

BLDSC no:- DX205122

LOUGHBOROUGH  
UNIVERSITY OF TECHNOLOGY  
LIBRARY

AUTHOR/FILING TITLE

CHAN, A.K.W.

ACCESSION/COPY NO.

040081994

VOL. NO.

CLASS MARK

- 1 JUL 1994

- 1 JUL 1994

07 OCT 1994

LOAN COPY

24 MAY 1995

21 JUN 1995

~~30 JUN 1995~~

~~30 JUN 1995~~

21 MAR 1997

14 JAN 2000

0400819945



**A Knowledge-based Approach for the  
Extraction of Machining Features from Solid Models**

by

**CHAN, Kit-Wah Alex**

A Doctoral Thesis

Submitted in partial fulfilment of the requirements

for the award of

Doctor of Philosophy of the

Loughborough University of Technology

May 1993

© by CHAN, Kit-Wah Alex, 1993.

Loughborough University of Technology Library	
Date	Jan 94
Class	
Acc. No.	040081994

w9921434

# ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to Dr. Keith Case, my research principal supervisor, for his expert guidance, considerate support and continual encouragement throughout my doctoral studies. His enlightening suggestions and earnest comments were invaluable to the development of this thesis.

I am also greatly indebted to Dr. S.T. Tan, my research co-supervisor, for his stimulating advice, thoughtful assistance and constant encouragement.

I sincerely acknowledge Dr. N.N.Z. Gindy, Prof. J.L. Murray and Prof. D.J. Williams for serving on my doctoral committee and for their valuable comments on my research.

My thanks are extended to the staff members and students of the CAD/CAM group of the department for their kindness and readiness to help at all times. In particular, I am grateful to Dr. K.C. Chan, Dr. K.C. Hui, Dr. K.Y. Hung, Mr. W.S. Sze, Mr. K.W. Wong, Mr. W.Y. Wong, Mr. W.K. Yeung, Dr. K.M. Yu and Dr. M.F. Yuen for sharing their experience and insight with me.

The Department of Mechanical Engineering of the University of Hong Kong has provided a friendly and supportive environment which helped to foster much of this research. The availability of the PADL-2 solid modelling system from Prof. H.B. Voelcker of Cornell University, is thankfully acknowledged.

Finally, I express my heartfelt appreciation to my wife, Tania, and my family, without whose understanding and encouragement I would never have attained this level of educational development. It is to them that I dedicate this work.

# SYNOPSIS

Computer understanding of machining features such as holes and pockets is essential for bridging the communication gap between Computer Aided Design and Computer Aided Manufacture. This thesis describes a prototype machining feature extraction system that is implemented by integrating the VAX-OPS5 rule-based artificial intelligence environment with the PADL-2 solid modeller. Specification of original stock and finished part geometry within the solid modeller is followed by determination of the nominal surface boundary of the corresponding cavity volume model by means of Boolean subtraction and boundary evaluation. The boundary model of the cavity volume is managed by using winged-edge and frame-based data structures. Machining features are extracted using two methods : (1) automatic feature recognition, and (2) machine learning of features for subsequent recognition.

In the first method, a machining feature recognition procedure which employs rule-based and procedural programming techniques has been devised. The feature recognizer uses built-in heuristics and tool accessibility analysis to identify and extract 2.5D machining features from a cavity volume. The tool accessibility analysis is based on a ray-casting technique, and the results are propagated into a frame-based data structure which acts as an agenda for guiding feature searching. A recognized machining feature is represented in terms of its tool entrance face and part face identities that are used in its winged-edge boundary model.

In the second method, a machine learning approach allows the user to interact with the wireframe display to define tool entrance and part faces of the cavity volume. These taught faces together with the boundary description of the cavity volume are converted into production rules. These new rules are incorporated into the knowledge base allowing subsequent recognition of similarly shaped cavity volumes and hence the generation of appropriate machining faces. This method is intended for customization to handle factory dependent machining features or machining features that cannot be machined by simple cylindrical cutters such as end-mills.

The validity and practical usefulness of the approach is demonstrated by the inclusion of a numerical control (NC) cutter path generating module that utilizes the winged-edge data structure for the post-processing of the extracted machining features into NC part programs.

# CONTENTS

	<i>Page</i>
<b>ACKNOWLEDGEMENTS</b> . . . . .	i
<b>SYNOPSIS</b> . . . . .	ii
<b>CHAPTER 1 INTRODUCTION</b> . . . . .	1
1.1 Computers in Manufacturing . . . . .	1
1.2 Moving towards Computer-Integrated Manufacturing . . . . .	3
1.3 Features : a Methodology for Integrating CAD and CAM . . . . .	4
1.4 Principles of Existing Feature Modelling Approaches . . . . .	7
1.5 Motivation of Research . . . . .	8
1.6 Research Objectives . . . . .	9
1.7 Research Methodologies . . . . .	10
<b>CHAPTER 2 LITERATURE REVIEW</b> . . . . .	12
2.1 Geometric Modelling Methods . . . . .	12
2.1.1 Constructive Solid Geometry (CSG) . . . . .	14
2.1.2 Boundary Representation (B-rep) . . . . .	17
2.2 Feature Modelling Methods . . . . .	20
2.2.1 Human-Assisted Feature Definition . . . . .	20
2.2.2 Design by Features . . . . .	21
2.2.3 Automatic Feature Recognition . . . . .	24
2.2.3.1 Recognition with CSG Models . . . . .	24
2.2.3.2 Recognition with B-rep Models . . . . .	27
2.3 Concluding Remarks . . . . .	36
<b>CHAPTER 3 ARTIFICIAL INTELLIGENCE TECHNIQUES</b> . . . . .	39
3.1 Knowledge-Based Systems (KBS) . . . . .	39
3.2 Problem-Solving Techniques . . . . .	41
3.2.1 Knowledge Representation . . . . .	42
3.2.1.1 Production Rules . . . . .	42
3.2.1.2 Semantic Nets . . . . .	44
3.2.1.3 Frames . . . . .	45
3.2.1.4 Object Orientation . . . . .	46

	<i>Page</i>
3.2.2 Search Techniques . . . . .	47
3.2.2.1 State-Space Approach . . . . .	47
3.2.2.2 Hill Climbing Approach . . . . .	48
3.2.2.3 Constraint Satisfaction Approach . . . . .	48
3.3 Machine Learning . . . . .	49
3.3.1 Learning by Rote . . . . .	49
3.3.2 Learning from Instruction . . . . .	50
3.3.3 Learning from Examples . . . . .	50
3.3.4 Learning by Analogy . . . . .	52
3.4 AI-based Manufacturing Researches . . . . .	52
3.5 Concluding Remarks . . . . .	55

**CHAPTER 4      MACHINING FEATURE  
                    DEFINITION AND REPRESENTATION . . . . . 56**

4.1 Parts Domain . . . . .	56
4.2 The Cavity Volume Model . . . . .	57
4.3 The Boundary of the Cavity Volume . . . . .	58
4.4 Machining Feature Representation . . . . .	62
4.5 Concluding Remarks . . . . .	70

**CHAPTER 5      MACHINING FEATURE  
                    RECOGNITION ALGORITHM . . . . . 71**

5.1 Criteria for Recognizable Machining Features . . . . .	71
5.2 Overview of the Algorithm . . . . .	73
5.3 Recognition of Machining Features from the Subvolume_1 . . . . .	79
5.3.1 Machining Heuristics . . . . .	79
5.3.2 Selection of the Group(1) Faces . . . . .	82
5.3.3 Geometric Reasoning for the Group(1) Faces . . . . .	82
5.3.3.1 The First Geometric Test for the Group(1) Faces . . . . .	82
5.3.3.2 The Second Geometric Test for the Group(1) Faces . . . . .	84
5.3.3.3 The Third Geometric Test for the Group(1) Faces . . . . .	89
5.3.4 Utilization of the Group(1) Face Testing Results . . . . .	92
5.3.5 Selection of the Group(2) Faces . . . . .	94
5.3.6 Geometric Reasoning for the Group(2) Faces . . . . .	95
5.3.6.1 The First and Second Geometric Tests for the Group(2) Faces . . . . .	95
5.3.6.2 Utilization of the Group(2) Face Testing Results . . . . .	97
5.3.6.3 Analysis of the Remaining Group(2) Faces . . . . .	98
5.3.7 Selection of the Group(3) Faces . . . . .	101
5.3.8 Geometric Reasoning for the Group(3) Faces . . . . .	102
5.3.9 Utilization of the Group(3) Face Testing Results . . . . .	104
5.4 Recognition of Machining Features from the Subvolume_2 . . . . .	105
5.5 Concluding Remarks . . . . .	111

	<i>Page</i>
<b>CHAPTER 6 MACHINE LEARNING OF FEATURES FOR RECOGNITION</b> . . . . .	113
6.1 The Role of the Machine Learning Approach . . . . .	113
6.2 The Methodology of the Approach . . . . .	114
6.2.1 Boundary Characteristics . . . . .	116
6.2.2 Teaching a Feature Description . . . . .	118
6.2.3 Memorizing the Taught Feature . . . . .	121
6.2.4 Recollecting the Learnt Feature . . . . .	122
6.3 Concluding Remarks . . . . .	123
<b>CHAPTER 7 IMPLEMENTATION</b> . . . . .	124
7.1 The Solid Modelling System . . . . .	124
7.2 The VAX-OPS5 AI Environment . . . . .	125
7.2.1 The Recognize-act Cycle . . . . .	125
7.2.2 The Command Interpreter . . . . .	127
7.2.3 The Run-time Compiler . . . . .	127
7.3 Linking PADL-2 and VAX-OPS5 . . . . .	128
7.4 Implementation of the Feature Recognition Approach . . . . .	129
7.4.1 Establishing Boundary Information . . . . .	130
7.4.1.2 Boundary Representation of the Stock and Part Models . . . . .	132
7.4.1.3 Boundary Representation of the Cavity Volume Model . . . . .	134
7.4.2 Describing Cavity Subvolumes in VAX-OPS5 . . . . .	140
7.4.3 Recognizing Machining Features . . . . .	143
7.5 Implementation of the Feature Learning Approach . . . . .	151
7.5.1 Teaching Feature Description . . . . .	151
7.5.2 Memorizing the Taught Feature . . . . .	154
7.5.3 Recollecting the Learnt Feature . . . . .	157
7.6 Discussion . . . . .	160
7.6.1 The Feature Recognition Approach . . . . .	160
7.6.2 The Feature Learning Approach . . . . .	164
<b>CHAPTER 8 VERIFICATION OF WORK</b> . . . . .	166
8.1 Grouping and Ordering Machining Features . . . . .	166
8.1.1 Retrieving Machining Features . . . . .	167
8.1.2 Grouping the Machining Features . . . . .	167
8.1.3 Resolving Identical Features Condition . . . . .	170
8.1.4 Sequencing Machining Features . . . . .	171
8.2 Cutter Path Generation . . . . .	173
8.3 Examples . . . . .	178
8.4 Concluding Remarks . . . . .	188



**CHAPTER 9      CONCLUSIONS AND FUTURE WORK . . .** *Page* 189

9.1 Conclusions . . . . .189  
9.2 Contributions to Knowledge . . . . .191  
9.3 Future Work . . . . .192  
    9.3.1 Feature Classification . . . . .192  
    9.3.2 Alternative Cutter Access Vectors . . . . .193  
    9.3.3 Instructing Multiple Features . . . . .194  
    9.3.4 Learning Technique Enhancement . . . . .194  
    9.3.5 General Research Directions . . . . .196

**REFERENCES . . . . .** 197

**APPENDICES**

Appendix A : Derivation of the cavity volume boundary expression . . . . .210  
Appendix B : Feature Representation - Illustrated Examples . . . . .213  
Appendix C : Line/surface Intersection . . . . .218  
Appendix D : Line/polygon Intersection . . . . .220  
Appendix E : The Modified Winged-edge Data Structure . . . . .222  
Appendix F : Aligning the Cutter Axis Vector with the Z-axis . . . . .226

# CHAPTER 1

## INTRODUCTION

### 1.1 Computers in Manufacturing

During the past four decades, major developments in the type and extent of manufacturing automation were made possible largely through rapid advances in the capacity and sophistication of computers. The significant stages of progress in the exploitation of computers in mechanical parts manufacturing industries is summarized in Fig. 1.1.

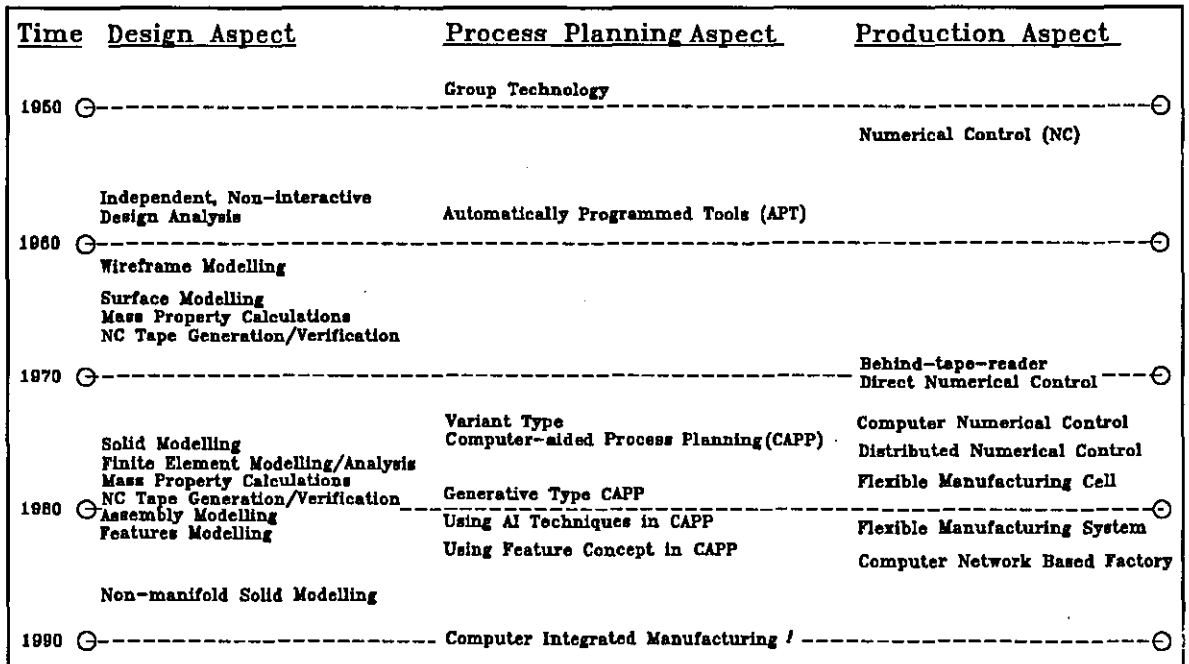


Figure 1.1 : Major developments of computer-based manufacturing automation.

As shown in the figure, the use of computers in design activities has evolved from non-interactive design analysis through simple wire-frame drafting to solid modelling and analysis. Non-manifold geometric modelling technologies [Weiler88] are also emerging. These technologies will have high potential value for applications such as laminate materials stress analysis.

A considerable research effort has been conducted to automate process planning which is the traditional link between design and manufacturing. The development of Computer-Aided Process Planning (CAPP) systems has advanced from the group technology [Gallagher73] coding based systems to the highly automatic systems that emphasize the integration with solid modellers for obtaining part description and the incorporation of planning logic using artificial intelligence (AI) techniques [Alting89].

The technology of computer control of production machines has progressed remarkably since the demonstration of the first stand-alone numerically controlled (NC) milling machine in 1952 at Massachusetts Institute of Technology [Pressman77]. Subsequently, the need for using large NC part programs has led to the development of direct numerical control technology by which NC part programs are transmitted directly from a central computer to serve a group of NC machines. The rapid technological advancement in manufacturing micro-electronic devices has accelerated the development and application of sophisticated computer numerical control (CNC) machines. Installation of highly computerized manufacturing systems, known as Flexible Manufacturing Systems (FMS), are proliferating throughout the world [Kochan86]. These systems have high adaptability to changes of manufacturing conditions, and hence they represent a strategy to increase productivity of batch production. Many conventional production management techniques are implemented as computer programs for enhancing the performance of various production functions such as production planning, material requirements planning, plant layout, and cost accounting. To strive for higher productivity and flexibility, modern factories have utilized computer-network based systems in the planning, management, and operational control functions through either direct or indirect computer interfaces with manufacturing resources.

In retrospect, it is found that much outstanding progress has been made in a variety of Computer-Aided Design (CAD) and Computer-Aided Manufacturing (CAM) applications. However, the past CAD/CAM development effort tended to be dispersed, and consequently, productivity improvement has been localized in individual 'islands of automation'.

## 1.2 Moving Towards Computer-Integrated Manufacturing

The type of manufacturing business environment that is of main concern today consists of a highly competitive, rapidly evolving market. Stringent customer expectations of faster delivery, shorter product life cycle, higher quality and less expensive products have made manufacturing support issues, such as the reduction of manufacturing lead times and the integrity of product information for efficient and effective sharing amongst various manufacturing functions, become more and more critical. In the ultimate effect, the hostile market environment has caused a change in manufacturing cost patterns such that direct manufacturing costs, such as material and labour costs, often represent only a small percentage of total production cost and indirect or manufacturing support costs are a very large portion of total cost [Thomson86].

Hence, to improve manufacturing productivity it is necessary to reduce heavy manufacturing support costs. The reduction of heavy manufacturing support costs is not to be accomplished by merely automating each step of the design and manufacturing cycle. It is also necessary to improve coordination and control between the automated steps of the entire manufacturing business. For instance, CAD technology has improved design productivity in terms of decreasing the product innovation lead times and costs. However, if the CAD information can also be utilized directly by other manufacturing functions such as process planning and inventory planning, any sudden design changes can then be propagated quickly and accurately throughout the manufacturing system. Appropriate corrective actions such as using alternative process plans can then be taken swiftly to bring the manufacturing system back to a stable condition. In effect, the overall productivity gain will be much more significant. Thus the trend in manufacturing automation is towards total factory automation or Computer-Integrated Manufacturing (CIM) which promotes the computer-integrated coordination of overall design and manufacturing functions.

Achievement of the goal of CIM requires a genuine integration of CAD and CAM into an integral computer-driven manufacturing system. However, the main

problem of combining these two principal manufacturing functions is in the communication of information between them. Although methods [Baer79, Requicha80] of developing CAD systems [Requicha82] that could manipulate modelled objects as complete geometric and topological solids are available, the diversities of CAM activities, such as process planning and automatic assembly, still cannot make full use of the CAD-generated object definition because it exists in terms of low-level geometric/topological data. Consequently, the current inter-linking of CAD and CAM has had to seek recourse to human assistance for confirming design purpose and manufacturing methods from the CAD models.

### **1.3 Features : a Methodology for Integrating CAD and CAM**

Years of research and development experience in the CAD/CAM research community has led to a consensus that a higher level of abstraction of design entities is needed for tightly coupling CAD and CAM. Such a collection of enhanced representations of design entities are generally referred to as "features".

There have been many different feature definitions found in the literature [Shah88a]. A feature definition given by an author basically reflects the insight, research approach and application context of that author. For instance, Henderson [Henderson84] identified and extracted manufacturing features, such as holes and pockets, from the boundary database of part models, and thus he defined a feature as : "a set of connected faces related to a specific manufacturing process". Cunningham and Dixon [Cunningham88] advocated design directly from features and saw a feature as : "a geometric form of entity that is used in reasoning in one or more design or manufacturing activities". To encompass the design-oriented and manufacturing-oriented views of features, Wilson and Pratt [Wilson88] gave a traditional and broad definition as : "a feature is a region of interest in a part model".

It is difficult to define features precisely because the interpretation of a feature is strongly associated with the feature's application and parameters. As illustrated in Fig. 1.2, applications are heterogeneous tasks in areas of activity, such as design, analysis, and manufacturing, within different engineering disciplines such as mechanical and electronic engineering. Parameters are attributes, such as dimensions, tolerances, and surface conditions, for supporting applications.

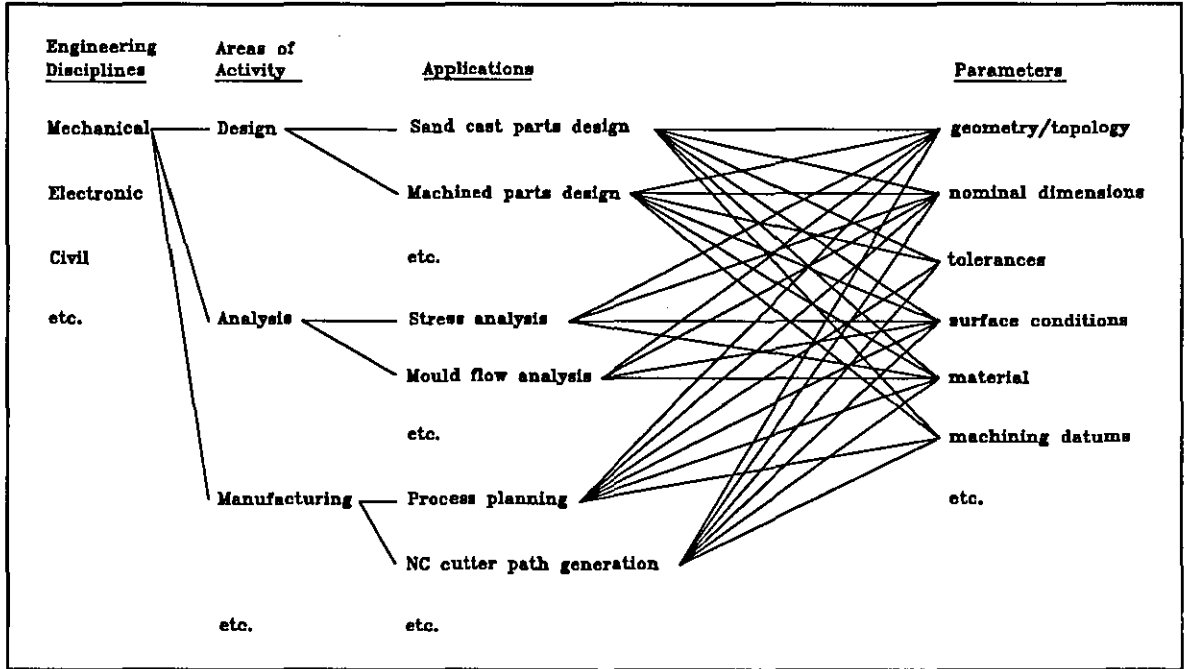


Figure 1.2 : The concept of application dependence of features.

For instance, in the mechanical engineering discipline, the feature "A" shown in Fig. 1.3(a) is generally considered to be a slot. In the eye of a designer, the slot may be viewed as one kind of functional feature that can be used to restrain the movement of a mating part. To a machinist however, the observation of the slot may stimulate the thinking of a slot machining operation.

To illustrate the significance of feature parameters on feature interpretation, the geometric aspect of the part shown in Fig. 1.3(a) is modified to become the part shown in Fig. 1.3(b). Many design/manufacturing engineers would now prefer to call the feature "B" as a notch or non-corner notch [Butterfield87]. However, the criteria, such

as the range of dimensional parameter values, for uniquely differentiating a notch from a slot are elusive.

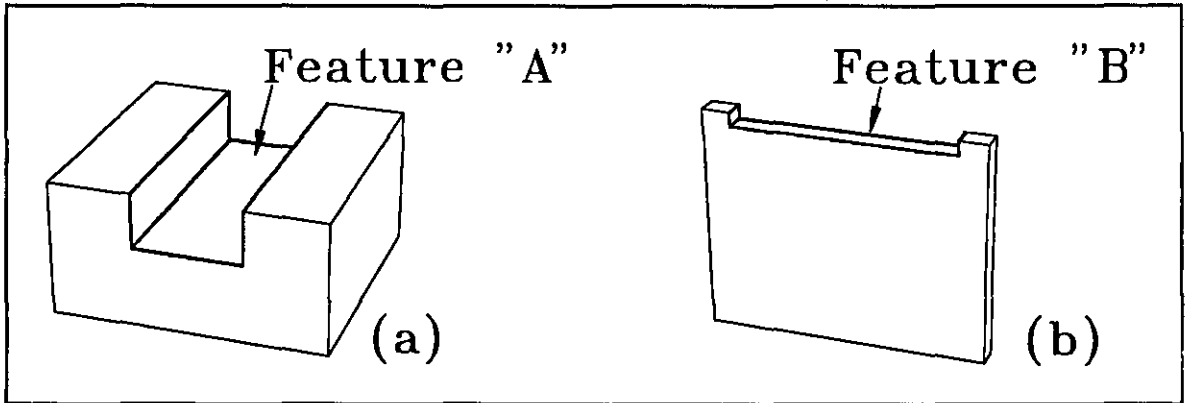


Figure 1.3 : Multiple views of features.

Another example, borrowed from the ideas of Pratt [Pratt87], is illustrated in Fig. 1.4.

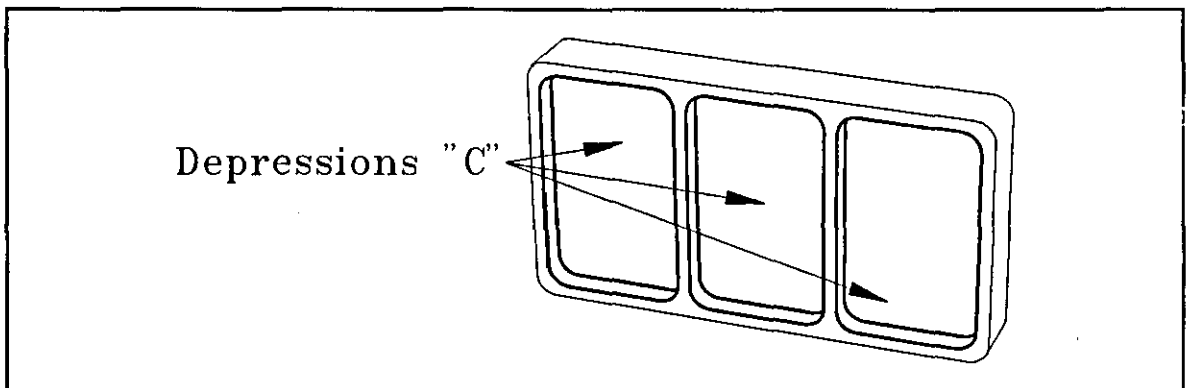


Figure 1.4 : Design oriented view and manufacturing oriented view of features.

As shown in the figure, if a machining-oriented interpretation is adopted, the three depressions "C" would be considered as three disjoint machining pockets. However, if the interpretation is design-oriented, the same depressions could be deemed as the web space formed within the boundaries of reinforcing ribs. With this form (basic geometry and topology) feature oriented interpretation, a variety of manufacturing processes such as casting; forging; sintering; chemical milling; etc., could also be conceived for creating the shape of the part (the two words, "form" and

"shape", will be used synonymously in this thesis). However, if the part is to be produced by a chemical milling process or attributes such as surface texture and over-etch factor are specified, then interpretation will be more certain in the sense that the depressions signify the design intent of increasing the strength/weight ratio of the part and that the formation of the depressions is likely to be due to the removal of material by the etching action of a chemical milling operation.

Thus the absence of a precise definition of features is very much due to the problems arising from the wide ranging applications in which different specialists share the same design model but reason about it using their own vocabulary. Nevertheless, features can be comprehended from the computer standpoint as some intelligent constructs of data and algorithms such that when the data/algorithms are processed by computer, they have the subtle effect of interpreting, generating, and propagating design purpose and manufacturing semantics amongst computer application modules. This important capability of conveying and manipulating design and manufacturing knowledge is fundamental to the objectives of linking CAD and CAM. In other words, a feature representation of manufacturing products can significantly enhance the integrity and semantics of product information so that it can be utilized as a common database to support a diversity of CAD/CAM applications.

#### **1.4 Principles of Existing Feature Modelling Approaches**

The approaches adopted by researchers for achieving a feature representation can be classified into three basic types : (i) human-assisted feature definition; (ii) automatic feature recognition; and (iii) design by features.

The human-assisted feature definition approach usually involves the construction of a 2D/3D wireframe or boundary database of solid model. The boundary database created is then rendered as an image of the model on a cathode-ray-tube display to allow the user to interactively pick topological entities, such as edges and faces, needed to define a feature such as a hole.



The automatic feature recognition approach simulates human design/manufacturing behaviour in interpreting the design information of a part through the implementation of reasoning abilities in computer programs to identify and extract relevant feature data from the part model database or some transformed version of the part model database.

The design by features approach aims to devise a feature-based modelling environment for design engineers to create part models directly from features right from the beginning of design. Generic feature definitions are maintained in a library from which features are instanced by specifying various parameters such as dimensions and locations.

### **1.5 Motivation of Research**

The human-assisted feature definition approach has been a traditional method used for inputting data for applications such as defining machining faces for NC cutter path generation. However, due to the need for human intervention, using this approach alone is not promising towards the goal of CIM.

The design by features approach is in agreement with the simultaneous engineering concept as the simultaneously enhanced feature model can retain design intent and manufacturing purpose for the concurrent support of other applications. However, the need for feature reasoning still exists because feature interpretation is application specific as discussed earlier. Moreover, feature characteristics may change when features interact during the design process. For instance, spatial interactions between the generic features that exist in the feature database can result in non-generic shapes. To reduce the complexity of these problems, many feature-based modelling systems have been implemented based on manufacturing-oriented features and restrictive criteria of feature interaction. As manufacturing features may not be compatible with design features, these systems have drawbacks such as low autonomy of design for function, limited choice of manufacturing processes, and problems related to the

determination of the level of abstraction and operations of the set of generic features.

The automatic feature recognition approach is attractive partly because it is automatic in nature and partly because the feature recognition algorithm can be constructed to suit different applications. Nevertheless, in the context of recognizing machining features such as holes and pockets from mechanical parts, the following drawbacks of the existing recognition technology have been identified :

1. Existing methods have not sufficiently exploited the tool accessibility information and machining heuristics (rules of thumb) in the process of feature recognition.
2. Shape complexity of both the machining features and mechanical parts considered tends to be relatively simple.
3. The 'recognizing intelligence' in the recognition algorithm is usually rigidly implemented with respect to the characteristics of a predefined set of feature primitive templates, and thus the approach has frequently been hindered by the limited range and complexity of features that can be recognized.

## 1.6 Research Objectives

The specific problems stated above initiate two general research objectives :

1. The first objective is to devise a feature recognition procedure that exploits the tool accessibility information and machining heuristics as clues for recognizing 2.5D machining features that exist in reasonably complex machining part designs such as those illustrated in Fig. 1.5.
2. The second objective is to devise a machine learning [Cohen83] approach by means of which the recognizing intelligence of the machining feature extraction system

developed in the first objective can be increased during the system service life.

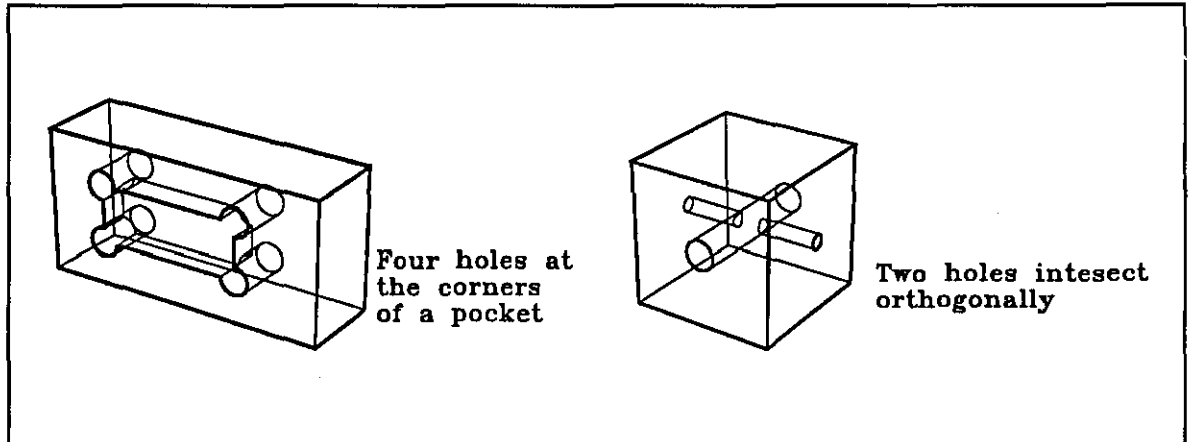


Figure 1.5 : Examples of complex machining parts to be handled.

## 1.7 Research Methodologies

The research methodologies employed involve the implementation of a prototype knowledge-based machining feature extraction system which is constructed by integrating the VAX-OPS5 [Forgy77] rule-based AI environment with the PADL-2 [Brown82] solid modeller. Specification of original stock and finished part geometry within the solid modeller is followed by determination of the nominal surface boundary of the corresponding machining volume model by means of Boolean subtraction and boundary evaluation [Requicha85a]. The boundary model of the machining volume is managed by using winged-edge [Baumgart74, Weiler85] and frame-based [Minsky75] data structures.

For the achievement of the first objective, a machining feature recognition procedure which employs rule-based and procedural programming techniques has been devised. The feature recognizer uses built-in machining heuristics and tool accessibility analysis to identify and extract 2.5D machining features from a machining volume. The tool accessibility analysis is based on ray-casting technique, and the results are propagated into a frame-based data structure which acts as an agenda for guiding feature searching. Instead of rigidly classifying the feature type, a recognized machining

feature is represented in terms of its tool entry face and part face identities that are used in its winged-edge boundary model.

For the achievement of the second objective, a machine learning approach is adopted by means of which the user is allowed to interact with the wireframe display to define tool entry and part faces of the machining volume. These taught faces together with the boundary description of the machining volume are converted into production rules. These new rules are incorporated into the knowledge base allowing subsequent recognition of similarly shaped machining volumes and hence the generation of appropriate machining faces. This method is intended for customization to handle factory dependent machining features or machining features that cannot be machined by simple cylindrical cutters such as end-mills.

The validity and practical usefulness of the approach is demonstrated by the inclusion of an NC cutter path generating module that utilizes the winged-edge data structure for the post-processing of the extracted machining features into NC part programs.

## CHAPTER 2

# LITERATURE REVIEW

Most feature modelling methods are based on a geometric modelling concept. Thus prior to the review of previous works on feature modelling, the geometric modelling methods are overviewed.

### 2.1 Geometric Modelling Methods

The geometric information of a solid part can be classified into two aspects : basic and variational. The basic geometric information refers to the ideal geometric (metric information) and topological configuration (shape information) of the part, while the variational geometric information refers to the allowable deviations of the ideal nominal shape such as geometrical tolerance and surface finish. The two approaches commonly used to model variational geometric information are parametrization and offsetting. Parametrization is done by using basic parameters to model the nominal geometry of a part. In turn, the basic parameters are associated with limiting parameters that correspond to the permissible variations of the basic shape. Offsetting is a non-parametric approach where the boundary of the nominal part is offset by the amount of the specified tolerances to generate the limiting parts.

Undoubtedly, variational geometric information is of paramount importance in the CAD/CAM context as many valuable design and manufacturing clues can be implied. Nevertheless, the formal study of representing variational geometric information is still an independent research issue [Requicha83, Juster92]. Most state-of-the-art geometric modelling systems still treat variational geometric information as precision features which are augmented in the basic geometric model as attributes based on the principle of the human-assisted feature definition approach. In this thesis, interest is focused on extracting machining features based on nominal shape information, and

hence only basic geometric information is considered.

There are three basic geometric modelling approaches : (1) 2D/3D wireframe modelling, (2) surface modelling, and (3) solid modelling.

Wireframe modelling only models the basic geometric framework of a solid part. For instance, a rotational part can be modelled by describing its surface profile as a 2D wireframe contour, while a prismatic part can be represented by its vertices and the edges joining the vertices. Thus wireframe representations do not provide a complete surface and volumetric description of physical parts, and hence human interpretation is necessary to define the missing information. Despite this, wireframe modelling is still an important basis for feature modelling. For example, 2D wireframe representations are often used for representing and extracting features of rotational parts [Joseph90] whereas 3D wireframe models are popularly adopted for quick display and verification of the geometry of feature models [Luby86].

Surface models take the modelling of an object one step beyond wireframe models by providing information on surfaces connecting the object edges. Typically, a surface model consists of wireframe entities that form the basis to create surface entities which can be analytic or synthetic. Analytic surface entities include planar surface, ruled surface, surface of revolution, etc., while synthetic surface entities include the bicubic Hermite spline surface, B-spline surface, rectangular and triangular Bezier patches, etc. [Rogers89]. As surface information is included, surface models are less ambiguous. They have been utilized in representing complex feature geometry such as in mould/die surface modelling, NC path generation, and interference detections [Choi88, Gandhi89].

Solid modelling is the highest level of geometric modelling technology in the sense that it can provide complete and unambiguous geometric and topological information of a part. Historically, several different solid modelling methods have been developed. The following five are typical :

- (1) Parametrized Shapes;
- (2) Spatial Occupancy Enumeration;
- (3) Sweep Representation;
- (4) Constructive Solid Geometry (CSG); and
- (5) Boundary Representation (B-rep).

CSG and B-rep are the best understood among these methods. They form the basis of most of the contemporary solid modelling systems and are widely used in feature modelling work. More importantly, they are also involved in this research, and hence their modelling principles and properties are briefly described below. For a formal discussion of solid modelling technologies, the reader is recommended to study references such as [Baer79, Requicha80 and Requicha82].

### 2.1.1 Constructive Solid Geometry (CSG)

In CSG modelling, an object is represented as an ordered, binary tree of primitives and regularized Boolean set-operations [Requicha78]. As shown in Fig. 2.1, the terminal nodes of a CSG tree are primitives, while the non-terminal nodes represent regularized Boolean set-operations applied to the two sub-nodes. The primitives can be solid primitives or half-spaces that are associated with necessary rigid-body transformations for achieving the desired position and orientation.

Commonly used solid primitives are blocks, cylinders, spheres, wedges, cones and tori. Each solid primitive is internally predefined as the volume bounded by a set of half-spaces which are closed (continuous without breaks) and orientable (side-wise distinguishable) surfaces such as planar and cylindrical surfaces. For instance, a cylinder primitive can be defined as the volume formed by the regularized intersection of two planar half-spaces and one positive cylindrical half-space as illustrated in Fig. 2.2. Some CSG-based systems, such as PADL-2 [Brown82] and TIPS-1 [Okino73], allow the use of both solid primitives and half-spaces to create solid models.

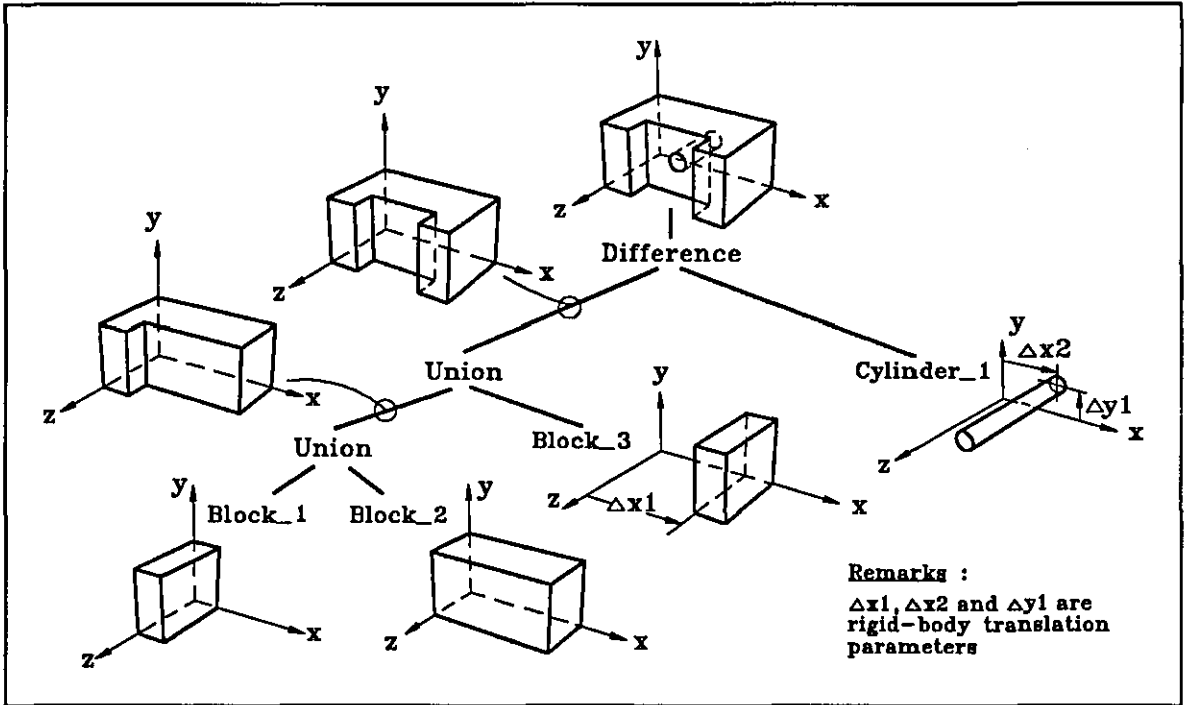


Figure 2.1 : CSG tree representation.

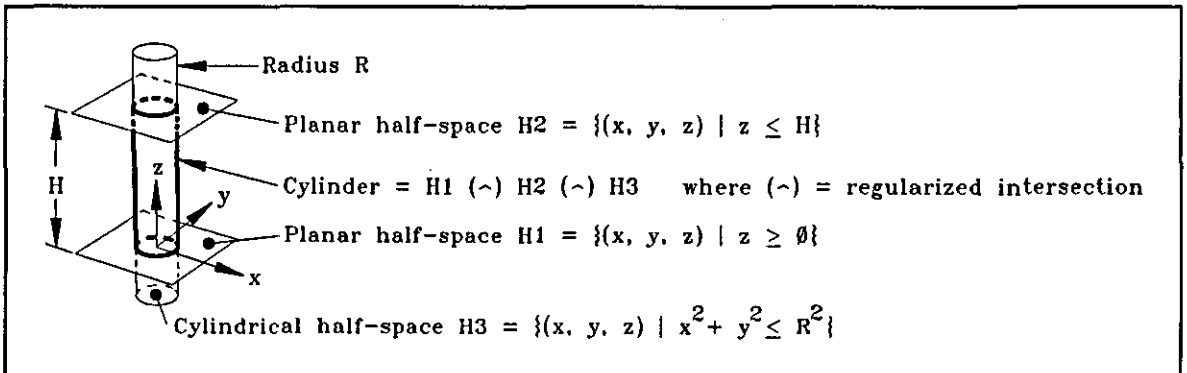


Figure 2.2 : Construction of cylinder primitive by using half-spaces.

The regularized Boolean set-operations are union, intersection, and difference which can be considered as the 3D versions of their respective conventional Boolean algebra counterparts, i.e. OR, AND, and NOT AND. Regularized Boolean set-operations are used to ensure that CSG objects are homogeneous solids which will not contain awkward components such as dangling faces and edges as illustrated in Fig. 2.3.



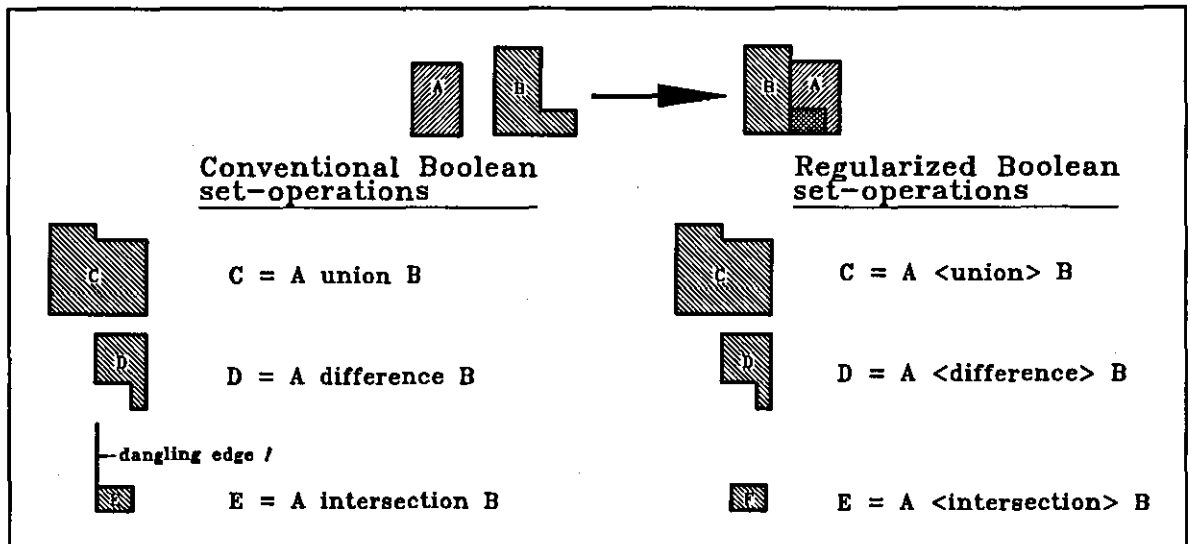


Figure 2.3 : Regularized Boolean operations ensure modelled solids are 'true' solids.

CSG representations are informationally complete but they do not provide the geometric and topological information explicitly. Whenever the boundary information is needed, the CSG representation has to be evaluated by using a procedure called boundary evaluation [Requicha85a, Voelcker81]. This procedure is computationally expensive but is necessary even for applications such as line drawing display of objects. Any changes made in the boundary information cannot be transmitted backward for updating the original CSG representations because the theories and algorithms [Shapiro91] for converting boundary information to the corresponding CSG representations are still not well researched. Moreover, due to the use of combinatorial Boolean operators, CSG representations are not unique. Figure 2.4 illustrates one of the many possible alternative CSG representations for the same object shown in Fig. 2.1.

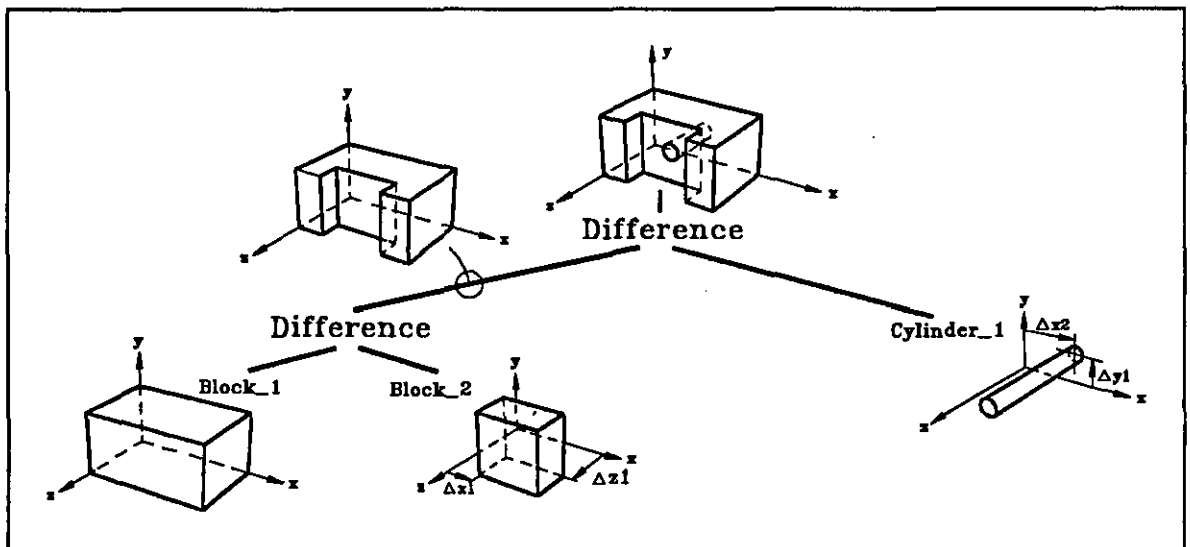


Figure 2.4 : Alternative CSG representation of the part shown in Figure 2.1.

