

LISP 言語の図形処理への応用とその評価

山本 強 青木 由直
(北海道大学工学部)

1. まえがき

以前からLISPはCAD/CAMといった図形処理の分野に対しても記述能力は高いと言われてきたが処理系の速度や容量の制限から実際にそういった応用システムがLISPで書かれたという例は少ないようである。しかし、いわゆるターンキーシステムと呼ばれるCADシステムではシステムとの高度な会話が必要となり、自然言語理解や状況認識などが要求されるようになってきており、今後はますます人工知能的なシステムの構築が求められてくると予想される。我々は以前からLISPを核言語とした図形処理システムの開発を行なってきたが、従来から言われているような処理速度や容量の問題については最近のマイクロプロセッサの処理能力の向上や大容量メモリの実現によってかなり改善されてきており、実用レベルのシステムの構築が可能となりつつある。その結果、図形処理への人工知能の手法の導入がより行ないやすくなり、実用システムへの人工知能の応用も可能になると思われる。。本稿では我々が開発しているLISPベースの階層的図形編集システム『AIDS』(Advanced Interactive Designer System)を例にしてこのような図形処理システムにおけるLISP言語の記述能力とその効率について述べる。

2. リスト構造による図形データ表現

AIDSで想定している図形データはLSIマスクパターンに代表される多層構造の面積パターン、あるいは線分で表現される図形である。この種の図形データは基本的なデータ要素の集まりとして表現され、各国形要素はその型、属性、座標点などの構造を持っている。これと対照的なのがいわゆるビットイメージであるがAIDSシステムではそれは取り扱わない事とする。AIDSで取り扱う基本データ構造は次の7種である。

LIN	連続する線分列
SHP	閉ループで囲まれた面積パターン
REC	四角形で囲まれた面積パターン
ARC	円及び円弧
TXT	テキストストリング
SYM	テンプレートパターン
FIG	以前に定義された図形データのネスティング(マクロセル)

これらはそれぞれ異なるパラメータで表現されており、定形的な配列によって表現するのは困難である。AIDSシステムでは内部表現として簡単なフレーム構造を採用している。ここで言うフレーム構造は現状ではむしろ属性リストと呼ぶべきものであるが少し拡張することにより共通の属性を継承によって表現する事ができるためあえてフレーム構造と呼ぶことにする。各データは図1に示されるような構造とS式で表現されている。このようなデータ表現を取ることで、各データプリミティブ毎に独立にデータ構造を定義でき以降の拡張が容易に行なえる利点がある。座標要素はX,Y座標値の点対を用いる。この点については線形リスト構造にすべきとの意見もあるが、構造的に明確でもありこの形式を採用している。

データプリミティブは1個のS式として表現され、それらを線形リストとして並べて全体の図形データが表現される。従って全国形要素はそのリストをrootから探索することによって洗いだす事ができる。図形の階層表現はFIG型プリミティブによって実現される。すなわち、

1. LINフレーム

Frame ID	LIN
Level_code	Positive_integer
List_of_Points	

S-Expression ((LIN Level_code) (X1.Y1) (X2.Y2) (X3.Y3).....)

2. SHPフレーム

Frame ID	SHP
Level_code	Positive_integer
List_of_Points	

S-Expression ((SHP Level_code) (X1.Y1) (X2.Y2) (X3.Y3).....)

3. RECフレーム

Frame_ID	REC
Level_code	Positive_integer
Rotation	Positive_integer
Hight	Positive_integer
Width	Positive_integer
Point	

S-Expression ((REC Level_code Rotation Hight Width) (X.Y))

4. ARCフレーム

Frame ID	ARC
Lavel code	Positive_integer
Radius	Positive_integer
Angle start	Positive_integer
Angle end	Positive_integer
Point	

S-Expression ((ARC Level-code R Begin End) (X.Y))

5. TXTフレーム

Frame_ID	TXT
Level_code	Positive_integer
Text	String,Atom or Number
Rotation	Positive_integer
Size	Positive_integer
Point	

S-Expression ((TXT Level_code "string" Rotation Size) (X.Y))

6. SYMフレーム

Frame_ID	SYM
Level_code	Positive_integer
Symbol_name	Atom
Rotation	Positive_integer
Size	Positive_integer
Point	

S-Expression ((SYM Level_code Symbol_name Rotation Size) (X.Y))

7. FIGフレーム

Frame_ID	FIG
Level_code	Positive_integer
Figure_name	ID
Rotation	Positive_integer
Size	Positive_integer
Point	

S-Expression ((FIG Level_code Figure_name Rotation Size) (X.Y))

