圏論原論 I

Sexytant

2023 年 1 月 21 日

目次

第Ⅰ部

準備

第 1 章

集合

1.1 集合

集合 set とは, ひとまず素朴に「ものの集まり」と定義される. 集合を構成する「もの」を**元 element** という.

記法 1.1. 集合は各元をカンマで区切り $\{\}$ で囲むことで表される.

記法 1.2. とある条件を満たす x 全体からなる集合は $\{x \mid x$ についての条件 $\}$ を用いて表す.

実例 1.3. $\{1,2,3\}$ や $\{a,b,c\}$, 自然数全体 $\mathbb{N}$, 整数全体 $\mathbb{Z}$, 実数全体 $\mathbb{R}$, 複素数全体 $\mathbb{C}$ は集合である.

実例 1.4. $\{2a \mid a \in \mathbb{N}\}$ や $\{a^3 \mid a \in \mathbb{N}\}$ は集合である.

実例 1.5. 空集合 empty set $\emptyset = \{\}$ は集合である.

記法 1.6. a が集合 S の元であることを $a \in S$ で表す.

定義 1.7. 集合 A の元のすべてが集合 B の元であるとき A **は** B **の部分集合 subset である**という.

記法 1.8. 集合 A が集合 B の部分集合であることを $A \subset B$ で表す.

定義 1.9. 集合 S が与えられたとき, S の部分集合の全体 $\mathscr{P}S = \{A \mid A \subset S\}$ を S **の冪集合 power set** という.

定義 1.10. 集合 A, B に対して $\{(a,b) \mid a \in A, b \in B\}$ を A **と** B **の直積 direct product** という.

記法 1.11. A と B の直積を $A \times B$ で表す.

1.2 二項関係・写像

1.2.1 二項関係

定義 1.12. 集合 A, B に対して $R \subset A \times B$ であるとき, R **は** A **と** B **の二項関係 binary relation** であるという.

定義 1.13. R が集合 A, B の二項関係であるとする. とある $a \in A, b \in B$ の組 (a,b) が R の元であるとき, a **と** b **に間に** R **の関係がある**という.

記法 1.14. a と b の間に R の関係があることを aRb と表す.

1.2.2 順序集合

定義 1.15. 次を満たす集合 X から X への二項関係 R を**順序 order** という.

1. 任意の $x \in X$ について $(x, x) \in R$
2. $(x, y) \in R \wedge (y, x) \in R \Rightarrow x = y$
3. $(x, y) \in R \wedge (y, z) \in R \Rightarrow (x, z) \in R$

定義 1.16. 順序をもつ集合を**順序集合 orderd set** という.

記法 1.17. 順序集合の元 x, y に順序関係があるとき, $x \preceq y$ で表す.

1.2.3 写像

定義 1.18. R が集合 A, B の二項関係であるとする. 任意の $a \in A$ について, とある $b \in B$ が一意に存在して aRb となるとき, R **は** A **から** B **への写像 map であるという.**

記法 1.19. 集合 A から集合 B への写像 f を $f : A \to B$ で表す.

記法 1.20. 集合 A から集合 B への写像に f に関して, $b \in B$ が $a \in A$ に対応することを

$$b = f(a)$$

で表す.

定義 1.21. 集合 A から集合 B への写像 f が, 任意の $a_1, a_2 \in A$ について

$$a_1 \neq a_2 \Rightarrow f(a_1) \neq f(a_2)$$

を満たすとき f **は単射 injection であるという.**

定義 1.22. 集合 A から集合 B への写像 f が, 任意の $b \in B$ に対してとある $a \in A$ が存在して

$$b = f(a)$$

であるとき f **は全射 surjection であるという.**

定義 1.23. 集合 A から集合 B への写像 f が, 単射であり, かつ全射であるとき f **は全単射 bijection であるという.**

定義 1.24. 集合 A から集合 B への写像 $f : A \to B$ について, 集合 B から集合 A への写像 $g : B \to A$ が存在して, A の任意の元 $a \in A$ に対して

$$f(a) = b \Leftrightarrow g(b) = a$$

が成り立つとき, g **は** f **の逆写像 inverse mapping であるという.**

記法 1.25. 写像 f の逆写像を f^{-1} で表す.

