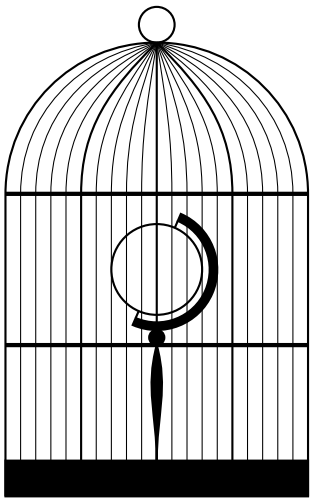

Wir müssen wissen. Wir werden wissen.

An Introduction to Lambda Calculus

ラ　ム　ダ　計　算　入　門

川向薫

はしがき

本書はひとつのラムダ計算の入門書です．第 1 章では型なしラムダ計算を扱い，第 2 章では単純型つきラムダ計算を扱いまして，第 3 章では単純型つきラムダ計算の型推論を扱います．

目次

第 1 章　型なしラムダ計算

§1.1.　構文

変数 (*variable*) をつぎのように定義します．

$$\begin{array}{rcl} x_1, x_2, \cdots & ::= & \mathrm{x}_1 \\ & | & \mathrm{x}_2 \\ & | & \vdots \end{array}$$

項 (*term*) をつぎのように帰納的に定義します．

$$\begin{array}{rcl} e_1, e_2, \cdots & ::= & x_1 \\ & | & (e_1\, e_2) \\ & | & (\lambda x_1\, e_1) \end{array}$$

x_1 や x_2 は (具体的な) 変数ですが x_1 や x_2 は変数を表すメタ変数です．e_1 や e_2 は項を表すメタ変数です．メタ変数が出現する項をメタ項と呼びます．たとえば $(\lambda \mathrm{x}_1 (\lambda \mathrm{x}_2 \mathrm{x}_1))$ は (具体的な) 項ですが $(\lambda x_1 (\lambda x_2 x_1))$ はメタ項です．(具体的な) 項を書くことはほとんどありませんので，メタ項を指して単に項と呼びますが，(具体的な) 項を指しているわけではないことに注意してください．このような区別をしばしばメタ (*meta*) レベルや対象 (*object*) レベルと呼びます．

e_1 の構造についての帰納法を e_1 の定義による帰納法と呼び，まず基礎ケース (*base case*) すなわち $e_1 = x_1$ について命題 $\mathsf{P}(e_1)$ が成りたつことを証明し，帰納ステップ (*induction step*) すなわち e_2, e_3, e_4 について命題 $\mathsf{P}(e_2), \mathsf{P}(e_3), \mathsf{P}(e_4)$ が成りたつという仮定——帰納法の仮定 (*induction hypothesis*)——のもとで $e_1 = (e_2\, e_3), (\lambda x_2\, e_4)$ について命題 $\mathsf{P}(e_1)$ が成りたつことを証明します．それで命題 $\forall e_5 (\mathsf{P}(e_5))$ が証明されたことになります．再帰に似ていますが，Coq の Fixpoint のように “構造的に小さな” ものでしか帰納法の仮定は成立しません．説明のためこれを推論規則 (*rule of inference*) として書きくだすと，つぎのようになります．

$$\frac{\Gamma \vdash \mathsf{P}(x_1) \quad \Gamma, \mathsf{P}(e_2), \mathsf{P}(e_3) \vdash \mathsf{P}(e_2\, e_3) \quad \Gamma, \mathsf{P}(e_4) \vdash (\lambda x_1\, e_4)}{\Gamma \vdash \forall e_5 (\mathsf{P}(e_5))}$$

§1.2.　長さ

項の長さ (*length*) をつぎのように帰納的に定義します．

$$\mathtt{length}(e_1) \coloneqq \begin{cases} 1 & e_1 = x_1 \\ \mathtt{length}(e_2) + \mathtt{length}(e_3) + 1 & e_1 = (e_2\, e_2) \\ \mathtt{length}(e_2) + 1 & e_1 = (\lambda x e_2) \end{cases}$$

