解　説

論　理　と　関　数†

田　村　浩一郎††　大谷木　重　夫††

1. はじめに

たとえば，$\forall x \forall y \forall z(x \leq y \wedge y \leq z \Rightarrow x \leq z)$ というような**論理文**と，$f(x)=x^2+3$ のような**関数**を見比べた時，論理文はある“事実”を叙述（宣言）しているのに対して，関数はある入力に対する出力(関数値)を規定する手続きを与えるものであるとみることができる．論理文は知識や事象を宣言的 (declarative) に表現するのに便利であり，一方，関数は手続き的 (procedural) に表現するのに便利である．

これまでの大多数のプログラム言語は基本的には手続き的表現を用いている．これはいわば“how”を表現するものと言って良い．それに対して“what”を与えれば，あとはコンピュータが“how”の部分をやってくれないかという要望は昔から強かった．近年ハードウェアのコストダウンやソフトウェア技術の進歩から再び非手続き的（宣言的）表現がみなおされ，その表現を許容する言語もではじめている．しかしそれによって生じる問題もまた根深いものがある．宣言的表現を与えるとコンピュータの実行時間が途端に長くなりすぎる．その効率を良くしようとすれば手続き的表現を与えなければならず，すると従来の手続き的言語の場合よりもむしろプログラミングがむずかしくなってしまう．この問題を純粋化すれば，論理表現と関数表現のミスマッチとしてみることができる．

また，もともと手続き的な表現を用いた方が素直な知識というものも数多い．道を教えたり料理の作りかたを教えたりする時，大半の人は手続き的に（こうしてあああして…）表現する．人工知能の研究ではある時期宣言的表現と手続き的表現との大論争があったが，結論は，それぞれ向き不向きがあるから両者をミックスできる表現形式が良い，と言うことになった．しかしその場合の問題もやはり両者を統合できる体系をいかに構築できるかであり，そこをあいまいのままにしておくと両者のミスマッチに悩まされることになる．

さらに最近論理型プログラミングと関数型プログラミングの提唱が盛んであるが，片方だけでは上と全く同じ理由から片手落ちであると思われ，真に実用的でしかも理論的にきれいな言語を作ろうとすればやはり論理と関数の統合体系が必要になろう．

この他に，プログラムの自動検証，自動合成といった分野でも同様な問題が考えられる．

もちろん応用的見地から離れて学問的見地からみても，表現力が豊かで強力な表現体系（理想言語）を求めるのはおそらくライプニッツ以来の夢であろう．Frege は Russel (のパラドックス) によって，Hilbert は Gödel (の不完全性定理) によって挫折させられたといわれるが，その後の理論的進展はどうなっているのだろうか．論理と関数という一見対立した概念の統合という立場で近年の理論的成果をみることも可能であろう．

本稿ではこういった動機から出発して，最終的には高階直観論理の一体系に達する．これは関数体系をまるごと埋め込むことができる強力な論理体系になっている．そこには論理と関数のある種の調和がある．

本稿の骨格は Scott 19) によっている．この論文を理解するには残念ながら多くの予備知識を必要としている．本稿はそのための入門としての役割を果たすと同時に紹介にもなることを意図して書いたものである．したがってできるだけわかりやすく述べるよう努力したつもりであるが，対象が対象だけにわれわれの多くにとってはなじみのない概念が続出し，その結果決して一読してすぐわかるというものにはなっていない．他人のプログラムを読む程度の忍耐を筆者らは読者に期待したい．

2. 関数と命題論理

関数体系は λ 計算体系として記号化できる．関数と論理の相互関係をみるために，まず λ 計算と命題論理

† Logic and Function by Koichiro TAMURA and Shigeo OYAGI (Control Devision, Electrotechnical Lab. MITI).

†† 電子技術総合研究所制御部

との対比をみておくことにしよう．その結果，λ計算を行うことは命題論理（の部分系）の証明過程と一致することがわかる．

2.1 関数とλ計算

関数を計算する基本操作は入力を変数に代入することにある．たとえば，$f(x)=x^2+3$ という関数に対して $f(5)$ を求めるには x を5でおきかえて計算すれば良い．この代入操作に注目して Church はλ計算の体系を構築した．その後 McCarthy がこの体系をもとに LISP を作ったのは良く知られている．

λ計算ではすべての対象 (object) を“関数”であるとみなし，計算は主として代入操作による変形であると考える．関数への入力に対応する変数を明示するために文字λを用いる*．たとえば，さきの $f(x)$ は，次のようにあらわす．

$$f\equiv\lambda x\cdot x^2+3$$

f は，関数 $\lambda x\cdot x^2+3$ の名前である．$f(5)$ は

$$f(5)\equiv(\lambda x\cdot x^2+3)\ (5)$$
$$\Rightarrow\ 25+3\ \Rightarrow\ 28$$

と変形される．ここで実は“+”も関数である．さらに，2乗操作 (x^2) も，そして5も25も3も“関数”である．たとえば，

$$1\equiv\lambda a.\,\lambda b.\,a(b)$$
$$2\equiv\lambda a\cdot\lambda b\cdot a(a(b))$$
$$\vdots$$

とあらわせば，

$$\text{add}\ 1\equiv\lambda x\cdot\lambda y\cdot\lambda z\cdot y((x(y))(z))$$

として，ある数に1を足す関数が定義できる．

たとえば add 1(1) は次のようになる．

$$\text{add}\ 1(1)$$
$$=\underbrace{(\lambda x\cdot\lambda y\cdot\lambda z\cdot y((x(y))(z))}_{\text{add }1}\ \underbrace{(\lambda a\cdot\lambda b\cdot a(b))}_{1}$$
$$=\lambda y\cdot\lambda z\cdot y(\underbrace{((\lambda a.\,\lambda b.\,a(b)(y))}_{\downarrow}(z))\quad(x\text{に}1\text{を代入})$$
$$=\lambda y\cdot\lambda z\cdot y(\overbrace{(\underbrace{\lambda b\cdot y(b)}_{\downarrow}(z))}^{\downarrow})\quad(a\text{に}y\text{を代入})$$
$$=\lambda y\cdot\lambda z\cdot y(\overbrace{y(z)}^{\downarrow})\quad(b\text{に}z\text{を代入})$$
$$=\lambda a\cdot\lambda b\cdot a(a(b))\quad\begin{pmatrix}y\text{を}a\text{に，}z\text{を}b\\\text{に記号を変える}\end{pmatrix}$$
$$=2.$$

高位プログラム言語の構成要素も面倒ではあるがすべて関数すなわちλ式であらわせる．この点に着目して Landin[24] や Strachy[23] らがプログラム言語をλ式で表現しなおすことによってその意味規則 (semantics) を厳密化しようとした．

しかしλ計算は単に形の上だけの操作であり，高位言語をたとえλ式に翻訳しても，それは単に別な言語体系に表現しなおしたにすぎない．λ式の意味を別な形式で与えることが必然的に必要になってくる．そこで Scott は，λ式のモデル作り，つまり数学的な構造による解釈の理論をはじめた．これによってλ計算体系に全く新しい視点が与えられ，外延意味論が誕生した．その詳細は Stoy[20] などにゆずる．本誌においても高橋[21]，伊藤[8] の解説がある．

λ式で定義可能な関数*は，以下に示す2個の関数 K, S から合成することができる．これは Schönfinkel が早くも1920年に発表したことである．

ⅰ) $K\equiv\lambda x\cdot\lambda y\cdot x$

ⅱ) $S\equiv\lambda x\cdot\lambda y\cdot\lambda z\cdot(x(z))(y(z))$

たとえば，恒等関数

ⅲ) $I\equiv\lambda x\cdot x$

も K と S の組合せで合成できる ($I=S(K)(K)$ となる)．

λ計算では関数の入力領域（定義域）と出力領域(値域）を指定する形式のものと指定しない形式のものが考えられる．前者を**有型 (typed) λ計算**と呼び，後者を**無型 (typeless) λ計算**と呼ぶ．しばらくは有型λ計算をもとに話しを進める．

2.2 有型λ計算と直観命題論理[12]

A，B を型 (type) とするとき，A から B への関数の型を $B\Leftarrow A$ とあらわすことにしよう．また a が A の型であるとき，$a\in A$ のようにあらわす．すると次のことが言える．

0) $a\in A, f\in B\Leftarrow A\ \vdash f(a)\in B$;

つまり，a の型が A，f の型が $B\Leftarrow A$ ならば，$f(a)$ の型は B である．(記号 $\vdash$ は左辺から右辺を導出することができることを示す)．

また，以下の I_A, K_{AB}, S_{ABC} のような定項を考えることができる．I_A, K_{AB}, S_{ABC} は前出の I, K, S に型を指定したものに他ならない．

1) $I_A\in A\Leftarrow A$ で，
$a\in A$ のとき，$I_A(a)=a$;

2) $K_{AB}\in(A\Leftarrow B)\Leftarrow A$ で，
$a\in A, b\in B$ のとき，$(K_{AB}(a))(b)=a$;

---

* 「関数 $f(x)$」という表現が良く用いられるが，これだと関数の形をあらわすのかそれとも x に対する関数値（出力）をあらわすのか区別がつかないため，λ を使って関数そのものを示すのである．

* ただし自由変数を含まないものに限る．自由変数とはλによって引数として指定されてない変数で，たとえば $\lambda x\cdot x+y$ の y のようなものである．

3) $S_{ABC}\in((A\Leftarrow C)\Leftarrow(B\Leftarrow C))\Leftarrow((A\Leftarrow B)\Leftarrow C)$ で，$c\in C, f\in(A\Leftarrow B)\Leftarrow C, g\in B\Leftarrow C$ のとき $((S_{ABC}(f))(g))(c)=(f(c))(g(c))$. この結果は型 A になる.

