実行回数を含む手続き間情報の解析と整理のツール

A Tool for Analyzing and Documenting Interprocedural Information Including Execution Counts

牛島和夫　　田町典子
Kazuo USHIJIMA　　Noriko TAMACHI

九州大学　工学部　情報工学科
Department of Computer Science and Communication Engineering, Kyushu University

１．はじめに

プログラムの規模がある程度以上大きくなるとその作成・改善・保守等に際して何らかの支援ツールが必須である。規模が大きいために生ずる困難として次のような点があげられる。

（１）　プログラムが大きくなるに従つて手続きの数も増え，これらの間の結合関係が複雑になるため，プログラム全体の構造や全体に対する各手続きの位置付けの把握が困難になる。

（２）　手続き間関係の複雑化に伴つて，これらの手続きと結合している大域的なデータの役割の把握が困難となる。

（３）　ソーステキストが大きくなるため，その中から特定の情報を得るのにかなり手間がかかるようになる（絶望的な場合もある）。

既存の静的解析ツール，例えばDAVE[1] やPFORT verifier [2] でも上記３点に関する情報を提示する。しかしその出力結果は大量になりがちで，利用しやすいように整理されているとはいいがたい。従つて，これらの支援ツールに求められる条件として次の点があげられる。

（１）出力が多すぎないこと。

（２）必要な情報が捜しやすいよう整理されていること。

一方，筆者らの研究室で作成したFORTRANプログラム実行解析システムFORDAPは，プログラムの能率改善やテストのツールとして役立つている[3]が，解析の対象となるプログラムが大きくなるにつれ，各手続き間や手続きとデータ間の関係が複雑になり，しかも解析の出力結果が増える。解析結果から有用な情報を得るためには，プログラムの静的構造を明確にしておいた方がよい場合がある。

これらの状況に鑑みて，FORTRANで記述されたプログラムを対象とし，手続き間情報を中心に静的解析を行つてデータベースを作成し，動的解析の結果（実行回数情報）とも併合して，表形式に整理して表示するシステムを作成した。以下にその機能，構成，使用例等について述べる。

２．機　能

２．１　文法エラー

このシステムでは，入力されるプログラムがコンパイラにかけてエラーの検出されないものであることを条件としている。手続き内で検出可能なエラーはコンパイラに任せることによつて，システムの規模を適度にとどめることができるからである。従つて，ここで検出する文法エラーはそれ以外の，特に手続き間の結合に伴つて生ずるエラーである。

（１）手続きの呼出において実引数と仮引数の数が一致しているか。各々の引数について型が一致しているか。

（２）名コモンブロックが，それぞれの手続きにおいて同じ構造をもつよう宣言されているか。

（３）再帰的呼出が行なわれる可能性はないか。

（４）非正規RETURN文，ENTRY文，EQUIVALENCE文がないか。

なお（２）に反するプログラムでもFORTRANの文法上は誤りではないが，この条件は各コモン変数に対しそれぞれの手続きにおける情報を比較するためには必須である。また，プログラムのわかりやすさから見てもこの条件を満たしているこ

とが望ましい。

このシステムでは，各手続きが1つの入口と1つの出口をもつことを条件としているため，(4)のような制限となる。これにより，非正規RETURN文は通常のRETURN文と見なし，ENTRY文，EQUIVALENCE文は無視する。このため，ENTRY文で宣言された手続きを呼出すとソースプログラム中にない手続きを呼出したものとみなす。またEQUIVALENCEの定義があつても異つた名前のデータは異つた領域にあるものとみなす。

CALLING CHART

CALLED	CALLER	BLACK	BLUE	BROWN	GOLD	GRAY	GREEN	INDIGO	LILAC	ORANGE	PINK	PURPLE	RED	SILVER	VIOLET	WHITE	YELLOW	MAINPR
BLACK		0	.	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.
BLUE		X	0	.	.	.	.	.	.	.	.	.	.	.	X	.	.	.
BROWN		.	.	0	.	.	.	.	.	.	X	.	.	.	X	.	.	.
CLOCKM		.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.
DABS		.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.
DARCOS		.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.
DCOS		.	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.
DCOT		X	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
DSIGN		.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.	.
DSIN		X	.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	.
DSQRT		.	.	.	X	X	.	X	X	X	X	X	.	.	.	X	X	.
GOLD		.	.	.	0	.	.	.	.	.	.	X	.	.	.	.	.	.
GRAY		.	.	.	.	0	.	.	.	.	.	X	.	.	.	.	.	.
GREEN		.	.	.	.	.	0	.	.	.	.	X	.	.	.	.	.	.
INDIGO		.	.	.	.	.	.	0	.	.	.	X	.	.	.	.	.	.
LILAC		.	.	.	.	X	.	.	0	.	.	.	.	.	.	.	.	.
ORANGE		.	.	.	.	.	.	X	.	0	.	.	.	.	.	.	.	.
PINK		.	.	.	.	.	.	.	.	.	0	.	.	.	.	.	.	X
PURPLE		.	.	.	.	.	.	.	.	.	.	0	.	.	.	.	X	.
RED		.	.	.	.	.	.	.	.	.	.	.	0	.	.	.	X	.
SILVER		.	.	.	.	.	.	.	.	.	X	.	.	0	.	.	.	.
VIOLET		.	.	.	.	.	.	.	.	.	X	.	.	.	0	.	.	.
WHITE		.	.	.	.	X	.	.	.	.	.	.	.	.	.	0	.	.
YELLOW		.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	0	.
MAINPROG		.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	0

