第８回補足資料

１２月３日

「λ項であらゆる計算は表現でき、β簡約規則だけであらゆる計算は実行できる」チャーチのテーゼ

- B簡約規則の適用で型の合致制約を外すと、あらゆる計算はβ簡約だけでできるとコンピュータ科学の基本原理となります。λ計算の理論の誕生(1936年）をもって、コンピュータの計算理論の誕生と考えられています。すこし遅れて同時期に、チューリングマシンモデルも提出され、計算能力に関する同値性が証明されました。

(ラムダ計算をチューリングマシンでシミュレートすることができ、逆もできることが示されました。)

学習に当たって：β簡約規則はこの意味で強い計算力を持つ規則です。日常使っている足し算や掛け算の計算手続きの先入観でλ計算に当てはめようとすると躓きます。B簡約規則に従ったラムダ項プログラムになっていることを注意してください。

ラムダ計算もチューリングマシンもプログラムは模様（図的パターン）の変換のイメージです。

- 先週第７回目に解説した＋を表すラムダ項（プログラム）について再確認します。

[復習]

- ＋はNat→(Nat →Nat)という型でいた。

3+2と略記したときは、実際のλ項は((＋３）２)のことになります。通常の算術言語文法では、＋だけや３＋だけでは文法違反になりますが、ラムダ計算では＋だけでも（＋３）だけでも型が付けられるラムダ項です。また、（＋３）だけでも計算できます（但し、もう一つの入力を待ってから出ないと自然数値を出力しません。）

束縛関係が変わらなければ同じラムダ項

ラムダ項における変項記号の選び方にhs自由度があります。

λ記号で束縛している変項がどれであるかがあいまいさなく示されれば、変項記号を変えても同じラムダ項と考えます（専門的にはα－conversionと呼ばれます。）

たとえば、自然数３は、

$\lambda\ f^{P\to P}\lambda\ x^{P}(f(f(fx)))$

$\lambda\ g^{P\to P}\lambda\ y^{P}(g(g(gy)))$

$\lambda\ x^{P\to P}\lambda\ f^{P}(x(x(xf)))$ はどれも同じラムダ項であり、３を表す。

３本の棒表す。コム言った計算でも３本棒を示せるようにλで束縛関係を与えている。

今、λの束縛変項に出てくる型表記を省略して書きます。提出レポートでは省略しないでください。

＋は、λmλnλfλx((mf)((nf)x))

3をλgλz(g(g(gz))), 2をλhλy(h(hy))と表記することとします。

((+3)2)▷(λnλfλx((3f)((nf)x))2) [赤部分と青部分はどちらもβ簡約できるリデックスの形。いま赤を先に簡約してみます。]

▷λfλx((3f)((2f)x)) [赤と青が簡約可能。青を簡約してみます.

3と２の部分を省略せずに書くとこのラムダ項は、

λfλx((λgλz(g(g(gz)))f)((λhλy(h(hy)) f)x))

▷ λfλx(λz(f(f(fz)))((λhλy(h(hy)) f)x))

▷ λfλx(λz(f(f(fz)))(λy(f(fy)) x)) [オレンジ部が簡約可能]

▷ λfλx(λz(f(f(fz)))(f(fx))) [赤部が簡約可能]

▷ λfλx(f(f(f(f(fx))))) これは５です。

(T or F)という中置記法は単純型t付きλ計算のλ項では、
((OR T)F)という一引数ずつの二回の関数適用で（前置記法的に）表現されます。これに2回β簡約規則で簡約すると次の最初のラムダ項となります。さらにβ簡約を進めると次のようになり、TorFの値はTとなります。[例題にはあるタイポがありますが、ラムダ項の定義に従えば直しやすいと思います。下は直してあります。]

$\lambda m^{Nat}\lambda n^{Nat}((T((Fm)m))((Fm)n))$　ここでTという省略記号の部分をλ項で示すと次のλ項のことであると分かります。、

$\lambda m^{Nat}\lambda n^{Nat\,Nat}((\lambda c^{Nat}\lambda d^{Nat}c((Fm)m))((Fm)n))$

$\triangleright \lambda m^{Nat}\lambda n^{Nat}(\lambda d^{Nat}((Fm)m)((Fm)n))$

$\triangleright \lambda m^{Nat}\lambda n^{Nat}(Fm)m)$ このFを省略しないでλ項で書くと、

$\equiv \lambda m^{Nat}\lambda n^{Nat}((\lambda c^{Nat}\lambda d^{Nat}d\ m)m)$

$\triangleright \lambda m^{Nat}\lambda n^{Nat}(\lambda d^{Nat}d\ m)$

$\triangleright \lambda m^{Nat}\lambda n^{Nat}m$ これはTの定義です。

≡ T　一回のβ簡約は▷で、簡約に関わらない同一項は≡で表します。

ここでは、たまたま外側の簡約可能部分（リデックス）を選びながらλ項プログラムを実行していきましたが、(Fm)もβ簡約でき、((Fm)n)や((Fm)m)を評価してから一番外側のORのリデックスをβ簡約することもできます。計算結果が順序（計算方略）によたないことがλ計算の合流性（チャーチ＝ロッサー性）で保証されています。

