コマンドプロセッサジェネレータとそのCADシステムへの応用

小島俊雄，井上久仁子（機械技術研究所）
小山田芳彰（東洋情報システム）

1．まえがき

CAD・CAMシステムなど，機械工業の生産システムにおけるコンピュータの利用が近年，飛躍的に大規模になってきている．そして，この分野のソフトウエア利用法は図形処理を中心とするシステムであり，その大きな特長の1つとして，(i)処理の多くが会話形式で実行される：(ii)プログラムやデータの準備・作成変更が，ソフトウェアシステムのユーザによって行なわれる必要のある場合が多い：(iii)たとえば，多様な機械工業の各分野で市民権をもっている専門用語を直接利用できることが必要である：などがあげられる．

会話形の処理システムにおいては，ユーザの入力操作によって，処理の流れを自由に変化することのできることが要求される．この入力操作は，一般にキーボードからのコマンドと呼ぶ文字列入力によって行なわれ，このコマンドの体系をコマンド言語と呼ぶ．CADシステム等においては，タブレットなど図形入力装置も用いられるがこれらの入力シーケンスも解析した後，コマンドと同等の形式に変換して統一的に処理することが可能である．コマンド言語は会話形式で処理されるため，記述言語よりずっと簡単で応答性のよいことが要求され，一般にインタプリタ形式で処理されている．しかしながら，これらの機械工業の分野別の特色を生かすコマンド処理系などソフトウェアシステムを開発してゆくための体系的な技術は未だ明確になっていない状態といってよいと思われる．[1]

本文では，CAD・CAMシステムなど上記のソフトウェアの作成・保守・評価などを体系的に支援するソフトウェアツールを明確にするという立場から開発・試作したコマンド処理プログラムについて，オイラーオペレータのプログラムパッケージ作成に関する例題に沿って考察評価した結果を述べる．オイラーオペレータは基本的なルーチン群ではあるが必要に応じて随時テストデータを用意し，手軽にテストをしたいなど，機械工業のソフトウェアに共通である．

2．ユーザコマンドの定義

本文で対象としている生産システムにおけるソフトウェアシステムでのコマンド言語の設計にあたっては，一般的な内容に関する配慮とともに，1で述べたように，ユーザがシステムのアルゴリズムの検証・改良に一定の役割を果たさねばならぬため，コマンドの拡張や変更が容易に実現されることが必要である．また，システムの対象とする分野で使用されている専門用語の使用が適切な文脈で利用できることは，その専門用語の背後にその分野におけるノウハウがこめられている点で，マン・マシンインタラクティブな操作上特に重要である．

本文で設計・試作の対象としたコマンド言語は，言語の完全さや機能よりもその分野における日常業務に立脚した書き易さに重点をおいた文法に従うものである点に特長がある．いずれにせよユーザの定義した文法はデータとして与えられ，

文法のチェックを行なって正しいコマンドであると認識されて，中間言語に翻訳され意味の処理手続きが実行されて所定の結果を得ることになる．

本文で記述することのできるユーザコマンドは次のような形式のものである．

$$\left.\begin{array}{l}\langle コマンド\rangle = \langle オペレーションコード\rangle \quad \langle オペランド\rangle \cdots \\ \langle オペランド\rangle = \langle サブオペランド\rangle \cdots \\ \langle サブオペランド\rangle = \langle トークン\rangle \cdots\end{array}\right\} \quad (1)$$

コマンドは処理の種類を指定するオペレーションコードと処理の対象や処理で必要なパラメータを定めるオペランドに分けられる．オペランドは必要に応じてサブオペランドに分割され，最終的には論理的に意味を有する最小単位であるトークンの列に分解される．トークンとしてはオペレーションコード，変数名，定数，区切り記号で構成される．なお，オペランドは省略される場合もある．

上記のコマンド言語の文法を入力記述するための定義した主なキーワードの一覧表を表1に示す．具体的なコマンド文法の記述は式(1)に示すコマンドの形式を各コマンドごとに，

```
$COMMAND ... ;
  ---                                  (2)
$END;
```

