# ループ書いたら負けかなと思っている

uskz

# 自己紹介

梶本裕介

25歳

# ループ嫌い

抽象度が低すぎる　accumulate?　transform?

範囲とかいちいち気にしないといけないのが嫌

形式的な扱いが面倒くさい

```
int f(int a, int b, int c)
{
    return 3 * a + 3 * b + 3 * c;
}
```

```
int f(int a, int b, int c)
{
    return 3 * (a + b + c);
}
```

```
bool p(bool a, bool b, bool c)
{
    return (!a && b || a && !b) != c;
}
```

```
bool p(bool a, bool b, bool c)
{
    return a == b == c;
}
```

# ホーア論理

Hoare Triple $\{P\}S\{Q\}$

Pは事前条件で，Qは事後条件．{P}S{Q}が正しいことはPが成り立つ全ての状態でSを実行するとQが成り立つ状態で終了することを主張している．

代入の公理図式 $\{P[x/E]\}x := E\{P\}$

逐次規則 $$\frac{\{P\}S\{Q\},\{Q\}T\{R\}}{\{P\}S;T\{R\}}$$

条件規則 $$\frac{\{B \wedge P\}S\{Q\},\{\neg B \wedge P\}T\{Q\}}{\{P\}\text{if } B \text{ then } S \text{ else } T \text{ fi}\{Q\}}$$

結果規則 $$\frac{P' \Rightarrow P,\{P\}S\{Q\},Q \Rightarrow Q'}{\{P'\}S\{Q'\}}$$

# ループの推論はめんどい

$n \geq 0$ の時

```
int r = 1;
for (int i = 1; i <= n; ++i)
    r *= i;
```

が実行された際に, $r = n!$ となることを示す（値の定義域は無視する）.

$$\{Q\}\,\text{initialization}\,;\,\textbf{while}\,B\,\textbf{do}\,S\,\textbf{done}\,\{R\}$$

・Pはループ不変条件. $\{P \wedge B\}S\{P\}$

・ループ実行前にPが成り立つ.

・ループの実行が停止する.

・ループの停止直後にRが成り立つ. $P \wedge \neg B \Rightarrow R$

ループの実行が停止することを示すには次の2点を満たす整数式Tを見つければ十分.

・Tは各繰り返しで減少する. すなわち新しい変数vに対し,
$\{P \wedge B\}v := T\,;S\{T < v\}$

・実行すべき繰り返しがある間はT>0が成り立つ.
$P \wedge B \Rightarrow T > 0$

# ループの推論はめんどい

まず，ループ不変条件Pは，

$$r = (i-1)! \wedge 1 \leq i \leq n+1$$

であり，これを示すには，

$$\{P \wedge i \leq n\}\, r := ir\,;\, i := i+1\, \{P\}$$

を証明すれば良い．

代入列$r := ir\,;\, i := i+1$ を真に定める最弱事前条件は $P[i/i+1][r/ir]$ なので，

$$P \wedge i \leq n \Rightarrow P[i/i+1][r/ir]$$

を示す．

$$\begin{aligned}
&P[i/i+1][r/ir] \\
\Leftrightarrow &(r = (i-1)! \wedge 1 \leq i \leq n+1)[i/i+1][r/ir] \\
\Leftrightarrow &(r = i! \wedge 1 \leq i+1 \leq n+1)[r/ir] \\
\Leftrightarrow &ir = i! \wedge 1 \leq i+1 \leq n+1 \\
\Leftrightarrow &ir = i! \wedge 0 \leq i \leq n \\
\Leftarrow &r = (i-1)! \wedge 0 \leq i \leq n \\
\Leftarrow &r = (i-1)! \wedge 1 \leq i \leq n+1 \wedge i \leq n \\
\Leftrightarrow &P \wedge i \leq n
\end{aligned}$$

# ループの推論はめんどい

次に，ループが停止することを示す為の整数式Tを探す．これは，$n+1-i$ である．

なぜなら，新しい変数vに対し，

$$(n+1-i)[i/i+1][r/ri][v/n+1-i]$$
$$\Leftrightarrow n-i \leq n+1-i$$
$$\Leftrightarrow \mathrm{True}$$

だから，

$$\{P \wedge i \leq n\} v := n+1-i\, r := ir\,;\, i := i+1 \{n+1-i < v\}$$

であり，$n+1-i$ は繰り返しのたびに減少し，また，

$$P \wedge i \leq n$$
$$\Leftrightarrow r = (i-1)! \wedge 1 \leq i \leq n+1 \wedge i \leq n$$
$$\Rightarrow i \leq n+1$$
$$\Leftrightarrow 0 \leq n+1-i$$

より，

$$P \wedge i \leq n \Rightarrow 0 < n+1-i$$

だから，$n+1-i$ は繰り返しが残っている間は0より大きい．

以上より，次のことが言えた．

$$\{P\} \text{while}\, i \leq n \,\text{do}\, r := ir\,;\, i := i+1 \,\text{done} \{P \wedge n < i\}$$

# ループの推論はめんどい

次に，$n \geq 0$ なら，ループ開始前にPが成り立つこと，すなわち次を示す．

$\{n \geq 0\} r := 1; i := 1 \{P\}$

を示す．

$P[i/1][r/1]$

$\Leftrightarrow (r = (i-1)! \wedge 1 \leq i \leq n+1)[i/1][r/1]$

$\Leftrightarrow 1 = 0! \wedge 1 \leq 1 \leq n+1$

$\Leftrightarrow n \geq 0$

最後に，$r = n!$ が $P \wedge n < i$ より弱いこと，すなわち，

$P \wedge n < i \Rightarrow r = n!$

を示す．

$P \wedge n < i$

$\Leftrightarrow r = (i-1)! \wedge 1 \leq i \leq n+1 \wedge n < i$

$\Rightarrow r = (i-1)! \wedge i = n+1$

$\Rightarrow r = n!$

以上で階乗が計算できることが示された．

# STLのalgorithm

```
transform(v.begin(), v.end(), back_inserter(u), _1 * _1)

int n = accumulate(v.begin(), v.end(), 1, _1 * _2);

int m = inner_product(v.begin(), v.end(), u.begin(), 0);
```

いちいちiteratorのpairを扱うのが面倒くさい.
containerに連続して適用したい.

