

Mathematical Theory of Computation に
おける D. Scott の 方法

京大 数研 小野 寛晰

§0. D. Scott による mathematical theory of computation
を[1]に基づいて紹介する。今までこの方面でおこなわれて
きた研究は、semantics の定義等が必ずしも明快であるとは
いえない。Scott は program 全体を束としてとらえ完備
束上の連続関数の性質をうまく利用して、“loop”を持つ
program (又は flow diagram) を自然に導入し、更に
program の semantics をエレガントに定義している。な
おここではふいふいふが、Strachey との共著論文[3]で、こ
こで展開される semantics の記述法と具体的な program に
対し、適用を試みている。program は基本的な構成要素か
らいかに構成されているかという operational な面と、
program の“内容”である、関数としての機能、いいか
えるならば mathematical な面の、二つの面を持つと考え
られる。mathematical theory of computation

は、この二つの面の関係を調べることである。そこでまず operational な面から見て, program を表わすような (formal な) expression を記述しなければならない。次にその expression が mathematical にはどのような関数をあらわしているかを定義する(§2.)。

いま expression の基本的な構成要素として関数 $f_0, f_1, f_2, \dots$, 及び述語 $b_0, b_1, b_2, \dots$ が与えられているとする。expression の基本的な operation としては次の二つが考えられる。

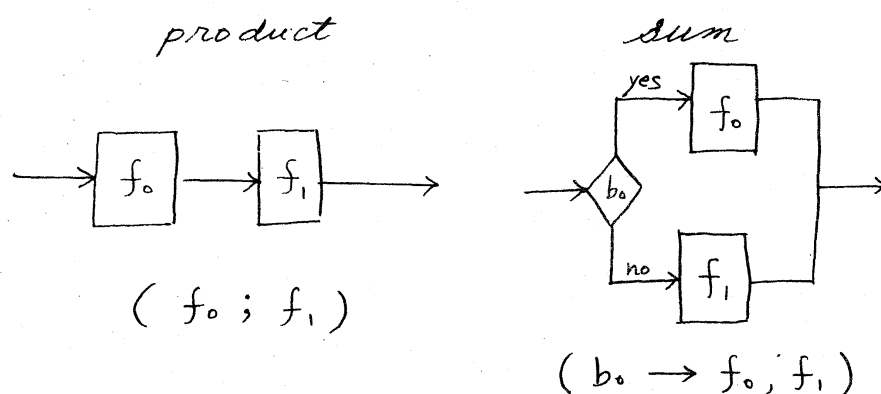


図 0.1

$f_0, f_1, \dots$ から product と sum により構成される expression を finite expression とよぶことにする。明らかに finite expression だけでは program 全体と表現することはできない。たとえば次のような, loop を持つ flow diagram を考える。

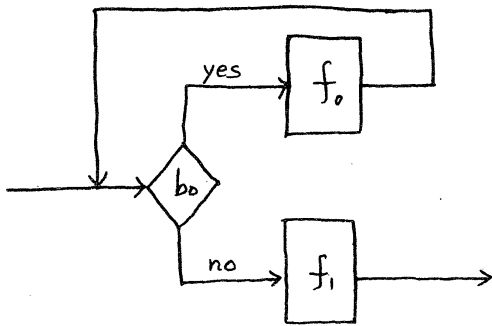


図 0.2

上の program 全体を g と書けば

$$g = (b_0 \rightarrow (f_0 ; g), f_1)$$

となる。このような g も expression とするために, "partial" な expression をめとめ, 上の g のような program を finite (partial) expression により "近似" することとを考える。

例えば, 上の g に対し, 次のような finite expressions $\{g_n\}$ を考える。(但し $\perp$ は内容的には totally undefined function を表わしてゐる。)

$$g_0 = \perp$$

$$g_{n+1} = (b_0 \rightarrow (f_0 ; g_n), f_1)$$

そうすれば g は $\{g_n\}_{n=0}^{\infty}$ の極限として表わされたい。と

ここでこの近似の考え方を次のように formulate してみる。

まず d, d' を $\perp$ の expression とした時に $d \leq d'$ とは、ある expression に対し d' が d より "よい" 近似になつて

