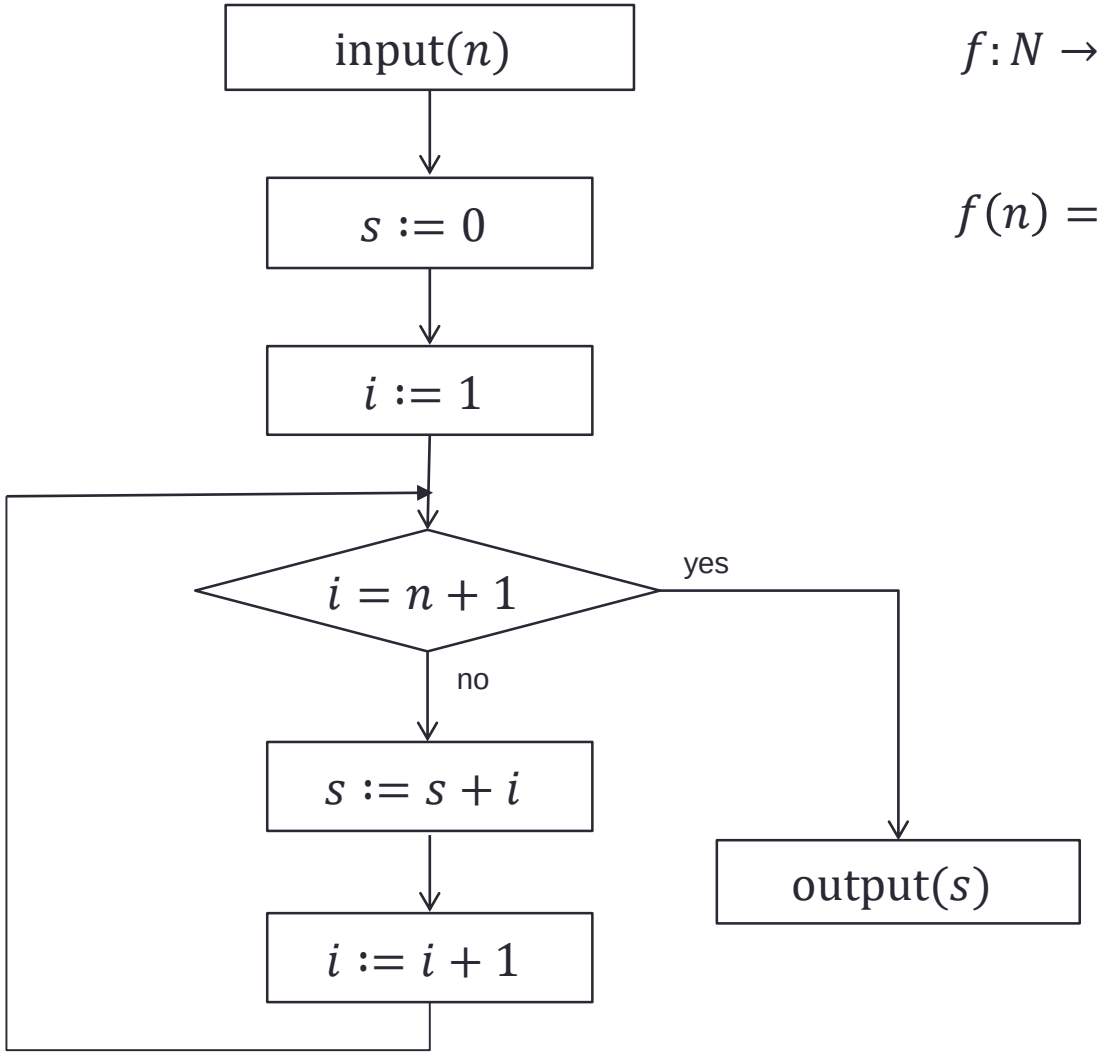


- 出力

- フローチャートプログラム
  - 入力から始まり，代入文と条件分岐を線で結び，出力で終わる．
  - 入力で与えられた変数に対する計算結果を出力とする．

$$f : \underbrace{N \times N \times \cdots \times N}_{\text{入力}} \rightarrow \underbrace{N}_{\text{出力}}$$

1+2+…+n を計算するプログラム

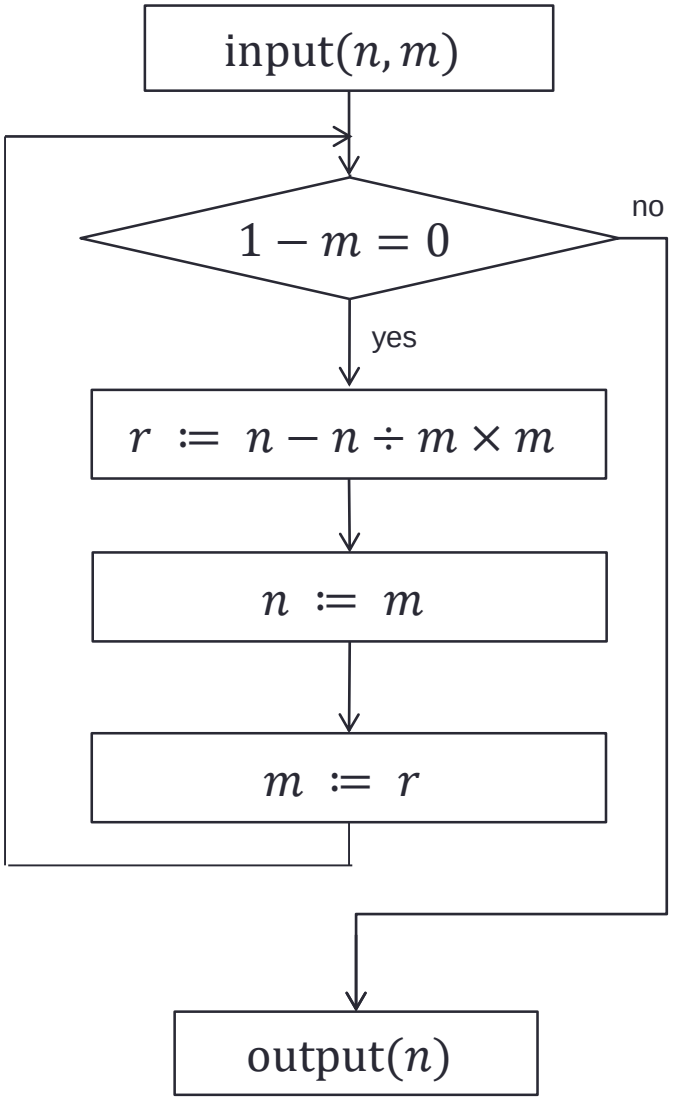


$$f: N \to N$$

$$f(n) = \sum_{i=1}^{n} i$$

ユークリッドの互除法

ユークリッドの互除法を
フローチャートで表しなさい



Whileプログラム

- プログラミング言語
  - 2次元的グラフであるフローチャートをプログラムとしてコンピュータに与えるのは難しい.
  - 1次元的な言語として表したい.

- Whileプログラム
  - $\mathrm{input}(x_1, x_2, \cdots, x_n)$
  - $\mathrm{output}(y)$
  - $x := e$
  - $\{P_1; P_2; \cdots; P_n\}$
  - if $(e_1 = e_2)$ then P else Q
  - while $(e_1 = e_2)$ P

Whileプログラムの例

- 1+2+…+n を計算するプログラム

```
input(n);
s := 0;
i := 1;
while (i <= n) {
  s := s + i;
  i := i + 1
}
output(s);
```

```
input(n);
s := 0;
i := 1;
while (1 - (i - n) = 1) {
  s := s + i;
  i := i + 1
}
output(s);
```

Whileプログラムの例

- ユークリッドの互除法をwhileプログラムとして書きなさい.

```
input(n,m);



 while (1 - m = 0) {

   r := n - n ÷ m × m;

   n := m;

   m := r
 }




output(n);
```

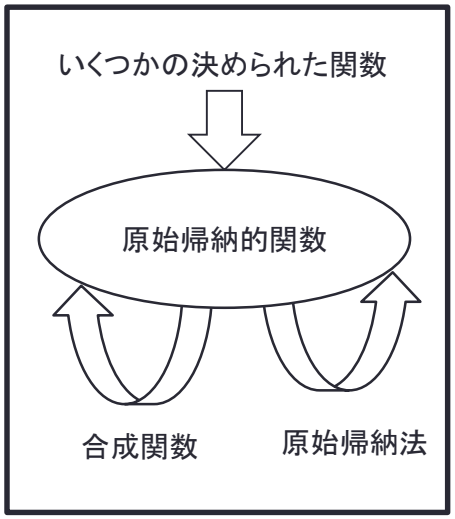
フローチャートとwhileプログラム

- **定理:**
  - 任意のwhileプログラムはフローチャートで表すことができる.
  - 任意のフローチャートはwhileプログラムで表すことができる.

- **系:**
  - 任意のwhileプログラムはwhile文が高々一つだけのプログラムに書き換えることができる.