### 2.1.2 Boundary Representation (B-rep)

In B-rep modelling, an object is represented in terms of its boundary faces. A face is conceived as a bounded region of a closed and orientable surface. It is usually defined in terms of its surface definition and bounding curves that are known as edges. In turn, each edge is expressed in terms of its curve definition and ending points that are known as vertices. Thus the database of a B-rep model basically contains the model's geometric entities, topological entities, and topological relationships as illustrated in Fig. 2.5. Strictly speaking, data such as surface equations of bounding faces and their spatial locations are referred to as geometric entities. Topological entities include faces, edges, vertices, etc., whereas topological relationships are structural connectivity pointers that specify how the geometric/topological entities are related with each other.

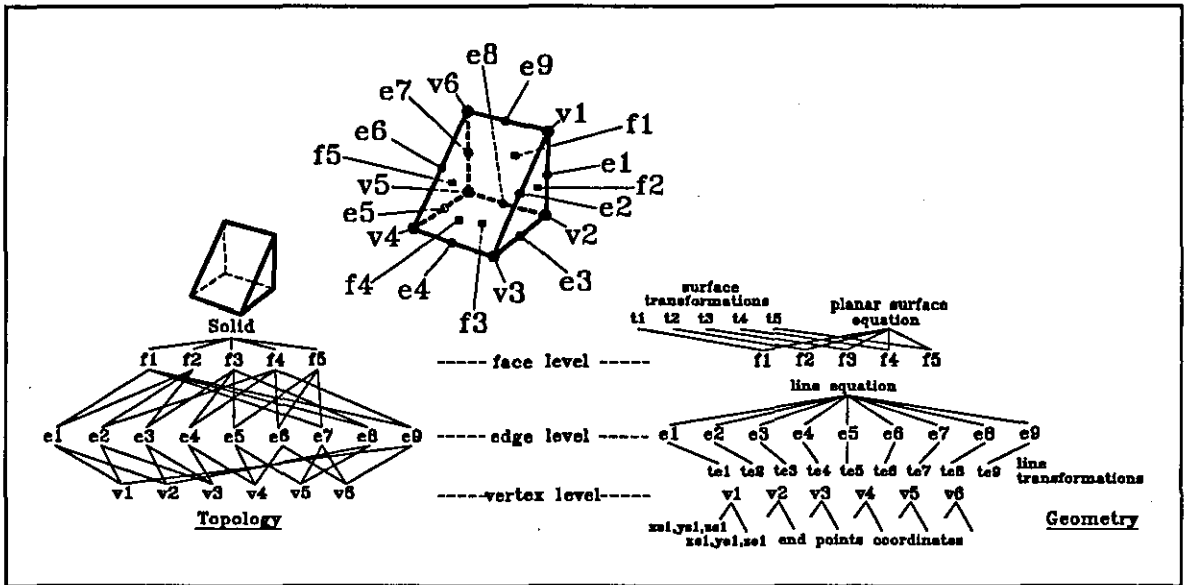


Figure 2.5 : B-rep of a polyhedral object.

Creation and manipulation of geometric/topological entities involve Euclidean (3D) geometry calculations and Euler operations. Euler operations [Eastman79] are based on the Euler-Poincare formula whose general form is :

$$v - e + f = 2 (s - h) + r$$

where  $v$  = number of vertices;

$e$  = number of edges;

$f$  = number of faces;

$s$  = number of shells (disconnected components);

$h$  = number of holes through the modelled solid; and

$r$  = number of rings (cavities) in faces.

This Euler-Poincare formula relates the number of basic topological entities in a polyhedral object and, consequently, is useful for checking the validity of B-rep models. In B-rep systems such as BUILD-2 and ROMULUS [Hillyard82], the rather

unintuitive Euler operations are upgraded to user-oriented operations such as chamfering and tweaking as illustrated in Fig. 2.6. The Boolean operations used in CSG systems are also used in B-rep systems to combine individual objects together to form a composite object. While the object modelling is performed incrementally via the use of these operations, the object's B-rep is constantly updated, and hence users can appreciate the instant change of shape of the modelled object.

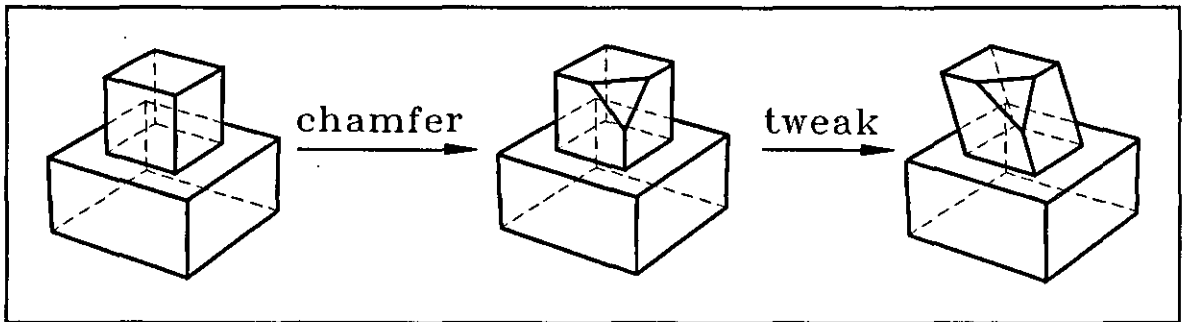


Figure 2.6 : Higher level B-rep operations.

Due to the provision of explicit boundary information, B-rep modelling schemes are very useful for applications such as graphic display and NC cutter path generation. Volumetric properties can also be computed by virtue of the Gauss divergence theorem which relates volume integrals to surface integrals [Lee82a, Lee82b].

In summary, both CSG and B-rep solid modellers can provide a database that describes the geometry and topology of physical objects. Nevertheless, the model database is only sound for describing the syntactic information content and not the semantic information content in terms of the engineering meaning of the model.

## 2.2 Feature Modelling Methods

As introduced in section 1.4, there have been three broad approaches adopted by researchers for modelling engineering meaning in the product database : (1) human-assisted feature definition, (2) automatic feature recognition, and (3) design by features. The literature on feature modelling is voluminous. Hence it is only possible to review a few representative examples that can help in understanding the methodologies used as well as possible shortcomings. As the focus of interest of this thesis is on the automatic feature recognition approach, the related investigations will be elaborated, while the background work of the other two approaches will only be briefly described. For a more comprehensive literature survey and discussion of feature modelling technologies, references such as [Pratt88, Shah88a, and Case92] can be pursued.

### 2.2.1 Human-Assisted Feature Definition

This approach has been widely used for augmenting data such as geometries and tolerances in the design for facilitating process planning and NC cutter path generation. For example, Chan [Chan82] developed a group of interactive commands on top of those provided by the ROMULUS [Hillyard82] B-rep modeller to manipulate the boundary model in such a way that appropriate faces and edges of a part can be tagged for automatic generation of the APT (Automatically Programmed Tools [IIT67]) geometry statements.

In [Requicha85b], an essentially B-rep data structure called a VGraph (variational graph) was implemented within the PADL-2 [Brown82] CSG modeller for interactive definition of features. The VGraph is needed since CSG schemes alone cannot support interactive manipulation of the boundary model. Features are defined in terms of groups of faces and edges with which attributes such as tolerances and datum systems are associated. The structure is utilized in a high level machining language called MPL (Machining Process Programming Language [Chan86]) for specifying

machining features such as holes and pockets.

### 2.2.2 Design by Features

Early works [Chang81, Descotte84, Berenji86] established feature model databases for facilitating computer aided process planning in the absence of a geometric modeller. The modelling process often involves textual input of machining features and part geometry information using a customized feature description language. For instance, Chang [Chang81] used a command language driven dialogue input method for the design of parts with holes that are described in terms of parameters such as diameter, upper chamfer, bottom chamfer and geometric tolerances. The 2D graphic image of the defined part is rendered on a CRT display screen, and the established hole database is processed by the APPAS [Wysk77] generative process planning system for the generation of appropriate machining sequences and parameters.

The modern implementation of the approach is similar to that of the CSG modelling method. It usually involves the definition of generic features in a library from which features are instanced by specifying relevant feature parameters such as dimension, location and various attributes for establishing the feature model of the part. The resulting feature model can provide additional information such as feature types, design rules, tool entrance directions and manufacturing sequences, and hence the need for inferring such important engineering meaning from other descriptions of the part model can be circumvented.

Most of the existent machining features based design systems [Hart86, Cutkosky88, Chang88/Turner88, Unger88, Tsang89, Hummel89/90] are based on Arbab's [Arbab82] 'deforming solid geometry' approach by which a part is modelled incrementally by subtracting features from a starting base solid. For example, the "First-Cut" system reported in [Cutkosky88] uses a solid modeller called Alpha\_1 and the starting base solids are meant to be extruded bar stocks. Design is conducted by

successively inputting a series of high-level manufacturing operation commands such as make hole and make pocket. Internally, the commands activate the corresponding Boolean subtractions of generic features from the based solid to yield the desired part. The system uses the renowned GARI [Descotte84] as the underlying process planner. NC code and inspection plans can also be generated.

Works such as [Luby86, Chung88, Cunningham88, Shah88b/Shah90] adopt a more flexible feature modelling process that allows user to design by adding, subtracting, and manipulating features. For example, Luby *et al* [Luby86] developed a metal castings design system which allows the use of both additive and subtractive features such as slabs, ribs and holes. Chung *et al* [Chung88] built a similar gating design system for investment castings by integrating a commercial B-rep modeller (I-DEAS, originally GEOMOD developed by [Baumgart74]) with an expert system shell (KEE). However, only additive features such as fillets and webs are allowed in the system. Rules of good casting practice are embedded in the feature definitions so that when features are instanced, the validity of the design can also be verified automatically.

Cunningham and Dixon [Cunningham88] proposed a comprehensive set of features based on an examination of the heuristics for a wide range of design and manufacturing process/activities such as functional design, manufacturability evaluation, and inspectability. The feature sets are related to a knowledge-based design by features system architecture that is not associated with a geometric modeller. The system basically consists of a user interface, a working features library, a feature operations library and an operations monitor. The authors advocated that such an intelligent design system will allow users to synthesize and transform the primary working features into a higher level secondary features that can provide the necessary information to support various manufacturing applications simultaneously.

Shah *et al* [Shah88b, Shah90, Shah91b] reported the development of a testbed system that consists of a feature based design shell and an application mapping shell.

The design shell allows one to integrate additive/subtractive features, geometry, topology, and design rules into a unified product description. The mapping shell performs feature reasoning and relates the established feature information to various applications such as manufacturability evaluation, GT coding, stress analysis, etc. A mechanism for interactive definition/recognition of features was also reported. The system can identify the entities that make up the features. The work for creating a feature model from the acquired information is still in progress.

Gindy [Gindy89] emphasized the need for a structural scheme to represent and manipulate features. He proposed a feature taxonomy where generic form features are conceived as volumes enveloped by entry/exit and depth boundaries. The feature classification is based on the "external access directions" (EADs) from which the feature volume could be removed by cutting tools. For instance, a through slot feature has three EADs whereas a step feature has four EADs as illustrated in Fig. 2.7.

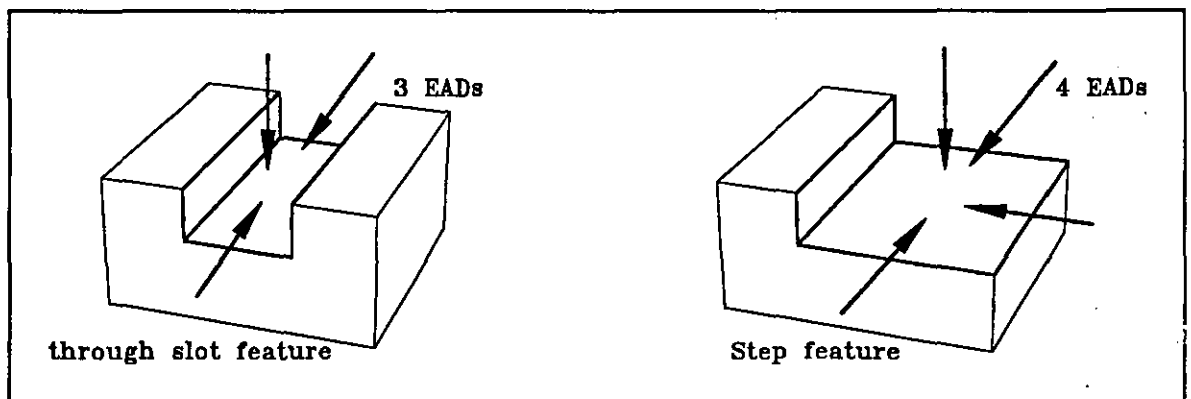


Figure 2.7 : External access directions for a through slot and a step.

At the highest level of the feature taxonomy (Fig. 2.8), form features are divided into three generic categories : protrusions, depressions and surfaces. Feature geometry is represented by defining the EADS, the boundary wall type (open or closed) and the exit boundary status (through or not through). Grouping feature geometry characteristics structurally in this way produces a list of form features classes that correspond to common geometric shapes such as bosses, pockets, holes, slots, notches,



and real and imaginary surfaces. Secondary feature forms such as gear teeth, screw threads, and knurl can also be described by the taxonomy as specific local geometry superimposed on the basic form feature. The feature taxonomy is useful not only to the design by features approach for structuring design features but also to the feature recognition approach for governing the design of feature recognition rules. In Gindy's reported work, the taxonomy is used as a generic data structure for conveying feature information of engineering parts to a prototype generative process planning system.

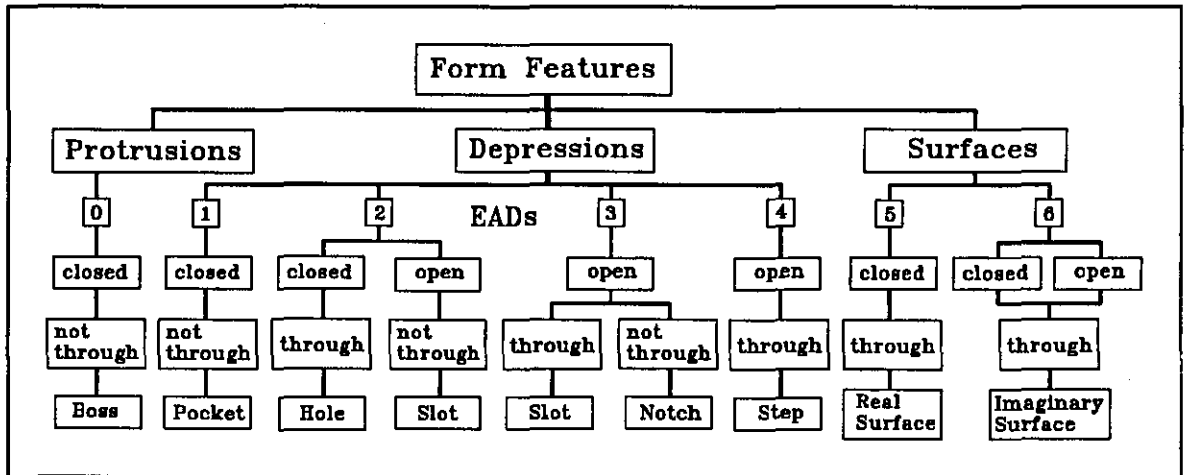


Figure 2.8 : Gindy's form feature taxonomy.

### 2.2.3 Automatic Feature Recognition

This approach assumes that the geometric model contains feature information that can be identified and exposed. The techniques used can be broadly classified into two groups : (1) recognition with CSG models, and (2) recognition with B-rep models.

#### 2.2.3.1 Recognition with CSG Models

One of the earliest works on feature recognition with CSG models was done by Woo [Woo75]. He used a restricted form of CSG with only ADD and SUBTRACT operators to define objects, and considered simple volumetric features such as slots and

holes. Features are extracted from an object's CSG representation by searching CSG patterns that match predefined feature definitions. The program can only recognize some elementary machining features from objects in a narrow domain.

Woo [Woo82] also used a decreasing convex hull algorithm to generate a CSG tree of convex volumes by recursively computing the Boolean difference between an object and its convex hull until the object equals its own convex hull. As illustrated in Fig. 2.9, the original object can be expressed as alternating sums of volumes in the form  $P_0 = H_0 - H_1 + H_2 - H_3$ . This form can be slightly rearranged as  $P_0 = H_0 - (H_1 - H_2 + H_3)$ , where  $H_0$  represents the stock and  $(H_1 - H_2 + H_3)$  represents a number of removal volumes or a sequence of machining operations.

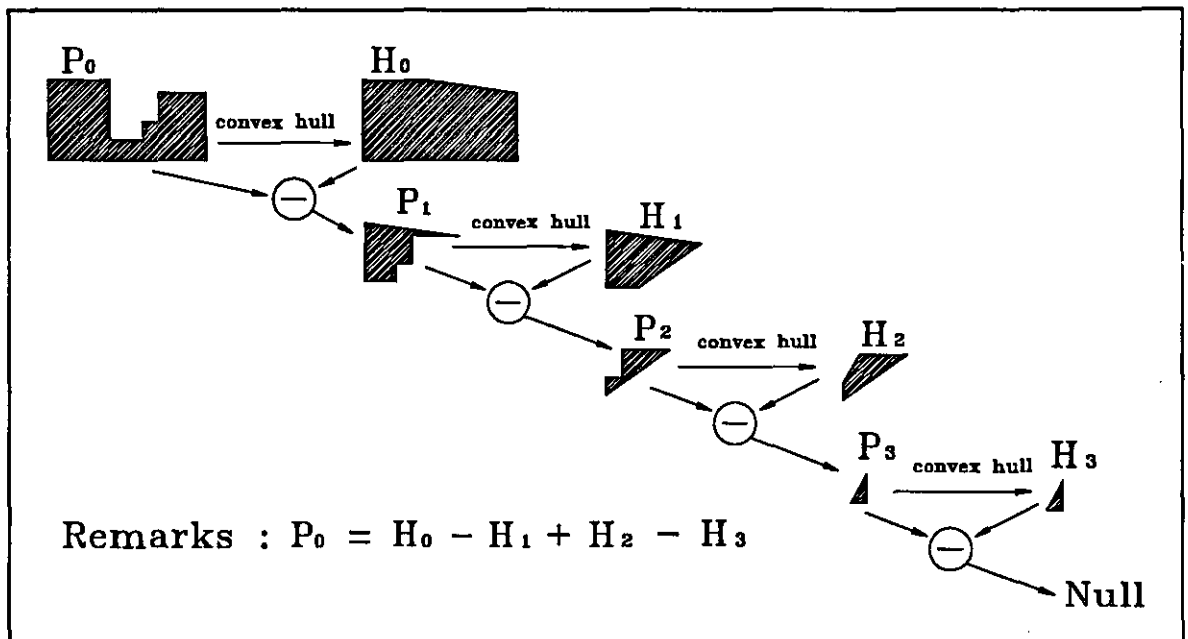


Figure 2.9 : Woo's decreasing convex hull algorithm.

However, the approach has several flaws : (1) the generation of convex volumes is solely based on geometry computation and therefore an odd-shaped removal volume that does not correspond to a single machining operation can result, (2) the sequence of generating removal volumes in the algorithm may not comply with a practical machining operation sequence, (3) the shape of the stock it assumes can be awkward

because it is simply the convex hull of the initial shape, (4) the algorithm is not purely CSG based since B-rep is involved in the convex hull determination, and (5) the algorithm does not converge when a null set condition does not exist. An algorithm for detecting the nonconvergent conditions is reported in a more recent work [Tang91] but the problem of nonconvergence still exists.

Lee and Fu [Lee87] utilized the principal axes of CSG primitives to extract features which are defined as CSG combinations of primitives whose principal axes of symmetry satisfy certain geometric relationships. For instance, the definition of a fillet involves the union and appropriate positioning of two cubes and a cylinder. Once the features are located, the CSG tree is rearranged by using tree manipulation techniques so as to group certain CSG patterns that correspond to solid features. Only a small set of features of very simple and restricted forms can be recognized and the non-uniqueness of CSG representations is not tackled. The main thrust of the work is the development of techniques for moving nodes in the CSG tree. Although a more efficient CSG tree reconstruction algorithm is subsequently reported in [Lee88], the feature extraction and unification methods remain basically the same.

More recently, Perng *et al* [Perng90] described a method for extracting machining features from CSG input. The method involves the conversion of a part's CSG tree representation into an equivalent DSG (destructive solid geometry) tree representation in which the part is expressed as  $S - E_1 - E_2 - E_3 \dots$ , etc., where  $S$  is the stock in the form of a block that bounds the given part, "-" is the Boolean difference operator, and  $E_1, E_2, E_3$ , etc., are the excess material volumes contained in the stock. The excess materials are classified into basic machining features by matching their face patterns with those of eighteen predefined machining feature primitives such as holes and pockets. The basic features were further grouped into composite machining features based on their adjacency relationships. The method has several shortcomings : (1) the original CSG tree input allows only union and difference operations, (2) the method is also not purely CSG based because the B-rep of the object is needed both in the CSG-DSG conversion process and in the feature recognition process, and (3) the

recognizable machining features are limited only to the eighteen predefined feature primitives. Further work from the same working group [Li91] reports an improvement to the method by taking into consideration of the original stock of non-block type in the CSG input.

### 2.2.3.2 Recognition with B-rep Models

The approach generally involves : (1) searching the B-rep model database to match geometric/topological patterns, (2) extracting recognized features from the database, and (3) organizing the recognized features to establish a corresponding feature model database.

The B-rep model database may be represented in different forms such as traditional hierarchical B-rep structures, AI-based representations, and boundary graphs. More than one representations may be used concurrently. Most researchers [Grayer77, Joshi88, etc.] recognize features directly from the finished part model, while a few [Henderson84, CAMI-ANC85] make use of the removal volume model obtained by subtracting the finished part model from the stock model.

Techniques for searching and matching feature patterns vary, from hard-coded, procedural data structure traversal/entity evaluation to AI-based pattern matching and boundary graph manipulation/matching.

Extraction of features ususally involves tagging/collecting face/edge sets of recognized features or generation of feature volumes that correspond to the recognized features. Organization of the recognized features often entails the enhancement of the original database with the inferred feature information or the establishment of a new database such as a feature graph to represent the derived engineering meanings of the model.

For example, early work by Grayer [Grayer77] extracted machining regions by sectioning the boundary model successively with a series of planar surfaces normal to the machine's spindle direction such that the boundary edge loops of 2.5D machining regions in the model can be revealed on the sectioning surfaces. The surfaces are processed by an area clearance machining procedure for cutter path generation. The method weakly assumes : (1) a spindle direction is given, (2) the gap increment between the sectioning surfaces is the desired depth of cut, and (3) the part can always be machined by a sequence of 2.5D pocket milling operations.

Kyprianou [Kyprianou80] described a syntactic pattern, edge-based recognizing algorithm which starts by classifying the B-rep entities. Edges are classified as convex, concave, smooth convex and smooth concave as illustrated in Fig. 2.10.

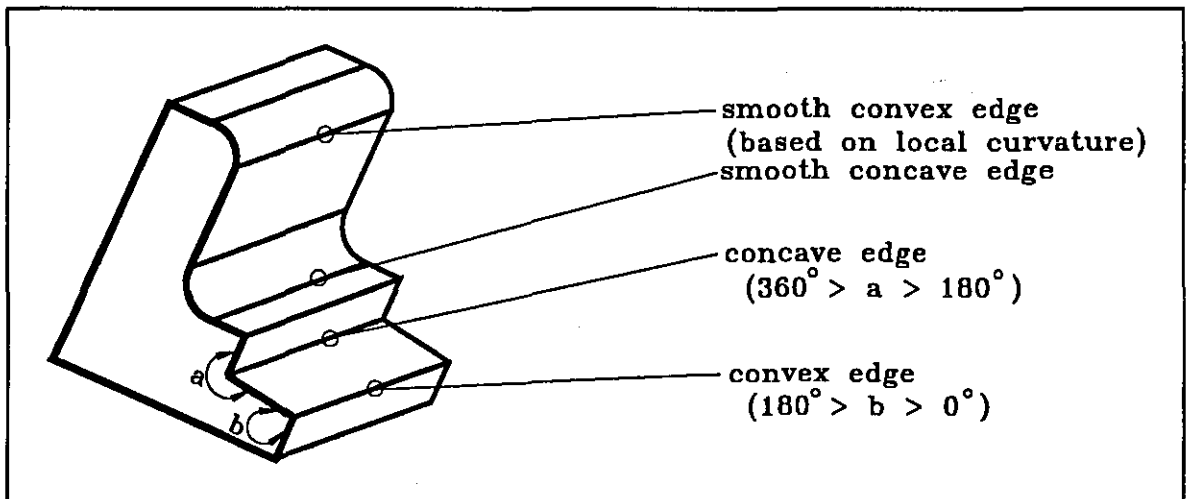


Figure 2.10 : Concavity classification of edges.

Similarly, the vertices and the edge loops are classified depending on the convexity/concavity of their incident edges and constituent edges respectively. Faces are labelled as primary if they contain a concave edge or an inner edge loop. Primary faces are further ordered based on the number of concave edgesets. A hierarchical faceset data structure is established by processing the entity-classified B-rep. Features are determined from the faceset data structure by using syntactic pattern parsing rules. For

instance, an inner loop of convex edges indicates a depression, and similarly, an inner loop of concave edges signifies a protrusion.. The face set data structure is used to generate group technology coding for classification of rotational and prismatic parts. The method works quite well with rotational parts but when used for prismatic parts, cannot recognize certain complex features such as T-slots. Recognized features are marked in terms of face sets, but accessibility information is absent.

The syntactic pattern recognition technique is also used by Choi *et al* [Choi84] to recognize simple features such as holes and pockets. For example, a hole is recognized by searching for circular edges lying in a plane. However, the method fails if a cylindrical hole opens non-orthogonally into a planar face or opens into a non-planar face since the types of edges thus formed are not always circular.

Henderson [Henderson84] used the AI language PROLOG to implement a rather sophisticated feature recognition algorithm that starts by subtracting the finished part from the stock, both represented in the ROMULUS [Hillyard82] solid modeller, to obtain the B-rep of the removal volume or cavity volume. The faces of the cavity volume are tagged as primary or secondary entrance faces according to their accessibility from the exterior or through other faces of the part. The B-rep of the cavity volume is converted into an equivalent set of PROLOG assertions. Features are face sets that satisfy relationships defined by PROLOG rules. For instance, a simple hole must have an entrance face associated with one or more coaxial side faces, and a bottom face. The algorithm searches for the PROLOG equivalent of the B-rep of the cavity volume for face patterns that match the feature rules. Features are searched in the following predetermined order : (1) slots, (2) pockets, and (3) holes. Once a feature is found, the corresponding feature volume is extracted by subtracting it from the initial cavity volume. This recognition procedure is applied recursively to the new cavity volume until the cavity volume is null. The recognized features are organized into a graph that represents the accessibility and adjacency relationships amongst the features.

In Henderson's method, the local accessibility of a feature is defined in terms of the presence of primary or secondary entrance faces in the pattern matching conditions of the corresponding feature rule. In order to ensure that the recognized features are machinable, Henderson performed further accessibility analysis by employing computationally expensive boundary intersection between the extracted volume and the finished part. His notion was that if the intersection is null then its removal will not gouge the finished part. However, this does not ensure global accessibility of a feature. For example, the hole shown in Fig. 2.11 cannot be machined by a cutting tool coming from the left hand side due to the obstruction caused by other components of the part.

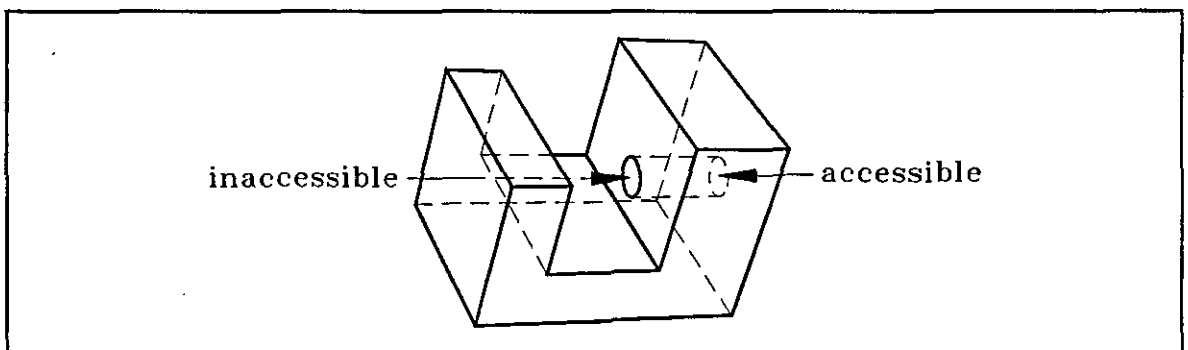


Figure 2.11 : A hole that is not accessible by cutting tool from one end.

Another problem with his algorithm is in dealing with interacting features. When the faces of a recognized feature do not enclose a volume, the algorithm generates volumetric features by computing cross-sections and sweeping them along linear or circular trajectories. The method fails in certain cases as the subtraction of features may cause alteration of the boundary pattern of features in the remaining cavity volume, and consequently, further recognition becomes increasingly complicated.

More recently, Dong [Dong88a, Dong88b] developed a feature extraction system using both procedural and declarative methods. His feature recognizer starts by converting the boundary information of the part from the PADL-2 [Brown82] CSG modeller into a set of LISP frames. A set of machining feature templates such as slots, pockets and holes are predefined, in terms of the feature recognition methods and the

sequence of recognition, in a set of hierarchical frames. The procedural oriented feature finder is a variant of Kyprianou's depression finding algorithm [Kyprianou80]. For instance, recognition of depressions is based on detecting a sequence of convex edges in a single face and recognition of slots is based on identifying two sets of edges that share the same set of adjacent faces. For finding pockets however, two methods are used. The first method looks for the pocket base that is totally enclosed by a concave edge periphery. But when the pocket is a through pocket, the first method fails as there is no concave edge periphery in the pocket base. The second method, searches for a set of faces whose edges form a closed loop of convex edges on a single top face. Feature volumes are created from the recognized features by a face extension technique which is based on the assumption that all surfaces required to bound the feature are planar and are present in the model.

Another part of Dong's work is the use of a declarative approach for users to define new feature types by using a special Feature Description Language (FDL). A FDL definition of a feature is equivalent to a semantic net, which consists of a set of nodes of geometric entities with labelled arcs indicating the relationships between the nodes. The FDL definitions are implemented also by LISP frames. A FDL interpreter is used to search for patterns that satisfy a feature's FDL definition thereby finding all the feature instances in the part model. His feature recognition algorithm can find several types of predefined machining features but is still unable to deal with general feature interactions. He weakly assumed that all machining features are machinable depressions and ignored features such as slabs and profiles. His idea of providing a means for the user to define new features is good but the definition of new features has to be based on the use of the non-interactive proprietary FDL.

A similar approach is also reported in [Sakurai88] where the recognizer is taught a new feature by interactively selecting faces of an example feature. The number and type of geometric entities as well as their connectivity are stored in a feature graph. Feature recognition is done by matching the feature graph with the subgraphs of the B-rep graph of the part model. However, the recognizer cannot recognize positive features



such as bosses and islands and features which can have a variable number of faces like general pockets and holes since the definition of features has to conform to a restricted set of feature facts. The system also has difficulty in recognizing interacting features.

Recently, quite a number of researchers have used graph-based techniques for feature representation and recognition. For example, Falcidieno [Falcidieno87] reported the use of a face-based B-rep called a Face Adjacency Hypergraph (FAH) for feature recognition. The nodes of a FAH are the object faces, whereas the arcs and the hyperarcs represent the relationships among the faces induced by sets of the edges and the vertices. She used Kyprianou's syntactic pattern recognition algorithm to recognize feature faces that do not form closed volumes. However, during the feature extraction process, a set of dummy faces, edges and vertices are generated to complement the recognized feature boundary to form volumetric features. This was achieved by extending those edges that do not belong to the convex boundary of the feature to form new vertices on the convex boundary. As illustrated in Fig. 2.12(a), edges  $e1$ ,  $e2$  and  $e3$  are extended to intersect thereby forming a new vertex  $v1$ . New dummy faces can then be formed to create a feature volume. However, as shown in Fig. 2.12(b), the method fails when all the new vertices required cannot be generated.

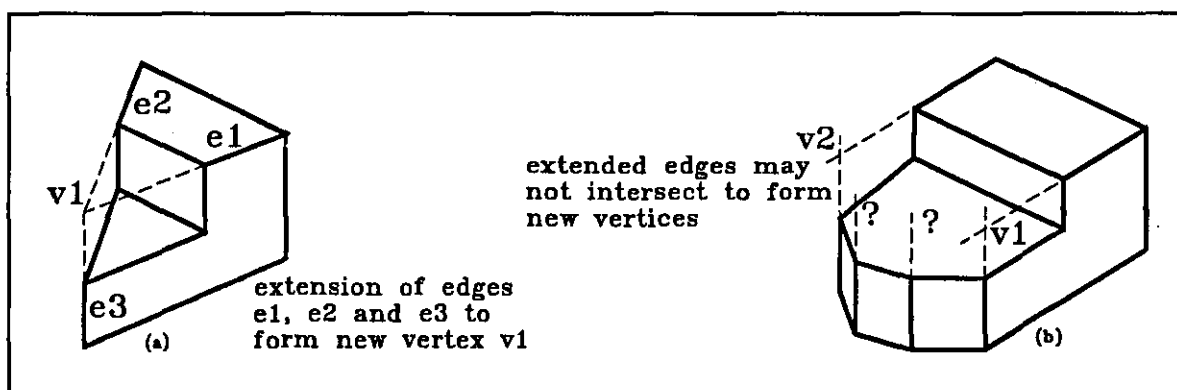


Figure 2.12 : Forming new feature volume by extending edges.

The extracted features are further organized into a hierarchical graph called a Structured Face Adjacency Hypergraph (SFAH) in which the nodes are a FAH representation of recognized features, and the arcs between nodes indicate the parent-

child relationships among features. Only simple planar face objects can be handled. It is unclear whether or not the method can be extended to cover wider part domains.

Floriani [Floriani88, Floriani89] described a B-rep structure called a Symmetric Boundary Graph (SBG) for modelling polyhedral objects. The SBG includes additional topological information such as face-loop and edge-loop relationships. Features such as protrusions/depressions and through holes are recognized and extracted based on inner loop identification and labelling of the connection faces between features. The extracted features are described by a directed graph called an Object Decomposition Graph (ODG). The nodes of an ODG represent the recognized features and the arcs represent the connection faces between features. For instance, a node with a single arc incident on it indicates the presence of a depression or protrusion, and a node with two or more arcs incident on it indicates a through hole, a handle or a bridge. Bruzzone [Bruzzone91] proposed another B-rep graph structure called a Face-Face Connection (FFC) for feature modelling. The FFC model is basically an enhanced version of the SBG and ODG.

Joshi [Joshi88] proposed the use of face-edge graphs called Attribute Adjacency Graphs (AAGs) for feature recognition. The nodes of an AAG represent faces and the arcs between the nodes represent edges. The attributes of the arcs can be 0 or 1 which represent concave and convex edges respectively. There is no smooth edge case as only polygonal features are dealt with. Features are classified into two levels. The higher level corresponds to feature families whose topological characteristics can be described by the characteristics of a AAG subgraph. For instance, a slot consists of three faces with concave edges between the slot base and the slot walls. The lower level stands for a particular feature type within a feature family, and geometric information is used to differentiate between individual feature types. For example, if the angle between the slot walls and the slot base is less than 90 degrees, then the slot is a dovetail slot.

Joshi's feature recognizer first uses a heuristic to dissect the AAG into simpler subgraphs by ignoring all the faces that contain only convex edges. The use of the

heuristic however, eliminates a group of features such as the top faces of islands and bosses. Feature recognition is then performed by matching each subgraph with the feature definitions described above in a fixed order : holes, slots, steps, pockets, blind steps and blind slots. Unsuccessful subgraph matching implies the existence of two types of feature interactions that must occur along a single face of one of the features : (1) features that share common edges, and (2) features that share common faces. For the former type, a heuristic is used to separate the interacting features by further dissecting the AAG sub-graphs at nodes that have more than one convex arc. A feature merge procedure is used to rejoin pairs of features that are dissected unnecessarily. For the latter type, a heuristic is again used to isolate the faces belonging to one feature and to combine the dissected face node pairs of an interacting feature member into a single face node. It is unclear how the artificial boundary of the combined face node pairs is created. Features that do not belong to the above two types are classified as virtual pockets by patching virtual faces to the features to form a closed loop of faces. Joshi's intention is to recognize machining features but the graph-based heuristics have no relationships with machining technology.

Corney and Clark, [Corney91a, Corney91b] described another graph-based feature recognition algorithm which starts by creating a Face-Edge Graph (FEG) within the B-rep model. The algorithm requires the specification of a ray casting vector called the aspect vector  $\vec{a}$  at the outset. The aspect vector corresponds to the orientation of the spindle axis of a milling machine that would be used for machining the recognized features. Based on the relationships between the surface normal and the aspect vector  $\vec{a}$ , the faces in the FEG are classified into three types : (1) vertical faces (v\_faces), (2) parallel faces (p\_faces), and (3) anti-parallel faces (ap\_faces). The v\_faces and p\_faces can be considered as the wall faces and base faces of 2.5D depressions/protrusions respectively, whereas the ap\_faces are assumed to be the remaining faces of the object (Fig. 2.13).

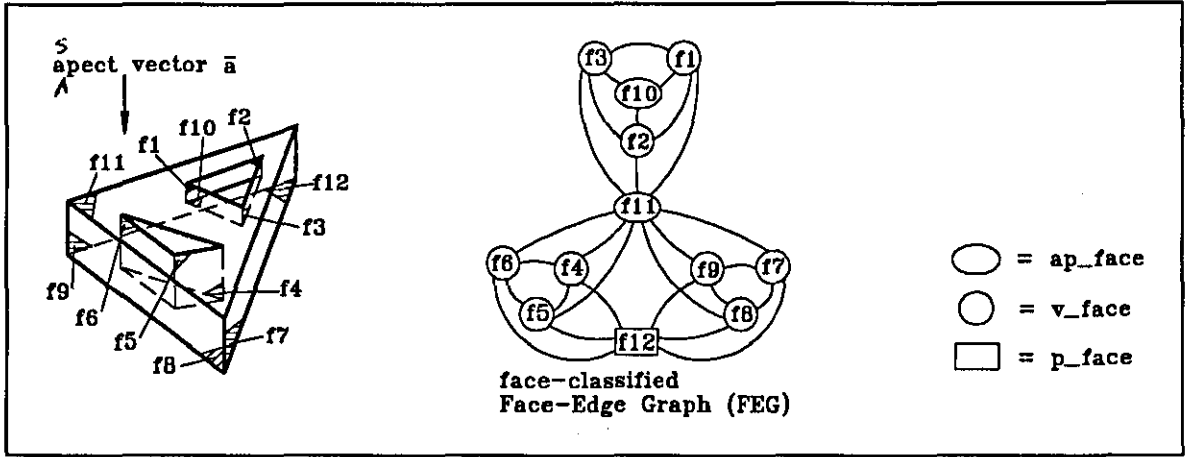


Figure 2.13 : Corney's face-classified Face-Edge Graph.

For exposing the features in the part model, rules are used to delete the p\_faces and ap\_faces from the face-classified FEG so as to generate two subgraphs called the Aspect Face Edge Graph (AFEG) and the UV-Graph (UVG). For instance, the AFEG is generated by deleting all the p\_faces and ap\_faces together with their adjacent edges as illustrated in Fig. 2.14. Both the AFEG and the UVG are then traversed and manipulated to generate some 2D polygon representations (p\_edge polygon) which are basically the virtual image of the vertical wall faces when projected along the aspect vector direction on to an imaginary plane. Each boundary segment of the p\_edge polygon is tagged with a surface normal code (either inward or outward facing) based on the face from which the segment is derived. As a result, a p\_edge polygon can be classified as : (1) inward, (2) outward, and (3) mixed (Fig. 2.14).

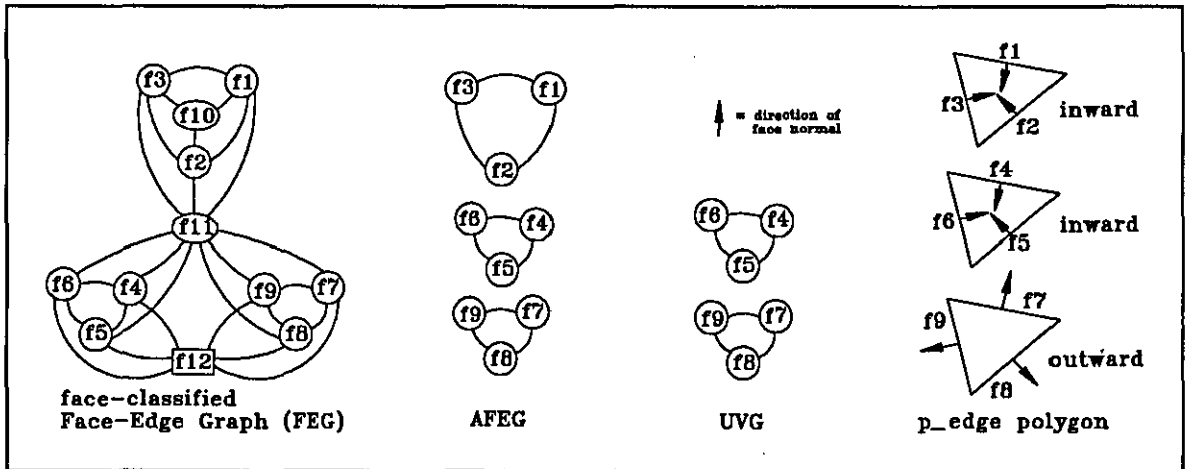


Figure 2.14 : AFEG, UVG, and p\_edge polygon representations derived from FEG.

For the extraction of features, rules are also used to classify the *p*\_edge polygons for representing different types of features. For instance, an inward facing *p*\_edge polygon of the AFEG represents a depression. By cross referencing the contents of the *p*\_edge polygons of the AFEG and UVG, more specific features can be inferred. For example, if a *p*\_edge polygon of the AFEG is classified as a depression but is not classified as a hole in the UVG, then the corresponding polygon or cycle would represent a pocket. The method also determines nested depressions by means of ordering the 'depth' of the depressions along the aspect vector direction.

Corney's method implicitly uses a machining heuristic in terms of specifying the aspect vector. Automatic determination of the many possible aspect vectors for a given part may be difficult although the authors suggested that a heuristic based on the presence of cylindrical faces can be used to ease the choice. The recognized features are basically arbitrarily shaped polyheral<sup>d</sup> pockets, protrusions, and through holes. As the pseudo-edges do not need to be the actual edges that lie on the object, the method of pseudo-edge polygon could be useful for dealing with feature interaction that usually cause boundary fragmentation.

### 2.3 Concluding Remarks

The design by features approach is attractive from the viewpoint of concurrent engineering. However, feature recognition is still required in the approach due to the fact that the interpretation of features is application dependent. Thus the study of automatic feature recognition techniques will continue to be a crucial research activity in the context of CAD/CAM integration.

Automatic feature recognition with CSG models has frequently been hampered by : (1) the non-uniqueness of CSG representations, (2) the implicit description of boundary information, and (3) the possible destruction of a feature by subsequent

Boolean operations, and hence searching for CSG patterns tends to be an unattractive approach to feature recognition.

One of the major difficulties found in most recognition methods with B-rep models is in dealing with complex shapes. To a large extent, this is due to the use of rigid feature definitions that rely heavily on face/edge topological information. When realistic parts and feature interactions occur, this face/edge adjacency information may not present in exactly the same manner as predefined in the feature templates, and hence recognition may not be reliable. Trying to fix the problems afterwards will be difficult if not impossible because many algorithms are implemented procedurally and also because many types of feature interactions can occur in real life parts.

Since feature interpretation is application dependent, recognition algorithms which rely solely on form reasoning without the ingredient of application specific information will inevitably face difficulty in handling complex feature interactions. For design by features systems, much of the application specific information can be obtained from the features that are instanced in the design. In a machining features recognition context however, application specific information such as machining heuristics and tool accessibility can be exploited and combined with form reasoning for developing more intelligent recognition algorithms.

It is believed that the implementation of such a recognition algorithm can be facilitated by using an integrated knowledge-based system and geometric modeller approach. The use of a knowledge-based system allows a clear demarcation between the feature (problem) description and the feature recognition (problem solving) procedures, and hence makes the implementation and subsequent maintenance of the system much easier. The tasks of searching and matching patterns involved in the feature recognition process can also be simplified by utilizing the inherent pattern matching mechanism of the knowledge-based system. Coupling a geometric modeller with a knowledge-based system enables efficient sharing of part modelling and reasoning information among the two environments. The geometric computation

routines available in the geometric modeller are also ideal tools for implementing the geometric tests.

To further enhance the capability and viability of a feature recognition system, the recognition power should be allowed to increase easily and perpetually. In this connection, the traditional human-assisted feature definition approach can be improved in such a way that the defined features are used as feature matching templates (or as generic features in a feature based design context). Although work [Dong88b, Sakurai88, Shah90] using this approach have been reported in recent years, the formal techniques for implementing the idea still have not been adequately exploited. A better scenario for the approach would be that a feature definition process based on interactive machine learning of an example feature under reasonable machine guidance.

Before embarking on the description of the core of the thesis, an overview of AI techniques concerning knowledge-based system and machine learning techniques is presented in the following chapter.

## CHAPTER 3

# ARTIFICIAL INTELLIGENCE TECHNIQUES

AI, as an important subfield of computer science, has the objective of studying and developing computer methods that solve problems in a way that would be considered intelligent if performed by a human. As AI scientists attempted to simulate the human thinking process, it was discovered that problem-solving techniques can be quite general for a wide range of problem domains (areas of expertise). It was also realized that the problem-solving power of a program mainly comes from the domain specific knowledge it possesses. Hence, much effort has been focused on developing generalized knowledge representations and search techniques for specialized computer programs. This has resulted in the development of knowledge-based systems.

