計算と論理

Jacques Garrigue, 2015 年 10 月 24 日

5　コンビネータ論理

λ 計算が発明される前に、証明の変換を表すために Shönfinkel と Curry がコンビネータ論理を考えた。

構文　コンビネータ論理は λ 計算よりさらに簡単で、変数や関数がなく、関数適用と二つのコンビネータ S と K しかない。

$$M ::= \mathsf{S} \mid \mathsf{K} \mid M\ M$$

ここでも、括弧は自由に使えて、括弧がない場合左に付ける。

$$\mathsf{S}\ \mathsf{K}\ (\mathsf{K}\ \mathsf{S}\ \mathsf{K}) = (\mathsf{S}\ \mathsf{K})\ ((\mathsf{K}\ \mathsf{S})\ \mathsf{K})$$

簡約規則　λ 計算同様、計算は書き換え規則で定義されている。

$$\begin{array}{rcl} \mathsf{S}\ M\ N\ L & \to & (M\ L)\ (N\ L) \\ \mathsf{K}\ M\ N & \to & M \end{array}$$

直感的には、S は分配コンビネータで、L を M と N に渡し、ML を NL に適用している。また K は二つ目の引数を捨てるコンビネータである。

定理 5.1 コンビネータ論理は**合流性**を持つ。$M \to \ldots \to N$ と $M \to \ldots \to P$ という二つの簡約列があれば、$N \to \ldots \to T,\ P \to \ldots \to T$ となるような T が存在する。

簡約の例　発案当初、S と K 意外に恒等関数 I も必要とされていた。しかし、S と K の組合せで I が作れることが分かった。

$$\mathsf{S}\ \mathsf{K}\ \mathsf{K}\ L \to (\mathsf{K}\ L)\ (\mathsf{K}\ L) \to L$$

これを使うと、λ 計算の任意の項がコンビネータで書ける。例えば、$\lambda x.\mathsf{c}_+\ x\ x$ は $\mathsf{S}\ \mathsf{c}_+\ (\mathsf{S}\ \mathsf{K}\ \mathsf{K})$ で表現できる。計算してみると

$$\mathsf{S}\ \mathsf{c}_+\ (\mathsf{S}\ \mathsf{K}\ \mathsf{K})\ M \to \mathsf{c}_+\ M\ (\mathsf{S}\ \mathsf{K}\ \mathsf{K}\ M) \to \mathsf{c}_+\ M\ M$$

で上記の λ 項と同じ動きをする。

翻訳　形式的に λ 計算からの翻訳を定義する。便宜のために、コンビネータ論理の式の中に変数を含むが、コンビネータ論理としてはその変数が未定の項を表していると思えばいい。

まず、変数 x を含むコンビネータ式から関数を作る演算子 $\lambda^*x.$ を定義する。($\lambda^*x.$ は構文ではなくて、操作である)

$$\begin{array}{rcll} \lambda^*x.M & = & \mathsf{K}\ M & (x \notin FV(M) \text{ のとき}) \\ \lambda^*x.x & = & \mathsf{S}\ \mathsf{K}\ \mathsf{K} & \\ \lambda^*x.M\ N & = & \mathsf{S}\ (\lambda^*x.M)\ (\lambda^*x.N) & \end{array}$$

補題 5.1 任意の *(変数を含む)* コンビネータ式 M と N について、

$$(\lambda^* x.M)\ N \to [N/x]M$$

証明 $\lambda^* x$ の定義に関する帰納法で証明する。

$x \notin FV(M)$ のとき、$(\lambda^* x.M)\ N = \mathsf{K}\ M\ N \to M = [N/x]M$

$M = x$ のとき、$(\lambda^* x.x)\ N = \mathsf{S}\ \mathsf{K}\ \mathsf{K}\ N \to N = [N/x]x$