帰納的関数

原始帰納的関数



- **Definition:**
  - 次の関数を原始帰納的関数（primitive recursive function
  - いくつかの決められた関数は原始帰納的関数である.
    - $zero: N^0 \to N \qquad zero() = 0$
    - $suc: N \to N \qquad suc(x) = x + 1 \quad (x\text{の次の数})$
    - $\pi_i^n : N^n \to N \qquad \pi_i^n(x_1, \dots, x_n) = x_i$

    簡単のために，今後 $zero()$ を 0 と書く
  - 原始帰納的関数の合成は原始帰納的関数である.
    - $g: N^m \to N$ と $h_i: N^n \to N$ $(i = 1,2, \dots, m)$ が原始帰納的関数のとき，合成した $f: N^n \to N$ も原始帰納的関数である.

      $$f(x_1, x_2, \dots, x_n) = g(h_1(x_1, x_2, \dots, x_n), \dots, h_m(x_1, x_2, \dots, x_n))$$
  - 原始帰納法による関数の定義
    - $g: N^n \to N$ と $h: N^{n+2} \to N$ が原始帰納的関数のとき，次のように定義した $f: N^{n+1} \to N$も原始帰納的関数である.

      $$f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n)$$
      $$f\big(x_1, \dots, x_n, suc(y)\big) = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y))$$

原始帰納的関数（例）

- $add: N^2 \to N$　　$add(x, y) = x + y$　　加算
  - $add(x, 0) = x$
  - $add\big(x, suc(y)\big) = suc\big(add(x, y)\big)$
- $mul: N^2 \to N$　　$mul(x, y) = x \times y$　　乗算
  - $mul(x, 0) = zero() = 0$
  - $mul\big(x, suc(y)\big) = add\big(x, mul(x, y)\big)$

- **定理:** 四則演算（加算，減算，乗算，除算）は原始帰納的関数である.

帰納的関数

- **定理**：原始帰納的関数は計算可能である.

- 原始帰納的関数は全域的関数である.
  - 計算可能な関数は全域的とは限らず部分的である.
- 計算可能な関数は原始帰納的関数より広い.

- 帰納的関数（recursive function）
  - 原始帰納的関数
  - 原始帰納的述語に対する最小解関数
  - 帰納的関数の合成

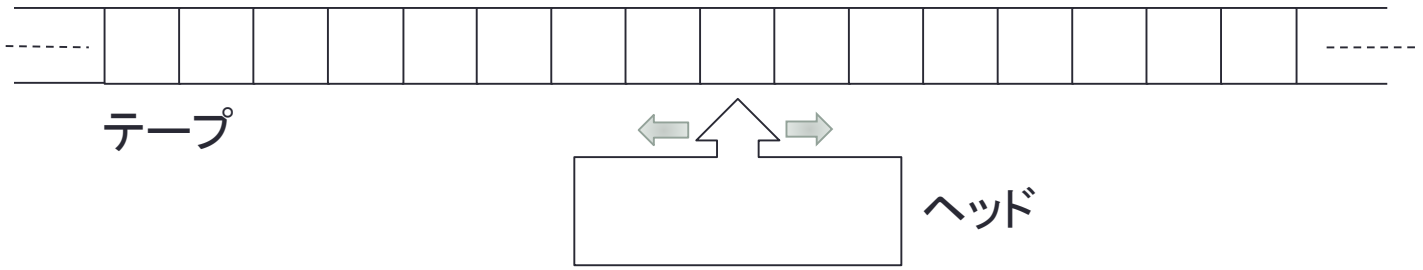
- **定理**：帰納的関数は計算可能である. 計算可能な関数は帰納的関数である.

チューリング機械

チューリング機械

- アラン・チューリング（Alan Turing）
  - イギリスの数学者（1912 年6月23日～1954年6月7日)
  - 「On Computable Numbers, with an Application to the Entscheidungsproblem」1936年5月28日
    - Entscheidungsproblem = 決定問題
    - The Entscheidungsproblem =「与えられた論理式が証明可能かどうかを決定する手法はあるか」ヒルベルトが1928年に出した問題.

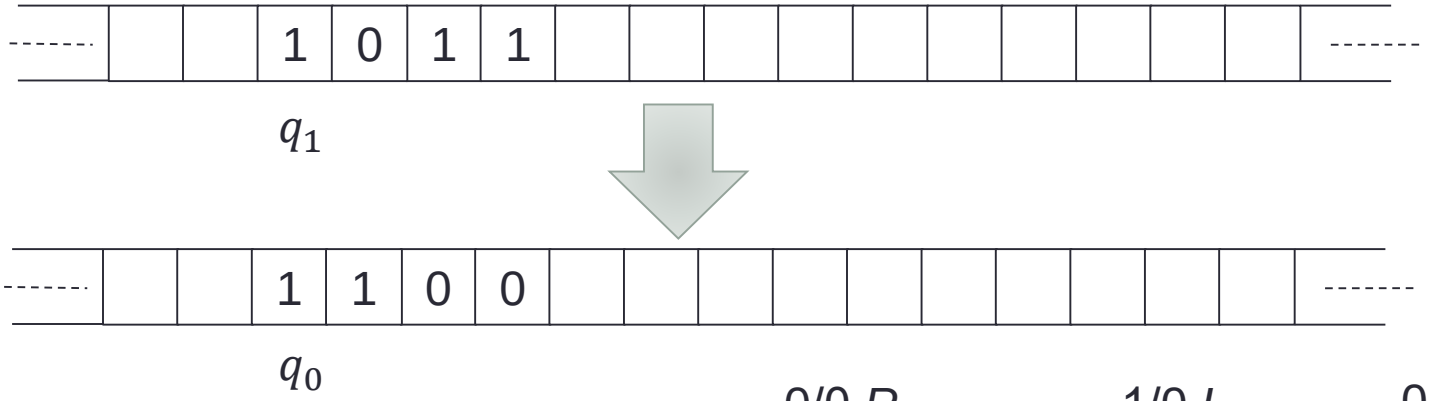
- チューリング機械
  - 左右無限に長いテープ
  - テープ上のデータを読み書きし，左右に動くヘッド



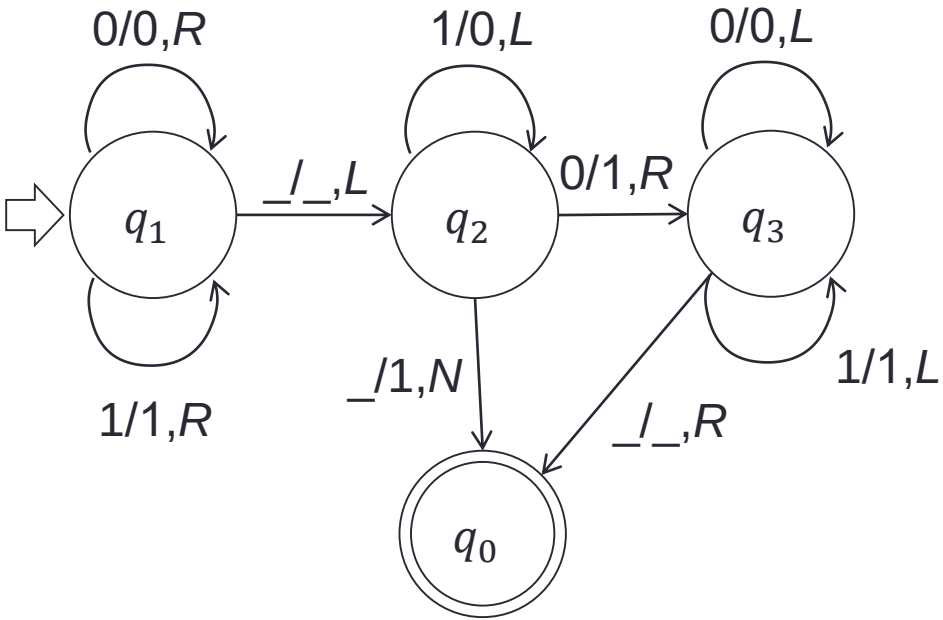
チューリング機械の例

- テープ上の二進数の数字に1を加えるチューリング機械を作りなさい.

$$M_5 = (\{\_, 0, 1\}, \{q_0, q_1, q_2, \cdots\}, T_5)$$



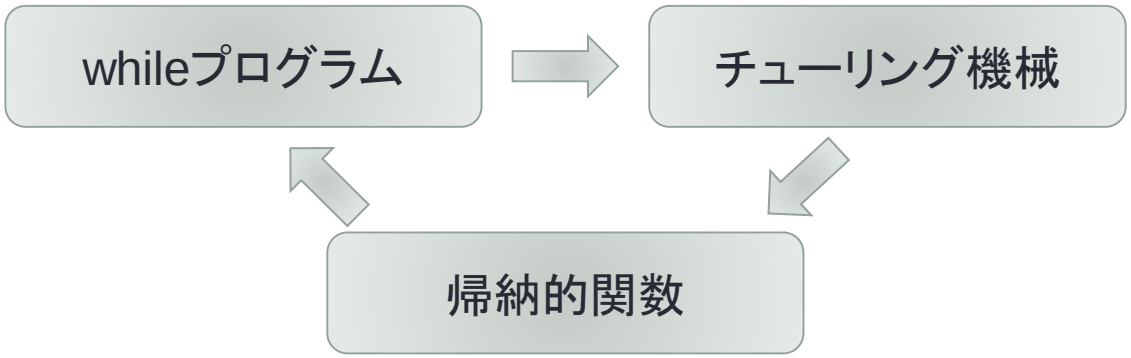
T_5	_	0	1
q_1	(q_2,_,L)	(q_1,0,R)	(q_1,1,R)
q_2	(q_0,1,N)	(q_3,1,L)	(q_2,0,L)
q_3	(q_0,_,R)	(q_3,0,L)	(q_3,1,L)



チューリング機械とプログラム

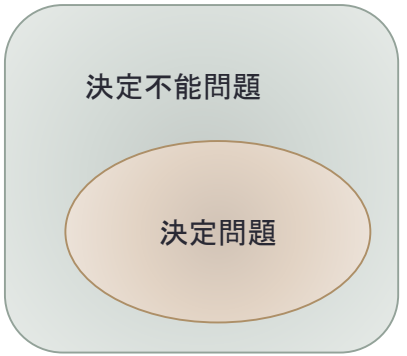
- チューリング機械は一般には停止しない.
  - 計算する関数は全域的ではなく部分的

- **定理**
  - チューリング機械が計算する関数 $f: N^n \to N$ はwhileプログラムで計算できる.
  - 帰納的関数 $f: N^n \to N$ はチューリング機械で計算することができる.



