Pascalプログラムに対する呼出数最小のモジュール宣言方法について†

阿部寅吉†† 有澤 誠††

大きなプログラムを作成するときに，幾つかのモジュールに分割して設計する方式は，構造化されたソフトウェアの基本である．この分割の後，構造化の進んだ言語では，モジュール宣言の階層構造を決定しなければならない．この階層構造は，モジュール間の呼出関係に依存する．本論文の目的は，モジュール結合度の考え方に基づき，Pascal を対象言語として，モジュールの呼出関係から望ましい階層構造を決定することである．Pascal のモジュールは，プログラム本体，手続き，関数のいずれかであると考える．このとき階層構造に対してひとつの評価値を定め，この値から望ましさを判断する．階層構造を決定する方法は，Pascal の特徴から forward 指定を必要としない場合と，必要とする場合とに分かれる．前者では，望ましい階層構造を求める方法を示す．しかし後者では，forward 指定に関する望ましさの判定が困難なため，単に階層構造を求める方法を示す．またこれらの過程から，forward 指定を必要とする条件や，特別な呼出関係に対する階層構造の性質などについても示す．

1. はじめに

大きなプログラムを作成するときに，幾つかのモジュールに分割して設計する方法は，構造化されたソフトウェアの基本である．このモジュール化の良否に対する評価尺度として強度と結合度[1]がある．結合度の中で最も基本的かつ重要な要素は，モジュール呼出しに関する尺度である．したがって，モジュールの呼出関係から各モジュールの位置付けをすることは，意義がある．本論文では言語を Pascal に限定し，モジュール結合度の考え方から，Pascal のモジュールを定義する．

本論文の目的は，Pascal において各モジュールの呼出関係から望ましいモジュールの階層構造（以後宣言関係と呼ぶ）を求めることである．特に，宣言関係に対してひとつの評価値を定め，この値から最も適当な宣言関係を求めることに限定する．パラメタ授受や大域変数へのアクセスなどは，本論文では考察の対象から除く．

本論文でとりあげる評価値は呼出数であり，与えられた宣言関係で呼出し可能な対の数を表す．プログラムで実際に生じる呼出関係に対して，どのくらい冗長な呼出しが可能かを示しており，他の条件が同じならこの値が小さいほどモジュールの独立性が高いとみることができる．与えられた呼出関係に対し，それ以外の冗長な呼出しの可能性がなるべく少ない宣言関係を求めることが，本論文の目的である．

第2章で，準備のため種々の定義や Pascal の性質を述べる．第3章で，宣言関係に対する評価値として呼出数を定義し，その特徴を述べる．第4章と第5章は，呼出関係から宣言関係を求める方法を示す．forward 指定の有無により方法が異なるため，第4章は forward 指定を必要としない場合について，最適な宣言関係を求める方法を示す．第5章は forward 指定を必要とする場合について，その条件と宣言関係を求める方法を示す．

2. 準　備

本章では，3章以下で引用する種々の定義や基本事項について述べる．

2.1 モジュールの定義

Pascal の手続き，関数およびプログラム本体を，モジュールと定義する．したがって，モジュールを並列におくほかに，モジュール内部に別のモジュールを含むという入れ子の状態が存在する．以下では特に断わらない限り，モジュール名を a，b，c，…の記号（ローマ小文字）で表す．

2.2 モジュールの呼出関係

モジュールの本体で，自己または他のモジュールを呼び出す文が存在するとき，モジュールの呼出しがあると定義する．実行中に実際の呼出しが起こるかどうかにはかかわらない．モジュール a がモジュール b を呼び出すとき，$a \Rightarrow b$ と表す．またモジュール $c_1, c_2, \cdots, c_k$ $(k \geqq 2)$ が，$c_i \Rightarrow c_{i+1}$ $(1 \leqq i \leqq k-1)$ ならば，$c_1 \overset{+}{\Rightarrow} c_k$ と表し，c_1 から c_k へ呼出経路が存在すると

† Minimum Calling Number Module Declaration Method for Pascal Programs by TORAKICHI ABE and MAKOTO ARISAWA (Department of Computer Science, Yamanashi University).

†† 山梨大学工学部計算機科学科

いう.