### 3.1 Knowledge-Based Systems (KBS)

A KBS is a computer program that uses domain specific knowledge and inference procedures to solve problems that are not amenable to procedural analysis and with incomplete information. The problem-solving knowledge is represented in an identifiable, separate part of the system rather than being dispersed throughout it. As illustrated in Fig. 3.1, the basic structure of a KBS consists of :

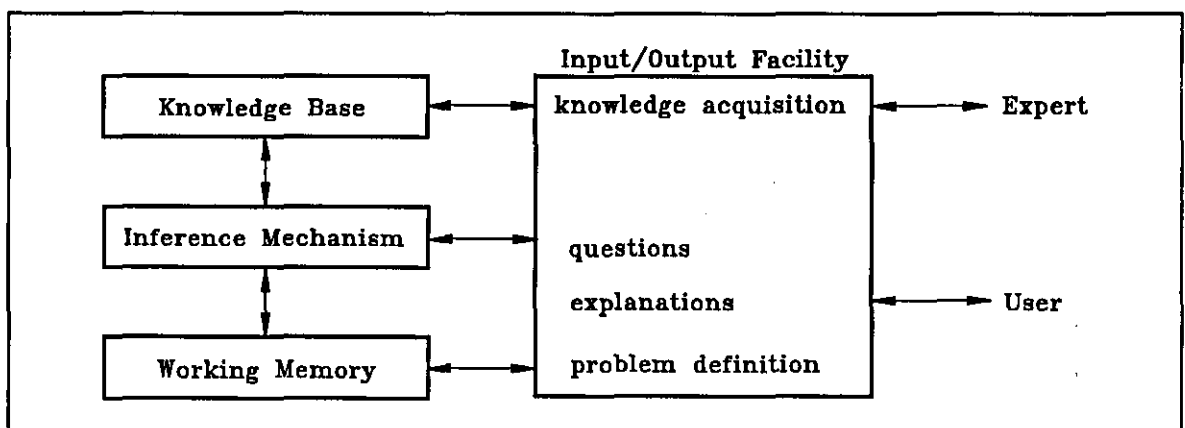


Figure 3.1 : Major elements of a Knowledge-based system.



**(1) an Input/Output facility** - The user can use this facility to : (i) input the facts about the problem to the working memory of the KBS, (ii) edit the problem model in the working memory, (iii) execute, control, and monitor the problem-solving process, and (iv) receive advice, explanations, and solutions from the KBS. If the KBS has a knowledge acquisition module, the facility will also be used to manage the knowledge acquisition process.

**(2) a Working Memory** - This serves as a global database containing the input descriptions of the problem to be solved. A subpart of the working memory is called the State Memory which stores a sequence of snapshots of the problem solving environment in the form of a record of the facts and rules that have been modified and applied.

**(3) a Knowledge Base** - This contains the knowledge specific to the problem domain of concern. Knowledge consists of symbolic descriptions about the factual relationships (assertions) and empirical relationships (heuristics) of a problem domain, as well as the procedures for manipulating those descriptions. For more intelligent KBS, this also includes a knowledge acquisition facility which enables the KBS to elicit additional knowledge about the problem domain from experts or other sources.

**(4) an Inference Mechanism** - This is also called a rule interpreter in a rule-based KBS. Basically, it utilizes the knowledge in the knowledge base to analyze the problem model described in the context, makes decisions, and draws logical conclusions. During the inference process, the problem model in the context is usually updated and a record of execution steps is produced so as to facilitate the provision of advice and explanations to user.

The major difference between a KBS and a conventional computer program is in the implementation of problem-solution logic. As shown in Fig. 3.2, the problem-solution logic in the conventional programming approach is implemented as rigid sequential procedures which consist of the user's predetermined problem-solution logic

intermingled with the computer control logic. Thus a change made in the problem may incur hectic re-analysis and costly re-coding of the program. In KBS however, the problem-solution logic is implemented usually as decision rules stored in the knowledge base. The coding sequence of a procedural program governs the proper execution of the program, whereas the coding sequence of decision rules in the knowledge base does not affect the execution of a KBS because a decision rule will be executed only when its conditions or conclusions are satisfied.

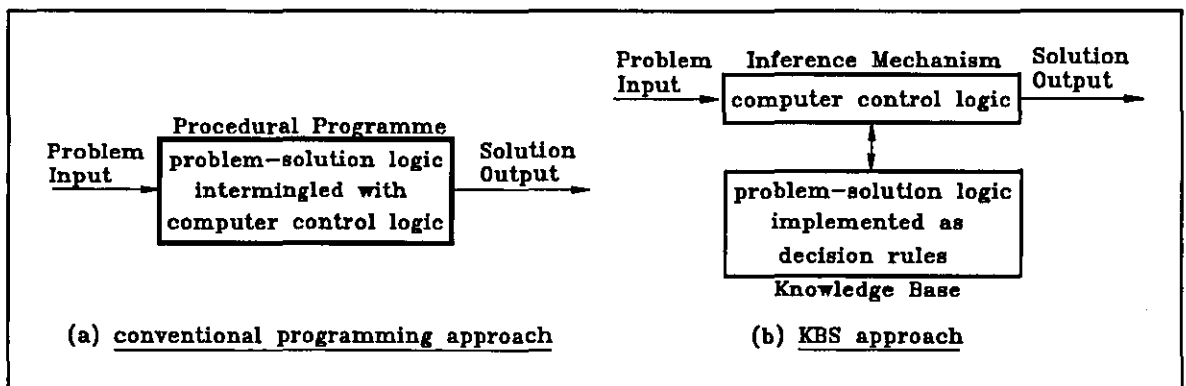


Figure 3.2 : Difference between conventional programming and KBS approaches.

Thus the conventional programming approach is ideal when the problem-solution procedure is of a stable and repetitive nature such as the determination of the convexity/concavity of all the edges in a B-rep model. The KBS approach would appear to be more attractive when it is difficult to predetermine a precise series of steps for solving a complex and empirical problem. The recognition of complex manufacturing features is such a problem.

### 3.2 Problem-Solving Techniques

The representation and retrieval of information are the two major tasks in problem solving. In the terminology of AI, these two tasks are known as knowledge representation and search techniques respectively. These two major components of problem solving are overviewed in the following sections as an aid to a better understanding of the methods employed in this thesis.

### 3.2.1 Knowledge Representation

A knowledge representation is a set of syntactic and semantic conventions for describing objects, relations and procedures of the appropriate knowledge. The four most popular schemes in use today are : (1) production rules, (2) frames, (3) semantic nets, and (4) object orientation.

#### 3.2.1.1 Production Rules

Production rules [Newell72], also called if-then, condition-action, or antecedent-consequent rules, are a natural way of expressing heuristics or procedural knowledge because they utilize the simple **IF condition THEN action** format. For example,

```

IF           the goal is to infer a face's accessibility
              and  the face is a machined face
              and  the face has a high priority ranking
THEN        perform accessibility test on that face
              and  modify the face's accessibility status according to the test result.

```

The **IF** part, or left-hand side (LHS), of a rule represents a condition that contains one or more clauses linked by logical connectives (AND, OR, etc.), whereas the **THEN** part, or right-hand side (RHS), of the rule specifies the corresponding consequence or action to be taken when the LHS pattern of the rule is satisfied. Depending on the system implementation, the RHS can take many forms, such as an interaction with the user, modification of an assertion in the working memory, addition of a new rule in the knowledge base, etc.. Certainty factors can also be used in a rule to indicate the degree of confidence attached to it. This enables a KBS to deal with information which is inexact or not completely reliable.

The mechanism for the selection and execution of rules is essentially the inference mechanism which can be implemented based on two basic strategies : (1) forward-chaining, and (2) backward-chaining. In forward-chaining, the contents of the working memory represent the current state of the problem. A rule is fired when its LHS pattern matches with the problem model data contained in the working memory. Since the inference chain progresses from the given data to a goal, so it is also known as data-driven or antecedent reasoning. By contrast, a backward-chaining strategy requires the setting up a set of hypotheses or goal data in the working memory and followed by firing of the rule whose RHS pattern matches with the goal. The LHS pattern of the fired rule is then added to the working memory as a new subgoal. This strategy is therefore also called goal-driven or consequent reasoning.

In general, a forward-chaining inference strategy is suitable when there is a single initial state and many equally acceptable goal states, while backward-chaining is appropriate for tasks that have a single goal state and considerable amount of relevant initial information. However, the distinction between the two inference strategies is not absolute. Many rule-based KBS systems use a mixed strategy that combines forward and backward reasoning. No matter which strategy is used, the basic cycling function of the inference mechanism is to recognize and act on a rule, and hence the inference mechanism is also called the recognize-act cycle.

Unlike the IF statements of conventional programming languages such as FORTRAN and PASCAL, the production rules are not executed in a predetermined, sequential order, and the flow of control is not limited to branching only at pre-coded points. The rules behave much like independent pieces of knowledge since they do not call each other directly but communicate only by means of the data in the working memory. This relative modularity and uniformity of rules enables easy addition and deletion of rules in the rule base, and hence rapidly changing conditions can be better accommodated. Moreover, production rules also provide a parallel reasoning capability because the inference mechanism can cycle back automatically to find all the satisfied rules. Nevertheless, production rules have some disadvantages. For instance, the

modularity and uniformity of rules make it difficult to visualize and follow through the flow of control logic in complicated problem solving. The recognize-act cycle mechanism is also not efficiently responsive to predetermined sequences of situations. Fortunately, these drawbacks can be remedied by incorporating external conventional programs in the forms of function calls and subroutines in the production rules. This hybrid-language programming technique is also employed in the research and is described in more detail in chapter 7.

### 3.2.1.2 Semantic Nets

A semantic net [Quillian68] is a form of associational representation that is composed of nodes, interconnected by various kinds of associative links. Each node represents an individual object and facts about the object. Each link explicitly expresses a relationship between a pair of objects. Fig. 3.3 illustrates the idea of a semantic net.

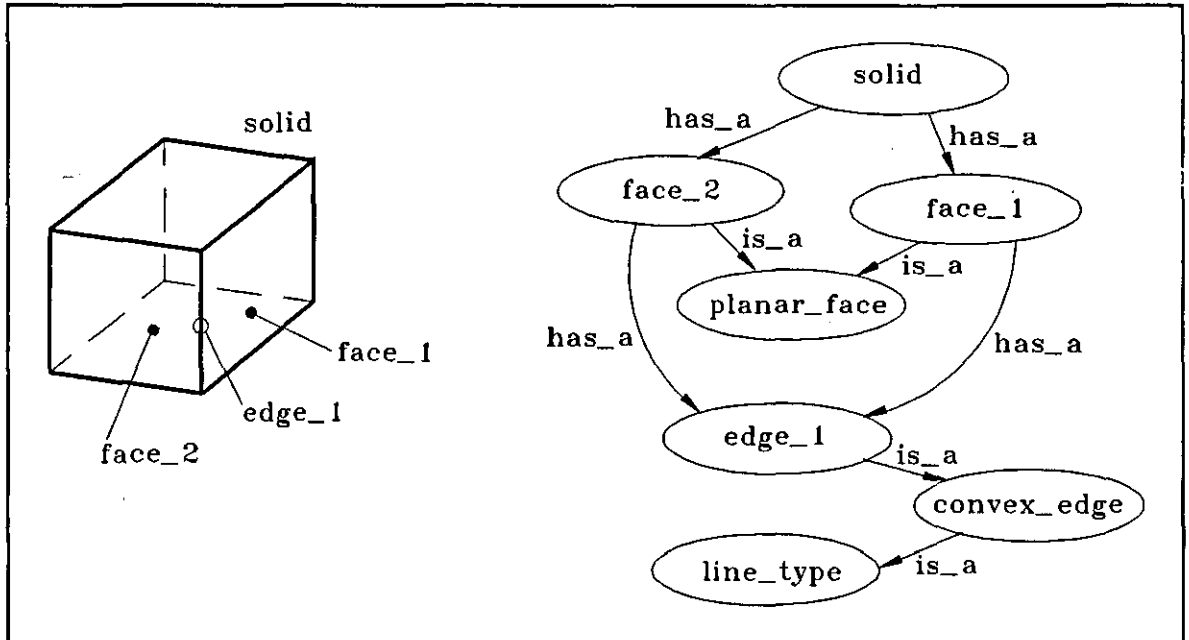


Figure 3.3 : Semantic network representation.

The reasoning mechanism used by most semantic network systems is based on matching network structures. For example, as described earlier in Dong's work [Dong88a], a network fragment representing a matching feature pattern template is constructed and is then matched against the network database to see if such a pattern exists. If the matching process is successful, variable nodes in the fragment can be determined by bounding them to appropriate values in the network. The graph-based feature recognition methods used by Joshi and Floriani [Joshi88, Floriani88], can also be considered as a variant of the semantic network based approach.

### 3.2.1.3 Frames

A frame [Minsky75] is a structured representation consisting of a set of standard characteristics that describe an object, act or event. It is rather similar to the record-like structure constructed in conventional programming languages. The characteristics in a frame are denoted in terms of attributes called slots. The contents of the slots can be either actual values or procedures for obtaining the desired values. The organization of a frame is very much like a semantic network that has a set of nodes and relations arranged in a hierarchical form. As an illustration, an instance of a frame representing an object's face is shown in Fig. 3.4. A frame-based representation is suitable for applications which are predictable because it supports the notion of standard stereotypes.

Frame : Face			
	<u>Slot Name</u>	<u>Contents</u>	<u>(comments)</u>
Slot 1 :	Name	1280	face integer pointer
Slot 2 :	Geometry Type	PLN	planar face
Slot 3 :	Surface Normal Code	1	positive normal
Slot 4 :	Classification	TEFACE	tool entrance face
Slot 5 :	No. of Bounding Edges	8	
Slot 6 :	Status Flag	NIL	

Figure 3.4 : Frame representation.

### 3.2.1.4 Object Orientation

This method of representing knowledge [Dahl73] is sometimes referred to as object oriented programming since the knowledge representation and knowledge manipulation procedures are programmed as a complete package. Objects, in the object-oriented paradigm, are entities that combine the properties of data and procedure. For example, the frame structure introduced above can be used to represent an object, but the frame slot must be able to contain working procedures which can communicate messages with other objects or frames. In most object-oriented languages such as Smalltalk80 [Goldberg83], objects are organized in a hierarchy of classes and instances. A class is a description of one or more similar objects. An instance is a manifestation of a class in the form of an object. Both classes and instances have a declarative structure that is defined in terms of object variables for storing states and methods or procedures for responding to messages. For example, the form feature taxonomy proposed by Gindy [Gindy89] can be implemented by using object orientation method as shown in Fig. 3.5.

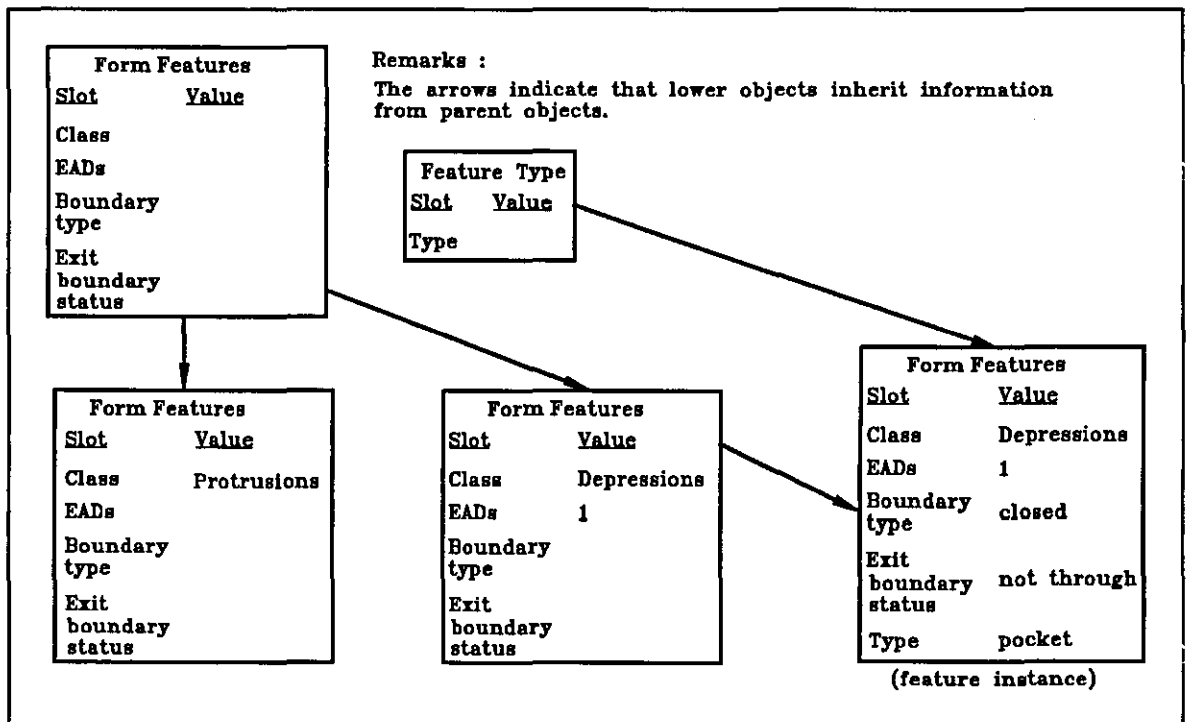


Figure 3.5 : An illustration of object orientation.

Object orientation has the profound advantage that classification and inheritance of attributes makes the maintenance of the system knowledge domain much easier. However, it requires a top-down design approach. Sometimes it is not clear on which level object attributes should be defined. This decision is usually a trade off between redundant specification of attributes at the lower levels and generic attributes which have no meaning at the level on which they are specified.

### **3.2.2 Search Techniques**

Simple search techniques such as the generate-and-test method can be used for trivial problems. For a complex problem, where the solution space is large or the number of alternative solutions are numerous, more sophisticated heuristic search techniques are essential. A heuristic search improves the efficiency of a searching process by guiding the search in fruitful directions, possibly at the price of failing to find the complete solution. However, good heuristics are just like golden rules of thumb; on the average, they do improve the quality of the search and provide good solutions to hard problems. Various search techniques [Nilsson71] have been developed and employed by AI scientists and practitioners. The following sections introduce three of the common ones.

#### **3.2.2.1 State-Space Approach**

In this popular approach, a problem is formulated with problem states, a set of operators, a search control procedure, and the desired goal state. A problem state is a description of a problem situation at a particular instance. An operator is a set of rules or computations which transforms the problem from one state to another state. The state space of a problem is conceived as all the possible states that can be reached from a given starting state via a series of transformations. A solution to this type of problem is obtained by following the search control procedure to successively apply the



operators to the starting state to produce new states until the generated new state is equal to the goal state. The two systematic search control procedures commonly used are : (1) breadth-first search, where all the search paths on the same level of the hierarchy are searched before examining any of the successor paths on the next lower level, and (2) depth-first search, where one path on the highest level is searched and then the successor paths immediately below that one are examined. Clearly, the efficiency of these two search techniques is affected by the position of the solutions in the search path hierarchy. For solving large state-space problems, a heuristic search that employs heuristic rules to determine which path should be searched next can improve the searching efficiency.

#### **3.2.2.2 Hill Climbing Approach**

This approach can be considered as an enhanced variant of the commonly known generate-and-test approach that simply generates a possible search path and then tests to see if the endpoint state of the path is actually a goal state, and so forth. The improvement is in the test process which uses a heuristic to evaluate an estimate of how close a generated state is to a goal state. The working principle of the heuristic is that if a generated state is not a goal state but is better than the current state, then the generated state will be used as the current state in the search, otherwise another new state will be generated and similarly tested. Thus the use of a heuristic-based test effectively injects application-specific knowledge into the search process.

#### **3.2.2.3 Constraint Satisfaction Approach**

In this approach, the goal of solving a problem is to determine some problem state that satisfies a defined set of constraints. It consists of two major steps : (1) constraints are determined and propagated as far as possible throughout the system, and (2) if there is still not a solution, a heuristic search begins to generate new constraints

which can again be proliferated in the system, and so forth. By viewing a problem as one of constraint satisfaction, it is possible to reduce the amount of search substantially. As a simple example, if a problem starts with the state,  $A = B + 2$ , and the constraint,  $A = 6$ , was known, a stronger constraint on  $B$  could be propagated as  $B = 4$ .

### 3.3 Machine Learning

Learning [Simon83] is a general term denoting the process of improving the long-term performance of a system. Machine learning is a subdomain of AI concerned with developing computational theories of learning and constructing computer programs with learning capabilities. By adding a learning mechanism to a computer system, the system developer expects that the user can extend the system's problem-solving capabilities through interaction with it, rather than by the process of reprogramming. Thus it would be an important problem-solving approach when the possible situations that a system will encounter are not known in advance.

Four basic learning strategies have been identified by AI researchers [Cohen83]: (1) learning by rote, (2) learning from instruction, (3) learning from examples, and (4) learning by analogy. These are introduced below.

#### 3.3.1 Learning by Rote

This is the basic learning activity by which information provided by the environment is stored and later on retrieved for use without much hypothesis or computation. The intelligent chess program developed by Samuel [Samuel63] is a typical example of this strategy. The program learns to play well by memorizing and recalling chess board positions that had been encountered in previous games.

### 3.3.2 Learning from Instruction

This refers to the process of transforming given general-purpose knowledge into a performing program. The transformation is called operationalization which can involve activities such as hypothesizing the missing details in the given information and deciding when to ask for more instruction. Mostow's program FOO (First Operational Operationalizer) [Mostow83] is one of the results of this strategy. The program makes use of a card playing game to investigate and demonstrate the principles, problems, and methods involved in converting general card playing advice into executable procedures.

### 3.3.3 Learning from Examples

This involves teaching a system how to perform (or how not to perform) a task by presenting it with a set of training examples. Training examples are usually very specific instances or detailed knowledge of behaviour that cannot be used efficiently by the system. Hence, the system needs to generalize the training examples into more general pieces of knowledge that can be used effectively. An important example of this learning strategy is Winston's [Winston75] work on learning simple structural concept descriptions that characterize some positive toy-block constructions and the corresponding near-miss cases (counter-examples) as shown in Fig. 3.6.

The original line drawing representation of the toy-block assemblies is converted into a semantic network description as illustrated in Fig. 3.7(a). His basic approach is to use a structural description of one known instance as a concept definition for examining other legal instances of the concept. The original structural definition is then generalized to include them as shown in Fig. 3.7(b). When descriptions of near-miss examples are given, the structural definition will be specialized to exclude them as shown in Fig. 3.7(c).

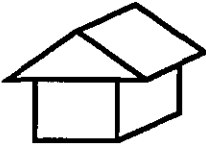
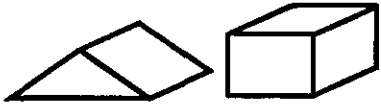

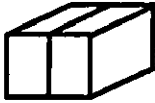
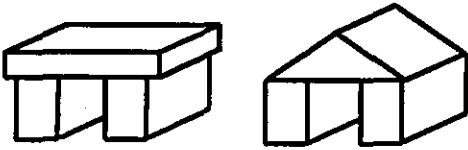
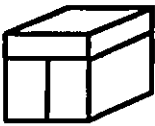
	Concept	Near Miss
House		
Tent		
Arch		

Figure 3.6 : Winstons's toy-block constructions and the corresponding near-miss cases.

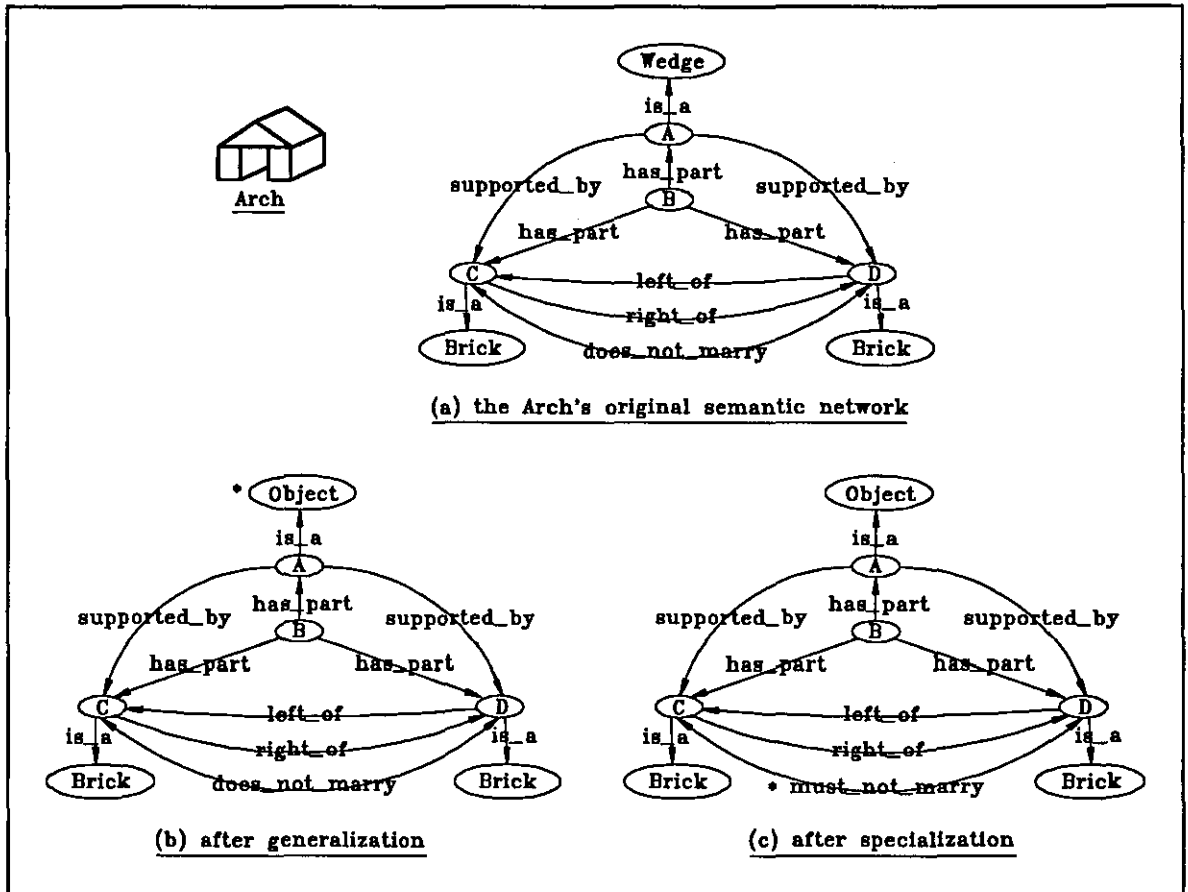


Figure 3.7 : Learning by generalization and specialization of concept.

### 3.3.4 Learning by Analogy

This strategy requires the learning system to detect the similarities between the old and the new situations, and to transform the old knowledge into analogous knowledge that can be used in the new situation. As an example, a transformational analogy presented by Anderson [Anderson79] is illustrated in Fig. 3.8. However, very little work has been done in this area since many issues such as the exact definition of analogy and the formal methods of recognizing analogies are still not well understood.

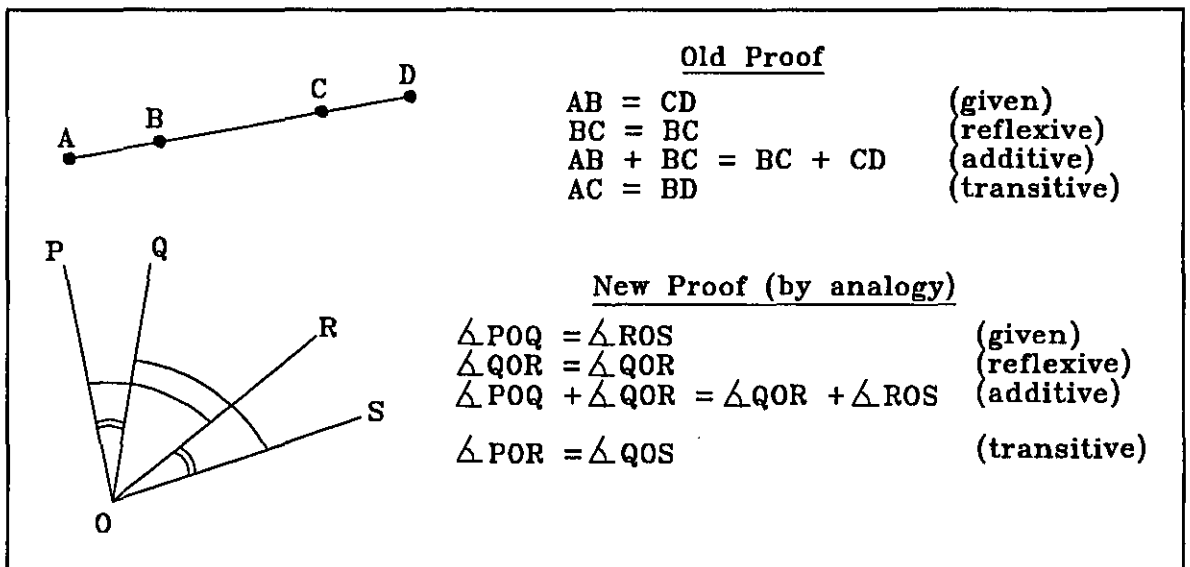


Figure 3.8 : Anderson's learning by transformational analogy.

### 3.4 AI-based Manufacturing Researches

The rapid advent of AI technology has enabled computers to be applied to less deterministic design tasks which require symbolic manipulation and reasoning, instead of only routine number processing. Design is now considered as a knowledge-based intelligent behaviour. Research interests are concentrated on how design knowledge is acquired, represented, organized, used, and generated. The aim is to develop an intelligent CAD system that can assist designers during the conceptual stage of design as well as detail design, and the design model established in the computer should be

able to provide qualitative and quantitative feature information that represents design intent and manufacturing purposes. Thus the current researches on AI-based design are relying on the principles of feature technology and simultaneous engineering. The literature on AI-based design research is voluminous. Some has been reviewed in the previous chapter in the section on feature modelling methods. Interested readers may like to refer to further references such as [Smithers89, Ishii89, Akagi91, Kroll91].

In the context of manufacturing resource planning and control, AI techniques have been exploited in diverse investigations such as manufacturing system layout design [Kusiak91], machine scheduling [Bullers80], process/machine diagnostics [Crawford87], robot-based sensory control [Simon88], etc..

As far as process planning is concerned, almost all the contemporary research systems use AI techniques. This is not surprising due to the fact that process planning is a knowledge intensive activity. A few representative works are briefly described below so as to obtain a general understanding of the techniques used and the level of achievement that has been accomplished.

Davies [Davies84] reported the implementation of a prototype system called EXCAP for process planning of rotational parts. Instead of conventional production rules, EXCAP uses fuzzy rules that define the extent to which the planning decision or hypothesis is justified when the certainty in the condition part of the rule is known. The component and blank are defined in terms of an ordered sequence of dimensioned features, such as face, cylinder or taper. A tree of possible operation sequences is formed. Nodes in the tree represent various intermediate workpiece states; the root node represents the finished part, and the terminal node represents the blank. The arcs linking the nodes represent the operations used. Rules set at a certainty value are attached to each operation arc for deciding the suitability for using operation. Planning is performed as the inverse of machining, and so works backwards from the finished part towards the blank state.

Berenji [Berenji86] used a general purpose rule-based expert system to develop an AI-based process planning system called Hi-Mapp, where a prismatic part is described by a set of form features such as notches, grooves, slots, and holes. The initial state of the expert system consists of information about the geometric description of the part in addition to the characteristics of the available machines, tools, and materials. The goal state consists of a partial ordering of the features to be processed. Process planning is viewed as the transformation from the initial state to the goal state. The transformation process is effected by firing the appropriate production rules stored in the knowledge base. The condition part of the production rules basically contains the feature type and additional characteristics such as surface finish, while the action part stores the recommended actions such as selection of process, machine, and operation.

More recently, Murray [Murray89] described the direct coupling of a knowledge-based system with a formerly developed automatic machining program called AMP, for enhancing the planning and part-programming process. Originally, AMP used the FORTRAN language to code six modules of machining knowledge about method of holding, blank size, profile division, bolt positioning, and dowelling. This machining knowledge was reformulated as a set of Prolog rules. The knowledge-based system can directly interrogate the model so that the effect of aspects of the geometry, such as thin walls and detachable waste, can be assessed. Another knowledge-based system to aid assembly design was also developed. The knowledge base was constructed as a sequence of frame-like modules. For example, the part modules define the attributes such as type and geometry associated with an assembly component, while the relation modules define inter-component relationship such as assembly fitting conditions. A set of assembly design rules are stored in the rule base to assist the designer to produce a complete design description.

From the above, it can be appreciated that nearly all knowledge-based CAPP systems use rules as a means of formulating the required knowledge. This is quite natural since many of the process planning decisions involve the use of alternative rules that can be empirical in nature. After all, a rule-based expert system is still one of the

most sophisticated computational methodologies that computer scientists can offer today.

However, despite a series of activities on advanced knowledge-based CAPP, there still exists some limitations in its capability and potential. These limitations are due to two main reasons : (1) a computer representation scheme that can provide both quantitative and qualitative information of a component is still not available, and (2) the process planning knowledge as well as its computer representation are still not well understood. The first obstacle has been discussed in length in the previous chapter, and is the focus of this research. Consideration of the second obstacle is not the main interest of this thesis although a simple machining planning and NC tool path generation program is also developed in this thesis as a proof-of-concept to validate the extracted feature information. Nevertheless, many other researchers [e.g. Houten90, Iwata90] are working earnestly on the process planning knowledge aspect. Until these two major obstacles are removed, a real knowledge-based CAPP system will not be established, and consequently, the goal of CIM will not be fully realized.

### **3.5 Concluding Remarks**

AI is actually a problem-solving methodology that can be applied to many fields such as design, engineering and management [Harmon88]. The KBS's architecture is very useful for manipulating the disparate information and knowledge elements typical of a CIM environment. Its capability of separating the problem-solution logic as knowledge from the computer control logic also facilitates the gradual addition of new knowledge to the system.

In this research, a significant step is made towards machining feature recognition by using a KBS approach which is supported by some of the general problem-solving and machine learning techniques introduced in this chapter. The approach is explained in detail in subsequent chapters.



## CHAPTER 4

# MACHINING FEATURE DEFINITION AND REPRESENTATION

### 4.1 Parts Domain

This thesis is primarily concerned with the recognition of machining features in non-rotational mechanical parts that are typically manufactured on 3-axis machining centres<sup>1</sup>. These parts are usually referred to as 2.5D solids<sup>2</sup>. The finished parts as well as their corresponding starting stocks are modelled by using the CSG method with solid primitives that are bounded by planar and cylindrical half-spaces. In other words, the stocks and parts are assumed to contain only planar and cylindrical faces which are subsets of smooth and mathematically perfect surfaces without any surface irregularities such as machining cusps and tool dwell marks. This represents an important geometric domain as it can describe a wide range of shapes [Samuel76, Yuen88] produced by the 3-axis machining centres.

---

<sup>1</sup>The generic term 3-axis machines includes : (1) the 2CL (2-axis Contouring and 1-axis Linear speed control) machines which are sometimes called the 2.5-axis machines, and (2) the 3C (3-axis Contouring) machines [BS3635]. A machine with 2CL kinematic capability controls the motions along the two orthogonal driving axes of the machine table simultaneously so that a 2D contour can be cut on a part. For a 3C machine, the motions along the three primary axes are simultaneously controlled so that a 'true' 3D surface can be cut. In this thesis, the parts are assumed to be manufactured on machines with 2CL kinematic capability.

<sup>2</sup>Solids such as the one shown in Fig. 4.1 are called 2.5D solids because the surface normals of their side faces are free to change two-dimensionally in the x-y plane. The surface normals however, remain unchanged in the y-z plane that is orthogonal to the former. Hence, 2.5D solids are sometimes called prismatic (column-like) or linearly sweepable solids.

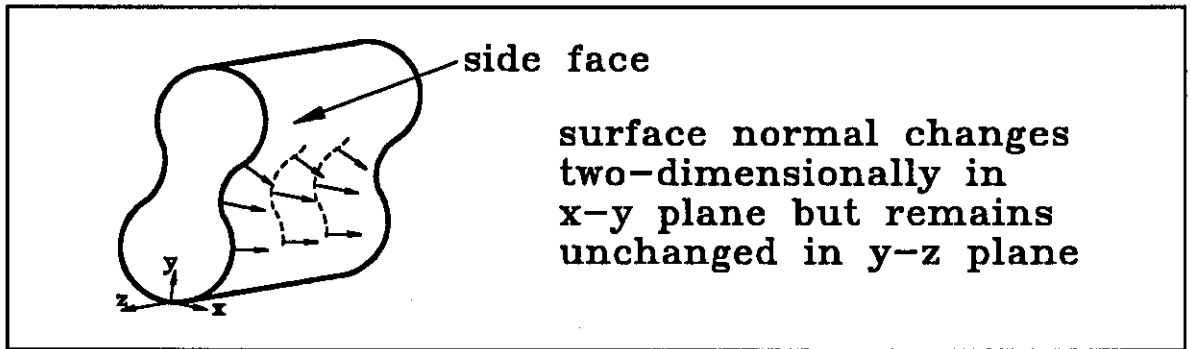


Figure 4.1 : Definition of a strictly 2.5D object.

## 4.2 The Cavity Volume Model

The recognition of machining features in this thesis is performed on the cavity volume model rather than on the part model. More formally, the modelling of a cavity volume is based on the notion that a machined part  $P$  is produced by removing a certain amount of material from a stock  $S$ . The necessary conditions that  $P$  must not be  $S$  and the volume of  $P$  must be smaller than the volume of  $S$  can be expressed as :

$$P \subset S \quad (1)$$

where  $\subset$  means 'is a proper subset of'.

The cavity volume model  $V$  is the total volume of material machined from  $S$  to produce  $P$ , which can be expressed as :

$$V = S \langle - \rangle P \quad (2)$$

where  $\langle - \rangle$  is the regularized Boolean subtraction operator.

When  $P$  contains  $n$  isolated machining features such as a pattern of holes lying on a pitch circle diameter,  $V$  will consist of the corresponding  $n$  disjoint subvolumes. This condition can be expressed as :

$$V = \sum_{i=1}^n v_i \quad (3)$$

where  $n$  is the number of subvolumes, and

$v_i$  is a subvolume instance.

There are four main reasons for using this cavity volume approach. Firstly, it is considered that recognizing machining features in a part model in the absence of the corresponding stock information is actually based on the assumption that the stock shape is a minimum convex envelope of the part, and thus all holes and cavities contained in the part will be recognized as machining features. Obviously, this is undesirable as a starting base stock can contain some depression features that were produced by previous manufacturing operations. Secondly, without the original stock information it is also difficult to recognize features such as surface milling and profile milling. Thirdly, feature recognition can be performed on the subvolumes one at a time, and the boundary database of a subvolume would be simpler than that of the corresponding part. Fourthly, obtaining the cavity volume model by means of subtraction is congenial with the perception of machining process. The cavity volume model provides a complete boundary description of the machining features which are actually present in the part. Clearly, this is important for reliable feature recognition and process planning. Although the derivation of the cavity volume requires boundary evaluation, it is not a major shortcoming because the development of more efficient boundary evaluation algorithms [Tilove84] and more powerful computing hardware has significantly lowered the computational cost of boundary evaluation.

### 4.3 The Boundary of the Cavity Volume

The surface boundary of the cavity volume  $V$  can be simplified and expressed formally as :

$$bV = b(S \langle - \rangle P) = (bS \wedge cP) \vee (iS \wedge bP) \quad (4)$$

where  $b$  is a boundary operator,  
 $\langle - \rangle$  is a regularized Boolean subtraction operator,  
 $\wedge$  is an intersection operator,  
 $c$  is a complement operator,  
 $\vee$  is a union operator, and  
 $i$  is an interior operator.

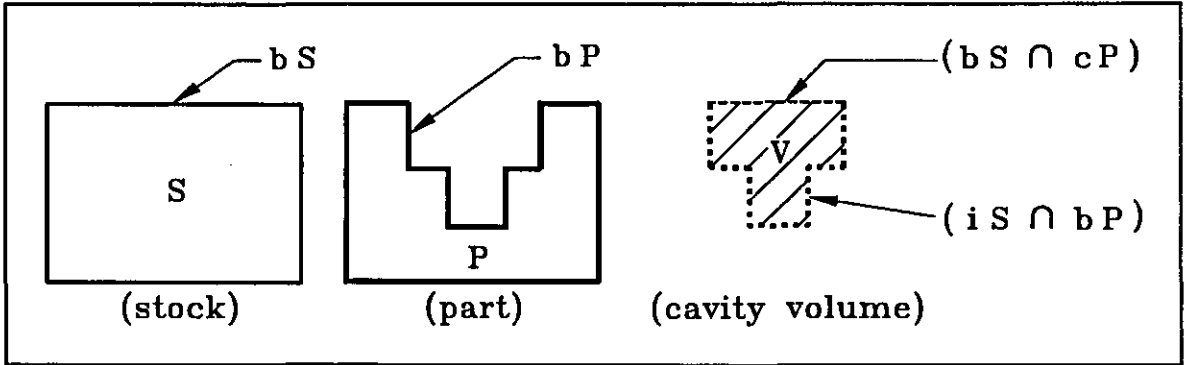


Figure 4.2 : The notion of the cavity volume boundary as described by expression (4).

Expression (4) is derived based on the regularized point-set theory. The derivation is explained and illustrated in Appendix A1, while the meaning of the expression is illustrated in Fig. 4.2. Intuitively, the first term,  $(bS \wedge cP)$ , refers to the point set formed by the intersection between the boundary point set of  $S$  and the complement (outside) point set of  $P$ . The second term,  $(iS \wedge bP)$ , refers to the point set formed by the intersection between the interior point set of  $S$  and the boundary point set of  $P$ .

Considering expression (4), since  $P \subseteq S$ , hence the term  $(bS \wedge cP) \neq 0$ . This term represents the portion of the stock boundary that is outside the part. From the machining viewpoint, it is the portion of the stock boundary through which a cutting tool can pass through without gouging the part. Hence, this term is defined as the tool entrance boundary, and a bounded region of this boundary is called a **tool\_entrance\_face**.

Again, since  $P \subseteq S$ , so the term  $(iS \wedge bP) \neq 0$ . This term represents the portion of the part boundary that is inside the stock. It can therefore be considered as the portion of the part boundary that is created due to the removal of material from the stock. So this term  $(iS \wedge bP)$  is defined as the machined boundary, and a bounded region of this boundary is called a **machined\_face**.

Furthermore, as described in expression (3),  $V$  may consist of several disjoint subvolumes. In the context of topology, the subvolumes are the shells of  $V$ . Hence, the boundary of  $V$  is equal to the sum of the boundary of all the subvolumes. This can be expressed as :

$$bV = \sum_{i=1}^n bv_i \quad (5)$$

where  $n$  is the number of subvolumes.

It follows that a cavity volume or each of its subvolume can be defined as a solid bounded by a set of `tool_entrance_faces` and `machined_faces`, which can be expressed as :

$$bV = \sum_{i=1}^m (\text{tool\_entrance\_face}_i) + \sum_{j=1}^n (\text{machined\_face}_j) \quad (6)$$

where  $m$  is the number of `tool_entrance_faces`, and  
 $n$  is the number of `machined_faces`.

The `tool_entrance_faces` and `machined_faces` of a cavity volume are like the doors and walls of a room respectively. This means that a cutting tool can access a cavity volume only through the `tool_entrance_faces`. Also if a portion of a cutting tool has already entered the cavity volume, that portion of cutting tool should not go beyond the `machined_faces`, otherwise gouging of the part will occur.

As the part is obtained by subtracting the cavity volume from the stock, the `machined_faces` of the cavity volume are basically the 'reverse image' of the `machined_faces` of the part. This means that for every `machined_facei` of the cavity volume there is a corresponding `machined_facej` of the part such that the two `machined_faces` have identical edge loop (edge boundary) and reversed surface normals. Using the half-space concept, this means that the half-space of `machined_facei` is the complement of the half-space of `machined_facej`, This idea is illustrated in Fig. 4.3 and

the condition can be expressed as :

$$\begin{aligned} V(\text{machined\_face}_i) &\in \text{half-space}_i \\ P(\text{machined\_face}_j) &\in \text{half-space}_j \\ \text{half-space}_i &= c(\text{half-space}_j) \end{aligned} \quad (7)$$

where  $\in$  means 'is a subset of', and  
 $c$  is a complement operator.

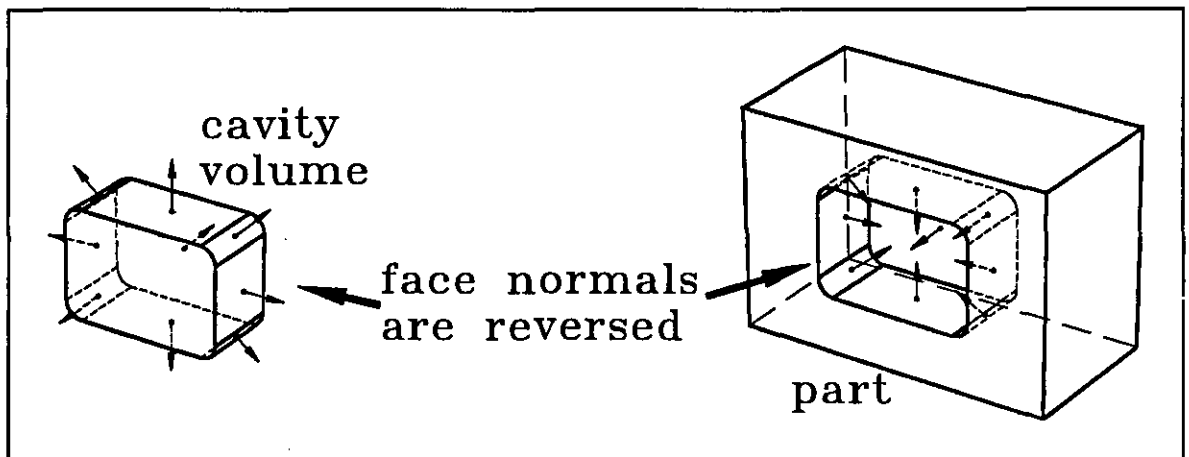


Figure 4.3 : The reverse image notion of machined\_faces.

In summary, by classifying the cavity volume boundary into tool\_entrance\_faces and machined\_faces, the semantic content of the cavity volume model is significantly enhanced. The geometry and topology of the tool\_entrance\_faces are also directly deduced from the starting stock during the Boolean subtraction operation.

In this research, the cavity volume boundary faces are further augmented with machining process related meanings during the recognition process, and in order to avoid confusion, this tool\_entrance\_face and machined\_face classification will hereafter be described as the 'nature' of the face. The procedure for determining the nature of the cavity volume boundary faces is described in the next chapter.

#### 4.4 Machining Feature Representation

A machining feature is defined as a machining region on a machined part that would be produced by a machining operation performed on a cutting machine with 2CL kinematic capability. Based on the cavity volume modelling concept discussed above, a machining feature can equally be conceived as a machining region on a cavity volume. Recognition of machining features is viewed as a process of identifying and extracting machining regions from the nominal geometry model of the cavity volume.

An extracted machining feature is represented as a set of faces on the cavity volume boundary : (1) the `part_face`, (2) the `check_face`, (3) the `primary_top_entrance_face`, (4) the `secondary_top_entrance_face`, and (5) the `side_entrance_face`. They are explained in detail below.

##### (1) the `Part_Face`

The name is borrowed from the term 'part surface' of the APT terminology [IIT67]. It serves to provide a reference surface for defining the limiting position of the bottom of a cutter when the represented feature is machined. Five `part_face` conditions are considered :

##### (condition 1) - when the nature of the `part_face` is a `tool_entrance_face`

This means that the represented feature is a through feature such as a simple through hole (Fig. 4.4).

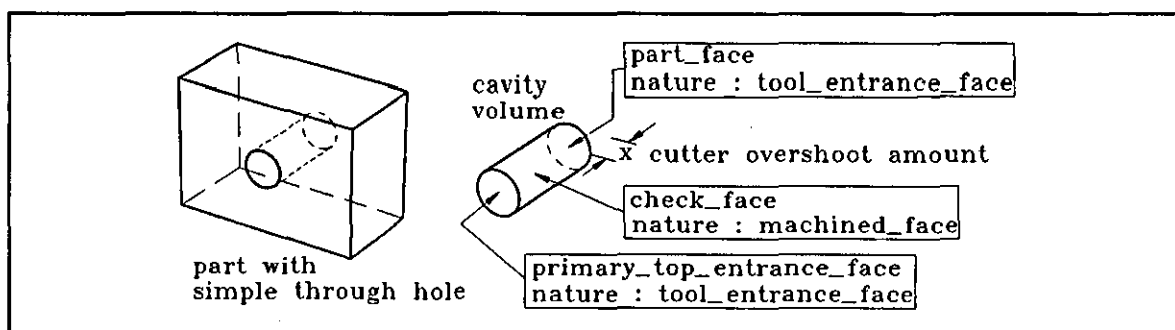


Figure 4.4 : `Part_face` condition 1 (orthogonal intersection with `check_face`).