図1　手続き間の結合

COMMON CHART

COMMON	RO UT IN	BLACK	BLUE	BROWN	GOLD	GRAY	GREEN	INDIGO	LILAC	ORANGE	PINK	PURPLE	RED	SILVER	VIOLET	WHITE	YELLOW	MAINPR
ALPHA		X	X	X	.	X	.	.	.	.	X	.	.	.	X	.	.	.
BETA		.	.	.	.	.	.	X	.	.	X	.	X	.	.	.	.	.
CHI		X	X	X	.	.	.	.	.	.	X	X	X	.	.	.	.	.
DELTA		.	.	.	.	.	.	.	.	.	X	X	.	.	.	.	.	.
GAMMA		X	X	X	.	.	.	.	.	.	X	.	.	.	.	.	.	.
IOTA		.	.	.	X	X	.	X	.	.	X	.	.	.	.	.	.	.
KAPPA		.	.	.	.	.	X	.	.	.	X	.	.	.	.	.	.	.
LAMBDA		.	.	.	.	.	.	.	.	.	X	.	X	.	.	.	.	.
MU		.	.	.	.	.	X	.	.	.	X	.	.	.	.	.	.	.
NU		.	.	.	X	X	.	.	X	.	X	.	.	.	.	X	.	.
PHI		.	.	.	X	.	.	.	.	.	X	.	.	.	.	.	.	.
PI		X	X	X	X	X	.	.	.	.	X	.	.	.	.	.	.	.
PSI		.	.	.	X	.	.	.	.	.	X	.	.	.	.	.	.	.
RHO		X	X	X	X	.	.	X	.	.	X	.	.	.	.	.	.	.
SIGMA		.	.	.	X	X	.	X	.	.	X	.	X	.	.	.	X	.
TAU		.	.	.	.	.	X	X	.	.	X	.	.	.	.	.	X	.
THETA		.	.	.	.	.	.	.	.	.	X	.	X	.	.	.	.	.
XI		.	.	.	.	.	.	.	.	.	X	.	.	.	.	.	X	.

図2　手続きとコモンブロックの結合

2．2　手続き間情報

以下の各項目は見やすさを考慮して，縦軸と横軸に，手続き名，コモンブロック名，コモン変数名等をとつた表の形で出力される。探索の容易さを考慮して，手続き名あるいはコモンブロック名は辞書式に整列されている。ただし手続き名においては主プログラムとブロックデータが最後に，コモンブロック名においては無名コモンブロックが最後におかれる。

手続き間情報は，コンパイラの作成する相互参照表等を用いても得られる。しかしこの場合は，呼出側手続きから被呼出側手続きを捜すのは容易であつてもその逆は手間がかかる。また，手続きに組込まれているコモンブロックを列挙することは容易であるが，特定のコモンブロックを指定してそれを組込んでいる手続きを列挙するのはめんどうである。表形式で出力されているとこれらの間の関係が対称なので，どちら側から捜す場合も容易である。

それぞれの表は，実行回数をのせるもの以外はすべて同じ書式仕様で記述されているので，重ね合わせて調べることができる。実行回数に関してはその出力にある程度の桁数が必要なため，他の表と同じにできなかつた。この大きさでも桁数が足りない場合（10万回以上）は＊＊＊＊＊で表示する。

以下のそれぞれの表は，選択的に出力できる。

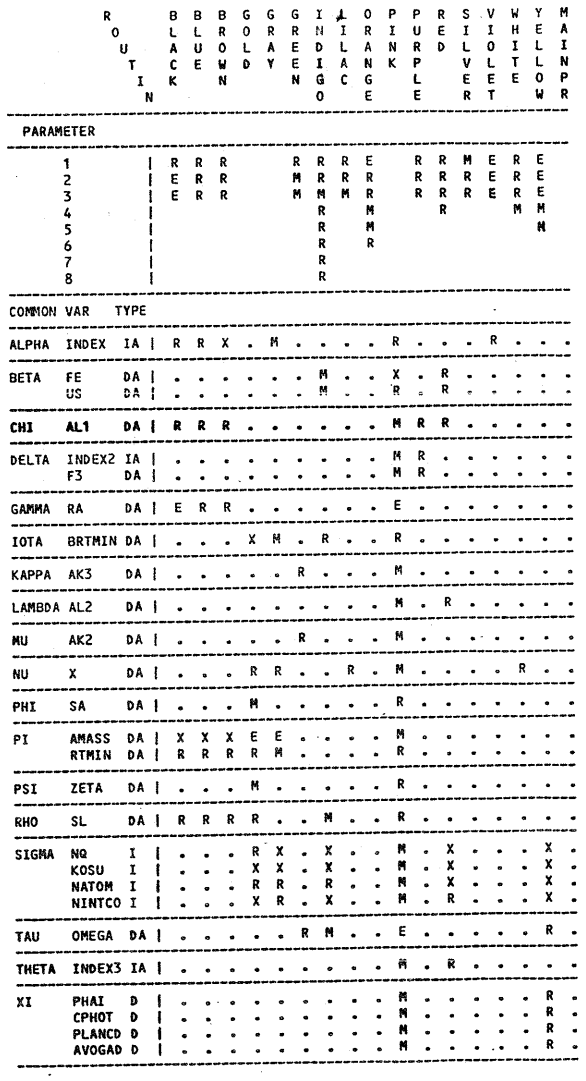
2．2．1　手続き間の結合（図1）

横軸に呼出側，縦軸に被呼出側の手続き名をとり，結合のある部分に×印を，また呼出側と被呼出側の手続きが一致している部分には○印をつける。

ソースプログラム中に含まれていない手続きの呼出があると，その手続き名は呼出側には現われず，被呼出側のみに存在する。従つてその並びには○印が存在せず，対角線上に並ぶ○印がひとつずれる。これによつてソースプログラム中に含まれない手続き名を知ることができる。