Pascal の呼出関係には，文法規則により性質 1a～性質 1b の二つがある.

[性質 1a] すべてのモジュールの中で，プログラム本体であるメイン・モジュールが唯一つ存在する. 以後，これを m と表す.

[性質 1b] m を呼び出すモジュールは存在しない.

また本論文では，冗長なモジュールが存在しないことを前提とする. これを性質 1c とする.

[性質 1c] m を除く任意のモジュール x において，m から x へ呼出経路が存在する. すなわち m⇒x である.

したがって，本論文で用いる呼出関係 ⇒ は，性質 1a～1c を満たす有向グラフになる.

ひとつのプログラムに含まれるすべてのモジュールの集合を S で表す. $S=\{m, a_1, a_2, \cdots, a_k\}$ とするとき，呼出関係の中で代表的なパターンを(a)～(d)に示す. 以下，モジュールの集合は，A, B, C, …などの記号（ローマ大文字）で表す.

(a) 線形呼出し：$m \Rightarrow a_1 \Rightarrow a_2 \Rightarrow \cdots \Rightarrow a_k$ となる呼出し.

(b) 循環呼出し：$m \Rightarrow a_1 \Rightarrow a_2 \Rightarrow \cdots \Rightarrow a_k \Rightarrow a_1$ となる呼出し.

(c) 木状呼出し：⇒ に関する有向グラフが，m を根とする木となる呼出し.

(d) 半順序呼出し：集合 S が関係 ⇒ に対して半順序となる呼出し.

以下 S の要素を節点，⇒ を枝とする有向グラフについて，呼出関係の性質をグラフ理論の用語を用いて表す.

部分集合 $T \subset S$ において，T に含まれる任意の要素 p，q が，$p \Rightarrow q$ かつ $q \Rightarrow p$ となるとき，T を強連結なグラフという. S を強連結なグラフごとに分割したとき，各成分を S の強連結成分という. 部分集合 $R \subset S$ において，$x \Rightarrow y\ (x \notin R, y \in R)$ となるとき，y を R の入口点という.

2.3 モジュールの宣言関係

Pascal の各モジュールは，互いに並列または入れ子の位置に宣言する. そこで，宣言の位置関係を家系図に対応させて表現する. たとえば，**図1**の宣言のときは，次のように親，子などの言葉を用いる.

親：m は a，b，d の親. b は c の親. d は e の親.

子：a，b，d は m の子. c は b の子. e は d の子.

兄：a と b は d の兄. a は b の兄.

伯父：a は c の伯父. a と b は e の伯父.

先祖：m と b は c の先祖. m と d は e の先祖.

子孫：a～e は m の子孫.

ここで，伯父系という位置関係を定義する. モジュール x の伯父系とは，x の兄，x の親，x の伯父，x の先祖，および x の先祖の兄のことである. 例えば図1では，次のようになる.

伯父系：m，a，b は c の伯父系. m，a，b，d は e の伯父系.

また，x の直系とは，x から長子だけでたどってゆける要素を指す. x の直系第一子とは，x の直系の中で，子孫側で x に最も近い要素を指す.

宣言関係は，**図2**のような二分木として表現する. この二分木による表現は，図1のような表現と等価である. 図1を二分木表現したものが，**図3**である.

2.4 モジュールの呼出規則

名前の有効範囲規則[2]から，宣言上の位置関係によって，呼出しが可能かどうかがわかる. すなわちモジュール x に対して，モジュール y が x 自身，x の子または m を除く伯父系のとき，x から y へ呼出可能である. これ以外の位置関係から呼び出す手段として，forward 指定が用意されている. モジュール x に対してモジュール y を forward 指定できる位置は，x の兄の位置でなければならない. x の forward 指定をすることにより新たに x が呼出可能となるのは，forward x と x の間にある任意のモジュール y とそ

```
program m;
  procedure a;
  procedure b;
    procedure c;
  procedure d;
    procedure e;
```

図1 モジュール宣言の例
Fig. 1 Example of module declaration.

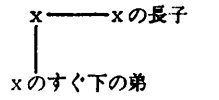

図2 宣言関係の二分木表現
Fig. 2 Representation for binary tree of declaration.

