プログラムにおける計算的誤りの一検出方法*

落 水 浩一郎** 田 中 幸 吉** 北 橋 忠 宏**

Abstract

The logical misses in programs are usually detected by checking up intermediate results of executed programs.

However, if the check-up process is performed by computer, the debugging time of programs will be greatly shorten. The present authors formulate this method by defining incorrect intermediate results as the "Computational Errors". In detail, constructing the predicate based on variables in a program which provides with the correct intermediate results, we detect the error by the T/F value of its predicate. This method is a kind of verification of accounts. and especially effective in the case where properties of variables or relations between variables are given.

1. まえがき

プログラムにおける，いわゆる論理ミスの検出は，現状では，たとえば FORTRAN においてはトレース・ステートメント，トレース・モードなどを使用して，プログラム実行時における計算の中間結果を人間が調べ，中間結果が正しくないときは原プログラムが論理ミスをもつと判断することによって行なわれている．これはコンパイラによって検出される文法的誤りが完全に除去されたのちにも存在する誤りであり，具体例としては算術ステートメント中の演算子の書き誤り，変数やステートメントの書き落しや書きまちがい，マトリックスの計算で行と列を逆に計算したものなどがあげられるであろう．ここで中間結果の正しさを計算機自身に統一的な方法で判定させるならば，プログラムのデバッグにかける時間を大幅に短縮できる．本論文においてはこれを定式化し，プログラムの実行時における正しくない中間結果を"計算的誤り"として定義し，その検出の自動化に関して考察した．具体的には計算の中間結果を規定するような述語をプログラム中の変数をもとにして構成し，その述語の真偽によって原プログラムにおける論理ミスを検出しようとする一種の検算方式である．この場合，対象とする論理ミスの範囲が問題となるが，本論文においては，プログラマ自身は与えられた問題の解を求める手順は正しく認識しているものと仮定し，解法のフローチャート化およびフローチャートのコード化に際して発生する誤りのみを取り扱うことにする．

このような問題は一般に"プログラムの正しさ"の問題として知られており，とくに R. W. Floyd, Z. Manna, D. C. Cooper らはプログラムの文法的な正しさのみならず，プログラムの意味的正しさをも証明できるシステムを考察しており[1)~4)]，本論文とは関連が深い．まず，Z. Manna はプログラムをアブストラクト・プログラムによって形式化し，アブストラクト・プログラムを一階の述語論理式に埋め込むことによって，プログラムの正しさを一階述語論理における充足可能性の問題に還元させることに成功した[2),4)]．さらに，D. C. Cooper は，Z. Manna の定式化の自然な拡張として，プログラムを Paterson のプログラムシェーマをもとにして形式化し，それを二階の述語論理式に埋め込むことによって，与えられたプログラムが正しいということを，そのプログラムに対応する二階述語論理における w・f・f（整合論理式）が定理として証明可能であるということに帰着させた[3)]．両者の定式化は文法的（構文的）な正しさの証明に関しては成功をおさめたが，意味的な正しさの保証に関しては，つぎに述べるように不十分である．

まず Z. Manna においてはアブストラクト・プログラムの各頂点に対応させた次数 n の述語変数，D. C. Cooper においては，コントロール・パスの始点と

* A Detection Method of Computational Errors in Programs, by Koichiro Ochimizu, Kokichi Tanaka, Tadahiro Kitahashi (Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University)

** 大阪大学基礎工学部情報工学科

終点に対応させられたレジスタの有限集合を変数とする述語に，それぞれ計算の中間結果を規定するような意味をもたせて定式化させてはいるが，いずれも適当な述語の存在を仮定したうえで議論が進められており，その具体的な内容は明らかにされていない．“プログラムの正しさ”を計算機によって証明しようとするとき，この述語の現実的な形を与えなければ“意味的な正しさ”は証明できない．筆者らは基本的には，Z. Manna, D. C. Cooper と同じ立場に立ち，述語による中間結果の規定の考え方をとったが，このような述語のみをもとにして誤り検出システムを構成することは，すなわち“プログラムの意味的な正しさ”の検出を行なうことであると考え，検出システム，検出述語の形を具体的に与えることによって，前述のように，プログラムの論理ミスの検出に適用可能であることを示した．次節以下に，まずプログラムの計算過程を記述するシステムを，プログラム中に出現する変数と，その内容の対の集合として定義し，正しくない中間結果の定式化として“計算的誤り”なる概念をそのシステムより導入し，つぎにその計算的誤りの検出システムの構成およびその簡単化を行ない，さらに検出述語の例をいくつかあげ，最後に FORTRAN によるプログラムを例として代表的な論理ミスを検出した結果を示した．変数自身，または変数間の関係があらかじめわかっているようなプログラム（値そのものはわからなくてもよい），すなわち与えられた条件を満たす解を求める型のプログラムに対してはこの方法は非常に有効であるといえる．