In practice, the bottom of a cutter will overshoot the part\_face by a certain distance so that the feature can be cut through. If the part\_face is a planar surface, the amount of overshoot ( $x$ ) can be determined based on a consideration of the cutter tip geometry and the amount of clearance gap allowed underneath the part\_face (Fig. 4.4).

If the part\_face is non-planar or the intersection between the part\_face and the check\_face(s) is non-orthogonal, the amount of overshoot will be  $(x + y)$  where  $y$  can be determined by calculating the difference between the 'highest' and the 'lowest' positions (in the direction of the machine spindle axis) on the edge loop curves formed between the part\_face and the check\_face(s) (Fig. 4.5).

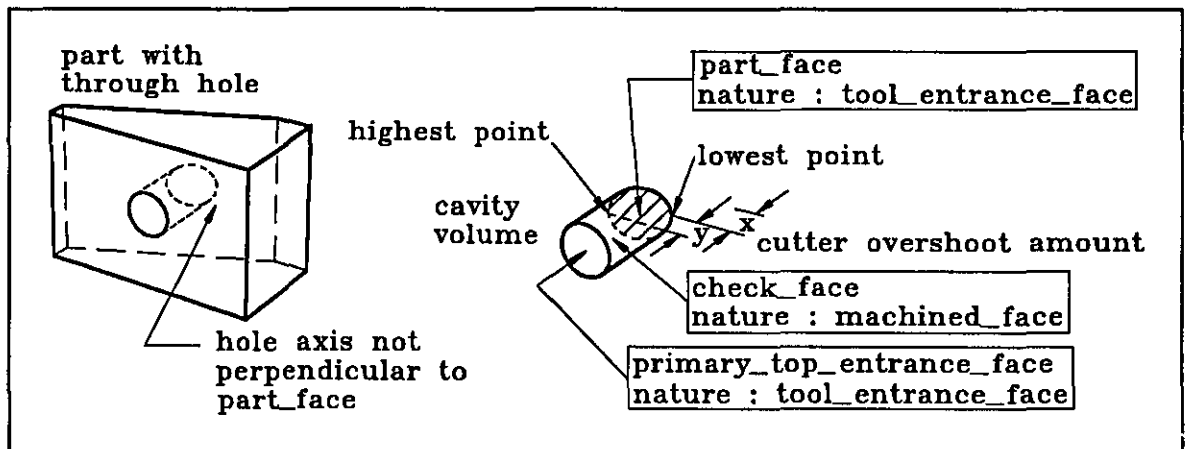


Figure 4.5 : Part\_face condition 1 (non\_orthogonal intersection with check\_face).

As the part\_face is a tool\_entrance\_face, it also means that the feature can be machined the other way round, i.e. using the original part\_face as a primary\_top\_entrance\_face and the original primary\_top\_entrance\_face as a part\_face. The recognition of these alternative machining directions is the task of the feature recognizer but the final interpretation or selection of a machining direction is considered to be a duty of process planning.



**(condition 2) - when the nature of the part\_face is a machined\_face, and its outer edge loop formed with the check\_face(s) consists of convex edges only**

This means that the represented feature is a non-through feature such as a blind hole and a pocket (Fig. 4.6). In practice, the bottom of a cutter stops on the part\_face, as in the case of blind hole drilling, or rides on the part\_face, as in the case of pocket milling.

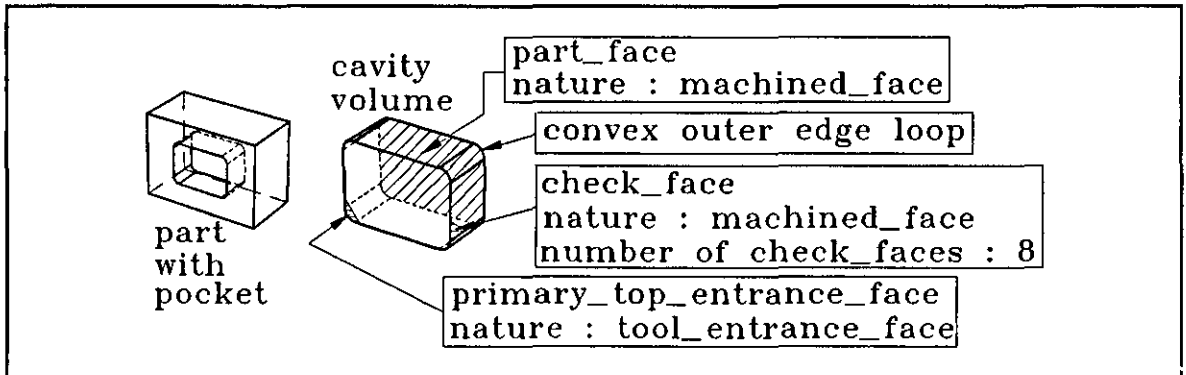


Figure 4.6: Part\_face condition 2.

**(condition 3) - when the nature of the part\_face is a machined\_face, and its outer edge loop formed with the check\_face(s) consists of concave edges only**

This means that the represented feature is a protrusion feature such as a boss or an island (Fig. 4.7). In practice, the bottom of a cutter rides on the part\_face which is actually the top face of the protrusion. The material surrounding a protrusion feature will be removed in another machining operation (i.e. another machining feature) such as surface milling (if the protrusion is on an open surface) and pocket milling (if the protrusion is contained in a pocket).

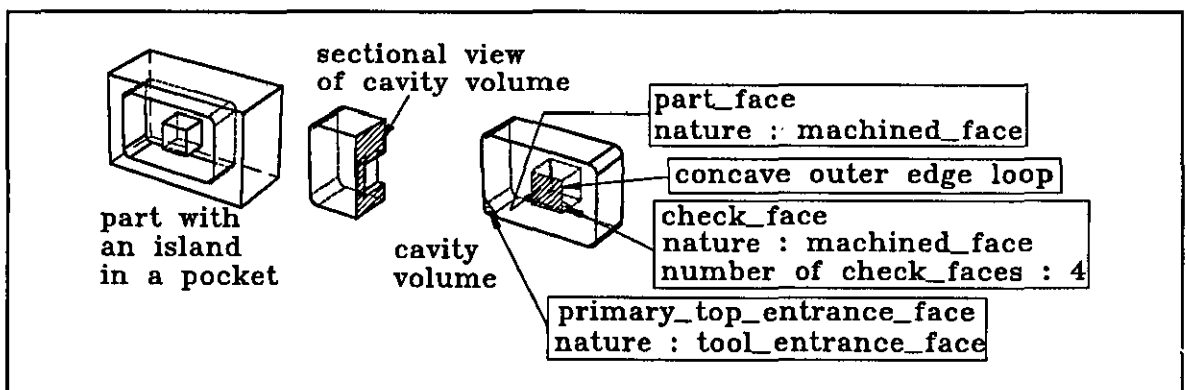


Figure 4.7 : Part\_face condition 3.

(condition 4) - when the nature of the `part_face` is a `machined_face`, and its outer edge loop formed with the `check_face(s)` consists of both concave and convex edges  
 This means that the represented feature interacts with another feature either in the manner as illustrated in Fig. 4.8 or in the manner as illustrated in Fig. 4.9. In practice, the bottom of a cutter rides on the `part_face`.

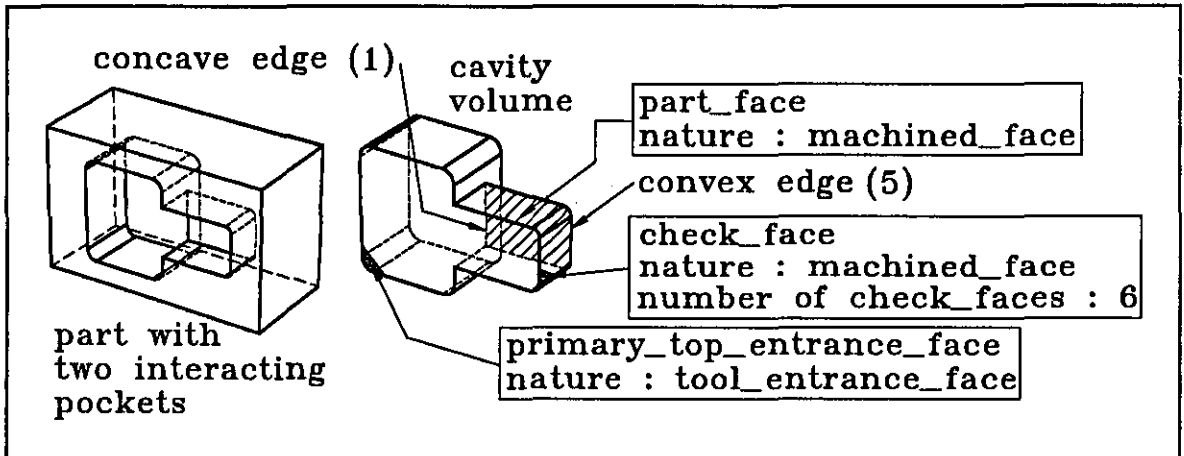


Figure 4.8 : Part\_face condition 4 (external interaction).

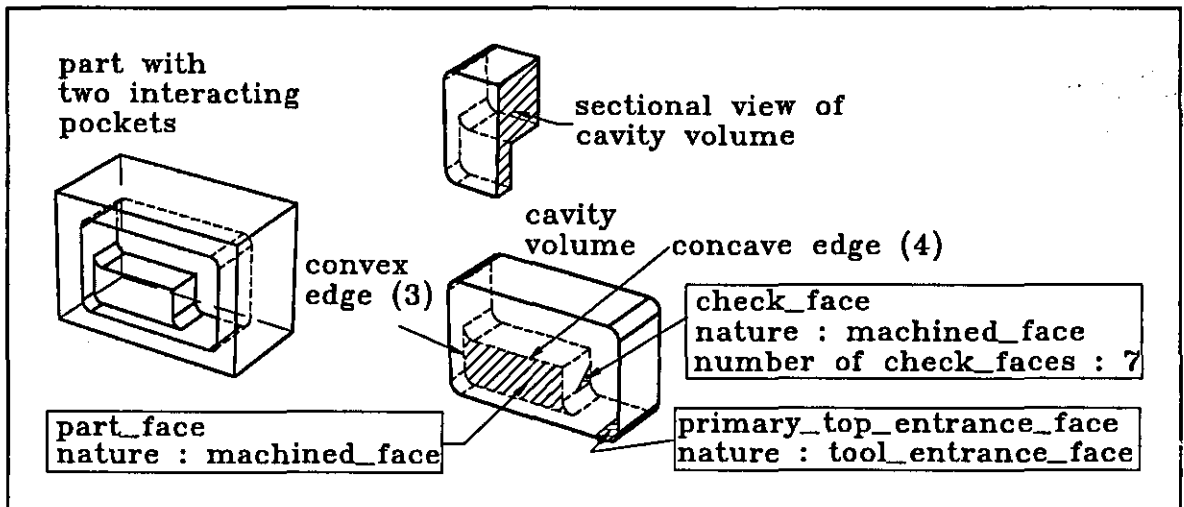


Figure 4.9 : Part\_face condition 4 (internal interaction).

condition 5) - when the nature of the `part_face` is a `machined_face`, and its edge loop formed with the `check_face(s)` is a concave, inner-loop

This means that the represented feature interacts with another feature in the manner as illustrated in Fig. 4.10, forming an inner edge loop in one of the faces of the interacting feature. In practice, the bottom of a cutter will overshoot the `part_face` by a distance that can be determined similarly as described above in (condition 1).

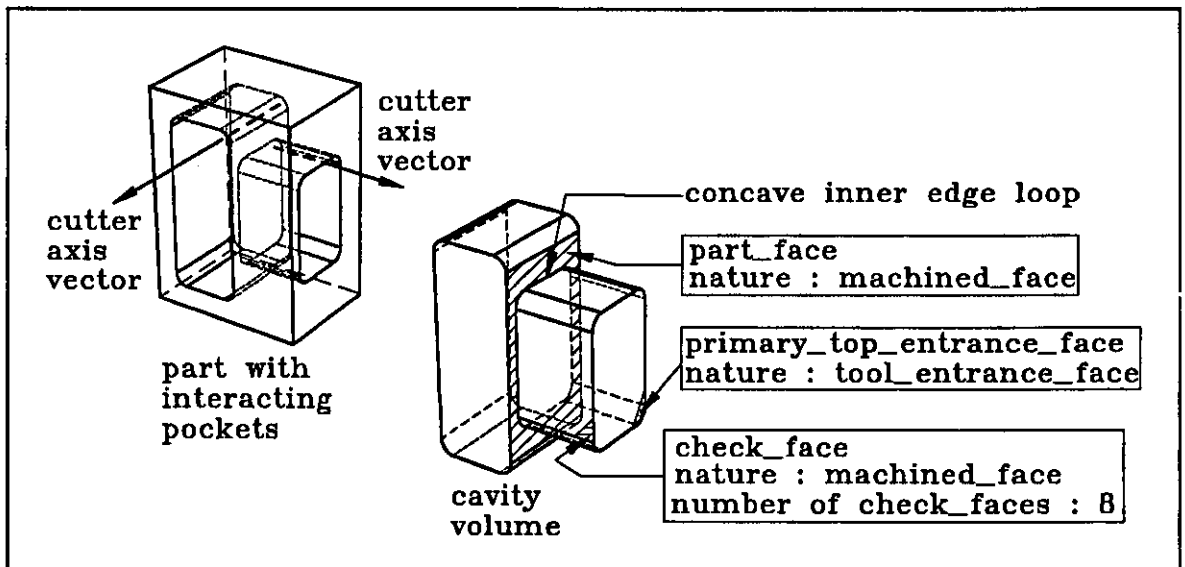


Figure 4.10 : Part\_face condition 5.

## (2) the Check\_Face

The name is also borrowed from the term 'check surface' of the APT terminology [IIT67]. The `check_face` can be conceived as the wall face of a machining feature. It serves to check or limit the lateral movement of a cutter. There may be only one `check_face`, as in the case of a cylindrical hole, or more than one `check_faces`, as in the case of a rectangular pocket. The nature of a `check_face` can be either a `tool_entrance_face` or a `machined_face`. A `check_face` of `tool_entrance_face` nature implies that it is a `side_entrance_face`. The `check_face` is assumed to be adjacent to the `part_face`. It facilitates the classification of the various `part_face` conditions as described above. Moreover, it is also used to determine the orientation of the machining feature

with respect to the spindle axis. The rule adopted is that if one of the `check_faces` is a cylindrical surface then the axis of the cylindrical surface is used as the cutter axis vector, otherwise, the line vector of a linear edge made between two `check_faces` will be used as the cutter axis vector (Fig. 4.10). This rule is based on the assumption that simple cylindrical cutters such as end mills are used and the formerly stated assumption that the machining operation is performed on a cutting machine with 2CL kinematic capability. It is emphasized that using the surface normal of the `part_face` as the cutter axis vector is unreliable because the `part_face` may not be planar and the intersection between the `part_face` and the adjacent `check_face(s)` may not be orthogonal as illustrated in Fig. 4.5.

### (3) the `Primary_Top_Entrance_Face`

The nature of this face is obviously a `tool_entrance_face`. It represents the area through which a cutter can enter axially into the machining feature as illustrated in the previous figures. The distance between this face and the `part_face` represents the total depth of cut required. It is assumed that a machining feature can have a cylindrical `primary_top_entrance_face` and or more than one `primary_top_entrance_faces` that are connected together in the form of a group of face patches as illustrated in Fig. 4.11.

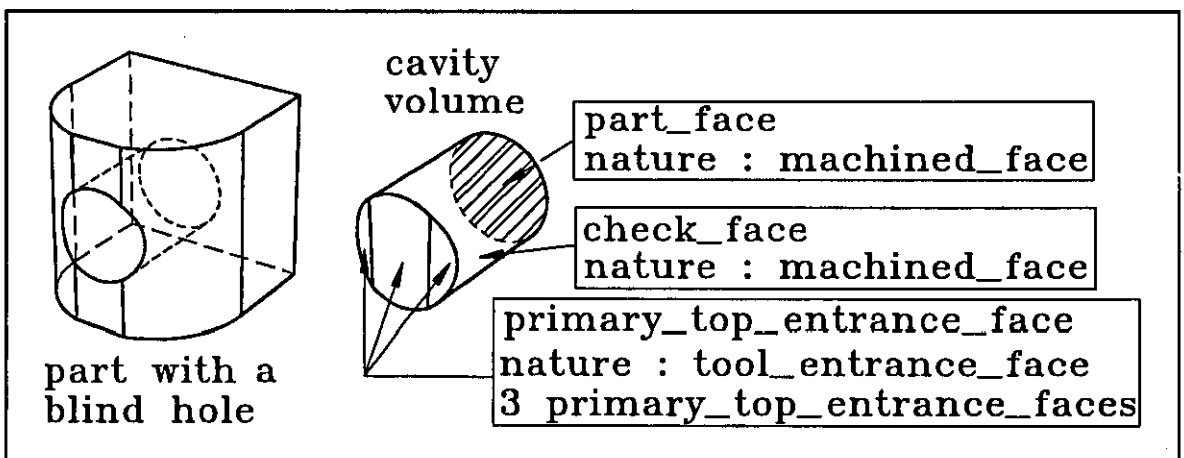


Figure 4.11 : A blind hole that has multiple, non-planar `primary_top_entrance_faces`.

**(4) the Secondary\_Top\_Entrance Face**

The nature of this face is a machined\_face. It lies between the primary\_top\_entrance face and the part\_face. The region within its inner edge loop or outside its outer edge loop represents the area through which a cutter can enter axially into the machining feature (Figs. 4.12 and 4.13). The secondary\_top\_entrance\_face is in fact used as a part\_face by another machining feature that is above the represented feature. In other words, when a represented feature has one or more secondary\_top\_entrance\_faces, it signifies that the represented feature can be machined after machining its upper features whose information can be addressed through the secondary\_top\_entrance\_faces.

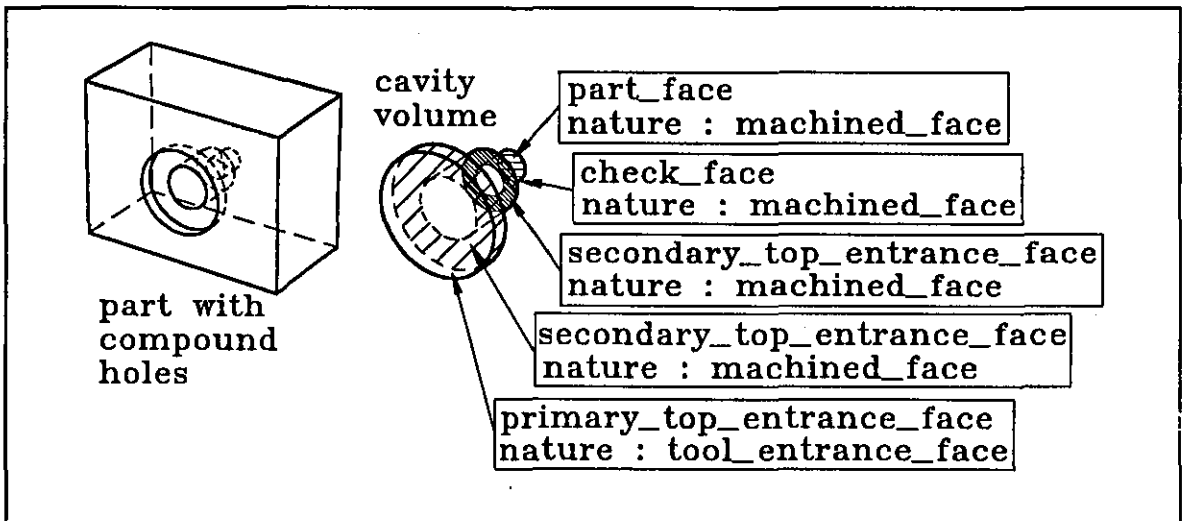


Figure 4.12 : The lowest hole has two secondary\_top\_entrance\_faces.

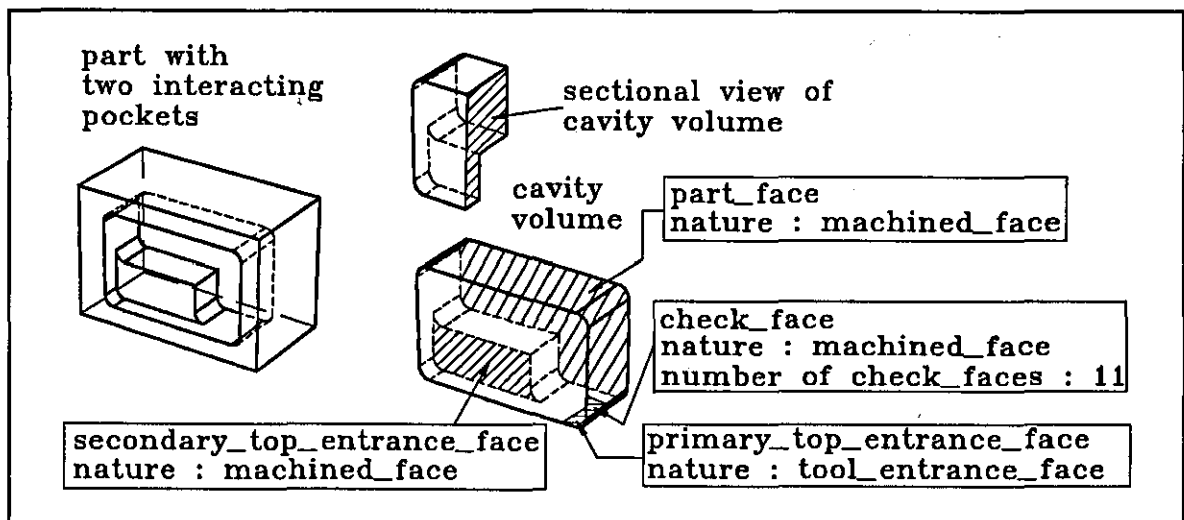


Figure 4.13 : The lower pocket has one secondary\_top\_entrance\_face.

**(5) the Side\_Entrance\_Face**

As mentioned above, this is a check\_face of tool\_entrance\_face nature. It represents the area through which a cutter can enter radially (or laterally) into the machining feature (Fig. 4.14). A tool\_entrance\_face is regarded as a side\_entrance\_face with respect to its adjacent recognised part\_face. Although this information can be obtained by means of interrogating the nature of the check\_face(s), the intention of including it is to provide a more direct representation of the presence of side\_entrance\_face possessed by a represented feature. For example, a feature without a side\_entrance\_face will immediately be interpreted as a feature that has no lateral openings, such as holes, and closed pocket, whereas features such as open slots and notches will have side\_entrance\_faces.

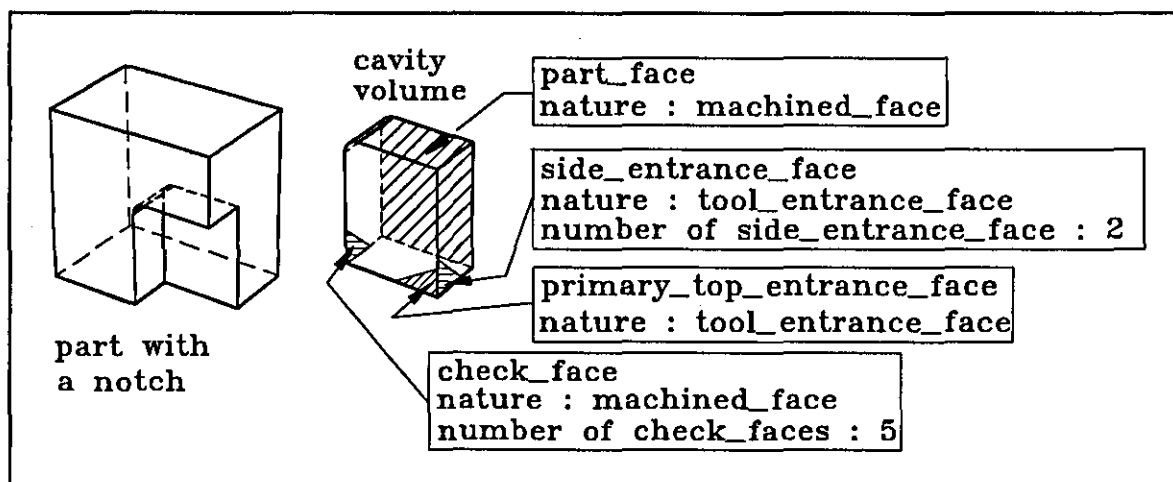


Figure 4.14 : A corner notch that has two side\_entrance\_faces.

Two illustrated examples are presented in Appendix B for giving a summarized view of the feature representation scheme.

#### 4.5 Concluding Remarks

In essence, the proposed machining feature representation scheme collaborates with the boundary databases of the cavity volume model to convey machining methods and machining process geometry of a machining region which can support both high level process planning and low level cutter path calculation. By virtue of the `secondary_top_entrance_face` information and the `part_face` conditions 4 and 5 described above, complex feature interactions can also be represented.

It is considered that given a feature classification taxonomy such as the one proposed by Gindy [Gindy89] (introduced in chapter 2), a more precise feature classification can be easily interpreted and obtained from the machining feature representation. In fact, an interestingly close comparison can be drawn between the feature representation adopted by Gindy and the one used in this thesis as shown below.

Feature attributes used by Gindy	Feature attributes used in this thesis
entry/exit face	<code>primary_top_entrance_face/part_face</code>
boundary type (closed/open)	<code>check_face/side_entrance_face</code>
exit boundary status	various <code>part_face</code> conditions
external access direction	cutter axis vector
depth axis	distance between <code>primary_ (or secondary) top_entrance_face</code> and <code>part_face</code> along cutter axis vector

Table 4.11 : Feature attributes comparison.

Moreover, the explicit machining method and geometry information provided by the representation scheme can facilitate the generation of non-invasive and collision free cutter paths even for an awkward shaped machining regions that cannot be classified into any specific feature type by a feature classification scheme. In the next chapter, a feature recognition algorithm that is based on this representation scheme is described.

## CHAPTER 5

# MACHINING FEATURE RECOGNITION ALGORITHM

This chapter describes an algorithm for the recognition of generic 2.5D machining features based on the feature representation scheme discussed in the last chapter. The criteria for recognizable machining features are defined first followed by the detailed description of the algorithm.

### 5.1 Criteria for Recognizable Machining Features

#### Criterion (1)

To be recognizable, the machining features should satisfy any one of the five `part_face` condition types described in section 4.4.

#### Criterion (2)

The machining features are machined by cylindrical cutters such as twist drills, end mills and slot drills on milling machines with the use of 2CL kinematic capability. The detailed cutter geometry such as the conical tip of a twist drill is ignored. Thus, for non-through features, such as non-through holes and pockets, the `part_face` is assumed to be planar, and the edge angle between the `part_face` and the surrounding `check_faces` of `machined_face` nature is a right angle as illustrated in Fig. 5.6. However, for through features, the `part_face` can be planar or cylindrical, and the edge angle between the `part_face` and the surrounding `check_faces` of `machined_face` nature is not necessarily a right angle (such as the situation illustrated in Fig. 4.5). If the surrounding `check_faces` are of `tool_entrance_face` nature, the edge angle is immaterial since the `check_faces` are basically `side_entrance_faces`.



### Criterion (3)

For a cylindrical check\_face of machined\_face nature, the cylindrical axis must be parallel to the cylindrical cutter axis  $z$ . For a planar check\_face of machined\_face nature, the face normal must be orthogonal to  $z$ . This means that the recognizable machining features must have non-sloping check\_faces of machined\_face nature with respect to the cutter axis as illustrated in Fig. 5.1.

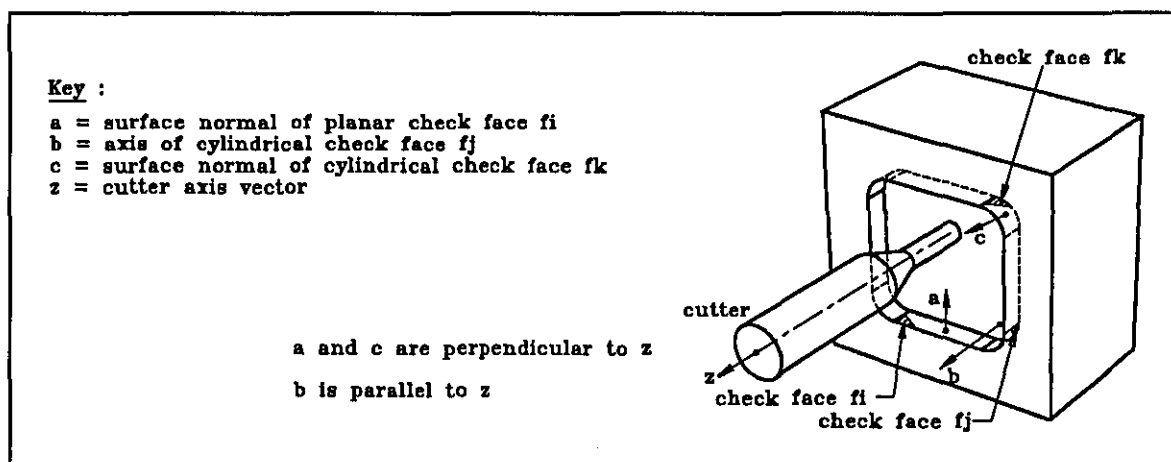


Figure 5.1 : Recognizable features have non-sloping check\_faces of machined\_face nature.

The machined\_face nature is emphasized because the geometric conditions need not apply to a check\_face of tool\_entrance\_face nature (such as the situation illustrated in Fig. 4.14) which is actually a side\_entrance\_face.

### Criterion (4)

The volumetric space above the part\_face is not interfered with by the other faces of the part as illustrated in Fig. 5.2. This basically implies that recognized machining feature must be accessible by an infinitely long cylindrical cutter without gouging the part.

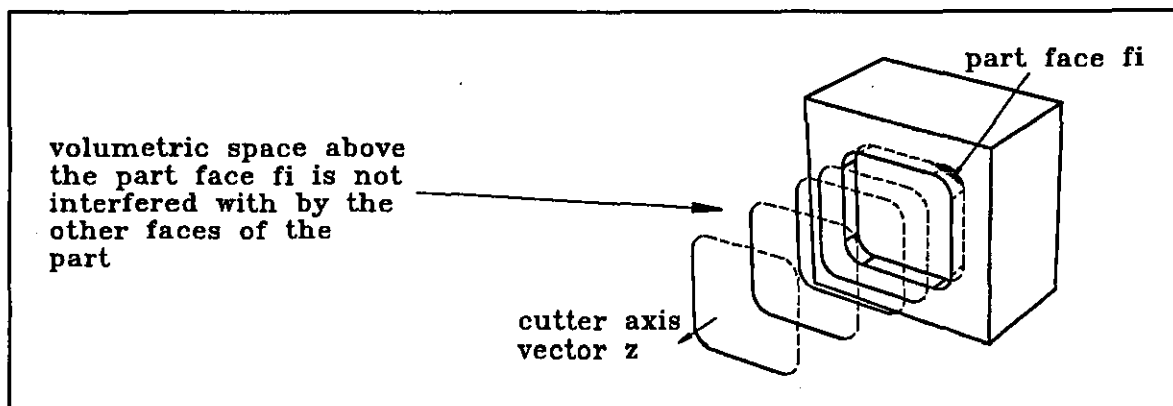


Figure 5.2 : Volumetric space above the part\_face is not obstructed.

In summary, each recognizable machining feature can be regarded as a 2.5D machining feature. However, the part or the cavity volume from which the machining features are recognized may not be a 2.5D solid as it can contain faces that are multiply-connected, i.e. with inner edge loop(s) such as the one illustrated in Fig. 4.10. Thus, the above constraints on the geometry that can be recognized still allow for the description of real parts which constitute a significant proportion of actual manufacture.

## 5.2 Overview of the Algorithm

The recognition is based primarily on the B-rep model of the cavity volume (or its subvolumes) obtained via Boolean subtraction between the corresponding stock and finished part models. However, as the bounding envelope of the stock and the boundary faces of the finished part are also utilized in the recognition process, the algorithm actually relies on the B-reps of the stock, part and cavity volume.

This chapter focuses on the description of the recognition algorithm with the assumption that the three B-rep models are available. In particular, if the cavity volume contains subvolumes as described by expression(3) in chapter 4, each individual subvolume is assumed to be addressable in terms of its complete set of boundary faces. Moreover, the convexity of the edges is assumed to have been determined and the

boundary faces are labelled as `machined_face` or `tool_entrance_face` as described by expression(6) in chapter 4. Details concerning the methods of establishing the B-rep models and the implementation of the algorithm are presented in chapter 7.

The recognition algorithm involves the geometric reasoning of three groups of candidate faces of a cavity volume (or its subvolumes) in the following sequence :

group(1) - in which the nature of the candidate face to be analyzed is of `machined_face` nature and the concerned edge loop is an outer edge loop of the candidate face. Hence, the group(1) faces are basically the `part_face` condition types 2, 3, and 4 as mentioned in section 4.4,

group(2) - in which the nature of the candidate face to be analyzed is of `tool_entrance_face` nature. Thus the group(2) faces represent the `part_face` condition type 1,

group(3) - in which the nature of the candidate face to be analyzed is of `machined_face` nature and the concerned edge loop is an inner edge loop of the candidate face. So the group(3) faces essentially represent the `part_face` condition type 5.

The reason for analyzing the group(1) faces before the group(2) faces is arbitrary and is based on the assumption that non-through features occur more frequently than through features. The group(3) faces are dealt with last since the method relies on the reasoning results of the former two groups of faces. This point is clarified in the following description of the algorithm.

For each of the above three groups of faces, the following three major steps are repeated until no candidate face can be selected :

- (1) select a candidate face,
- (2) perform geometric reasoning on the candidate face,
- (3) utilize the geometric reasoning results.

An outline of the complete algorithm looks like :

```

Procedure Recognize_Machining_Features (cavity volumeB-rep, partB-rep, stockB-rep)
  For each subvolume in the cavity volumeB-rep, Repeat
    For the faces in group(1), Repeat
      (1) select a candidate face
      (2) perform geometric reasoning on the candidate face
      (3) utilize the geometric reasoning results
    Until no candidate face can be selected
    For the faces in group(2), Repeat
      (1) select a candidate face
      (2) perform geometric reasoning on the candidate face
      (3) utilize the geometric reasoning results
    Until no candidate face can be selected
    For the faces in group(3), Repeat
      (1) select a candidate face
      (2) perform geometric reasoning on the candidate face
      (3) utilize the geometric reasoning results
    Until no candidate face can be selected
  Until no more subvolume
End {Procedure}

```

Although the three major steps appear in each face group, the detailed mechanisms of the steps for each face group have some variations. A hypothetical part shown in Fig. 5.3 is used to explain the algorithm step by step.

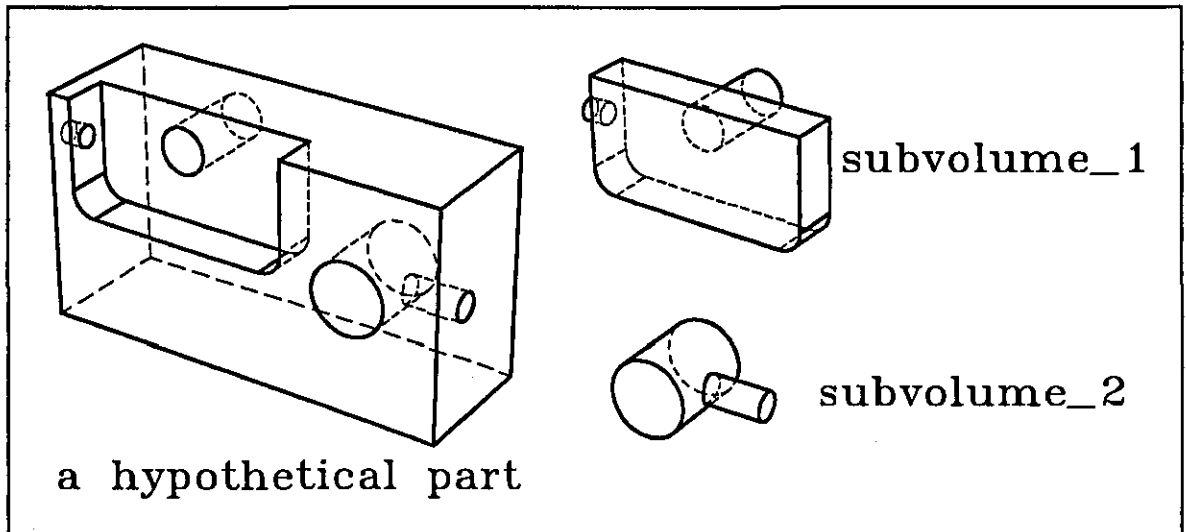


Figure 5.3 : A hypothetical part used to illustrate the explanation of the algorithm.

As can be seen in Fig. 5.3, the hypothetical part is assumed to be machined from a starting stock in the form of a rectangular block. The cavity volume thus formed consists of two subvolumes, i.e. the `subvolume_1` and the `subvolume_2`. For illustration purposes, the subvolumes are represented in Figs. 5.4 and 5.5 by means of a face-edge graph.

In the face-edge graph, the rectangular nodes represent the boundary faces of a subvolume, while an arc joining two face nodes represents an edge shared between the two faces. A small circle attached to the side of a face node indicates that the face is multiply-connected, i.e. has inner edge loop(s). Edges belonging to an inner edge loop of a face will link to the circle rather than to the side of the rectangular face node. Understandably, the number of circles attached to a face node indicates the number of inner edge loops owned by the corresponding face. The 'surface type' and 'face nature' attributes are depicted in every face node. The 'access' and 'status' attributes are utility flags used in the recognition algorithm. The convexity of an edge is shown as an integer code by the side of the edge arc. For the working of the algorithm, every edge also has a status flag. The initial value of the status of every edge is nil.

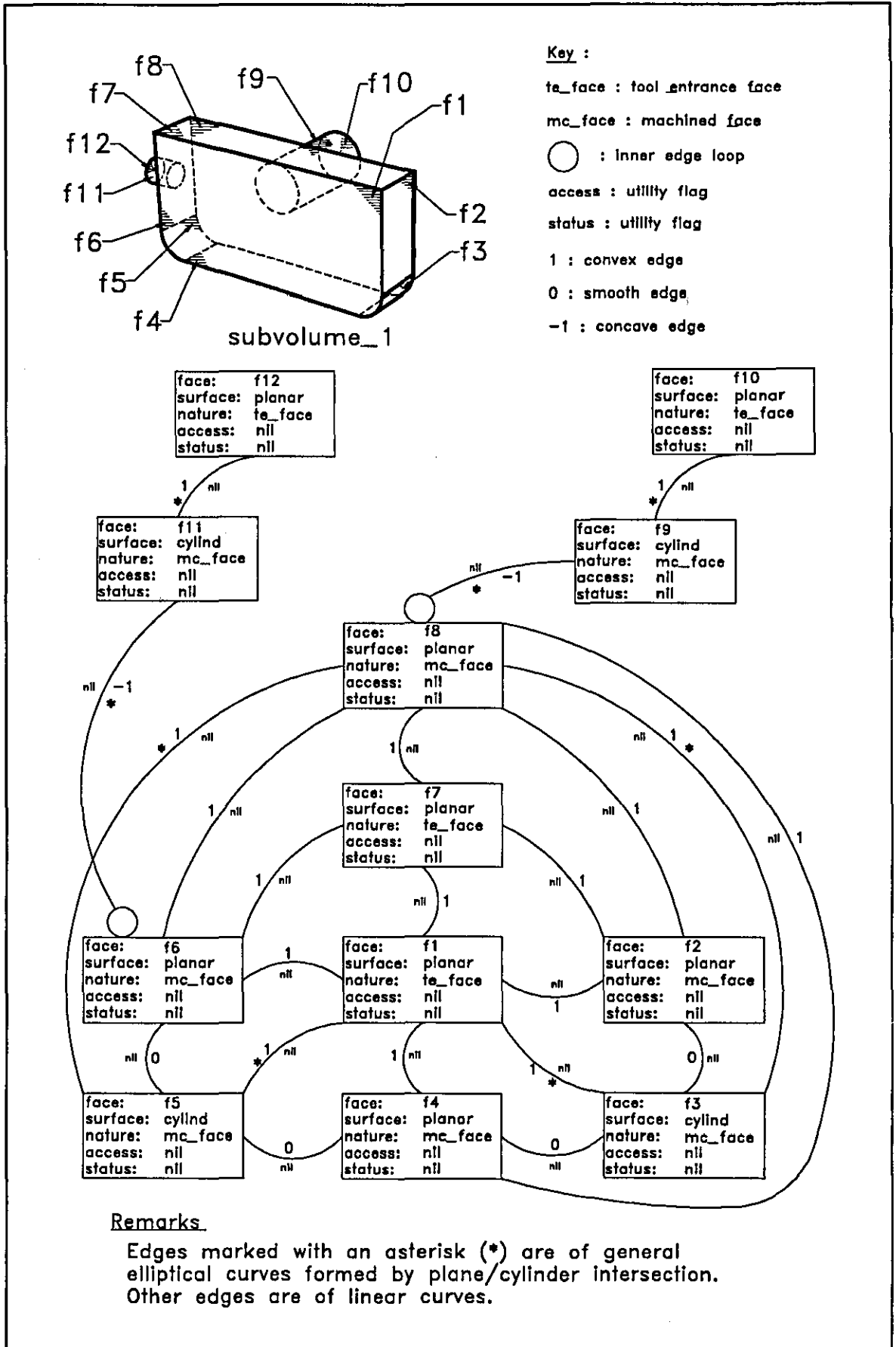


Figure 5.4 : The face/edge graph representation of subvolume\_1.

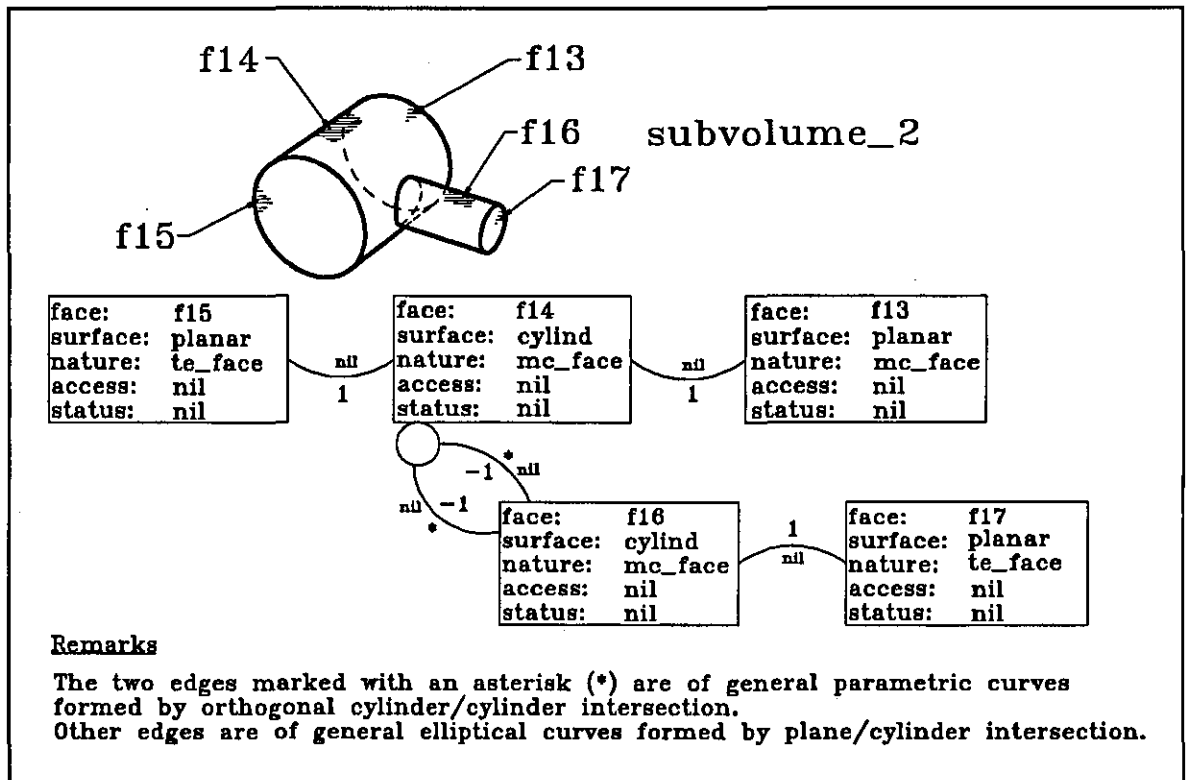


Figure 5.5 : The face/edge graph of subvolume\_2.

In the context of AI problem solving, the face-edge graphs depicted in Figs. 5.4 and 5.5 can be viewed as a representation of the initial condition states of two problem spaces. The recognition algorithm basically tries to analyze the problem states according to the criteria described in section 5.1 by means of geometric reasoning, and to transform the problem states by means of propagating the geometric reasoning results as new problem states or constraints in the problem space until no more state transformation is possible. Each intermediate transformation represents a success or a failure of recognizing a machining feature.

### 5.3 Recognition of Machining Features from the Subvolume\_1

Since the subvolumes are disjoint solids as described by expression(3) in section 4.2, the machining features in one subvolume will not interact geometrically with the machining features in another subvolume. As the objective of the recognition algorithm is to expose all the independent machining features, it is considered that the sequencing of subvolumes for recognition is not important at this stage. However, the fact that machining features extracted from individual subvolumes may constitute to a higher level feature pattern, such as a pattern of holes lying on a pitch circle diameter, will be discussed in chapter 8. For the purpose of better explaining the algorithm, the subvolume\_1 is chosen first.

#### 5.3.1 Machining Heuristics

To facilitate the search for candidate faces, two heuristics are used to rank the tool\_entrance\_faces and machined\_faces in terms of their selection priority. The first heuristic is that :

- If** a planar face  $f$  (of either tool\_entrance\_face or machined\_face nature) contains one or more non-linear edges in its boundary,
- Then**  $f$  is more likely to be used as a part\_face.

This is because the presence of a non-linear edge signifies the presence of an adjacent cylindrical check\_face that can be easily machined by the revolving action of a cylindrical cutter (Fig. 5.6).



