E1　汎用グラフィック言語UNGL

山本欣子，出羽　洋（（財）日本情報処理開発センター）

1. UNGLの概要

UNGL（Uniuersal Graphic Language）はグラフィック・オペレーティング・システム（CGOS）の下で開発を進めている汎用グラフィック言語である。UNGLはFORTRANをベース言語とするプリプロセッサ形式の言語で，FORTRANに次の様な面で機能的拡張を加えたものである。

グラフィック・データの導入とそれに関する処理機能，グラフィック入出力機能，割り込み処理に関する機能，汎用データ・ストラクチャへのアクセスの手段。

また設計にあたって特に留意した事は次の様な点であった。

① 汎用性があること，すなわち特定のアプリケーション向きの言語でなく，なるべく広範囲のアプリケーション・プログラムが実現できる様な言語であることが望ましい。

② グラフィック・システムの持つ機能が十分発揮できること。

③ マシン・インディペンデントであること。グラフィックスのハードウェア，ソフトウェアがまだ流動的であることを考えると，なかなか設定しにくい目標である。少なくとも特定のハードウェア，ソフトウェアの存在を前提とするような機能はとり入れるべきでなく，ごく一般的なグラフィック・システムで実現できる様な言語であることが望ましい。

④ ハイレベルであること。グラフィック言語においては特に，図形データ，関連データ，割込み処理等，一般のプログラミング言語からみると，新しい概念を必要とするものが多く，その扱いも複雑である。使い易さを前提とするには，これらの扱いに関する概念を明確にし，適当な表現形式を考える必要がある。

UNGLのこれらに対する扱いに関して，特徴的な点を以下に記す。

a）図形データの扱いに関して

① 図形データの概念が明確である。

図形データを表現するデータ・タイプとして新たに4つのデータ・タイプ，イメージ，イメージ・データ，図形変換マトリックス，論理化因子が設けられている。これらはいずれも図形に関する量を表現し，その概念上の区別も明確である。

② 3次元図形の扱いが可能である。

3次元イメージによる3次元図形の扱いも可能である。

③ 図形操作が容易である。

図形操作を表わすいくつかのオペレータがあり，プログラム上で自由に図形操作を表現することができる。

④ 表現形式が優れている。

複雑な図形も，イメージ・エクスプレションの中に端的に表現される。

⑤ 画面管理の機能がある。

画面はフレームという単位で管理され，保存しておくことができる。

⑥ 各種の画面操作が可能である。

フレームに出力された図形に対して，各種の操作を加えることができる。

b）割り込み処理に関して

① プログラム・ステートの概念で取り扱うことができる。

割り込み処理過程をプログラム・ステートの遷移という形式で表現することができる。また従来のステート・ダイヤグラム・アプローチではプログラム・ステートがスタティックに定義されるのに対し，UNGLではこれを動的に再定義することも可能である。またシステムの保有する状態をステート0として扱うことにより，従来のON文等による割り込み処理の概念もそのまま通用する形になっている。

② 非同期処理が可能である。

リエントラント構造を備えた割り込み処理ルーチンに対して非同期処理が認められる。

c）データ・ストラクチャ操作に関して

① LEAP タイプの連想記憶機構へのアクセスが可能である。

② 論理化因子により，図形構造と問題用データ構造の関連がつけられる様になっている。

2. UNGLの機能

2.1 図形データ処理

A．図形データの扱いに関する機能

a）グラフィック・データ

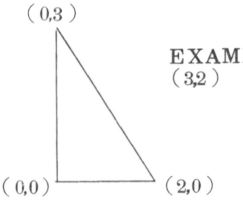
UNGLが扱うグラフィック・データには4つのタイプ，すなわちイメージ，イメージ・データ,図形変換マトリックス，論理化因子，がある。またこれらのグラフィック・データ間に適用されるいくつかのグラフィック・オペレータがあり，図形操作，図形の構造化はグラフィック・データ，グラフィック・オペレータによるイメージ・エクスプレションとして表現される。以下にこれらの関係について概略を述べる。

