北海道大学工学部研究報告
第149号（平成２年）

Bulletin of the Faculty of Engineering,
Hokkaido University. No. 149 (1990)

視覚的ソフトウェア環境における知的処理

南　　弘征　　水田正弘

（平成元年９月28日受理）

An AI Application under a Visual Software Environment

Hiroyuki MINAMI and Masahiro MIZUTA

(Received September 28, 1989)

Abstract

Recently, it has been important for end-users in a computer society to build user-friendly softwares. With respect to it, "Visual programming" is one of the most attractive conception for man-machine interface. Using a pointing device (mouse), and operating objects directly on CRT, we can clearly understand the relation between the action of computer and one's mental image. But, the useful environment itself may simply increase the breadth of operations, so that the end-user may be confused when to use it. For improvement of usefulness, we may assume that a new approach which differs from previous ones should be applied. Thus it is considered that tiny AI programs utilize for its development as a new side.

Then, we add "Validity Check" (tiny AI module) to the data analysis system which is already made, with a point of view that the fusion of "Visual Programming" and AI program is one part of a development of the visual environments. We propose that this module is available for a confidence of operations by examples with executions.

1. はじめに

昨今のコンピュータの普及に伴い，プログラミングや計算機についてほとんど知識を持たないエンドユーザと呼ばれる人々にも，容易でかつ有効に利用できるようなシステムの構築が重要な課題となってきた。その際，考慮すべき点として，『操作性の向上』および『知的処理の自動化』の二つがある。『操作性の向上』のために視覚的ソフトウェア環境[1)2)]やダイレクトマニピュレーション[3)]の研究がなされている。例えば，ポインティングデバイス（マウス）等を用い，画面上で直接，対象を操作できることにより，コンピュータの動作と，動作に対するユーザの心的なイメージとの結びつきはより密接になる。また，計算機に期待される処理内容が，ハードウェア・ソフトウェアの進歩に従って複雑かつ高度になってきており，そのために初心者が誤った操作や不適切な処理方法を実行するのを防ぐため，『知的処理の自動化』が第二の課題として考えられる。

以上の背景から，本報告では視覚的ソフトウェア環境と知的処理との融合をヒューマンインタ

情報科学

フェースの一環として捉えることを念頭に置いて開発したデータ解析システムについて報告する。はじめにデータ解析システムの概要を紹介した後，知的処理によって実現している部分，すなわち個々のデータ処理間の結合関係を評価するモジュールについて詳述する。

なお，本システムはC言語で，また本モジュールはPrologで記述されており，PC-9801シリーズのMS-DOS環境上で動作する。以下，説明のためにDEC-10 Prologの表記[4]を用いることがある。

2. 視覚的ソフトウェア環境について

ヒューマンインタフェースを考える際，視覚的ソフトウェア環境は重要な研究の一つである。これは近年の計算機環境，特にグラフィックディスプレイの普及に伴って急速に発展してきた研究であるため，研究の分類がほとんどなされていない。その数少ない分類の例として，Shu[1]は「データの視覚化」，「プログラムとその実行過程の視覚化」，「視覚的コーチング」といった分類を試みている。

「データの視覚化」はいわゆるプレゼンテーションの問題であり，これまでにも多くの研究がなされている。また，数値データの出力に関してはグラフィカルな表現法として数多く報告されている。「プログラムとその実行過程の視覚化」はプログラムソースの適切な印字から，実行過程のグラフィカルな表示までを範疇としている。特に，実行過程の表示（トレース）は，効率的なソフトウェア開発に際して，重要かつ実際的なテーマである。以上の2つと一線を画しているのが「視覚的コーチング」である。これは，ユーザがマウス等で対象を直接操作し，その過程を計算機が学習することで同様の過程を実行するというもので，機械学習の分野では「例示学習」と呼ばれることがある。

本報告で説明するシステムは「プログラムとその実行過程の視覚化」の範疇に属する。さらに，知的処理を行なうモジュールを付加したことで拡張がなされている。前述した3つの範疇のうち知的処理を含んでいるものとして，「例示学習」に当たる「視覚的コーチング」がある。しかし，「視覚的コーチング」の目的はプログラムの自動生成にあり，操作性そのものには直接関与していない。システム全体で考えた場合，「視覚的コーチング」の下で，視覚的環境と知的処理は全く異なる目的で用いられている。本報告で説明するシステムは，視覚的ソフトウェア環境と知的処理を同一の目的，すなわち『操作性の向上』をめざして構築されており，「視覚的コーチング」とは異なった立場を取っている。

