二次元言語 AMBIT/G とその試作について*

鳥　居　宏　次**

Abstract

AMBIT/G is a two-dimensional language proposed by Christensen to manipulate directed graphs. This paper reports the implementation of AMBIT/G in the small computer PDP-7, using Ring Structure Processor and Down Compiler at Mathematical Laboratory, the University of Cambridge.

This implementation divides all the program into five modules so that the users can understand this program easily using light buttons according to each module, and that the structure of program is simple.

1. まえがき

二次元言語，とくにグラフ処理を目的とする AMBIT/G と，その製作について述べる．AMBIT/G は Christensen により開発された言語で，記号処理を目的とする AMBIT に底流をおくものである．

コンピュータ・グラフィックスの普及に伴い，図形処理を目的とする言語が多く提案されている．ところが一般には，図形，すなわち平面上に広がりを持つ物体の処理のアルゴリズムの記述の手段として，直線状に並べられた文字系列によるのが大部分である．これに対し，ライトペンなどをおもな手段として図形の処理操作を指示する方法は，筆者の知る限り，AMBIT/G だけである．AMBIT/G は，上述のように，一般図形処理言語というより，むしろ線形グラフの処理を目的にして提案されたのであるが，図形またはグラフの処理手続きを図形で与える点は大いに注目に値する．

本稿では，上述のような興味およびコンピュータ・グラフィックスの普及にかかわらず，この種の処理ルーチンがあまり注目されてないことなどの理由から，その試作の結果と，それに関する問題について論じる．

この試作は，リング構造*** のデータを処理する Ring Structure Processor (RSP と略記する)[1] を用いて書かれている．RSP で作られたデータを，ディスプレイ・ファイルに変換するプログラム Down Compiler (DC と略記する)[3] を通すことにより，RSP により得られたデータ構造は，ディスプレイ上に図形として描き出される．したがって，本試作は，Wiseman などにより作られた RSP と DC の一応用例とみなすことができる．

2. では AMBIT/G の簡単な説明，**3.** では本試作における AMBIT/G のとらえ方，および RSP, DC の概略を述べる．本試作の特長として，モードの概念を定義して，ユーザが使いやすくなるとともに，プログラム構成の単純化を目的にしているが，**4.** では，その点を中心にして，ユーザからみた本プログラムとデータ構造，およびプログラム内部の構成について述べる．

2. AMBIT/G について

AMBIT/G は，方向を持つ線形グラフ処理用のグラフィック・プログラミング言語である．これについて，文献 4) をもとに，簡単に説明する．

入力データは，節点と節点間を結ぶ方向を持った枝を要素とする二次元図形である．しかし，普通の線形グラフと異なる点は，通常の線形グラフでは，各節点が形のない単なる点を意味するのに対し，AMBIT/G では，三角形，四角形のように，各節点はある定められた形からなる．以下では，形を持った図形であるにもかかわらず，便宜上それらをも節点と呼ぶことにする．各節点は名前を持つことができる．さらに，節点をなす「形」を持つ図形には，その辺の上にいくつかの特別の場所，すなわち点が，システムのほうですで

* On AMBIT/G and its Implementation, by Koji Torii (Electrotechnical Laboratory)

** 電気試験所電子計算機部

*** リング構造とは，データ構造のうち，一つのリングスタートと，任意個のリングポインタとが一つの輪のように接がれたものをいう．

に決められていて，それらの点からのみ枝が出ることができる*．ただし，枝の入る場所についてはこの種の制限はない．また，原則として，各点から2本以上の枝が出てはならない（後述のLINKモードと，RULEモードの図形では実質上，2本以上出ることがある）．

AMBIT/Gプログラムでは，入力データを変換する操作を記述するのに，二次元図形からなるいくつかのステートメントが用いられる．ステートメントには，図形処理プログラムということから当然考えられるように，次の2種類がある．その一つは，ある図形が入力データ中に存在するか否かを調べ，“Yes”ならもとの図形の一部を変換する場合であり，他方は，入力データ中に図形が存在するか否かの結果，“Yes”と“No”の場合につき，それぞれ次に実行すべきステートメントを示す場合である．これらを詳しく分ければ，次のように書き得る．

