解　説

スコット理論†

高　橋　正　子††

1. 序

新しいプログラム言語を定義したりマニュアルを作ったりするとき，その言語の構文は主としてBNF記法を用いて表わし，意味（セマンティクス）は英語や日本語などで書くことが多い．この伝統的な意味の記述方法は分かり易く便利であるが，近年，数式や論理式を用いてより形式的にこれを表わそうとする動きや，その方向を目指す研究が多く見られるようになった．こうした動きの背景には，日常言語による意味記述にはとかくあいまいさが伴い不正確になりやすいこと，特に最近注目を集めているプログラムの自動検証等のために完全にあいまいさのない形式的表現が要求されることがあげられるが，更に数学サイドの見方をするならプログラムの意味の世界に数学的に非常に興味ある新しい構造が見え始めたことが，この方面の研究を推進する大きな力となったと言えるのではないかと思う．

以下，本章ではプログラムの意味を考えるとはどういうことか，従来の方法と数式や論理式を用いる方法の考え方の違いなどについて簡単に述べ，次章以下のイントロダクションとする．2章ではスコットの理論がプログラムの意味のどのような側面に注目し，どんなアプローチで何を示そうとするのかについてなるべく数学用語を用いずに解説し，3章および4章で理論の展開の概説を試みる．

1.1 プログラム言語の意味を最も詳細かつあいまいさなしに記述したものとして，その言語のコンパイラやインタプリタがある．実際，言語Lのコンパイラ Cを一つ作り，それが機械Mの上で稼動されたとき実行される事柄をもってその言語の「意味」を定義することにすれば，そこにあいまいさや不正確さは全くない．しかしその場合Lの定義が正確なのはよいが，同じLの名で呼ばれる言語の意味が，使用する機械や処理系によってまちまちで一般性を欠く．また人間がそれをきちんと理解しようとするといちいちコンパイラやインタプリタを解読しなければならないことになって大変不便である．

1.2 コンパイラや自然言語以外にプログラムの意味を正確に，しかも人間に分かり易く書く方法はないものだろうか．そう考えたとき候補として頭に浮かぶのが数学で用いられる表現形式，すなわち数式や論理式を用いる方法である．数式を用いるというのは概略次のような考え方に基づく．ふつうコンパイラ等をプログラムの意味記述とみなす場合の不都合な点は上に述べた(イ)製作上の事情に依存し一般性を欠く，ということの他に，(ロ)基本的な部分と細部が同等の詳しさで（すなわちすべてが非常に詳しく）記述されるため人間にとって理解し難い．(ハ)処理系記述用言語がまだ必ずしも十分発達していない現状では，扱う機種によってそれぞれ異なる言語で書かれる場合が多く，それらの各々に一人の人間が精通することは困難である，などの点が上げられるだろう．そこでこれらの障害を回避するため，なるべく個々の機械や言語に依存せず，しかもプログラムにとって本質的と思われる概念を（今，必要とされる詳しさに応じて）抽出

---

† Introduction to the Theory of Denotational Semantics by Masako TAKAHASHI (Department of Information Science, Tokyo Institute of Technology).

†† 東京工業大学理学部情報科学科

本文で用いた特殊な記号

(記号)	(用例)	(意味)
$\sqsubseteq$	$x \sqsubseteq y$	半順序
$\bigsqcup$	$\bigsqcup X$,	上限
$\sqcup$	$x \sqcup y$	
$\bigsqcap$	$\bigsqcap X$,	下限
$\sqcap$	$x \sqcap y$	
$\top$		最大元
$\bot$		最小元
$\in$	$x \in X$	属する
$\subseteq$	$X \subseteq D$	含まれる
$\to$	$D \to D'$	関数
$+$		プラス
$\times$		かける
$\Leftarrow$	$f(x) \Leftarrow$ if …	代入
$\Rightarrow$		ならば
$\forall$	$\forall x \in D$	任意の
$\exists$	$\exists e_n \subseteq x$	適当な
$\prec$	$e \prec x$	特種な大小関係

し，それらが個々のプログラム言語でどのように互に関係づけられているかを人間に分かり易くかつ正確に記述する方法を見い出すことができれば大変望ましい．その場合，抽出された概念とそれらの関係を表現するのに，集合，関数，順序，束などの数学用語を用いるのが自然で有効ではないか．おおよそこのような考え方に立ってプログラムの意味の世界を探ろうとしたのが次章以下で述べる数学的または表示的意味論 (mathematical または denotational semantics) とよばれるものである．

論理式を用いてプログラムの意味を記述する研究も自動検証を目指す実際的な立場から，或いはそれにともなう種々の理論的問題の解明を目指して活発に進められているが本稿の主題から離れるのでここでは触れない．なお，スコットらの表示的意味論と論理式を用いる公理的アプローチは互に排反の性格のものではなく，両者を併用して具体的な言語の意味を記述しプログラムの自動検証に役立てようとする試みも近年報告されているが，ここではその話題も割愛し，表示的意味論の基礎的問題に焦点を絞って話をすすめる．

