プログラム構造からみた変異の分類法†

佐藤 匡正††

本論文では，プログラム変異を定式的に分類する方法を提案する．プログラム変異とは，一つの問題に対して作成された同値プログラムのうち，プログラムの基本要素は同じであるがその連なり方の規定に違いのあるものをいう．本方法では，プログラム記法 HCP 図法で書かれたプログラムを機構と構造という二つの成分に分けて構造の方を代数式によって表現した式を得る．得られた式を変形して式同士の関係を求め，これによって変異を分類する．ここで用いる代数は積に関する可換律の成り立たない環を基本として拡張したものである．この方法を，既に発表されている二つの事例について適用し，方法論としての妥当性を確認している．

1. 序　論

一つの問題に対して複数個の相異なるプログラムが存在することがある．コードは異なるが，想定された入力に対して得られた出力が等しいプログラムは，一般に，同値プログラムと呼ばれる．この同値プログラムの存在が最近，著者権に関係して問題になってきている．プログラムの著作権の保護の範囲が広まり，従来のコード・レベルから「構造」へと拡大してきている．米国では1987年に Whelan 事件で「構造」は保護の対象になりうるという判決が出された．この判決では，流れ図で表現された「構造」が同じであればコードは異なっていても著作権法上の複製にあたる，という判断を示したものとされている[1]．この判断は同値プログラムに分類の基準を与えたものと捉えることができるが，この基準は直接的というよりは間接的である．構造が同じか否かということが，技術的な判断基準によって直接的に判断されているのではなく，コードが作られるに際して，その構造に接近できるか否かという基準によって判断されている．今後は訴訟の内容がさらに高度になり，このような間接的な判断では不十分で，「構造」が同じか否かを直接判断する必要が生じるものと予想される．このような要求に応えるための一つとして，ここでは同値プログラムの一種であるプログラム変異の分類法について論ずることにする．

同値プログラムについては，流れ図プログラムに関して，同値およびより強い同値性である同型を判断するという考えがよく知られている[2]．しかし，この考えでは，定義した同値の性質上から，正当性や停止性の証明と同様に，プログラムの意味を数学的に取り扱わねばならない．ところが，この取扱いは複雑であり，実務プログラムに適用するという点から見ると，十分な方法とはいえない．実用上からは定式的に扱えて，簡便に同値プログラムか否かが判断できれば効果的である．

すべての同値プログラムに対しての適用は無理であるが，限定された範囲なら簡便に扱えるというのであれば実用の価値がある．そこで，対象とする同値プログラムを限定する．同値プログラムとは，動作の結果が同一でプログラムとしての記述が異なっているものといえる．この記述の違いには，①プログラムの基本要素そのものに違いのあるもの，②基本要素は互いに同じものであるがその連なり方の規定に違いのあるもの，の二種類が考えられる．ここで，基本要素とは JIS 用語 (X 0128) であり，データの加工や変換などを担っているもので，流れ図でいえば処理ボックスに当たる．①は一般的な場合であるが，②は同値プログラムを限定したもので，プログラム理論では同型と呼んでいる[2]．しかし，ここではこれに，特に具体的なイメージを与えたい．そこで，生物学において同種の生物にみられる形態上の違いを変異と呼ぶことに倣って，これを改めてプログラム変異と呼ぶことにする．

この変異では基本要素が同じであるので，連なり方に着目する必要がある．一般に，連なり方に着目したプログラムは流れ図プログラムといわれる．流れ図プログラムを定式的に表現するには少々工夫が要る．単純に考えれば，構造化された，いわゆる GOTO-less であればよく知られているように，原則として連接，選択，繰返しの組合せによって規定されているので，基本要素を変数で表し，連接，選択の二つの基本操作を環のような代数の積と和の演算子に対応づければ，代数式として表現できる．ところが，このように

† Program Variation Classification by Program Structure by TADAMASA SATOU (Department of Information Science, Yokohama Souei Junior College).

†† 横浜創英短期大学情報処理学科

して得られた流れ図プログラムに対する代数式では式変形はうまくゆかない．プログラム上の意味が式変形によって変わってしまうためと考えられる．ここでいうプログラム上の意味とは流れの制御に関するものである．代数式には，このような意味は含められないから，何らかの手段で意味を含んでいる成分を取り除かなければ式変形は行えない．

このための考え方として，プログラム理論では，流れ図プログラムを「プログラムスキーマ（以下，単にスキーマという）」と「解釈」に分けて意味の分離を図っている．流れ図プログラムの族としてスキーマを考え，これに対して変数の領域，関数，述語からなる「解釈」が与えられて流れ図プログラムになるものとしている．こうするとスキーマには意味は含まれない．しかし，このスキーマは流れの制御の大枠を表しているだけで，プログラム処理を反映しているわけではないし，分離しきれない流れの制御に関する意味も存在する．例えば，本来は流れの制御の成分であるスイッチやフラグなど，が取り除けず，スキーマの一部を成すことになる．こうすると，実際には変異であっても，式表現からは変異とは判断できないことになる．流れの制御成分も取り除けるように工夫できれば具合がよい．

この一つとして，流れ図プログラムにおいてスイッチやフラグを含む流れの制御成分を取り除く手法が提案された[3]．これはプログラムの成分を構造と機構に分けるという考えである．機構は流れを制御する成分であり，選択や繰返しの部分だけでなく，これに関連している基本要素をも含んだ一つの機構を形作る部分である．構造はプログラムが行うべき本来の処理であり，機構によって生成された基本要素の列から成る．つまり，流れ図プログラムは，基本要素の列を要素とする集合が構造として定義され，この集合の要素が機構によって部分化されたものと解釈される．この機構によって部分化された集合がプログラムとしての動作を与えている．