決定可能問題

- **決定可能**問題
  - プログラムでyesかnoか判定できる問題
  - プログラムは停止する必要がある.
  - 対応する帰納的関数は全域的.



- **決定不能**問題
  - 決定可能でない問題
  - プログラムはyesとは答えるが，noが出せないかもしれない.
  - 対応する関数は帰納的ではあるが，全域的ではない.

- **停止問題**:
  - 任意のプログラムPとその入力$a_1, \ldots, a_n$が与えられた時に，プログラムPがこの入力に対して停止し計算結果を出すかどうかを調べるプログラムは存在するか?

決定不能述語

- $\mathrm{halt}_n(z, x_1, \ldots, x_n)$
  - プログラム z が入力 $x_1, \ldots, x_n$ に対して停止するのかどうか.
- $\mathrm{total}_n(z)$
  - プログラム z が常に停止するかどうか.
- $\forall x_1 \cdots \forall x_n(\mathrm{comp}_n(z, x_1, \ldots, x_n) = 0)$
  - プログラム z がどんな入力に対しても常に0を出力するかどうか.
- $\exists x_1 \cdots \exists x_n(\mathrm{comp}_n(z, x_1, \ldots, x_n) = 0)$
  - プログラム z がある入力に対して0を出力することがあるかどうか.
- z に関して $\mathrm{comp}_n(z, x_1, \ldots, x_n)$ の定義域が有限集合である.
  - プログラム z が有限個の入力に対して定数かどうか.
- z に関して $\mathrm{comp}_n(z, x_1, \ldots, x_n)$ が定数関数である.
  - プログラム z が常に同じ値を出力するかどうか.
- z と z' に関して $\mathrm{comp}_n(z, x_1, \ldots, x_n) = \mathrm{comp}_n(z', x_1, \ldots, x_n)$
  - 2つのプログラム z と z' が同じかどうか.

Post対応問題

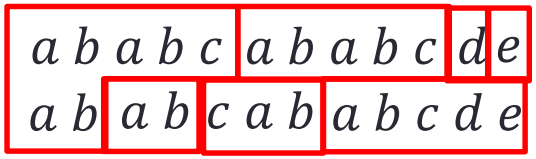
問題:有限個の文字列の対の集合が与えられた時,

$$\{(s_1, t_1), (s_2, t_2), \ldots, (s_n, t_n)\}$$

文字列の連結を使って,次のようになる数列 $i_1, \ldots, i_m$ が存在するか?

$$s_{i_1} s_{i_2} \ldots s_{i_m} = t_{i_1} t_{i_2} \ldots t_{i_m}$$

例: $\{(e, abcde), (ababc, ab), (d, cab)\}$



- この問題は決定不能(計算機で解く一般的な方法はない.プログラムは止まらない.)

ラムダ計算

ラムダ式（λ式）

- **定義:** λ式を以下のように定義する.
  (1) 変数 $x, y, z, x_1, x_2, y', \ldots$ は λ式である.
  (2) λ式 M と変数 x に対して
  $(\lambda x.M)$
  は λ式である.（関数抽象, function abstraction）
  (3) λ式 M と N に対して
  $(M\ N)$
  は λ式である.（関数適用, function application）

- **例:** x, y, f を変数とする.
  - $\left(\lambda x.\left(f\left(f\ x\right)\right)\right)$
  - $\left(\left(\lambda f.\left(f\ x\right)\right)\left(\lambda y.\ y\right)\right)$

- 省略記法
  - $\lambda x_1 x_2 \cdots x_n.M \equiv \left(\lambda x_1.\left(\lambda x_2.\left(\cdots\left(\lambda x_n.M\right)\cdots\right)\right)\right)$
  - $M_1 M_2 M_3 \cdots M_n \equiv \left(\left(\cdots\left(\left(M_1 M_2\right)M_3\right)\cdots\right)M_n\right)$

α変換とβ変換

- 束縛変数を交換しても意味は変わらない.
  - $\lambda x.x \xrightarrow{\alpha} \lambda y.y$
  - $\lambda x.(y(\lambda x.y\ x)x) \xrightarrow{\alpha} \lambda x.(y(\lambda z.y\ z)x)$

- α変換によって自由変数と束縛変数が異なるようにすることができる.

- **定義**: β基 $(\lambda x.M)N$ を $M[x := N]$ に置き換える.

$$(\lambda x.M)N \xrightarrow{\beta} M[x := N]$$

N の自由変数が束縛される場合は α変換を先に行う

- $(\lambda x.x)y \xrightarrow{\beta} y$
- $(\lambda x.x\ y)(\lambda x.x) \xrightarrow{\beta} (\lambda x.x)y \xrightarrow{\beta} y$
- $(\lambda x\ y.x\ y)(\lambda x.y) \xrightarrow{\alpha} (\lambda x\ z.x\ z)(\lambda x.y) \xrightarrow{\beta} \lambda z.(\lambda x.y)z \xrightarrow{\beta} \lambda z.y$

基本的なλ式

- $I \equiv \lambda x.\, x$
  - $I\ M \xrightarrow{\beta} (\lambda x.\, x)\ M \xrightarrow{\beta} M$
- $K \equiv \lambda x\ y.\, x$
  - $K\ M\ N \xrightarrow{\beta} (\lambda x\ y.\, x)\ M\ N \xrightarrow{\beta} (\lambda y.\, M)\ N \xrightarrow{\beta} M$
- $S \equiv \lambda x\ y\ z.\, x\ z\ (y\ z)$
  - $S\ P\ Q\ R \xrightarrow{\beta} P\ R\ (Q\ R)$
  - $S(\lambda x.\, M)(\lambda x.\, N) \xrightarrow{\beta} \lambda z.\, (\lambda x.\, M)z\big((\lambda x.\, N)z\big) \xrightarrow{\beta} \lambda z.\, M[x := z]\ N[x := z] \xrightarrow{\alpha} \lambda x.\, M\ N$
  - $S\ K\ K \xrightarrow{\beta} \xrightarrow{\beta} \lambda z.\, Kz\ (Kz) \xrightarrow{\beta} \xrightarrow{\beta} \lambda z.\, z \xrightarrow{\alpha} I$

- SK式 :
  - 変数と S と K を関数適用して構成された λ式
  - 任意の λ式 M に対してSK式 X が存在して $X \overset{\alpha\beta}{\Rightarrow} M$ とすることができる.

特別なλ式

- $Z \equiv (\lambda x.\, x\ x)(\lambda x.\, x\ x)$
  - $(\lambda x.\, x\ x)(\lambda x.\, x\ x) \xrightarrow{\beta} (\lambda x.\, x\ x)(\lambda x.\, x\ x) \xrightarrow{\beta} (\lambda x.\, x\ x)(\lambda x.\, x\ x) \xrightarrow{\beta}$
- $Y \equiv \lambda y.\, \big(\lambda x.\, y(x\ x)\big)\big(\lambda x.\, y(x\ x)\big)$
  - Curryの不動点演算子
  - $Y\ M \overset{\alpha\beta}{\Leftrightarrow} M(Y\ M)$
  - $Y\ M \xrightarrow{\beta} \big(\lambda x.\, M(x\ x)\big)\big(\lambda x.\, M(x\ x)\big) \xrightarrow{\beta} M\big(\lambda x.\, M(x\ x)\big)\big(\lambda x.\, M(x\ x)\big)$
  - $Y(\lambda x.\, x) \xrightarrow{\beta} (\lambda x.\, x\ x)(\lambda x.\, x\ x)$
- $Y' \equiv \big(\lambda x\ y.\, y(x\ x\ y)\big)\big(\lambda x\ y.\, y(x\ x\ y)\big)$
  - Turingの不動点演算子
  - $Y'\ M \overset{\alpha\beta}{\Rightarrow} M(Y'\ M)$

自然数のλ表現

- 自然数のλ表現:
  - $[0] \equiv \lambda x\ y.\ y$
  - $[1] \equiv \lambda x\ y.\ x\ y$
  - $[2] \equiv \lambda x\ y.\ x(x\ y)$
  - $[3] \equiv \lambda x\ y.\ x\big(x(x\ y)\big)$
  - $\vdots$
  - $[n] \equiv \lambda x\ y.\ \overbrace{x\big(x(\cdots(x}^{n}\ y)\cdots)\big)$
- 演算:
  - $[\mathrm{suc}] \equiv \lambda x\ y\ z.\ y(x\ y\ z)$
  - $[\mathrm{add}] \equiv \lambda x\ y\ z\ w.\ x\ z(y\ z\ w)$
  - $[\mathrm{mul}] \equiv \lambda x\ y\ z\ w.\ x(y\ z)w$
  - $[\mathrm{pred}] \equiv \lambda x\ y\ z.\ x\big(\lambda u\ v.\ v(u\ y)\big)(\lambda a.\ z)(\lambda a.\ a)$
  - $[\mathrm{zero?}] \equiv \lambda x.\ x(\lambda x.\ [\mathrm{false}])[\mathrm{true}]$

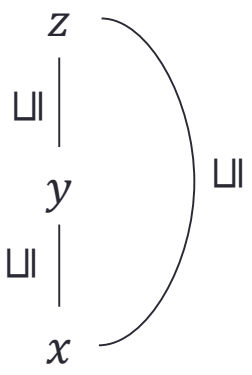
計算可能性

- 計算可能な関数はλ表現可能である.
  - 計算可能な関数は帰納的関数である.
  - 帰納的関数はλ表現を持つ.