視覚的ソフトウェア環境と知的処理は，各々単独でもソフトウェア操作性の向上に有益である。この二つをうまく融合し，相乗効果を引き出すことでさらに有効な環境を作成し，ユーザーの負担を減らすことが，本報告で述べる研究の目的である。

3. 視覚的データ解析システム

今日，データ解析を行なうためには計算機の利用が不可欠であり，数多くのデータ解析システムが開発されている。しかし，多くのシステムは，コマンド方式やメニュー方式で操作を行なうため，自分の使用目的と直接的には無関係なコマンドや操作法を会得する必要があり，初心者にとって負担が大き過ぎる。そこで，視覚的ソフトウェア環境の考え方に基づいたデータ解析システムを試作した[5]。本報告では，このシステムに付加した知的処理の機能を扱うため，はじめに本データ解析システムの概略を説明する。

データに対する解析処理や出力方法，および処理される原データは全て箱で表わされる。各々の箱には，その処理特有のオプションを設定するためのプルダウンメニューが用意されている。また，画面右上には「画面コピー」等，システムそのものに関する「コマンド」を選択するためのプルダウンメニューがある。このような環境で，ユーザは利用したい処理の箱を選択し，マウスを用いてディスプレイ上の任意の位置に配置する。次いで，オプションを適正な値にし，箱と箱を線分でつなぐことによってデータ処理手順を作成，実行し，結果の出力や処理手順の修正を行なうことができる（画面の例は6章の図2参照）。

この環境は，処理手順の組み立て作業を非常に容易なものとしているが，データ解析処理を行なうソフトウェアとして見た場合，操作が簡単なだけに，データ解析に関する知識を持たないユーザに対しては，不適切な処理選択をする危険を増加させる，という欠点も併せ持つことになる。

そこで，手順が正当かどうかを判断し，不適当である場合や最適な手順を特定できる場合は，直接指示するような知的処理が付加機能として望まれる。そのうち今回は，2処理間の結合関係に対する判断を行なうことが可能になったので報告する。理想的には3処理以上の結合関係を総合的に判断する必要があるが，これは後で述べるように現段階での実現が難しい。そこで今回は基礎的研究として，判断の最小単位である2処理間の判断に限定した。

なお，本システムで処理される原データはファイルで与えられ，各処理間の情報の受け渡しは一時ファイルを用いて行なわれる。加えて，個々の処理はそれ自身が一つの実行ファイルであることから，この環境は移植性が極めて高く，解析処理の追加は容易である。

4. 知識ベース

処理間の結合を判断するためには，当然各処理の内容を予め計算機側が把握している必要がある。通常，このような知識の表現は（特に自然世界などでは）属性の継承等，非常に扱いにくい側面を持つ。しかし，データ解析処理における知識は統計学を基礎としているため，必要な知識がある程度整理されており，簡潔に記述することが可能である。今回，具体的には以下のような表記法を取った。

処理名：：：：：対象データタイプ
　　：：：：最適入力データ属性
　　：：：可能入力データ属性
　　：：出力データ属性
　　：オプション

知識の要素各々に関して具体的に詳述する。

(A) 処理名

データ解析や出力の処理名が入る。現在までに用意されている処理は以下の通りである。

データ入力
キャラクタ出力（数字のままでの出力）
ヒストグラム
棒グラフ
散布図
レーダーチャート
主成分分析

距離

多次元尺度構成法（MDS）

クラスター分析

(B) 対象データタイプ

処理の適用対象となるデータの型を記述したもの。現在の版で扱えるデータの型は2種類ある。

① データ行列（datamatrix）

各行がデータの1レコードを，その各々の列はデータの種類を指す。

② 正方行列（squarematrix）

行・列ともに同一の範疇に属するデータの並びを示す。この型は類似性・非類似性を指すかどうかの属性値を持つことができる。

(C) 最適入力データ属性

当該処理の適用が望ましいとされるデータの属性が記述される。

記述に用いた表記は表1の通り。

(D) 可能入力データ属性

当該処理の適用が（形式的にでも）可能なデータの属性が記述される。

記述法は(C)に同じ。

(E) 出力データ属性

当該処理から出力されるデータの属性が記述される。この記述は(C)，(D)と対応づけられ，処理の前後でのデータ属性の推移を示す。

(F) オプション

当該処理中，評価に直接必要なオプション名を列挙する。データをPrologの節に変換する際に述語名や引数を付加され，getOpt/3という述語として登録される。

前述の表記法によって記述された知識ベースはPrologの節に変換され，実行時に用いられる。つまり本モジュールの実行部分とは切り離されており，従って知識の追加は容易に行うことができる。