$\mathtt{length}(e_1) = \mathrm{n}$ についての帰納法を e_1 の長さに関する帰納法と呼びます．長さに関する帰納法では，まず $\mathrm{n} = 1$ すなわち $e_1 = x_1$ について命題が成りたつことを証明し，$\mathrm{n} = 1, \cdots, \mathrm{k}$ について命題が成りたつと仮定

したうえで $\mathrm{n} = \mathrm{k} + 1$ について命題が成りたつことを証明します．長さに関する帰納法では構造的に小さくなっていなくても，長さが同じなら帰納法の仮定が成立すると考えます．言いかえれば帰納法の仮定で $\mathrm{P}(e_1)$ が成立し，なおかつ $\mathbf{length}(e_2) \leqq \mathbf{length}(e_1)$ ならば $\mathrm{P}(e_2)$ も成立するということです．定義による帰納法では，長さが同じだからといって帰納法の仮定が成立するというわけにはいきません．説明のためこれを推論規則として書きくだすと，つぎのようになります．

$$\frac{\Gamma, \mathbf{length}(x_1) = 1 \vdash \mathrm{P}(1) \quad \Gamma, \mathbf{length}(x_1) = \mathrm{k}+1, \mathrm{P}(1), \cdots, \mathrm{P}(\mathrm{k}) \vdash \mathrm{P}(\mathrm{k}+1)}{\Gamma \vdash \forall \mathrm{n}(\mathrm{P}(\mathrm{n}))}$$

§1.3. 自由変数

項 e_1 に含まれる自由変数の集合 $\mathrm{FV}(e_1)$ をつぎのように帰納的に定義します．

$$\mathrm{FV}(e_1) \coloneqq \begin{cases} \{x_1\} & e_1 = x_1 \\ \mathrm{FV}(e_2) \cup \mathrm{FV}(e_3) & e_1 = (e_2\, e_3) \\ \mathrm{FV}(e_2) \setminus \{x_1\} & e_1 = (\lambda x_1\, e_2) \end{cases}$$

§1.4. 束縛変数

項 e_1 に含まれる自由変数の集合 $\mathrm{BV}(e_1)$ をつぎのように帰納的に定義します．

$$\mathrm{BV}(e_1) \coloneqq \begin{cases} \emptyset & e_1 = x_1 \\ \mathrm{BV}(e_2) \cup \mathrm{BV}(e_3) & e_1 = (e_2\, e_3) \\ \mathrm{BV}(e_2) \cup \{x_1\} & e_1 = (\lambda x_1\, e_2) \end{cases}$$

§1.5. 変数

項 e_1 に含まれる変数の集合 $\mathrm{V}(e_1)$ をつぎのように定義します．

$$\mathrm{V}(e_1) \coloneqq \mathrm{FV}(e_1) \cup \mathrm{BV}(e_1)$$

§1.6. 代入

代入 (*substitution*)$[e_1/x_1]e_2$ をつぎのように帰納的に定義します．

$$[e_1/x_1]e_2 \coloneqq \begin{cases} e_1 & e_2 = x_2 \wedge x_1 = x_2 \\ e_2 & e_2 = x_2 \wedge x_1 \neq x_2 \\ ([e_1/x_1]e_3[e_1/x_1]e_4) & e_2 = (e_3\, e_4) \\ e_2 & e_2 = (\lambda x_2\, e_3) \wedge x_1 = x_2 \\ (\lambda x_3[e_1/x_1][x_3/x_2]e_3) & e_2 = (\lambda x_2\, e_3) \wedge x_1 \neq x_2 \wedge x_1 \neq x_3 \wedge x_2 \neq x_3 \wedge x_3 \notin \mathrm{V}(e_1) \wedge x_3 \notin \mathrm{V}(e_2) \end{cases}$$

補題 1.6.1. $\forall x_1 \forall x_2 \forall e_1(\mathbf{length}([x_1/x_2]e_1) = \mathbf{length}(e_1))$.