1.2.4 合成と結合律

定義 1.26. 集合 A, B, C に関する二項関係 $F \in A \times B$ と $G \in B \times C$ について, その**合成 composition** $G \circ F \in A \times C$ を

$$G \circ F = \{(a, c) \in A \times C | \text{ とある } b \in B \text{ が存在して } (a, b) \in F \text{ かつ } (b, c) \in G \text{ である } \}$$

で定義する.

命題 1.27. 二項関係の合成は**結合律 associative law** を満たす. 結合律とは, 集合 A, B, C, D に対する 3 つの二項関係 $F \in (A \times B), G \in (B \times C), H \in (C \times D)$ について

$$(H \circ G) \circ F = H \circ (G \circ F)$$

が成り立つことをいう.

定義 1.28. 集合 A, B, C に関する写像 $f : A \to B$ と $g : B \to C$ について, その合成 $g \circ f : A \to C$ を

$$(g \circ f)(a) = g(f(a))$$

で定義する. ここで a は A の元である.

命題 1.29. 写像の合成は結合律を満たす. すなわち, 集合 A, B, C, D に対する 3 つの写像 $f : A \to B, g : B \to C, h : C \to D$ について

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

が成り立つ.

第 2 章

代数系

2.1 マグマ・半群・群

2.1.1 マグマ

定義 2.1. 集合 S に関する写像 $\mu : S \times S \to S$ を S **上の二項演算 binary operation** μ という.

定義 2.2. 集合 M と集合 M 上の二項演算 μ の組 (M, μ) を**マグマ magma** という.

2.1.2 半群

定義 2.3. マグマ (G, μ) について, μ が結合律を満たす, すなわち任意の元 $g, h, k \in G$ に対して

$$\mu(g, \mu(h, k)) = \mu(\mu(g, h), k)$$

が成り立つとき, (G, μ) **は半群 semigroup である**という.

2.1.3 群

定義 2.4. マグマ (G, μ) について, とある元 $e \in G$ が存在し, 任意の元 $g \in G$ に対して

$$\mu(e, g) = \mu(g, e) = g$$

が成り立つとき, e を G **の単位元 identity element** という.

定義 2.5. マグマ (G, μ) について, 任意の元 $g \in G$ に対して, とある元 $h \in G$ が存在して

$$\mu(g, h) = \mu(h, g) = e$$

が成り立つとき, h を g **の逆元 inverse element** という. ここで e は G の単位元である.

記法 2.6. マグマ (G, μ) における元 $g \in G$ の逆元を g^{-1} で表す.

定義 2.7. 半群 (G, μ) が単位元をもち, かつ任意の元に対して逆元が存在するとき (G, μ) **は群 group であるという**

定義 2.8. 群 (G_1, μ_1) から群 (G_2, μ_2) への写像 f が任意の G_1 の元 g, g' について

$$f(\mu_1(g, g')) = \mu_2(f(g), f(g'))$$

を満たすとき、f **は** (G_1, μ_1) **から** (G_2, μ_2) **への準同型写像 homomorphism である**という.

定義 2.9. 群 (G_1, μ_1) から群 (G_2, μ_2) への準同型写像 f が全単射であるとき, f **は** (G_1, μ_1) **から** (G_2, μ_2) **への同型写像 isomorphism であるという**

定義 2.10. 群 (G, μ) において, 任意の $a, b \in G$ に対して $\mu(a, b) = \mu(b, a)$ が成り立つとき, **群** (G, μ) **は可換群 commutative group であるという**[*1].

[*1] アーベル群 abelian group ともいう

第 3 章

Haskell の基礎

関数型プログラミング言語 Haskell では圏論的な視点からライブラリが構築されている.