の形で記述される．

表1　キーワード一覧

キーワード	説明
$A	自然数
$C	識別子
$I	整数
$Q	’の付く文字列
$R	実数
$COMMAND	コマンド定義の開始の宣言
$GO	飛越し
$IF	条件付飛越し(の場合)
$IGNORE	オペランドとみなさない区切記号の宣言
$LEVEL1, $LEVEL2, $LEVEL3	コマンドのレベル
$SELECT	オペランドの選択
$SHORT	コマンド及びオペランドの圧縮形の入力を許す宣言
$L	ラベル
[ ]	区切記号
.	オペランドの終了
$END	$COMMANDの記述終了
$NOTIF	条件付飛越し（偽の場合）
;	文の終わり

例題として，

CHAN = <自然数> [-<自然数>] (3)
[, … <自然数> [-<自然数>][-<自然数>][, INC/<自然数>]

で与えられる式は，

```
$COMMAND CHAN $IGNORE ',' '_' '=' ;
    = ; $LEVEL1 ;
    $1; $A $SELECT 0[-$A]1 .;
        $NOTIF , $2 ;
        $NOTIF INC $1 ;
        / $A.;
    $2; $END
```

で記述することになる．

3．コマンドプロセッサジェネレータ

1および2で述べた種々の要求を満足するコマンド言語とその処理系の作成を簡単にするために，表駆動形式の汎用インタプリタやBNF記法のコマンド言語の文法を処理し，そのパーザを生成するコマンドプロセッサジェネレータが開発されている．[2] 本文で述べるコマンドプロセッサはこのうち後者のコマンドプロセッサジェネレータによる方式に属する．図1は2章で述べた文法に基くコマンドプロセッサの作成法を示す模式図である．システムの開発者（ユーザ）は，表1の文法書を参考にコマンド言語の規則をファイル化する．Gプログラムはこのファイルを読み込み，パーザの一部を構成する．GCプログラム（FORTRANソースプログラム）を出力する．次に開発者は，オペランドを