① イメージ

イメージは構造化，図形操作の対象となる基本的な図形要素（図形のまとまり）を表わす。イメージの実体は次に述べるイメージ・データによって定義されるが，これを既に定義されている他のイメージを引用して，階層的に記述することもできる。しかしユーザは特に階層構造を意識する必要はない。この様にイメージはサブピクチャとか図形モデルと呼ばれるものに近い概念を表わす。またイメージには2次元図形を表わす2次元イメージと，3次元図形を表わす3次元イメージがある。

② イメージ・データ

イメージ・データは2つのグラフィック・ファンクションLINE，TEXTにより定義され，イメージに蓄えられる実際の図形データとして，それぞれ線分，文字ストリングを表現する。例えば，〔図1〕に示された図形はイメージ PICTURE として次のように定義される。



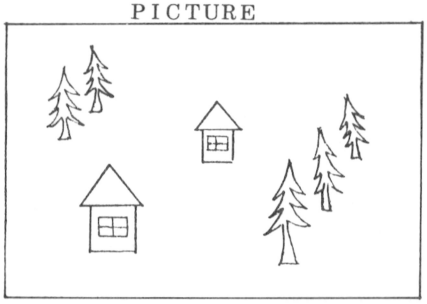
〔図1〕

PICTURE<=❜LINE(0,0,2,0,0,3,0,0),TEXT(❜EXAMPLE❜,3,2)❜

ここで <= は，イメージの代入操作を表わすグラフィック・オペレータである。UNGLではこの他に次のようなグラフィック・オペレータがある。

グラフィック・オペレータ	操作
＜＝	代入操作を表わす
＋	併合操作を表わす
.TRS.	変換（拡大縮少，回転，平行移動）操作を表わす
.PERS.	透視変換操作を表わす
.ORTH.	平行投影操作を表わす
|	論理的なグルーピングを表わす

〔図－２〕は既に定義されているイメージHOUSE，TREEを引用してイメージPICTUREを階層的に表現したものである。



〔図－２〕

HOUSES ＜＝HOUSE. TRS.（M1）
＋HOUSE. TRS.（M2）
TREES ＜＝TREE. TRS.（M3）
＋TREE. TRS.（M4）＋……
PICTURE ＜＝HOUSES＋TREES

注）（Mi）は後述する変換マトリックスを表わす。

③ 図形変換マトリックス

図形変換マトリックスは，イメージに対して各種の図形変換を行う時に，その変換量を表わすデータとして使用される。これには次の３種類のマトリックスがある。

㋐ 図形変換マトリックス

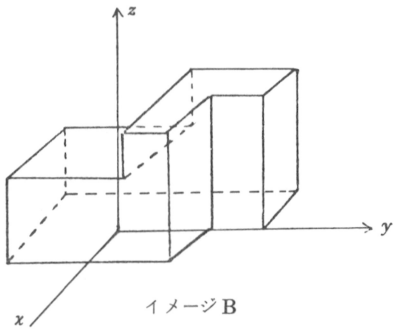
イメージに対する図形の拡大縮少，回転，平行移動に必要な情報を蓄えている。２次元イメージに対する２次元図形変換マトリックスと３次元イメージに対する３次元図形変換マトリックスがある。

