ホモトピー型理論

上村 太一

2023-10-01

[*1] https://creativecommons.org/licenses/by/4.0/

貢献者 [007H]

執筆、誤字脱字の報告などで本書の内容に直接影響を与えた者たちの一覧。

- Taichi Uemura (https://github.com/uemurax, https://twitter.com/t_uemura669101)
- 立腹層 (https://twitter.com/rippukusou)

目次

1 はじめに [0000]

ホモトピー型理論 (homotopy type theory, HoTT) [HoTT-Book] は**ホモトピー論** (homotopy theory) と**型理論** (type theory) が融合した分野である。ホモトピー論は古典的には空間のホモトピー型 (良い位相空間をホモトピー同値で同一視したもの) を調べる分野で、現代ではより一般的、抽象的になんらかの「ホモトピー」と呼ばれる緩い同一視の概念がある状況に使われる。一方、型理論は形式体系の一種で、数学の基礎言語、の内部言語、プログラミング言語、定理証明支援系などに使われる。

型理論をホモトピー論的な文脈で解釈することでこの二つの分野は密接に繋がれる。典型的には、空間のホモトピー型の集まりが型理論のモデルになる。これによって、型理論の中でのあらゆる構成は空間のホモトピー型に関する構成に翻訳できる。逆に、空間のホモトピー型のなすモデルにおいて妥当な構成を型理論に取り込むこともできる。ホモトピー型理論と言った場合、型理論とホモトピー論を関連付けて双方を調べる研究分野のことを指す場合と、ホモトピー論的に妥当な構成を取り入れた型理論のことを指す場合がある。本書では、後者のホモトピー論的に妥当な構成を取り入れた型理論とはどのようなものなのかを解説する。

ホモトピー論が型理論にもたらした重要な概念の二つは**一価性公理** (univalence axiom) と**高次帰納的型** (higher inductive type) である。一価性公理は、二つの型の間に同値 (適当な意味で逆関数を持つ関数) がある時にそれらの型は同一視されることを保証する。従来の数学でも、二つの集合の間に全単射がある時にそれらの集合を同一視するのが自然であるが、この同一視は非形式的である。例えば公理的集合論において二つの集合が同一視されるのはそれらの要素が一致する時であり、全単射があるだけで本当に同一視してしまうと当然矛盾する。一方、一価性公理は形式体系における公理であり、同値がある二つの型は本当に同一視される。この同一視は型理論をホモトピー論的に解釈して初めて正当化される。

高次帰納的型は帰納的型の一般化である。通常の帰納的型はいくつかの要素の構成で自由に生成された型で、自然数のなす型、リスト型、余積などがその例である。高次帰納的型は要素の同一視の構成も含めることができる。商型、押し出しなどの余極限が典型例である。ホモトピー論的解釈を念頭に置くと、ここでいう余極限はホモトピー余極限と呼ばれるもので、円周、球面、トーラスなどの高次のホモトピー論的な情報を豊富に持つ空間を表す型を含む。

[0077] 1.1 型理論

型理論は形式体系の一種で、様々な用途がある。歴史的には**数学の基礎付け** *(foundations of mathematics)* が元々の動機である。Russell[Russell--1908-0000] が型理論を導入した当時はパラドックスの発見により数学の基礎付けが揺らいでいた時期である。Russell[Russell--1908-0000] の型理論は後に Church[Church--1940-0000] によって単純型付きラムダ計算として整備され、元々の動機であった数学の基礎付けとは別に**プログラミング言語** *(programming language)* の理論においても活発に研究されている[Pierce--2002-0000]。型理論はさらに別の方向で圏論との関わりもあり、例えば単純型付きラムダ計算はデカルト閉圏の**内部言語** *(internal language)* である[Lambek--Scott--1986-0000]。ホモトピー型理論の直接の基盤となる型理論は Martin-Löf 型理論[Martin-Loef--1975-0000] である。これは Martin-Löf[Martin-Loef--1975-0000] が構成的数学の基礎付けとして導入したもので、**定理証明支援系** *(interactive theorem prover, proof assistant)* の基礎でもある。

型理論が圏の内部言語であるという意味を説明する。圏とは、集合と写像のような、なんらかの対象とその間の射のなる構造を抽象化した概念である。単純型付きラムダ計算がデカルト閉圏の内部言語であるとは、デカルト閉圏と呼ばれる特別な構造を持つ圏は型を対象、要素を射と解釈することで単純型付きラムダ計算の**モデル**になるという意味である(実際にはデカルト閉圏の構造が単純型付きラムダ計算を解釈するために必要最低限のものであるというようなことも言える)。例えば集合と写像のなす圏はデカルト閉圏であり、型を集合、要素を写像と解釈することで単純型付きラムダ計算のモデルになる。他にも、群 G に対して G の作用を持つ集合のなす圏、空間 X に対して X 上の層のなす圏などはデカルト閉圏である。したがって、単純型付きラムダ計算の中での構成は自動的にこれらの圏の中での構成に翻訳される。

定理証明支援系とは、コンピュータ上で定理を証明または証明の正しさを検証するソフトウェアで、Coq[*2]や Agda[*3]や Isabelle[*4]などがある。このうちは Coq と Agda は Martin-Löf 型理論 (の派生) を基盤としている。

型理論はしばしば**構成的** *(constructive)* であることが強調される。構成的でない議論の例としては、排中律や二重否定の除去がある。排中律は、すべての命題は真または偽のいずれかであるという公理であるが、構成的な立場からは真か偽のどちらなのかを決定できない限りは真または偽のいずれかであるとは言えない。二重否定の除去は例えばある性質を満たす x が存在しないと仮定して矛盾を導き、よってその性質を満たす x が存在するというような議論である。実際に証拠となる x を構成しなくても x の存在を証明できてしまう場合があるので構成的ではない。型理論では、このような非構成的な公理は一般には仮定しない。

非構成的な公理を仮定しないということは当然証明力は落ちる。構成主義者でもない限りあまり構成的であることにこだわる意味はないように思うかもしれない。しかし、型理論は圏の内部言語という観点から見ると、証明力を落とすことはモデルが増えるという意

[*2] https://coq.inria.fr/
[*3] https://wiki.portal.chalmers.se/agda/pmwiki.php
[*4] https://isabelle.in.tum.de/

味で利点でもある。実際、排中律を適切に解釈できる圏というのは特殊なもので、排中律を仮定した型理論は圏の内部言語としてはあまり使い勝手が良いものではない。このことから、構成主義者でなくとも構成的な型理論を使う価値は十分にある。

型理論が構成的であることはプログラミング言語の観点からは重要である。構成がはっきりしない部分があると計算が進まないのでプログラムとは言えない。逆に、構成的な証明からはプログラム抽出 (program extraction) ができる。例えば任意の x に対してある性質を満たす y が存在するという命題を構成的に証明できたとしよう。証明が構成的であるということは、x から具体的に y が構成されたということであるが、これは x から y を計算するプログラムであると言える。このように証明から抽出されたプログラムは定義により満たすべき性質を満たしていることが保証されている。

1.2 統一された正しい同一視の概念

[0076]

ホモトピー型理論を数学の言語 (数学の基礎付けまたは圏の内部言語) と見た場合、長所として「統一された正しい同一視の概念」が挙げられる。比較のために、事実上標準的な数学の基礎付けである公理的集合論では同一視の概念が「正しい」とは言い難いことを説明する。公理的集合論では、あらゆる数学的対象は集合として実装される。二つの集合が同一であるのはそれらが全く同じ要素を持つ場合である。この同一視の概念は統一されてはいるが、「正しい」同一視ではない。数学者が実際に二つの集合を同一視するのはそれらの間に全単射がある時である。また、なんらかの構造 (例えば群構造) を持った二つの集合を同一視するのはそれらの間に同型写像 (例えば群同型) がある時であるし、場合によっては同型よりも弱い概念 (例えば圏同値やホモトピー同値) によって二つの対象を同一視する。つまり、言語が提供する同一視の概念と数学者が本当に求めている同一視の概念が乖離している。

集合としての同一性よりも弱い同一視の概念を使うということは、各種概念が弱い同一視の下で**不変**であるかどうかを常に気にしなければならない。例えば、二つの集合の間に全単射があればそれらの濃度は同じなので、濃度は全単射の下で不変であり、濃度について論じる時は集合を全単射の下で同一視しても問題はない。一方、集合 X についての性質「X は空集合を要素に持つ」は全単射の下で不変ではない。このように、公理的集合論の言語を使う限り、全単射の下で不変でないような性質は簡単に書けてしまい、そのような性質に言及した場合には全単射による同一視は (一般には) 正当化されない。「空集合を要素に持つ」はやや人工的な例だが、もう少し自然な例だと「圏が有限個の対象と射を持つ」といった性質は圏同値で不変でないにもかかわらず圏論の教科書に登場することがある。

対して、ホモトピー型理論には統一された同一視の概念があり、あらゆる性質はその同一視の下で不変であり、かつその同一視の概念は「正しい」ものであることを証明できる。一価性公理は二つの型の間に同値があればそれらは同一視されるというものであったが、実はこの公理があればあらゆる構造について同一視の概念が「正しい」ものであることが導出される。この現象は**構造同一原理** *(structure identity principle)* と呼ばれる。例えば、二つの群の間に群同型があればそれらは同一視され、二つの圏の間に圏同値があればそれらは同一視される、などが**定理**として得られる。よって、あらゆる性質は群同型や圏同値の下で不変である。群同型や圏同型の下で不変でないようなものはそもそも書けな

いように言語が設計されていると言ってもよい。

[0078]

1.3 一価性公理とホモトピー論

一価性公理は雑に言えば同値の概念と同一視の概念が一致するという公理だが、そんなことが本当に起こり得るだろうか。少なくとも、同一視の概念を「等しさ」と解釈する限りは不可能である。例えば、等しくない集合の間にも全単射はあり得るので、集合の間の同値と集合の等しさは完全に異なる概念である。一価性公理は同一視の概念をホモトピー論的に解釈すれば正当化されることを説明する。

ホモトピー論ではその名の通り**ホモトピー** *(homotopy)* の概念が重要である。ホモトピーの概念は様々な場所で現れ、それに応じて様々な「ホモトピー論」があるが、まずは古典的な位相空間のホモトピー論を思い出そう。位相空間の間の連続写像 $f : U \to A$ と $g : U \to A$ に対して、f から g へのホモトピーとは f を g に「連続的に変形」させるもので、連続写像の間の緩い同一視の概念である。形式的には、連続写像 $H : [0,1] \times U \to A$ であって $H(0,x) = f(x)$ かつ $H(1,x) = g(x)$ となるものである。位相空間のホモトピー論ではホモトピーの概念により連続写像を同一視して位相空間を調べる。

ホモトピーで連続写像を同一視するといっても、ホモトピーはたくさんあり得るので、どのように同一視するかというのも重要な情報になる。ホモトピー自身も連続写像であることから、ホモトピーの間のホモトピーやホモトピーの間のホモトピーの間のホモトピーといった高次のホモトピー概念も定義される。さらに、f から g へのホモトピー H と g から h へのホモトピー K は合成することができ、f から h へのホモトピー $K \circ H$ を得る。合成演算はすべての次元のホモトピーに対して定義され、互いに複雑な関係を持つ。このような構造は**∞グルーポイド** *(∞-groupoid)* と呼ばれる。一点からの連続写像とその間の高次ホモトピーを考えると、任意の位相空間から∞グルーポイドを得る。∞グルーポイドの写像の間にもホモトピーの概念が定義され、この構成は位相空間のホモトピー論から∞グルーポイドのホモトピー論への「ホモトピー論の射」になる。

位相空間から∞グルーポイドの構成を見ると、位相空間や連続関数の具体的な定義はあまり関係なく、対象の射の間の高次のホモトピーの概念さえあればよい。したがって、どのようなホモトピー論に対しても、対象の間の射の集まりは∞グルーポイドをなす。このことから、∞グルーポイドのホモトピー論は数あるホモトピー論の中でも特別な役割を担う。そして、一価性公理を満たすホモトピー論の典型例はまさに∞グルーポイドのホモトピー論である。

∞グルーポイドは点の集まりと、点の間のホモトピーの集まりと、点の間のホモトピーの間のホモトピーの集まりなどの無限個の構成要素からなる構造である。点の間のホモトピーとは点の同一視のしかたと思ってもよい。∞グルーポイドの**同値** *(equivalence)* とは、∞グルーポイドの写像 $f : A \to B$ であって、逆写像 $g : B \to A$ と合成 $g \circ f$ から A 上の恒等写像へのホモトピーと合成 $f \circ g$ から B 上の恒等写像へのホモトピーがあるものである。ホモトピー同値は∞グルーポイドの「正しい」同一視のしかたと考えられる。よって、∞グルーポイドを点、∞グルーポイドの同値を点の間のホモトピーとするような∞グルーポイドが考えられる。この∞グルーポイドの∞グルーポイドがあることで、∞グルーポイドのホモトピー論は一価性公理を満たす。定義により∞グルーポイドの間の同一視のしかたと∞グルーポイドの同値が一致するので当然である。

この議論は際限なく高次のホモトピーの概念があることによって成り立つ。比較のために集合のホモトピー論を考える。集合の二つの要素が等しい時にそれらの間にただ一つホモトピーがあるとする。このホモトピーの概念から得られる集合の同値の概念は全単射である。すると、集合を点とする構造を考えると点の間のホモトピーは全単射となる。等しくない集合の間にも全単射はあり得るので、この集合を点とする構造は集合の枠組みには収まらずグルーポイドと呼ばれる一つ上の次元のホモトピーの概念を持つ構造になる。これを修正するためにグルーポイドのホモトピー論を考えると、今度はグルーポイドを点とする構造が 2 グルーポイドというさらに一つ上の次元のホモトピーの概念を持つ構造になる。以下同様に、n グルーポイドを点とする構造は $n+1$ グルーポイドというさらに一つ上の次元のホモトピーの概念を持つ構造になる。∞グルーポイドはこの議論の「極限」に位置する概念で、∞グルーポイドを点とする構造はまた∞グルーポイドになるというわけである。

ちなみに、Grothendieck の**ホモトピー仮説** *(homotopy hypothesis)* によると位相空間のホモトピー論と∞グルーポイドのホモトピー論は同値である。∞グルーポイドをどう厳密に定義するかというのは自明ではなく、ホモトピー仮説は∞グルーポイドの定義によって定理であったり予想であったりする。例えば Kan 複体 (Kan complex) は∞グルーポイドの定義の一つであり、位相空間のホモトピー論と Kan 複体のホモトピー論は同値である (モデル圏 [Quillen--1967-0000] の言葉で定式化される。証明は例えば [Hovey--1999-0000] を参照)。さらに、一価性公理は Kan 複体を∞グルーポイドの定義として示された [Kapulkin--Lumsdaine--2021-0000]。

1.4 識別子 [000R]

本書ではすべての章や定理などに一意で恒久的な**識別子**が割り当てられている。章番号や定理番号は本書の更新に伴って変わることがあるのに対し、識別子は変わることはない。そのため、本書の特定の箇所を参照する場合は識別子を使うことを強く推奨する。HTML 版では識別子ごとのページも作成されており、`https://uemurax.github.io/hott-ja/<識別子>.html` からアクセスできる。PDF 版では識別子ごとのページは作成していない。

ちなみに、識別子を使うというアイディアは Stacks project[*5]や Kerodon[*6]を参考にしている。

[*5] `https://stacks.math.columbia.edu/`
[*6] `https://kerodon.net/`

2 型理論 [0001]

ホモトピー型理論は**型理論**の一種であり、型理論は一般になんらかの対象を**構成**するための規則群と構成されたものに関する**定義**を表す規則群によって定められる。2 章では、ホモトピー型理論の基礎となる構成規則を導入する。

型理論一般における概念や表記をいくつか導入する。以下の用語は形式的に定義することもできるが、当面は普通の自然言語の意味で解釈して問題ない。

2*1 記法: 同じ種類の対象 α と β が**定義により等しい**ことを $\alpha \equiv \beta$ と書く。 [0086]

定義により等しいとは、定義を表す規則を使って等しいと導けることを意味する。定義により等しい対象は全く同じものとして扱ってよい。

構成にはいくつかの**仮定**が置かれることもある。仮定 x の下での対象とは、どんな対象 x に対しても妥当な構成で得られる対象である。

2*2 記法: α を仮定 $x_1, \ldots, x_n$ の下での対象、a_i を x_i と同じ種類の対象とする。α の構成において各 x_i に a_i を**代入**したものはまた妥当な構成である。代入の結果得られる対象を $\alpha[x_1 \mapsto a_1, \ldots, x_n \mapsto a_n]$ と書く。 [0087]

どのような種類の対象を構成できるかは型理論によって異なるが、通常は**型**とその**要素**が最も興味のある対象である。各要素には型が割り当てられている。

2*3 記法: 要素 a の型が A であることを明示するときは $a : A$ と書く。 [0088]

2.1 宇宙 [0009]

型理論における対象は型と要素の二種類に分けられるが、型と要素を厳格に区別し続けることは時に理論を煩雑にする。そこで**宇宙**を導入する。宇宙とは要素が型であるような型である。すべての型がある宇宙の要素になるように規則を設計することで、型を特別な要素とみなすことができる。本書で考える型理論は「可算個」の非有界な宇宙の列

$$\mathcal{U}(0) : \mathcal{U}(1) : \mathcal{U}(2) : \ldots$$

を持つ。

数学を基礎付ける文脈では $0, 1, 2, \ldots$ が何なのか分からないので、**階数**の概念を導入する。階数は型でも要素でもない別の種類の対象である。

2.1*1 規則: **階数** *(level)* についての規則は次で与えられる。 [000D]

- 階数 0 を構成できる。
- 階数 i に対し、階数 $\mathsf{succ}(i)$ を構成できる。

宇宙に関する規則は次の通りである。

[000E] **2.1*2 規則**: **宇宙** *(universe)* についての規則は次で与えられる。

- 階数 i に対して、型 $\mathcal{U}(i)$ を構成できる。
- 階数 i と要素 $A : \mathcal{U}(i)$ に対して、型 $\mathsf{El}(A)$ を構成できる。
- 階数 i に対して、要素 $\lceil\mathcal{U}\rceil(i) : \mathcal{U}(\mathsf{succ}(i))$ を構成できる。
- 階数 i に対して、$\mathsf{El}(\lceil\mathcal{U}\rceil(i)) \equiv \mathcal{U}(i)$ と定義される。
- 階数 i と要素 $A : \mathcal{U}(i)$ に対して、$\mathsf{Lift}(A) : \mathcal{U}(\mathsf{succ}(i))$ を構成できる。
- 階数 i と要素 $A : \mathcal{U}(i)$ に対して、$\mathsf{El}(\mathsf{Lift}(A)) \equiv \mathsf{El}(A)$ と定義される。

[000F] **2.1*3 記法**:

- El はよく省略する。つまり、要素 $A : \mathcal{U}(i)$ そのものを型とみなす。
- Lift はよく省略する。つまり、要素 $A : \mathcal{U}(i)$ は $\mathcal{U}(\mathsf{succ}(i))$ の要素でもあるとみなす。
- $\mathcal{U}(i)$ と $\lceil\mathcal{U}\rceil(i)$ を表記上区別しない。つまり、$\mathcal{U}(i)$ そのものを $\mathcal{U}(\mathsf{succ}(i))$ の要素とみなす。

記法 2.1*3 の下で、規則 2.1*2 は次のようにも書ける。

- 階数 i に対して、$\mathcal{U}(i) : \mathcal{U}(\mathsf{succ}(i))$ を構成できる。
- 階数 i と要素 $A : \mathcal{U}(i)$ に対して、A は型である。
- 階数 i と要素 $A : \mathcal{U}(i)$ に対して、$A : \mathcal{U}(\mathsf{succ}(i))$ である。

本書で考える型理論では規則 2.1*2 の他には形式的な意味での型を構成する規則は与えず、代わりに $\mathcal{U}(i)$ の要素を構成する規則を与える。以降、$\mathcal{U}(i)$ の要素を (階数 i の) **型**と呼ぶ。

[000A] 2.2 関数型

関数は型理論において最も基本的な概念である。関数の導入によって、仮定の下での対象をも型理論の対象にすることができ、これは型理論の表現力を飛躍的に向上させる。

[000H] **2.2*1 規則**: i を階数、$A : \mathcal{U}(i)$ を型、$B : \mathcal{U}(i)$ を仮定 $x : A$ の下での型とする。

- **関数型** *(function type)* $\prod_{x:A} B : \mathcal{U}(i)$ を構成できる。$\prod_{x:A} B$ の要素を**関数** *(function)* と呼ぶ。
- 仮定 $x : A$ の下での要素 $b : B$ に対し、**ラムダ抽象** *(lambda abstraction)* $\lambda x.b : \prod_{x:A} B$ を構成できる。
- 要素 $f : \prod_{x:A} B$ と $a : A$ に対し、**関数適用** *(function application)* $f(a) : B[x \mapsto a]$ を構成できる。
- $b : B$ を仮定 $x : A$ の下での要素、$a : A$ を要素とする。このとき、$(\lambda x.b)(a) \equiv b[x \mapsto a]$ と定義される。

- 要素 $f : \prod_{x:A} B$ に対し、$f \equiv \lambda x.f(x)$ と定義される。

関数型の導入により、仮定 $x : A$ の下での B の要素と $\prod_{x:A} B$ 型の関数は同じように振る舞う。以降は仮定の下での要素の代わりに関数を使う。

B が $x : A$ に依存しない場合は次のように略記する。

2.2*2 定義: i を階数、$A, B : \mathcal{U}(i)$ を型とする。このとき、$A \to B : \mathcal{U}(i)$ を $\prod_{x:A} B$ と定義する。 [000G]

2.2*3 記法: [000J]

- $\to$ は右結合の演算子である。例えば、$A \to B \to C$ は $A \to (B \to C)$ と読む。
- $\lambda x_1 \ldots . \lambda x_n.b$ は $\lambda(x_1, \ldots, x_n).b$ と略記することがある。
- $f(a_1) \ldots (a_n)$ は $f(a_1, \ldots, a_n)$ と略記することがある。
- $\prod_{x:A}$ の結合は弱い。例えば、$\prod_{x:A} \prod_{y:B} C \to D$ は $\prod_{x:A}(\prod_{y:B}(C \to D))$ と読む。

引数が多い関数を適用する時、すべての引数を明示するのはいささか煩雑である。そこで、引数が他の引数の型から推論できる場合に省略できるようにする。

2.2*4 記法: $f : \prod_{\{x:A\}} \prod_{y:B} C$ のように引数を { } で囲った場合、その引数は**暗黙的** *(implicit)* であると約束する。つまり、要素 $a : A$ と $b : B[x \mapsto a]$ に対して、関数適用を $f(a, b)$ の代わりに a を省略して $f(b)$ と書く。B の構成で x を明示的に使う場合には実用上は曖昧性は無い。暗黙的な引数 a だけを適用したい場合は $f\{a\} : \prod_{y:B[x\mapsto a]} C[x \mapsto a]$ と書く。 [000Q]

$A \to \mathcal{U}(i)$ 型の関数は A で添字付けられた**型の族**と考えられる。

2.2*5 用語: i を階数、$A : \mathcal{U}(i)$ を型とする。関数 $B : A \to \mathcal{U}(i)$ を A 上の**型の族** *(type family)* と呼ぶ。 [000I]

いくつかの簡単な関数の例を挙げる。

2.2*6 記法: 関数を定義するには次のいずれかような言い回しをする。 [0089]

- 関数 $f : \prod_{x:A} B(x)$ を $\lambda x.b$ と定義する。
- 関数 $f : \prod_{x:A} B(x)$ を $f(x) \equiv b$ と定義する。
- $x : A$ を要素とする。要素 $f(x) : B(x)$ を b と定義する。

最後の記法において、f の引数のリストに現れない仮定は f の暗黙的引数 (記法 2.2*3) と解釈する。

2.2*7 例: i を階数、$A : \mathcal{U}(i)$ を型とする。**恒等関数** *(identity function)*$\mathsf{id} : A \to A$ を $\mathsf{id}(x) \equiv x$ と定義する。 [0010]

2.2*8 例: i を階数、$A, B, C : \mathcal{U}(i)$ を型、$f : A \to B$ と $g : B \to C$ を関数とする。**合成関数** *(composed function)*$g \circ f : A \to C$ を $(g \circ f)(x) \equiv g(f(x))$ と定義する。 [0011]

2.2*9 演習: i を階数、$A, B, C, D : \mathcal{U}(i)$ を型、$f : A \to B$ と $g : B \to C$ と $h : C \to D$ [0012]

を関数とする。

1 $f \equiv f \circ \mathsf{id}$ であることを確かめよ。
2 $\mathsf{id} \circ f \equiv f$ であることを確かめよ。
3 $(h \circ g) \circ f \equiv h \circ (g \circ f)$ であることを確かめよ。

[0013] **2.2*10 例**: i を階数、$A, B : \mathcal{U}(i)$ を型、$C : A \to B \to \mathcal{U}(i)$ を型の族、$f : \prod_{x:A} \prod_{y:B} C(x,y)$ を関数とする。関数 $\mathsf{swap}(f) : \prod_{y:B} \prod_{x:A} C(x,y)$ を $\mathsf{swap}(f,y,x) \equiv f(x,y)$ と定義する。

[000B] 2.3 レコード型

レコード型は構造を記述するのに便利な型である。本書では、組み込み型としては単位型と対型を導入し、レコード型は記法として実現する。これは体系を単純なものに抑えるためである。

単位型は要素を丁度一つ持つ型である。

[000K] **2.3*1 規則**:

- **単位型** *(unit type)* $\mathbf{1} : \mathcal{U}(0)$ を構成できる。
- 要素 $\star : \mathbf{1}$ を構成できる。
- 要素 $a : \mathbf{1}$ に対し、$a \equiv \star$ と定義される。

対型は要素の対を要素に持つ型である。

[000L] **2.3*2 規則**: i を階数、$A : \mathcal{U}(i)$ を型、$B : A \to \mathcal{U}(i)$ を型の族とする。

- **対型** *(pair type)* $\sum_{x:A} B(x) : \mathcal{U}(i)$ を構成できる。
- 要素 $a : A$ と $b : B(a)$ に対し、**対** *(pair)* $\mathsf{pair}(a,b) : \sum_{x:A} B(x)$ を構成できる。
- 要素 $c : \sum_{x:A} B(x)$ に対し、**射影** *(projection)* $\mathsf{proj}_1(c) : A$ と $\mathsf{proj}_2(c) : B(\mathsf{proj}_1(c))$ を構成できる。
- 要素 $a : A$ と $b : B(a)$ に対し、$\mathsf{proj}_1(\mathsf{pair}(a,b)) \equiv a$, $\mathsf{proj}_2(\mathsf{pair}(a,b)) \equiv b$ と定義される。
- 要素 $c : \sum_{x:A} B(x)$ に対し、$c \equiv \mathsf{pair}(\mathsf{proj}_1(c), \mathsf{proj}_2(c))$ と定義される。

[000M] **2.3*3 定義**: i を階数、$A, B : \mathcal{U}(i)$ を型とする。型 $A \times B : \mathcal{U}(i)$ を $\sum_{x:A} B$ と定義する。

[000N] **2.3*4 記法**:

- $\times$ は右結合の演算子である。例えば、$A \times B \times C$ は $A \times (B \times C)$ と読む。
- $\sum_{x:A}$ の結合は弱い。例えば、$\sum_{x:A} \sum_{y:B} C \times D$ は $\sum_{x:A} (\sum_{y:B} (C \times D))$ と読む。

対型を繰り返し使って得られる型の要素は何らかの**構造**だと考えられる。構造の成分に名前をつけるための記法を導入する。

[000O] **2.3*5 記法**: 記法 $\mathsf{Record}\{x_1 : A_1, \ldots, x_n : A_n\}$ を次のように定める。

- $\mathsf{Record}\{\}$ は $\mathbf{1}$ のこととする。

- $\mathsf{Record}\{\!|x_1 : A_1, \ldots, x_{n+1} : A_{n+1}|\!\}$ は $\sum_{x_1:A_1} \mathsf{Record}\{\!|x_2 : A_2, \ldots, x_{n+1} : A_{n+1}|\!\}$ のこととする。

この形の型を**レコード型** *(record type)* と呼ぶ。また、記法 $\mathsf{record}\{\!|x_1 \equiv a_1, \ldots, x_n \equiv a_n|\!\}$ を次のように定める。

- $\mathsf{record}\{\!|\,|\!\}$ は $\star$ のこととする。
- $\mathsf{record}\{\!|x_1 \equiv a_1, \ldots, x_{n+1} \equiv a_{n+1}|\!\}$ は $\mathsf{pair}(a_1, \mathsf{record}\{\!|x_2 \equiv a_2, \ldots, x_{n+1} \equiv a_{n+1}|\!\})$ のこととする。

これらの記法は、各 a_i が $A_i[x_1 \mapsto a_1, \ldots, x_{i-1} \mapsto a_{i-1}]$ の要素の時に $\mathsf{record}\{\!|x_1 \equiv a_1, \ldots, x_n \equiv a_n|\!\}$ が $\mathsf{Record}\{\!|x_1 : A_1, \ldots, x_n : A_n|\!\}$ の要素になるように設計されている。さらに、$a : \mathsf{Record}\{\!|x_1 : A_1, \ldots, x_n : A_n|\!\}$ を要素、y を x_1 から x_n のいずれかとする時、記法 $a.y$ を次のように定める。

- y が x_1 の時、$a.x_1$ は $\mathsf{proj}_1(a)$ のこととする。
- y が x_2 から x_n のいずれかの時、$a.y$ は $(\mathsf{proj}_2(a)).y$ のこととする。

この記法は $a.x_i : A_i[x_1 \mapsto a.x_1, \ldots, x_{i-1} \mapsto a.x_{i-1}]$ となるように設計されている。また、$(\mathsf{record}\{\!|x_1 \equiv a_1, \ldots, x_n \equiv a_n|\!\}).x_i \equiv a_i$ であることと $a \equiv \mathsf{record}\{\!|x_1 \equiv a.x_1, \ldots, x_n \equiv a.x_n|\!\}$ であることを確かめられる。

2.3*6 記法: 大きなレコード型を定義する際には、文章内で $\mathsf{Record}\{\!|x_1 : A_1, \ldots, x_n : A_n|\!\}$ と書く代わりに縦に並べて [000W]

- $x_1 : A_1$
- $\vdots$
- $x_n : A_n$

と書くことがある。レコード型の要素を定義する際にも同様に縦に並べて書くことがある。

2.3*7 例: i を階数、$A, B : \mathcal{U}(i)$ を型とする。型 $A \leftrightarrow B : \mathcal{U}(i)$ を $\mathsf{Record}\{\!|\mathsf{to} : A \to B, \mathsf{from} : B \to A|\!\}$ と定義する。$A \leftrightarrow B$ の要素がある時、A と B は**論理的に同値** *(logically equivalent)* であると言う。**反射律**、**対称律**、**推移律**を次のように構成できる。 [001T]