この構造には，プログラムとして実施すべき本来の基本要素の連なりのみが含まれ，流れの制御に関する意味は含まれていないので，代数式として扱えることになる．また，この構造の概念からプログラム変異をみれば，共通部分のある構造という見方で定義できることになる．

以下，流れ図に示されている変異プログラムを構造からみて定式的に分類する方法を提案する．次の第2章では，代数式の変形をするための準備として，代数式を適用する方法や用語の定義を与える．第3章では，二つの例題に本手法を適用して方法論としての妥当性を確かめる．

2. 準　　備

2.1 代数の適用法—機構と構造[3]

HCP 図法での基本要素の連なり方の規定は，原則として，いわゆる GOTO-less である．階層化され，階層内では連接，選択，繰返しから成っている．このうち繰返しは，基本形であれば連接と選択で展開できるので，連接と選択が基本である．そこで，連接を積，選択を和に対応させれば，基本要素に対する和と積の代数式によって HCP 図のプログラムが表現できる．ところが，式の演算で積の分配律を適用すると，プログラムとしての意味に不都合を生ずることがある．

いま，積を・，和を＋で表すものとして代数式 $a\cdot(b+c)$ を想定する．この式に分配律を適用して式変形を行うと $(a\cdot b+a\cdot c)$ となる．この式変形に関してプログラム上の意味を考えると，変形後の $(a\cdot b+a\cdot c)$ がプログラムとしての意味をなさないことがある．変形前の $(b+c)$ において，基本要素 a によって，b,c のどちらかが選択されるための条件が与えられている場合である．この場合，（ ）で示される選択とこれに条件を与えている a に関してプログラムとして意味をもつ順序は，a の次に（ ）が続く $a\cdot(b+c)$ である．a と（ ）の順序が逆転した $(a\cdot b+a\cdot c)$ では意味をなさない．これは代数式の変形ができないことを意味する．

この不都合は，基本要素と選択が「条件を介して結合しているため」と考えられる．この結合を切るために，プログラムの処理上の意味を「機構」と「構造」の二つに分ける．機構には，各々の選択や繰返しにおいて，それに含まれている選択するための仕掛けが形作られている．この仕掛けによって実行の流れが制御され，構造に対して一定の制約が与えられる．一方，構造には，そのプログラムが本来果たすべき役割が処理ステップとして組み立てられたものである．

この機構の内容を示す例として，スイッチやフラグを考える．機構に含まれるものは，オン/オフを判断している条件選択部分のほかに，基本要素である初期設定，オン/オフ操作部分である．一方，構造はプログラムからこの機構を除いた部分であり，そのプログラムの入力と出力を結び付ける基本要素の連なりである．

基本的には，プログラムから機構を取り除くことによって構造が取り出される．ただし，プログラムには機構と構造が混在していて互いに共用している部分がある．この部分が構造として残される．

次の手順で構造が取り出される．

〈S1〉 繰返しおよび選択における条件付の記述が削除される．

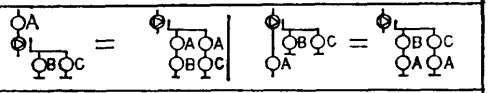
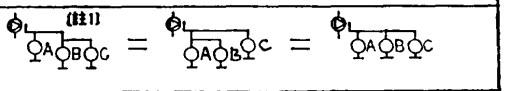
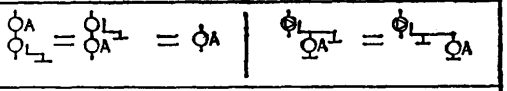
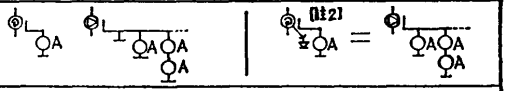
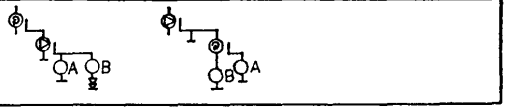
〈S2〉 条件付記述からデータ名が得られる．このうち条件記述以外では，参照されていないデータ名を機構用データ名という．

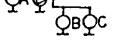
〈S3〉 機構用データ名に対して値を与えているか，参照している操作をもつ基本要素が削除される．

このように機構と構造に分割することによって，上の積の分配律における不都合は機構の方に閉じ込められるので，構造には代数が適用できることになる．

この代数では，機構についての言及ができないので，プログラム全体の形式化はできない．したがって，これまでに「プログラムの理論化」として行われてきたようなプログラムの正当性について論ずることはできないが，変異を分類する場合のように，プログラムを包括的に議論するには有効である．

表 1 S 代数の基本式と HCP 図法の対応
Table 1 Correspondence between S-algebra fundamental formula and HCP notations.