3.1 型

3.1.1 定義済みの型

Haskell には標準ライブラリに

表 3.1 Haskell の標準ライブラリに定義済みの型

`Int`	固定長整数
`Integer`	多倍長整数
`Char`	文字
`Float`	単精度浮動小数点数
`Double`	倍精度浮動小数点数
`Bool`	ブール代数

3.1.2 型構築子

定義済みの型をもとにタプル, リストあるいは `Maybe` のような型構築子によって新たな型を無限に作り出すことができる.

実例 3.1. `[Integer]`, `Maybe Int`, `(Int,[Char])` などはすべて *Haskell* の型である.

3.1.3 データ宣言

例として `Red`, `Green`, `Blue` からなるデータ `Color` の宣言は以下のように記述する.

```
1 data Color = Red | Green | Blue
```

データを表示するためには, `Color` を型クラス `Show` のインスタンスにする必要があることに注意する.

```
1 data Color = Red | Green | Blue deriving Show
```

3.1.4　型クラス

```
class Foo a where
    foo :: a -> String
instance Foo Bool where
    foo True = "Bool: True"
    foo False = "Bool: False"
instance Foo Int where
    foo x = "Int: " ++ show x
instance Foo Char where
    foo x = "Char: " ++ [x]

main = do
    putStrLn $ foo True      -- Bool: True
    putStrLn $ foo (123::Int)    -- Int: 123
    putStrLn $ foo 'A'       -- Char: A
```

Foo 型クラスは任意の型 (a) を受け取り、String を返却するメソッド foo を持っている．instance を用いてそれぞれの型が引数に指定された場合の処理を実装している．

第 II 部

圏論の諸概念

第 4 章

圏

4.1 圏

定義 4.1. **対象**の集合 Obj と**射**の集合 Mor からなり, 以下の演算[*1]が定義されているものを**圏**という.

1. 射 $f \in \mathrm{Mor}$ には**始域**および**終域**となる対象がそれぞれ一意に定まる
2. 射 $f, g \in \mathrm{Mor}$ について, f の終域と g の始域が一致するとき, **合成射** $g \circ f$ が一意に定まる
3. 射の合成は結合律を満たす. すなわち, 射 f, g, h について, ... であるとき

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

が成り立つ.

4. 任意の対象 $A \in \mathrm{Obj}$ について, 次の条件を満たす**恒等射** $1_A : A \to A$ が存在する[*2]
条件：任意の射の組 $f : A \to B, g : B \to A$ に対して $f \circ 1_A = f$ かつ $1_A \circ g = g$

記法 4.2. 圏 $\mathscr{C}$ の対象を $\mathrm{Obj}(\mathscr{C})$ で表す.

記法 4.3. 圏 $\mathscr{C}$ の射を $\mathrm{Mor}(\mathscr{C})$ で表す.

記法 4.4. 射 f の始域を $\mathrm{dom} f$ で表す.

記法 4.5. 射 f の終域を $\mathrm{cod} f$ で表す.

記法 4.6. 始域が A, 終域が B である射 f を $f : A \to B$ で表す.

記法 4.7. 圏 $\mathscr{C}$ の対象 A, B に対して $f : A \to B$ となる射の全体を $\mathrm{Hom}_{\mathscr{C}}(A, B)$ で表す.

定義 4.8. 圏 $\mathscr{C}$ の射 $f : A \to B$ に対して, ある射 $g : B \to A$ が存在して $g \circ f = 1_A$ かつ $f \circ g = 1_B$ となるとき, f **は同型射である**という. また, このとき g を f の**逆射**という.

記法 4.9. 射 f の逆射を f^{-1} で表す.

命題 4.10. ある射の逆射は存在すれば, 一意に定まる

[*1] ?

[*2] 恒等写 1_A は一意に定まるので, 定義に一意性を加えても問題ない.