図1 AIDSシステムのデータ構造とS式表現

FIG型プリミティブは以前に定義された図形そのものを表すが、図形デ-タの実体はそのプリミティブではなく、アクセスパスだけが示される。従って図形リストは表向き(トップレベル)は単純な線形リストであるが実際の構造はTreeとなっている。AIDSの基本思想として図形デ-タの操作はトップレベルから見える要素のみを対象とするため、構造の複雑さと関係なく編集操作を画一化することができる。

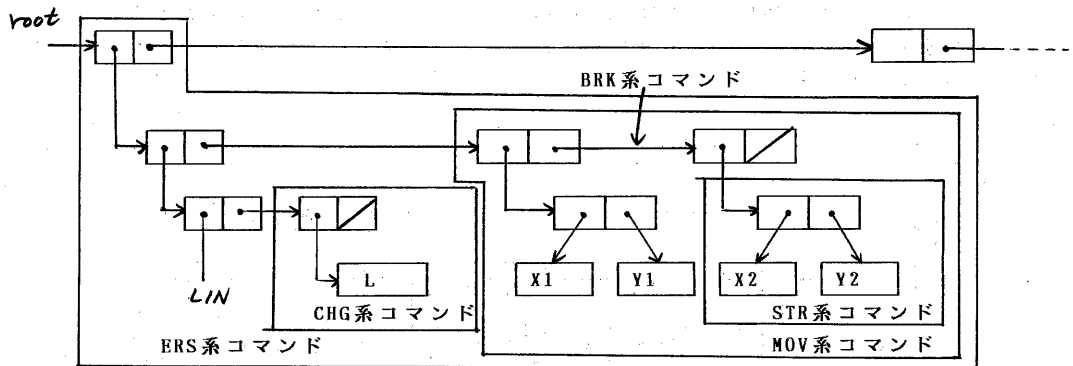
図形デ-タをリストで表現することは構造的なエレガントさはあるがメモリ-効率の点から見ると決して有利とは言えない。これは座標デ-タのようにあらかじめ構造が固定されているものに対してもセルを消費してしまうためでもある。より専用化された処理系を作成し、座標もアトミックな要素として取り扱うことによりこの問題はかなり改善されると思われる。しかし、それによって処理系の一般性は失われることになり、ポストプロセッシングヘデ-タを渡す際の障害ともなる可能性があり、むしろデ-タ構造はなるべく純粋なリスト構造としておきメモリ-の大容量化で対処するほうが良いと思われる。

3. 図形編集

図形編集は次に上げるいくつかの基本パターンに分類される。

- | | |
|-------------|----------------------------------|
| 1. INS系コマンド | 現在アクティブな図形にたいするデ-タの追加(Insert) |
| 2. ERS系コマンド | 指定されたデ-タエントリ-に削除(Erase) |
| 3. MOV系コマンド | 指定されたエントリ-の移動(Move) |
| 4. STR系コマンド | 指定されたエントリ-の位相を変えないような変形(Stretch) |
| 5. BRK系コマンド | 指定されたエントリ-の変形(Break) |
| 6. CHG系コマンド | 指定されたエントリ-の属性値の変更(Change) |

これらはデ-タプリミティブのどの部分を変更するかによっている。図2に編集コマンド毎の操作の対象を示すが、このように体系化することによって編集操作関数をデ-タプリミティブと独立に設計できるようになる。実際の編集は操作関数を図形リストの要素に対してMAP関数によって作用させる事により行なわれる。この時、操作対象を指定するために座標に対するサーチを行なう場合があり、その処理速度は操作性の善し悪しに関係してくる。そのためAIDSシステムでは3種類の干渉チェック(二つの四角形が共通領域を持つか、線分と四角形が交差するか、点が四角形の中に含まれるか)の述語をSUBR関数として用意した。



(図)2 基本編集コマンドと操作対象(線分の例)

本システムのような図形処理システムでは取り扱えるデータ量の上限が一つの処理能力の指標となる。AIDSシステムでは階層的な図形表現を行なうため実際に記述されているデータ量(Top Level)よりもはるかに多くの図形が表示される事がある。これは同一の図形がマクロセルとして複数回用いられた場合であるが、ここでは単純にトップレベルでデータを記述した場合、どの程度のデータを取り扱えるかを検討してみる。AIDSシステムで最もプリミティブなデータ構造は線分である。現在、2点を結ぶ線分は図2からも分かるように13セルで表現される。従って単純に線分を入力していくと約2000ベクトルで全セルを消費してしまう。これは全ての線分が異なる属性を持っている場合であり、実際には何種類かのタイプに集約されるためハッシュによって先頭部は共有されるようになり、ほぼ10セルで1ベクトルが表示されるようになる。プログラム実行中はオーバーレイ関数がロードされているため更に有効セル数は減っており、約2500ベクトルが最大記述可能データ量であるといえる。もっとも効率の良いのはREC型であり(図3)、この場合は4000個のRECがトップレベルで記述できることが確認できた。この場合も単純にセル数を計算すると1REC当たり15セルを消費するが先頭部を共有することによって6セルまで減少する。この量は決して多いとは言えないがトップレベルで多量のデータを記述する事は編集操作や画面のリフレッシュに要する時間が増加しインタラクティブな操作が行なえなくなるためなるべく同一レベルの図形要素は1000個以下とすべきである。また、このような配慮を行なえばAIDSではデータが階層的に記述された場合にデータのオーバーレイを行なうため実質的に取り扱えるデータ量には制限が無くなる。