$M = M_1\ M_2$ のとき、$(\lambda^* x.M_1\ M_2)\ N = \mathsf{S}\ (\lambda^* x.M_1)\ (\lambda^* x.M_2)\ N \to$
$((\lambda^* x.M_1)\ N)\ ((\lambda^* x.M_2)\ N) \to [N/x]M_1\ [N/x]M_2 = [N/x](M_1\ M_2)$ □

この定義を使うと翻訳関数 $[\![\_]\!]$ が以下のように定義できる。

$$\begin{array}{rcl} [\![x]\!] & = & x \\ [\![\lambda x.M]\!] & = & \lambda^* x.[\![M]\!] \\ [\![M\ N]\!] & = & [\![M]\!]\ [\![N]\!] \end{array}$$

例 チャーチ数に関する定義を翻訳する。読み易さのやめに、S K K を I で略している。

$$\begin{array}{rcl} [\![\mathsf{c}_0]\!] & = & [\![\lambda f.\lambda x.x]\!] = \lambda^* f.\lambda^* x.x = \lambda^* f.\mathsf{I} = \mathsf{K}\ \mathsf{I} \\ [\![\mathsf{c}_1]\!] & = & \lambda^* f.\lambda^* x.f\ x = \lambda^* f.\mathsf{S}\ (\lambda^* x.f)\ (\lambda^* x.x) = \lambda^* f.\mathsf{S}\ (\mathsf{K}\ f)\ \mathsf{I} \\ & = & \mathsf{S}\ (\mathsf{S}\ (\mathsf{K}\ \mathsf{S})\ (\mathsf{S}\ (\mathsf{K}\ \mathsf{K})\ \mathsf{I}))\ (\mathsf{K}\ \mathsf{I}) \\ [\![\mathsf{c}_\times]\!] & = & \lambda^* m.\lambda^* n.\lambda^* f.m\ (n\ f) \\ & = & \lambda^* m.\lambda^* n.\mathsf{S}\ (\lambda^* f.m)\ (\lambda^* f.n\ f) \\ & = & \lambda^* m.\lambda^* n.\mathsf{S}\ (\mathsf{K}\ m)\ (\mathsf{S}\ (\mathsf{K}\ n)\ \mathsf{I}) \\ & = & \lambda^* m.\mathsf{S}\ (\lambda^* n.\mathsf{S}\ (\mathsf{K}\ m))\ (\lambda^* n.\mathsf{S}\ (\mathsf{K}\ n)\ \mathsf{I}) \\ & = & \lambda^* m.\mathsf{S}\ (\mathsf{K}\ (\mathsf{S}\ (\mathsf{K}\ m)))\ (\mathsf{S}\ (\mathsf{S}\ (\mathsf{K}\ \mathsf{S})\ (\mathsf{S}\ (\mathsf{K}\ \mathsf{K})\ \mathsf{I}))\ (\mathsf{K}\ \mathsf{I})) \\ & = & \mathsf{S}\ (\mathsf{K}\ \mathsf{S})\ (\mathsf{S}\ (\mathsf{K}\ \mathsf{K})\ (\mathsf{S}\ (\mathsf{K}\ \mathsf{S})\ (\mathsf{S}\ (\mathsf{K}\ \mathsf{K})\ \mathsf{I})))\ (\mathsf{K}\ (\mathsf{S}\ (\mathsf{S}\ (\mathsf{K}\ \mathsf{S})\ (\mathsf{S}\ (\mathsf{K}\ \mathsf{K})\ \mathsf{I}))\ (\mathsf{K}\ \mathsf{I}))) \end{array}$$

