グラフィクスとCAD 17－2
（1985・5・31）

形状モデラーと知識処理システムの結合方式
A Connection between Solid Modeller and knowlege Based System

今村 聡　小島俊雄　井上久仁子　関口 博
Satoshi Imamura　Toshio Kojima　Kuniko Inoue　Hiroshi Sekiguchi

機械技術研究所
(Mechanical Engineering Laboratory)

A construction method of knowledge based system with geometric data modeling/handling capability is presented. The system is composed of prolog program and a set of pascal boolean functions used as predicates in prolog program. The geometric data model developed is basically represented as gluing of linearly swept planar figures and is featured in that technical information as well as geometric properties can be obtained easily. The procedure how the system organized is also discussed following example of computer aided process planning.

1．はじめに

機械工業におけるCAD/CAMを代表とする生産向けソフトウェアシステムでは、形状のモデリングが中心的な役割を果たしてきたが、より高度な生産活動の支援は、形状と共に精度、組立関係などの技術情報の保持するデータモデルが必要であることはすでに多くの場で指摘されている。[1) 2)]

一方設計活動に対してより高度な支援を行ったり、設計情報から工程設計、作業設計、組立手順などの製造情報を生成するには、定型的な処理しか行われないプログラムでは不十分なことも明らかになってきた。この目的には専門家の知識を組み込んだ知識処理システムが有効だと考えられ、このプログラミング言語として論理型言語であるPrologが注目されている。そして、このような考え方に基づいた研究もいくつか報告されている。[3) 4)]

上記のようなアプリケーションにおいて、知識処理システムで必要とするデータは、ほとんど形状モデラー上にあり、これらをすべて述語論理で表現し、Prolog上に記述し直すことは手間がかかり効率も良いとはいえない。むしろ、Prolog上には必要な知識のみを置き、必要なデータはプログラムの実行時に形状モデラーに問い合わせる方が全体的な効率は上がるであろう。

本報は生産向け知識処理システムの試みに関し、具体的には、新しい生産向けデータモデルG-repの提案と、G-repとPrologの結合に関するものである。

2．システム構成

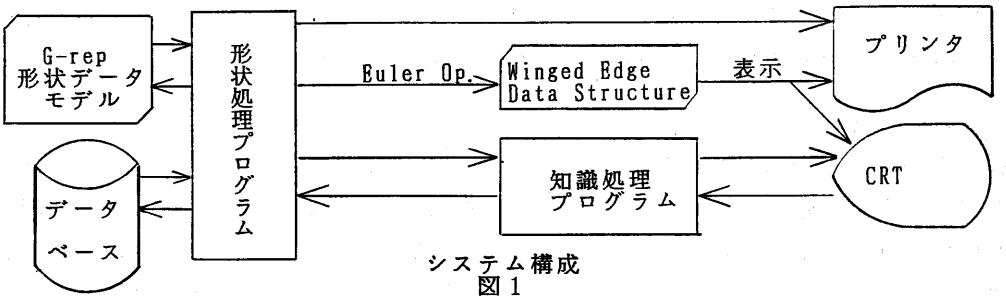
図1に試作したシステムの構成を示す。システムは主に形状処理プログラムと知識処理プログラムから構成されている。ユーザーは基本的には知識処理プログラムを介してシステムと対話し、それが形状処理に関係する場合はさらに形状処理プログラムを介してG-repデータ構造にアクセスすることができる。また形状データの変更結果などをCRT上に図形表示したり、テキストデータとしてプリンタに出力したり、あるいはデータベースにデータを格納

したりすることができる。形状処理は基本的にG-rep上で行われるが、表示や干渉チェックなど必要に応じてG-repデータ構造を全体的または局所的にCompound Euler Operation、Euler Operationを介してWinged Edge Data Structureに変換することができる。