ここで，型を“命題”とみなし，$\Leftarrow$を含意 (imply) とみなせば

0′) $A, B\Leftarrow A \vdash B$;

1′) $A\Leftarrow A$;

2′) $(A\Leftarrow B)\Leftarrow A$;

3′) $((A)\Leftarrow C)\Leftarrow(B\Leftarrow C))\Leftarrow((A\Leftarrow B)\Leftarrow C)$;

について，0′) はいわゆる3段論法のひとつである Modus Ponens, 1′)〜3′) は命題論理，特に直観命題論理の恒真式 (の一部) に他ならない. 0)〜3) において $a\in A$ を，a は A の“証明”であると読むことにすれば，0′) を推論規則とし 1′)〜3′) を公理に持つ命題論理体系において，

λ 計算＝証明過程

であると考えることができる.

この論理体系は命題論理体系の一部をなすにすぎない. もう少しふつうの体系に近づけるために，論理積 ($\wedge$) や真理値 (T) を導入したものを考える. ここでは Gentzen 流の推論規則の表現を用いることにする. ラベルのついた矢印 ($\rightarrow$) は，左辺の論理式が真ならば右辺の論理式も真であることを意味し (entailment)*，その上に表示したラベルは後述するように証明過程を示すのに用いる. また，横棒 (——) は棒線の上の部分が言えれば下の部分が導出できるという推論規則を示す (既出の記号 $\vdash$ と同じである).

4) $A\xrightarrow{1_A}A$, $\dfrac{A\xrightarrow{f}B \quad B\xrightarrow{g}C}{A\xrightarrow{gf}C}$;

5) $A\xrightarrow{O_A}\mathsf{T}$;

6) $A\wedge B\xrightarrow{\pi_{A,B}}A$, $A\wedge B\xrightarrow{\pi'_{A,B}}B$, $\dfrac{C\xrightarrow{f}A \quad C\xrightarrow{g}B}{C\xrightarrow{\langle f,g\rangle}A\wedge B}$;

7) $(A\Leftarrow B)\wedge B\xrightarrow{\varepsilon_{A,B}}A$, $\dfrac{C\wedge B\xrightarrow{h}A}{C\xrightarrow{h^*}A\Leftarrow B}$.

この体系からたとえば論理積の交換則を次のように証明することができる.

$$\dfrac{\dfrac{A\wedge B\xrightarrow{\pi'_{A,B}}B \qquad A\wedge B\xrightarrow{\pi_{A,B}}A}{A\wedge B\xrightarrow{\langle\pi'_{A,B},\ \pi_{A,B}\rangle}B\wedge A}}{}$$

この下部にあらわれた矢印のラベル $\langle\pi'_{A,B},\pi_{AB}\rangle$ が証明過程を示すと見ることができよう.

* entailment は一般的には次のようなものである. Γ, Δ を論理式の集合とした時，$\Gamma\rightarrow\Delta$ というのは，「Γ 内のすべての論理式が真ならば，Δ 内の論理式のいずれかが真である」という意味である.

また，先きの 2′) と 3′) が恒真 (tautology) であることが言える. ここでいう恒真とは $\mathsf{T}\rightarrow\varphi$ なる φ を言う.

4)〜7) は命題論理，特に直観命題論理，の体系の一部をなしている. このように限定したのは，4)〜7) が次のように λ 計算体系と密接な関係があるからである.

4) の 1_A は恒等関数 I である. また gf は関数の合成で，$g\circ f$ とも書く.

5) の 0_A は値が定数の関数である.

6) の $A\wedge B$ は，型の直積としてみるならば $\pi_{A,B}$ は順序対の左，$\pi'_{A,B}$ は右を値とする関数である. 順序対は関数の体系のなかで大きな役割を持つ. 順序対を合成して一般に n 項組を作ることができる.

7) における $A\Leftarrow B$ は A^B つまり型 B から型 A への関数の型としてみることができる. $\varepsilon_{A,B}$ は，$f\in A\Leftarrow B$ と $x\in B$ が与えられた時 $f(x)$ を評価 (evaluate) する関数である. LISP で言えば関数 apply に相当する. また，h は $x\in C, y\in B$ の2引数をとる関数であるが，λ 計算では多引数関数は1引数関数の合成であらわす (Curry 化と呼ばれる) ことにしており，$h(x,y)$ を $(h^*(x))(y)$ とする. これを可能にすることを 7) の後半が示していることになる.

先きの 2) 3) が成立することから，(有型の) K と S の存在が言え，この意味でこの体系は有型 λ 計算体系を構成する.

このように，4)〜7) は，$A, B, C\cdots$ を命題とみなし，$\wedge$ を論理積，$\Leftarrow$ を含意とみなせば命題論理の部分体系をなすものと見ることができ，一方，$A, B, C\cdots$ を型，$\wedge$ を型の直積，$A\Leftarrow B$ を型 A^B とみなせば，λ 計算体系をなすものとして見ることができる.

両者を統合する，つまり両者を抽象化してひとつの体系とみなすことはできないだろうか. それによって

証明過程＝λ 計算

のテーゼを統合的な視点から見ることができるようになるだろう.

それを論じる前に，4)〜7) を構成する，$1_A, 0_A$ あるいは $\pi_{A,D}$, $\varepsilon_{A,D}$ といった特殊な関数の相互の関係 (等値関係) についての要請を以下に与えよう.

4′) すべての $f: A\rightarrow B$, $g: B\rightarrow C$, $h: C\rightarrow D$ に対して，

$f1_A=f$, $1_Bf=f$, $(hg)f=h(gf)$;

5′) すべての $f: A\rightarrow\mathsf{T}$ に対して

$f=0_A$

6′) すべての $f: C\to A$, $g: C\to B$, $h: C\to A\wedge B$に対して,

$\pi_{A,B}<f,g>=f, \pi'_{A,B}<f,g>=g,$

$<\pi_{A,B}h, \pi'_{A,B}h>=h$;

7′) すべての $h: C\wedge B\to A$, $k: C\to A\Leftarrow B$ に対して,

$\varepsilon_{A,B}<h^*\pi_{C,B}, \pi'_{C,B}>=h,$

$(\varepsilon_{A,B}<k\pi_{C,B}, \pi'_{C,B}>)^*=k.$

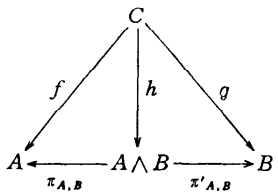
これらの要請は 4)～7) を命題論理体系としてみても，また λ 計算体系として見てもいずれも自然なものである.

4)～7) および 4′)～7′) によって実は**積閉包圏** (Cartesian closed category) が規定されたことになる.

まず，4)4′)によって**圏** (category) が定義される. 圏は**対象** (object), **射** (arrow, または morphism) および**射の合成**から構成される. 対象はここでは命題と型を抽象化したものとみることができる. 射は entailment と関数を抽象化したものである. 4′) の後半は射の合成を規定している.

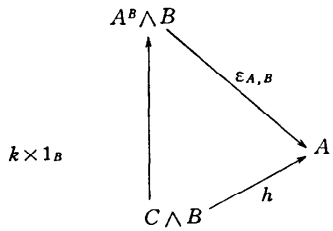
5)5′)における **T**は圏理論では終対象 (terminal object) と呼ばれる. 集合でいえば要素が 1 個の集合に相当する. 圏においては終対象は本質的には 1 個である. 圏理論では **T**をふつう **1** と書く.

6)6′)では圏における対象の直積(cartesian product)を規定する. これは集合の直積に相当する. $A\wedge B$ はふつう $A\times B$ と書く. 6)6′) は次の図式



が可換であることを示すものに他ならない (つまり，$f=\pi_{A,B}\circ h$ かつ $g=\pi'_{A,B}\circ h$ が成立つ).

7) 7′) は対象のベキ乗 (exponentiation) を規定する. $A\Leftarrow B$ は A^B とも書く. 7) 7′) は次の図式



が可換であることを示す. ここで $k\times 1_B$ は射の積を示し，射を集合間の写像として見る時には，入力 $<c, b>\in C\wedge B$ に対して出力が $<k(c), 1_B(b)>=<k(c), b>$ となるものに対応している.