証明. x_1, x_2, e_1 を固定し，$[x_1/x_2]e_1$ の定義による帰納法で証明する．

- $[x_1/x_2]e_1 = x_1$ の場合．$e_1 = x_2$ なので，$\mathbf{length}([x_1/x_2]e_1) = 1 = \mathbf{length}(e_1)$.

- $[x_1/x_2]e_1 = e_1$ の場合．明らかに $\mathtt{length}([x_1/x_2]e_1) = \mathtt{length}(e_1)$.
- $[x_1/x_2]e_1 = ([x_1/x_2]e_2[x_1/x_2]e_3)$ の場合．帰納法の仮定:

$$\begin{aligned}\mathtt{length}([x_1/x_2]e_2) &= \mathtt{length}(e_2)\\ \mathtt{length}([x_1/x_2]e_3) &= \mathtt{length}(e_3)\end{aligned}$$

したがって，

$$\begin{aligned}\mathtt{length}([x_1/x_2]e_1) &= \mathtt{length}([x_1/x_2]e_2[x_1/x_2]e_3)\\ &= \mathtt{length}([x_1/x_2]e_2) + \mathtt{length}([x_1/x_2]e_3) + 1\\ &= \mathtt{length}(e_2) + \mathtt{length}(e_3) + 1\\ &= \mathtt{length}(e_2\,e_3)\\ &= \mathtt{length}(e_1)\end{aligned}$$

- $[x_1/x_2]e_1 = (\lambda x_4[x_1/x_2][x_4/x_3]e_2)$ の場合．帰納法の仮定:

$$\begin{aligned}\mathtt{length}([x_1/x_2][x_4/x_3]e_2) &= \mathtt{length}([x_4/x_3]e_2)\\ \mathtt{length}([x_4/x_3]e_2) &= \mathtt{length}(e_2)\end{aligned}$$

したがって，

$$\begin{aligned}\mathtt{length}([x_1/x_2]e_1) &= \mathtt{length}(\lambda x_4[x_1/x_2][x_4/x_3]e_2)\\ &= \mathtt{length}([x_1/x_2][x_4/x_3]e_2) + 1\\ &= \mathtt{length}([x_4/x_3]e_2) + 1\\ &= \mathtt{length}(e_2) + 1\\ &= \mathtt{length}(e_1)\end{aligned}$$

□

補題 1.6.2. $\forall x_1 \forall x_2 \forall x_3 \forall e_1([x_1/x_2][x_2/x_3]e_1 = [x_1/x_3]e_1)$.

証明. 証明は簡単． □

§1.7. α-変換

α-同値 (*α-congruence*) $e_1 \equiv e_2$ をつぎのように帰納的に定義します．

$$e_1 \equiv e_2 \coloneqq \begin{cases} x_1 = x_2 & e_1 = x_1 \wedge e_2 = x_2 \\ e_3 \equiv e_5 \wedge e_4 \equiv e_6 & e_1 = (e_3\,e_4) \wedge e_2 = (e_5\,e_6) \\ [x_3/x_1]e_3 \equiv [x_3/x_2]e_4 & e_1 = (\lambda x_1\, e_3) \wedge e_2 = (\lambda x_2\, e_4) \wedge x_3 \notin \mathrm{V}(e_1) \wedge x_3 \notin \mathrm{V}(e_2) \end{cases}$$

$e_1 \equiv e_2$ が定義されないことを単に $e_1 \not\equiv e_2$ と書きます．$e_1 \equiv e_2$ は簡単には変数名の違いを無視した同値関係です．たとえば $(\lambda \mathrm{x}_1 \mathrm{x}_1) \not\equiv (\lambda \mathrm{x}_2 \mathrm{x}_2)$ ですが $(\lambda \mathrm{x}_1 \mathrm{x}_1) \equiv (\lambda \mathrm{x}_2 \mathrm{x}_2)$ です．

補題 1.7.1. $\forall e_1 \forall e_2(e_1 \equiv e_2 \to \mathtt{length}(e_1) = \mathtt{length}(e_2))$.