# Range

Nを1から初めてその2乗を足していき,
和が2000を初めて超えたとき和はいくつになるか

http://www.cworld2000.com/blog/archives/2008/05/post_19.html
http://d.hatena.ne.jp/NyaRuRu/20080527/p1

```
using namespace pstade::oven;
using namespace boost::lambda;

int v = counting(1, max_count)
        |transformed(regular(_1 * _1))
        |scanned(regular(_1 + _2))
        |dropped_while(regular(_1 <= 2000))
        |front;
```

# リスト

Rangeの推論にリスト（列）の理論が使える．気がする．

$$\mathrm{Listr}\, A = \varepsilon \mid A :_r \mathrm{Listr}\, A$$

$$\mathrm{Listl}\, A = \varepsilon \mid \mathrm{Listl}\, A :_l A$$

$$[1,2,3] = 1:(2:(3:\varepsilon)) = 1:2:3:\varepsilon$$

# リストの連結

$$[1,2]{+}{+}[3,4,5]=[1,2,3,4,5]$$

$$(+\!+):\mathrm{Listr}\,A\times\mathrm{Listr}\,A\to\mathrm{Listr}\,A$$
$$\varepsilon +\!+\, ys = ys$$
$$(x:xs) +\!+\, ts = x:(xs +\!+\, ys)$$

**結合律** $(xs +\!+\, ys) +\!+\, zs = xs +\!+ (ys +\!+\, zs)$

**単位元** $xs +\!+\, \varepsilon = \varepsilon +\!+\, xs = xs$

$$\mathrm{concat}[[1,2],[\,],[3,4,5]]=[1,2,3,4,5]$$

$$\mathrm{concat}:\mathrm{Listr}(\mathrm{Listr}\,A)\to\mathrm{Listr}\,A$$
$$\mathrm{concat}\,\varepsilon=\varepsilon$$
$$\mathrm{concat}(xs:xss)=xs +\!+\,\mathrm{concat}\,xss$$

**++上のconcatの分配律**

$$\mathrm{concat}(xs +\!+\, ys)=\mathrm{concat}\,xs +\!+\,\mathrm{concat}\,ys$$

# リストの反転

$$\mathrm{reverse}[1,2,3,4,5]=[5,4,3,2,1]$$

$$\mathrm{reverse} : \mathrm{Listr}\ A \to \mathrm{Listr}\ A$$
$$\mathrm{reverse}\ \varepsilon = \varepsilon$$
$$\mathrm{reverse}(x : xs) = \mathrm{reverse}\ xs +\!+ [x]$$

$$\mathrm{reverse}(\mathrm{reverse}\ xs) = xs$$

# リストの長さ

$$\mathrm{length}[1,2,3,4,5]=5$$

$$\mathrm{length} : \mathrm{Listr}\ A \to \mathbb{N}$$
$$\mathrm{length}\ \varepsilon = 0$$
$$\mathrm{length}(x : xs) = 1 + \mathrm{length}\ xs$$

++上の#の分配律

$$\mathrm{length}(xs +\!+ ys) = \mathrm{length}\ xs + \mathrm{length}\ ys$$

# take, drop

$$\mathrm{take}\,3\,[1,2,3,4,5]=[1,2,3]$$
$$\mathrm{drop}\,3\,[1,2,3,4,5]=[4,5]$$

$$\mathrm{take} : \mathbb{N} \to \mathrm{Listr}\ A \to \mathrm{Listr}\ A$$
$$\mathrm{take}\,0\,xs = \varepsilon$$
$$\mathrm{take}\,(n+1)\,\varepsilon = \varepsilon$$
$$\mathrm{take}\,(n+1)\,(x : xs) = x : \mathrm{take}\,n\,xs$$

$$\mathrm{drop} : \mathbb{N} \to \mathrm{Listr}\ A \to \mathrm{Listr}\ A$$
$$\mathrm{drop}\,0\,xs = xs$$
$$\mathrm{drop}\,(n+1)\,\varepsilon = \varepsilon$$
$$\mathrm{drop}\,(n+1)\,(x : xs) = \mathrm{drop}\,n\,xs$$

$$\mathrm{take}\,n\,xs +\!\!+ \mathrm{drop}\,n\,xs = xs$$
$$\mathrm{take}\,m \circ \mathrm{take}\,n = \mathrm{take}\,(m \downarrow n)$$
$$\mathrm{drop}\,m \circ \mathrm{drop}\,n = \mathrm{drop}\,(m+n)$$
$$\mathrm{take}\,m \circ \mathrm{drop}\,n = \mathrm{drop}\,n \circ \mathrm{take}\,(m+n)$$

$$\mathrm{splitAt}\,n\,xs = (\mathrm{take}\,n\,xs\,,\,\mathrm{drop}\,n\,xs)$$

# Listr

$$\mathrm{Listr}\,(\lambda x.\,x^2)\,[1,2,3,4,5]=[1,4,9,16,25]$$

$$\mathrm{Listr} : (A \to B) \to \mathrm{Listr}\,A \to \mathrm{Listr}\,B$$
$$\mathrm{Listr}\,f\,\varepsilon = \varepsilon$$
$$\mathrm{Listr}\,f\,(x : xs) = f\,x : \mathrm{Listr}\,f\,xs$$

$$\mathrm{Listr}\,\mathrm{id} = \mathrm{id}$$
$$\mathrm{Listr}\,f \circ g = \mathrm{Listr}\,f \circ \mathrm{Listr}\,g$$

$$f \circ \mathrm{head} = \mathrm{head} \circ \mathrm{Listr}\,f$$
$$\mathrm{Listr}\,f \circ \mathrm{tail} = \mathrm{tail} \circ \mathrm{Listr}\,f$$
$$\mathrm{Listr}\,f \circ \mathrm{reverse} = \mathrm{reverse} \circ \mathrm{Listr}\,f$$
$$\mathrm{Listr}\,f \circ \mathrm{concat} = \mathrm{concat} \circ \mathrm{Listr}\,(\mathrm{Listr}\,f)$$

特に, $\mathrm{concat}\,[xs, ys] = xs +\!+ ys$ なので,

$$\mathrm{Listr}\,f\,(xs +\!+ ys) = \mathrm{Listr}\,f\,xs +\!+ \mathrm{Listr}\,f\,ys$$

# filter

$$\text{filter}\,\text{odd}\,[1,2,3,4,5]=[1,3,5]$$

$$\text{filter}:(A\to\text{Bool})\to\text{Listr}\,A\to\text{Listr}\,A$$
$$\text{filter}\;p=\text{concat}\circ\text{Listr}\,(p\to\text{wrap},\text{const}\,\varepsilon)$$

$$\begin{array}{l}(p\to f,g)\,a\\ \quad\mid p\,a\qquad\quad=f\,x\\ \quad\mid otherwise=g\,x\end{array}$$

$$\text{wrap}\,x=x:\varepsilon$$
$$\text{const}\,x\,a=x$$

$$\text{filter}\;p\circ\text{filter}\,q=\text{filter}\,((\wedge)\circ\langle p,q\rangle)$$
$$\langle f,g\rangle\,x=(f\,x,g\,x)$$

$$\text{filter}\;p\circ\text{concat}=\text{concat}\circ\text{Listr}\,(\text{filter}\,f)$$

