International Journal of Computer Integrated Manufacturing
Vol 7, No 2, 77-99, 1994

Process planning by recognizing and learning machining features

A. K. W. CHAN and K. CASE

Abstract. We present two methods for process planning of 2.5D machined parts. The first method is based on feature recognition from a 3D model. We embedded the shape and the machining method of two generic classes of machining features in a set of OPS5 rules to form a machining feature recognizer. When successfully recognizing a machining feature, machining instructions, in terms of the tool entrance face, drive face and part face, for cutting the machining feature will be generated and further processed to produce NC codes.

The second method is based on learning the shape and the machining method of the machining feature. When a machining feature cannot be recognized by the former feature recognizer, the user can use the machining feature as a positive training example to instruct the system about the tool entrance face, drive face and part face of the machining feature. The system then builds a new rule, using the boundary shape of the unrecognized machining feature as the rule's matching condition and the acquired machining instruction as the rule's action. The new rule can be used subsequently for process planning of machining features that have shapes similar to the memorized one.

1. Introduction

Process planning has been the focus of much CAD/CAM automation research which has led to the emergence of the 'feature' concept. The abstraction of feature tends to depend on the researcher's points of interest. In the context of process planning of machined parts, however, feature can be regarded as a collection of shapes and technological attributes (Shah *et al*. 1988) associated with the machining processes of the machined parts.

Based on the ways of creating and using features, two approaches used by researchers can be discerned. One is called feature recognition and the other, the design by feature method. The former approach has been used by researchers (Grayer 1977, Joshi and Chang 1988) to recognize machining form features from the computer model with the assumptions that the features are present in the model and that they can be identified and extracted for further communication to support other design and manufacturing activities. The latter approach has been adopted by researchers (Descotte and Latcombe 1984, Cutkosky *et al*. 1988) who emphasize the incorporation of feature information more explicitly in a computer model during the design process so as to minimize or eliminate the expensive feature recognition process.

From the design and manufacturing standpoints, features which are relevant to functional performance may not be relevant to manufacturing and vice versa. For instance, the use of strengthening ribs is a common design feature to reinforce the structural strength of castings. However, the ribs can be obtained by methods, such as casting or machining the surrounding material of the ribs, or can be fabricated by welding. Many design-by-feature-based prototype systems (Turner and Anderson 1988) resolved this design and manufacturing features mapping problem by using a predefined set of manufacturing oriented features for design purposes. As a consequence, these systems tend to limit the designer's ability in designing with design features and restrict the choice of manufacturing processes.

Some feature-recognition-based approaches (Henderson 1984, Requicha and Vandenbranke 1989) employed AI techniques in their work, yet they are basically relying on a finite set of primitive machining feature templates, such as hole, pocket, etc., together with their syntactic interaction patterns for recognition. Since primitive machining features can combine in an unpredictable manner to form compound machining features, pre-defining the syntactic pattern of feature combination will limit the system's recognition power. To deal with the process planning of compound machining features, we believe that the system should possess learning ability so that its process planning capability can grow.

Authors: A. K. W. Chan, Department of Mechanical Engineering, University of Hong Kong, Pokfulam Road, Hong Kong; K. Case, Department of Manufacturing Engineering, Loughborough University of Technology, Loughborough, Leicestershire, LE11 3TU, UK.

Four basic machine learning strategies have been described in (Cohen and Feigenbaum 1982). They are:

1. Rote learning, in which the environment provides explicit information which is then memorized by the learning agent without much inference.
2. Learning from instruction, in which the learning agent has to hypothesis the missing details from the provided information, then transform and integrate the new knowledge with prior knowledge so that it can be used in the future.
3. Learning from examples, in which the learning agent develops a general scheme for handling future situations after it has encountered a number of positive and negative examples of situations.
4. Learning by analogy, in which the learning agent extends the scope of its existing knowledge by detecting the similarities and differences between the old and new situations.

Our research is on the automation of process planning of machined parts which consist of compound machining features. We used the PADL-2 solid modeller (Volecker and Rosenberg 1986) as a design front end for modelling the material stock and the finished part. By a Boolean subtraction operation between the stock and the part models we obtained the boundary representation (BRep) of the cavity volumes of material. We integrated the PADL-2 modeller with the rule-based VAX-OPS5 AI environment to facilitate the implementation of two process planning methods. The first method uses a set of OPS5 rules to recognize machining features from the cavity volume model and generate appropriate machining instruction when recognition is successful. The second method attempts to use the machine learning strategies 1, 2 and 3, as mentioned above, to handle compound machining features that cannot be recognized by the first method.

The sequel of this paper is presented in the following manner. Section 2 introduces the architecture of our experimental system. Section 3 describes the representation of the cavity volume model in the solid modelling environment. Section 4 describes the transfer of the cavity volume information from the solid modelling domain to the OPS5 AI environment. Section *5* defines the representation of two categories of machining features, their recognition method and rules. Section 6 describes the post-processing of the recognition result to generate NC machine codes. Section 7 describes the mechanism of the learning method. Section 8 concludes our work.

2. System's architecture

Figure 1 illustrates our method of linking the VAX-OPS5 AI environment with the PADL-2 solid modeller. The VAX-OPS5 AI environment is an extended implementation of the OPS5 language (Forgy and McDermott 1977) which consists of a database and production rules that manipulate the database. Data in the database is represented by the Object-Attribute-Value (O-A-V) triplets method. The VAX-OPS5 run-time system controls the execution of OPS5 programs. It consists of a recognize-act cycle, command interpreter and run-time compiler. During the recognize phase, the system compares working-memory elements of the database with the condition elements on the left-hand side of each rule. As the left-hand sides of rules are satisfied, the run-time system creates a conflict set that contains records of the working-memory elements that match the condition elements of a rule. Each record includes a rule name and a list of the time tags of working-memory elements that match the condition elements on the rule's left-hand side. The run-time system uses either the Lexicographic-Sort (LEX) or the Means-Ends-Analysis (MEA) conflict resolution strategy to order and select the records in the conflict set. Both strategies apply in the order of the following rules: refraction, recency and specificity (McDermott and Forgy 1978). The MEA strategy is similar to the LEX strategy except that it includes an extra step after refraction, which orders the records in the conflict set according to the recency of the working-memory element matching the first condition element in each rule. Our testbed system uses the MEA strategy as the most important condition element is always placed first on the left-hand side of each rule. After the run-time system selects a record from the conflict set, the recognize-act cycle enters the act phase. During this phase, variables assigned in the rule's left-hand side are bound to values and the actions on the right-hand side of the rule to which the record refers execute. The VAX-OPS5 command interpreter is used to control the execution of a program interactively. By virtue of the run-time compiler, new rules can be added to an executing program.

We installed two new input command parsers, one in front of PADL-2 and the other in the AI environment, for filtering our system commands. When the command, 'ALEX', is intercepted in the solid modeller, the system will switch to the command parser of the AI environment. The command, 'OPS', is for going into the OPS5 command interpreter and the command, 'MODEL', is for returning from the AI environment to the solid modeller. The 'new definitional statement' decision block is to catch the use of a group of specially designed non-PADL solid expression statements which have to be converted into the PADL-2 legal syntax before passing to the PADL-2

parser. The purpose of implementing the non-PADL statements is to provide an optional input format to facilitate the solid modelling process.

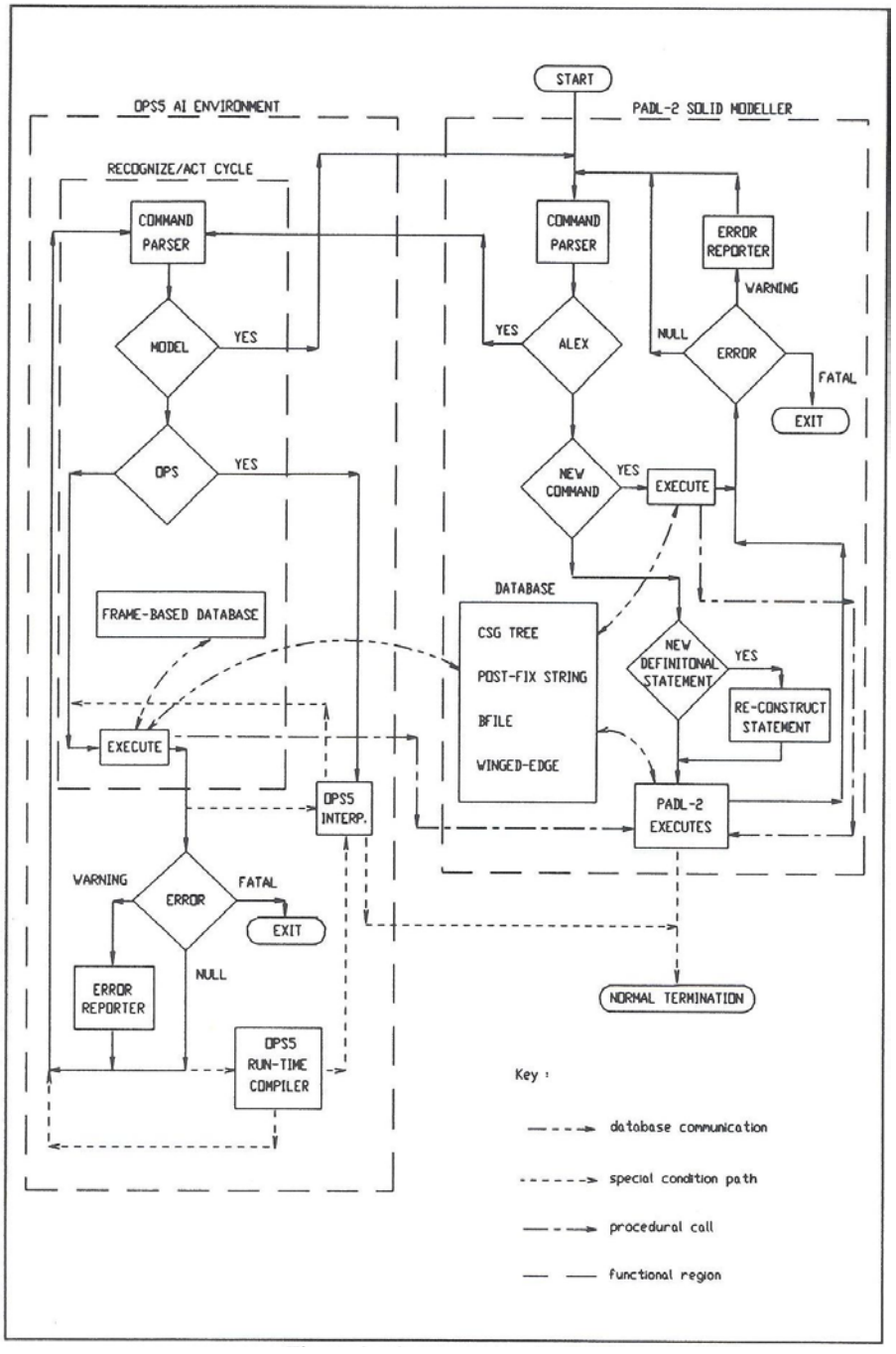


Figure 1. System's architecture

3. BRep in the solid modeller

PADL-2 records the construction history of a solid model, in terms of the type and size of simple solid primitives, Boolean and spatial operators used, in a constructive solid geometry (CSG) binary tree data structure. When the explicit BRep of the solid model is required, the CSG data is manipulated by a set of boundary evaluation routines. The generated BRep is maintained internally in a hierarchical boundary file (bflle). The design of the bflle is based on the so-called maximal-face (Silva 1981) scheme in which surface regions (or faces) of the same halfspace are collectively addressed by a single logical pointer. For many applications, this maximal-face representation scheme is undesirable, especially from the standpoint of machining process planning which demands distinction of individual face entities. Moreover, the hierarchical nature of the bflle also restricts the design of procedural routines as the bfile information has to be addressed in a top-down manner.