2.3 無型 λ 計算と積閉包圏

前節では有型 λ 計算の型を命題に対応させて論じたが，無型 λ 計算についても同様の議論ができる.

Scott[17] の理論ではもともと無型 λ 計算をあつかっている. 型に対応するものとして“データ型”と呼ぶものを考える. これは λ 式で表現可能である. ここでいうデータ型はもちろんプログラム言語でいうデータ型を意識しているがここでは仮りにそう呼ぶものと考えておく. Scott のいうデータ型は位相空間論でいうレトラクトと密接な関係があり概念的には同じものである. この概念に基づくデータ型が前節での型に対応しそれが圏の対象とみなされて積閉包圏が以下のように構成できる.

データ型 A はある λ 項 (λ 計算での対象物) であり，次の式が成立つものとする.

$$A=\lambda x\cdot A(A(x)) \quad \cdots\cdots(\text{i})$$

この式の成立つものを圏の「対象」とする. 次に任意の射 $f: A\to B$ は

$$f=\lambda x\cdot B(f(A(x))) \quad \cdots\cdots(\text{ii})$$

をみたす λ 項であるとする. ここで,

$$1_A=A, \quad \cdots\cdots(\text{iii})$$

$$f\circ g=\lambda x\cdot f(g(x)); \quad \cdots\cdots(\text{iv})$$

とすれば，圏を作ることができる. すなわち，このような 1_A や射の合成について **2.2** の 4) 4′) が言える.

さらに，この圏を積閉包圏とするために，以下の定義を与える.

$A\times B\equiv\lambda u.\lambda z.z(A(u(\lambda x.\lambda y.x)))(B(u(\lambda x.\lambda y.y)))$,

ただし $\pi_{A,B}\equiv\lambda u.(A\times B)(u)(\lambda x.\lambda y.x)$,

$\pi'_{A,B}\equiv\lambda u.(A\times B)(u)(\lambda x.\lambda y.y)$;

$\mathbf{1}\equiv\lambda u.\lambda x.x$,

ただし

$0_A\equiv\mathbf{1}$;

$f: C\to A$, $g: C\to B$ のとき

$<f,g>\equiv\lambda t.\lambda z.z(f(t))(g(t))$;

$A\Rightarrow B\equiv\lambda f\cdot B\circ f\circ A$,

ただし

$\varepsilon_{B,C}\equiv\lambda u.C(u(\lambda x.\lambda y.x))(B(u(\lambda x.\lambda y.y)))$,

また

$h: A\times B\to C$ のとき

$h^*\equiv\lambda x.\lambda y.h(\lambda z.z(x)(y))$.

これらの定義によって 4)～7), 4')～7') が成り立ち，こうして積閉包圏が構成できる．証明は面倒であるが式の変形だけである．λ計算になじみのある人は試してみると良い．

ここで，“データ型”が圏における一種のレトラクトであることを述べておこう．

圏において，対象 A, B 間の**ひきこみ (retraction)** とは，$i: A\to B$, $j: B\to A$ とした時 $j\circ i=1_A$ が成り立つ対 (i, j) を言う．このとき A を B の**素縮またはレトラクト (retract)** と呼ぶ．

$$A \underset{j}{\overset{i}{\rightleftarrows}} B$$

特別の対象 U を

$$U\equiv\lambda x\cdot x$$

と定義する．

$$U\circ U=\lambda x\cdot U(U(x))=U$$

となるから，U は対象であるとして良い．するとこの圏においてはすべての対象(データ型)が U のレトラクトになる．なぜなら各対象 A について，$U\circ A\circ A=A$, $A\circ A\circ U=A$ であるから，

$$A: A\to U,\quad A: U\to A,$$

とでき，さらに，

$$A\circ A=A=1_A,$$

であるから，(A, A) が A, U 間のひきこみになっている．

特に U については，

$$U\Rightarrow U=\lambda f.\lambda x.f(x)$$

と書け，$U\Rightarrow U$ もまた U のレトラクトになることが言える．したがって $U\Rightarrow U$ (U の“関数空間”) が U のレトラクトである．このような U を「反射領域」(reflexive domain) と呼び，外延意味論で定義される反射領域に対応している．

以上みてきたように反射領域 U のレトラクトとしてデータ型を与え，それを圏の対象とみなし，データ型間の写像 (関数) を射とみなせば積閉包圏が構成できる．つまり，無型λ計算での“データ型”を有型λ計算の“型”とみなせば無型λ計算も有型λ計算も実質的には同じであると考えることができる．

3. 高階直観論理とトポス

前節では直観命題論理 (の部分体系) とλ計算体系との類似を検討し，両者を抽象化して積閉包圏となることを示した．しかし，論理側にしても関数側にしてもいずれもその性質のごく一部を見たにすぎない．論理側では，命題論理の部分系であるし，関数の方でも，自由変数が使えない．集合論で言うならば要素に関係する議論を全くしないようなものである．

その種の議論を進めるためには積閉包圏にいくつかの新しい概念を追加し，それらの関係を定義して行かなければならない．そうして行くと，積閉包圏にある条件を追加してできるトポス (topos) になる．このトポスは非常に有効な概念で，古典的な一階述語論理のみならず，一般に高階直観論理のモデルとしても使える．そして，うまいことに，関数体系 (λ計算体系) もまたこのトポスの中に埋め込むことができる．それを以下にみて行こう．

3.1 積閉包圏からトポスへ

積閉包圏は，対象を集合，射を集合間の写像とみなせば集合のあつまりを抽象化したものであるとみることができる．事実，任意の集合を対象とし，任意の集合間の写像を射とする圏 **Sets** はそのようにして積閉包圏となる．しかし集合を論じる時に必ず生じる部分集合の全体すなわちベキ集合の概念が積閉包圏にはあらわれてこない．つまり個々の集合の内部に自由に立ちいって論じることができない．そこでこのような概念を明確に導入する必要がある．そうしてできるのがトポスである．

トポスは正式には

積閉包圏＋部分対象分類器
(subobject classifier)

として定義される．ここで「部分対象」とは部分集合に対応する概念であり，部分対象分類器とは部分対象を識別するための圏理論における一種のしくみである．

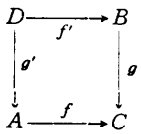
部分対象分類器の定義を与える前に，ふたつほど圏用語について述べなくてはならない．**単射** (monic) と **ひきもどし** (pullback) である．

$f: A\to B$ を圏 C の射；任意の g, h をともに $C\to A$ なる C の射としたとき，$f\circ g=f\circ h$ ならば $g=h$ になる f を**単射**という．

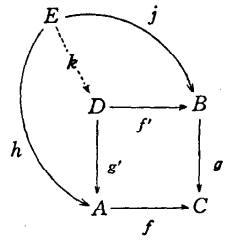
$$C \underset{h}{\overset{g}{\rightrightarrows}} A \xrightarrow{f} B$$

これは集合の単射に対応している．

ひきもどしの 4 角形



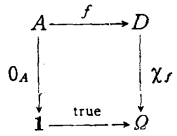
とは，この図形が可換（つまり，$f\circ g'=g\circ f'$）であること，そして



上の図で外側の4角が可換ならば必ず $h=g'\circ k$，$j=f'\circ k$ をみたす k がただひとつ存在するものを言う．$A\xleftarrow{g'}D\xrightarrow{f'}B$ を $A\xrightarrow{g}C\xleftarrow{f}B$ の**ひきもどし**と言う．

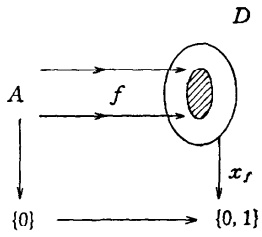
さて，この準備の上で，部分対象分類器の定義を与える．

圏 $\mathcal{C}$ は終対象 $\mathbf{1}$ を持つものとする：そのとき，任意の単射 $f: A\to D$ に対して，



なる4角形がひきもどしになるような射 $\chi_f: D\to\Omega$ がただひとつ存在するとき，$\mathrm{true}: \mathbf{1}\to\Omega$ と合わせて Ω を**部分対象分類器**と呼ぶ．

A, D を集合，$\mathbf{1}=\{0\}$，$\Omega=\{0,1\}$，$\mathrm{true}(0)=1$ として集合の世界に対応させてみると，A は D の部分集合と同型であるとみなせる．



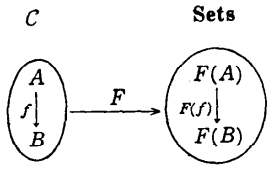
χ_f は A の特性関数に相当する．こうして **Sets** はトポスになる．部分集合の識別を射としてとらえる機構が部分対象分類器に他ならない．しかし一度抽象化するとそれが1人歩きをはじめる．たとえば $\Omega=\{0,1\}$ という2分法に限らず，Ω をもっと自由にとれる．この Ω はトポスの性格を示すものになる．

