論　文

グラフ処理システム GMS とその応用*

真野芳久** 杉藤芳雄** 鳥居宏次**

Abstract

An Interactive Graph Manipulation System-GMS is described which is implemented on the basis of Graph Manipulation Language GML. It is mainly to manipulate a given graph conversationally, according to a transformation program written in GML. It may be used as a tool for constructing a procedure to solve a problem step by step using computer graphics based on its quick response. Since the user has complete control over GMS's running and can solve problems heuristically making use of its facilities, it may be called as an integrated programming system.

An application of GMS is shown which generates a tree representation for a given plane figure.

1. まえがき

回路網，組み合わせ問題，PERT，有機化学，データ構造，等の様々な分野の様々な問題がグラフで表現され，グラフ理論の用語を使って研究されている．グラフの応用範囲は広いが，その理由として，理論としてほぼ完成されている点の他に，関係を記述するための視覚的にすぐれた表現法であることがあげられる．

このように利用価値の大きいグラフを計算機によって容易に扱えるようにする試みは既にいくつか報告されている．例えば文献 1),2),3) では，ALGOL や FORTRAN あるいは LISP をベースとしたグラフ向き言語について報告している．これらの試みの多くでは，二次元に広がるグラフを一次元に写像する知的作業が要求され，グラフの視覚性の良さという利点が生かされていない．また，これら言語の処理プロセッサはプリコンパイラ形式のものが多く，プログラムの変更やグラフの入出力時の表現形式の変換等のためのプログラミング支援機能は不十分である．

本稿では，グラフィック・ディスプレイの利用によりこれらの欠点を補い，かつその利用形態を簡潔にすることにより，グラフ本来の特徴を生かしつつ容易に計算機を利用することが可能になることを目的として設計された言語 GML に基づくシステムとその応用について述べる．また，システム作成と使用経験をもとにして，この種の言語や言語システムについての一般的な検討も行う．

GML (Graph Manipulation Language)[4] は，対象とする入力グラフやグラフ変換のための規則を二次元的に表現できる言語である．またそのプログラムは，グラフの変換を行う特殊な形のブロックを含むフローチャート方式によるものである．このように GML はプログラム及び扱うデータ共に二次元図形であり，GML をインプリメントすることによって，人間の創造力，直観力が視覚の面からも更に助長されることになると考えられる．

GML の1つのインプリメントである本稿で述べる GMS (Interactive Graph Manipulation System) は，会話型システムであり，更に利用者の便宜を計るために，問題解決システムとしての機能，プログラミング・システムとしての機能を持っている．

2. GML 概略

グラフを対象とするように句構造文法を二次元に拡張した1つにウェブ文法[5] がある．GML は，思想としてはウェブ文法に制御構造を付加してプログラミン

---

* An Interactive Graph Manipulation System GMS and Its Applications by Yoshihisa MANO, Yoshio SUGITO and Koji TORII (Computer Science Division, Electrotechnical Laboratory).

** 電子技術総合研究所ソフトウェア部

グ言語としての機能を持たせたものと考えられる．GML におけるグラフ処理は，グラフの置換，即ち，指定されたグラフと同型な部分グラフを入力グラフ中に発見し，その部分グラフを他の指定されたグラフで置き換える，という操作を中心にしてなされる*．

GML プログラムでは，ルールグラフが二次元的な線グラフとして記述されるのみならず，制御の流れもフローチャート形式のブロック線図によって表現される．これらのブロックはその形状によって処理の内容が定められている．GML の構成要素を付録Ⅰに示す．GML はグラフの置換の記述の他に，テキストのステートメントの記述を許している．

GML の対象とするグラフは，節点にラベルを持つグラフであり，ラベルは $X(a_1, a_2, \cdots, a_n)$ の形をなす．ここで X は文字列，a_i は整数である**．ルールグラフの場合，ラベルの a_i 部分にはラベル変数と呼ばれる変数を置くことができる．ラベル変数は，確定あるいは未定の2つの状態のいずれかをとる．確定状態のときは，特定の整数値を指定するものとみなされる．未定状態のラベル変数は特定の値を指定せず，これはマッチング時に don't care の意味で用いられる．しかし，マッチングが成功した場合には，確定状態へと移行し，入力グラフ中の対応する値を指定するようになる．