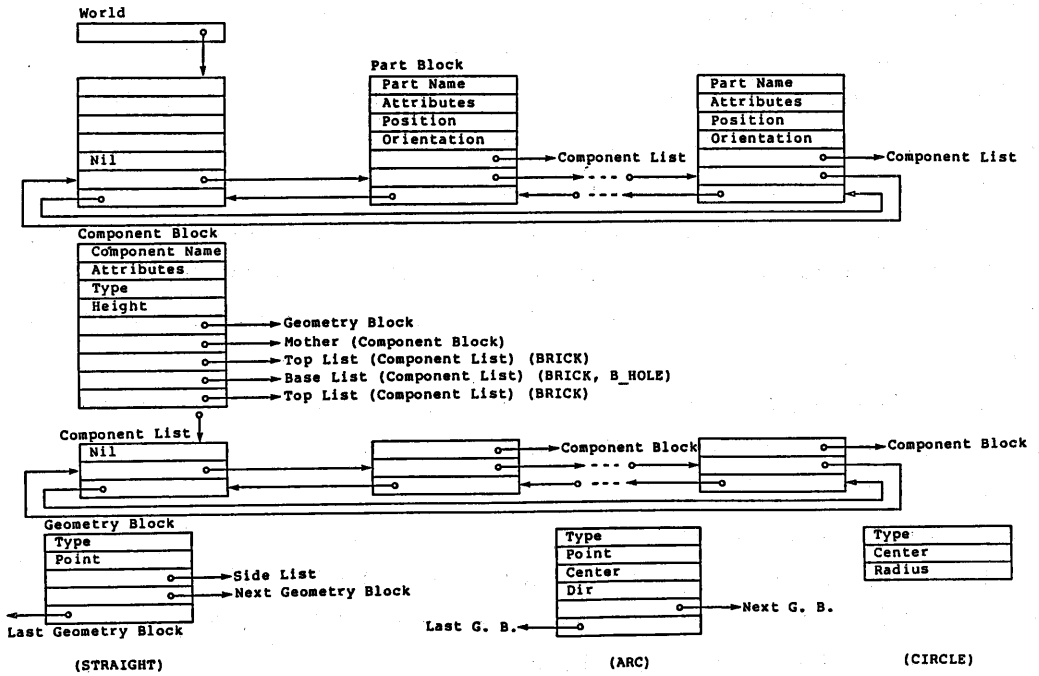
3．形状処理システム

3.1 G-rep (Gluing Representation))

G-repは２1/2次元形状に対象を限定し、平面図形を平行掃引して作られるComponentを貼り合わせて立体形状を表現するデータモデルである[5)]。図２にそのブロックの構造、ブロック間の接続関係を示す。G-repは精度、材質及び機能要素名（歯車、スプライン、カムetc）などの技術情報を便利に表現できるように工夫されている。データ構造はPart Block、Component Block、Geometry Blockの３種類のBlockとそれらをつなげるリストからなる。



システム構成
図１



G-repデータ構造
図２

次に各ブロックのデータ構造について述べる。

Part Blockの位置ベクトルは親ブロックの座標を原点としており、方向ベクトルはX、Y、Z軸回りの回転角で表現している。親ブロックの位置はワールド座標系を基準にしており、その底面はまずX-Y平面に平行に設定された後、方向ベクトルにより回転角が決まる。Part Blockを構成するBrickタイプのComponentはすべてComponent Listにより接続され、Component ListはComponent Pointerにより指定される。Part BlockはPointerにより双方向に接続される。

Component Blockは３種類のTypeをもち、Brickは平面形状を掃引して作られる固体形状、B_holeは止まり穴、T_holeは貫通穴を表わす。X_Y_ThetaはMother Componentの座標系を基準にしたX、Y位置及びZ軸の回りの回転角を表わす。Z軸位置は貼り合わせる場所により自動的に決まる。Heightはコンポーネントの高さ（掃引した長さ）を表わす。Geometry Pointerは平面形状を定義するGeometry Listを指定する。座標の基準となるMother ComponentはMother Pointerにより指定されている。Top List PointerはTypeがBrickの場合に存在し、上部に貼り合わされるComponentのListを指定する。Base List PointerはTypeがBrick、