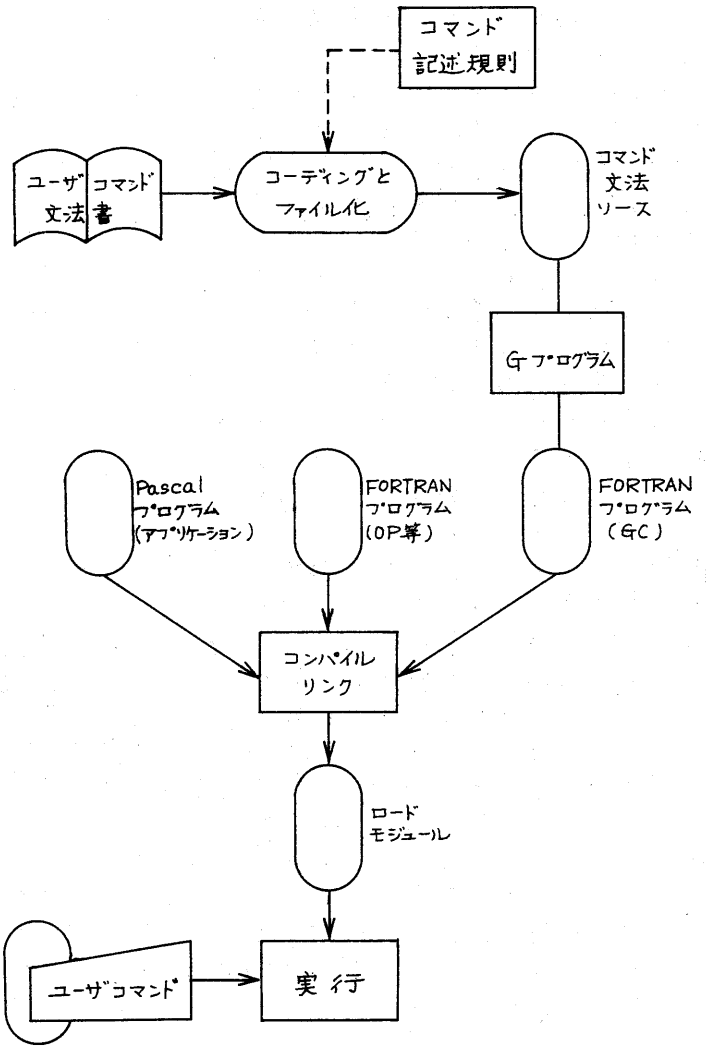

図1 コマンドプロセッサの作成手順

とり出して意味の解析ルーチンが動作しうる形式にリンクするためのOPプログラムや他の補助的プログラム（FORTRAN）をコンパイルし，ユーザの意味解析プログラム（Pascal ソースプログラム）をコンパイルした結果とリンクする．ロードモジュールが作成されるとユーザがコマンドを入力し，システムが稼動状態となる．以上の一連の処理はカタログされており，人間の介入のない形で実現できる．

図2はコマンドプロセッサの動作を示す流れ図である．ユーザの入力するコマンドは1コマンドずつ読み込まれ，GCプログラムでトークンに分解され，構文解析が行なわれる．そして，文法に合ったコマンドである場合には各トークンが文法に沿って中間言語に変換される．次にオペランドが存在する場合にはオペランドがOPプログラムによって取り出され，CEプログラムによってコマンドの処理が実行される．現在，コマンドの処理プログラムCEは，OPプログラムで取り込まれたパラメータを利用する各コマンドごとの独立したプログラムの集合体として開発者が作成することになる．