# zip, unzip

$$\mathrm{zip}([1,2,3],[a,b,c])=[(1,a),(2,b),(3,c)]$$

$$\mathrm{zip}:\mathrm{Listr}\,A\times\mathrm{Listr}\,A\to\mathrm{Listr}\,(A\times B)$$
$$\mathrm{zip}(\varepsilon,\varepsilon)=\varepsilon$$
$$\mathrm{zip}(x:xs,y:ys)=(x,y):\mathrm{zip}(xs,ys)$$

$$\mathrm{unzip}:\mathrm{Listr}\,(A\times B)\to\mathrm{Listr}\,A\times\mathrm{Listr}\,A$$
$$\mathrm{unzip}=\langle\mathrm{Listr}\,\pi_0,\mathrm{Listr}\,\pi_1\rangle$$

$$\mathrm{zip}\circ\mathrm{unzip}=\mathrm{id}$$

$$\mathrm{zip}\circ(\mathrm{Listr\ f}\times\mathrm{Listr\ g})=\mathrm{Listr}\,(f\times g)\circ\mathrm{zip}$$

$$(\mathrm{Listr\ f}\times\mathrm{Listr\ g})\circ\mathrm{unzip}=\mathrm{unzip}\circ\mathrm{Listr}\,(f\times g)$$

$$f\times g=\langle f\circ\pi_0,g\circ\pi_1\rangle$$

# fold

$$\text{foldr}(c, \star)(x_1 : (x_2 : (x_3 : (x_4 : \varepsilon)))) = x_1 \star ((x_2 \star (x_3 \star (x_4 \star c))))$$

$$\text{foldr} : A \times (B \times A \to A) \to \text{List}\, B \to A$$
$$\text{foldr}(c, h)\, \varepsilon = c$$
$$\text{foldr}(c, h)(x : xs) = h(x, \text{foldr}(c, h)\, xs)$$

$$\text{concat} = \text{foldr}(\varepsilon, ++)$$
$$\text{sum} = \text{foldr}(0, +)$$
$$\text{product} = \text{foldr}(1, \times)$$
$$\text{and} = \text{foldr}(\text{True}, \wedge)$$
$$\text{List}\, f = \text{foldr}(\varepsilon, h)\ \textit{where}\ h(x, xs) = f\, x : xs$$
$$\text{length} = \text{sum} \circ \text{List}(\text{const}\, 1)$$
$$\text{filter}\, p = \text{foldr}(\varepsilon, (p \circ \pi_0 \to (:), \pi_1))$$

# Listl上のfold

$$\mathrm{foldl} : A \times (B \times A \to A) \to \mathrm{Listl}\, B \to A$$
$$\mathrm{foldl}(c, h)\, \varepsilon = c$$
$$\mathrm{foldl}(c, h)(xs : x) = h(\mathrm{foldl}(c, h)\, xs, x)$$

実際にはListr上で定義されていることが多い.

$$\mathrm{foldl} : A \times (B \times A \to A) \to \mathrm{Listr}\, B \to A$$
$$\mathrm{foldl}(c, h)\, \varepsilon = c$$
$$\mathrm{foldl}(c, h)(x : xs) = \mathrm{foldl}(h(c, x), h)\, xs$$
$$\mathrm{reverse} = \mathrm{foldl}(\varepsilon, \mathrm{cons})\ \mathit{where}\ \mathrm{cons}\, xs\, x = x : xs$$

# 第一双対性定理

☆が結合的で単位元eを持つなら,

$$\mathrm{foldr}(e, ☆)\, xs = \mathrm{foldl}(e, ☆)\, xs$$

# 第二双対性定理

任意のx,y,zに対し, ☆と★とeについて,

$$x ☆ (y ★ z) = (x ☆ y) ★ z$$

$$x ☆ e = e ★ x$$

ならば,

$$\mathrm{foldr}(e, ☆)\, xs = \mathrm{foldl}(e, ★)\, xs$$

# 第三双対性定理

$$\mathrm{foldr}(e, f)\, xs = \mathrm{foldl}(e, \mathrm{flip}\, f)(\mathrm{reverse}\, xs)$$

## foldrの融合律

$f\ a = b \wedge \forall x, y[f\ (g(x, y)) = h(x, f\ y)]$ ならば,

$f \circ \mathrm{foldr}(a, g) = \mathrm{foldr}(b, h)$

## foldlの融合律

$f\ a = b \wedge \forall x, y[f\ (g(x, y)) = h(f\ x, y)]$ ならば,

$f \circ \mathrm{foldl}(a, g) = \mathrm{foldl}(b, h)$

## foldとListrの融合律

$\mathrm{foldr}(a, f) \circ \mathrm{Listr}\ g = \mathrm{foldr}(a, f \circ g)$

# 非空リスト

$$\mathrm{Listr}^{+}\,A = \mathrm{wrap}\,A \mid A :_{r} \mathrm{Listr}^{+}\,A$$

$$\mathrm{Listl}^{+}\,A = \mathrm{wrap}\,A \mid \mathrm{Listl}^{+}\,A :_{l} A$$

$$\mathrm{foldr}^{+}(f, g)(\mathrm{wrap}\,x) = f\,x$$
$$\mathrm{foldr}^{+}(f, g)(x : xs) = g(x, \mathrm{foldr}^{+}(f, g)\,xs)$$

$$\mathrm{head} = \mathrm{foldr}^{+}(\mathrm{id}, \pi_0)$$

$$\mathrm{foldr1}\,f = \mathrm{foldr}^{+}(\mathrm{id}, f)$$

$$\mathrm{maximum} = \mathrm{foldr1}\,(\uparrow)$$

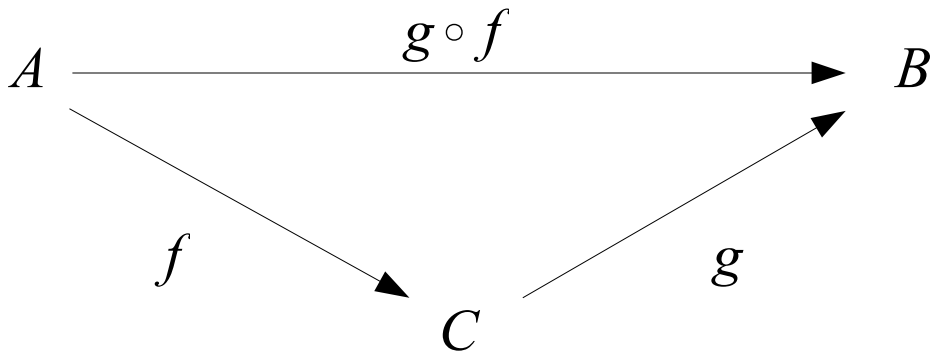
# 圏

圏 $\boldsymbol{C}$ は対象(object)の集まりと射(morphism)の集まりからなる.