証明. 証明は簡単． □

簡単のため $e_1 \equiv e_2$ のとき，e_1, e_2 両方の長さに関する帰納法で証明することがあります．2 重帰納法ではありませんので注意してください．帰納法は基本的にひとつの自然数や構造に対してしかできませんが，項

の n-組 (*n-tuple*)$(e_1, \cdots, e_n)$ というひとつの構造を考えれば，$e_1 \equiv e_2$かつ$\cdots$かつ$e_{n-1} \equiv e_n$ という前提なら $\mathrm{length}(e_1) = \cdots = \mathrm{length}(e_n)$ ですから，

$$\mathrm{length}_n(e_1, \cdots, e_n) \coloneqq \mathrm{length}(e_1)$$

を与えれば，項の n-組の長さに関する帰納法ができます．

補題 1.7.2. $\forall f \forall e_1 \forall e_2 (e_1 \equiv e_2 \leftrightarrows f(e_1) \equiv f(e_2))$.

証明. 証明は簡単． □

補題 1.7.3. α-同値は反射的である．すなわち，$\forall e_1 (e_1 \equiv e_1)$.

証明. e_1を固定し，e_1 の長さに関する帰納法で証明する．

- $\mathrm{length}(e_1) = 1$ の場合．
  - $e_1 = x_1$ の場合．$x_1 = x_1$ なので，α-同値の定義により $e_1 \equiv e_1$ である．
- $\mathrm{length}(e_1) = k + 1$ の場合．
  - $e_1 = (e_2 e_3)$ の場合．帰納法の仮定:

    $$e_2 \equiv e_2$$
    $$e_3 \equiv e_3$$

    したがって $e_2 \equiv e_2 \wedge e_3 \equiv e_3$ なので，α-同値の定義により $e_1 \equiv e_1$ である．
  - $e_1 = (\lambda x_1 e_2)$ の場合．帰納法の仮定:

    $$e_2 \equiv e_2$$

    また補題 1.6.1 により帰納法の仮定は $[x_2/x_1]e_2 \equiv [x_2/x_1]e_2$ と変形できるから，α-同値の定義により $e_1 \equiv e_1$ である．

□

補題 1.7.4. α-同値は対称的である．すなわち，$\forall e_1 \forall e_2 (e_1 \equiv e_2 \to e_2 \equiv e_1)$.

証明. e_1, e_2 を固定し，e_1, e_2 の長さに関する帰納法で証明する．前提:

$$e_1 \equiv e_2$$

- $\mathrm{length}(e_1) = \mathrm{length}(e_2) = 1$ の場合．
  - $e_1 = x_1, e_2 = x_2$ の場合．前提を計算すれば $x_1 = x_2$ となるので，$x_2 = x_1$．したがって α-同値の定義により $e_2 \equiv e_1$ である．
- $\mathrm{length}(e_1) = \mathrm{length}(e_2) = k + 1$ の場合．
  - $e_1 = (e_3 e_4), e_2 = (e_5 e_6)$ の場合．帰納法の仮定:

    $$e_3 \equiv e_5 \to e_5 \equiv e_3$$
    $$e_4 \equiv e_6 \to e_6 \equiv e_4$$

    また前提を計算すれば $e_3 \equiv e_5 \wedge e_4 \equiv e_6$ となるので，$e_5 \equiv e_3 \wedge e_6 \equiv e_4$ であり，α-同値の定義により $e_2 \equiv e_1$ である．