- $\mathsf{record}\{\!|\mathsf{to} \equiv \mathsf{id}, \mathsf{from} \equiv \mathsf{id}|\!\} : A \leftrightarrow A$
- $\lambda e.\mathsf{record}\{\!|\mathsf{to} \equiv e.\mathsf{from}, \mathsf{from} \equiv e.\mathsf{to}|\!\} : (A \leftrightarrow B) \to (B \leftrightarrow A)$
- $\lambda(e, f).\mathsf{record}\{\!|\mathsf{to} \equiv f.\mathsf{to} \circ e.\mathsf{to}, \mathsf{from} \equiv e.\mathsf{from} \circ f.\mathsf{from}|\!\} : (A \leftrightarrow B) \to (B \leftrightarrow C) \to (A \leftrightarrow C)$

2.3*8 演習: i を階数、$A_1, A_2, A_3 : \mathcal{U}(i)$ を型とする。関数 $f : A_1 \to A_2$ と $g : A_2 \to A_3$ と $h : A_3 \to A_1$ がある時、各 A_n と A_m は論理的に同値であることを確かめよ。 [001U]

2.3*9 例: i を階数とする。型 $\mathcal{U}_\bullet(i) : \mathcal{U}(\mathsf{succ}(i))$ を次のレコード型と定義する。 [008A]

- $\mathsf{Carrier} : \mathcal{U}(i)$
- $\mathsf{point} : \mathsf{Carrier}$

$\mathcal{U}_\bullet(i)$ の要素を (階数 i の) **点付き型** *(pointed type)* と呼ぶ。

[008B] **2.3*10 例**: i を階数とする。型 $\mathsf{Magma}(i) : \mathcal{U}(\mathsf{succ}(i))$ を次のレコード型と定義する。

- $\mathsf{Carrier} : \mathcal{U}(i)$
- $\mathsf{op} : \mathsf{Carrier} \to \mathsf{Carrier} \to \mathsf{Carrier}$

$\mathsf{Magma}(i)$ の要素を**マグマ** *(magma)* と呼ぶ。

例 2.3*9 と例 2.3*10 で名前の衝突がある (Carrier) が、型によってどの構造について話しているのか分かるので混乱は無いだろう。

[008C] **2.3*11 例**: i を階数とする。型 $\mathsf{ReflGraph}(i) : \mathcal{U}(\mathsf{succ}(i))$ を次のレコード型と定義する。

- $\mathsf{Vertex} : \mathcal{U}(i)$
- $\mathsf{Edge} : \mathsf{Vertex} \to \mathsf{Vertex} \to \mathcal{U}(i)$
- $\mathsf{refl} : \prod_{x:\mathsf{Vertex}} \mathsf{Edge}(x, x)$

$\mathsf{ReflGraph}(i)$ の要素を**反射的グラフ** *(reflexive graph)* と呼ぶ。

対型に関する基本的な関数をいくつか定義する。

[0014] **2.3*12 例**: i を階数、$A : \mathcal{U}(i)$ を型、$B : A \to \mathcal{U}(i)$ を型の族、$C : \prod_{x:A} B(x) \to \mathcal{U}(i)$ を型の族とする。

- 関数 $f : \prod_{z:\sum_{x:A} B(x)} C(\mathsf{proj}_1(z), \mathsf{proj}_2(z))$ に対し、**カリー化** *(currying)*
$$\mathsf{curry}(f) : \prod_{x:A} \prod_{y:B(x)} C(x, y)$$
を $\lambda(x, y).f(\mathsf{pair}(x, y))$ と定義する。
- 関数 $g : \prod_{x:A} \prod_{y:B(x)} C(x, y)$ に対し、**逆カリー化** *(uncurrying)*
$$\mathsf{uncurry}(g) : \prod_{z:\sum_{x:A} B(x)} C(\mathsf{proj}_1(z), \mathsf{proj}_2(z))$$
を $\lambda z.g(\mathsf{proj}_1(z), \mathsf{proj}_2(z))$ と定義する。

[0015] **2.3*13 演習**: 例 2.3*12 において、$\mathsf{uncurry}(\mathsf{curry}(f)) \equiv f$ と $\mathsf{curry}(\mathsf{uncurry}(g)) \equiv g$ を確かめよ。

[0016] **2.3*14 例**: i を階数、$A : \mathcal{U}(i)$ を型、$B : A \to \mathcal{U}(i)$ を型の族、$C : \prod_{x:A} B(x) \to \mathcal{U}(i)$ を型の族とする。

- 関数 $\mathsf{assoc}(C) : (\sum_{z:\sum_{x:A} B(x)} C(\mathsf{proj}_1(z), \mathsf{proj}_2(z))) \to (\sum_{x:A} \sum_{y:B(x)} C(x, y))$ を $\lambda w.\mathsf{pair}(\mathsf{proj}_1(\mathsf{proj}_1(w)), \mathsf{pair}(\mathsf{proj}_2(\mathsf{proj}_1(w)), \mathsf{proj}_2(w)))$ と定義する。
- 関数 $\mathsf{assoc}^{-1}(C) : (\sum_{x:A} \sum_{y:B(x)} C(x, y)) \to (\sum_{z:\sum_{x:A} B(x)} C(\mathsf{proj}_1(z), \mathsf{proj}_2(z)))$ を $\lambda w.\mathsf{pair}(\mathsf{pair}(\mathsf{proj}_1(w), \mathsf{proj}_1(\mathsf{proj}_2(w))), \mathsf{proj}_2(\mathsf{proj}_2(w)))$ と定義する。

[0017] **2.3*15 演習**: 例 2.3*14 において、$\mathsf{assoc}^{-1}(C) \circ \mathsf{assoc}(C) \equiv \mathsf{id}$ と $\mathsf{assoc}(C) \circ \mathsf{assoc}^{-1}(C) \equiv \mathsf{id}$ を確かめよ。

[0018] **2.3*16 例**: i を階数、$A, B : \mathcal{U}(i)$ を型とする。関数 $\mathsf{sym}(A, B) : A \times B \to B \times A$ を

$\lambda z.\mathsf{pair}(\mathsf{proj}_2(z), \mathsf{proj}_1(z))$ と定義する。

2.3*17 演習: 例 2.3*16 において、$\mathsf{sym}(B,A) \circ \mathsf{sym}(A,B) \equiv \mathsf{id}$ であることを確かめよ。 [0019]

2.3*18 例: i を階数、$A : \mathcal{U}(i)$ を型、$B : A \to \mathcal{U}(i)$ を型の族、$C : \prod_{x:A} B(x) \to \mathcal{U}(i)$ を型の族とする。 [001A]

- 関数 $\mathsf{dist}(C) : (\prod_{x:A} \sum_{y:B(x)} C(x,y)) \to (\sum_{f:\prod_{x:A} B(x)} \prod_{x:A} C(x, f(x)))$ を $\lambda h.\mathsf{pair}(\lambda x.\mathsf{proj}_1(h(x)), \lambda x.\mathsf{proj}_2(h(x)))$ と定義する。
- 関数 $\mathsf{dist}^{-1}(C) : (\sum_{f:\prod_{x:A} B(x)} \prod_{x:A} C(x, f(x))) \to (\prod_{x:A} \sum_{y:B(x)} C(x,y))$ を $\lambda k.\lambda x.\mathsf{pair}(\mathsf{proj}_1(k(x)), \mathsf{proj}_2(k(x)))$ と定義する。

2.3*19 演習: 例 2.3*18 において、$\mathsf{dist}^{-1}(C) \circ \mathsf{dist}(C) \equiv \mathsf{id}$ と $\mathsf{dist}(C) \circ \mathsf{dist}^{-1}(C) \equiv \mathsf{id}$ を確かめよ。 [001B]

2.3*20 演習: i を階数、$A : \mathcal{U}(i)$ を型とする。関数 $\mathsf{diag}(A) : A \to A \times A$ であって、任意の $a : A$ に対して $\mathsf{proj}_1(\mathsf{diag}(A,a)) \equiv a$ かつ $\mathsf{proj}_2(\mathsf{diag}(A,a)) \equiv a$ となるものを構成せよ。 [002M]

2.4 同一視型 [000C]

同一視型はホモトピー型理論において最も特徴的な型である。

2.4*1 規則: i を階数、$A : \mathcal{U}(i)$ を型、$a_1 : A$ を要素とする。 [000P]

- 要素 $a_2 : A$ に対し、**同一視型** *(identity type)* $a_1 = a_2 : \mathcal{U}(i)$ を構成できる。$a_1 = a_2$ の要素を a_1 と a_2 の**同一視** *(identification)* と呼ぶ。
- 要素 $\mathsf{refl}\{a_1\} : a_1 = a_1$ を構成できる。
- $a_2 : A$ と $p : a_1 = a_2$ を要素、j を階数、$B : \prod_{\{x:A\}} a_1 = x \to \mathcal{U}(j)$ を型の族、$b : B(\mathsf{refl})$ を要素とすると、要素 $\mathsf{ind}_=(p, B, b) : B(p)$ を構成できる。
- j を階数、$B : \prod_{\{x:A\}} a_1 = x \to \mathcal{U}(j)$ を型の族、$b : B(\mathsf{refl})$ を要素とすると、$\mathsf{ind}_=(\mathsf{refl}, B, b) \equiv b$ と定義される。

[HoTT-Book] にならって同一視型に等号の記号を使うが、その意味は従来の数学の等号とは大きく異なる。従来の数学では、$a_1 = a_2$ といえば a_1 と a_2 が等しいという命題であるが、型理論では $a_1 = a_2$ はあくまで型である。従って、$a_1 = a_2$ の要素というものを考えることができ、それは非形式的には a_1 と a_2 の同一視のしかたと解釈される。

規則 2.4*1 について説明する。$\mathsf{refl} : a_1 = a_1$ は a_1 とそれ自身の自明な同一視を表す。残りの規則はいわゆる**帰納法原理**の一例で、任意の $a_2 : A$ と同一視型の要素 $p : a_1 = a_2$ を使ってなんらかを構成するためには p が refl の場合の構成 $(b : B(\mathsf{refl}))$ を与えれば十分であると読める。注意するべきこととして、この帰納法原理から $a_1 = a_1$ の要素は refl しか無いことは**導出されない**。同一視型の規則の正しい読み方は型の族 $\lambda x.(a_1 = x) : A \to \mathcal{U}(i)$ が $\mathsf{refl} : a_1 = a_1$ で自由に生成されることであって、個々の型 $a_1 = a_2$ については特に言えることはない。

同一視型からの関数を定義するには**パターンマッチ**が便利である。パターンマッチは一

般に帰納的型からの関数を構成子による場合分けで定義する手法である。同一視型の構成子は refl だけなのでその場合の定義だけを与えれば関数を定義できることになる。

[008D] **2.4*2 記法**: $\mathsf{ind}_=$ を使って関数を定義するには次のいずれかのような言い回しをする。

- 関数 $f : \prod_{\{x:A\}} \prod_{p:a_1=x} B(p)$ を $\lambda(x,p).\mathsf{ind}_=(p,B,b)$ と定義する。
- 関数 $f : \prod_{\{x:A\}} \prod_{p:a_1=x} B(p)$ を $f(\mathsf{refl}) \equiv b$ で定義する。
- $x : A$ を要素、$p : a_1 = x$ を同一視とする。要素 $f(p) : B(p)$ を $f(\mathsf{refl}) \equiv b$ で定義する。

パターンマッチが有効なのはもちろん $\mathsf{ind}_=$ を使ったものに書き直せる時だけである。例えば、「関数 $f : a = a \to B$ を $f(\mathsf{refl}) \equiv b$ と定義する」という使い方はできない。

$a_1 = a_2$ が同一視型と呼ぶに価することを確認するために、いくつかの期待される関数を構成しよう。

[001C] **2.4*3 定義**: i を階数、$A : \mathcal{U}(i)$ を型、j を階数、$B : A \to \mathcal{U}(j)$ を型の族とする。$a_1, a_2 : A$ と $p : a_1 = a_2$ を要素とする。**輸送関数** *(transport function)*

$$\mathsf{transport}(B,p) : B(a_1) \to B(a_2)$$

を $\mathsf{transport}(B, \mathsf{refl}) \equiv \mathsf{id}$ で定義する。

[001G] **2.4*4 比較**: 述語論理において、B を一変数の述語とすると、$x_1 = x_2 \to B(x_1) \to B(x_2)$ が成り立つ。これは等しい対象は述語によって区別できないことを表す。定義 2.4*3 はこの類似で、型理論においては同一視される要素を型の族では区別できないことを表す。

[001D] **2.4*5 定義**: i を階数、$A : \mathcal{U}(i)$ を型、$a_0, a_1, a_2 : A$ を要素、$p_1 : a_0 = a_1$ と $p_2 : a_0 = a_2$ を同一視とする。同一視 $\mathsf{ext}(p_1,p_2) : a_1 = a_2$ を $\mathsf{transport}(\lambda x.(x = a_2), p_1, p_2)$ と定義する。

[001E] **2.4*6 定義**: i を階数、$A : \mathcal{U}(i)$ を型とする。

- 関数 $\mathsf{sym} : \prod_{\{x_1,x_2:A\}} x_1 = x_2 \to x_2 = x_1$ を $\lambda(x_1,x_2,z).\mathsf{ext}(z, \mathsf{refl})$ と定義する。$\mathsf{sym}(p)$ を p^{-1} とも書く。
- 関数 $\mathsf{trans} : \prod_{\{x_1,x_2,x_3:A\}} x_1 = x_2 \to x_2 = x_3 \to x_1 = x_3$ を $\lambda(x_1,x_2,x_3,z,w).\mathsf{ext}(z^{-1}, w)$ と定義する。$\mathsf{trans}(p,q)$ を $q \circ p$ とも書く。

[001H] **2.4*7 比較**: 述語論理において、**対称律** $x_1 = x_2 \to x_2 = x_1$ と**推移律** $x_1 = x_2 \to x_2 = x_3 \to x_1 = x_3$ が成り立つ。定義 2.4*6 はこの類似である。ちなみに、$\mathsf{refl} : x = x$ は**反射律**とも思える。

[001F] **2.4*8 定義**: i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数とする。関数

$$\mathsf{ap}(f) : \prod_{\{x_1,x_2:A\}} x_1 = x_2 \to f(x_1) = f(x_2)$$

を $\lambda(x_1,x_2,z).\mathsf{transport}(\lambda x.(f(x_1) = f(x)), z, \mathsf{refl})$ と定義する。文脈上わかる場合は、$\mathsf{ap}(f,p)$ のことを $f(p)$ と書いてしまうこともある。

[001I] **2.4*9 比較**: 述語論理において、f を一変数の関数とすると、$x_1 = x_2 \to f(x_1) = f(x_2)$

が成り立つ。これは関数が等しさを保つことを表す。定義 2.4*8 はこの類似で、型理論において関数は同一視を保つことを表す。

2.4.1 高次グルーポイド構造 [002N]

要素 $a_1, a_2 : A$ に対して、$a_1 = a_2$ がまた型であるということは、同一視 $p_1, p_2 : a_1 = a_2$ に対してもまた同一視型 $p_1 = p_2$ がある。同一視 $q_1, q_2 : p_1 = p_2$ に対してさらに同一視型 $q_1 = q_2$ があり、これを繰り返すと好きなだけ「高次の」同一視型を得る。実は、これらの高次の同一視型たちは**∞グルーポイド**と呼ばれる構造の一部であることが知られている [Lumsdaine--2010-0000][van-den-Berg--Garner--2011-0000]。∞グルーポイドの構造は無限個の演算を持つ複雑なもので深入りはしない。ただ覚えておくとよいのは、あらゆる型が自動的にそのような豊富な構造を持ち、あらゆる構成が自動的にその構造と整合的になるということである。

∞グルーポイドの持つ構造の一部を挙げる。∞グルーポイドの文脈では要素 $p : a_1 = a_2$ は**射**と呼ばれる。$\mathsf{refl} : a_1 = a_1$ は**恒等射**、$p^{-1} : a_2 = a_1$ は**逆射**、$q \circ p : a_1 = a_3$ は**合成射**と呼ばれる。$\mathsf{ext}(p_1, p_2) : a_1 = a_2$ は $p_2 : a_0 = a_2$ の $p_1 : a_0 = a_1$ に沿った**拡張**である。

2.4.1*1 例: i を階数、$A : \mathcal{U}(i)$ を型、$a_1, a_2 : A$ を要素、$p : a_1 = a_2$ を同一視とする。同一視 $\mathsf{ext\text{-}self}(p) : \mathsf{ext}(p, p) = \mathsf{refl}$ を構成する。定義より $\mathsf{ext}(\mathsf{refl}, \mathsf{refl}) \equiv \mathsf{refl}$ なので帰納法により $\mathsf{ext\text{-}self}(\mathsf{refl}) \equiv \mathsf{refl}$ と定義すればよい。 [001M]

2.4.1*2 例: i を階数、$A : \mathcal{U}(i)$ を型、$a_1, a_2 : A$ を要素、$p : a_1 = a_2$ を同一視とする。同一視 $\mathsf{sym\text{-}sym}(p) : (p^{-1})^{-1} = p$ を帰納法により $\mathsf{sym\text{-}sym}(\mathsf{refl}) \equiv \mathsf{refl}$ と定義する。 [0027]

2.4.1*3 例: i を階数、$A : \mathcal{U}(i)$ を型、$a_1, a_2, a_3, a_4 : A$ を要素、$p_1 : a_1 = a_2$ と $p_2 : a_2 = a_3$ と $p_3 : a_3 = a_4$ を同一視とする。同一視 $\mathsf{assoc}(p_3, p_2, p_1) : (p_3 \circ p_2) \circ p_1 = p_3 \circ (p_2 \circ p_1)$ を構成する。同一視型の帰納法により、$\mathsf{assoc}(p_3, p_2, \mathsf{refl}) \equiv \mathsf{refl}$ と定義すればよい。 [002O]

2.4.1*4 演習: i を階数、$A : \mathcal{U}(i)$ を型、$a_1, a_2 : A$ を要素、$p : a_1 = a_2$ を同一視とする。同一視 $\mathsf{unit\text{-}l}(p) : \mathsf{refl} \circ p = p$ と $\mathsf{unit\text{-}r}(p) : p = p \circ \mathsf{refl}$ を構成せよ。 [002P]

2.4.1*5 演習: i を階数、$A : \mathcal{U}(i)$ を型、$a_1, a_2 : A$ を要素、$p : a_1 = a_2$ を同一視とする。同一視 $\mathsf{inv\text{-}l}(p) : p^{-1} \circ p = \mathsf{refl}$ と $\mathsf{inv\text{-}r}(p) : p \circ p^{-1} = \mathsf{refl}$ を構成せよ。 [0047]

2.4.1*6 演習: いわゆる *Mac Lane* **五角形**を表す型とその要素を構成せよ。 [002R]

2.4.1*7 例: i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数とする。 [002Q]

1. 要素 $a_1, a_2, a_3 : A$ と同一視 $p_1 : a_1 = a_2$ と $p_2 : a_2 = a_3$ に対して、同一視 $\mathsf{ap\text{-}comp}(f, p_2, p_1) : \mathsf{ap}(f, p_2 \circ p_1) = \mathsf{ap}(f, p_2) \circ \mathsf{ap}(f, p_1)$ を構成できる。実際、$\mathsf{ap\text{-}comp}(f, p_2, \mathsf{refl}) \equiv \mathsf{refl}$ と定義すればよい。
2. 要素 $a_1, a_2 : A$ と同一視 $p : a_1 = a_2$ に対して、同一視 $\mathsf{ap\text{-}inv}(f, p) : \mathsf{ap}(f, p^{-1}) = (\mathsf{ap}(f, p))^{-1}$ を構成できる。実際、$\mathsf{ap\text{-}inv}(f, \mathsf{refl}) \equiv \mathsf{refl}$ と定義すればよい。

2.4.1*8 例: i を階数、$A : \mathcal{U}(i)$ を型、$B : A \to \mathcal{U}(i)$ を型の族とする。要素 $a_1, a_2, a_3 : A$ と同一視 $p_1 : a_1 = a_2$ と $p_2 : a_2 = a_3$ に対して、同一視 [002S]

transport-comp(B, p_2, p_1) : transport$(B, p_2 \circ p_1) =$ transport$(B, p_2) \circ$ transport(B, p_1) を構成できる。実際、transport-comp$(B, p_2, \mathsf{refl}) \equiv \mathsf{refl}$ と定義すればよい。

[002T] 2.5 自然数

今までに導入した型だけでは何も面白い数学はできない。数学を面白くするには**自然数**の概念は必須であろう。自然数の型は**帰納的型**の例として定義される。

[002V] **2.5*1 規則**:

- **自然数型** *(type of natural numbers)*$\mathbb{N} : \mathcal{U}(0)$ を構成できる。$\mathbb{N}$ の要素を**自然数** *(natural number)* と呼ぶ。
- 要素 $0 : \mathbb{N}$ を構成できる。
- 要素 $n : \mathbb{N}$ に対して、要素 $\mathsf{succ}(n) : \mathbb{N}$ を構成できる。
- $n : \mathbb{N}$ を要素、i を階数、$A : \mathbb{N} \to \mathcal{U}(i)$ を型の族、$a : A(0)$ を要素、$f : \prod_{\{x:\mathbb{N}\}} A(x) \to A(\mathsf{succ}(x))$ を関数とすると、要素 $\mathsf{ind}_{\mathbb{N}}(n, A, a, f) : A(n)$ を構成できる。
- i を階数、$A : \mathbb{N} \to \mathcal{U}(i)$ を型の族、$a : A(0)$ を要素、$f : \prod_{\{x:\mathbb{N}\}} A(x) \to A(\mathsf{succ}(x))$ を関数とすると、$\mathsf{ind}_{\mathbb{N}}(0, A, a, f) \equiv a$ と定義される。
- $n : \mathbb{N}$ を要素、i を階数、$A : \mathbb{N} \to \mathcal{U}(i)$ を型の族、$a : A(0)$ を要素、$f : \prod_{\{x:\mathbb{N}\}} A(x) \to A(\mathsf{succ}(x))$ を関数とすると、$\mathsf{ind}_{\mathbb{N}}(\mathsf{succ}(n), A, a, f) \equiv f(\mathsf{ind}_{\mathbb{N}}(n, A, a, f))$ と定義される。

一般に帰納的型はいくつかの**構成子** *(constructor)* によって定められる。$\mathbb{N}$ の場合、0 と succ が構成子である。これらは $\mathbb{N}$ の要素を構成する方法を与える。任意の $\mathbb{N}$ の要素がこれらの構成子のみを使って構成されることを表すために、**帰納法原理** *(induction principle)* を規則として認める。$\mathsf{ind}_{\mathbb{N}}$ に関する規則は、自然数 n を使ってなんらかを構成するには、0 の場合の構成 $(a : A(0))$ と $\mathsf{succ}(x)$ の場合の構成 $(f : \prod_{\{x:\mathbb{N}\}} A(x) \to A(\mathsf{succ}(x)))$ を与えれば十分であることを意味する。つまり、型理論の中の人にとっては自然数とは 0 と succ のみを使って構成されたものである。さらに、succ の場合の構成においては、x の場合の構成 (f の引数 $y : A(x)$) は既に定義されたと仮定してよい。これは**再帰的定義** *(recursive definition)* を可能にする。

自然数型からの関数を定義するには**パターンマッチ**が便利である。

[008E] **2.5*2 記法**: $\mathsf{ind}_{\mathbb{N}}$ を使って関数を定義するには次のいずれかのような言い回しをする。

- 関数 $h : \prod_{x:\mathbb{N}} A(x)$ を $\lambda x.\mathsf{ind}_{\mathbb{N}}(x, A, a, f)$ と定義する。
- 関数 $h : \prod_{x:\mathbb{N}} A(x)$ を $h(0) \equiv a$ と $h(\mathsf{succ}(x)) \equiv f(h(x))$ で定義する。
- $x : \mathbb{N}$ を要素とする。要素 $h(x) : A(x)$ を $h(0) \equiv a$ と $h(\mathsf{succ}(x)) \equiv f(h(x))$ で定義する。

現実では多変数の関数の引数の一つにパターンマッチを使う場合がある。自然数のパターンマッチの succ の場合 $h(\mathsf{succ}(x)) \equiv f(h(x))$ では右辺に $h(x)$ のパターンを見出すのに慣れが要ることもある。

[002W] **2.5*3 例**: $n_1, n_2 : \mathbb{N}$ を要素とする。要素 $\mathsf{plus}(n_1, n_2) : \mathbb{N}$ を構成する。自然数の帰納法を

使い、$\mathsf{plus}(0, n_2) \equiv n_2$ と $\mathsf{plus}(\mathsf{succ}(n_1), n_2) \equiv \mathsf{succ}(\mathsf{plus}(n_1, n_2))$ と定義する。形式的には $\mathsf{plus}(n_1, n_2) \equiv \mathsf{ind}_{\mathbb{N}}(n_1, \lambda x.\mathbb{N}, n_2, \lambda(x, y).\mathsf{succ}(y))$ と定義できる。

2.5*4 例: $n : \mathbb{N}$ を要素、$f : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$ を関数とする。**原始再帰** *(primitive recursion)* による関数 $\mathsf{prim\text{-}rec}(n, f) : \mathbb{N} \to \mathbb{N}$ が $\mathsf{prim\text{-}rec}(n, f, 0) \equiv n$ と $\mathsf{prim\text{-}rec}(n, f, \mathsf{succ}(m)) \equiv f(m, \mathsf{prim\text{-}rec}(n, f, m))$ で定義される。形式的には $\mathsf{prim\text{-}rec}(n, f) \equiv \lambda m.\mathsf{ind}_{\mathbb{N}}(m, \lambda x.\mathbb{N}, n, \lambda(x, y).f(x, y))$ である。よって、いわゆる**原始再帰的関数**はすべて構成できる。 [002X]

2.5*5 演習: *Ackermann* **関数**は二変数の関数 $\mathsf{ack} : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$ で、次のように定義される。 [002Y]

- $\mathsf{ack}(0, n) \equiv \mathsf{succ}(n)$
- $\mathsf{ack}(\mathsf{succ}(m), 0) \equiv \mathsf{ack}(m, \mathsf{succ}(0))$
- $\mathsf{ack}(\mathsf{succ}(m), \mathsf{succ}(n)) \equiv \mathsf{ack}(m, \mathsf{ack}(\mathsf{succ}(m), n))$

Ackermann 関数の構成を規則 2.5*1 に基づいて正当化せよ。ちなみに、Ackermann 関数は原始再帰的でないことが知られているので、例 2.5*4 の特別な場合としては定義できない。関数型を使えることに注意するとよい。

2.6 有限余積 [002U]

自然数型の他にも重要な帰納的型がある。最も簡単な帰納的型は**空型**である。

2.6*1 規則: [0030]

- **空型** *(empty type)* $\mathbf{0} : \mathcal{U}(0)$ を構成できる。
- $c : \mathbf{0}$ を要素、i を階数、$A : \mathbf{0} \to \mathcal{U}(i)$ を型の族とすると、要素 $\mathsf{ind}_{\mathbf{0}}(c, A) : A(c)$ を構成できる。

空型には構成子が一つも与えられていない。よって、型理論の中の人にとっては $\mathbf{0}$ の要素は存在しないはずである。言い換えれば、$\mathbf{0}$ の要素は**矛盾** *(contradiction)* を表し、帰納法原理は矛盾からはすべてを導けること、**爆発原理** *(principle of explosion* **または** *ex falso quodlibet)* を意味する。

次に簡単な帰納的型は**余積**である。

2.6*2 規則: i を階数、$A, B : \mathcal{U}(i)$ を型とする。 [002Z]

- **余積** *(coproduct)* $A + B : \mathcal{U}(i)$ を構成できる。
- 要素 $a : A$ に対して、要素 $\mathsf{in}_1(a) : A + B$ を構成できる。
- 要素 $b : B$ に対して、要素 $\mathsf{in}_2(b) : A + B$ を構成できる。
- $c : A + B$ を要素、j を階数、$D : A + B \to \mathcal{U}(j)$ を型の族、$d_1 : \prod_{x:A} D(\mathsf{in}_1(x))$ を要素、$d_2 : \prod_{y:B} D(\mathsf{in}_2(y))$ を要素とすると、要素 $\mathsf{ind}_+(c, D, d_1, d_2) : D(c)$ を構成できる。
- $a : A$ を要素、j を階数、$D : A + B \to \mathcal{U}(j)$ を型の族、$d_1 : \prod_{x:A} D(\mathsf{in}_1(x))$ を要素、$d_2 : \prod_{y:B} D(\mathsf{in}_2(y))$ を要素とすると、$\mathsf{ind}_+(\mathsf{in}_1(a), D, d_1, d_2) \equiv d_1(a)$ と定義

される。

- $b : B$ を要素、j を階数、$D : A + B \to \mathcal{U}(j)$ を型の族、$d_1 : \prod_{x:A} D(\mathsf{in}_1(x))$ を要素、$d_2 : \prod_{y:B} D(\mathsf{in}_2(y))$ を要素とすると、$\mathsf{ind}_+(\mathsf{in}_2(b), D, d_1, d_2) \equiv d_2(b)$ と定義される。

in_1 と in_2 が $A + B$ の構成子であり、帰納法原理はやはりこれらの構成子のみを使って得られるものだけが $A + B$ の要素であることを表す。

余積のパターンマッチは次のようになる。

[008F] **2.6*3 記法**: ind_+ を使って関数を定義するには次のような言い回しをする。

- 関数 $f : \prod_{z:A+B} D(z)$ を $\lambda z.\mathsf{ind}_+(z, D, d_1, d_2)$ と定義する。
- 関数 $f : \prod_{z:A+B} D(z)$ を $f(\mathsf{in}_1(x)) \equiv d_1(x)$ と $f(\mathsf{in}_2(y)) \equiv d_2(y)$ で定義する。
- $z : A + B$ を要素とする。要素 $f(z) : D(z)$ を $f(\mathsf{in}_1(x)) \equiv d_1(x)$ と $f(\mathsf{in}_2(y)) \equiv d_2(y)$ で定義する。

3 一価性公理 [0002]

一価性公理 (univalence axiom) はホモトピー型理論において最も重要な公理で、型理論をまさに「ホモトピー論的」なものに強制する公理である。非形式的には、型 $A, B : \mathcal{U}(i)$ に対して同一視型 $A = B$ と**同値**のなす型 $A \simeq B$ が同値になることを要請する。

一価性について述べるには同値の概念が不可欠である。同値の定義はいくつか考えられるが、本書では関数であって各点の逆像が**可縮**であるものという定義を採用する。型が可縮であるとは非形式的には要素をただ一つ持つということであり、この同値の定義は直観的にも自然であろう。技術的な利点として、可縮性は非常に扱いやすい性質であるという点、ホモトピー型理論における他の概念も可縮性を軸に定義することができるという点がある。3.1 節で可縮性についての基本事項まとめる。

