

An Interactive Graph Manipulation System GMS and Its Applications

Yoshihisa MANO*, Yoshio SUGITO* and Koji TORII*

Abstract

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

1. Introduction

Graph has wide application area, partly because graph theory has been established almost completely and the graphs are adequate diagrammatical means to represent "relations". GML (Graph Manipulation Language) is designed so that one could directly treat graphs represented in two-dimensional form by using a computer graphic display. GML is a graphical language, in which the graph transformation procedure is written in flowchart-like block diagrams.

It can be expected that implementations of GML processor, one of which is GMS (Graph Manipulation System) described here, encourage users' creative and intuitive power of problem solving, and allow users to find solutions for his problems heuristically.

2. Summary of GML

We could regard GML basically as Web grammar⁽¹⁾ augmented with control structures. Graph processing in GML principally consists of a sequence of graph rewriting, that is to find in an input graph a subgraph isomorphic to a given graph, and to replace the subgraph with another given graph. These operations are called "matching" and "embedding" respectively, and thus the given graphs are called a "matching graph" and an "embedding graph" (a "rule graph" in a generic term)

This paper first appeared in Japanese in Joho-Shori (Journal of the Information Processing Society of Japan), Vol. 18, No. 3 (1977), pp. 257~264.

* Computer Science Division, Electrotechnical Laboratory

respectively.

GML programs take the form of block diagrams, and each different block has its own meaning. Constructs in GML are shown in Table 1. Graphs manipulated by GML programs have node labels whose form is of $X(a_1, \dots, a_n)$, where X is a character string and a_i 's are integers, while a_i 's may be integer-valued variables in case of rule graphs. The specification of GML and its descriptive power in detail are discussed in (2).

3. Features of GMS

GMS has the following features:

- 1) It has facilities as a programming supporting system. In particular it supplies users with several tools as an interactive system.
- 2) It is well suited for heuristic and/or trial-and-error approach to solve problems.
- 3) It enables us to regard graph processing as consisting of dynamic graph transformation.
- 4) GMS makes use of the full facilities of graphic terminal with a light pen to receive commands quickly, to display graphs and GML programs, to monitor the execution, and so on.

Data treated in GMS are shown in Table 2, and the flow of processes in GMS is summarized in Fig.1.

4. GMS from users' viewpoint

4.1 GMS as an interactive system

GMS users only deal with the graphic display. The commands, such as mode change, editing graphs or GML programs and so on, are transmitted using light pen, function key or typewriter attached to the display. In most cases users can operate by only hitting one of the "menus" or an element of figures found on the display using light

(a) GML blocks

form	name	content	function
	start block	program name	to begin the execution of the program
	end block	EXIT	to exit from the program
	text block	declaration, assignment	to execute the statement
	test block	logical expression	to test the expression
	matching block	matching graph	to execute the matching
	embedding block	embedding graph	to execute the embedding
	subroutine block	program name	to execute the program as a subroutine

(b) GML flow lines

flow line	function
	to execute B after A
	to execute B if the execution of A results in true/success
	to execute B if the execution of A results in false/failure
	to execute B with "rematching", that is to find other matching than those found previously

(A and B stand for blocks)
Table 1 Constructs in GML.

data	element	meaning
control	program name, block, block name, flow line	graph transformation process
statement	text	
rule graph	node, edge, label, number for correspondence	block contents
skeleton graph	skeleton graph name, node, edge, label	
input graph	input graph name, node, edge, label	input
output graph	node, edge, label	output
current input graph	node, edge, label	input graph being processed

Table 2 Data treated in GMS.

pen. The display is also used to show diagrammatical output graph immediately, and to monitor the execution of GML program.

Two blocks shown in Table 3 are added to extend interactive facilities. When program execution is interrupted at a breakpoint block, the current input graph is displayed and users can utilize GMS's all facilities including modification of the GML program and the current input graph. Breakpoint blocks and continuation mode described later have been proved greatly useful to develop new algorithms. Another extension of GML is an introduction of some built-in functions, such as the degree of a specified node, the number of nodes with a given label and so on.

GMS is divided into several modes according to the roles, which are shown in Table 4. Because of the physical size of the display, GML programs are constructed separately in flow and content mode. Except in execution or continuation mode users can select an arbitrary mode.

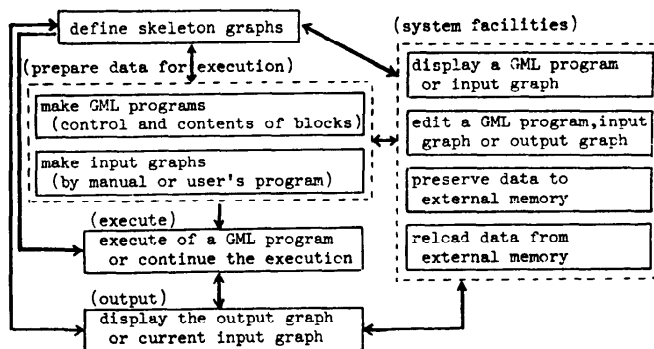


Fig.1 The flow of processes in GMS.