4.1.1　情報隠蔽された対象の探究に圏論が提供する方法論

オブジェクト指向プログラミングでは,「知らせる必要のない情報は隠蔽しておくほうが安全である」という**情報隠蔽**の考え方が重要視される．これに対して, 圏論は, 対象がもつ情報が隠蔽されている状況下で, 射のみから対象について探究するという方法論を提供する．

実例 4.11. どのような要素をもつかわからない集合 A について写像 $f : A \to A$ が定義されていて $f \circ f \circ f$ が恒等写像になるとする．このとき A が 3 つの要素 a_1, a_2, a_3 をもつと仮定することができ,

$$f(a_1) = a_2, f(a_2) = a_3, f(a_3) = a_1$$

というように, これらの要素が写像 f によって回転していると考えることができる．

4.1.2　小圏・局所小圏・大圏

小圏

定義 4.12. 対象の集合, 射の集合がともに小集合である圏を**小圏**という．

実例 4.13. [*3] 対象も射もない圏 $\mathbf{0}$ は小圏である．

実例 4.14. 対象が 1 つで, 恒等射のみをもつ圏 $\mathbf{1}$ は小圏である．

実例 4.15. 対象が A, B の 2 つで, 恒等射と射 $f : A \to B$ のみをもつ圏 $\mathbf{2}$ は小圏である．

実例 4.16. 順序集合は小圏である．

定義 4.17. すべての対象の組 A, B に対して $\mathrm{Hom}_{\mathscr{C}}(A, B)$ が小集合である圏 $\mathscr{C}$ を**局所小圏**という．

大圏

定義 4.18. 小圏でない圏を**大圏**という．

実例 4.19. すべての小集合[*4]を対象とし, それらの間の写像を射とする圏 Set は大圏である．

実例 4.20. すべての群を対象とし, それらの間の準同型写像を射とする圏 Grp は大圏である．

実例 4.21. すべてのアーベル群を対象とし, それらの間の準同型写像を射とする圏 Ab は大圏である．

実例 4.22. すべての位相空間[*5]を対象とし, それらの間の連続写像[*6]を射とする圏 Top は大圏である．

実例 4.23. ある体[*7] k に対して, すべての k 次線形空間[*8]を対象とし, それらの間の k 次線形写像[*9]を射とする圏 Vect_k は大圏である．

[*3] 実例と命題の区別が曖昧...
[*4] 「宇宙」で定義しているが, 現在本文から外している
[*5] 定義を本文から外している
[*6] 定義を本文から外している
[*7] 定義した箇所をコメントアウトしている
[*8] 未定義？
[*9] 未定義？

4.1.3 部分圏

記法 4.24. 圏 C における射 $f, g \in \mathrm{Mor}(\mathscr{C})$ の合成を $g \circ_{\mathscr{C}} f$ で表す.

定義 4.25. 圏 $\mathscr{A}$ が圏 $\mathscr{B}$ に対して, 以下の 3 条件を満たすとき, $\mathscr{A}$ **は** $\mathscr{B}$ **の部分圏である**という.

1. $\mathrm{Obj}(\mathscr{A})$ が $\mathrm{Obj}(\mathscr{B})$ の部分集合である.
2. 圏 $\mathscr{A}$ の 2 つの対象の組の全体が, 圏 $\mathscr{B}$ の 2 つの対象の組の全体の部分集合である.
3.

実例 4.26. 圏 Ab[*10]は圏 Grp[*11]の部分圏である.

4.1.4 双対

定義 4.27. 任意の圏 $\mathscr{C}$ に対して, 対象が $\mathscr{C}$ と同じで, 射の向きが $\mathcal{C}$ と反対になっている圏を**双対圏という.**

記法 4.28. 圏 $\mathscr{C}$ の双対圏を $\mathscr{C}^{\mathrm{op}}$ と表す.

注意 4.29. 双対の原理

4.1.5 圏の生成

命題 4.30. 集合 A に対して写像 $f : A \to A$ を定める. このとき, 対象の集合 $\mathrm{Obj} = \{A\}$ と, 射の集合 $\mathrm{Mor} = \{1_A, f\}$ からなる圏を得ることができる. ここで, 射 1_A は A についての恒等射である.