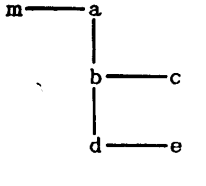

図3 図1の二分木表現
Fig. 3 Representation for binary tree of Fig. 1.

⇒	m	a	b	c	d	e
m	×	○	○	×	○	×
a	×	○	×	×	×	×
b	×	○	○	○	×	×
c	×	○	○	○	×	×
d	×	○	○	×	○	○
e	×	○	○	×	○	○

図4 図3の呼出可能な位置
Fig. 4 Position possible to call of Fig. 3.

の子孫である.

図3に対して，呼出し可能な位置を**図4**に示す．○印は呼出し可能を表し，×印は不可能を表す.

3. モジュール呼出数

本章では，宣言関係から呼出数という値を算出し，これが何を意味するのかについて述べる．呼出数は宣言関係固有の値で，呼出関係から宣言関係を導くときに用いる基準値となる.

3.1 呼出数の定義

呼出数とは，宣言関係の中で呼出し可能な位置関係の総和であり，次のように定義する.

[呼出数の定義]

モジュールxに対して，xから呼出し可能な位置にあるすべてのモジュール数を C_x とする．プログラム中の全モジュールの集合を S とするとき，呼出数は $\sum_{x \in S} C_x$ で与えられる．■

x から呼出し可能な位置とは，2.4節で述べたように，x 自身，x の子および x の伯父系である．S のモジュール総数を n とすると，自分自身の呼出しに関するものは $n-1$，自分の子の呼出しに関するものも $n-1$ だけある．したがって，n に対して呼出数が変化するのは，伯父系の数である.

3.2 二分木の路長

呼出数は，宣言関係を表す二分木の内部路長[3),4)]と，次のような関係がある.

[定理1] 呼出数を c，内部路長を l，モジュール総数を n とすると，$c=l+n-1$ である.

[証明] 内部路長とは，二分木の各節点から根までの距離の総和である．任意の節点に対し根までの距離を p とすると，その経路上にあるmを除く伯父系の数は $p-1$ である．$\Sigma p=l$ であるから，全伯父系の数は，$\Sigma(p-1)=l-(n-1)$ である．呼出数はこのほかに，自分自身と子への呼出しも含むので，

$$c=l-(n-1)+2(n-1)=l+n-1$$ ■

これにより，二分木の路長に関する性質はそのまま呼出数に適用できる．呼出数の最大値は，

$$\frac{1}{2}(n-1)(n+2)$$

であり，最小値は $q=\lfloor \log_2 n \rfloor$ とすると，

$$n(q+2)-2^{q+1}$$

である．(ただし $\lfloor\ \rfloor$ は floor 記号[3)]である.)

3.3 呼出数の最小化と最適性

ここでは，呼出数の最も小さい宣言関係が最適な宣言である，ということを提案する.

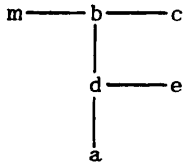

図5 図3のaを末子に移した宣言
Fig. 5 Declaration moved a in Fig. 3 to last.

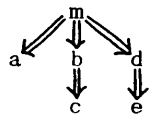

図6 図3と図5を満足する呼出関係の例
Fig. 6 Example of calling relation to satisfy Fig. 3 and Fig. 5.

図3の宣言関係に対し，aをmの長子から末子へ入れ替えたものが**図5**である．図3と図5の宣言関係は，例えば**図6**の呼出関係を満足する．図6において呼出しは五つあるが，図3と図5から得られる呼出し可能な位置の総数すなわち呼出数は，それぞれ18と16である．このことから，図3よりも図5のほうが，図6の呼出関係を忠実に表現していると考えられる．なぜなら，実際に呼出しがないにもかかわらず，宣言上呼出し可能となってしまう位置関係が少ないほど，呼出関係をよく表しているからである.

呼出数は，宣言関係に対する値であるから，宣言関係を組み替えれば，当然変化する．ある呼出関係を満足する宣言関係の中で，呼出数最小のものは，無駄な呼出し可能位置が少なく，最も呼出関係を反映した宣言関係である．よって，呼出数最小の宣言関係が最適であると考え，呼出関係から呼出数最小の（最適な）宣言関係を得ることを目標とする.

