特性多項式によるプログラム複雑度の特徴づけ†

有澤 誠††

プログラムの複雑度の尺度の一つである McCabe 尺度の拡張について考察する．McCabe 尺度は，プログラム中の独立な実行可能径路数を複雑度とするものであるが，ループについてはダミー変数をあてはめて，特性多項式の形に拡張するという Cantone の提案がある．また構造化されたプログラムについては，制御フローグラフの代りに木構造でモデル化する Tamine の提案もある．本稿では，Jackson の構造図に再帰構造を補い，その新しい構造図で表現したプログラムの複雑度を，特性多項式で特徴づける方法を提案する．この特徴づけは，Cantone や Tamine によるものの一般化にあたっている．とくに再帰構造をもつプログラムについても，特性多項式，あるいは多項式を閉じた形にした特性式によって，複雑度を特徴づけできることを示す．

1. まえがき

プログラムの複雑度を客観的定量的に評価することの重要性は，ソフトウェア製品のライフサイクルで，保守の部分にコストの大半がさかれている事実によって，はっきり示されている．またソフトウェアの開発工程に品質管理や品質保証を導入するためにも，理解しやすさあるいは改修しやすさの尺度としての，プログラムの複雑度のもつ意義は大きい．

これまで提案されてきた尺度は，プログラムの制御フローグラムの性格をもとにした McCabe 尺度[1]と，プログラム中のオペレータおよびオペランドの出現率をもとにした Halstead 尺度[2]とが，有望視されている．このどちらについても，種々の改良案や拡張案が発表されている．

本稿では，McCabe 尺度の一つの拡張として提案されている特性多項式を用いる手法をとりあげる．まず2章で，これまでに提案されている内容を述べる．次に，Jackson[3]による構造図を，再帰構造を記述できるように拡張することについて3章で述べる．この拡張された Jackson 図を用いて，特性多項式による複雑度の特徴づけを行う手法について，4章で論じる．

2. Cantone と Tamine の特性多項式

McCabe によるプログラムの複雑度は，プログラムを制御フローグラフに変換し，そのグラフのサイクロマティック数を求めて複雑度とする．この数は，

(枝の数)−(節点数)+2

で計算でき，開始点から終了点への独立な径路の数に等しい．しかし，単なる条件分岐による場合とループによる場合を区別せずに独立な径路を求めるため，プログラムの複雑度として不十分な面をもつ．直観的には，ループは単純な分岐よりも複雑度を増すとみることができるからである．

Cantone[4]は，分岐によって生じる独立な径路数の増加と，ループとを分離することを考えた．一つループがある場合，ダミー変数 c を導入して，c と書く，ループが二つ並んでいれば c^2 である．分岐とループの組合せは，たとえばループの中味が2分岐であれば $2c$ となり，2分岐のそれぞれの中味がループであっても，$2c$ となる．ループが2重にネストした場合は c^2 である．Cantone による複雑度の計算例を**図1**にあげる．Cantone の複雑度は，McCabe の複雑度のような整数値ではなく，ダミー変数 c に関する多項式になる．これを特性多項式とよぶ．

Cantone の方法では，二つのループがネストしていても，縦列に並んでいても，同じ c^2 という複雑度になってしまう点が望ましくない．プログラムが構造化されていれば，制御フローグラフの代りに，木構造を用いてプログラムの構造を表せる点を合わせて，改良案を Tamine[5]が示した．Tamine は，正方形の節点で一般の文を表し，円形の節点で分岐構文を表す．文の連接は，まとめる範囲の節点の下に矢印を導く．ループについては，一段深くネストさせる．すなわち，木構造のネストはループの深さに対応する．ここで，異なるネスト水準のループには，異なるダミー変数をわりあてるのが Tamine の方法である．ループが二つの縦列に並んだものは $2a$ となり，ループを二つネストさせたものは ab となって区別できる．

Tamine の特性多項式は，文の連接についてはそれぞれの部分の積をとり，分岐については分岐先それぞ

† Characteristic Polynomial Oriented Program Complexity by MAKOTO ARISAWA (Department of Computer Science, Faculty of Engineering, Yamanashi University).