4.2 GMS as a problem solving system

GMS is useful for solving general problems described by using "relations". The graph model of such problem plays an important role, and various graphs could be candidates for the model. GMS allows users to include in the system their own programs to generate input graphs as the models. The program written by users, called modelling routine, takes the form of a Fortran subroutine which receives an integer parameter given by users themselves at run time.

One can trace the graph transformation process dynamically by using breakpoint blocks and continuation mode. For example, it is possible to write a GML program to simulate the derivation process of a Web grammar.



form	name	content	function
	error block	ERR	to terminate the program and output an error message
	breakpoint block	an integer	to break the execution and display the current input graph

Table 3 Appended blocks.

4.3 GMS as an integrated programming system

It may be impossible to have any consensus as for what general purpose programming system is to be. GMS is considered to be a pilot system towards a problem oriented programming language system. Users in front of the display can use various GMS facilities and I/O devices interactively, such as :

- 1) To list input graph names or GML program names.
- 2) To edit two-dimensional graphs and GML programs, for example deleting nodes, edges, blocks, or flow lines, and moving nodes or blocks.
- 3) To construct rule graphs easily by copying skeleton graphs. These rule graphs are also modifiable.
- 4) Hard copy facility is supported by software.
- 5) Data treated in GMS can be preserved in MT or cards, and can be reloaded. This serves for portable file system.

mode	task to be performed
declaration mode	to make skeleton graphs
input mode	to make input graphs
flow mode	to make control part of GML programs
content mode	to fill the content of blocks with a statement or rule graph
output mode	an output graph or current input graph at breakpoint block is displayed
execution mode	a GML program begins to execute for given input graph interpretively
continuation mode	to continue the execution of a GML program interrupted at breakpoint block

Table 4 GMS modes.

5. Internal structure of GMS

5.1 Data structure

From the nature of GMS, dynamic modification of data must be done easily and data retrieval must be done efficiently. One of such data structures is associative triples, introduced by Feldman⁽³⁾. EDSP (ETL's Data Structure Package)⁽⁴⁾ supports such data structure which GMS makes use of. Triples are denoted by [Attribute, Object, Value], where each component is an item with its own identifier and data. Triples in GMS are, for example, as follows, where upper case letter items are keywords.

[INPUT GRAPH, graph name, graph id] [NODE, graph id, node id]
 [EDGE, node id, node id] [LABEL of graph id, node id, label]

5.2 System structure

GMS, written in Fortran and assembly language, is implemented on HITAC 8410 and H-8283-1 computer graphic display. GLP (Graphic Line Printer) as a hard copy device, disc as the secondary storage, and MT or card for data preservation are also used.

Modules composing GMS are as follows, where F and A stand for steps in Fortran and in assembly language respectively : main module (250F), interaction module (2810F)

and 960A) in which GML programs and graphs are constructed and/or edited interactively, GML interpreter (3870A) including text processing and graph rewriting, a module to support modelling routine (100F) and a module for data preservation (2130A). GSP (Graphic Subroutine Package) supports input/output processing via the display.

6. Conclusions

Buneman⁽⁵⁾ has presented an algorithm which generates a tree representing topology of a plane black and white figure. By using GMS, modified algorithm which also processes area information simultaneously was implemented easily. This is one of the GMS applications, and it is expected that GMS will be utilized in wider fields. One can state some effects GMS has caused :

- 1) It makes possible to understand graphs not only from theoretical viewpoint but also from the dynamic transformation one.
- 2) GMS offers programming environment, in which users can use various facilities and I/O devices freely and try to find solutions interactively.
- 3) One can solve problems reducible to graphs rather quickly, since it is no need to elaborate the details such as special data structures for graphs.

Acknowledgement

The authors wish to thank Dr. O.Ishii and Dr. H.Nishino for encouraging them to do this project. We also thank Mr. K.Furukawa, and Mr. M.Arisawa (Assistant Professor of Yamanashi Univ. currently) for their assistances and advices in many ways in the development of the system.

References

- (1) U.G.Montanari : "Separable graphs, planar graphs and web grammars", Information and Control, Vol.16 No.3, pp.243-267 (1970).
- (2) Y.Sugito, Y.Mano and K.Torii : "On a two-dimensional graph manipulation language GML", Trans. IECE Vol.59-D No.9, pp.597-604 (1976), in Japanese.
- (3) J.A.Feldman and P.D.Rovner : "An Algol-based associative language", CACM, Vol.12 No.8, pp.439-449 (1969).
- (4) K.Furukawa and S.Yamazaki : "EDSP: A general purpose data structure manipulating system", Bulletin of the Electrotechnical Laboratory, Vol.37 No.1,2, pp.91-100 (1973), in Japanese.
- (5) O.P.Buneman : "A grammar for the topological analysis of plane figures", Machine Intelligence Vol.5, pp.383-393 (1969).