文献4) では GML の仕様が詳細に述べられ，その能力が検討されているので，ここでは詳細を省略する．

3. GMS 概要

GML の視覚性にすぐれた点を生かし，解法についての発想を速かに実現し実行されることが可能となるように，GMS は設計，開発された．GMS の主な特徴としては，

1) プログラミング支援システムとしての機能を持つ．特に，会話型システムとして，プログラムの作成から完成まで利用者に様々な便宜を与えている．
2) 試行錯誤的方法や発見的思考による解決法に適している．
3) グラフ処理をダイナミックなグラフ変換としてとらえることを可能にしている．
4) 二次元図形表示のためグラフィック・ディスプレイ（ライトペン方式）を端末として用い，コマンドの指示や実行のモニタ等にもその利点を生かしている．

GMS で扱うデータを **Table 1** に示す．また，GMS を使用するときの処理内容とその流れを **Fig. 1** に示す．これらについての詳細は **4.** で述べる．

Table 1 Data treated in GMS

データ		構成要素	意味
GMLプログラム	制御構造	プログラム名，ブロック，ブロック名，流れ線	グラフの変換手続き
	ブロックの内容 文	テキスト	
	ブロックの内容 ルールグラフ	節点，枝，ラベル，対応番号	
骨格グラフ		骨格グラフ名，節点，枝，ラベル	類似のルールグラフを複製するために用いられるグラフ
入力グラフ		入力グラフ名，節点，枝，ラベル	GML プログラムへの入力となるグラフ
出力グラフ		節点，枝，ラベル	GML プログラムの実行によって得られるグラフ
中間グラフ		節点，枝，ラベル	GML プログラム実行中の変換されつつある入力グラフ

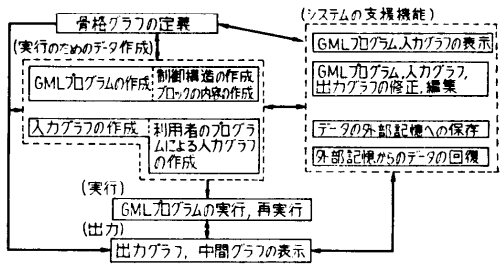


Fig. 1 The flow of processes in GMS

4. 利用者から見た GMS

4.1 会話型システムとしての GMS

GMS の利用者にとっては，グラフィックスの画面のみが相手であり，コマンドの指示や画面上の図形の作成，修正等は，グラフィックス装置に備え付けられたライトペン，ファンクション・キー，タイプライタを用いる．しかし多くの場合は，メニュと呼ばれるグラフィックス上に表示されているコマンド，あるいは図形の一部をライトペンで指示するのみで，自然で迅速な操作ができるようになっている．グラフィックスはまた，GML プログラムの実行結果を直ちに図形としてのグラフとして表現したり，GML プログラムの実行経過をモニタする道具としても使われる．

* これらの操作をそれぞれマッチング，エンベッディングと呼び，指定されるグラフをそれぞれマッチング・グラフ，エンベッディング・グラフと呼ぶ．両グラフを総称してルールグラフと呼ぶ．従ってマッチングという言葉は通常のグラフ理論における使われ方と異なっている点に注意．

** GMS では，X は1文字，n は3以下と制限している．

Table 2 Appended blocks

ブロック	ブロック名	ブロックの内容	ブロックの働き
ERR	エラーブロック	ERR	プログラムの実行を終了させエラーメッセージを表示する
ブレイクポイントブロック		整数	プログラムの実行を中断して中間グラフを整数とともに表示する

更に会話型システムとして充実させるためには，GML 言語だけでは不十分であり，その仕様が拡張された．**Table 2** に見られる2種類のブロックの追加である．特にブレイクポイント・ブロックで実行が中断される場合に中間グラフが表示されるが，この時点でGMS の持つあらゆる機能が，中間グラフや GML プログラムの変更をも含めて，利用できる．この機能と中断された実行を再開させる再実行機能とは，アルゴリズムの開発の道具として大きな役割を果たしている．GML の他の拡張として，中間グラフの種々の数量的性質を求める組込関数の導入がある．これらは**付録II**にまとめられている．