$\lambda m^{Nat}\lambda n^{Nat}((T((Fm)m))((Fm)n))$　ここでTという省略記号の部分をλ項で示すと次のλ項のことであると分かります。、

$\lambda m^{Nat}\lambda n^{Nat\,Nat}((\lambda c^{Nat}\lambda d^{Nat}c((Fm)m))((Fm)n))$

- ✂ $\lambda m^{Nat}\lambda n^{Nat}(\lambda d^{Nat}((Fm)m))((Fm)n))$
- ✂ $\lambda m^{Nat}\lambda n^{Nat}(Fm)m)$ このFを省略しないでλ項で書くと、
- ≡ $\lambda m^{Nat}\lambda n^{Nat}((\lambda c^{Nat}\ \lambda d^{Nat}\,d\ m)m)$
- $\lambda m^{Nat}\lambda n^{Nat}(\ \lambda d^{Nat}\,d\ m)$
- $\lambda m^{Nat}\lambda n^{Nat}\,m$ これはTの定義です。
- ≡ T

、Fm)もβ簡約でき、(Fm)n)や((Fm)m)を評価してから一番外側のORのリデックスをβ簡約することもできます。計算結果が順序（計算方略）によたないことがλ計算の合流性（チャーチ＝ロッサー性）で保証されています。この例では外側からの評価方略（ストラテジー）のほうが無駄な計算をしなくて済むこ とが分かります。内側からすべてを評価してく方略はEAGER EVALUATION,内側の部分項の評価は必要に
なったときにだけ行うよう外側からの評価して計算コストを抑える仕方はLAZY EVALUATIONに当たると言えます。計算モデルレベルではどんな順序でも計算結果が分かっている場合も、実装レベルではどのような方略をとるかが重要となります。ここでは、たまたま外側の簡約可能部分（リデックス）を選びながらλ項プログラムを実行していきましたなります。

λ m^{Nat} λ n^{Nat}((T((Fm)m))((Fm)n)) ここでTという省略記号の部分をλ項で示すと次のλ項のことであると分かります。、

λ m^{Nat} λ n$^{Nat\ Nat}$ ((λ c^{Nat} λ d^{Nat}c((Fm)m))((Fm)n))

- ✂ λ m^{Nat} λ n^{Nat}(λ d^{Nat} ((Fm)m)((Fm)n))
- ✂ λ m^{Nat} λ n^{Nat} (Fm)m) このFを省略しないでλ項で書くと、
- ≡ λ m^{Nat} λ n^{Nat} ((λ c^{Nat} λ d^{Nat} d m)m)
- λ m^{Nat} λ n^{Nat} (λ d^{Nat} d m)
- λ m^{Nat} λ n^{Nat} m これはTの定義です。
- ≡ T

同様に、いろいろは順番の計算実行を考えてください。

- F OR F 、つまり　((OR F)F)というλ項の評価（β簡約規則により、規約形（正規形、即ち、それ以上簡約できない項）にたどり着くβ簡約のいろいろな（リデックスの）選び方を考えてみてください。

これまでの型の定義に積型を加えます。（「ならば」だけの論理式に「かつ」を加えたことに対応します。　これまでのラムダ計算は単純型付きラムダ計算と呼ばれています。）

4.2　積タイプを持つタイプ付ラムダ計算

タイプの定義を次のように拡張する。

定義 2.20’ (タイプ)

1 及び 2 は前の通り。

3. A、B がタイプのとき、$A \times B$ もタイプである。

ここで $A \times B$ は**積タイプ**と呼ばれる．積タイプ $A \times B$ は命題論理における $A \wedge B$ に対応している．

次にラムダ項の定義を拡張する。

定義 2.21’ (ラムダ項)

積型を持つラムダ項へのラムダ項定義の拡張

- 定義 2.21’ (ラムダ項)

1,2,3 は前の通り。

4. もし t がタイプ A のラムダ項で s がタイプ B のラムダ項なら、＜t, s＞はタイプ A × B のラムダ項である。

5. もし t がタイプ A × B のラムダ項ならば、π^1(t) はタイプ A のラムダ項であり、又π^2(t)はタイプ B のラムダ項である。

- ここで＜t, s＞は t と s との対を意味する．即ち，積タイプは対のタイプを表している.