コンビネータ式はとても長くなるが、λ計算と同値である。

冠頭標準形 S, S M, S M N, K, K M の形をしたコンビネータ式は冠頭標準形だという。先頭を含む簡約が存在しないという意味である。

定理 5.2 M が弱冠頭標準形を持つことと $[\![M]\!]$ が冠頭標準形を持つことが同値である。

6 コンビネータ簡約器

万能チューリング機械と同様に、コンビネータ式をデータとしてもらい、それをこれ以上簡約できない正規形になるまで簡約規則で書き換えるλ項が定義できる。

符号化 まず、データの表現を定めなければならない。コンビネータ式は 3 種類しかないので、ここで対と真偽値の組合せを使う。コンビネータ式をデータに変換する過程を符号化とおう。

$$\begin{array}{rcl} \overline{\mathsf{S}} & = & \mathsf{pair\ t\ t} \\ \overline{\mathsf{K}} & = & \mathsf{pair\ t\ f} \\ \overline{M\ N} & = & \mathsf{pair\ f}\ (\mathsf{pair}\ \overline{M}\ \overline{N}) \end{array}$$

対の左が t のとき、コンビネータを表し、コンビネータの種類 (S か K) は対の右にある。適用のときは対の左が f で、右は符号化された式の対である。

簡約器　次に以上の形で符号化されたコンビネータ式を簡約するλ項を定義する。まず、補助関数を計算する項を定義する。

$$
\begin{aligned}
\mathsf{eqb} = \quad & \lambda x.\lambda y.x\ y\ (\mathsf{not}\ y) \\
\mathsf{andb} = \quad & \lambda x.\lambda y.x\ y\ \mathsf{f} \\
\mathsf{cmp} = \quad & \lambda x.\lambda y.\mathsf{fst}\ x\ (\mathsf{andb}\ (\mathsf{fst}\ y)\ (\mathsf{eqb}\ (\mathsf{snd}\ x)\ (\mathsf{snd}\ y))) \\
& \qquad (\mathsf{fst}\ y\ \mathsf{f}\ (\mathsf{snd}\ x\ (\lambda x_1.\lambda x_2.\mathsf{snd}\ y\ (\lambda y_1.\lambda y_2.\mathsf{andb}\ (\mathsf{cmp}\ x_1\ y_1)\ (\mathsf{cmp}\ x_2\ y_2))))) \\
\mathsf{app} = \quad & \lambda x.\lambda y.\mathsf{pair}\ \mathsf{f}\ (\mathsf{pair}\ x\ y)
\end{aligned}
$$

cmp は二つの符号化されたコンビネータ式を比較する。そのために、二つの真偽値を比較する eqb と二つの真偽値の論理積を計算する andb も使う。app は二つの符号化されたコンビネータ式の適用を符号化する。

以下の ev が一回の冠頭簡約を行う。eval は冠頭標準形になる (冠頭簡約がなくなる) までそれを繰り返す。

$$
\begin{aligned}
\mathsf{ev} = \quad & \lambda x.\mathsf{fst}\ x\ x \\
& \quad (\mathsf{snd}\ x\ (\lambda m.\lambda n. \\
& \qquad \mathsf{fst}\ m\ x \\
& \qquad (\mathsf{snd}\ m\ (\lambda p.\lambda q. \\
& \qquad\quad \mathsf{fst}\ p\ (\mathsf{snd}\ p\ x\ q) \\
& \qquad\quad (\mathsf{snd}\ p\ (\lambda r.\lambda t. \\
& \qquad\qquad \mathsf{fst}\ r\ (\mathsf{snd}\ r\ (\mathsf{app}\ (\mathsf{app}\ t\ n)\ (\mathsf{app}\ q\ n))\ (\mathsf{app}\ t\ n)) \\
& \qquad\qquad (\mathsf{app}\ (\mathsf{ev}\ m)\ n))))))) \\
\mathsf{eval} = \quad & (\lambda x.\mathsf{cmp}\ x\ y\ x\ (\mathsf{eval}\ y))\ (\mathsf{ev}\ x)
\end{aligned}
$$