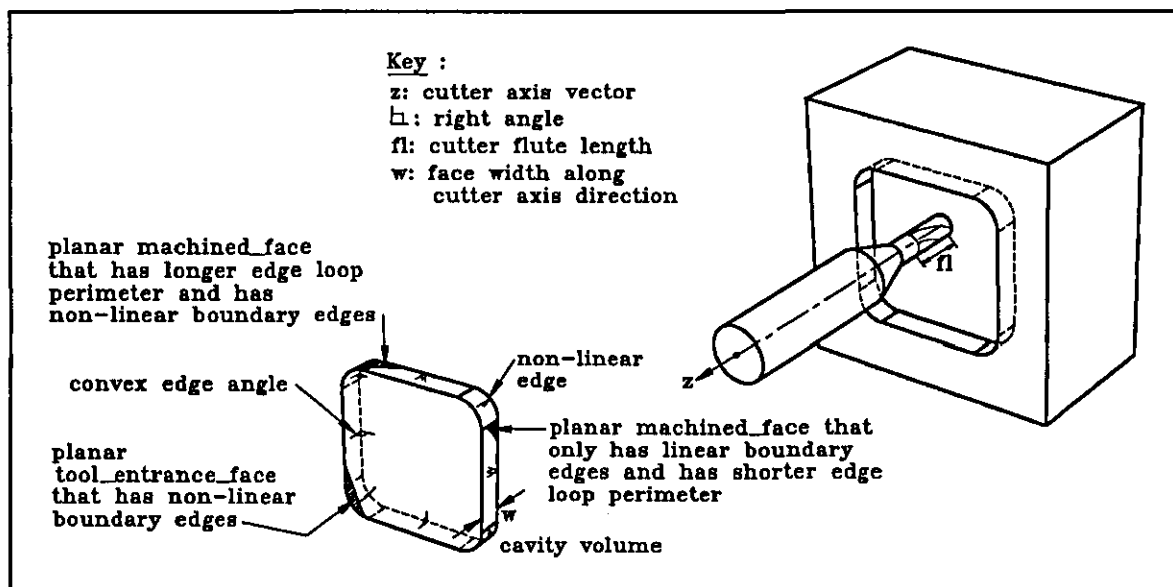


Figure 5.6 : Illustration of the two machining heuristics used.

The second heuristic is that :

**If** a planar face  $f$  has a longer edge loop perimeter,  
**Then**  $f$  is more likely to be used as a part\_face.

This implies that the adjacent faces of face  $f$  would have a shorter edge loop perimeter. Since each of the adjacent faces shares an edge with face  $f$ , the face width of the adjacent faces measured along the cutter axis direction is more likely to be shorter. As cutters, such as drills and end mills, have a finite cutter flute length, it is more likely that the adjacent faces can be machined by the cutter flutes (Fig. 5.6). These two heuristics are applied by means of sorting the tool\_entrance\_faces and machined\_faces in the following manner :

$$\{ \text{pne1, pne2, ... p1, p2, ... c1, c2 ... } \}$$

where pne1, pne2, etc. are planar faces that contain non-linear edge(s) and the edge loop perimeter of pne1 is longer than that of pne2,  
 p1, p2, etc. are planar faces that do not contain non-linear edge and the edge loop perimeter of p1 is longer than that of p2, and  
 c1, c2, etc. are cylindrical faces sorted in descending order of their edge loop perimeter.

The procedure for sorting the `tool_entrance_faces` is illustrated below, while the procedure for sorting the `machined_faces` is similar.

```

Procedure Sort_Tool_Entrance_Faces (subvolumeB-rep,           {input parameter}
                                     tool_entrance_face_list) {output parameter}
  create a working list_1 and a working list_2
  put the planar tool_entrance_faces of subvolumeB-rep into the working list_1
  put the cylindrical tool_entrance_faces of subvolumeB-rep into the working list_2
  create a working list_3 and a working list_4
  For the faces in the working list_1, Repeat
    if      the face contains a non-linear edge
    then    put the face into the working list_3
    else    put the face into the working list_4
  Until all the faces have been checked
  sort the faces in the working list_3 in descending order of face edge loop perimeter
  sort the faces in the working list_4 in descending order of face edge loop perimeter
  sort the faces in the working list_2 in descending order of face edge loop perimeter
  copy the faces in the working lists_3, 4, and 2 to the tool_entrance_face_list
End {Procedure}

```

For the purpose of explaining the algorithm, the sorted faces are assumed to be stored in two linear lists, i.e. a `tool_entrance_face` list and a `machined_face` list. However, in the actual implementation the sorted faces are represented in the context of a KBS as a set of frames of faces. Details about this point is described in chapter 7.

For the `subvolume_1`, the `tool_entrance_face` list contains faces arranged as : {f1, f10, f12, f7}, while the `machined_face` list contains faces arranged as : {f8, f4, f2, f6, f3, f5, f9, f11} (please refer to Fig. 5.4 for face notations).

### 5.3.2 Selection of the Group(1) Faces

As described in section 5.2, the algorithm examines the group(1) faces first. As the group(1) faces are of machined\_face nature, the following rule is used to select a candidate face from the machined\_face list :

**If**  $f$  is a face to be selected from the machined\_face list,  
 $f$  is planar,  
the value of the access attribute of  $f$  is not zero, and  
the value of the status attribute of  $f$  is neither 'part\_face' nor 'check\_face',  
**Then** select  $f$  as the candidate face.

The first face f8 in the machined\_face list satisfies the above rule. So it is selected as a candidate face for geometric reasoning.

### 5.3.3 Geometric Reasoning for the Group(1) Faces

For analyzing the group(1) candidate faces, three major geometric tests are conducted to ensure that the criteria (2), (3) and (4) described in section 5.1 are satisfied.

#### 5.3.3.1 The First Geometric Test for the Group(1) Faces

In the first geometric test, the edges in the outer edge loop of the candidate face are examined in turn by the following rules :

**If** an edge is concave, or  
an edge is convex and its adjacent face is of tool\_entrance\_face nature,  
**Then** no further test for the edge is necessary (since its adjacent face will not obstruct the cutter to reach the candidate face as illustrated in Figs. 4.8, 4.9, and 4.14)

**If** an edge is convex and its adjacent face is of machined\_face nature,  
**Then** the convex edge angle is computed to see whether or not it is equal to a right angle. If the convex edge angle is a right angle, then the conditions stated in criterion (1) are satisfied and the candidate face passes the test, otherwise the candidate face fails the test.

**If** an edge is smooth,  
**Then** the candidate face fails the test (since the smooth edge signifies that the axis of its adjacent cylindrical face is not parallel with the cutter axis).

For the candidate face f8, convex edge e6 is exempted from the convex edge angle test since its adjacent face f7 is of tool\_entrance\_face nature (Fig. 5.7).

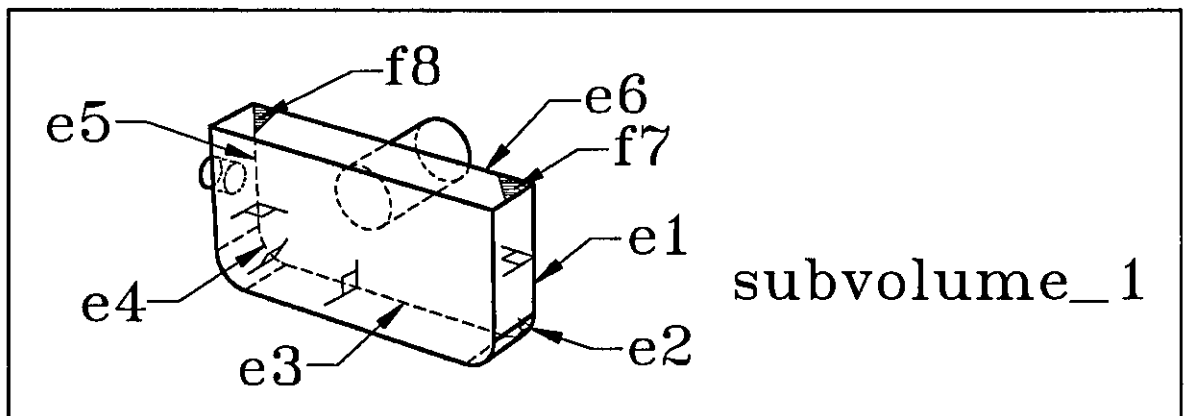


Figure 5.7 : Convex edge angles of candidate face f8 are right angles.

Convex edges e1, e2, e3, e4, and e5 have edge angles equal to a right angle. So the first test is successful and the second test can proceed. In the event that this first test is not satisfied, the geometric reasoning for the candidate face will terminate as the candidate face does not constitute to a valid part\_face of a machining feature.

### 5.3.3.2 The Second Geometric Test for the Group(1) Faces

In the second test, the edges of the candidate face are slightly offset towards the inside of the subvolume as illustrated in Fig. 5.8.

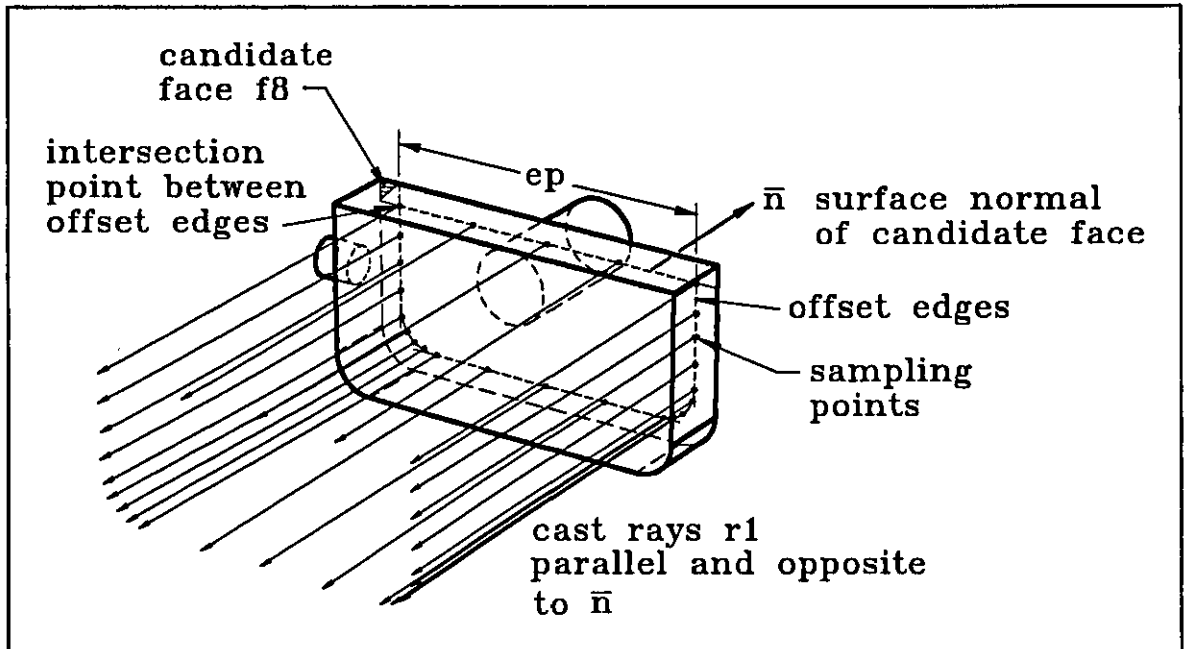


Figure 5.8 : Offsetting edges and casting rays  $r1$  from candidate face  $f8$ .

When a candidate face has only one elliptical (circular) edge as in the case of a circular hole, the edge offset operation amounts to a shrinkage of the circular edge diameter [Tiller84, Saeed88]. When a candidate face has several boundary edges, the offset edges may intersect each other as shown in Fig. 5.8. For an offset edge that has intersection with its adjacent edges, the portion between the intersection points is taken as  $ep$ , otherwise, the full length of the offset edge is taken as  $ep$ . Sampling points with equal interval between them are taken on  $ep$ . In the actual implementation, five sampling points, including the two end points, are used. From each sampling point, a semi-infinite line or ray  $r1$  is projected such that the ray is parallel but opposite in direction to the surface normal of the candidate face. The cast rays  $r1$  are tested for possible intersection with the half-spaces of the subvolume by means of a line/surface intersection computation [Roth82]. The principle of the line/surface intersection computation is presented in Appendix C.

To speed up the test, a preliminary test is used to sort out some of the half-spaces that do not require the line/surface intersection test. For instance, planar half-spaces whose surface normals are perpendicular to the surface normal of the candidate face and cylindrical half-spaces whose axes are parallel to the surface normal of the candidate face can be excluded from the line/surface intersection test because they do not intersect with the cast rays. In the current example, planar half-spaces whose surface normals are square with the surface normal of the candidate face  $f_8$  are those of faces  $f_2$ ,  $f_4$ ,  $f_6$  and  $f_{12}$ . Cylindrical half-spaces whose axes are parallel to the surface normal of the candidate face are those of faces  $f_3$ ,  $f_5$ , and  $f_9$ . Besides, planar half-space of face  $f_{10}$  also does not intersect the cast rays because it lies 'behind' the origins of the cast rays. As can be seen from the illustration in Fig. 5.9, all the cast rays  $r_1$  intersect the planar half-spaces of face  $f_1$  and some cast rays intersect the cylindrical half-space of face  $f_{11}$ .

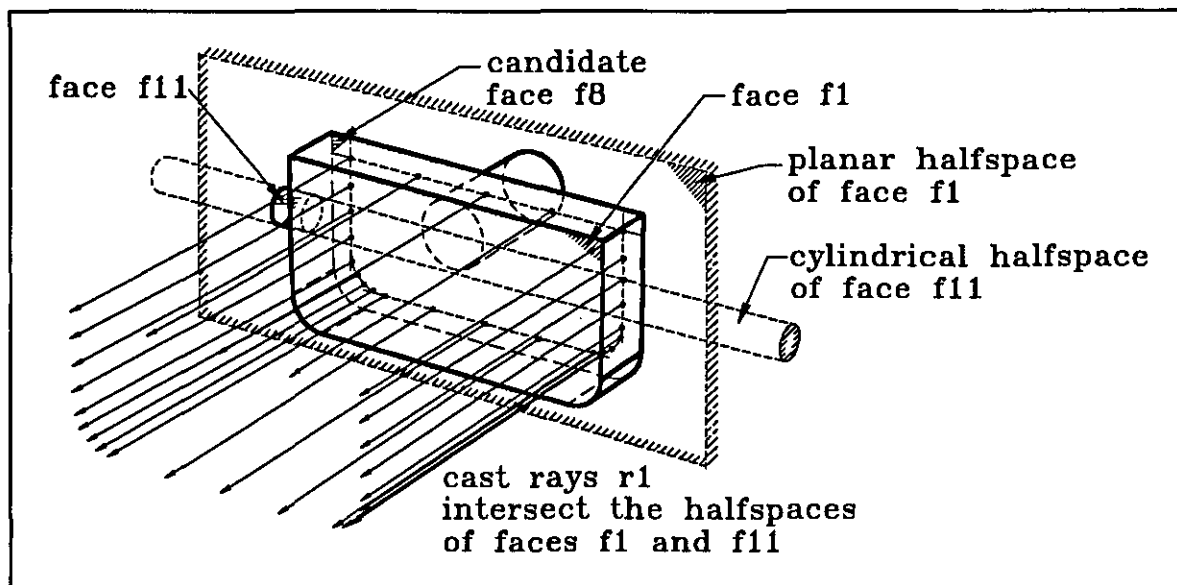


Figure 5.9 : The half-spaces intersected by rays  $r_1$  projected from face  $f_8$ .

Each intersection point  $p$  formed on an intersection half-space is further tested to see if the intersection point lies inside or outside the bounded region of the face belonging to the intersection half-space. This is done by means of a line/polygon intersection test [Tilove80, Tilove81], which again involves the casting of a semi-infinite ray  $r_2$  from the intersection point  $p$  across the face boundary edges (Fig. 5.10). The principle of the line/polygon intersection test is presented in Appendix D.

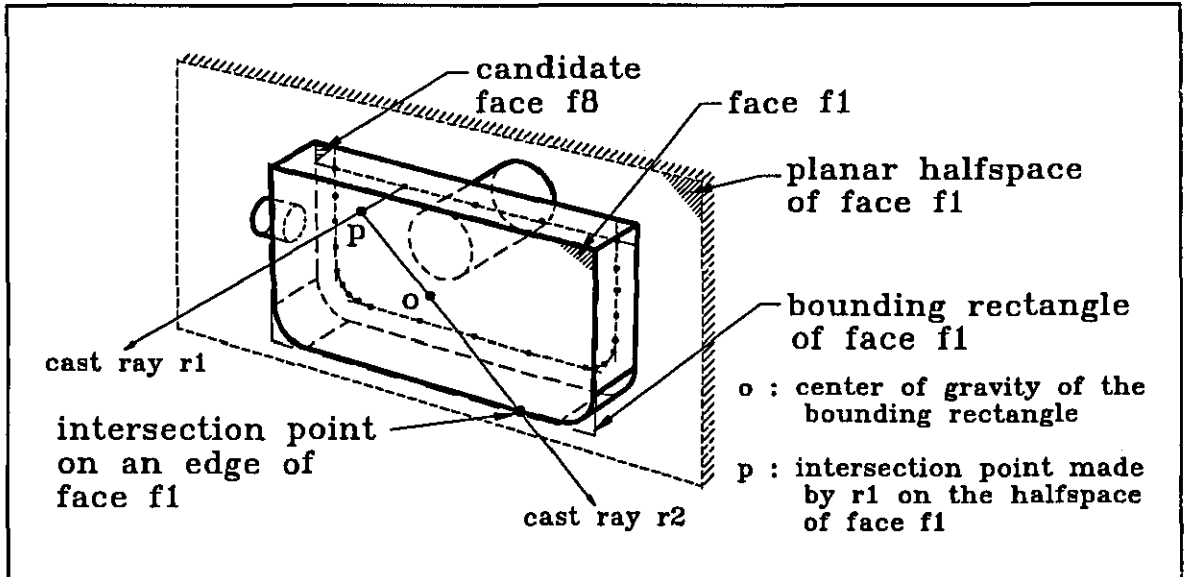


Figure 5.10 : Cast rays  $r_2$  projected from point  $p$  across the boundary of face  $f_1$ .

If an intersection half-space is planar, ray  $r_2$  passes through a point  $o$  which is the centre of gravity of a rectangle that bounds the face belonging to the intersection half-space (Fig. 5.10). If an intersection half-space is cylindrical, there will be two intersection points  $p$  for each cast ray  $r_1$ . The rays  $r_2$  are cast such that they are parallel to the axis of the cylindrical half-space (Fig. 5.11).

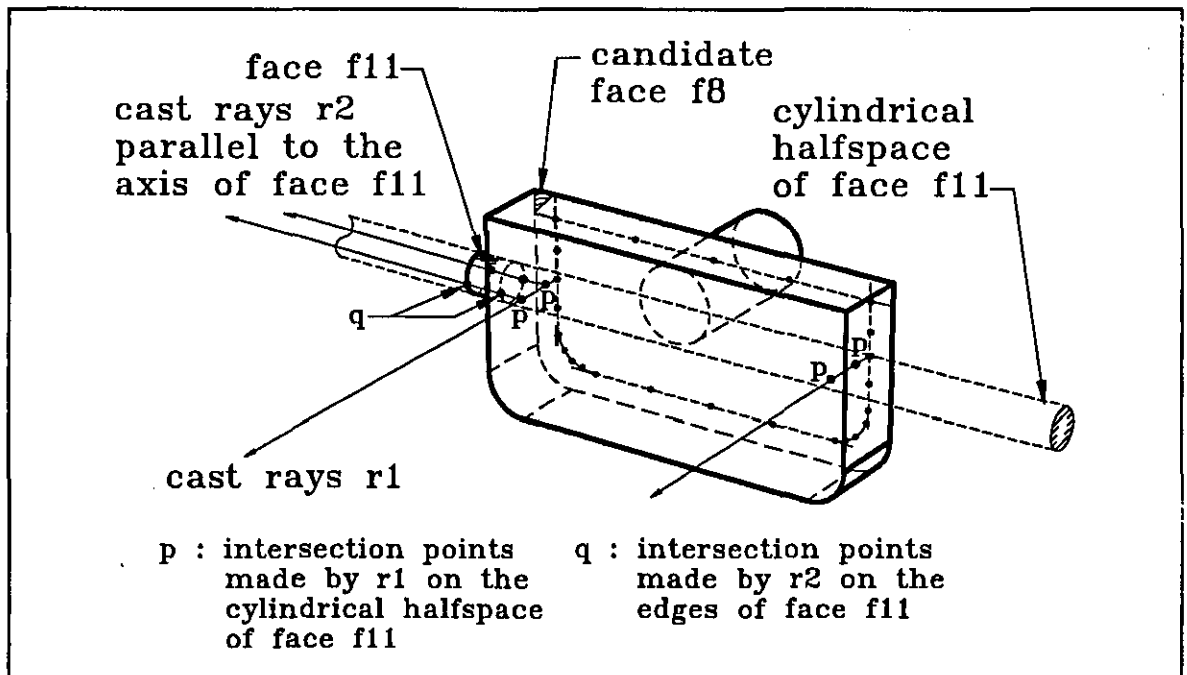


Figure 5.11 : Cast rays  $r_2$  projected from point  $p$  across the boundary of face  $f_{11}$ .

The boundary edges of the intersection face are then tested to see if they intersect with the cast ray  $r2$ . The number of intersection points made by a cast ray  $r2$  on the boundary edges of the intersection face are counted. If the number of intersection points are even, then the intersection point  $p$  lies outside the intersection face, otherwise it lies inside the intersection face. The following rules are used to handle the different results of  $p$  :

- If**  $p$  lies outside an intersection face  $f$ ,
- And If**  $f$  is planar,  
the nature of  $f$  is machined\_face, and  
the value of the status attribute of  $f$  is 'part\_face',
- Then**  $f$  is a secondary\_top\_entrance\_face for the candidate face test, and  
record  $f$  in the working list  $B$ ,
- Then**  $f$  does not obstruct a cutter to access the candidate face, and  
continue the geometric reasoning,
- Else If**  $p$  lies inside an intersection face  $f$ ,  
 $f$  is the only intersection face, and  
the nature of  $f$  is tool\_entrance\_face,
- Then**  $f$  is the primary\_top\_entrance\_face for the candidate face,  
record  $f$  in the working list  $A$ , and  
continue the geometric reasoning,
- Else If**  $p$  lies inside an intersection face  $f$ , and  
the nature of  $f$  is machined\_face,
- Then**  $f$  obstructs a cutter to reach the candidate face, and  
terminate the geometric reasoning.

As illustrated in Fig. 5.11, the intersection point  $p$  lies outside the cylindrical face  $f11$ . By the first rule above, face  $f11$  does not cause cutter interference and the geometric test continues. Fig. 5.10 also shows that the intersection point  $p$  lies inside



face *f1*. By the second rule above, face *f1* is the `primary_top_entrance_face` for the candidate face *f8*, and hence face *f1* is recorded in a working list *A*.

The line/surface and line/polygon intersection tests are performed on all the sampling points. If a candidate face has multiple `primary_top_entrance_faces` that are connected together in the form of a patch of faces (as illustrated previously in Fig. 4.11), the patch of `primary_top_entrance_faces` can also be detected by the cast rays *rI*. Similarly, if a candidate face has several `secondary_top_entrance_faces`, they can also be detected by the cast rays *rI*.

For the current example, the intersection points lie outside face *f11* and inside face *f1*, implying that the entire test on the candidate face is successful. As there are six offset edges on the candidate face *f8* and five sampling points per edge used in the implementation, there would have been 30 sampling points to test. However, if two offset edges intersect each other, their end sampling points overlap. To avoid testing of overlapping sampling points, the overlapped points are sorted out before carrying out the line/surface and line/polygon intersection tests. Thus, in the current example, only 24 intersection points on the half-space of face *f1* are actually tested. This also implies that the identity (integer pointer) of face *f1* is recorded 24 times in the working list *A*. Hence, when the test is successfully completed, an operation is performed to eliminate duplicate integers in the working lists *A* and *B* so as to ensure that they contain only unique integers. For the current example, the working list *A* contains only one integer pointer of face *f1* after the duplicate integer elimination operation. This means that face *f1* is the only `primary_top_entrance_face` that can be used by a cutter to reach the candidate face *f8*.

As face *f11* is cylindrical, it cannot completely satisfy the above first rule. This means that face *f11* does not cause cutter interference but is not a `secondary_top_entrance_face` for the candidate face *f8*. So the working list *B* remains empty at the end of the test.

### 5.3.3.3 The Third Geometric Test for the Group(1) Faces

A candidate face that has passed the above first and second geometric tests only means : (1) that the candidate face can be used as the part\_face of a machining feature, and (2) the candidate face can be locally accessible by a cutter. However, criterion (4) actually requires that a recognized machining feature should also be globally accessible. A globally accessible machining feature is considered as a locally accessible machining feature whose cutter access path is also not obstructed by any protrusions or overhangs of the part as well as other possible obstacles in the machining environment such as clamping and locating devices. An example of a locally accessible but not globally accessible machining feature is illustrated in Fig. 2.11. In this thesis, the machining environment is not considered. The extent of global accessibility is confined to consideration of the part shape.

The global accessibility analysis is performed in the third test, where the cast rays  $r1$  used in the second test are tested for intersection with the half-spaces of the part. Like the second test, a preliminary test is performed to reduce the number of half-spaces required for the line/surface intersection test. For the current example, the following faces (please refer to Fig. 5.12 for face notations) are exempted from the line/surface intersection test as there will be no intersection point formed due to the reasons explained :

Faces	Reason for Exemption from the Test
fb	its half-space is complementary to the half-space of the candidate face f8 as described by expression(7) in section 4.3,
fc, fe, fg, fl, fm, fn, fo	their planar surface normals are perpendicular to the cast rays $r1$ ,
fd, ff, fh, fi	their cylindrical surface axes are parallel to the cast rays,
fj and fp	they lie behind the origins of the cast rays.

Table 5.1 : Faces exempted from the intersection test.

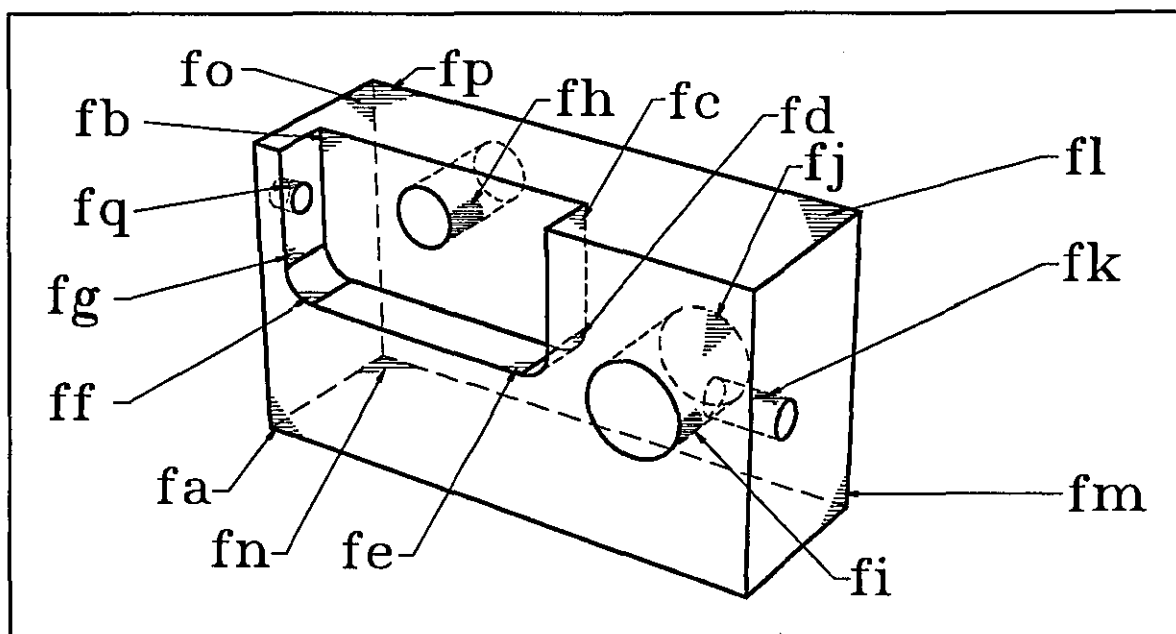


Figure 5.12 : The boundary faces of the hypothetical part.

Thus, the line/surface intersection test is performed only on faces  $fa$ ,  $fk$  and  $fq$ . As illustrated in Fig. 5.13, the cast rays  $r1$  intersect the half-spaces of faces  $fa$ ,  $fk$  and  $fq$ .

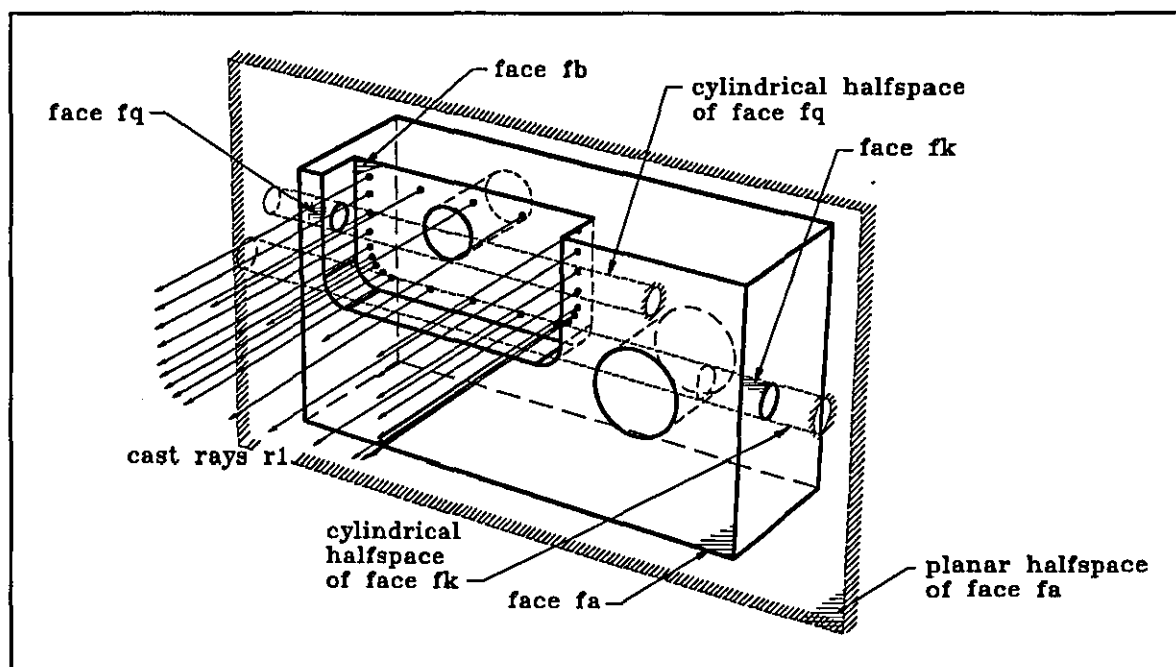


Figure 5.13 : Cast rays  $r1$  intersect with the relevant halfspaces of the part.

For each of the three intersection faces, a line/polygon intersection test is performed in a way similar to that used in the second test for the purpose of determining whether the intersection point  $p$  lies inside or outside the intersection face. For instance, the line/polygon intersection test for the intersection face  $fa$  is illustrated in Fig. 5.14.

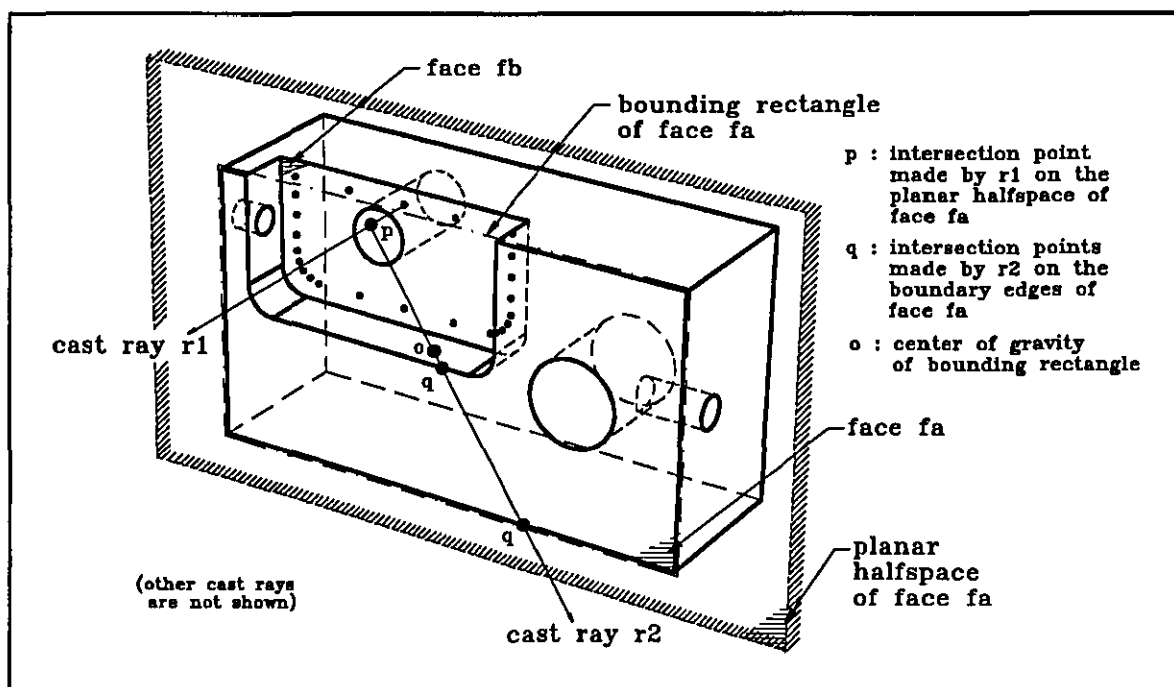


Figure 5.14 : Cast ray  $r2$  projected from point  $p$  across the boundary of face  $fa$ .

The following rules are used to handle the different results of  $p$  :

- If**  $p$  lies outside an intersection face  $f$ ,  
**Then**  $f$  does not obstruct a cutter to access the candidate face, and continue the geometric reasoning,  
**Else If**  $p$  lies inside an intersection face  $f$ ,  
**Then**  $f$  obstructs a cutter to access the candidate face, and terminate the geometric reasoning,  
**Else** computational error, and terminate the geometric reasoning.

As can be seen in the above rules, the decision for analyzing the 'in/out' conditions of  $p$  is straightforward since the intersection faces in this third test are the boundary faces of the finished part rather than those of the subvolume.

With the illustration in Fig. 5.13, it can be perceived that the intersection points lie outside faces  $f_a$ ,  $f_k$  and  $f_q$ , implying that the three faces do not cause cutter interference. In summary, the three tests for the candidate face  $f_8$  are successful. So the algorithm moves on to utilize the geometric reasoning results.

#### 5.3.4 Utilization of the Group(1) Face Testing Results

The geometric reasoning results are handled according to the following rule :

- If** either one of the above three geometric tests fails,
- Then** the value of the access attribute of the candidate face  $f$  is changed from the string 'nil' to the integer '0',
- Else** (a) generate the following feature record and top entrance face lists :
- record heading : machining feature
  - fields 1-3 : the cutter axis vector, i.e. cast ray  $rI$
  - field 4 : a constituent edge  $e$  of the outer edge loop of  $f$
  - field 5 : a pointer to a primary\_top\_entrance face list
  - field 6 : a pointer to a secondary\_top\_entrance\_face list
- (b) primary\_top\_entrance\_face list = linear list  $A$
- (c) secondary\_top\_entrance\_face list = linear list  $B$
- (d) insert the above feature record in the machining feature list of  $f$ ,
- (e) change the value of the access attribute of  $f$  from 'nil' to  $f$ ,
- (f) change the value of the status attribute of  $f$  from 'nil' to the string 'part\_face',
- (g) change the value of the status attribute of the adjacent faces of  $f$  according to the following rule :

**If** the value of the status attribute of a face  $f$  is 'part\_face',  
 edge  $e$  is an edge belonging to the outer edge loop of  $f$ ,  
 $e$  is convex,  
 the value of the status attribute of  $e$  is 'nil',  
 the adjacent faces of  $e$  are  $f$  and  $g$ , and  
 the value of the status attribute of  $g$  is 'nil',  
**Then** change the value of the status attribute of  $g$  to the string  
 'check\_face'.

In the B-rep database of the cavity volume, every face record has a field assigned for storing a pointer to a machining feature list. When the above three geometric tests are satisfied, a feature record is created and appended in the machining feature list of the candidate face. More details about the B-rep database is described in chapter 7.

As the current candidate face  $f_8$  satisfies the three tests, the above rule records the recognition of a valid machining feature in the B-rep database. The rule also transforms the initial face/edge graph or problem states shown in Fig. 5.4 to the problem states shown in Fig. 5.15.

Having completed the first problem state transformation, the algorithm recurs to use the rule stated in section 5.3.2 to select the next candidate face from the machined\_face list for geometric reasoning. However, for the current example, no more faces from the machined\_face list can be selected as the value of the status attribute of faces  $f_4$ ,  $f_2$ , and  $f_6$  has been changed to 'check\_face', while the surface type of faces  $f_3$ ,  $f_5$ ,  $f_9$ , and  $f_{11}$  is cylindrical (Fig. 5.15). So the algorithm proceeds to consider the group(2) faces.

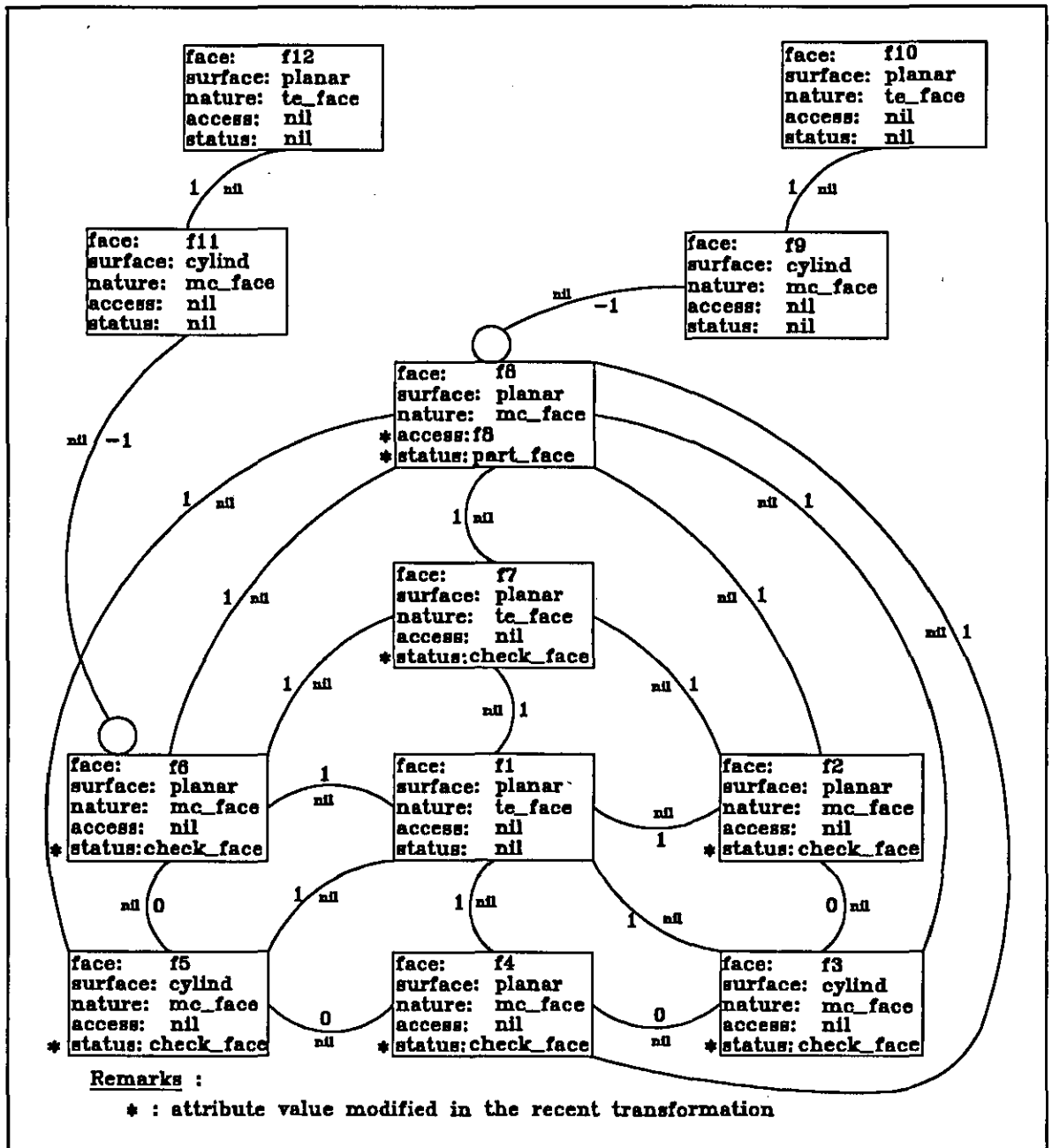


Figure 5.15 : The first transformation of the face/edge graph of subvolume\_1.

### 5.3.5 Selection of the Group(2) Faces

For analyzing the group(2) candidate faces, the tool\_entrance\_face list is relevant. The following rule is used to select a candidate face from the tool\_entrance\_face list :

**If**  $f$  is a face to be selected from the `tool_entrance_face` list,  
the value of the access attribute of  $f$  is not zero, and  
the value of the status attribute of  $f$  is neither 'part\_face' nor 'check\_face',  
**Then** select  $f$  as the candidate face.

With the `tool_entrance_face` list {f1, f10, f12, f7} and the problem states shown in Fig. 5.15, the above rule selects face f1 as the candidate face.

### 5.3.6 Geometric Reasoning for the Group(2) Faces

For analyzing the group(2) candidate faces, two major tests similar to the previous second and third geometric tests are conducted to ensure that the criteria (3) and (4) described in section 5.1 are satisfied. The previous first geometric test is not used because criterion (1) is not necessary for group(2) faces since the intersection between the `part_face` and its adjacent `check_faces` may not be orthogonal such as the case illustrated in Fig. 4.5.

#### 5.3.6.1 The First and Second Geometric Tests for the Group(2) Faces

The procedures employed in the first and second geometric test are essentially the same as those described in sections 5.3.3.2 and 5.3.3.3 respectively. However, as the intersection between the candidate face and its adjacent faces may not be orthogonal, the direction of the cast rays  $rI$  is determined as follow :

**If** the candidate face  $f$  has an adjacent cylindrical face,  
**Then** the cast rays  $rI$  are parallel to the axis of the cylindrical face and towards the inside of the cavity volume as illustrated in Fig. 5.16(a),  
**Else** the cast rays are parallel to a linear edge that is shared between two planar adjacent faces and towards the inside of the cavity volume as illustrated in Fig. 5.16(b).



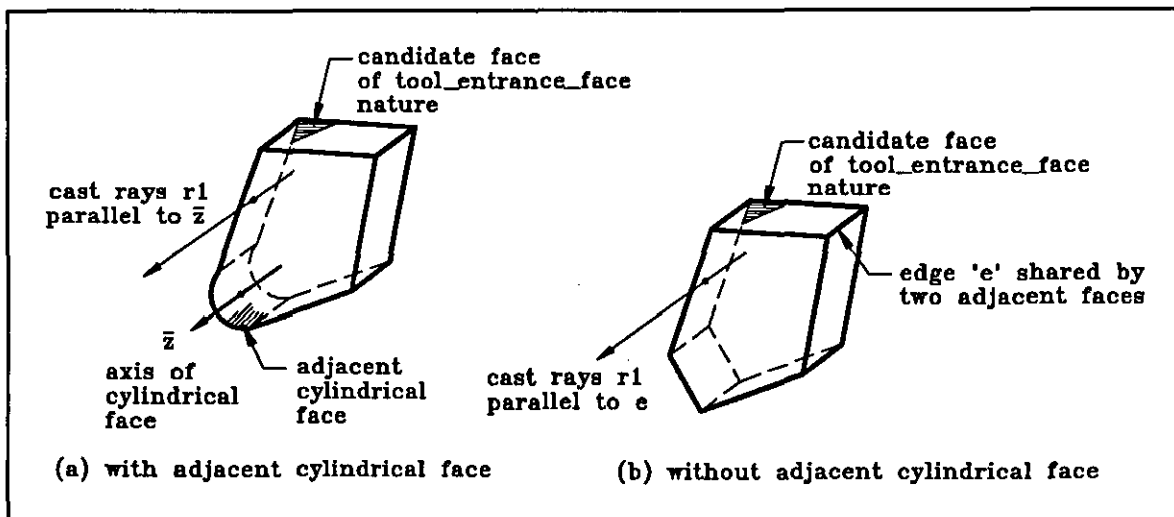


Figure 5.16 : Determining the projection direction of cast rays  $r1$  for group (2) faces.

As illustrated in Fig. 5.17, the very first intersection point of the cast ray  $r1$  is on the face  $f8$  which is of `machined_face` nature. This implies that cutter access to the candidate face  $f1$  is blocked by face  $f8$ . So by the last rule stated in section 5.3.3.2, the geometric reasoning for the candidate face  $f1$  stops.

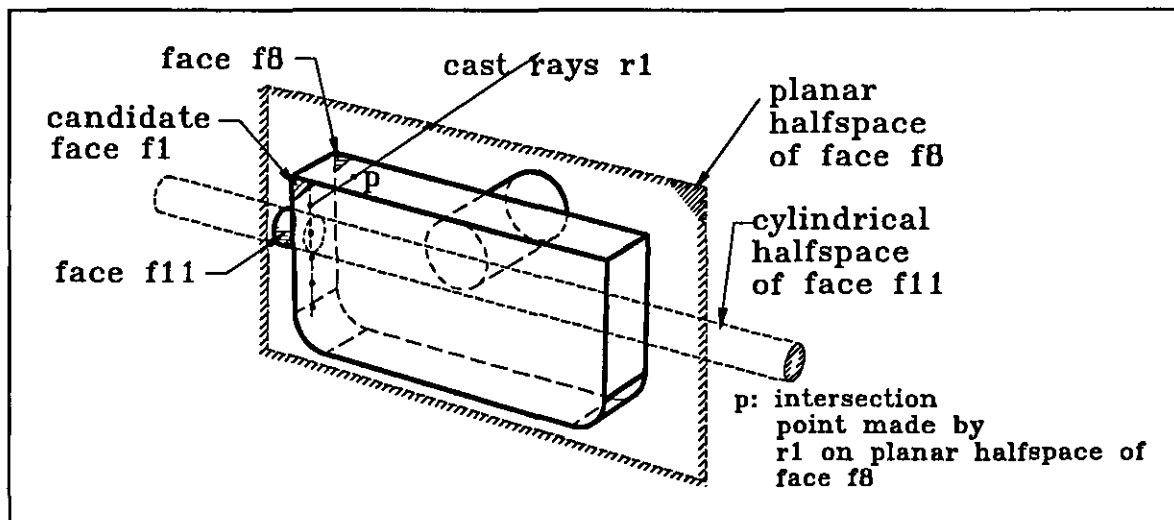


Figure 5.17 : Cast rays  $r1$  projected from the surface of face  $f1$ .

## 5.3.6.2 Utilization of the Group(2) Face Testing Results

The geometric reasoning results for the group(2) faces are handled similarly as described in section 5.3.4. Since the candidate face f1 fails the first geometric test, the value of its access attribute is changed from nil to zero. As a result, the problem space shown in Fig. 5.15 is transformed to that shown in Fig. 5.18.