†† 山梨大学工学部計算機科学科

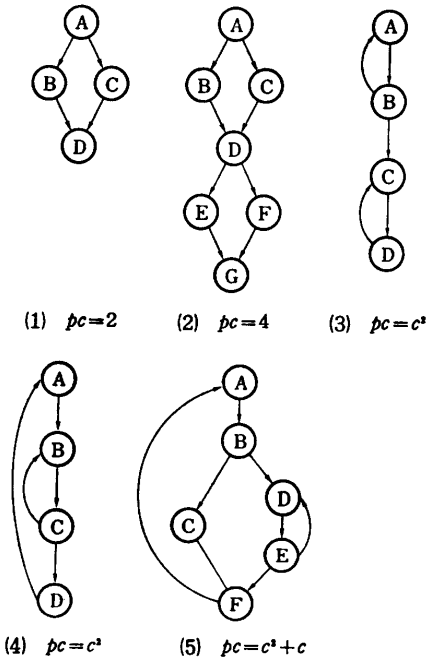


図 1 Cantone の特性多項式の計算例

Fig. 1 Examples of Cantone's characteristic polynomial.

れの多項式の和をとる．ループについてはそのネスト水準に対応するダミー変数を与える．この三つを組合せて，プログラムの特性多項式を計算する．さきほどと同じ例題について Tamine の方法を適用したものを**図 2** に示す．この図からわかるように，2分岐の先がそれぞれループである場合の多項式は $2a$ となり，ループの中味が2分岐の場合の多項式もやはり $2a$ となる．このことは，たとえば次の二つの文の複雑度を等しいとみなすことである．

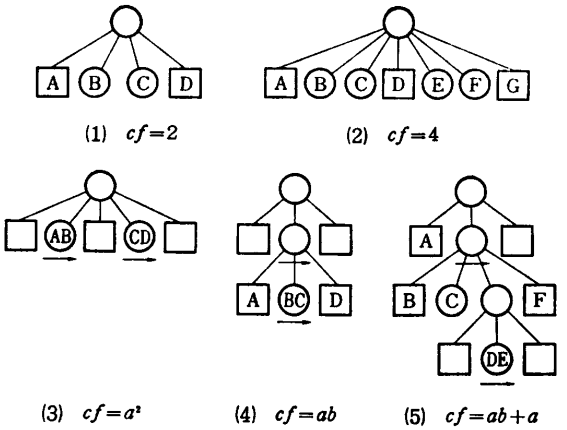


図 2 Tamine の記法で図1の例を書いた結果

Fig. 2 Same examples as Fig. 1 in Tamine's notation.

(1) **if** p **then for** i:=1 **to** n **do** $a[i]:=f(i)$
else for i:=1 **to** n **do** $a[i]:=g(i)$

(2) **for** i:=1 **to** n **do if** p **then** $a[i]:=f(i)$
else $a[i]:=g(i)$

この例は人工的であって現実的ではないが，実行効率からみると(1)がすぐれており，複雑度からみてもキイストローク数は多いが(1)のほうが単純である．(2)は p が i に依存する述語であるような一般形の特別な場合にあたるとみるべきで，(1)より複雑度は高いと考えるのが自然であろう．