上のλ項が読み難いので、ここでもっとプログラミング言語らしい構文で書く。

```
let rec ev x =
  if fst x then x else                                  (* 引数がない *)
  let (m, n) = snd x in
  if fst m then x else                                  (* 引数が足りない *)
  let (p, q) = snd m in
  if fst p then (if snd p then x else q) else           (* Kなら足りる *)
  let (r, t) = snd p in
  if fst r then (if snd p then app (app t n) (app q n) else app t n)
  else app (ev m) n

let rec eval x =
  let y = eval x in
  if cmp x y then x else eval y
```

復号器　簡約をλ計算に完全に任せると、もっと単純な方法がある。符号化されたコンビネータ式をそれと同等なλ項に置き換える関数 comp である。例えるならば、eval がコンビネータ論理のインタプリタで、comp はλ計算へのコンパイラである。

$$
\begin{aligned}
\mathsf{comp} = \quad & \lambda x.\mathsf{fst}\ x\ (\mathsf{snd}\ x\ (\lambda f.\lambda g.\lambda x.f\ x\ (g\ x))\ (\lambda x.\lambda y.x)) \\
& \qquad (\mathsf{snd}\ x\ (\lambda m.\lambda n.\mathsf{comp}\ m\ (\mathsf{comp}\ n))
\end{aligned}
$$

7 停止問題

λ計算 (またはコンビネータ論理) が定義する計算可能性とチューリング計算可能性が同値であるので、判定不能問題をλ計算で定義しても良い。

あるλ項 L がある条件を判定するとは、判定対象となる項 M について、M が条件を満すときに $L\ M$ が最左戦略で t に簡約され、満さないときに f に簡約されなければならない。

定理 7.1 符号化されたコンビネータ式が冠頭標準形を持つかどうかを判定するλ項 H が存在しない。

証明 任意のλ項 M について、$M^* = \overline{[\![M]\!]}$ が M のコンビネータ訳の符号化である。M が弱冠頭標準形を持つことと $[\![M]\!]$ が冠頭標準形を持つことが同値であることが定理 5.2 から分かる。eval M^* と comp M^* の弱冠頭標準形の存在も同値である。

符号化されたコンビネータ式に対して、それをもう一度符号化する関数 quote を定義する。定義から quote $M^* = M^{**}$。

$$\begin{aligned}\mathsf{quote} &= \lambda x.\mathsf{fst}\ x\ (\mathsf{snd}\ x\ (\mathsf{pair\ t\ t})^*\ (\mathsf{pair\ t\ f})^*) \\ &\qquad (\mathsf{snd}\ x\ (\lambda m.\lambda n.\mathsf{app}\ (\mathsf{app}\ \mathsf{app}^*\ (\mathsf{quote}\ m))\ (\mathsf{quote}\ n)))\end{aligned}$$

もしも定理にあるような H が存在すれば、$F = \lambda x.H\ (\mathsf{app}\ x\ (\mathsf{quote}\ x))\ \Omega\ \mathsf{t}$ とおく (ここでは $\Omega = (\lambda x.x\ x)\ (\lambda x.x\ x)$ は冠頭標準形を持たないλ項である)。$H\ (\mathsf{app}\ M^*\ (\mathsf{quote}\ M^*)) = H\ (\mathsf{app}\ M^*\ M^{**}) = H\ (\overline{[\![M]\!]\ [\![M^*]\!]}) = H\ (M\ M^*)^*$ が t のときに $F\ M^*$ 弱冠頭標準形を持たず、f のときに標準形 t を持つ。

$F\ F^*$ が冠頭標準形を持つかどうかを考えよう。持っているためには、$H\ (\mathsf{app}\ F^*\ F^{**})$ は f を返さなければならない。しかし、それは $F\ F^*$ が弱冠頭標準形を持たないということになる。逆に、持っていなければ、$H\ (\mathsf{app}\ F^*\ F^{**})$ は t を返さなければならない。しかし、それは $F\ F^*$ が弱冠頭標準形を持つということになる。 □

チューリング機械の場合と同様に、この定理がヒルベルトの Entscheidungsproblem の反例となる。