利用者の立場から見たとき，GMS は処理の内容に従って **Table 3** に示すいくつかのモードに分けて考えられる．しかし，GML プログラムを作成する部分は，グラフィックスの画面の大きさの制約から，プログラムの制御構造を構成するモードと，ブロックの内容を書き込むモードに分けられている．実行，再実行モード中にあるときを除き，任意の時点で任意のモードに移行できる．また，そのモードで必要とされるすべての支援機能が利用できる．

4.2 問題解決システムとしての GMS

前節でグラフィックスを前にして利用者が完全に全体を制御できることを GMS の会話型機能として述べた．GMS の応用として，単にグラフのみならず「関係」が重視される一般の問題への拡張が考えられる．

Table 3 GMS's modes

モード	そのモードでなされる内容
declaration モード	骨格グラフが作成される
入力モード	入力グラフが作成される
flow モード	GML プログラムの制御構造が作成される
content モード	ブロックの内容が作成される．ブロックの形状により文あるいはルールグラフがブロックの内容となる
出力モード	出力グラフあるいはブレイクポイント・ブロックによる中断時の中間グラフが表示される
実行モード	指定された GML プログラムを指定された入力グラフを入力データとしてインタプリティブに実行が開始される
再実行モード	ブレイクポイント・ブロックで中断している GML プログラムの実行が再開される

その場合，問題のグラフへのモデル化が重要であり，モデル化の方法によって様々なグラフが得られる．

GMS では，問題のモデル化の部分を発見的に解決するための支援機能として，入力グラフ作成用のプログラムを GMS 内に組み込むことを許している．利用者によって書かれるこのプログラム（モデル化ルーチンと呼ぶ）は，利用者がグラフィックス上に与える数値をパラメータとして受け取る FORTRAN サブルーチンの形をなす．このプログラムでは，定められた形式の隣接行列とラベルのための配列を入力グラフの定義として設定しなければならないことを除いて，任意の実行や入出力を行うことができる．例えば，GML プログラム中で宣言される配列（それが実際に占める記憶領域は利用者側即ちモデル化ルーチン側に開放されている）に，その初期値を設定しておくことが可能である．**6.** で述べる応用例ではこの機能を利用している．

前節で述べたブレイクポイント・ブロックと再実行機能とを活用することにより，グラフが変換されていく様子をダイナミックにとらえることを目的とする場合にも GMS を利用できる．例えば，GML の言語能力からしてウェブ文法のシミュレータとして GML プログラムを作ることは可能であるが，その生成過程をダイナミックにトレースすることができる．

4.3 集約化されたプログラミング・システムとしての GMS

現状ではソフトウェア作成環境としての汎用的なプログラミング・システムの具体的なイメージが確立していない．GMS は，問題向きのプログラミング言語システムとしての1つの試作としてとらえることができる．即ち，グラフに帰着される問題を解決するために，様々な GMS の機能や様々な入出力装置をグラフィックスを前にした利用者がインタラクティブに試行錯誤的に駆使できる．GMS に用意されているこの種の機能には次のものがある．

1) 既に作成されている入力グラフ，GML プログラムの名前をすべて表示させる．
2) GML プログラムの制御構造やグラフの二次元図形を編集するための手段として，節点，枝，ブロック，流れ線を削除したり，節点，ブロックを移動させる．
3) ルールグラフを書き込む時点で，declaration モードで作成した骨格グラフ名を指定すると，その骨格グラフが複製されてブロックの内容となる．

このグラフは修正することが可能であり，類似のルールグラフを多数作成する場合の便宜を与えている．

4) グラフィックス上の図形のハードコピーをとる機能がソフトウェアによって支援されている．

5) 磁気テープあるいはカードによるデータの保存や回復ができる．これはファイル・システムの代用であるが，持ち運び可能であり，GML プログラム等の蓄積や利用者間の情報交換を容易にしている．

5. GMS 内部構造

5.1 データ構造

GMS ではその性質上データの追加，削除が任意の順序でなされる．また，GML プログラム実行時にはデータの検索が頻繁になされることが予想される．ダイナミックなデータ変更が容易で効率の良いデータの検索が可能であるデータ構造の1つとして，LEAP[6]で実現されているような連想三ツ組データ構造がある．この種のデータ構造を支援するものとして我々が利用可能な EDSP (ETL's Data Structure Package)[7]があり，ハッシュ技法を用いたすぐれた検索能力を持っている．GMS は EDSP を用いてインプリメントされている．