2. 諸 定 義

2.1 計算過程記述システム

まずプログラム中に出現する変数（アドレス名）をもとにして計算過程記述システム (Computational process description system, 以下 CPDS と略す) を定義する．

定義 1.

$$\mathrm{CPDS}=\begin{Bmatrix}\mathrm{cpds}_0\\ \mathrm{cpds}_1\\ \vdots\\ \mathrm{cpds}_m\end{Bmatrix}=\|(x_k, a_{ki})\|_{\substack{1\leqq k\leqq n\\ 0\leqq i\leqq m}}$$

cpds_0: $\{(x_1, a_{10}), (x_2, a_{20}), \cdots\cdots, (x_n, a_{n0})\}$

初期状態における変数とその内容の対の集合．

cpds_i: $\{(x_1, a_{1i}), (x_2, a_{2i}), \cdots\cdots, (x_n, a_{ni})\}$ $(1\leqq i\leqq m-1)$

i 番目の実行命令が実行されたときの変数とその内容の対の集合．

cpds_m: $\{(x_1, a_{1m}), (x_2, a_{2m}), \cdots\cdots, (x_n, a_{nm})\}$

計算の実行終了時の変数とその内容の対の集合．

$x_k(1\leqq k\leqq n)$: 変数名．

表 1 コンパイラで除去できなかったプログラム・エラー
Table 1 Undetectable errors by compiler.

No.	種 類	内 容	使 用 言 語
1	ループに関するもの	ループの初期値の設定が正しく行なわれていない	FORTRAN, COBOL
		ループのインデッキシングが正しく行なわれていない． 1. インデックスが常に初期値に設定される． 2. インデッキシング関数（例 $I=I+1$）などが書き誤られているか，または脱落している． 3. Do-loop のインデックス多重使用．	FORTRAN, COBOL etc.
2	ステートメント，変数に関するもの	命令コードの書きまちがい．	PAL Ⅲ（アセンブラ）
		DIMENSION 宣言を忘れると Function 文と解釈される．	FORTRAN
		算術ステートメント中の演算子の書きまちがい．	FORTRAN
		変数の書きまちがい 1. 算術ステートメント中において 2. IF 命令内において etc.	FORTRAN etc.
3	サブルーチン関係	サブルーチン，サブプログラムとのデータのやりとりにおいて，タイプおよびデータサイズが不一致でもわからない．	FORTRAN
4	初期値関係	初期値設定なし．	一 般
		アキュムレータを空にすることを忘れる．	アセンブラー般
5	マトリクス	行と列を逆に計算する．	一 般
6	そ の 他	オーバー・フロー，I/O 装置の故障	

$a_{ki}(0 \leqq i \leqq m)$: 変数 x_k の cpds_i における内容.

∧: 変数 x_k の内容が cpds_i において初めて定義されるとき $a_{kj}(0 \leqq j \leqq i-1)$ はすべて ∧ とする.

2.2 計算的誤り

論理ミスの種類としては，筆者が 10 名程度のプログラミング経験者に対して，“コンパイラで除去できなかったプログラム・エラー” としてアンケート調査を行なったところ **Table 1** に示すような結果が得られた.

Table 1 において No. 6 のエラーは計算機のハードウェアと密接に関連しており，われわれがいう論理ミスの範囲からは逸脱するものである．ゆえに，このようなものを除いて考えるとつぎのようなことがいえるのではなかろうか．プログラムが論理ミスをもつならば，その実行時において変数の内容の中間結果が所定の値をとらなくなり（ただし所定の中間結果があらかじめわかっているとは限らない），その結果，計算結果が正しくなかったり，ループして結果がでなくなったりしてしまう．Table 1 において No. 1, No. 2, No. 4, No. 5 などはまさにその好例であるといえよう．したがって以後，逆に CPDS における変数の内容が正しくないことを “計算的誤り” をもつと定義し，計算的誤りをもつようなプログラムを論理ミスをもつプログラムとする.

定義 2 計算的誤り

CPDS における内容 a_{ki} が所定の値と異なるとき，a_{ki} を計算的誤りとよび，CPDS は計算的誤りをもつとする.

3. 計算的誤り検出システム

本節では，CPDS をもとにして，計算的誤りの検出システムを構成する．筆者らの目的は端的にいうならば，プログラムの実行段階の各点における変数の内容が妥当であるか否かを，その値に対して，あらかじめ述語を用いて与えられた条件を満たしているかどうかによって判断しようとするものである．そのために，まず基本的な計算的誤り検出システム (Computational error detection system, 以下 CEDS と略す) を構成し，さらに検出述語の具体例をあげ，CEDS についていくつかの性質を導き，それをもとにして CEDS の簡単化をおこなう.