（1）　入力データ，またはすでにいくつかのステートメントの実行後のグラフについて，その中に一致している部分があるか否かを調べられるべき部分図形．

（2）　部分図形が入力データ中に見付かれば，その部分に関する枝の接ぎ換えを示す図形．

（3）　部分図形が入力データの一部に一致するとき，次に実行されるべきステートメントを示す図形．

（4）　部分図形が入力データ中に見付からないとき，次に実行されるべきステートメントを示す図形．

の四つの指示を与える．

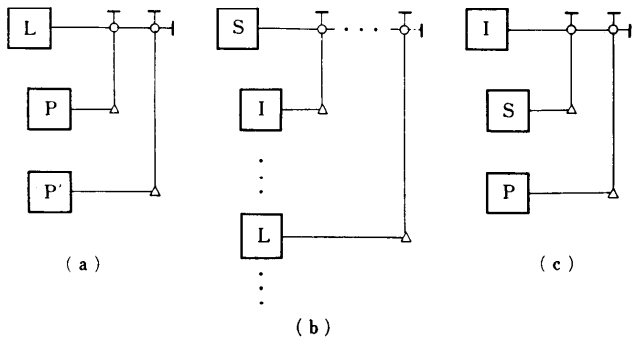
これら4種類のいくつかのステートメントに従って，与えられた入力図形が書き換えられ，その出力として入力図形と異なる節点と枝との接続からなる図形が得られる．

3. RSP と DC

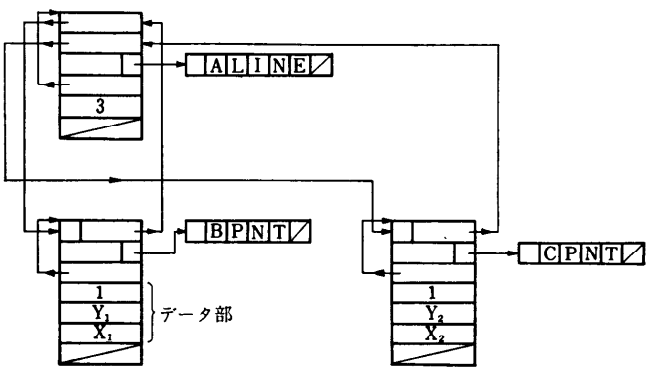
小形計算機用のデータ構造処理を目的とするRSP (Ring Structure Processor) は一方向性リング構造を持つ．データ構造をなす各エレメントは，エレメント相互の関係を示す部分と，ユーザが任意に使えるデータ部とからなる（第2図参照）．

DC (Down Compiler) は，RSPにより作られたデータ構造をディスプレイ・ファイルに変換するルーチンで，作製者 Wiseman[3] の初期の目的は，電子回路の自動設計である．したがって，DCが取り扱えるデータ構造のエレメントの種類は，回路部品を示すInstance，それらの結合を表わすLine，その位置を示すPoint，そして複雑な構造にも役だつために導入されたSubpictureだけである．DCが，これらの区別を行なうために，ユーザが自由に使用できるエレメントのデータ部に，特別の約束のもとに置かれた数字を書くことになっている（第2図参照）．たとえばLineエレメントは3を，Pointエレメントなら1を，それぞれ，エレメントのデータ部の最初の場所に置くわけである．

RSPの構造を示すのに**第1図**のような記法を用いる．これはASP[7]に従ったもので，四角形はエレメントの値，三角形，円，および直線はエレメントの関係を表わす．第1図(a)は，Line（Lで示す）エレメントが二つのPointエレメント（P, P′で示す）を端点に持つことを表わし，P, P′から各々一つずつのリングスタートが出，Lに2個のリングポインタがある



第1図　DC の RSP 上の基本的な構造



第2図　第1図(a)の具体例

* たとえば，三角形では，底辺上の左端点，右端点，中点である（第3図などを参照）．

ことを示している．

第1図を通じて，LはLineエレメント，PはPointエレメント，IはInstanceエレメント，そしてSはSubpictureエレメントを示す．第1図の接続は，DCで許されるそれらの間の可能なすべての接続関係を表わす．

第2図は，第1図(a)に対応して，実際のエレメントの構造を示す．仮りにLineの名前をALINE，2点の名前をそれぞれBPNT, CPNTとし，それらの点の座標を (X_1, Y_1), (X_2, Y_2) としている．

前述のように，AMBIT/Gでは，各種の図形を節点とし，それら相互の関係を示す直線と，その二つの端点とを表現できればよいから，上述のRSPをもとにするDCを使うことができる．

4. プログラムの構成

本試作では，**2.** で述べたようなステートメントを，できるだけモジュール化した．これは，各種の指示はライトボタンによるため，ユーザに理解しやすいこと，およびプログラムの構成がみやすくなることによる．各モジュールをユーザからみたとき，モード* と呼ぶことにし，次の5種類からなる．

(1) **LINKモード**　以後のすべてのモードに使われるグラフ上の節点間の接続関係を示す**．このモードと(3)のRULEモードにおいてだけ，節点の辺上の一点から何本かの枝が出ることも許される．