番号	基本式 (S代数)	HCP図法
1	A(B+C)=(AB+AC) (B+C)A=(BA+CA)	
2	(A+(B+C))=((A+ B) +C)=(A+B +C)	
3	(A+B)=(B+A) (A+A)=(A)=A	
4	Aε=εA=A (A+ε)=(ε+A)	
5	$(A)^{*}=(\varepsilon+A+AA+\cdots)$ $(A)^{+}=(A+AA+\cdots)$	
6	$((A+\underline{B})^{*}=((A)^{*}B+\varepsilon)$	

註 1) 選択ネストの略記法である．正式には，  である．

註 2) 出口付繰返しの略記法である．正式には，  である．

$(A(\varepsilon+\underline{\varepsilon}))^{*}=(A)^{*}A=A(A)^{*}=(A)^{+}$

2.2 S 代 数[3]

S代数は，プログラムの記法である HCP 図法[8)~10)]の制御構造を表すための代数である．代数系としては，積に関する可換律が成り立たない環を基本として HCP 図法の階層飛び越し操作に対応させた拡張演算子；入口子，出口子をもつ．HCP 図法の一つの基本要素を変数または定数 (ε, ϕ) で表し，この連なり方の規定を積，和で表す．この代数系を S 代数と呼ぶ．HCP 図法との対応を**表 1** に整理する．

なお，以下では，積の演算子を省略するために代数式の変数を大小英1文字で表すことにする．ただし，大小の使い分けは，単に式の見やすさを狙いとしたもので，特別な意味をもつものではない．

2.3 変 異

オートマトンでは有限個の記号の集合をアルファベット，アルファベットから有限個の記号を重複を許して選び出し，一列に並べたものを語，語を要素とする集合を言語と言っている．言語は，よく知られているように，正規表現の代数式で規定される．

そこで，これに倣って代数式と集合の関係によってプログラム構造を扱うことにする．アルファベットは基本要素の集合に，語は基本要素の連なりに，言語はプログラム構造にそれぞれ対応させることができる．二つのアルファベット $\{y_i\}$, $\{z_i\}$ $(i=1, 2, \cdots, n)$ に対して，言語 F を規定する代数式 $f(y_1, y_2, \cdots, y_n)$ と言語 G を規定する代数式 $g(z_1, z_2, \cdots, z_n)$ が考えられる．

これらを用いて変異を次のように定義する．

基本要素の集合 $\{y_i\}$ に対して，プログラム構造 F を規定する代数式 f と $\{z_i\}$ に対する G と g を想定する．

$F \cap G \neq \phi$ かつ $\{y_i\}=\{z_i\}$ であるとき，f と g は「変異」であるという．

1) $F=G$ のとき，f と g は一致変異という．
また，f は g に「一致する」といい，$f \doteqdot g$ と書く．

2) $F \subseteq G$ のとき，f と g は包含変異という．
また，g は f を「包含する」といい，$f \subseteq g$ または $g \supseteq f$ と書く．

3) $F \subseteq S$, $G \subseteq S$ かつ $S \subseteq F \cup G$ なる集合 S が存在するとき，f と g は交差変異という．

包含変異および一致変異による式変形を変異変形という．なお，変異プログラムの動作が一致する理由については付録1に示す．

2.4 変異と同値プログラムの関係

（1） 一致変異

二つのプログラム構造が互いに一致しており，これらの構造に対して適当な機構を与えることによって構造の全部もしくは一部を動作の範囲とする同値プログラムが得られる．

（2） 包含変異

二つのプログラム構造 F, G において一方の，例えば G が F を包含しており，これらの構造に対して適当な機構を与えることによって F の全部もしくは一部を動作の範囲とする同値プログラムが得られる．

（3） 交差変異

二つのプログラム構造 F, G が互いに交差しており，これらの構造に対して適当な機構を与えることによって交差している共通部分の全部もしくは一部を動作の範囲とする同値プログラムが得られる．

3. 分　　析

実際の変異プログラムがどのような関係にあるかを次の手順によって調べる．

[1] HCP 図法で書かれたプログラムを構造と機構に分ける．

[2] 上の構造の方のみをS代数式として表す．

[3] 式変形を行い，構造の関係を式の関係から観察する．

[4] 式の関係から，どの変異に分類できるかを観察する．

[5] 結果について考察する．

事例としては，周知の二つの問題を取り上げる．一つは文字列の切り出しに関するもの[4]で，他は倉庫管理業務に関するもの[6]である．なお，ここで用いる定理は付録2に示す．

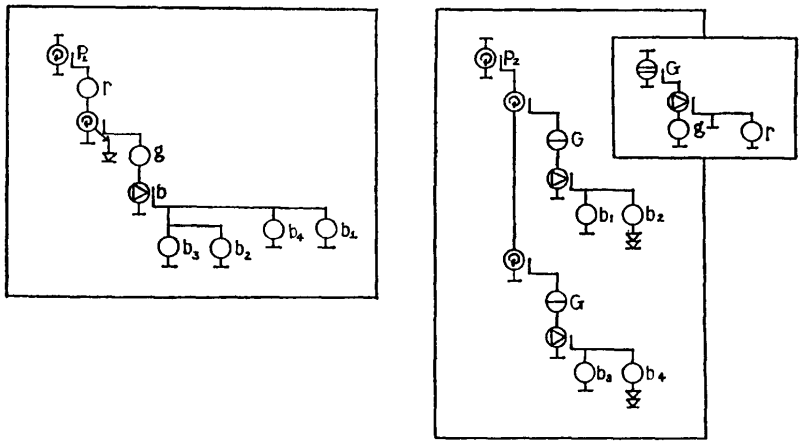
3.1 文字列の切り出し問題

文献4)には，ファイルに格納されている文字列を切り出す問題を例として，機能モデルに従って，系統的に作成されたプログラムと直観的に作成されたプログラムが示されている．この事例を取り上げる．