表1 知識ベース中の表記

表　記	意　味	表　記	意　味
datamatrix	データ行列である	sim, dissim	正方行列での属性(類似性, 非類似性)
squarematirx	正方行列である	type	データ行列の各列における属性の列
dim	各行列の次元	all（Attr）	データ値の属性が全てAttrである
num	データ行列でのデータ数	not（Any）	Anyではない
disc, cont	データ行列での属性(離散，連続)	=	左辺と右辺が等しい(数，文字とも)

※その他，算術記号を本来の意味で用いた。

変換を行うプログラムはPrologで記述されている。知識ベースは，本モジュール実行時，最適入力データ属性，可能入力データ属性との照合を処理系のパターンマッチングで行なう関係上，Prologの節に変換された形で用いられる。判断の際に必要なオプション値等は，変換された節に適当な述語を挿入し，探索時にその述語の変数にオプション値を返すことで取り込んでいる。

変換の一例を示す（図1)。

知識ベース

```
pca :::::datamatrix
     ::::[dim >= 2,dim >= p]
      :::[dim = 1,dim >= p]
       ::[datamatrix,p*cont,num]
        :[p].
```

実行用Prolog節

```
action(pca,datamatrix,fit,ProcID,
       InList,[datamatrix,P*cont,num]) :-
  getOpt(ProcID,p,P),
  member1(dim(X),InList),
  X >= 2,
  member1(dim(Y),InList),
  Y >= P.

action(pca,datamatrix,possible,ProcID,
       InList,[datamatrix,P*cont,num]) :-
  getOpt(ProcID,p,P),
  member1(dim(X),InList),
  X >= 1,
  member1(dim(Y),InList),
  Y >= P.

action(_,_,discord,_,_,_).
```

図1 知識ベースから実行用 Prolog 節への変換例

5. 評価のアルゴリズム

評価のアルゴリズムは以下の通りである。なお，評価の対象となる2処理において，データを出力する側をS処理，S処理の出力を受ける側をD処理とする。以下，[ ]内が評価値，{ }内が修正手続名である。これらが，アイコン番号等と共にシステム本体へ返される。

I S処理の出力データ属性をD処理の入力データ属性と比較する。評価は3段階に分かれる。

① S処理の出力データ属性⊂D処理の最適入力データ属性→ [fit]

② S処理の出力データ属性⊂D処理の可能入力データ属性→ [possible]

③ それ以外の場合 → [discord]

①の場合は適切である旨を出力し，終了。②③の場合はIIへ。

II D処理の入力データ属性がオプション値と関連している場合，オプション値を変えることでS処理出力データと属性が一致するか調べる。一致する場合 {changeOpt} は，その変更すべきオプションと望ましい値を出力し，終了。

III 入力データ属性がS処理の出力属性と [fit] であり，出力データ属性がD処理の出力データ属性と一致するような処理を知識ベース中から探す。見つかれば，②③の違いを含めて，処理の変更 {changeAct} を提案して，終了。

IV 入力データ属性がS処理の出力属性と [fit] であり，出力データ属性がD処理の入力データ属性と一致するような処理をIIIと同様に知識ベース中から探す。見つかれば，IIIと同じようにして，処理の挿入 {insertAct} を提案して，終了。

V　I～Ⅳのいずれでも属性間の評価値が改善されなかった場合は，その旨を出力し，終了。

本アルゴリズムの適用時，P1—P2—P3 という処理の結合がなされ，P1—P2 は［possible］あるいは［discord］とされた場合，P2 からの出力データ属性をそのまま評価に用いることは適切ではない。なぜなら，ユーザはこの処理結合を，全ての処理間で［fit］と考えて作成したのであり，［possible］や［discord］という評価はそれ以前に何らかの間違いがあったことを意味している。つまり，この状態で得られる P2 の出力データ属性が正常とは限らないため，P2—P3 結合の評価が正確になされるとは考え難いのである。

この場合，P1—P2 の結合が［fit］以外とされた時点で判断を終了し，ユーザが修正してから次の判断を行なうというのが正しい順序である。しかし，3 つ以上の処理の結合では複数の誤りがあることも考えられる。その場合，ユーザは誤りが完全に修正されるまで本モジュールを実行する必要があり，冗長な作業を強いられる。そこで，あらかじめ P1—P2 の結合関係が修正され［fit］になることを仮定し，その仮定から得られる出力データの属性を用いて次以降の処理結合に関する評価を行なうようにした。よって，評価に際して，出力データ属性は常に正しいものが出力され，各々の 2 処理間の評価が一度に可能となった。