2. スコット理論における基本的概念と考え方

2.1 プログラムの中で表現されるものの中には，さまざまな種類の数，配列，リスト構造，関数等が含まれているが，その中で例えばある桁数（例えば2進32桁）以下の自然数のように，そのもの自身が直接計算機のメモリに記憶されたりあるいは，1,256番目の素数のように値そのものは直接記憶されていなくても有限時間内にその正確な値を求めることができるものと，そうでないものとがある．円周率πを考えてみると，その上位n桁を（nが指定されたとき）求めることは有限時間内にできるがその全桁の正確な値を得ることはできない．しかし，n桁まで正しい値π_n($n=1, 2, \cdots$) が得られればこれらによってπを近似し，その極限としてπが「表現された」とみることはできよう．ここでπ_{n+1}はπ_nより精度の高い（より多くの情報を含む）近似値であるが，スコットはこの近似の良さという（半順序）関係に注目する．そしてこれをプログラムの意味の世界における一つの基本的概念とみなし，$\pi_n \sqsubseteq \pi_{n+1}$のように表わす．

次に非負整数（本稿ではこれを自然数とよぶ）の上の関数$f(x)=x!$について考えよう．この関数は再帰呼び出しの機能をもつプログラム言語で，例えば

$f(x) \Leftarrow$ **if** $x>0$ **then** $x \cdot f(x-1)$ **else** 1

のような手続きによって表現される関数であるが，このfの値を有限時間内に全部求めることは不可能である．いま，上記のプログラムの再帰呼び出しの部分を順次展開すると

$f(x) \Leftarrow$ **if** $x>0$ **then**
$x \cdot$ (**if** $x-1>0$ **then**
$(x-1) \cdot$ (**if** $x-2>0$ **then**
$(x-2) \cdot$ (⋯
⋯) **else** 1
) **else** 1
) **else** 1

の形の無限に長いプログラムになるが，ここで展開をn回で打ち切りそれまでに値が求まらないときは未定義とするプログラムを考え，それによって表現される関数をf_nとする．このときf_nは$x=0, 1, 2, \cdots, n$に対してのみ$x!$値を取る自然数上の部分関数である．これらの部分関数は有限時間で値が（定義されているものは）すべて求まり，かつ各々fを「近似」する部分関数である．この場合にもf_{n+1}はf_nより多くの情報を提供するので$f_n \sqsubseteq f_{n+1}$と書く．fは関数列$f_0 \sqsubseteq f_1 \sqsubseteq f_2 \sqsubseteq \cdots$の極限として表わされる．

スコットの表示的意味論では，プログラムによって表わされる数や関数のなす集合（意味空間，semantic space, domain, data type などと呼ばれる）の中のこのような「近似」の概念およびその「極限」（近似を先のように半順序関係で表わすときその順序における上限）を基本とし，その上に議論を組み立てる．そうすることによってプログラム言語に特徴的と思われる繰り返しや再帰呼び出しを用いて表現される意味の世界の構造が極めて自然に捉えられる．

2.2 スコットは更に意味空間に対する条件として（イ）この空間の上で考える関数を連続関数――ある意味で極限を保存する関数――に制限すること，（ロ）有限的に表現される元の極限としてすべての元が表わされる（つまり有限的に表現される元全体が意味空間内で十分稠密であり，丁度実数全体の中で有理数全体が（位相的に）果たすのと似た役割を果たす）こと，（ハ）極限に収束する方法に制限を加え，例えば互に全く比較不能な二つの上昇列が同一の極限を持つことを禁ずる（もしそのようなものを許すと極限を保存する関数――連続関数――の概念が著しく制限されてしまう），などを直観的または理論的見地からつけ加え，それらの条件の下で理論を展開する．（条件の正確な定義は

次章で述べるが束および位相空間の初歩の言葉を用いてなされる.) これらの前提条件の当否によってスコット理論の評価も当然異なる訳であるが筆者の考えではそれらは弱過ぎるきらいはあっても強過ぎることはない, つまり意味空間として過重な条件ではないと思う.

2.3 プログラム格納式の通常の計算機で, ある命令を自分自身 (のコード) に対して適用したり, また関数呼び出しの際, 実パラメタとしてその関数自身を用いること (これを関数の自己適用, self-application とよぶ) ことが少なくとも原理的に可能である. つまり我々はプログラムの意味の世界を考える際, 暗黙のうちに自己適用可能な空間を考えていることになる.

ところで関数の自己適用に関して次の事実が知られている: 集合Xが, XからXへの関数全体の集合と等濃度 (1対1 onto の対応がつく) なら, Xの濃度は1以下である. つまり X は2個以上の元を持ち得ない. この事実とプログラムの意味の世界の自己適用性とはどう関係するのか, 矛盾は生じないだろうか. スコットが意味の世界の数学的基礎を確立したいと考えた動機の一つはこの点を解明することにあったのではないかと考えられる. (このことは別の言い方をすれば, チャーチの λ-calculus の自明でない集合論的モデルが作られるかという問題になる.) そして実際彼はそのような条件を満たす空間で自明でない (元を2個以上, 実際無限個, 持つ) ものを作ることに成功した. 3章で D_∞ モデルと呼ばれる空間の構成を, 4章で P_ω モデルを紹介する.

2.4 ある計算機Mの上でプログラムによって表現される意味の世界をVとする. Vは自然数の集合 N, M の上で考えられるすべての手続き (の表わす関数) の集合 P, アドレスの集合A, などで構成されるであろう. いま, 簡単のために $V=N+P+A$ とおく. (+ は直和を表わす.) 計算機Mの状態の集合をSとすると, $S=(A\to V)$, すなわちSはAからVへの関数の集合に等しい. このときプログラムで定義される関数の性質を考えると次の同型関係が成り立つ.

$$P\cong((V\times S)\to(V\times S))$$

すなわちPの元は, 実パラメタ ($\in V$) と計算機Mの状態 ($\in S$) に依存して出力結果 ($\in V$) を出し同時にMの状態を更新する一つの関数を表わす. ($\times$ は直積を, $\cong$ は左辺と右辺が1対1 onto に対応づけられることを意味する.) **2.3** で述べた λ-calculus のモデルとは $D\cong(D\to D)$ なる関係を満たす自明でない集合 D のことで, それについて次節でやや詳しく述べるが, $\to$の他に$\times$や$+$を含んだ上のような式の場合にもこれを満足する自明でない集合 (P, V, Sなど)が存在することが同じような考え方で, または D_∞ や P_ω の性質を使って, 導くことができる. したがってこのような集合を用いてプログラムが表わす意味を上のように (ただし実際はもっと厳密に) 記述して矛盾がないことが保証される訳である.