㋑ 透視変換マトリックス

３次元イメージを透視する時の投象面（$x-y$平面，$y-z$平面，$z-x$平面のうちいずれか），視点の座標を与える。

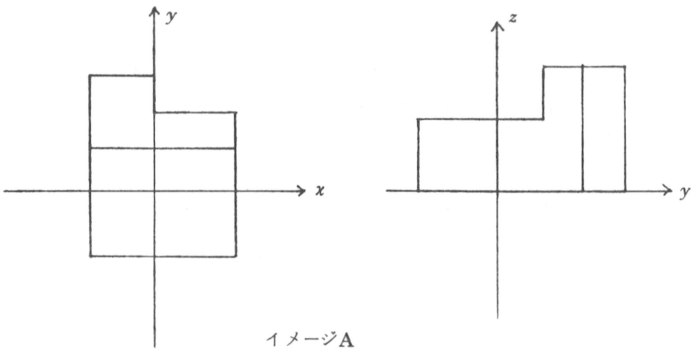
㋒ 平行投影マトリックス

３次元イメージを平行投影する時の投象面を与える。これらのマトリックスは，それぞれグラフィック・オペレータ.TRS.，.PERS.，.ORTH.によってイメージに作用させる。例えば〔図－３〕のイメージBを$x-y$平面，$y-z$平面に平行投影すると〔図－４〕，〔図－５〕のようになる。ここで(1)，(2)はそれぞれ$x-y$平面，$y-z$平面を表わす平行投影マトリックスである。このように平行投影の機能は３次元物体の三面図を得る時などに利用できる。



〔図－３〕

A<=B. ORTH.(1)　　A<=B.ORTH.(2)



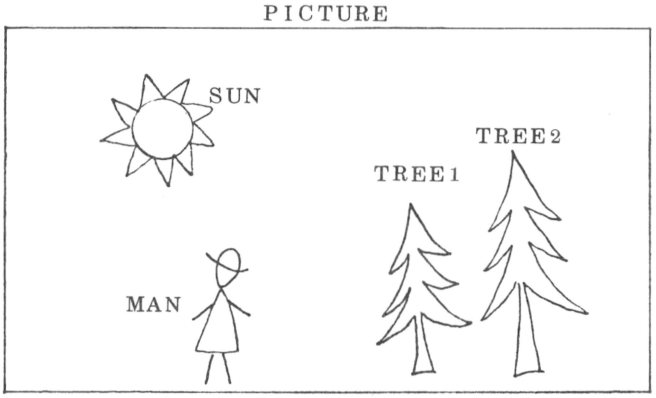
イメージA

〔図－4〕　〔図－5〕

④　論理化因子

一般に，問題解析を含んだ高度なグラフィック・アプリケーション・プログラムにおいて，画面からの図形を仲介とするユーザ・インタラクションは，問題モデルを表現する問題用データ・ストラクチャへの直接のアクセスを必要とする。これにはあらかじめ個々の図形要素と，図形モデルを表現する関連データ（アイテム）との間に何らかのつながりが保たれていなくてはならない。

UNGLでは，論理化因子およびグルーピング・オペレータを用いて，任意の図形エクスプレションに対して，それを論理的な単位とし，それに論理名称を与え，関連データとの結びつきが可能になるように考慮している。すなわち，論理名称にはアイテム名称そのものを与えることができるから，ライトペン・ディテクト等により，図形が指摘されるとそれによって返されるインターナル・ネームで直接プログラム・データ・ストラクチャにアクセスする形式でプログラム表現することが可能である。



〔図－6〕

以上に述べたように，論理化因子は図形の幾何学的表現に対し，グルーピング・オペレータで付加され，それを論理的なまとまりとする働きをする。またこれ以外に，図形の非幾何学的な特質（輝度，ライトペン条件）を記述する働きもかね備えている。

次に論理化因子の具体的な使用例をあげる。〔図－6〕のようなイメージPICTUREが次のように定義されていたとする。

PICTURE <=MAN+SUN+TREE1+TREE2

この時，次の3つの方法でこれをグルーピングした時，PICTUREが画面表示された時の違いについて述べる。

㋐ PICTURE <=(MAN+SUN+TREE1+TREE2)|<1>

㋑ PICTURE <=(MAN+SUN)<1>+(TREE1+TREE2)|<2>

㋒ PICTURE <=MAN|<1>+SUN|<2>+TREE1|<3>+TREE2|<4>

どの場合もPICTUREが表示されると〔図－6〕の絵がCRTに表示される。

㋐の場合は，PICTURE全体で一つの識別対象になっている。すなわちSUN，MAN，TREE1，TREE2 のどれが指摘されても，全く同じレスポンス▼1▼ が返ってくる。