To overcome these problems, the hierarchical bfile representation is converted into the modified winged-edge data structure representation (Weiler 1985). In the winged-edge data structure, the BRep is based on the connected-face (Silva 1981) scheme which is congenial with human interpretation of an engineering object boundary. The procedures of converting bfile to a winged-edge data structure can be found in (Chan and Tan 1988).

3.1 *The cavity volume model*

In our feature recognition we used the cavity volume model because we considered that depressions identified on a part model may not necessarily require machining operations as they may have been produced by some former manufacturing processes. Recognizing machining features solely based on the part model also can hardly identify such aspects as the need of a surface milling operation. For machine understanding of the net machining regions, we believed that the stock information is essential and the use of the regularized Boolean subtraction operator between the stock and part models can simplify the task of revealing the genuine machining regions. Although this method is at the expense of boundary evaluation, we considered that the shortcoming can be compensated by the advantage that the cavity volume model, hereafter called the MC**_VOL,** offers explicitly the complete geometrical and topological bounding envelope of the net machining regions.

Based on the user specified stock and part names, our system performs the regularized Boolean subtraction and boundary evaluation operations to obtain the intermediate bflies of the stock, part and MC**_VOL** models, and afterwards, performs their bfile-to-winged-edge data structure conversion. The sequence for establishing the winged-edge data structures of stock, part and MC**_VOL** is illustrated in Figure 2. Figure 3 is a condensed picture of the entire BRep structures which are maintained in the system together with the PADL-2 inherent data representation methods.

As MC**_VOL** may represent several spatially disjointed sub-volumes, the sub-volumes are virtually separated by checking their face connection relationship. A pointer (MC_VOL_CHILD in Figure 3) is assigned for each identified sub-volume to provide an access path to its own sets of faces, edges and vertices. The pointers to the sub-volumes are stored in a list, MC_VOL_LIST, as shown in Figure 3.

3.2 *MC_VOL face classification*

The surface boundary of MC_VOL can be modelled as follows:

$$\text{bMC\_VOL} = b(S \leftrightarrow P)$$
$$= (bS \cap cP) \cup (iS \cap bP) \cup [bS \cap bP \cap k(iS \cap cP)]$$

where b :: = the surface boundary point set of
S :: = stock solid model
$\leftrightarrow$ :: = regularized Boolean difference
P :: = part solid model
$\cap$:: = intersects
c :: = the complement point set
$\cup$ :: = unions
i :: = the interior point set
k :: = the closure point set

(For rigorous understanding of the above expressions please refer to Requicha and Tilove 1978).

For a valid machining operation, P is a proper sub-set of S :. $(bS \cap cP) \neq 0$
$(bS \cap cP)$ is defined as the *tool entrance boundary* (TE_FACE)
Again since P is a proper sub-set of S :. $(iS \cap bP) \neq 0$
$(iS \cap bP)$ is defined as the machining generated boundary (MC_FACE)

Since bS and bP are two-dimensional $(bS \cap bP)$ is either :
(a) null (i.e. disjoint)
(b) two-dimensional (overlap)
(c) one-dimensional (intersect)

The last term, $[bS \cap bP \cap k(iS \cap cP)]$ excludes cases (a) and (b) and refers to those relevant point sets that lie on the

```
; Scsg and Pcsg are CSG structures of S and P
Procedure Build_winged-edge_data_structure (Scsg, Pcsg)
; deduce CSG structure of MC_VOL
        MC_VOLcsg ← Scsg <-> Pcsg
; perform boundary evaluation to obtain their bfiles
                Sbfile ← BEval (Scsg)
                Pbfile ← BEval (Pcsg)
                MC_VOLbfile ← BEval (MC_VOLcsg)
; perform m-face to c-face conversion and build winged-edge structures
                Swed ← MFace_To_CFace (Sbfile)
                Pwed ← MFace_To_CFace (Pbfile)
                MC_VOLwed ← MFace_To_CFace (MC_VOLbfile)
end /procedure/
```

Figure 2. Building the winged-edge based BRep data.