4. forward 指定が必要な呼出関係

呼出関係から宣言関係を生成するとき，呼出関係の中に特別なパターンのグラフが存在すると，forward 指定が必要になる．ところが，forward 指定を必要とする場合，最適な宣言関係を導出するのが困難なため forward 指定の有無により区別して取り扱う.

本章では，forward 指定が不必要な呼出関係について，宣言関係を生成するアルゴリズムと，最適な宣言関係を生成する方法を示す.

4.1 宣言生成アルゴリズム

まず，基本的な考え方を示す．任意の要素xの宣言に対して，重要な性質が二つある.

[性質2a] x の任意の子孫を y とすると，m から y へ至る呼出経路中に，必ず x が存在する．すなわち，m ⇒ x ⇒ y である.

[証明] 性質1cより，mからyへ呼出経路が存在

する．xの子孫の集合を D とすると，m⇏y より，D の外部のある要素 p から D 内のある要素 q へ呼出しがある．宣言上呼び出せる位置は，子または伯父系であるから，p=x かつ q は x の子でなければならない．結局 m⇏y の経路中に，必ずxが存在する．■

［性質 2b］ xの任意の弟側の要素をzとすると，xを通らずにmからzへ至る呼出経路が存在する．すなわち，m⇏z（xを通らない）である．

［証明］ xの弟側の集合をBとすると，Bの外部から内部へ呼出経路があり，外部はxの親で内部はxの弟である．xの親は，xを通らずにmから呼出経路があり，またzへ呼出経路がある．よって，xを通らずにmからzへ至る呼出経路が存在する．■

これらのことから図7のアルゴリズムを得る．手続き make_dcl は，宣言関係を表す二分木を top-down で再帰的に生成する．再帰の各段階で，渡されたモジュールの集合 S から節点となる要素をひとつ選出し，残りの要素を左右の枝へ振り分ける．make_dcl の最初に与えるものは，全モジュールの集合 S と呼出関係 C，さらに生成する二分木の根を指すポインタ b である．S 内のすべての要素について until 節が偽となると，このアルゴリズムは失敗する．このとき，宣言を生成するためには forward 指定が必要になるので，5章で詳しく論じる．宣言の生成が滞りなく行われたときアルゴリズムは成功である．そこで，アルゴリズムが成功したとき，得られる宣言関係が与えられた呼出関係を満足していることを，証明する．

［証明］ 任意の呼出し x⇒y が，アルゴリズムの適用によって，保たれることを示す．このとき，節点のとりようによって(a)～(c)の三つの場合がある．

```
procedure make_dcl (S : set of modules ; C; var b) ;
  var   R, L, T : set of modules ;
        pι : module in S;
begin {make_dcl}
  if  S = ∅  then
    b := nil
  else
  begin
    repeat
      /*S内の任意の要素pι を次々に選ぶ*/;
      T:=S − {pι };
      R:= {x | x∈T, pι ⇏x};
      L:= {x | x∈T, m⇏x (pι を通らない)}
    until   R∩L = ∅;
    new(b);
    /*pι をbの先に登録する*/;
    make_dcl( R, C, b^.right );
    make_dcl( L, C, b^.left );
  end {if}
end {make-dcl};
```

図 7 宣言生成アルゴリズム
Fig. 7 An algorithm for producing the module declaration.

(a) xが先に節点に選ばれたとき

yは R に含まれるので，yはxの子孫である．このときyはxの子であることを示す．背理法を用いて，yがxの子でないとすると，xのある子wの子孫でなければならない．するとwも R に含まれて，x⇏w かつ w⇏y である．ところが x⇒y なので，wが節点に選ばれるとき $y\in(R\cap L)$ となり，矛盾が生じる．したがって，yはxの子であり，x⇒y は保たれる．

