数理解析研究所講究録
第 322 巻 1978 年 75-98

再帰サブルティンにおける
パラメタの数について

電子技術総合研究所ソフトウェア部
二木厚吉

1. まえがき

再帰的に自分自身を呼び出すことが出来るようなサブルティンは，アルゴリズムの記述において基本的である。実際多くの問題に対して，再帰サブルティンを使った簡単で効率のよいアルゴリズムが示されている。（例えば[7]参照）このような再帰サブルティンにおいて，情報の授受の総量を規定するパラメタの数によりアルゴリズムの記述力にどのような差が生ずるかは，興味深い問題である。

本稿では，高々 n 個の入力パラメタと m 個の出力パラメタを持つ再帰サブルティンの使用が許されるプログラミング言語をモデル化したプログラム図式のクラス $P_R[m,n]$ を定義し，この問題をプログラム図式比較論（comparative schematology）的観点から考察する。

プログラミング言語のある特徴の有無による表現力の差異

を形式化して捉える手段として，プログラム図式のレベルでの能力比較が有力であることは，Paterson & Hewitt[1]により最初に指摘された。その後この方法に基づいた研究がいくつか行なわれている[2, 3, 4, 5]。

2. プログラム図式(Program Schema)によるモデル化

2.1 $P_R[m, n]$, $P_R^a[m, n]$ の定義

プログラム図式は各節点に適当な命令がラベル付けされたフローグラフの有限集合として定義される。フローグラフは図1の4種類の節点を矢印の方向が一致するようにつないで得られる有限有向グラフで，次の(i), (ii), (iii)の条件を満たすものである。(i)ただ1つの開始節点を持つ，(ii)すべての演算節点および判定節点は開始節点から停止節点に至るパス上にある，(iii)すべての停止節点へは開始節点からのパスがある。

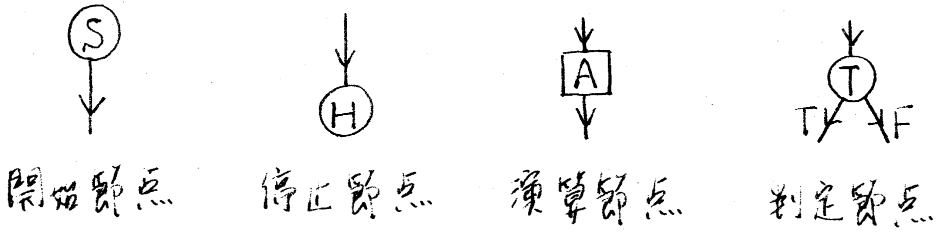


図1　フローグラフを構成する4種類の節点

本稿で考察する種々のプログラム図式を定義するためには

次のような命令が必要である。

開始節点にラベル付けされる命令

(S1) START$(x_1, x_2, \cdots, x_l)$

(S2) $F_i^{m,n}(x_{i1}, x_{i2}, \cdots, x_{im})$

演算節点にラベル付けされる命令

(A1) $x_j \leftarrow x_i$ (A2) $x_j \leftarrow f_i^m(x_{i1}, x_{i2}, \cdots, x_{im})$

(A3) $(x_{j1}, x_{j2}, \cdots, x_{jn}) \leftarrow F_i^{m,n}(x_{i1}, x_{i2}, \cdots, x_{im})$

判定節点にラベル付けされる命令

(T1) $p_i^m(x_{i1}, x_{i2}, \cdots, x_{im})$

停止節点にラベル付けされる命令

(H1) HALT(x_i) (H2) RETRURN$(x_{i1}, x_{i2}, \cdots, x_{in})$

ここで，x_i, x_jは変数記号，f_i^mはm-ary基本関数記号，p_i^mはm-ary基本判定記号，$F_i^{m,n}$はm入力n出力再帰サブルティン記号である。ただし，$i, j, l, m, n = 0, 1, 2, \cdots$とする。

以上の準備のもとに，高々m個の値を入力し，高々n個の値を出力する再帰サブルティンの使用が許されたプログラミング言語をモデル化したプログラム図式のクラスを定義することが出来る。

〔定義1〕上で導入された命令を各節点にラベル付けされたフローグラフの有限集合で，次の(a), (b), (c), (d)の条件を

満たすものを (m, n)再帰プログラム図式（RPS：Recursive Program Schema）といい，そのクラスを$\underline{P_R[m, n]}$で表わす。特に，各フローグラフが有向閉路を持たないとき，その(m, n)-RPSをアサイクリックと呼び，そのクラスを$\underline{P_R^a[m, n]}$で表わす。

(a) ただ1つのフローグラフだけが開始節点に(S1)タイプの命令をラベル付けされている。

このフローグラフを主プログラム図式，その他のフローグラフを副プログラム図式という。

(b) 主プログラム図式の停止節点には(H1)タイプの命令が，各副プログラム図式の停止節点には，その開始節点にラベル付けされた(S2)タイプの命令とnの値が一致した(H2)タイプの命令が，それぞれラベル付けされている。