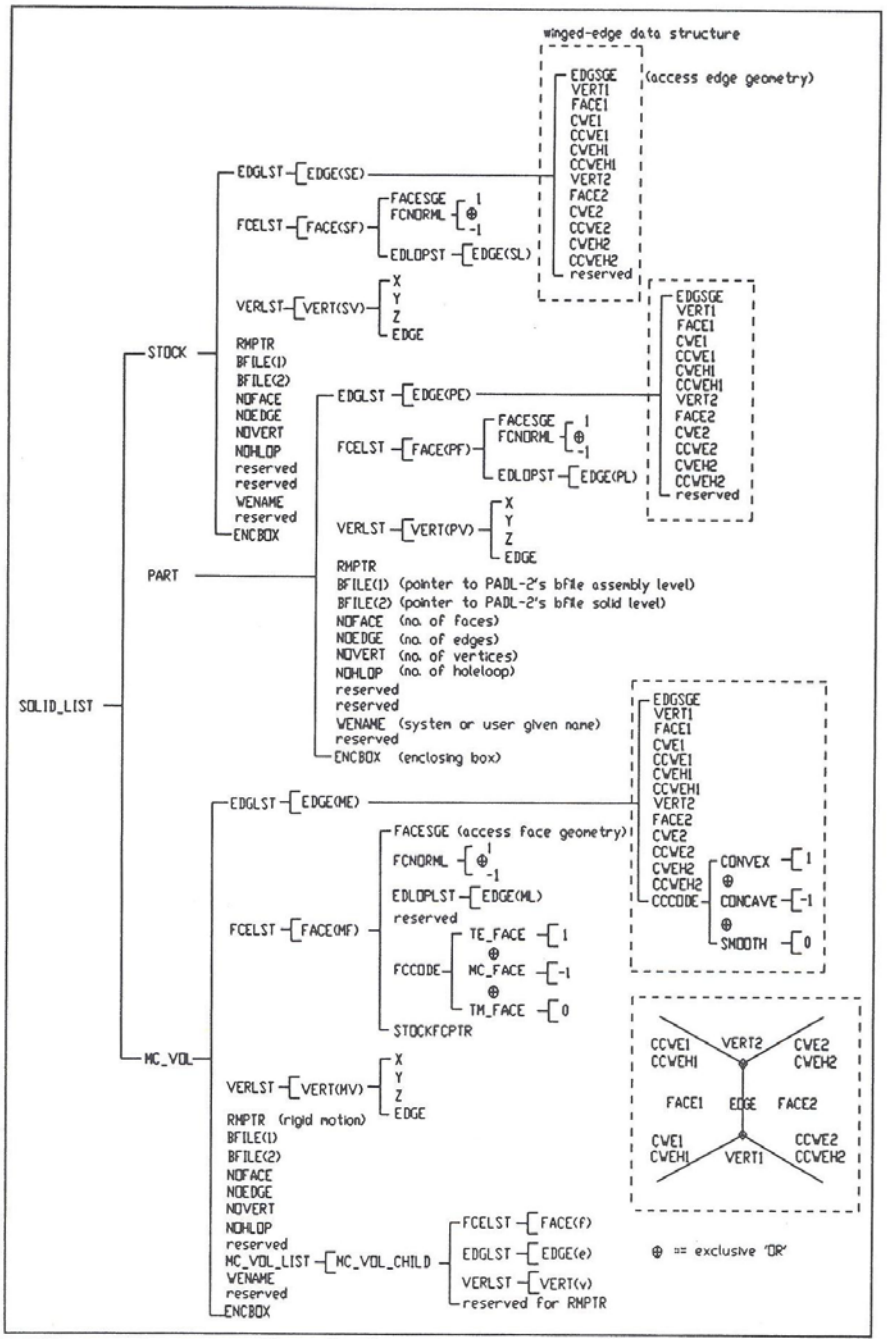


Figure 3. System's BRep structure

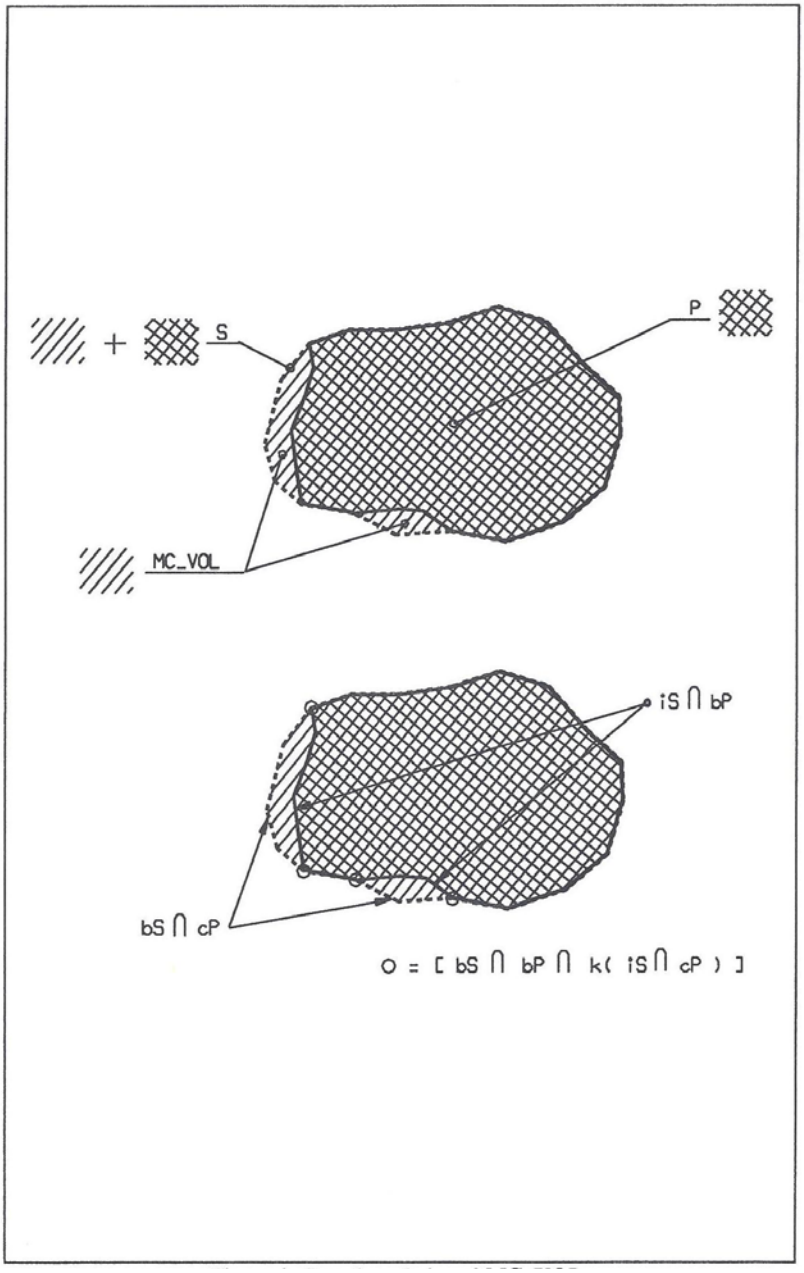


Figure 4. Face boundaries of MC_VOL

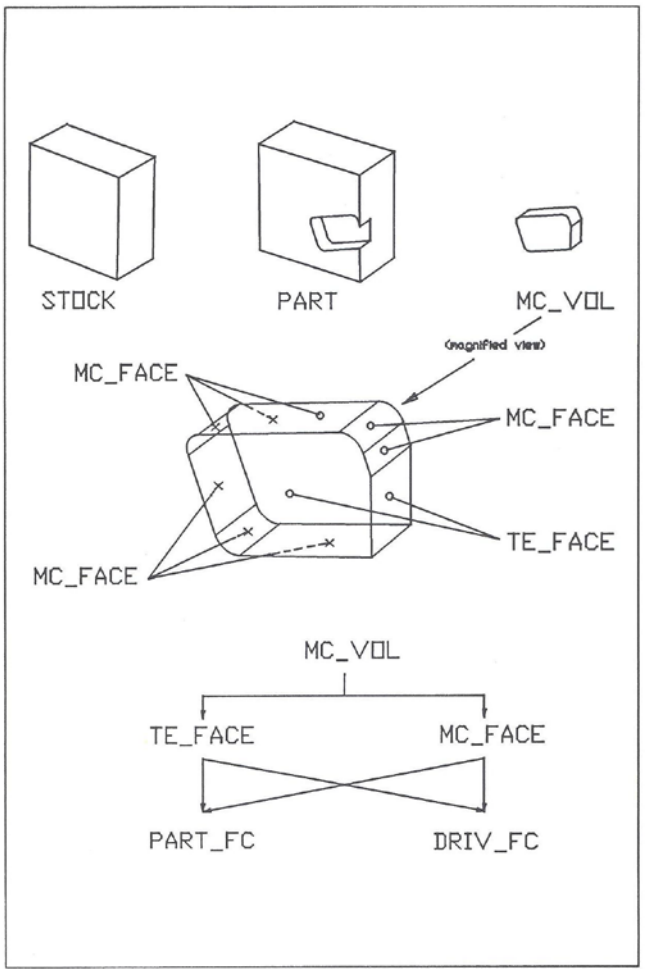


Figure 5. Face taxonomy of MC_VOL

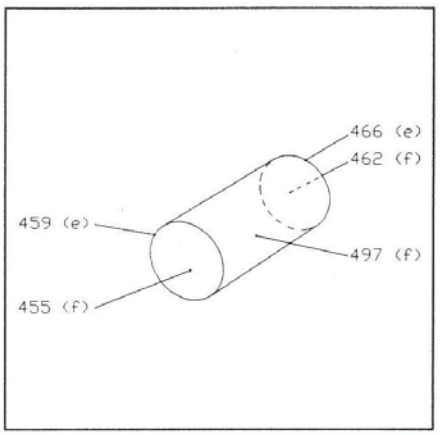
intersection edge formed by bS and bP as illustrated in Figure 4.

Based on the surface boundary expression shown above, each face of MC_VOL is classified as TE_FACE or MC_FACE by testing the face's halfspace parameters and boundary edges with those of the faces of *S*. The face classification result is stored in the integer variable, FCCODE, of each face as shown in Figure 3.

Each edge of the MC_VOL is also classified into either, convex, concave or smooth indicating that the solid angle θ between two adjacent faces of MC_VOL is either $0° < \theta < 180°$, $180° < \theta < 360°$ or $\theta = 180°$ respectively. The edge convexity is augmented in the BRep of MC_VOL (Figure 3) for feature recognition purpose.

3.3 *MC_FACE classification*

With the above face classification, the MC_VOL can be considered to be bounded by a collection of TE_FACE and MC_FACE. Using the drive face and part face concept of the Automatically Programmed Tools (APT) NC language, it can be conceived that the TE_FACE or MC_FACE can become a part face or a drive face during machining operation. This idea




```
(SOLID ^IDENTY 501 ^SYSNAM MV_1 ^RGMOTN 0 ^CLASCD NIL ^FACLST 500
 ^UTFLG1 NIL)

(FACLST ^IDENTY 500 ^FACNUM 3 ^UTFLG1 NIL ^FACESS 497 462 455)

(FACE ^IDENTY 497 ^FACTYP CYL ^CLASCD MC_FACE ^FACNOR 1
 ^FACLOP 498 ^UTFLG1 NIL)
(FACE ^IDENTY 462 ^FACTYP PLN ^CLASCD TE_FACE ^FACNOR 1
 ^FACLOP 463 ^UTFLG1 NIL)
(FACE ^IDENTY 455 ^FACTYP PLN ^CLASCD TE_FACE ^FACNOR 1
 ^FACLOP 456 ^UTFLG1 NIL)

(FACLOP ^IDENTY 498 ^LOPNUM 2 ^UTFLG1 NIL ^EDGLOP 466 459)
(FACLOP ^IDENTY 463 ^LOPNUM 1 ^UTFLG1 NIL ^EDGLOP 466)
(FACLOP ^IDENTY 456 ^LOPNUM 1 ^UTFLG1 NIL ^EDGLOP 459)

(EDGE ^IDENTY 459 ^EDGTYP ELP ^CLASCD CONVEX ^LFTFAC 455
 ^LFENUM 1 ^RHTFAC 497 ^RFENUM 1 ^UTFLG1 NIL)
(EDGE ^IDENTY 466 ^EDGTYP ELP ^CLASCD CONVEX ^LFTFAC 462
 ^LFENUM 1 ^RHTFAC 497 ^RFENUM 1 ^UTFLG1 NIL)
```

Figure 6. Frame-based data example.

is illustrated in Figure 5. We made use of this change of MC_VOL face status to drive the recognition process which will be described in section 5.1.

4. Representing the MC_VOL in the AI environment

The O-A-V triplets data format of OPS5 has led to the design of a frame-based data structure in the AI environment. By issuing the command 'BFRAME mv_x', where mv_x is assumed to be the system name of a sub-volume of MC_VOL, the BRep of mv_x is extracted and copied internally from the winged-edge data structure to the frame-based data structure. Presented in Appendix 1 is the data structure declaration section of the system program in the original OPS5 language syntax. As an example, when a straight cylindrical hole is drilled perpendicularly through two parallel planar faces of a stock model, the form of MC_VOL generated will be a cylindrical column. Its BRep represention in the AI environment is illustrated in Figure 6. To utilize the strength of OPS5 in symbolic pattern matching and to facilitate human interpretation/interrogation of the BRep information, some attribute values are converted from integer codes into explicit symbolic strings during the data copying process. For example, the edge convexity has been translated from integer code 1, -1, or 0 to CONVEX, CONCAVE or SMOOTH respectively.

5. Classification of machining features

We classified 2.5D machining features into two categories. Category 1 (MF_{cat1} refers to one-dimensional machining features that involve only the feed motion of the machine spindle axis for machining. Category 2 (MF_{cat2}) covers those that will involve the interpolated feed motion of the machine's x and y axes for contour machining.

The following criteria are assumed for MF_{cat1}:

(i) the MC_VOL can be machined by one or more one-dimensional machining operations, i.e.
$MC\_VOL_{cat1} :: = \{1DMOP_n\}$ where $n \geq 1$ and n = positive integers;

(ii) each lDMOP(i) consists of the following structure which is similar to that used by Choi et al. (1984):

lDMOP(i) ::= HSS HES EDG HBS
where HSS is a planar TE_FACE,

HES is a positive surface normal cylindrical or conical MC_FACE,

EDG is a closed circular edge with HES and HBS as adjacent faces,

and HBS is either a planar TE_FACE, planar MC_FACE or conical MC_FACE;

(iii) the joining condition between cylindrical and planar faces or between conical and planar faces must be that the axis vector of the cylindrical or conical face is parallel to the surface normal vector of the planar face;

(iv) joining condition between cylindrical and conical faces or between conical and conical faces must be co-axial.

Similarly, the following criteria are assumed for MF_{cat2}:

(i) the MC_VOL can be machined by one or more 2.5D machining operations, i.e.
$MC\_VOL_{catl} :: = \{2.5DMOP_m\}$ where $m \geq 1$ and m = positive integers,

(ii) each 2.5DMOP(i) consists of the following structure:
2.5DMOP(i) ::= PSS $\{PES_n\}$ $\{EDG_n\}$ PBS

where PSS is a planar TE_FACE,

$\{PES_n\}$ is a set of connected planar or cylindrical faces with the number of faces = n and their face classification can be TE_FACE or MC_FACE,
$\{EDG_n\}$ is a list of n linked edges and each edge has a face of $\{PES_n\}$ and PBS as adjacent faces,

and PBS is either a planar TE_FACE or MC_FACE.