B_holeの場合に存在し、底部に貼り合わされるComponentのListを指定する。Hole List PointerはTypeがBrickの場合にのみ存在し、Componentに接続するB_hole、T_holeのListを指定する。

Geometry BlockにはStraight、Arc、Circleの３Typeがあり、それぞれデータ構造が異なる。Circleは１つだけで円形状を定義するが、StraightとArcは１点を指定し、次の点まで結ぶ線分または円弧を定義し、複数のBlockにより一つの閉平面を構成する。Straightの場合、その線分を掃引して作られる側面に接続されるComponent Listを指定するSide List Pointerがある。円弧の場合は円弧の中心と円弧の向きをそれぞれCenter、Dirで指定する。Arc、Straightは互いに双方向ポインタにより接続される。

G-repはある意味でCSGとB-repの中間的な性格をもつ形状データモデルといえよう。現在は平面への貼り合わせのみが実現されているが、円筒面への貼り合わせも可能であり、またラウンドやフィレットなども局所的な属性として表現することができよう。データ量はWinged Edge Data Structureに比べ数１０分の１程度とかなり小さくてすむ。

3.2　形状データモデルのユーティリティプログラム

PrologからG-repに対して形状属性を問い合わせたり、形状や属性の変更を行わせるためのユーティリティプログラムをいくつか用意してある。それらには各データブロックを、生成、削除、変更する基本プログラムのほかに以下のようなものが用意されている。

ａ）属性抽出

指定されたPart Blockに指定されたAttributeを持つComponentがあればtrue、ないならばfalseを返す。

入力：PART_NAME
　　COMPONENT_ATTRIBUTE
出力：検出フラグ

このプログラムはAttributeに記述された

技術情報をPrologに伝えるために用いる。

b）属性の変化

指定されたPart Blockに指定されたAttributeをもつComponentがあれば、新しいAttributeの値に変化させ、trueを返す。

入力：PART_NAME
COMPONENT_ATTRIBUTE(old)
COMPONENT_ATTRIBUTE(new)
出力：フラグ
結果：COMPONENT_ATTRIBUTEが変化する

c）線対称

指定されたComponentの平面形状が線対称かどうか判断する。線対称であればFlagをtrueにし、対称軸の方程式の係数を返す。線対称でなければ単にfalseを返す。

入力：COMPONENT_NAME
出力：検出フラグ
線対称軸方程式の係数

d）点対称

同じような判断を点対称について行う。

入力：COMPONENT_NAME
出力：検出フラグ
点対称の中心点座標

e）内接円

指定されたComponentの平面形状の内接円を求める。内接円が求まればFlagをtrueにし、中心座標、半径を返す。求まらなければ単にfalseを返す。

入力：PART_NAME
COMPONENT_NAME
出力：検出フラグ
内接円の中心座標、半径

f）外接円

同じことを外接円について行う。b）～e）はいずれも指定されたComponentの平面形状を定義するGeometry Listのデータをもとに幾何学的計算を行い、出力結果を求めている。

入力：PART_NAME
COMPONENT_NAME
出力：検出フラグ
外接円の中心座標、半径

g）Componentの消去

指定されたComponentをComponent List、Top List、Base List、Hole Listからはずす。はずされたComponentはGarbage Component Listにつなげられ、必要に応じて参照が可能である。

入力：PART_NAME
COMPONENT_NAME
出力：フラグ
結果：指定したComponentを消去する

h）回転体への変形プログラム

指定されたComponentのタイプはBrickの場合は外接円を求め、外接円を掃引した形状にComponentを変化させる。タイプがT_holeまたはB_holeの場合は内接円を求め、内接円を