㋑の場合は，MANとSUN，TREE1とTREE2が各々一まとまりで画面は2つの識別対象からなっている。例えばMANかSUNが指摘されるとレスポンス▼1▼が返り，一方TREE1かTREE2が指適されるとレスポンス▼2▼ が返ってくる。

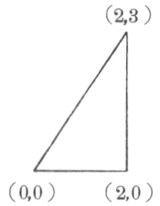
㋒の場合は，SUN，MAN，TREE1，TREE2 が各々1つの識別対象であり，画面は4つの識別可能な図形から構成されている。すなわちMANが指摘されるとレスポンス▼1▼ が返り，同様にSUN，TREE1，TREE2に対してはレスポンス▼2▼，▼3▼，▼4▼ が返される。

b）イメージ・エクスプレションとデータ・ストラクチャ

これまでに見てきたように，イメージは単にイメージ・データによって定義されるだけでなく，他のイメージを図形モデルとして引用したり，これに適当な操作を加えたものを引用したりしながら次々に複雑なイメージとして定義していくことができる。このようなグラフィック・データとグラフィック・オペレータによるイメージの表現形式をイメージ・エクスプレションと呼ぶ。

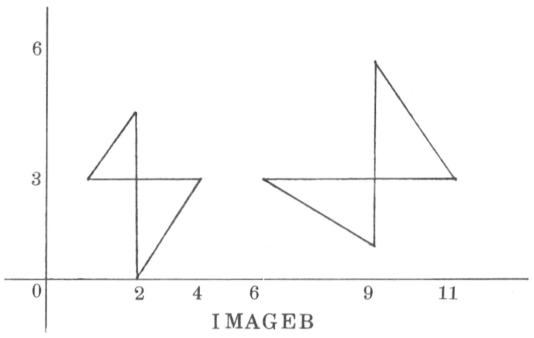
イメージ・エクスプレションは，次の例にみるように内部的にはトリー構造として表現されている。

〔図－7〕のように，既に定義されているIMAGEAに変換をほどこした図形を〔図－8〕のようなIMAGEBと定義する。



IMAGEA

〔図－7〕



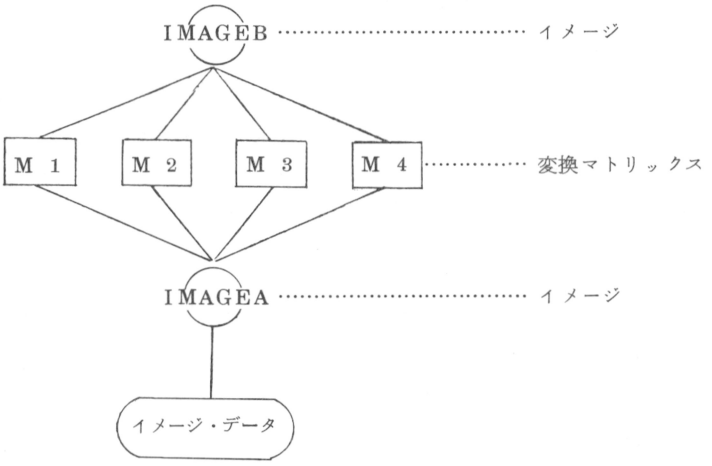
IMAGEB

〔図－8〕

IMAGEB は次のようなイメージ・エクスプレションとして表現される。

```
IMAGEB <= IMAGEA. TRS. (1, 3, 0, 0.5, 0.5)
        +IMAGEA. TRS. (4, 3, 180)
        +IMAGEA. TRS. (9, 6, -90, 1.5, 0.667)
        +IMAGEA. TRS. (9, 1, 90)
```