$\boldsymbol{C}$ の任意の射fには始域と呼ばれる対象と終域と呼ばれる対象が備わっている.

$$dom\ f = A \wedge cod\ f = B \qquad f : A \to B \qquad A \overset{f}{\to} B$$

$\boldsymbol{C}$の任意の射f, gに対して, dom g = cod fのとき射の合成 $g \circ f$ が存在する.




射の合成に対して結合律が成り立つ.

$$(\forall\ f : A \to B, g : B \to C, h : C \to D)[h \circ (g \circ f) = (h \circ g) \circ f]$$

$\boldsymbol{C}$の任意の対象Aに対して次を満たす恒等射 $id_A : A \to A$ が存在する.

$$(\forall\ f : A \to B)[id_B \circ f = f = f \circ id_A]$$

# 集合と全域関数の圏

圏 ***Set***の対象は集合

射は型付の全域写像 $(f, A, B)\, where\ Dom\ f = A, Ran\ f \subset B$

恒等射 $id_A : A \to A$ はA上の恒等写像

射 $(f, A, B)$ と射 $(g, C, D)$ の合成射はB=Cのとき，またそのときにのみ定義され，$(g, C, D) \circ (f, A, B) = (g \circ f, A, D)$

# 積圏

圏$\boldsymbol{A}\times\boldsymbol{B}$の対象は$\boldsymbol{A}$の対象$A$と$\boldsymbol{B}$の対象$B$の対$(A,B)$

射は$\boldsymbol{A}$の射$f$と$\boldsymbol{B}$の射$g$の対$(f,g)$

恒等射$id_{A\times B}$は$(id_A, id_B)$

$(f,g)\circ(h,k)=(f\circ h, g\circ k)$

## モノ射

$(\forall f, g: C \to A)[f = g \Leftrightarrow m \circ f = m \circ g]$

が成り立つ時, $m: A \to B$ をモノ射と言う.

***Set*** では単射.

## エピ射

$(\forall f, g: B \to C)[f = g \Leftrightarrow f \circ e = g \circ e]$

が成り立つ時, $e: A \to B$ をエピ射と言う.

***Set*** では全射.

# 同型射

$i : A \to B$ に対して$j : B \to A$ が存在し，$j \circ i = \mathrm{id}_A \wedge i \circ j = \mathrm{id}_B$
が成り立つ時，$i : A \to B$ を同型射と言う.

このような $j$ は存在するなら多くとも1つなので，これを$i^{-1}$ と書く.

***Set***では全単射.

同型射はモノ射かつエピ射であるが，その逆は必ずしも成り立たない.

$i : A \to B$ が同型射のときAとBは同型と言う.

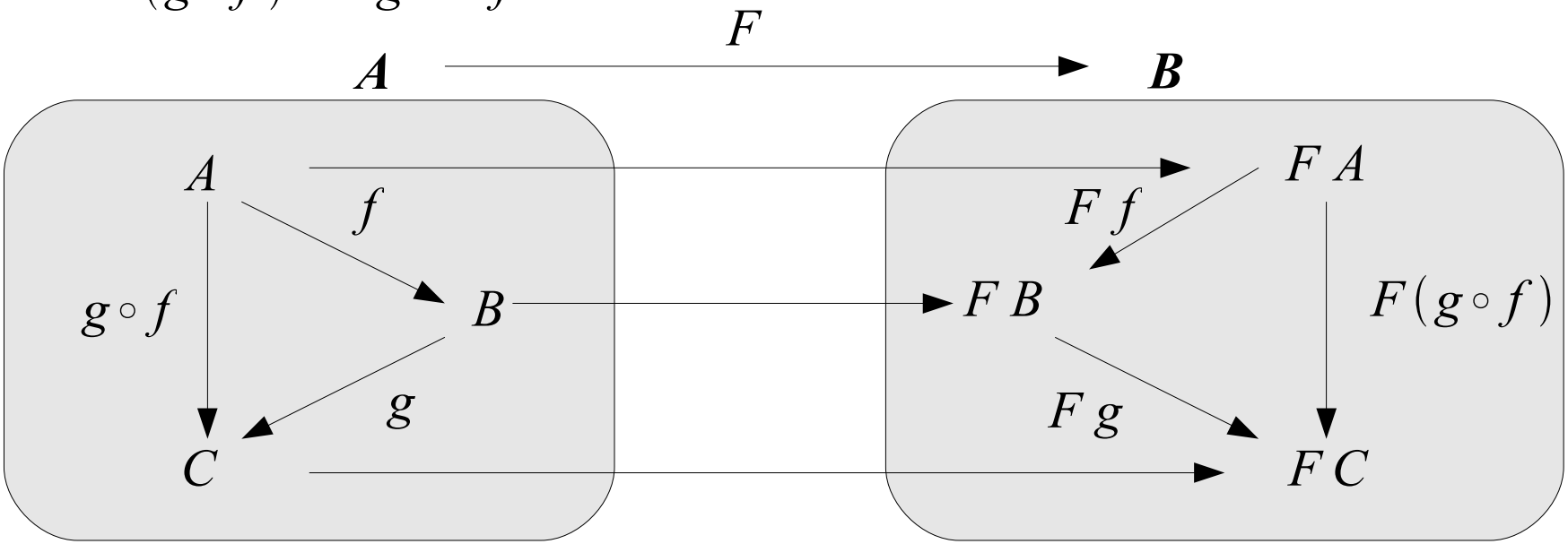
# 関手

**A,B**を圏とする. このとき関手 $F : \boldsymbol{A} \to \boldsymbol{B}$ は対象から対象への写像と射から射への写像からなる.

$$f : A \to B \Rightarrow F\,f : F\,A \to F\,B$$

$$F(id_A) = id_{F\,A}$$

$$F(g \circ f) = F\,g \circ F\,f$$




射の合成同様関手の合成が定義される. $(G \circ F)\,f = G(F\,f)$

関手は(小さな)圏の圏の射となる.