2.5 次章で取り上げる D_∞ は, 任意の意味空間から出発し, これを順次拡張することにより得られる自己適用可能な空間である. 4章で紹介する P_ω はこれと異なり, 自然数の集合の上に見い出された具体的な一つの空間であるが, 任意の意味空間をその中に埋め込むことができるという面白い性質を持っている. 4章では P_ω のこうした性質のほか, ある簡単な言語のセマンティクスを P_ω の上に実際に定義することにより, 従来の計算の理論 (帰納的関数の理論) とスコット理論の間に直接の結びつきが見られることについて簡単に触れる.

3. D_∞ の構成

3.1 はじめに, 次の(1), (2)の条件を満足する集合Dについて考える.

(1) Dは半順序関係* $\sqsubseteq$ を持つ.

(2) Dの任意の部分集合Xに対し, Xの上限$\bigsqcup X$および下限$\bigsqcap X$が存在する.

このような集合Dを完備束とよぶ. 条件(2)よりDは必ず最大元$\bigsqcup D$と最小元$\bigsqcap D$を持つがこれらをそれぞれ $\top$, $\bot$ で表わす. (近似の精度 $\sqsubseteq$ に関して $\bot$ は何らの情報も含まない元, 「未定義」を意味し, $\top$ は如何なる情報もその上につけ加えることのできない元——例えば既に相矛盾する情報を含んだ要素, 「過定義」とよばれる——を意味する. $\bot$ はともかく, $\top$ は直観的に把握し難い存在であるがこのようなものを仮定する根拠は, もしDが単なる半順序集合で完備束でないとき, これを適当に拡張してDを完備束の中に埋め込むことができる[13]こと, そしてそのような枠組の中でより明解な議論が展開されるためである.)

3.2 Dと D' が完備束で, 関数 $f: D\to D'$ が

* $\sqsubseteq$ が D の半順序であるとは (i)$x\sqsubseteq x$, (ii)$x\sqsubseteq y$, $y\sqsubseteq x\Rightarrow x=y$, (iii)$x\sqsubseteq y$, $y\sqsubseteq z\Rightarrow x\sqsubseteq z$ がすべての $x, y, z(\in D)$ について成り立つことを言う. $(\forall x\in X)(x\sqsubseteq y)$ のときyを $X(\subseteq D)$ の上界とよび $X\sqsubseteq y$ と書く. X の上界の最小元が存在するときこれを X の上限 $\bigsqcup X$ とよぶ. 同様に X の下界を定義し下界の最大元が存在するときそれを下限$\bigsqcap X$ とよぶ. $X=\{x, y\}$ の とき, $\bigsqcup X$ を $x\sqcup y$, $\bigsqcap X$ を $x\sqcap y$ と書く.

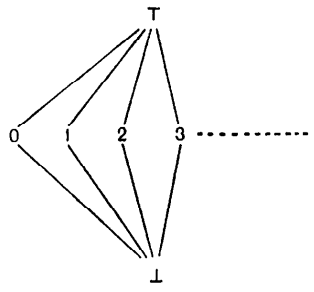

図-1

$f(\bigsqcup X)=\bigsqcup\{f(x)|x\in X\}$

を任意の有向集合*$X\subseteq D$に対し満たしているとき，fを連続関数とよぶ．$x\sqsubseteq y$のとき$X=\{x,y\}$は明らかに有向集合で，したがってfが連続ならば

$f(x)\sqsubseteq f(x)\sqcup f(y)=f(x\sqcup y)=f(y)$

が成立する．ゆえに連続関数は常に単調関数である．

DからD'への連続関数の全体を$[D\to D']$と書く．この集合に適当な半順序を導入することによってこれを再び完備束にすることができる．実際，関数fおよび$g(\in[D\to D'])$の間に

$f(x)\sqsubseteq g(x)\quad(\forall x\in D)$

の関係があるとき$f\sqsubseteq g$と書くことにすればこの関係が$[D\to D']$の半順序であること，また各$F\subseteq[D\to D']$に対し

$g(x)=\bigsqcup\{f(x)|f\in F\}\quad(\forall x\in D)$

なる関数gを考えると，gが$[D\to D']$におけるFの上限であることが容易に確かめられる．例として，DとD'がともに図-1で示される簡単な完備束である場合（すなわち，$\bot$と$\top$の他には自然数が互に比較不能な元として存在するのみの空間である）を考えると，DからDへの関数

$$f_n(x)=\begin{cases}\top & x=\top\ \text{のとき}\\ x! & x=0,1,2,\cdots,n\ \text{のとき}\\ \bot & \text{上記以外のとき}\end{cases}$$