(b) yが先に節点に選ばれたとき

xが R と L のどちらに含まれても，xからみてyは伯父系となるので，x⇒y は保たれる．

(c) xとy以外の要素zが選ばれたとき

z⇏x ならば，x⇒y より z⇏y も成り立つので，xとyは共に R に含まれる．z⇏x でなければ，x は L に含まれる．このとき z⇏y とすると，m⇏z⇏y かつ m⇏x⇒y であるから，$y\in(R\cap L)$ となって矛盾する．よって y も L に含まれる．いずれの場合も，xとyは同じ側へ含まれるので，帰納的に(a)または(b)へ帰着する．■

4.2 循環呼出し

呼出関係の中で特に単純なものは，容易に最適な宣言関係を生成することができる．また，このような呼出関係は種々の性質をもっている．ここでは循環呼出しについて，種々の性質と最適な宣言方法を示す．

循環呼出しとは定義 1b で示したとおり，モジュールの集合 $S=\{m, a_0, a_1, a_2, \cdots, a_k\}$ に対して，

$$m \Rightarrow a_0,\ a_0 \Rightarrow a_1, \cdots, a_i \Rightarrow a_{i+1}, \cdots, a_k \Rightarrow a_0$$

という呼出関係である．循環呼出しを可能にする宣言関係を循環宣言と呼ぶ．特に S が $k+2$ 個のモジュールからなるとき，k 次循環宣言と呼ぶ．循環宣言は，以下に示す 3a～3f の性質がある．

［性質 3a］ a_0 はmの長子にあたり，a_1～a_k は a_0 の子孫である．

［性質 3b］ a_k は a_0 の直系第一子である．二分木で表現すると，a_k は最上段の右端に位置する．

［性質 3c］ $(k+1)$ 次循環宣言は，k 次循環宣言から生成できる．

［系］ 性質 3b より，a_{k+1} を a_k の子，兄または先祖の兄の位置に挿入する．

［性質 3d］ k 次循環宣言は，k 個の節をもつ二分木構造と1対1対応する．

(a) 循環宣言 ⇒ 二分木構造

循環宣言を二分木表現し，各節の内容と根(m)およ

び節 a_0 を取除く.

(b) 二分木構造 ⇒ 循環宣言

二分木をインオーダ[3]で探索し，節に順次 $a_1, a_2, \cdots, a_k$ を割り当てる．そして，根の親の位置に節 a_0 を，根の祖父の位置に節 m を付け加える.

[系] k 次循環宣言は，スタック操作，カッコ列，多角形の三角形分割と，それぞれ1対1対応する[3].

[性質 3e] k 次循環宣言と，どの部分列 $\cdots a_i \cdots a_j \cdots a_l \cdots$ も不等式 $j<l<i$ を満足しないような順列 $\{a_1, a_2, \cdots, a_k\}$ とは，1対1対応する.

[性質 3f] k 次循環宣言の総数は，Catalon 数 $\frac{2 \cdot {}_nC_n}{n+1}$ である.

循環呼出しから最適な宣言を求める方法を示す.

[操作1]

(1) 内部路長が最小となる二分木を求める．(必ずしも完全二分木でなくてもよい．二分木の全長を k とすると，$k-1$ 段目が節点で埋め尽くされていればよい.)

(2) この二分木から性質3dを用いて宣言を生成する.

4.3 一般呼出しの最適な宣言方法

特徴のある呼出関係の中には，最適な宣言関係を求めるより効率的な方法をもつものがある．ここでは，木状呼出しと半順序呼出しについて，最適な宣言関係を求める方法を示し，forward 指定を必要としない一般の呼出関係から最適な宣言関係を求める指針を示す.

木状呼出しの場合，操作2を行って最適な宣言関係を求める.

[操作2]

(1) 呼出関係から宣言関係の初期状態を求める．初期状態とは，a ⇒ b という呼出しに対しbをaの子に宣言した状態である．したがって，初期状態は木状呼出しの関係を満足する.

(2) 初期状態から，浮上操作と呼ぶ操作を可能な限り行う．操作の各段階（初期状態を含む）で得られる宣言関係は，すべて最適の候補となる．浮上操作とは，図8に示すように宣言関係の中でpとqが親子の関係にあるとき，q以下の部分宣言をpの兄弟の位置に移す（浮上する）ことである．ただし，次の(a)と(b)の場合は操作してはならない.