ゐることとを示す。(例 $g_n \leq g_{n+1}$)。 $\leq$ は順序関係であり、

更にこの順序に対し product と sum が次の意味で単

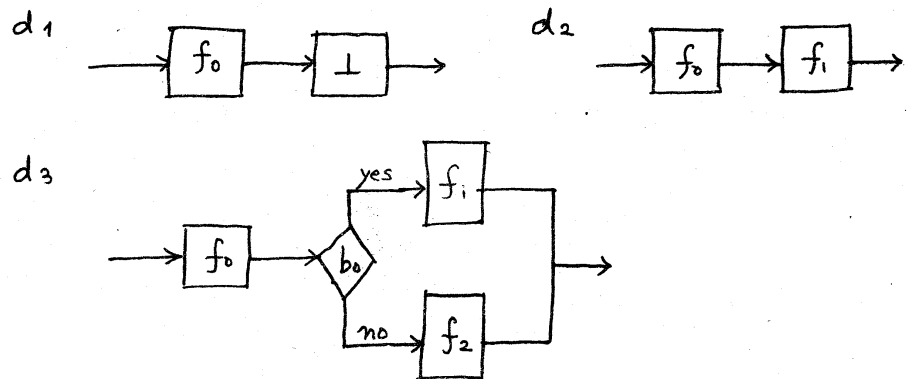
調になつてゐると考えるのは自然である。

$d_0 \leq d_1$ かつ $d'_0 \leq d'_1$ ならば

$$1) (d_0; d'_0) \leq (d_1; d'_1) \quad (0.3)$$

$$2) (b \rightarrow d_0, d'_0) \leq (b \rightarrow d_1, d'_1)$$

symbol $\perp, \top$ を導入し、それらを "underdetermined" program, "overdetermined" program を表わすものとする。従つて任意の expression d に対し $\perp \leq d \leq \top$ になりたつ。このように順序を入れると, finite expression 全体は最小元 $\perp$, 最大元 $\top$ を持つ束と考えることが出来る。(例えば $d \cup d' = \top$ は d と d' が incompatible な program であることと等しい。)



$$d_1 \leq d_2 \quad d_1 \leq d_3 \quad d_2 \cup d_3 = \top$$

そして, finite expression 全体の束を $\mathcal{L}$ で説明する方法で完備化すれば, loop を与えるような program を expression と $\mathcal{L}$ とらえることが可能になる。

§ 1. expression のなす束

expression のなす束を具体的に定義する前に [1] の key になっている定理 (定理 1.2) にふれておく。

束 D の部分集合 X の任意の有限部分集合が X の中に上界を持つ時、 X は 有向 (directed) であるという。

定義 1.1 関数 $f: D \rightarrow D'$ (D, D' はともに束) が 連続 であるとは、 D の任意の有向部分集合 X に対し

$$f(\cup X) = \cup \{f(x) \mid x \in X\}$$

がなりたつことをいう。

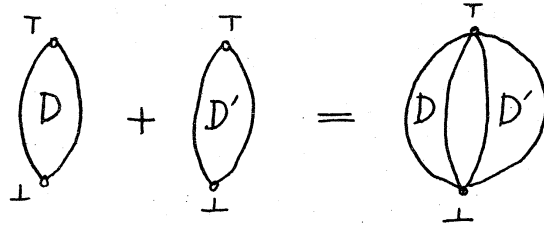
定理 1.2 D を完備束、 $f \in D \rightarrow D$ なる連続関数とする。
と f は不動点を持つ。更に不動点のうち最小のものが存在する。
(最小の不動点を $Y(f)$ と書き、 Y を fixed point operator とよぶ。)
minimal

証 $\perp$ を D の最小元とし、 $p = \bigcup_{n=0}^{\infty} f^n(\perp)$ とする。但し

f^0 は identity function, $f^{m+1} = f \cdot f^m$ とする。

すると p が f の最小の不動点になる。

二つの完備束 D, D' があたえられた時、union $D + D'$ 及び product $D \times D'$ を次のように定める。但し union を考える場合には D と D' とが disjoint であるとする。

定義 1.3

定義 1.4 $D \times D'$ は集合 $\{(d, d') \mid d \in D, d' \in D'\}$ に束演算を *component wise* に定義して得らる束である。

明らかに $D + D'$ 及び $D \times D'$ も完備になる。

まず *finite expression* のなす束を作る。基本的な関数 $f_0, f_1, \dots$ 及び述語 $b_0, b_1, \dots$ に対し次のような束 F, B を作る。