[問題]

文字列から項目名を切り出す．文字種別は空白文字，英字およびデリミタである．項目名は英字で始まり，デリミタ［；］で区切られている．項目名に先立つ空白文字は読み飛ばす．項目中には空白文字は現れないものとする．なお，文字列は固定長レコード形式のファイルに格納されており誤りはないものとする．また，項目名は1レコードに閉じているものとする．

文献2)の二つのプログラムから「構造」を得る．得られた構造を**図1**（A），（B）に示す．変数の集合は一致する．変数によって表される基本要素の意味は次



（A） 構造1　　（B） 構造2

変　数	意　味
G	文字を取り出す
g	一文字を取り出す
r	レコードを得る
b	項目名を切り出す
b_1	空白文字を読み飛ばす
b_2	先頭の文字を始末する
b_3	途中の文字を始末する
b_4	項目名を始末する

図1 文字列から項目名を切り出すための二つの構造
Fig. 1 Two structures for delimiting item names from input strings.

のとおりである[7].

G：文字を取り出す，g：一文字を取り出す，r：レコードを得る，b：項目名を切り出す，b_1：空白を読み飛ばす，b_2：先頭の文字を始末する，b_3：途中の文字を始末する，b_4：項目名を始末する．

（1） 構造の関係

プログラム 1，2 の構造を P_1，P_2 で表すと，

$P_1 \doteqdot (r(g(b_1+b_2+b_3+b_4))^+)^*$

$P_2 \doteqdot ((G(b_1+\underline{\underline{b_2}}))^+(G(b_3+\underline{\underline{b_4}}))^+)^*$，$G \doteqdot (r+\varepsilon)g$

定理 8 の系 1 において，$A \doteqdot r$，$x \doteqdot g(b_1+b_2+b_3+b_4)$ とすると，

$P_1 \subseteq ((r+\varepsilon)g(b_1+b_2+b_3+b_4))^*$

一方，

$P_2 \doteqdot ((G(b_1+\underline{\underline{b_2}}))^+(G(b_3+\underline{\underline{b_4}}))^+)^*$

出口の定義から，

$\doteqdot ((Gb_1)^*Gb_2(Gb_3)^*Gb_4)^*$

定理 7 において，$x \doteqdot Gb_1$，$A \doteqdot Gb_2$，$y \doteqdot Gb_3$，$B \doteqdot Gb_4$ とすると，

$\subseteq (Gb_1+Gb_2+Gb_3+Gb_4)^*$

$\doteqdot (G(b_1+b_2+b_3+b_4))^*$

G を展開して，

$\doteqdot ((r+\varepsilon)g(b_1+b_2+b_3+b_4))^*$

以上により，P_1，$P_2 \subseteq K$

ここで，$K \doteqdot ((r+\varepsilon)gb)^*$，$b \doteqdot (b_1+b_2+b_3+b_4)$

K は P_1, P_2 を包含し，かつ明らかに $P_1 \doteqdot P_2$ が成り立つことがある．よって P_1 と P_2 は交差変異である．

（2） 考　　察

K における選択機構 $(r+\varepsilon)$ および繰返し機構の組合せで，K を部分化して P_1 に一致させることができる．例えば，レコード入力表示フラグを用いることによって $K \doteqdot ((r+\varepsilon)gb)^*$ の構造に制約を与え，$P_1 \doteqdot (r(gb)^+)^*$ に一致させた例を図 2（B）に示す．

K と P_2 についても同様である．プログラムとして示すことは避けるが，選択機構 $(b_1+b_2+b_3+b_4)$ と繰返し機構の組合せによって K を部分化し，

$P_2 \doteqdot ((Gb_1)^*Gb_2(Gb_3)^*Gb_4)^*$

に一致させることができる．

設計上からみた P_1 と P_2 の構造上の違いは，二つある繰返し式を共通化して一つにするか否かにある．上の観察より明らかなように，P_1 では，P_2 における二つの式

$(G(b_1+\underline{\underline{b_2}}))^*$ および $(G(b_3+\underline{\underline{b_4}}))^*$

が両者に共通な一つの式 $(G(b_3+b_4+b_1+b_2))^*$ に統合されている．このような共通化によって，P_2 では分散されている式 G［レコードから文字を取り出す］が P_1 では一つで済むようになり，レコードと文字とを同時に扱っている構造の式 $(r(gb)^+)^*$ の中に展開されている．

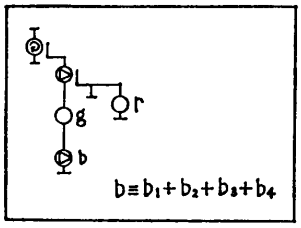
実際のプログラミングでは，このような共通化がよく見掛けられる．この共通化により処理能率の向上や修正箇所の局所化などの効果が得られるが，問題固有の骨組みがプログラム上の構造には歪められて反映される．この歪みがわかりやすさや変更しやすさなどに影響を及ぼすことが少なくない．

3.2 Bergland の倉庫管理の問題

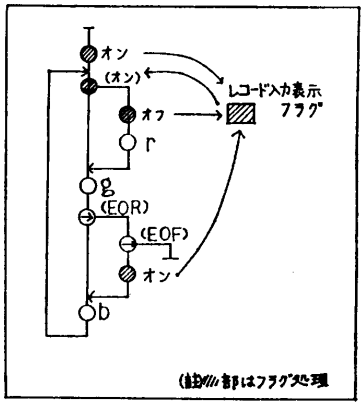
文献 6) には，倉庫管理業務を例題として，複数のプログラム設計方法を適用して作成されたプログラムが示されている．ここでは，これらのプログラムのうち，機能分割設計法によるプログラム，およびジャクソン設計法によるプログラムについて分析を行う．