右と左の区別のある「順序対」です。

- 積タイプの対は順序対です。前頁のラムダ項の定義の４及び５でこの左右を区別する扱いがあらわされています。

- 対というデータ型には順序なしの対もあり得ます。ここでは取り扱いません。対の一般化としてリスト型があります。時間に余裕があれば授業で取り扱いますが、基本教材には入っておらず、レポート課題の範囲外です。

上級者へ：データ型「リスト」はHeadとTailの順序対として定義できます。例えば最も単純な自然数のリストList[Nat]の型を持つラムダ項を定義してみてください。

積型を持つラムダ項の定義に対応する型推論

ラムダ項の形成規則は次のように拡張できる。

$$\frac{t : A \quad s : B}{< t, s >: A \times B} \times -I$$

$$\frac{t : A \times B}{\pi^1(t) : A} \times -E \text{左}$$

$$\frac{t : A \times B}{\pi^2(t) : B} \times -E \text{右}$$

積型についての β 簡約規則の追加（上級者は証明の正規化との対応も見ておいてください。）

ラムダ項の書き換え規則 ($\beta-$ 簡約規則) を次のように拡張する。

1. これまでの $\beta-$ 簡約規則
2. $\pi^1 < t, s > \triangleright t$
   $\pi^2 < t, s > \triangleright s$

積型が入った次の表現が型付きラムダ項であることを型推論によって確かめた例。のラムダ項をFと書くと、(F＋）は２引数を対として待つ型の足し算に変わる。ここでは型推論規則の形式で型チェックをしていますが、型推論による調べ方でなく定義のテキストに沿って型を調べることでも、もちろんよいです。

(4) $(A \to (B \to C)) \to (A \wedge B \to C)$ のタイプを持つラムダ項の例

$$\dfrac{\dfrac{\dfrac{\dfrac{x : A \times B}{\pi^2 x : B} \quad \dfrac{\dfrac{x : A \times B}{\pi^1 x : A} \quad y : A \to (B \to C)}{y\pi^1 x : B \to C}}{(y\pi^1 x)\pi^2 x : C}}{\lambda x^{A \times B}(y\pi^1 x)\pi^2 x : A \wedge B \to C}}{\lambda y^{A \to (B \to C)} \lambda x^{A \times B}(y\pi^1 x)\pi^2 x : (A \to (B \to C)) \to (A \wedge B \to C)}$$

練習問題

- 前ページのラムダ項をHとして、

（H *Times)*を β 簡約で計算してください。その結果をGとします。次に、(G <2,3>)を計算してください。

型付きλ計算について

型付きラムダ計算はλ項プログラムが文法通りに型付けされている場合にはそのことの証明を与えてくれるわけです。現在は命題論理という最も小さい論理体系に限定して話していますが、述語論理という次の論理言語では、例えば

「あるリスの型の任意のリストデータに対して、辞書的順序にソートしたリストも存在する」

というような表現が述語論理で書けて、この論理式を「型」とみなすと、この論理式の証明は「ソート型」を持つラムダ項プログラムともなせることに成ります。「型」はプログラムの一般的「仕様」に当たるわけです。異なる種類のソートアルゴリズムは、異なる証明や対応するラムダ項ということになります。これはごく大雑把な説明ですので、もっと深く勉強したい人がいれば参考文献などを紹介します。

型付きラムダ計算について（続き）

今、命題論理から述語論理への型付きラムダ計算の話に触れました。もう一つの方向の型付きラムダ計算として重要なのは、高階論理への拡張です。それに対応する型は「多相型(Polymorphic Types)」と呼ばれます。

基本教材で取り扱った、命題論理に基づくIf-Then-Elseのλ項プログラムでは、Ifのあとのブール型の値がTかFかにより二つのラムダ項プログラムのどちらを走らせるかを決めましたが、どちらもNat型のプログラムでした。どんな型のプログラムを走らせる場合にも、このIF－THEN－ELSEを使えるようにしようとすると、多相型を導入することになります。多相型にはF系の強い多相型がありますが、MLやHASKELなどは弱い多相型で、時間が許せばその周辺の話題も載せたいと思います。（ただし、基本教材にないことはレポート課題の基本問題としては出題することはありません。）