なお，Tamine の多項式で，$a=b=c$ のようにすべてのダミー変数を同一視してしまうと，Cantone の多項式に帰着する．

Tamine の木構造は，分岐を文の連接の中に並べてしまうので，木の深さは浅く幅は広がった形になる．

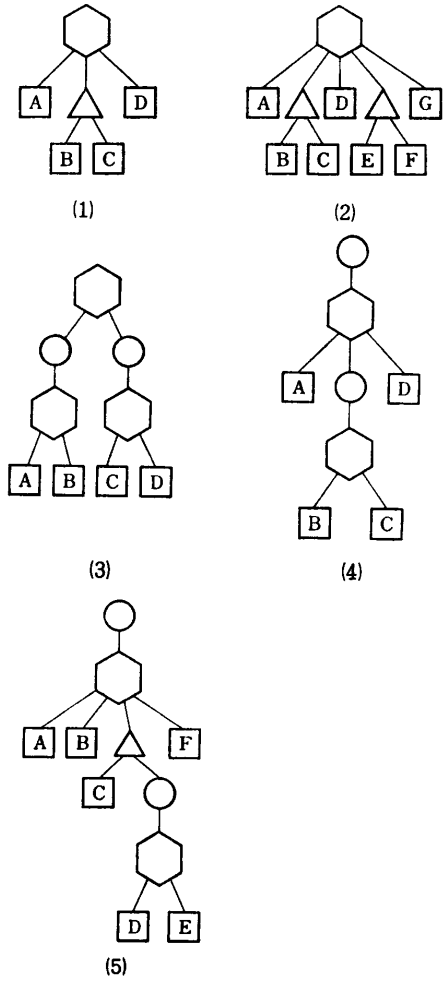


図 3 Wingerter の記法で図1の例を書いた結果

Fig. 3 Same examples as in Fig. 1 in Wingerter's notation.

木構造を，より構文が明確になるように記法を変える提案を Wingerter[6]が示している．正方形の節点で基本文を表し，六角形の節点で文の連接，三角形の節点で分岐，円形の節点でループと，構文によって節点の形を変えて表す．分岐やループの中味は，ネストを一段深くして，プログラムの構造がより見やすくなるようにする．前に用いた同じ例題を，Wingerter の記法にしたものを**図3**に示す．

Wingerter の方法は，特性多項式自体は Tamine のものと同じである．どちらの方法でも，和や積についての式の可約条件をはずせば，さきにあげた2分岐の中味がそれぞれループとなる(1)は $a+a$ となり，ループの中味が2分岐となる(2)は $(1+1)a$ となって，別々に特徴づけできる．このことを利用すれば，よりよい複雑度の尺度を導くことができる．また Wingerter の木構造を用いるなら，むしろ Jackson の構造図をそのまま利用するほうが一般性がある．さらに3基本構文のほかに再帰的な手続に呼出しを加えたものを対象としたい．以上の諸点について筆者が拡張した複雑度の特徴づけについて，以下の章で述べていく．

3. 再帰的な Jackson 構造図

構造化されたプログラムやデータ構造は，基本要素(代入文や入出力文などの基本文，あるいはスカラデータなどの基本データ）の上に，三つの基本構造を組合せて構成できる．Jackson は長方形の節点を木構造状に配し，この3基本構造を表現する記法を提唱している．これが Jackson の構造図で，**図4**のようにまとめることができる．

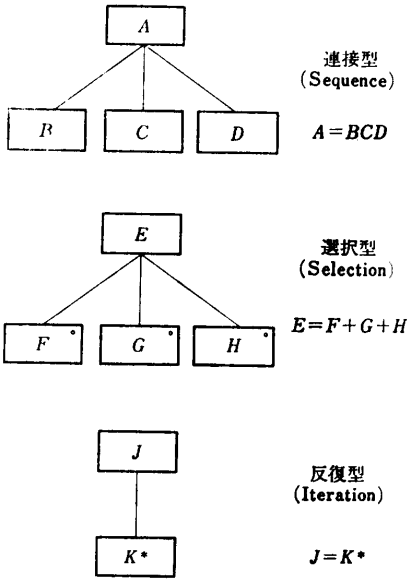


図 4 Jackson の3基本構造図

Fig. 4 Jackson's three basic structure diagrams.

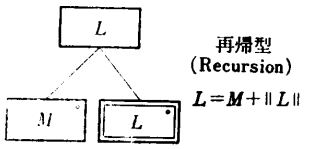


図 5 再帰的な構造図

Fig. 5 Recursive structure diagram.

連接構造は，通常の親節点に複数の子節点がついている形，分岐構造を子節点の右肩に小さい丸印をつけた形，ループ構造は子節点の右肩に小さい星印をつけた形である．木構造の中で，子節点をもたない末端節点が基本要素に相当する．木構造全体は，これらの基本要素上の正規表現の形で表現することもできる．すなわち，連接構造は記号列の積（並べて書くこと）にあたり，分岐構造は記号列の和（プラス記号で結ぶこと）にあたり，ループ構造は Kleene のスター（かっこでくくって右肩に星印をつけること）にあたる．