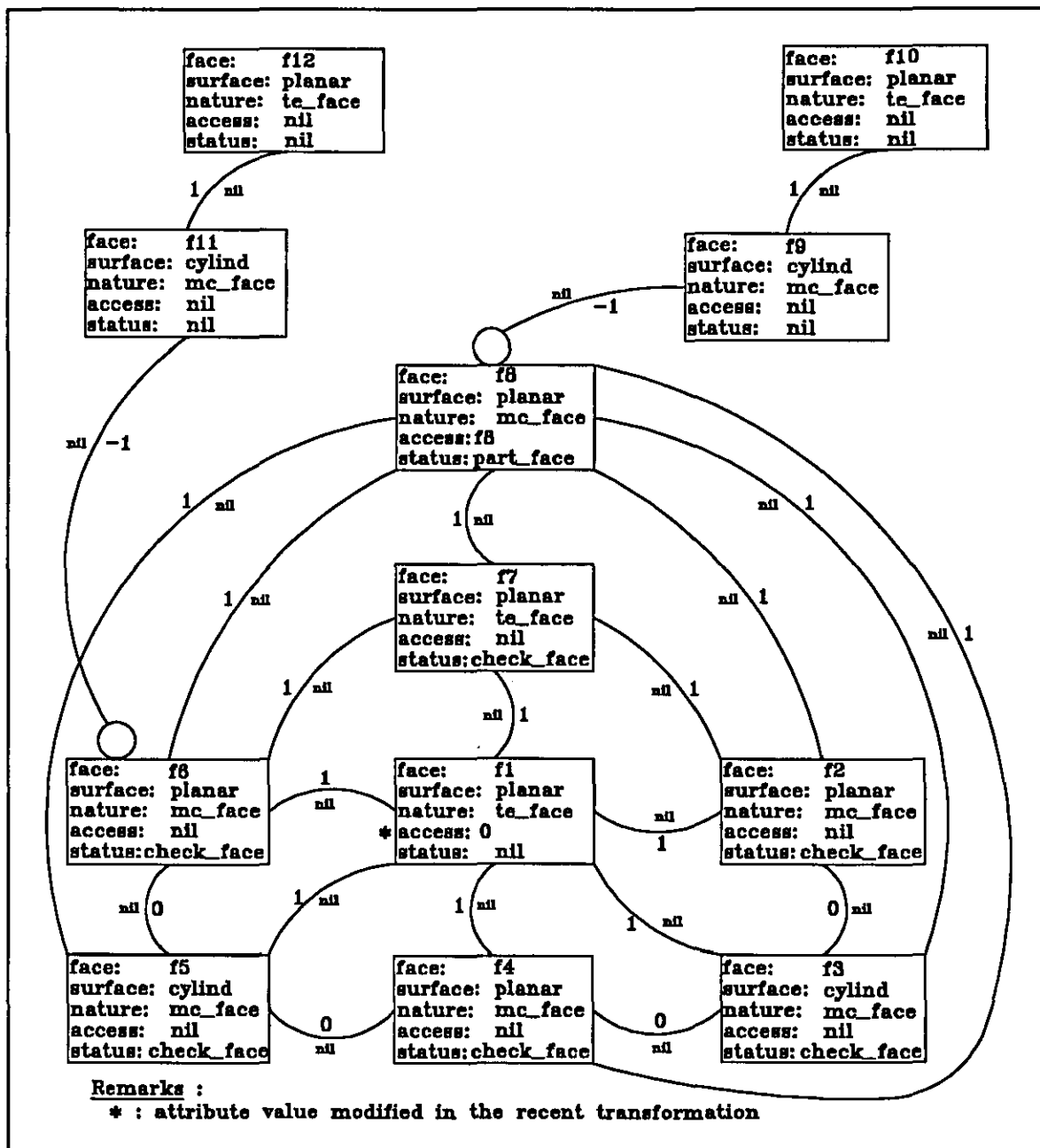


Figure 5.18 : The second transformation of the face/edge graph of subvolume\_1.

### 5.3.6.3 Analysis of the Remaining Group(2) Faces

The algorithm loops back to use the rule stated in section 5.3.5 to select the next candidate face from the `tool_entrance_face` list. The second face `f10` in the list satisfies the rule and is therefore chosen for geometric reasoning. The relevant half-spaces involved in the first and second geometric tests of the candidate face `f10` are illustrated in Fig. 5.19.

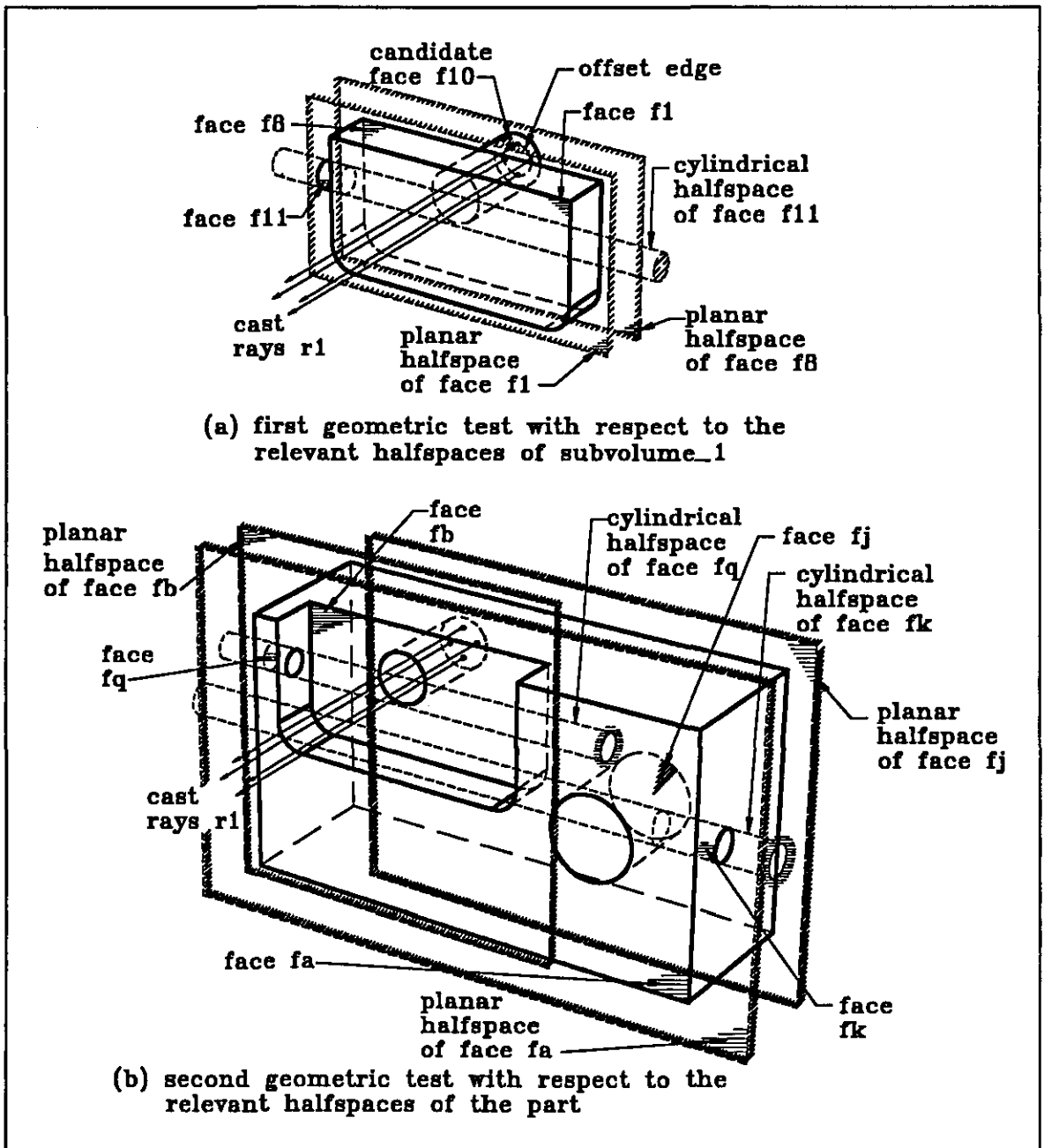


Figure 5.19 : The first and second geometric tests performed on face `f10`.

With the illustration in Fig. 5.19, it can be understood that face f10 satisfies both the first and second tests. Face f8 is recorded as a `secondary_top_entrance_face` in the working list *B* and face f1 is recorded as a `primary_top_entrance_face` in the working list *A*. By the rule described in section 5.3.4, the problem space shown in Fig. 5.18 is transformed to that shown in Fig. 5.20.

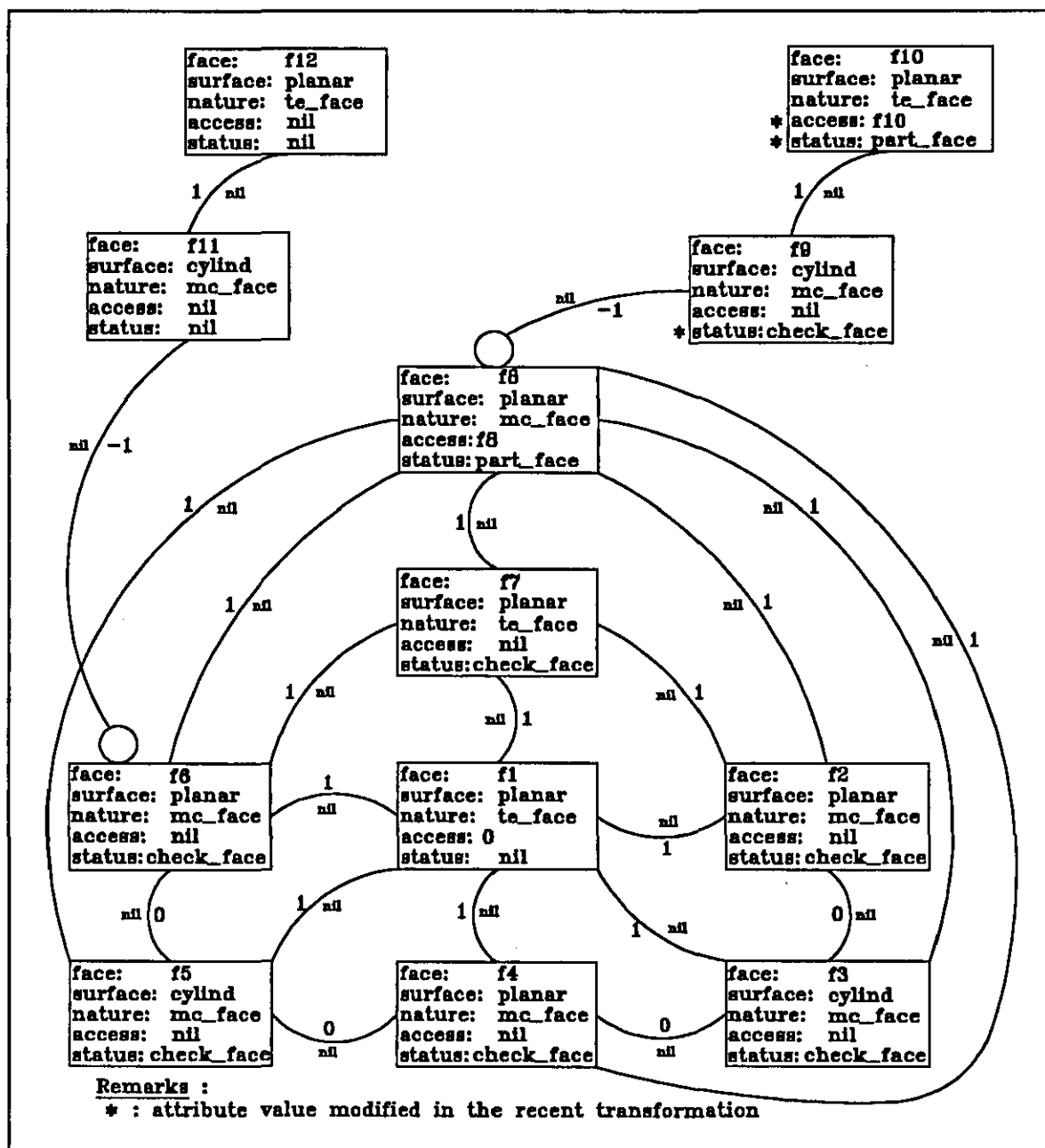


Figure 5.20 : The third transformation of the face/edge graph of subvolume\_1.

The algorithm returns to select another candidate face from the `tool_entrance_face` list. This time face `f12` is chosen as the candidate face. However, face `f12` does not pass the first test as face `f2` causes cutter interference (Fig. 5.21).

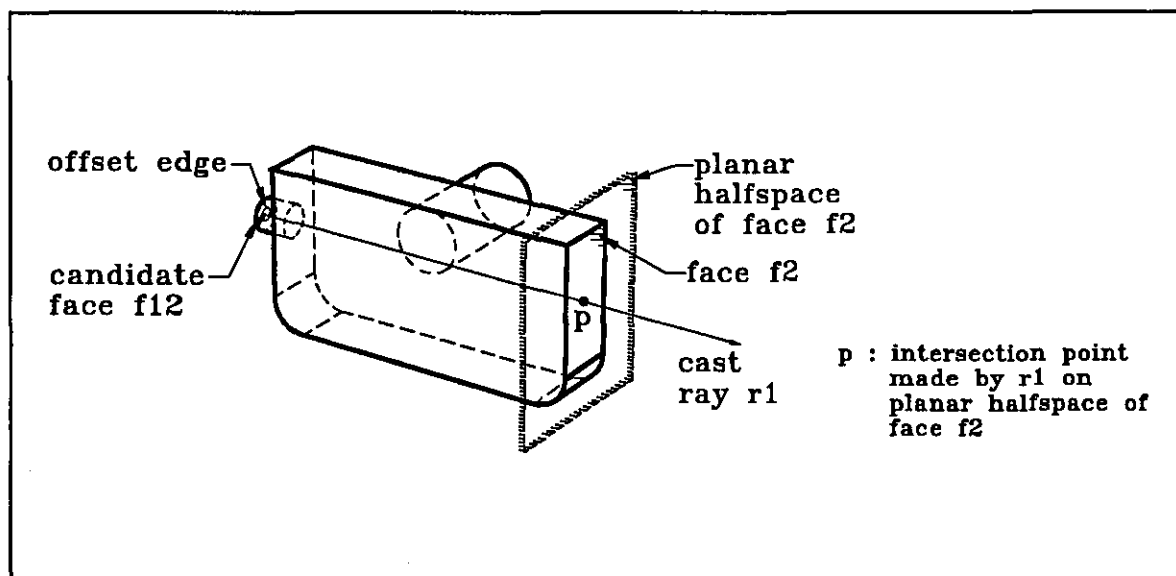


Figure 5.21 : The first geometric test performed on face `f12`.

Consequently, the problem space shown in Fig. 5.20 is transformed to that shown in Fig. 5.22.

The algorithm attempts to select the last face `f7` from the `tool_entrance_face` list. However, face `f7` does not satisfy the selection rule stated in section 5.3.5 because its access attribute value has been modified to 'check\_face' in the previous transformation (Fig. 5.15). As there are no more selectable candidate faces in the `tool_entrance_face` list, the algorithm directs the focus of interest on the group(3) faces.

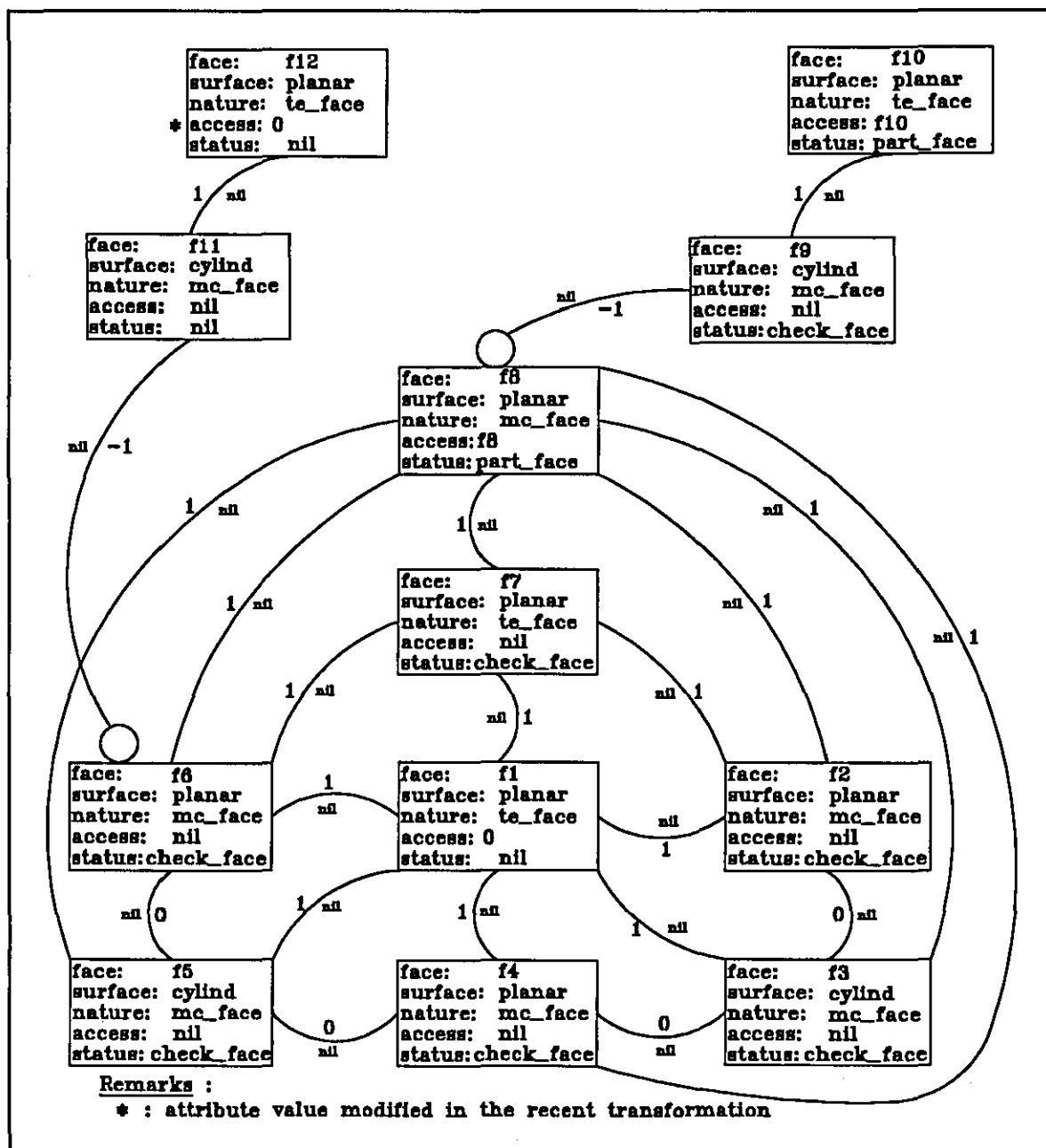


Figure 5.22 : The fourth transformation of the face/edge graph of subvolume\_1.

### 5.3.7 Selection of the Group(3) Faces

After performing geometric reasoning on the groups(1) and group(2) faces, the occurrence of the group(3) faces will be on those faces that are of machined\_face nature, have inner edge loop, and the status attribute value has been modified to 'check\_face' or 'part\_face' during the previous tests.

Since the nature of the group(3) faces is `machined_face`, the `machined_face` list {f8, f4, f2, f6, f3, f5, f9, f11} can be used again as an agenda for governing the selection sequence of the candidate faces. The following rule is used to select the group(3) candidate faces for geometric reasoning :

**If**  $f$  is a face to be selected from the machine face list,  
the value of the status attribute of  $f$  is either 'check\_face' or 'part\_face',  
 $f$  has an inner edge loop  $el$ ,  
the status of the constituent edges of  $el$  is nil,  
the adjacent faces of the constituent edges are  $gs$ , and  
the value of the status attribute of  $gs$  is not 'check\_face',  
**Then** select  $f$  as a candidate face.

Based on the problem states shown in Fig. 5.22, the two `machined_faces` that have inner edge loop are faces f8 and f6. Face f8 does not satisfy the above rule because the adjacent face of its inner edge is f9 whose status attribute value has been modified to 'check\_face'. Face f6, however, satisfies the above rule, and so it is chosen as a candidate face for geometric reasoning.

### 5.3.8 Geometric Reasoning for the Group(3) Faces

For analyzing the group(3) candidate faces, two major geometric tests similar to the two tests used for the groups(2) faces are conducted to ensure that the criteria (3) and (4) described in section 5.1 are satisfied. The first geometric test used for the group(1) faces is also not used here. This is because criterion (1) is not obligatory for group(3) faces since the intersection between the `part_face` and the adjacent `check_faces` may not be orthogonal. The first and second tests for the candidate face f6 is illustrated in Fig. 5.23. Both the first and second tests are successful. During the first test, face f12 is detected as a `primary_top_entrance_face`.

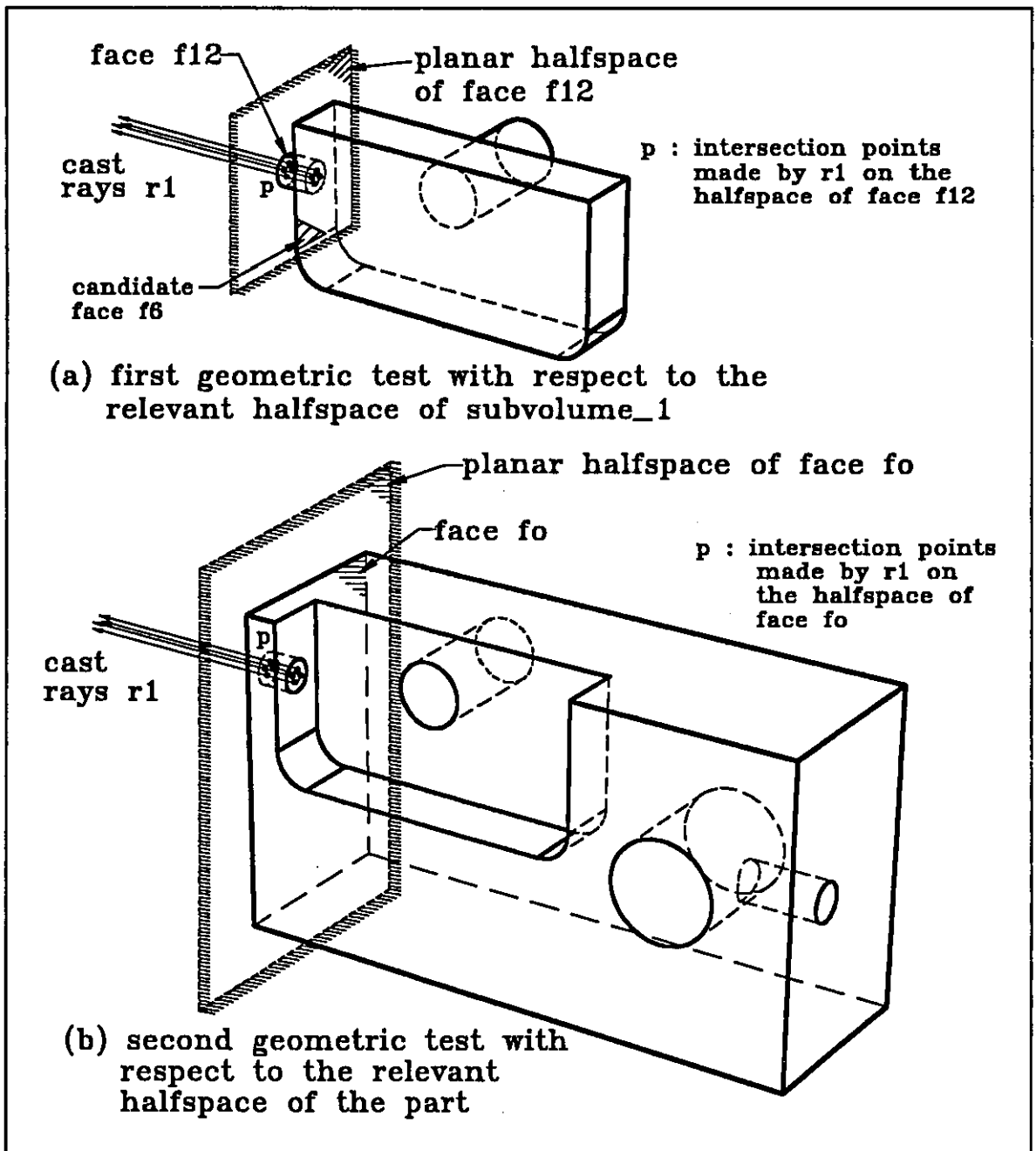


Figure 5.23 : The first and second geometric tests performed on face  $f_6$ .



### 5.3.9 Utilization of the Group(3) Face Testing Results

For the group(3) faces, the geometric reasoning results are treated according to the following rule :

- If** either one of the two geometric tests fails,
- Then** change the value of the access attribute of the candidate face  $f$  to zero, and change the status of the constituent edges of the inner edge loop  $el$  of  $f$  from 'nil' to the string 'marked',
- Else** (a) generate the following feature record and top entrance face lists :
- record heading : machining feature
  - fields 1-3 : the cutter axis vector, i.e. cast ray  $r1$
  - field 4 : a constituent edge  $e$  of the inner edge loop  $el$
  - field 5 : a pointer to a primary\_top\_entrance\_face list
  - field 6 : a pointer to a secondary\_top\_entrance\_face list
- (b) primary\_top\_entrance\_face list = linear list  $A$
- (c) secondary\_top\_entrance\_face list = linear list  $B$
- (d) insert the above feature record in the machining feature list of  $f$ ,
- (e) change the value of the status attribute of the faces adjacent to the constituent edges of  $el$  from 'nil' to the string 'check\_face'.

By the above rule, the problem state shown in Fig. 5.22 is transformed to that shown in Fig. 5.24. As there are no more selectable group(3) faces, the algorithm stops. Fig. 5.24 therefore also represents the final problem space of subvolume\_1.

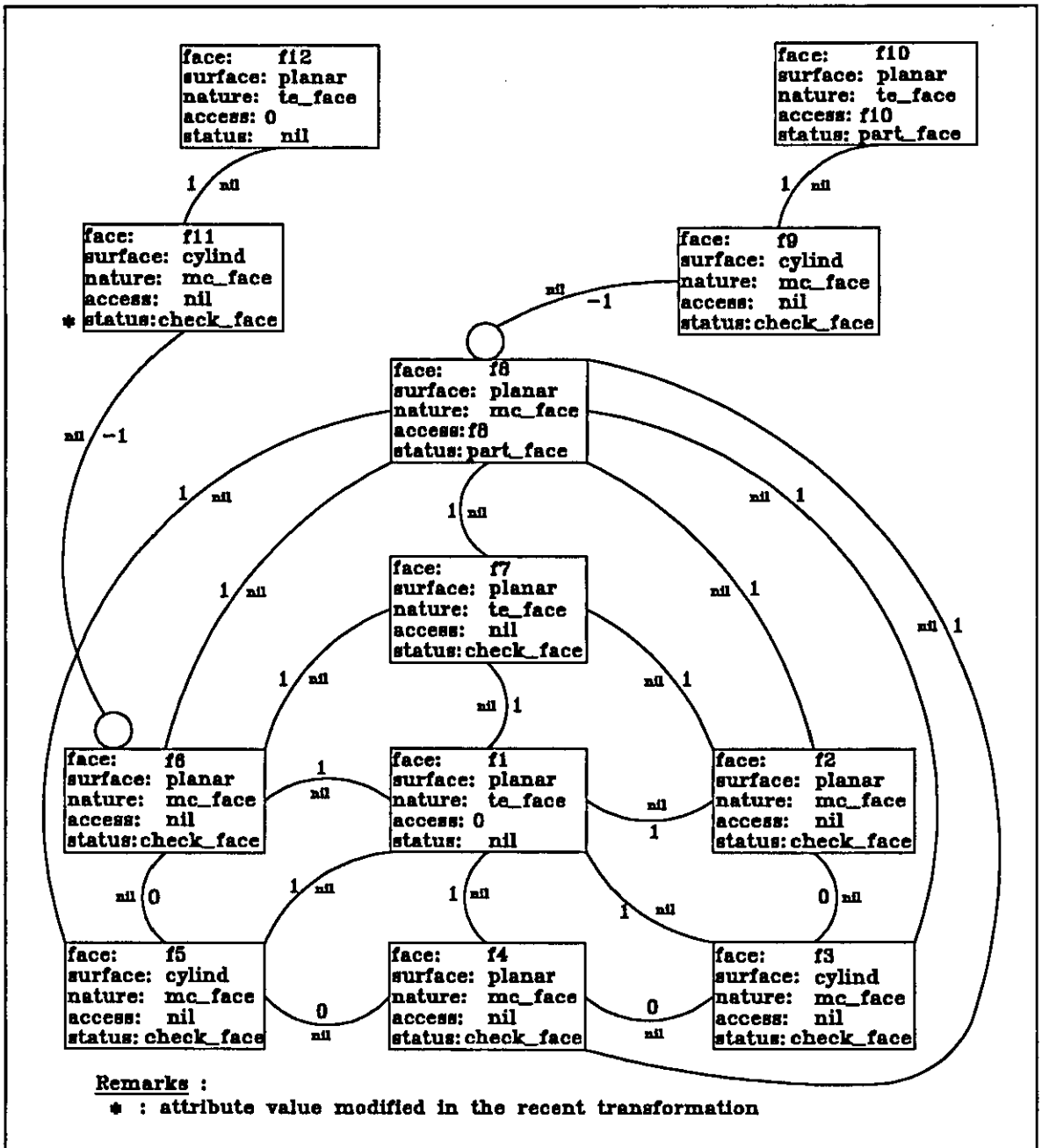


Figure 5.24 : The fifth transformation of the face/edge graph of subvolume\_1.

#### 5.4 Recognition of Machining Features from the Subvolume\_2

The problem space of the subvolume\_2 in terms of the face/edge graph shown in Fig. 5.5 is analyzed similarly by the algorithm. By means of the heuristics-based sorting procedures described in section 5.3.1, the tool\_entrance\_face and the

machined\_face lists are created. The tool\_entrance\_face list contains faces f15 and f17, while the machined\_face list contains faces f13, f14 and f16 (Fig. 5.5). By using the rule stated in section 5.3.2, face f13 is chosen as the first candidate face for geometric reasoning.

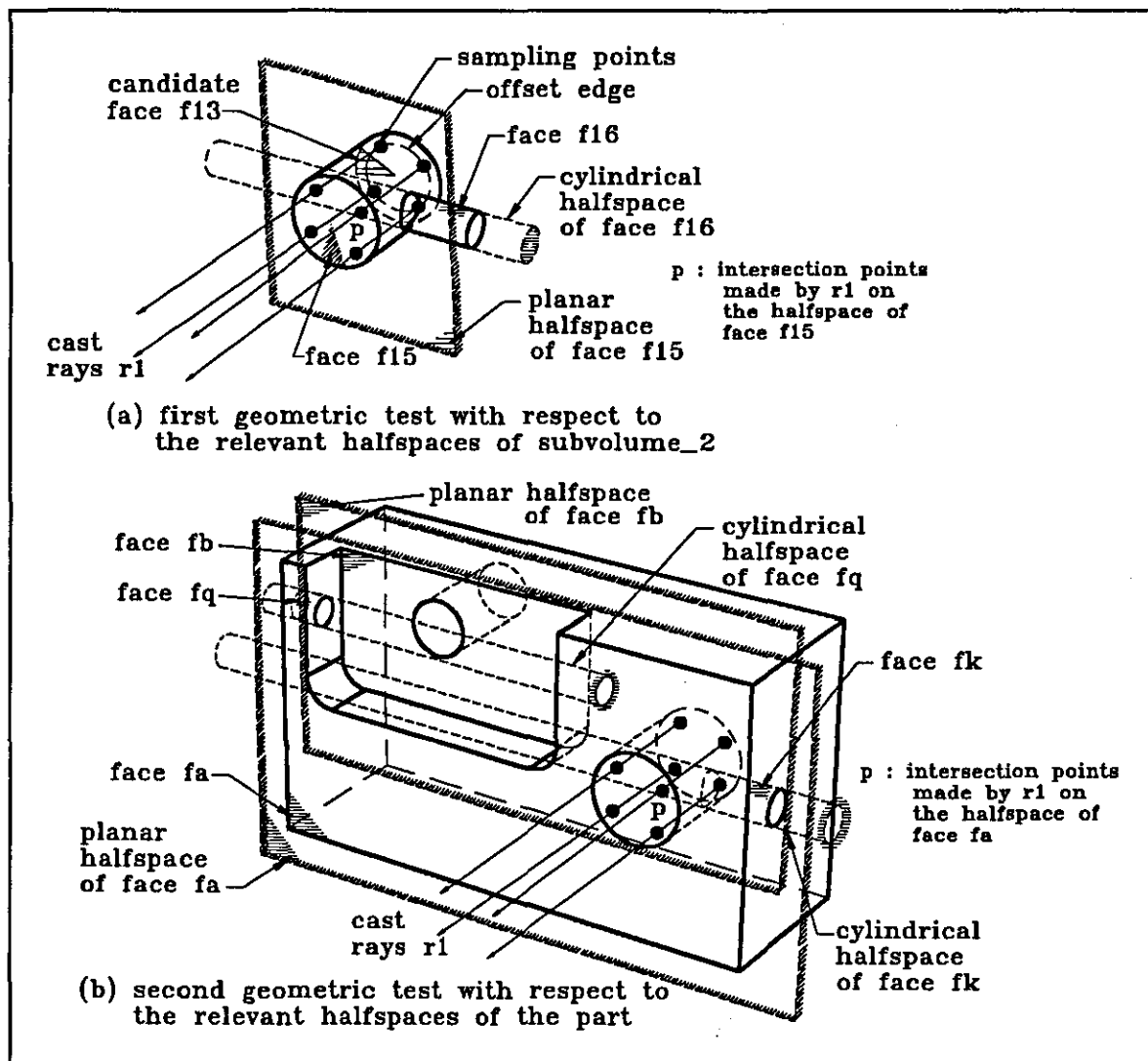


Figure 5.25 : The first and second geometric tests performed on face f13.

Face f13 passes the first test as its surface normal is parallel to the axis of its adjacent cylindrical face f14. The second and third tests for the candidate face f13 are illustrated in Fig. 5.25.

It can be observed that face f16 does not cause cutter interference and face f15 is detected as a `primary_top_entrance_face`. Consequently, a feature record is created and augmented in the machining feature list of the candidate face f13. The problem space shown in Fig. 5.5 is transformed to that shown in Fig. 5.26.

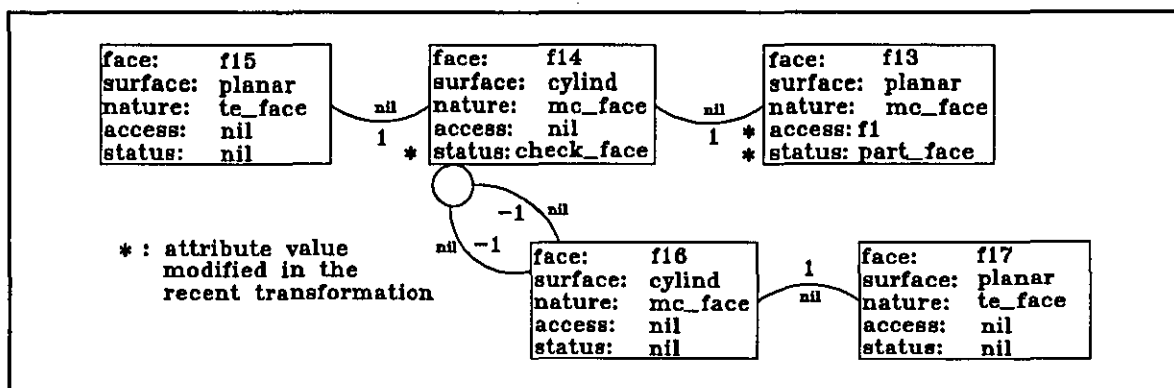


Figure 5.26 : The first transformation of the face/edge graph of subvolume\_2.

As the remaining faces in the `machined_face` list are cylindrical, they do not satisfy the candidate face selection rule stated in section 5.3.2. Thus, the algorithm turns to consider the `group(2)` faces.

According to the candidate face selection rule defined in section 5.3.5, face f15 is first selected from the `tool_entrance_face` list for geometric reasoning. With the illustration in Fig. 5.27, it can be understood that the candidate face f15 fails the first test as described in section 5.3.6.1 due to the fact that face f13 causes cutter interference.

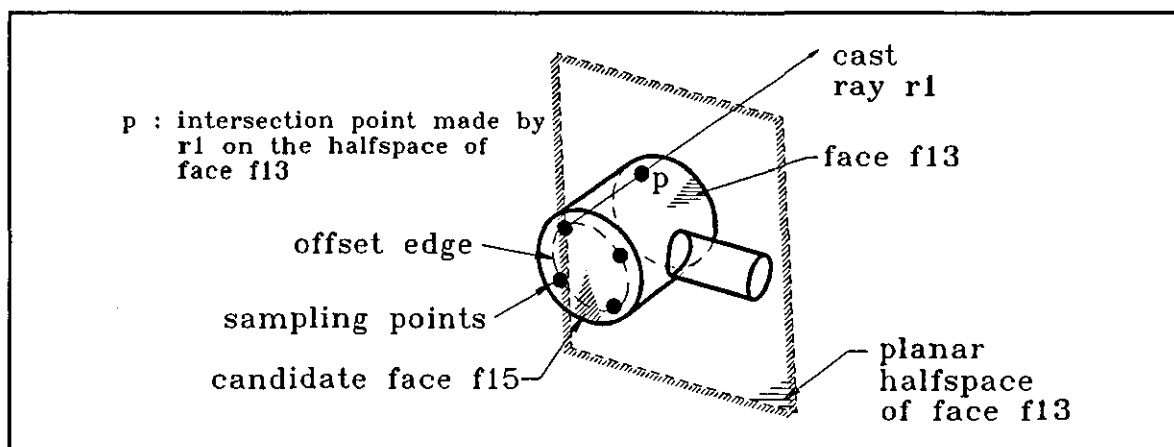


Figure 5.27 : The first geometric test performed on face f15.

The problem space is transformed to that shown in Fig. 5.28.

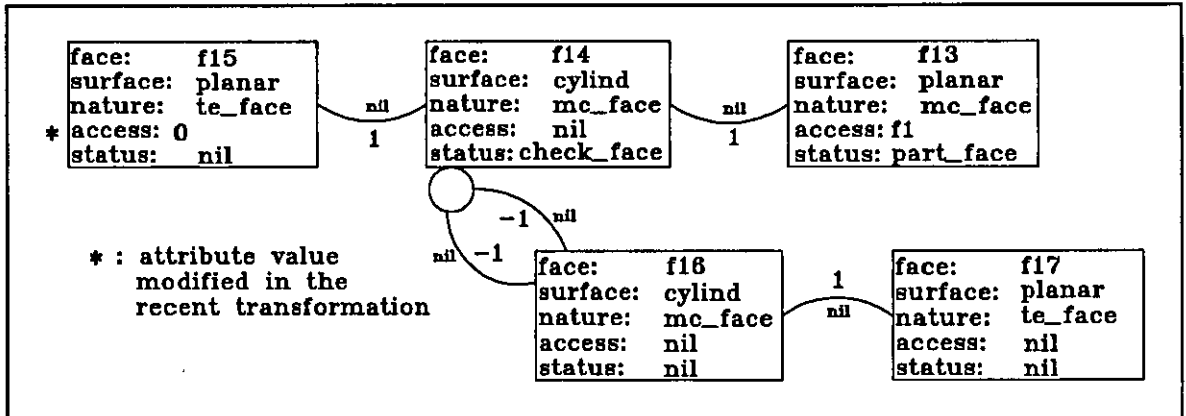


Figure 5.28 : The second transformation of the face/edge graph of subvolume\_2.

The next chosen candidate face f17 also fails the first test as face f14 causes cutter interference (Fig. 5.29).

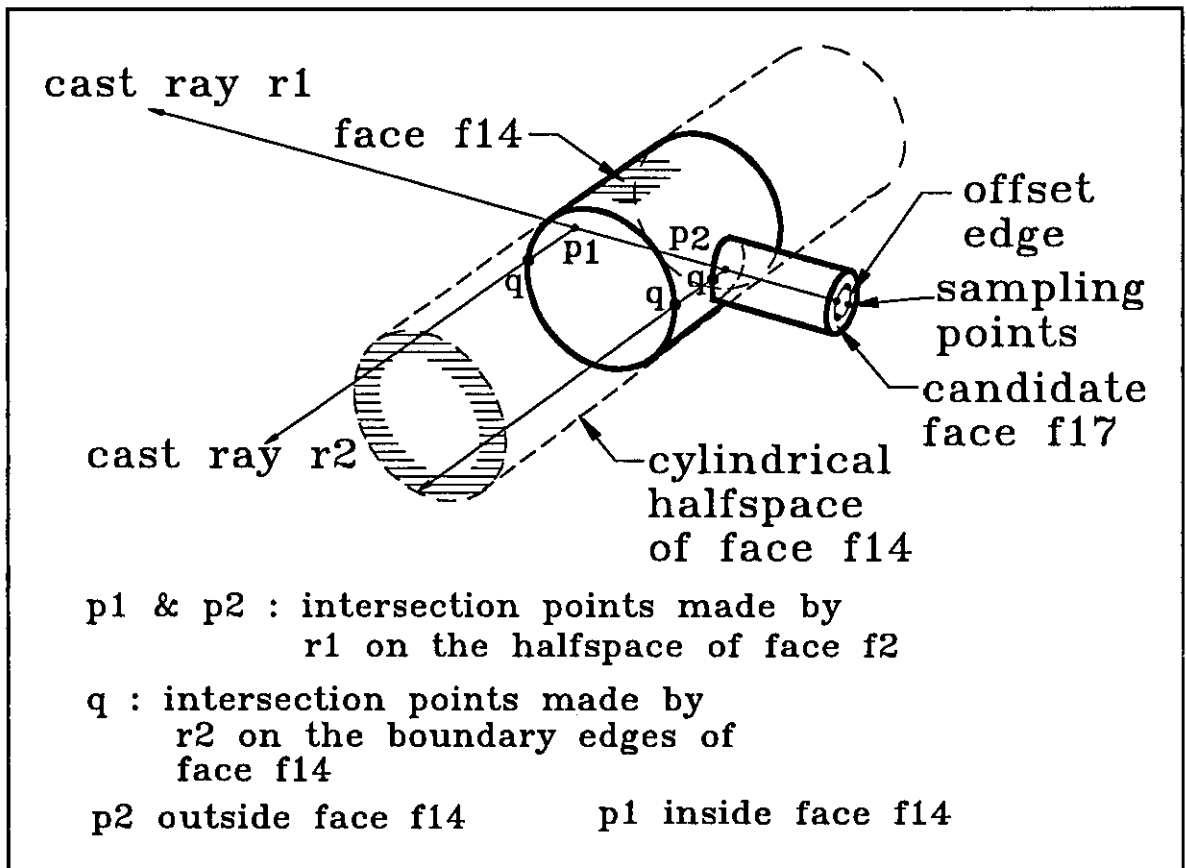


Figure 5.29 : The first geometric test performed on face f17.

As a result, the problem space is transformed to the one shown in Fig. 5.30.

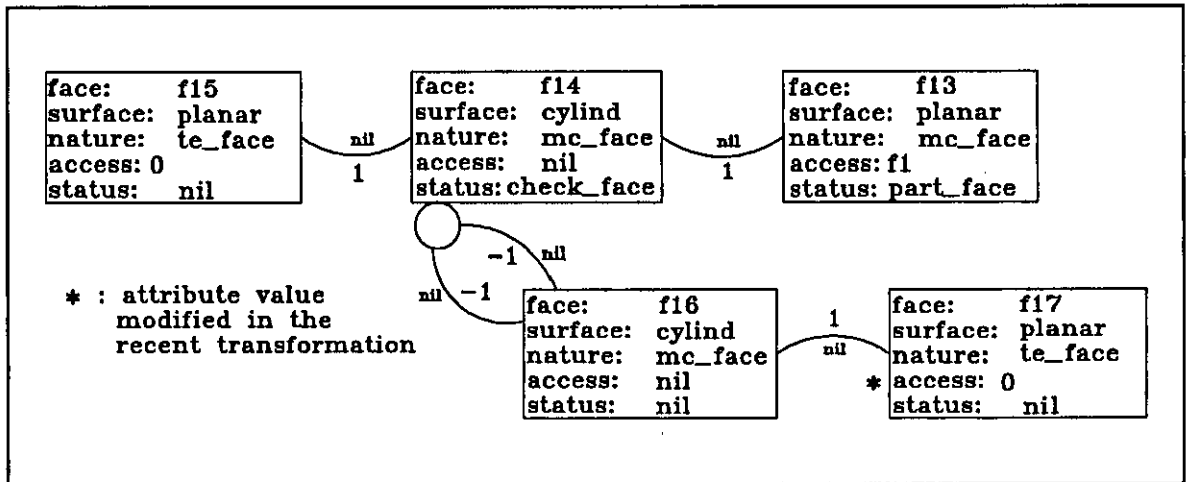


Figure 5.30 : The third transformation of the face/edge graph of subvolume\_2.

The algorithm proceeds to consider group(3) faces. By using the group(3) candidate face selection rule described in section 5.3.7, face f14 is selected for testing. With the illustration in Fig. 5.31, it can be perceived that face f14 passes the two tests and face f17 is detected as a primary\_top\_entrance\_face.

A machining feature record is created and is appended in the machining feature list within the face record of face f14. The problem space is transformed to that shown in Fig. 5.32. The algorithm stops since there are no more selectable group(3) faces.

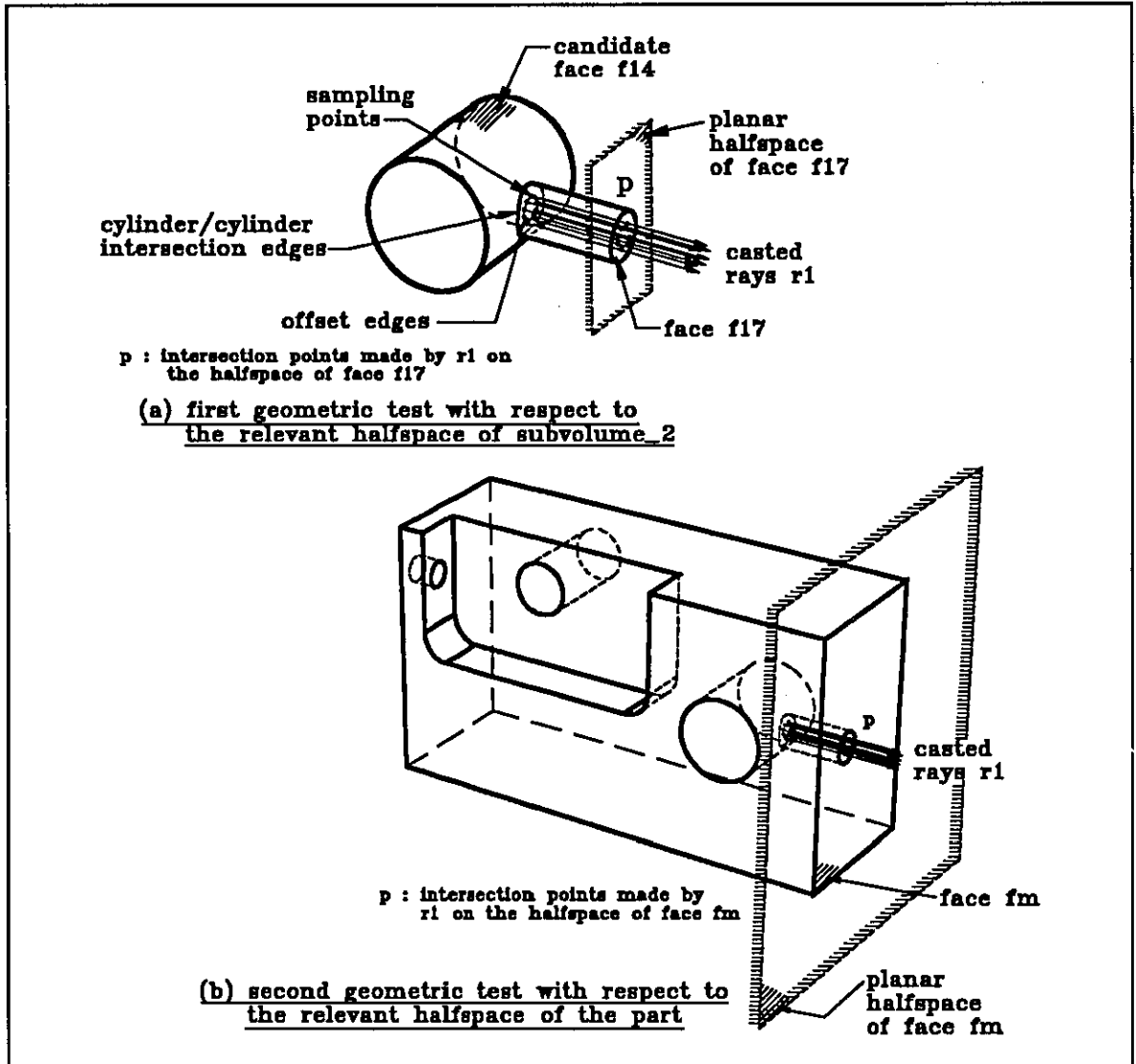


Figure 5.31 : The first and second geometric tests performed on face f14.