(2) **INPUT モード**　入力データがこのモードで与えられる．入力データは，節点が図形（三角形，四角形など）からなる方向を持ったグラフである．

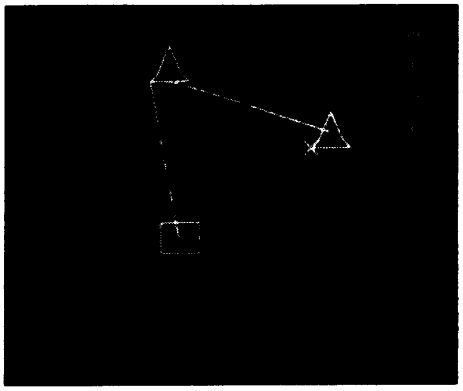
(3) **FLOW モード**　部分グラフが入力データ中に見付かるときと，そうでないときに従い，次に実行されるべきステートメントは異なる．このモードでは，各ステートメントに名前を付し，その実行順序を与える．

(4) **RULE モード**　FLOWモードで与えられたステートメント（以後，接ぎ換え規則，またはルールと呼ぶ）の名前に従い，その内容，すなわち部分図形とそれが入力データ中に見付けられた後，接ぎ換えられるべき接続関係を与える．したがって，部分図形と，接ぎ換えられた後を示す二つの図形が描かれるわけだが，後述の例題で明らかなように，実際の操作においては，太線と細線を使用して，一つの図形で描くようにしている．これはInstanceエレメントの増減がないことによる．

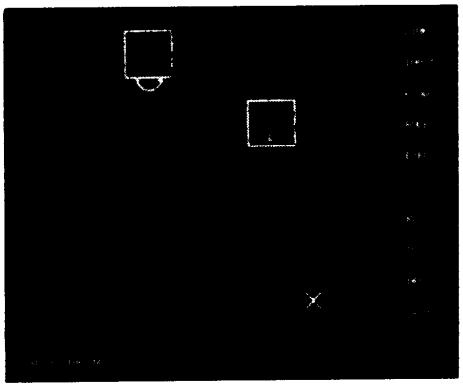
(5) **EXEC モード**　このモードでは，上述の四つのモードで与えられた情報に従って得られる出力を求める．すなわちプログラムの実行をする．

第3図～第6図は，簡単な例である．第3図は入力データを示す．方向性グラフを取り扱うプログラムではあるが，図には矢印は現われないで，各枝の始点は節点の辺上に，終点は節点の中に位置することにより代用している．したがって第3図は，Aなる名前を持つ三角形の節点（簡単のため，三角形Aと呼ぶ）の左下端点から長方形Bに，および，三角形Aの底辺の中点から三角形Cに枝が出ている入力データを表わす．

この実際の操作は，トラッキング・クロスで位置を指示し，ライト・ボタンRCTGをヒットして長方形を，同様にTRGLのヒットにより三角形が描かれる．また，RLのヒット後，二つの節点をヒットすることにより太線が描かれる．各節点の名前はテレタイプから打ち込む．



第3図　INPUT モードでの例



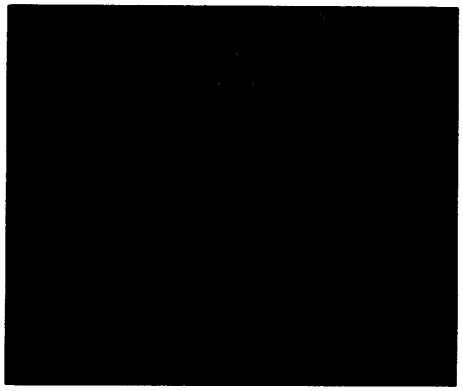
第4図　FLOW モードでの例

* 文献 4), 5) では現われない．
** このモードは，節点の接続に関する宣言とみなせば理解しやすい．

ここで，LINKモードの図形としては，第3図で各節点の名前を除いたものを考えている（図省略）．

第4図は，FLOWモードを表わす，正方形（以後，便宜上ブロックと呼ぶ）はルールを，その中の文字はそのルールの名前を表わす．このモードでは各ブロックの接続には，太線と細線の2種類を用いる．そのルールを実行して，入力データ中に部分グラフとして見付かれば，太線が指すブロック名を名前とするルールを次に行なうことを示し，見付からないときは，細線が指す方のルールを行なうことを示す．また太線で，矢印をその先に持つ半円により，同じルールを繰り返し実行することを表わす．したがって，第4図では，Sルールを実行可能な限り繰り返し，部分図形が見付からなくなると，Eルール（実際は空のルール）として，終了することを表わす．第3図と同様，BLボタンをヒットしてブロックを，ARCSボタンと一つのブロックとのヒットにより半円を，そしてDLと二つのブロックをヒットすることにより細線が描かれる．