3.1 基本的な計算的誤り検出システム

定義 3

CPDS に対してつぎのような述語の行列 CEDS を対応させる.

$$\text{CEDS}=\|P_{ki}(x_k)\|_{\substack{1\leqq k\leqq n\\ 0\leqq i\leqq m}}$$

$P_{ki}(x_k)$: 変数 x_k の cpds_i における内容に対してのみ定義される一変数述語.

定義 4

・なる演算をつぎのように定義する.

$$(\text{CEDS})\cdot(\text{CPDS})=\|P_{ki}(a_{ki})\|_{\substack{1\leqq k\leqq n\\ 0\leqq i\leqq m}}$$

ただし，$a_{ki}=\wedge$ のときは $P_{ki}(a_{ki})=T$ とする.

定義 5

i) 弱い検出の述語 $P_{ki}(x_k)$ はつぎの性質を満たす.

a_{ki} が計算的誤りでない$\Rightarrow P_{ki}(a_{ki})=T$

ii) 強い検出の述語 $P_{ki}(x_k)$ はつぎの性質を満たす.

a_{ki} が計算的誤りでない$\Leftrightarrow P_{ki}(a_{ki})=T$

注1) 定義5について少し説明をつけ加える．まず弱い検出の述語とは，たとえば a_{ki} が三角関数によって計算される値のとき，$P_{ki}(x_k)$ として $|x_k|\leqq 1$ のような述語を構成することを意味しており，このとき，a_{ki} が正しいならば当然 $P_{ki}(a_{ki})=T$ となるが，逆に $P_{ki}(a_{ki})=T$ となっても a_{ki} が正しいとは断言できない．すなわち，計算結果があらかじめわからないような変数に対して，その誤りを検出しようとするとき，その内容のある程度の正しさを証明しようとする立場である．一方強い検出の述語とは，たとえば DO-loop における制御変数のように，その変数の内容自体の推移があらかじめわかっているようなときや，また計算された値の正しさを保証しうる必要かつ十分な条件が存在するとき（たとえば2次方程式の解法プログラムにおける根）のように，その変数の内容を完全に規定できるときに構成される述語をさす.

定義 6

CEDS よりつぎのような述語を構成する.

$$P_{\text{detection}}=\bigwedge_{\substack{k=1,n\\ i=0,m}} P_{ki}(x_k)$$

$$\hat{P}_{\text{detection}}=\bigwedge_{\substack{k=1,n\\ i=0,m}} P_{ki}(a_{ki})$$

$\hat{P}_{\text{detection}}$ はその構成法から明らかなようにつぎの性質をもつ.

(P 1) $P_{ki}(x_k)$ がすべて強い検出の述語のとき，

CPDS が計算的誤りをもたない$\Leftrightarrow\hat{P}_{\text{detection}}=T$

(P 2) $P_{ki}(x_k)$ のうち少なくとも一つが弱い検出の

述語のとき，

CPDS が計算的誤りをもたない⇒$\hat{P}_{\text{detection}}=T$

3.2 検出述語

本節では検出述語 $P_{ki}(x_k)$ の具体例を与える方法を考察する．一般には中間結果や最終結果をあらかじめ知ることができないので，これは一見不可能であるかのように思える．しかし前節において導入した弱い検出の概念を使用すれば，値そのものはわからなくてもその値の正しさに対する必要条件を与えることは容易であるので，検出述語は，たとえば不等号関係などによって容易に与えられるであろう．しかし，弱い検出によっては必ずしも計算的誤りが検出されるとは限らないので，強い検出述語の成立条件が問題となる．これは変数自身，変数間の関係，変数の値に対する条件などがあらかじめ決定できるときに初めて可能になる．

以上を **Table 2** にまとめる．

表 2　検出述語の決定条件

Table 2　Decision conditions of error-detecting predicate.

No.	条　　件	検出述語
1	変数の内容に対する制限条件があらかじめ決定できる．	強い検出述語 弱い検出述語
2	変数の内容があらかじめ決定できる．	強い検出述語
3	変数間の関係があらかじめ決定できる．	強い検出述語 弱い検出述語

Table 3 に Table 2 に対応する具体例を示そう．

表 3　Table 2 の具体例

Table 3　Example of Table 2.