(iii) the joining condition between cylindrical and planar faces in $\{PES_n\}$ must be that the axis vector of the cylindrical face is orthogonal to the surface normal vector of the planar face,

(iv) edges in $\{EDG_n\}$ have solid angle between adjacent faces either equal to 90° or 270°.

The structures of lDMOP(i) and 2.5DMOP(i) described above can be illustrated by using entity relationship (E-R) diagrams shown in Figures 7 (a) and (b) respectively.

The criteria defined above for MF_{cat1} also include the definition of a compound configuration of MF_{cat1} because the number of machining operations *n* of $\{lDM0P_n\}$ can be greater than one. Physically, this refers to multi-diameter stepped holes which can be regarded as a string of co-axial holes which have a common HSS. This idea is illustrated by Figure 8 which shows a machined part with a counter-sunk, non-through drilled hole. Since machining of the hole will involve only z-

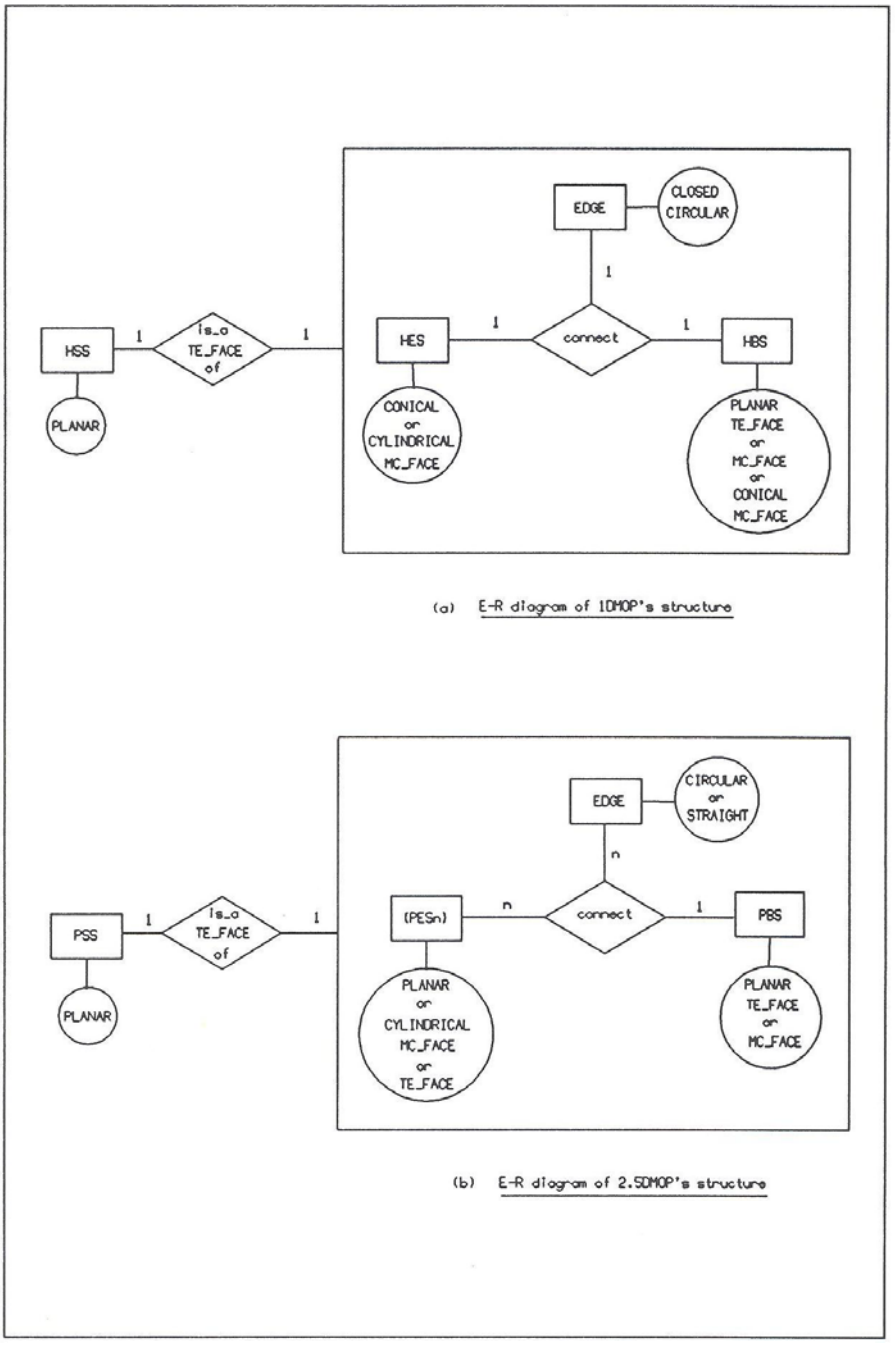
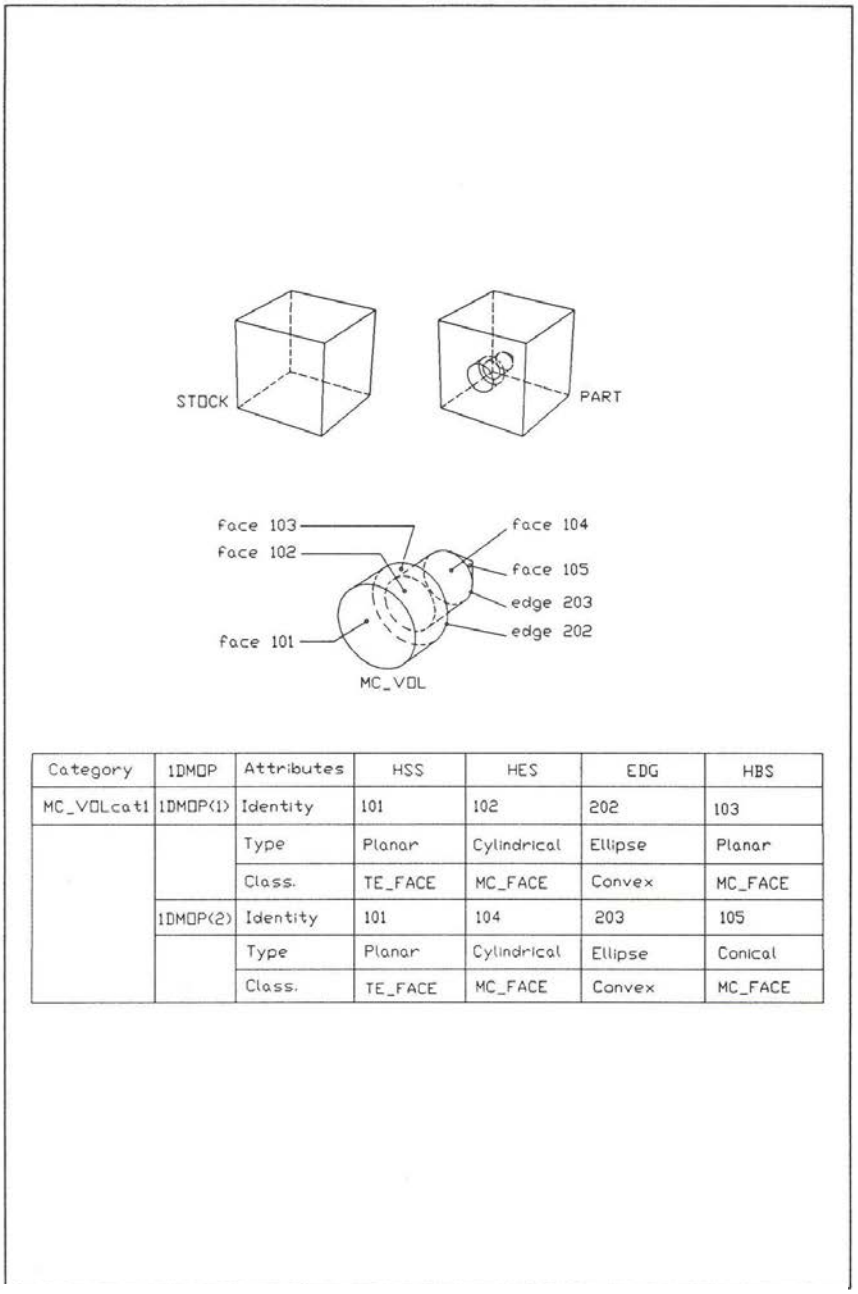
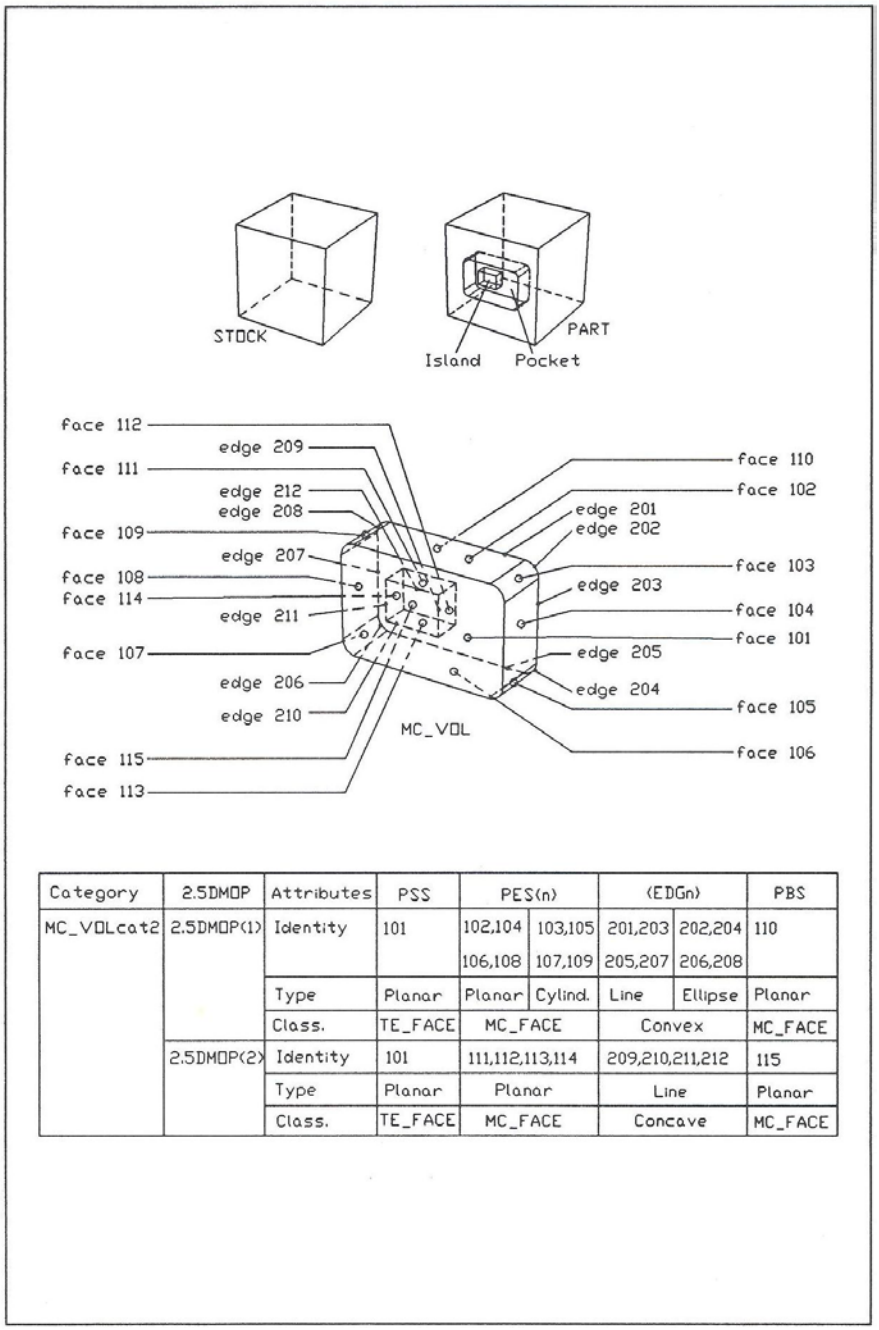