なお，本アルゴリズムはデータ解析処理を例として説明されているが，計算機で扱われるデータの多くは各々何がしかの属性（またはサブゴール）を持っており，それに対する処理とは属性を変換することである場合が多い。このような類のシステムにおいて，本アルゴリズムは概ね適用可能である。

6. 実　行　例

実際にこの判断ルーチンを用いた実行例を示す(図 2)。この例では，原データを多次元尺度構成法（MDS）にかけ，その結果をヒストグラムとして出力すること，また更にクラスター分析を行ない，数値出力を行なうという処理が示されている。

実際の処理内容について簡単に説明しておく。多次元尺度構成法はデータのうち，類似しているものを近くに，そうでないものを遠くに配置する処理である。クラスター分析は，（ここでは）多次元尺度構成法の結果をより鮮明にするために分類する目的で使用されている。また，ヒスト

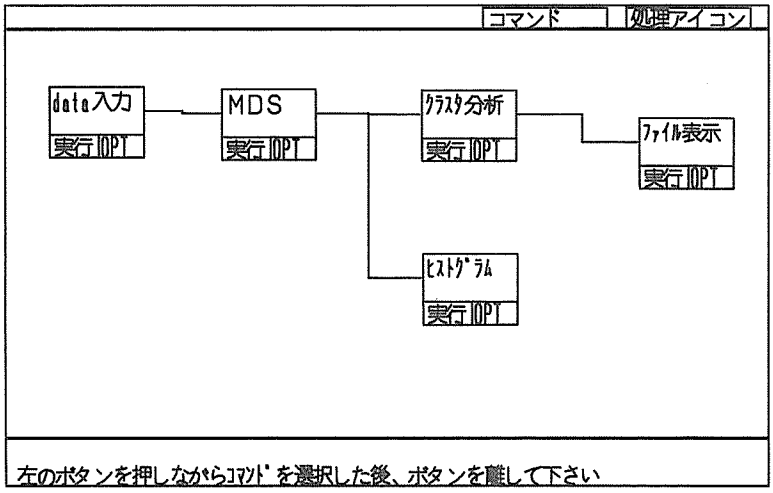

図 2　データ解析システムの画面

グラムは多次元尺度構成法の結果を図的に表示するために使用されている処理であると考えられる。しかし，多次元尺度構成法の出力は2変量以上であり，ヒストグラムは1変量を図示する方法である。つまり，これは明らかな誤りである。

本モジュールは，この状態から「コマンド」をプルダウンし「結合確認」を選択することで実行される。選択されるとシステムは，本モジュールへシステムの状態を伝達するために一時ファイルを作成する。ファイルに記述される情報は，各処理の番号や結合関係，オプションとその値である（図3）。次いで実際に本モジュールがシステムから呼ばれ，一時ファイル中の各情報はモジュールの内部にPrologの述語として宣言される。そして，前述のアルゴリズムにより評価が行なわれる。各2処理間に対する評価結果は，システムからの一時ファイルに上書きされる。返される情報は2項の評価と，もしあれば補正の提案である（図4）。これを受けてシステムは該当する処理を枠で囲み，評価を基に指示メッセージを出力する（図5）。この例では「MDS」および「クラスター分析」のオプション値の誤りに対しての指示と，「ヒストグラム」の「散布図」への変更を提案している。これらを基にユーザは指示に従って修正を行なうか，別な処理手順の検討を始めることになる。

```
action(0,0,[file=data/dat]).      data入力.
action(1,7,[method=torgerson,p=2,sim=sim]).      ＭＤＳ.
action(2,10,[input=squarematrix,sim=sim]).      ｸﾗｽﾀ分析.
action(3,1,[file=crt]).      ﾌｧｲﾙ表示.
action(4,4,[]).      ﾋｽﾄｸﾞﾗﾑ.
path(0,0,1,0).
path(1,0,2,0).
path(2,0,3,0).
path(1,0,4,0).
[0,squarematrix,4,disc,sim].
```

図3 図2の状態でのシステムからの情報

```
0 0 1 0 POSSIBLE CHANGEOPT METHOD KRUSKAL
1 0 2 0 DISCORD CHANGEOPT INPUT DATAMATRIX
2 0 3 0 FIT
1 0 4 0 DISCORD CHANGEACT 2
```