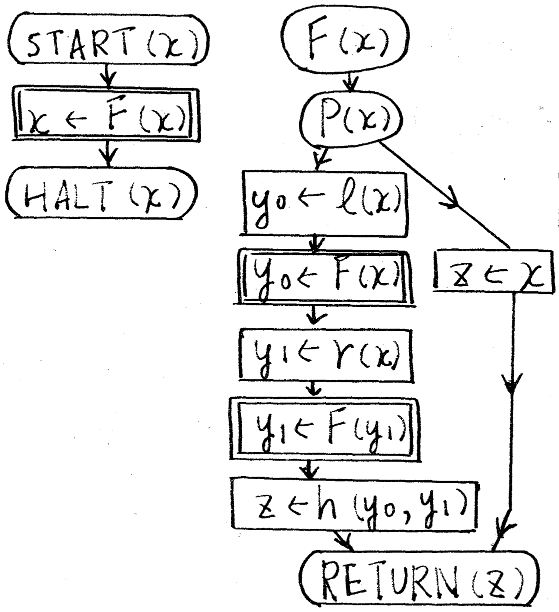
(c) (A3)タイプの命令がラベル付けされているとき，そしてそのときに限り，その命令とi，m，nの値が一致した(S2)タイプの命令を開始節点にラベル付けされた副プログラム図式がただ1つ存在する。

(d) 副プログラム図式がk個あり，それらの開始節点に，$F_i^{m_i, n_i}(x_{i1}, x_{i2}, \cdots, x_{im})$ $(1 \leqq i \leqq k)$のような命令がラベル付けされているとすると，

$$\forall i \in \{1, 2, \cdots, k\} : m_i \leq m \wedge n_i \leq n$$

である。 ∎

（例1）図2にプログラム図式の一例として，アサイクリック(1,1)-RPS S_0 とその線形記法 (linear notation) を示す。



［(A3)タイプの命令は節点を二重に囲って示している。］

```
┌START(x);
│ x←F(x);
└HALT(x)
```

```
┌F(x):
  if p(x)
  then begin
         y0←l(x);
         y0←F(x);
         y1←r(x);
         y1←F(y1);
         z←h(y0,y1)
       end
  else z←x;
└RETURN(z)
```

$S_0 \in P_R^a[1,1]$　　　　　　　　S_0 の線形記法

図2　プログラム図式　$S_0 \in P_R^a[1,1]$

線形記法はフローグラフをAlgol-likeな記法で示したものである。両者の対応は明らかであろう。以後は簡単のために

線形記法を用いてプログラム図式を表わす。また，x，y，z，…で変数記号を，f，g，h，…で基本関数記法を，p，q，r，…で基本判定記号を，F，G，H，…で再帰サブルーティン記号を表わす。

2.2 プログラム図式類の等価性[4]と[5]

プログラム図式は，データ領域が特徴付けられず，基本関数及び基本判定がそれぞれ f_i^m 及び P_i^m ($i, m=0,1,\cdots$) といった記号としてだけ示されたプログラムである。従って，プログラム図式はプログラムの集合を表わしていると考えられる。この集合に属するプログラムは，基本関数記号及び基本判定記号に解釈を与えることにより得られる。つまり，データ領域を与え，基本関数記号 f_i^m 及び基本判定記号 P_i^m にそれぞれデータ領域上の関数及び判定関数（true または false を値として返す関数）を割当てることにより，プログラム図式から1つのプログラムが得られる。

2つのプログラム図式は，START命令中で示される入力変数ベクトル，基本関数記号の集合，基本判定記号の集合の3つがともに等しいとき比較可能であるという。等価性は比較可能なプログラム図式についてのみ問題とされ，その定義は自由解釈（free interpretation）だけに基づいて行なわれ

る。自由解釈だけを考慮することが一般性を失なわないことについては〔6〕p260～262などを参照されたい。

自由解釈は，データ領域として入力変数と基本関数記号から生成される項（term）の集合*を持ち，各基本関数には，

$$f_i^m : t_1, t_2, \cdots t_m \rightarrow f_i^m(t_1, t_2, \cdots, t_m)$$

で定義される関数を割当てる。従って自由解釈は，各基本判定記号にどのような項の集合上の判定関数を割当てるかで特徴付けられる。以下では自由解釈のことを単に解釈という。

プログラム図式に解釈を与えれば実行可能なプログラムが得られる。この実行に際し，入力変数以外のすべての変数はある特別な初期値を与えられているとする。ただし，この初期値が，基本関数又はSTART，RETURN文の引数として使われることはないとし，この規則に反する実行は停止しないと約束する。また，各サブルティン・コールで呼び出される副プログラムの変数はすべてそのコールに固有で局所的なものであり，パラメタの授受はcall by valueで行なわれるとする。