ある任意の圏 $\mathcal{C}$ から，圏 **Sets** への対応を考え，次のようにしてあるトポスを作ることができる．

$\mathcal{C}$ から **Sets** への対応 F を次のように構成する．

（1）　$\mathcal{C}$ の各対象 A に対して $F(A)$ は **Sets** のなかのひとつの集合

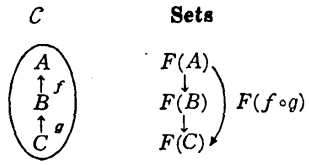
（2）　$\mathcal{C}$ の各射 $f: B\to A$ に対して $F(f): F(A)\to F(B)$ は **Sets** のひとつの射（向きが逆になることに注意）



（3）　$F(1_A)=1_{F(A)}(F(A)\to F(A)$ なる恒等写像）

（4）　$\mathcal{C}$ において $f: B\to A$，$g: C\to B$ ならば **Sets** において

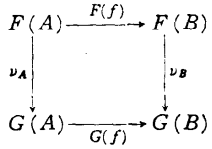
$F(f\circ g)=F(g)\circ F(f)$　　である．



このような対応 F を $\mathcal{C}$ から **Sets** への（反変）**関手**（functor）と呼んでいる．関手“全体”が実は圏を作り，これは積閉包圏でもありさらにトポスにもなる．それを通常 $\mathbf{Sets}^{\mathcal{C}^{op}}$ とあらわす*．積閉包圏になることを以下に述べよう．

まず，$\mathbf{Sets}^{\mathcal{C}^{op}}$ の対象は関手である．そして射は次のように定義される関手間の“自然変換”である．

F と G を関手とした時，自然変換 $\nu: F\to G$ とは，$\mathcal{C}$ において $f: B\to A$ のとき，次の図式が **Sets** においてかならず可換になるように $\mathcal{C}$ の各対象 A に対して $\nu_A: F(A)\to G(A)$ を対応させるものである．

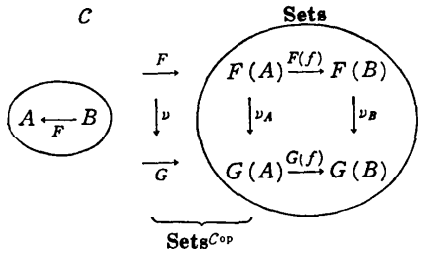


これはわかりにくいかも知れないが，次の図を参考

---

* 圏 $\mathcal{C}$ のすべての射の向きを逆にして圏を作ることができる．これを通常 $\mathcal{C}^{op}$ とあらわす．したがって，$\mathbf{Sets}^{\mathcal{C}^{op}}$ とは $\mathcal{C}^{op}$ から **Sets** への関手（とその間の射）から作る圏であるとも言える．

にして意味をくみとって欲しい.



Cの中で“状態” Aが $f: B\to A$によって“状態” Bに“推移”したとする（話がややこしいが，$f: B\to A$ のとき，AからBへ移るとみなす．**Sets** 側で逆になるからである）．すると，ある関手 F を決めておけば，Cにおけるこの推移を集合 $F(A)$ から集合 $F(B)$ への推移 $F(f)$ として反映させることになる．別な関手GによればCの推移fを別なかたちで集合の推移に反映させる．そこでFという反映の仕方からGという反映の仕方への変換を考えると，**Sets** の中で集合 $F(A)$ 内の任意の点から $\nu_B{}^0F(f)$ という道を辿って着く $G(B)$ の中の点と，$G(f)\circ\nu_A$ という道を辿った時の点とが一致するように，ν_A, ν_B を選ぶのが自然であろう．圏 **Sets** では任意の集合間の任意の写像を持っているから，そのような ν_A, ν_B を選ぶことができる．そこで，Cの対象Aに ν_A，Bに ν_B を対応させるものをνとする．この対応はCの任意の対象に与えることができる．こうしてできたのが自然変換 $\nu: F\to G$ である．ν_A, ν_B などを ν の**成分**と呼ぶ．そしてさらに，任意の関手を対象とし，関手間の自然変換を射として作られる圏が $\mathbf{Sets}^{C^{op}}$ である．

まず，$\mathbf{Sets}^{C^{op}}$ が圏となることを確かめておこう．

（1） 任意の関手 F に対して自然変換 $1_F: F\to F$ はCの任意の対象Aに対して，$(1_F)_A=1_{F(A)}: F(A)\to F(A)$ を対応させるものである．これが必要な性質をみたすことは容易に確かめられよう．

（2） 任意の自然変換 ν, μ, ξ について．

$$\nu\circ(\mu\circ\xi)=(\nu\circ\mu)\circ\xi$$

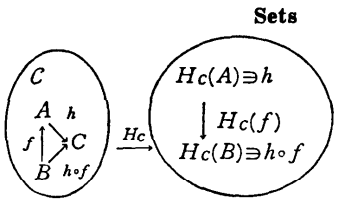
が成立つことも明らかである．

このような関手および自然変換の具体例をひとつ述べる．これは後に使用する．

Cの対象 A, C について，

$$H_C(A)=\{h\mid h: A\to C\}$$

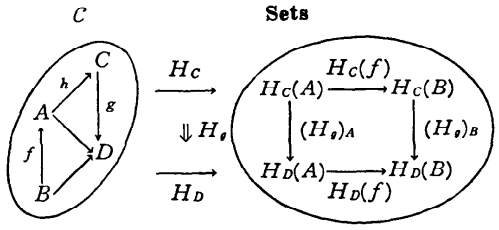
とする．つまり，$A\to C$ なる射すべてのあつまりである．**Sets** はすべての集合を対象として持つからこれも **Sets** の対象のひとつである．次に $f: B\to A$ のとき，$H_C(f)$ は $h\in H_C(A)$ を $h\circ f\in H_C(B)$ に移す写像であるとする．



こうして H_C は（反変）関手となる．

自然変換は次のように考える．

Cにおいて $g: C\to D$ とする．このとき $H_g: H_C\to H_D$ を次のようにきめる．Cの任意の対象Aについて，H_g の成分 $(H_g)_A$ は $h\in H_C(A)$ を $g\circ h\in H_D(A)$ に移すものとする．そして，



上の図の **Sets** 内で，$h\in H_C(A)$ を $(H_g)_A$ によって $g\circ h\in H_D(A)$ にうつし，さらにそれを $H_D(f)$ によって $g\circ h\circ f\in H_D(B)$ にうつすルートと，hを $H_C(f)$ によって $h\circ f\in H_C(B)$ にうつし，さらにそれを $(H_g)_B$ によって $g\circ h\circ f\in H_D(B)$ うつすルートの両方が（Cを見ることによって）可能であり，両者のゴールが一致する．したがって H_g は自然変換になる．

次に関手圏 $\mathbf{Sets}^{C^{op}}$ が積閉包圏になるかどうかを見て行く．

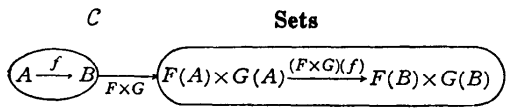
（3） まず終対象（「単位関手」）$\mathbf{1}$はCの任意の対象Aに対して $\mathbf{1}(A)=\{0\}$ となるものとする．また任意の射 $f: B\to A$ に対して $\mathbf{1}(f): \{0\}\to\{0\}$ とする．さらに，任意の関手 F から $\mathbf{1}$ への自然変換を ν とすれば，Cの各Aに対して，その成分は $\nu_A: F(A)\to\{0\}$ になる．

（4） 関手 F, G の積 $F\times G$ については，Cの各Aに対して

$$(F\times G)(A)=F(A)\times G(A)$$

（ただし右辺の×は集合の直積）

とし，さらに，$a\in F(A)$, $b\in G(A)$ そして $f: B\to A$ ならば，



$(F\times G)(f)(\langle a,b\rangle)=\langle F(f)(a),G(f)(b)\rangle$

とする．自然変換 $p: F\times G\to F$ および $q: F\times G\to G$ についても同様に C の対象と **Sets** の集合との対応をとって考え，たとえば

$p_A=\pi_{F(A)G(A)}: F(A)\times G(A)\to F(A)$

のように定義する．さらに，自然変換の対 $\langle\mu,\nu\rangle$ の定義も全く同様に“点”（対象↔集合）ごとに考え，$\mu: W\to U$，$\nu: W\to V$ とすれば，$\langle\mu,\nu\rangle: W\to U\times V$ は

$\langle\mu,\nu\rangle_A=\langle\mu_A,\nu_A\rangle: W(A)\to U(A)\times V(A)$

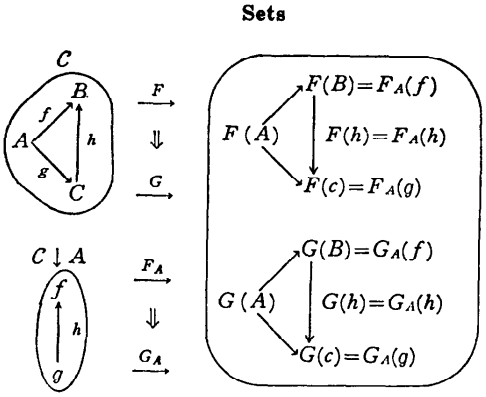
として定義する．以上の定義により $\mathbf{Sets}^{C^{op}}$ における積の性質がみたされる．