- λ表現可能な関数は計算可能である.
  - λ表現可能な関数は最左β変換により実行可能である.
  - λ式を数字でコード化できる.
  - 最左β変換を行うプログラムを作成することができる.

完全半順序

半順序関係



- $\sqsubseteq$ が集合 D 上の順序関係:
  - 反射律(reflective): $x \sqsubseteq x$
  - 推移律(transitive): $x \sqsubseteq y$ かつ $y \sqsubseteq z$ ならば $x \sqsubseteq z$
  - 反対称律(antisymmetric): $x \sqsubseteq y$ かつ $y \sqsubseteq x$ ならば $x = y$
- $(D, \sqsubseteq)$ は半順序集合(partially ordered set)である
  - $\sqsubseteq$ は半順序

- 情報は半順序集合である.

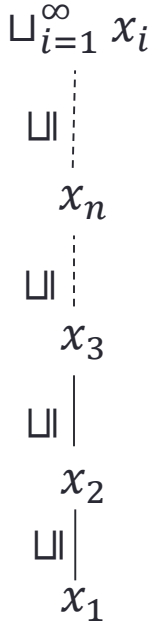
- 情報には最小の情報がある.
  - どんな x に対しても $\bot \sqsubseteq x$

- 情報をあわせることができる.
  - $x \sqcup y$

完全半順序集合（Complete Partial Ordered Set）

- 半順序集合 $(D, \sqsubseteq)$ が**完全半順序集合**（complete partial ordered set, cpo）であるとは：
  - 最小元 $\bot$ がある.
  - $x_1 \sqsubseteq x_2 \sqsubseteq x_3 \sqsubseteq \cdots \sqsubseteq x_n \sqsubseteq \cdots$, となる要素列に対して上限 $\sqcup_{i=1}^{\infty} x_i$ がある.

- 最大元があるとは限らない.



- 情報の集合は完全半順序集合である.

プログラムは連続関数

- プログラムは，完全半順序間の関数である.

  (1) $f(\bot) = \bot$
  (2) $x \sqsubseteq y$ ならば $f(x) \sqsubseteq f(y)$
  (3) $x_1 \sqsubseteq x_2 \sqsubseteq x_3 \sqsubseteq \cdots$ に対して $f(\sqcup_{i=1}^{\infty} x_i) = \sqcup_{i=1}^{\infty} f(x_i)$

- f が(2)を満たすとき，f は単調(monotonic)関数である.
- f が(2)と(3)を満たすとき，f は連続(continuous)関数である.
- f が(1)を満たすとき，f は正格(strict)である.

- プログラムは連続関数である.

平坦領域（flat domain）

- 集合 S に最小元 $\bot$ を追加する.
  - $x \in S$ に対して $\bot \sqsubseteq x$

- 例:
  - 真偽値の集合 $B = \{\mathrm{tt}, \mathrm{ff}\}$ の平坦領域は,

$$B_\bot = \left\{ \begin{array}{ccc} \mathrm{tt} & & \mathrm{ff} \\ & \diagdown \quad \diagup & \\ & \bot & \end{array} \right\}$$

  - 自然数の集合 $N = \{0, 1, 2, 3, 4, \cdots\}$ の平坦領域は,

$$N_\bot = \left\{ \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & \cdots & \\ & \diagdown & \diagdown & | & \diagup & \diagup & \\ & & & \bot & & & \end{array} \right\}$$

直積（product）

- CPO D_1 と D_2 の直積 $D_1 \times D_2$ とは，
  - $D_1 \times D_2 = \{\langle x, y\rangle \mid x \in D_1, y \in D_2\}$
  - $\langle x, y\rangle \sqsubseteq \langle x', y'\rangle \Leftrightarrow x \sqsubseteq x'$ かつ $y \sqsubseteq y'$

- $D_1 \times D_2$ はCPOである．
  - 上記の $\sqsubseteq$ は半順序になっている．
  - $\bot_{D_1 \times D_2} = \langle \bot_{D_1}, \bot_{D_2} \rangle$
  - $\langle x_1, y_1\rangle \sqsubseteq \langle x_2, y_2\rangle \sqsubseteq \langle x_3, y_3\rangle \sqsubseteq \cdots \sqsubseteq \langle x_i, y_i\rangle \sqsubseteq \cdots$ に対して，

$$\sqcup_{i=1}^{\infty} \langle x_i, y_i\rangle = \langle \sqcup_{i=1}^{\infty} x_i \, , \sqcup_{i=1}^{\infty} y_i \rangle$$

or の連続関数への拡張

- $\mathrm{or}_{\bot} : B_{\bot} \times B_{\bot} \to B_{\bot}$

$\mathrm{or}_{\bot}$	$\bot$	ff	tt
$\bot$	$\bot$	$\bot$	$\bot$
ff	$\bot$	ff	tt
tt	$\bot$	tt	tt

- $\mathrm{or}_{L} : B_{\bot} \times B_{\bot} \to B_{\bot}$

or_{L}	$\bot$	ff	tt
$\bot$	$\bot$	$\bot$	$\bot$
ff	$\bot$	ff	tt
tt	tt	tt	tt

- $\mathrm{or}_{R} : B_{\bot} \times B_{\bot} \to B_{\bot}$

or_{R}	$\bot$	ff	tt
$\bot$	$\bot$	$\bot$	tt
ff	$\bot$	ff	tt
tt	$\bot$	tt	tt

- $\mathrm{or}_{P} : B_{\bot} \times B_{\bot} \to B_{\bot}$

or_{P}	$\bot$	ff	tt
$\bot$	$\bot$	$\bot$	tt
ff	$\bot$	ff	tt
tt	tt	tt	tt

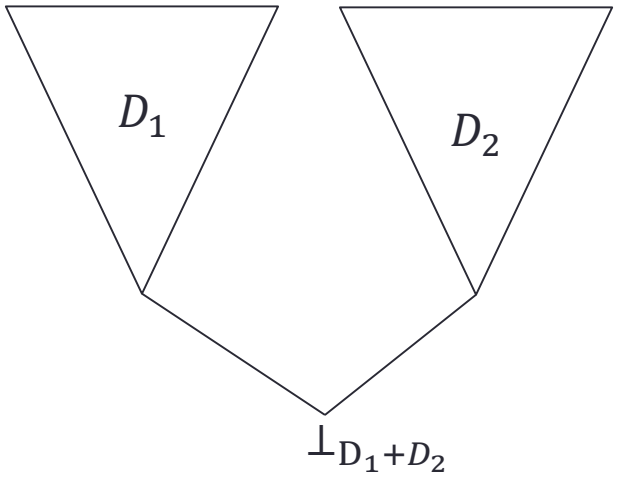
条件関数

- $\mathrm{cond}: B_\perp \times N_\perp \times N_\perp \to N_\perp$
  - $\mathrm{cond}(\perp, x, y) = \perp$
  - $\mathrm{cond}(\mathrm{tt}, x, y) = x$
  - $\mathrm{cond}(\mathrm{ff}, x, y) = y$

直和（co-product）

- CPO D_1 と D_2の直和 $D_1 + D_2$ とは，
  - $D_1 + D_2 = \{\langle x, 1\rangle \mid x \in D_1\} \cup \{\langle y, 2\rangle \mid y \in D_2\} \cup \{\bot_{D_1+D_2}\}$
    - $\langle x, 1\rangle \sqsubseteq \langle x', 1\rangle \Leftrightarrow x \sqsubseteq x'$
    - $\langle y, 2\rangle \sqsubseteq \langle y', 2\rangle \Leftrightarrow y \sqsubseteq y'$
    - $\bot_{D_1+D_2} \sqsubseteq \langle x, 1\rangle$
    - $\bot_{D_1+D_2} \sqsubseteq \langle y, 2\rangle$

- $D_1 + D_2$ はCPOである．
  - 上記 $\sqsubseteq$ は半順序になっている．
  - $\bot_{D_1+D_2}$ が最小元．



関数空間（function space）

- CPO D_1 から D_2 への**関数空間** $[D_1 \to D_2]$ とは，
  - $[D_1 \to D_2] = \{f : D_1 \to D_2 \mid f$ は連続である$\}$
  - $f \sqsubseteq f' \Leftrightarrow x \in D_1$ に対して $f(x) \sqsubseteq f'(x)$

- $[D_1 \to D_2]$ はCPOである．
  - 上記の $\sqsubseteq$ は半順序になっている．
  - $\bot_{[D_1 \to D_2]}(x) = \bot_{D_2}$
  - $f_1 \sqsubseteq f_2 \sqsubseteq f_3 \sqsubseteq \cdots \sqsubseteq f_i \sqsubseteq \cdots$ に対して，

$$\left(\bigsqcup_{i=1}^{\infty} f_i\right)(x) = \bigsqcup_{i=1}^{\infty} f_i(x)$$

不動点定理

- **定理**: 連続関数 $f: D \to D$ は最小不動点を持つ.
  - f の不動点 $u \Leftrightarrow f(u) = u$

- $\mathrm{fix}: [D \to D] \to D$
  - $\mathrm{fix}(f) = \sqcup_{i=0}^{\infty} f^i(\bot)$
  - これも連続

不動点意味論

- 再帰プログラムは分かりにくい？
  - $\mathrm{fact}(x) \equiv \mathrm{if}\ x = 0\ \mathrm{then}\ 1\ \mathrm{else}\ x \times \mathrm{fact}(x-1)$

