7.12 体系の「性質」

型付きラムダ計算のような計算体系に対して，成立が期待される主な性質には以下の 3 つがある[11].

- 型システムの健全性…型が整合的なプログラムを計算するとき，計算の途中の任意の時点 (計算終了後を含む) で，型が整合的であるという性質.
- 合流性…どのような順番で計算を行ったとしても，計算結果が一意的であるという性質.
- 停止性 (強い停止性，弱い停止性)…

  どのような順番で計算を行ったとしても，計算がいつか必ず停止するという性質 (強い停止性)，ある (適当な) 順番で計算を行ったときに，計算がいつか必ず停止するという性質 (弱い停止性).

これらの性質が成立することにより，どのような「良い」ことがいえるかは後の章で考えることにして，さしあたり，これらの性質を数学的に厳密に述べ，証明を考えることにする．なお，証明の詳細について完全に理解することは本講義の範囲をこえるので，以下で述べる定理の主張のみを理解する程度でよい (どういう定理かは理解すべきであるが，証明は理解できなくてもよい．)

7.13 型システムの健全性

定理 7 (サブジェクトリダクション (Subject Reduction)) $\Gamma \vdash M : A$ が *(*前に述べた推論規則により*)* 導出可能であり，$M \to^* N$ であれば，$\Gamma \vdash N : A$ が導出可能である.

ここで $M \to^* N$ は，$M \to N$ の反射的・推移的閉包であり，0 ステップ以上の計算により M が N に変形できることを意味している．また，以下の証明で $M \to_k^* N$ という記法を使うが，これは k ステップの計算で M が N に変形できることを意味している.

証明の概要: まず，以下の性質を $\phi(k)$ とする．これは，ちょうど k ステップで M から N に簡約 (計算) 可能である場合に限定した定理の内容である.

$\Gamma \vdash M : A$ が導出可能であり，$M \to_k^* N$ であれば，$\Gamma \vdash N : A$ が導出可能である.

「任意の自然数 k に対して $\phi(k)$ が成立する」ことを証明できれば定理は証明されたことになる．これを，k に関する数学的帰納法で証明する．すなわち，以下の 2 つを証明できればよい.

- $\phi(0)$ を証明する.
- $\phi(k)$ が成立することを仮定して $\phi(k+1)$ を証明する.

$\phi(0)$ は，この場合，trivial に成立する内容である.「$\phi(k)$ ならば $\phi(k+1)$」 を証明するためには，以下の性質 (これを性質 (*) と呼ぶことにする) が証明できればよい.

$\Gamma \vdash M : A$ が導出可能であり，$M \to N$ であれば，$\Gamma \vdash N : A$ が導出可能である.

これはちょうど 1 ステップの計算で型が保たれる，という性質である．この性質を証明するために，もう一度帰納法を使う．今度は自然数に関する帰納法ではなく,「$\Gamma \vdash M : A$ の導出」の構成に関する帰納法 (木に関する帰納法) を使う．この帰納法を詳しく書くと，以下のようになる.

[11] ここでの説明は，わかりやすさのために数学的厳密さを若干犠牲にしている.

- $\Gamma \vdash M : A$ の最終導出規則が var 規則であり，そのすべての部分導出に対して性質 (*) が成立し，さらに，$M \to N$ であれば，$\Gamma \vdash N : A$ が導出可能である．
- 上記の var を λ で置きかえたもの
- 上記の var を apply で置きかえたもの
- 上記の var を pair で置きかえたもの
- 上記の var を left で置きかえたもの
- 上記の var を right で置きかえたもの
- 上記の var を inl で置きかえたもの
- 上記の var を inr で置きかえたもの
- 上記の var を case で置きかえたもの

「これら 9 つが全て証明できれば，任意の導出 $\Gamma \vdash M : A$ に対して性質 (*) が成立する」という推論法が,「導出の構成に関する帰納法」である[12].

上記 9 つの項目を見てみると，たとえば，*var* 規則の場合は,「部分導出」はないので，以下のことを証明すればよい．

> M が変数 x であり，$x : A$ が Γ に含まれていて，さらに，$M \to N$ であれば，$\Gamma \vdash N : A$ が導出可能である．

この場合 $x \to N$ となる N は存在しないので，仮定が成立せず，したがって，全体としては trivial に成立する．

また，*pair* 規則の場合は,「部分導出」が 2 つあるので，以下のことを証明すればよい．

> M が項 $< L, N >$ であり，A が型 $B \times C$ であり，$\Gamma \vdash L : B$ の導出が存在して，かつ，その導出に対して性質 (*) が成立し，$\Gamma \vdash N : C$ の導出が存在して，かつ，その導出に対して性質 (*) が成立し，さらに，$M \to N$ であれば，$\Gamma \vdash N : A$ が導出可能である．