（5） 関手 F,G について $F\Rightarrow G$ はどのように定義したら良いだろうか．これは今までのように簡単に点ごとに与える，つまり，$(F\Rightarrow G)(A)=(F(A)\Rightarrow G(A))$（集合 $F(A)$ から集合 $G(A)$ への写像全体）とするわけにはいかない．$(F\Rightarrow G)(A)$ は，C の中の“状態” A からのさまざまの“推移”（Aに入る射）全体を何らかのかたちで **Sets** の中で反映させるべきものであろう．そのように考えると $F(A)\Rightarrow G(A)$ では Aからの推移を反映するものがなく情報不足である．そこで，ちょっと複雑になるが，$(F\Rightarrow G)(A)$ を次のように定義して行く．以下ではしばらく Aを固定して考える．

まず $C\downarrow A$という圏を作る．これは Aに入る Cの射を対象とする．$f: B\to A$, $g: C\to A$ とすれば，f も gも $C\downarrow A$での対象である．この時，$h: C\to B$, $h\circ f=g$ とすれば $C\downarrow A$ の中で gから f への射を hとする．こうして圏 $C\downarrow A$を任意の Aについて作ることができる．

次に，関手 $F: C\to$**Sets** に対応して，関手 $F_A: C\downarrow A\to$**Sets** を作る．これは Cの任意の $f: B\to A$ に対し $F_A(f)=F(B)$ とし，また Cで $h\circ f=g$, すなわち $C\downarrow A$ で $h: g\to f$ ならば $F_A(h)=F(h)$ となるものとする．

$C\downarrow A$ は Aからの推移（つまり Aに入る射）に関する情報をすべて持ち，しかも，それだけに限られているから，F_A は Fに沿ってその情報を **Sets** の中に忠実に反映させるものになる．同様に Gに対して G_Aを作る．

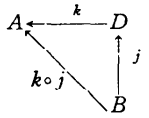


最後に仕あげとして，“関手” $F\Rightarrow G$ の定義を与える．Cの任意の Aについて，

$(F\Rightarrow G)(A)=\mathbf{Nat}\,[F_A,G_A]$

（≡「F_A から G_A へのすべての自然変換からなる集合」）

とし，C の射 $k: D\to A$ に対しては，$(F\Rightarrow G)(k)$ は $\mathbf{Nat}\,[F_A,G_A]$ から $\mathbf{Nat}\,[F_D,G_D]$ への写像とする．ただしここで ν を F_A から G_A への，ν' を F_D から G_D への自然変換とした時，次の図式



において，$C\downarrow D$ の対象 jについて，$\nu_j'=\nu_{k\circ j}$ であるという条件をつける．こうして“関手” $F\Rightarrow G$ が定義された．実際にこれが関手であることは容易に確かめられる．

（6） 関手の積とベキが定義されたので，$\mathbf{Sets}^{C^{op}}$ が積閉包圏になるための条件として，この上で評価関数や，2引数の関数を1引数の関数に分解するものを与えなければならない．

評価関数を εであらわすことにしよう．$\mathbf{Sets}^{C^{op}}$ において εは $(F\Rightarrow G)\times F\to G$ なる自然変換である．これを **Sets** の中で見ると，C の対象 Aについてその成分は，

$\varepsilon_A: ((F\Rightarrow G)\times F)(A)=(F\Rightarrow G)(A)\times F(A)\to G(A)$

となる．そこで，$x\in F(A)$ および $\tau\in(F\Rightarrow G)(A)$ ならば

$\varepsilon_A(\langle\tau,x\rangle)=\tau_{1A}(x)$.

と定義する．ここで τは F_A から G_A への自然変換であり，また τ_{1A} はその τが $C\downarrow A$ の対象 1_A に対応づけられた成分であるから，確かに $\tau_{1A}: F(A)\to$

$G(A)$ であり，この定義が有効であることがわかる．

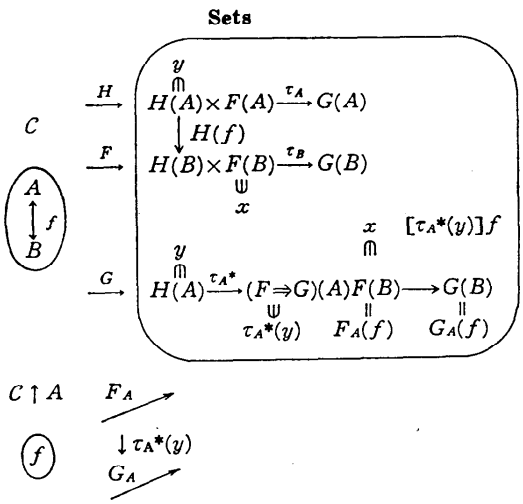
次に自然変換 $\tau: H\times F\to G$ に対する $\tau^*: H\to(F\Rightarrow G)$ を定義する．$\mathcal{C}$ の対象 A に対応する成分は

$$\tau_A^*: H(A)\to(F\Rightarrow G)(A)$$

である．すると，各 $y\in H(A)$ に対して $\tau_A^*(y)$ は F_A から G_A への自然変換になる．そこで $\mathcal{C}\downarrow A$ の各対象 $f: B\to A$ について，$[\tau_A^*(y)]_f$ を対応させれば，これは $F_A(f)=F(B)$ から $G_A(f)=G(B)$ への写像であり，$x\in F(B)$ について

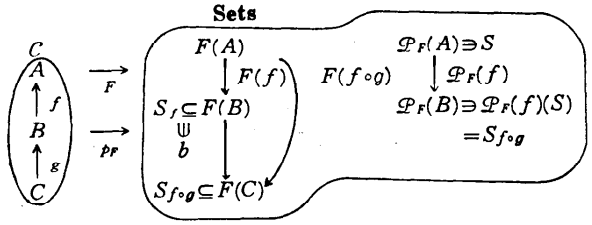
$$[\tau_A^*(y)]_f(x)=\tau_B(\langle H(f)(y),x\rangle)$$

となるものと定義する．一見ややこしいが，図を参照してみれば，τ と τ^* との連関を与えるこの定義は当然であることがわかる．



以上で $\mathbf{Sets}^{\mathcal{C}^{op}}$ が積閉包圏となることの説明を終える．この他 $\mathbf{Sets}^{\mathcal{C}^{op}}$ の性質として，後の説明で必要となるベキ対象 $\mathcal{P}_F$（F は $\mathbf{Sets}^{\mathcal{C}^{op}}$ の任意の対象＝関手）を持つことが言える．**ベキ対象** (power object) とは集合でいえばある集合 S の部分集合すべてからなる集合 $\mathcal{P}_S$ に対応する概念である．

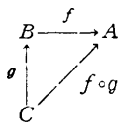
任意の関手 F に対して $(\mathcal{P}_F)(A)$ の要素は，$f: B\to A$ に対応した $S_f\subseteq F(B)$ の全体 S で，$b\in S_f$, $g: C\to B$ としたとき



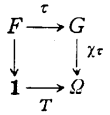
$F(f)(b)\in S_{f\circ g}$ となるものである．また $\mathcal{P}_F(f)$ については，$S\in\mathcal{P}_F(A)$ のとき，$(\mathcal{P}_F(f)(S))_g=S_{f\circ g}$ が成立つものとする．以上で $\mathcal{P}_F$ が "関手" として定義された．これが実際に関手であることは容易に確かめられよう．

ベキ対象の存在が言えると，積閉包圏＋ベキ対象によってトポスとなる．つまりベキ対象を使って部分対象分類器 Ω を作ることができる．このトポスでの Ω がどうなっているかをみておこう．

$\mathcal{C}$ の A について，S_A を A に入る射すべてからなる集合とする．つまり，$\mathcal{C}\downarrow A$ の対象すべてからなる集合である．この S_A の部分集合 S で，$f\in S$, $f: B\to A$ ならば，$\mathcal{C}$ の任意の射 $g: C\to B$ に対して必ず $f\circ g\in S$ となるような S を **A-crible** と呼ぶ．A-crible には少なくとも ϕ（空集合）と S_A のふたつがある．



このとき，"関手" Ω は，任意の A について $\Omega(A)$ ＝「A-crible の全体」であり，また，$f: B\to A$ について $\Omega(f): \Omega(A)\to\Omega(B)$ となるものとして定義できる．



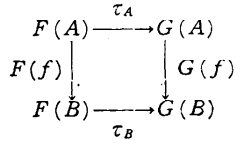
上図がひきもどしの4角形であるように自然変換 τ, T, χ_τ を考えたとき，T の成分 T_A は，$T_A: \{0\}\to\Omega(A)$ について，

$T_A(0)=S_A$　($\Omega(A)$ 中の A-crible の中で最大のものである)

とする．Ω とこの T で $\mathbf{Sets}^{\mathcal{C}^{op}}$ の部分対象分類器になる．このとき χ_τ の成分 $(\chi_\tau)_A$ は，$x\in G(A)$ として，

$$(\chi_\tau)_A(x)=\{f: B\to A\mid G(f)(x)\in\tau_B(F(B))\}$$

となるようにすれば良い．



$\mathbf{Sets}^{\mathcal{C}^{op}}$ におけるこの Ω（と T）は，後述する

Kripke モデルのある本質を示すものになる.