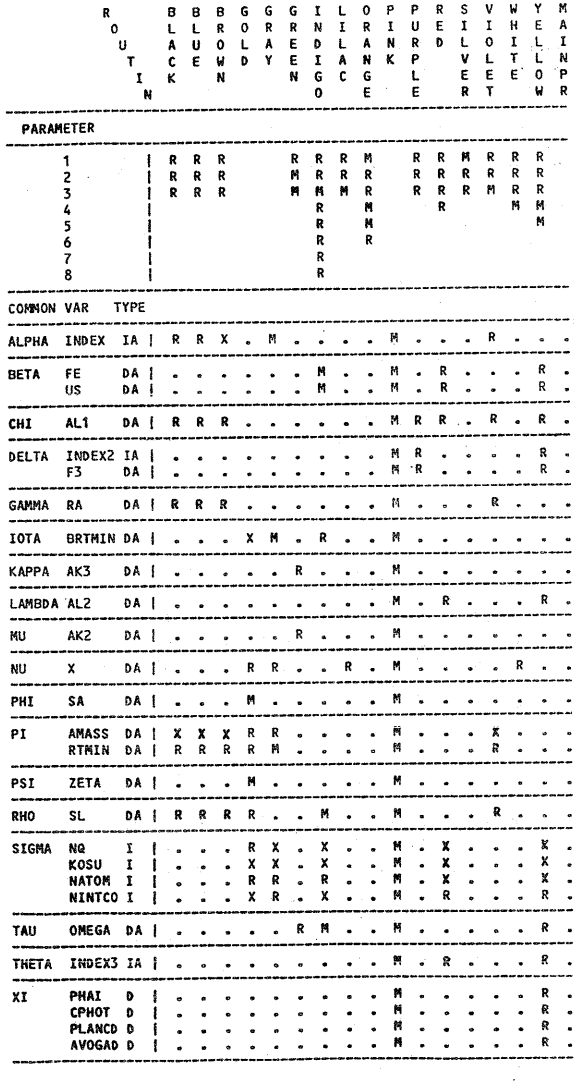
2．2．2　手続きとコモンブロックの結合（図2）

横軸に手続き名，縦軸にコモンブロック名をとり，結合のある部分に×印をつける。



ROUTIN	BLACK	BLUE	BROWN	GOLD	GRAY	GREEN	INDIGO	ILAC	ORANGE	PINK	PURPLE	RED	SILVER	VIOLET	WHITE	YELLOW	MAINPR
PARAMETER																	
1	R	R	R			R	R	R	E		R	R	M	E	R	E	
2	E	R	R			M	R	M	R		R	R	E	E	R	E	
3	E	R	R			M	R	M	R		R	R	E	E	M	M	
4							R		M				R				
5							R		M								
6							R		R								
7							R										
8							R										
COMMON VAR TYPE																	
ALPHA INDEX IA	R	R	X	.	M	.	.	.	.	R	.	.	.	R	.	.	.
BETA FE DA	.	.	.	.	.	.	M	.	.	X	.	R	.	.	.	.	.
US DA	.	.	.	.	.	.	M	.	.	R	.	R	.	.	.	.	.
CHI AL1 DA	**R**	**R**	**R**	.	.	.	.	.	.	M	R	R	.	.	.	.	.
DELTA INDEX2 IA	.	.	.	.	.	.	.	.	.	M	R	.	.	.	.	.	.
F3 DA	.	.	.	.	.	.	.	.	.	M	R	.	.	.	.	.	.
GAMMA RA DA	E	R	R	.	.	.	.	.	.	E	.	.	.	.	.	.	.
IOTA BRTMIN DA	.	.	.	X	M	.	R	.	.	R	.	.	.	.	.	.	.
KAPPA AK3 DA	.	.	.	.	.	R	.	.	.	M	.	.	.	.	.	.	.
LAMBDA AL2 DA	.	.	.	.	.	.	.	.	.	M	.	R	.	.	.	.	.
MU AK2 DA	.	.	.	.	.	R	.	.	.	M	.	.	.	.	.	.	.
NU X DA	.	.	.	R	R	.	.	R	.	M	.	.	.	.	R	.	.
PHI SA DA	.	.	.	M	.	.	.	.	.	R	.	.	.	.	.	.	.
PI AMASS DA	X	X	X	E	E	.	.	.	.	M	.	.	.	.	.	.	.
RTMIN DA	R	R	R	R	M	.	.	.	.	R	.	.	.	.	.	.	.
PSI ZETA DA	.	.	.	M	.	.	.	.	.	R	.	.	.	.	.	.	.
RHO SL DA	R	R	R	R	.	.	M	.	.	R	.	.	.	.	.	.	.
SIGMA NQ I	.	.	.	R	X	.	X	.	.	M	.	X	.	.	.	X	.
KOSU I	.	.	.	X	X	.	X	.	.	M	.	X	.	.	.	X	.
NATOM I	.	.	.	R	R	.	R	.	.	M	.	X	.	.	.	X	.
NINTCO I	.	.	.	X	R	.	X	.	.	M	.	R	.	.	.	X	.
TAU OMEGA DA	.	.	.	.	.	R	M	.	.	E	.	.	.	.	.	R	.
THETA INDEX3 IA	.	.	.	.	.	.	.	.	.	M	.	R	.	.	.	.	.
XI PHAI D	.	.	.	.	.	.	.	.	.	M	.	.	.	.	.	R	.
CPHOT D	.	.	.	.	.	.	.	.	.	M	.	.	.	.	.	R	.
PLANCD D	.	.	.	.	.	.	.	.	.	M	.	.	.	.	.	R	.
AVOGAD D	.	.	.	.	.	.	.	.	.	M	.	.	.	.	.	R	.

図3　変数の使用状況（呼出を含まない）



ROUTIN	BLACK	BLUE	BROWN	GOLD	GRAY	GREEN	INDIGO	LILAC	ORANGE	PINK	PUPLE	RED	SILVER	VIOLET	WHITE	YELLOW	MAINPR
PARAMETER																	
1	R	R	R			R	R	R	M			R	M	R	E	R	
2	R	R	R			M	R	R	R			R	R	R	E	E	
3	R	R	R			M	R	M	R			R	R	M	M	M	
4							R		M				R				
5							R		M								
6							R		R								
7							R										
8							R										
COMMON VAR TYPE																	
ALPHA INDEX IA	R	R	X	.	M	.	.	.	.	M	.	.	.	R	.	.	.
BETA FE DA	.	.	.	.	.	.	M	.	.	M	.	R	.	.	.	R	.
US DA	.	.	.	.	.	.	M	.	.	M	.	R	.	.	.	R	.
CHI AL1 DA	R	R	R	.	.	.	.	.	.	M	R	R	.	R	.	R	.
DELTA INDEX2 IA	.	.	.	.	.	.	.	.	.	M	R	.	.	.	.	R	.
F3 DA	.	.	.	.	.	.	.	.	.	M	R	.	.	.	.	R	.
GAMMA RA DA	R	R	R	.	.	.	.	.	.	M	.	.	.	R	.	.	.
IOTA BRTMIN DA	.	.	.	X	M	.	R	.	.	M	.	.	.	.	.	.	.
KAPPA AK3 DA	.	.	.	.	.	R	.	.	.	M	.	.	.	.	.	.	.
LAMBDA AL2 DA	.	.	.	.	.	.	.	.	.	M	.	R	.	.	.	R	.
MU AK2 DA	.	.	.	.	.	R	.	.	.	M	.	.	.	.	.	.	.
NU X DA	.	.	.	R	R	.	.	R	.	M	.	.	.	.	R	.	.
PHI SA DA	.	.	.	M	.	.	.	.	.	M	.	.	.	.	.	.	.
PI AMASS DA	X	X	X	R	R	.	.	.	.	M	.	.	.	X	.	.	.
RTMIN DA	R	R	R	R	M	.	.	.	.	M	.	.	.	R	.	.	.
PSI ZETA DA	.	.	.	M	.	.	.	.	.	M	.	.	.	.	.	.	.
RHO SL DA	R	R	R	R	.	.	M	.	.	M	.	.	.	R	.	.	.
SIGMA NQ I	.	.	.	R	X	.	X	.	.	M	.	X	.	.	.	X	.
KOSU I	.	.	.	X	X	.	X	.	.	M	.	X	.	.	.	X	.
NATOM I	.	.	.	R	R	.	R	.	.	M	.	X	.	.	.	X	.
NINTCO I	.	.	.	X	R	.	X	.	.	M	.	R	.	.	.	R	.
TAU OMEGA DA	.	.	.	.	.	R	M	.	.	M	.	.	.	.	.	R	.
THETA INDEX3 IA	.	.	.	.	.	.	.	.	.	M	.	R	.	.	.	R	.
XI PHAI D	.	.	.	.	.	.	.	.	.	M	.	.	.	.	.	R	.
CPHOT D	.	.	.	.	.	.	.	.	.	M	.	.	.	.	.	R	.
PLANCD D	.	.	.	.	.	.	.	.	.	M	.	.	.	.	.	R	.
AVOGAD D	.	.	.	.	.	.	.	.	.	M	.	.	.	.	.	R	.