- $e_1 = (\lambda x_1 e_3), e_2 = (\lambda x_2 e_4)$ の場合．帰納法の仮定:

$$e_3 \equiv e_4 \to e_4 \equiv e_3$$

また前提を計算すれば $[x_3/x_1]e_3 \equiv [x_3/x_2]e_4$ となり，補題 1.6.1 により帰納法の仮定は

$$[x_3/x_1]e_3 \equiv [x_3/x_2]e_4 \to [x_3/x_2]e_4 \equiv [x_3/x_1]e_3$$

と変形できるから，$[x_3/x_2]e_4 \equiv [x_3/x_1]e_3$ である．したがって α-同値の定義により $e_2 \equiv e_1$ である．

- $\mathrm{length}(e_1) \neq \mathrm{length}(e_2)$ の場合．前提と補題 1.7.1 が矛盾するので，このような場合は起こりえない．

□

今回は最後の $\mathrm{length}(e_1) \neq \mathrm{length}(e_2)$ の場合を明示しましたが，今後は省略します．

補題 1.7.5. α-同値は推移的である．すなわち，$\forall e_1 \forall e_2 \forall e_3 (e_1 \equiv e_2 \to e_2 \equiv e_3 \to e_1 \equiv e_3)$.

証明. e_1, e_2, e_3 を固定し，e_1, e_2, e_3 の長さに関する帰納法で証明する．前提:

$$e_1 \equiv e_2$$
$$e_2 \equiv e_3$$

- $\mathrm{length}(e_1) = \mathrm{length}(e_2) = \mathrm{length}(e_3) = 1$ の場合．
  - $e_1 = x_1, e_2 = x_2, e_3 = x_3$ の場合．前提を計算すれば $x_1 = x_2$ かつ $x_2 = x_3$ となる．したがって $x_1 = x_3$ なので，α-同値の定義により $e_1 \equiv e_3$ である．
- $\mathrm{length}(e_1) = \mathrm{length}(e_2) = \mathrm{length}(e_3) = \mathrm{k} + 1$ の場合．
  - $e_1 = (e_4 e_5), e_2 = (e_6 e_7), e_3 = (e_8 e_9)$ の場合．帰納法の仮定:

$$e_4 \equiv e_6 \to e_6 \equiv e_8 \to e_4 \equiv e_8$$
$$e_5 \equiv e_7 \to e_7 \equiv e_9 \to e_5 \equiv e_9$$

また前提を計算すれば $e_4 \equiv e_6 \wedge e_5 \equiv e_7$ かつ $e_6 \equiv e_8 \wedge e_7 \equiv e_9$ となるので，$e_4 \equiv e_8 \wedge e_5 \equiv e_9$ であり，α-同値の定義により $e_1 \equiv e_3$ である．

  - $e_1 = (\lambda x_1 e_4), e_2 = (\lambda x_2 e_5), e_3 = (\lambda x_3 e_6)$ の場合．帰納法の仮定:

$$e_4 \equiv e_5 \to e_5 \equiv e_6 \to e_4 \equiv e_6$$

また前提を計算すれば $[x_4/x_1]e_4 \equiv [x_4/x_2]e_5$ かつ $[x_5/x_2]e_5 \equiv [x_5/x_3]e_6$ となり，補題 1.7.2 により $[x_6/x_4][x_4/x_1]e_4 \equiv [x_6/x_4][x_4/x_2]e_5$ かつ $[x_6/x_5][x_5/x_2]e_5 \equiv [x_6/x_5][x_5/x_3]e_6$ すなわち補題 1.6.2 により $[x_6/x_1]e_4 \equiv [x_6/x_2]e_5$ かつ $[x_6/x_2]e_5 \equiv [x_6/x_3]e_6$ である．補題 1.6.1 によりにより帰納法の仮定は

$$[x_6/x_1]e_4 \equiv [x_6/x_2]e_5 \to [x_6/x_2]e_5 \equiv [x_6/x_3]e_6 \to [x_6/x_1]e_4 \equiv [x_6/x_3]e_6$$