第5図はRULEモードで，ルール名がSなることを示す（画面左下に“S”が現われている）．各節点は名前を持っていてもいなくてもよい．FLOWモードと同様に，各節点間の接続には太線と細線の2種類を用いる．図は，太線とそれに接がる節点とからなる部分図形を入力データ中で探し，見付かれば，始点を同じくする細線とそれに接がる節点とからなる図形に接ぎ換えることを意味する．したがって，三角形Aの左下端点から，任意の名前を持つ長方形に接った部分図形が入力データ中にあるか否か，もしあれば，長方形に接っている直線を三角形Cに接ぎ換えることを意味する．



第5図 RULEモードでの例

第6図 出力図形の例

LINKモードの図形と，第3図～第5図が与えられたとき，EXECボタンをヒットすることにより，上例の出力図形は第6図のようになる．

本試作のプログラムの占めるコア・エリアは，RSPとDCに1.4kW，その他に2.5kWである．プログラムはRSP命令とPDP-7のアセンブラ命令で書かれており，PDP-7と340ディスプレイ装置間には，バッファメモリがない．したがって，ディスプレイ上の図の入れ換えは，コアの別の場所に，新たに作られるディスプレイ・ファイルを先に用意し，ファイル位置を指示する方式をとっている．この結果，瞬時の時間遅れで新たな図が得られる．

入力データ中の部分図形の判定は，各ルールが少なくとも一つの名前を持った節点が存在することから簡単にできる．すなわち，名前の一致する節点を入力データ中に求め，それから始める．しかも，直線すなわち矢印の始点は，指定された点のみからで，かつ，1本しか出ないことから，ルールの節点の個数だけ，入力データ中の節点を調べればよい．

5. まとめ

本稿では，線形グラフ的な図形処理を目的とするAMBIT/Gと，その試作について述べた．この言語では，取り扱かう対象が図形のみとした場合の，フローチャートがFLOWモードで描かれていると考えることができる．したがって，判定条件が一致したか否かの場合のみのとき，たとえば，SNOBOL言語のある部分クラスについては明らかに，フローチャートをFLOWモードのように表現することができる．さらに，FLOWモードをいくつか組み合わすことにより，フローチャートからプログラムを生成する問題の一つの方法とも考えることができる．

この言語では，各ルールには少なくとも一つは名前

を持つ節点が必要である．このために，部分図形を入力データ中で探す手続きは簡単になるが，この制限により，必ずしもあらゆるケースのグラフ処理が可能ではないように思える．しかし，この言語の応用例が広くないことから，この点については文献 4) でも触れられていない．今後の応用とともに考慮されるべき興味ある点であろう．

本製作はケンブリッジ大学数学研究所で行なったものである．使用した計算機は PDP-7 であるが，このメモリは，1ワードが18ビットの 8K ワードであるため，本プログラムが取り扱える例題は簡単なものになってしまった．なお，プログラムのデバッグには，DDT オンライン・デバッギングシステムが有効であったことを付記しておく．

末筆ながら，本機会を与えていただいた，電気試験所電子計算機部野田克彦部長，同情報システム研究室西野博二室長，同計算機方式研究室相磯秀夫室長，およびケンブリッジ大学 M. V. Wilkes 教授，N. E. Wiseman と CAD グループに深謝する．

参考文献

1) N. E. Wiseman and J. O. Hiles: A Ring Structure Processor for a Small Computer, Comp. J., **10**, No. 4, pp. 338-346 (1968).
2) J. S. Grant and N. E. Wiseman: Programming with RSP. Internal Rep. of Univ. Math. Lab. Cambridge (Oct. 1967).
3) N. E. Wiseman: A Note on Compiling Display File from a Data Structure, Comp. J., **11**, No. 2, pp. 141-147 (1968).
4) C. Christensen: An Example of the Manipulation of Directed Graphs in the AMBIT/G Programming Language, Proc. of the Symposium on Interactive Systems for Experimental Applied Mathematics (1967).
5) P. D. Rovner: An AMBIT/G Programming Ianguage Implementation, Int. Rep. of MIT (1968).
6) K. Torii: An Implementation of AMBIT/G, Int. Rep. of The Univ. of Cambridge, Math. Lab., (1969).
7) C. A. Lang and J. C. Gray: ASP-A Ring Implemented Associative Structure Package, Comm. of ACM, **11**, No. 8, pp. 550-555 (1968).

(昭和45年1月13日受付)