## 恒等関手

$\mathrm{Id} : \boldsymbol{C} \to \boldsymbol{C}$

$\mathrm{Id}\, A = A$

$\mathrm{Id}\, f = f$

## 定関手

$K_B : \boldsymbol{A} \to \boldsymbol{B}$

$K_B A = B$

$K_B f = id_B$

## 二項関手

始域が積圏の関手.

$F(\mathrm{id}_A, \mathrm{id}_B) = id_{F(A,B)}$

$F(f \circ g, h \circ k) = F(f, h) \circ F(g, k)$

## リスト関手

$\mathrm{Listr} : \boldsymbol{Set} \to \boldsymbol{Set}$

$\mathrm{Listr}(f \circ g) = \mathrm{Listr}\, f \circ \mathrm{Listr}\, g$

$\mathrm{Listr}\, \mathrm{id} = \mathrm{id}$

# 始対象と終対象

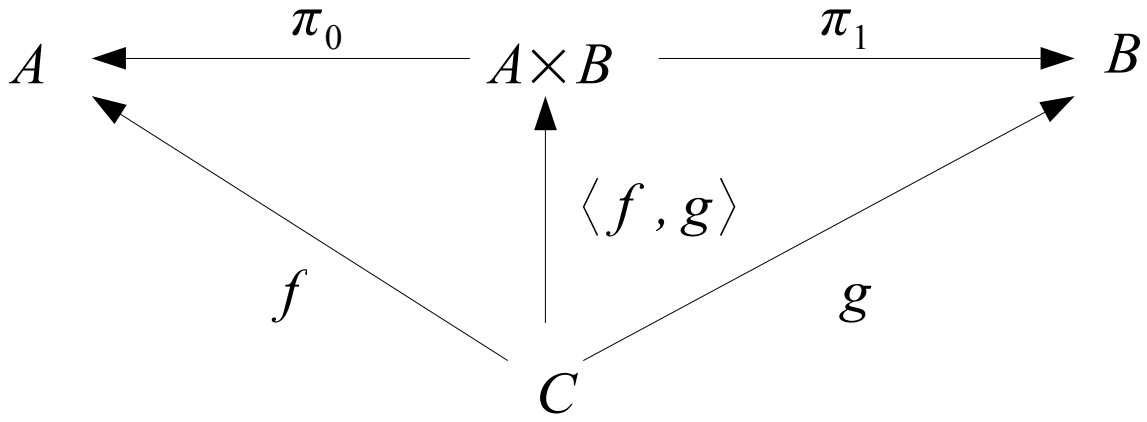
$\boldsymbol{C}$ の任意の対象 $A$ に対し，$0 \to A$ なる射が唯一存在するとき，$0$ を始対象と言う.

$\boldsymbol{C}$ の任意の対象 $A$ に対し，$A \to 1$ なる射が唯一存在するとき，$1$ を終対象と言う.

$\boldsymbol{Set}$ では空集合が始対象で，singleton setが終対象.

# 積

対象 $A$ と対象 $B$ の積とは，任意の対象 $C$ と射の組 $f:C\to A$ と $g:C\to B$ に対し，次の図式を可換にするような射 $\langle f,g\rangle:C\to A\times B$ が唯一存在する，2つの射 $\pi_0:A\times B\to A$ と $\pi_1:A\times B\to B$ を伴った対象 $A\times B$ のこと.



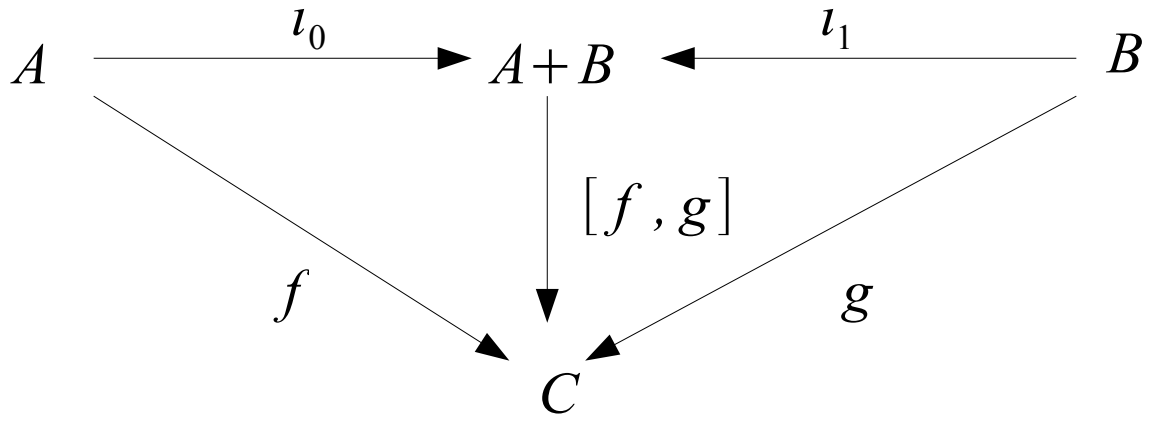

$$h=\langle f,g\rangle \Leftrightarrow \pi_0\circ h=f\wedge\pi_1\circ h=g$$

$$\pi_0\circ\langle f,g\rangle=f\wedge\pi_1\circ\langle f,g\rangle=g$$
$$\mathrm{id}=\langle\pi_0,\pi_1\rangle$$
$$\langle f,g\rangle\circ m=\langle f\circ m,g\circ m\rangle$$

圏の任意の対象の対が積を持つ時，その圏は積を持つと言う.

# 余積

対象 $A$ と対象 $B$ の余積とは，任意の対象 $C$ と射の組 $f:C\to A$ と $g:C\to B$ に対し，次の図式を可換にするような射 $[f,g]:A+B\to C$ が唯一存在する，2つの射 $\iota_0:A\to A+B$ と $\iota_1:A+B\to B$ を伴った対象 $A+B$ のこと.




$$h=[f,g]\Leftrightarrow h\circ\iota_0=f\wedge h\circ\iota_1=g$$

$$[f,g]\circ\iota_0=f\wedge[f,g]\circ\iota_1=g$$

$$\mathrm{id}=[\iota_0,\iota_1]$$

$$m\circ[f,g]=[m\circ f,m\circ g]$$

# 積関手

$\boldsymbol{C}$ が積を持つ時，射に対し $f \times g = \langle f \circ \pi_{0,} g \circ \pi_1 \rangle$ と定義することで2項関手 $\times : \boldsymbol{C} \times \boldsymbol{C} \to \boldsymbol{C}$ が考えられる.