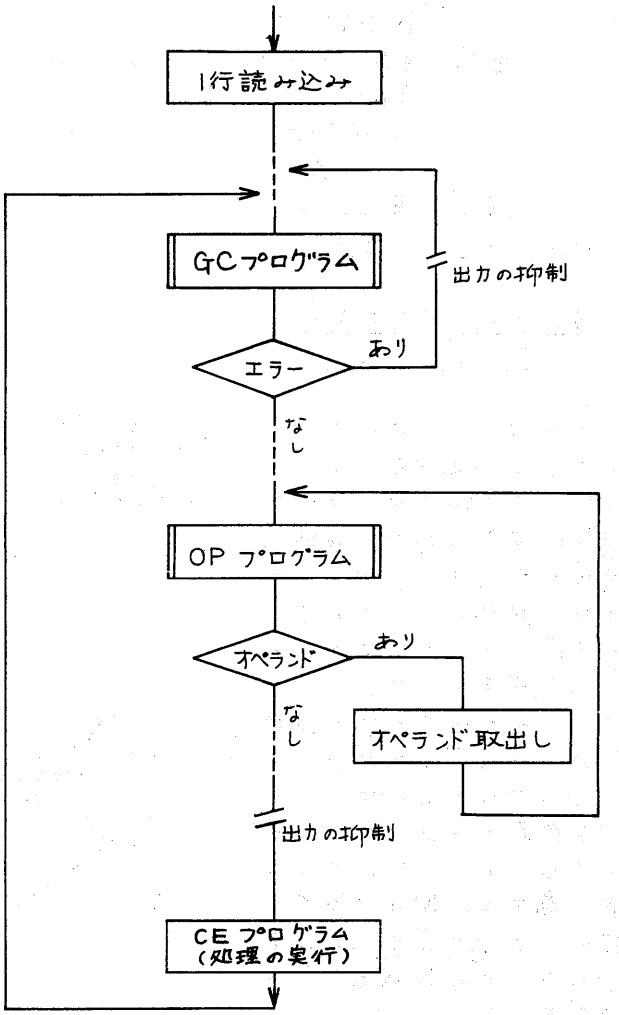

図2　コマンドプロセッサの動作

本文のコマンドプロセッサの特徴をまとめると次のようになろう．(i) ユーザの文法定義が容易である．(ii) ユーザの入力するコマンドは自動的にコマンドファイルに格納され，再実行や修正が自由に行なえる．(iii) 文法チェックにおけるエラー回数を計数し，定められた数になるまでは次のコマンドの解釈を続ける形にセットできる．(iv) 図形処理プログラムの実行を伴う場合が多くコマンド実行時の画面の制御を行う必要がある．このため，コマンドのエコーバック，出力の抑制が設定できる．(v) ユーザの定義したコマンド全体について分類を行ない，オペレーションコードの判別が出来る範囲でのコマンドの省略が可能である．(vi) パラメータの指定において順序が意味を持たない場合の指定が容易である．(vii) コマンドの種類や構造を反映したコンパクトなコマンドプロセッサが構成できる．

4. オイラーオペレータ群の試作への利用

3で述べた方式のコマンドプロセッサをオイラーオペレータプログラム群の開発とテストに適用した．ここでは，その例題に沿って効果を評価する．

オイラーオペレータは，境界によって立体を表現するデータ構造を用いたソリッドモデルを作成する基本的な手続き群である．そして，データ構造の個別的な特徴に直接依存しない形で部品形状を段階的に構成してゆく手法であり，互に素の立体の数を b，貫通する穴の数を h，面の数を f，面に属する穴ループの数を l，辺の数を e，そして頂点の数を v とするとき，オイラーの標数に関する，

$$v - e + f - l - 2(b - h) = 0 \qquad (5)$$

が常に成立する拘束条件を有している．Mantylaらによる GWB に沿って表2に示した12個のオイラーオペレータとその結果の図形表示，データ構造の出力ルーチンを加えたプログラム群を試作した．これらの組合わせで，基本的には多面体の形状を全て記述することが可能とされている．[3]

表2　オイラーオペレータの例

オペレータ	説　明
mbflv (B, F, V)	物体，面，頂点の作成
kbflv (B)	物体の削除
mev (V_1, E, V_2)	辺，頂点の作成
kev (E, V)	辺，頂点の削除
mef (V_1, V_2, F_1, F_2, E)	面，辺の作成
kef (E)	面，辺の削除
kemr (E)	辺の削除，穴ループの作成
mekr (V_1, V_2, E)	辺の作成，穴ループの削除
kfmrh (F_1, F_2)	面の削除，貫通穴の作成
mfkrh (F_1, F_2)	貫通穴の削除，面の作成
semv (E_1, E_2, V)	辺の分割，頂点の作成
jekv (E_1, E_2)	辺の合成，頂点の削除

各オイラーオペレータはデータ構造のブロック（物体，面などのブロック）の作成・挿入・削除そしてデータの設定を行う基本ルーチンの上に作成されており，トポロジカルな拘束条件の検査を含んでいる．オイラーオペレータのプログラムは Pascal で記述し，最大100行である．

開発段階からコマンドインタプリタを利用した主な理由としては，先ず，立体を構成する境界要素のデータブロック間の関係はポインタ変数を利用した複雑なリスト構造であり，その関係を図形的にあるいは名前による関係として適宜，表現することが有効である点があげられる．また，各オイラーオペレータの全ての使用法をつくしたテストデータをあらかじめ用意して個々のオペレータについて作業を進めることは困難であり，部分的に制御可能な形にオイラーオペレータ群を構成しておき，総合的・段階的に作業を進めることが効率的であること，将来的にオイラーオペレータが複合されて使用される場合に発見される誤りに対しても柔軟に対処できる点などを考慮した．

オイラーオペレータの文法を図3(a)に，また，そのコマンドプロセッサジェ

MBFLV (x, y, z)
KBFLV B/n
MEV { V/n1 (x, y, z) | (x, y, z) V/n1 } [E/n2 ({ CW | CCW })]