と変形できるから，$[x_6/x_1]e_4 \equiv [x_6/x_3]e_6$ である．ゆえに α-同値の定義により $e_1 \equiv e_3$ である．

□

$e_1 \equiv e_2$ のとき，e_1 を e_2 で置きかえることを α-変換 (*α-conversion*) と呼びます．以後，$e_1 \equiv e_2$ は通常の等号と同様に，すなわち $e_1 \equiv e_2$ のときはいつでも e_1 を e_2 で置きかえてもよいと考えます．

§1.8. β-変形

e_1, e_2 が項ならば $e_1 \triangleright_1 e_2$ と $e_1 \triangleright e_2$ を式 (*expression*) と呼びます．1 ステップの β-変形 (β-*reduction*) の推論規則 (*rule of inference*) をつぎのように定義します．

$$\frac{}{((\lambda x_1\, e_1)\, e_2) \triangleright_1 [e_1/x_1]e_2}$$

$$\frac{e_1 \triangleright_1 e_2}{(e_1\, e_3) \triangleright_1 (e_2\, e_3)}$$

$$\frac{e_1 \triangleright_1 e_2}{(e_3\, e_1) \triangleright_1 (e_3\, e_2)}$$

$$\frac{e_1 \triangleright_1 e_2}{(\lambda x_1\, e_1) \triangleright_1 (\lambda x_1\, e_2)}$$

さらに n ステップの β-変形の推論規則をつぎのように定義します．

$$\frac{}{e_1 \triangleright e_1}$$

$$\frac{e_1 \triangleright_1 e_2 \quad e_2 \triangleright e_3}{e_1 \triangleright e_3}$$

横線より上の部分を前提 (*premise*)，下の部分を結論 (*conclusion*) と呼びます．推論規則をいくつか組みあわせた図を推論図 (*deduction*) と呼びます．推論図をつぎのように帰納的に定義します．

- 式 A は推論図である．この形の推論図を仮定 (*assumption*) と呼び，前提は $\{\mathrm{A}\}$ で結論は A である．
- $\Pi_1, \cdots, \Pi_n$ を推論図とし，それぞれの結論を $\mathrm{A}_1, \cdots, \mathrm{A}_n$ とする．$\mathrm{A}_1, \cdots, \mathrm{A}_n$ を前提とする推論規則が存在し，その結論が B ならば，

  $$\frac{\Pi_1 \quad \cdots \quad \Pi_n}{\mathrm{B}}$$

  は推論図である．この形の推論図の前提は Π_1の前提 $\cup \cdots \cup$ Π_nの前提 であり，結論は B である．

推論図の概念は，帰納的に定義するより例示したほうがわかりやすいかもしれません．つぎの推論図は，$(((\lambda x_1(\lambda x_2 x_1))x_3)x_4)$ という項が n ステップの β-変形で x_3 になるということを表しています．読みやすくするために代入はあらかじめ等価な項で置きかえてありますが，厳密には代入が出現する場所もあることに注意してください．

例 1.8.1.

$$\dfrac{\dfrac{\dfrac{}{((\lambda x_1(\lambda x_2 x_1))x_3) \triangleright_1 (\lambda x_2 x_3)}}{(((\lambda x_1(\lambda x_2 x_1))x_3)x_4) \triangleright_1 ((\lambda x_2 x_3)x_4)} \quad \dfrac{\dfrac{}{((\lambda x_2 x_3)x_4) \triangleright_1 x_3} \quad \dfrac{}{x_3 \triangleright x_3}}{((\lambda x_2 x_3)x_4) \triangleright x_3}}{(((\lambda x_1(\lambda x_2 x_1))x_3)x_4) \triangleright x_3}$$