$$\mathrm{id} \times \mathrm{id} = \langle \mathrm{id} \circ \pi_{0,} \mathrm{id} \circ \pi_1 \rangle = \mathrm{id}$$

$$(f \times g) \circ \langle h, k \rangle = \langle f \circ h, g \circ k \rangle$$

$$(f \times g) \circ (h \times k) = (f \circ h) \times (g \circ k)$$

# 余積関手

$f+g=[\iota_0\circ f\,,\iota_1\circ g]$ と定義することで2項関手 $+:\boldsymbol{C}\times\boldsymbol{C}\to\boldsymbol{C}$ が考えられる.

$$\mathrm{id}+\mathrm{id}=[\iota_0\circ\mathrm{id}\,,\iota_1\circ\mathrm{id}]=\mathrm{id}$$

$$[f\,,g]\circ(h+k)=[f\circ h\,,g\circ k]$$

$$(f+g)\circ(h+k)=(f\circ h)+(g\circ k)$$

# 交換律

$$\langle[f\,,g],[h,k]\rangle=[\langle f\,,h\rangle,\langle g\,,k\rangle]$$

# 多項式関手

恒等関手と定関手は多項式関手

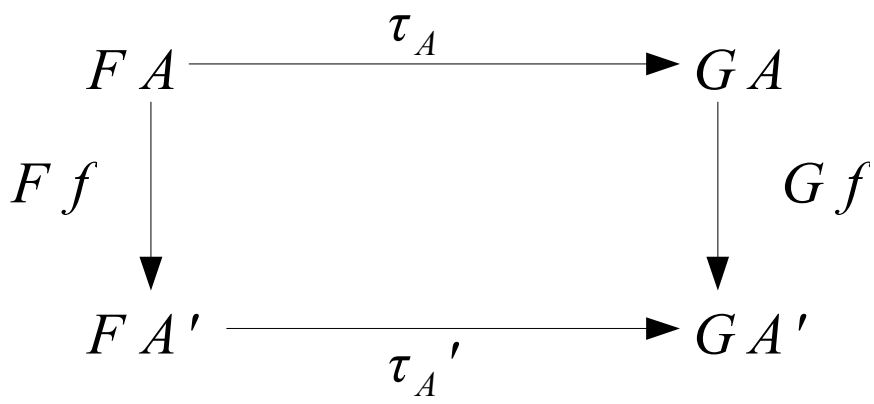
FとGが多項式関手のとき，FG, F+G, F×Gは多項式関手

$$(F+G)h = Fh + Gh$$
$$(F\times G)h = Fh \times Gh$$

例えば $FX = A + K\times A$ と $Ff = \mathrm{id}_A + f\times \mathrm{id}_A$ によって定まる関手Fは，
$F = K_A + (\mathrm{Id}\times K_A)$ なので多項式関手

# 自然変換

関手 $F, G : \boldsymbol{A} \to \boldsymbol{B}$ に対して，自然変換 $\tau : F \to G$ とは，$\boldsymbol{A}$ の各対象 $A$ に，$\boldsymbol{B}$ の射 $\tau_A : FA \to GA$ を割り当てる写像で，$\boldsymbol{A}$ の任意の射 $f : A \to A'$ について次の図式を可換にするもの．




$$Gf \circ \tau_A = \tau_{A'} \circ Ff$$

# 自然変換の例

$$f \circ \mathrm{head} = \mathrm{head} \circ \mathrm{Listr}^{+} f$$

$$\mathrm{Listr}\, f \circ \mathrm{tail} = \mathrm{tail} \circ \mathrm{Listr}^{+} f$$

$$\mathrm{Listr}\, f \circ \mathrm{reverse} = \mathrm{reverse} \circ \mathrm{Listr}\, f$$

$$(\mathrm{Listr} \circ \mathrm{Listr})\, f \circ \mathrm{inits} = \mathrm{inits} \circ \mathrm{Listr}\, f$$

$$\mathrm{Listr}\, f \circ \mathrm{concat} = \mathrm{concat} \circ (\mathrm{Listr} \circ \mathrm{Listr})\, f$$

$$\mathrm{zip} \circ (\mathrm{Listr}\ \mathrm{f} \times \mathrm{Listr}\ \mathrm{g}) = \mathrm{Listr}(f \times g) \circ \mathrm{zip}$$

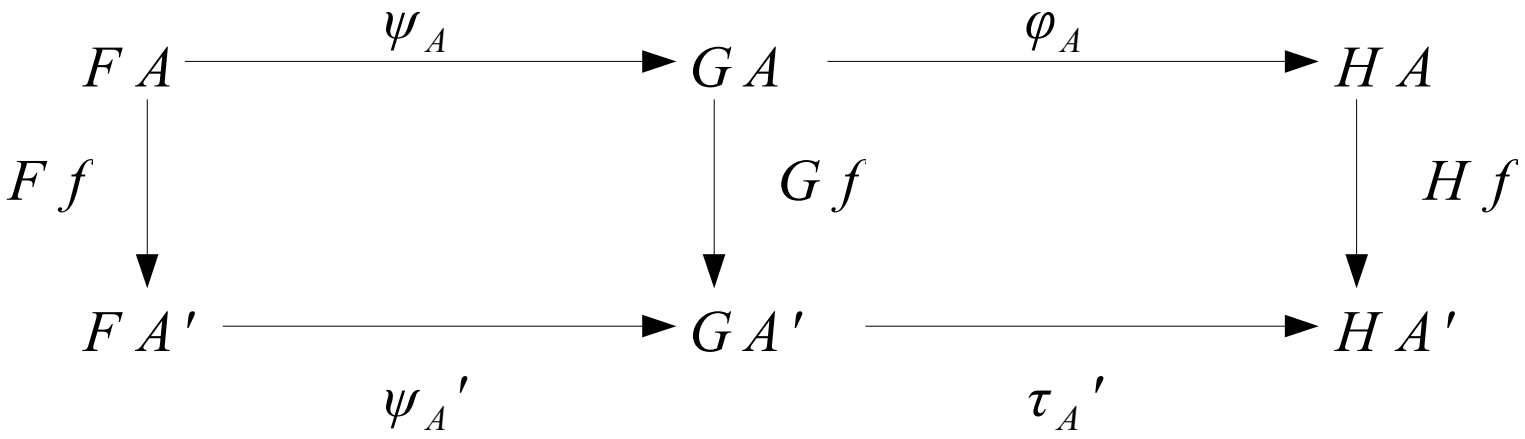
$$(\mathrm{Listr}\ \mathrm{f} \times \mathrm{Listr}\ \mathrm{g}) \circ \mathrm{unzip} = \mathrm{unzip} \circ \mathrm{Listr}(f \times g)$$