また〔図－9〕はその内部表現を示したものである。



〔図－9〕

B. 図形入出力

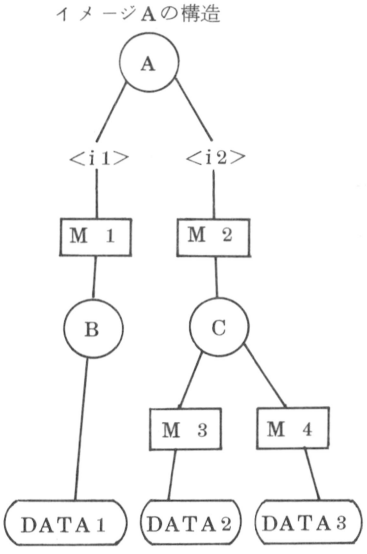
a) 表示データ・リスト

表示はイメージを対象にして行われる。イメージを指定して出力ステートメントが実行されると，システムはイメージ構造をたどり，各種の図形変換を行いながら最終的に次の形式の表示用図形データを作成する。これを表示データ・リストと呼ぶ。

論理化因子(i)	変換マトリックス (M)	図形データ	論理化因子(i')	変換マトリックス (M')	図形データ	

〔図－10〕表示データ・リストの概念図

ここで〔図－11〕はイメージ構造と，それから作成される表示データ・リストの関係を示したものである。



イメージAを指定して表示命令を実行した時の
表示データ・リスト

i 1	M1	DATA1	i 2	M5	DATA2	M6	DATA3

$M5 = M2 \times M3$
$M6 = M2 \times M4$

○ イメージ
□ 変換マトリックス
⬭ イメージ・データ
< > 論理化因子

〔図－11〕

表示データ・リストはまだ完全な表示データではない。すなわちこれを表示コマンド・オーダに変換するまでには次のような操作が残されている。

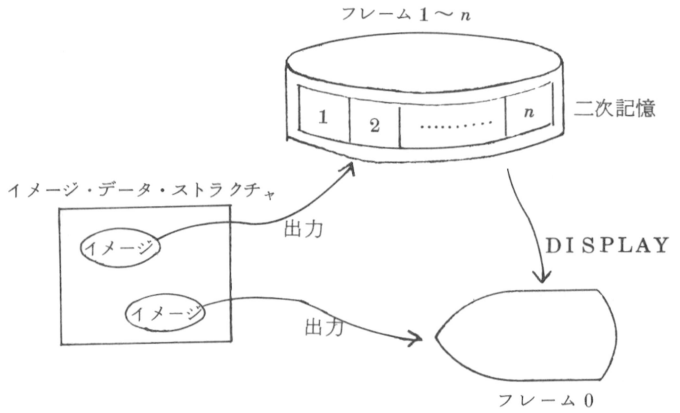
① 変換マトリックスと個々のデータ間の演算

② 論理化因子に従って，コリレーション・リストと表示コマンドをジェネレートする。

b) フレームの概念

イメージが図形として出力される媒体をフレームと呼ぶ。フレームは0～n（整数）で表わされる識別名を持ち，各々CRT画面の1画面に相当する表示データを蓄えている。

一担イメージがフレームに出力されると，フレーム上の図形データは論理化因子で指定されたまとまりで管理される表示データ・リストに変換され，イメージ構造は失われる。従ってフレーム上の図形を識別するには，論理化因子で与えた論理名を用いなくてはならない。論理化因子で与えた論理名で識別される図形のまとまりをエンティティ（インスタンス）と呼ぶ。フレーム0はディスプレー・ファイルそのものに相当する。すなわちフレーム0に出力することは画面表示することである。フレーム0に対して為された図形の動的な変更に対してそのヒストリーを残すことはできない。一方フレーム1～nは表示されない画面で，具体的にはストレージ・デバイスへの出力を意味する。フレーム1～nに対して為された出力はパーマネントなものであり，必要な時点まで残しておくことができる。またフレーム1～nを画面表示するためには，後述するDISPLAYステートメントが使用される。



