圏論原論

Hirokichi Tanaka

2022 年 12 月 30 日

目次

第 1 章

準備

1.1 集合とユニバース

定義 1.1. 順序集合

定義 1.2. べき集合

定義 1.3. ユニバース

定義 1.4. 小さい集合

命題 1.1. ユニバース U は小さい集合ではない

内包原理

定義 1.5. クラス

命題 1.2. ユニバース U はクラスである.

定義 1.6. 真のクラス

命題 1.3. ユニバース U は真のクラスである.

定義 1.7. 部分集合

1.2 二項関係・部分関数・写像（関数）

定義 1.8. 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**であるという.

定義 1.9. R が集合 A, B の二項関係であるとする.
各集合の要素 $a \in A$ と $b \in B$ について $(a, b) \in R$ であるとき, a **と** b **に間に** R **の関係がある**という.

記法 1.1. a と b の間に R の関係があることを aRb と表す.

定義 1.10. 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 **への写像である**という.

注意 1.1. 以下では, 「**関数**」と「写像」を同じ意味で用いる.

定義 1.11. 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 $b, b' \in B$ が存在し，

$$aRb \wedge aRb' \Rightarrow b = b'$$

が成り立つとき，R **は** A **から** B **への部分関数**という．

命題 1.4. 関数は部分関数である．これは定義より明らかである．

命題 1.5. 部分関数は二項関係である．これは定義より明らかである．

1.2.1　合成と結合律

定義 1.12. 集合 A, B, C に対する *2* つの写像 $f : A \to B$ と $g : B \Rightarrow C$ について，その**合成写像** $g \circ f : A \to C$ を

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

で定義する．ここで a は A の要素である．

定義 1.13. 部分関数の合成

定義 1.14. 二項関係の合成を次で定義する
…

命題 1.6. 二項関係の合成は結合律を満たす

注意 1.2. 一般に，結合律とは...

命題 1.7. 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．

命題 1.8. 部分関数の合成は結合律を満たす．

1.3　群・準同型写像・同型写像

定義 1.15. 群

定義 1.16. 準同型写像

定義 1.17. 同型写像

定義 1.18. **アーベル群**

定義 1.19. n 次ホモロジー群

第 2 章

圏・関手

2.1 圏

定義 2.1. 圏 $\mathscr{C}$ は**対象**の集合 $\mathrm{Obj}(\mathscr{C})$ と**射**の集合 $\mathrm{Mor}(\mathscr{C})$ からなる，以下の演算が定義されているもののことをいう．

1. 射 $f \in \mathrm{Mor}(\mathscr{C})$ には**始域** $\mathrm{dom}f$ および**終域** $\mathrm{cod}f$ となる対象がそれぞれ一意に定まる[*1]
2. $\mathrm{dom}g = \mathrm{cod}f$ を満たす射 $f, g \in \mathrm{Mor}(\mathscr{C})$ に対して，**合成射** $g \circ f : \mathrm{dom}f \to \mathrm{cod}g$ が一意に定まる
3. 射の合成は結合律を満たす．すなわち，対象 $A, B, C, D \in \mathrm{Obj}$ についての射の列 f, g, h が与えられたとき
$$h \circ (g \circ f) = (h \circ g) \circ f$$
が成り立つ．
4. 任意の対象 $A \in \mathrm{Obj}(\mathscr{C})$ 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 $1_A \circ g = g$

定義 2.2. 同型射と逆射

記法 2.1. 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．

命題 2.1. ある射の逆射は存在すれば，一意に定まる

2.1.1 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．

例 2.1. 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．

[*1] $\mathrm{dom}f = A \in \mathrm{Obj}(\mathscr{C}), \mathrm{cod}f = B \in \mathrm{Obj}(\mathscr{C})$ のとき $f : A \to B$ とかく．
[*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．

2.1.2　小圏・局所小圏・大圏

定義 2.3. 対象の集合, 射の集合がともに小さい集合である圏を**小圏**という.

例 2.2. 順序集合は小圏である.

定義 2.4. すべての対象の組 A, B に対して $\mathrm{Hom}_{\mathscr{C}}(A, B)$ が小さい集合である圏 $\mathscr{C}$ を**局所小圏**という.

定義 2.5. 小圏でない圏を**大圏**という.

例 2.3. すべての小さな集合を対象とし, それらの間の写像を射とする圏 Set は大圏である.

例 2.4. 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.