(a) 文法の一例

```
$SHORT;
/*-------------------------------------------------------------*/
$COMMAND MBFLV    $IGNORE  '(',')',',';
                  $LEVEL1;
                    ($R,$R,$R) ;.;
                  $END;
/*-------------------------------------------------------------*/
$COMMAND KBFLV    $IGNORE  '/';
                  $LEVEL1;
                    B / $A ;.;
                  $END
/*-------------------------------------------------------------*/
$COMMAND MEV      $IGNORE '(',')','/',',';
                  $LEVEL1;
                    $NOTIF V $1;
                       / $A ;.; ($R,$R,$R) ;.;
                       $GO $2;
                    $1;
                       ($R,$R,$R) ;.; V / $A ;.;
                    $2; $NOTIF E $3;
                       / $A ;.; ($SELECT 1[CW?CCW]1) ;.;
                  $3; $END
```

(b) 文法定義のデータ

図3 オイラーオペレータ用コマンド

オペレータプログラムへの入力データを図3(b)に示した。(部分) 文法の定義は70行(コマンド数15)であり，生成されたパーザ部分はFORTRANで430行である。

図4に三角柱に三角柱の貫通穴を有しつつの頂点について面とりをほどこした形状を作成するオイラーオペレータのコマンド列を示す。図5はその結果の図形出力である。

プログラムの開発中に，オイラーオペレータの操作で指定する引数の仕様が何回か変更された。その際にはオイラーオペレータに対応する手続きの引数を変更すると共に文法の記述を修正しコンパイル・リンクをやり直したが全体的なオペレータの動作に至るまで効率よく実施することができた。

このようなインタプリタ形式のコマンド言語を用いたプログラム開発は，部分的・段

```
MBFLV (0.0, 0.0, 0.0)
DSSET VIEW(10.0, 12.0, 8.0, 5.0)
MEV V/1 (-2.0, 2.0, 0.0) E/1(CCW)
MEV V/2 (2.0, 2.0, 0.0) E/1(CCW)
MEF V/3 E/2(CW) V/1 E/1(CCW)
MEV V/1 (0.0, 0.0, 1.0) E/3(CW)
                  . . .
MEF V/9 E/12(CW) V/7 E/11(CCW)
KEMR E/10 S
SEMV E/7 (-0.5, 0.5, 1.0) S
SEMV E/9 (0.5, 0.5, 1.0) P
SEMV E/4 (0.0, 0.0, 0.5) P
MEF V/10 E/14(CCW) V/11 E/15(CCW)
MEF V/11 E/15(CW) V/12 E/16(CCW)
MEF V/12 E/16(CW) V/10 E/7(CCW)
KEF E/7
KEF E/15
KEV E/16 V/4
MEV V/7 (0.0, 1.0, 0.0) E/13(CW)
MEV V/8 (-0.5, 1.5, 0.0) E/11(CW)
MEV V/9 (0.5, 1.5, 0.0) E/12(CW)
MEF V/13 E/20(CCW) V/14 E/21(CW)
MEF V/14 E/21(CCW) V/15 E/22(CW)
MEF V/15 E/22(CCW) V/13 E/20(CW)
KFMRH F/5 F/2
```