このように 1 ステップの β-変形と n ステップの β-変形を別々に定義する意味論を Small Step Semantics と呼びます．Small Step Semantics の関々は 1 ステップの β-変形にありますので，n ステップの β-変形を定義しないこともあります．1 ステップの β-変形を定義せず，いきなり n ステップの β-変形を定義する意味論を Big Step Semantics と呼びます．Big Step Semantics では $\Downarrow$ という記号をもちいることが一般的です．Big Step Semantics のほうがより実際の実装に近い意味論なのでプログラム言語の形式化ではよくもちいられますが，理論的な計算模型を考える場合には Small Step Semantics がよくもちいられます．

Π の前提が $\{\mathrm{A}_1, \cdots, \mathrm{A}_n\}$ で結論が B のとき, Π を B に至る推論図と呼び, $\Gamma = \mathrm{A}_1, \cdots, \mathrm{A}_n, \mathrm{A}_{n+1}, \cdots, \mathrm{A}_{n+m}$ を Π の仮定と呼びます．仮定は集合と同様に，順序や重複は気にしませんし，前提にない式が仮定に含まれて

いてもかまいません．さらに $\Gamma \vdash \mathrm{B}$ と書き，仮定 Γ から B がでると読みます．とくに $\mathrm{n} = 0$ かつ $\mathrm{m} = 0$ すなわち Π の仮定が空のとき Π を証明図と呼び，B は証明可能 (*provable*) であるといいます．$e_1 \triangleright_1 e_2$ が証明できるような e_2 が存在しないとき，e_1 は β-正規形 (β-*redux*) であるといいます．$e_1 \triangleright e_2$ が証明でき，かつ e_2 が正規形であるとき，e_1 は正規形をもつといいます．正規形をもたない項が存在することに注意してください．たとえば，$((\lambda \mathrm{x}_0 \mathrm{x}_0)(\lambda \mathrm{x}_0 \mathrm{x}_0))$ は正規形をもちません．推論図の長さ (*length*) をつぎのように帰納的に定義します．

$$\mathrm{length}(\Pi) ::= \begin{cases} 1 & \Pi = \mathrm{A} \\ \mathrm{length}(\Pi_1) + \cdots + \mathrm{length}(\Pi_\mathrm{n}) + 1 & \Pi = \dfrac{\Pi_1 \quad \cdots \quad \Pi_\mathrm{n}}{\mathrm{A}} \end{cases}$$

補題 1.8.1. $(\vdash e_1 \triangleright_1 e_2) \to (\vdash e_1 \triangleright e_2)$.

証明.

$$\dfrac{\dfrac{}{e_1 \triangleright_1 e_2} \quad \dfrac{}{e_2 \triangleright e_2}}{e_1 \triangleright e_2}$$

□

補題 1.8.2. n ステップの β-変形は推移的である．すなわち，$(\vdash e_1 \triangleright e_2) \to (\vdash e_2 \triangleright e_3) \to (\vdash e_1 \triangleright e_3)$.

証明. $e_1 \triangleright e_2$ に至る証明図を Π とし，Π の長さに関する帰納法で証明する．

- $\mathrm{length}(\Pi) = 1$ の場合．$e_1 \triangleright e_3$ に至る証明図は

  $$\dfrac{\dfrac{}{e_1 \triangleright e_1} \quad \dfrac{\Pi_1 \quad \cdots \quad \Pi_\mathrm{n}}{e_1 \triangleright e_3}}{e_1 \triangleright e_3}$$

  という形をしているから，Π はいわば “むだな” 証明図なので，単にはぶけば

  $$\dfrac{\Pi_1 \quad \cdots \quad \Pi_\mathrm{n}}{e_1 \triangleright e_3}$$

  と変形できるからよい．
- $\mathrm{length}(\Pi) = \mathrm{k} + 1$ の場合．$e_1 \triangleright e_3$ に至る証明図は

  $$\dfrac{\dfrac{\Pi_1 \quad \Pi_2}{e_1 \triangleright e_2} \quad \dfrac{\Pi_3 \quad \cdots \quad \Pi_{\mathrm{n}+2}}{e_2 \triangleright e_3}}{e_1 \triangleright e_3}$$