Jackson の構造図の表現能力は，正規文法の水準であり，一般の文脈自由文法には及ばない．そこで再帰的な手続き呼出しや，ポインタでリスト要素をつないだリスト構造を扱うには，これだけでは不十分である*．そこで Jackson の構造図に，再帰構造を導入する．**図5**のように2重線の正方形とし，その中に書く識別子でどの部分の再帰構造かを示す．2重線の正方形は，埋込み構造を暗示するよう選んだ．再帰節点の代りに，同じ識別子をもつ節点から下の構造を埋込んでやったものが，再帰展開形となる．一般には無限に埋込みがつづく．

代表的な再帰構造は次の二つである．

(1) $F(x)\equiv$ **if** $x<0$ **then** $f:=f_0$
else $\{f_1:=F(x-1);$
$f:=f_0*f_1\}$

(2) $H(k,x,y,z)\equiv$ **if** $k=1$ **then** $m(k,x,z)$
else $\{H(k-1,x,z,y);$
$m(k,x,z);$
$H(k-1,y,x,z)\}$

はじめの例は階乗計算，もう一つはハノイの塔の問題にあたる．それぞれを再帰的な構造図にすると，**図6**のようになる．

* Jackson の JSP 法では，バックトラッキングを記述できるようにしてこの点の不備を補っている．

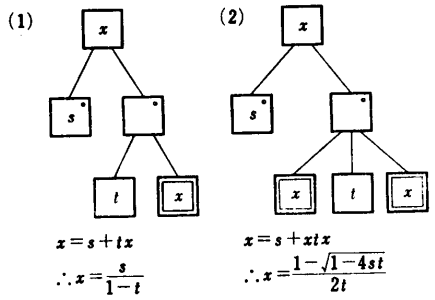


図 6 二つの代表的な再帰構造例
Fig. 6 Two typical recursive structure diagrams.

再帰節点を含む構造図は，正規表現では記述できないが，Salomaa[7]による擬似正規表現の記法を用いることができる．これは，$A\rightarrow xAy|z$ という生成規則に対して，$(xAy+z)^{+A}$ を擬似正規表現として与えるもので，非終端記号Aが消えるまで，Aの位置にxAyかzかを埋込んでいくことを示す．この記号を用いるとKleeneのスターは $x^*\equiv(xA+\varepsilon)^{+A}$ と書くことができるので，擬似正規表現は正規表現を含む．さきの図6の構造図に対応する擬似正規表現は，次のようになる．

(1) $(s+tX)^{+X}$

(2) $(s+XtX)^{+X}$

4. 新しい複雑度の尺度

これまでの章で準備が整ったので，新しい複雑度の尺度を導入する．複雑度を評価すべきプログラムは構造化されていることを前提とする．

まず，プログラムを再帰構造の記法を含めて拡張したJacksonの構造図で記述する．末端節点には，a, b, c など小文字の終端識別子をわりあてる．再帰節点を含まない部分は，木の末端の節点から始めて，根の節点にむかって，正規表現を作っていく．2章であげた諸例について正規表現を対応づけしたものを**図7**に示す．

再帰節点には，A, B, C など大文字の非終端識別子をわりあてる．根の節点に近いほうがふつうの長方形，末端に近いほうが2重の長方形になる．このとき，さきの図6について，次のように計算する．

まず(1)については，全体の値を未知数 x とおいて，再帰呼出しのところも同じxをあてはめる．このとき等式 $x=s+tx$ が成立する．これを移項して整理すると，$x=\frac{s}{1-t}$ が得られる．同様に(2)についても，全体の値を未知数xとおき，等式 $x=s+xtx$ を整理して，$x=\frac{1\pm\sqrt{1-4st}}{2t}$ を得る．