図4　変数の使用状況（呼出を含む）

2.2.3 各手続きにおけるコモン変数の使用状況（図3，図4）

横軸に手続き名，縦軸にコモンブロック名をとり，各コモンブロックについて，その構成要素であるコモン変数をコモンブロック内での宣言順に並べる。各コモン変数については，その名前（これは，それが属する手続きによつて異ることがあるが，代表名をひとつあげている），型，及び配列であるか否かの別を出力する。

表中のM，R，Xは，各変数がその手続き内で代入（Modify）参照（Reference），未使用（X = not used）を表わしたものである。これにより，各コモン変数がそれぞれの手続きでどのように使用されているかがわかる。同時に各手続きの仮引数についても使用状況を出力する。

図3はその手続き内のみの使用状況であり，他の手続きの呼出に関しては，実引数として陽に渡されるもののみE（External）で表わし，コモン変数として渡されるものは無視している。

一方図4は，その手続きが持つ呼出をすべて含んだ使用状況であつて，その手続きが他の手続きから実際に呼ばれた場合どのように使用されるかを表わしている。従つて図4をみると次のようなことがわかる。

（1）実行に必要ないコモン変数や組込む必要のないコモンブロックがあればそれを指摘できる。

（2）プログラム中で定数として扱われているコモン変数を知ることができる。

（3）引数については，それが入力用に使用されているかあるいは（入）出力用に使用されているかの別がわかる。

これらの情報により，プログラムの一部を書き替える場合，変数に与える影響の検討が容易に行える。また必要なコモン変数のみが組込まれるようにコモンブロックを再構成しようとする場合にも手がかりを与える。

2.2.4 静的出現回数（図5，図6）

それぞれの名前がソースプログラム中に出現する回数を図1，図2と同様の表の上に記述する。図5では各手続きが他の手続きを何ケ所で呼んでいるか，図6では各手続きにおいてあるコモン変数あるいは仮引数が陽に何ケ所で使われているかを示す。

このような情報は，部分的な修正がプログラム

CALLING CHART

CALLED	CALL	BLACK	BLUE	BROWN	CLOCKM	DABS	DARCOS	DCOS	DCOT	DSIGN	DSIN	DSQRT	GOLD	GRAY	GREEN	INDIGO	LILAC	ORANGE	PINK	PURPLE	RED	SILVER	VIOLET	WHITE	YELLOW	MAINPROG
BLACK		.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
BLUE		1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
BROWN		.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
CLOCKM		.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
DABS		.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
DARCOS		.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
DCOS		.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
DCOT		1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
DSIGN		.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
DSIN		1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
DSQRT		.	.	1	2	.	1	1	1	1	12	.	.	.	.	.	.	.	.	.	2	1	.	.	.	.
GOLD		.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
GRAY		.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
GREEN		.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
INDIGO		.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
LILAC		.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
ORANGE		.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
PINK		.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1
PURPLE		.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.	.
RED		.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	2	.	.	.	.
SILVER		.	.	.	.	.	.	.	.	.	4	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
VIOLET		.	.	.	.	.	.	.	.	.	4	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
WHITE		.	.	.	2	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
YELLOW		.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
MAINPROG		.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

図5　静的呼出回数

COMMON CHART

COMMON VAR	VAR	TYPE	ROUTINE	BLACK	BLUE	BROWN	CLOCKM	DABS	DARCOS	DCOS	DCOT	DSIGN	DSIN	DSQRT	GOLD	GRAY	GREEN	INDIGO	LILAC	ORANGE	PINK	PURPLE	RED	SILVER	VIOLET	WHITE	YELLOW	MAINPROG
PARAMETER			1	3	3	2			7	2	1	26		1	4	1	5	2	7
			2	4	2	1			4	2	1	12		1	1	1	3	10	
			3	4	2	1			4	2	3	3		1	1	1	3	8	
			4							8		9			1			6	
			5							2		2						3	
			6							1		2							
			7							1									
			8							3									
ALPHA	INDEX	IA	69	6	2	-	.	.	.	.	.	1	.	.	.	2	.	.	.
BETA	FE	DA	69	.	.	.	.	.	.	1	.	.	-	.	1	.	.	.	.
	US	DA	69	.	.	.	.	.	.	1	.	.	2	.	1	.	.	.	.
CHI	AL1	DA	69	1	1	1	.	.	.	.	.	.	4	1	1	.	.	.	.
DELTA	INDEX2	IA	69	.	.	.	.	.	.	.	.	.	11	4	.	.	.	.	.
	F3	DA	69	.	.	.	.	.	.	.	.	.	3	1	.	.	.	.	.
GAMMA	RA	DA	69	3	1	2	.	.	.	.	.	.	15	.	.	.	.	.	.
IOTA	BRTMIN	DA	69	.	.	.	-	1	.	1	.	.	2	.	.	.	.	.	.
KAPPA	AK3	DA	69	.	.	.	.	.	2	.	.	.	4	.	.	.	.	.	.
LAMBDA	AL2	DA	69	.	.	.	.	.	.	.	.	.	9	.	1	.	.	.	.
MU	AK2	DA	69	.	.	.	.	.	2	.	.	.	4	.	.	.	.	.	.
NU	X	DA	69	.	.	.	2	2	.	.	1	.	33	.	.	.	3	.	.
PHI	SA	DA	69	.	.	.	1	.	.	.	.	.	7	.	.	.	.	.	.
PI	AMASS	DA	69	-	-	-	1	1	.	.	.	.	6	.	.	.	.	.	.
	RTMIN	DA	69	1	1	2	2	2	.	.	.	.	2	.	.	.	.	.	.
PSI	ZETA	DA	69	.	.	.	1	.	.	.	.	.	3	.	.	.	.	.	.
RHO	SL	DA	69	1	1	2	3	.	.	1	.	.	5	.	.	.	.	.	.
SIGMA	NQ	I	69	.	.	.	3	.	.	-	.	.	7	.	-	.	.	.	-
	KOSU	I	69	.	.	.	-	.	.	-	.	.	2	.	-	.	.	.	-
	NATOM	I	69	.	.	.	4	4	.	2	.	.	11	.	-	.	.	.	-
	NINTCO	I	69	.	.	.	-	2	.	-	.	.	4	.	2	.	.	.	-
TAU	OMEGA	DA	69	.	.	.	.	.	2	1	.	.	8	.	.	.	.	1	.
THETA	INDEX3	IA	69	.	.	.	.	.	.	.	.	.	16	.	4	.	.	.	.
XI	PHAI	D	69	.	.	.	.	.	.	.	.	.	3	.	.	.	.	1	.
	CPHOT	D	69	.	.	.	.	.	.	.	.	.	2	.	.	.	.	1	.
	PLANCD	D	69	.	.	.	.	.	.	.	.	.	2	.	.	.	.	1	.
	AVOGAD	D	69	.	.	.	.	.	.	.	.	.	2	.	.	.	.	1	.