$(n=0,1,\cdots)$はすべて連続で，しかも$f_0\sqsubseteq f_1\sqsubseteq f_2\sqsubseteq\cdots$の関係にある．また，これらの関数の上限$\bigsqcup\{f_0,f_1,\cdots\}$として関数

$$f(x)=\begin{cases}\top & x=\top\ \text{のとき}\\ x! & x=0,1,2,\cdots\ \text{のとき}\\ \bot & x=\bot\ \text{のとき}\end{cases}$$

が得られる．$[D\to D]$は完備束だからfも当然$[D\to D]$の元である．次いで上のDをatomの集合とするリスト構造の全体に半順序

$\langle x_1,x_2,\cdots,x_n\rangle\sqsubseteq\langle y_1,y_2,\cdots,y_m\rangle$
$\Leftrightarrow x_i\sqsubseteq y_i\quad(i=1,2,\cdots,n),\ n\leq m$

を入れた半順序集合を考える．この集合自身は完備束でない（部分集合が必ずしも上限を持たない）．これを完備化して得られる空間をD'とすると，D'は例えば次のような無限上昇列

$\langle x_1\rangle\sqsubseteq\langle x_1,x_2\rangle\sqsubseteq\langle x_1,x_2,x_3\rangle\sqsubseteq\cdots$

の上限（無限リスト$\langle x_1,x_2,x_3,\cdots\rangle$に相当する元）を含む空間となる．この$D'$の上でLISPの*car*, *cdr*, *cons*, *append*等に相当する関数を連続関数として定義することができる．任意の意味空間の上で定数関数，恒等関数，$x\sqcup y$，**if** $x\sqsubseteq a$ **then** b_1 **else** b_2（ただし$b_1\sqsubseteq b_2$）やそれらを合成して得られる関数などが連続であることも容易に確かめられる．

3.3 以上の定義の下で先述の方程式$D\cong[D\to D]$の自明でない解（反射的空間，reflexive domainとよばれる）をどのようにして作ることができるか，以下途中の証明は省きその構成の大筋を示す．

はじめに，任意に与えられた完備束D_0に対して$D_1=[D_0\to D_0]$とおく．そして，D_0の各元xに対し定数関数xを対応させるD_0からD_1への関数をϕ_0とおく．LISP等で使われるλ-記法を用いれば

$\phi_0(x)=\lambda y.\,x\quad(\forall x\in D_0)$

である．また，D_1の元fに$f(\bot)$を対応させるD_1からD_0への関数をψ_0とおく．このとき定義より，任意の$x\in D_0$に対して

$\psi_0(\phi_0(x))=\psi_0(\lambda y.\,x)=(\lambda y.\,x)(\bot)=x$

したがって合成関数$\psi_0\circ\phi_0$がD_0上の恒等関数に等しいことが分かる．また任意の$f\in D_1$に対して

$\phi_0(\psi_0(f))=\phi_0(f(\bot))=\lambda y.\,(f(\bot))\sqsubseteq f$

（なぜなら，fは連続，したがって単調だから各$x\in D_0$について$(\lambda y.\,(f(\bot)))(x)=f(\bot)\sqsubseteq f(x)$，すなわち$\lambda y.\,(f(\bot))\sqsubseteq f$）となる．$\phi_0,\psi_0$はともに連続関数で，特に$\psi_0$は$D_0$の束構造をすべて保存したままそれを$D_1$の中へ送る，すなわち$\psi_0$は$D_0$から$\psi_0(D_0)$への束同型である．これらをまとめて

$D_0\overset{\phi_0,\psi_0}{\simeq}D_1$

と書く．

一般に完備束DとD'の間に連続関数$\phi:D\to D'$，$\psi:D'\to D$が定義され，かつ$\psi(\phi(x))=x\,(\forall x\in D)$，$\phi(\psi(x'))\sqsubseteq x'\,(\forall x'\in D')$が成り立つとき

$D\overset{\phi,\psi}{\preceq}D'$

と書く．このとき，直観的に言うとD'はDより内容

* Xが有向集合であるとは，Xの有限部分集合が常に少なくとも一つの上界をXの中にもつことを意味する．

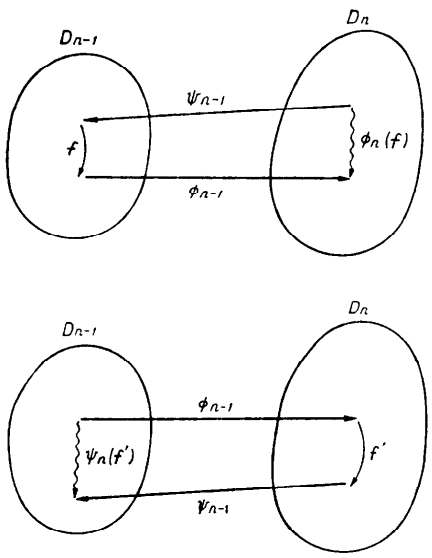

図-2

の豊富な空間で，ϕ はDの D' への埋め込みを，また ψ は D' のDへの射影 (projection) を表わす．(各 $x' \in D'$ に対して $\psi(x')=\bigsqcup\{x\in D \mid \phi(x)\sqsubseteq x'\}$，すなわち $\psi(x')$ は（x と $\phi(x)$ を同一視したとき）Dの元による x' の最良近似を表わす．)

今，D_0 より始めて $D_1, D_2, \cdots, D_n$ がすでに定義され，

$$D_0 \overset{\phi_0,\psi_0}{\lesssim} D_1 \overset{\phi_1,\psi_1}{\lesssim} D_2 \lesssim \cdots \lesssim D_{n-1} \overset{\phi_{n-1},\psi_{n-1}}{\lesssim} D_n$$