命題 4.31. 集合 A に対して写像 $f : A \to A$ を定め, 正の整数 n に対して $f^n = \underbrace{f \circ f \circ \cdots \circ f}_{n}$ とする. このとき, 対象の集合 $\mathrm{Obj} = \{A\}$ と, 射の集合 $\mathrm{Mor} = \{1_A, f, f^2, \ldots, f^n, \ldots\}$ からなる圏を得ることができる. ここで, 射 1_A は A についての恒等射である.

命題 4.32. 集合 A, B に対して, 写像 $f : A \to B$ と $g :\to A$ を定める. このとき, 対象の集合 $\mathrm{Obj} = \{A, B\}$ と射の集合 $\mathrm{Mor} = \{1_A, 1_B, f, g, f \circ g\}$ からなる圏を得ることができる. ここで, 射 $1_A, 1_B$ はそれぞれ A, B についての恒等射である.

定義 4.33. 集合 $S_1, \ldots, S_N$ に関する写像 $f_1, \ldots, f_M$ について, 対象の集合を $\{S_1, \ldots, S_N\}$, 射の集合を恒等射と $f_1, \ldots, f_M$ およびそれらの合成のみからなる集合とする圏が存在するとき, 射の列 $\{f_1, \ldots, f_M\}$ を**生成系 system of generators** という. **生成系**

生成元
関係

定義 4.34. 圏の積を次で定義する

命題 4.35. 圏の積は圏である.

[*10] ref
[*11] ref

4.2 Haskell における圏 (Hask)

命題 4.36. すべての *Haskell* の型を対象とし, それらの間の関数を射とする圏 *Hask* は小圏である.

証明. □

注意 4.37. *Haskell* において, 型 `A`,`B` に対して, 型構築子によってつくられる `A->B` は *1* つの型となる.

第 5 章

関手

5.1 関手

定義 5.1. 圏 $\mathscr{A}$ の対象 $\mathrm{Obj}(\mathscr{A})$ から圏 $\mathscr{B}$ の対象 $\mathrm{Obj}(\mathscr{B})$ への関数を $\mathscr{A}$ **から** $\mathscr{B}$ **への対象関数**という.

定義 5.2. 射関数

定義 5.3. 圏 $\mathscr{A}, \mathscr{B}$ に対する対称関数と射関数の組を $\mathscr{A}$ **から** $\mathscr{B}$ **への関手**という.

実例 5.4. 順序を保存する写像

実例 5.5. n 次ホモロジー関手

実例 5.6.

実例 5.7.

5.1.1 反変関手

定義 5.8. 圏 A^{op} から圏 B への関手を**圏** A **から圏** B **への反変関手**という.

実例 5.9.

記法 5.10.

注意 5.11.

5.1.2 定数関手

定義 5.12. 圏 $\mathscr{A}$ の任意の対象 A を圏 $\mathscr{B}$ のただ一つの対象 B_0 に写し, $\mathscr{A}$ の任意の射 f を圏 $\mathscr{B}$ の恒等射に写す関手を**定数関手**という

5.1.3 忠実関手と充満関手

定義 5.13. 圏 $\mathscr{A}$ から圏 $\mathscr{B}$ への関手 F に関して, 集合 $\{(A_1, A_2) \mid A_1, A_2 \in \mathrm{Obj}(\mathscr{A})\}$ から集合 $\{(F(A_1), F(A_2)) \mid \mathrm{Obj}(\mathscr{A}))\}$ への写像が単射となっているとき, **関手** F **は忠実**であるという.

定義 5.14. 圏 $\mathscr{A}$ から圏 $\mathscr{B}$ への関手 F に関して, 集合 $\{(A_1, A_2) \mid A_1, A_2 \in \mathrm{Obj}(\mathscr{A})\}$ から集合

$\{(F(A_1), F(A_2)) \mid \mathrm{Obj}(\mathscr{A}))\}$ への写像が全射となっているとき, **関手 F は充満**であるという.