EDSP における連想三ツ組は，3つの item の順序組として表現され，各 item は識別子と任意長のデータを持つ．また，三ツ組に識別子を持たせ，これを1つの item とみなすことができる．データの検索は，1〜2 個の item と三ツ組中での位置とを指定して，これらの item をその位置に含む連想三ツ組の他の成分にある item を集合として求めたり，item の三ツ組が連想三ツ組であるか否かを調べる等の方法による．

Table 1 GMS's data and the internal representation

item	識別子の表現	item の持つデータ
入力グラフ	16ビットで表現できる，それぞれ固有の範囲内の数値	なし
ルールグラフ		
入力グラフ名	それ自身	
節点	それを含むグラフの識別子と，グラフィックス上での識別子であるそれぞれ固有の範囲内の16ビットで表現される数値との対	座標，(マッチング・グラフのとき) 節点のタイプ
枝		枝を表現する三ツ組
ラベル	それ自身	なし
対応番号		

連想三ツ組

[INPUT GRAPH, graph name, graph id]
[FRAME GRAPH, graph name, graph id]
[NODE, graph id, node id]
[LABEL OF graph id, node id, label]
[NUMBER OF graph id, node id, number] (ルールグラフのみ)
[EDGE, node id, node id]≡edge id
[NO EDGE, node id, node id]≡edge id (マッチング・グラフのみ)

(a) グラフに関する部分

item	識別子の表現	item の持つデータ
GML プログラム	16ビットで表現できる固有の範囲内の値	
GML プログラム名	それ自身	なし
ブロック名		
ブロック	GML プログラムの識別子と，グラフィックス上での識別子であるそれぞれ固有の範囲内の16ビットで表現される数値との対	座標，ブロックのタイプ，ブロックの内容であるテキストまたはルールグラフ
流れ線		流れ線を表現する三ツ組，(折れ線で描かれているときは) 中継点の座標

連想三ツ組

[PROGRAM, program name, program id]
[program name, block name, block id]
[START BLOCK, program name, block id]
[NEXT BLOCK WITH SUCCESS, block id, block id]≡flow line id
[NEXT BLOCK WITH FAILURE, block id, block id]≡flow line id
[NEXT BLOCK WITH SUCCESS AND REMATCH, block id, block id]≡flow line id
[NEXT BLOCK WITH FAILURE AND REMATCH, block id, block id]≡flow line id

(b) GML プログラムの制御構造に関する部分

Table 5 GMS's modules

モジュール（ステップ数 (F: Fortran, A: アセンブラ)，サイズ (kB)）	内容または方法	支援ソフトウェア（プログラムサイズ，GMS 側で用意するデータ領域サイズ）
GMS メインルーチン (250(F), 5.4)	全体の流れを制御する	
画面処理 (2810(F)+960(A), 67.6		
グラフの会話的作成・修正 GML プログラムの会話的作成・修正 グラフの表示	座標未定の節点は，それと隣接するすべての節点に近く，かつ節点の疎な領域に表示する	GSP (40.2, 32.8)
GML プログラムの表示 画面要素の消去と移動	ブロック間の相対位置関係により流れ線の結線方法を変える	
インタプリタ (3870(A), 22.6)		
テキスト処理 マッチング エンベッディング	逆ポーランド記法に変換 ラベルの一致，次数の条件等による前処理後バックトラッキング法を用いる	EDSP (30.3, 32.0)
磁気テープまたはカード上のデータの入出力 (2130(A), 23.4)	内部表現と印字可能な外部表現との相互変換	
モデル化ルーチンの支援 (100(F), 14.4)	行列と内部表現との変換	
ハードコピー (20(F), 0.6)		ハードコピー用サブルーチンパッケージ (15.3, 0)

グラフィックス上の図形要素はライトペンによる指示のための 16 ビットの識別子を持ち，各 item の識別子は 32 ビットからなる．グラフィックスによる会話を容易にインプリメントするために，両識別子の対応を明確にし，識別子の値からその種類を判別できることが望ましい．また，GML プログラム，特にマッチングとエンベッディングの実行効率は，GMS の実行性にとって大きな要因であり，そこでのデータ検索は効率的になされなければならない．これらを考慮して item の識別子や各種連想三ツ組の表現形式が **Table 4** (前頁参照)のように定められた．