図6　静的出現回数

CALLING CHART

CALLED \ CALLER	BLACK	BLUE	BROWN	GOLD	GRAY	GREEN	INDIGO	LILAC	ORANGE	PINK	PURPLE	RED	SILVER	VIOLET	WHITE	YELLOW	MAINPROG
BLACK	.	.	.	.	.	.	.	.	.	.	.	.	.	684	.	.	.
BLUE	684	.	.	.	.	.	.	.	.	.	.	.	.	741	.	.	.
BROWN	.	.	.	.	.	.	.	.	.	.	.	.	.	5	.	.	.
CLOCKM	.	.	.	.	.	.	.	.	.	2	.	.	.	.	.	.	.
DABS	.	.	.	.	.	.	.	.	*****	.	.	.	.	.	.	.	.
DARCOS	.	.	.	.	.	.	.	.	.	2	.	.	.	.	.	.	.
DCOS	.	.	.	.	.	.	.	.	.	10	.	.	.	.	.	.	.
DCOT	684	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
DSIGN	.	.	.	.	.	.	.	.	814	.	.	.	.	.	.	.	.
DSIN	684	.	.	.	.	.	.	.	.	12	.	.	.	.	.	.	.
DSQRT	.	.	.	8	9	.	18	13	4	16	.	.	.	.	48	60	.
GOLD	.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.
GRAY	.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.
GREEN	.	.	.	.	.	.	.	.	.	5	.	.	.	.	.	.	.
INDIGO	.	.	.	.	.	.	.	.	.	4	.	.	.	.	.	.	.
LILAC	.	.	.	.	13	.	.	.	.	.	.	.	.	.	.	.	.
ORANGE	.	.	.	.	.	.	4	.	.	.	.	.	.	.	.	.	.
PINK	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1
PURPLE	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	60	.
RED	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	95	.
SILVER	.	.	.	.	.	.	.	.	.	4	.	.	.	.	.	.	.
VIOLET	.	.	.	.	.	.	.	.	.	1430	.	.	.	.	.	.	.
WHITE	.	.	.	.	24	.	.	.	.	.	.	.	.	.	.	.	.
YELLOW	.	.	.	.	.	.	.	.	.	60	.	.	.	.	.	.	.
MAINPROG	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

図 7　動的呼出回数

COMMON CHART

PARAMETER \ ROUTINE	BLACK	BLUE	BROWN	GOLD	GRAY	GREEN	INDIGO	LILAC	ORANGE	PINK	PURPLE	RED	SILVER	VIOLET	WHITE	YELLOW
1	2052	6327	10			190	96872	39	*****		420	4900	624	3606	144	335
2	4104	8436	5			70	15600	39	22537		420	*****	4	1430	216	485
3	4104	8436	5			70	4200	117	891		420	*****	24	1430	216	390
4							16275		85848			*****			216	275
5							600		818							120
6							4		1636							
7							18									
8							732									

COMMON	VAR	TYPE	BLACK	BLUE	BROWN	GOLD	GRAY	GREEN	INDIGO	LILAC	ORANGE	PINK	PURPLE	RED	SILVER	VIOLET	WHITE	YELLOW
ALPHA	INDEX	IA	4104	4218	–	.	254	.	.	.	.	1	.	.	.	2176	.	.
BETA	FE	DA	.	.	.	.	.	.	104	.	.	–	.	*****	.	.	.	.
	US	DA	.	.	.	.	.	.	624	.	.	2	.	*****	.	.	.	.
CHI	AL1	DA	684	2109	5	.	.	.	.	.	.	11702	420	*****	.	.	.	.
DELTA	INDEX2	IA	.	.	.	.	.	.	.	.	.	29	1320	.	.	.	.	.
	F3	DA	.	.	.	.	.	.	.	.	.	7	420	.	.	.	.	.
GAMMA	RA	DA	2052	2109	10	.	.	.	.	.	.	31	.	.	.	.	.	.
IOTA	BRTMIN	DA	.	.	.	–	624	.	14976	.	.	11258	.	.	.	.	.	.
KAPPA	AK3	DA	.	.	.	.	.	60	.	.	.	121	.	.	.	.	.	.
LAMBDA	AL2	DA	.	.	.	.	.	.	.	.	.	2830	.	*****	.	.	.	.
MU	AK2	DA	.	.	.	.	.	60	.	.	.	121	.	.	.	.	.	.
NU	X	DA	.	.	.	1296	2	.	.	39	.	77	.	.	.	.	216	.
PHI	SA	DA	.	.	.	162	.	.	.	.	.	139	.	.	.	.	.	.
PI	AMASS	DA	–	–	–	8	8	.	.	.	.	34	.	.	.	.	.	.
	RTMIN	DA	2052	6327	10	1296	632	.	.	.	.	240	.	.	.	.	.	.
PSI	ZETA	DA	.	.	.	972	.	.	.	.	.	344	.	.	.	.	.	.
RHO	SL	DA	2052	6327	10	9072	.	.	432	.	.	11634	.	.	.	.	.	.
SIGMA	NQ	I	.	.	.	66	–	.	–	.	.	47	.	–	.	.	.	–
	KOSU	I	.	.	.	–	–	.	–	.	.	2	.	–	.	.	.	–
	NATOM	I	.	.	.	1135	53	.	8	.	.	1435	.	–	.	.	.	–
	NINTCO	I	.	.	.	–	2	.	–	.	.	4	.	*****	.	.	.	–
TAU	OMEGA	DA	.	.	.	.	.	10	18	.	.	655	.	.	.	.	.	60
THETA	INDEX3	IA	.	.	.	.	.	.	.	.	.	16	.	4900	.	.	.	.
XI	PHAI	D	.	.	.	.	.	.	.	.	.	3	.	.	.	.	.	60
	CPHOT	D	.	.	.	.	.	.	.	.	.	2	.	.	.	.	.	60
	PLANCD	D	.	.	.	.	.	.	.	.	.	2	.	.	.	.	.	60
	AVOGAD	D	.	.	.	.	.	.	.	.	.	2	.	.	.	.	.	60