Figure 7. Structures of 1DMOP and 2.5DMOP



Category	1DMOP	Attributes	HSS	HES	EDG	HBS
MC_VOLcat1	1DMOP(1)	Identity	101	102	202	103
		Type	Planar	Cylindrical	Ellipse	Planar
		Class.	TE_FACE	MC_FACE	Convex	MC_FACE
	1DMOP(2)	Identity	101	104	203	105
		Type	Planar	Cylindrical	Ellipse	Conical
		Class.	TE_FACE	MC_FACE	Convex	MC_FACE

Figure 8. Compound MC_VOL_{cat1} matching feature



Category	2.5DMOP	Attributes	PSS	PES(n)		⟨EDGn⟩		PBS
MC_VOLcat2	2.5DMOP(1)	Identity	101	102,104	103,105	201,203	202,204	110
				106,108	107,109	205,207	206,208	
		Type	Planar	Planar	Cylind.	Line	Ellipse	Planar
		Class.	TE_FACE	MC_FACE		Convex		MC_FACE
	2.5DMOP(2)	Identity	101	111,112,113,114		209,210,211,212		115
		Type	Planar	Planar		Line		Planar
		Class.	TE_FACE	MC_FACE		Concave		MC_FACE

Figure 9. Compound MC_VOL_{cat2} machining feature

axis feed motion of the spindle, the corresponding MC_VOL belongs to MC_VOL_{cat1} which consists of two lDMOP and hence *n* is 2. The upper counter- sunk hole is lDMOP(l) which has planar face 101 as HSS, cylindrical face 102 as HES, planar face 103 as HBS and circular edge 102 as EDG. The lower drilled hole is 1DMOP(2) which has planar face 101 as HSS, cylindrical face 104 as HES, conical face 105 as HBS and circular edge 203 as EDG.

Similarly, the criteria defined above for MF_{cat1} also describe compound configurations when *m* of $\{2.5DMOP_m\}$ is greater than one. Physical examples are a nested pocket (a pocket within a pocket) and a pocket that has an island on its base. For the former example, there are two 2.5DMOP machining operations, one for the upper pocket and the other for the lower pocket. The upper and the lower pockets will each has its own $\{PES_n\}$ and PBS but they will share the same PSS. For the latter example there are also two 2.5DMOP machining operations, one for the pocket and the other for the island as illustrated in Figure 9.

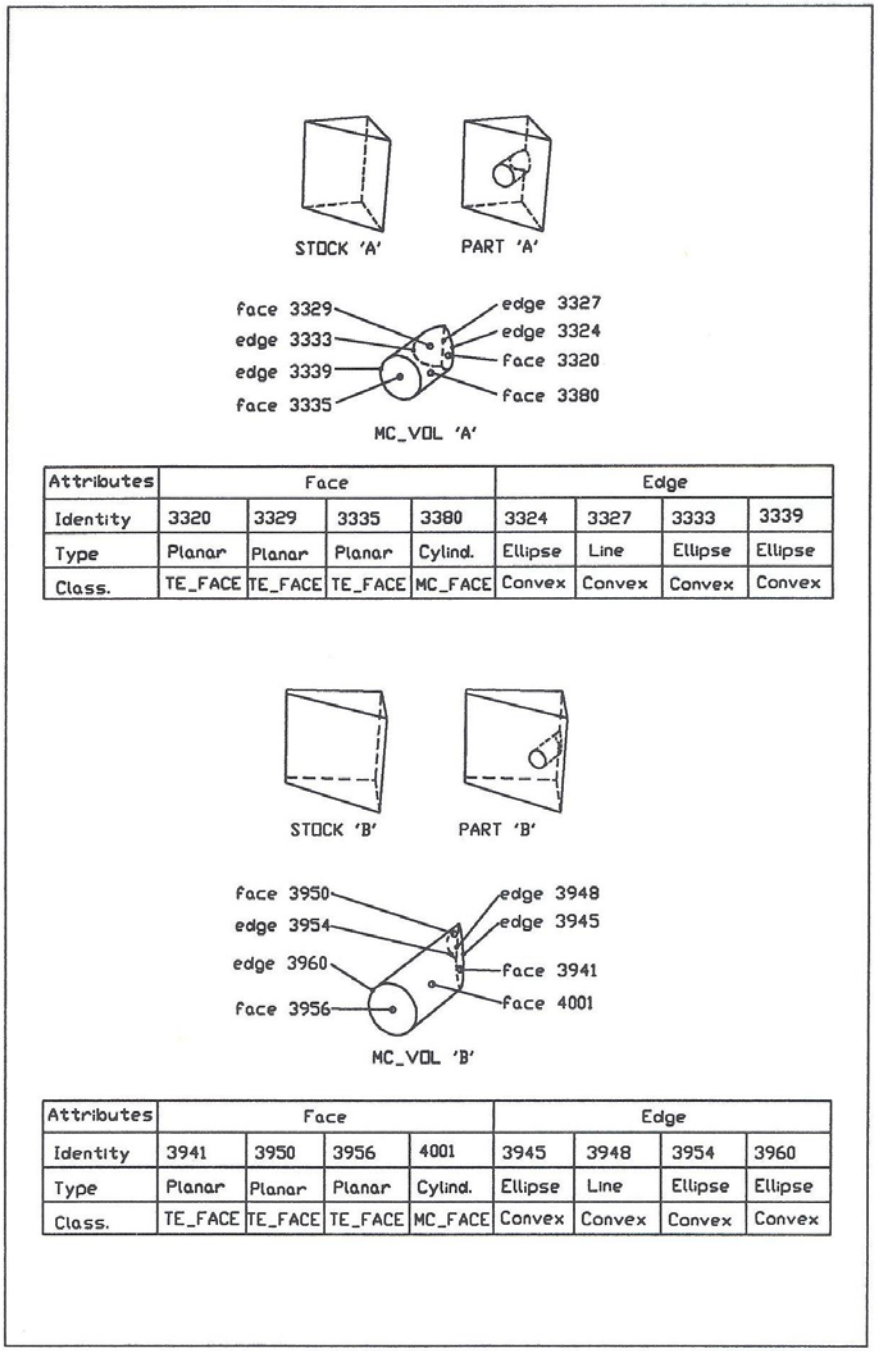
5.1 *Method and rules for recognizing machining features*

Forward chaining rules which define the dissected geometrical/topological structure of MC_VOL in accordance with the previously defined criteria for MF_{cat1} and MF_{cat2} as well as their possible compound configuration relationships are constructed. The rules are arranged in a cluster of context trees so that the rule searching is based on depth-first strategy. The current system default sequence is to hunt for MF_{cat1} first, then MF_{cat2} followed by MF_{cat1}/MF_{cat2} compound configurations.

The system currently handles the sub-volumes of MC_VOL one at a time and order is decided by the user. The frame-based BRep of a specific sub-volume of MC_VOL is established in the AI environment as described in Section 4. Based on the recognize-act cycle mechanism of OPS5, the system continually tries to match the conditions of a selected rule with the MC_VOL's BRep as well as the run-time generated data. During the inference process, the actions of the rules perform various important functions. For instance, they may change the inference results into new facts and inherit them to the subsequent rules, modify or remove some of the existing facts, or directly access the winged-edge data and the procedural routines of the solid modeller via OPS5 foreign procedural languages interface facility.

The sample part shown in Figure 12 is used to illustrate the method and the rules used to find machining operation face sets. The part has a cylindrical stepped hole below the base of the upper pocket so it represents a MF_{cat2} compounded with a MF_{cat1}. As there are quite a number of rules invoked during the recognition process, only the rules for finding the upper pocket and the hole are described. The important rules are shown in the readable OPS5 language syntax with comments in Appendix 2. The function and effect of the rules are also summarized below in the order of their firing sequence.

Rule	Function and effect
OPLAN	The user enters the command 'OPLAN' to activate this rule which makes some preparation work for later returning to the command parser when no more machining face sets can be found. It then calls for the finding of MF_{cat1} and MF_{cat2} and activates RULE_DETECT_NON_CC_DEP.
RULE_DETECT_NON_CC_DEP	This rule finds the TE_FACE that has the largest surface area from the boundary faces of MC_VOL.
RULE_UPDATE_LGFAC_TIME_FLAG	Based on the founded TE_FACE, this rule activates two rules, one finding MF_{cat2} with round corners and the other finding MF_{cat2} without round corners.
RULE_FIND_NON_CC_DEP_TYPEl	Since the pocket of the part has round corners, this rule is invoked. It stores the TE_FACE in a temporary working frame and calls for the search of the DRIV_FC of the pocket.
RULE_FIND_DRIVFC_LIST	This rule is activated many times recording all the DRIV_FC, i.e., the wall faces of the pocket, in a temporary working frame.
RULE_FIND_PLN_PARTFC_TYPEl	Finds the PART_FC, i.e., the base face of the pocket, transfers the identities of the found TE_FACE, DRIV_FC and PART_FC from the temporary working frame to a new frame. It also calls for the test of compound machining features that may exist below the base face of the pocket.
RULE_PLN_PARTFC_CYL_TEST	Tests if there is any cylindrical hole below the base face of the pocket.
RULE_PLN/CYL/PLN_l	Finds the cylindrical hole below the base face of the pocket and transfers the identities of the TE_FACE, DRIV-FC and PART-FC to a new frame.