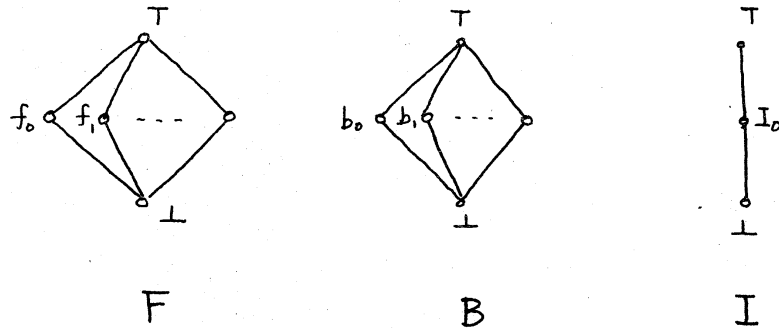


図 1.5

又, *identity function* I_0 に対し, 上のような束 I を作る。 $D_0 = F + I$ とする。 D_0 は基礎となる関数のみからなる束である。次に D をすてに得られた, *expression* のある集合のなす束とするとき $\{(d; d') \mid d \in D, d' \in D\}$ 及び

$\{(b \rightarrow d, d') \mid b \in B, d, d' \in D\}$ を又束とすると考えられる。と $\exists$ が (0.3) よりこれらの束は $D \times D' \cup B \times D \times D$ と同一視できることがわかる。したがって

定義 1.6 $D_0 = F + I$

$$D_{n+1} = D_0 + (D_n \times D_n) + (B \times D_n \times D_n)$$

(カンジを出すために $D_{n+1} = D_0 + (D_n ; D_n) + (B \rightarrow D_n, D_n)$ と書く。)

定理 1.7 $m \leq n$ ならば D_m は D_n の部分束である。

さて 集合 $\bigcup_{n=0}^{\infty} D_n$ を考えよう。定理 1.7 より $\bigcup_{n=0}^{\infty} D_n$ は束となることがわかるが、完備ではない。 $(\bigcup_{n=0}^{\infty} D_n$ は finite expression 全体の集合である。) そこで $\bigcup_{n=0}^{\infty} D_n$ を完備化することとを考える。

定義 1.8 $\psi_m : D_{m+1} \rightarrow D_m$ を m についての帰納法を用いて次のように定義する。

$$\psi_0(d) = \begin{cases} d & \text{if } d \in D_0 \\ \perp & \text{otherwise} \end{cases}$$

$$\psi_{n+1}(d) = \begin{cases} d & \text{if } d \in D_0 \\ (\psi_n(d'); \psi_n(d'')) & \text{if } d = (d'; d'') \\ (b \rightarrow \psi_n(d'), \psi_n(d'')) & \text{if } d = (b \rightarrow d', d'') \end{cases}$$

(但し $d', d'' \in D_n$)

$\mathcal{M}$ に関する帰納法を用いて次のことがわかる。

定理 1.9 $d \in D_{m+1}$ に対し

$$d \in D_m \iff \psi_m(d) = d$$

定理 1.10 各 m について

- 1) ψ_m は連続関数
- 2) ψ_m は D_{m+1} の各元に対し D_m の元で best approximation になるものをあたえる関数である。つまり $\psi_m(d) \leq d$,
かつ $d' \in D_m$ かつ $d' \leq d$ ならば $d' \leq \psi_m(d)$.

さて 次のような $\bigcup_{n=0}^{\infty} D_n$ の元の sequence $\langle d_n \rangle_{n=0}^{\infty}$ を考えてみる。

- 1) 各 n について $d_n \in D_n$
- 2) 各 n について $\psi_n(d_{n+1}) = d_n$

定理 1.10 2) と考え合わせるとこのような列 $\{d_n\}_{n=0}^{\infty}$ の 極限 $\star$ を $\bigcup_{n=0}^{\infty} D_n$ につけ加えてやればよいことになる。このことを formal に表現すると次のようになる。

定義 1.11 E を次の 1), 2) をみたすような $\bigcup_{n=0}^{\infty} D_n$ の sequence $\langle d_n \rangle_{n=0}^{\infty}$ の集合とする。

- 1) 各 n について $d_n \in D_n$
- 2) 各 n について $\psi_n(d_{n+1}) = d_n$.