定義 5.15. 圏 $\mathscr{A}$ から圏 $\mathscr{B}$ への関手 F が忠実かつ充満であるとき**関手 F は充満忠実である**という

定義 5.16. 圏 $\mathscr{A}$ が圏 $\mathscr{A}$ の部分圏であり, 関手 $F : \mathscr{A} \to \mathscr{B}$ が充満であるとき, **圏 $\mathscr{A}$ は圏 $\mathscr{B}$ の充満部分圏である**という.

実例 5.17. ... 充満忠実である.

実例 5.18. ... 忠実だが充満でない

実例 5.19. 充満だが忠実でない

実例 5.20. 複素数...

...

... 忠実だが充満でない. (例 *1.30*)

5.1.4 埋め込み関手

5.1.5 忘却関手

5.2 Haskell の型構築子と関手

List 関手

Haskell における型構築子 `[]` は任意の型 `A` に対して型 `[A]` を対応させる. これは, Hask から Hask への対称関数とみなせる. 型 `A` と型 `B` および関数 `f::A->B` が与えられとき `map f::[A]->[B]` が決定される.

...

型構築子 `[]` は **List 関手**と呼ばれる

Maybe 関手

Haskell における型構築子 `Maybe` は...

...

型構築子 `Maybe` は **Maybe 関手**と呼ばれる.

Tree 関手

一般に木構造を生成する型構築子は関手にできる. これを **Tree 関手**と呼ぶ.

```
1 module Tree where
2 import Data.Char
3
4 data Tree a = Empty | Node a (Tree a) (Tree a)
5
6 instance (Show a) => Show (Tree a) where
7   show x = show1 0 x
8
9 show1 :: Show a => Int -> (Tree a) -> String
```

```
10 show1 n Empty = ""
11 show1 n (Node x t1 t2) =
12   show1 (n+1) t2 ++
13   indent n ++ show x ++ "\n" ++
14   show1 (n+1) t1
15
16 indent :: Int -> String
17 indent n = replicate (n*4) ' '
18
19 instance Functor Tree where
20   fmap f Empty = Empty
21   fmap f (Node x t1 t2) =
22     Node (f x) (fmap f t1) (fmap f t2)
23
24 -- test data
25 tree2 = Node "two" (Node "three" Empty Empty)
26                    (Node "four" Empty Empty)
27 tree3 = Node "five" (Node "six" Empty Empty)
28                     (Node "seven" Empty Empty)
29 tree1 = Node "one" tree2 tree3
30
31 string2int :: String -> Int
32 string2int "one"   = 1
33 string2int "two"   = 2
34 string2int "three" = 3
35 string2int "four"  = 4
36 string2int "five"  = 5
37 string2int "six"   = 6
38 string2int "seven" = 7
39 string2int _       = 0
40
41 tree0 = fmap string2int tree1
42
43 {- suggested tests
44 fmap (map toUpper) tree1
45 fmap string2int tree1
46 fmap length tree1
47 -}
48
49 {- another possible instance of Show
50 show1 :: Show a => Int -> (Tree a) -> String
51 show1 n Empty = indent n ++ "E"
```

```
show1 n (Node x t1 t2) =
  indent n ++ show x ++ "\n" ++
  show1 (n+1) t1 ++ "\n" ++
  show1 (n+1) t2
-}

{- original instance
instance (Show a) => Show (Tree a) where
  show x = show1 0 x

show1 :: Show a => Int -> (Tree a) -> String
show1 n Empty = ""
show1 n (Node x t1 t2) =
  indent n ++ show x ++ "\n" ++
  show1 (n+1) t1 ++
  show1 (n+1) t2
-}
```