図4 図3に対する本モジュールの出力情報

（「ＭＤＳ」を赤で囲んで）オプションを[KRUSKAL]にしてみてください。
（「クラスター分析」を赤で囲んで）オプションを[DATAMATRIX]にしてください。
（「ヒストグラム」を赤で囲んで）この処理を「散布図」にすることを検討して下さい。

※（）内はシステムの動作

図5 図4によるシステムのメッセージ出力

7. 考察と問題点

本モジュールの当初の目的である，視覚的プログラミング環境下でのユーザタスクの一部負担は一応達成できたと思われる。エキスパートシステムのような本格的な知的処理は，視覚的ソフトウェアなる環境とは直接関連しないと考えられるが，本システムと本モジュールのように，操作性の向上という目的を掲げた時の二つのテーマの融合は，非常に強力であることを示唆しているように思われる。また，システムはC言語でプログラムされており，それぞれの言語の特性を生かした上で同一環境を構築できたという点も，意義は大きいと考えられる。

しかし，現在の版ではいくつかの問題を依然残している。それらを以下に記す。

① 3処理以上の結合関係の総合判断

先に述べたように，当然3処理以上の結合に関しても判断を行なう必要があるのだが，これを帰納的に解釈すると「原データと最終出力方法を特定することで最適な解析法を構築する」という目的になってしまい，計算機による問題解決に帰着してしまう。知識ベースを質・量ともに増やすことで実現はある程度可能であるが，あくまでシステムの一助であるべき本モジュールの性格や実行速度を考える時，判断の対象をどの範囲とするかは直ちに決められることではなく，今後，システム利用履歴の分析などによる，充分な検討が必要である。

② 学習のインプリメント

処理の結合が適した組というのは大量に存在する。①とも関連するが，それを記憶しておき，また，分析・再構成するという機械学習の概念を本モジュールに導入することが考えられる。しかし，これも知識の爆発等，計算機による知的処理研究に際して問題となっている事柄と直結するため，①同様，本モジュールの性格とも併せ，実現には慎重であるべきと考えている。

③ 処理の拡張

データ解析処理そのものと本モジュールは無関係であるが，「n番目のデータが1であるもの」というようなデータの内容に即した抽出の指定等は知的処理の一つと考えることができるため，前述したような各言語の特性といった点を考えると，このように処理対象に即した部分も知的処理の一環として実現する必要がある。

8. システムの今後の発展性

前述したような各問題点に関しての検討と並行して，本システムと本モジュールに関して，各種の改良が行なわれている。まず，ワークステーションへの移植である。今後，システム・モジュール双方の拡張に伴って，処理の充実と実行速度との兼ね合いを考える必要がある。そこで，多くのワークステーションで主流となりつつあるX-Window上で本システムを動作させるべく，現在移植作業を進めている。

次に，他のデータ解析システムのフロントエンドとしてのシステム構築である。最近，非常に強力なデータ解析システムが開発されてきている。本システムは自身，いわゆるフロントエンド的な面を持つものであるから，それらと結合することで，よりユーザフレンドリな環境を提供できることが期待される。そこで，ワークステーションへの移植と合わせ，UNIX上で稼働する統計パッケージであるS言語のフロントエンドとして本システムを応用する作業の検討に入っている。

また，本モジュールにおいて，ユーザモデルの構築による視覚的なフロントエンドでの「対話」

管理も検討中である。視覚的な環境下でのマウスの動作が自然言語同様，ユーザからの「発話」であるという視点に立てば，いわゆる「対話」管理を行なうことも可能である。この「発話」は，構文解析の必要が少なく，また自然言語と比べて語彙が少ないこと等から，「対話」管理は自然言語の場合よりも容易である。それを導入することによって，ユーザに対しより柔軟なシステムを作成できると考えられる。

参 考 文 献

1) Shu, N. C.: Visual Programming (1988), Van Nostrand Reinhold Company, New York.
2) 西川博昭，寺田浩詔：情報処理，Vol. 29, No. 5, (1989), p. 485-504.
3) 渕　一博監修：インタフェースの科学 (1987)，共立出版.
4) 中島秀之：Prolog (1983), 産業図書.
5) 水田正弘：第 56 回日本統計学会講演報告集，(1988)，p. 116-117.
6) 水田正弘，南　弘征：第 57 回日本統計学会講演報告集，(1989)，p. 148-150.