（1） 構造の関係

上の二つのプログラムからフラグやスイッチなどからなる機構を取り除いてしまうと，図 3 のような構造 F, J が得られる．ここで，F は機能分割法によるプログラムの構造であり，J はデータ構造法による構造である．原著では両プログラムの処理記述文の言い回しは一致してはいないが，問題記述を考え合わせれば同一の基本要素とみなせる．変数の対応を図 3 中の表に示すようにする．

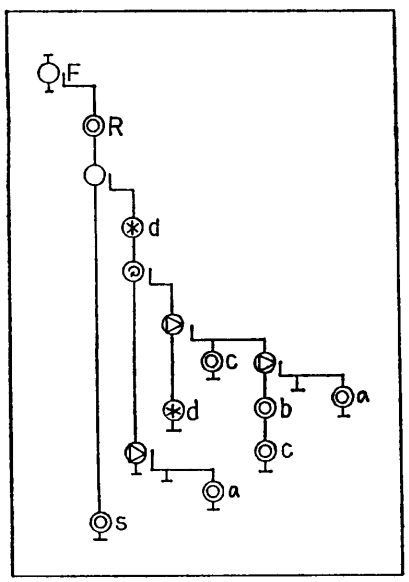


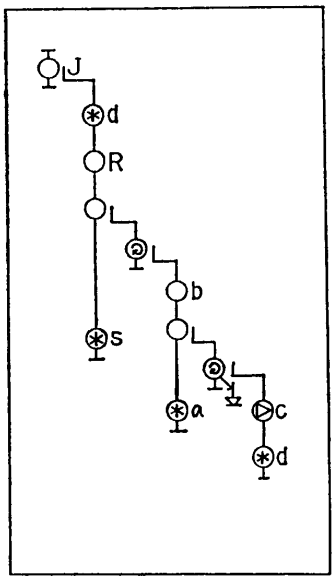
（A） $((r+\varepsilon)gb)^*$ の構造



（B） (A)構造にフラグで制約をつけて $(r(gb^+))^*$ と同値にしたプログラム［Cp 図法］[11]

図 2 同一連なりをうるための機構例
Fig. 2 A mechanism example for the same sequence.





記号	機能分割法による解法の構造 F	データ構造（ジャクソン）法による解法の構造 J
a	PROCESS END OF LAST /PREVIOUS GROUP	Write net-chg
b	PROCESS START OF NEW GROUP	item-grps:=item-grps+1/critem :=next item
c	PROCESS CARD	CTR BDY
d	read (STF)	read STF
R	PRODUCE HEADING	write heading
S	PRODUCE SUMMARY	write item grps

図 3 Bergland の例題と構造
Fig. 3 Bergland example and its structures.

$$F \doteqdot R(d(((a+\varepsilon)bc+c)d)^*(a+\varepsilon))S$$
$$\doteqdot Rd((ab+b+\varepsilon)cd)^*(a+\varepsilon)S$$
$$J \doteqdot dR(b(cd)^+a)^*S$$

F と J の関係を分析する．

定理 8 において，$A \doteqdot b$，$x \doteqdot cd$，$B \doteqdot a$ とすると，

$$(b(cd)^+a)^* \subseteq (\varepsilon+((b+ab+\varepsilon)cd)^*a)$$
$$\subseteq ((b+ab+\varepsilon)cd)^*(a+\varepsilon) \quad [\leftarrow \text{定理 2}] \qquad (1)$$

したがって，

$$J \doteqdot dR(b(cd)^+a)^*S \subseteq dR((b+ab+\varepsilon)cd)^*(a+\varepsilon)S \qquad (2)$$

である．

ところで，プログラムの意味上からは $dR \doteqdot Rd$ である．「d：ファイルを読む」であり，「R：見出し行を書く」とは互いに独立であるので，両者を入れ換えてもプログラムとしての意味は変わらない．そこで，式(2)の右辺は F と等しくなる．よって，

$$J \subseteq F$$

また，明らかに $J \doteqdot F$ が成り立つことがある．つまり，J と F は包含変異である．

（2） 考　　察[5]

機構の働きによって，F は J と同値のプログラムとなる．J には，スイッチ SW_1，SW_2 が選択機構 $(a+\varepsilon)$，$(bc+c)$ に使用され，繰返し機構と組み合わされて基本要素の連なりが制御される．最初の繰返しサイクルでは，最初の $(a+\varepsilon)$ において，SW_1 によって ε が選択される．次の $(bc+c)$ では，SW_2 によって bc が選択される．つまり，最初のサイクルでは，bcd という連なりが生ずる．次のサイクルでは，各々で ε と c が選択され，cd が生ずる．このサイクルが繰り返され，あるタイミング（具体的には，カード・グループの変化）で SW_1 がオンになり，$(a+\varepsilon)$ で a が選択され，$abcd$ となる．そして，再度 cd が繰り返される．最後には，二番目の $(a+\varepsilon)$ で a が選択される．また，繰返しが ε であったときは ε が選択され，繰返しは終了する．このような機構の働きで，両者は同値となる．