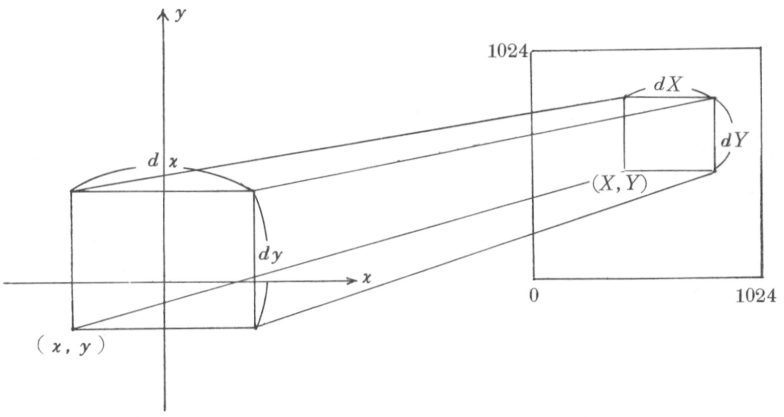
〔図－12〕フレームの概念図

c) グラフィック入出力ステートメント

図形出力は，「あるイメージをあるフレームに出力する」という形式で行われる。またフレームに出力された図形に対しては，論理化因子によってまとめられたエンティティの単位で修正を加えることができる。具体的な修正動作として，置換，削除，CRT画面制御情報の変更等がある。

① イメージの出力

イメージを出力する時には，「イメージの座標系のある領域を，フレーム座標系のある領域に表示する」という形でウインドイング情報を与えることができる。この時，イメージの座標系の範囲は制限されていないが，フレームの座標系は1024×1024の正方形の座標系とする。次にこれらの関係を〔図－13〕に示す。

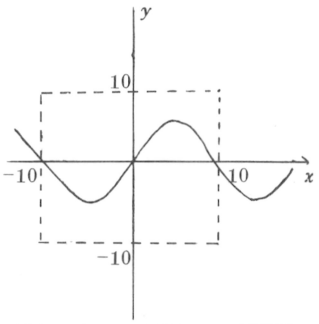


イメージの座標系　　フレームの座標系

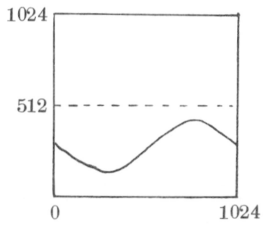
〔図－13〕

例えば〔図－14〕のように定義されたIMAGEAにウインドイングを施して，フレームに出力すると〔図－15〕のようになる。

DRAW φ, IMAGEA, (−10, −10, 20, 20, 0, 0, 1024, 512)



〔図−14〕IMAGEAの座標系



〔図−15〕フレーム0の座標系

〔図－14〕

この他に特殊な機能として，簡単なアニメーションを行うANIMATEステートメントが用意されている。

② フレーム内の図形消去

フレームに描かれた図形はエンティティ単位で消去することができる。フレーム0を指定した場合はCRT画面上の図形が消え，フレーム1～nを指定した時はそのフレーム内のそのフラブに対応するディスプレー・オーダが取り除かれる。例えば次のようにするとITEM1を名称とする図形が画面から消える。

DELETE φ, ＜ITEM1＞

③ 制御情報の変更

一担フレームに出力された図形に対して，その制御情報を動的に変更することができる。例えば，

CONTROL n, ＜▼FLAG▼, I＞

とすると，フレームnの中の▼FLAG▼という名称をもつエンティティの制御情報がIに変更される。

④ フレーム操作

二次記憶上のフレーム1～nをCRT画面上に表現することができる。これにはDISPLAYステートメントが使用される。例えば，

DISPLAY　K

とすると，フレームKをCRT画面上に表示する。CRT画面上になんらかの図形が表示されていても重ね書きをする。CRT画面に表示された図形はすべてフレーム0の図形とみなし操作される。