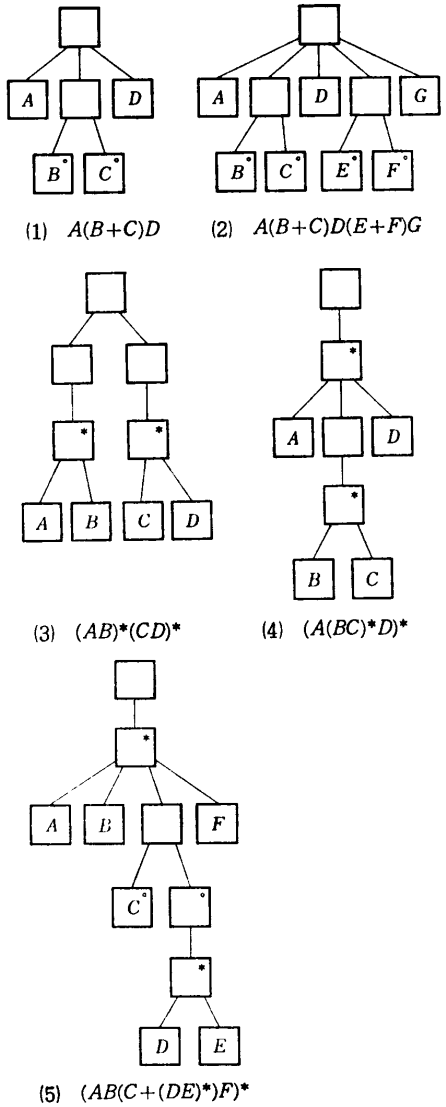
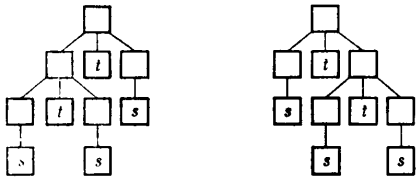


図 7 提案している記法で図1の例を書いた結果
Fig. 7 Same examples as in Fig. 1 in the proposed notation.

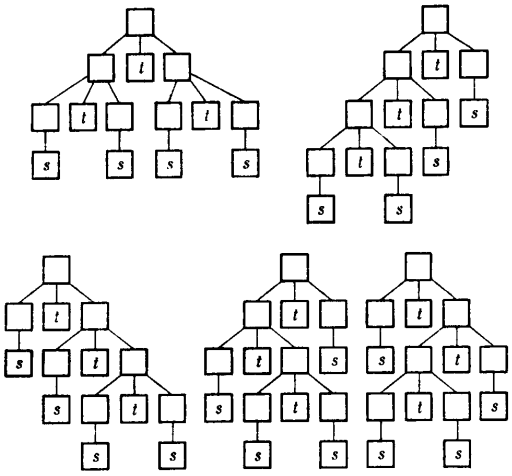
どちらのxもこのままでは多項式ではないが，Knuth[8]などにみられる母関数の考えかたを用いて級数展開すれば，多項式とみなすことができる．

はじめの例は，$\frac{1}{1-t}=1+t+t^2+t^3+\cdots$ を用いて，$x=s+ts+t^2s+t^3s+\cdots$ となる．これは，プログラムの実行径路の可能性を表現している．ただし式の形からは，sをt^nの左におくか右におくかは区別できず，プログラムの意味論を調べて定めなければならない．

もう一つの例は，次の式を利用する．



(a) 再帰呼出し2回の場合の2通りのパターン
(2 different patterns for 2 recursive calls)



(b) 再帰呼出し3回の場合の5通りのパターン
(5 different patterns for 3 recursive calls)

図 8 再帰呼出し数を固定したときの異なるパターン例
Fig. 8 Different pattern examples for fixed recursive calls.

$$\sqrt{1-4u}=1-2u-2u^2 -\frac{2}{3}\binom{4}{2}u^3-\frac{2}{4}\binom{6}{3}u^4-\frac{2}{5}\binom{8}{4}u^5-\cdots$$
$$=1-2u-2u^2-4u^3-10u^4-28u^5\cdots$$

ここで $u=st$ とおき，複号は負のほうをとると，次式が得られる．

$$x=s+s^2t+2s^3t^2+5s^4t^3+14s^5t^4+\cdots$$

ここでも，s と t の順序は元のプログラムの情報を反映させて s と t が交互に現れるようにすると，次のようになる．

$$x=s+sts+2ststs+5stststs+14ststststs+\cdots$$

この係数の1, 2, 5などは Catalan 数で，このパターンの可能な組合せの数を表し，たとえば *ststs* は図8(a)に示す2通りのパターン，*stststs* は図8(b)に示す5通りのパターンにそれぞれ対応している．

実は図6の構造図をハノイの塔のプログラムとみると，図8に示したパターンの多くは現実には生じない．ハノイの塔では，再帰呼出しが左右対称に広がる形のみ生じる．しかし図6の構造図にはその情報は含まれていない．左側と右側の再帰呼出しの回数が等しいという制約は，この図では記述できない．