(a) 親がmのとき

(b) 子が木状呼出しの末端のとき

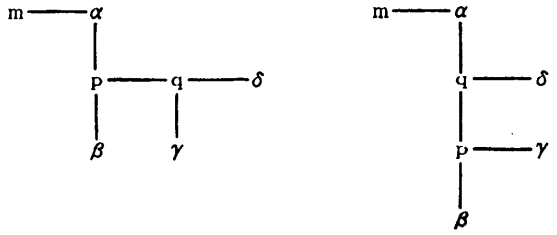

図 8 浮上操作の例
Fig. 8 Example of the Bubble-Operation.

浮上操作は(a),(b)を除く任意の要素に適用できるうえ，操作の順序によって得られる宣言が異なるため，すべての場合を尽くさなければならない.

(3) 得られた候補の各々について兄弟の順序を最適な形に入れ替え，呼出数を計算し，最適な宣言を求める．初期状態では兄弟の順序に制約がない．この場合，子孫の数の多い順に入れ替える．しかし，浮上操作を行う度に兄弟の順序に制約が加わるので，このときは制約の範囲内で子孫の数の多い順に入れ替える. ■

半順序呼出しの場合は，複数の要素から呼び出されるコモン・モジュールが存在するため，木状呼出しと同じように(1)を行うことができない．そこで，(1)について修正した操作3を行う.

[操作3]

(1) コモン・モジュール以外については，操作2(1)に従う．コモン・モジュールの中には，mからの任意の経路中に他のコモン・モジュールを含まないものが必ず存在する．このような要素をcとすると，mからcへ至るすべての経路中に共通に存在する要素の中で最もcに近いものを選び（これをdとする），cをdの子として宣言する．そして，呼出関係からcを呼び出す関係をすべて取り除き，d ⇒ c という関係を追加する．このようにして，コモン・モジュールがすべて宣言されるまでこの操作を繰り返す.

(2) 操作2(2)と同じ.

(3) 操作2(3)と同じ. ■

こうして得られる初期状態は，既に兄弟の順序に制約をもっている.

forward 指定を必要としない一般の呼出関係については，今までのことから操作4を行えば最適な宣言関係が求められる.

[操作4]

(1) 呼出関係を表すグラフを，強連結成分ごとに分解する.

(2) 各成分内で，最適化を図る．

(3) 各成分間は半順序なので，操作3(1)により成分単位の初期状態を求める．

(4) 成分単位で浮上操作および兄弟の入れ替えを行い，最適な宣言を求める．■

しかし，現在のところ(2)の方法が確立していない．成分内は循環呼出しを含んでいるが，それを取り除いても残るグラフがやはり一般の呼出関係となるため，最適化を図るのは難しい．

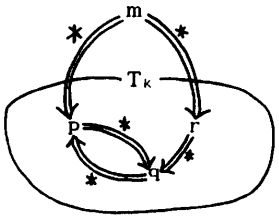

図9 極大成分 T_k
Fig. 9 A maximum element T_k.

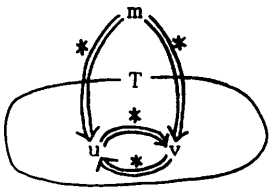

図10 複数の入口点を持つ強連結なグラフ T
Fig. 10 A strong components graph T with plural entry points.

5. forward 指定を必要とする呼出関係

呼出関係から宣言を構成するとき，forward 指定を必要とするかどうかによって，方法が分かれる．必要とする場合，最適な宣言を求めることは困難である．よって本章では，呼出関係から宣言関係を求める方法を示す．5.1 節で forward 指定の必要条件や宣言生成アルゴリズムとの関係を述べ，5.2 節で forward 指定をする宣言方法を示す．

5.1 forward 指定を必要とする条件

forward 指定が必要かどうか調べるために，次の定理がある．

[定理 2] 与えられた呼出関係に対し，(1)，(2)，(3)は互いに同値である．

(1) 宣言関係は，forward 指定を必要とする．

(2) 図7の宣言生成アルゴリズムが失敗する．

(3) 呼出関係の中に，複数の入口をもつ強連結なグラフが存在する．

[証明] (1)→(2)，(2)→(3)，(3)→(1)をそれぞれ示す．

(1)→(2)：

図7の宣言生成アルゴリズムが成功したと仮定する．これは forward 指定をしなくても，呼出関係に矛盾することなしに，全節点を選べることを意味する．こうして生成した宣言関係には forward 指定がなく，(1)と矛盾する．したがって，宣言生成アルゴリズムは失敗する．