CRT画面をクリアする時には，RESETステートメントを用いる。また二次記憶上のフレーム1～nをクリアするにはフレームIDを指定する。例えば，

RESET　4

とすると，フレーム4がクリアされる。クリアされた領域はフリー・ストレッジとしてシステムに返される。

2.2　割り込み処理

UNGLでは，次の2つの形式で割り込み処理過程を表現することが可能である。

① ONステートメントにより直接システムに対し，割り込み動作に関する指示を与える。

② あらかじめプログラム・ステートを定義しておき，プログラム・ステートの遷移という形式でこれを表現する。

①は通常の割り込み処理の概念にあてはまるもので，ON文により割り込み制御に関する情報を登録する方法である。②では直接割り込み制御表を意識せずに，これをプログラム・ステートという概念で扱う方法である。特に後者は複雑な割り込み処理過程を表現する時に，その流れをマクロに把握できるという意味で優れた表現形式を与えるだろう。

a）プログラム・ステート

その時に許されるアクション，リアクションの関係で性格づけられるプログラムの状態をUNGLではプログラム・ステートと呼ぶ。

プログラム・ステートはユーザによって定義され，その遷移のコントロールを受ける。ユーザ・プログラムがイニシェートされた時プログラム・ステートは「状態0」にあるという。「状態0」はシステムが定義するプログラム・ステートで，その初期状態においてあらゆる割り込み要因に対して割り込み不可の状態にある。プログラム・ステートはスタティックに定義され，プログラムの実行に先だってシステムに通知されるが，実行時にその内容を変更することも可能である。前述した①の形式は「状態0」を動的に再定義するという形式で実現されることになる。

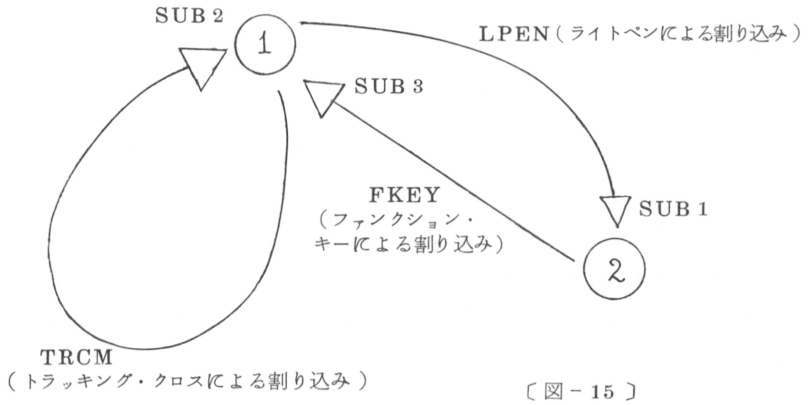
b）プログラム・ステートの定義

プログラム・ステートはDEFINE文からENDS文の間で定義され，STATE文により識別番号（ステート番号）が与えられる。一つのプログラム・ステートはSTATE文から次のSTATE文，またはENDS文の間で定義され，割り込み要因とその処理ルーティンの関係，および処理後のプログラム・ステートの遷移に関する情報が与えられる。

次に状態1，状態2を定義する例を示す。

```
DEFINES                          ―――― プログラム・ステートの
                                      定義開始を示す
    STATE1                       ―――― 状態1の定義
      ON LPEN DO SUB1, 2
      ON TRCM DO SUB2  1
    STATE2                       ―――― 状態2の定義
      ON FKEY DO SUB3, 1
ENDS                             ―――― プログラム・ステートの
                                      定義終了を示す
```

これをステート・ダイヤグラムで表現すると次のようになる。



〔図－15〕

ここでON文は，割り込み要因と処理ルーティン名，復帰後の状態を与える。

c）プログラム・ステートの遷移

割り込み処理過程の遷移はプログラム・ステートの遷移という形式で表現される。プログラム・ステートの遷移を指示する方法は2通りある。1つは前述のプログラム・ステートの定義の中でのON文のオプションとして復帰後のプログラム・ステートを指示する方法である。もう1つは実行文TRANSFERによる方法である。これはステート番号を指示することにより，プログラム・ステートを実行時に自由に遷移させることができる。

d）プログラム・ステートの動的な変更

プログラム・ステートに関する情報は実行時に先だってスタティックに定義されるが，これを実行時に動的に修正することも可能である。これは特にプログラム・ステートの遷移という形式をとらずに割り込み処理過程を表現する時にも必要となる。

