”Proofs and Types” ノート

Rei Tomori

2025 年 7 月 17 日

1 Curry-Howard 同型

1.1 表示的意義

型は現在議論されている対象の種類を表わす．たとえば型 $U \to V$ の対象は U から V への関数，型 $U \times V$ の対象は U の対象と V の対象の順序対といったように．

原子型の意味は重要ではない[*1]．文脈による．

項は我々が Heyting の意味論と自然演繹で用いた 5 つの図式に極めて正確に従う．

1. 型 T の変数 x^T は，型 T の任意の項 t を表わす．ただし x^T は t で置換されるとする．
2. $\langle u, v\rangle$ は u と v の順序対である．
3. $\pi^1 t$ と $\pi^2 t$ はそれぞれ t の第一成分と第二成分への射影である．
4. $\lambda x^U.v$ は型 U の各項 u を $v\,[u/x]$，すなわち x^T を u の略記としたときの v に写す関数である．

表示的には，以下の secondary な等式

$$\begin{aligned}\left\langle \pi^1 t, \pi^2 t\right\rangle &= t\\ \lambda x^U.tx &= t\ (x \notin \mathrm{FV}(t))\end{aligned}$$

に加え，(primary な) 等式がある:

$$\begin{aligned}\pi^1 \langle u, v\rangle &= u\\ pi^2 \langle u, v\rangle &= v\\ (\lambda x^U.v)u &= v\,[u/x]\end{aligned}$$

1.2 操作的な意義

等式規則により，λ 項の表示的意味が定められた．次に，λ 項の操作的意味について考察しよう．

一般に，λ 項はプログラムを表わす．プログラムの目的はその操作的意味を計算することである．(たとえば，$(\lambda x.x + 1)2$ を計算することで，その表示的意味である 5 を得る)

対して，型はプログラムの仕様を定める．プログラムの型が与えられたとき，仕様について何が言えるだろうか．

[*1] 命題変数と思うことにする

仕様に関する言明として「このプログラムは 2 つの整数の和を計算する」を例に取り考える．この言明は十分に詳細だろうか．つまり，この計算がどう行なわれるか主張できるだろうか.[*2]それとも，この言明は詳細すぎるだろうか．たとえば，プログラムが 2 つの整数値を取り，整数値を返すことはいえるだろうか.[*3]

構文に限っていえば，この答は型システムに依存し，明確でない．たとえば，本書で導入する型システム (System F) でこのプログラムに型を付ける場合，$\texttt{int} \to \texttt{int} \to \texttt{int}$ となり，最も明白な情報しか提供しない．対して，Martin-Löf の型システムなどの型体系はプログラムが何を計算するかに関する情報を与える.[*4]つまり，型システムが表示的情報を与える．

より一般的なレベルにおいては，構文的差異を無視することにすると，型は対象を一緒に繋げる命令と見做せる．たとえば，モジュール[*5]を用いてプログラムを構成することを考えよう．モジュールは OOP におけるクラス同様，内部は外から隠蔽されているものとする．モジュールの型は (衝突しないように選ばれた) 全ての可能な引数の型によって決定される．したがって，特にモジュールは同じ型シグネチャを持つ他のモジュールに置換できる．しかし，この見方を数学的に定式化することは難しい．

型 $U_1, \ldots, U_n$ の変数 $x_1, \ldots, x_n$ に依存する型 T の項 t を考えよう．この t は，§3.2 における関数型の解釈，つまり型 $U \to V$ を持つ項 $\lambda x^U.v$ を型 U の各項 u に対して $v\,[u/x]$ を返す関数と解釈するわけにはいかない．むしろ，パラメタ同士を繋ぎ合わせる命令と見做すべきである．モジュールを表わす項には適切な型の入力を挿入する場所がある．たとえば，モジュールにおけるパラメタ x_i の各出現は項 $u_i : U_i$ が挿入される可能性を指す．x_i に u_i が代入される各インスタンスにおいて，u_i は x_i に同時代入される．さらに，項 t も他のモジュールのパラメタとして代入される可能性がある．つまり，モジュールのインスタンスを作るために入力として引数に値が代入され，さらにモジュール自身も他のモジュールの引数として取られうることが分かる．

変数と値を同じ現象の双対的な側面と見做すこの見方によれば，アルゴリズムの実行とは対称的な入出力のプロセスとして理解できる．アルゴリズムはいくつかの引数を取り，更にアルゴリズム自身も (高階関数である) 別のアルゴリズムの引数となりうる．

項の振る舞いを操作的に説明するために，表示的意味を定める等式規則から，左辺から右辺への書き換え規則を作ろう．この書き換え規則は項を計算するプログラム (i.e. インタプリタ) を定めるものと見做せる．プログラムの最終的な結果こそがプログラムの意味を与えるので，操作的意味の理解を深めるために以後正規化について考察する．

1.3 変換

これ以上評価規則を適用できない項を正規形と呼ぶ.

Definition 1. 項 t が正規形であるとは，t の任意の部分項が以下の形でないことである*:*

$\pi^1 \langle u, v \rangle, \pi^2 \langle u, v \rangle, (\lambda x^U.v)u$

[*2] 機械語レベルにおいて?

[*3] いえない．2 整数が浮動小数点値にキャストされる可能性があるため.