# 恒等変換

任意の関手$F$に対して，恒等変換 $\mathrm{id}_F : F \to F$ は，$(\mathrm{id}_F)_A = \mathrm{id}_{FA}$

# 自然変換の合成

$\varphi : G \to H$と$\psi : F \to G$ に対し，$(\varphi \circ \psi)_A = \varphi_A \circ \psi_A$



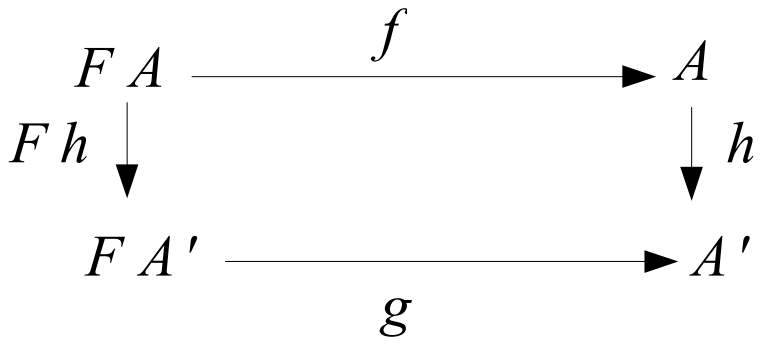

自然変換はその対象が関手である圏の射となる

各componentが全単射の時，自然変換は自然同系射と呼ばれる

# 代数

関手$F : \boldsymbol{C} \to \boldsymbol{C}$ に対して，F-代数とは，型 $A \to F\ A$を持つ射のこと.

$f : A \to F\ A$ から $g : A' \to F\ A'$ へのF-準同型射とは，次の図式を可換にする射 $h : A \to A'$




$$h \circ f = g \circ F\ h$$

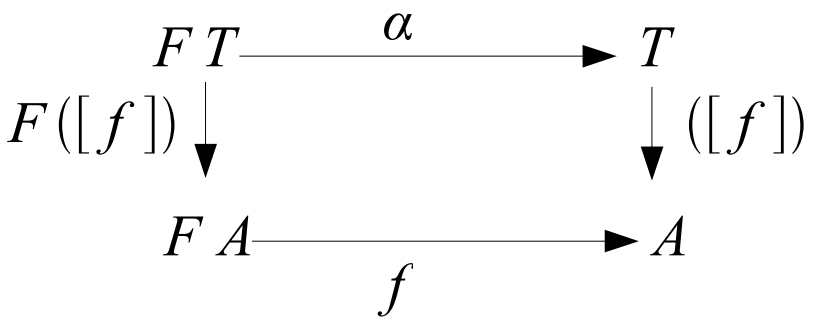
恒等射は準同型射であり，準同型射の合成は準同型射なので

F-代数は射が準同型射な圏 $\boldsymbol{Alg}(F)$ の対象となる.

# 始代数

***Set*** の多項式関手を初め，多くの関手に対して $\boldsymbol{Alg}(F)$ は始対象を持ち，これを $\alpha : F\,T \to T$ で表す.

F-始代数が存在すると言うことは，任意のF-代数 $f : A \to F\,A$ に対し，$\alpha$ から $f$ への準同型射が唯一存在すると言うこと. この準同型射を $([\,f\,]) : T \to A$ で表す.




$$h = ([\,f\,]) \Leftrightarrow h \circ \alpha = f \circ F\,h$$

$([\,f\,])$ の形の射をcatamorphismと言う.

## 評価規則

$$([f])\circ\alpha = f\circ F([f])$$

## 反射律

$$([\alpha]) = \mathrm{id}$$

$$\mathrm{id} = ([f])$$
$$\Leftrightarrow \mathrm{id}\circ\alpha = f\circ F\,\mathrm{id}$$
$$\Leftrightarrow f = \alpha$$

## 融合律

$$h\circ([f]) = ([g]) \Leftarrow h\circ f = g\circ F\,h$$

$$h\circ([f]) = ([g])$$
$$\Leftrightarrow h\circ([f])\circ\alpha = g\circ F(h\circ([f]))$$
$$\Leftrightarrow h\circ([f])\circ\alpha = g\circ F\,h\circ F([f])$$
$$\Leftrightarrow h\circ f\circ F([f]) = g\circ F\,h\circ F([f])$$
$$\Leftarrow h\circ f = g\circ F\,h$$

# 型関手

$\mathrm{Listr}\,A = \varepsilon \mid A :_r \mathrm{Listr}\,A$ は $[\varepsilon, (:_r)] : F_A(\mathrm{Listr}\,A) \to \mathrm{Listr}\,A$ を
$F_A B = 1 + (A \times B)$ かつ $F_A f = \mathrm{id}_1 + (\mathrm{id}_A \times f)$ で定義される関手 $F_A$
の始代数として宣言する.
$F_A$ は二項関手 $F$ の第一引数を $A$ で固定したものと見なす.

$F$ を始代数のコレクション $\alpha_A : F(A, T\,A) \to T\,A$ を伴う二項関手とすると,

$T\,f = ([\alpha \circ F(f, \mathrm{id})])$
と定義することによってTを関手にすることが出来る.

例えばリスト関手は,
$\mathrm{Listr}\,f = ([[\varepsilon, (:_r)] \circ (\mathrm{id} + (f \times \mathrm{id}))])$
により定義される. 変形すると,
$\mathrm{Listr}\,f = ([\varepsilon, (:_r) \circ (f \times \mathrm{id})])$

# catamorphismはfold

型 $\mathrm{List}\, A$ に対し,

$$
\begin{aligned}
& h = ([c, f]) \\
\Leftrightarrow\ & h \circ \alpha = [c, f] \circ Fh \\
\Leftrightarrow\ & h \circ \alpha = [c, f] \circ (\mathrm{id} + (\mathrm{id} \times h)) \\
\Leftrightarrow\ & h \circ \alpha = [c, f \circ (\mathrm{id} \times h)] \\
\Leftrightarrow\ & h \circ [\varepsilon, (:_r)] = [c, f \circ (\mathrm{id} \times h)] \\
\Leftrightarrow\ & [h \circ \varepsilon, h \circ (:_r)] = [c, f \circ (\mathrm{id} \times h)] \\
\Leftrightarrow\ & h \circ \varepsilon = c \wedge h \circ (:_r) = f \circ (\mathrm{id} \times h)
\end{aligned}
$$

これは,

$$
\begin{aligned}
& h\, \varepsilon = c \\
& h\, (x, xs) = f\, (x, h\, xs)
\end{aligned}
$$

ということであり, すなわち,

$$([c, f]) = \mathrm{foldr}\, (c, f)$$

ということ.