また，ベキ対象との関連については，≅を圏の意味での対象間の同型として，一般に

$$\Omega \cong \mathcal{P}_1$$

となる（説明は省略する）．したがってベキ対象を定義することによってもトポスを作ることができる.

ともかく以上のように構成した関手圏はトポスの代表例である．その便利な点は C が任意の圏（積閉包圏である必要はない）であり，また関手によって移る先きが集合なので，何かと話しをしやすい点にある．$\mathbf{Sets}^{C^{op}}$ の関手は，直観的に言えば，C の中の“推移”（射）を **Sets** の中の集合の推移に忠実に伝えるものであり，この意味で“可変集合”とみなすことができる．実際に，**Sets** もトポスの例であるが，$\mathbf{Sets}^{C^{op}}$ はそれを可変構造にしたと言う意味でその関手は集合の拡張であるとみなすことができよう．事実そういうアイデアにもとづいて圏理論によって作られた概念がトポスに他ならないのである.

3.2 高階直観論理

2.2 では λ 計算体系と命題論理の部分系とが相似関係にあることを見，両者がともに積閉包圏をなすことを示した．以下では λ 計算体系を埋め込める論理体系を構築する．これによって，手続き的知識の表現（関数体系）を宣言的知識の表現（論理体系）の中に自然なかたちでとり込むことが可能になる.

そのためにはまず高階性が要求される．関数体系（λ 計算体系）では任意の関数を許容し，たとえば引数に関数を用いることのできる関数も認められるからである.

また，詳しい説明は略すが，排中律を要請する古典論理では任意の関数を埋込むことに無理が生じ，直観論理が必要になる.

こうして関数を埋込む論理体系として高階直観論理を考えることになる.

オランダの数学者 Brouwer がはじめた直観論理は，ある命題が真か偽かを決定するのは証明が構成できた時に限るという構成主義が基本になっている．たとえば有名なフェルマの定理 α はいまもって証明されていないから，α は真とも偽とも言えない．するとこの場合 $\alpha \vee \sim\alpha$ も構成的な証明が与えられず（$\sim\alpha$ は「α は偽である」の意味），$\alpha \vee \sim\alpha$ は天下り的な公理とは認めがたい．「α が真ではない」というのは，「いまの所構成的な証明が与えられていない」ということを意味するだけで，これがすなわち「α は偽である（$\sim\alpha$）」ということにはならない．明日にも α は真であると言えるかも知れないからである．見方を変えれば，真と偽の間の“灰色地帯”を認める立場であるとも言えよう.

こういった思想的な“気持ち”を定式化するにはどうすれば良いだろうか．一般には Hilbert の公理系から排中律をとったものであると言え，また，Gentzen の自然演繹法（LK と呼ばれる）で，$\Gamma \to \Delta$（Γ の中の論理式のすべてが真ならば，Δ の中の少なくともひとつの論理式が真である）という推論規則中の式において，Δ にはたかだかひとつの論理式しか含まないような推論規則体系としても定式化できる（LJ と呼ばれる）[14].

さて，本稿で考察するのは多種高階直観論理（many-sorted higher-order intuitionistic logic）である．ここで多種（many-sorted）というのは，記述に用いる記号がそれぞれある型（タイプ）を持つ（と想定する）ことを言う．また，高階というのは，型のある集合がまた型のひとつであることを認めることである．そうすると，型の集合の型の集合の……という具合にいくらでも高くして行くことができる．ただし，型の集合の型は自分自身を要素として含まない.

型の理論は Russel が自分の見つけたパラドックスを避けるためにはじめたもので，特に高階論理を論じる時に必要な手法になっている.

多種高階直観論理（あまりにも長すぎるので以下では単に“高階論理”と略す）はこのように Brouwer と Russel のアイデアを源流にしているが，現代では数学的に整備され安定した論理体系になっている．特に 1970 年代になってその意味解釈にトポスが有効なことがわかった．つまりこの論理体系（公理と推論規則からなる）の theory* $\mathcal{T}$ に対してある対応づけによってトポス $E(\mathcal{T})$ を作ることができ，逆に任意のトポス E から theory $\mathcal{T}(E)$ を作ることができ，しかも $\mathcal{T}(E)$ から圏 $C(\mathcal{T}(E))$ を作れば，$C(\mathcal{T}(E))=E$ となる[22]．一般に，記号操作の世界である論理体系に対してある種の数学的枠組（たとえば集合やトポスなど）を対応づける意味構成をモデルと呼びその対応によって記号の“意味”を与える．したがって高階論理の“意味”を与えるものとしてトポスが有効なことがわかる.

---

* 一般に論理体系の theory Γ とは，論理体系から導出される論理式の集合で閉じているものをいう．つまり，φ を論理式とすれば，$\Gamma \vdash \varphi$ ならば $\varphi \in \Gamma$ である.

3.3 Kripke モデル

高階論理とトポスとの一般的な対応づけについては，たとえば竹内[22]などを参考にしてもらうことにして，ここでは，関数を論理体系のなかに埋め込むことを主眼にし，またトポスとして $\mathbf{Sets}^{\mathcal{C}^{op}}$ を考えることにする．

後に与えるモデルは Kripke のモデルを土台にしている．様相論理のモデル作りをした Kripke は 1960 年代の中頃，同様な考え方で直観論理のモデル作りを行った．

Kripke の発想は，ある論理式 α の真偽が決まるのは必ずしも絶対的なものではなく，α について言及する時点や状態によって変わり得るものであるとする．この「言及する時点」や「状態」を「仮想世界 (possible world)」と呼ぶ．こういう考え方は，われわれの日常生活で良く体験することであり，むしろ自然な発想であると言えよう．

この立場に立って直観論理を見るとどうなるだろうか．ある論理式 α の真偽は構成的に証明された時はじめて確定する．この証明の段階(あるいは知識の状態)をそれぞれ仮想世界とみなす．すると，仮想世界は時間とともに推移するが，現在の段階から未来の段階をみた時，それは不確定で，複数個考えられる．こうして仮想世界全体のあつまりをみた時，段階(仮想世界)の推移を順序関係としてみれば，半順序集合になる．そして，もしある段階 p で α が真ならば，α は証明されたわけであるから，それに続く任意の段階 q でも α は真であるとする．また，もし仮想世界が一個しかないとすれば，古典論理と全く同じになるとする．この意味で直観論理は古典論理の拡張になっている．

順序関係を射としてみれば，半順序集合を圏とみなすことができる．それを $\boldsymbol{p}$ とする．さらに，$\boldsymbol{p}$ の任意の対象 p (つまり仮想世界) に対応して，仮想世界を構成する個体の集合を考えることができる．これは圏 **Sets** の対象としてみることができる．そこで $\boldsymbol{p}$ から **Sets** への対応として関手 F を考えてみる．この関手は，$\boldsymbol{p}$ での動き (推移) を **Sets** での集合の推移に忠実に反映させるものである．こういう任意の関手全体とそれらの間の自然変換を与えれば，圏 $\mathbf{Sets}^{\boldsymbol{p}}$ が得られ，ひとつのトポスになる．このトポスの上で Kripke モデルにもとづく論理体系を定式化することができる．その詳細は別書 (Goldbratt[6] など) にゆずるが，同様な考え方で組み立てた高階直観論理体系については，後に詳しく述べる．

ここで Kripke モデルと古典論理とのちがいを部分対象分類器についてみておく．古典論理では，$\Omega=\{0,1\}$，true $(0)=1$ として部分対象分類器を構成した．Kripke モデルにおいて，順序関係を $\sqsubseteq$ であらわした時，$p\sqsubseteq q$ のとき射 $p\leftarrow q$ として半順序集合 $\boldsymbol{p}$ を圏とみなせば，トポス $\mathbf{Sets}^{\boldsymbol{p}}$ においては，$\boldsymbol{p}$ の“対象” p に対して $\Omega(p)=$「p-crible の全体」となり，$T_{\boldsymbol{p}}(0)=S_p$，つまり，$p$ に入る射全体からなる集合となる．いいかえれば

$$\begin{array}{ccc} \{0\} & \xrightarrow[\text{true}]{} & \{0,1\} \\ \text{古典論理} & & \end{array} \qquad \begin{array}{ccc} \mathbf{1}_{\boldsymbol{p}} & \longrightarrow & \Omega_{\boldsymbol{p}} \\ \| & & \| \\ \{0\} & & T_{\boldsymbol{p}} \\ & & \{\phi,\cdots,S_p\} \\ & & \text{直観理論} \end{array}$$