Attributes	Face				Edge			
Identity	3320	3329	3335	3380	3324	3327	3333	3339
Type	Planar	Planar	Planar	Cylind.	Ellipse	Line	Ellipse	Ellipse
Class.	TE_FACE	TE_FACE	TE_FACE	MC_FACE	Convex	Convex	Convex	Convex

Attributes	Face				Edge			
Identity	3941	3950	3956	4001	3945	3948	3954	3960
Type	Planar	Planar	Planar	Cylind.	Ellipse	Line	Ellipse	Ellipse
Class.	TE_FACE	TE_FACE	TE_FACE	MC_FACE	Convex	Convex	Convex	Convex

Figure 10. Part ‘A’ and Part ‘B’

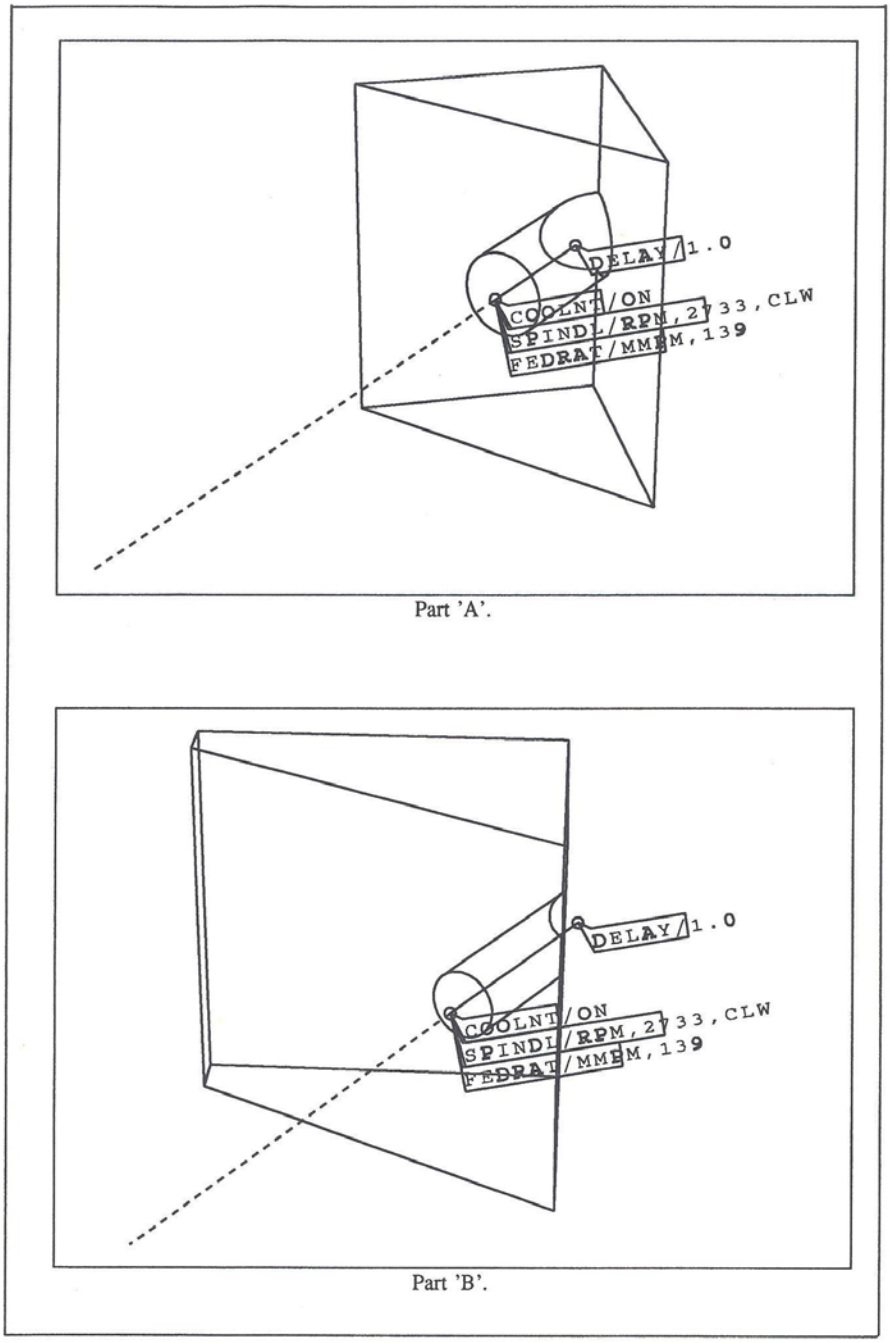


Figure 11. Cutter oaths of part ‘A’ and part ‘B’

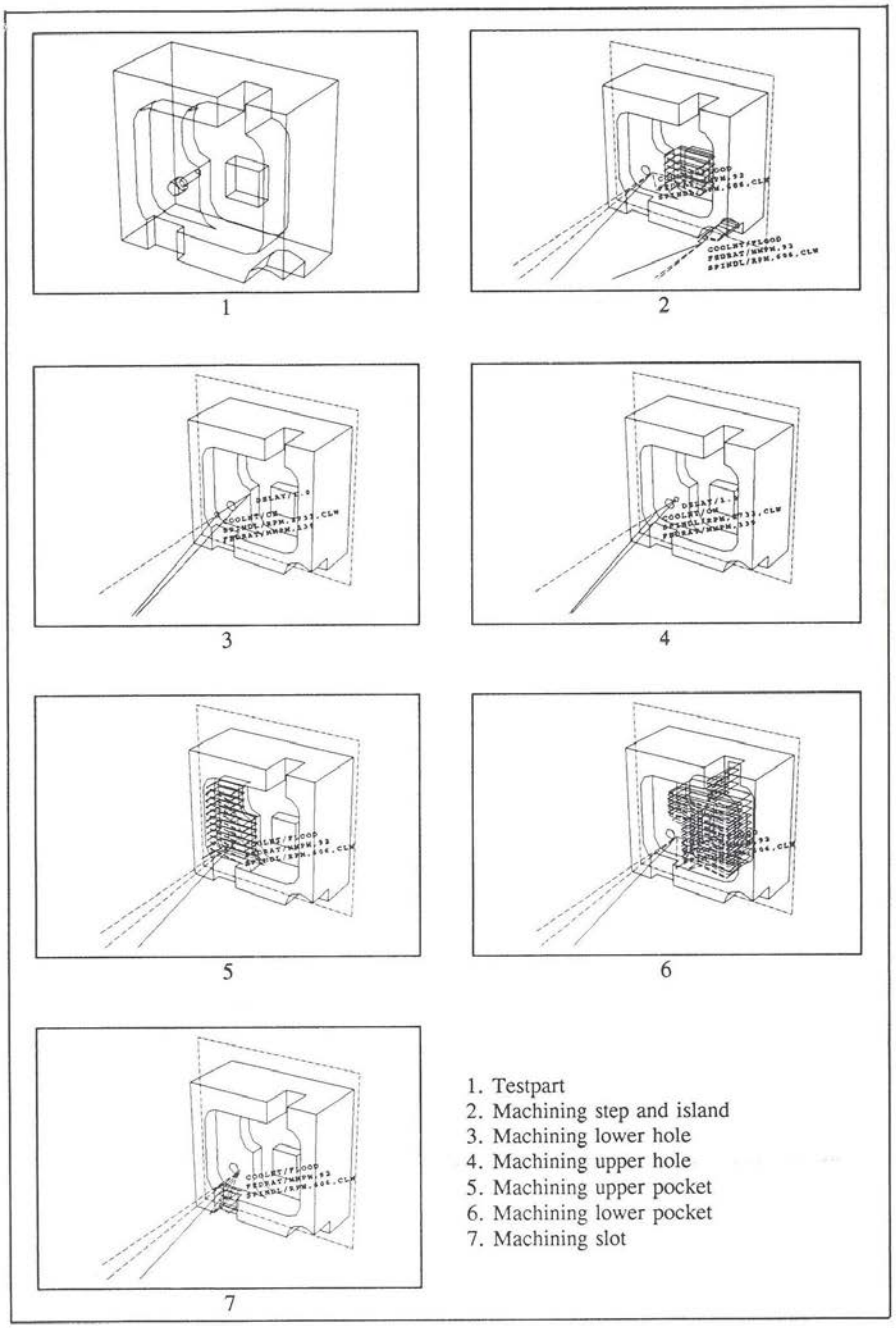


Figure 12. An illustrated test part

When no more machining features are found, the system will return to the command parser to wait for the user's command. By issuing another command, 'WOPLAN', the machining operation face sets stored in the above two frames will be written to a machining operation file.

6. Tool path generation

The machining operation file stores the TE_FACE, DRIV_FC and PART_FC information of each MF_{cat1} or MF_{cat2} in an independent record. We used a modified NC tool path generation module (Wong 1989) to read the machining operation file and process the machining operations sequentially. For MF_{cat1} the tool path is a straight line from the HSS to the HBS. For MF_{cat2}, the $\{PES0_n\}$ faces are offset towards the inner side of MC_VOL by an amount equal to the radius of the cutter used plus a default clearance allowed for subsequent fine machining operation. The PBS face is offset towards the inner side by an amount conceived to be the distance between the bottom face of a flat-end milling cutter and the PBS. This offset distance is a linear function of the axial depth of cut and is determined automatically based on the material type. The offset surfaces will intersect to form a group of virtual two-dimensional curve segments which are further processed to produce a list of zig-zag cutter paths for rough machining. The zig-zag path direction is always along the x-axis of the modelling space coordinate system. Similarly, a list of cutter paths around the $\{PES_n\}$ is formed for fine machining. Whenever cutter retraction is required, it will be raised to a safe height which is determined by evaluating the bounding envelope of the stock. A simple cutter file and a machinability data file are maintained in the system so that cutting speed, feed and axial depth of cut are generated automatically. The speed, feed, delay time and coolant postprocessor-based instructions are also inserted in the cutter path list at appropriate positions. They can be displayed on the screen together with the cutter path to facilitate visual verification. The cutter path can also be edited interactively. The cutter path data will be further processed to produce NC program.