図 8　動的アクセス回数

全体に影響を及ぼす可能性のある場合には有効である。手続きの仮引数の数や順序をかえようとする場合にはそれを呼んでいる実引数列も変更しなければならないが，その際，その呼出をもつ手続き名とその手続き中何ケ所にあるかがわかつていれば見落しが少くてすむ。あるコモン変数の扱い方を変えようとする場合，例えば配列の次元の扱い方を変える場合などにも役立つ。

これらの用途には相互参照表を利用することも可能であるが，手続き数が多くなるとこれも膨大なものとなり，数多い局所変数の中から手続き名やコモン変数名を捜し出すのはかなり手間がかかる作業となる。最終的には相互参照表を使用するにしても，あらかじめ大局的な情報を図５，図６から得ておくとその手間は少くてすむ。

２．２．５　動的アクセス回数（図７，図８）

プログラムの各文の実行回数を計数することによつて得たデータをもとに，手続きの呼出回数（図７），変数のアクセス回数（図８）を図５，図６と同じ表の上に出力する。

FORDAPの出力はプログラムの解析に有用であるが，ソースプログラムの行数とほぼ同じ行数が出力されるため，プログラムが大きくなると結果を把握しにくくなる。このような時，出力の要約としての表があると理解を助ける。FORDAPでは各手続きが何回呼ばれたかはわかつても，いくつかの手続きからそれぞれ何回呼ばれたかという情報は得にくい。静的構造の上にのせて表の形で出力を与えると，この点でも便利である。

実行回数情報から，実行の集中している部分を発見して，そこを局所的に手直しするだけで，全体として効率の向上が可能である［３，４］。さらに，実行の集中した手続きの呼出回数を減らすことができればその効果は一層大きい。しかしこの手直しはプログラム全体に影響を及ぼすので，図７などの助けなしには困難である。このほか呼出頻度の多い小さな手続きを呼出側に組込んだり，使用される回数の多い変数に対しては，小さな手続きであれば，常にレジスタにのせておくようアセンブリ語で書き直すべきかどうかの指針等が得られる。

プログラムテストの立場から見ると，実行回数またはアクセス回数の少いところは大きな情報を与えてくれる。実行回数ゼロというのはプログラム中に未テスト部分があることを示している。このような部分の中には通常は実行されない部分，例えばエラー処理部等も含まれる。エラー処理ルーチンはいろいろな部分から呼ばれる可能性はあるが，実際にはほとんど実行されない。何らかの条件によりその手続きが実行されたとき，それがどこから呼ばれて走つたのかを知るのには図７が役立つ。めつたに起こらないことが起こつたことを知らせるフラグ変数の値がどこで書きかえられたかは図８からわかる。

３．構　成

全体は４つの部分，実行回数計数部（FORDAP），入力部（BASE），解析部（ANALYSIS），出力部（OUTPUT）から成る（図９参照）。実行回数計数部は，ソースプログラムと実行時のデータを入力し，実行結果と各文に対する実行回数付きのリストを出力する。その際，実行回数に関するデータがプログラムデータベースＤ（Dynamic）に作成される。入力部はソースプログラムを入力し，それを静的に解析して，得た情報をプログラムデータベースＳ（Static）に蓄える。この際文法上の誤り（２．１節参照）があれば検出する。解析部は，プログラムデータベースＳ及びＤを入力して，出力すべき情報を抽出し，出力用作業ファイル（Information for Output）に出力する。出力部ではこれを入力して編集し，表形式で出力する。

プログラムデータベースＳとＤはいずれも解析部に使用される。Ｄは実行回数のデータだから，実行時のデータを変える等していくつかの異る実行状態に対する解析を必要とする場合は，それぞれに対応して作成されねばならない。これに対してＳはソースプログラムの静的解析によつて作成されるので，ソースプログラム自体が書き替えられない限り作り直す必要はない。

３．１　実行回数計数部

これは我々の研究室で作成し，現在九大情報工学科FACOM230-45S，九大，東大，名大の各大型計算機センター等で使用されているFORDAPを利用して，実行回数データをファイルに出力するように修正したものである。

３．２　入力部

ソースプログラムを入力し，構文解析を行い，情報を蓄積する。蓄積される情報のうち，局所的なものは入力される手続きごとに作成され，ファ

イルに出力される。大域的な情報は手続きの区切りごとに追加され，ソースプログラム中の最後の手続きの処理が終つた段階で完成する。これらのデータをプログラムデータベースSと呼び，局所的な部分をL(Local)，大域的な部分をG(Global)と呼ぶ。

データベースL中の情報は，名前，実行に関する基本ブロック，及び両者の結合の3つに分けて蓄積される。名前データにはその属性，型，及び仮引数ならばその引数列における順序，コモン変数ならばその属するコモンブロック名とその中における順序等がある。基本ブロックに関するデータには，その対応する文の種類，コントロールフローの行先等がある。結合データには，名前と基本ブロックを指定してその名前がその基本ブロックにおいてどのように使われているか（参照，代入，及び配列添字ならその配列名と添字の次元，呼出の実引数ならその手続き名と引数列における順序等）がある。

データベースGには，手続きとコモンブロック及びそれらの結合に関するデータが蓄積される。手続きについてはその持つ仮引数の数，それぞれの仮引数に対して型，配列であればその大きさを蓄える。コモンブロックについてはその要素とするコモン変数の数，それぞれのコモン変数に対して名前，型，配列であればその大きさを蓄える。また手続き間の結合，手続きとコモンブロックの結合についても必要な情報を蓄える。