級数展開して多項式にするよりも，閉じた式のままで特性式として用いることが，複雑度の特徴づけには便利である．こうして，再帰節点を含めて，構造図が一つの特性式で表現できる．

この特性式について，終端記号 *a*, *b*, *c* 等にすべて1を与え，Kleene のスター（星印）がかぶさっている部分にダミー変数 *c* を与えれば，このまま Cantone の特性多項式が得られる．また星印のネスト水準ごとにダミー変数を変えれば，Tamine の特性多項式になる．図6に示した例はすべて図1の Cantone の例と同じであり，再帰構造を含んでいない．

終端記号 *a*, *b*, *c* 等にすべて1を与える代りに，プログラム上で対応する個所の複雑度に関する重みを与えることがより現実的である．その重みの選びかたとしては，Halstead 尺度のような，式や文の複雑度をオペレータおよびオペランドの種類数や出現総数で表す方法も使える．Halstead 尺度は，構造のもつ複雑度を表現する能力には乏しいけれども，式や文の平板な羅列の複雑度を表現するには適しているからである．

Jackson 構造図を拡張したものを，複雑度尺度のモデルに用いることの利点として，この手法がプログラムの複雑さだけでなく，データの複雑さの尺度にも使えることをとりあげることができる．連接，選択，反復の3基本構造はデータの構造にもあてはまり，再帰構造も，ポインタによるリスト構造のデータにあてはまる．したがって，データの複雑さも，正規表現をもとにした特性式で記述することによって，特徴づけできる．

さらに JSD 法[9]にみられるように，プログラム（動的なもの）とデータ（静的なもの）を合わせて一つの構造図に記述することも可能である．そのような形でのプログラムの複雑さの特徴づけについては，今後の課題として検討していく予定である．

再帰的な構造をもつ場合には，3章でふれた Salomaa の擬似正規表現を用いる方法と，本章で述べた方法で特性式を求める方法とがある．特性式を用いた場合，級数展開によって具体的な記号列が得られる利点がある．擬似正規表現には，ネストの水準など構造的な複雑度がみやすいという利点がある．双方の利点を生かすには，擬似正規表現から特性式を得るほうがその逆よりも容易であることを考えて，記述は擬似正規表現をとり，終端記号 *a*, *b*, *c* 等に重みづけする前に特性式を求めるというやりかたがよいであろう．

5. むすび

本稿での提案は，Jackson の構造図に再帰構造を導入することと，構造化されているプログラムの McCabe 風の複雑度尺度として正規表現あるいは擬似正規表現をもとにした特性式を用いること，の2点である．本稿ではこの2点を結びつけて論じたが，概念は独立であり，一方だけを生かして利用することにも十分意義があると考えている．

謝辞 本研究を支援していただいた，山梨大学工学部計算機科学科の教職員および大学院生の皆様に感謝いたします．

参考文献

1) McCabe, T. J.: A Complexity Measure, *IEEE Tr. Softw. Eng.*, Vol. 2, No. 4, pp. 308-320 (1976).
2) Halstead, M. H.: *Elements of Software Science*, Elsevier North Holland, Amsterdam (1977).
3) Jackson, M. A.: *Principles of Program Design*, Academic Press, New York (1975).
4) Cantone, G., Cimitile, A. and Sansone, L.: Complexity in Program Schemes—The Characteristic Polynomial, *Sigplan Notices*, Vol. 18, No. 3, pp. 22-31 (1983).
5) Tamine, J. J.: On the Use of Tree-like Structures to Simplify Measure of Complexity, *Sigplan Notices*, Vol. 18, No. 9, pp. 62-69 (1983).
6) Wingerter, R.: A Note on Determining the Complexity of Algorithms, *Sigplan Notices*, Vol. 19, No. 3, pp. 73-78 (1984).
7) Salomaa, A.: *Fomal Languages*, Academic Press, New York (1973).
8) Knuth, D. E.: *The Art of Computer Programming I, Fundamental Algorithms*, Addison-Wesley, New York (1968, 1973).
9) Jackson, M. A.: *System Development*, Prentice-Hall, Englewood Cliffs (1983).

(昭和59年10月29日受付)
(昭和60年 2 月21日採録)