プログラムの実行の途中のある時点で変数$y_1, y_2, \cdots, y_m$

---

*正確には次のように定義される。

i) 入力変数及び0-ary基本関数記号は項である。

ii) f_i^mがm-ary基本関数記号で$t_1, t_2, \cdots, t_m$が項ならば，

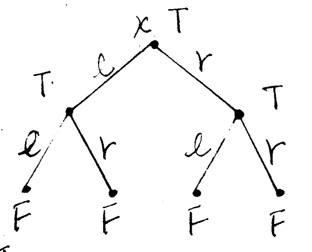
$f_i^m(t_1, t_2, \cdots, t_m)$も項である。

iii) i) ii)で項とされるものだけが項である。

の内容が項$t_1, t_2, \cdots, t_m$であったとする。このとき，$y \leftarrow f_i^m(y_1, y_2, \cdots, y_m)$のような演算命令に出会ったとすれば，この実行は項$f_i^m(t_1, t_2, \cdots, t_m)$を算出したという。また，$p_i^m(y_1, y_2, \cdots, y_m)$のような判定命令に出会い，かつその値がtrueのときは原子式$p_i^m(t_1, t_2, \cdots, t_m)$を，falseのときは否定原子式$\neg p_i^m(t_1, t_2, \cdots, t_m)$を，それぞれ判定したという。

〔定義2〕プログラム図式Sに解釈Iを与えて得られるプログラムの実行が出力を出すまでに算出または判定する項及び基本式（原子式と否定原子式を総称して基本式という）の系列の前後に入力と出力を付加した系列を<u>計算系列</u>といい，$\langle S, I \rangle$で表わす。また，その出力を$Val(S, I)$で表わす。プログラムが停止しなければ，$\langle S, I \rangle$及び$Val(S, I)$は定義されない。さらに，すべての解釈に対する$\langle S, I \rangle$の集合を$\Pi(S)$で表わし，Sの<u>計算集合</u>という。 ■

（例2）図2のプログラム図式S_0に対する次のような2進木で示される解釈I_0を考える。ここで，2進木の各節点はαx ($\alpha \in \{l, r\}^*$)のような項を表わし，それにラベル付けされたTまたはFは，それぞれ$I(p)(\alpha x) =$ ture または false を示している。他のすべての項に対して$I(p)$はfalseとなるとする。



$$
\begin{aligned}
\langle S_0, I_0\rangle = (x)\ \$\ p(x)\ \$\ \ell x\ \$\ p(\ell x)\ \$\ \ell\ell x\ \$\ \neg p(\ell\ell x) \\
\$\ r\ell x\ \$\ \neg p(r\ell x) \\
\$\ h(\ell\ell x, r\ell x) \\
\$\ rx\ \$\ p(rx)\ \$\ \ell rx\ \$\ \neg p(\ell rx) \\
\$\ rrx\ \$\ \neg p(rrx) \\
\$\ h(\ell rx, rrx) \\
\$\ h(h(\ell\ell x, r\ell x), h(\ell rx, rrx)) \\
\$\ (h(h(\ell\ell x, r\ell x), h(\ell rx, rrx)))
\end{aligned}
$$

となる。ここで，区切り記号として$を用い，入出力はそれぞれ（ ）でくくって示している。

〔定義3〕比較可能なプログラム図式SとS'は，

(i) 任意の解釈Iに対して$Val(S, I) \equiv Val(S', I)$のとき出力等価といい$S \overset{\mathrm{Val}}{\equiv} S'$と記し，

(ii) 任意の解釈Iに対して$\langle S, I\rangle \equiv \langle S', I\rangle$のとき動作等価といい$S \overset{\mathrm{op}}{\equiv} S'$と記す。

ここで，記号$\equiv$は，両辺が共に定義されないか，または共に定義されてその値が等しいかのいずれかである，という意味を持つ。∎

次の命題が成立することは容易に示される。証明は略す。

〔命題1〕(i) $\overset{\mathrm{Val}}{\equiv} \supsetneq \overset{\mathrm{op}}{\equiv}$

(ii) $S \overset{\mathrm{op}}{\equiv} S' \Leftrightarrow \Pi(S) = \Pi(S')$ ∎

（例3）図3の2つのプログラム図式 $S_1 \in P_R[0,*]$*, $S_2 \in P_R^a[1,1]$ を考える。

```
┌START(x);
  y ← x;
  while p(x)
  do x ← f(x);
  while p(y)
  do begin
     y ← f(y);
     x ← g(x)
     end;
└HALT(x)
```

$S_1 \in P$

```
┌START(x);
  x ← F(x);
└HALT(x)

┌F(x):
  if p(x)
  then begin
       x ← f(x);
       x ← F(x);
       x ← g(x)
       end;
└RETURN(x)
```