更に E の上の順序 $\leq$ を次のように定める。

$$\langle d_n \rangle_{n=0}^{\infty} \subseteq \langle e_n \rangle_{n=0}^{\infty} \iff \text{各 } n \text{ について } d_n \subseteq e_n.$$

定理 1.12 E は完備束である。

この E を expression 全体のなす束 とよぶことにする。

まず $d \in D_m$ とし、 $\langle d_n \rangle_{n=0}^{\infty} \in E$ として $n \geq m$ なるすべての n に対し $d_n = d$ となるものを d^* と書く。すると $\{d^* \mid d \in D_m\}$ と D_m とは束として同型になることがわかる。従って d と d^* を以後同一視して考えることにする。つまり、各 m について D_m は E の部分束となる。更に任意の $d = \langle d_n \rangle_{n=0}^{\infty} \in E$ について $d = \bigcup_{n=0}^{\infty} d_n$ がなりたつ。 E の元 $\langle d_n \rangle_{n=0}^{\infty}, \langle d'_n \rangle_{n=0}^{\infty}$ に対し product と sum を次のように定める。

$$(\langle d_n \rangle_{n=0}^{\infty}; \langle d'_n \rangle_{n=0}^{\infty}) = \langle (d_n; d'_n) \rangle_{n=0}^{\infty}$$

$$(b \rightarrow \langle d_n \rangle_{n=0}^{\infty}, \langle d'_n \rangle_{n=0}^{\infty}) = \langle (b \rightarrow d_n, d'_n) \rangle_{n=0}^{\infty}$$

定理 1.13 $E = D_0 + (E; E) + (B \rightarrow E, E)$

E が図 0.2 のように loop を持つ program の expression を本当に含んでいることを次に示す。

図 0.2 の例は

$$g = (b_0 \rightarrow (f_0; g), f_1)$$

と書ける。

関数 $\pi: E \rightarrow E$ を次のように定義する。

$$\Phi(x) = (b_0 \rightarrow (f_0; x), f_1)$$

明らかに Φ は連続である。すると定理 1.2, 定理 1.12 より Φ は不動点 $g (\in E)$ を持つ。 $\Phi(g) = g$ 。したがって $g = (b_0 \rightarrow (f_0; g), f_1)$ となる g が E の中に存在することが証明された。これは次のように一般化できる。

finite expression の中にあらわれる関数を適当に variable $x_0, x_1, \dots$ で置きかえたものを polynomial とよぶことにする。

定理 1.14 $\pi_0(x_0, \dots, x_m), \dots, \pi_m(x_0, \dots, x_m)$ を, $x_0, x_1, \dots, x_m$ 以外の variable は含んでいないような $m+1$ 個の polynomial とする。すると

$$x_0 = \pi_0(x_0, \dots, x_m)$$

$$x_m = \pi_m(x_0, \dots, x_m)$$

の解 $(d_0, \dots, d_m)$ が E の中に存在する。

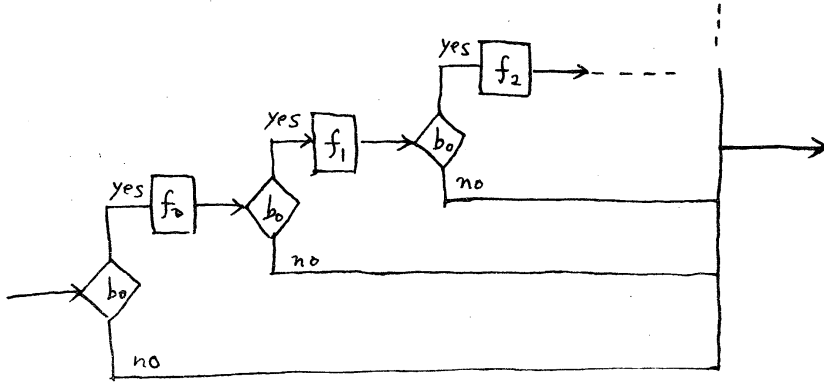
証. 関数 $\Phi: E^{m+1} \rightarrow E^{m+1}$ を次のように定義する。

$$\Phi(x_0, \dots, x_m) = (\pi_0(x_0, \dots, x_m), \dots, \pi_m(x_0, \dots, x_m))$$

Φ は連続であるから, 定理 1.2 より Φ の不動点 $(d_0, \dots, d_m)$ が存在する。(すなわち $(d_0, \dots, d_m) = Y(\lambda x_0 \dots \lambda x_m \Phi)$ 。)