5.2 システム構成

GMS を構成するハードウェアは，HITAC 8410 計算機と H-8283-1 型コンピュータ・グラフィックス装置が主体である．この他に，ハードコピー装置である GLP (Graphic Line Printer)，二次記憶装置としてのディスク，ファイルシステムの代用としての磁気テープあるいはカードの入出力装置等である．

GMS は FORTRAN とアセンブラを用いてインプリメントされており，次の既存ソフトウェアを利用している．データ構造のための EDSP，グラフィックスからの入出力を支援する GSP (Graphic Subroutine Package)，及びハードコピーをとるためのサブルーチン・パッケージである．

各モードの主部分，複数個のモードから共通に利用される部分，という基準によって，GMS は多くのモジュールに分割されている．主要なモジュールとその大きさ，そこでとられている特記すべき方法が **Table 5** に示される．システム全体で約 320 kB (1 B=8 bits) であるが，オーバレイにより 187 kB の占有記憶領域に抑えられている．これらモジュールの中でアルゴリズム的に興味深いものとして，マッチングとグラフ表示がある．前者は効率の改善の面で，後者は画面上での理解し易い節点の配置という面で今後更に研究されるべき問題である．

6. 応用例

Fig. 2(a)のような平面図形において，連続する部分図形間の包含関係は (b), (c) のように木や文字列で表現できる．Buneman[8] は，この変換を行う方法として生成文法を用いた．その方法が面積情報をも考慮したものに拡張されること，及びそれが GMS を用いることにより容易に実現されることを示すことができる[9]．本章では，簡単のため面積情報の考慮は省略して，平面図形より木表現を得るための GMS を利用した方法について述べる．

まず変換規則を Buneman の方法に従って述べる．

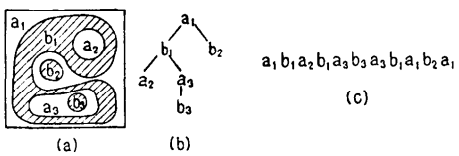


Fig. 2 An example of plane figure and its representation

平面図形は白，黒のメッシュによる表現とし，最も外側はすべて白であるとする．行間の各線に接続関係を示す分岐数（線の上部にある接続領域が次の行で分割されている数）の列を与える．**Fig. 3** に分岐数列の例が示される．線 l の分岐数列 11201 は，次の行で接続領域 a, b, e は変化せず，c は 2 つに分割され，d は消滅することを示す．変換規則は，この分岐数列と，非終端節点の string 表現を示す補助配列 F を使って記述される．(変換規則中，a_i, b_i はそれぞれ白，黒の接続領域を表わす．)

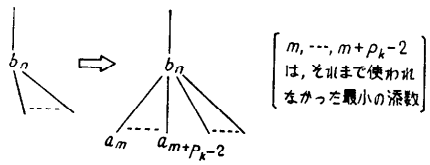
変換規則 1 (初期設定)

a_1 なる 1 点からなるグラフを作る．$F(1):=a_1$．各分岐数列の k 番目の値 P_k，及び配列 F の内容（分岐数 P_k の非終端節点は $F(j)$ であるとする）によって，変換規則 2〜5 のいずれかがとられる．

変換規則 2 ($P_k=1$ のとき)

何も行わない．

変換規則 3 ($P_k>1$ のとき，$F(j)=b_n$ として)



$$F(i):=\begin{cases} F(i) & i\leqq j \\ a_m & i=j+1 \\ b_n & i=j+2 \\ \vdots & \vdots \\ a_{m+P_k-2} & i=j+2P_k-3 \\ b_n & i=j+2P_k-2 \\ F(i-2P_k+2) & i>j+2P_k-2 \end{cases}$$

変換規則 4 ($P_k=0$ かつ $F(j-1)=F(j+1)$ のとき)

$$F(i):=\begin{cases} F(i) & i<j \\ F(i+2) & i\geq j \end{cases}$$

変換規則 5 ($P_k=0$ かつ $F(j-1)\neq F(j+1)$ のとき)