$S_2 \in P_R^a[1,1]$

図3　プログラム図式　S_1 と S_2

この2つのプログラム図式に対しては，$S_1 \overset{\mathrm{Val}}{\equiv} S_2$ であるが，$S_1 \overset{\mathrm{OP}}{\equiv} S_2$ は成立しない。

---

* 入力変数のないサブルーティンは必ず発散して停まらないとしているので，(P7参照) 任意の $n \geq 0$ に対し $P_R[0,n]$ は主プログラム図式だけから成る単純プログラム形のクラスを表わすとする。このクラスを $P_R[0,*]$ または単に P と記す。同様の理由で $\forall n \geq 0 \ P_R^a[0,n] \overset{\mathrm{def}}{\equiv} P_R^a[0,*]$ とする。

3. 能力比較

CとC'をプログラム図式のクラスとする。プログラム図式間の2つの等価性に基づいてCとC'の間には次のような2つの変換可能性が定義される。

〔定義4〕Cに属する任意のプログラム図式Sに対して$S \overset{\alpha}{\equiv} S'$なるプログラム図式S'がC'中に存在するとき，CはC'に等価性$\overset{\alpha}{\equiv}$に基づいて変換可能であるといい，$C \overset{\alpha}{\leq} C'$と記す。ここでαはValまたはopである。 ∎

以下では，2.1で定義した種々のプログラム図式のクラス，$P_R[m,n]$，$P_R^a[m,n]$，$m,n=0,1,2,\cdots$，に対し，それらが相互にどのように変換可能であるのかを考察していく。ただし，定理の証明は付録にまとめて示す。

〔記法〕(i) $C \overset{\alpha}{\equiv} C' \overset{def}{\iff} C \overset{\alpha}{\leq} C' \wedge C \overset{\alpha}{\geq} C'$

(ii) $C < C' \overset{def}{\iff} C \overset{\alpha}{\leq} C' \wedge \neg(C \geq C')$ ∎

フローグラフにループを許す影響に関しては次の定理が成立する。

〔定理1〕(i) $\bigcup_{m,n=0}^{\infty} P_R[m,n] \overset{op}{\equiv} \bigcup_{m,n=0}^{\infty} P_R^a[m,n]$

(ii) $\forall m \geq 0\ \forall n \geq 0 : \neg(P_R[0,*] \overset{Val}{\leq} P_R^a[m,n])$ ∎

入力パラメタの数を制限する影響に関しては次の定理が成

立する。

〔定理2〕 $\forall m \geq 1\ \forall n \geq 0 : \neg(P_R^a[m,0] \overset{\mathrm{val}}{\leq} P_R[m-1,n])$ ▮

出力パラメタの数を制限する影響に関しては次の定理が成立する。

〔定理3〕(i) $\forall m \geq 1\ \forall n \geq 1 : P_R^a[m,n] \overset{\mathrm{val}}{\equiv} P_R[m,1]$

(ii) $\forall m \geq 1\ \forall n \geq 1 : P_R[m,n] \overset{\mathrm{val}}{\equiv} P_R[m,1]$

(iii) $\forall m \geq 1 : \neg(P_R^a[1,1] \overset{\mathrm{val}}{\leq} P_R[m,0])$

(iv) $\forall n \geq 1\ \forall m \geq 0 : \neg(P_R^a[n,n] \overset{\mathrm{op}}{\leq} P_R[m,n-1])$ ▮

定理1-(i)は，フローグラフにループを含んだどんな再帰プログラム図式に対しても，それと動作等価なループフリーな再帰プログラム図式（RPS）が，（新たな再帰サブルティンを導入することにより）書けることを示している。しかし，定理1-(ii)から，再帰サブルティンの入出力パラメタの数に上界を設けてしまえば，再帰サブルティンを含まない単純プログラム図式のクラス$P(\overset{\mathrm{def}}{=} P_R[0,*])$の中にさえ，ループフリーなどんな再帰プログラム図式（RPS）とも出力等価にさえ（従って当然動作等価にも）なり得ないものが存在する。

定理2は，出力パラメタの数を制限すれば再帰プログラム図式の能力は出力等価の意味で制限されることを示している。

つまり，$P_R^a[m,0] \subseteq P_R^a[m,n] \subseteq P_R[m,n]$であり，$P_R[m-1,n] \subseteq P_R[m,n]$だから，定理2から次の系が導びかれる。

〔系1〕(i) $\forall m \ge 1\ \forall n \ge 0 \quad P_R^a[m-1,n] \overset{Val}{<} P_R^a[m,n]$

(ii) $\forall m \ge 1\ \forall n \ge 0 \quad P_R[m-1,n] \overset{Val}{<} P_R[m,n]$ ▮

m変数再帰図式(recursive schema)のクラスを$R[m]$で表わすことにする。(再帰図式の定義については[6]p319参照)ここでm変数とは，再帰図式のどの関数変数記号(本稿での再帰サブルーティン記号に対応する)も高々m個のパラメタしか持たないことを意味する。[6]の定理4-5, p324及び[2]の3-6, p73の構成法から直ちに次の命題が導びかれる*。