一価性公理は宇宙 $\mathcal{U}(i)$ の同一視型の特徴付けを与えているとも解釈できる。諸々の型の同一視型の特徴付けについての一般論として、**同一視型の基本定理**を 3.2 節で証明する。「基本定理」と名付けられる通り、この定理は本書の至る所で使われる。3.3 節で一価性公理を導入する。また、関数型の同一視型の特徴付け (いわゆる**関数外延性**) にもなんらかの公理が必要であることが知られているので、3.4 節で**関数外延性公理**を導入する。

一価性公理は後に導入する高次帰納的型と合わせてその真価を発揮するのだが、一価性公理だけでもいくつか興味深い帰結を得られる。その一つは**構造同一原理** (structure identity principle) と呼ばれるもので、なんらかの数学的構造の同一視型はその構造を保つ同値の型と同値であるという原理である。構造同一原理を一般的に述べるには「構造」とは厳密にどう定義されるかという問題が大きいので立ち入らず、3.5 節でいくつかの例を挙げるにとどめる。

3.6 節はやや技術的な話で、同値の概念の基本性質を見る。

3.1 可縮性 [000S]

可縮性はホモトピー型理論において中心的な役割を果たす概念である。

3.1*1 定義: i を階数、$A : \mathcal{U}(i)$ を型とする。型 $\mathsf{IsContr}(A) : \mathcal{U}(i)$ を次のレコード型と定義する。 [000T]

- $\mathsf{center} : A$
- $\mathsf{contr} : \prod_{x:A} \mathsf{center} = x$

$\mathsf{IsContr}(A)$ の要素がある時、A は**可縮** *(contractible)* であると言う。

つまり、型 A が可縮であるとは、ある要素 $\mathsf{center} : A$ があり、すべての A の要素は

center と同一視されるということである。やや非形式的に言うと、A はただ一つの要素を持つということである。

後に示すことだが、$\mathsf{IsContr}(A)$ の任意の二つの要素は (関数外延性の下で) 同一視される (命題 4.1*7)。したがって、A が可縮であると言った場合、$\mathsf{IsContr}(A)$ の要素があることは重要だが、その要素の具体的な定義は気にしなくてよい。あるいは、可縮性は型についての**命題**であり、それを**証明**できることは重要だがどのように証明されたかは重要ではないとも言える。

[001O] **3.1*2 例**: $\mathbf{1}$ は可縮である。実際、$\mathsf{record}\{\mathsf{center} \equiv \star, \mathsf{contr} \equiv \lambda x.\mathsf{refl}\} : \mathsf{IsContr}(\mathbf{1})$ を確かめられる。

型は∞グルーポイドの構造を持っているので、ただ一つの要素を持つとはいっても異なる同一視のしかたがある可能性が懸念される。次の命題 3.1*3 は、可縮性は見た目よりもずっと強い条件で、任意の二つの要素の同一視もただ一つだけあることを示す。命題 3.1*3 を繰り返し使えば、任意に高次の同一視もただ一つだけあることが分かる。

[001L] **3.1*3 命題**: i を階数、$A : \mathcal{U}(i)$ を型、$a_1, a_2 : A$ を要素とする。A が可縮ならば $a_1 = a_2$ も可縮である。

証明: $c : \mathsf{IsContr}(A)$ と仮定する。要素 $p : a_1 = a_2$ と $q : \prod_{z:a_1=a_2} p = z$ を構成すればよい。仮定 c より $a_0 : A$ と $r : \prod_{x:A} a_0 = x$ を得る。$p \equiv \mathsf{ext}(r(a_1), r(a_2))$ と定義する。q については、一般化したもの $q' : \prod_{x:A} \prod_{z:a_1=x} \mathsf{ext}(r(a_1), r(x)) = z$ を構成し、$q \equiv q'(a_2)$ と定義する。同一視型の帰納法により、要素 $s : \mathsf{ext}(r(a_1), r(a_1)) = \mathsf{refl}$ を構成すればよいが、これは例 2.4.1*1 で構成した。 ■

型の可縮性は次の命題 3.1*4 や命題 3.1*6 を使って示される場合が多い。

[001N] **3.1*4 命題**: i を階数、$A : \mathcal{U}(i)$ を型、$a : A$ を要素とする。型 $\sum_{x:A} a = x$ は可縮である。

証明: 要素 $c : \sum_{x:A} a = x$ と $r : \prod_{z:\sum_{x:A} a=x} c = z$ を構成すればよい。$c \equiv \mathsf{pair}(a, \mathsf{refl})$ と定義する。r については、カリー化 (例 2.3*12) により、$\prod_{x:A} \prod_{w:a=x} c = \mathsf{pair}(x, w)$ の要素を構成すればよい。同一視型の帰納法により、$c = \mathsf{pair}(a, \mathsf{refl})$ の要素を構成すればよいが、c の定義より refl とすればよい。 ■

命題 3.1*4 は同一視型の帰納法の別表現と考えられる。同一視型の帰納法は型の族 $\lambda x.(a = x) : A \to \mathcal{U}(i)$ が $\mathsf{refl} : a = a$ で自由に生成されることを表す。これは、対型 $\sum_{x:A} a = x$ が $\mathsf{pair}(a, \mathsf{refl})$ で自由に生成されると言い換えられる。一つの要素で生成される型はその要素しか持たないと期待され、命題 3.1*4 は実際にそうだと言っている。

命題 3.1*6 は可縮性が**レトラクト**で閉じることを主張する。

[001J] **3.1*5 定義**: i を階数、$A, B : \mathcal{U}(i)$ を型とする。型 $\mathsf{Retract}(A, B) : \mathcal{U}(i)$ を次のレコード型と定義する。

- $\mathsf{retraction} : B \to A$
- $\mathsf{section} : A \to B$
- $\mathsf{r\text{-}s} : \prod_{x:A} \mathsf{retraction}(\mathsf{section}(x)) = x$

$\mathsf{Retract}(A,B)$ は $A \triangleleft B$ と書くこともある。$A \triangleleft B$ の要素がある時、A は B の**レトラクト** *(retract)* であると言う。また、$A \triangleleft\triangleright B$ を $(A \triangleleft B) \times (B \triangleleft A)$ と定義する。

$\mathsf{Retract}(A,B)$ は $\mathsf{IsContr}(A)$ と違って命題ではないが、実用上は $\mathsf{Retract}(A,B)$ の要素の具体的な定義は重要ではない場合が多い。これは命題 3.1*6 のように、ある性質がレトラクトで閉じるという使われ方をする場合が多いからである。

3.1*6 命題: i を階数、$A, B : \mathcal{U}(i)$ を型、$r : \mathsf{Retract}(A,B)$ を要素とする。B が可縮ならば A も可縮である。 [001K]

証明: $d : \mathsf{IsContr}(B)$ と仮定する。要素 $a : A$ と $p : \prod_{x:A} a = x$ を構成すればよい。仮定 d から $b : B$ と $q : \prod_{y:B} b = y$ を得る。仮定 r から $f : B \to A$ と $g : A \to B$ と $t : \prod_{x:A} f(g(x)) = x$ を得る。$a \equiv f(b)$ と定義する。p を定義するために、$x : A$ を仮定する。$q(g(x)) : b = g(x)$ と $t(x) : f(g(x)) = x$ に注意して、$p(x) \equiv t(x) \circ \mathsf{ap}(f, q(g(x)))$ と定義すればよい。 ■

可縮性を使って型の**同値**が定義される。

3.1*7 定義: i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数、$b : B$ を要素とする。型 $\mathsf{Fiber}(f,b) : \mathcal{U}(i)$ を $\mathsf{Record}\{\!|\mathsf{elem} : A, \mathsf{id} : f(\mathsf{elem}) = b|\!\}$ と定義する。 [001P]

3.1*8 定義: i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数とする。型 $\mathsf{IsEquiv}(f) : \mathcal{U}(i)$ を $\prod_{y:B} \mathsf{IsContr}(\mathsf{Fiber}(f,y))$ と定義する。$\mathsf{IsEquiv}(f)$ の要素がある時、f は**同値** *(equivalence)* であると言う。 [001Q]

3.1*9 定義: i を階数、$A, B : \mathcal{U}(i)$ を型とする。型 $A \simeq B : \mathcal{U}(i)$ を $\mathsf{Record}\{\!|\mathsf{fun} : A \to B, \mathsf{is\text{-}equiv} : \mathsf{IsEquiv}(\mathsf{fun})|\!\}$ と定義する。 [000V]

この定義による同値の概念が妥当なものであるかは自明ではない。直観的には、$\mathsf{Fiber}(f,b)$ は b の f による逆像であり、$\mathsf{IsContr}(\mathsf{Fiber}(f,b))$ は逆像がただ一つの要素を持つことを表す。これはもっともらしい定義だが、$\simeq$ が反射的、対称、推移的であることすら定義 3.1*9 から直ちに分かることではない。3.6 節でこの同値の概念が妥当であることを説明するが、その前にいくつか重要な定理と概念を導入する。

3.2 同一視型の基本定理 [001R]

同一視型はすべての型に対して一様に定義されているが、個々の型については具体的な同一視のしかたが期待される。例えば、対型の要素 $c_1, c_2 : A \times B$ の「自然な」同一視のしかたは $\mathsf{proj}_1(c_1)$ と $\mathsf{proj}_1(c_2)$ を同一視しかつ $\mathsf{proj}_2(c_1)$ と $\mathsf{proj}_2(c_2)$ を同一視することである。つまり、同値 $(c_1 = c_2) \simeq (\mathsf{proj}_1(c_1) = \mathsf{proj}_1(c_2)) \times (\mathsf{proj}_2(c_1) = \mathsf{proj}_2(c_2))$ を構成できると期待される (例 3.2*9)。**同一視型の基本定理** (定理 3.2*4) は、この手の同値を構成する手順を与える。

型 $A : \mathcal{U}(i)$ に興味があるとして、要素 $a : A$ に対して同一視型の族 $\lambda x.(a = x) : A \to \mathcal{U}(i)$ を特徴付けることを考える。具体的に特徴付けの候補 $B : A \to \mathcal{U}(i)$ を見つけたとしよう。これが正しいものなら、refl に対応する要素 $b : B(a)$ があるはずである。同一視型の帰納法により、b は関数 $b' : \prod_{x:A} a = x \to B(x)$ に拡張される。$b'(x) : a = x \to B(x)$

は「標準的」な比較関数であり、これが同値であることを示したい。$b'(x)$ の同値性を定義に従って示すことは難しくはないが筋が良いとも言えない。同一視型の基本定理は、すべての $b'(x)$ が同値であることと $\sum_{x:A} B(x)$ が可縮であることが論理的に同値であると主張する。次の点から後者の方が示しやすい性質であると思われる。

- 可縮性は様々な型の構成について閉じる (例えば命題 3.1*3 や命題 3.1*4)。
- $\sum_{x:A} B(x)$ の可縮性は a や b や b' に依らない性質である。

[001V] **3.2*1 補題**: i を階数、$A, B : \mathcal{U}(i)$ を型、$e : A \simeq B$ を同値とすると、$\mathsf{Retract}(B, A)$ の要素を構成できる。

証明: 仮定 e から $f : A \to B$ と $H : \mathsf{IsEquiv}(f)$ を得る。任意の $y : B$ に対して $H(y) : \mathsf{IsContr}(\mathsf{Fiber}(f, y))$ を得るので、特に関数 $G : \prod_{y:B} \mathsf{Fiber}(f, y)$ を得る。例 2.3*18 の要領で G から関数 $g : B \to A$ と同一視 $p : \prod_{y:B} f(g(y)) = y$ を得る。これで要素 $\mathsf{record}\{\!|\mathsf{retraction} \equiv f, \mathsf{section} \equiv g, \mathsf{r\text{-}s} \equiv p|\!\}$: $\mathsf{Retract}(B, A)$ を構成できた。 ■

[001X] **3.2*2 補題**: i を階数、$A : \mathcal{U}(i)$ を型、$B : A \to \mathcal{U}(i)$ を型の族、$c_1, c_2 : \sum_{x:A} B(x)$ を要素とすると、

$$\mathsf{Retract}(\textstyle\sum_{z:\mathsf{proj}_1(c_1)=\mathsf{proj}_1(c_2)} \mathsf{transport}(B, z, \mathsf{proj}_2(c_1)) = \mathsf{proj}_2(c_2), c_1 = c_2)$$

の要素を構成できる。

証明: 関数 $f : (\sum_{z:\mathsf{proj}_1(c_1)=\mathsf{proj}_1(c_2)} \mathsf{transport}(B, z, \mathsf{proj}_2(c_1)) = \mathsf{proj}_2(c_2)) \to c_1 = c_2$ と $g : c_1 = c_2 \to (\sum_{z:\mathsf{proj}_1(c_1)=\mathsf{proj}_1(c_2)} \mathsf{transport}(B, z, \mathsf{proj}_2(c_1)) = \mathsf{proj}_2(c_2))$ と同一視 $p : \prod_w g(f(w)) = w$ を構成する。f についてはカリー化、一般化して

$$f' : \textstyle\prod_{\{x:A\}} \prod_{\{y:B(x)\}} \prod_{z:\mathsf{proj}_1(c_1)=x} \mathsf{transport}(B, z, \mathsf{proj}_2(c_1)) = y \to c_1 = \mathsf{pair}(x, y)$$

を構成すればよいが、同一視型の帰納法により $f'(\mathsf{refl}, \mathsf{refl}) \equiv \mathsf{refl}$ と定義できる。g は一般化して

$$g' : \textstyle\prod_{\{y:\sum_{x:A} B(x)\}} c_1 = y \to (\sum_{z:\mathsf{proj}_1(c_1)=\mathsf{proj}_1(y)} \mathsf{transport}(B, z, \mathsf{proj}_2(c_1)) = \mathsf{proj}_2(y))$$

を帰納法で $g'(\mathsf{refl}) \equiv \mathsf{pair}(\mathsf{refl}, \mathsf{refl})$ と定義する。p も f と同様にカリー化、一般化して

$$p' : \textstyle\prod_{\{x\}} \prod_{\{y\}} \prod_{z:\mathsf{proj}_1(c_1)=x} \prod_{w:\mathsf{transport}(B,z,\mathsf{proj}_2(c_1))=y} g'(f'(z, w)) = \mathsf{pair}(z, w)$$

を帰納法により $p'(\mathsf{refl}, \mathsf{refl}) \equiv \mathsf{refl}$ と定義する。 ■

[001W] **3.2*3 補題**: i を階数、$A : \mathcal{U}(i)$ を型、$B, C : A \to \mathcal{U}(i)$ を型の族、$r : \prod_{x:A} \mathsf{Retract}(B(x), C(x))$ を要素とすると、$\mathsf{Retract}(\sum_{x:A} B(x), \sum_{x:A} C(x))$ の要素を構成できる。

証明: 仮定 r から関数 $f : \prod_{\{x:A\}} B(x) \to C(x)$ と $g : \prod_{\{x:A\}} C(x) \to B(x)$ と同一視 $p : \prod_{x:A} \prod_{y:B(x)} g(f(y)) = y$ を得る。関数 $F : (\sum_{x:A} B(x)) \to (\sum_{x:A} C(x))$ を $\lambda z.\mathsf{pair}(\mathsf{proj}_1(z), f(\mathsf{proj}_2(z)))$ と定義する。関数 $G : (\sum_{x:A} C(x)) \to (\sum_{x:A} B(x))$ も同様に g を使って定義される。同一視 $P : \prod_{z:\sum_{x:A} B(x)} G(F(z)) = z$ を定義するために、$z : \sum_{x:A} B(x)$ を仮定する。構成から $\mathsf{proj}_1(G(F(z))) \equiv \mathsf{proj}_1(z)$ であり、$p(\mathsf{proj}_1(z), \mathsf{proj}_2(z)) : \mathsf{proj}_2(G(F(z))) = \mathsf{proj}_2(z)$ を得る。補題 3.2*2 を使って、

$\mathsf{pair}(\mathsf{refl}, p(\mathsf{proj}_1(z), \mathsf{proj}_2(z)))$ から $G(F(z)) = z$ の要素を構成できる。 ■

3.2*4 定理 (同一視型の基本定理): i を階数、$A : \mathcal{U}(i)$ を型、$a : A$ を要素、$B : A \to \mathcal{U}(i)$ を型の族、$b : B(a)$ を要素とする。次の型は論理的に同値である。 [001S]

1. $\prod_{x:A} \mathsf{IsEquiv}(\lambda(p : a = x).\mathsf{transport}(B, p, b))$
2. $\prod_{x:A} (a = x) \simeq B(x)$
3. $\prod_{x:A} \mathsf{Retract}(B(x), a = x)$
4. $\mathsf{IsContr}(\sum_{x:A} B(x))$

証明: $\lambda(p : a = x).\mathsf{transport}(B, p, b)$ の型が $a = x \to B(x)$ であることから、1 から 2 は $\simeq$ の定義から自明である。

2 から 3 は補題 3.2*1 による。

3 から 4 を示す。3 を仮定すると、補題 3.2*3 から $\mathsf{Retract}(\sum_{x:A} B(x), \sum_{x:A} a = x)$ の要素を得る。すると、命題 3.1*4 と命題 3.1*6 より $\sum_{x:A} B(x)$ は可縮である。

最後に 4 から 1 を示す。4 を仮定し、$x : A$ と $y : B(x)$ を仮定する。$\mathsf{Fiber}(\lambda p.\mathsf{transport}(B, p, b), y)$ が可縮であることを示す。Fiber の定義より、$\sum_{p:a=x} \mathsf{transport}(B, p, b) = y$ が可縮であることを示せばよい。補題 3.2*2 と命題 3.1*6 から、$\mathsf{pair}(a, b) = \mathsf{pair}(x, y)$ が可縮であることを示せばよいが、これは仮定と命題 3.1*3 から従う。 ■

副産物として、対の同一視型の特徴付けは既に得られている。

3.2*5 系: i を階数、$A : \mathcal{U}(i)$ を型、$B : A \to \mathcal{U}(i)$ を型の族、$c_1, c_2 : \sum_{x:A} B(x)$ を要素とすると、同値 [002B]

$$(c_1 = c_2) \simeq (\textstyle\sum_{z:\mathsf{proj}_1(c_1)=\mathsf{proj}_1(c_2)} \mathsf{transport}(B, z, \mathsf{proj}_2(c_1)) = \mathsf{proj}_2(c_2))$$

を構成できる。

証明: 定理 3.2*4 と補題 3.2*2 による。 ■

定理 3.2*4 を適用する際に便利な補題を用意する。

3.2*6 補題: i を階数、$A : \mathcal{U}(i)$ を型、$B : A \to \mathcal{U}(i)$ を型の族、$a : A$ を要素とする。A が可縮ならば $(\sum_{x:A} B(x)) \mathrel{◁▷} B(a)$ の要素を構成できる。 [0024]

証明: A が可縮であると仮定する。命題 3.1*3 より、関数 $p : \prod_{x:A} x = a$ を得る。関数 $f : (\sum_{x:A} B(x)) \to B(a)$ を $\lambda z.\mathsf{transport}(B, p(\mathsf{proj}_1(z)), \mathsf{proj}_2(z))$ と定義し、関数 $g : B(a) \to (\sum_{x:A} B(x))$ を $\lambda y.\mathsf{pair}(a, y)$ と定義する。定義より、任意の $z : \sum_{x:A} B(x)$ に対して、$p(\mathsf{proj}_1(z)) : \mathsf{proj}_1(z) = \mathsf{proj}_1(g(f(z)))$ と $\mathsf{refl} : \mathsf{transport}(B, p(\mathsf{proj}_1(z)), \mathsf{proj}_2(z)) = \mathsf{proj}_2(g(f(z)))$ を得るので、補題 3.2*2 より同一視 $q : \prod_z g(f(z)) = z$ を得る。また、命題 3.1*3 より同一視 $r : p(a) = \mathsf{refl}$ も得られるので、同一視 $\lambda y.\mathsf{ap}(\lambda w.\mathsf{transport}(B, w, y), r) : \prod_{y:B(a)} f(g(y)) = y$ を得る。 ■

3.2*7 補題: i を階数、$A : \mathcal{U}(i)$ を型、$B : A \to \mathcal{U}(i)$ を型の族、$C : \prod_{x:A} B(x) \to \mathcal{U}(i)$ を型の族、$a : A$ と $b : B(a)$ を要素とする。$\sum_{x:A} B(x)$ が可縮ならば $(\sum_{x:A} \sum_{y:B(x)} C(x, y)) \mathrel{◁▷} C(a, b)$ の要素を構成できる。 [0025]

証明: 補題 3.2*6 からすぐに従う。 ■

[002C] **3.2*8 例**: $a_1, a_2 : \mathbf{1}$ を要素とする。同値 $(a_1 = a_2) \simeq \mathbf{1}$ を構成しよう。定理 3.2*4 を適用する。$B : \mathbf{1} \to \mathcal{U}(0)$ を $\lambda x.\mathbf{1}$ と定義する。要素 $\star : B(a_1)$ を得る。例 3.1*2 と補題 3.2*6 により、$\sum_{x:\mathbf{1}} B(x)$ は $B(a_1)$ のレトラクトである。再び例 3.1*2 により $B(a_1)$ は可縮なので、命題 3.1*6 により $\sum_{x:\mathbf{1}} B(x)$ は可縮である。

[002D] **3.2*9 例**: i を階数、$A, B : \mathcal{U}(i)$ を型、$c_1, c_2 : A \times B$ を要素とする。同値 $(c_1 = c_2) \simeq (\mathsf{proj}_1(c_1) = \mathsf{proj}_1(c_2)) \times (\mathsf{proj}_2(c_1) = \mathsf{proj}_2(c_2))$ を構成しよう。定理 3.2*4 を適用する。$E : A \times B \to \mathcal{U}(i)$ を $\lambda z.(\mathsf{proj}_1(c_1) = \mathsf{proj}_1(z)) \times (\mathsf{proj}_2(c_1) = \mathsf{proj}_2(z))$ と定義する。要素 $\mathsf{pair}(\mathsf{refl}, \mathsf{refl}) : E(c_1)$ を得る。レトラクトの列

$$\begin{aligned} & \textstyle\sum_{z:A\times B} E(z) \\ \lhd \quad & \{\text{並び換え}\} \\ & \textstyle\sum_{x:A} \sum_{p:\mathsf{proj}_1(c_1)=x} \sum_{y:B} \mathsf{proj}_2(c_1) = y \\ \lhd \quad & \{\text{命題 3.1*4 と補題 3.2*7}\} \\ & \textstyle\sum_{y:B} \mathsf{proj}_2(c_1) = y \end{aligned}$$

を得て、最後の型は命題 3.1*4 により可縮なので、命題 3.1*6 より $\sum_{z:A\times B} E(z)$ も可縮である。

[000X] 3.3 一価性

同値 $A \simeq B$ は型 $A, B : \mathcal{U}(i)$ の「正しい」同一視のしかたと考えられる。つまり、同値 $(A = B) \simeq (A \simeq B)$ が期待される。しかし、関数 $A \simeq B \to A = B$ は宇宙、関数型、対型、同一視型の規則からは導出できない。実際、型を従来の意味での集合と解釈するモデルを考えると、$A \simeq B$ は A から B への全単射のなす集合と解釈される一方、$A = B$ は A と B が等しい時に限り (一つだけ) 要素を持つような集合と解釈される。異なる集合の間にも全単射は存在する場合があり、その時には関数 $A \simeq B \to A = B$ は存在しない。

一価性公理は同値 $(A = B) \simeq (A \simeq B)$ を導出する公理である。先に説明したように、この同値は型理論の集合論的解釈とは相反するものである。一価性公理の下では、型は空間のホモトピー型のように振る舞う。その意味で、一価性公理は型理論をホモトピー論的なものに強制する公理と言える。

[0026] **3.3*1 補題**: i を階数、$A : \mathcal{U}(i)$ を型とすると、関数 $\mathsf{id} : A \to A$ は同値である。

証明: $a : A$ を要素とする。$\mathsf{Fiber}(\mathsf{id}, a)$ の定義から、$\sum_{x:A} x = a$ が可縮であることを示せばよい。例 2.4.1*2 より $x = a$ は $a = x$ のレトラクトなので、定理 3.2*4 より $\sum_{x:A} x = a$ は可縮である。 ■

[000Y] **3.3*2 定義**: i を階層とする。宇宙 $\mathcal{U}(i)$ が**一価性** *(univalence)* を満たすとは、$\prod_{X:\mathcal{U}(i)} \mathsf{IsContr}(\sum_{Y:\mathcal{U}(i)} X \simeq Y)$ の要素があることである。

補題 3.3*1 から定理 3.2*4 を適用できて、$\mathcal{U}(i)$ が一価性を満たす時、任意の型 $A, B : \mathcal{U}(i)$ に対して同値 $(A = B) \simeq (A \simeq B)$ を構成できる。

3.3*3 公理 (一価性公理): **一価性公理** *(univalence axiom)* は任意の階数の宇宙が一価性を満たすことを要請する。 [000Z]

ここで公理という言葉を使ったが、規則と公理に本質的な違いは無い。つまり、公理 3.3*3 は任意の階数 i に対して要素 $\mathsf{ua}(i) : \prod_{X:\mathcal{U}(i)} \mathsf{IsContr}(\sum_{Y:\mathcal{U}(i)} X \simeq Y)$ を構成できるという規則だとも言える。本書では、次のように感覚的に使い分ける。

- 規則は新しい型の種類を導入する際に使われ、その型が何なのかを説明するものである。例えば規則 2.2*1 は関数型を説明するための規則だが、$\mathsf{ua}(i)$ は複合的な型の要素でしかなく、新しい型を説明するものではない。
- 規則は導入した後はいつでも暗黙的に使うが、公理を使用する際は明示する。これは公理がどのように使われるかを強調するためというのもあるが、相反する公理を導入する可能性も考えると公理を大域的に仮定するのは良くないというのがある。あるいは、規則は型理論の基盤にあたるものだが公理は選択的な機能だと言ってもよい。

ちなみに、公理はある型が可縮であるという形で述べられることが多い。よって、$\mathsf{ua}(i)$ のような名前を付ける必要はない。

3.4 関数外延性 [001Y]

関数 $f, g : \prod_{x:A} B(x)$ の「正しい」同一視のしかたは任意の $x : A$ に対して $f(x)$ と $g(x)$ を同一視することである。つまり、同値 $(f = g) \simeq (\prod_{x:A} f(x) = g(x))$ が期待される。しかし、関数 $(\prod_{x:A} f(x) = g(x)) \to f = g$ は関数型の規則と同一視型の規則からは構成できないことが知られている [Streicher--1993-0000]。そのため、この同値を得るためには何らかの公理が必要である。

3.4*1 定義: i を階数、$A : \mathcal{U}(i)$ を型、$B : A \to \mathcal{U}(i)$ を型の族とする。関数型 $\prod_{x:A} B(x)$ が **関数外延性** *(function extensionality)* を満たすとは、$\prod_{f:\prod_{x:A} B(x)} \mathsf{IsContr}(\sum_{g:\prod_{x:A} B(x)} \prod_{x:A} f(x) = g(x))$ の要素があることである。 [001Z]

要素 $\lambda x.\mathsf{refl}\{f(x)\} : \prod_{x:A} f(x) = f(x)$ があるので定理 3.2*4 を適用できて、$\prod_{x:A} B(x)$ が関数外延性を持つ時、任意の関数 $f : \prod_{x:A} B(x)$ と $g : \prod_{x:A} B(x)$ に対して同値 $(f = g) \simeq (\prod_{x:A} f(x) = g(x))$ を得る。

3.4*2 公理 (関数外延性公理): **関数外延性公理** *(function extensionality axiom)* はすべての関数型が関数外延性を満たすことを要請する。 [0020]

3.4*3 命題: i を階数とする。次は論理的に同値である。 [0029]

1 $\mathcal{U}(i)$ のすべての関数型が関数外延性を満たす。
2 任意の型 $A : \mathcal{U}(i)$ と型の族 $B : A \to \mathcal{U}(i)$ に対して、$(\prod_{x:A} \mathsf{IsContr}(B(x))) \to \mathsf{IsContr}(\prod_{x:A} B(x))$ の要素がある。

証明: 1 から 2 を示す。$c : \prod_{x:A} \mathsf{IsContr}(B(x))$ を仮定する。仮定 c より、関数 $f : \prod_{x:A} B(x)$ と同一視 $p : \prod_{x:A} \prod_{y:B(x)} f(x) = y$ を得る。同一視 $q : \prod_{g:\prod_{x:A} B(x)} f = g$

を構成すればよい。任意の $g : \prod_{x:A} B(x)$ に対して、$\lambda x.p(x, g(x)) : \prod_{x:A} f(x) = g(x)$ を得るが、関数外延性と定理 3.2*4 により $f = g$ の要素を得る。

2 から 1 を示す。$A : \mathcal{U}(i)$ を型、$B : A \to \mathcal{U}(i)$ を型の族、$f : \prod_{x:A} B(x)$ を関数とする。例 2.3*18 より、$\sum_{g:\prod_{x:A} B(x)} \prod_{x:A} f(x) = g(x)$ は $\prod_{x:A} \sum_{y:B(x)} f(x) = y$ のレトラクトであるが、後者は仮定と命題 3.1*4 により可縮である。よって、命題 3.1*6 から前者も可縮である。 ■

[0021] ### 3.4.1 一価性から関数外延性を導く

実は、一価性公理から関数外延性公理が導出される (定理 3.4.1*5)。

[0079] **3.4.1*1 定義**: i を階数、$A : \mathcal{U}(i)$ を型、$B : A \to \mathcal{U}(i)$ を型の族とする。関数 $p : (A \to (\sum_{x:A} B(x))) \to (A \to A)$ を $\lambda f.\lambda x.\mathsf{proj}_1(f(x))$ とし、型 $\mathsf{Sect}(A, B) : \mathcal{U}(i)$ を $\mathsf{Fiber}(p, \mathsf{id})$ と定義する。

[007A] **3.4.1*2 補題**: i を階数、$A : \mathcal{U}(i)$ を型、$B : A \to \mathcal{U}(i)$ を型の族とすると、$\prod_{x:A} B(x)$ は $\mathsf{Sect}(A, B)$ のレトラクトである。

証明: 関数 $F : (\prod_{x:A} B(x)) \to \mathsf{Sect}(A, B)$ を

$$\lambda f.\mathsf{record}\{\mathsf{elem} \equiv \lambda x.\mathsf{pair}(x, f(x)), \mathsf{id} \equiv \mathsf{refl}\}$$

と定義し、関数 $G : \mathsf{Sect}(A, B) \to (\prod_{x:A} B(x))$ を

$$\lambda z.\lambda x.\mathsf{transport}(\lambda g.B(g(x)), z.\mathsf{id}, \mathsf{proj}_2(z.\mathsf{elem}(x)))$$