F と J の構造上の違いは，前の例題と同様に，繰返し式の共通化の度合にある．J では，カードとカード・グループという入力データ構造を反映した繰返し式の入れ子構造をなしているのに対して，F では，式(1)の左辺 $(b(cd)^+a)^*$ において，内側の $(cd)^+$ と外側の $(b(cd)^+a)^*$ を一つの繰返し式として共通化されたものとみることができる．このような共通化は，前の例題と同様に，構造自体は単純化されるが，制御のための機構が複雑になり，了解性や耐変更性に問題が生ずる．

3.3 議　　論

（1） プログラム設計における変異の意味

包含変異は構造の「拡張効果」をもつ．プログラム設計において，このような構造の拡張はよく見受けられるが，構造内部の構成要素間での共通化が一つの要因と考えられる．この共通化の狙いは汎用化と効率化にあると推定される．共通化を図れば，構造に含まれ

る基本要素の連なりは多くなる．問題の変形に対しても，機構に手を入れるだけで，構造には手をいれずに済む．例えば，付録1の例を考えて，$(AB+BA)\subseteq(A+B)^2$ において，何らかの要求により $(AB+BA)$ を $(AB+BA+BB)$ とするような変更が必要になったとする．左辺の $(AB+BA)$ という構造では，変更に対処するために構造自体に新たに BB を加える必要がある．しかし，$(AB+BA)\subseteq(AB+BA+BB)\subseteq(A+B)^2$ であるから，AB と BA が共通化されている右辺の構造 $(A+B)^2$ の変更は，AA を ϕ にするよう機構に手を入れるだけで済む．したがって，修正量，大きさ，実行させるべき処理要素数についての効率はよいといえる．

このような共通化の手法は従来から行われている．この手法が効果的かどうかについては議論の余地がある．上の変更は容易であるようにみえるが，実際のプログラムでは，付図1(D)にあるように，構造と機構が一体化しており，かつ機構が複雑であることも少なくない．機構だけを変更しようとしても，構造と一体化している現状のプログラムでは，変更は容易ではない．よって現状では，構造が汎用的になっているといっても，変更が容易とはいえない．ただし，機構と構造が分けて扱えるようになれば，この汎用化策が生きてくる．

効率化の問題についても同様に，機構と構造が一体化している現状では，了解性と効率化とが相反している．プログラム設計に当たっては，この相反性に配慮する必要がある．

（2） 積の可換律について

S代数では，積の可換律は成り立たないとしている．しかし，3.2節の Bergland の例題の分析では，プログラム処理としての意味を配慮して，例外的に可換律が成り立つとした．この取扱いは，厳密にいえば，変異として取り扱える範囲を越えている．ここでは，常識の範囲として便宜的に取り扱っている．本来は，処理の意味を背景とする変異として，別に議論を進める必要がある．今後の課題とする．

（3） 複製判断基準への応用について

本提案によって，複数のプログラムが複製されたものか否かを直接結論づけることはできないが，複製について一定の判断基準を与える可能性が考えられる．この基準として，例えば，「一致変異と包含変異では複製の可能性があり，包含変異でない交差変異の関係についてはその可能性が薄い」などが考えられる．

（4） 基本要素の記述について

流れ図プログラムの基本要素の記述方法として，暗黙に自然語を用いた厳密でないものを想定している．したがって，基本要素に関して，構造の取り出し方，および同一か否か，無意味か否かの判断等については，情報処理技術者としてのある程度の固有な知識が前提となっている．

4. 結　　論

同値プログラムのうち，プログラム構造の基本要素は等しいが連なり方の規定に違いのあるものを変異と名付け，一致，包含，交差の三つの種類に分類する方法を提案した．

この方法を既に発表されている二つのプログラム変異の事例について適用し，これらが交差，包含の変異であることを示した．

謝辞 本論文を書くにあたってご協力頂いた皆様に感謝いたします．インドネシア国在住の Go Sing Fung さんと御一家の皆さんは本論文を書くための環境を提供して励ましてくださった．NTT インターナショナル社の佐野淑郎さんには文献 1) の入手にお骨折り頂いた．また，本学の久米頼子さんには図表のトレースをして頂いた．ここで皆さんに改めて感謝の気持ちを表します．

参 考 文 献

1) 松田政行：コンピュータ時代の知的所有権，pp. 22-37，(株)ぎょうせい (1988)．
2) Manna, Z.: *Mathematical Theory of Computation*, McGraw-Hill Inc. (1974)；五十嵐訳，プログラムの理論，日本コンピュータ協会 (1975)．
3) 佐藤匡正：HCP図法で記述されたプログラム解法の S 代数による定式化，情報処理学会論文誌，Vol. 27, No. 6, pp. 612-620 (1986)．
4) 佐藤匡正：プログラム処理の多様化要因と処理の標準化，第27回情報処理学会全国大会論文集，pp. 463-464 (1983)．
5) 佐藤匡正：プログラム処理要素の共用化度合からみた解法の相違—Bergland の例題を題材として，情報処理学会論文誌，Vol. 26, No. 4, pp. 759-762 (1985)．
6) Bergland, G. D.: A Guided Tour of Program Design Methodologies, *Computer*, Vol. 14, No. 10, pp. 13-37 (1981)．
7) 佐藤匡正：プログラム差異の代数的分析，電子通信学会総合全国大会，p. 6-83 (1986)．
8) 佐藤匡正，浅見秀雄：フローチャート階層的表

現のための一提案，第20回情報処理学会全国大会論文集，pp. 285-286 (1979).
9) Satoh, T. et al.: Documentation Technology for Packing Hierarchical Function, Data, and Control Structures, *COMPCON (IEEE) '81 Fall Proceedings*, pp. 284-290 (1981).
10) 佐藤匡正：プログラミング用ドキュメンテーション，情報処理，Vol. 22, No. 5, pp. 383-389 (1981).
11) 佐藤匡正，長野宏宣：HIPO 記述の一手法，第17回情報処理学会全国大会論文集，pp. 489-490 (1976).