  という形をしていて，Π_1 の結論は $e_1 \triangleright_1 e_4$ で，Π_2 の結論は $e_4 \triangleright e_2$ である．帰納法の仮定により $(\vdash e_4 \triangleright e_2) \to (\vdash e_2 \triangleright e_3) \to (\vdash e_4 \triangleright e_3)$ だから，$e_1 \triangleright e_3$ に至る証明図は

  $$\dfrac{\Pi_1 \quad \dfrac{\Pi_2 \quad \dfrac{\Pi_3 \quad \cdots \quad \Pi_{\mathrm{n}+2}}{e_2 \triangleright e_3}}{e_4 \triangleright e_3}}{e_1 \triangleright e_3}$$

  と変形できるからよい．

□

補題 1.8.2 により，$\vdash e_1 \triangleright e_2$ かつ $\vdash e_2 \triangleright e_3$ ならば

$$\dfrac{e_1 \triangleright e_2 \quad e_2 \triangleright e_3}{e_1 \triangleright e_3}$$

という推論規則を β-変形の体系に加えても，証明できる式が増えないことがわかります．このような推論規則を許容規則 (*admissible rule*) といいます．このような許容規則は，β-変形の推論規則のみを使って直接導くことはできません．先の証明では証明図自体の変形をともなっています．いわば Lisp のマクロのようなメタプログラミングによる証明をしているわけです．推論規則のみを使って直接に導ける推論規則は派生規則 (*derivable rule*) といいます．より一般には

$$\mathrm{A}_1, \cdots, \mathrm{A}_n \vdash \mathrm{B}$$

のとき

$$\frac{\mathrm{A}_1 \quad \cdots \quad \mathrm{A}_n}{\mathrm{B}}$$

を派生規則といい，

$$(\vdash \mathrm{A}_1) \to \cdots \to (\vdash \mathrm{A}_n) \to (\vdash \mathrm{B})$$

のとき

$$\frac{\mathrm{A}_1 \quad \cdots \quad \mathrm{A}_n}{\mathrm{B}}$$

を許容規則といいます．派生規則ならば許容規則です．先に述べたように派生規則や許容規則を体系に加えても証明できる式は増えませんので，体系の性質を壊さずに短く書けるようになります．

§1.9. 並行簡約

e_1, e_2 が項ならば $e_1 \sqsupset_1 e_2$ を式 (*expression*) と呼びます．1 ステップの並行簡約 (*parallel reduction*) の推論規則 (*rule of inference*) をつぎのように定義します．

$$\frac{}{x_1 \sqsupset_1 x_1}$$

$$\frac{e_1 \sqsupset_1 e_2 \quad e_3 \sqsupset_1 e_4}{(e_1\, e_3) \sqsupset_1 (e_2\, e_4)}$$

$$\frac{e_1 \sqsupset_1 e_2}{(\lambda x_1\, e_1) \sqsupset_1 (\lambda x_1\, e_2)}$$

$$\frac{e_1 \sqsupset_1 e_2 \quad e_3 \sqsupset_1 e_2}{((\lambda x_1\, e_1)\, e_3) \sqsupset [e_2/x_1]\, e_4}$$

§1.10. 合流性定理

定理 1.10.1. $\forall e_1 \forall e_2 \forall e_3 \exists e_4 ((\vdash e_1 \triangleright e_2) \to (\vdash e_1 \triangleright e_3) \to (\vdash e_2 \triangleright e_4) \wedge (\vdash e_3 \triangleright e_4))$

第 2 章　単純型つきラムダ計算

第 3 章　単純型つきラムダ計算の型推論

Although you do not know that the bridge will fall if there are no contradictions, yet it is almost certain that if there are contradictions it will go wrong somewhere.