7. The learning agent

We use the two parts, part A and part B, illustrated in Figure 10 to describe the learning agent. As the cylindrical through hole of part A passes right through edge 3327, the boundary structure of the MC_VOL is different from the defined structures of MF_{cat1} or MF_{cat2} and therefore it cannot be recognized by the feature recognizer of the system. To activate the learning agent, the user can issue the command 'LEARN mv_x ', where mv_x is the system given name of the MC_VOL. When activated, the learning agent will ask the user to indicate which faces are to be used as the TE_FACE, PART_FC and DRIV_FC for machining the MC_VOL. The input process is assisted by interactive graphic facility, by which the user manipulates the mouse cursor to pick the correct faces. As for the MC_VOL of part A, face 3335 will be used as TE_FACE since it is a planar surface orthogonal to the entrance direction of the cutting tool. Face 3329 (or 3320) will be regarded as the PART_FC since it will be a tool exit face for machining the through hole and face 3380 will be conceived as the DRIV_FC for drilling the through hole.

Having acquired the three machining faces, the learning agent asks the user for a rule name. The agent then extracts the MC_**VOL** BRep from the frame-based database and uses the information as condition clauses to build a new OPS5 rule. The actions of the new rule are to create a new frame for storing the three acquired machining faces and to write them to a machining operation file. The new rule generated after learning part A is shown in Appendix 3 with explanation notes. Building of the rule is by using the run-time compiling facility of VAX-OPS5. The execution codes of the new rule are added to the rule base of the system in the same working session. The context of the new rule is also stored in a file, called OPS$BUILD.OPS.

The new rule can be considered as an independent process planning rule not only for part A but also for other machined parts that have MC_**VOL** BRep pattern similar to that of part A. For instance, the part B, shown in Figure 10, has similar BRep pattern and hence it can also be handled by the new rule. Activation of the new rule is by issuing the command 'RECALL name', where name is the user's given rule name during the learning process. The new rule uses the memorized BRep pattern to match with the BRep pattern of part B and the actions of the new rule generates the machining operation file. Using the NC module to process the machining operation file, the corresponding cutter path can be produced. The generated cutter paths for the two parts are illustrated in Figure 11.

8. Conclusions

We integrated the VAX-OPS5 AI environment with the PADL-2 solid modeller for implementing two process planning methods for 2.5D mechanical parts. The resultant system was developed using both the rule-based OPS5

language and FORTRAN procedural languages. The tight connection enables direct internal information transfer between the two environments.

The first feature-recognition-based method can be used to recognize machining features whose shapes are in conformance with the system defined machining feature templates. It can handle machined parts with reasonable complex shapes such as the one illustrated in Figure 12.

The second method is an attempt to emulate the mental activities of a human process planner when tackling an 'odd shaped' machining feature without prior process planning knowledge of that particular machining feature. The method requires the user's explicit instruction concerning the machining method of the odd shaped machining feature in the first encounter. The system then memorizes by hard its shape and the associated machining method as an independent rule for process planning of machining features that have shape similar to the memorized one. The method is based on the assumption that a subset of complex machining features have generic shape patterns but the generic shape patterns are factory dependent and the range of the patterns is so wide that it is difficult, if not impossible, to pre-define and implement them either in a feature-recognition-based system or in a design by feature-based system. We consider that an intelligent process planning system that can be taught to acquire new process planning knowledge to adapt to different factory working environments during its service life is a more desirable solution.

Acknowledgements

The authors wish to thank the Committee on Research and Conference Grants of the University of Hong Kong for financial support. Thanks are also due to the Department of Mechanical Engineering and the Computer Centre of the University of Hong Kong for providing the interactive computing facilities.

Appendix 1. Declaration of frame-based data structure.

```
(LITERALIZE SOLID        ; a solid frame with the following attributes or (slots) :
                IDENTY   ; identity pointer
                SYSNAM   ; user/system given ASCII name
                RGMOTN   ; rigid motion pointer
                CLASCD   ; reserved
                FACLST   ; pointer to face list frame
                UTFLG1)  ; utility flag for internal use
;
(VECTOR-ATTRIBUTE FACESS) ; a list for storing face pointers
(LITERALIZE FACLST       ; aface list frame
                IDENTY
                FACNUM   ; number of faces
                UTFLGl
                FACESS)  ; pointers to face frames
;
(LITERALIZE FACE          ; a face frame
                IDENTY
                FACTYP   ; face swface type
                CLASCD   ; face classification
                FACNOR   ; face swface normal code
                FACLOP   ; a pointer to face loop frame
                UTFLGl)
;
(VECTOR-ATTRIBUTE EDGLOP) ; a list for storing face edge loop pointers
(LITERALIZE FACLOP       ; a face loop frame
                IDENTY
                LOPNUM   ; number of face edge loops
                UTFLGl
                EDGLOP)  ; pointers to edge(i) frame of loop(i)
;
```

```
(LITERALIZE EDGE          ; an edge frame
                IDENTY
                EDGTYP    ; edge curve type
                CLASCD    ; convexity classification
                LFTFAC    ; left adjacent face
                LFENUM    ; number of edges of left adjacent face
                RHTFAC    ; right adjacent face
                RFENUM    ; number of edges of right adjacent face
                UTFLGl)
```

Appendix 2. Recognition rules for the part shown in Figure 12.

```
;assuming that the MC_VOL shown in Figure 15 has been established in the data base and the command, 'OPLAN' is issued in the A1
environment, then the following rules will be activated (P OPLAN
   {<oplan> (OPLAN)}; match the 'OPLAN' command
   (SOLID "FACLST <faclst>); make sure that there is a MC_VOL present
   ··>
   (REMOVE <oplan>) ; remove OPLAN command to avoid re-firing of this rule
   (MAKE ALEX) ; go back to AI command parser when no more rules can be invoked
(MAKE DETECT_NON_CC_DEP); want to hunt for MF_cat2
   (MAKE DETECT_CC_DEP) ; want to hunt for MF_cat1
;
   (P RULE_DETECT_NON_CC_DEP ; find the largest TE_FACE
   (DETECT_NON_CC_DEP); receive the message token from the above rule
   (SOLID ^FACLST <faclst>) ; get the face list of MC_VOL
   ··>
   (MAKE FIND_PLN_PARTFC_TYPE3),·
   (MAKE FIND_PLN_PARTFC_TYPE2) ; want to find different PART_FC
   conditions
   (MAKE FIND_PLN_PARTFC_TYPEI) ;
   (MAKE TOOL_ENT_FACE (XLGFAC <faclst>))) ;find the largest TE_FACE by the external procedure XLGFAC
;
   (P RULE_UPDATE_LGFAC_TIME_FLAG ;increase the priority of the largest TE_FACE
   {<token> (TOOL_ENT_FACE <fl>)}; receive TE_FACE f1
   {<face> (FACE "IDENTY <fl>)} ; inherit the largest TE_FACE found
   ··>
   (REMOVE <token>); remove the message token
   (MODIFY <face>); update TE_FACE's time flat
   (MAKE FIND_NON_CC_DEP_TYPE2) ; find MF_cat2 without round corners
   (MAKE FIND_NON_CC_DEP_TYPEl)) ;find MF_cat2 with round corners
;
   (P RULE_FIND_NON_CC_DEP_TYPEl; MF_cat2 with round corners
   (FIND_NON_CC_DEP_TYPEl); receive the message token
   {<face> (FACE ^IDENTY <fl> ^FACTYP PLN ^CLASCD TE_FACE
   ^UTFLGl NIL)}; TE_FACE is planar
   (EDGE ^EDGTYP ELP ^CLASCD CONVEX ^LFTFAC <f1>
   ^LFENUM > 1 ^RHTFAC <f2> ^UTFLGl NIL); edge is convex, left face is f1 and right face is f2, f1 has more than 1 edges
   (FACE ^IDENTY <f2> ^FACTYP CYL) ; f2 is a cylindrical face
```

```
   ‥>
   (MODIFY <face> ^UTFLGl NON_CC_DEP); change the utility flag status
   (MAKE MC/OPER ^NUMBER TEMPOR ^TOENFC <fl> ^DFCNUM 0);create and initialize a temporary frame for storing the machining operation face sets
   (MAKE NON_CC_DEP_FOUND_TOKEN)); issue a message token
   ; this rule will be re-invoked many times to find all the DRIV FC
(P RULE_FIND_DRIVFC_LIST
   (NON_CC_DEP_FOUND_TOKEN)
   (FACE ^IDENTY <f1> ^UTFLGl <>NIL); f1's utility flag is not nil
   {<edge> (EDGE ^CLASCD CONVEX ^LFI'FAC <f1> ^RHTFAC <12> ^UTFLGl NIL)}
   {<face2> (FACE ^IDENTY <f1> ^UTFLGl NIL)}
   {<mdoper> (MC/OPER ^TOENFC <f1> ^DFCNUM <number>)};get the pointer of the temporary machining operation frame
   ‥>
   (MAKE TEFCLS ^IDENTY <fl> ^TEFACE <fl>); make a frame for recording f1 as the TE_FACE of f2
   (MODIFY <edge> ^UTFLGl FIND_DRIVFC_LIST) (MODIFY <face2> ^UTFLGl FIND_DRIVFC_LIST) (MODIFY <mdoper> ^DFCNUM (COMPUTE 1 + <number>)
   "DRIVFC <f1> (SUBSTR <mdoper> DRIVFC INF))) ; update the found DRIV _FC in the temporary machining operation frame by the SUBSTR function and update the number of DRIV _FC found by the COMPUTE function
;
   (P RULE_FIND_PLN_PARTFC_TYPEl
   (FIND_PLN_PARTFC_TYPEI)
   {<face1> (FACE ^IDENTY <f1> ^FACTYP PLN ^CLASCD MC_FACE ^UTFLGl NIL)}
   (EDGE ^CLASCD CONVEX ^LFTFAC <f1> ^LFENUM > 1 ^RHTFAC <f1>)
   (FACE ^IDENTY <f1> ^UTFLGl << FIND_DRIVFC_LIST
   FIND_NEST_DEP_DRIVFC_LIST >>) ; flag status is either FIND_DRIVFC_UST or
   FIND-NEST-DEP-DRIVFC-liST
   {<mdoper> (MC/OPER ^NUMBER TEMPOR)}
   ‥>
   (MODIFY <face1> ^CLASCD PART_FC ^UTFLGl FIND_PLN_PARTFC)
   ; change f1' s face classification attribute value from MC_FACE to PART_FC and its utility flag status to FIND_PLN_PARTFC
   (MAKE TEFCLS ^IDENTY <f1> ^TEFACE (SUBSTR <mdoper> TOENFC
   TOENFC)) ; make a frame for recording the TE_FACE of f1, which is inherited from the TE_FACE of the temporary machining operation record
   (MAKE MC/OPER ^NUMBER (GENATOM) ^TOENFC (SUBSTR <mdoper>
   TOENFC TOENFC) ^PARTFC <f1> ^DFCNUM (SUBSTR <mdoper>
   DFCNUM DFCNUM) ^DRIVFC (SUBSTR <mdoper> DRIVFC INF)); make a permanent machining operation record and inherit the information from the temporary machining operation record. The record number is generated by the run-time system using the GENATOM function.
   (REMOVE <mdoper>) ; remove the temporary machining operation record
   (MAKE FIND_PLN_PARTFC_NON_CC_DEP); try to hunt for nested pocket or island on f1 which is a PART _FC
   (MAKE RULE_PLN_PARTFC_CON_TEST); try to hunt for conical hole on f1
   (MAKE RULE_PLN_PARTFC_CYL_TEST)); try to hunt for cylindrical hole on f1
   ;
   (P RULE_PLN_PARTFC_CYL_TEST; preliminary test for PART_FC
   (RULE_PLN_PARTFC_CYL_TEST)
   (FACE ^IDENTY <f1> ^FACTYP PLN ^CLASCD PART_FC ^FACLOP <faclop1> AUTFLG1 <>NIL)
   (FACLOP ^IDENTY <faclop1> ^LOPNUM > 1) ; f1 has an inner edge loop
   (EDGE ^ IDENTY <e1> ^EDGTYP ELP ^CLASCD CONCAV ^LFTFAC <f1> ^LFENUM 1 ^RHTFAC <f2> ^UTFLG1 NIL)
   (FACE ^IDENTY <f2> ^FACTYP CYL ^CLASCD MC FACE ^UTFLG1 NIL)
```

```
   -->
   (MAKE PLN_PARTFC_CYL_TEST_TOKEN)); try to hunt for a cylindrical hole below this PART_FC
;
   (P RULE_PLN/CYLIPLN_1; a cylindrical hole below a PART_FC
   {<token> (PLN_PARTFC_CYL_TEST_TOKEN)}
   (FACE ^IDENTY <f1> ^FACTYP PLN ^CLASCD PART_FC ^ UTFLG1 <> NIL)
   {<tefcls> (TEFCLS ^IDENTY <f1>)}; get the TE_FACE of f1
   {<edge1> (EDGE ^EDGTYP ELP ^CLASCD CONCAV ^LFTFAC <f1> ^LFENUM 1 ^RHTFAC <f2> ^UTFLG1 NIL)}
   {<face2> (FACE ^IDENTY <f2> ^FACTYP CYL ^CLASCD MC_FACE
   ^FACNOR 1 ^UTFLG1 NIL)}
   {<edge2> (EDGE ^EDGTYP ELP ^CLASCD CONVEX ^LFTFAC <f3> ^LFENUM 1^RHTFAC <f2> ^UTFLG1 NIL)}
   {<face3> (FACE ^IDENTY <f3> ^FACTYP PLN^CLASCD MC_FACE ^ UTFLG1 NIL)}
   -->
   (REMOVE <token>)
   (MODIFY <edge1> ^UTFLG1 PLN/CYLIPLN_1)
   (MODIFY <face2> ^CLASCD DRIV_FC ^UTFLG1 PLN/CYLIPLN_1) (MODIFY <edge2> ^UTFLG1 PLN/CYLIPLN_1)
   (MODIFY <face3> ^CLASCD PART_FC ^UTFLG1 PLN/CYLIPLN_1) (MAKE TEFCLS ^IDENTY <f2> ^TEFACE (SUBSTR
<tefcls> TEFACE
   INF));make a frame for recording the TE_FACE of f3 which is inherited from that of f1
   (MAKE TEFCLS ^IDENTY <f3> ^TEFACE (SUBSTR <tefcls> TEFACE
   INF));make a frame for recording the TE_FACE of f3 which is inherited from that of f1
   (MAKE MC/OPER ^NUMBER (GENATOM) ^TOENFC (SUBSTR <tefcls>
   TEFACE INF) ^PARTFC <f3> ^DFCNUM 1 ^DRIVFC <f2>) ; make a new machining operation record frame for storing the
faces involved in the hole drilling operation (MAKE RULE_PLN_PARTFC_CON_TEST); try to hunt for conical hole (MAKE
RULE_PLN_PARTFC_CYL_TEST)); try to hunt for cylindrical hole
```