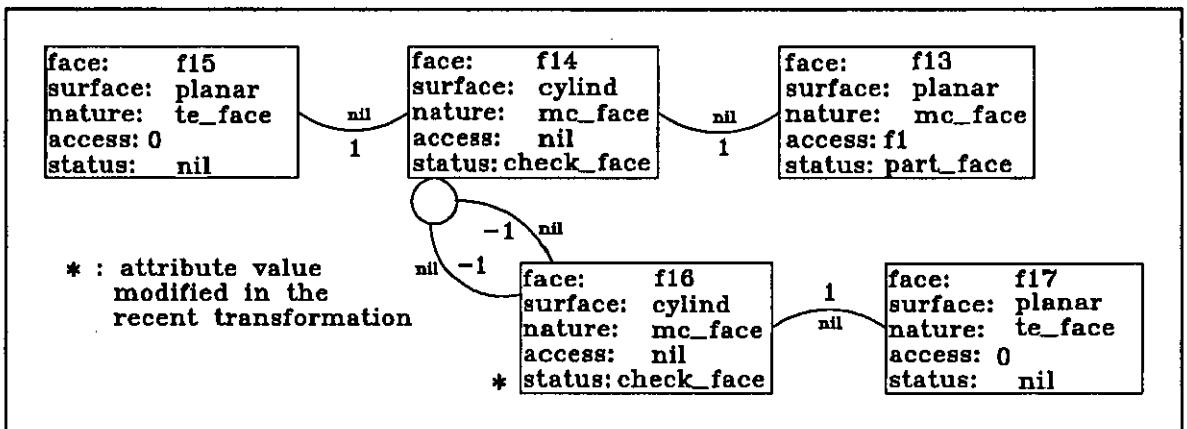


Figure 5.32 : The fourth transformation of the face/edge graph of subvolume\_2.

The machining features recognized from the subvolume\_1 and subvolume\_2 are summarized in Tables 5.2 and 5.3 respectively :

#### From the Subvolume\_1

<b>machining feature</b>	<b>part face</b>	<b>check face</b>	<b>primary top entrance face</b>	<b>secondary top entrance face</b>	<b>side entrance face</b>
1	f8	f2, f3, f4, f5, f6, f7	f1	nil	f7
2	f10	f9	f1	f8	nil
3	f6	f11	f12	nil	nil

Table 5.2 : The machining features recognized from the subvolume\_1.

#### From the Subvolume\_2

<b>machining feature</b>	<b>part face</b>	<b>check face</b>	<b>primary top entrance face</b>	<b>secondary top entrance face</b>	<b>side entrance face</b>
1	f13	f14	f15	nil	nil
2	f14	f16	f17	nil	nil

Table 5.3 : The machining features recognized from the subvolume\_2.

### 5.5 Concluding Remarks

The devised feature recognition algorithm basically has two major steps : (1) searching a potential part\_face on the cavity volume model by means of matching the cavity volume boundary with the set of geometric and topological relationships defined in section 5.1, and (2) performing accessibility analysis on the potential part\_face by means of the ray casting technique. Effectively, the recognized features are ensured to be accessible 2.5D machining features. As the search for a potential part\_face is also performed on faces that have inner edge loops, the algorithm can extract 2.5D



machining features from the reasonably complex machining features that are formed due to feature interaction.

Understandably, machining features that violate the defined recognition mechanism of the algorithm will not be recognized by the system. As it is not feasible to predict and precode every possible feature pattern and recognition algorithm in a computer program, a more desirable approach to improve the capability of the system would be to separate feature definition from feature recognition. Attempts at using this approach are described in the next chapter.

## CHAPTER 6 MACHINE LEARNING OF FEATURES FOR RECOGNITION

### 6.1 The Role of the Machine Learning Approach

The motive for using the machine learning approach is to handle machining features that cannot be recognized by the devised feature recognition algorithm. Figure 6.1 illustrates the working idea of the machine learning approach in relation to the former recognition approach.

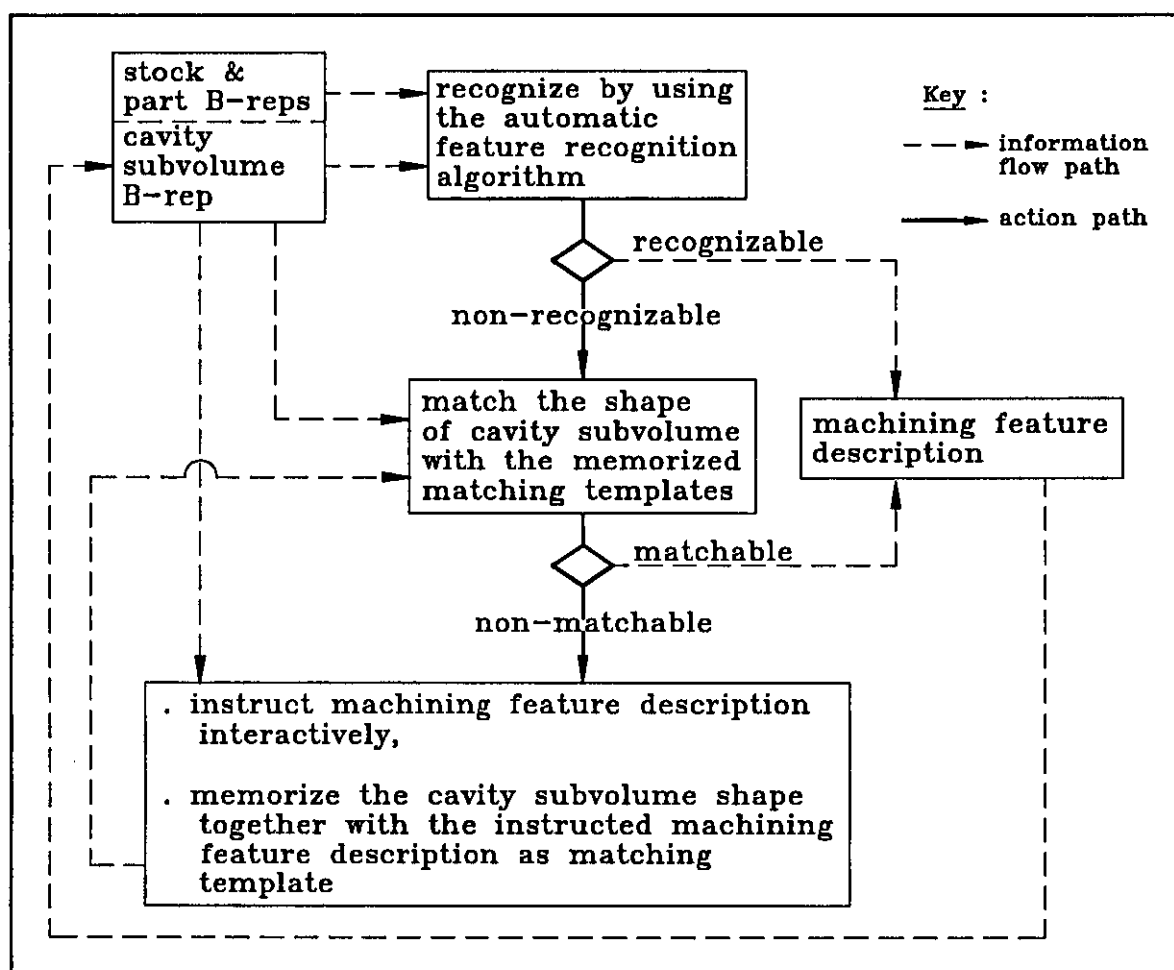


Figure 6.1 : The working concept of the research system.

The devised feature recognition algorithm is used in the front end to extract machining features that obey the predefined feature definitions without the need of human intervention. The machine learning approach, however, is used to learn a non-recognizable machining feature as a piece of new knowledge so as to enable the system to recognize similarly shaped machining features that would be encountered subsequently. A non-recognizable machining feature need not be learnt if it is not expected to be met again in the future. In that circumstance, it can be handled as a one-off job by using the human-assisted part programming approach. The decision as to whether or not a non-recognizable machining feature is worth learning should be made by the user of the system. For instance, it would be useful to learn factory dependent machining features which can be grouped into families based on their parameterizable shapes.

## 6.2 The Methodology of the Approach

As introduced in section 3.3, a learning process can be conducted by using different learning strategies. This thesis basically adopts the learning by rote strategy as a first attempt to study the machine learning of features for recognition. According to the principle of the learning by rote strategy, the system should be capable of performing four major tasks : (1) acquire information and the associated actions about an exemplary situation provided by the user, (2) memorize the acquired information and actions as an internal representation or matching pattern in the system, (3) recognize similarity between the memorized matching pattern and new situation, and (4) retrieve and apply the memorized actions to the new situation. The methodology of the machine learning approach used in this thesis is based on these four major learning activities.

The first task represents the use of an exemplary cavity volume  $V$  that cannot be handled by the recognition algorithm for the user to instruct the corresponding feature description  $F$  in terms of the machining face scheme adopted in the first approach.

The second task involves the conversion of the boundary characteristics of **V** together with the instructed feature description **F** into a set of production rules which represent a memorized matching pattern. The definition of boundary characteristics will be discussed later on. Symbolically, the first and second tasks can be expressed as :

$$\{ b[\mathbf{V}] + \mathbf{F} \} \Rightarrow \mathbf{M}$$

where **b[V]** = the boundary characteristics of cavity volume **V**  
 + = association of a feature description instructed by user  
**F** = the instructed feature description  
 => = conversion from B-rep data to production rules  
**M** = matching pattern rules

The third task is performed when the system subsequently encounters a new cavity volume **W** whose cavity volume boundary characteristics match with the conditions of the memorized set of production rules. This can be expressed as :

$$b[\mathbf{W}] < > \mathbf{M}(b[\mathbf{V}])$$

where **b[W]** = the boundary characteristics of cavity volume **W**  
 < > = successful pattern matching  
**M(b[V])** = the boundary characteristics portion of matching pattern rules **M**

The last task is performed at the result of firing the set of production rules **M** whose actions lead to the retrieval and substitution of the previously instructed feature description **F** to the new feature cavity volume **W**, i.e.

$$\mathbf{M}(\mathbf{F}) \rightarrow \mathbf{W}$$

where **M(F)** = the feature description portion of matching pattern rules **M**  
 -> = retrieval and substitution of feature description

Two major issues arise from the third task : the definition of boundary characteristics (or shape), and the conditions for testing shape similarity. These two issues are elaborated below.

### 6.2.1 Boundary Characteristics

The boundary  $bV$  of cavity volume  $V$  is considered to be composed of  $m$  number of faces and  $n$  number of edges as :

$$bV = \{fv_1 + fv_2 + fv_3 + \dots fv_m\} + \{ev_1 + ev_2 + ev_3 + \dots ev_n\}$$

For a face  $f$ , a set of characteristic conditions (or constraints)  $g$  can be defined. Symbolically, the set of constraints  $g$  of a face  $f$  is expressed as  $f[g]$ . For instance, the set of face conditions considered in this approach are :

- (1) face type (planar or cylindrical),
- (2) face nature (machined\_face or tool\_entrance\_face),
- (3) number of boundary edges, and
- (4) instructed machining feature description (part\_face, side\_entrance\_face, or primary\_top\_entrance\_face).

Similarly, a set of conditions  $h$  can be defined for an edge  $e$ . For example, the set of edge conditions used in this approach are :

- (1) edge type (line, ellipse (including circular), or general cylinder/cylinder intersection parametric curve),
- (2) convexity (convex, concave, or smooth),
- (3) left adjacent face (face identity used in the winged-edge B-rep database), and
- (4) right adjacent face (face identity).

The first two conditions are basically geometric information of the edge, whereas the last two conditions are topological information that help to define the 'shape' of the object. Symbolically, the conditions of an edge is expressed as  $e[h]$ .

In the approach, the boundary characteristics  $b[V]$  of a cavity volume  $V$  are expressed as a combination of the following conditions :

- (1) the number of boundary faces  $m$ ,
- (2) the number of boundary edges  $n$ ,
- (3) the geometric conditions of each face, i.e.  
 $\{fv_1[g], fv_2[g], \dots fv_m[g]\}$ , and
- (4) the geometric and topological conditions of each edge, i.e.  
 $\{ev_1[h], ev_2[h], \dots ev_n[h]\}$

Two candidate cavity volumes  $V$  and  $W$  are considered to be similar in shape if their boundary characteristics are identical, i.e.

$$b[V] = b[W]$$

This implies that the following four conditions are satisfied :

- (1)  $\{fv_1[g], fv_2[g], \dots fv_m[g]\} < > \{fw_1[g], fw_2[g], \dots fw_p[g]\}$
- (2)  $\{ev_1[h], ev_2[h], \dots ev_n[h]\} < > \{ew_1[h], ew_2[h], \dots ew_q[h]\}$
- (3)  $m = p$
- (4)  $n = q$

where  $p$  = the number of faces of cavity volume  $W$ ,  
 $q$  = the number of edges of cavity volume  $W$ ,  
 $< >$  = successful pattern matching,  
 $fw$  = face of cavity volume  $W$ , and  
 $ew$  = edge of cavity volume  $W$ .

Hence, the test for shape similarity performed in the third task is to test whether or not the above four conditions of a previously learnt feature can match the corresponding conditions of a new feature. The question of whether such a set of matching conditions are sufficient for a reliable shape comparison will be discussed in the next chapter.

With the help of a hypothetical part shown in Fig. 6.2, the approach is now further elaborated based on three main steps : (1) teaching a feature description, (2) memorizing the taught feature, and (3) recollecting the learnt feature.

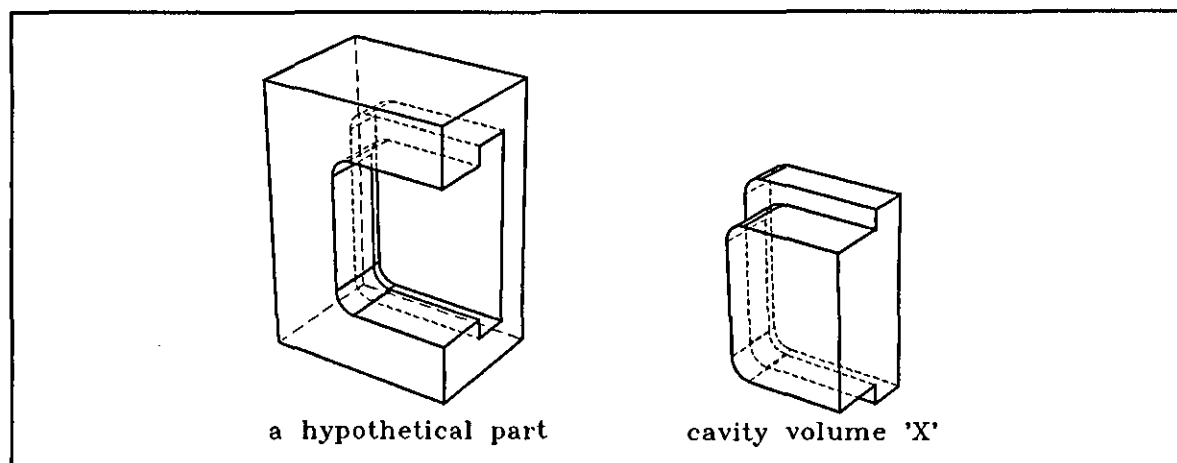


Figure 6.2 : A hypothetical part for explaining the machine learning approach.

### 6.2.2 Teaching a Feature Description

The hypothetical part contains only one cavity volume 'X'. For the present implementation, it is assumed that a cavity volume contains only one machining feature to be learnt. However, it is considered that the principle of the method can be applied on a cavity volume that contains more than one machining feature.

It can be observed that 'X' is a non-2.5D, T-slot-like machining feature. Also it can be appreciated that 'X' cannot pass the geometric test of the devised feature recognition algorithm as its T-slot-like undercut feature cannot be machined by the use of a simple cylindrical cutter. For explanation purposes, it is assumed that the system has not encountered machining feature cavity volumes with a shape similar to that of 'X' before. This implies that the system has no prior knowledge about 'X' or its similarly shaped counterparts. Thus the machine learning method can be used to learn 'X' with the intention that after the learning process the system will be able to recognize 'X' or similarly shaped machining features automatically.

The learning strategy is based on using 'X' as a positive teaching example for the user to teach the corresponding machining feature description to the system. The machining feature description is in terms of the `part_face`, `side_entrance_face`, and `primary_top_entrance_face` that have already been used in the generic feature definitions. There are three main reasons for using the same feature description. Firstly, as mentioned in section 4.4, the three faces serve to describe the machining method of a machining feature. Secondly, maintaining a uniform feature representation in the system standardizes the communication of feature information to other manufacturing applications such as process planning. Lastly, the manufacturing meaning of the three machining faces could be easily understood by a general user such as a CAD/CAM engineer or a CNC machine operator, so that the teaching of new features to the system would not need to be performed by special experts.

Hence, the teaching of the feature description of 'X' is equivalent to the specification of the three machining faces on the boundary of 'X' by the user. In the system, it is implemented in such a way that the user specifies the three faces interactively with the help of the wireframe display of 'X' and the pointing device of the computer system.

As the machining method used for machining 'X' is similar to the general T-slot machining operation, faces `f1`, `f2` and `f3` would be specified as `part_face`, `side_entrance_face` and `primary_top_entrance_face` respectively (Fig. 6.3).

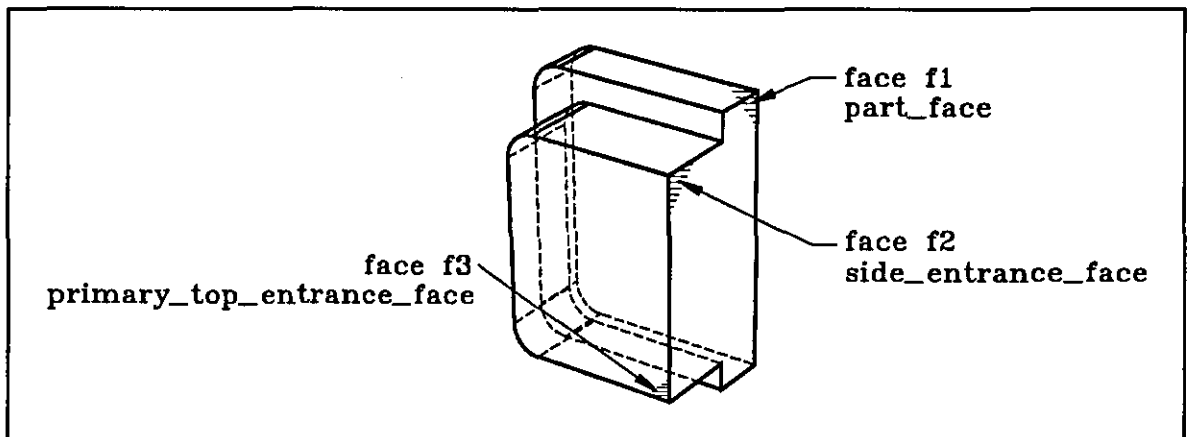


Figure 6.3 : Specifying the three machining faces as feature description.



It is realized that a machining feature could have several side\_entrance\_faces and primary\_top\_entrance\_faces. However, the present implementation of the approach assumes that only one side\_entrance\_face and one primary\_top\_entrance\_face need to be specified. The selection of the three machining faces is decided by the user. The specified machining face information is incorporated in the corresponding face records in the B-rep of 'X'. This is illustrated by using the face/edge graph shown in Fig. 6.4.

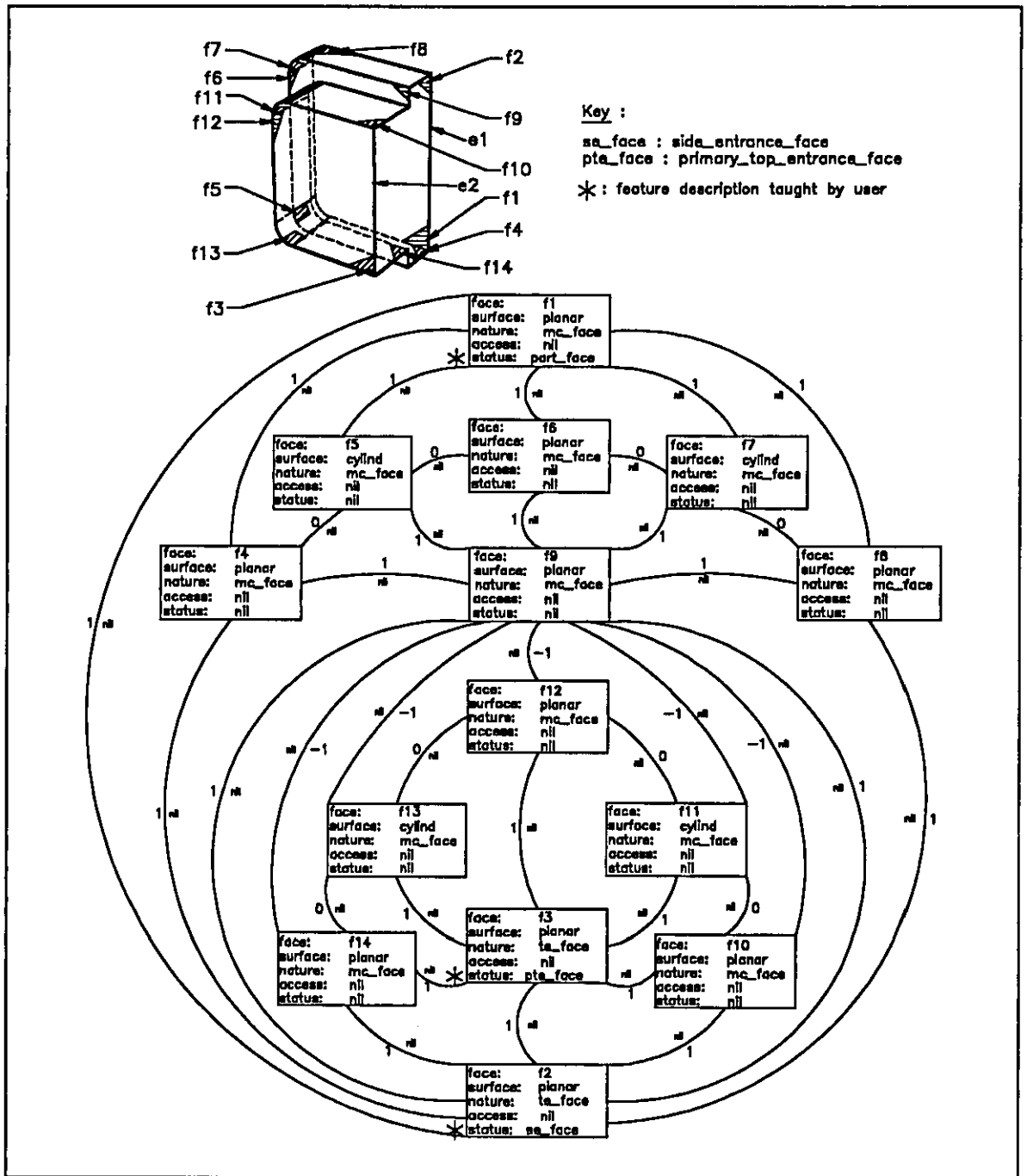


Figure 6.4 : The face/edge graph of cavity volume 'X'.

### 6.2.3 Memorizing the Taught Feature

The taught machining feature is memorized as a piece of new knowledge by automatically converting the boundary characteristics of 'X' into a set of production rules. As mentioned earlier, the boundary characteristics of 'X' essentially represent the geometric and topological conditions of the faces and edges of 'X' to be matched. For instance, as for 'X', the boundary characteristics described in the condition part of the production rules would be :

face  $f1$  is planar, nature is machined\_face, number of boundary edges is 6, status is specified as part\_face;  
 edge  $e1$  is linear, convex, left adjacent face is  $f2$ , right adjacent face is  $f1$ ;  
 face  $f2$  is planar, nature is tool\_entrance\_face, number of boundary edges is 8, status is specified as side\_entrance\_face;  
 edge  $e2$  is linear, convex, left adjacent face is  $f3$ , right adjacent face is  $f2$ ;  
 face  $f3$  is planar, nature is tool\_entrance\_face, number of boundary edges is 6, status is specified as primary\_top\_entrance\_face;  
 ... etc. for the remaining faces and edges.

As a production rule in the KBS can only contain a limited number of conditional elements, the boundary characteristics are specified in a set of rules rather than in a single rule. However, the set of rules is virtually linked together as a total set which also implies that the matching for shape similarity is on the basis of the total set of rules rather than on a rule-by-rule basis. During the rule construction process, there is no checking whether an identical rule exists in some previously generated set of rules. More details about the construction of rules will be described in the next chapter.

The new rules are incorporated in the system by compiling them into an object code module which is then linked with the old object code modules of the system to produce a new executable program.

The teaching and memorization of new machining features can be viewed as the customization and re-configuration of the system. The feature extraction capability of the re-configured system improves due to an increase of feature recognition rules that are established without being concerned with the programming problems of new feature descriptions and recognition.

#### 6.2.4 Recollecting the Learnt Feature

When another non-recognizable machining feature is encountered subsequently, the system attempts to recall the learnt feature in terms of its shape and instructed feature description by matching the incorporated rules with the boundary characteristics of the non-recognizable machining feature. For instance, assuming that a similarly shaped cavity volume 'Y' (Fig. 6.5) is encountered subsequently, as it will not be recognized by the feature recognition module, the incorporated set of rules will match the boundary characteristics of 'Y'. As 'X' and 'Y' have identical boundary shape, the entire set of rules can be matched and, as a result, the three instructed machining faces of 'X' are retrieved and substituted as machining feature description for 'Y' as illustrated in Fig. 6.5.

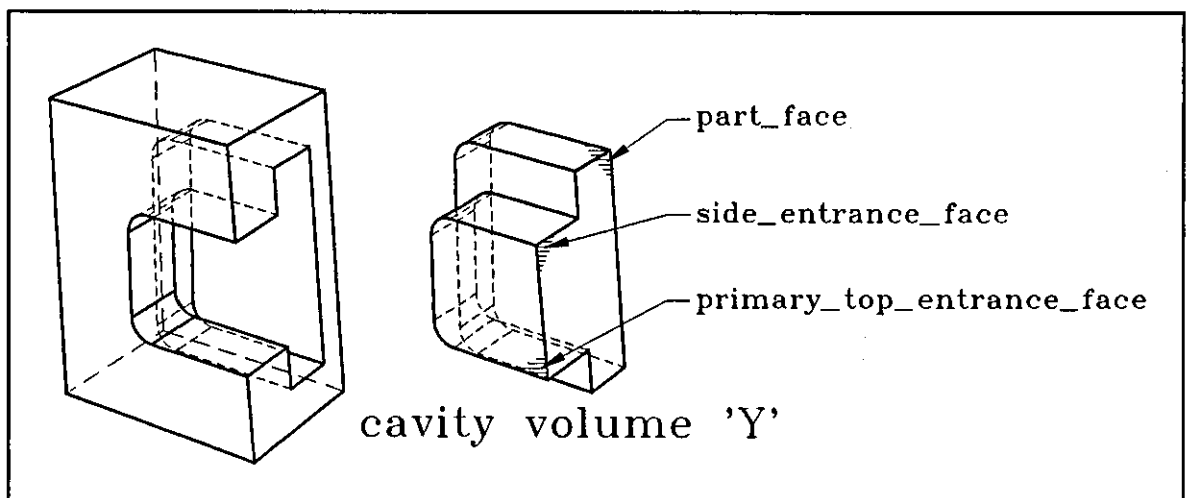


Figure 6.5 : A similarly shaped cavity volume 'Y'.

Should the matching fail, the next set of rules previously incorporated in the system will be used to match with the boundary characteristics of 'Y'. If there is no set of rules that can match the boundary characteristics of 'Y', the approach assumes that 'Y' is another new machining feature that could be learnt by the system in the same manner.

### **6.3 Concluding Remarks**

The automatic generation of a set of production rules as the result of learning an exemplary machining feature virtually represents the ability of developing a piece of new knowledge in the system for extracting machining features that have shapes similar to that of the exemplary feature. In effect, the machine learning approach improves the feature extraction capability by perpetually expanding the system knowledge base. This is in contrast with the former recognition algorithm approach in which the knowledge is precoded rigidly as a mixture of feature pattern declarations and geometric testing procedures.

## CHAPTER 7

# IMPLEMENTATION

An experimental prototype system has been implemented to study the feasibility of using the two described approaches for extracting machining features from a CAD database. The system is implemented by integrating a solid modeller with a rule-based AI environment.

The solid modeller is used as a CAD system for defining the nominal geometry of the starting stock and finished part, and for generating the boundary information of the corresponding cavity volume model by means of Boolean subtraction and boundary evaluation. The established B-reps of the solids interface directly with the feature recognition process and the machine learning process. This is necessary as the feature recognition algorithm is designed to store the extracted features information directly in the B-rep of the cavity volume, while the machine learning approach also requires interactive access to the B-rep of the cavity volume during the feature teaching and memorizing phases.

The main reason for using the rule-based AI environment is that its structure is basically a knowledge based system whose characteristics are described in section 3.1. The AI environment has a global database, a rule base and an inference mechanism. The global database is used for storing the shape definition of cavity subvolume, while the rule base and the inference mechanism are used for fast prototyping of the rule-based recognition approach and the machining learning approach.

### 7.1 The Solid Modelling System

The solid modelling system is the PADL-2 CSG modeller [Brown82]. The principle of CSG modelling method has been introduced in Chapter 2. As a typical CSG

system, PADL-2 uses a CSG tree data structure as the primary representational medium for maintaining the construction history of a user defined solid. The CSG tree is operated on by a set of boundary evaluation [Requicha85a] procedures to obtain the corresponding boundary information which is then stored and managed in an auxiliary boundary representational scheme, called the BFILE [Hartquist81]. The logical entities in the BFILE are linked collections of assemblies, solids, faces and edges.

PADL-2 software consists of functional modules that are organized as procedurally accessible subsystems that can be used through subroutine or function calls rather than by directly accessing the internal data structures. This open architecture of PADL-2 simplifies the task of binding it with the rule-based AI environment.

## **7.2 The VAX-OPS5 AI Environment**

The VAX-OPS5 [Digital85] AI environment is used for prototyping of the two feature extraction approaches. The VAX-OPS5 is an extended implementation of the OPS5 production rule language [Forgy77, Brownston85] which consists of a global database and production rules that manipulate the database. Data or working-memory elements in the database is represented in a frame format as illustrated in Fig. 7.9.

The VAX-OPS5 run-time system controls the execution of OPS5 programs and consists of a recognize-act cycle, command interpreter and run-time compiler.

### **7.2.1 The Recognize-act Cycle**

This is essentially the inference mechanism or pattern matcher of the system. During the recognize phase of the recognize-act cycle, the system compares working-memory elements of the database with the condition elements on the left-hand side of each rule (Fig. 7.1). When working-memory elements match all the condition elements,

the rule is ready for execution. 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, called an instantiation, 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.

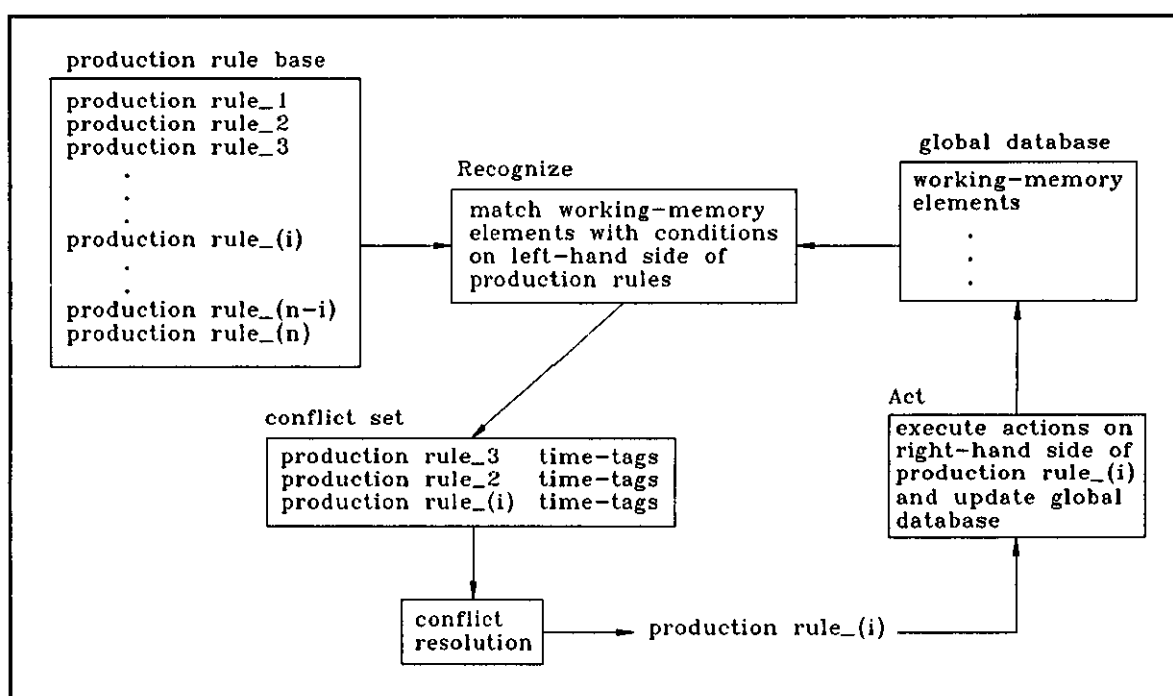


Figure 7.1 : The recognize-act cycle.

The run-time system uses either the Lexicographic-Sort (LEX) or the Means-Ends-Analysis (MEA) conflict resolution strategy to order and select one of the instantiations in the conflict set. Both strategies apply in the order of the following built-in rules : refraction, recency, specificity and arbitrary [McDermott78]. By the refraction rule, an instantiation is selected only once. This prevents a program from looping infinitely on the same data. The recency rule selects the instantiation that refers to the most recent data in working memory. This means that the system selects the instantiation that contains the highest time tags. The specificity rule selects an instantiation of a rule whose left-hand side is the most specific. Specificity is measured by the number of conditional tests on a rule's left-hand side. If more than one instantiation has the highest level of specificity, an instantiation is selected arbitrarily.

The MEA strategy is similar to the LEX strategy except that it includes an extra step after refraction, which orders the instantiations in the conflict set according to the recency of the working-memory element matching the first condition element in each rule. In the prototype system, the most important condition element is always placed first on the left-hand side of each rule, and hence, the MEA strategy is used in the system.

After the run-time system selects an instantiation 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 instantiation refers execute. The execution of the rule actions may effect changes in the working-memory elements of the database. When the act phase completes, the cycle goes back to the recognize phase.

### **7.2.2 The Command Interpreter**

The VAX-OPS5 command interpreter is used to control the execution of a program interactively. A special set of interpreter commands can be used for setting up initial conditions, executing recognize-act cycles, debugging OPS5 programs, controlling input/output, calling external routines and controlling program loops.

### **7.2.3 The Run-time Compiler**

By using the VAX-OPS5 'BUILD' action in a rule, new rules can be added to an executing program. Each time a 'BUILD' action executes, the run-time compiler creates a new version of the file named OPS\$BUILD.OPS for storing the source code of the new rule and also includes the execution codes of the new rule in the executing program. This new rule generation facility appears to be a very convenient means of implementing a learning agent in the system. However, as the source code of the new



rule is always stored in the same file, old rules stored in the file will be overwritten by new rules. Besides, information about the interface to the run-time compiler is also limited. Hence, this run-time rule generation facility is not used in the prototype system.

### 7.3 Linking PADL-2 and VAX-OPS5

As shown in Fig. 7.2, the prototype system is built by coupling the PADL-2 solid modeller with the VAX-OPS5 system. The system programs are developed by using the OPS5 production rule language and the FORTRAN language. During program development, the object files of the developed programs are linked with the object files of the PADL-2 programs to form one binary executable image that runs on a MircoVAX II workstation under the VAX/VMS operating system. The two systems communicate through the use of a set of VAX-OPS5's foreign language interface facility which is basically a set of support routines that enable external programs written in other languages to communicate with OPS5 programs.

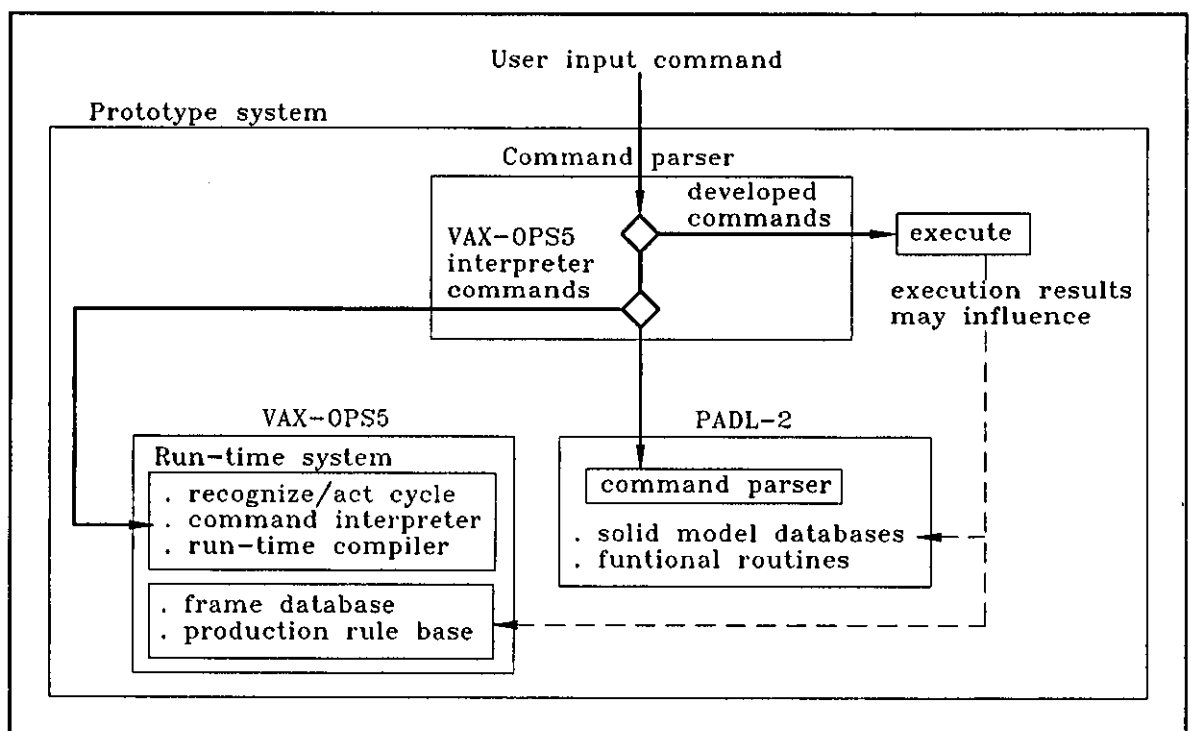


Figure 7.2 : Linking PADL-2 and VAX-OPS5.

The user-system interface is effected through the use of command line input method. A command parsing routine is developed to interpret the user input commands. The command parsing routine works like a two-stage filter. The first stage intercepts a set of new commands developed for carrying out a variety of functions such as enquiring of a geometric entity, activating the feature recognition process, etc.. The second stage catches the VAX-OPS5 interpreter commands and directs them to the VAX-OPS5 environment for execution. Commands that leak through the two stages are passed to the PADL-2 system for execution. If a wrong command is input, PADL-2 will return appropriate error message to prompt the user.

#### **7.4 Implementation of the Feature Recognition Approach**

At the outset, the CSG models of the part and the corresponding starting stock are defined through the use of PADL-2's commands. To facilitate the modelling of some typical cavity shapes in a part, several generic meta-primitives such as blocks with round corners are also predefined in the system. However, it is the duty of the user to ensure that the shape and size of the stock are correctly defined for making the part. Moreover, since the cavity volume is obtained via a Boolean subtraction operation between the part and the stock, the user must also ensure that the relative position and orientation between the part and stock are correct so that the desired cavity volume is obtained.

The feature recognition process involves the following three major steps : (1) establishing boundary information, (2) describing cavity subvolume in VAX-OPS5, and (3) recognizing machining features.

### 7.4.1 Establishing Boundary Information

After defining the CSG models of the stock and part, a command 'makwed/stock, part' is issued to the system. The command accepts the names of the defined stock and part as command arguments and activates the corresponding set of command procedures which perform two main functions :

- (1) establish the CSG data structure of the cavity volume model as described by expression (2) in section 4.2,
- (2) activate the boundary evaluation procedures of PADL-2 to establish the boundary information of the stock, part and cavity volume.

As mentioned in section 7.2, the boundary information is stored and managed in a hierarchical BFILE in PADL-2. However, there are two problems with the use of the BFILE. The first problem is that the definition of faces in the BFILE has been based on the so-called maximal-face [Silva81] scheme in which faces belonging to the same half-space are collectively addressed by a single logical pointer. For many applications, especially in feature recognition, this maximal-face representation method is undesirable. For example, using the maximal-face scheme, the two areas 'A' and 'B' shown in Fig. 7.3(a) are represented as a single face. In contrast, the connected-face scheme [Silva81] illustrated in Fig. 7.3(b) is congenial with the human perceived definition of an object face and is a more sensible segmentation of the surface boundary for feature recognition. Unfortunately, changing the maximal-face scheme of the BFILE is a formidable task as the scheme is implemented as part of the sophisticated boundary evaluation algorithm.

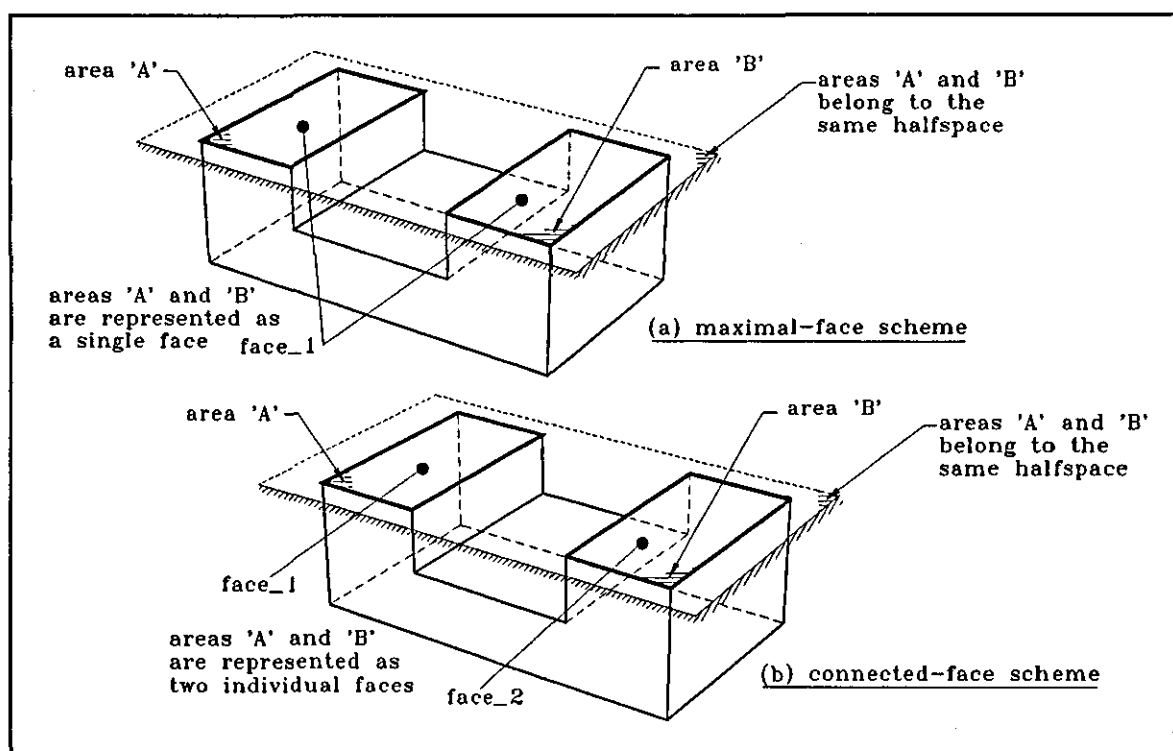


Figure 7.3 : Maximal-face and connected-face schemes.

The second problem is that the hierarchical structure of the BFILE also imposes restrictions on computer program design due the fact that the BFILE entities ( i.e. assemblies, solids, faces, edges, and vertices ) have to be traversed and accessed in a top-down manner.

To overcome these problems, a new B-rep database structure is developed in the prototype system for maintaining the B-reps of the stock, part and cavity volume. The basic B-rep information in the new database is derived from that of the BFILE. However, the boundary faces of the stock, part and cavity volume are represented as connected-faces in their corresponding new B-rep databases. The conversion from maximal-face to connected-face is performed by an implementation of a conversion algorithm proposed by Chan [Chan88], which basically determines all the closed edge loops for each maximal-face represented in the BFILE and then groups the identified edge loops into outer edge loop and inner edge loops of a connected-face.

7.4.1.2 Boundary Representation of the Stock and Part Models

A data structure has been defined and implemented for handling the converted boundary representation. The B-rep structures of the stock and the part are basically the same. For explanation purposes, the B-rep structure of the stock is shown in Fig. 7.4.