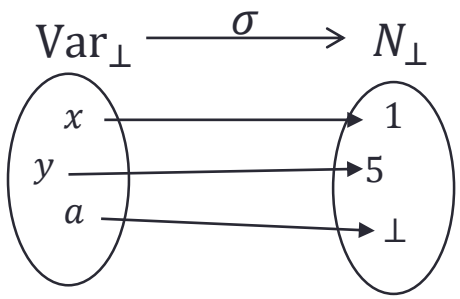
- $\mathrm{fact}: N_\perp \to N_\perp$
  - $\mathrm{fact} = \lambda x.\,\mathrm{cond}\big(x = 0, 1, x \times \mathrm{fact}(x-1)\big)$
  - fact は次の関数 F の不動点
    - $F: [N_\perp \to N_\perp] \to [N_\perp \to N_\perp]$
    - $F(f) = \lambda x.\,\mathrm{cond}\big(x = 0, 1, x \times f(x-1)\big)$
  - fact を F の最小不動点として理解する.
    - $F(\perp) =$
    - $F^2(\perp) =$
    - $F^3(\perp) =$
    - $\vdots$
    - $\mathrm{fix}(F) = \sqcup_{n=0}^{\infty} F^n(\perp)$

プログラミング言語の意味

- プログラミング言語の構文
  - BNF（あるいは文脈自由文法）を使って形式的に定義されることが多い
- プログラミング言語の意味
  - 自然言語で説明されることが多く，あいまいである
- 形式的意味
  - 公理的意味
    - プログラムを論理式の中に埋め込む
  - 操作的意味
    - プログラムを別の良くわかっているシステムで模倣する
  - 表示的意味
    - プログラムを数学的対象として埋め込む

式の表示

- 式の抽象構文 $e \in \mathrm{Exp}$
  - $n \in N$
  - $v \in \mathrm{Var}$
  - $e ::= n \mid v \mid e_1 + e_2 \mid e_1 - e_2 \mid e_1 * e_2 \mid e_1 \,/\, e_2$
- $e \in \mathrm{Exp}$ の表示は現在の変数の値に依存する
- 状態 S
  - $S = [\mathrm{Var}_\perp \to N_\perp]$
  - $\sigma \in S$ は変数を値に対応させる
- Exp の表示
  - $\mathcal{E} \in \mathrm{Exp} \to [S \to N_\perp]$



- $\mathcal{E}[\![n]\!]\sigma = n$
- $\mathcal{E}[\![v]\!]\sigma = \sigma[\![v]\!]$
- $\mathcal{E}[\![e_1 + e_2]\!]\sigma = \mathcal{E}[\![e_1]\!]\sigma + \mathcal{E}[\![e_2]\!]\sigma$
- $\mathcal{E}[\![e_1 - e_2]\!]\sigma = \mathcal{E}[\![e_1]\!]\sigma - \mathcal{E}[\![e_2]\!]\sigma$
- $\mathcal{E}[\![e_1 * e_2]\!]\sigma = \mathcal{E}[\![e_1]\!]\sigma \times \mathcal{E}[\![e_2]\!]\sigma$
- $\mathcal{E}[\![e_1 \,/\, e_2]\!]\sigma = \mathcal{E}[\![e_1]\!]\sigma \div \mathcal{E}[\![e_2]\!]\sigma$

文の表示

- 文の抽象構文 $c \in \mathrm{Cmd}$
  - $c ::= \mathrm{null} \mid x := e \mid \mathrm{if}\ (e_1 = e_2)\ c_1\ \mathrm{else}\ c_2 \mid \mathrm{while}\ (e_1 = e_2)\ c \mid c_1; c_2$
- Cmd の表示
  - $\mathcal{C} \in \mathrm{Cmd} \to [S \to S]$

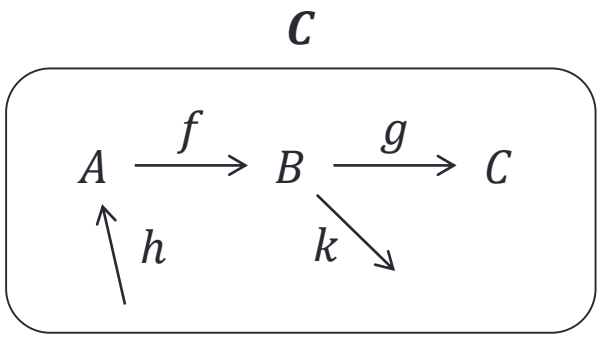
$$S \xrightarrow{\mathcal{C}[\![c]\!]} S$$

- $\mathcal{C}[\![\mathrm{null}]\!]\sigma = \sigma$
- $\mathcal{C}[\![c_1; c_2]\!] = \mathcal{C}[\![c_2]\!] \circ \mathcal{C}[\![c_1]\!]$
- $\mathcal{C}[\![x := e\ ]\!]\sigma = \sigma[\mathcal{E}[\![e]\!]\sigma / x]$
- $\mathcal{C}[\![\mathrm{if}\ (e_1 = e_2)\ c_1\ \mathrm{else}\ c_2]\!]\sigma = \mathrm{cond}(\mathcal{E}[\![e_1]\!]\sigma = \mathcal{E}[\![e_2]\!]\sigma, \mathcal{C}[\![c_1]\!]\sigma, \mathcal{C}[\![c_2]\!]\sigma)$
- $\mathcal{C}[\![\mathrm{while}\ (e_1 = e_2)\ c]\!] = \mathrm{fix}\big(\lambda w.\, \lambda\sigma.\, \mathrm{cond}(\mathcal{E}[\![e_1]\!]\sigma = \mathcal{E}[\![e_2]\!]\sigma, w(\mathcal{C}[\![c]\!]\sigma), \sigma)\big)$

圏論

圏

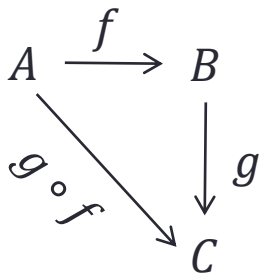
- **圏** C は次のものからなる:
  - **対象**(object)の集まり: $\boldsymbol{C} = \{A, B, C, \ldots\}$
  - 対象 A と B に対して, A から B への**射**(arrow, morphism)の集まり: $\hom_{\boldsymbol{C}}(A, B) = \{f, g, h, \ldots\}$
    - $f \in \hom_{\boldsymbol{C}}(A, B)$ のとき $f: A \to B$ と書く
    - A は f の**定義域**(domain)
    - B は f の**値域**(codomain, range)



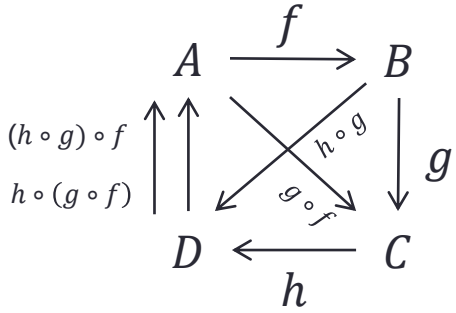
圏（つづき）

- 圏 $\boldsymbol{C}$ は次の条件を満たさなくてはならない：

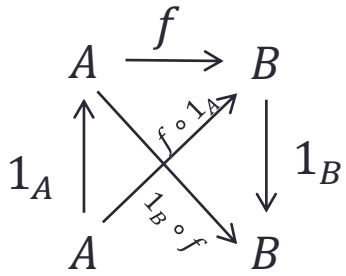
  - $f: A \to B$ および $g: B \to C$ のとき
    $g \circ f: A \to C$



  - $f: A \to B$, $g: B \to C$, $f: C \to D$ のとき
    $h \circ (g \circ f) = (h \circ g) \circ f$



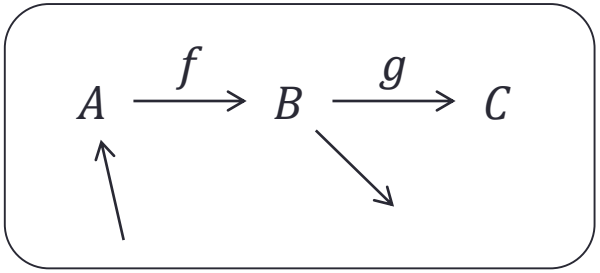
  - 対象 A にたいして射 $1_A: A \to A$ があり，
    $f: A \to B$ に対して
    $f \circ 1_A = f$ および $1_B \circ f = f$



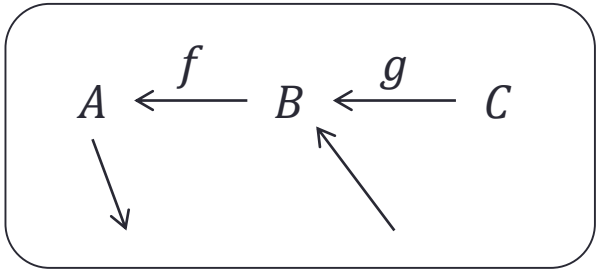
双対圏

- 圏 $\boldsymbol{C}$ の双対圏 $\boldsymbol{C}^{op}$ を次のように定義する
  - $\boldsymbol{C}^{op}$ の対象 = $\boldsymbol{C}$ の対象
  - $\boldsymbol{C}^{op}$ の射 $\hom_{\boldsymbol{C}^{op}}(A,B) = \hom_{\boldsymbol{C}}(B,A)$ $\boldsymbol{C}$ の射
  - 射の向きを反対にする

$\boldsymbol{C}$



$\boldsymbol{C}^{op}$



- 圏 $\boldsymbol{C}$ で成り立つことは，双対圏 $\boldsymbol{C}^{op}$ でも成り立つ．
  - $(\boldsymbol{C}^{op})^{op} = \boldsymbol{C}$

始対象と終対象

- 始対象（initial object）I
  - すべての対象 A に対して唯一の射が存在する

$$I \overset{!}{\dashrightarrow} A$$

- 終対象（final object）F
  - すべての対象 A から唯一の射が存在する

$$A \overset{!}{\dashrightarrow} F$$

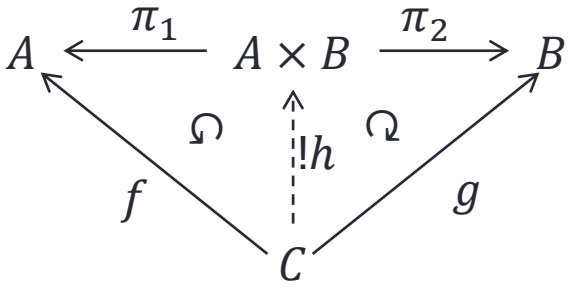
- **定理**: 始対象は存在すれば同型を除いて一意的である.