入力：PART_NAME
COMPONENT_NAME
出力：フラグ
結果：Componentの平面形状が内接円または外接円の形状に変化する

掃引した形状にComponentを変化させる。形状の変化は、外接円（内接円）の中心座標と半径をもとに新たにCircleタイプのGeometry Listを作り、これをComponentのGeometry Listと交換することによって行う。

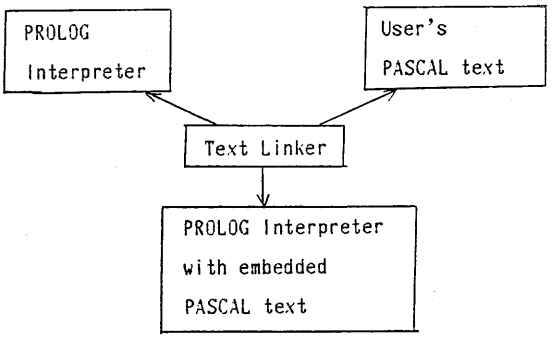
以上のユーティリティプログラムによりProlog処理系の中でG-repのもつデータを利用し、またG-repのデータ構造を更新させることができる。またユーザーは必要に応じてユーティリティプログラムを作成し、追加することもできる。

4．知識処理システム

4.1　PrologとPascalの結合

知識処理はPrologの処理系として実現されている。利用した処理系は東京理科大学情報科学科井上研究室で開発されたProlog-Likaである。[6] これはPascalで書かれたProlog InterpreterでPascalのProcedure、Boolean Functionをpredicateとして呼び出すことができ、またProlog内で実数を扱える点に特長がある。Prolog-Pascal間の引数には整数、実数、文字列が許される。

次にPrologとPascalのプログラムをリンクする手順を図３に沿って説明する。Pascalプログラムは宣言部とPrologで呼び出すサブルーチンを集めたファイルとそれ以外にサブルーチンのみを集めたファイルに分けておく。Text LinkerはUserのText Fileにアクセスし、連結の制限条件をチェックした後、Prolog InterpreterとUser Programをファイル上でマージし、Pascalコンパイル・リンクを行い、Prolog Interpreterを作り上げる。このようにしてPascalで記述されたUser Programをbuilt-in関数として使える。



PrologとPascalのリンク
図３

4.2　バックトラック処理

PrologではPredicateのユニフィケーションに失敗した場合、自動的にバックトラックを起こす。これがPascalで記述された形状処理用ユーティリティプログラムまで戻ってきた場合、G-repデータ構造を更新前の状態に戻す必要が生ずる。これは更新前のデータの状態を別の場所に保持しておき、バックトラックにより再びプログラムが呼び出された場合、元の状態に復帰させることによって行っている。

このメカニズムを実現する方法についてより具体的な例によって説明することにする。たとえば、貫通穴を埋めてなくす動作を行う場合は、Component の消去用のプログラムによりタイプがT_holeのComponentを各Listからはずし、Garbage Component Listにつなぐ。

Garbage Component Listは次のような構造をもつ。

GARBAGE_COMPONENT_LIST

COMPONENT_NAME : character string;

TOP_COMPONENT_POINTER : pointer;

BASE_COMPONENT_POINTER : pointer;

HOLE_COMPONENT_POINTER : pointer;

COMPONENT_POINTER : pointer;

LAST、NEXT : pointer.

Top_Component_Pointer、Base_Component_Pointer、Hole_Component_Pointerはそれぞれが所属していたTop_List、Base_List、Hole_Listをもつ Componentを示す。Component_Pointerは、もとのデータ構造からはずされたComponentを示すためにある。Last、Nextは双方向に接続するためのポインタである。

バックトラックが発生し、Component消去用プログラムが再び呼び出されると、まず指定したComponentを消去しようとする。しかし、それはすでに消去されているので、この手続きは失敗し、次にGarbage Component Listを調べる。ここにある場合は、Componentをもとのデータ構造に戻し、Boolean Functionとしてはfailを返す。ここにもない場合は単にfailを返すことになる。

このようにしてバックトラック時のデータの復帰動作が完了する。

別の例としてスプライン加工やキー溝加工などを行う前の形状を求めるために、外接円または内接円を求めて形状変化を行った場合は、交換したGeometry ListはGarbage Geometry Listにつなげられる。それは次のような構造をもつ。

GARBAGE_GEOM_LIST

COMPONENT_NAME : character string;

GEOMETRY_POINTER : pointer;

LAST、NEXT : pointer.