と定義すると、$G \circ F \equiv \mathsf{id}$ である。 ■

[007B] **3.4.1*3 補題**: i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数とする。$\mathcal{U}(i)$ が一価性を満たし、f が同値ならば、任意の型 $X : \mathcal{U}(i)$ に対して関数 $\lambda g.f \circ g : (X \to A) \to (X \to B)$ は同値である。

証明: 一価性より、f が恒等関数の場合を示せば十分であるが、$\lambda g.\mathsf{id} \circ g \equiv \lambda g.g$ なのでこれは同値である。 ■

[007C] **3.4.1*4 補題**: i を階数、$A : \mathcal{U}(i)$ を型、$B : A \to \mathcal{U}(i)$ を型の族とする。$\prod_{x:A} \mathsf{IsContr}(B(x))$ の要素があるならば、関数 $\lambda z.\mathsf{proj}_1(z) : (\sum_{x:A} B(x)) \to A$ は同値である。

証明: 任意の $a : A$ に対して、レトラクト

$$\begin{aligned} & \mathsf{Fiber}(\lambda z.\mathsf{proj}_1(z), a) \\ \lhd\ & \{\text{並び換え}\} \\ & \textstyle\sum_{x:A} \sum_{p:x=a} B(x) \\ \lhd\ & \{\text{補題 3.3*1}\} \\ & B(a) \end{aligned}$$

を得て、最後の型は仮定より可縮である。 ■

3.4.1*5 定理: i を階数とする。$\mathcal{U}(i)$ が一価性を満たすならば、$\mathcal{U}(i)$ のすべての関数型は関数外延性を満たす。 [007D]

証明: 命題 3.4*3 を適用する。$A : \mathcal{U}(i)$ を型、$B : A \to \mathcal{U}(i)$ を型の族とし、各 $B(x)$ は可縮であると仮定する。補題 3.4.1*4 より、関数 $\lambda z.\mathsf{proj}_1(z) : (\sum_{x:A} B(x)) \to A$ は同値である。これと補題 3.4.1*3 より、$\mathsf{Sect}(A, B)$ は可縮である。したがって、補題 3.4.1*2 と命題 3.1*6 より $\prod_{x:A} B(x)$ は可縮である。 ■

3.5 構造同一原理 [0022]

一価性 (と関数外延性) の帰結として、**構造同一原理** *(structure identity principle)* をいくつか例示する。構造とは厳密に定義はしないが、なんらかの数学的構造を表すレコード型のこととする。構造同一原理は、構造の同一視型は構造を保つ同値の型と同値であることを主張する。

ある構造 A と要素 $a : A$ に対し、同一視型 $\lambda x.(a = x)$ を特徴付けるには、候補 $E : A \to \mathcal{U}(i)$ を見つけて $\sum_{x:A} E(x)$ が可縮であることを示す。要素 $e : E(a)$ を自然に見つけられる場合がほとんどで、定理 3.2*4 から $(a = x) \simeq E(x)$ を得る。基本的な戦略は目的の型から始めて、既に可縮と分かっている型になるまでレトラクトの列を作ることである (実際には同値の列を作れる場合が多い)。すると命題 3.1*6 により目的の型が可縮であることが従う。レトラクトの列を作る際には補題 3.2*7 が便利で、大きなレコード型の一部に同一視型の特徴付けを知っている部分があれば、そのレコード型はより単純な型のレトラクトであることを示せる。また、レコード型の要素を並び換える関数はレトラクションであることも便利である。

3.5*1 例: i を階数とする。$\mathcal{U}_\bullet(i)$ (例 2.3*9) の同一視型を特徴付ける。$A : \mathcal{U}_\bullet(i)$ に対し、$E : \mathcal{U}_\bullet(i) \to \mathcal{U}(i)$ を $\lambda Z.(\sum_{e:A.\mathsf{Carrier} \simeq Z.\mathsf{Carrier}} e(A.\mathsf{point}) = Z.\mathsf{point})$ と定義する。要素 $\mathsf{pair}(\mathsf{id}, \mathsf{refl}) : E(A)$ を得る。レトラクトの列 [0023]

$$
\begin{aligned}
& \textstyle\sum_{Z:\mathcal{U}_\bullet(i)} E(Z) \\
\triangleleft \quad & \{\text{並び換え}\} \\
& \textstyle\sum_{X:\mathcal{U}(i)} \sum_{e:A.\mathsf{Carrier} \simeq X} \sum_{x:X} e(A.\mathsf{point}) = x \\
\triangleleft \quad & \{\text{一価性}\} \\
& \textstyle\sum_{x:A.\mathsf{Carrier}} A.\mathsf{point} = x
\end{aligned}
$$

を得て、最後の型は命題 3.1*4 より可縮なので、$\sum_{Z:\mathcal{U}_\bullet(i)} E(Z)$ は可縮である。

3.5*2 例: i を階数とする。$\mathsf{Magma}(i)$ (例 2.3*10) の同一視型を特徴付ける。$A : \mathsf{Magma}(i)$ に対し、$E : \mathsf{Magma}(i) \to \mathcal{U}(i)$ を [0028]

$$\lambda Z.(\textstyle\sum_{e:A.\mathsf{Carrier} \simeq Z.\mathsf{Carrier}} \prod_{x_1,x_2:A.\mathsf{Carrier}} e(A.\mathsf{op}(x_1, x_2)) = Z.\mathsf{op}(e(x_1), e(x_2)))$$

と定義する。要素 $\mathsf{pair}(\mathsf{id}, \lambda(x_1, x_2).\mathsf{refl}) : E(A)$ を得る。レトラクトの列

$$
\begin{aligned}
& \textstyle\sum_{Z:\mathsf{Magma}(i)} E(Z) \\
\triangleleft \quad & \{\text{並び換え}\} \\
& \textstyle\sum_{X:\mathcal{U}(i)} \sum_{e:A.\mathsf{Carrier} \simeq X} \sum_{f:X \to X \to X} \prod_{x_1,x_2:A.\mathsf{Carrier}} e(A.\mathsf{op}(x_1, x_2)) = f(e(x_1), e(x_2))
\end{aligned}
$$

$\triangleleft$ {一価性}
$\sum_{f:A.\mathsf{Carrier}\to A.\mathsf{Carrier}\to A.\mathsf{Carrier}} \prod_{x_1,x_2:A.\mathsf{Carrier}} A.\mathsf{op}(x_1,x_2) = f(x_1,x_2)$

を得て、最後の型は関数外延性により可縮なので、$\sum_{Z:\mathsf{Magma}(i)} E(Z)$ は可縮である。

[002A] **3.5*3 例**: i を階数とする。$\mathsf{ReflGraph}(i)$ (例 2.3*11) の同一視型を特徴付ける。$A : \mathsf{ReflGraph}(i)$ と $Z : \mathsf{ReflGraph}(i)$ に対し、$E(Z) : \mathcal{U}(i)$ を次のレコード型と定義する。

- $v : A.\mathsf{Vertex} \simeq Z.\mathsf{Vertex}$
- $e : \prod_{\{x_1,x_2:A.\mathsf{Vertex}\}} A.\mathsf{Edge}(x_1,x_2) \simeq Z.\mathsf{Edge}(v(x_1),v(x_2))$
- $r : \prod_{x:A.\mathsf{Vertex}} e(A.\mathsf{refl}(x)) = Z.\mathsf{refl}(v(x))$

要素 $\mathsf{record}\{\!|v \equiv \mathsf{id}, e \equiv \lambda(x_1,x_2).\mathsf{id}, r \equiv \lambda x.\mathsf{refl}|\!\} : E(A)$ を得る。レトラクトの列

$\sum_{Z:\mathsf{ReflGraph}(i)} E(Z)$
$\triangleleft$ {並び換え}
$\sum_{X:\mathcal{U}(i)} \sum_{v:A.\mathsf{Vertex}\simeq X} \sum_Y \sum_e \sum_z \prod_x —$
$\triangleleft$ {一価性}
$\sum_{Y:A.\mathsf{Vertex}\to A.\mathsf{Vertex}\to\mathcal{U}(i)} \sum_{e:\prod_{x_1,x_2:A.\mathsf{Vertex}} A.\mathsf{Edge}(x_1,x_2)\simeq Y(x_1,x_2)} \sum_z \prod_x —$
$\triangleleft$ {関数外延性と一価性}
$\sum_{z:\prod_{x:A.\mathsf{Vertex}} A.\mathsf{Edge}(x,x)} \prod_{x:A.\mathsf{Vertex}} A.\mathsf{refl}(x) = z(x)$

を得て、最後の型は関数外延性により可縮なので、$\sum_{Z:\mathsf{ReflGraph}(i)} E(Z)$ は可縮である。

数学でよく登場する構造といえば群や環などであるが、これらについての構造同一原理を正しく述べるには 4 章で導入する n **型**の概念を用いる。4.6 節でさらなる構造同一原理を見る。

[000U] ## 3.6 同値

同値の定義 (定義 3.1*8) が妥当なものであることを見る。同値の概念にふさわしい性質として次を示す。

- 恒等関数は同値である (補題 3.3*1)
- 同値の概念は**ホモトピー不変**である (命題 3.6*6)
- **六分の二性** *(2-out-of-6 property)*: 合成可能な関数 f,g,h に対して、$g\circ f$ と $h\circ g$ が同値ならば $f,g,h,h\circ g\circ f$ も同値である (命題 3.6*7)

さらに、関数全体のうちの同値のなすクラスはこれらの性質を満たすものの中で最小であることを示す。つまり、任意の同値はこれらの性質のみを使って得られる (命題 3.6*8)。また、もう一つ重要な事実として、(関数外延性の下で)$\mathsf{IsEquiv}(f)$ は**命題**であるということがある (系 4.1*9)。命題は後に導入する概念なので今は説明しないが、関数が同値であることをどのように証明したかは気にしなくてよいということが分かる。

[002L] **3.6*1 補題**: i を階数、$A,B,C : \mathcal{U}(i)$ を型、$f : A\to B$ と $g : B\to C$ を型、$c : C$ を要素とすると、$(\sum_{y:\mathsf{Fiber}(g,c)} \mathsf{Fiber}(f, y.\mathsf{elem})) \diamond\!\!\!\triangleright \mathsf{Fiber}(g\circ f, c)$ の要素を構成できる。

証明: 次のようにわかる。

$$
\begin{aligned}
& \textstyle\sum_{y:\mathsf{Fiber}(g,c)} \mathsf{Fiber}(f, y.\mathsf{elem}) \\
\lhd\rhd \quad & \{\text{並び換え}\} \\
& \textstyle\sum_{x:A} \sum_{y:B} \sum_{p:f(x)=y} g(y) = c \\
\lhd\rhd \quad & \{\text{補題 3.2*3 と補題 3.2*7 と命題 3.1*4}\} \\
& \textstyle\sum_{x:A} g(f(x)) = c
\end{aligned}
$$

■

3.6*2 補題: i を階数、$A, B, C : \mathcal{U}(i)$ を型、$f : A \to B$ と $g : B \to C$ を関数とする。f と g と $g \circ f$ のうちいずれか二つが同値ならば残りの一つも同値である。つまり、次の型の要素を構成できる。 [002E]

- $\mathsf{IsEquiv}(f) \to \mathsf{IsEquiv}(g) \to \mathsf{IsEquiv}(g \circ f)$
- $\mathsf{IsEquiv}(f) \to \mathsf{IsEquiv}(g \circ f) \to \mathsf{IsEquiv}(g)$
- $\mathsf{IsEquiv}(g) \to \mathsf{IsEquiv}(g \circ f) \to \mathsf{IsEquiv}(f)$

証明: f が同値であると仮定すると、補題 3.6*1 と命題 3.1*6 から $\mathsf{IsEquiv}(g) \leftrightarrow \mathsf{IsEquiv}(g \circ f)$ が従う。

g と $g \circ f$ が同値であると仮定する。$b : B$ を仮定する。$r : \mathsf{Fiber}(g, g(b))$ を $\mathsf{record}\{\!|\mathsf{elem} \equiv b, \mathsf{id} \equiv \mathsf{refl}|\!\}$ と定義する。レトラクト

$$
\begin{aligned}
& \mathsf{Fiber}(g \circ f, g(b)) \\
\lhd\rhd \quad & \{\text{補題 3.6*1}\} \\
& \textstyle\sum_{y:\mathsf{Fiber}(g,g(b))} \mathsf{Fiber}(f, y.\mathsf{elem}) \\
\lhd\rhd \quad & \{g \text{ が同値}\} \\
& \mathsf{Fiber}(f, r.\mathsf{elem})
\end{aligned}
$$

を得て、$r.\mathsf{elem} \equiv b$ であることに注意すると、$g \circ f$ が同値なので $\mathsf{Fiber}(f, b)$ は可縮である。 ■

3.6*3 補題: i を階数、$A : \mathcal{U}(i)$ を型とする。関数 $\lambda z.\mathsf{proj}_1(z) : (\sum_{x_1,x_2:A} x_1 = x_2) \to A$ は同値である。 [002J]

証明: 任意の要素 $a : A$ に対して、レトラクトの列

$$
\begin{aligned}
& \mathsf{Fiber}(\lambda z.\mathsf{proj}_1(z), a) \\
\lhd \quad & \{\text{並び換え}\} \\
& \textstyle\sum_{x_1:A} \sum_{p:x_1=a} \sum_{x_2:A} x_1 = x_2 \\
\lhd \quad & \{\text{補題 3.3*1}\} \\
& \textstyle\sum_{x_2:A} a = x_2
\end{aligned}
$$

を得て、最後の型は命題 3.1*4 により可縮である。 ■

3.6*4 補題: i を階数、$A : \mathcal{U}(i)$ を型とする。関数 $\lambda x.\mathsf{pair}(x, \mathsf{pair}(x, \mathsf{refl})) : A \to (\sum_{x_1,x_2:A} x_1 = x_2)$ と $\lambda z.\mathsf{proj}_1(\mathsf{proj}_2(z)) : (\sum_{x_1,x_2:A} x_1 = x_2) \to A$ は同値である。 [002K]

証明: $f \equiv \lambda z.\mathsf{proj}_1(z)$ と $g \equiv \lambda x.\mathsf{pair}(x, \mathsf{pair}(x, \mathsf{refl}))$ と $h \equiv \lambda z.\mathsf{proj}_1(\mathsf{proj}_2(z))$ と定義す

ると、$f \circ g \equiv \mathsf{id}$ かつ $h \circ g \equiv \mathsf{id}$ である。補題 3.6*3 より f は同値であるから、補題 3.6*2 と補題 3.3*1 から g も同値であると分かる。すると、再び補題 3.6*2 と補題 3.3*1 から h も同値であると分かる。 ■

[002I] **3.6*5 定義**: i を階数、$A : \mathcal{U}(i)$ を型、$B : A \to \mathcal{U}(i)$ を型の族、$f, g : \prod_{x:A} B(x)$ を関数とする。型 $f \sim g : \mathcal{U}(i)$ を $\prod_{x:A} f(x) = g(x)$ と定義する。$f \sim g$ の要素を f と g の間の **ホモトピー** *(homotopy)* と呼ぶ。

[002G] **3.6*6 命題**: i を階数、$A, B : \mathcal{U}(i)$ を型、$f, g : A \to B$ を関数、$p : f \sim g$ をホモトピーとする。f が同値ならば g も同値である。

証明: $q : A \to (\sum_{y_1, y_2 : B} y_1 = y_2)$ を $\lambda x.\mathsf{pair}(f(x), \mathsf{pair}(g(x), p(x)))$ と定義すると、$(\lambda z.\mathsf{proj}_1(z)) \circ q \equiv f$ かつ $(\lambda z.\mathsf{proj}_1(\mathsf{proj}_2(z))) \circ q \equiv g$ である。補題 3.6*2 と補題 3.6*3 と仮定から q が同値であると分かり、すると補題 3.6*2 と補題 3.6*4 から g が同値であると分かる。 ■

[002F] **3.6*7 命題**: i を階数、$A, B, C, D : \mathcal{U}(i)$ を型、$f : A \to B$ と $g : B \to C$ と $h : C \to D$ を関数とする。$g \circ f$ と $h \circ g$ が同値ならば f と g と h と $h \circ g \circ f$ も同値である。

証明: 任意の要素 $d : D$ に対して、レトラクト

$$
\begin{aligned}
& \mathsf{Fiber}(h \circ g \circ f, d) \\
\lhd\ & \{\text{補題 } 3.6^*1\} \\
& \textstyle\sum_{z:\mathsf{Fiber}(h,d)} \sum_{y:\mathsf{Fiber}(g,z.\mathsf{elem})} \mathsf{Fiber}(f, y.\mathsf{elem}) \\
\lhd\ & \{\text{演習 } 2.3^*20\} \\
& \textstyle\sum_{z:\mathsf{Fiber}(h,d)} \sum_{y':\mathsf{Fiber}(g,z.\mathsf{elem})} \sum_{y:\mathsf{Fiber}(g,z.\mathsf{elem})} \mathsf{Fiber}(f, y.\mathsf{elem}) \\
\lhd\ & \{h \circ g \text{ が同値}\} \\
& \textstyle\sum_{y:\mathsf{Fiber}(g,\_)} \mathsf{Fiber}(f, y.\mathsf{elem})
\end{aligned}
$$

を得て、最後の型は $g \circ f$ が同値なので可縮である。よって、$h \circ g \circ f$ は同値である。すると補題 3.6*2 より残りの関数もすべて同値である。 ■

[002H] **3.6*8 命題**: i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数とする。f が同値ならば、関数 $g, h : B \to A$ とホモトピー $p : f \circ g \sim \mathsf{id}$ と $q : h \circ f \sim \mathsf{id}$ を構成できる。

証明: f が同値であると仮定する。関数 $G : \prod_{y:B} \mathsf{Fiber}(f, y)$ と同一視 $P : \prod_{y:B} \prod_{z:\mathsf{Fiber}(f,y)} G(y) = z$ を得る。$g \equiv \lambda y.(G(y)).\mathsf{elem}$ と $p \equiv \lambda y.(G(y)).\mathsf{id}$ と定義する。$r : \prod_{x:A} \mathsf{Fiber}(f, f(x))$ を $r \equiv \lambda x.\mathsf{record}\{\!|\mathsf{elem} \equiv x, \mathsf{id} \equiv \mathsf{refl}|\!\}$ と定義する。$h \equiv g$ と定義し、$q \equiv \lambda x.\mathsf{ap}(\lambda z.z.\mathsf{id}, P(f(x), r(x)))$ と定義すればよい。 ■

[004I] 3.6.1 他の同値の概念

定義 3.1*8 の他にも同値の概念の定義はある。

[004J] **3.6.1*1 定義**: i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数とする。

- 型 $\mathsf{LInv}(f) : \mathcal{U}(i)$ を $\mathsf{Record}\{\!|\mathsf{inv} : B \to A, \mathsf{is\text{-}linv} : \mathsf{inv} \circ f \sim \mathsf{id}|\!\}$ と定義する。

- 型 RInv$(f) : \mathcal{U}(i)$ を Record$\{\!|$inv : $B \to A$, is-rinv : $f \circ \mathsf{inv} \sim \mathsf{id}|\!\}$ と定義する。
- 型 IsBiinv$(f) : \mathcal{U}(i)$ を Record$\{\!|$linv : LInv(f), rinv : RInv$(f)|\!\}$ と定義する。

IsBiinv(f) の要素がある時、f は**両側可逆** *(biinvertible)* であると言う。

3.6.1*2 命題: i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数とする。型 IsEquiv(f) と IsBiinv(f) は論理的に同値である。 [004K]

証明: 補題 3.3*1 と命題 3.6*6 と命題 3.6*7 と命題 3.6*8 から従う。 ■

定義 3.6.1*1 の利点は比較的低級な概念で定義できる点である。この定義で使われている概念は恒等関数、合成、ホモトピーだけである。これらの概念は一般の高次圏で意味をなすものであり、実際、高次圏での同値は両側可逆性で定義できる。一方、ファイバーや可縮性には型理論の言語が本質的に使われている。ただ、定義 3.6.1*1 から直接取り出せる情報はそれほど有益でなく、逆関数の取り方も二種類あるので使い勝手が良いとは言えない。

3.6.1*3 定義: i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数とする。型 IsHAE$(f) : \mathcal{U}(i)$ を次のレコード型と定義する。 [004L]

- inv : $B \to A$
- unit : inv $\circ f \sim$ id
- counit : $f \circ$ inv $\sim$ id
- coh : $\prod_{x:A} f(\mathsf{unit}(x)) = \mathsf{counit}(f(x))$

IsHAE(f) の要素がある時、f は**半随伴同値** *(half adjoint equivalence)* であると言う。

3.6.1*4 演習: i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数とする。型 IsEquiv(f) と IsHAE(f) は論理的に同値であることを示せ。(ヒント: 命題 3.6*8 の証明を拡張すれば IsEquiv$(f) \to$ IsHAE(f) を示せる。) [004M]

定義 3.6.1*3 は定義 3.6.1*1 と比べて有益な情報が多い。その分、その定義には一つ高次の同一視 (coh) を使う。

次の定義 3.6.1*5 も同値の概念を定義しているように見えるかもしれないが、これは正しい同値の定義**ではない**。

3.6.1*5 定義: i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数とする。型 QInv$(f) : \mathcal{U}(i)$ を次のレコード型と定義する。 [004T]

- inv : $B \to A$
- unit : inv $\circ f \sim$ id
- counit : $f \circ$ inv $\sim$ id

3.6.1*6 命題: i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数とする。型 IsEquiv(f) と QInv(f) は論理的に同値である。 [004U]

証明: IsHAE$(f) \to$ QInv(f) と QInv$(f) \to$ IsBiinv(f) は容易に示せるので、命題 3.6.1*2 と演習 3.6.1*4 を使う。 ■

命題 3.6.1*6 により、関数 f が同値であることを示す目的では $\mathsf{QInv}(f)$ を使って何も問題はない。$\mathsf{QInv}(f)$ が他に挙げた正しい同値の概念と決定的に違うのは、同一でない二つの $\mathsf{QInv}(f)$ の要素がありうる点である。後に示すが、(関数外延性の下で)$\mathsf{IsEquiv}(f)$ と $\mathsf{IsBiinv}(f)$ と $\mathsf{IsHAE}(f)$ はいずれも任意の二つの要素が同一視される (系 4.1*9 と命題 4.1.1*3 と命題 4.1.1*5)。$\mathsf{QInv}(f)$ についてはこの性質は示せないどころか、一価性公理の下で $\mathsf{QInv}(f)$ が同一でない二つの要素を持つような f を構成できることが知られている。

4 n 型 [003W]

2.4.1 節で型は高次グルーポイドの構造を持つと説明したが、その豊富な構造をすべて把握するのは容易ではない。n 次元より上の構造が自明になっているような型は n **型**と呼ばれ、比較的解析が容易である。

次数 n は -2 から数えるのが都合がよい。

4*1 定義: [003X]

- 型 $\mathsf{TruncLevel} : \mathcal{U}(0)$ を $\mathbb{N} + (\mathbf{1} + \mathbf{1})$ と定義する。
- 要素 $-2 : \mathsf{TruncLevel}$ を $\mathsf{in}_2(\mathsf{in}_2(\star))$ と定義する。
- 要素 $-1 : \mathsf{TruncLevel}$ を $\mathsf{in}_2(\mathsf{in}_1(\star))$ と定義する。
- 要素 $n : \mathsf{TruncLevel}$ に対して、要素 $\mathsf{succ}(n) : \mathsf{TruncLevel}$ を $\mathsf{succ}(-2) \equiv -1$ と $\mathsf{succ}(-1) \equiv \mathsf{in}_1(0)$ と $\mathsf{succ}(\mathsf{in}_1(n)) \equiv \mathsf{in}_1(\mathsf{succ}(n))$ と定義する。

要素 $n : \mathbb{N}$ に対しては、in_1 を省略して n 自身を TruncLevel の要素とみなす。

TruncLevel は実質 $\mathbb{N}$ と同じであるが、0 の代わりに -2 から数えたものである。特に、次の**帰納法原理**を満たす: 型の族 $A : \mathsf{TruncLevel} \to \mathcal{U}(i)$ に対して、関数 $h : \prod_{x:\mathsf{TruncLevel}} A(x)$ を構成するためには、

- $a : A(-2)$
- $f : \prod_{x:\mathsf{TruncLevel}} A(x) \to A(\mathsf{succ}(x))$

を構成すれば十分である。

4*2 定義: i を階数、$A : \mathcal{U}(i)$ を型とする。$n : \mathsf{TruncLevel}$ に対して、型 $\mathsf{IsTrunc}(n, A) : \mathcal{U}(i)$ を次のように定義する。 [003Y]

- $\mathsf{IsTrunc}(-2, A) \equiv \mathsf{IsContr}(A)$
- $\mathsf{IsTrunc}(\mathsf{succ}(n), A) \equiv \prod_{x_1, x_2 : A} \mathsf{IsTrunc}(n, x_1 = x_2)$

$\mathsf{IsTrunc}(n, A)$ の要素がある時、A は n **型** *(n-type)* である、または n-truncated であると言う。

4*3 定義: i を階数、n : TruncLevel を要素とする。型 $\langle n \rangle\text{-}\mathsf{Type}(i)$: $\mathcal{U}(\mathsf{succ}(i))$ を $\mathsf{Record}\{\!|\mathsf{Type} : \mathcal{U}(i), \mathsf{is\text{-}trunc} : \mathsf{IsTrunc}(n, \mathsf{Type})|\!\}$ と定義する。 [0053]

-1 型は特別に**命題**と呼ばれ、4.1 節でより詳しく調べる。0 型は特別に**集合**と呼ばれ、4.2 節でより詳しく調べる。

n 型の一般的な性質をいくつか見る。まず、n 型はレトラクトで閉じる (命題 4*5)。

[0046] **4*4 補題**: i を階数、$A, B : \mathcal{U}(i)$ を型、$r : \mathsf{Retract}(A, B)$ を要素、$a_1, a_2 : A$ を要素とすると、$\mathsf{Retract}(a_1 = a_2, r.\mathsf{section}(a_1) = r.\mathsf{section}(a_2))$ の要素を構成できる。

証明: $f \equiv r.\mathsf{section}$ と $g \equiv r.\mathsf{retraction}$ と $p \equiv r.\mathsf{r\text{-}s}$ と定義する。$F : a_1 = a_2 \to f(a_1) = f(a_2)$ を $\mathsf{ap}(f)$ と定義する。$G : f(a_1) = f(a_2) \to a_1 = a_2$ を $\lambda q.(p(a_2) \circ \mathsf{ap}(g, q)) \circ (p(a_1))^{-1}$ と定義する。$\prod_{z:a_1=a_2} G(F(z)) = z$ を示すには、同一視型の帰納法により $G(F(\mathsf{refl}\{a_1\})) = \mathsf{refl}\{a_1\}$ を示せばよいが、$G(F(\mathsf{refl}\{a_1\})) \equiv p(a_1) \circ (p(a_1))^{-1}$ なので演習 2.4.1*5 を適用すればよい。 ■

[0045] **4*5 命題**: i を階数、$A, B : \mathcal{U}(i)$ を型、$r : \mathsf{Retract}(A, B)$ と $n : \mathsf{TruncLevel}$ を要素とする。B が n 型ならば、A も n 型である。

証明: n についての帰納法による。n が -2 の時は命題 3.1*6 による。

n について主張が成り立つと仮定し、$\mathsf{succ}(n)$ の場合を示す。$x_1, x_2 : A$ に対して、$\mathsf{IsTrunc}(n, x_1 = x_2)$ を示せばよいが、補題 4*4 と帰納法の仮定から直ちに従う。 ■

n 型はいくつかの型の構成で閉じる。

[0048] **4*6 命題**: 関数外延性を仮定する。i を階数、$A : \mathcal{U}(i)$ を型、$B : A \to \mathcal{U}(i)$ を型の族、$n : \mathsf{TruncLevel}$ を要素とする。$\prod_{x:A} \mathsf{IsTrunc}(n, B(x))$ の要素があるならば、$\prod_{x:A} B(x)$ は n 型である。

証明: n についての帰納法による。n が -2 の場合は命題 3.4*3 による。

n の場合に主張が成り立つと仮定し、$\mathsf{succ}(n)$ の場合を示す。$f : \prod_{x:A} B(x)$ と $g : \prod_{x:A} B(x)$ に対し、$f = g$ が n 型であることを示す。関数外延性より、同値 $(f = g) \simeq (\prod_{x:A} f(x) = g(x))$ を得る。各 $f(x) = g(x)$ は B についての仮定より n 型であるから、帰納法の仮定と命題 4*5 から $f = g$ は n 型である。 ■

[004X] **4*7 命題**: i を階数、$A : \mathcal{U}(i)$ を型、$B : A \to \mathcal{U}(i)$ を型の族、$n : \mathsf{TruncLevel}$ を要素とする。A が n 型で、任意の $x : A$ に対して $B(x)$ が n 型ならば、$\sum_{x:A} B(x)$ も n 型である。

証明: n についての帰納法による。n が -2 の場合は容易である。

n の場合に主張が成り立つと仮定し、$\mathsf{succ}(n)$ の場合を示す。$c_1, c_2 : \sum_{x:A} B(x)$ に対し、$c_1 = c_2$ が n 型であることを示す。系 3.2*5 より、同値

$$(c_1 = c_2) \simeq (\textstyle\sum_{z:\mathsf{proj}_1(c_1)=\mathsf{proj}_1(c_2)} \mathsf{transport}(B, z, \mathsf{proj}_2(c_1)) = \mathsf{proj}_2(c_2))$$

を得る。仮定より、$\mathsf{proj}_1(c_1) = \mathsf{proj}_1(c_2)$ と各 $\mathsf{transport}(B, z, \mathsf{proj}_2(c_1)) = \mathsf{proj}_2(c_2)$ は n 型である。よって、帰納法の仮定を適用すればよい。 ■

[004F] **4*8 命題**: i を階数、$A : \mathcal{U}(i)$ を型、$n : \mathsf{TruncLevel}$ を要素とする。A が n 型ならば、A は $\mathsf{succ}(n)$ 型である。

証明: n についての帰納法による。n が -2 の場合は命題 3.1*3 による。

n の場合に主張が成り立つと仮定して、$\mathsf{succ}(n)$ の場合を示す。A が $\mathsf{succ}(n)$ 型と仮定して、任意の $x_1, x_2 : A$ に対して $x_1 = x_2$ が $\mathsf{succ}(n)$ 型であることを示せばよいが、仮定と帰納法の過程から直ちに従う。 ■

4*9 系: i を階数、$A : \mathcal{U}(i)$ を型、$a_1, a_2 : A$ を要素、n : TruncLevel を要素とする。A が n 型ならば、$a_1 = a_2$ も n 型である。 [0052]

相対的な n 型の概念も導入する。

4*10 定義: i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数、n : TruncLevel を要素とする。型 $\mathsf{IsTruncMap}(n, f) : \mathcal{U}(i)$ を $\prod_{y:B} \mathsf{IsTrunc}(n, \mathsf{Fiber}(f, y))$ と定義する。 [005Q]