(2)→(3)：

宣言生成アルゴリズムが失敗したとき，渡されたモジュールの部分集合を F とする．失敗すれば，任意の要素 $x\in F$ に対し，

$$x \Rightarrow y \text{ かつ } m \not\Rightarrow y \text{ (xを通らない)} \qquad (*)$$

となる要素 $y\in F$ $(x\neq y)$ が存在する．F を強連結成分ごとに分割し，その各成分を T_1, T_2, …, T_n $(n\geqq 2)$ とする．成分間の関係は半順序であるから，この中に極大成分が存在する．極大成分はすべて二つ以上の要素をもつ．なぜなら要素数1の成分が存在すれば，その成分の要素を節点に選ぶとき，(*)で $x \Rightarrow y$ となる y が存在せず，宣言生成アルゴリズムが成功するからである．極大成分のひとつを T_k とし，T_k のある入口点を p とする．$p\in F$ だから(*)よりある $q\in F$ が存在して，$p \Rightarrow q$ かつ $m \not\Rightarrow q$ (pを通らない) となる．さらに T_k が極大成分であるから，$p \Rightarrow q$ ならば $q\in T_k$ である．また T_k は強連結なグラフでもあるので $q \Rightarrow p$ である．これらの関係をまとめると**図9**のようになる．ここで $m \Rightarrow q$ よりこの経路中に T_k の入口点が存在し，これを r とする．$m \Rightarrow q$ は p を通らないから $p\neq r$ だが，$q=r$ であってもよい．結局，F の中に複数の入口点 p，r をもつ強連結成分 T_k が存在する．これは，S から見ると強連結なグラフである．

(3)→(1)：

図10のように複数の入口点 u，v を持つ強連結なグラフ T があるとき，forward 指定をせずに宣言関係を生成できたと仮定する．u に対して v のとりうる宣言上の位置は，次の三つに分かれる．

(a) u が v の子孫のとき

v を通らない呼出経路 $m \not\Rightarrow u$ は存在しない．

(b) v が u の子孫のとき

u を通らない呼出経路 $m \not\Rightarrow v$ は存在しない．

(c) (a)や(b)以外のとき

T は循環呼出でもあるから，性質 3a より宣言において根となる要素 w が存在し，u，v も含め他の要素は w の子孫となる．このとき性質 2a より，呼出経路 $m \Rightarrow u$ および $m \Rightarrow v$ は必ず w を経由する．ところがこれは，u や v が T の入口点であることと矛盾する．

(a)，(b)，(c)いずれの場合も，矛盾を生じる．したがって，図10のような呼出関係があるとき，forward 指定なしで宣言を生成することはできない．■

5.2 forward 指定をする宣言生成方法

これは4.1節の方法に対して，失敗したときの操作を含めたものである．4.1節の方法が失敗したときにforward指定が必要であることを，5.1節で証明した．そこで失敗したときに節点をforward指定し，呼出グラフから今後吟味する必要のない呼出関係を削除して，続行する操作を示す．4.1節の方法で宣言生成の最中，ある節点で失敗したとき，操作5を行って続行する．

[操作5]

(1) 渡された集合 F から，まだforward指定されていない要素をひとつ選ぶ．これをxとする．

(2) xをforward指定し，節点に登録する．

(3) F に含まれる任意の要素yに対し，$y \Rightarrow x$ となる呼出しがあれば，これを呼出グラフから削除する．

(4) F をすべてforward xの弟側に渡す．■

この方法が正しいことを保証するために，定理3の(a)～(c)を証明する．

[定理3]

(a) 生成した宣言関係は，呼出関係を満足する．

(b) forward指定を繰り返すと，操作(1)で選べる要素が減ってゆく．しかし，要素が尽きてこの方法が失敗することはない．

(c) forward指定をした要素は，必ず指定位置の弟として登録される．

[証明]