〔命題2〕 $\forall m \ge 1$: $R[m] \overset{op}{\equiv} P_R^a[m,1]$ ▮

この命題と定理2及び，$P_R^a[m-1,1] \subseteq P_R[m-1,1]$，$P_R^a[m,0] \subseteq P_R^a[m,1]$から次の系が導びかれる。

〔系2〕 $\forall m \ge 1$: $R[m] \overset{Val}{<} R[m+1]$ ▮

また，定理1-(i)，定理3-(i)から $P \overset{op}{\le} P_R^a[m,n] \overset{Val}{\le} P_R^a[m,1]$ だから，定理2から次の系も導びかれる。

〔系3〕 $P \overset{Val}{<} R \overset{def}{=} \bigcup_{m=0}^{\infty} R[m]$ ▮

系2は[4]で，系3は[2]でそれぞれ示された結果である。

---

* 再帰図式における計算規則(computation rule)は，leftmost-onnermost rule ([6], p321)とする。再帰図式に対しても定義2, 3と同様にして出力，動作の両等価性が定義される。

定理3-(i),(ii)は，出力パラメタの数を1つに制限しても出力等価の意味では能力を小さくすることにならないことを示している。しかし，定理3-(iii)は，出力パラメタの数が0と1では，能力に出力等価の意味で差が出ることを示し，定理3-(iv)は，入力パラメタの数を越えない範囲で，出力パラメタの数に応じて，動作等価の意味で能力に差が出ることを意味している。つまり次の系が導びかれる。詳細は略す。

〔系4〕(i) $\forall m \geq 1 : P_R^a(m,0) \overset{val}{<} P_R^a(m,1)$

(ii) $\forall m \geq 1 : P_R(m,0) \overset{val}{<} P_R(m,1)$ ■

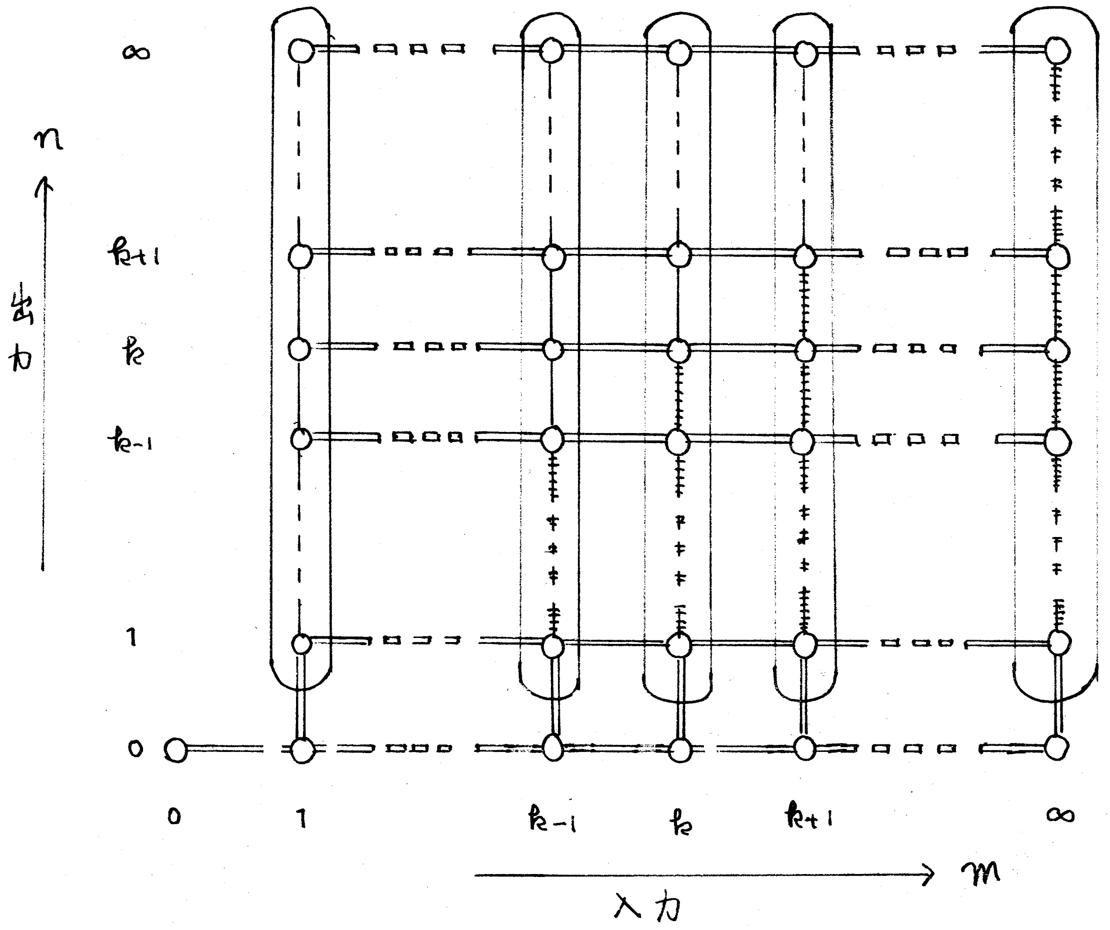
〔系5〕(i) $\forall m \geq 1\ \forall n \in \{1,2,\cdots,m\} : P_R^a(m,n-1) \overset{op}{<} P_R^a(m,n)$