- **双対定理:** 終対象は存在すれば同型を除いて一意的である.

直積と直和

- $A \times B$ は A と B の直積（product）$\Leftrightarrow$
  - 射 $\pi_1 : A \times B \to A$ と $\pi_2 : A \times B \to B$ が存在する
  - 任意の対象 C と射 $f : C \to A$, $g : C \to B$ に対して，

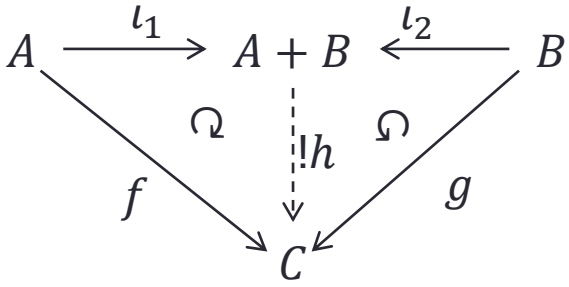
    次の図を可換にする射 $h : C \to A \times B$ が唯一存在する



$\pi_1 \circ h = f$　　$\pi_2 \circ h = g$

- $A + B$ は A と B の直和（co-product）$\Leftrightarrow$
  - 射 $\iota_1 : A \to A + B$ と $\iota_2 : B \to A + B$ が存在する
  - 任意の対象 C と射 $f : A \to C$, $g : B \to C$ に対して，

    次の図を可換にする射 $h : A + B \to C$ が唯一存在する

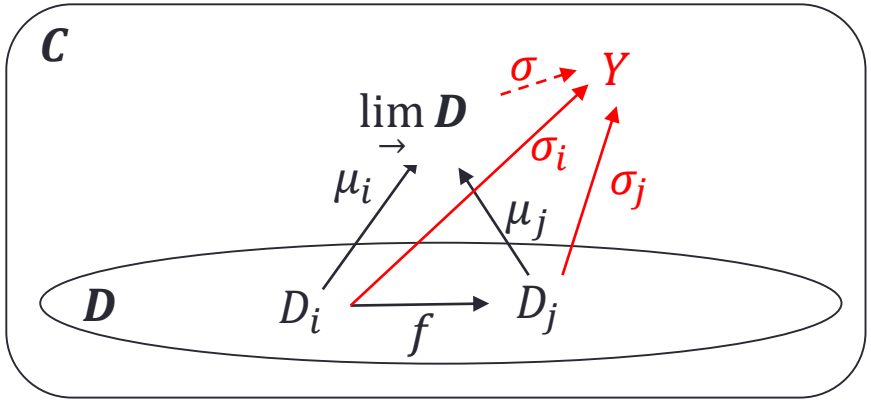


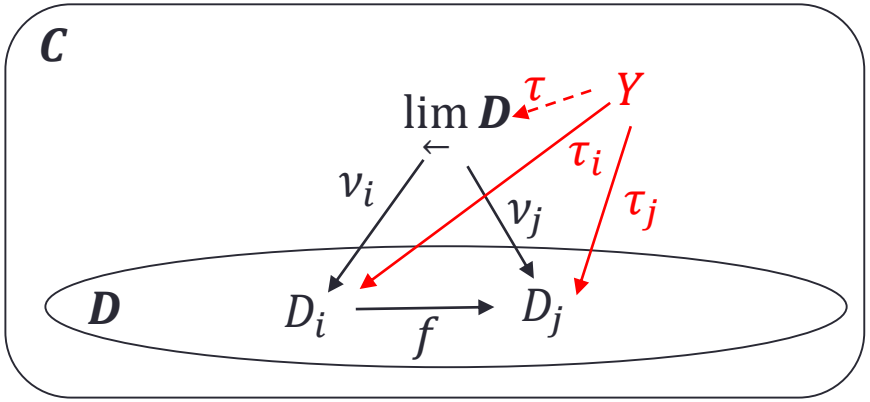
$h \circ \iota_1 = f$　　$h \circ \iota_2 = g$

極限(limit)と余極限(colimit)

- 圏 $\boldsymbol{C}$ の部分圏 $\boldsymbol{D}$ の余極限(colimit)とは，次の条件を満たす $\boldsymbol{C}$ の対象 X である．
  - X から $\boldsymbol{D}$ のすべての対象 D_i に対して射 $\mu_i: D_i \to X$ が存在する．
  - $f \in D$ で $f: D_i \to D_j$ ならば $\mu_j \circ f = \mu_i$
  - $\boldsymbol{C}$ の対象 Y と，Y から $\boldsymbol{D}$ の対象 D_i からの射 $\sigma_i: D_i \to Y$ で，$f \in D$，$f: D_i \to D_j$ に対して $\sigma_j \circ f = \tau_i$ であれば，$\sigma: X \to Y$ で $\sigma_i = \sigma \circ \mu_i$ となるような σ が唯一存在する．

- 圏 $\boldsymbol{C}$ の部分圏 $\boldsymbol{D}$ の極限(limit)とは，次の条件を満たす $\boldsymbol{C}$ の対象 X である．
  - X から $\boldsymbol{D}$ のすべての対象 D_i に対して射 $\nu_i: X \to D_i$ が存在する．
  - $f \in D$ で $f: D_i \to D_j$ ならば $f \circ \nu_i = \nu_j$
  - $\boldsymbol{C}$ の対象 Y と，Y から $\boldsymbol{D}$ の対象 D_i への射 $\tau_i: Y \to D_i$ で，$f \in D$，$f: D_i \to D_j$ に対して $f \circ \tau_i = \tau_j$ であれば，$\tau: Y \to X$ で $\tau_i = \nu_i \circ \tau$ となるような τ が唯一存在する．