(a) xをforward指定すると，F に含まれる任意の要素は，xへ呼出し可能となる．xから他への呼出しは，forward指定以後xを節点に登録するときに，4.1節の方法で吟味できる．xへの呼出しもxからの呼出しも保証できるから，生成した宣言関係は呼出関係を満足する．

(b) 失敗したとき，xはある強連結成分に含まれている．しかし操作5(3)によりxはその成分から外れる．forward指定ごとに強連結成分から要素がひとつずつ外れてゆくから，F の要素数を n とすると，最悪でも $(n-1)$ 回forward指定すれば強連結成分はなくなり，4.1節の方法が適用できる．したがって，要素が尽きて失敗することはない．

(c) xをforward指定する．もしxがforward xの弟でなければ，forward xのある弟zの子孫である．zを節点に登録するとき4.1節の方法を用いるから，$z \Rightarrow x$ である．ところが操作5(3)で，xを呼び出す関係をすべて削除しているはずだから，これは矛盾である．したがって，zは存在せずxはforward xの弟である．■

6. おわりに

本論文は，呼出関係から宣言関係を求める方法を述べ，特に呼出関係の一部に対しては最適な宣言関係を求める方法を示した．しかし，以下の問題点が残っている．

(1) 3.3節で述べた呼出数の最小化は，読みやすさ，理解しやすさという点で必ずしも最適とは言えない．何が最適なのかということも明確ではない．

(2) 同様なことはforward指定を必要とするときにも言える．forward指定の数や位置関係も最適さの要因となるため，最適な宣言関係の定義が難しい．

(3) 木状や半順序の呼出関係から最適な宣言関係を求める方法は，効率が良くない．

なお，本論文と同様の目的で，ALGOL 60に対して考察した論文[5]が最近発表された．Pascalと異なりforward指定を必要としないスコープ規則であるために，結果はかなり異なっている．

現在，本論文で述べた方法を具体化して，Pascalソース・プログラムのモジュールを最適な形に組み替えるトゥールを開発中である[6),7)]．この場合，本論文で議論したモジュールの呼出関係と宣言関係だけでなく，パラメタ授受，大域変数へのアクセスなども考慮する必要があり，さらに幾つかの問題が残されている．

参考文献

1) Myers, G. J.(國友義久，伊東武夫訳)：ソフトウェアの複合/構造化設計，pp. 57-94，近代科学社，東京 (1979).
2) Jensen, K. and Wirth, N. (原田賢一訳)：PASCAL, pp. 75-90，培風館，東京 (1981).
3) Knuth, D. E.: *The Art of Computer Programming, Vol. 1: Fundamental Algorithms*, pp. 234-239, 316-320, 399-405, Addison-Wesley, Reading, Mass. (1973).
4) Wirth, N.(片山卓也訳)：アルゴリズム＋データ構造＝プログラム，pp. 215-276，日本コンピュータ協会，東京 (1976).
5) Crookes, D. and Devlin, C. J.: An Algorithms for Optimal Procedural Nesting, *Comput. J.*, Vol. 28, No. 1, pp. 17-21 (1985).
6) 阿部寅吉ほか：プログラム評価トゥールの開発(1)，第28回情報処理学会全国大会論文集，4B-1，p. 627 (1984).
7) 阿部寅吉，有沢　誠：Pascalプログラムのモジ

ュール組換えに関する考察，第31回全国大会講演論文集，2E-5，p. 315，情報処理学会（1985）．

（昭和60年9月27日受付）
（昭和61年7月16日採録）

阿部　寛吉（正会員）

1962年生．1984年山梨大学工学部計算機科学科卒業．1986年同大学院工学研究科修士課程修了．現在NTTに勤務．在学中の研究テーマは，ソフトウェア工学．特にソフトウェアの評価，アルゴリズムの解析に興味をもっている．

有澤　誠（正会員）

1944年生．1967年東京大学工学部計数工学科卒業．電子技術総合研究所を経て，現在山梨大学工学部計算機科学科に勤務．工学博士．ソフトウェア工学，特にソフトウェアの評価，アルゴリズムの解析，オブジェクト指向システムなどに興味をもっている．