その他，これらの蓄積されたデータとソースプログラムを対応づけるためのいくつかのポインタを持つ。

なお入力部では，手続き間の情報を抽出，蓄積するに際して，2.1節で述べたような文法エラーを検出する。

3．3　解析部

手続き間の結合関係を考慮して解析を行うためには，現在解析中の手続きから呼ばれている手続きすべてが既に解析済でなければならない。すべての手続きをこのような条件下で解析するためには手続きの解析順序が問題となる。ここでは上の条件を満たし，かつファイルアクセスがなるべく少くてすむ順序で手続きを解析する。

解析は主にプログラムデータベースLのうえで行なわれるが，他の手続き呼出があるとGから必要な情報を得る。Lについては，現在解析中の手続きに対応する部分だけしか主記憶上にないので他の手続きに関する情報はGから得るしかない。従つて手続きを解析した後で，それを呼ぶ側からみて必要な情報はその都度Gに追加しておかねばならない。

実行回数の情報はプログラムデータベースD上にある。これは実行回数計数部から得るもので，入力ソースプログラムの各レコードに対応して実行回数情報が蓄えられている。従つて，プログラムデータベースSとソースプログラムを対応づけるポインタによつて，SとDも対応づけることができる。

これらのデータベース上で解析を行つた結果，Gには追加情報が蓄えられ，出力すべき情報が出力用作業ファイルに作成される。

3．4　出力部

出力されるべき情報を出力用作業ファイルから入力し，プログラムデータベースSの助けをかりてこれを表の形に編集する。出力部では，必要な表だけを選択的に出力することができる。

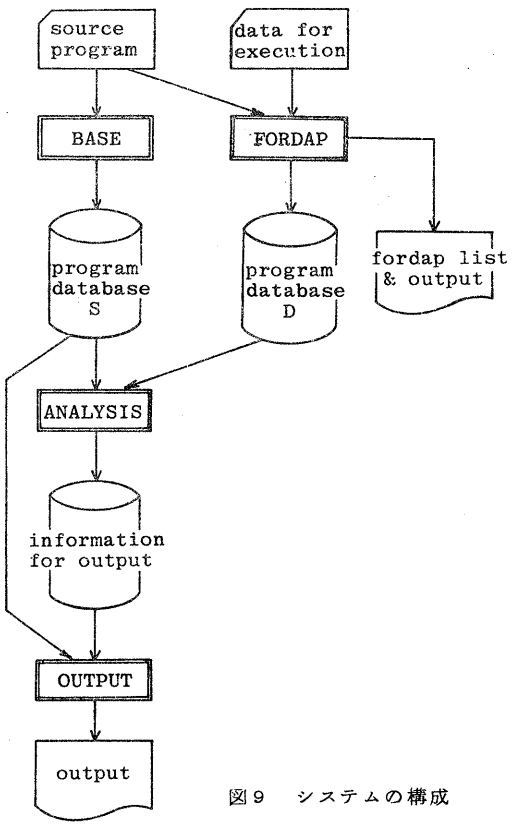


図9　システムの構成

4．使用例

既に述べたように，このシステムでは呼出側手続きと被呼出側手続き，あるいは手続きとデータを表の縦軸と横軸に並べてそれらの間の関係を表示する。従つてこれらが対称な形で出力されることになり，コンパイラの出力等では得難かつた，被呼出側手続きから呼出側手続きを見る，あるいはデータの側から手続きを見るという新しい視野に立つてプログラムを見ることができるようになつた。

以下に，プログラムを保守する立場，テストする立場の2つの面からこのシステムの使用例を示す。

4．1　プログラム保守への使用例

他人の作成したプログラムの保守等を行う場合には，そのプログラムの構造を知る必要がある。ここでは図1～図8を用いて未知のプログラムについて得られる情報について述べる。実際に動かすことのできないプログラムに対しても，図1～図6は出力されるのでかなりのことがわかる。

図1によりこのプログラムは8つの組込関数あるいはライブラリ手続きを使用していることを知る。特に未だ動かしていないプログラムの場合，ソースプログラム中にない手続きがその計算機システム中で使用できることを確認しておく必要がある。

いろいろな手続きから呼ばれている手続きは比較的小さな下請けルーチンであり，いろいろな手続きを呼んでいるものはかなりまとまつた仕事をしていると考えられる。このプログラムでは組込関数のみが前者に該当し，その他の手続きは BLUE を除いては それぞれ唯1つの手続きから呼ばれているという，比較的わかりやすい構造をもつている。

呼出関係において上位の（即ち呼ぶ側の）手続きで宣言されていないコモンブロックは，その上位手続きに対して局所的である。このようなことがあれば注意を要するので図2でチエツクしておいた方がよい。図2によれば，すべてのコモンブロックが主プログラムではなくPINKに組込まれている。また図1によれば，主プログラムは手続きPINKのみを呼出している。従つてPINKは事実上の主プログラムの役割を演じていることがわかる。

	図3	図4
PINK	M	M
GOLD	E	R
GRAY	E	R
VIOLET		X
BLACK	X	X
BLUE	X	X
BLUE	X	X
BROWN	X	X

図10　コモン変数AMASSの使用状況

各手続きと大域的データの関係について概略を知るためには，図3，図4を用いることができる。今，コモンブロックPI中のコモン変数AMASSを考えると，PIを組込む6つの手続きの結合関係は図1により図10のようになる。図3によれば，変数AMASSは手続きBLACK, BLUE, BROWN 中では使用されておらず，手続きGOLD, GRAY中では引数として他手続きに渡され，手続きPINK 中では値を代入されている。図4によればGOLD, GRAYからの呼出を含めてもAMASSは参照されるのみである。従つてAMASSはPINK 中で値を与えられ，PINKから直接呼ばれるGOLD, GRAYがその値を参照し，何らかの処理を行う。なお，図3ではPIを組込んでいない手続きVIOLETが図4ではこれを組込んでいるとしているのは，その呼出関係の上下の手続きが共にこれを組込んでいるために暗黙に組込まれていると見なすからである。

実行の集中は，図8における手続きREDで使用される各変数のアクセス回数から推察できる。FORDAPの出力結果を参照するとプログラムの実行時間の約3/4がこの手続きに集中しているので，REDの内部を局所的に手直しすることによつて実行時間の短縮ができた。さらに，REDは手続きYELLOWのみから95回呼ばれている（図7参照）ので，もしこの回数を減らすことができれば著しい効率改善が期待できる。しかしそのような修正はプログラム全体にわたる可能性がある。 REDがYELLOW内2ケ所に現れていること(図5参照)，REDの仮引数やRED中に組込まれているコモン変数が参照されるだけ（図3，図4参照）である等の情報が，手直し（の可否を含めた）作業の参考になる。