4*11 補題: i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数、$a : A$ と $c : \mathsf{Fiber}(f, f(a))$ を要素とすると、同値 [005S]

$$(\mathsf{record}\{\!|\mathsf{elem} \equiv a, \mathsf{id} \equiv \mathsf{refl}|\!\} = c) \simeq \mathsf{Fiber}(\mathsf{ap}(f)\{\mathsf{proj}_1(c), a\}, \mathsf{proj}_2(c))$$

を構成できる。

証明: 定理 3.2*4 を適用する。レトラクト

$$\begin{aligned}
& \textstyle\sum_{z:\mathsf{Fiber}(f,f(a))} \mathsf{Fiber}(\mathsf{ap}(f)\{\mathsf{proj}_1(c), a\}, \mathsf{proj}_2(c)) \\
\triangleleft \quad & \{\text{並び替え}\} \\
& \textstyle\sum_{x:A} \sum_{p:x=a} \sum_{q:f(x)=f(a)} f(p) = q \\
\triangleleft \quad & \{\text{補題 3.3*1}\} \\
& \textstyle\sum_{q:f(a)=f(a)} f(\mathsf{refl}) = q
\end{aligned}$$

を得て、最後の型は命題 3.1*4 より可縮である。 ■

4*12 命題: i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数とする。 [005R]

1 型 $\mathsf{IsTruncMap}(-2, f)$ と $\mathsf{IsEquiv}(f)$ は論理的に同値である。
2 要素 n : TruncLevel に対して、次の型は論理的に同値である。
- $\mathsf{IsTruncMap}(\mathsf{succ}(n), f)$
- $\prod_{x_1,x_2:A} \mathsf{IsTruncMap}(n, \mathsf{ap}(f)\{x_1, x_2\})$

証明: 1 は定義から自明である。
2 は次のように分かる。

$$\begin{aligned}
& \mathsf{IsTruncMap}(\mathsf{succ}(n), f) \\
\leftrightarrow \quad & \{\text{定義}\} \\
& \textstyle\prod_{y:B} \prod_{z_1,z_2:\mathsf{Fiber}(f,y)} \mathsf{IsTrunc}(n, z_1 = z_2) \\
\leftrightarrow \quad & \{\text{並び替え}\} \\
& \textstyle\prod_{x_1:A} \prod_{y:B} \prod_{p_1:f(x_1)=y} \prod_{z_2:\mathsf{Fiber}(f,y)} \mathsf{IsTrunc}(n, \mathsf{record}\{\!|\mathsf{elem} \equiv x_1, \mathsf{id} \equiv p_1|\!\} = z_2) \\
\leftrightarrow \quad & \{p_1 \text{ についての帰納法}\} \\
& \textstyle\prod_{x_1:A} \prod_{z_2:\mathsf{Fiber}(f,f(x_1))} \mathsf{IsTrunc}(n, \mathsf{record}\{\!|\mathsf{elem} \equiv x_1, \mathsf{id} \equiv \mathsf{refl}|\!\} = z_2) \\
\leftrightarrow \quad & \{\text{補題 4*11}\} \\
& \textstyle\prod_{x_1:A} \prod_{z_2:\mathsf{Fiber}(f,f(x_1))} \mathsf{IsTrunc}(n, \mathsf{Fiber}(\mathsf{ap}(f), \mathsf{proj}_2(z_2))) \\
\leftrightarrow \quad & \{\text{並び替え}\} \\
& \textstyle\prod_{x_2,x_1:A} \prod_{p:f(x_2)=f(x_1)} \mathsf{IsTrunc}(n, \mathsf{Fiber}(\mathsf{ap}(f), p)) \\
\leftrightarrow \quad & \{\text{定義}\} \\
& \textstyle\prod_{x_2,x_1:A} \mathsf{IsTruncMap}(n, \mathsf{ap}(f)\{x_2, x_1\})
\end{aligned}$$

■

[003Z] ## 4.1 命題

-2 型は可縮な型であり、すべての次元の構造が自明であるという最も単純な型である。次に単純な型は -1 型である。定義から、A が -1 型であるとは、任意の $x_1, x_2 : A$ に対して $x_1 = x_2$ が可縮であることである。つまり、任意の二つの要素の間にただ一つだけ同一視がある。A を -1 型とすると、A の要素が存在するという情報には意味があるが、A の二つの要素が同一かどうかは考える意味がない。このような A を**命題**と考え、A の要素を命題の**証明**と考える。

[0040] **4.1*1 定義**: i を階数、$A : \mathcal{U}(i)$ を型とする。型 $\mathsf{IsProp}(A) : \mathcal{U}(i)$ を $\mathsf{IsTrunc}(-1, A)$ と定義する。$\mathsf{IsProp}(A)$ の要素がある時、A は**命題** *(proposition)* であると言う。

[004G] **4.1*2 例**: $\mathbf{0}$ は命題である。実際、帰納法で直ちに示せる。

型が命題であることを示すには次が便利である。

[0041] **4.1*3 命題**: i を階数、$A : \mathcal{U}(i)$ を型とする。次の型は論理的に同値である。

1. $\mathsf{IsProp}(A)$
2. $\prod_{x_1, x_2 : A} x_1 = x_2$
3. $A \to \mathsf{IsContr}(A)$

証明: 1 から 2 は定義からすぐである。

2 から 3 を示す。$H : \prod_{x_1, x_2 : A} x_1 = x_2$ と $a : A$ を仮定する。a があるので、A が可縮であることを示すには $\prod_{x:A} a = x$ の要素を構成すればよいが、$\lambda x.H(a, x)$ でよい。

3 から 1 を示す。$H : A \to \mathsf{IsContr}(A)$ と $x_1, x_2 : A$ を仮定する。$x_1 = x_2$ が可縮であることを示すが、$H(x_1) : \mathsf{IsContr}(A)$ があるので、命題 3.1*3 を適用すればよい。 ■

レコード型の同一視型を決定する場面において、命題の部分は無視できることを示す (命題 4.1*5)。

[006B] **4.1*4 補題**: i を階数、$A : \mathcal{U}(i)$ を型、$P, B : A \to \mathcal{U}(i)$ を型の族、$c : \sum_{x:A} P(x)$ を要素、$b : B(\mathsf{proj}_1(c))$ を要素とする。$\prod_{x:A} \mathsf{IsProp}(P(x))$ の要素があり、$\sum_{x:A} B(x)$ は可縮であるならば、$\sum_{z:\sum_{x:A} P(x)} B(\mathsf{proj}_1(z))$ は可縮である。

証明: 並び換えによりレトラクト $(\sum_{z:\sum_{x:A} P(x)} B(\mathsf{proj}_1(z))) \triangleleft (\sum_{z:\sum_{x:A} B(x)} P(\mathsf{proj}_1(z)))$ を得る。後者は命題 4*8 と命題 4*7 より命題であり、要素 $\mathsf{pair}(\mathsf{pair}(\mathsf{proj}_1(c), b), \mathsf{proj}_2(c))$ を持つので、命題 4.1*3 より可縮である。 ■

[0049] **4.1*5 命題**: i を階数、$A : \mathcal{U}(i)$ を型、$B : A \to \mathcal{U}(i)$ を型の族、$c_1, c_2 : \sum_{x:A} B(x)$ を要素とする。$\prod_{x:A} \mathsf{IsProp}(B(x))$ の要素があるならば、同値 $(c_1 = c_2) \simeq (\mathsf{proj}_1(c_1) = \mathsf{proj}_1(c_2))$ を得る。

証明: 定理 3.2*4 を適用する。補題 4.1*4 より、$\sum_{x:A} \mathsf{proj}_1(c_1) = x$ が可縮であることを示せばよいが、これは命題 3.1*4 から従う。 ■

命題 4.1*5 より、各 $B(x)$ が命題の時は、$\sum_{x:A} B(x)$ の要素の同一視に関しては二番目の要素は完全に無視される。そのため、$\sum_{x:A} B(x)$ は要素 $a : A$ と要素 $b : B(a)$ の対のなす型というよりは、要素 $a : A$ であって**性質** $B(a)$ を満たすもののなす A の**部分型**であると考えられる。この視点を強調するために記法を導入する。

4.1*6 記法: i を階数、$A : \mathcal{U}(i)$ を型、$B : A \to \mathcal{U}(i)$ を型の族とする。関数外延性の下で $\prod_{x:A} \mathsf{IsProp}(B(x))$ の要素がある時、$\sum_{x:A} B(x)$ のことを $\{x : A \mid B(x)\}$ と書くことがある。さらに、要素 $c : \{x : A \mid B(x)\}$ に対して、proj_1 を省略して c そのものを A の要素とみなすことがある。 [004A]

さて、これまでいくつかの型に IsXXX という形の命名をしてきたが、これは (関数外延性の下で) その型が命題であることを意図している。多くの概念が可縮性を軸に定義されるので、$\mathsf{IsContr}(A)$ が命題であること (命題 4.1*7) が基本的である。

4.1*7 命題: 関数外延性を仮定する。i を階数、$A : \mathcal{U}(i)$ を型とすると、型 $\mathsf{IsContr}(A)$ は命題である。 [0042]

証明: 命題 4.1*3 より、$\mathsf{IsContr}(A) \to \mathsf{IsContr}(\mathsf{IsContr}(A))$ を示せばよい。A が可縮であると仮定する。要素 $a_0 : A$ を得る。レトラクトの列

$$\begin{aligned} & \mathsf{IsContr}(A) \\ \lhd \quad & \{\text{定義}\} \\ & \textstyle\sum_{a:A} \prod_{x:A} a = x \\ \lhd \quad & \{A \text{ は可縮}\} \\ & \textstyle\prod_{x:A} a_0 = x \end{aligned}$$

を得て、最後の型は命題 3.1*3 と命題 3.4*3 より可縮である。 ■

4.1*8 系: 関数外延性を仮定する。i を階数、$A : \mathcal{U}(i)$ を型、$n : \mathsf{TruncLevel}$ を要素とすると、$\mathsf{IsTrunc}(n, A)$ は命題である。 [0043]

証明: 命題 4.1*7 と命題 4*6 から、n についての帰納法で示せる。 ■

4.1*9 系: 関数外延性を仮定する。i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数とすると、$\mathsf{IsEquiv}(f)$ は命題である。 [0044]

4.1.1 節で諸々の同値の概念も命題であることを見る。

$\mathsf{IsTrunc}(n, A)$ が命題である (系 4.1*8) ということは、$\langle n \rangle\text{-}\mathsf{Type}(i)$ は $\mathcal{U}(i)$ の部分型となる。一価性公理から、その同一視型を計算できる。

4.1*10 命題: 一価性と関数外延性を仮定する。i を階数、$n : \mathsf{TruncLevel}$ を要素とする。$\langle n \rangle\text{-}\mathsf{Type}(i)$ は $\mathsf{succ}(n)$ 型である。 [0054]

証明: $A, B : \langle n \rangle\text{-}\mathsf{Type}(i)$ を要素とする。一価性と命題 4.1*5 と系 4.1*8 より、同値 $(A = B) \simeq (A.\mathsf{Type} \simeq B.\mathsf{Type})$ を得て、後者は命題 4*6 と命題 4*7 と系 4*9 から n 型である。 ■

命題の相対版も考える。

[005Y] **4.1*11 定義**: i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数とする。型 $\mathsf{IsEmb}(f) : \mathcal{U}(i)$ を $\mathsf{IsTruncMap}(-1, f)$ と定義する。$\mathsf{IsEmb}(f)$ の要素がある時、f は**埋め込み** *(embedding)* であると言う。

[0062] **4.1*12 命題**: i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数とする。次は論理的に同値である。

1 f は埋め込みである。
2 任意の $x_1 : A$ に対して、$\sum_{x_2:A} f(x_1) = f(x_2)$ は可縮である。

証明: 命題 4*12 より、1 は任意の $x_1, x_2 : A$ に対して $\mathsf{ap}(f) : x_1 = x_2 \to f(x_1) = f(x_2)$ が同値であることと論理的に同値である。定理 3.2*4 より、これは 2 と論理的に同値である。 ■

[004N] 4.1.1 同値の概念

系 4.1*9 より、関数が同値であることは命題であることが分かったが、3.6.1 節で導入した他の同値の概念が命題であることも示す。

まずは両側可逆性 (定義 3.6.1*1) を考える。

[004P] **4.1.1*1 演習**: 関数外延性を仮定する。i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数とする。f は同値であると仮定して、次を示せ。

1 任意の型 $X : \mathcal{U}(i)$ に対して、関数 $\lambda g.f \circ g : (X \to A) \to (X \to B)$ は同値である。
2 任意の型 $Y : \mathcal{U}(i)$ に対して、関数 $\lambda h.h \circ f : (B \to Y) \to (A \to Y)$ は同値である。

[004Q] **4.1.1*2 系**: 関数外延性を仮定する。i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数とする。f が同値ならば、$\mathsf{LInv}(f)$ と $\mathsf{RInv}(f)$ は可縮である。

証明: 関数外延性から、レトラクト $\mathsf{RInv}(f) \triangleleft \mathsf{Fiber}(\lambda(g : B \to A).f \circ g, \mathsf{id})$ を得て、右辺は演習 4.1.1*1 より可縮である。$\mathsf{LInv}(f)$ についても同様である。 ■

[004O] **4.1.1*3 命題**: 関数外延性を仮定する。i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数とすると、型 $\mathsf{IsBiinv}(f)$ は命題である。

証明: 命題 4.1*3 より、f が両側可逆であると仮定して $\mathsf{IsBiinv}(f)$ が可縮であることを示せばよいが、命題 3.6.1*2 より f は同値なので、系 4.1.1*2 から $\mathsf{IsBiinv}(f)$ が可縮であることが従う。 ■

次に、半随伴同値 (定義 3.6.1*3) を考える。

[004S] **4.1.1*4 補題**: i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数、$a_1, a_2 : A$ を要素とする。f が同値ならば $\mathsf{ap}(f) : a_1 = a_2 \to f(a_1) = f(a_2)$ は同値である。

証明: 命題 4*8 と命題 4*12 から従う。 ■

[004R] **4.1.1*5 命題**: 関数外延性を仮定する。i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数とすると、型 $\mathsf{IsHAE}(f)$ は命題である。

証明: 命題 4.1*3 より、f が半随伴同値であると仮定して $\mathsf{IsHAE}(f)$ が可縮であることを示せばよい。演習 3.6.1*4 より f は同値である。レトラクト

$$\begin{aligned}
&\mathsf{IsHAE}(f)\\
\triangleleft\quad&\{\text{並び替え}\}\\
&\textstyle\sum_{g:B\to A}\sum_{\varepsilon:f\circ g\sim\mathsf{id}}\sum_{\eta:g\circ f\sim\mathsf{id}}\prod_{x:A} f(\eta(x))=\varepsilon(f(x))\\
\triangleleft\quad&\{\text{系 4.1.1*2。可縮性から適当な } g \text{ と } \varepsilon \text{ を取れる。}\}\\
&\textstyle\sum_{\eta:g\circ f\sim\mathsf{id}}\prod_{x:A} f(\eta(x))=\varepsilon(f(x))\\
\triangleleft\quad&\{\text{例 2.3*18}\}\\
&\textstyle\prod_{x:A}\sum_{p:g(f(x))=x} f(p)=\varepsilon(f(x))
\end{aligned}$$

を得て、最後の型は補題 4.1.1*4 と命題 3.4*3 より可縮である。 ■

4.2 集合 [004B]

-1 型の次に単純な型は 0 型である。A が 0 型の時は、任意の $x_1, x_2 : A$ に対して $x_1 = x_2$ は命題である。よって、二つの要素が同一視されるかどうかには興味があるが、どう同一視されるかは考える意味がない。このような型を**集合**であると考える。

4.2*1 定義: i を階数、$A : \mathcal{U}(i)$ を型とする。型 $\mathsf{IsSet}(A) : \mathcal{U}(i)$ を $\mathsf{IsTrunc}(0, A)$ と定義する。$\mathsf{IsSet}(A)$ の要素がある時、A は**集合** *(set)* であると言う。 [004C]

4.2*2 命題: i を階数、$A : \mathcal{U}(i)$ を型とする。次の型は論理的に同値である。 [004D]

1. $\mathsf{IsSet}(A)$
2. $\prod_{x:A} \mathsf{IsContr}(x = x)$
3. $\prod_{x:A}\prod_{z:x=x} \mathsf{refl} = z$ ($Axiom\ K$ と呼ばれる)
4. $\prod_{x_1,x_2:A}\prod_{z_1,z_2:x_1=x_2} z_1 = z_2$ (*uniqueness of identity principle (UIP)* と呼ばれる)

証明: 1 から 2 を示す。A が集合であると仮定する。要素 $x : A$ に対して、$x = x$ は命題である。要素 $\mathsf{refl} : x = x$ があるので、命題 4.1*3 より $x = x$ は可縮である。

2 から 3 は命題 3.1*3 による。

4 において z_1 について帰納法を使うと 3 そのものになるので 3 から 4 が従う。

4 から 1 は命題 4.1*3 による。 ■

型が集合であることの十分条件として、同一視型が**決定可能**であるというのがある (定理 4.2*4)。ここで、型 P が決定可能とは、$P + (P \to \mathbf{0})$ の要素があることを言う。

4.2*3 補題: i を階数、$A : \mathcal{U}(i)$ を型、$E : A \to A \to \mathcal{U}(i)$ を型の族、$r : \prod_{x:A} E(x, x)$ と $f : \prod_{x_1,x_2:A} E(x_1, x_2) \to x_1 = x_2$ を関数とする。$\prod_{x_1,x_2:A} \mathsf{IsProp}(E(x_1, x_2))$ の要素がある時、同値 $\prod_{x_1(x_2,A)}(x_1 = x_2) \simeq E(x_1, x_2)$ を構成でき、特に A は集合である。 [004E]

証明: r から同一視型の帰納法より関数 $g : \prod_{x_1,x_2:A} x_1 = x_2 \to E(x_1, x_2)$ を得る。$E(x_1, x_2)$ が命題であるという仮定から、f と g はレトラクト $E(x_1, x_2) \triangleleft (x_1 = x_2)$ を定める。よって、定理 3.2*4 より同値 $(x_1 = x_2) \simeq E(x_1, x_2)$ を得る。 ■

[004H] **4.2*4 定理** (Hedberg): i を階数、$A : \mathcal{U}(i)$ を型、$d : \prod_{x_1,x_2:A}(x_1 = x_2)+(x_1 = x_2 \to \mathbf{0})$ を関数とすると、A は集合である。[Hedberg--1998-0000]

証明: 補題 4.2*3 を適用する。$E : A \to A \to \mathcal{U}(i)$ を次のように定義する。$x_1, x_2 : A$ に対して、$T(x_1, x_2) : (x_1 = x_2) + (x_1 = x_2 \to \mathbf{0}) \to \mathcal{U}(i)$ を $T(x_1, x_2, \mathsf{in}_1(z)) \equiv \mathbf{1}$ と $T(x_2, x_2, \mathsf{in}_2(z)) \equiv \mathbf{0}$ で定義する。$E(x_1, x_2) \equiv T(x_1, x_2, d(x_1, x_2))$ と定義する。例 3.1*2 と命題 4*8 と例 4.1*2 から、各 $E(x_1, x_2)$ は命題である。関数 $g : \prod_z T(x_1, x_2, z) \to x_1 = x_2$ を $g(\mathsf{in}_1(u), w) \equiv u$ と $g(\mathsf{in}_2(v), w) \equiv \mathsf{ind}_{\mathbf{0}}(w, \_)$ で定義できるので、関数 $E(x_1, x_2) \to x_1 = x_2$ を得る。最後に、任意の $x : A$ に対して同値 $h : \prod_z T(x, x, z) \simeq \mathbf{1}$ を $h(\mathsf{in}_1(u)) \equiv \mathsf{id}$ と $h(\mathsf{in}_2(v)) \equiv \mathsf{ind}_{\mathbf{0}}(v(\mathsf{refl}), \_)$ と定義できるので、関数 $\prod_{x:A} E(x, x)$ を得る。 ■

[004Z] 4.3 切り詰め

型は∞グルーポイドという高次元の構造を持つのであったが、その豊富な構造ゆえ型を解析するのは必ずしも容易ではない。高次元の構造を解析する常套手段はより低次元の構造で近似することである。型の n 型による最良の近似を導入する。

[0050] **4.3*1 規則**: i を階数、$A : \mathcal{U}(i)$ を型、$n : \mathsf{TruncLevel}$ を要素とする。

- n **切り詰め** *(n-truncation)* $\|A\|_n : \mathcal{U}(i)$ を構成できる。
- $\|A\|_n$ は n 型である (ことを示す要素を構成できる)。
- 要素 $a : A$ に対して、要素 $|a|_n : \|A\|_n$ を構成できる。
- $c : \|A\|_n$ を要素、j を階数、$B : \|A\|_n \to \mathcal{U}(j)$ を型の族、$f : \prod_{x:A} B(|x|_n)$ を関数とする。$\prod_{z:\|A\|_n} \mathsf{IsTrunc}(n, B(z))$ の要素があるなら、要素 $\mathsf{ind}_{\|A\|_n}(c, B, f) : B(c)$ を構成できる。
- $a : A$ を要素、j を階数、$B : \|A\|_n \to \mathcal{U}(j)$ を型の族、$f : \prod_{x:A} B(|x|_n)$ を関数とする。$\prod_{z:\|A\|_n} \mathsf{IsTrunc}(n, B(z))$ の要素があるなら、$\mathsf{ind}_{\|A\|_n}(a, B, f) \equiv f(a)$ を定義される。

規則 4.3*1 は帰納的型の規則と類似している。実際、$\|A\|_n$ を高次帰納的型 (5 章) の一種と捉えることもできる。$\lambda a.|a|_n$ は構成子である。それより下の規則は**帰納法原理**を表すが、帰納法が使える状況は B が n 型の族である場合に制限されている。

[0051] **4.3*2 命題**: i を階数、$A : \mathcal{U}(i)$ を型、$n : \mathsf{TruncLevel}$ を要素とする。次は論理的に同値である。

1 A は n 型である。
2 関数 $\lambda x.|x|_n : A \to \|A\|_n$ は同値である。
3 A は $\|A\|_n$ のレトラクトである。

証明: 2 から 3 は自明である。3 から 1 は命題 4*5 による。

1 から 2 を示す。A が n 型なので、帰納法より関数 $g : \|A\|_n \to A$ であって任意の $a : A$ に対して $g(|a|_n) \equiv a$ となるものを構成できる。特に、$g \circ (\lambda x.|x|_n) \sim \mathsf{id}$ である。$(\lambda x.|x|_n) \circ g \sim \mathsf{id}$ を示す。各 $z : \|A\|_n$ に対して $|g(z)|_n = z$ は系 4*9 より n 型なので、

帰納法により $\prod_{x:A} |g(|x|_n)|_n = |x|_n$ を示せばよいがこれは定義から明らかである。 ■

$\|A\|_n$ の性質を調べる際には一価性が不可欠である。一価性をどう使うかは次の定理 4.3*3 にまとめられる。

4.3*3 定理: 一価性と関数外延性を仮定する。i を階数、$A : \mathcal{U}(i)$ を型、$B : A \to \mathcal{U}(i)$ を型の族、$n :$ TruncLevel を要素とする。任意の $x : A$ に対して $B(x)$ は n 型であると仮定する。命題 4.1*10 より、帰納法で型の族 $T : \|A\|_{\mathsf{succ}(n)} \to \mathcal{U}(i)$ であって、任意の $u : \|A\|_{\mathsf{succ}(n)}$ に対して $T(u)$ は n 型であり、任意の $x : A$ に対して $T(|x|_{\mathsf{succ}(n)}) \equiv B(x)$ であるものを構成できる。命題 4*7 と命題 4*8 より $\sum_{u:\|A\|_{\mathsf{succ}(n)}} T(u)$ は $\mathsf{succ}(n)$ 型なので、関数 $\lambda z.\mathsf{pair}(|\mathsf{proj}_1(z)|_{\mathsf{succ}(n)}, \mathsf{proj}_2(z)) : (\sum_{x:A} B(x)) \to (\sum_{u:\|A\|_{\mathsf{succ}(n)}} T(u))$ は関数 [0056]

$$H : \|\textstyle\sum_{x:A} B(x)\|_{\mathsf{succ}(n)} \to (\sum_{u:\|A\|_{\mathsf{succ}(n)}} T(u))$$

を誘導する。この時、関数 H は同値である。

証明: $\sum_{u:\|A\|_{\mathsf{succ}(n)}} T(u)$ が $\|\sum_{x:A} B(x)\|_{\mathsf{succ}(n)}$ と同じ帰納法原理を満たすことを見る。$C : (\sum_{u:\|A\|_{\mathsf{succ}(n)}} T(u)) \to \mathcal{U}(i)$ を型の族とし、各 $C(u)$ は $\mathsf{succ}(n)$ 型であると仮定する。$f : \prod_{z:\sum_{x:A} B(x)} C(\mathsf{pair}(|z|_{\mathsf{succ}(n)}, \mathsf{proj}_2(z)))$ を関数とする。$D : \prod_{u:\|A\|_{\mathsf{succ}(n)}} T(u) \to \mathcal{U}(i)$ を C のカリー化、$g : \prod_{x:A} \prod_{y:B(x)} D(|x|_{\mathsf{succ}(n)}, y)$ を f のカリー化とする。命題 4*6 より各 $\prod_{v:T(u)} D(u, v)$ は $\mathsf{succ}(n)$ 型なので、帰納法で関数 $h : \prod_{u:\|A\|_{\mathsf{succ}(n)}} \prod_{v:T(u)} D(u, v)$ を得る。h を逆カリー化すればよい。 ■

定理 4.3*3 を使う例として、$\|A\|_n$ の同一視型の特徴付けを与える。n が -2 の場合は自明なので、興味があるのはそれ以外の場合である。

4.3*4 命題: 一価性と関数外延性を仮定する。i を階数、$A : \mathcal{U}(i)$ を型、$a_1, a_2 : A$ を要素、$n :$ TruncLevel を要素とすると、同値 $(|a_1|_{\mathsf{succ}(n)} = |a_2|_{\mathsf{succ}(n)}) \simeq \|a_1 = a_2\|_n$ を構成できる。 [0055]

証明: 定理 4.3*3 を型の族 $\lambda x.\|a_1 = x\|_n : A \to \mathcal{U}(i)$ に適用すると、定理 3.2*4 より、$\|\sum_{x:A} \|a_1 = x\|_n\|_{\mathsf{succ}(n)}$ が可縮であることを示せば十分である。要素 $c_1 : \|\sum_{x:A} \|a_1 = x\|_n\|_{\mathsf{succ}(n)}$ を $|\mathsf{pair}(a_1, |\mathsf{refl}|_n)|_{\mathsf{succ}(n)}$ と定義する。任意の $w : \|\sum_{x:A} \|a_1 = x\|_n\|_{\mathsf{succ}(n)}$ に対して同一視 $c_1 = w$ を構成する。系 4*9 よりこの同一視型は $\mathsf{succ}(n)$ 型なので、帰納法より任意の $x : A$ と $v : \|a_1 = x\|_n$ に対して同一視 $c_1 = |\mathsf{pair}(x, v)|_{\mathsf{succ}(n)}$ を構成すればよい。この同一視型は定義から n 型なので、帰納法より任意の $x : A$ と $y : a_1 = x$ に対して同一視 $c_1 = |\mathsf{pair}(x, |y|_n)|_{\mathsf{succ}(n)}$ を構成すればよいが、同一視型の帰納法で refl を与えればよい。 ■

4.4 述語論理 [0057]

命題の概念の導入により、**一階述語論理**を型理論の中で模倣できる。

4.4*1 記法: i を階数とする。 [0058]

- 型 $\top : \mathcal{U}(0)$ を $\mathbf{1}$ と定義する。

- 型 $P, Q : \mathcal{U}(i)$ に対して、型 $P \wedge Q : \mathcal{U}(i)$ を $P \times Q$ と定義する。
- 型 $\bot : \mathcal{U}(0)$ を $\mathbf{0}$ と定義する。
- 型 $P, Q : \mathcal{U}(i)$ に対して、型 $P \vee Q : \mathcal{U}(i)$ を $\|P + Q\|_{-1}$ と定義する。
- 型 $P, Q : \mathcal{U}(i)$ に対して、型 $P \Rightarrow Q : \mathcal{U}(i)$ を $P \to Q$ と定義する。
- 型 $P : \mathcal{U}(i)$ に対して、型 $\neg P : \mathcal{U}(i)$ を $P \Rightarrow \bot$ と定義する。
- 型 $P, Q : \mathcal{U}(i)$ に対して、型 $P \Leftrightarrow Q : \mathcal{U}(i)$ を $(P \Rightarrow Q) \wedge (Q \Rightarrow P)$ と定義する。
- 型 $A : \mathcal{U}(i)$ と型の族 $P : A \to \mathcal{U}(i)$ に対して、型 $\forall_{x:A} P(x) : \mathcal{U}(i)$ を $\prod_{x:A} P(x)$ と定義する。
- 型 $A : \mathcal{U}(i)$ と型の族 $B : A \to \mathcal{U}(i)$ に対して、型 $\exists_{x:A} B(x) : \mathcal{U}(i)$ を $\| \sum_{x:A} B(x) \|_{-1}$ と定義する。

これらの記法は P や Q が命題である場合に使い、結果も命題であることが分かる。

[005A] **4.4*2 用語**: i を階数、$P : \mathcal{U}(i)$ を型とする。P が命題である場合、「P の要素がある」の代わりに「P は**真**である」という言い方をする。

ただし、特別な公理を課さない限り型理論で模倣できる論理は**直観主義論理**である。特に、命題 P に対して $P \vee \neg P$ が真であるとは限らない。

[0059] **4.4*3 公理** (排中律): **排中律** *(law of excluded middle)* は任意の階数 i と型 $P : \mathcal{U}(i)$ に対して、P が命題ならば $P \vee \neg P$ が真であることを要請する。

排中律は従来の数学では当たり前のように使われるが、ホモトピー型理論ではほとんど**排中律を仮定しない**。

[005T] 4.5 連結性

n 型は n 次元より上の構造が自明な型であったが、逆に n 次元以下の構造が自明な型を導入する。

[005U] **4.5*1 定義**: i を階数、$A : \mathcal{U}(i)$ を型、n : TruncLevel を要素とする。型 IsConnected$(n, A) : \mathcal{U}(i)$ を IsContr$(\|A\|_n)$ と定義する。IsConnected(n, A) の要素がある時、A は n **連結** *(n-connected)* であると言う。

[005V] **4.5*2 定義**: i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数、n : TruncLevel を要素とする。型 IsConnMap$(n, f) : \mathcal{U}(i)$ を $\prod_{y:B}$ IsConnected$(n,$ Fiber$(f, y))$ と定義する。IsConnMap(n, f) の要素がある時、f は n **連結**であると言う。