仮想世界 p からの推移全体となる．このように真理値の 2 分法から離れ，真理値が相対化される．$\Omega_{\boldsymbol{p}}$ の要素を継承 (heritage) と呼ぶこともある．

Kripke モデルで恒真 (valid) な論理式というのは，どの仮想世界に対しても真であることが言えるものである．すると，たとえば $\alpha\vee\sim\alpha$ は必ずしも恒真ではない．こうしてモデルから恒真式を作ると，古典論理の公理のうち排中律 $(\alpha\vee\sim\alpha)$ 以外のものは恒真となり，Kripke モデルによる解釈は直観論理の解釈として妥当なことがわかる．

3.4 高階論理とトポス $\mathbf{Sets}^{\mathcal{C}^{op}}$

さて，以上の準備のうえで，いよいよ関数と論理との調和をはかる高階の論理体系を与えることにする．ここで述べるものは Joyal-Reyes の強制充足 (forcing satisfaction) と呼ばれるもので，基本的な考え方は Kripke モデルによっている．

(1) **基礎型** (ground type) は，圏 $\mathcal{C}$ の対象に対して 1 対 1 の対応があるものとする．基礎型も対象も $A,B,C,\cdots$ などであらわす．また基礎型は同時に“仮想世界”ともみなされる．

(2) 基礎型から一般的な**型**を次の手段で合成する．

- 基礎型は型である．
- T,S を型とした時，$\mathbf{1}$, $T\times S$, $T\Rightarrow S$, $\mathcal{P}_T$ もまた型である．

(3) **原子論理式**として以下のものを用いる．ただし，$x,y\cdots$ はある型を持つ変数である．

- $\perp$，特別の定数．
- $x=y$，x と y は同じ型とする．
- $y=fx$，f は定数記号で，x の型を A, y の型を B とした時，射 $f: A\to B$ に対応するものとする．

• $z=\langle x,y\rangle$, xの型を T, yの型を S, zの型を $T\times S$ とする.

• $z=x(y)$, zの型を S, yの型を T, xの型を $T\Rightarrow S$ とする.

• $y\in x$, yの型を T, xの型を $\mathcal{P}_T$ とする.

(4) 一般の**論理式**は次のようにして作る.

• 原子論理式は論理式である.

• Φ, Ψ を論理式としたとき, $\Phi\wedge\Psi$, $\Phi\vee\Psi$, $\Phi\Rightarrow\Psi$, $\forall x.\Psi$, $^{\exists}x.\Psi$ もまた論理式である. ここには〜(否定)がないが, $\sim\Phi$ は $\Phi\Rightarrow\bot$ と同じとみなす.

いうまでもなく型や論理式はここで与えられた方法で作れるものに限る.

以上から記号の世界が形成できる. ここで, 以下のようにして記号の"意味"を与える.

(5) $A\Vdash\Phi[s]$ と書いた時, その意味は, sを付値関数 (valuation function) とした時, 世界 (あるいは時点) Aで Φ が成立する, ということである. ここに付値関数 sというのは, 式 Φ の中にあらわれる変数を世界Aに対応する集合の中の個体に対応させる関数である. xには型があるからその型によって値が異なる. そこで, xの型がTとすれば, H_T なる関手 (代表関手—前述—) を用いて, $s(x)$ は $H_T(A)$ なる集合に属することにする. ただし, 関手の合成は型の合成に合わせる. つまり

$$H_T\times s=H_T\times Hs,$$
$$H_T\Rightarrow s=(H_T\Rightarrow Hs).$$
$$H_{\mathcal{P}_T}=\mathcal{P}_{H_T},$$

そして,

$$H_1=\mathbf{1}\ (\text{関手として}).$$

たとえば, xの型が $B\times C$ ならば, 個体 $s(x)$ は $H_{B\times C}(A)=(H_B\times H_C)(A)=H_B(A)\times H_C(A)$, つまり, 集合 $H_B(A)$ と $H_C(A)$ の直積の要素になる. したがって, $s(x)=\langle f,g\rangle$, ここに $f:A\to B$, $g:A\to C$ なるある射である.

(6) 原子論理式の意味を次のように定める. ここで iff は if and only if の略で, 必要十分条件を与える.

$A\Vdash\bot[s]$　　iff　偽 (false)
$A\Vdash x=y[s]$　　iff　$s(x)=s(y)$
$A\Vdash y=fx[s]$　　iff　$s(y)=f\circ s(x)$
$A\Vdash z=\langle x,y\rangle[s]$　　iff　$s(z)=\langle s(x),s(y)\rangle$
$A\Vdash z=x(y)[s]$　　iff　$s(z)=s(x)_{1A}(s(y))$

zの型を T, yの型を S とすれば, x の型は $S\Rightarrow T$ であるから $s(x)\in H_{S\Rightarrow T}(A)=(H_S\Rightarrow H_T)(A)=\mathbf{Nat}[(H_S)_A,(H_T)_A]$ となる. ここで $(H_S)_A,(H_T)_A$ ともに, $C\downarrow A\to\mathbf{Sets}$ の関手である. $(H_S)_A\to(H_T)_A$ の自然変換のひとつを νとすれば, $1_A:A\to A$ であるから, ν_{1A} は $H_S(A)\to H_T(A)$ なる写像のひとつになる. したがって, $s(z)=s(x)_{1A}(s(y))$ は有効である. (以上は関手のベキの復習である).

$A\Vdash y\in x[s]$　iff　$s(y)\in s(x)_{1A}$

ここで左側の$\in$は単なる記述記号で, 右側の$\in$は集合の要素を示すものである. yの型を T, xの型を $\mathcal{P}_T$ とする. $s(x)\in H_{\mathcal{P}_T}(A)=\mathcal{P}_{H_T}(A)$. ベキ対象の復習をすると, $s(x)_{1A}$ は $H_T(A)$ の部分集合である. $S(y)\in H_T(A)$ であるから, 上式右辺は有効である.

このようにして原子論理式に意味を与えることができた.

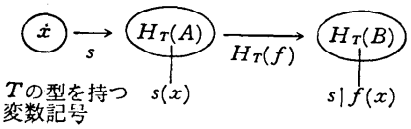
(8) 合成論理式の意味を次に与える. Φ, Ψ を任意の論理式とする.

$A\Vdash[\Phi\wedge\Psi][s]$　iff　$A\Vdash\Phi[s]$ かつ $A\Vdash\Psi[s]$

$A\Vdash[\Phi\vee\Psi][s]$　iff　$A\Vdash\Phi[s]$ または $A\Vdash\Psi[s]$

ここまでは当然であろう.

$A\Vdash[\Phi\Rightarrow\Psi][s]$　iff　$f:B\to A$ で $B\Vdash\Phi[s|f]$ ならば単に $B\Vdash\Psi[s|f]$. ここで, $s|f$ は"sのfによる制限"を示す. つまり, xの型をTとすれば, $s|f(x)=H_T(f)(s(x))$ となる. 直観的な解釈をすれば,



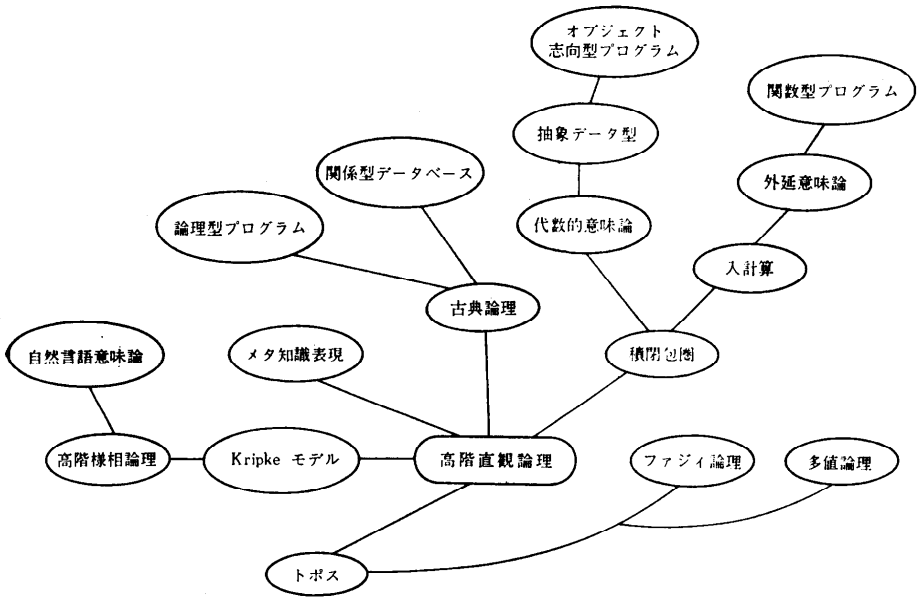
変数xに対応する個体が"世界" Aから"世界" Bへ移る時の移り方を示すことになる. それは当然 $f:B\to A$ に従うことになる.