が成り立っているとする．このとき $D_{n+1}=[D_n\to D_n]$ とおき，連続関数 $\phi_n : D_n\to D_{n+1}$，$\psi_n : D_{n+1}\to D_n$ を

$$\phi_n(f)=\phi_{n-1}\circ f\circ\psi_{n-1} \quad (\forall f\in D_n=[D_{n-1}\to D_{n-1}])$$

$$\psi_n(f')=\psi_{n-1}\circ f'\circ\phi_{n-1} \quad (\forall f'\in D_{n+1}=[D_n\to D_n])$$

により定義する．直観的に言うと，$\phi_n(f)$ は $f(\in D_n)$ をより豊かな構造をもつ空間 D_{n+1} に持ち込み，これを見かけ上 D_n から D_n への関数に昇格せしめたもの，また $\psi(f')$ は $f'(\in D')$ の機能を ϕ_{n-1}, ψ_{n-1} を介して D_n 上に射影したものである (**図-2** 参照)．

このとき $\psi_n(\phi_n(f))=f$ $(\forall f\in D_n)$ および $\phi_n(\psi_n(f'))\sqsubseteq f'$ $(\forall f'\in D'_{n+1})$，したがって

$$D_n \overset{\phi_n,\psi_n}{\lesssim} D_{n+1}$$

が再び成り立つことが示される．

このようにして無限列

$$D_0 \overset{\phi_0,\psi_0}{\lesssim} D_1 \overset{\phi_1,\psi_1}{\lesssim} D_2 \lesssim \cdots$$

が得られるがそれらの「極限」D_∞ を次のように定義する．

$$D_\infty=\{\langle x_0, x_1, x_2, \cdots\rangle \mid x_n\in D_n,\ x_n=\psi_n(x_{n+1})\ (n=0,1,\cdots)\}$$

D_∞ の元 $x=\langle x_0, x_1, \cdots\rangle$ と $y=\langle y_0, y_1, \cdots\rangle$ の間に成分ごとに $x_n\sqsubseteq y_n$ $(n=0,1,\cdots)$ の関係があるとき $x\sqsubseteq y$ と書く．このように定義された半順序 $\sqsubseteq$ に関して D_∞ は完備束である．

さて，D_∞ は $D_0\times D_1\times D_2\times\cdots$ の部分集合であるが，D_n は $D_{n+1}, D_{n+2}, \cdots$ の中に埋め込まれるので D_n から D_∞ への自然な埋め込み $\phi_{n\infty} : D_n\to D_\infty$ が定義できる．(注：D_∞ の各元は第 n 成分 x_n を与えるとそれより左側の成分 $x_{n-1}, x_{n-2}, \cdots$ は定義より一意に定まる．) 各 $x\in D_\infty$ の第 n 成分を $\phi_{\infty n}(x)$ とすると $\phi_{\infty n} : D_\infty\to D_n$ は連続関数で，上の $\phi_{n\infty}$ との間に

$$D_n \overset{\phi_{n\infty},\phi_{\infty n}}{\lesssim} D_\infty \quad (n=0,1,2,\cdots)$$

の関係が成り立つ．こうして各 D_n を D_∞ の中へ埋め込むことができた．ここで，D_n の $\phi_{n\infty}$ による像 $\phi_{n\infty}(D_n)$ を $D_n^{(\infty)}$ とおくと

$$D_0^{(\infty)}\subseteq D_1^{(\infty)}\subseteq D_2^{(\infty)}\subseteq\cdots \quad (\subseteq D_\infty)$$

が成り立ち，更に D_∞ の各元 $x=\langle x_0, x_1, \cdots\rangle$ が

$$\phi_{0\infty}(x_0)\sqsubseteq\phi_{1\infty}(x_1)\sqsubseteq\phi_{2\infty}(x_2)\sqsubseteq\cdots$$

なる上昇列の上限として表わされることが分かる．このことは D_∞ が $D_0^{(\infty)}, D_1^{(\infty)}, D_2^{(\infty)}, \cdots$ をすべて含む最小の完備束であることを示している．

このようにして組み立てられた空間 D_∞ において件の同型式 $D_\infty\cong[D_\infty\to D_\infty]$ が成り立つことが概略次のようにして示される．D_∞ は D_0 から始めてその上の連続関数空間 D_1，連続関数空間上の連続関数空間 $D_2, \cdots$ と次々に前の空間に関数空間を積み上げていく形で拡張を続け，最後にそれらの極限（正確には射影極限，projective limit または inverse limit）を取ることによって得られた空間であった．$[D_\infty\to D_\infty]$ はその上の連続関数全体のなす空間である．そこで，こうした両者の構成に着目して，D_∞ から $[D_\infty\to D_\infty]$ への関数 Φ を次のように定義する．

$$\Phi(x)=\bigsqcup\{\phi_{n\infty}\circ x_{n+1}\circ\psi_{\infty n} \mid n=0,1,2,\cdots\} \quad (\forall x=\langle x_0, x_1, x_2, \cdots\rangle\in D_\infty)$$

右辺に現われる各項 $\phi_{n\infty}\circ x_{n+1}\circ\psi_{\infty n}$ は，任意の $y=\langle y_0, y_1, \cdots\rangle\in D_\infty$ に対して，まず $\psi_{\infty n}$ によりその第 n 成分 $y_n(\in D_n)$ を取りこれに $x_{n+1}(\in D_{n+1}=[D_n\to D_n])$ を施し，その結果を $\phi_{n\infty}$ で D_∞ に送り返すという操作を表わし，$\Phi(x)$ はそれらの上限である．次に