$F(j-1)=b_n$, $F(j)=a_m$, $F(j+1)=b_l$ として，

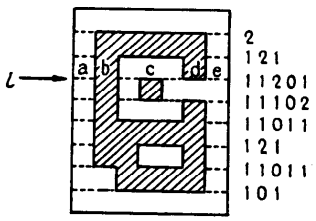
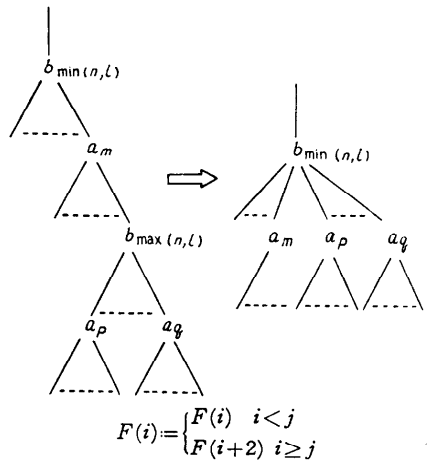


Fig. 3 An example of sequences of branching numbers



$$F(i):=\begin{cases} F(i) & i<j \\ F(i+2) & i\geq j \end{cases}$$

ただし，F 中に $b_{max(n,l)}$ があれば，$b_{min(n,l)}$ に置き換える．

変換規則 3, 5 では a, b を逆にした変換規則 3*, 5* もある．

これら変換規則を実現している GML プログラムの概略を **Fig. 4** に示す（見やすくするために一部 GML では許されない記法を用いている）．分岐数列の情報を得るために，GML プログラム実行前にモデル化ルーチンが実行される．このルーチンは，メッシュ図形を表わす行列を読み込んで，GML プログラム中に定

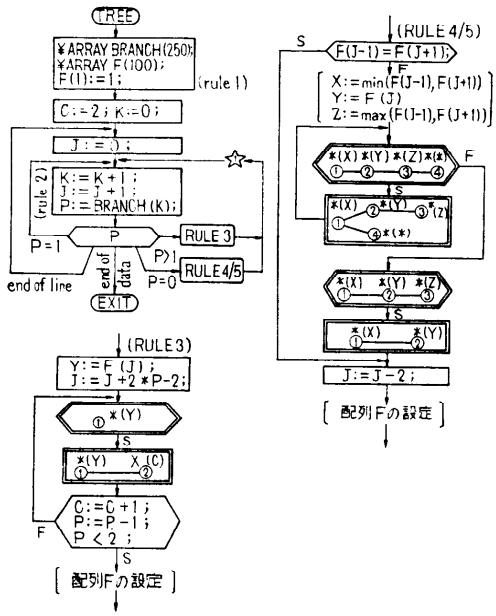


Fig. 4 Outline of a GML program which generates the tree representation

義されている配列 BRANCH に分岐数列を設定するもので，FORTRAN を使い 65 行で書かれている．Fig. 4 の GML プログラムは，各接続領域に対してユニークな番号をラベルの第1添数部分に置いている．なお，入力グラフはラベル $A(1)$ を持つ1点からなるグラフとしている．

7. おわりに

図形としてのグラフを対象とし，プログラムも二次元的である会話型システム GMS について述べた．GMS が今後広く応用されていくことが期待されている．このようなシステムが製作されたことによる成果として次のようなことがあげられる．

1) 従来の理論中心のグラフの概念をそのダイナミックな変換としてとらえることが可能となった．これにより間接的にはグラフ利用の可能性を高め得る．
2) グラフに帰着し得る問題に対しては，GML の能力や各種入出力装置の機能を試行錯誤的に駆使してインタラクティブに問題の解法を求める環境を作り得た．この点からプログラミング・システムの問題向きの試作である．
3) グラフに帰着される問題の解法を，グラフに関するデータ構造等の詳細を定めずとも実現できるので，実行可能なプログラムを速かに得ることができる．

GMS はこのような成果を得ているが，新しい型のシステムを作成する最初の試みの常として，いくつかの不備な点を見出すことができる．これらは次のようなものである．

1) 汎用的データ構造とその支援ソフトを使用しているために生ずる効率低下が存在する．例えば，データ量が増大したとき二次記憶装置との入出力が頻繁に起こる．
2) 入力グラフの節点の数などに強い制限が付いており，実用面への障害となっている．
3) 枝にラベルを付けることができないのは，応用分野によっては大きな制限である．
4) GML プログラムを表現するとき，画面の大きさによる制約から制御構造と各ブロックの内容が同時に表示されないため，全体の理解は容易でない．表示方法に何らかの検討が必要である．
5) GMS は計算機の CPU や主記憶装置を占有して動くので，気軽に利用できる状態になっていない．よりすぐれたハードウェア，OS の支援が望まれる．