No.	例	検出述語
1	不等号関係 ・ある数が正数である． ・上限をもつ． ・下限をもつ．	弱い検出述語
	数列の一般項	強い検出述語
2	FORTRAN の DO ステートメント中の制御変数	強い検出述語
3	数列の漸化式	強い検出述語
	ある変数間の内容の和が上限をもつ．	弱い検出述語

Table 2, Table 3 において No. 2, No. 3 については CEDS の簡単化とも関係が深いので後節にその説明はゆずることにして，No. 1 の場合についてディオファントス的述語* を用いた検出述語の具体例をあげることにしよう．

定義 7

ⅰ）多項式とはつぎのようなものをいう．

$$\sum_{\substack{0\leqq i_1\leqq n_1\\ 0\leqq i_2\leqq n_2\\ \cdots\cdots\\ 0\leqq i_k\leqq n_k}} a_{i_1 i_2\cdots\cdots i_k}x_1^{i_1}x_2^{i_2}\cdots\cdots x_k^{i_k}$$

$a_{i_1 i_2\cdots\cdots i_k}$ は“正，負の整数または 0”，

$x_1, x_2, \cdots\cdots, x_k$ の変域は“負でない整数”の集合．

ⅱ）$P(\xi^{(k)})$ が多項式述語のとき，$P(\xi^{(k)})=0$ を多項式述語と呼ぶ．

ⅲ）ディオファントス的述語とはつぎのようなものをいう．

$$\bigvee_{\eta^{(m)}} S(\xi^{(n)}, \eta^{(m)})$$

ただし，$S(\xi^{(n)}, \eta^{(m)})$ は多項式述語．

ディオファントス的述語を用いると正整数に対して，**Table 4** に示すような検出述語を構成できる．

ディオファントス的述語は存在記号，論理積，論理和のもとで閉じ，否定および全称記号のもとでは閉じていないことが知られているので，Table 4 の No. 1～7 までの述語に存在記号の付加，論理積，論理和の演算を適用することによって，さらに複雑な検出述語を構成できるであろう．

(例 1)　DO 10 $I=1, 11, 2$ なる命令に対して，変数 I に対する強い検出述語はつぎのように与えられる．

$$\{\bigvee_{w}[(I-1-2w=0)]\}\wedge\{\bigvee_{z}[(z\neq 0)\wedge(11-I-z=0)]\}$$

表 4　Table 2, 3 の No. 1 に対する検出述語をディオファントス的述語を用いて実現した例

Table 4　Examples of error-detecting predicates using Diophantos-type predicates for Table 2, Table 3 (No. 1).

No.	条　　件	検出述語
1	変数 x_k の値が 0 でない．$x_k\neq 0$	$\bigvee_{y}(x_k-y-1=0)$
2	変数 x_k の値がある上限 a でおさえられる．$x_k<a$	$\bigvee_{z}[(z\neq 0)\wedge(a-x_k-z=0)]$
3	変数 x_k の値が下限 b をもつ．$b<x_k$	$\bigvee_{z}[(z\neq 0)\wedge(x_k-b-z=0)]$
4	変数 x_k の値は c でない．$x_k\neq c$	$(x_k<c)\vee(x_k>c)$
5	変数 x_k の値が初期値 a より b ずつふえてゆく．$x_k\equiv a(\bmod b)$	$\bigvee_{w}(x_k-a-bw=0)$
6	変数 x_k の値は素数でない．	$\bigvee_{y,z}[x_k=(y+2)(z+2)]$
7	変数 x_k の値は 2 のベキでない．	$\bigvee_{z,w}[x_k=(2z+3)w]$

* ディオファントス的述語の詳細に関しては文献 5)を参照されたい

もちろん，ディオファントス的述語は論理積に関して閉じているので，検出述語をすべてディオファントス的述語で表わした場合は，$P_{\text{detection}}$ もディオファントス的述語となる．さらに検出システムをディオファントス的述語をもとにして構成する利点は，CEDS を実現するさい，CEDS に高々，多項式の計算と，ある数が 0 であるか否かの判断機能をもたせるのみでよいことにある．

3.3 CEDS の簡単化

3.1, 3.2 によって計算的誤りを検出するシステムの基本構成を定めることができた．しかし，現実問題として，各 cpds_i のそれぞれの変数に対して一つ一つ述語を対応させることは不可能である．本節では，CEDS の簡単化について考察する．

(a) $\wedge$ の除去

CPDS に $a_{ki}=\wedge$ となる内容 a_{ki} が存在するとき，つぎの関係が成立する．

Th. 1

$$\bigwedge_{\substack{k=1,n\\ i=0,m}} P_{ki}(x_k) \Leftrightarrow \bigwedge_{k=1,n} \bigwedge_{i=i_k,m} P_{ki}(x_k)$$