例 2.5. 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.

例 2.6. すべての位相空間を対象とし, それらの間の連続写像を射とする圏 Top は大圏である.

例 2.7. ある体 k に対して, すべての k 次線形空間を対象とし, それらの間の k 次線形写像を射とする圏 Vect_k は大圏である.

2.1.3　部分圏

定義 2.6. 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.
3.

例 2.8. 圏 Ab は圏 Grp の部分圏である.

2.1.4　双対

定義 2.7. 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 2.2. 圏 $\mathscr{C}$ の双対圏を $\mathscr{C}^{\mathrm{op}}$ と表す.

注意 2.1. 双対の原理

2.1.5　圏の生成

定義 2.8. 生成元

定義 2.9. 生成系

定義 2.10. 圏の積

2.2　関手・反変関手

定義 2.11. 圏 $\mathscr{A}$ から $\mathscr{B}$ への**関手**は**対象関数** $F_0 : \mathrm{Obj}(\mathscr{A}) \to \mathrm{Obj}(\mathscr{B})$ と**射関数** F_1 からなるもののことをいう.

定義 2.12. **反変関手**

2.3　忠実関手と充満関手

定義 2.13. **忠実**

定義 2.14. **充満**

定義 2.15. **充満忠実**

定義 2.16. **充満部分圏**

2.4　Haskell における圏 (Hask)

Haskell には標準ライブラリに `Int`, `Integer`, `Char`, `Float`, `Double`, `Bool` などの型が定義済みである. これらをもとにタプル, 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.

例 2.9. `[Integer]`, `Maybe Int`, `(Int,[Char])` などはすべて *Haskell* の型である.

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.

注意 2.2. *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.

2.4.1　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.

```haskell
module Tree where
import Data.Char

data Tree a = Empty | Node a (Tree a) (Tree a)

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) =
  show1 (n+1) t2 ++
  indent n ++ show x ++ "\n" ++
  show1 (n+1) t1

indent :: Int -> String
indent n = replicate (n*4) ' '

instance Functor Tree where
  fmap f Empty = Empty
  fmap f (Node x t1 t2) =
    Node (f x) (fmap f t1) (fmap f t2)

-- test data
tree2 = Node "two" (Node "three" Empty Empty)
                   (Node "four" Empty Empty)
tree3 = Node "five" (Node "six" Empty Empty)
                    (Node "seven" Empty Empty)
tree1 = Node "one" tree2 tree3

string2int :: String -> Int
string2int "one"   = 1
string2int "two"   = 2
string2int "three" = 3
string2int "four"  = 4
string2int "five"  = 5
string2int "six"   = 6
string2int "seven" = 7
string2int _       = 0

tree0 = fmap string2int tree1

{- suggested tests
fmap (map toUpper) tree1
fmap string2int tree1
fmap length tree1
-}

{- another possible instance of Show
show1 :: Show a => Int -> (Tree a) -> String
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
-}
```

2.4.2 2変数の関手

2.4.3 Hask の部分圏

第 3 章

自然変換・定数関手

3.1 自然変換

定義 3.1. 自然変換

3.2 定数関手

定義 3.2. 定数関手

定義 3.3.

3.3 Hask における自然変換・定数関手

3.3.1 Hask における自然変換

concat
safehead
concat と safehead の垂直合成
flatten 関数

3.3.2 Hask における定数関手

length 関数

3.3.3 Hask の部分圏

第 4 章

関手圏・圏同値

4.1 関手圏

定義 4.1. 自然同型

4.2 圏同値

定義 4.2. 圏同値

4.3 Hask における...

4.3.1 Hask における自然同型

mirror 関数

Maybe 関手と Either() 関手の間の自然同型

第 5 章

普遍性と極限

第 6 章

随伴

第 7 章

モナドと Haskell の Monad

第 8 章

表現可能関手

定義 8.1. 局所小圏 $\mathscr{A}$ の対象 A を固定して, 対象 A' ごとに射の集合 $\mathrm{Hom}_{\mathscr{A}}(A, A')$ を考える... 以下のように, 対象の対応を射の対応に拡張して得られる関手を **Hom 関手**という. ...

定義 8.2. $\mathscr{A}$ を局所小圏とする. 関手 $X : \mathscr{A} \to \mathrm{Set}$ が, とある $A \in \mathrm{Obj}(\mathscr{A})$ によって... となるとき, X **は表現可能である**という.