型付きから型なしラムダ計算へ

- tが入力として待っている型とｓの型が合致していなくても関数適用(ts)を許すとする場合を型なしラムダ計算と言います。
- すべての表現ん方は同じとみるという見方をして、型なしラムダ計算を型付きラムダ計算の特殊例とみつこともできます。
- 通常は型の定義なしにラムダ項の定義とβ簡約規則の定義をするとき、型なしラムダ計算と呼びます。（どれも計算面からみると同じことです。）
- 型付きラムダ計算のメリットとして型推論により型チェックが計算でき、文法（や仕様）に合ったラムダ項プログラムであることが証明という保証付きであること、
- 型付きラムダ項プログラムは停止性が保証されること（ｔづまり、プログラムを実行するとかんらず計算結果が出ることです。
- プログラムが意図している仕様通りであることとプログラムの停止性はプログラムのCORRECTNESSのの最も基本的要請です。この二つの性質は全く独立な性質です。
- これらを備える点は型付きラムダ計算言語のメリットと言えます。

型なしラムダ計算

- これまで基本教材で扱ってきたβ簡約による計算例は型条件を課しても成立する計算でした。逆に、自然数０，１，２，３、…の定義やBOOL値T,Fについている変項の型宣言を削除して、β簡約規則にも型の一致条件を入れず、＋やTIMESやIF-THEN-ELSEなどの定義にも型宣言を無視すれば、それらの計算をすることには何の問題もありません。
- いままで授業や基本教材で計算したことは型なしでもそのまま計算できるわけです。逆は成立しません。型付き項としては認められない計算ができることとなり、計算としてのパワーや計算力は上がります。「すべての計算は型なしラムダ項で表現でき、β簡約だけで計算できる」というチャーチのテーゼがλ計算の確立とともに唱えられ（1936年）、コンピュータ科学で同意を得ています。
- ただし、注意を要するのは、「計算できることはすべてこの枠組みでできるが、計算が止まらない場合も含む」という点です。

停止しない型なしラムダ項の典型例

(xx)は型がつかないことが分かります。（どうしてか、型付きラムダ項の定義に戻って確かめてみてください。）

いまつぎのような型なしラムダ項を考えます。

$(\lambda x(xx)\ \lambda x(xx))$

これそのものは（型を無視すれば）β簡約のリデックスですので、β簡約します。するとその結果全く同じλ項となり、さらにβ簡約できます。この簡約が無限に続き、停止しないことが分かります。

実はこの特殊例は、前に言及したラッセルのパラドクスとも深い関係にあります。

計算表記から集合表記へ：ラッセルノパラドクスとの関係

- 太郎は学生である

（学生　太郎）と表し、Bool型で真偽ができるとする。

λx(学生　x) は学生の集合を表すこととなる。

このような集合表現を使って、前に言及したラッセルノパラドクスを取り扱うと、前頁の停止しない型なしラムダ項の無限計算過程と似た構造が現れることが分かる。

少し変形すると、ゲーデルの不完全性定理、カントールの集合論における対角線論法も同様の構造で著せることが分かる。

停止しない型なしラムダ項とラッセルノパラドクス

述語論理言語では、主語と述語を分節化し、「太郎は学生である」を

（学生　太郎）のように表示します（カッコの使い方を少し変えて　学生（太郎）と書くことも多いです。）

ここで「太郎は学生である」は真偽の値をとるのでブール型と呼ぶことにしておきます。太郎は個有名型を持つとします。すると、

学生の型は「個有名→ブール」となります。学生は個有名の集合で、例えば太郎がその集合に入れば「太郎は学生である」が真となる、これがこの型理論の背後にある（表示的）意味論です。

このような集合論や述語論理の解釈を念頭に置くと、一般に,xが入力を待って、その入力ｙがｘのメンバーであるかどうかで（ｘが著す集合に所属するかどうかで）ブール値の真偽を返すとします。この時(xy)はブール値をとるので、¬(xy)も有意味です。太郎は学生ではない、もこの形となります。(xy)は「ｘはｙを含む」、¬(xy)は「ｘはｙを含まない」、(xx)は「ｘはｘ自身を含む」、¬(xx)は「xはx自信を含まない」、λｘ¬(xx)は「自分自身を含まないものからなる集合」、(対象の入力を待ち真偽値を返す関数は集合と同一視できます。）。

(λｘ¬(xx) λｘ¬(xx)) は「自分自身を含まないものからなる集合を自分自身を含まないものの集合が含む」という（前に言及した）らせる文となります。ここでもβ簡約という代入操作事態は意味があるのでβ簡約してみると、ラッセル文RはNotRが同にゅつされ、NotRから逆にNotNotRつまり、Rが導出されます。B簡約で意味は変わらないと考えるのでぱあ即すが生じます。これを契機に(xx)のような事故言及ができないよう型付け理論が誕生しました。

(λｘ¬(xx) λｘ¬(xx))　＞　¬(λｘ¬(xx) λｘ¬(xx))　(λｘ¬(xx) λｘ¬(xx))