4．2　テスト・デバッグへの使用例

図11及び図12は，あるプログラムに対して3節であげた図5，図7に対応する出力である。図13はこのプログラムの呼出における階層構造を木状に表示したものであつて，図11より求まる。

手続きOVFSは図11からわかるように9ケ所，

7つの手続きから呼ばれている。このプログラムでは実行が進むに従つていくつかの配列にデータを蓄積してゆくが，実行時のデータの内容によつては配列がいつぱいになつても処理が終らないことがある。OVFSはいつぱいになつた配列になおデータを追加しようとした時に呼ばれる手続きである。従つてこのプログラムが正常に終了する場合にはOVFSは1回も呼ばれることがない。

配列がいつぱいであると，その処理を行つていた手続きは配列名を引数としてOVFSを呼ぶ。OVFSは配列名を伴つたオーバフローメッセージを出力する。従つて一般には，メッセージを見てその配列を大きくしてやればプログラムは正常に動くようになる。

しかし，何らかの理由で（例えば無限ループに入る等）必然的に配列を食いつぶしてしまうこと

CALLING CHART

CALLED	BASE	CLTBL	EXTNLP	EXTNLS	GVBLKS	NDLINK	OVFS	PARSER	UNITS	USARCH	MAINPR
BASE											1
BSINIT	1										
BSOPEN	1										
BUFOUT								6			
CKIND								1			
CLTBL			1						1		
CONST								1			
EXTNLP	1										
EXTNLS			3								
GVBLKS			1						2		
MEMCL									2		
NDLINK	1										
OVFS			1	1	1	1		1	2	2	
PARSER	1										
UNITS	1										
USARCH			2						2		
VDCLR								2			
VEXEC								8			
MAINPROG											

図11　静的呼出回数

```
MAINPROG
  BASE
  -BSINIT
  -BSOPEN
  -EXTNLP
    -CLTBL
    -EXTNLS
        OVFS  (1)
    -GVBLKS
        OVFS  (2)
    -OVFS
    -USARCH
        OVFS  (3)
  -NDLINK
      OVFS
  -PARSER
    -BUFOUT
    -CKIND
    -CONST
    -OVFS
    -VDCLR
    -VEXEC
  -UNITS
    -CLTBL
    -GVBLKS
        OVFS  (4)
    -MEMCL
    -OVFS
    -USARCH
        OVFS  (5)
```

図13　階層的呼出グラフ

CALLING CHART

CALLED	BASE	CLTBL	EXTNLP	EXTNLS	GVBLKS	NDLINK	OVFS	PARSER	UNITS	USARCH	MAINPR
BASE											1
BSINIT	1										
BSOPEN	6										
BUFOUT								0			
CKIND								46			
CLTBL			159						150		
CONST								47			
EXTNLP	6										
EXTNLS			369								
GVBLKS			27						70		
MEMCL									166		
NDLINK	6										
OVFS			0	0	18	0		0	0	11	
PARSER	272										
UNITS	6										
USARCH			33						41		
VDCLR								246			
VEXEC								10			
MAINPROG											

図12　動的呼出回数

も考えられる。このような徴候がある場合にはどこでOVFSが呼ばれたかが問題になる。図12によればOVFSはGVBLKSから18回，USARCHから11回呼ばれ，EXTNLP, EXTNLS, NDLINK, PARSER, UNITSからは，呼ばれる可能性はあるが実際には呼ばれていないことがわかる。

このプログラムでは，配列オーバフローを検出してOVFSが呼出された後，制御がBASEに戻ると，そこで計算が停止するようなアルゴリズムになつている。従つてOVFSが複数回呼出されているのはその間BASEに制御が戻つていないことを意味する。ところで，図13において(1)の呼出ルートからは1回も呼出されていない(図12参照)。実際にEXTNLPのソースプログラムを調べてみると，(2)と(3)のルートが実行されて(1)がされないということは起こり得ないことがわかる。従つて異常はUNITSの側のルート(4)(5)で発生したことがつきとめられる。

5．おわりに

このシステムはportable FORTRAN [2] (カード枚数＝約4000,ステツブ数＝約2700，ただし実行回数計数部を除く)で記述され，ファイル編成はすべて順編成である。従つて他の計算機への移しかえも容易に行なえる。九大情報工学科のFACOM230-45Sではシステムの大きさは約120キロバイト，CPU時間は約2000枚(手続き＝35,コモンブロツク＝9，注釈行を含む)のソースプログラムに対して約260秒である。これを九大大型計算機センターのFACOM-M200に移しかえたところ，約180キロバイト，CPU時間は上記と同じデータに対して約5.6秒(入力部が3.0秒，解析部と出力部とで2.6秒)である。またPFORT verifierのソースプログラム(カード枚数＝約8000，手続き＝99，コモンブロツク＝23)に対しては30.5秒(12.4+18.1)である。

このシステムでは，手続き間情報に関して，静的解析結果と実行回数情報をもとに，それらを関連付けて大局的に把握しやすい形で出力することを目的とする。従つて出力は膨大なデータの要約という形で与えられ，より詳しい情報を得たい場合には別の手段を考えねばならない。これには相互参照表，FORDAPの出力リスト，DAVEやPFORT verifierの出力，あるいはソースリストそのもの等を用いることができる。しかし一般には，詳細な情報が必要なのはごく一部であることが多い。それも，全体の概略がわかつたうえでどの情報をより詳しく知りたいかを指定する方法がより効果的である。

このようなことを可能にするためには，システムと会話を進めながら必要な情報を得る必要がある。このシステムで作成されたプログラムデータベースにはそのような情報が各種蓄積されているので，これをもとに会話的により詳しい情報を探索し出力するシステムを作成中である。

参考文献

1. L. J. Osterweil & L. D. Fosdick, "DAVE: A Validation Error Detection and Documentation System for Fortran Programs," Software-Practice and Experience, vol.6, pp.473-486,1976.
2. B. G. Ryder, "The PFORT Verifier," Software-Practice and Experience, vol. 4, pp.359-377,1974.
3. 牛島和夫, "Fortranプログラミングツール," 産業図書，1979.
4. D. E. Knuth, "An Empirical Study of FORTRAN Programs," Software-Practice and Experience, vol.1, pp.105-133, 1971.