(証明)　定義 4 より $a_{ki}=\wedge$ のとき $P_{ki}(a_{ki})=T$

変数 x_k に対して，$i=i_k$ にてはじめてその内容 a_{ki} が定義されるとすると，

$$\bigwedge_{i=0,m} P_{ki}(x_k) \Leftrightarrow \overbrace{T\wedge T\wedge\cdots\wedge T}^{i_k-1\text{個}}\wedge\Big(\bigwedge_{i=i_k,m} P_{ki}(x_k)\Big)$$

$$\Leftrightarrow \bigwedge_{i=i_k,m} P_{ki}(x_k)$$

$$\therefore \bigwedge_{\substack{k=1,n\\ i=0,m}} P_{ki}(x_k) \Leftrightarrow \bigwedge_{k=1,n} \bigwedge_{i=i_k,m} P_{ki}(x_k)$$

$Q\cdot E\cdot D$

これは a_{ki} が定義されているところでのみ検出述語を構成すればよいことを示す．

(b) ブロック化

x_k の内容 a_{ki} の CPDS における推移を $A_k=(a_{k0}, a_{k1}, \cdots\cdots, a_{km})$ なる順序対であらわし，A_k につぎのような仮定を与える．

1. x_k の初期値は $\text{cpds}_{j_1}(0\leqq j_1\leqq m)$ で定義される．
2. x_k に無関係なステートメントが実行されているときは，x_k の内容は保持される．

A_k に 1, 2 のような仮定を与えると，A_k に関してつぎの性質が得られる．

$$\left.\begin{array}{l} a_{k,j_1-1}=a_{k,j_1-2}=\cdots\cdots=a_{k0}\\ a_{k,j_2-1}=a_{k,j_2-2}=\cdots\cdots=a_{kj_1}\\ \cdots\cdots\cdots\cdots\cdots\cdots\\ a_{k,j_m}=a_{k,j_m-1}=\cdots\cdots=a_{k,j_m-1}\end{array}\right\}$$

ただし，$0=j_0<j_1<\cdots\cdots<j_{m-1}<j_m=m$

ここで，$A'_k=(a_{kj_1}, a_{kj_2}, \cdots\cdots, a_{kj_{m-1}})$ なる順序対を考えればつぎのような関係が成立する．

L. 1

$P_{\text{detection}(A_k)}=\bigwedge_{i=0,m} P_{ki}(x_k)$

$P_{\text{detection}(A_k')}=\bigwedge_{i=j_1,j_{m-1}} P_{ki}(x_k)$ とすると

$P_{\text{detection}(A_k)}\Leftrightarrow P_{\text{detection}(A_k')}$

(証明)

$\hat{P}_{\text{detection}(A_k)}=\bigwedge_{i=0,m} P_{ki}(a_{ki})$

$\hat{P}_{\text{detection}(A_k')}=\bigwedge_{i=j_1,j_{m-1}} P_{ki}(a_{ki})$ とおき

$a_{k,j_1-1}=\wedge$ とする（すなわち，a_{kj_1} にて x_k の内容は初めて定義される）．

$\Rightarrow$

i) $\hat{P}_{\text{detection}(A_k)}=T$ のとき

$A_k=(a_{k0}, a_{k1}, \cdots\cdots, a_{km})$ の各要素 $a_{ki}(0\leqq i\leqq m)$ はすべて計算的誤りでない．

したがって，その部分列である $A_k'=(a_{kj_1}, \cdots\cdots, a_{kj_{m-1}})$ の各要素 $a_{ki}(j_1\leqq i\leqq j_{m-1})$ も計算的誤りでない．

すなわち，$\hat{P}_{\text{detection}(A_k')}=T$

ii) $\hat{P}_{\text{detection}(A_k)}=F$ のとき

A_k の要素のうち少なくとも一つは計算的誤りであり，最初の計算的誤りを $a_{ki}(0\leqq i\leqq m)$ とする．

もし，$a_{ki}\in A_k'$ ならば，ただちに $\hat{P}_{\text{detection}(A_k')}=F$

もし，$a_{ki}\notin A_k'$ ならば，たとえばそれを $a_{ki}(j_p\leqq l\leqq j_{p+1}-1)$ とするとき仮定より $a_{ki}=a_{kj_p}$

$\therefore \hat{P}_{\text{detection}(A_k')}=F$

$\Leftarrow$

iii) $\hat{P}_{\text{detection}(A_k')}=T$ のとき

等しい内容に対しては同じ述語がわりあてられるとすると $\hat{P}_{\text{detection}(A_k)}=T$

iv) $\hat{P}_{\text{detection}(A_k')}=F$ のとき

明らかに，$\hat{P}_{\text{detection}(A_k)}=F$　　$Q\cdot E\cdot D$

Th. 2

$$P_{\text{detection}}\Leftrightarrow \bigwedge_{k=1,n} P_{\text{detection}(A_k')}$$