# 型関手の融合律

$$([h])\circ T\,g=([h\circ F(g,\mathrm{id})])$$

$$([h])\circ T\,g=([h\circ F(g,\mathrm{id})])$$
$$\Leftrightarrow([h])\circ([\alpha\circ F(g,\mathrm{id})])=([h\circ F(g,\mathrm{id})])$$
$$\Leftarrow([h])\circ\alpha\circ F(g,\mathrm{id})=h\circ F(g,\mathrm{id})\circ F(\mathrm{id},([h]))$$
$$\Leftrightarrow h\circ F(\mathrm{id},([h]))\circ F(g,\mathrm{id})=h\circ F(g,\mathrm{id})\circ F(\mathrm{id},([h]))$$
$$\Leftrightarrow \mathrm{True}$$

例えば $\mathrm{sum}=([0,(+)])$ とすると,

$$\mathrm{sum}\circ\mathrm{Listr}\,f=([0,(+)\circ(f\times\mathrm{id})])$$

# 分配圏

積と余積を持つ圏において，対象 $A\times(B+C)$ と対象$(A\times B)+(A\times C)$ が同型の時，その圏を分配圏と言う.

積と余積を持つ圏では，

$\mathrm{undistl}:(A\times B)+(A\times C)\to A\times(B+C)$

は常に存在する.（例えば $\mathrm{undistl}=[\mathrm{id}\times\iota_0,\mathrm{id}\times\iota_1]$ とすれば良い）

しかし，逆射

$\mathrm{distl}:A\times(B+C)\to(A\times B)+(A\times C)$

が存在するとは限らない.

例えば ***Set*** が分配圏.

# 論理値

分配圏では論理値を$\mathrm{Bool} = 1+1, \mathrm{True} = \iota_0, \mathrm{False} = \iota_1$ とすることにより扱える.

$$\neg = [\mathrm{False}, \mathrm{True}]$$

quad を $\mathrm{Bool}^2 \to 1+1+1+1$ な同型射として,

$$\wedge = [\mathrm{True}, \mathrm{False}, \mathrm{False}, \mathrm{False}] \circ \mathrm{quad}$$

# 条件式

$$(p \to f, g) = [f, g] \circ (\pi_0 + \pi_0) \circ \mathrm{distl} \circ \langle \mathrm{id}, p \rangle$$

$$h \circ (p \to f, g) = (p \to h \circ f, h \circ g)$$

$$(p \to f, g) \circ h = (p \circ h \to f \circ h, g \circ h)$$

$$(p \to f, f) = f$$

# filter

$$\mathrm{filter}\ p = ([\varepsilon, (p \circ \pi_0 \to (:_r), \pi_1)])$$

# Banana-splitの法則

$$\langle([h]),([k])\rangle = ([\langle h \circ F\pi_0, k \circ F\pi_1\rangle])$$

catamorphismの普遍性より,

$$\langle([h]),([k])\rangle \circ \alpha = \langle h \circ F\pi_0, k \circ F\pi_1\rangle \circ F\langle([h]),([k])\rangle$$

を示せばよい.

$$\begin{aligned}
&\langle([h]),([k])\rangle \circ \alpha \\
&= \langle([h \circ \alpha]),([k \circ \alpha])\rangle \\
&= \langle h \circ F([h]), k \circ F([k])\rangle \\
&= \langle h \circ F(\pi_0 \circ \langle([h]),([k])\rangle), k \circ F(\pi_1 \circ \langle([h]),([k])\rangle)\rangle \\
&= \langle h \circ F\pi_0 \circ F\langle([h]),([k])\rangle, k \circ F\pi_1 \circ F\langle([h]),([k])\rangle\rangle \\
&= \langle h \circ F\pi_0, k \circ F\pi_1\rangle \circ F\langle([h]),([k])\rangle
\end{aligned}$$

# リストの横断を纏める

$$\mathrm{average} = (/) \circ \langle \mathrm{sum}, \mathrm{length} \rangle$$

このaverageの定義はリストを2度横断する.

$$\begin{aligned}
&\langle \mathrm{sum}, \mathrm{length} \rangle \\
&= \langle ([0,(+)]), ([0,(+1)\circ\pi_1]) \rangle \\
&= ([\langle [0,(+)]\circ F\pi_0, [0,(+1)\circ\pi_1]\circ F\pi_1 \rangle]) \\
&= ([\langle [0,(+)]\circ(\mathrm{id}+(\mathrm{id}\times\pi_0)), [0,(+1)\circ\pi_1]\circ(\mathrm{id}+(\mathrm{id}\times\pi_1)) \rangle]) \\
&= ([\langle [0,(+)\circ(\mathrm{id}\times\pi_0)], [0,(+1)\circ\pi_1\circ(\mathrm{id}\times\pi_1)] \rangle]) \\
&= ([\langle 0,0 \rangle, \langle (+)\circ(\mathrm{id}\times\pi_0), (+1)\circ\pi_1\circ(\mathrm{id}\times\pi_1) \rangle]) \\
&= ([\langle 0,0 \rangle, \langle (+)\circ(\mathrm{id}\times\pi_0), (+1)\circ\pi_1\circ\pi_1 \rangle])
\end{aligned}$$

λ式を用いると,

$$([\langle 0,0 \rangle, \lambda(a,(b,n)).(a+b, n+1)])$$

# やり残しとこれから

currryingの圏論的扱い

Curry–Howard–Lambek同型対応

F-代数とcatamorphismの双対概念としてのF-余代数とanamorphism

モナドとコモナド

集合と全域関数の圏ではなくpointed CPOと連続関数の圏

関数とcategoryではなく関係とallegory

などなど

# 参考文献


Antony J. T. Davie. An Introduction to Functional Programming Systems Using Haskell. Cambridge University Press, 1992.

Benjamin C. Pierce. Basic Category Theory for Computer Scientists. MIT Press, 1991.

David Gries, and Fred B. Schneider. A Logical Approach to Discrete Math. Springer-Verlag, 1993.

Jeremy Gibbons and Oege De Moor (Eds). The Fun of Programming. Palgrave Macmillan, 2003.

Mac Lane, S. Categories for the Working Mathematician. Springer, 1998.

Richrd Bird. Introduction to Functional Programming using Haskell. Prentice Hall, 1998.

Richrd Bird and Oege De Moor. Algebra of Programming. Prentice Hall, 1997.

Roland Backhouse, Roy Crole and Jeremy Gibbons (Eds). Algebraic and Coalgebraic Methods in the Mathematics of Program Construction. Springer-Verlag, 2002.

Steve Awodey. Category Theory. Oxford University Press, 2006.