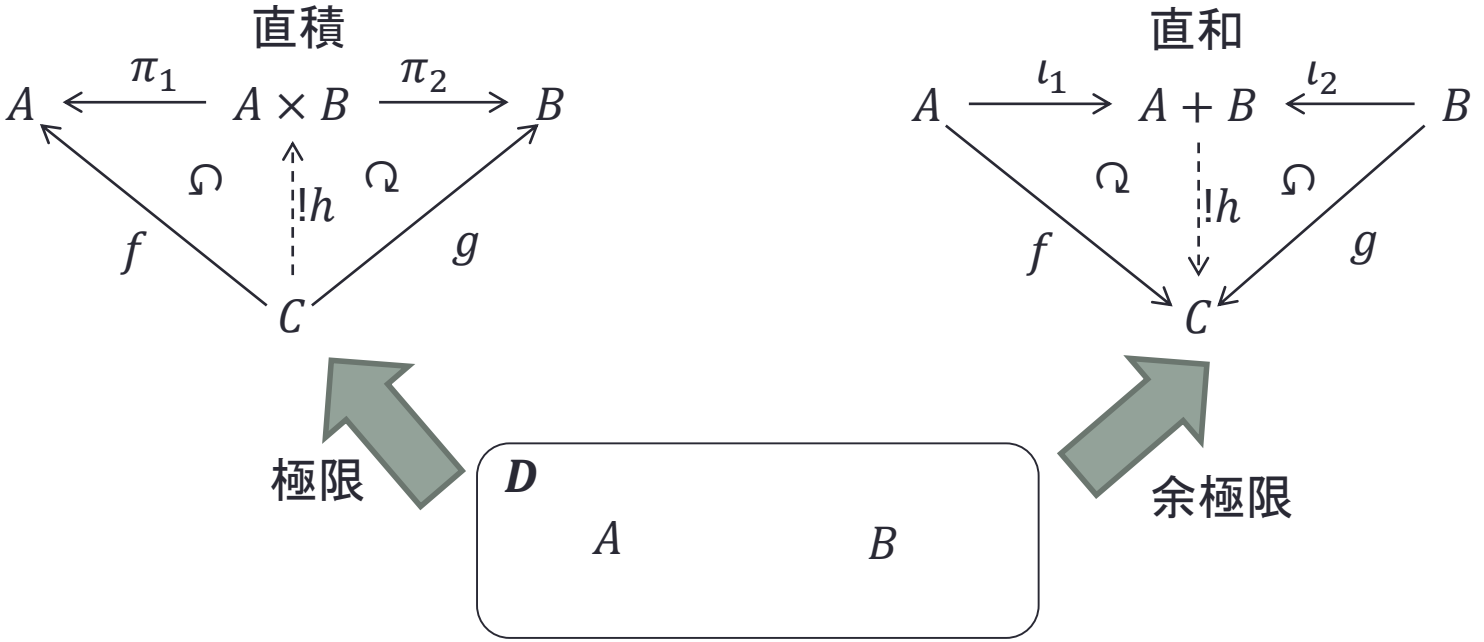




$\varinjlim \boldsymbol{D}$ と $\varprojlim \boldsymbol{D}$ は双対

すべては極限と余極限

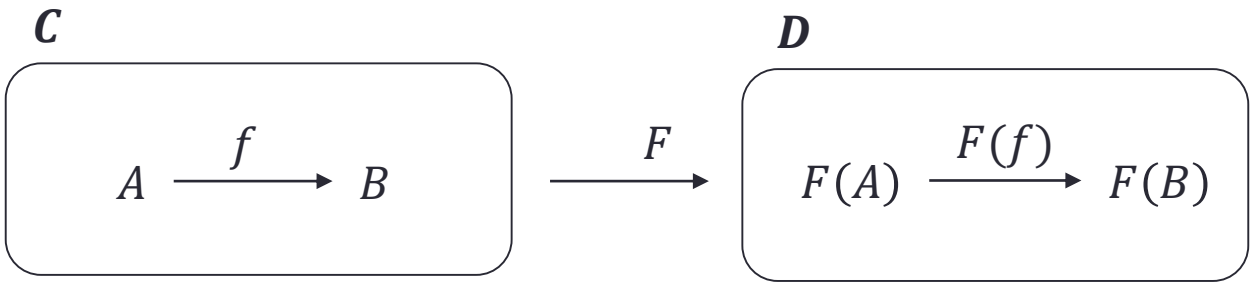
- 積と和



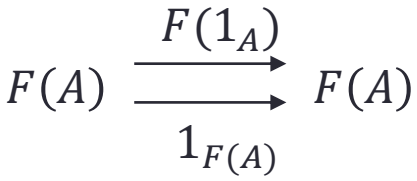
- 終対象と始対象

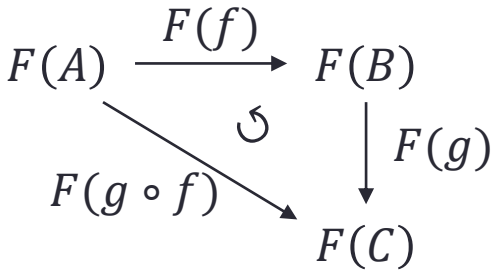
関手（Functor）

- 圏 $\boldsymbol{C}$ から 圏 $\boldsymbol{D}$ への関手 $F\colon \boldsymbol{C} \to \boldsymbol{D}$
  - 対象 $A \in \boldsymbol{C}$ に対して，$F(A) \in \boldsymbol{D}$
  - 射 $f\colon A \to B \in \boldsymbol{C}$ に対して，$F(f)\colon F(A) \to F(B) \in \boldsymbol{D}$



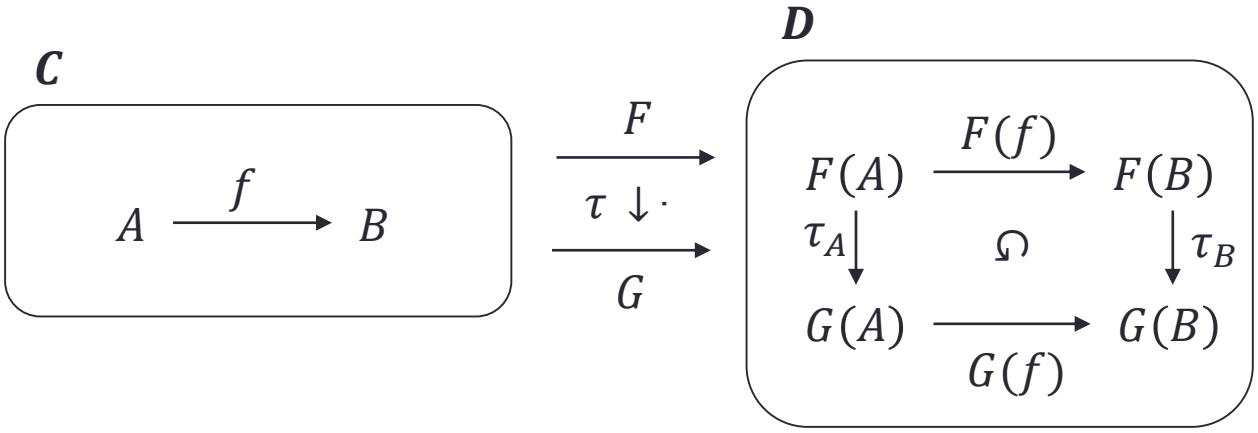
- $A \in \boldsymbol{C}$ に対して，$F(1_A) = 1_{F(A)}$
- $f\colon A \to B,\ g\colon B \to C \in \boldsymbol{C}$ に対して，$F(g \circ f) = F(g) \circ F(f)$



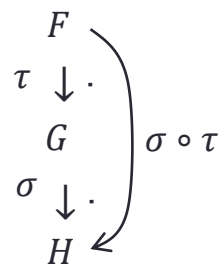


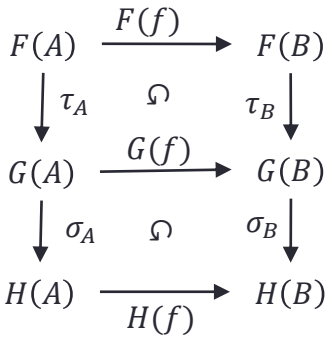
自然変換（natural transformation）

- 圏 $\boldsymbol{C}$ から 圏 $\boldsymbol{D}$ への2つの関手 $F, G: \boldsymbol{C} \to \boldsymbol{D}$ に対して $\tau: F \dot{\to} G$ が自然変換であるとは：
  - 対象 $A \in \boldsymbol{C}$ に対して，$\boldsymbol{D}$ の射 $\tau_A: F(A) \to G(A)$ が存在する．
  - 射 $f: A \to B \in C$ に対して，$\tau_B \circ F(f) = G(f) \circ \tau_A$ が成り立つ．



- 圏 $\boldsymbol{D}^{\boldsymbol{C}}$
  - 対象：圏 $\boldsymbol{C}$ から 圏 $\boldsymbol{D}$ への関手
  - 射：関手間の自然変換

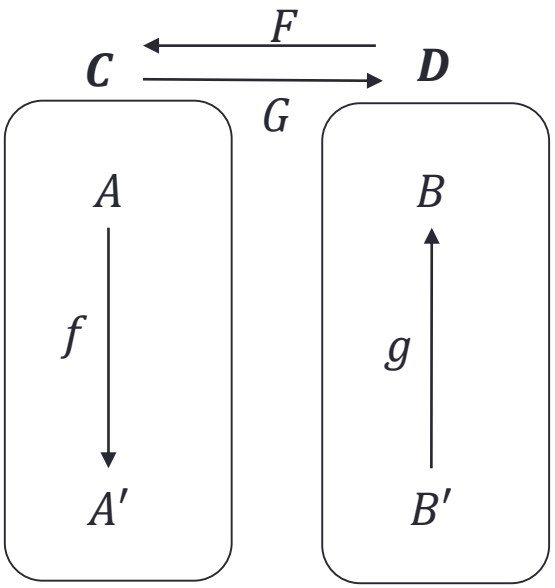




随伴（adjunction）

- 圏 $\boldsymbol{C}$ から 圏 $\boldsymbol{D}$ の随伴（adjunction）
  - 左随伴関手（left adjoint functor）$F\colon \boldsymbol{D} \to \boldsymbol{C}$
  - 右随伴関手（right adjoint functor）$G\colon \boldsymbol{C} \to \boldsymbol{D}$
  - $\hom_{\boldsymbol{C}}(F(B), A) \simeq \hom_{\boldsymbol{D}}\big(B, G(A)\big)$ が $A \in \boldsymbol{C}$ と $B \in \boldsymbol{D}$ に対して自然な同型となっている.
  - $F \dashv G$ と書かれる

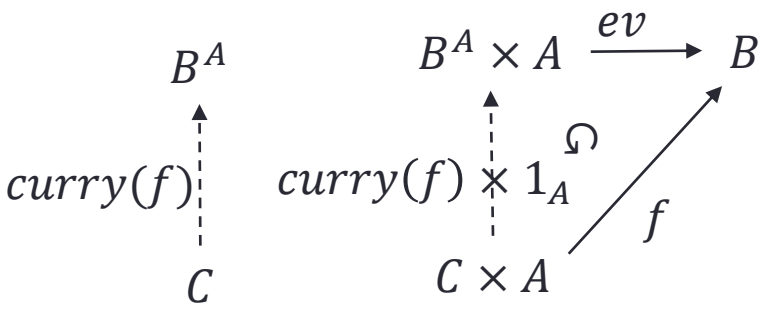
$$\hom_{\boldsymbol{C}}(F(-), -) \simeq \hom_{\boldsymbol{D}}\big(-, G(-)\big) : \boldsymbol{D}^{op} \times \boldsymbol{C} \to \boldsymbol{Set}$$



$$\begin{array}{ccc} \hom_{\boldsymbol{C}}(F(B), A) & \xleftrightarrow{\cong} & \hom_{\boldsymbol{D}}\big(B, G(A)\big) \\ \Big\downarrow \hom_{C}(F(g), f) & \circlearrowleft & \Big\downarrow \hom_{D}\big(g, G(f)\big) \\ \hom_{\boldsymbol{C}}(F(B'), A') & \xleftrightarrow{\cong} & \hom_{\boldsymbol{D}}\big(B', G(A')\big) \end{array}$$