バックトラックが生じた場合は、上記と同様な手続きをふんで、データの復帰が行われることになる。このようにして、PrologとG-repのデータ構造の整合性を常に保つことができることになる。

5．例題

丸物の工程設計の例題を図4に示す。このプログラムでは加工形状から素材形状に達するまでの加工工程を逆に創生する。工程設計解は、基本的には、このプログラムが求めた加工工程を逆にたどれば求められる。加工工程の順序決定はProlog上の9つのルールによっており、それらは加工優先順位番号とその順位で加工する加工形状単位のリストをもつ。

加工形状単位とはこの例題で定義したもので、表1に示すとおりである。F_Screwのように加工形状単位の先頭にF_のついているものは仕上加工（finishing）を意味する。これらはいずれも、ある一定の加工法によって加工される形状単位で、対象部品に存在するか否かにより加工工程が変化する。しかしこれらの存在を形状のみから判断するにはパターン認識

的手法が必要であり、困難かつ時間も要する。そこで加工形状単位は、Component_BlockのAttrib- uteに技術情報として記述している。このようなことは、機械図面上でも注釈として行われており、不自然なことではない。

プログラムはStart(*X)の*Xに対象とする部品名を代入して動作を始める。Lathe(*X)は*Xが丸物部品であれば成功する。Findall(*Y,Proc(*Y,*X),*L)は、Proc(*Y,*X)を満たす*Yすべてを集めたリスト*Lを求め、Proc(*Y,*X)は対象部品名*Xがもつ加工形状単位を*Yに返す働きをもつ。

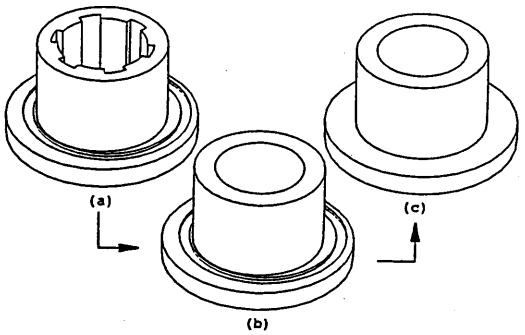
Apply_RuleはRuleを1から順に適用し、各順位のRuleに記されている加工形状単位のList*Mと*LをOperationに渡す。Operationは*Mと*Lの共通部分を順次*Lから取り除き、同時にShape_ConvによりG-repに働きかけてデータを変化させる。その方法は加工形状単位により異なり、Screw、Spline、Gear、Cam、Plane、Sp_Groove、Key、Cy_Screwについては、TypeがB_holeまたはT_holeの場合は最大内接円に、Brickの場合は最小外接円に形状を拡張させる。Hole、Tap、Ec_Groove、Nc_Grooveは形状の消去を行う。Cylinderはタイプがbrickで中心軸が一致しているComponentのうち最大径のものにすべての径を揃える。F_で示される仕上加工につ

```
START(*X) :-
        LATHE(*X),
        FINDALL(*Y,PROC(*Y,*X),*L),
        APPLY_RULE(1,*L,*X).
APPLY_RULE(10,*,*).
APPLY_RULE(*NO,*L,*NAME)  :-
        RULE(*NO,*M),
        OPERATION(*M,*L,*NAME),
        ADD(*NO,1,*NO1),
        APPLY_RULE(*NO1,*L,*NAME).
OPERATION(( ),*,*).
OPERATION((*HM.*TM),*L,*NAME)  :-
        EFFACE(*HM,*L,*L1),
        SHAPE_CONV(*HM,*NAME),
        WRITE((*HM, OF ,*NAME)), NL,
        OPERATION(*TM,*L1,*NAME).
OPERATION((*HM.*TM),*L,*NAME) :-
        OPERATION(*TM,*L,*NAME).
RULE(1,(F_SCREW,F_SPLINE,F_GEAR)).
RULE(2,(SCREW,SPLINE,GEAR)).
RULE(3,(F_CAM,F_CRANK)).
RULE(4,(F_CYLINDER,F_PLANE)).
RULE(5,(HOLE,TAP)).
RULE(6,(PLANE,KEY,SP_GROOVE,
        EC_GROOVE,NC_GROOVE)).
RULE(7,(CAM,CRANK)).
RULE(8,(CY_SCREW)).
RULE(9,(CYLINDER)).
```