(証明)　$P_{\text{detection}}=\bigwedge_{\substack{k=1,n\\ i=0,m}} P_{ki}(x_k)$

$$=\bigwedge_{k=1,n} P_{\text{detection}(A_k)}$$

L. 1 より $P_{\text{detection}(A_k)} \Leftrightarrow P_{\text{detection}(A_k')}$

$\therefore P_{\text{detection}} = \underset{k=1,n}{\wedge} P_{\text{detection}(A_k')}$ $\quad Q \cdot E \cdot D$

(c) 列方向述語

定義 8

cpds$_i$ における変数 x_k の内容 a_{ki} が cpds$_j$($j<i$) における変数 x_k の内容 a_{kj} によってのみ決まるとき，x_k は CPDS において独立であるという．

Th. 3

変数 x_k が CPDS において独立であるとき，$a_{ki}=g(a_{kj})$ なる関数 g が全単射であれば，つぎに示す性質をもつ，x_k に対する強い検出の述語 $P_k(x_k)$ が存在する．

$$\underset{i=i_1,m}{\wedge} P_{ki}(x_k) \Leftrightarrow P_k(x_k)$$

(証明) Th. 1, Th. 2 の結果より，x_k の内容は A_k' を考えることにする．

$A_k'=(a_{kj_1}, a_{kj_2}, \cdots\cdots, a_{kj_m})$ とすると $(j_1<j_2\cdots<j_m)$ 仮定より $2 \leqq i \leqq m$ なる任意の i に対して

$$a_{kj_i}=g(a_{kj_{i-1}})$$

$$\therefore a_{kj_i}=\overbrace{g\cdot g\cdot\cdots\cdots\cdot g}^{i-1}(a_{kj_1})$$

すなわち $a_{kj_i}=(g)^{i-1}(a_{kj_1})$

仮定より g が全単射であるから g^{-1} が存在して，

$$a_{kj_1}=((g)^{i-1})^{-1}(a_{kj_i})=(g^{-1})^{i-1}(a_{kj_i})$$

$$\therefore P_k(x_k)=\{(g^{-1})^{i-1}(a_{kj_i})-a_{kj_1}=0\} \quad Q \cdot E \cdot D$$

(例 2)

われわれがよくおかす誤りの一つに，IF 命令（またはそれに相当する命令）により構成したループからコントロールがいつまでたっても抜け出ない場合がある．これを検出する述語を Th. 3 によって構成し実験例を示す．いまループの構造を **Fig. 1** のように仮定する．

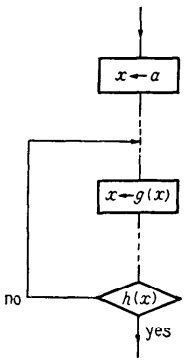

図 1 ループの例

Fig. 1 An example of loop.

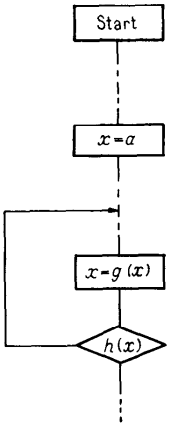

図 2 原プログラム

Fig. 2 An original program.

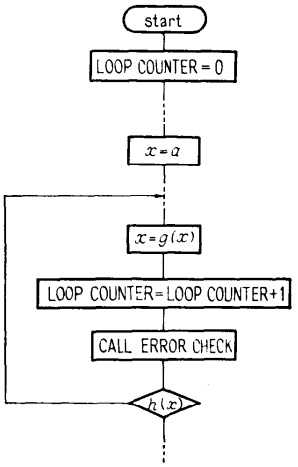

図 3 修飾された原プログラム

Fig. 3 A modified program of Fig. 2.

このとき L. 1 で与えられた順序対 A_k' は $(a, g(a), \cdots\cdots, g^p(a))$ となる．

$\therefore P(x)=\{(g^{-1})^i(x)-a=0,\ 1 \leqq i \leqq P\}$

を考えると，これはループのインデックスの値が正しく変化し，かつある上限でおさえられたか否かを知る強い検出の述語となる．この述語を使って FORTRAN によって実験を行なった．**Fig. 2** は原プログラム中におけるループの例，**Fig. 3** はエラーチェックを行なうために原プログラムに修飾を行なった例，**Fig. 4** は $P(x)$ の FORTRAN による実現例である．

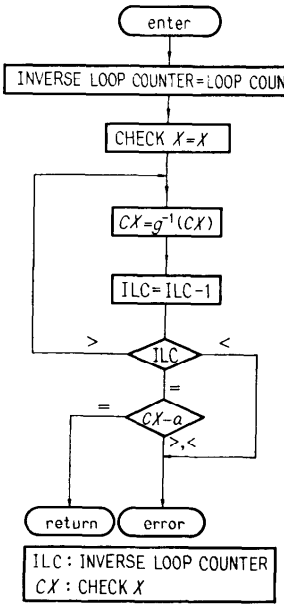

図 4 $P(x)$ の FORTRAN による表現

Fig. 4 Realization of $P(x)$ by FORTRAN.