付　　録

1. 変異の同値性

変異プログラムの実体が同じになるのは機構の働きによる．つまり，変異関係の構造に対して適当な機構を与えることによって同値プログラムが得られる．この状況を簡単なプログラム例で観察する．

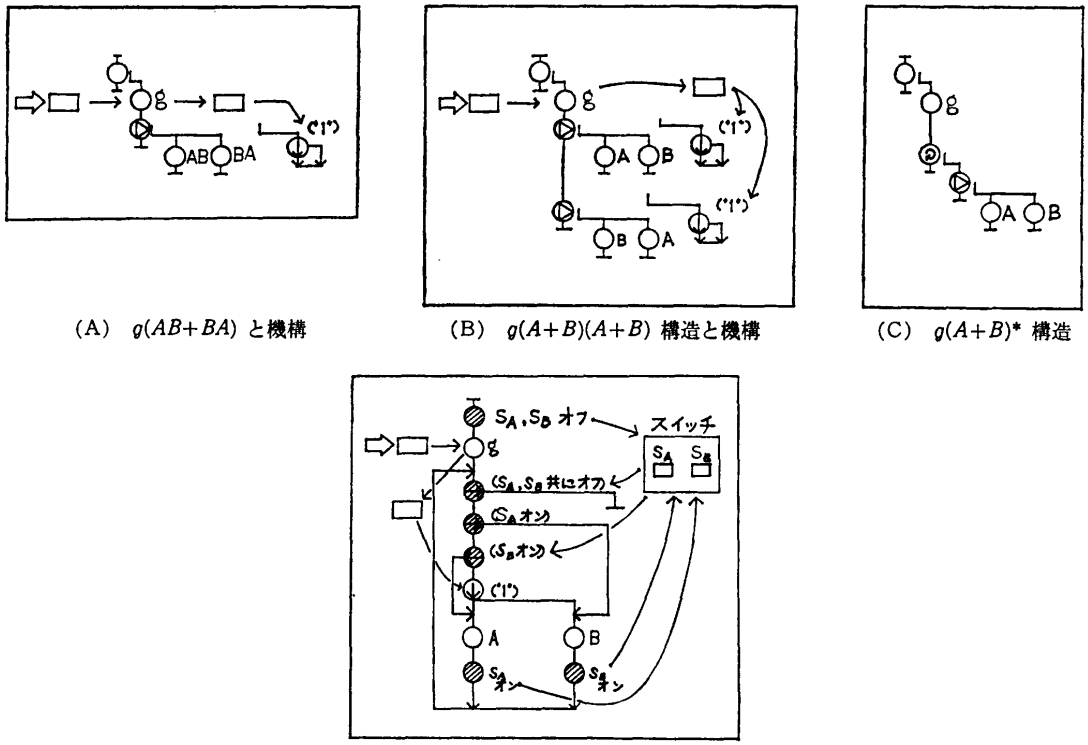
入力を得て，これがある値，例えば“1”なら基本要素の連なりは AB で，そうでなければ連なりは BA であるようなプログラムを考える．いま，基本要素 A では入力を変換するものとする．このプログラムの構造は，「入力を得る」を g とすると，$g(AB+BA)$ である．この構造の変異を考察する．包含変異変形で拡張すると，

$$g(AB+BA)$$
$$\subseteq g(A+B)(A+B) \qquad (A.1)$$
$$\subseteq g(A+B)^* \qquad (A.2)$$

$g(AB+BA)$ は式(A.1)，(A.2)の変異である．これらが同値プログラムになるのは次の理由による．

式(A.1)では最初と次の $(A+B)$ の選択機構の組合せで，ここでの基本要素の連なりを AB と BA に限定することができる．これを実現するには，例えば，最初の機構では，入力値が“1”であれば A が選択され，次の機構では，“1”であれば B が選択されるようにする．この機構の例を付図1(B)に示す．

式(A.2)では繰返しと選択の機構を組み合わせて，必要な基本要素の連なりのみに限定することができれば，プログラムの同値性は保てる．例えば，A, B の選択切り替えに二つのスイッチを使う．構造と機構を一体化したプログラムを CP 図法[11]で付図1(D)に示す．図中，⊖○などの記号に斜線を入れたものが A, B の選択切り替えのための機構の規定である．



(A) $g(AB+BA)$ と機構　(B) $g(A+B)(A+B)$ 構造と機構　(C) $g(A+B)^*$ 構造

(D) (C)構造のプログラム

付図 1 構造の包含関係と変異
Appendix fig. 1 Structure inclusion and variations.