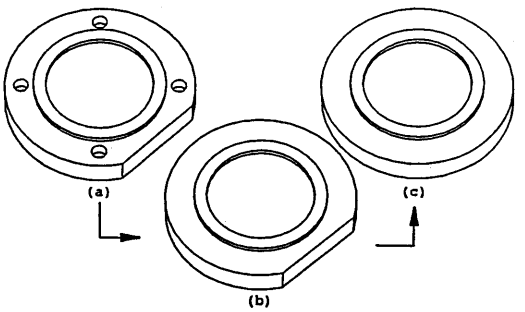
図4　丸物工程設計プログラム

表1　プログラムで用いた加工形状単位

SCREW	ねじ　切り	HOLE	中心軸の異なる穴
GEAR	歯　　車	PLANE	外周面の平面
CRANK	クランク軸	CY_SCREW	旋削ねじ切り
SPLINE	スプライン	SP_GROOVE	外周面のラセン溝
KEY	キ　ー　溝	EC_GROOVE	端面の偏心溝
TAP	ね　じ　穴	NC_GROOVE	端面の非円形溝
CAM	カ　　ム	CYLINDER	内外周面旋削



形状変化の例1
図5



形状変化の例2
図6

いては、ComponentのAttributeのみを変化させる。

形状の変形プロセスの例を図６、図７に示す。

６．まとめ

１）知識処理システムとの相性の良い形状データモデルG-repを提案した。これは技術情報を格納でき、形状情報の抽出や形状の変化、復帰が容易という特徴をもち、また必要に応じてWinged Edge Data Structureに変換して表示などを行うことができる。

２）Prologと形状モデラの一結合方式について述べた。これによりPrologが直接形状モデラーのデータを取り入れてプログラムを実行することが可能になった。CADシステムの知能化、工程設計など設計情報から製造情報を生成する過程の自動化などに有効であろうと考えている。今後はより多くの例題を手がけることにより、問題点を明確にして行かねばならない。

おわりにProlog_likaを研究開発し、提供して下さった、東京理科大学情報科学科井上研究室の有山隆史氏、小林生明氏に厚くお礼を申し上げます。

参考文献

１．穂坂衛：CAD/CAMにおける情報の役割、情報処理 Vol24 No1 pp3～10(1983)

２．木村文彦：FAにおける情報処理技術の役割、情報処理 Vol25 No4 pp283～295(1984)

３．上原貴夫：CADにおけるProlog、情報処理 Vol25 No12 pp1373～1379(1984)

４．森、光本、藤田、後藤：配線エキスパートシステム、情報処理学会第28回全国大会 pp1073～1074(1984)

５．T.Kojima et al."Development of Machine Part Model Consistent with Designer's Shape Description in CAD CAM System" Annals of the CIRP Vol33/1/1984

６．松波、桐生、栗原、白石：東京理科大学情報科学科昭和58年度卒業論文「拡張Prologシステム」の試作