[005W] **4.5*3 注意**: 定義 4.5*2 の意味での n 連結関数は古典的なホモトピー論における $n+1$ 連結写像に対応するものである。

[005X] **4.5*4 命題**: i を階数、$A : \mathcal{U}(i)$ を型、n : TruncLevel を要素とする。A が n 型かつ n 連結ならば、A は可縮である。

証明: 命題 4.3*2 から従う。 ■

定義から、任意の関数は -2 連結である。-1 連結な関数は**全射**であると考える。

4.5*5 定義: i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数とする。型 $\mathsf{IsSurj}(f) : \mathcal{U}(i)$ を $\forall_{y:B}\exists_{x:A} f(x) = y$ と定義する。$\mathsf{IsSurj}(f)$ の要素がある時、f は**全射** *(surjection)* であると言う。 [005Z]

4.5*6 命題: i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数とする。次は論理的に同値である。 [0074]

1 f は全射である。
2 f は -1 連結である。

証明: $y : B$ を要素とする。命題 4.1*3 より、$\|\mathsf{Fiber}(f, y)\|_{-1}$ と $\mathsf{IsContr}(\|\mathsf{Fiber}(f, y)\|_{-1})$ は論理的に同値であるから主張が従う。 ■

4.6 構造同一原理 [004V]

3.5 節で構造同一原理の例をいくつか見たが、群や環などの数学的に興味深い構造については後回しにしていた。これは、型理論において「$a_1 = a_2$」は単なる型なので公理を適切に記述するためには 4.1 節の意味での命題の概念が必要だからである。例えば、ある二項演算 $\times$ の結合律を素直に $(a_1 \times a_2) \times a_3 = a_1 \times (a_2 \times a_3)$ と書くと、これは公理というよりも構造の一部になってしまう。結合律を適切に公理として扱うためには、同一視型が命題であることを要請すればよい。

4.6*1 例: i を階数とする。型 $\mathsf{GroupStr}(i) : \mathcal{U}(\mathsf{succ}(i))$ を次のレコード型と定義する。 [004W]

- $\mathsf{Carrier} : \mathcal{U}(i)$
- $\mathsf{unit} : \mathsf{Carrier}$
- $\mathsf{mul} : \mathsf{Carrier} \to \mathsf{Carrier} \to \mathsf{Carrier}$
- $\mathsf{inv} : \mathsf{Carrier} \to \mathsf{Carrier}$

要素 $G : \mathsf{GroupStr}(i)$ に対して、型 $\mathsf{GroupAxiom}(G) : \mathcal{U}(i)$ を次のレコード型と定義する。

- $\_ : \mathsf{IsSet}(G.\mathsf{Carrier})$
- $\_ : \prod_{x:G.\mathsf{Carrier}} G.(\mathsf{mul}(G.\mathsf{unit}, x)) = x$
- $\_ : \prod_{x:G.\mathsf{Carrier}} G.\mathsf{mul}(x, G.\mathsf{unit}) = x$
- $\_ : \prod_{x_1,x_2,x_3:G.\mathsf{Carrier}} G.\mathsf{mul}(G.\mathsf{mul}(x_1, x_2), x_3) = G.\mathsf{mul}(x_1, G.\mathsf{mul}(x_3, x_3))$
- $\_ : \prod_{x:G.\mathsf{Carrier}} G.\mathsf{mul}(G.\mathsf{inv}(x), x) = G.\mathsf{unit}$
- $\_ : \prod_{x:G.\mathsf{Carrier}} G.\mathsf{mul}(x, G.\mathsf{inv}(x)) = G.\mathsf{unit}$

型 $\mathsf{Group}(i) : \mathcal{U}(\mathsf{succ}(i))$ を $\sum_{X:\mathsf{GroupStr}(i)} \mathsf{GroupAxiom}(X)$ と定義する。$\mathsf{Group}(i)$ の要素を (階数 i の) **群** *(group)* と呼ぶ。群 $A, B : \mathsf{Group}(i)$ に対して、同一視型 $A = B$ を計算しよう。命題 4*7 と命題 4*6 より $\mathsf{GroupAxiom}(X)$ は命題であることが分かるので、命題 4.1*5 より同値 $(A = B) \simeq (\mathsf{proj}_1(A) = \mathsf{proj}_1(B))$ を得る。例 3.5*2 と同様に、$\mathsf{proj}_1(A) = \mathsf{proj}_1(B)$ は次のレコード型と同値であることが分かる。

- $f : (\mathsf{proj}_1(A)).\mathsf{Carrier} \simeq (\mathsf{proj}_1(B)).\mathsf{Carrier}$
- $\_ : f((\mathsf{proj}_1(A)).\mathsf{unit}) = (\mathsf{proj}_1(B)).\mathsf{unit}$

- _ : $\prod_{x_1,x_2} f((\mathsf{proj}_1(A)).\mathsf{mul}(x_1,x_2)) = (\mathsf{proj}_1(B)).\mathsf{mul}(f(x_1),f(x_2))$
- _ : $\prod_{x} f((\mathsf{proj}_1(A)).\mathsf{inv}(x)) = (\mathsf{proj}_1(B)).\mathsf{inv}(f(x))$

この型の要素はいわゆる**群同型** *(group isomorphism)* である。GroupAxiom(G) の定義で IsSet(G.Carrier) を落とすと、Group(i) の同一視の特徴付けは演算だけでなく残りの構造も保つ同値というものになってしまう。

ちなみに、群の間の乗法を保つ関数は単位元と逆元も保つことを示せるので、群同型の型は次のレコード型とも同値である。

- $f : (\mathsf{proj}_1(A)).\mathsf{Carrier} \simeq (\mathsf{proj}_1(B)).\mathsf{Carrier}$
- _ : $\prod_{x_1,x_2} f((\mathsf{proj}_1(A)).\mathsf{mul}(x_1,x_2)) = (\mathsf{proj}_1(B)).\mathsf{mul}(f(x_1),f(x_2))$

[004Y] **4.6*2 例**: i を階数とする。型 RingStr(i) : $\mathcal{U}(\mathsf{succ}(i))$ を次のレコード型と定義する。

- Carrier : $\mathcal{U}(i)$
- zero : Carrier
- plus : Carrier → Carrier → Carrier
- neg : Carrier → Carrier
- one : Carrier
- mul : Carrier → Carrier → Carrier

要素 R : RingStr(i) に対して、型 RingAxiom(R) : $\mathcal{U}(i)$ を次のレコード型と定義する。

- _ : IsSet(R.Carrier)
- _ : $\prod_{x:R.\mathsf{Carrier}} R.\mathsf{plus}(R.\mathsf{zero},x) = x$
- _ : $\prod_{x_1,x_2,x_3:R.\mathsf{Carrier}} R.\mathsf{plus}(R.\mathsf{plus}(x_1,x_2),x_3) = R.\mathsf{plus}(x_1,R.\mathsf{plus}(x_2,x_3))$
- _ : $\prod_{x_1,x_2:R.\mathsf{Carrier}} R.\mathsf{plus}(x_1,x_2) = R.\mathsf{plus}(x_2,x_1)$
- _ : $\prod_{x} R.\mathsf{plus}(x,R.\mathsf{neg}(x)) = R.\mathsf{zero}$
- _ : $\prod_{x} R.\mathsf{mul}(R.\mathsf{one},x) = x$
- _ : $\prod_{x} R.\mathsf{mul}(x,R.\mathsf{one}) = x$
- _ : $\prod_{x_1,x_2,x_3:R.\mathsf{Carrier}} R.\mathsf{mul}(R.\mathsf{mul}(x_1,x_2),x_3) = R.\mathsf{mul}(x_1,R.\mathsf{mul}(x_2,x_3))$
- _ : $\prod_{x,y_1,y_2:R.\mathsf{Carrier}} R.\mathsf{mul}(x,R.\mathsf{plus}(y_1,y_2)) = R.\mathsf{plus}(R.\mathsf{mul}(x,y_1),R.\mathsf{mul}(x,y_2))$
- _ : $\prod_{x_1,x_2,y:R.\mathsf{Carrier}} R.\mathsf{mul}(R.\mathsf{plus}(x_1,x_2),y) = R.\mathsf{plus}(R.\mathsf{mul}(x_1,y),R.\mathsf{mul}(x_2,y))$

型 Ring(i) : $\mathcal{U}(\mathsf{succ}(i))$ を $\sum_{X:\mathsf{RingStr}(i)} \mathsf{RingAxiom}(X)$ と定義する。Ring(i) の要素を (階数 i の) **環** *(ring)* と呼ぶ。環 A,B : Ring(i) に対して、同一視型 $A = B$ は次のレコード型と同値であることが分かる。

- $f : (\mathsf{proj}_1(A)).\mathsf{Carrier} \simeq (\mathsf{proj}_1(B)).\mathsf{Carrier}$
- _ : $f((\mathsf{proj}_1(A)).\mathsf{zero}) = (\mathsf{proj}_1(B)).\mathsf{zero}$
- _ : $\prod_{x_1,x_2} f((\mathsf{proj}_1(A)).\mathsf{plus}(x_1,x_2)) = (\mathsf{proj}_1(B)).\mathsf{plus}(f(x_1),f(x_2))$
- _ : $\prod_{x} f((\mathsf{proj}_1(A)).\mathsf{neg}(x)) = (\mathsf{proj}_1(B)).\mathsf{neg}(f(x))$
- _ : $f((\mathsf{proj}_1(A)).\mathsf{one}) = (\mathsf{proj}_1(B)).\mathsf{one}$
- _ : $\prod_{x_1,x_2} f((\mathsf{proj}_1(A)).\mathsf{mul}(x_1,x_2)) = (\mathsf{proj}_1(B)).\mathsf{mul}(f(x_1),f(x_2))$

この型の要素はいわゆる**環同型** *(ring isomorphism)* である。

5 高次帰納的型

[0003]

高次帰納的型 *(higher inductive type)* は一価性公理と並んで、ホモトピー型理論において重要な概念である。通常の帰納的型は要素を構成するいくつかの構成子によって自由に生成される型であった。高次帰納的型は要素の構成に加えて、**同一視**の構成子も許したものである。

最も汎用的な高次帰納的型は**ファイバー余積**または**押し出し**で、5.1 節で導入する。ファイバー余積を使って、**懸垂**、n **次元球面**、**コファイバー**を構成できて、有限の CW **複体**を構成できることがわかる。このように、高次帰納的型は豊富に高次元の同一視の構造を持つ型を構成するのに使われる。

高次帰納的型は従来の数学 (例えば集合論) でも意味をなす概念である。例えばファイバー余積は集合論でもよく使われる。しかし、集合論で同一視を追加することは要素を等しくすることになるが、追加された同一視の情報が失われる。例えば、二つの要素の間の同一視を一つ追加することと二つ追加することは区別されない。このことに起因して、集合論においては高次帰納的型は必ずしも良く振る舞わないことが知られている。一方、ホモトピー型理論においては、高次帰納的型で追加された同一視の情報が失われることはない。この事実は**降下性**と呼ばれる性質にまとめられ、一価性公理の下で証明される (5.2 節)。

高次帰納的型を説明するためにいくつか準備をしておく。

5*1 定義: i を階数、$A : \mathcal{U}(i)$ を型、$B : A \to \mathcal{U}(i)$ を型の族、$a_1, a_2 : A$ を要素、$p : a_1 = a_2$ を同一視、$b_1 : B(a_1)$ と $b_2 : B(a_2)$ を要素とする。型 $b_1 =^B_p b_2 : \mathcal{U}(i)$ を $\mathsf{transport}(B, p, b_1) = b_2$ と定義する。$b_1 =^B_p b_2$ の要素を b_1 と b_2 の p **上の同一視** *(identification over p)* と呼ぶ。 [003L]

5*2 定義: i を階数、$A : \mathcal{U}(i)$ を型、$B : A \to \mathcal{U}(i)$ を型の族、$f : \prod_{x:A} B(x)$ を関数とする。関数 [007I]

$$\mathsf{apd}(f) : \prod_{\{x_1, x_2 : A\}} \prod_{p : x_1 = x_2} f(x_1) =^B_p f(x_2)$$

を $\mathsf{apd}(f, \mathsf{refl}) \equiv \mathsf{refl}$ で定義する。

5.1 ファイバー余積

[0033]

ファイバー余積は汎用的な高次帰納的型である。

5.1*1 規則: i を階数、$A, B, C : \mathcal{U}(i)$ を型、$f : C \to A$ と $g : C \to B$ を関数とする。 [003R]

- **ファイバー余積** *(fiber coproduct)* $A\ {}_f{+}_g\ B : \mathcal{U}(i)$ を構成できる。
- 要素 $a : A$ に対して、要素 $\mathsf{in}_1(a) : A\ {}_f{+}_g\ B$ を構成できる。
- 要素 $b : B$ に対して、要素 $\mathsf{in}_2(b) : A\ {}_f{+}_g\ B$ を構成できる。
- 要素 $c : C$ に対して、同一視 $\mathsf{glue}(c) : \mathsf{in}_1(f(c)) = \mathsf{in}_2(g(c))$ を構成できる。
- $d : A\ {}_f{+}_g\ B$ を要素、j を階数、$E : A\ {}_f{+}_g\ B \to \mathcal{U}(j)$ を型の族、$e_1 : \prod_{x:A} E(\mathsf{in}_1(x))$ と $e_2 : \prod_{y:B} E(\mathsf{in}_2(y))$ を関数、$p : \prod_{z:C} e_1(f(z)) =^E_{\mathsf{glue}(z)} e_2(g(z))$ を同一視とすると、要素 $\mathsf{ind}_{\_+\_}(d, E, e_1, e_2, p) : E(d)$ を構成できる。
- $a : A$ を要素、j を階数、$E : A\ {}_f{+}_g\ B \to \mathcal{U}(j)$ を型の族、$e_1 : \prod_{x:A} E(\mathsf{in}_1(x))$ と $e_2 : \prod_{y:B} E(\mathsf{in}_2(y))$ を関数、$p : \prod_{z:C} e_1(f(z)) =^E_{\mathsf{glue}(z)} e_2(g(z))$ を同一視とすると、$\mathsf{ind}_{\_+\_}(\mathsf{in}_1(a), E, e_1, e_2, p) \equiv e_1(a)$ と定義される。
- $b : B$ を要素、j を階数、$E : A\ {}_f{+}_g\ B \to \mathcal{U}(j)$ を型の族、$e_1 : \prod_{x:A} E(\mathsf{in}_1(x))$ と $e_2 : \prod_{y:B} E(\mathsf{in}_2(y))$ を関数、$p : \prod_{z:C} e_1(f(z)) =^E_{\mathsf{glue}(z)} e_2(g(z))$ を同一視とすると、$\mathsf{ind}_{\_+\_}(\mathsf{in}_2(b), E, e_1, e_2, p) \equiv e_2(b)$ と定義される。
- $c : C$ を要素、j を階数、$E : A\ {}_f{+}_g\ B \to \mathcal{U}(j)$ を型の族、$e_1 : \prod_{x:A} E(\mathsf{in}_1(x))$ と $e_2 : \prod_{y:B} E(\mathsf{in}_2(y))$ を関数、$p : \prod_{z:C} e_1(f(z)) =^E_{\mathsf{glue}(z)} e_2(g(z))$ を同一視とすると、同一視 $\mathsf{ind}_{\_+\_}\text{-}\mathsf{glue}(c, E, e_1, e_2, p) : \mathsf{apd}(\lambda d.\mathsf{ind}_{\_+\_}(d, E, e_1, e_2, p), \mathsf{glue}(c)) = p(c)$ を構成できる。

高次帰納的型も通常の帰納的型と同様にいくつかの**構成子**によって定められる。ファイバー余積の場合、in_1 と in_2 と glue が構成子である。in_1 と in_2 は $A\ {}_f{+}_g\ B$ の要素を構成する普通の構成子である。glue が**同一視**を構成するという点が新しい。要素を構成する構成子は *point constructor* と、同一視を構成する構成子は *path constructor* と呼ばれることもある。

残りの規則はファイバー余積の**帰納法原理**で、型 $A\ {}_f{+}_g\ B$ が in_1 と in_2 と glue で自由に生成されることを表す。ただし、$\mathsf{ind}_{\_+\_}(\mathsf{in}_1(a), E, e_1, e_2, p) \equiv e_1(a)$ と $\mathsf{ind}_{\_+\_}(\mathsf{in}_2(b), E, e_1, e_2, p) \equiv e_2(b)$ は定義であるのに対し、$\mathsf{apd}(\lambda d.\mathsf{ind}_{\_+\_}(d, E, e_1, e_2, p), \mathsf{glue}(c)) = p(c)$ は同一視を構成する。理想的には $\mathsf{apd}(\lambda d.\mathsf{ind}_{\_+\_}(d, E, e_1, e_2, p), \mathsf{glue}(c)) \equiv p(c)$ と定義したいところだが、これは単なる $\mathsf{ind}_{\_+\_}$ の定義ではなく $\mathsf{ind}_{\_+\_}$ と apd に関する複合的な定義である。apd はプリミティヴではなく我々が構成した関数 (定義 5*2) なので、それに関する新たな定義の正当性は無い。

ファイバー余積を使って構成できる型には次のようなものがある。

[007V] **5.1*2 定義**: i を階数、$A : \mathcal{U}(i)$ を型とする。関数 $f : A \to \mathbf{1}$ を $\lambda\_.\star$ とする。**懸垂** *(suspension)* $\mathsf{Susp}(A) : \mathcal{U}(i)$ を $\mathbf{1}\ {}_f{+}_f\ \mathbf{1}$ と定義する。

[003M] **5.1*3 定義**: 自然数 $n : \mathbb{N}$ に対して、n **次元球面** *(n-dimensional sphere)* $\mathbb{S}^n : \mathcal{U}(0)$ を次のように定義する。

- 便宜的に -1 次元球面 $\mathbb{S}^{-1}$ を $\mathbf{0}$ と定義する。
- $\mathbb{S}^0 \equiv \mathsf{Susp}(\mathbb{S}^{-1})$
- $\mathbb{S}^{\mathsf{succ}(n)} \equiv \mathsf{Susp}(\mathbb{S}^n)$

[007W] **5.1*4 定義**: i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数とする。関数 $g : A \to \mathbf{1}$ を $\lambda\_.\star$ とする。f の**コファイバー** *(cofiber)* $\mathsf{Cofiber}(f)$ を $B\ {}_f{+}_g\ \mathbf{1}$ と定義する。

位相幾何学において、CW **複体**は、n 次元球を境界 $(n-1$ 次元球面) に沿って貼り付ける操作を繰り返して得られる空間である。n 次元球が可縮であることをふまえると、n 次元球を境界 $f : \mathbb{S}^{n-1} \to X$ に沿って空間 X に貼り付けたものは f のコファイバーとホモトピー同値である。したがって、型理論において、$f : \mathbb{S}^{n-1} \to X$ の形の関数のコファイバーを取る操作を繰り返すことで有限 CW 複体 (のホモトピー型) を構成できる。

5.2 降下性 [0034]

高次帰納的型は一価性公理の下で**降下性** *(descent property)* という著しい性質を持つ。降下性という言葉は∞トポス理論 [Lurie--2009-0000] から取ったもので、余極限と有限極限を強く関連付ける性質である。

ファイバー余積の場合の降下性について説明する。少し長くなるが、ファイバー余積に関連する図式に名前を付ける。

5.2*1 定義: i を階数とする。型 $\mathsf{Span}(i) : \mathcal{U}(\mathsf{succ}(i))$ を次のレコード型と定義する。 [007J]

- $\mathsf{Left} : \mathcal{U}(i)$
- $\mathsf{Right} : \mathcal{U}(i)$
- $\mathsf{Center} : \mathcal{U}(i)$
- $\mathsf{leg\text{-}l} : \mathsf{Center} \to \mathsf{Left}$
- $\mathsf{leg\text{-}r} : \mathsf{Center} \to \mathsf{Right}$

$\mathsf{Span}(i)$ の要素を (階数 i の) **スパン** *(span)* と呼ぶ。

5.2*2 定義: i を階数、$A : \mathsf{Span}(i)$ をスパンとする。型 $\mathsf{Cocone}(A) : \mathcal{U}(\mathsf{succ}(i))$ を次のレコード型と定義する。 [007K]

- $\mathsf{Vertex} : \mathcal{U}(i)$
- $\mathsf{in}_{\mathsf{Left}} : A.\mathsf{Left} \to \mathsf{Vertex}$
- $\mathsf{in}_{\mathsf{Right}} : A.\mathsf{Right} \to \mathsf{Vertex}$
- $\mathsf{in}_{\mathsf{Center}} : A.\mathsf{Center} \to \mathsf{Vertex}$
- $\mathsf{in}_{\mathsf{leg\text{-}l}} : \prod_{x:A.\mathsf{Center}} \mathsf{in}_{\mathsf{Center}}(x) = \mathsf{in}_{\mathsf{Left}}(A.\mathsf{leg\text{-}l}(x))$
- $\mathsf{in}_{\mathsf{leg\text{-}r}} : \prod_{x:A.\mathsf{Center}} \mathsf{in}_{\mathsf{Center}}(x) = \mathsf{in}_{\mathsf{Right}}(A.\mathsf{leg\text{-}r}(x))$

$\mathsf{Cocone}(A)$ の要素を A 下の**余錐** *(cocone)* と呼ぶ。

5.2*3 定義: i を階数、$A : \mathsf{Span}(i)$ をスパン、$C : \mathsf{Cocone}(A)$ を余錐とする。関数 [007L]

$$\mathsf{cmp}(C) : A.\mathsf{Left} \,{}_{A.\mathsf{leg\text{-}l}}{+}_{A.\mathsf{leg\text{-}r}}\, A.\mathsf{Right} \to C.\mathsf{Vertex}$$

を次のように定義する。

- $x : A.\mathsf{Left}$ に対して $\mathsf{cmp}(C, \mathsf{in}_1(x)) \equiv C.\mathsf{in}_{\mathsf{Left}}(x)$
- $y : A.\mathsf{Right}$ に対して $\mathsf{cmp}(C, \mathsf{in}_2(y)) \equiv C.\mathsf{in}_{\mathsf{Right}}(y)$
- $z : A.\mathsf{Center}$ に対して $\mathsf{cmp}(C, \mathsf{glue}(z)) = C.\mathsf{in}_{\mathsf{leg\text{-}r}}(z) \circ (C.\mathsf{in}_{\mathsf{leg\text{-}l}}(z))^{-1}$

5.2*4 定義: i を階数、$A : \mathsf{Span}(i)$ をスパン、$C : \mathsf{Cocone}(A)$ を余錐とする。型 [007M]

IsUniversal$(C) : \mathcal{U}(i)$ を IsEquiv(cmp(C)) と定義する。IsUniversal(C) の要素があるとき、C は**普遍余錐** *(universal cocone)* であると言う。

降下性は「相対的な」余錐を使って説明されるので、それを導入する。

[007N] **5.2*5 定義**: i を階数、A : Span(i) をスパンとする。型 SpanOver$(A) : \mathcal{U}$(succ(i)) を次のレコード型と定義する。

- Left : A.Left $\to \mathcal{U}(i)$
- Right : A.Right $\to \mathcal{U}(i)$
- Center : A.Center $\to \mathcal{U}(i)$
- leg-l : $\prod_{\{x:A.\mathsf{Center}\}}$ Center$(x) \to$ Left$(A.\text{leg-l}(x))$
- leg-r : $\prod_{\{x:A.\mathsf{Center}\}}$ Center$(x) \to$ Right$(A.\text{leg-r}(x))$

SpanOver(A) の要素を A **上のスパン**と呼ぶ。

[007O] **5.2*6 定義**: i を階数、A : Span(i) をスパン、B : SpanOver(A) を A 上のスパン、C : Cocone(A) を余錐とする。型 CoconeOver$(B, C) : \mathcal{U}$(succ(i)) と次のレコード型と定義する。

- Vertex : C.Vertex $\to \mathcal{U}(i)$
- $\mathsf{in}_{\mathsf{Left}} : \prod_{\{x:A.\mathsf{Left}\}} B.\mathsf{Left}(x) \to \mathsf{Vertex}(C.\mathsf{in}_{\mathsf{Left}}(x))$
- $\mathsf{in}_{\mathsf{Right}} : \prod_{\{x:A.\mathsf{Right}\}} B.\mathsf{Right}(x) \to \mathsf{Vertex}(C.\mathsf{in}_{\mathsf{Right}}(x))$
- $\mathsf{in}_{\mathsf{Center}} : \prod_{\{x:A.\mathsf{Center}\}} B.\mathsf{Center}(x) \to \mathsf{Vertex}(C.\mathsf{in}_{\mathsf{Center}}(x))$
- $\mathsf{in}_{\text{leg-l}} : \prod_{\{x:A.\mathsf{Center}\}} \prod_{y:B.\mathsf{Center}(x)} \mathsf{in}_{\mathsf{Center}}(y) =^{\mathsf{Vertex}}_{C.\mathsf{in}_{\text{leg-l}}(x)} \mathsf{in}_{\mathsf{Left}}(B.\text{leg-l}(y))$
- $\mathsf{in}_{\text{leg-r}} : \prod_{\{x:A.\mathsf{Center}\}} \prod_{y:B.\mathsf{Center}(x)} \mathsf{in}_{\mathsf{Center}}(y) =^{\mathsf{Vertex}}_{C.\mathsf{in}_{\text{leg-r}}(x)} \mathsf{in}_{\mathsf{Right}}(B.\text{leg-r}(y))$

CoconeOver(B, C) の要素を C **上** B **下の余錐**と呼ぶ。

相対的な余錐は対型を取ることで普通の余錐にすることができる。

[007P] **5.2*7 定義**: i を階数、A : Span(i) をスパン、B : SpanOver(A) を A 上のスパンとする。スパン Total(B) : Span(i) を次のように定義する。

- Left $\equiv \sum_{x:A.\mathsf{Left}} B.\mathsf{Left}(x)$
- Right $\equiv \sum_{x:A.\mathsf{Right}} B.\mathsf{Right}(x)$
- Center $\equiv \sum_{x:A.\mathsf{Center}} B.\mathsf{Center}(x)$
- leg-l$(x) \equiv$ pair$(A.\text{leg-l}(\mathsf{proj}_1(x)), B.\text{leg-l}(\mathsf{proj}_2(x)))$
- leg-r$(x) \equiv$ pair$(A.\text{leg-r}(\mathsf{proj}_1(x)), B.\text{leg-r}(\mathsf{proj}_2(x)))$

[007Q] **5.2*8 定義**: i を階数、A : Span(i) をスパン、B : SpanOver(A) を A 上のスパン、C : Cocone(A) を余錐、D : CoconeOver(B, C) を C 上の余錐とする。余錐 Total(D) : Cocone(Total(B)) を次のように定義する。

- Vertex $\equiv \sum_{x:C.\mathsf{Vertex}} D.\mathsf{Vertex}(x)$
- $\mathsf{in}_{\mathsf{Left}}(x) \equiv \mathsf{pair}(C.\mathsf{in}_{\mathsf{Left}}(\mathsf{proj}_1(x)), D.\mathsf{in}_{\mathsf{Left}}(\mathsf{proj}_2(x)))$
- $\mathsf{in}_{\mathsf{Right}}(x) \equiv \mathsf{pair}(C.\mathsf{in}_{\mathsf{Right}}(\mathsf{proj}_1(x)), D.\mathsf{in}_{\mathsf{Right}}(\mathsf{proj}_2(x)))$
- $\mathsf{in}_{\mathsf{Center}}(x) \equiv \mathsf{pair}(C.\mathsf{in}_{\mathsf{Center}}(\mathsf{proj}_1(x)), D.\mathsf{in}_{\mathsf{Center}}(\mathsf{proj}_2(x)))$

- $\mathsf{in}_{\mathsf{leg\text{-}l}}(x)$ は $\mathsf{pair}(C.\mathsf{in}_{\mathsf{leg\text{-}l}}(\mathsf{proj}_1(x)), D.\mathsf{in}_{\mathsf{leg\text{-}l}}(\mathsf{proj}_2(x)))$ に系 3.2*5 を適用して定義する。
- $\mathsf{in}_{\mathsf{leg\text{-}r}}(x)$ は $\mathsf{pair}(C.\mathsf{in}_{\mathsf{leg\text{-}r}}(\mathsf{proj}_1(x)), D.\mathsf{in}_{\mathsf{leg\text{-}r}}(\mathsf{proj}_2(x)))$ に系 3.2*5 を適用して定義する。

最後にカルテシアンという性質を導入する。

5.2*9 定義: i を階数、$A : \mathsf{Span}(i)$ をスパン、$B : \mathsf{SpanOver}(A)$ を A 上のスパンとする。型 $\mathsf{IsCart}(B) : \mathcal{U}(i)$ を $(\forall_{x:A.\mathsf{Center}} \mathsf{IsEquiv}(B.\mathsf{leg\text{-}l}\{x\})) \wedge (\forall_{x:A.\mathsf{Center}} \mathsf{IsEquiv}(B.\mathsf{leg\text{-}r}\{x\}))$ と定義する。$\mathsf{IsCart}(B)$ の要素があるとき、B は**カルテシアン** *(cartesian)* であると言う。 [007R]

5.2*10 定義: i を階数、$A : \mathsf{Span}(i)$ をスパン、$B : \mathsf{SpanOver}(A)$ を A 上のスパン、$C : \mathsf{Cocone}(A)$ を余錐、$D : \mathsf{CoconeOver}(B, C)$ を C 上の余錐とする。型 $\mathsf{IsCart}(D) : \mathcal{U}(i)$ を $(\forall_{x:A.\mathsf{Left}} \mathsf{IsEquiv}(D.\mathsf{in}_{\mathsf{Left}}\{x\})) \wedge (\forall_{x:A.\mathsf{Right}} \mathsf{IsEquiv}(D.\mathsf{in}_{\mathsf{Right}}\{x\})) \wedge (\forall_{x:A.\mathsf{Center}} \mathsf{IsEquiv}(D.\mathsf{in}_{\mathsf{Center}}\{x\}))$ と定義する。$\mathsf{IsCart}(D)$ の要素があるとき、D は**カルテシアン** *(cartesian)* であると言う。 [007S]

いよいよ本題に入る。次の演習 5.2*11 はファイバー余積が**引き戻しで安定** *(stable under pullback)* であることを表し、これには一価性は必要ない。

5.2*11 演習: i を階数、$A : \mathsf{Span}(i)$ をスパン、$B : \mathsf{SpanOver}(A)$ を A 上のスパン、$C : \mathsf{Cocone}(A)$ を余錐、$D : \mathsf{CoconeOver}(B, C)$ を C 上の余錐とする。B がカルテシアンで、C が普遍余錐で、D がカルテシアンならば、$\mathsf{Total}(D) : \mathsf{Cocone}(\mathsf{Total}(B))$ は普遍余錐である。 [007T]