$\Psi : [D_\infty \to D_\infty] \to D_\infty$ を

$$\Psi(f) = \langle x_0, x_1, x_2, \cdots \rangle \quad (\forall f \in [D_\infty \to D_\infty])$$

ただし

$$x_{n+1} = \phi_{\infty n} \circ f \circ \phi_{n\infty} \quad (n = 0, 1, \cdots)$$

$$x_0 = \phi_0(x_1)$$

とおく．ここで各 x_{n+1} は，D_n の元を $\phi_{n\infty}$ で D_∞ の中へ持ち込み，そこで f を適用して再び D_n に引き戻す操作を表わし，それ自身 D_n から D_n への連続関数（したがって D_{n+1} の元）である．これらを成分として構成される $\Psi(f)$ が D_∞ に属し，かつ関数 Ψ 自身が連続であることも定義に照らして容易に調べられる．

以上の定義の下で，若干の計算により

$$D_\infty \overset{\Phi,\Psi}{\lesssim} [D_\infty \to D_\infty] \quad \text{かつ} \quad [D_\infty \to D_\infty] \overset{\Psi,\Phi}{\lesssim} D_\infty$$

がともに成り立つこと，したがって $\Psi(\Phi(x)) = x\ (\forall x \in D_\infty)$ および $\Phi(\Psi(f)) = f\ (\forall f \in [D_\infty \to D_\infty])$ が成立することが導かれる．このことから Φ および Ψ がともに束同型，すなわち $D_\infty \cong [D_\infty \to D_\infty]$ が成り立つことが示される．

3.4 次に D を前記の(1)，(2)の条件の他に次の(3)も満たす集合とする．

(3) 可附番集合 $E(\subseteq D)$ が存在して，D の各元 x を

$$x = \bigsqcup \{e \in E \mid e \prec x\}$$

の形に書くことができる．

このような D を（可附番基をもつ）連続束 continuous lattice とよび，スコットはその上の連続関数とともにこれを意味空間の数学的な定義とした．E の直観的なイメージとして，意味空間の元の中で有限時間内に正確な値が求まるものの集合を考えるとよい．上で用いた記号 $e \prec x$ は，e が下から x へ近づく近づき方を規定したものであるが，この定義を述べるためにまず D の中に位相を導入し，D を位相空間にする必要がある．

D の部分集合 U が次の(i)，(ii)を満足するとき U を D の開集合とよぶ．

(i) $x \in U,\ x \sqsubseteq y \Rightarrow y \in U$

(ii) $\bigsqcup X \in U,\ X$ 有向集合 $\Rightarrow X \cap U \neq \phi$

このように開集合を定義したとき，x を含む少なくとも一つの開集合 U が $e \sqsubseteq U$ を満足する，ということを記号 $e \prec x$ で表わす*．

D_0 を任意の連続束とするとき，D_0 から出発して **3.3** の手続きにより $D_1, D_2, \cdots$ を得，D_∞ を構成すれば $D_1, D_2, \cdots$ および D_∞ がやはり連続束となり，結局 $D_\infty \cong [D_\infty \to D_\infty]$ でかつ(1)～(3)を満足する空間 D_∞ の存在が証明できる．（束に上の位相を導入したとき，普通の位相空間の意味での連続関数と，**3.2** で定義した連続関数の概念とが一致する．）

(3)は直観的に把握し難い条件であるが次章で述べる自然数の集合から成る束の上で成り立っていることからも分かるとおり不自然な条件でないことは言える．これは意味空間に対する直観的な要請からというよりむしろ自然数の集合（或いはより一般の集合）のベキ集合のなす束の性質から帰納的に導かれた条件でこれが理論全体を非常に円滑に運ぶ鍵となっているように思われる．連続束に関する議論の詳細については Scott[1),3)], Reynolds[4)] を参照されたい．

4. P_ω

4.1 自然数の集合 $\omega = \{0, 1, 2, \cdots\}$ のベキ集合を $P_\omega = \{x \mid x \subseteq \omega\}$ とおく．P_ω は集合の包含関係 $\subseteq$ に関して束をなし，任意の $X(\subseteq P_\omega)$ に対してその和集合 $\cup X = \cup \{x \mid x \in X\}$ がその上限，共通部分 $\cap X = \cap \{x \mid x \in X\}$ がその下限となる．すなわち P_ω は $\subseteq$ に関して完備束をなす．

ω の有限部分集合の全体を E とおくと，E は可附番集合 $\{e_0, e_1, e_2, \cdots\}$（実際，

$$e_n = \{k_0, k_1, \cdots, k_{m-1}\} \Leftrightarrow n = \sum_{i<m} 2^{k_i}$$

なる数え上げを考えればよい）であり，かつ P_ω の各元 x を E の元によって $x = \cup \{e_n \mid e_n \subseteq x\}$ と表わすことができる．