[*4] Agda でベクトルの末尾を取る関数 `tail` は,

```
1 tail : {A:Set}{n:Nat}Vec A (suc n) -> Vec A n
2 tail (_::xs) = xs
```

[*5] ML における?

1 ステップの評価を変換という．

Definition 2 (変換)**.** 項 t が t' に変換されるとは，t，t' が以下の条件を満たすことである*:*

1. $\exists u, v. t = \pi^1 \langle u, v \rangle \wedge t' = u$.
2. $\exists u, v. t = \pi^2 \langle u, v \rangle \wedge t' = v$.
3. $\exists u, v, t = (\lambda x^U.v)u \wedge t' = v\,[u/x]$.

t をレデックス (簡約基)，t' をコントラクタムという．以下の進行補題により，redex と contractum の型は常に等しい．

Lemma 1 (進行)**.** 任意の項 t, t' に対して，$t : T$ かつ $t \to t'$ ならば $t' : T$ である．

Proof. t, t' を任意に取り，$t : T$ とする．$t \to t'$ を仮定し，t の形で場合を分ける．

- t が組の射影の形で書けるとき．第一成分の場合を示す．t が $\pi^1 \langle u, v \rangle$ のとき．$v : T'$ とすると，$\pi^1 : T \times T' \to T$ より，$t : T$ である．評価規則より $t' = v : T$ なので従う．
- t が β 簡約基の形で書けるとき．$u : U, v : T$ および v に束縛出現を持たない $x^U : U$ により，t は $(\lambda x^U.v)u$ の形で書ける．評価規則を適用することにより，t のコントラクタム t' は $t' = v\,[u/x]$ であり，型は (x^U の各出現を v で置き換えているので)T のままである．

□

多ステップの変換を簡約という．

Definition 3 (簡約)**.** 項 u が項 v に簡約されるとは，$n \in \mathbb{N}$ と列 $u = t_0, t_1, \ldots, t_{n-1}, t_n = v$ が存在し，$\forall i \in \mathbb{N}, i < n \Rightarrow t_i \to t_{i+1}$ なることであり，$u \rightsquigarrow v$ と書く．

Proposition 1. 二項関係 $(\rightsquigarrow)$ は $(\to)$ の反射推移閉包である．

Proof. 反射性．各項 t に対して簡約列を $t_0 = t$ とすれば即座に従う．推移性．任意の t, u, v に対して $t \rightsquigarrow u, u \rightsquigarrow v$ を仮定する．定義より，リダクション列 $\{t_i\}_{i=0}^m, \{u_i\}_{i=0}^n$ を取れる．そこで，新たなリダクション列 $\{t'_i\}_{i=0}^{m+n}$ を，$\forall i \leq m, t'_i = t_i$，$\forall i \leq n, t'_{m+i} = u_i$ と定めることができる．以上より，$t \rightsquigarrow v$ が従う． □

Definition 4 (正規形)**.** t の正規形とは，$t \rightsquigarrow u$ なる正規形 u である．

以後，STLC において正規形が一意的に存在することを示す．その準備として，頭部正規形と正規形の等価性を示そう．

Definition 5 (頭部正規形)**.**

Lemma 2. 項 t が正規形であることの必要十分条件とは，t が頭部正規形

$$\lambda x_1.\lambda x_2.\ldots\lambda x_n.y\ u_1\ u_2\ \ldots\ u_m \tag{1}$$

であることである．ただし y は x_i の何れかに一致しうる．また，各 j に対して u_j は正規形である．

2 正規化性定理

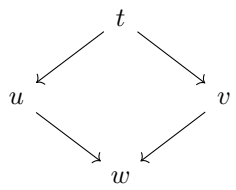
この章は，型付き λ 計算が計算論的に良い振る舞いをする事を保証する 2 つの結果について扱う; **正規化性定理**は正規形の存在性を，*Church-Rosser* **性**はその一意性を保証する．[*6]正規化性定理には 2 つの形がある．

- **弱正規化性定理** (正規化を行なうための評価戦略で，停止するものが存在する．) この主張はこの節で扱う．
- **強正規化性定理** (全ての評価戦略において正規化が停止する); これは §6 で示す．

2.1 Church-Rosser 性

この性質は，正規形の一意性を，その存在性とは独立して主張する．実際，それは型無し λ 計算のような正規化定理の成り立たない体系にとって意味がある．

Thm 1. $t \rightsquigarrow u$ ならば，$u, v \rightsquigarrow w$ であるような w が存在する．



Corollary 1 (A)**.** 項 t は高々 *1* つの正規形を持つ．

Proof. $t \rightsquigarrow u, v$ なる正規形 u, v を任意に取る．このとき，定理より w が存在して $u, v \rightsquigarrow w$.
しかし u, v は正規形なのでそれら自身にのみ簡約される．つまり $u = v, v = w$．ゆえに $u = v$. □

Church-Rosser の証明には多少デリケートな部分がある (少なくとも力ずくで示そうとする場合)．この性質は多種多様な体系で成り立ち，証明は常にほぼ一緒である．

Church-Rosser から即座に従う補題として，体系の健全性がある．

[*6] Church-Rosser は示さない．Barendregt などを見よ．