図4 オイラーオペレータプログラム

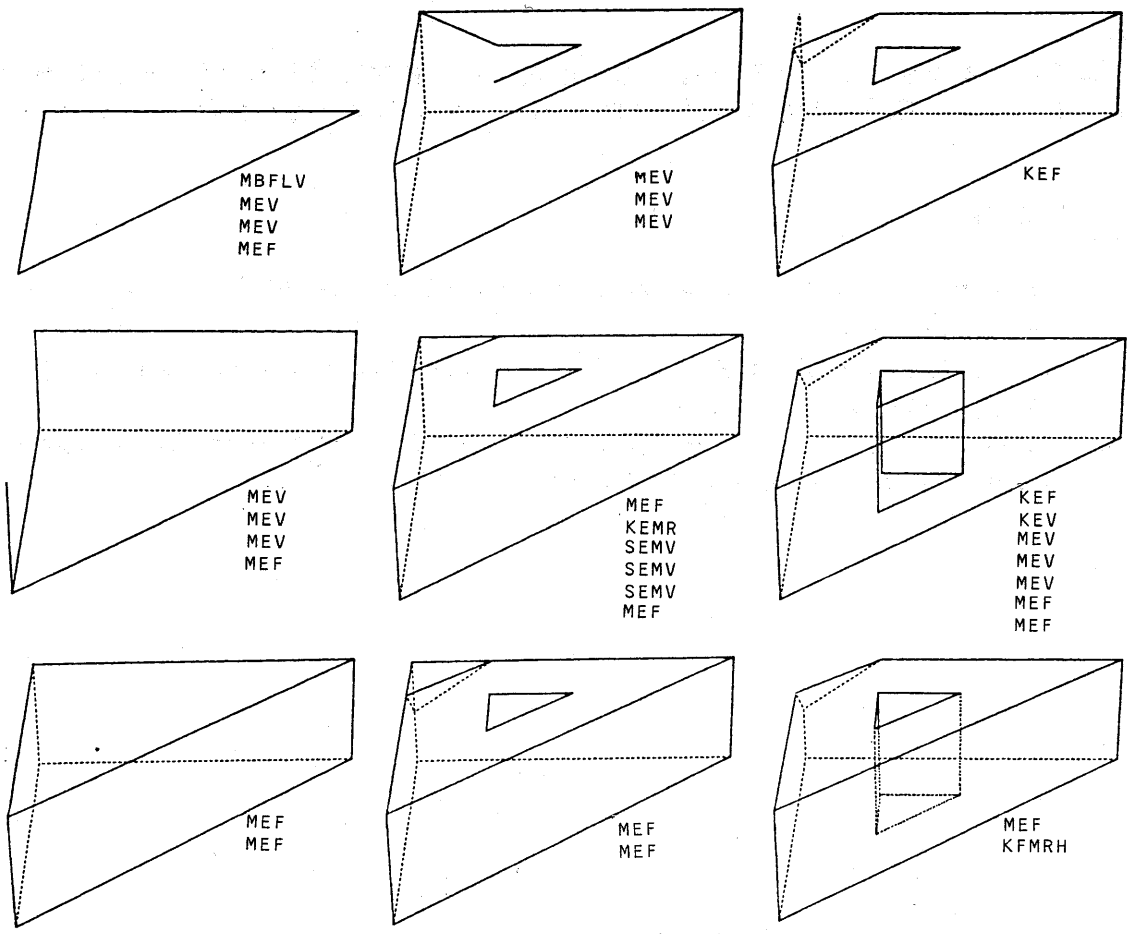

図5　オイラーコマンドによる形状の生成例

階的な開発やテストに向いており，コマンドプロセッサジェネレータが基本的なソフトウェアツールとして有効であることが確認された．

また，文法の記述がユーザ・アプリケーションの立場から設計されており，必ずしも厳密でない点や文法の記述の誤りに関する検査機能についての問題点が明確になった．

5．まとめ

システム開発者やユーザが自分の定義したコマンドを用いて自由にプログラムモジュール単位で作業を実行することの可能なコマンド処理系の作成法について検討し，コマンドプロセッサジェネレータの形で実現した．そして，その結果をオイラーオペレータプログラム群の作成に適用し，有効性を確認することができた．残された課題は多いが，当面，コマンド文法の記述の厳密化とエラー処理の拡充とファイルや図形入力装置の利用を含めた統一的な取扱いが重要であると考えている．

また，コマンド言語に対してはマクロ機能，変数や制御構造の付加が必要であ

る．

最後に，本研究について有益な討論を頂戴した機械技術研究所 関口博主任研究官ならびに4の実験を実行した東京理科大生 三木和男君に厚くお礼申し上げます．

参考文献

1) 木村文彦： CAD/CAMシステム構築のための基礎技術，精機学会 第2回サマーセミナ，1/23，(1983)．
2) S. C. Johnson & M.E. Leik： Language Development Tools，BSTJ，57-6，2155/2175，(1978)．
3) M. Mantyla & R. Sulonen：GWB：A Solid Modeler with Euler Operators，IEEE CG & A，2-2，17/31，(1982)．