2.3 データ・ストラクチャ操作

関連データの扱いに関しては，特に一般性を考慮してLEAPタイプの連想記憶機構を採用している。連想記憶は関連表現において優れた力を持っている。すなわち，その記憶機構に対してユーザは全く気を配る必要がないということである。例えば従来のリング構造においては，一般に「どのアイテムをどのアイテムのリングの何番目に挿入する」というような内部構造を意識した関連の表現形式が必要とされたが，連想記憶機構では単に関連の本質的な部分だけを述べれば十分である。

UNGLでは，関連を表現する上で基本となるアイテム，セットを扱い，機能的にもほぼLEAPに近いものを考

えた。

参考文献

1. An experimental display programming language for the PDP−10 computer
   by William M. Newman October, 1969.
2. GRAF : Graphic Additions to FORTRAN
   by Carl Christensen and Elliot N. Pinson FJCC, 1967.
3. CPL／I : A PL／I extension for computer graphics
   by David N. Smith SJCC, 1971.
4. The LEAP language and data structure
   by Paul D. Rouner and Jerome A. Feldman IFIP, 1968.
5. An Algol−based associative language
   by Jerome A. Feldman and Paul D. Rouner ACM Vol 12, No.8 August, 1968.

本 PDF ファイルは 1965 年発行の「第 6 回プログラミング―シンポジウム報告集」をスキャンし、項目ごとに整理して、情報処理学会電子図書館「情報学広場」に掲載するものです。

この出版物は情報処理学会への著作権譲渡がなされていませんが、情報処理学会公式 Web サイトの https://www.ipsj.or.jp/topics/Past_reports.html に下記「過去のプログラミング・シンポジウム報告集の利用許諾について」を掲載して、権利者の捜索をおこないました。そのうえで同意をいただいたもの、お申し出のなかったものを掲載しています。

過去のプログラミング・シンポジウム報告集の利用許諾について

情報処理学会発行の出版物著作権は平成 12 年から情報処理学会著作権規程に従い、学会に帰属することになっています。

プログラミング・シンポジウムの報告集は、情報処理学会と設立の事情が異なるため、この改訂がシンポジウム内部で徹底しておらず、情報処理学会の他の出版物が情報学広場 (=情報処理学会電子図書館) で公開されているにも拘らず、古い報告集には公開されていないものが少からずありました。

プログラミング・シンポジウムは昭和 59 年に情報処理学会の一部門になりましたが、それ以前の報告集も含め、この度学会の他の出版物と同様の扱いにしたいと考えます。過去のすべての報告集の論文について、著作権者（論文を執筆された故人の相続人）を探し出して利用許諾に関する同意を頂くことは困難ですので、一定期間の権利者捜索の努力をしたうえで、著作権者が見つからない場合も論文を情報学広場に掲載させていただきたいと思います。その後、著作権者が発見され、情報学広場への掲載の継続に同意が得られなかった場合には、当該論文については、掲載を停止致します。

この措置にご意見のある方は、プログラミング・シンポジウムの辻尚史運営委員長 (tsuji@math.s.chiba-u.ac.jp) までお申し出ください。

加えて、著作権者について情報をお持ちの方は事務局まで情報をお寄せくださいますようお願い申し上げます。

期間：2020 年 12 月 18 日 〜 2021 年 3 月 19 日
掲載日：2020 年 12 月 18 日

プログラミング・シンポジウム委員会

情報処理学会著作権規程
https://www.ipsj.or.jp/copyright/ronbun/copyright.html