(ii) $\forall m \geq 1\ \forall n \in \{1,2,\cdots,m\} : P_R(m,n-1) \overset{op}{<} P_R(m,n)$ ■

動作等価性はある意味で厳しい等価性と考えられる。定理3-(i),(ii)がこの厳しい等価性について成立し得ないことは，定理3-(iii),(iv)で示されたが，これらがどの程度まで厳しい等価性に基づいて成立するのかは，今後に残された問題である。また，$m<n$なるm, nに対して，$P_R^a(m,n-1) \overset{op}{<} P_R^a(m,n)$，$P_R(m,n-1) \overset{op}{<} P_R^a(m,n)$が成立するかどうかも未解決の問題である。

4. 比較図

3節で得た結果を図示すると，図4になる。図4の各節点



$\left[\begin{array}{l} \forall n \geq 0 \quad P_R[0,n] \overset{def}{=} P_R[0,*] \text{ なので，このクラスを} \\ P_R[0,0] \text{ で示している。} P_R^a[0,n] \text{ についても同じ。} \end{array}\right]$

—— ：包含関係がある。

┼┼┼┼ ：op の意味の真の能力差がある。

═══ ：Val の意味の真の能力差がある。

⬭ ：Val の意味で等価である。

図4 $P_R[m,n]$ ，$P_R^a[m,n]$ の比較図

は，プログラム図式のクラス $P_R[m,n]$ 又は $P_R^a[m,n]$ を表わしている。つまり，$P_R[m,n]$ と $P_R^a[m,n]$ $(m, n = 0, 1, 2, \cdots)$ は同じ構造の比較図を持つ。また，$P_R[m,n]$ と $P_R^a[m,n]$ の間の関係は，

(1) $\forall m \geq 0 \ \forall n \geq 0 \ : \ P_R^a[m,n] \subseteq P_R[m,n]$

(2) $\bigcup_{m,n=0}^{\infty} P_R[m,n] \overset{op}{\equiv} \bigcup_{m,n=0}^{\infty} P_R^a[m,n]$

(3) $\forall m \geq 0 \ \forall n \geq 0 \ : \neg \left(P_R[0,*] \overset{val}{\leq} P_R^a[m,n] \right)$

ですべてがつくされる。

終りに，本研究の機会を与えられた石井ソフトウェア部長，日頃御指導いただく鳥居言語処理研究室長に感謝いたします。

［参考文献］

[1] Paterson & Hewitt : Record of Proj. MAC Conf. P119 (1970)

[2] Constable & Gries : SIAMJ. Computing 1, P66 (1972)

[3] Brown et al. : SIAMJ. Computing 1, P242 (1972)

(4) Chandra & Manna : J. Computer Languages 1, P219 (1975)

[5] 二木, 木村 : 信学論D Vol 59-D, No.10 P725 (1976)

[6] Manna : Mathmatical Theory of Computation, McGraw-Hill, N.Y., (1974)

[7] Aho, Hopcroft & Ullman : The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, Mass., (1974)

付 録（定理の証明の概要）

〔定理1－(i)〕 任意の $S \in P_R[m, n]$ に対して，$S \cong S'$ なる $S' \in P_R^a[m', n']$ を構成出来ることを示す。Sを構成する各フローグラフごとに次の操作により，それに対応した有限個のアサイクリック・フローグラフを作る。

1. フローグラフ中のどのループも少くとも1つの切断点を含むように，適当なエッジ上に切断点を作り，各切断点に新しい再帰サブルティン記号を対応させる。
2. フローグラフを切断点で分断して，次のようなアサイクリック・フローグラフを作る。(a) 各アサイクリック・フローグラフの開始節点は元のフローグラフ開始節点又は切断点である。(b) 各アサイクリック・フローグラフの停止節点は元のフローグラフの停止節点又は切断点である。(c) 各切断点及び元のフローグラフの開始節点を開始節点とするアサイクリック・フローグラフを正確に1つ作る。
3. 切断点以外の節点へは元のフローグラフと同じ命令をラベル付けし，切断点へは次のような命令をラベル付けする。(a) 各アサイクリック・フローグラフの開始節点となっている切断点へは，その切断点に対応する再帰サブ