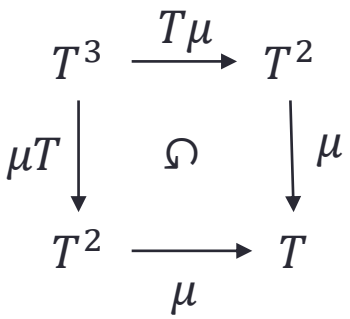
すべては随伴

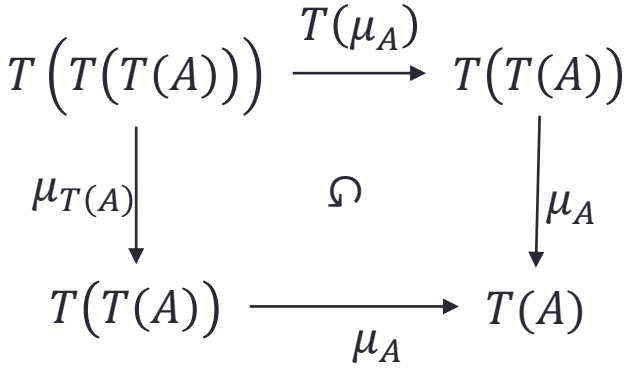
- $\Delta: \boldsymbol{C} \to \boldsymbol{C} \times \boldsymbol{C}$ を $\Delta(A) = (A, A)$ とする対角線関手とすると
  - $+ \dashv \Delta \dashv \times$
  - 直和は Δ の左随伴関手
  - 直積は Δ の右随伴関手
- $! : \boldsymbol{C} \to \cdot$ を $!(A) = \cdot$ とする関手とすると
  - 始対象は $!$ の左随伴関手
  - 終対象は $!$ の右随伴関手
  - ここで，$\cdot$ は対象が1つ，射が恒等射だけの圏
- $\boldsymbol{D}$ を $\boldsymbol{C}$ の部分圏としたとき，$\Delta: \boldsymbol{C} \to \boldsymbol{C}^{\boldsymbol{D}}$ を $\Delta(A)(B) = (A)$, $\Delta(A)(f) = 1_A$ とする関手とすると
  - 余極限 $\varinjlim \boldsymbol{D}$ は Δ の左随伴関手
  - 極限 $\varprojlim \boldsymbol{D}$ は Δ の右随伴関手
- $(-) \times A: \boldsymbol{C} \to \boldsymbol{C}$ の右随伴関手 $(-)^A: \boldsymbol{C} \to \boldsymbol{C}$ が関数空間
  - $\hom_{\boldsymbol{C}}(C \times A, B) \simeq \hom_{\boldsymbol{C}}(C, B^A)$



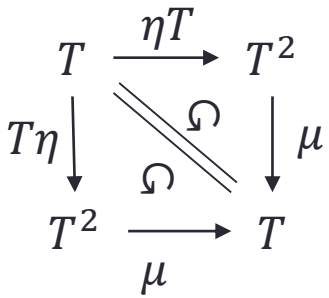
モナド（Monad）

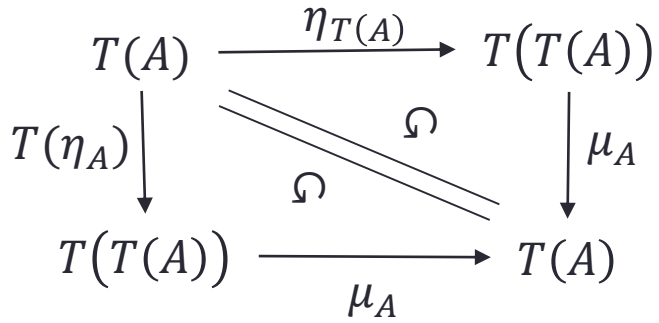
- 圏 $\boldsymbol{C}$ 上のモナドは関手 $T: \boldsymbol{C} \to \boldsymbol{C}$ と2つの自然変換 $\eta: 1_{\boldsymbol{C}} \dot{\to} T$ と $\mu: T^2 \dot{\to} T$ からなり，次の条件をみたす：
  - $\mu \circ T\mu = \mu \circ \mu T$
  - $\mu \circ T\eta = \mu \circ \eta T = 1_T$





半群の結合法則





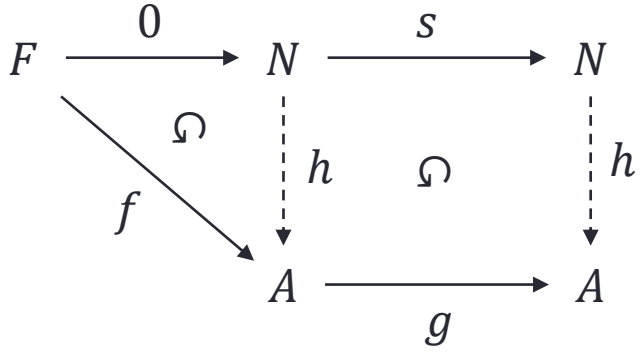
半群の単位元の存在

- $F \dashv G$ のとき $T = G \circ F$ はモナドである.

自然数対象（Natural Number Object）

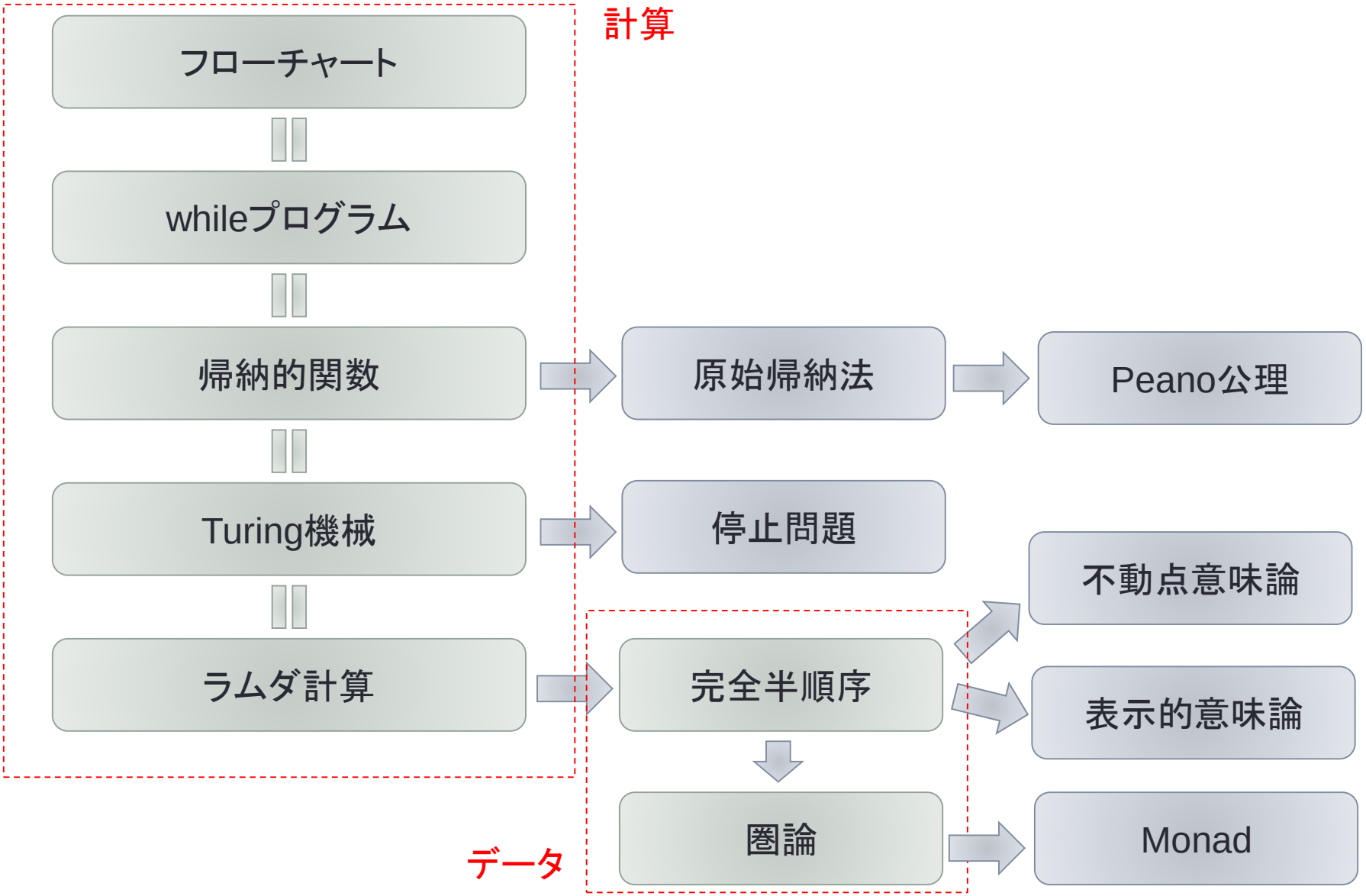
- N が**自然数対象**（Natural Number Object）であるとは
  - 2つの射がある
    - $0: 1 \to N$
    - $s: N \to N$
  - $f: 1 \to A, g: A \to A$ に対して，$h: N \to A$ が唯一存在し
    - $h \circ 0 = f$
    - $h \circ s = g \circ h$

    となる．



- h を $pr(f, g)$ と書くことにする．

情報数学


計算
フローチャート
whileプログラム
帰納的関数
Turing機械
ラムダ計算
原始帰納法
停止問題
Peano公理
不動点意味論
完全半順序
表示的意味論
圏論
Monad
データ