5.3　２変数の関手

Hom 関手

```
{- defined in default startup environment
class Functor f where
  fmap :: (a -> b) -> f a - f b

instance Functor ((->) r) where
  fmap = (.)

instance Functor ((,) a) where
  fmap f (x,y) = (x, f y)
-}

class Contra f where
  pamf ::  (a -> b) -> f b -> f a

newtype Moh b a = Moh {getHom :: a -> b}

instance Contra (Moh b) where
  pamf f (Moh g) = Moh (g . f)

newtype Riap b a = Riap {getPair :: (a,b)}
```

```
21
22 instance Functor (Riap b) where
23   fmap f (Riap (x,y)) =  Riap (f x,y)
24
25 {- suggested tests
26 fmap (\x -> x*x) (\x -> x + 1) 10 == 121
27 getHom (pamf (\x -> x*x) (Moh (\x -> x + 1))) 10 == 101
28
29 fmap (\x -> x*x) (10,10) == (10,100)
30 getPair (fmap (\x -> x*x) (Riap (10,10))) == (100,10)
31 -}
```

5.4 型クラスと Hask の部分圏

ソート関手

```
1  import Tree
2  import Flatten
3
4  iSort :: Ord a => [a] -> [a]
5  iSort xs =
6    foldr ins [] xs
7      where
8        ins x [] = [x]
9        ins x (y:ys)
10         | x <= y     = x:y:ys
11         | otherwise = y: ins x ys
12
13 list2tree :: Ord a => [a] -> Tree a
14 list2tree xs =
15   foldl sni Empty xs
16     where
17       sni = flip ins
18       ins x Empty    = Node x Empty Empty
19       ins x (Node y t1 t2)
20         | x <= y     = Node y (ins x t1) t2
21         | otherwise = Node y t1 (ins x t2)
22
23 iSort2 :: Ord a => [a] -> [a]
24 iSort2 = flatten . list2tree
25
26 {- suggested tests
```

```
map (\x -> x*2) . iSort $ [1,5,3,4,2]
iSort . map (\x -> x*2) $ [1,5,3,4,2]
-}
```

第 6 章

自然変換

6.1 自然変換

定義 6.1. 圏 $\mathscr{A}$ から圏 $\mathscr{B}$ への関手 F, G について,

1. $\mathscr{A}$ の任意の対象 A に対して $\mathscr{B}$ の射 $\alpha_A : FA \to GA$ が存在し
2. $\mathscr{A}$ の任意の射 $f : A_1 \to A_2$ が

$$\alpha_{A_2} \circ Ff = Gf \circ \alpha_{A_1}$$

を満たすとき

に対して**集合** $\alpha = \{\alpha_A \mid A \in \mathrm{Obj}(\mathscr{A})\}$ **は関手** F **から関手** G **への自然変換である**という.

記法 6.2. 関手 F から関手 G への自然変換 α を $\alpha : F \to G$ で表す.

記法 6.3. 関手 F から関手 G への自然変換 α を次の図表で表す.

6.2 Hask における自然変換

6.2.1 concat

6.2.2 List 関手から Maybe 関手への自然変換 safehead

6.2.3 concat と safehead の垂直合成

6.2.4 二分木からリストへの flatten 関数

6.3 Hask における定数関手

6.3.1 length 関数

第 7 章

関手圏

7.1　関手圏

命題 7.1. … は圏である.

定義 7.2. $\mathscr{A}$ **から** $\mathscr{B}$ **への関手圏**という.

記法 7.3. $\mathscr{A}$ から $\mathscr{B}$ への関手圏を $[\mathscr{A}, \mathscr{B}]$ で表す

定義 7.4. 関手圏における同型射を自然同型という. **自然同型**

命題 7.5. 自然変換… が自然同型であること,
… が同型者であることは同値である.

第 8 章

圏同値

8.1 圏同値

定義 8.1. 圏同値

8.2 Hask における自然同型

8.2.1 mirror 関数

Tree 関手から Tree 関手への自然同型

8.2.2 Maybe 関手と Either() 関手の間の自然同型

8.3 まとめ

aa