ルティン記号F_iを使った(S2)タイプの命令 $F_i(\bar{x})$。

(b) 停止節点となっている切断点へは(H2)タイプの命令RETURN($\bar{x}$)。

ここで$\bar{x}$は，元のフローグラフの各節点にラベル付けされた命令中に現われた変数記号全部を集めたベクトルである。

各フローグラフに対し，上述の1,2,3の操作を施こして得られたアサイクリック・フローグラフの集合がS'となる。

〔定理1-(ii)〕 次のようなプログラム図式 $S_{RWH}(m+1) \in P_R[0,*]$ を考える。

```
┌ START(x);
  y1 ← x;
  y2 ← g(y1); y3 ← g(y1); … ; ym+1 ← g(ym);
  L:  if ¬p(y1)      then  HALT(y1)
                     else  y1 ← f(y1);
      if ¬p(y2)      then  HALT(y2)
                     else  y2 ← f(y2);
            ⋮
      if  p(ym+1)    then  HALT(ym+1)
                     else  ym+1 ← f(ym+1);
└ goto L.
```

このプログラム図式は，Restricted Witch Hunt と呼ばれ，Chandra & Manna [4] において導入されたものであり，次のような性格を持つ。

<u>if</u> $(\exists i \geq 0\ \exists j \leq m : p(f^i g^j x) = false)$

<u>then</u> return $f^i g^j x$

<u>else</u> diverge .

[4] ではこういった性質を持つプログラム図式が決して $R[m]$ の中には存在しえないことが証明されている。ところがこの証明が，そのまま $\bigcup_{n=0}^{\infty} P_R^a[m,n]$ に対しても有効であることは容易に了解される．従って，

$$\forall S \in \bigcup_{n=0}^{\infty} P_R^a[m,n] : \neg (S \overset{\text{val}}{\equiv} S_{RWH}(m+1))$$

が示されることになり，定理が証明される。

〔定理2〕 次のような性質を持つプログラム図式を考える。入力 x，unay基本関数 l，r，基本判定 p，ある自然数 m に対し，

<u>if</u> $(\exists a_0, a_1, a_2, \cdots : (a_0 = \varepsilon)$

$\wedge\ (\forall i \in \{0, 1, 2, \cdots\} : a_{i+1} = l \cdot a_i \vee a_{i+1} = r \cdot a_i)$

$\wedge\ (\forall t \in \{lx, lrx, lr^2x, \cdots, lr^{m-2}x, r^{m-1}x\}$

$\forall a \in \{a_0, a_1, a_2, \cdots\} : p(at) = True\ )$

<u>then</u> diverge

```
else  return x.
```

ここで，εは空語を表わしている。このような性質を持つプログラム図式を今後 m-TS (Tree Search) 図式と呼ぶ。

次の $S_{TS}(m) \in P_R^a(m, 0)$ が m-TS 図式であることは容易に了解される。

```
┌ START(x); i←0; x₀←x
  while i < m-1 do
  begin yᵢ ← l(x₀); x₀←r(x₀); i←i+1 end;
  y_{m-1} ← x₀;
  ( ) ← F(y₀, y₁, ···, y_{m-1})
└ HALT(x).
┌ F(x₀, x₁, ···, x_{m-1}):
  if p(x₀) ∧ p(x₁) ∧ ··· ∧ p(x_{m-1})
  then begin
        y₀ ← l(x₀); y₁ ← l(x₁); ···; y_{m-1} ← l(x_{m-1});
        ( ) ← F(y₀, y₁, ···, y_{m-1});
        y₀ ← r(x₀); y₁ ← r(x₁); ···; y_{m-1} ← r(x_{m-1});
        ( ) ← F(y₀, y₁, ···, y_{m-1});
        end;
└ RETURN( ).
```

ここで，mは定数なので，便法として使っている，

while $i < m-1$ do --- といった，一見プログラム図式の定義に反するような記法は取り除くことが出来る点に注意されたい。

ところが，$\bigcup_{n=0}^{\infty} P_R[m-1,n]$ に属するどんなプログラム図式も m-TS図式にはなり得ない。つまり，

$$\forall S \in \bigcup_{n=0}^{\infty} P_R[m-1,n] : \neg(S \overset{val}{\equiv} S_{TS}(m))$$

を示される。直観的には，$m-1$個の入力パラメタしか持たない再帰サブルティンでは，$lx, lrx, \cdots lr^{m-1}x$ の下にぶらさがるm個の木(tree)を同時に探索するアルゴリズムが書けないということである。これから定理が示される。詳細は略す。

〔定理3-(i),(ii)〕 $P_R[m,n]$ に属する任意のプログラム図式Sに対して，$S \overset{val}{\equiv} S'$ なる $S' \in P_R[m,1]$ の存在を示せばよい。S中のn個の出力パラメタを持つサブルティンに対して，S'ではそれをシミュレートするn個のサブルティンを作る。すなわち，S中のサブルティン$F_i^{m,n}$に対し，S'中には $F_{i1}^{m,1}, F_{i2}^{m,1}, \cdots, F_{in}^{m,1}$ なるn個のサブルティンがある。ここで，$F_{ij}^{m,1}$ は $F_i^{m,n}$ と同じ入力ベクトルに対し，$F_i^{m,n}$ の出力ベクトルの第j番目の要素を返すサブルティンである。$F_i^{m,n}$ から $F_{i1}^{m,1}, F_{i2}^{m,1}, \cdots, F_{in}^{m,n}$ を構成するには$F_i^{m,n}$ の