$A\Vdash\forall x\cdot\Phi[s]$　iff　$B\to A$ なる任意のBについて, $f:B\to A$とし, $b\in H_T(B)$ ならば常に, $B\Vdash\Phi[s|f(b/x)]$

ここで, $s(b/x)$ は, 変数xの値がbに一致するように付値関数 sを固定することを示す.

$A\Vdash\exists x.\Phi[s]$　iff　$A\Vdash\Phi[s(a/x)]$ なる $a\in H_T(A)$ が存在する.

以上のように基本的には全く Kripke モデルと同じである. Kripke モデルと同じく, 任意の $f:B\to A$, 任意の Φ について, $A\Vdash\Phi[s]$ ならば $B\Vdash\Phi[s|f]$ となる. **妥当な式** Φは, 任意のAとそれに付随して定



める任意の s に対して $A \Vdash \Phi[s]$ が成立つことであるとする.

ここで与えたモデルが集合論のモデルとして意義を持つためには，少なくとも2つの基本的な性質を確かめなければならない.第1は内包公理(comprehension axion)，第2は外延公理 (extentionality axion)，が成立することである．どちらも集合論の基礎になっている.

内包公理というのは，大まかに言えば，ある論理式 Φ を真とする実体の“あつまり”もまた実体として認められる，つまり体系のなかに存在しうるということである．ここで与えた高階論理体系の記号で厳密に言えば次のようになる．ここで $\Phi \Leftrightarrow \Psi$ は，$(\Phi \Rightarrow \Psi) \wedge (\Phi \Leftarrow \Psi)$ と同じであるとする.

論理式 Φ にあらわれる自由変数*は $u_0, \cdots, u_{n-1}$, y のいずれかであるとし，x は Φ の中に自由変数としてはあらわれない新しい変数とする．x の型を $\mathcal{P}T$, y の型を T とする．すると，任意の Φ について,

$$\forall u_0, \cdots, u_{n-1}\ \exists x \forall y[y \in x \Leftrightarrow \Phi]$$

は妥当である.

これを示すには，C の任意の A，任意の $b_0, \cdots, b_{n-1}$ (型は u_i に合わせる) について,

$$A \Vdash \forall y[y \in x \Leftrightarrow \Phi][s]$$

になるような $c \in H_{\mathcal{P}T}(A)$，ただし，$s(x)=c$，が存在することを言えば良い．ただし，$s(u_i)=b_i$ とする．そのような c は，各 $f: B \to A$ に対して

$$cf = \{t \in H_T(B) \mid B \Vdash \Phi[s \mid f(t/y)]\}$$

とし，これら cf のうちのひとつに対応するものを c とすれば良い．証明は省略する.

外延公理とは，大まかに言えば，同じ要素からなるふたつの“あつまり”は同じである，ということである．これをこの高階論理体系で記述すれば,

$$\forall x, y[\forall z[z \in x \Leftrightarrow z \in y] \Rightarrow x = y]$$

ということになる．これも妥当であることが証明できる.

また，関数についても同様に内包，外延公理が考えられるが，これらが妥当であることも言える.

この他にも関数に関する興味深い式が得られる．たとえば，$h = g \circ f$ のとき

$$\forall x, y, z[[y = fx \wedge z = gy] \Rightarrow z = hx]$$

は妥当である．また，C において $f = g$ のとき，かつそのときに限り,

$$\forall x, y[y = fx \Leftrightarrow y = gx]$$

が妥当となり，これは e を高階論理に“埋め込む”ことができることを示すものに他ならない．特に C が積閉包圏ならば，(有型) λ 計算体系はすっかり高階論理体系に埋め込まれることになる．そこで，λ 計算の原

* λ 式での自由変数と同じく，$\forall$ や $\exists$ に束縛されてない変数

理はすべて高階論理の論理式として記述できる．無型λ計算についても，Scott 流にレトラクトを考えることによってやはり同様に高階論理体系に埋込める可能性がある．

このように，論理（高階直観論理）と関数（λ計算）との一種の調和を形成することができるのである．

4. おわりに

ここまで述べたことをまとめると，

（1） 直観論理とλ計算とは密接な関係がある．λ計算は，積閉包圏を介して，直観命題論理の部分体系と同じである．

（2） 高階直観論理体系を作ると，トポスを介して，その中にλ計算体系を埋込むことができる．

これだけでは直接にはコンピュータ科学への応用には結びつかない．しかし，その土台のひとつが築かれたとみることができる．

まず最も直接的な効果を持つのは，関数型プログラムの正当性証明にここで述べた高階論理を利用することである．

さらに予想としては，コンピュータ科学のさまざまな分野の相互関連を論じる基礎にもなり得るだろう．考えられる相互関連を図に示す．

さらに想像の翼をひろげるならば，トポスを基礎にして，自然言語からあいまいさをとり除いた高レベルの表現力を持ち得るような，万能純粋言語（universal pure language）の構築に進み得るかも知れない．もちろん一挙にそこに到達できるとは思われないが，少なくともこのような地道な方法で高位言語を追求することは，コンピュータ科学を「科学」とするために重要であると考えられる．

筆者らの力不足によってこの分野の研究動向を十分伝えることができなかったが，本稿がわが国におけるこの分野の研究を活発化するためのひとつの刺激になり得るならば，それは筆者らの望外の喜びである．

謝辞

初稿については，電総研の白井良明，古川康一，梅山伸二の諸氏から貴重なコメントをいただいた．ここに感謝の意を表したい．

参考文献

1) Barendregt, H.: The Lambda Calculus its Syntax and Semantics, North Holland (1980).
2) Church, A.: The Calculi of lambda conversion, Princeton University Press (1941).
3) Curry, H.B. and Feys, R.: Combinatory logic, Vol. 1., North Holland (1958).
4) Curry, H.B., Hindley, J.R. and Seldin, J.P.: Combinatory logic, Vol. 2, North Holland (1972).
5) Gierz, G., Hofmann, K.H. Keimel, K., Lawson, J.D., Mislove, M., and Scott, D.S.: A Compendium of continuous Lattices, Springer-Verlag, (1980).
6) Goldblatt, R.: Topoi: The Categorial Analysis of Logic. North-Holland (1979).
7) 後藤滋樹：プログラム・シンセシス，情報処理，Vol. 22, No. 3, pp. 218-225 (1981).
8) 伊藤貴康：プログラミング言語の意味論—入門的解説，情報処理，Vol. 21, No. 6, pp. 660-670 (1980).
9) 伊藤貴康：プログラムの理論とその応用(1)，情報処理，Vol. 22, No. 7.
10) 伊藤貴康：プログラムの理論とその応用(2)，情報処理，Vol. 22, No. 9, pp. 884-895 (1981).
11) Joyal, A. and Reyes, G.E.: Forcing and generic models in categorical logic, J. Pure and applied algebra (1980).
12) Lambek, J.: From λ-Calculus to Cartesian closed categories, Essays on combinatory logic, Lambda Calculus and Formalism, North-Holland pp. 375-402.
13) Maclane, S.: Categories to the working Mathematicians, Springer (1971).
14) 松本和夫：数理論理学，共立出版 (1970).
15) Schönfinkel, M.: Über die Bausteine der mathematischen Logik, Math. Ann. 92, pp. 305-316 Translated in "From Frege to Gödel" (1924).
16) Scott, D.: Continuous lattices, Lecture Notes in Mathematics Vol. 274, pp. 97-136, Springer Verlag (1972).
17) Scott, D.: λ-calculus and recursion theory, Prc. 3rd Scandinavian Logic Symposium pp. 154-193, North-Holland (1972).
18) Scott, D.: Lambda calculus Some models, some philosophy, in Kleene Symposium, North-Holland, pp. 381-421, (1980).
19) Scott, D.: Related theories of the λ-Calculus, Essays on Combinatory logic, Lambda Calculus and Formalism, North-Holland, pp. 402-448 (1980).
20) Stoy, J.E.: Denotational Semantics The Scott-Strachey Approach to Programming Language Theory, The MIT. Press (1977).
21) 高橋正子：スコット理論，情報処理，Vol. 20, No. 11, pp. 983-990 (1979).
22) 竹内外史：層．圏．トポス，日本評論社 (1977).
23) Strachey, C.: Towards a Formal Semantics,

Formal Language Description Languages for Computer Programming, Proceedings of IFIP Working Conference on Formal Language Description Languages, North-Holland Publishing pp. 198-220 (1966).
24) Landin, P.: A Formal Description of Algol 60, Formal Language Description Languages for Computer Programming, Proceedings of IFIP Working Conference on Formal Language Description Languages, North Holland Publishing, pp. 266-294 (1966).

(昭和57年6月10日受付)