P_ω における開集合の概念は，各 $x \in P_\omega$ に対し

$$x \in U \Leftrightarrow \exists e_n \subseteq x,\ e_n \in U$$

を満足する集合 U として特徴づけられる．特に e_n $(\in E)$ を含む集合の全体 $U_n = \{x \mid e_n \subseteq x \subseteq \omega\}$ は P_ω の開集合である．これらのことを用いて

$$x = \cup \{e_n \mid e_n \prec x\} \quad (\forall x \in P_\omega)$$

が導かれ，したがって P_ω が **3.4** の条件(3)を満たすこと，すなわち P_ω が連続束であることが確かめられる．P_ω 上の関数 $f : P_\omega \to P_\omega$ が連続であるということは f が

$$f(x) = \cup \{f(e_n) \mid e_n \subseteq x\} \quad (\forall x \in P_\omega)$$

を満たすということと同値である．（よって f は P_ω の部分集合 E の上で値が決まれば一意的に定まる．）

---

* Stoy[7)] では $\prec$ の代わりに，位相を使わずに定義される関係 $\ll$ を用いている．$\ll$ は2項関係として $\prec$ より弱いが，(3)の中の $\prec$ を $\ll$ で置きかえても条件としては(3)と同値である．(Yuyama[15)])

4.2 P_ωは，唯二つの元から成る完備束$\{\bot, \top\}$に前章の意味で位相を導入したものの積空間$\{\bot, \top\}^\omega$と同型な位相空間である．このように極めて単純な構造を持ちながらP_ωは次のような興味深い性質を示す[2),6)]．

（i） 任意の連続束はP_ωのある部分空間と同型である．

（ii） Yを任意の位相空間，Xをその部分空間とする．そのとき，任意の連続関数$f: X \to P_\omega$を，YからP_ωへの連続関数$\bar{f}: Y \to P_\omega$に拡張することができる．

このことから，任意の連続束D, D'とその上の連続関数$f: D \to D'$が与えられたとき，DおよびD'をP_ωの中に埋め込み，fをP_ω上の連続関数$\bar{f}: P_\omega \to P_\omega$に拡張できることが分かる．つまり$P_\omega$とその上の連続関数を調べることによって連続束とその上の連続関数一般の性質が知られる．P_ωはそのような普遍的性格を持つ空間である．

4.3 前章で我々は連続束でかつ$D \cong [D \to D]$を満足するDが存在することを知った．P_ωは任意の連続束をその部分空間として埋め込むことができるのだから，当然上のような自己適用可能な空間もこの中に入る筈である．それが実際どのようにして行われるかを見るために，我々はまずP_ω上の連続関数を如何に「コーディング」してP_ωの元で表わすか，つまりP_ωの元に関数とパラメタの二面性を持たせる方法を考える必要がある．

はじめに，自然数の対を自然数に，

$$\tau(n, m) = \sum_{i=0}^{n+m} i + m \quad (\forall n, m \in \omega)$$

によって1対1に対応させる．**4.1**でP_ω上の連続関数fは$E = \{e_0, e_1, e_2, \cdots\}$の上で値が決まれば一意に定まることを見たが，このことを利用し，fのコードgraph(f)を次のように定める．

$$\mathrm{graph}(f) = \{\tau(n, m) \mid m \in f(e_n)\}$$

このとき，graphは$[P_\omega \to P_\omega]$からP_ωへの1対1対応で，$u = \mathrm{graph}(f)$が与えられたとき，それからfを次のようにデコードすることができる．

$$f(x) = \{m \mid \exists e_n \subseteq x,\ \tau(n, m) \in u\} \quad (\forall x \in P_\omega)$$

一般に，どのような$u(\in P_\omega)$に対しても上式で規定される連続関数fが存在するが，これを$\mathrm{fun}(u)$と書くことにすると，

$$\mathrm{fun}(\mathrm{graph}(f)) = f \quad (\forall f \in [P_\omega \to P_\omega])$$

$$\mathrm{graph}(\mathrm{fun}(u)) \supseteq u \quad (\forall u \in P_\omega)$$

が成り立つ．任意の$u \in P_\omega$が連続関数のコードになっている訳ではないので上の第二式の包含関係は必ずしも$=$で置きかえられない．$u = \mathrm{graph}(f)$ならば$=$が成り立つ．

4.4 P_ωの元に関数とパラメタの二面性を持たせる手段を得たので，次に，簡単な数の体系と自己適用の機能を含む言語を定義しそのセマンティクスをP_ωの上で記述することを考える．

言語LAMBDAのwff（式，well formed formula）は定数0，変数$x, y, \cdots$，および，もしα, β, γがwffなら$\alpha+1$，$\alpha-1$，$\alpha \supset \beta, \gamma$，$\alpha(\beta)$，$\lambda x.\alpha$（それ以外はwffでない）とする．これらの各wffに対して，P_ω上にその意味を次のように定める．（左がシンタクス，右がセマンティクス）

0	$: \{0\}$
$\alpha+1$	$: \{n+1 \mid n \in \alpha\}$
$\alpha-1$	$: \{n \mid n+1 \in \alpha\}$
$\alpha \supset \beta, \gamma$	$: \{n \in \beta \mid 0 \in \alpha\}$ $\cup \{n \in \gamma \mid \exists k,\ k+1 \in \alpha\}$
$\alpha(\beta)$	$: \mathrm{fun}(\alpha)(\beta)$
$\lambda x.\alpha$	$:$ graph(f)，ただしfは$f(e_n) = \alpha[e_n/x]$ $(\forall e_n \in E)$なるP_ω上の連続関数

ここで右側に現われるα, β, γはそれらのセマンティクスを表わし，$\alpha[e_n/x]$はαのセマンティクスで変数xの所にe_nを代入したものを表わす．一般に変数は意味空間P_ωの上を動く．