RETURN 文を，RETURN$(z_1,\cdots z_n)$ から，RETURN(z_j) $(1\le j\le n)$ とするだけでよい。ただし，S中のサブルティンの呼び出し文，

$$(y_1,\cdots,y_n)\leftarrow F_i^{m,n}(x_1,\cdots,x_m)$$

は，

$$\underline{\text{begin}}\quad y_1\leftarrow F_{i1}^{m,1}(x_1,\cdots,x_m);$$
$$y_2\leftarrow F_{i2}^{m,1}(x_1,\cdots,x_m)$$
$$\vdots$$
$$y_n\leftarrow F_{in}^{m,1}(x_1,\cdots,x_n)\quad \underline{\text{end}}$$

に変えなければならない。

$S\in R_R^a[m,n]$ に対しても同様である。

〔定理3-(iii),(iv)〕次のプログラム図式 $S_{TK}(n)\in P_R^a[n,n]$ を考える。

```
START(x); i ← 0;
while  i < n-1  do
begin  y0 ← l(x); x ← r(x); i ← i+1  end;
yn-1 ← x;
(y0, y1, ---, yn-1) ← F(y0, y1, ---, yn-1);
z ← h(yn-2, yn-1); z ← h(yn-3, z); ---;
z ← h(y0, z);
HALT(z).
```

$$
\begin{aligned}
&\ulcorner F(x_0, x_1, \cdots, x_{n-1});\\
&\quad \underline{\text{if}}\ p(x_0) \wedge p(x_1) \wedge \cdots \wedge p(x_{n-1})\\
&\quad \underline{\text{then}}\ \underline{\text{begin}}\\
&\qquad y_0 \leftarrow l(x_0);\ y_1 \leftarrow l(x_1);\ \cdots;\ y_{n-1} \leftarrow l(x_{n-1});\\
&\qquad (yl_0, yl_1, \cdots, yl_{n-1}) \leftarrow F(y_0, y_1, \cdots, y_{n-1});\\
&\qquad y_0 \leftarrow r(x_0);\ y_1 \leftarrow r(x_1);\ \cdots;\ y_{n-1} \leftarrow r(x_{n-1});\\
&\qquad (yr_0, yr_1, \cdots, yr_{n-1}) \leftarrow F(y_0, y_1, \cdots, y_{n-1});\\
&\qquad z_0 \leftarrow h(yl_0, yr_0);\ z_1 \leftarrow h(yl_1, yr_1);\ \cdots;\\
&\qquad\qquad\qquad z_{n-1} \leftarrow h(yom_{-1}, yr_{n-1});\\
&\qquad \underline{\text{end}}\\
&\quad \underline{\text{else}}\ \underline{\text{begin}}\ z_0 \leftarrow x_0;\ z_1 \leftarrow x_1;\ \cdots;\ z_{n-1} \leftarrow x_{n-1}\\
&\qquad \underline{\text{end}}\\
&\llcorner \text{RETURN}(z_0, z_1, \cdots, z_{n-1}).
\end{aligned}
$$

次のような解釈I_mを考える。

$$
\begin{aligned}
I_m(p)(t) &= \text{False if } (t = a \cdot b \cdot x \wedge |a| = m\\
&\qquad\qquad \wedge b \in \{l, lr, \cdots, lr^{n-2}, r^{n-1}\})\\
&= \text{True otherwise}
\end{aligned}
$$

ただし，$n=1$ のとき $\{l, lr, \cdots, lr^{n-2}, r^{n-1}\} = \{\varepsilon\}$ と考える。このような解釈に対して，STK(n) は次のような項τを値として返す。

$$\tau = h(\tau_m(lx), h(\tau_m(lrx), h(\cdots h(\tau_m(lr^{n-2}x, r^{n-1}x))\cdots))$$

ここで，$\tau_m(y)$は次のように，帰納的に定義される。

$$\tau_0(y) = y,$$

$$\tau_{i+1}(y) = h(\tau_i(l(y)), \tau_i(r(x)))$$

このようなプログラム図式 STK (n) に対して，

$$\forall S \in \bigcup_{m=0}^{\infty} P_R[m, 0] : \neg(S \overset{val}{\equiv} STK(1))$$

$$\forall n > 1 \quad \forall S \in \bigcup_{m=0}^{\infty} P_R[m, n-1] :$$

$$\neg(S \overset{op}{\equiv} STK(n))$$

が示され定理が証明される。詳細は略す。