謝辞 本研究の機会と各種の有益な助言を与えていただいた電子技術総合研究所石井治ソフトウェア部長，西野博二パターン情報部長に感謝する．また，システム作成に関して様々な助言と協力をしていただいた同所古川康一氏，山梨大学有沢誠氏に感謝する．

参考文献

1) S. Crespi-Reghizzi, R. Morpurgs : A language for treating graphs, CACM, 13-5, pp. 319～323 (1970).
2) W.C. Rheinboldt, V.R. Basili, C.K. Mesztenyi : On a programming language for graph algorithms, Univ. of Maryland, Computer Science Center, Technical Report TR-158 (1971).
3) T. W. Pratt, D. P. Friedman : A language extension for graph processing and its formal semantics, CACM, 14-7, pp. 460～467 (1971).
4) 杉藤，真野，鳥居：グラフ処理用二次元言語とその機能，電子通信学会論文誌 D，J 59-D, 9, p. 597 (昭 51).
5) U. G. Montanari : Separable graphs, planar graphs and web grammars, Information and Control, 16-3, pp. 243～267 (1970).
6) J. A. Feldman, P. D. Rovner : An Algol-based associative language, CACM, 12-8, pp. 439～449 (1969).
7) 古川，山崎：汎用データ構造処理システム EDSP について，電総研彙報，37, pp. 91～100 (1973).
8) O. P. Buneman : A grammar for the topological analysis of plane figures, Machine Intelligence 5, pp. 383～393 (1969).
9) 鳥居：接続図形抽出と変換文法のプログラム，電子通信学会全国大会予稿，No. 1625 (昭 49).

(付録次ページにつづく)

付録 I GML 構成要素

ブロック	ブロック名	ブロックの内容	ブロックの働き
P	開始ブロック	プログラム名	プログラムまたはサブルーチンPの実行の開始点であることを示す
EXIT	終了ブロック	EXIT	プログラムの実行終了，またはサブルーチン・リターンを示す
S	テキストブロック	宣言文 代入文	文Sの実行
S e F	テストブロック	関係式 論理式	式eを評価し，その真偽により次の実行ブロックを選択する
S g (R) F	マッチングブロック	マッチング グラフ	マッチングを実行し，その成否により次の実行ブロックを選択する
g	エンベッディングブロック	エンベッディング グラフ	エンベッディングを実行する
P	サブルーチンブロック	プログラム名	サブルーチンPを実行させる

(a) GMLブロック

流れ線	流れ線の意味
Ⓐ—Ⓑ	Ⓐの実行後Ⓑを実行する
Ⓐ —S→ Ⓑ	Ⓐの実行結果が真/成功のとき続けてⒷを実行する
Ⓐ —F→ Ⓑ	Ⓐの実行結果が偽/失敗のとき続けてⒷを実行する
Ⓐ —R→ Ⓑ	Ⓑ(マッチング・ブロック)をリマッチング(これまでⒷによって見つけられた以外の部分グラフを見つけること)によって実行する

(b) 流れ線

付録 II 組込関数

組込関数	値
¥ POINT	中間グラフの節点の個数
¥ LINE	中間グラフの枝の個数
¥ DEG	中間グラフ中の直前にマッチングされた節点の次数
¥ DEGI	中間グラフ中の直前にマッチングされた節点の入次数
¥ DEGO	中間グラフ中の直前にマッチングされた節点の出次数
¥ MAXDEG	中間グラフ中の節点の最大次数
¥ MAXDEGI	中間グラフ中の節点の最大入次数
¥ MAXDEGO	中間グラフ中の節点の最大出次数
¥ MINDEG	中間グラフ中の節点の最小次数
¥ MINDEGI	中間グラフ中の節点の最小入次数
¥ MINDEGO	中間グラフ中の節点の最小出次数
¥ LABEL (label)	指定されたラベルを持つ中間グラフ中の節点の個数

(昭和51年7月8日受付)