上のような polynomial の解として表わされるような expression は program としても意味のあるものだが, E

はこのような解として表わされないような expression を非可算個含んでいる。次に一例を示す



⊗ 1. / 5

§2. program の semantics

定義 2.1 D, D' を完備束とする。 $[D \rightarrow D']$ を D から D' への連続関数の集合に、次の順序を付けたものとする。

$$f \leq g \iff f(d) \leq' g(d) \text{ for any } d \in D.$$

定理 2.2 $[D \rightarrow D']$ は完備束となる。

次に semantics の記述に必要な束 $\perp$ 及 v relation $\rhd$ を定義する。

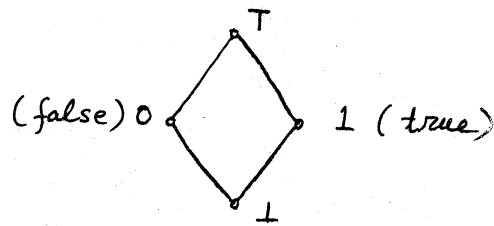


図 2.3 束 T (the lattice of truth values)

任意の束 D に対し " $\supset$ " は $T \times D \times D \rightarrow D$ なる関数で次のように定義する。但し $\supset (t, d, d')$ と書くかわりに $t \supset (d, d')$ と書く。

$$t \supset (d, d') = \begin{cases} d \vee d' & \text{if } t = T \\ d & \text{if } t = 1 \\ d' & \text{if } t = 0 \\ \perp & \text{if } t = \perp \end{cases}$$

E の semantics を考える際に次の仮定をおく。

"information" 全体は完備束をなし、更に "information" の変換は連続的におこなわれる。

具体的には次のように表現される。まず 1) ~ 3) をみたすような triple $(S, \mathcal{F}, B)$ を考える。

1) S は完備束

2) $\mathcal{F} \in [F \rightarrow [S \rightarrow S]]$ かつ $\mathcal{F}(\perp) = \perp$ かつ $\mathcal{F}(T) = T$

3) $B \in [B \rightarrow [S \rightarrow T]]$ かつ $B(\perp) = \perp$ かつ $B(T) = T$ 。

このような triple $(S, \mathcal{F}, B)$ に対し、次のような

$\nu \in [E \rightarrow [S \rightarrow S]]$ を考える。

任意の $d \in E$ 及び $\sigma \in S$ に対し

$$\nu(d)(\sigma) = \begin{cases} F(d)(\sigma) & \text{if } d \in F \\ \sigma & \text{if } d = I_0 \\ \nu(d_2)(\nu(d_1)(\sigma)) & \text{if } d = (d_1; d_2) \\ B(b)(\sigma) \supset (\nu(d_1)(\sigma), \nu(d_2)(\sigma)) & \text{if } d = (b \rightarrow d_1, d_2) \end{cases}$$

このような ν が存在することは次のようにして確かめられる。

上の式において右辺を Φ とあらわすと

$\Phi : [E \rightarrow [S \rightarrow S]] \rightarrow [E \rightarrow [S \rightarrow S]]$ は ν についての連続関数とみなせる。従って定理 1.2 より Φ の不動点として ν が定まるが、特に ν を Φ の最小の不動点にとる。このように (S, F, B) から定まる ν のことを E の 解釈 とよぶ。このようにして得られる解釈は通常の意味と一致している。(例えば d が "recursion schema" で定まるような expression であるならば $\nu(d)$ は recursion schema に対して定義されている d の解釈と一致する。[4], [5] 参照。)

定義 2.4 E の二つの元 d, d' が 同値 である ($d \cong d'$)

とは、任意の E の解釈 ν に対して $\nu(d) = \nu(d')$ となることをいう。

REFERENCES

- [1] D. Scott, *The lattice of flow diagrams*, (1970).
- [2] D. Scott, *Outline of a mathematical theory of computation*, *Proceedings of the Fourth Annual Princeton Conference on Information Sciences and Systems* (1970).
- [3] C. Strachey and D. Scott, *Mathematical semantics for two simple languages*, (1970).
- [4] D. Scott, *Some definitional suggestions for automata theory*, *J. Computer and System Sciences*, 1 (1967).
- [5] J. McCarthy, *A basis for a mathematical theory of computation*, in "Computer Programming and Formal Systems," North-Holland, 1963.