この程度のデータ量でもLSIセルデザインや中規模のPCB設計には実用可能である。本システムの最大の特徴はデータ構造が明確である事であり、これによって一旦図形データファイル化された図形情報を他の処理系へ渡すことが容易となる。また本システムの図形データ表現形式は『絵』としての情報よりもネットワークやトポロジ-的な情報を重視して設計されており、図形の認識が行ないやすくなっている。そのような応用例として電子回路シミュレータへの応用を図4に示す。この例では回路の結線情報やシンボルはAIDSの図形ファイルとして記述されたものがそのまま回路シミュレータに渡され、ネットワークを解析することによって回路方程式を組み立てる。方程式化されたデータは過渡解析プログラムによってシミュレートされ結果は再びAIDS図形ファイルとして返される。回路シミュレータの自体は連立一次

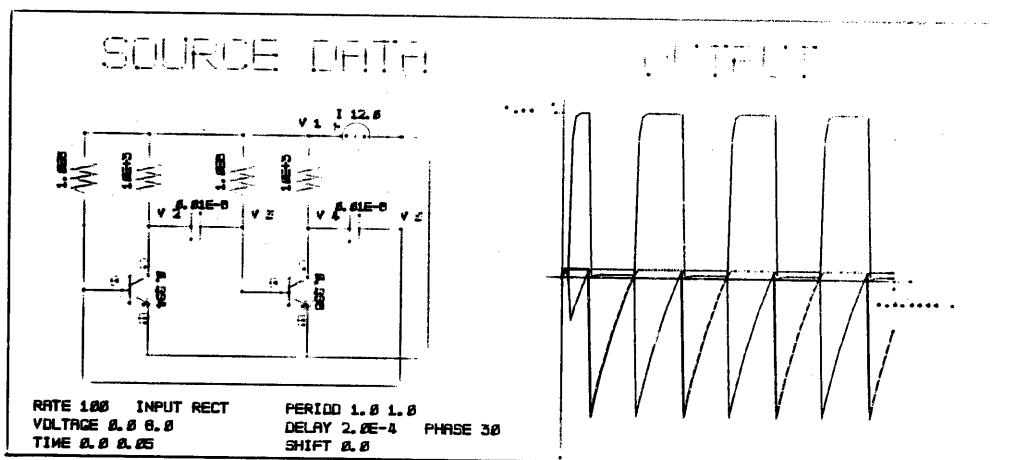


図4. AIDSシステムの応用例(回路シミュレータのデータ記述)

方程式の解を求める問題になるためC言語を用いたが、図形データの構造が明確であるため回路方程式の認識は容易であった。

6. まとめ

LISP言語の特性上、数値計算的な部分は他の手続き型の言語に劣るがデータ表現、制御構造についてはかなり有利な点が認められる。実際に試作システムを作成した結果、実行速度についても仮想画面のサポート、基本サ-チ関数のSUBR化によってインタープリタ上でも十分な速度が得られることが分かった。LISPの問題点としては、データ領域の消費量が他の言語の場合とくらべて数倍必要となる事があるが、これについてもハッシュ技法の採用によってかなり改善されることが確認できた。この点については最近はパーソナルマシン上でも1Mバイトを越えるメモリ空間を確保出来るようになってきておりそれをサポートするようなLISP処理系の開発によってより改善されるであろう。

LISPの言語仕様はその応用を記号処理や知識表現といった分野として設計されていると考えられ、本稿で述べたような図形処理などには決して満足の行くものではないが多少の拡張仕様によって無理なく応用できた。実際にプログラミングして見るとMAP関数のように並行処理的な記述が自然に行なえるなど今後の図形処理システムの記述法にも一つの示唆を与えている。今後、コンカレント処理を陽に記述出来る言語仕様や処理系が得られるようになった場合、より実用的システムがLISP上で記述出来ると期待される。

参考文献:

1. 小林、山本、青木『リスト処理言語による対話型図形処理システム』電子通信学会情報システム部門全国大会予稿集、講演番号187,昭和58年9月
2. 山本、小林、青木『マイクロコンピュータによるLISP汎用図形編集システム』情報処理学会記号処理研究会資料、WGSYM27-3,昭和59年3月
3. J.B.Marti,A.C.Hearn,M.L.Griss,"STANDARD LISP REPORT",University of Utah,UUCS-78-101,1978