$P(x)$ によって検出された論理ミスの例をつぎに示す.

(1) x のインデッキシングのまちがい ($g(x)$ の書きまちがい).

(2) IF 命令による飛び先 (返り先) のまちがい.

(3) IF 命令内の変数の書きまちがい.

(4) $g(x)$ の書きおとし.

(5) x の初期値の書き忘れ, 書きまちがい.

(d) 行方向述語

ある cpds$_i$ における変数の内容間の関係があらかじめわかっているときはつぎの関係を満たす述語,

$$P_{ki}(x_{k_1}, x_{k_2}, \cdots\cdots, x_{k_m})$$

を構成できる.

$$\bigwedge_{k=k_1,k_m} P_{ki}(x_k)=P_{ki}(x_{k_1}, x_{k_2}, \cdots\cdots, x_{k_m})$$

このとき, つぎの関係が成立する.

Th. 4

$$P_{\text{detection}}\Leftrightarrow(\bigwedge_{\substack{i\neq l\\ k=1,n}} P_{ki}(x_k))\wedge(\bigwedge_{\substack{i=l\\ k\neq k_1,k_m}} P_{ki}(x_k))\wedge(P_{ki}(x_{k_1}, x_{k_2}, \cdots\cdots, x_{k_m}))$$

(証明)

$$P_{\text{detection}}=\bigwedge_{\substack{i=0,m\\ k=1,n}} P_{ki}(x_k)=(\bigwedge_{\substack{i\neq l\\ k=1,n}} P_{ki}(x_k))\wedge(\bigwedge_{\substack{i=l\\ k=1,n}} P_{ki}(x_k))$$

```
C     FIBONACCI NUMBER
C     ERROR CHECK TEST
C     ORIGINAL
      DIMENSION IIF(30)
      IIF(1)=0
      IIF(2)=1
      WRITE(20,100) IIF(1)
      WRITE(20,100) IIF(2)
 100  FORMAT(I5)
      I=1
      J=2
      K=3
 3    IIF(K)=IIF(J)+IIF(I)
      IF(IIF(K)-10000) 1,1,2
 1    WRITE(20,100) IIF(K)
      I=I+1
      J=J+1
      K=K+1
      GO TO 3
 2    STOP
      END
```

図 5 原プログラム

Fig. 5 An original program.

$$\bigwedge_{k=1,n} P_{ki}(x_k)=(\bigwedge_{\substack{i=l\\ k\neq k_1,k_m}} P_{ki}(x_k))\wedge(\bigwedge_{\substack{i=l\\ k=k_1,k_m}} P_{ki}(x_k))=(\bigwedge_{\substack{i=l\\ k\neq k_1,k_m}} P_{ki}(x_k))\wedge P_{ki}(x_{k_1}, x_{k_2}, \cdots\cdots, x_{k_m})$$

$Q\cdot E\cdot D$

(例 3)

$x+y=5$ を満たす (x, y) の自然数解を求めるようなプログラムにおいて, 解 (x, y) に対する検出述語は, $P(x, y)=\{x-y-5=0\}$ のように与えることができる.

4. 実 験 例

フィボナッチ数を 10000 まで計算するようなプログラムを例として, 小規模のプログラムに対する適用例を示そう. **Fig. 5** に原プログラムを示す.

Table 5 に Fig. 5 のプログラム例に対する検出述語の例, および $P_{\text{detection}}$ を与える.

表 5 検出述語

Table 5 Error-detecting predicates.

$P_{\text{detection}}=P_1(I)\wedge P_2(J)\wedge P_3(K)\wedge P_4(IIF(K), IIF(J), IIF(I))\wedge P_5(IIF(K))$
$P_1(I)=\{(g^{-1})^k(I)=1, g^{-1}(I)=I-1\}$
$P_2(J)=\{(g'^{-1})^k(J)=2, g'^{-1}(J)=J-1\}$
$P_3(K)=\{(g''^{-1})^k(K)=3, g''^{-1}(K)=K-1\}$
$P_4(IIF(K), IIF(J), IIF(I))=\{IIF(K)-IIF(J)-IIF(I)=0\}$
$P_5(IIF(K))=\{IIF(K)-10000<0\}$

ただし k はループの深さを示す.