一価性はカルテシアンな C 上の余錐を構成するのに使われる。

5.2*12 演習: 関数外延性と一価性を仮定する。i を階数、$A : \mathsf{Span}(i)$ をスパン、$B : \mathsf{SpanOver}(A)$ を A 上のスパン、$C : \mathsf{Cocone}(A)$ を余錐とする。B がカルテシアンで、C が普遍余錐であると仮定すると、型 [003V]

$$\{D : \mathsf{CoconeOver}(B, C) \mid \mathsf{IsCart}(D)\}$$

は可縮である。さらに、一意に存在するカルテシアンな $D : \mathsf{CoconeOver}(B, C)$ に対して、$\mathsf{Total}(D) : \mathsf{Cocone}(\mathsf{Total}(B))$ は普遍余錐である。

∞トポス理論で降下性と呼ばれる性質は演習 5.2*13 のように述べられる。

5.2*13 演習: 関数外延性と一価性を仮定する。i を階数、$A : \mathsf{Span}(i)$ をスパン、$B : \mathsf{SpanOver}(A)$ を A 上のスパン、$C : \mathsf{Cocone}(A)$ を余錐、$D : \mathsf{CoconeOver}(B, C)$ を C 上の余錐とする。B がカルテシアンで、C が普遍余錐であるとすると、D がカルテシアンであることと $\mathsf{Total}(D)$ が普遍余錐であることは論理的に同値である。 [007U]

5.3 局所化 [007Y]

ファイバー余積以外の重要な高次帰納的型として**局所化** *(localization)* がある。

[007Z] **5.3*1 定義**: i を階数とする。型 $\mathsf{LocalGen}(i) : \mathcal{U}(\mathsf{succ}(i))$ を次のレコード型と定義する。

- $\mathsf{Index} : \mathcal{U}(i)$
- $\mathsf{Dom} : \mathsf{Index} \to \mathcal{U}(i)$
- $\mathsf{Cod} : \mathsf{Index} \to \mathcal{U}(i)$
- $\mathsf{fun} : \prod_{k:\mathsf{Index}} \mathsf{Dom}(k) \to \mathsf{Cod}(k)$

$\mathsf{LocalGen}(i)$ の要素を (階数 i の) **局所生成系** *(local generator)* と呼ぶ。

[0080] **5.3*2 定義**: i を階数、$G : \mathsf{LocalGen}(i)$ を局所生成系、$A : \mathcal{U}(i)$ を型とする。型 $\mathsf{IsLocal}(G, A) : \mathcal{U}(i)$ を

$$\forall_{k:G.\mathsf{Index}} \mathsf{IsEquiv}(\lambda(f : G.\mathsf{Cod}(k) \to A).f \circ G.\mathsf{fun}(k))$$

と定義する。$\mathsf{IsLocal}(G, A)$ の要素があるとき、A は G-**局所的** *(G-local)* であると言う。

型 A の G-**局所化** (定義 5.3*6) は A に最も近い G-局所的な型のことである。G-局所化の構成への最初の段階として、次の高次帰納的型を導入する。

[0081] **5.3*3 規則**: i を階数、$G : \mathsf{LocalGen}(i)$ を局所生成系、$A : \mathcal{U}(i)$ を型とする。

- A の**弱** G-**局所化** $\mathsf{WLoc}(G, A) : \mathcal{U}(i)$ を構成できる。
- 要素 $a : A$ に対して、要素 $\mathsf{in}(a) : \mathsf{WLoc}(G, A)$ を構成できる。
- $k : G.\mathsf{Index}$ を要素、$f : G.\mathsf{Dom}(k) \to \mathsf{WLoc}(G, A)$ を関数、$y : G.\mathsf{Cod}(k)$ を要素とすると、要素 $\mathsf{ext}(f, y) : \mathsf{WLoc}(G, A)$ を構成できる。
- $k : G.\mathsf{Index}$ を要素、$f : G.\mathsf{Dom}(k) \to \mathsf{WLoc}(G, A)$ を関数、$x : G.\mathsf{Dom}(k)$ を要素とすると、同一視 $\mathsf{is\text{-}ext}(f, x) : \mathsf{ext}(f, G.\mathsf{fun}(k, x)) = f(x)$ を構成できる。
- $b : \mathsf{WLoc}(G, A)$ を要素、j を階数、$C : \mathsf{WLoc}(G, A) \to \mathcal{U}(j)$ を型の族、$s : \prod_{a:A} C(\mathsf{in}(a))$ と $h : \prod_{\{k:G.\mathsf{Index}\}} \prod_{\{f:G.\mathsf{Dom}(k)\to\mathsf{WLoc}(G,A)\}} (\prod_{x:G.\mathsf{Dom}(k)} C(f(x))) \to (\prod_{y:G.\mathsf{Cod}(k)} C(\mathsf{ext}(f, y)))$ と $p : \prod_{\{k\}} \prod_{\{f\}} \prod_{g:\prod_{x:G.\mathsf{Dom}(k)} C(f(x))} \prod_{x:G.\mathsf{Dom}(k)} h(g, G.\mathsf{fun}(k, x))\ g(x)$ を関数とすると、要素 $\mathsf{ind}_{\mathsf{WLoc}}(b, C, s, h, p) : C(b)$ を構成できる。
- $a : A$ を要素、j を階数、$C : \mathsf{WLoc}(G, A) \to \mathcal{U}(j)$ を型の族、$s : \prod_{a:A} C(\mathsf{in}(a))$ と $h : \prod_{\{k:G.\mathsf{Index}\}} \prod_{\{f:G.\mathsf{Dom}(k)\to\mathsf{WLoc}(G,A)\}} (\prod_{x:G.\mathsf{Dom}(k)} C(f(x))) \to (\prod_{y:G.\mathsf{Cod}(k)} C(\mathsf{ext}(f, y)))$ と $p : \prod_{\{k\}} \prod_{\{f\}} \prod_{g:\prod_{x:G.\mathsf{Dom}(k)} C(f(x))} \prod_{x:G.\mathsf{Dom}(k)} h(g, G.\mathsf{fun}(k, x)) =^{C}_{\mathsf{is\text{-}ext}(f,x)} g(x)$ を関数とすると、$\mathsf{ind}_{\mathsf{WLoc}}(\mathsf{in}(a), C, s, h, p) \equiv s(a)$ と定義される。
- $k' : G.\mathsf{Index}$ を要素、$f' : G.\mathsf{Dom}(k') \to \mathsf{WLoc}(G, A)$ を関数、$y' : G.\mathsf{Cod}(k')$ を要素、j を階数、$C : \mathsf{WLoc}(G, A) \to \mathcal{U}(j)$ を型の族、$s : \prod_{a:A} C(\mathsf{in}(a))$ と $h : \prod_{\{k:G.\mathsf{Index}\}} \prod_{\{f:G.\mathsf{Dom}(k)\to\mathsf{WLoc}(G,A)\}} (\prod_{x:G.\mathsf{Dom}(k)} C(f(x))) \to (\prod_{y:G.\mathsf{Cod}(k)} C(\mathsf{ext}(f, y)))$ と $p : \prod_{\{k\}} \prod_{\{f\}} \prod_{g:\prod_{x:G.\mathsf{Dom}(k)} C(f(x))} \prod_{x:G.\mathsf{Dom}(k)} h(g, G.\mathsf{fun}(k, x)) =^{C}_{\mathsf{is\text{-}ext}(f,x)} g(x)$ を関数とすると、$\mathsf{ind}_{\mathsf{WLoc}}(\mathsf{ext}(f', y'), C, s, h, p) \equiv h(\lambda x.\mathsf{ind}_{\mathsf{WLoc}}(f'(x), C, s, h, p), y')$ と定義される。
- $k' : G.\mathsf{Index}$ を要素、$f' : G.\mathsf{Dom}(k') \to \mathsf{WLoc}(G, A)$ を関数、$x' : G.\mathsf{Dom}(k')$ を要素、j を階数、$C : \mathsf{WLoc}(G, A) \to \mathcal{U}(j)$ を型の族、$s : \prod_{a:A} C(\mathsf{in}(a))$ と $h : \prod_{\{k:G.\mathsf{Index}\}} \prod_{\{f:G.\mathsf{Dom}(k)\to\mathsf{WLoc}(G,A)\}} (\prod_{x:G.\mathsf{Dom}(k)} C(f(x))) \to (\prod_{y:G.\mathsf{Cod}(k)} C(\mathsf{ext}(f, y)))$ と $p : \prod_{\{k\}} \prod_{\{f\}} \prod_{g:\prod_{x:G.\mathsf{Dom}(k)} C(f(x))} \prod_{x:G.\mathsf{Dom}(k)} h(g, G.\mathsf{fun}(k, x)) =^{C}_{\mathsf{is\text{-}ext}(f,x)} g(x)$ を関数とすると、同一視 $\mathsf{ind}_{\mathsf{WLoc}}\text{-}\mathsf{is\text{-}ext}(f', x', C, s, h, p) : \mathsf{apd}(\lambda b.\mathsf{ind}_{\mathsf{WLoc}}(b, C, s, h, p), \mathsf{is\text{-}ext}$

$p(\lambda x.\mathsf{ind}_{\mathsf{WLoc}}(f'(x), C, s, h, p), x')$ を構成できる。

$\mathsf{WLoc}(G, A)$ は**再帰的**な高次帰納的型の例である。ここでいう再帰的とは、ext と is-ext の引数 f の型 $G.\mathsf{Dom}(k) \to \mathsf{WLoc}(G, A)$ に $\mathsf{WLoc}(G, A)$ 自身が現れるということである。ファイバー余積 (規則 5.1*1) は再帰的な高次帰納的型ではない。

定義から、関数

$$\lambda f.f \circ G.\mathsf{fun}(k) : (G.\mathsf{Cod}(k) \to \mathsf{WLoc}(G, A)) \to (G.\mathsf{Dom}(k) \to \mathsf{WLoc}(G, A))$$

のファイバーは (関数外延性の下で) 要素を持つが可縮であるとまでは言えない。可縮性を保証するためにはより高次の同一視の構成子を追加するのも一つの手だが、実は G に関数を追加することでも実現できる。

5.3*4 定義: i を階数、$A, B : \mathcal{U}(i)$ を型、$f : A \to B$ を関数とする。関数 $\mathsf{codiag}(f) : B \,{}_f{+}_f\, B \to B$ を [0082]

- $\mathsf{codiag}(f, \mathsf{in}_1(y)) \equiv y$
- $\mathsf{codiag}(f, \mathsf{in}_2(y)) \equiv y$
- $\mathsf{codiag}(f, \mathsf{glue}(x)) = \mathsf{refl}$

で定義する。

5.3*5 定義: i を階数、G : $\mathsf{LocalGen}(i)$ を局所生成系とする。局所生成系 $\mathsf{D}(G)$: $\mathsf{LocalGen}(i)$ を次のように定義する。 [0083]

- $\mathsf{Index} \equiv G.\mathsf{Index} + G.\mathsf{Index}$
- $\mathsf{fun}(\mathsf{in}_1(k)) \equiv G.\mathsf{fun}(k)$
- $\mathsf{fun}(\mathsf{in}_2(k)) \equiv \mathsf{codiag}(G.\mathsf{fun}(k))$
- Dom と Cod は fun から決まる。

5.3*6 定義: i を階数、$G : \mathsf{LocalGen}(i)$ を局所生成系、$A : \mathcal{U}(i)$ を型とする。A の G-**局所化** *(G-localization)* $\mathsf{Loc}(G, A) : \mathcal{U}(i)$ を $\mathsf{WLoc}(\mathsf{D}(G), A)$ と定義する。 [0084]

$\mathsf{Loc}(G, A)$ は次の**普遍性**を満たす。

5.3*7 演習: 関数外延性を仮定する。i を階数、$G : \mathsf{LocalGen}(i)$ を局所生成系、$A : \mathcal{U}(i)$ を型とする。 [0085]

1 $\mathsf{Loc}(G, A)$ は G-局所的である。
2 $X : \mathcal{U}(i)$ を型とし、X は G-局所的であると仮定する。関数

$$\lambda f.f \circ \mathsf{in} : (\mathsf{Loc}(G, A) \to X) \to (A \to X)$$

は同値である。

6 ホモトピー論

[0005]

7 圏論

[0004]

圏論は数学における対象とその間の写像の概念を抽象化した構造を調べる分野である。特に、対象の間の**同型**の概念が一般的に定義され、同型こそが対象の「正しい」同一視のしかたと考えられる。しかし、例えば集合論を数学の基礎とした場合、同型な対象であっても等しいとは限らないので、それらを厳密に同一視するわけにはいかない。一方、型理論を数学の基礎とした場合、一価性公理と同様の発想で同型の型と同一視型が同値になるような条件を圏に課す余地があり、実際に多くの圏の具体例は一価性公理の下でその条件を満たす。その意味で、圏論はホモトピー型理論を基礎とするのが適切である分野の一つである。

7.1 圏

[005B]

7.1*1 定義: i を階数とする。型 $\mathsf{PreCat}(i) : \mathcal{U}(\mathsf{succ}(i))$ を次のレコード型と定義する。 [005C]

- **対象** *(object)* の型 $\mathsf{Obj} : \mathcal{U}(i)$
- **射** *(morphism)* の型の族 $\mathsf{Map} : \mathsf{Obj} \to \mathsf{Obj} \to \mathcal{U}(i)$
- **恒等射** *(identity)*$\mathsf{id} : \prod_{\{x:\mathsf{Obj}\}} \mathsf{Map}(x, x)$
- 射の**合成** *(composition)*$\mathsf{comp} : \prod_{\{x_1,x_2,x_3:\mathsf{Obj}\}} \mathsf{Map}(x_2, x_3) \to \mathsf{Map}(x_1, x_2) \to \mathsf{Map}(x_1, x_3)$
- $\_ : \prod_{x_1,x_2:\mathsf{Obj}} \mathsf{IsSet}(\mathsf{Map}(x_1, x_2))$
- $\_ : \prod_{x_1,x_2:\mathsf{Obj}} \prod_{f:\mathsf{Map}(x_1,x_2)} \mathsf{comp}(\mathsf{id}, f) = f$
- $\_ : \prod_{x_1,x_2:\mathsf{Obj}} \prod_{f:\mathsf{Map}(x_1,x_2)} \mathsf{comp}(f, \mathsf{id}) = f$
- $\_ : \prod_{x_1,x_2,x_3,x_4:\mathsf{Obj}} \prod_{f_1:\mathsf{Map}(x_1,x_2)} \prod_{f_2:\mathsf{Map}(x_2,x_3)} \prod_{f_3:\mathsf{Map}(x_3,x_4)} \mathsf{comp}(\mathsf{comp}(f_3, f_2), f_1) = \mathsf{comp}(f_3, \mathsf{comp}(f_2, f_1))$

$\mathsf{PreCat}(i)$ の要素を (階数 i の) **前圏** *(precategory)* と呼ぶ。

射の型 $\mathsf{Map}(x_1, x_2)$ は集合であると要請される。これは、群の定義 (例 4.6*1) において Carrier は集合としたのと同様である。対して、Obj は集合とは限らない。

7.1*2 記法: i を階数、$C : \mathsf{PreCat}(i)$ を前圏とする。x が C の対象であることを $x : C.\mathsf{Obj}$ の代わりに単に $x : C$ と書く。対象 $x_1, x_2 : C$ に対して、$C.\mathsf{Map}(x_1, x_2)$ の代わりに単に $\mathsf{Map}(x_1, x_2)$ と書く。$x_1 : C$ と書いた時点で C の前圏の構造が暗黙に了解されるのでこの表記で曖昧性はない。同様に、対象 $x : C$ に対して、$C.\mathsf{id}\{x\}$ の代わりに単に $\mathsf{id}\{x\}$ と書く。合成 $\mathsf{comp}(f_2, f_1)$ は二項演算子を使って $f_2 \circ f_1$ と書く。 [005D]

7.1*3 例: 関数外延性を仮定する。i を階数とする。前圏 $\mathsf{Set}^{(\mathsf{Cat})}(i) : \mathsf{PreCat}(\mathsf{succ}(i))$ を [006H]

次のように定義する。

- Obj $\equiv \{A : \mathcal{U}(i) \mid \mathsf{IsSet}(A)\}$
- Map $\equiv \lambda(A, B).(A \to B)$
- 恒等射は例 2.2*7、合成は例 2.2*8 による。
- 命題 4*6 より、各 Map(A, B) は集合である。
- 他の公理は自明である。

[006N] **7.1*4 例**: i を階数、$C : \mathsf{PreCat}(i)$ を前圏とする。前圏 $\mathsf{Op}(C) : \mathsf{PreCat}(i)$ を次のように定義する。

- Obj $\equiv C.\mathsf{Obj}$
- Map $\equiv \lambda(x_1, x_2).C.\mathsf{Map}(x_2, x_1)$
- id $\equiv \lambda x.C.\mathsf{id}\{x\}$
- comp $\equiv \lambda(x_1, x_2, x_3).\lambda(g : \mathsf{Map}(x_2, x_3), f : \mathsf{Map}(x_1, x_2)).C.\mathsf{comp}(f, g)$
- 前圏の公理は C のそれから分かる。

前圏 C の対象の間には、同一視型の他に**同型**という同一視の概念が考えらる。

[005E] **7.1*5 定義**: i を階数、$C : \mathsf{PreCat}(i)$ を前圏、$x_1, x_2 : C$ を対象、$f : \mathsf{Map}(x_1, x_2)$ を射とする。

- 型 $\mathsf{LInv}(f) : \mathcal{U}(i)$ を $\mathsf{Record}\{\!|\mathsf{inv} : \mathsf{Map}(x_2, x_1), \mathsf{id} : \mathsf{inv} \circ f = \mathsf{id}|\!\}$ と定義する。
- 型 $\mathsf{RInv}(f) : \mathcal{U}(i)$ を $\mathsf{Record}\{\!|\mathsf{inv} : \mathsf{Map}(x_2, x_1), \mathsf{id} : f \circ \mathsf{inv} = \mathsf{id}|\!\}$ と定義する。
- 型 $\mathsf{IsIso}(f) : \mathcal{U}(i)$ を $\mathsf{Record}\{\!|\mathsf{linv} : \mathsf{LInv}(f), \mathsf{rinv} : \mathsf{RInv}(f)|\!\}$ と定義する。

$\mathsf{IsIso}(f)$ の要素がある時、f は**同型** *(isomorphism)* であるという。

[005G] **7.1*6 補題**: i を階数、$C : \mathsf{PreCat}(i)$ を前圏、$x_1, x_2 : C$ を対象、$f : \mathsf{Map}(x_1, x_2)$ を射とする。次は論理的に同値である。

1 f は同型である。
2 任意の対象 $y : C$ に対して、関数 $\lambda g.f \circ g : \mathsf{Map}(y, x_1) \to \mathsf{Map}(y, x_2)$ は同値である。
3 任意の対象 $y : C$ に対して、関数 $\lambda g.g \circ f : \mathsf{Map}(x_2, y) \to \mathsf{Map}(x_1, y)$ は同値である。

証明: 1 から 2 を示す。定義から、関数 $\lambda g.f \circ g : \mathsf{Map}(y, x_1) \to \mathsf{Map}(y, x_2)$ は両側可逆であることが分かる。よって、命題 3.6.1*2 よりこれは同値である。

2 から 1 を示す。$\lambda g.f \circ g : \mathsf{Map}(x_2, x_1) \to \mathsf{Map}(x_2, x_2)$ が同値なので、射 $h : \mathsf{Map}(x_2, x_1)$ と同一視 $p : f \circ h = \mathsf{id}$ を得る。関数 $\lambda g.f \circ g : \mathsf{Map}(x_1, x_1) \to \mathsf{Map}(x_1, x_2)$ が同値なので、同一視

$$\begin{array}{ll} & f \circ h \circ f \\ = & \{p\} \\ & f \\ = & \{\text{前圏の公理}\} \end{array}$$

$$f \circ \mathsf{id}$$

から同一視 $q : h \circ f = \mathsf{id}$ を得る。よって、f は同型である。

1 と 3 の同値性も同様である。 ■

7.1*7 命題: i を階数、$C : \mathsf{PreCat}(i)$ を前圏、$x_1, x_2 : C$ を対象、$f : \mathsf{Map}(x_1, x_2)$ を射とすると、型 $\mathsf{IsIso}(f)$ は命題である。 [005H]

証明: 命題 4.1*3 より、f が同型であると仮定して $\mathsf{IsIso}(f)$ が可縮であることを示せばよいが、これは補題 7.1*6 からすぐに従う。 ■

7.1*8 定義: i を階数、$C : \mathsf{PreCat}(i)$ を前圏、$x_1, x_2 : C$ を対象とする。型 $x_1 \cong x_2 : \mathcal{U}(i)$ を $\{f : \mathsf{Map}(x_1, x_2) \mid \mathsf{IsIso}(f)\}$ と定義する。 [005F]

同型の基本性質は次のようにまとめられる。

7.1*9 命題: i を階数、$C : \mathsf{PreCat}(i)$ を前圏とする。 [006X]

1 任意の対象 $x : C$ に対して、恒等射 $\mathsf{id} : \mathsf{Map}(x, x)$ は同型である。
2 任意の対象 $x_1, x_2, x_3, x_4 : C$ と射 $f_1 : \mathsf{Map}(x_1, x_2)$ と $f_2 : \mathsf{Map}(x_2, x_3)$ と $f_3 : \mathsf{Map}(x_3, x_4)$ に対して、$f_2 \circ f_1$ と $f_3 \circ f_2$ が同型ならば、f_1 と f_2 と f_3 と $f_3 \circ f_2 \circ f_1$ も同型である。

証明: 1 は定義からすぐに確かめられる。2 は補題 7.1*6 と命題 3.6*7 による。 ■

圏とは、前圏であって対象の間の同型の型と同一視型が同値になるようなものである。

7.1*10 定義: i を階数とする。 [005I]

- 前圏 $C : \mathsf{PreCat}(i)$ に対して、型 $\mathsf{IsCat}(C) : \mathcal{U}(i)$ を

$$\prod_{x:C} \mathsf{IsContr}(\textstyle\sum_{y:C} x \cong y)$$

と定義する。
- 型 $\mathsf{Cat}(i) : \mathcal{U}(\mathsf{succ}(i))$ を $\{C : \mathsf{PreCat}(i) \mid \mathsf{IsCat}(C)\}$ と定義する。

$\mathsf{Cat}(i)$ の要素を (階数 i の) **圏** *(category)* と呼ぶ。

C を圏とすると、恒等射は同型なので定理 3.2*4 を適用できて、同値

$$\prod_{x_1, x_2 : C} (x_1 = x_2) \simeq (x_1 \cong x_2)$$

を得る。逆に、前圏 C が圏であることを示すにはこのような同値を構成すれば十分である。

圏の典型例として、$\mathsf{Set}^{(\mathsf{Cat})}(i)$ (例 7.1*3) が圏であることを見る。

7.1*11 補題: 関数外延性を仮定する。i を階数、$A, B : \mathsf{Set}^{(\mathsf{Cat})}(i)$ を対象、$f : \mathsf{Map}(A, B)$ を射とする。次は論理的に同値である。 [006I]

1 f は $\mathsf{Set}^{(\mathsf{Cat})}(i)$ の射として同型である。
2 f は関数として同値である。

証明: 命題 3.6.1*2 による。 ■

[006J] **7.1*12 定理**: 関数外延性と一価性を仮定する。i を階数とすると、$\mathsf{Set}^{(\mathsf{Cat})}(i)$ は圏である。

証明: 対象 $A, B : \mathsf{Set}^{(\mathsf{Cat})}(i)$ に対して、補題 7.1*11 より $(A \cong B) \simeq (A \simeq B)$ を得て、一価性と命題 4.1*5 より $(A \simeq B) \simeq (A = B)$ を得る。 ■

[005J] 7.2 関手

関手は前圏の間の構造を保つ関数である。

[005K] **7.2*1 定義**: i を階数、$C, D : \mathsf{PreCat}(i)$ を前圏とする。型 $\mathsf{Fun}(C, D) : \mathcal{U}(i)$ を次のレコード型と定義する。

- $\mathsf{obj} : C.\mathsf{Obj} \to D.\mathsf{Obj}$
- $\mathsf{map} : \prod_{\{x_1, x_2 : C.\mathsf{Obj}\}} C.\mathsf{Map}(x_1, x_2) \to D.\mathsf{Map}(\mathsf{obj}(x_1), \mathsf{obj}(x_2))$
- $\_ : \prod_{x:C.\mathsf{Obj}} \mathsf{map}(C.\mathsf{id}(x)) = D.\mathsf{id}(\mathsf{obj}(x))$
- $\_ : \prod_{x_1,x_2,x_3:C.\mathsf{Obj}} \prod_{f_1:C.\mathsf{Map}(x_1,x_2)} \prod_{f_2:C.\mathsf{Map}(x_2,x_3)} \mathsf{map}(C.\mathsf{comp}(f_2, f_1)) = D.\mathsf{comp}(\mathsf{map}(f_2), \mathsf{map}(f_1))$

$\mathsf{Fun}(C, D)$ の要素を C から D への**関手** *(functor)* と呼ぶ。

[005L] **7.2*2 記法**: i を階数、$C, D : \mathsf{PreCat}(i)$ を前圏、$F : \mathsf{Fun}(C, D)$ を関手とする。

- 対象 $x : C$ に対して、$F.\mathsf{obj}(x) : D$ の代わりに単に $F(x)$ と書く。
- 対象 $x_1, x_2 : C$ と射 $f : \mathsf{Map}(x_1, x_2)$ に対して、$F.\mathsf{map}(f) : \mathsf{Map}(F(x_1), F(x_2))$ の代わりに単に $F(f)$ と書く。

[0060] **7.2*3 演習**: i 階数、$C, D : \mathsf{PreCat}(i)$ を前圏、$F : \mathsf{Fun}(C, D)$ を関手、$x_1, x_2 : C$ を対象、$f : \mathsf{Map}(x_1, x_2)$ を射とする。f が同型ならば、$F(f)$ は同型であることを示せ。

3.5 節や 4.6 節と同様に、前圏の**構造同一原理**が得られる。

[005M] **7.2*4 定義**: i を階数、$C, D : \mathsf{PreCat}(i)$ を前圏、$F : \mathsf{Fun}(C, D)$ を関手とする。型 $\mathsf{IsIso}(F) : \mathcal{U}(i)$ を次のレコード型と定義する。

- $\_ : \mathsf{IsEquiv}(F.\mathsf{obj})$
- $\_ : \prod_{x_1,x_2:C} \mathsf{IsEquiv}(F.\mathsf{map}\{x_1, x_2\})$

$\mathsf{IsIso}(F)$ の要素がある時、F は**前圏の同型** *(isomorphism of precategories)* であると言う。

[005N] **7.2*5 演習** (前圏の構造同一原理): 一価性と関数外延性を仮定する。i を階数、$C, D : \mathsf{PreCat}(i)$ を前圏とする。同値

$$(C = D) \simeq \{F : \mathsf{Fun}(C, D) \mid \mathsf{IsIso}(F)\}$$

を構成せよ。

従来の圏論においては、定義 7.2*4 は圏の同一視の概念としては強過ぎて、次の**弱圏同**

値こそが「正しい」圏の同一視の概念とされる。

7.2*6 定義: i を階数、$C, D : \mathsf{PreCat}(i)$ を前圏、$F : \mathsf{Fun}(C, D)$ を関手とする。 [0050]

- 型 $\mathsf{IsFF}(F) : \mathcal{U}(i)$ を $\forall_{x_1,x_2:C}\mathsf{IsEquiv}(F.\mathsf{map}\{x_1, x_2\})$ と定義する。$\mathsf{IsFF}(F)$ の要素がある時、F は**充満忠実** *(fully faithful)* であると言う。
- 型 $\mathsf{IsEssSurj}(F) : \mathcal{U}(i)$ を $\forall_{y:D}\exists_{x:C} F(x) \cong y$ と定義する。$\mathsf{IsEssSurj}(F)$ の要素がある時、F は**本質的全射** *(essentially surjective)* であると言う。
- 型 $\mathsf{IsWCatEquiv}(F) : \mathcal{U}(i)$ を $\mathsf{IsFF}(F) \wedge \mathsf{IsEssSurj}(F)$ と定義する。$\mathsf{IsWCatEquiv}(F)$ の要素がある時、F は**弱圏同値** *(weak categorical equivalence)* であると言う。

ホモトピー型理論での圏論では、圏の間の関手が前圏の同型であることと弱圏同値であることは同値であることに差はなくなる (定理 7.2*12)。演習 7.2*5 と合わせると、圏の間に弱圏同値があればそれらの圏は同一視される (系 7.2*13)。よって、弱圏同値が正しい同一視の概念であることが厳密な定理として得られる。

7.2*7 演習: i を階数、$C, D : \mathsf{PreCat}(i)$ を前圏、$F : \mathsf{Fun}(C, D)$ を関手、$x_1, x_2 : C$ を対象、$f : \mathsf{Map}(x_1, x_2)$ を射とする。F は充満忠実であると仮定する。$F(f)$ が同型ならば f は同型であることを示せ。 [0061]

7.2*8 定義: i を階数、$C, D : \mathsf{PreCat}(i)$ を前圏、$F : \mathsf{Fun}(C, D)$ を関手、$y : D$ を対象とする。型 $\mathsf{Fiber}^{\cong}(F, y) : \mathcal{U}(i)$ を次のレコード型と定義する。 [006Y]

- $\mathsf{obj} : C$
- $\mathsf{iso} : F(\mathsf{obj}) \cong y$

7.2*9 補題: i を階数、$C, D : \mathsf{PreCat}(i)$ を前圏、$F : \mathsf{Fun}(C, D)$ を関手、$y : D$ を対象とする。C が圏で、F が充満忠実ならば、$\mathsf{Fiber}^{\cong}(F, y)$ は命題である。 [006Z]

証明: 命題 4.1*3 を適用し、$a : \mathsf{Fiber}^{\cong}(F, y)$ を仮定して $\mathsf{Fiber}^{\cong}(F, y)$ が可縮であることを示す。レトラクト

$$
\begin{aligned}
& \mathsf{Fiber}^{\cong}(F, y) \\
\lhd\ & \{\text{定義}\} \\
& \textstyle\sum_{x:C} F(x) \cong y \\
\lhd\ & \{a.\mathsf{iso}\ \text{と合成}\} \\
& \textstyle\sum_{x:C} F(a.\mathsf{obj}) \cong F(x) \\
\lhd\ & \{\text{演習 } 7.2*3 \text{ と演習 } 7.2*7\} \\
& \textstyle\sum_{x:C} a.\mathsf{obj} \cong x
\end{aligned}
$$

を得て、最後の型は C が圏なので可縮である。 ■

7.2*10 補題: i を階数、$C, D : \mathsf{PreCat}(i)$ を前圏、$F : \mathsf{Fun}(C, D)$ を関手とする。C と D が圏で、F が充満忠実ならば、$F.\mathsf{obj} : C.\mathsf{Obj} \to D.\mathsf{Obj}$ は埋め込みである。 [0063]