LAMBDAは自己適用可能な関数空間を研究するためチャーチによって考えられた論理体系λ-calculusに数の簡単な計算と条件式の拡張（これによって多値関数や部分関数が定義できる）をつけ加えて作られた言語である．LAMBDAのセマンティクスを上のように定めたとき，各wffはその自由変数に値$(\in P_\omega)$を与えるとそれに対してP_ωの元を一つ定める，つまりP_ω上の関数を表わす．このようにして表わされるP_ω上の関数（それをLAMBDA定義可能な関数とよぶ）はすべて連続であるが，その中には例えばP_ω上の連続関数のコードgraph(f)を与えると，fの最小不動点（$f(x) = x$を満たし，包含関係に関して最小であるx）を出すfixed point operator：なども含まれている．fixed point operatorは再帰呼び出しを使って書かれた手続きの意味を記述するのに用いられ，その意味でプログラムの意味記述上極めて重要な要素である．

P_ωの上でこのように表現されたセマンティクスと

帰納的関数の理論 (recursive function theory[14]) の関係を知る上で興味深い次の結果がある[2),6)]: P_ω 上の関数 $f(x_1, x_2, \cdots, x_n)$ が LAMBDA 定義可能なとき，そのときに限り，集合 $\{(m, n_1, n_2, \cdots, n_k) | m \in f(e_{n_1}, e_{n_2}, \cdots, e_{n_k})\}$ が帰納的可算 (recursively enumerable) となる．すなわち，適当なチューリング機械を作りこの集合を数え上げることができる．この意味で LAMBDA 定義可能な関数は従来の計算可能関数の概念を P_ω 上の関数に自然に拡張したものとなっている．(従来，計算可能な (部分) 関数とよばれるのは，上記の概念を ω から $\omega \cup \{\perp\}$ ($\perp$ は未定義を表わすとして) への関数に制限したものに相当する.)

5. あ と が き

スコット理論への入門としてなるべく数学的知識を仮定せずに概観を述べた積りであるが，理論の規模の大きさに比し筆者の未熟なため甚だ不十分な記述に終ったことを御容赦頂きたい．今後この理論がどのように発展していくか予測は困難であるが，筆者個人の考えでは最後に述べた従来の計算の理論との関係が更に深く追究され，この二者を包括する形でより大きな計算の理論が展開されれば非常に興味深いと思う．従来の理論 (例えば Rogers[14] 参照) がどちらかと言うと計算可能関数から可能でない関数，そしてその不可能な度合いという方向に興味の中心が広がって行く傾向を見せたのに対して，スコットの理論は計算可能な関数の世界の中に興味深い数学的構造を見るという点で従来の理論を補う性格を持つように思われるからである．また，最近多くの関心を集めている同時処理に対する理論的アプローチとしてもスコット理論の今後に期待される所は大きいのではないかと思う．

最後に，これからこの理論を勉強する方々のために文献について述べる．まず Scott[1),2)] は最も基本的な文献であるが必ずしも読み易くない．D_ω を中心に読み易く書かれたものとして Scott[3)], Reynolds[4)], Sanderson[5)] がある．P_ω については Scott[6)], Stoy[7)] が理解し易い．スコット理論の主旨や考え方を平易に解説したものとして Scott[8)~10)], Tennent[11)], Stoy[7)], また数学的内容を整理し簡潔に述べたものに Barendregt[12)] がある．

本稿を書くにあたりお世話になった湯山恭史氏他の方々に感謝の意を表します．

参 考 文 献

1) Scott, D.: Continuous lattice, *Lecture Notes in Mathematics*, Vol. 274, pp. 97-136 (1972).
2) Scott, D.: Data types as lattices, *SIAM J. Comput.*, Vol. 5, pp. 522-587 (1976).
3) Scott, D.: Models for various type-free calculi, *Logic, Methodology and Philosophy of Science* (P. Suppes et al. eds., N-H) pp. 157-187 (1973).
4) Reynolds, J. C.: *Notes on a lattice-theoretic approach to the theory of computation*, Systems & Information Sci. Dept. Syracuse Univ., Syracuse, N. Y. (1972).
5) Sanderson, J. G.: *The lambda calculus, lattice theory and reflexive domains*, Math. Institute Lecture Notes, Oxford, England (1973).
6) Scott, D.: Lambda calculus and recursion theory, *Proc. 3rd Scandinavian Logic Symp.* (S. Kanger, ed., N-H) pp. 154-193 (1975).
7) Stoy, J. E.: *Denotational Semantics—The Scott—Strachey Approach to Programming Language Theory*, MIT Press (1977).
8) Scott, D.: Outline of a mathematical theory of computation, *Proc. 4th Ann. Princeton Conf. on Information Sciences and Systems*, pp. 169-176 (1970).
9) Scott, D.: Lattice theory, data types, and semantics, *Formal Semantics of Prog. Lang.* (R. Rustin, ed., Prentice-Hall) pp. 65-106 (1972).
10) Scott, D.: Logic and programming languages, *CACM*, Vol. 20, pp. 634-641 (1977).
(中島玲二訳,「論理学とプログラミング」bit Vol. 10, pp. 1, 404-1, 414, 1977).
11) Tennent, R. D.: The denotational semantics of programming languages, *CACM*, Vol. 19, pp. 437-453 (1976).
12) Barendregt, H. P.: The type free lambda calculus, *Handbook of Math. Logic* (J. Barwise, ed., N-H) pp. 1, 091-1, 132 (1977).
13) 岩村 聯: 束論，共立全書 (1966).
14) Rogers, H. R.: *Theory of Recursive Functions and Effective Computability*, McGraw-Hill, (1967).
15) Yuyama, K,: *A note on continuous latticeo*, Research Reports in Inf. Sci., Tokyo Institute of Technology (to appear).

(昭和 54 年 8 月 20 日受付)