Fig. 6 に検出システムを付加したプログラム例を示す. Fig. 6 のプログラム中, ブロックAは $P_4(IIF(K), IIF(J), IIF(I))$ の実現部, ブロックBは $P_1(I)$, $P_2(J)$, $P_3(K)$ の実現部であり, ブロックAにおいてエ

```
C       FIBONACCI NUMBER
C       ERROR CHECK TEST
C       PROGRAM AND CEDS
        LC=0
        DIMENSION IIF(30)
        IIF(1)=0
        IIF(2)=1
        WRITE(20,100) IIF(1)
        WRITE(20,100) IIF(2)
  100   FORMAT(I5)
        I=1
        J=2
        K=3
    3   IIF(K)=IIF(J)+IIF(I)
        IF(IIF(K)-10000) 1,1,2
    1   WRITE(20,100) IIF(K)
        MZENK=IIF(K)-IIF(J)-IIF(I)
        IF(MZENK) 88,66,88
ブロックA
   88   WRITE(20,103)
  103   FORMAT(11HZENKA ERROR)
        PAUSE
   66   I=I+1
        J=J+1
        K=K+1
        LC=LC+1
        CALL! 99
        GO TO 3
   99   IIC=0
        JC=0
        KC=0
        IIC=I
        JC=J
        KC=K
        LLC=LC
   77   IIC=IIC-1
        JC=JC-1
        KC=KC-1
ブロックB
        LLC=LLC-1
        IF(LLC) 11,22,77
   22   IF(IIC-1) 11,23,11
   23   IF(JC-2) 11,24,11
   24   IF(KC-3) 11,25,11
   11   WRITE(20,102)
  102   FORMAT(10HLOOP ERROR)
        PAUSE
   25   RETURN!
    2   STOP
        END
```

図 6 検出システムを付加したプログラム

Fig. 6 A program with the detection system.

ラーが検出されたときは，ZENKA ERROR (すなわち漸化式のコード化部分に論理ミスがある)，ブロックBにおいてエラーが検出されたときは，LOOP ERROR (すなわちループの部分に論理ミスがある) とプリントアウトされる．

種々の論理ミスの検出例を **Table 6** に示す．

Table 6 をまとめると **Table 7** のようになる．

5. ま と め

プログラムの論理ミスを本論文で述べたような方法で検出しようとする試みは，変数自身または変数間のみたすべき条件があらかじめ決定できるようなときは有効である．難点は検出システム，すなわち検出述語をプログラマ自身が与えなければならない点にあるが，これはプログラムの意味論的な問題を解決しようとする場合にはつねに直面する壁であり，現在では有効な解決手段が見つからない．本論文で考察した方法において残された課題は，

表 6 検 出 例

Table 6 Examples of detection.

論理	ミス	TYPE OUT MODE
正	誤	
$J=J+1$	$J=J+2$	LOOP ERROR
$I=1$ ⋮ 3 $IIF(K)=IIF(J)+IIF(I)$	3 $I=1$ ⋮ $IIF(K)=IIF(J)+IIF(I)$	LOOP ERROR
$I=I+1$	な し (脱落)	LOOP ERROR
3 $IIF(K)=IIF(J)+IIF(I)$	3 $IIF(K)=IIF(J)+IIF(J)$	ZENKA ERROR
	3 $IIF(J)=IIF(J)+IIF(I)$	ZENKA ERROR
	3 $IIF(K)=IIF(J)-IIF(I)$	ZENKA ERROR

表 7 Table 6 の概要

Table 7 Interpretation of Table 6.

インデッキシングのあやまり
返り先まちがい
変数の書き誤り
演算子の書き誤り

1. 検出述語の統一的な決定法．
2. 検出プログラム作成の自動化 (すなわちコンパイラによる論理ミスの検出)

などであろう．

終わりに，プログラム理論その他の分野において，日ごろご助言いただく大阪大学基礎工学部助教授 豊田順一氏，またご討論いただく田中研究室諸氏に感謝の意を表します．

参考文献

1) R. W. Floyd: "Assigning meanings to Programs", Proc. Amer. Math. Soc. Symposia in Applied Mathematics, 19, pp. 19～31.
2) Z. Manna: "Properties of Programs and the First-order Predicate Calculus", JACM, vol. 16, No. 2, pp. 244～255 (April 1969).
3) D. C. Cooper: "Program Scheme Equivalences and Second-order Logic", Fourth Ann. Machine Intelligence, pp. 3～15 (1968).
4) Z. Manna: "The Correctness of Programs", JCSS, 3, pp. 119～127 (1969).
5) M. Davis 著，渡辺 茂訳: "計算の理論"，岩波書店，東京 (1966).

(昭和 46 年 9 月 23 日受付)