これは，以下のようにして簡単に証明できる．$< L, P > \to N$ ということから，N が $< L1, P >$ の形で，かつ，$L \to L1$ であるか，または，N が $< L, P1 >$ の形で，かつ，$P \to P1$ であるかのいずれかである．前者の場合，$\Gamma \vdash L : B$ の導出に対して性質 (*) が成立するので，$\Gamma \vdash L1 : B$ という導出を作ることができる．そこで，この導出と，もともとあった $\Gamma \vdash P : C$ の導出を pair 規則により組み合わせれば，$\Gamma \vdash < L1, P > : B \times C$ の導出を作ることができる．後者の場合も同様である．

ほかの規則の場合も同様に証明できる．(ただし，apply 規則や case 規則の場合に「導出に対する代入」という概念を必要とするので，かなり面倒な議論が必要になる．その詳細は省略する．)

[証明の概要おわり]

定理 8 (canonical form に関する性質) $\vdash M : A$ が導出可能であれば，M は計算可能であるか，または，*canonical form (*最終規則が λ*, pair, inl, inr* のいずれかで導出された項*)* である．

[12] これは，n 分木に対する帰納法の一般化である．

$\vdash M : A$ が導出可能である，ということは，M が型がつく項であるだけでなく，$\Gamma = \{\}$ ということを意味している．つまり，M は変数の自由出現を持たない項である．通常「プログラム」は，自由な変数出現を持たない項であるが，この定理は,「プログラム」の計算結果は，canonical form であることを意味している．

逆にいうと，計算結果 (canonical form) 以外の形で計算が進まなくなることはない (計算途中でひっかかってしまう) ことがないことを意味している．

7.14 合流性

定理 9 (合流性) 型つきラムダ計算の項 M, N_1, N_2 に対して *(*つまり，M, N_1, N_2 は適当な宣言 $\Gamma_1, \Gamma_2, \Gamma_3$ と型 A_1, A_2, A_3 に対して，$\Gamma_1 \vdash M : A_1$, $\Gamma_2 \vdash N_1 : A_2$, $\Gamma_3 \vdash N_2 : A_3$ が導出可能であるとする*)*，$M \rightarrow^* N_1$ かつ $M \rightarrow^* N_2$ であれば，ある項 L があって，$N_1 \rightarrow^* L$ かつ $N_2 \rightarrow^* L$ となる．

この定理は，Church-Rosser 定理とも呼ばれる[13]．型のないラムダ計算に対しても成立する重要な定理であり，計算の順序によらず結果が一意的であることを意味している．

この定理の証明は省略する．

7.15 停止性

定理 10 (強い停止性) $\Gamma \vdash M : A$ が推論できるなら，$M \rightarrow N_1 \rightarrow N_2 \rightarrow N_3 \rightarrow \cdots$ となる無限の計算列は存在しない．

これは，型がきちんとつく項 M から始めた計算は，どのように計算しても，必ず有限回の計算により結果 (値) に到達することを意味している．

定理 11 (弱い停止性) $\Gamma \vdash M : A$ が推論できるなら，これ以上計算可能でない N に対して，$M \rightarrow^* N$ となる．

これは，型がきちんとつく項 M から始めた計算は，うまくやれば，有限回の計算により結果 (値) に到達することを意味している．

明らかに，強い停止性が成立すれば，弱い停止性は成立する．本講義でのべた型付きラムダ計算の体系に対しては，強い停止性が成立するので弱い停止性も成立する．一方，型のないラムダ計算の体系では，(強い/弱い) 停止性は成立しない．実際，$(\lambda x.xx)(\lambda x.xx)$ という項の計算は無限ループとなる．この定理の証明は省略する．

現在利用可能なほとんど全てのプログラム言語に対して,「停止しないプログラム」を書くことが可能であるという事実からすると，型付きラムダ計算の体系が停止性を満たす，という定理は奇異にうつるかもしれない．しかし，通常，人間は，停止しないことを目的にプログラムを書くことはないだろう.「どんな入力に対しても有限時間内に停止して答えを出す」プログラムを書きたいのがほとんどの場合である[14]．そのような観点からすると,「プログラム言語の機能のうち，この部分はきちんと停止する機能である」「停止しないのは，この機能である」という風にきちんと区別できることが好ましい．区別することにより,「このプログラムが停止するためには，どの部分をチェックすればよいか」が明確になるからである．

[13] Church も Rosser も人の名前であり，この分野を築いた偉人たちである．

[14] OS のカーネルやネットワーク・サーバなど，無限に動き続けるのが普通であるようなプログラムもある．しかし，これらのプログラムでも，リクエストに対しては有限時間内に応答することが求められているのであり，そのような応答をモデル化する場合は，やはり，停止性が満たされる必要がある．

言いかえると，多くのプログラミング言語は，型付きラムダ計算を基礎として，それに機能を追加する形で理解することができる．たとえば,「代入によって値を変更できる変数」,「再帰呼び出し」「繰り返し (C 言語の for 文など)」「ファイルへの入出力」などを加えると，現実的なプログラミング言語にかなり近付けることができる．そのとき，型の健全性，停止性，合流性などの性質がどのようになるか (保たれるのか，破られるのか) を観察することにより，プログラミング言語の構造が良く見えてくるのである．