以上により，変異の構造は機構を規定することによって同値プログラムになりうる．

2. 変異変形の定理

式 A, B について次の定理が成り立つ．

定理 1 $(A)^*\doteqdot((A)^*)^*\doteqdot((A)^*(A)^*)^*$

証明] 略．

定理 2 $((A)^*B+\varepsilon)\subseteq(A)^*(B+\varepsilon)$

証明] $(A)^*(B+\varepsilon)\doteqdot((A)^*B+(A)^*)$

$\supseteq((A)^*B+\varepsilon)\doteqdot$左辺 ■

定理 3 $(A)^+\subseteq(A)^*$

証明] 略．

定理 4

ⅰ) $(A)^*B\subseteq(A+B)^*$,

ⅱ) $A(B)^*\subseteq(A+B)^*$

証明]

ⅰ) 右辺$\doteqdot(A+B)^*\supseteq(A+B)^+$ [←定理 3]

$\doteqdot(A+B)^*(A+B)$

$\supseteq(A)^*(A+B)$

$\supseteq(A)^*B\doteqdot$左辺 ■

ⅱ) 同様に，

右辺$\supseteq(A+B)(A+B)^*\supseteq(A)^*B\doteqdot$左辺 ■

系 $A(B)^+\subseteq((A+\varepsilon)B)^+$

証明] 右辺$\doteqdot((A+\varepsilon)B)^+\doteqdot(A+\varepsilon)B((A+\varepsilon)B)^*$

$\supseteq AB((A+\varepsilon)B)^*$

$\supseteq AB(B)^*\doteqdot A(B)^+\doteqdot$左辺 ■

定理 5 $(A+B)^*\doteqdot((A)^*+(B)^*)^*\doteqdot((A)^*(B)^*)^*$

証明]

$(A+B)^*\doteqdot((A+B)^*)^*\doteqdot((A+B)^*(A+B)^*)^*$ [←定理 1]

$\supseteq((A)^*(B)^*)^*\doteqdot(((A)^*(B)^*)+((A)^*(B)^*))^*$

$\supseteq((A)^*+(B)^*)^*\supseteq(A+B)^*$ (A.3)

$\therefore (A+B)^*\supseteq((A)^*(B)^*)^*\supseteq(A+B)^*$

よって，$(A+B)^*\doteqdot((A)^*(B)^*)^*$

次に，

$(A+B)^*\doteqdot((A)^*+(B)^*)^*\doteqdot(((A)^*+(B)^*)^*)^*$

$\doteqdot(((A)^*+(B)^*)^*((A)^*+(B)^*)^*)^*$

$\supseteq(((A)^*)^*((B)^*)^*)^*\doteqdot((A)^*(B)^*)^*$

一方，式(A.3)により，

$((A)^*(B)^*)^*\supseteq((A)^*+(B)^*)^*$

$((A)^*(B)^*)^*\doteqdot((A)^*+(B)^*)^*$

以上により，

$(A+B)^*\doteqdot((A)^*+(B)^*)^*\doteqdot((A)^*(B)^*)^*$ ■

系 $(A+\varepsilon)^*\doteqdot(A)^*$

証明] 上の定理で B を ε とすれば，

$(A+\varepsilon)^*\doteqdot((A)^*(\varepsilon)^*)^*\doteqdot((A)^*)^*\doteqdot(A)^*$ ■

定理 6 $(AB)^*\doteqdot(\varepsilon+A(BA)^*B)$

証明]

$(AB)^*\doteqdot(\varepsilon+AB+ABAB+ABABAB+\cdots)$

$\doteqdot(\varepsilon+A(\varepsilon+BA+BABA+\cdots)B)$

$\doteqdot(\varepsilon+A(BA)^*B)$ ■

定理 7 $((x)^*A(y)^*B)^*\subseteq(x+A+y+B)^*$

証明]

$((x)^*A(y)^*B)^*\subseteq((x+A)^*(y+B)^*)^*$ [←定理 4]

$\doteqdot((x+A)+(y+B))^*$ [←定理 5]

$\doteqdot(x+A+y+B)^*$ ■

定理 8 $(A(x)^+B)^*\subseteq(\varepsilon+((A+BA+\varepsilon)x)^*B)$

証明]

$(A(x)+B)^*\doteqdot(\varepsilon+A(x)^+(BA(x)^+)^*B)$ [←定理 6]

$A(x)^+(BA(x)^+)^*\subseteq(A(x)^++BA(x)^+)^*$ [←定理 4]

$\subseteq(((A+\varepsilon)x)^++((BA+\varepsilon)x)^+)^*$ [←定理 4・系 1]

$\subseteq(((A+\varepsilon)x)^*+((BA+\varepsilon)x)^*)^*$ [←定理 3]

$\doteqdot((A+\varepsilon)x+(BA+\varepsilon)x)^*$ [←定理 5]

$\doteqdot((A+BA+\varepsilon)x)^*$

$\therefore (A(x)^+B)^*\subseteq(\varepsilon+((A+BA+\varepsilon)x)^*B)$ ■

系 $(A(x)^+)^*\subseteq((A+\varepsilon)x)^*$

証明] 上の定理で B を ε とすると得られる．

(平成元年 12 月 20 日受付)
(平成 2 年 11 月 13 日採録)

佐藤 匡正（正会員）

昭和 42 年横浜国立大学電気工学科卒業．同年日本電信電話公社(現，日本電信電話(株)) 入社．電気通信研究所，データ通信本部などにおいて銀行業務処理システム，言語処理プログラムの開発等を通じてソフトウェア開発技術の研究実用化を推進する．現在，横浜創英短期大学に在籍，実用的なソフトウェア開発技術，特にプロジェクトを管理するための方法やプログラムを規定するための方法に興味をもっている