証明: $y : D$ を対象とする。D が圏であることから、$\mathsf{Fiber}(F.\mathsf{obj}, y)$ は $\mathsf{Fiber}^{\cong}(F, y)$ のレトラクトであることが分かる。後者は補題 7.2*9 より命題である。 ■

[0075] **7.2*11 補題**: i を階数、$C, D : \mathsf{PreCat}(i)$ を前圏、$F : \mathsf{Fun}(C, D)$ を関手とする。D が圏で、F が本質的全射ならば、$F.\mathsf{obj} : C.\mathsf{Obj} \to D.\mathsf{Obj}$ は全射である。

証明: 定義からすぐである。 ■

[005P] **7.2*12 定理**: i を階数、$C, D : \mathsf{PreCat}(i)$ を前圏、$F : \mathsf{Fun}(C, D)$ を関手とする。C と D が圏ならば、次は論理的に同値である。

1 F は前圏の同型である。
2 F は弱圏同値である。

証明: 1 から 2 は自明である。
2 から 1 を示す。$F.\mathsf{obj} : C.\mathsf{Obj} \to D.\mathsf{Obj}$ が同値であることを示せばよいが、これは命題 4.5*4 と補題 7.2*10 と命題 4.5*6 と補題 7.2*11 から従う。 ■

[0064] **7.2*13 系** (圏の構造同一原理): 一価性と関数外延性を仮定する。i を階数、$C, D : \mathsf{Cat}(i)$ を圏とすると、同値

$$(C = D) \simeq \{F : \mathsf{Fun}(C, D) \mid \mathsf{IsWCatEquiv}(F)\}$$

を得る。

[0065] 7.3 自然変換

自然変換は関手の間の射の概念である。実際、関手を対象、自然変換を射とする前圏を構成できる (定義 7.3*4)。さらに、終域が圏であるような関手のなす前圏は圏であることを示す (命題 7.3*6)。

[0066] **7.3*1 定義**: i を階数、$C, D : \mathsf{PreCat}(i)$ を前圏、$F, G : \mathsf{Fun}(C, D)$ を関手とする。型 $\mathsf{NatTrans}(F, G) : \mathcal{U}(i)$ を

$$\{t : \textstyle\prod_{x:C} \mathsf{Map}(F(x), G(x)) \mid \forall_{x_1,x_2:C} \forall_{f:\mathsf{Map}(x_1,x_2)} G(f) \circ t(x_1) = t(x_2) \circ F(f)\}$$

と定義する。$\mathsf{NatTrans}(F, G)$ の要素を F から G への**自然変換** *(natural transformation)* と呼ぶ。自然変換 $t : \mathsf{NatTrans}(F, G)$ が満たすべき性質 $(\forall_{x_1,x_2:C} \forall_{f:\mathsf{Map}(x_1,x_2)} G(f) \circ t(x_1) = t(x_2) \circ F(f))$ を t の**自然性** *(naturality)* と言う。

[0067] **7.3*2 演習**: i を階数、$C, D : \mathsf{PreCat}(i)$ を前圏とする。

1 関手 $F : \mathsf{Fun}(C, D)$ に対して、**恒等自然変換** $\mathsf{id}\{F\} : \mathsf{NatTrans}(F, F)$ を $\lambda x.\mathsf{id}\{F(x)\}$ と定義する。$\mathsf{id}\{F\}$ の自然性を確認せよ。
2 関手 $F_1, F_2, F_3 : \mathsf{Fun}(C, D)$ と自然変換 $t_1 : \mathsf{NatTrans}(F_1, F_2)$ と $t_2 : \mathsf{NatTrans}(F_2, F_3)$ に対して、**合成** $t_2 \circ t_1 : \mathsf{NatTrans}(F_1, F_3)$ を $\lambda x.t_2(x) \circ t_1(x)$ と定義する。$t_2 \circ t_1$ の自然性を確認せよ。

[0068] **7.3*3 命題**: 関数外延性を仮定する。i を階数、$C, D : \mathsf{PreCat}(i)$ を前圏、$F, G : \mathsf{Fun}(C, D)$ を関手、$s, t : \mathsf{NatTrans}(F, G)$ を自然変換とすると、同値

$$(s = t) \simeq (\forall_{x:C} s(x) = t(x))$$

を構成できる。

7.3*4 定義: 関数外延性を仮定する。i を階数、C, D : PreCat(i) を前圏とする。前圏 Fun$^{(\mathsf{Cat})}(C, D)$: PreCat(i) を次のように定義する。 [0069]

- Obj $\equiv$ Fun(C, D)
- Map $\equiv \lambda(F, G)$.NatTrans(F, G)
- 恒等射と合成は演習 7.3*2 の通り。
- 残りは命題 7.3*3 から容易に示せる。

7.3*5 命題: 関数外延性を仮定する。i を階数、C, D : PreCat(i) を前圏、F, G : Fun(C, D) を関手、t : NatTrans(F, G) を自然変換とする。次は論理的に同値である。 [006A]

1 任意の $x : C$ に対して、$t(x)$: Map$(F(x), G(x))$ は同型である。
2 t は前圏 Fun$^{(\mathsf{Cat})}(C, D)$ の射として同型である。

証明: 2 から 1 は自明である。

1 から 2 を示す。各 $x : C$ に対して、射 $s(x), u(x)$: Map$(G(x), F(x))$ と同一視 $p(x) : s(x) \circ t(x) =$ id と $q(x) : t(x) \circ u(x) =$ id を得る。命題 7.3*3 より、s と u の自然性を確認すれば十分である。$x_1, x_2 : C$ を対象、f : Map(x_1, x_2) を射とする。$F(f) \circ s(x_1) = s(x_2) \circ G(f)$ を言うには、補題 7.1*6 より $F(f) \circ s(x_1) \circ t(x_1) = s(x_2) \circ G(f) \circ t(x_1)$ を示せば十分で、これは次のように分かる。

$$\begin{aligned} & F(f) \circ s(x_1) \circ t(x_1) \\ = \quad & \{p(x_1)\} \\ & F(f) \\ = \quad & \{(p(x_2))^{-1}\} \\ & s(x_2) \circ t(x_2) \circ F(f) \\ = \quad & \{t \text{ の自然性}\} \\ & s(x_2) \circ G(f) \circ t(x_1) \end{aligned}$$

u の自然性も同様である。 ■

7.3*6 命題: 関数外延性を仮定する。i を階数、C, D : PreCat(i) を前圏とする。D が圏ならば、Fun$^{(\mathsf{Cat})}(C, D)$ は圏である。 [006C]

証明: F : Fun$^{(\mathsf{Cat})}(C, D)$ を対象とする。命題 7.3*5 よりレトラクト $(\sum_{G:\mathsf{Fun}^{(\mathsf{Cat})}(C,D)} F \cong G) \triangleleft (\sum_{G:\mathsf{Fun}(C,D)} \{t : \mathsf{NatTrans}(F, G) \mid \forall_{x:C} \mathsf{IsIso}(t(x))\})$ を得る。後者が可縮であることを示すには、補題 4.1*4 より

$$\sum_{G_0:C.\mathsf{Obj}\to D.\mathsf{Obj}} \sum_{G_1:\prod_{\{x_1,x_2:C\}} \mathsf{Map}(x_1,x_2)\to\mathsf{Map}(G_0(x_1),G_0(x_2))} \{t : \prod_{x:C} F(x) \cong G_0(x) \mid \forall_{x_1,x_2:C} \forall_{f:\mathsf{Map}(x_1,x_2)} G_1(f) \circ t(x_1) = t(x_2) \circ F(f)\}$$

が可縮であることを示せばよい。$\sum_{G_0:C.\mathsf{Obj}\to D.\mathsf{Obj}} \prod_{x:C} F(x) \cong G_0(x)$ の部分は $\prod_{x:C} \sum_{y:D} F(x) \cong y$ のレトラクトなので、命題 3.4*3 と D が圏であるという仮定から可縮である。よって、件の型は $\{G_1 : \prod_{\{x_1,x_2:C\}} \mathsf{Map}(x_1, x_2) \to \mathsf{Map}(F(x_1), F(x_2)) \mid$

$\forall_{x_1,x_2:C}\forall_{f:\mathsf{Map}(x_1,x_2)} G_1(f) \circ \mathsf{id} = \mathsf{id} \circ F(f)\}$ のレトラクトである。この型はさらに $\prod_{x_1,x_2:C}\prod_{f:\mathsf{Map}(x_1,x_2)}\{g : \mathsf{Map}(F(x_1),F(x_2)) \mid g = F(f)\}$ のレトラクトであるが、これは命題 3.4*3 と補題 3.3*1 より可縮である。 ■

$\mathsf{Fun}^{(\mathsf{Cat})}(C,D)$ は**普遍性**でも特徴づけられる (演習 7.3*8)。

[006Q] **7.3*7 定義**: i を階数、$C_1, C_2, D : \mathsf{PreCat}(i)$ を前圏とする。型 $\mathsf{BiFun}(C_1, C_2; D) : \mathcal{U}(i)$ を次のレコード型と定義する。

- $\mathsf{obj} : C_1.\mathsf{Obj} \to C_2.\mathsf{Obj} \to D.\mathsf{Obj}$
- $\mathsf{map}_1 : \prod_{\{x_{11},x_{12}:C_1\}} \mathsf{Map}(x_{11},x_{12}) \to (\prod_{x_2:C_2} \mathsf{Map}(\mathsf{obj}(x_{11},x_2),\mathsf{obj}(x_{12},x_2)))$
- $\mathsf{map}_2 : \prod_{x_1:C_1}\prod_{\{x_{21},x_{22}:C_2\}} \mathsf{Map}(x_{21},x_{22}) \to \mathsf{Map}(\mathsf{obj}(x_1,x_{21}),\mathsf{obj}(x_1,x_{22}))$
- $\_ : \forall_{x_1:C_1}\forall_{x_2:C_2}\mathsf{map}_1(\mathsf{id}\{x_1\},x_2) = \mathsf{id}\{\mathsf{obj}(x_1,x_2)\}$
- $\_ : \forall_{x_{11},x_{12},x_{13}:C_1}\forall_{x_2:C_2}\forall_{f_1:\mathsf{Map}(x_{11},x_{12})}\forall_{f_2:\mathsf{Map}(x_{12},x_{13})}\mathsf{map}_1(f_2 \circ f_1,x_2) = \mathsf{map}_1(f_2,x_2)\circ\mathsf{map}_1(f_1,x_2)$
- $\_ : \forall_{x_1:C_1}\forall_{x_2:C_2}\mathsf{map}_2(x_1,\mathsf{id}\{x_2\}) = \mathsf{id}\{\mathsf{obj}(x_1,x_2)\}$
- $\_ : \forall_{x_1:C_1}\forall_{x_{21},x_{22},x_{23}:C_2}\forall_{f_1:\mathsf{Map}(x_{21},x_{22})}\forall_{f_2:\mathsf{Map}(x_{22},x_{23})}\mathsf{map}_2(x_1,f_2 \circ f_1) = \mathsf{map}_2(x_1,f_2)\circ\mathsf{map}_2(x_1,f_1)$
- $\_ : \forall_{x_{11},x_{12}:C_1}\forall_{x_{21},x_{22}:C_2}\forall_{f_1:\mathsf{Map}(x_{11},x_{12})}\forall_{f_2:\mathsf{Map}(x_{21},x_{22})}\mathsf{map}_1(f_1,x_{22})\circ\mathsf{map}_2(x_{11},f_2) = \mathsf{map}_2(x_{12},f_2)\circ\mathsf{map}_1(f_1,x_{21})$

$\mathsf{BiFun}(C_1,C_2;D)$ の要素を C_1, C_2 から D への**双関手** *(bifunctor)* と呼ぶ。

[006R] **7.3*8 演習**: 関数外延性を仮定する。i を階数、$C, D, X : \mathsf{PreCat}(i)$ を前圏とする。同値

$$\mathsf{Fun}(X, \mathsf{Fun}^{(\mathsf{Cat})}(C,D)) \simeq \mathsf{BiFun}(C, X; D)$$

を構成せよ。

[006D] 7.4 前層

圏にとっての**前層**は、群にとっての作用、環にとっての加群に相当するものである。

[006E] **7.4*1 定義**: i を階数、$C : \mathsf{PreCat}(i)$ を前圏とする。型 $\mathsf{Psh}(C) : \mathcal{U}(\mathsf{succ}(i))$ を次のレコード型と定義する。

- $\mathsf{Carrier} : C.\mathsf{Obj} \to \mathcal{U}(i)$
- $\mathsf{act} : \prod_{\{x_1,x_2:C\}} \mathsf{Carrier}(x_2) \to \mathsf{Map}(x_1,x_2) \to \mathsf{Carrier}(x_1)$
- $\_ : \forall_{x:C}\mathsf{IsSet}(\mathsf{Carrier}(x))$
- $\_ : \forall_{x:C}\forall_{a:\mathsf{Carrier}(x)}\mathsf{act}(a,\mathsf{id}) = a$
- $\_ : \forall_{x_1,x_2,x_3:C}\forall_{f_1:\mathsf{Map}(x_1,x_2)}\forall_{f_2:\mathsf{Map}(x_2,x_3)}\forall_{a:\mathsf{Carrier}(x_3)}\mathsf{act}(a, f_2\circ f_1) = \mathsf{act}(\mathsf{act}(a,f_2),f_1)$

$\mathsf{Psh}(C)$ の要素を C 上の**前層** *(presheaf)* と呼ぶ。

[006F] **7.4*2 記法**: i を階数、$C : \mathsf{PreCat}(i)$ を前圏、$A : \mathsf{Psh}(C)$ を前層とする。

- 対象 $x : C$ に対して、$A.\mathsf{Carrier}(x)$ の代わりに単に $A(x)$ と書く。

- 対象 $x_1, x_2 : C$ と射 $f : \mathsf{Map}(x_1, x_2)$ と要素 $a : A(x_2)$ に対して、要素 $a \cdot f : A(x_1)$ を $A.\mathsf{act}(a, f)$ と定義する。

つまり、C 上の前層は C の対象で添え字付けられた集合の族で、C の射が右から作用しているものである。

7.4*3 定義: i を階数、$C : \mathsf{PreCat}(i)$ を前圏、$A, B : \mathsf{Psh}(C)$ を前層とする。型 $\mathsf{Hom}(A, B) : \mathcal{U}(i)$ を [006G]

$$\{h : \prod_{\{x:C\}} A(x) \to B(x) \mid \forall_{x_1,x_2:C} \forall_{f:\mathsf{Map}(x_1,x_2)} \forall_{a:A(x_2)} h(a \cdot f) = h(a) \cdot f\}$$

と定義する。$\mathsf{Hom}(A, B)$ の要素を**前層の射** *(morphism of presheaves)* と呼ぶ。

7.4*4 演習: i を階数、$C : \mathsf{PreCat}(i)$ を前圏とする。 [006L]

1 前層 $A : \mathsf{Psh}(C)$ に対して、前層の射 $\mathsf{id}\{A\} : \mathsf{Hom}(A, A)$ を $\lambda x.\mathsf{id}\{A(x)\}$ と定義する。これが前層の射の公理を満たすことを確かめよ。
2 前層 $A_1, A_2, A_3 : \mathsf{Psh}(C)$ と前層の射 $f_1 : \mathsf{Hom}(A_1, A_2)$ と $f_2 : \mathsf{Hom}(A_2, A_3)$ に対して、前層の射 $f_2 \circ f_1 : \mathsf{Hom}(A_1, A_3)$ を $\lambda x.f_2\{x\} \circ f_1\{x\}$ と定義する。これが前層の射の公理を満たすことを確かめよ。

前層と前層の射は前圏をなし、関手のなす前圏としても定義できる (演習 7.4*6)。

7.4*5 定義: 関数外延性を仮定する。i を階数、$C : \mathsf{PreCat}(i)$ を前圏とする。前圏 $\mathsf{Psh}^{(\mathsf{Cat})}(C) : \mathcal{U}(\mathsf{succ}(i))$ を次のように定義する。 [006K]

- $\mathsf{Obj} \equiv \mathsf{Psh}(C)$
- $\mathsf{Map} \equiv \lambda(A, B).\mathsf{Hom}(A, B)$
- 恒等射と合成は演習 7.4*4 の通りである。
- 前圏の公理は関数外延性から分かる。

7.4*6 演習: 関数外延性を仮定する。i を階数、$C : \mathsf{PreCat}(i)$ を前圏とする。$\mathsf{Psh}^{(\mathsf{Cat})}(C)$ と $\mathsf{Fun}^{(\mathsf{Cat})}(\mathsf{Op}(C), \mathsf{Set}^{(\mathsf{Cat})}(i))$ の間の前圏の同型を構成せよ。 [006M]

7.4*7 系: 関数外延性と一価性を仮定する。i を階数、$C : \mathsf{PreCat}(i)$ を前圏とすると、$\mathsf{Psh}(C)$ は圏である。 [006O]

7.4*8 系: 関数外延性を仮定する。i を階数、$C : \mathsf{PreCat}(i)$ を前圏、$A, B : \mathsf{Psh}^{(\mathsf{Cat})}(C)$ を前層、$f : \mathsf{Map}(A, B)$ を射とする。次は論理的に同値である。 [0072]

1 f は $\mathsf{Psh}^{(\mathsf{Cat})}(C)$ の射として同型である。
2 任意の対象 $x : C$ に対して、関数 $\lambda a.f(a) : A(x) \to B(x)$ は同値である。

さて、圏論において最も重要な**米田の補題** (定理 7.4*12) を導入する。

7.4*9 定義: i を階数、$C : \mathsf{PreCat}(i)$ を前圏とする。$\mathsf{Map}^{(\mathsf{Fun})}(C) : \mathsf{BiFun}(\mathsf{Op}(C), C; \mathsf{Set}^{(\mathsf{Cat})}(i))$ [006S]
を次のように定義する。

- $\mathsf{obj} \equiv \lambda(x_1, x_2).\mathsf{Map}(x_1, x_2)$
- $\mathsf{map}_1 \equiv \lambda(x_{11}, x_{12}, f, x_2).\lambda h.h \circ f$

- $\mathsf{map}_2 \equiv \lambda(x_1, x_{21}, x_{22}, f).\lambda h.f \circ h$

[006P] **7.4*10 定義**: 関数外延性を仮定する。i を階数、$C : \mathsf{PreCat}(i)$ を前圏とする。**米田埋め込み** *(Yoneda embedding)* $\mathbf{よ}(C) : \mathsf{Fun}(C, \mathsf{Psh}^{(\mathsf{Cat})}(C))$ を演習 7.3*8 と演習 7.4*6 で $\mathsf{Map}^{(\mathsf{Fun})}(C)$ に対応するものと定義する。

$\mathbf{よ}(C)$ が埋め込みと呼ばれるのは系 7.4*13 による。定義から、任意の対象 $x, y : C$ に対して $\mathbf{よ}(C)(x)(y) \equiv \mathsf{Map}(y, x)$ である。特に、$\mathsf{id} : \mathsf{Map}(x, x)$ は $\mathbf{よ}(C)(x)(x)$ の要素とも思える。id をどのように見ているかを区別するために別の表記を導入する。

[006U] **7.4*11 記法**: 関数外延性を仮定する。i を階数、$C : \mathsf{PreCat}(i)$ を前圏、$x : C$ を対象とする。要素 $\mathsf{gen}(x) : \mathbf{よ}(C)(x)(x)$ を $\mathsf{id}\{x\}$ と定義する。

米田の補題が主張するのは、$\mathbf{よ}(C)(x)$ は $\mathsf{gen}(x) : \mathbf{よ}(C)(x)(x)$ で自由に生成された C 上の前層であることである。

[006T] **7.4*12 定理**: 関数外延性を仮定する。i を階数、$C : \mathsf{PreCat}(i)$ を前圏、$x : C$ を対象、$A : \mathsf{Psh}^{(\mathsf{Cat})}(C)$ を前層とする。関数

$$\lambda h.h(\mathsf{gen}(x)) : \mathsf{Map}(\mathbf{よ}(C)(x), A) \to A(x)$$

は同値である。

証明: 射 $h : \mathsf{Map}(\mathbf{よ}(C)(x), A)$ と対象 $y : C$ と要素 $f : \mathbf{よ}(C)(x)(y)$ に対して、同一視

$$\begin{aligned} & h(f) \\ = \quad & \{\text{前圏の公理}\} \\ & h(\mathsf{gen}(x) \cdot f) \\ = \quad & \{\text{前層の公理}\} \\ & h(\mathsf{gen}(x)) \cdot f \end{aligned}$$

を得るので、h は $\mathsf{gen}(x)$ における値のみで決まる。つまり、任意の要素 $a : A(x)$ に対して、$\mathsf{Fiber}(\lambda h.h(\mathsf{gen}(x)), a)$ は命題であることが分かる。この型が要素を持つことを確認するには、$\lambda(y, f).a \cdot f : \prod_{y:C} \mathbf{よ}(C)(x)(y) \to A(y)$ が前層の射であることを確かめればよいが、それは前層の公理から容易である。 ■

[006V] **7.4*13 系**: 関数外延性を仮定する。i を階数、$C : \mathsf{PreCat}(i)$ を前圏とする。米田埋め込み $\mathbf{よ}(C) : \mathsf{Fun}(C, \mathsf{Psh}^{(\mathsf{Cat})}(C))$ は充満忠実である。

証明: $x_1, x_2 : C$ を対象とする。関数 $\lambda f.\mathbf{よ}(C)(f) : \mathsf{Map}(x_1, x_2) \to \mathsf{Map}(\mathbf{よ}(C)(x_1), \mathbf{よ}(C)(x_2))$ と $\lambda h.h(\mathsf{gen}(x_1)) : \mathsf{Map}(\mathbf{よ}(C)(x_1), \mathbf{よ}(C)(x_2)) \to \mathbf{よ}(C)(x_2)(x_1)$ の合成は $\mathsf{Map}(x_1, x_2)$ 上の恒等関数と同一であることが分かる。よって、補題 3.6*2 と定理 7.4*12 から $\lambda f.\mathbf{よ}(C)(f)$ は同値である。 ■

多くの圏論的な概念は、ある前層が**表現可能**であるという性質で定義される。

[0070] **7.4*14 定義**: i を階数、$C : \mathsf{PreCat}(i)$ を前圏、$A : \mathsf{Psh}(C)$ を前層とする。型 $\mathsf{IsRepr}(A) : \mathcal{U}(i)$ を次のレコード型と定義する。

- obj : C
- elem : $A(\mathsf{obj})$
- _ : $\forall_{x:C}\mathsf{IsEquiv}(\lambda(f : \mathsf{Map}(x, \mathsf{obj})).\mathsf{elem} \cdot f)$

$\mathsf{IsRepr}(A)$ の要素がある時、A は**表現可能** *(representable)* であると言う。また、$\mathsf{IsRepr}(A)$ の要素のことを A の**普遍要素** *(universal element)* と呼ぶ。

従来の圏論では、C 上の前層 A の普遍要素は「同型を除いて一意」であることが知られている。ホモトピー型理論においては、C が圏と仮定して普遍要素は一意であること、つまり $\mathsf{IsRepr}(A)$ は命題であることが分かる (命題 7.4*16)。

7.4*15 補題: 関数外延性を仮定する。i を階数、$C : \mathsf{PreCat}(i)$ を前圏、$A : \mathsf{Psh}(C)$ を前層とすると、同値 [0071]

$$\mathsf{IsRepr}(A) \simeq \mathsf{Fiber}^{\cong}(\textbf{よ}(C), A)$$

を構成できる。

証明: 定理 7.4*12 と系 7.4*8 から従う。 ■

7.4*16 命題: 関数外延性を仮定する。i を階数、$C : \mathsf{PreCat}(i)$ を前圏、$A : \mathsf{Psh}(C)$ を前層とする。C が圏ならば、$\mathsf{IsRepr}(A)$ は命題である。 [0073]

証明: 補題 7.4*15 と系 7.4*13 と補題 7.2*9 から従う。 ■

参考文献

[007E]

文献: Alonzo Church. A Formulation of the Simple Theory of Types. *The Journal of Symbolic Logic*, 5(2):56–68, 1940. https://doi.org/10.2307/2266170 [Church--1940-0000]

文献: Michael Hedberg. A coherence theorem for Martin-Löf's type theory. *Journal of Functional Programming*, 8(4):413–436, 1998-07. https://doi.org/10.1017/S0956796898003153 [Hedberg--1998-0000]

文献: The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study, 2013. http://homotopytypetheory.org/book/ [HoTT-Book]

文献: Mark Hovey. *Model categories*. American Mathematical Society, 1999. [Hovey--1999-0000]

文献: Krzysztof Kapulkin & Peter LeFanu Lumsdaine. The simplicial model of univalent foundations (after Voevodsky). *Journal of the European Mathematical Society (JEMS)*, 23(6):2071–2126, 2021. https://doi.org/10.4171/JEMS/1050 https://arxiv.org/abs/1211.2851v5 [Kapulkin--Lumsdaine--2021-0000]

文献: J. Lambek & P. J. Scott. *Introduction to higher order categorical logic*. Cambridge University Press, 1986. [Lambek--Scott--1986-0000]

文献: Peter LeFanu Lumsdaine. Weak ω-categories from intensional type theory. *Logical Methods in Computer Science*, 6(3), 2010-09-17. https://doi.org/10.2168/lmcs-6(3:24)2010 https://arxiv.org/abs/0812.0409v4 [Lumsdaine--2010-0000]

文献: Jacob Lurie. *Higher Topos Theory*. Princeton University Press, 2009. https://www.math.ias.edu/~lurie/papers/HTT.pdf [Lurie--2009-0000]

文献: Per Martin-Löf. An Intuitionistic Theory of Types: Predicative Part. *Studies in Logic and the Foundations of Mathematics*, 80:73–118, 1975. https://doi.org/10.1016/S0049-237X(08)71945-1 [Martin-Loef--1975-0000]

文献: Benjamin C. Pierce. *Types and programming languages*. MIT Press, 2002. [Pierce--2002-0000]

文献: Daniel G. Quillen. *Homotopical algebra*. Springer, 1967. https://doi.org/doi:10.1007/BFb0097438 [Quillen--1967-0000]

文献: Bertrand Russell. Mathematical Logic as Based on the Theory of Types. [Russell--1908-0000]

American Journal of Mathematics, 30(3):222–262, 1908. https://doi.org/10.2307/2369948

[Streicher--1993-0000] **文献**: Thomas Streicher. *Investigations into Intensional Type Theory*. Habilitation Thesis, Ludwig-Maximilians-Universität München, 1993. https://www2.mathematik.tu-darmstadt.de/~streicher/HabilStreicher.pdf

[van-den-Berg--Garner--2011-0000] **文献**: Benno van den Berg & Richard Garner. Types are weak ω-groupoids. *Proceedings of the London Mathematical Society*, 102(2):370–394, 2011. https://doi.org/10.1112/plms/pdq026 https://arxiv.org/abs/0812.0298v2

記法の一覧

[007X]

- $\neg P$ → 記法 4.4*1
- -1 (truncation level) → 定義 4*1
- -2 (truncation level) → 定義 4*1
- $\mathbf{0}$ → 規則 2.6*1
- 0 (自然数) → 規則 2.5*1
- 0 (階数) → 規則 2.1*1
- $\mathbf{1}$ → 規則 2.3*1
- $A \,{}_f{+}_g\, B$ → 規則 5.1*1
- $A(x)$ (前層) → 記法 7.4*2
- $A + B$ → 規則 2.6*2
- $A \to B$ → 定義 2.2*2
- $A \leftrightarrow B$ → 例 2.3*7
- $A \triangleleft\triangleright B$ → 定義 3.1*5
- $A \triangleleft B$ → 定義 3.1*5
- $A \simeq B$ → 定義 3.1*9
- $A \times B$ → 定義 2.3*3
- $\mathsf{BiFun}(C_1, C_2; D)$ → 定義 7.3*7
- $\mathsf{Cat}(i)$ → 定義 7.1*10
- Cocone → 定義 5.2*2
- CoconeOver → 定義 5.2*6
- Cofiber → 定義 5.1*4
- D → 定義 5.3*5
- $F(f)$ (関手を射に適用) → 記法 7.2*2
- $F(x)$ (関手を対象に適用) → 記法 7.2*2
- Fiber → 定義 3.1*7
- $\mathsf{Fiber}^{\simeq}$ → 定義 7.2*8
- Fun → 定義 7.2*1
- $\mathsf{Fun}^{(\mathsf{Cat})}$ → 定義 7.3*4
- $\mathsf{Group}(i)$ → 例 4.6*1
- Hom → 定義 7.4*3
- IsBiinv → 定義 3.6.1*1
- IsCart (余錐) → 定義 5.2*10
- IsCart (スパン) → 定義 5.2*9
- IsCat → 定義 7.1*10
- IsConnMap → 定義 4.5*2
- IsConnected → 定義 4.5*1
- IsContr → 定義 3.1*1
- IsEmb → 定義 4.1*11
- IsEquiv → 定義 3.1*8
- IsEssSurj → 定義 7.2*6
- IsFF → 定義 7.2*6
- IsHAE → 定義 3.6.1*3
- IsIso (前圏の同型) → 定義 7.2*4
- IsIso → 定義 7.1*5
- IsLocal → 定義 5.3*2
- IsProp → 定義 4.1*1
- IsRepr → 定義 7.4*14
- IsSet → 定義 4.2*1
- IsSurj → 定義 4.5*5
- IsTrunc → 定義 4*2
- IsTruncMap → 定義 4*10
- IsUniversal → 定義 5.2*4
- IsWCatEquiv → 定義 7.2*6
- LInv (前圏) → 定義 7.1*5
- LInv → 定義 3.6.1*1
- Loc → 定義 5.3*6
- $\mathsf{LocalGen}(i)$ → 定義 5.3*1
- $\mathsf{Magma}(i)$ → 例 2.3*10
- $\mathsf{Map}^{(\mathsf{Fun})}$ → 定義 7.4*9
- $\mathbb{N}$ → 規則 2.5*1
- NatTrans → 定義 7.3*1
- Op → 例 7.1*4
- $P \wedge Q$ → 記法 4.4*1
- $P \Leftrightarrow Q$ → 記法 4.4*1

- succ(i) (階数) → 規則 2.1*1
- $t_2 \circ t_1$ (自然変換) → 演習 7.3*2
- transport → 定義 2.4*3
- $x_1 \cong x_2$ → 定義 7.1*8
- $\{x : A \mid B(x)\}$ → 記法 4.1*6
- $|a|_n$ → 規則 4.3*1
- $\|A\|_n$ → 規則 4.3*1
- $\lambda(x_1, \ldots, x_n).b$ → 記法 2.2*3
- $\lambda x.b$ → 規則 2.2*1
- $\star$ → 規則 2.3*1
- **よ** → 定義 7.4*10

索引

[007F]