Appendix 3. New rule built for part 'A'

The following new rule is built by applying the learning agent on part 'A' which is shown in Figure 10.

```
(P PART_A
   { <RECALL> ( RECALL PART_A) }
   ( EDGE ^ IDENTY <3339> ^ EDGTYP ELP ^ CLASCD CONVEX
            ^ LFTFAC <3335> ^ RHTFAC <3380>)
   ( EDGE ^ IDENTY <3333> ^ EDGTYP ELP ^ CLASCD CONVEX
            ^ LFTFAC <3380> ^ RHTFAC <3329> )
   ( EDGE ^ IDENTY <3327> ^ EDGTYP LIN ^ CLASCD CONVEX
            ^ LFTFAC <3329> ^ RHTFAC <3320> )
   ( EDGE ^ IDENTY <3324> ^ EDGTYP ELP ^ CLASCD CONVEX
            ^ LFTFAC <3380> ^ RHTFAC <3320>)
   (FACE ^ IDENTY <3329> ^ FACTYP PLN ^ CLASCD TE_FACE
            ^ FACNOR 1)
   ( FACE ^ IDENTY <3320> ^ FACTYP PLN ^ CLASCD TE_FACE
            ^ FACNOR 1)
   (FACE ^ IDENTY <3335> ^ FACTYP PLN ^ CLASCD TE_FACE
            ^ FACNOR -1)
   (FACE ^ IDENTY <3380> ^ FACTYP CYL ^ CLASCD MC_FACE
```

```
        ^ FACNOR 1)
-->
( REMOVE <RECALL> )
( MAKE MC/OPER ^ NUMBER ( GENATOM )  ^ TOENFC <3335>
                                      ^ PARTFC <3329>
                                     ^ DRIVFC <3380>
                                     ^ DFCNUM 1)
                                     ( MAKE WOPLAN ))
```

Remarks:

1. 'PART_A' is an arbitrary name of the rule provided by user during the learning process.
2. 'RECALL PART_A' is a variable frame added by the system during the rule building process. It is used for the purpose of later on matching a user's command so that the new rule can be recalled.
3. The integer values enclosed in < > brackets are originally the system assigned integer identities of the corresponding geometric entities. A special routine of the learning agent encapsulates them in < > brackets thereby converting them into variables of OPS5 format.
4. Using the user instructed machining faces, the 'MAKE MC/OPER ... ' action generates a machining operation frame.
5. The 'MAKE WOPLAN' action retrieves the machining faces from the machining operation frame and writes them in a file for the NC module to read.

References

CHAN, K. C., and TAN, S. T., 1988, Hierarchical structure to winged-edge structure: a conversion algorithm, The Visual Computer, 4, 133-141.

CHOI, B. K., 1984, Automatic recognition of machined surfaces from a 3D solid model, Computer Aided Design, Vol. 16, No.2, March, 81-86.

COHEN, P. R. and FEIGENBAUM, E. A., (Eds), 1982, The Handbook of Artificial Intelligence, Volume 3, Chapter XIV : Learning and Inductive Inference, (Pitman Books).

CUTKOSKY, M., 1988, Features in process based design, ASME Computers in Engineering Conference, San Francisco.

DESCOTTE, Y., and LATOMBE, J., 1984, GARI: an expert system for process planning, In PICKETT, M. S., and BOYSE, J. W., Eds., Solid modelling by computers : From theory to applications, (Plenum Press), 329-346.

FORGY, C., and MCDERMOTT, J., 1977, OPS, a domain-independent production system language, Proceedings of the Fifth International Joint Conference on Artificial Intelligence, 933-939.

GRAYER, A. R., 1977, The automatic production of machined components starting from a stored geometric description, In MCPHERSON, D., Ed., Advances in Computer-Aided Manufacture, (North-Holland), 137-152.

HENDERSON, M. R., 1984, Extraction of feature information from three dimensional CAD data, Ph.D. thesis, Purdue University.

JOSHI, S. and CHANG, T. C., 1988, Graph-based heuristics for recognition of machined features from a 3D solid model, Computer Aided Design, 20, (2), 58-66.

MCDERMOTT, J. and FORGY, C., 1978, Production system conflict resolution strategies, In Waterman, D.A., and Hayes-Roth, F., Eds., Pattern-directed inference systems, (Academic Press), 177-199.

REQUICHA, A. A. G., and TILOVE, R. B., 1978, Mathematical foundations of constructive solid geometry: general topology of closed regular sets, Tech. Memo. No. 27a, Production Automation Project, University of Rochester.

REQUICHA, A. A. G., and VANDENBRANDE, J. H., 1989, Form features for mechanical design and manufacturing, ASME Computers in Engineering, 1, 47-52.

SHAH, J.J., 1988, Current status of features technology (revised report), Computer Aided Manufacturing - International, Inc., November.

SILVA, C.E., 1981, Alternative definitions of faces in boundary representations of solid objects. Tech. Memo. No. 36, Production Automation Project, University of Rochester.

TURNER, G. P., and ANDERSON, D. C., 1988, An object-oriented approach to interactive, feature-based design for quick turnaround manufacturing, Computers in Engineering, 1, 551-555.

VOLECKER, H. B., and ROSENBERG, A. V., 1986, Dissemination of PADL-2 software, ADM-03, (Cornell Programmable Automation, Cornell University).

WEILER, K.,1985, Edge-based data structures for solid modelling in curved-surface environments, IEEE Computer Graphics & Applications, January, 21-40.

WONG, W.Y., 1989, An octree and face oriented approach for NC machining, M.Phil. thesis, Department of Mechanical Engineering, University of Hong Kong.