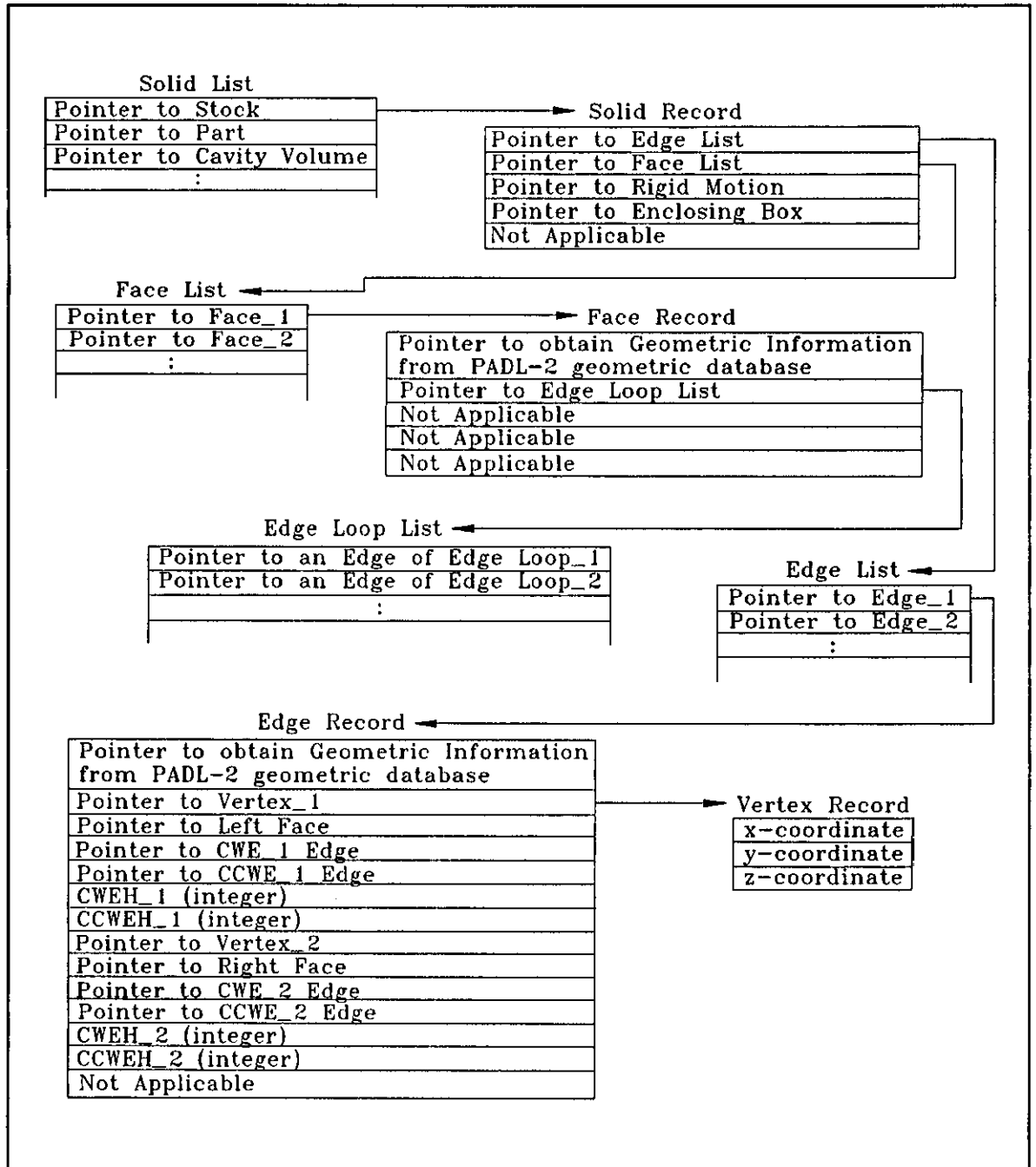


Figure 7.4 : B-rep data structure of the stock and part.

At the highest level, the structure uses a list for storing the pointers to the records of the three modelled solids, namely the stock, part, and cavity volume. The solid record has 5 fields. The first field stores a pointer to an edge list in which the edge record pointers are maintained. Similarly, the second field stores a pointer to a face list for storing the face record pointers. The purpose of these edge and face lists is to facilitate sequential traversal of the edges and faces when necessary. The third field stores a pointer for obtaining the rigid motion transformation matrix (location and orientation) of the solid, while the fourth field stores a pointer for obtaining the enclosing box size of the solid. The fifth field is not applicable for the stock and part models.

The face record also has 5 fields. The first field contains a pointer that can be used to obtain the face's geometric information, such as surface type, surface normal, etc., from the PADL-2 geometric database. The second field is a pointer to an edge loop list. The number of elements in the edge loop list represents the number of edge loops of the face. For instance, if a face has an inner edge loop, then its edge loop list will have two elements. The first element is a pointer to an edge belonging to the outer edge loop, while the second element is a pointer to an edge belonging to the inner edge loop. The last three fields are not relevant for the stock and part models.

The edge record contains 14 fields. The first field is used to obtain the edge's geometric information, such as curve type, curve parameters, etc., from the PADL-2 geometric database. The design of the subsequent twelve fields (i.e. from the second field to the thirteenth field) is based on the modified winged-edge data structure proposed by Weiler [Weiler85]. The modified winged-edge structure and the symbols used in the fields are explained in Appendix E. In summary, the winged-edge structure is a non-hierarchical, edge-based data structure which maintains explicitly the adjacency relationships of faces, edges and vertices, and thus offers more freedom in accessing boundary information in the B-rep. The last field is not applicable for the stock and part models. A fragment of the B-rep of the hypothetical part used in chapter 5 is illustrated in Fig. 7.5.

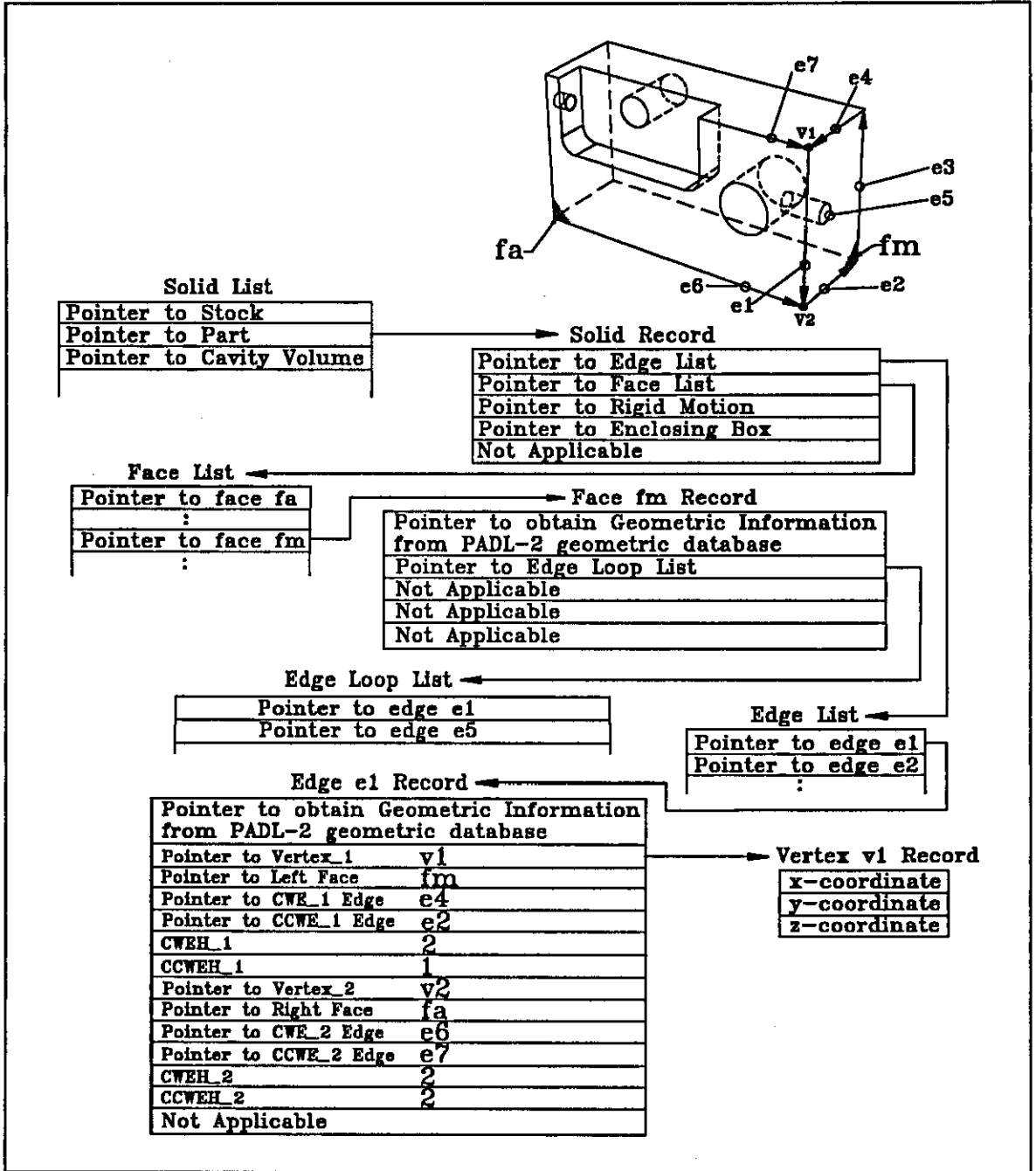


Figure 7.5 : A fragment of the B-rep of the hypothetical part.

### 7.4.1.3 Boundary Representation of the Cavity Volume Model

The B-rep of the cavity volume is an extended version of the B-reps of the stock and part models. As can be seen in Fig. 7.6, the fifth field of the solid record now stores a pointer to a cavity subvolume list. This is necessary because the cavity volume

may consist of several disjoint subvolumes as described by expression (3) in section 4.2.

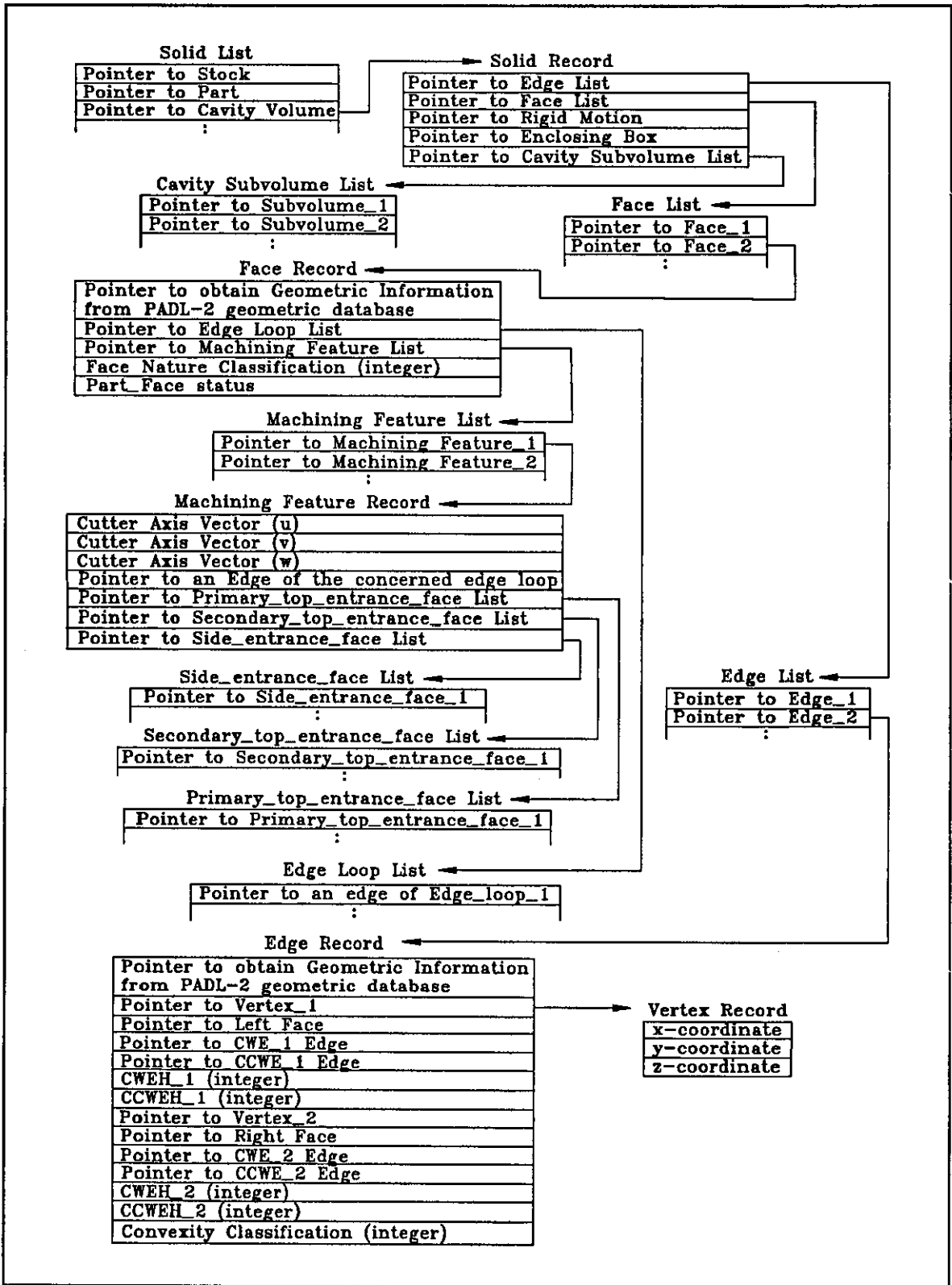


Figure 7.6 : B-rep structure of the cavity volume model.



The cavity subvolume list stores the pointers for addressing the individual subvolumes. In turn, each individual subvolume is represented by the same solid record structure that maintains its own lists of faces and edges as shown in Fig. 7.7.

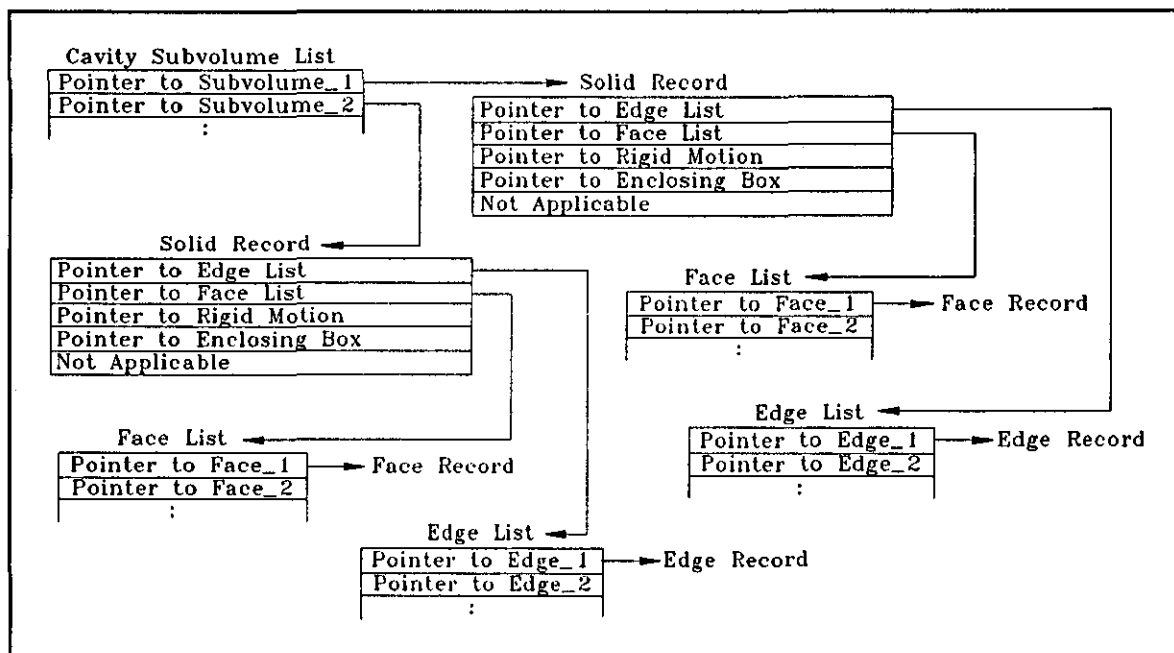


Figure 7.7 : Cavity subvolume list structure.

The faces of each individual subvolume are determined by classifying the faces of the cavity volume into face groups such that the faces within each face group are adjacent to each other. The edges of each individual subvolume can then be determined simply from the boundary edges of the faces in each face group.

As mentioned earlier, the last three fields of the face record are not used for the stock and part models. However, for the cavity volume model, the third field of the face record is designed to store the recognized machining features. As can be seen in Fig. 7.6, if a face is recognized as a part\_face, then the third field of the face record will store a pointer to a machining feature list. If the face is not a part\_face then the third field of the face record is not used. In turn, the machining feature list stores the pointers to the corresponding machining feature records of the face. This means that each machining feature record represents a machining feature that uses the face as a

part\_face. It will be recalled from section 4.4 that a multi-connected face (i.e. face with inner edge loops) can be a condition type 5 part\_face, and can be used by several machining features as illustrated in Fig. 7.8. This is why a machining feature list is used for maintaining the recognized machining features that are associated with a part\_face.

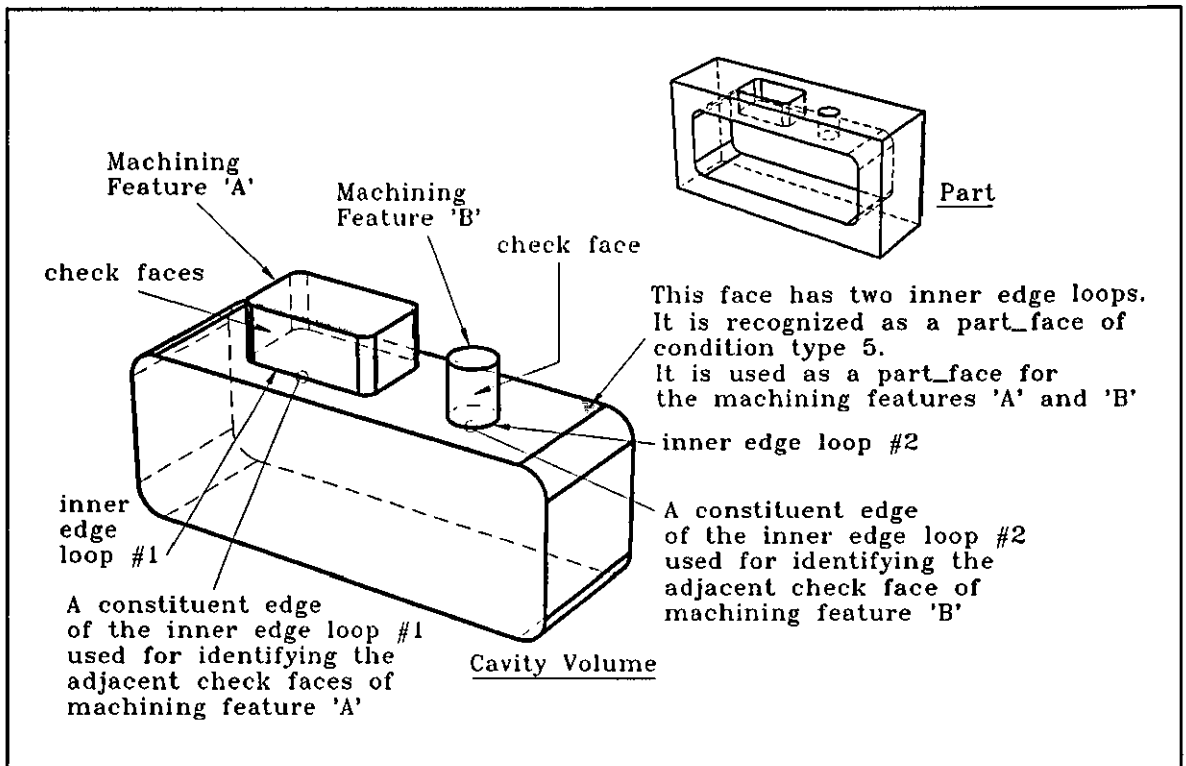


Figure 7.8 : A condition type 5 part\_face.

The fourth field of the face record stores an integer which represents the face nature classification of the cavity volume as discussed in section 4.3. As described by expression (4) in section 4.3, the tool\_entrance\_face is defined as  $(bS \cap cP)$ , hence the face nature is determined by classifying each of the cavity volume faces against the stock faces. If a cavity volume face is contained in a stock face (i.e. same half-space) then the cavity volume face is classified as tool\_entrance\_face, and an integer 1 is stored in the fourth field of the face record. Otherwise, the cavity volume face is tagged as machined\_face and an integer -1 is stored. The fifth field is used to indicate whether or not the face is a part\_face.

The machining feature record contains 7 fields. The first 3 fields are used to store the cutter axis vector parameters that represent the cutter approach direction to the `part_face` as described in section 5.3.4. The fourth field stores a pointer to an edge which is shared between the `part_face` and the `check_face` of the corresponding machining feature as illustrated in Fig. 7.8. The fifth, sixth and seventh fields store the pointers to the `primary_top_entrance_face` list, `secondary_top_entrance` face list and `side_entrance_face` list respectively. The three lists are used for maintaining the face identities of the three different types of `tool_entrance_faces`.

The last field of the edge record stores an integer which represents the convexity classification of the edge. The edge convexity of an edge is determined by evaluating the inner angle of the two adjacent faces meeting at the edge according to the conditions illustrated previously in Fig. 2.8. In the implementation, convex, concave and smooth edges are represented by integers 1, -1 and 0 respectively.

When the B-reps of the stock, part and cavity volume are successfully established, the system will report the number of subvolumes contained in the cavity volume. The subvolumes maintained in the cavity subvolume list are arranged in ascending order of their number of boundary faces, and are also given system default names as `mv_1`, `mv_2`, etc.. Thus the first subvolume `mv_1` has the least number of faces.

Three interactive enquiry commands are developed for the user to interrogate the established models : (1) `asksvol` (2) `asksolid/<x>`, (3) `askface/<x>`, and (4) `askedge/<x>`. The facilities of these commands are outlined below :

#### (1) `asksvol`

This command is used to interrogate the cavity subvolumes available in the system. The system uses the pointers maintained in the cavity subvolume list (Fig. 7.6) to access the corresponding B-reps of the cavity subvolumes. A wireframe display of all the disjoint cavity subvolumes is rendered on the screen, and the total number of

cavity subvolumes with their system given names, i.e. mv\_1, mv\_2, etc., are reported.

### (2) **asksolid/ <x>**

This is used for obtaining information about a solid <x> maintained in the B-rep database. The command argument 'x' can be either one of the followings :

- (a) the user given name of the original stock used in the CSG design stage,
- (b) the user given name of the part used in the CSG design stage,
- (c) the system given name of any cavity subvolume.

The system renders a wireframe display of the enquired solid, and reports the following information :

- (a) the total number of boundary faces,
- (b) the total number of boundary edges,
- (c) the total number of machined\_faces (if enquired solid is a cavity subvolume), and
- (d) the total number of tool\_entrance\_faces (if enquired solid is a cavity subvolume).

### (3) **askface/ <x>**

This command is used to enquire about a boundary face of a cavity subvolume 'x'. The system displays the wireframe image of the desired subvolume on the screen and prompts the user to input the enquired face by means of picking any two boundary edges of the face with the use of the 'mouse' pointing device. The system acknowledges the input face by highlighting the boundary edges of the face with a different colour, and reports the following textual information on the screen :

- (a) integer identity of the enquired face,
- (b) surface type (planar or cylindrical),
- (c) face nature classification (machined\_face or tool\_entrance\_face), and
- (d) machining feature description (part\_face, check\_face, etc.).

### (4) **askedge/ <x>**

This is used to ask about a boundary edge of a cavity subvolume 'x'. The user input the enquired edge again by means of interactively picking the graphical display

image of the edge with the use of the 'mouse'. The system highlights the input edge with different colour and returns the following message on the screen :

- (a) integer identity of the edge,
- (b) curve type (line, ellipse or general cylinder/cylinder intersection curve), and
- (c) convexity (convex, concave or smooth).

#### 7.4.2 Describing Cavity Subvolumes in VAX-OPS5

A cavity subvolume 'mv\_1' is presented to the feature recognizer for recognition by using the command 'bframe/mv\_1'. The corresponding command procedure collects relevant information from the cavity volume B-rep database and establishes a frame-based description of mv\_1 in the global database of the AI environment.

The frame-based description of a cavity subvolume consists of : (1) a solid frame, (2) face frames, (3) edge frames, and (4) inner edge loop frames.

The structure of a solid frame is :

```
Frame :      solid
Attribute1 : name
```

A face frame has the following structure :

```
Frame :      face
Attribute1 :  face identity
Attribute2 :  face's surface type
Attribute3 :  face nature classification
Attribute4 :  number of boundary edges
Attribute5 :  access
Attribute6 :  status
```

The structure of an edge frame is :

Frame : edge  
Attribute1 : edge identity  
Attribute2 : edge's curve type  
Attribute3 : convexity classification  
Attribute4 : left adjacent face identity  
Attribute5 : right adjacent face identity  
Attribute6 : status

The structure of an inner edge loop is :

Frame : inner edge loop  
Attribute1 : edge identity of an edge belonging to the inner edge loop  
Attribute2 : face identity of a face that owns the inner edge loop

As an illustration, the frame-based description of the subvolume\_2 used in chapter 5 is shown in Fig. 7.9. It can be seen that the frame-based description is essentially an implementation of the face/edge graph representation used in chapter 5.

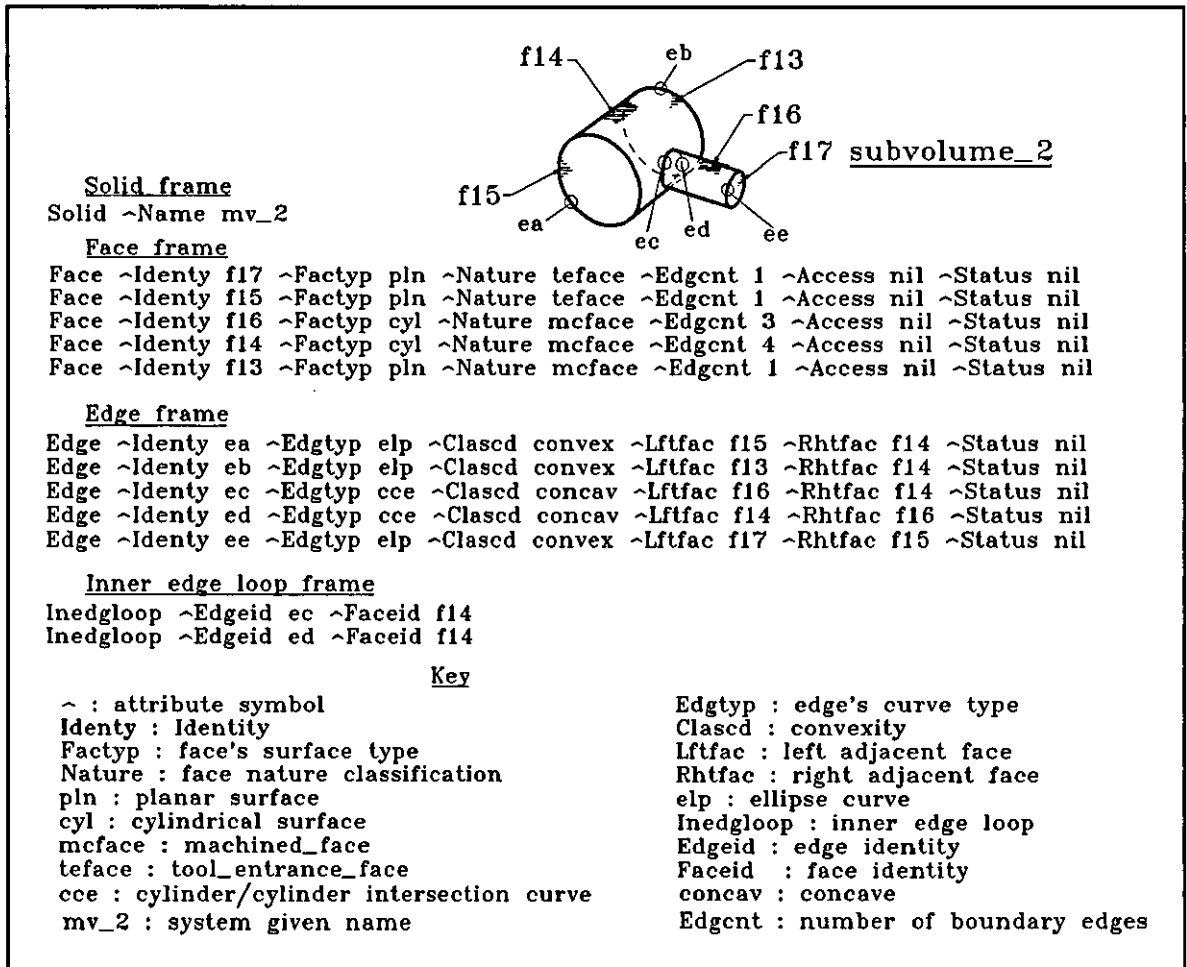


Figure 7.9 : Frame-based representation of the subvolume\_2.

The candidate face selection mechanism described in section 5.3.1 is implemented by making use of the recency selection rule described earlier in section 7.2.3. The recency selection rule selects the instantiation that refers to the most recent data in working memory, and hence the creation of face frames in the global database is designed to follow the chronological sequence :

{... c2, c1, ... p2, p1, ... pne2, pne1},

where the symbols c2, c1, p2, p1, pne2, and pne1 have the same meanings as used in section 5.3.1. For instance, pne1 was the last face frame created, and hence it is the most recent face frame (with the highest time tag) in comparison with the others. So according to the recency selection strategy, pne1 is the first candidate face to be chosen. After the selection of pne1, the next most recent face frame is pne2, and so on. In

effect, the recency selection strategy is used as a sequential data accessing method similar to the way of popping data out of a stack memory.

The face frames shown in Fig. 7.9 are listed in their actual sequence created in the global database. It can be seen that the `tool_entrance_face` frames are created first followed by the `machined_face` frames. As a result, the `machined_face` group will be selected before the `tool_entrance_face` group, and thus this is in accordance with the recognition algorithm which considers the group(1) faces before the group(2) faces.

Apart from creating the above frame-based description, the 'bframe' command also creates several other utility frames in the global database. The utility frames are used mainly for the purpose of passing messages amongst the rules. The idea of using the utility frames will become clearer in the next section.

### 7.4.3 Recognizing Machining Features

The machining feature recognition algorithm is implemented by means of a mixture of OPS5 language production rules and FORTRAN language procedures. The use of OPS5 production rules is to utilize the recognize-act cycle mechanism as a pattern matcher for searching and matching the rule conditions as discussed in chapter 5. The FORTRAN procedures are used for performing tasks such as line/surface intersection computation and communication with the B-rep database/geometric database of the solid modeller.

Before starting the recognition process, the user can use some optional commands to monitor the recognition process such as the (1) `watch <integer>`, and (2) `raydisp <on>/<off>`. The 'watch' command is an OPS5 interpreter command which sets the amount of trace information that the system displays while executing the recognition program. The command argument is an integer in the range of 0 to 3 (default value is 1), which represents the trace level to be set as follows :



Level	Trace Information Displayed
0	None
1	The instantiations selected by the conflict resolution strategy for execution
2	The same as level 1, plus the working memory elements that are added to and deleted from the global database
3	The same as level 2, plus changes that occur in the conflict set.

Thus by examining the trace information, the user can obtain an explanation of how the reasoning process is performed during the entire recognition process. For instance, the faces included in the conflict set and the reasoning results of a selected candidate face can be monitored. The recognition process can also be animated by using the 'raydisp' command which toggles the on/off display of the casted rays used in the line/surface and line/curve intersection tests.

The recognition process is initiated by issuing the command 'recognize/mv\_<integer>', where 'mv\_<integer>' is assumed to be the name of the cavity subvolume to be recognized. For instance, to recognize the first subvolume mv\_1, the command is 'recognize/mv\_1'. The command is intercepted by the command parser described formerly in section 7.3. A utility frame 'command ^type recognize ^argument1 mv\_1' is then created in the global database. This utility frame is matched by the following production rule (expressed in OPS5 language syntax) in the rule base:

```
(p recognize
  {<recogn> (command ^type recognize ^argument <subvolume_name>)}
  (solid ^name <subvolume_name>)
-->
  (remove <recogn>)
  (make return_to_command_parser)
  (make goal ^context report ^argument <subvolume_name>)
  (make goal ^context reason_group(3)_face ^argument <subvolume_name>)
  (make goal ^context reason_group(2)_face ^argument <subvolume_name>)
  (make goal ^context reason_group(1)_face ^argument <subvolume_name>))
```

The meaning of the above rule is as follows :

Rule name : recognize

- If . in the global database there is a 'recognize' command whose argument is <subvolume\_name>, and  
 . there is a solid whose name is <subvolume\_name> ,

Then

- . remove the command utility frame from the global database to avoid recursive firing of this rule,
- . generate five new utility frames in the global database in the following order:
  - goal ^context return\_to\_command\_parser
  - goal ^context report ^argument <subvolume\_name>
  - goal ^context reason\_group(3)\_face ^argument <subvolume\_name>
  - goal ^context reason\_group(2)\_face ^argument <subvolume\_name>
  - goal ^context reason\_group(1)\_face ^argument <subvolume\_name> .

According to the above sequence of utility frame generation, the last frame 'goal ^context reason\_group(1)\_face...' is the most recent, while the second last frame 'goal ^context reason\_group(2)\_face...' is the next most recent, and so on. Due to the built-in recency selection mechanism, the last frame has higher matching priority than the second last frame, and so on. Thus, in effect, the last three utility frames act as an agenda to control the recognition process to consider the group(1) faces first, then the group(2) faces and lastly the group(3) faces.

The utility frame 'goal ^context report...' is for reporting any faces that have failed the geometric tests. The frame 'goal ^context return\_to\_command\_parser' is for returning the system control to the command parser at the end of the recognition process.

The recognition process is driven on the forward chaining strategy as the cavity subvolume frame-based description created in the global database is matched and tested by a set of production rules for determining the three groups of faces contained in the cavity subvolume. For instance, the selection and testing of a group(1) face is initiated

by the firing of the following production rule :

```
(p reason_group(1)_face
(goal ^context reason_group(1)_faces ^argument <subvolume_name>)
{<face> (face ^factyp {<> cyl} ^nature mcfac ^access {<> 0}
^status {<> part_face <> check_face} ^identity <mcfac-id>)}
(edge ^lftfac <mcfac-id> ^identity <edge-id>)
- (inedgloop ^edgeid <edge-id>)
-->
(call xwedfc <mcfac-id> <subvolume_name>)
(modify <face> ^access (xmfact <mcfac-id> <edge-id> <subvolume_name>))
(make goal ^context change_adjacent_face_status ^argument <mcfac-id>))
```

The above rule reads as :

Rule name : reason\_group(1)\_face

- If
- . the goal is to reason a group(1) face of a subvolume whose name is addressed by the pointer <subvolume\_name>, and
  - . in the global database there is a face frame whose characteristics are as follows:
    - face surface type - not cylindrical
    - face nature - machined\_face
    - access utility flag - not zero
    - status utility flag - neither 'part\_face' nor 'check\_face'
    - face identity - addressed by the pointer <mcfac-id> ,
  - . in the global database there is an edge frame whose adjacent face is the same as the face addressed by the pointer <mcfac-id>; the identity of the edge is addressed by the pointer <edge-id>, and
  - . the edge addressed by the pointer <edge-id> is not an edge member of an inner edge loop,

Then

- . use the names of the face and subvolume as input parameters to call the external subroutine 'xwedfc' for highlighting the wireframe image of the face,
- . pass the names of the face, edge and subvolume as input parameters to the external function 'xmfact' which, in turn, activates a series of procedural routines to perform the first, second and third geometric tests on the face as described in sections 5.3.3, and

- . generate a utility frame 'goal ^context change\_adjacent\_face\_status ^argument <mface-id>' in the global database.

The first condition of the above rule is to match the utility frame 'goal ^context reason\_group(1)\_face ...'. In effect, the utility frame functions as an agenda item for initiating the recognition of group(1) faces. The remaining matching conditions in the above rule are essentially the characteristic pattern of a group(1) face as defined in section 5.3.2. Apart from performing the geometric tests, the external function 'xmfact' also modifies the access and status utility flags of the face frame according to the geometric test results and adds the recognized feature information in the cavity volume B-rep if recognition is successful.

For example, Fig. 7.10 shows a fragment of the recognized feature information stored in the B-rep of the subvolume\_2.

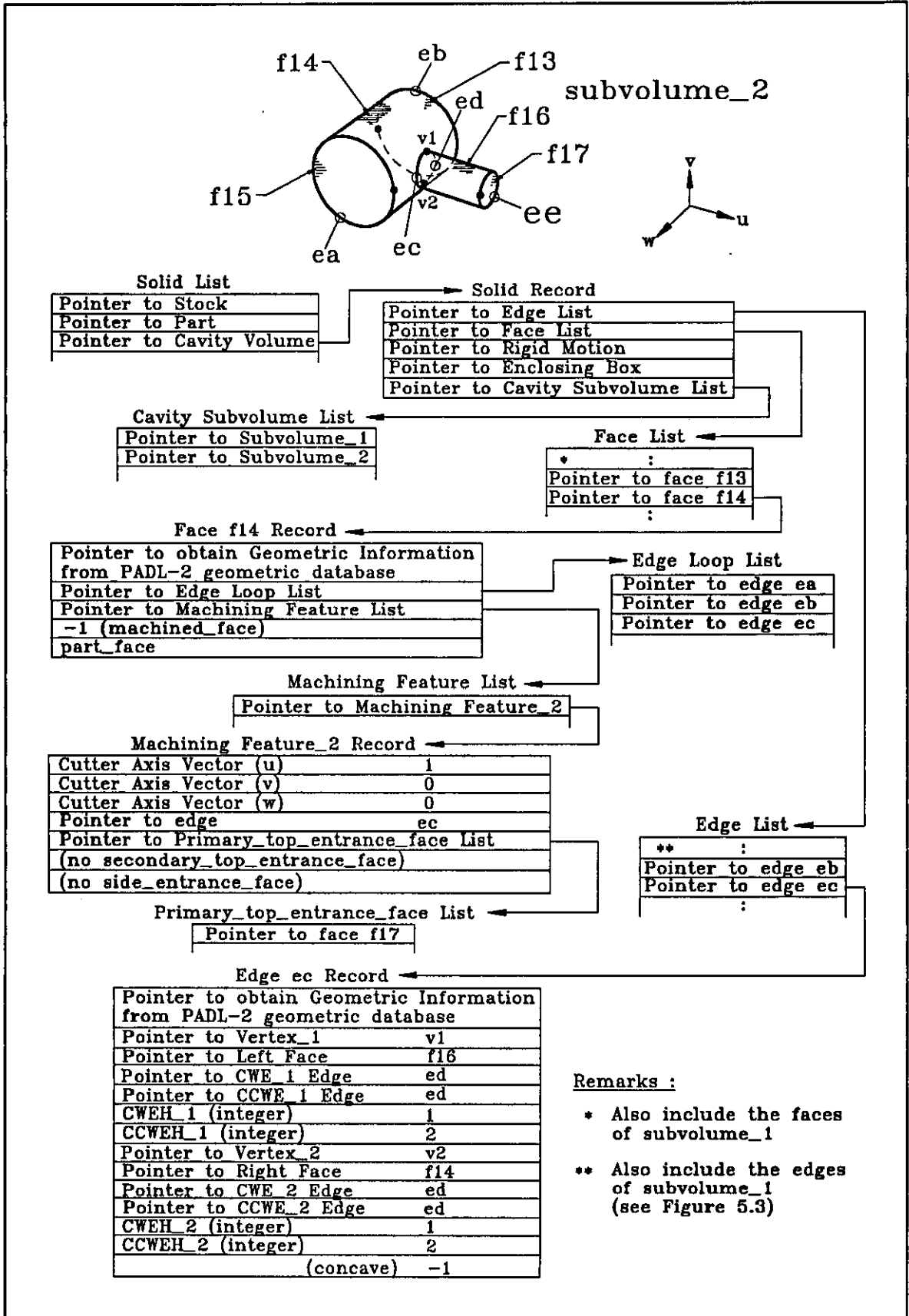


Figure 7.10 : A fragment of the recognized feature information of the subvolume\_2.

The utility frame 'goal ^context change\_adjacent\_face\_status ...' is used to activate the following rule for changing the status of the adjacent faces :

```
(p change_adjacent_face_status
  (goal ^context change_adjacent_face_status ^argument <mface-id>)
  (face ^identity <mface-id> ^status part_face)
  (edge ^lftfac <mface-id> ^rhtfac <checkface-id> ^status nil ^clascd convex
    ^identity <edge-id>
    {<fac> (face ^identity <checkface-id> ^status nil)})
-->
  (modify <fac> ^status check_face) )
```

The above rule reads as :

Rule name : change\_adjacent\_face\_status

- If
- . the goal is to change the status of the adjacent faces of <mface-id> ,
  - . there is a face frame whose face identity is addressed by the pointer <mfaceid> and whose status utility flag is part\_face,
  - . there is an edge whose characteristics are as follows :
    - left adjacent face is the same as the face addressed by the pointer <mface-id>
    - right adjacent face is addressed by the pointer <checkface-id>
    - status utility flag is addressed by the pointer <edge-id> , and
  - . the status utility of the face addressed by the pointer <checkface-id> is nil,

Then

- . change the status utility flag of <checkface-id> to 'check\_face'.

The above rule is essentially an implementation of the rule stated in section 5.3.4 (e). It can be seen that the above rule will not fire if the tested candidate face fails either one of the three geometric tests. This is because the status of the candidate face will not be changed to 'part\_face' by the geometric testing routines if the geometric test is not successful.

The implementation of the algorithm for the recognition of the group(2) face and group(3) face is done in a similar manner. For instance, the utility frame 'goal ^context reason\_group(2)\_face ...' is used to activate another set of production rules to select

and test the group(2) faces. Similarly, the utility frame 'goal ^context reason\_group(3) face ...' is used to invoke another set of rules to test the group(3) faces.

As the status and access utility flags in the face frames and edge frames of the global database are modified by the action of the rules during the recognition process, this has the effect of adding constraints in the cavity subvolume frame-based description in terms of reducing the number of frames matchable by the rules. As a result, the efficiency of the recognize-act cycle increases.

Since the access flag of a face will be changed to zero when the face fails the geometric test, the following rule is designed to match this access flag signal so as to report any faces that have failed the geometric test :

```
(p report_failed_face
  (goal ^report ^argument <subvolume_name>)
  {<fac> (face ^access 0 ^identity <face-id>)})
-->
  (call xwedfc <face-id> <subvolume_name>)
  (write crlf | Face | <face-id> | fails the geometric tests ! | )
  (modify <fac> ^access -1) )
```

The meaning of the above rule is :

Rule name : report\_failed\_face

If . the goal is to report a face that has failed the geometric test, and  
 . there is a face whose access utility flag has been changed to zero by the  
 geometric testing procedures,

Then

- . highlight the wireframe image of the face,
- . inform the user that the face fails the geometric test, and
- . change the access utility flag to -1 so as to avoid recursive firing of this rule.

Recalling that the utility frame 'goal ^context report ...' is less recent than the three frames used for activating the testing of the three groups of faces, the above rule will fire only after the testing process of the three groups of faces. Similarly, the

following rule is designed to make use of the least recent utility frame 'goal\_return\_to\_command\_parser' to return the system control to the command parser when all the rules relevant to the recognition process cannot be fired :

```
(p return_to_command_parser
  {<ret> (goal ^return_to_command_parser)}
-->
  (remove <ret>)
  (call xcompars)) ; external routine to call the command parser
```

## 7.5 Implementation of the Feature Learning Approach

The hypothetical part shown in Fig. 6.2 is again used here to facilitate the description of the implementation. The following initial conditions are assumed :

- (1) the system has not learnt the shape of the cavity volume 'X' before,
- (2) the feature recognizer has been used but fails to recognize the cavity volume 'X' due to its T-slot-like shape,
- (3) the cavity volume 'X' is going to be used as a positive teaching example, and its B-rep database still exists in the solid modeller.

### 7.5.1 Teaching Feature Description

The feature learning process is initiated by inputting the command 'learn/mv\_<integer>', where mv\_<integer> is the system given name of the cavity subvolume. For the current example, the cavity volume 'X' will be named as mv\_1 as it does not contain any subvolume. The corresponding command procedure displays the wireframe image of the cavity volume on the screen. At the same time the user is prompted to select a face of the cavity volume which will be used as a part\_face. As described in section 6.2.1, the face f1 shown in Fig. 7.11 is to be selected as the part\_face.



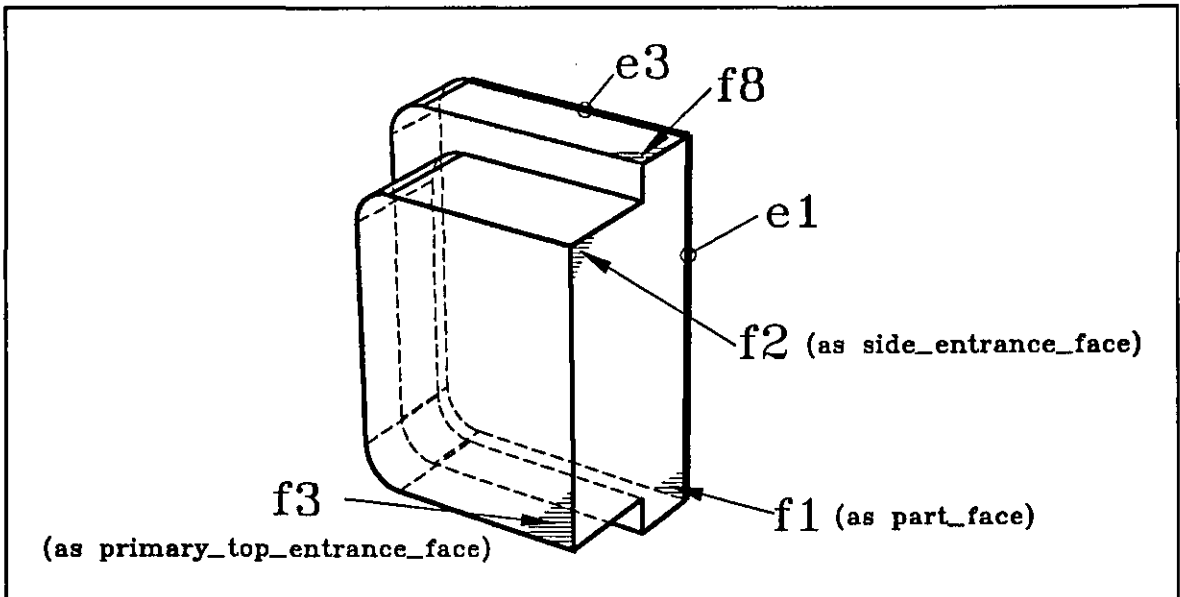


Figure 7.11 : Instructing machining faces.

The implementation of the face selection process makes use of the interactive input facility of the system. More specifically, a face is selected by means of inputting two of its boundary edges. The input of an edge is done interactively by positioning the cursor of the mouse pointing device near the wireframe display of the edge. The coordinates of the confirmed cursor point 'p' is obtained via the use of the GKS [Bono87] input device support routines. The perpendicular distance between the point 'p' and a boundary edge of the cavity volume is then calculated. The edge that is nearest to the picked cursor point is highlighted. The second edge is input in the same manner.

With the two input edges, the desired input face is determined as follows. Assuming that e1 and e3 are the two input edges (Fig. 7.11), and using the winged-edge B-rep database, their adjacent faces can be retrieved as :

<u>Picked Edge</u>	<u>Adjacent Face #1</u>	<u>Adjacent Face #2</u>
e1	f1	f2
e3	f1	f8

By simple comparison, the face (f1) that owns both of the two input edges is determined as the selected part\_face. The selection of the side\_entrance\_face (f2) and the primary\_top\_entrance\_face (f3) is done in the same manner. The instructed machining face information is stored in the B-rep database as shown in Fig. 7.12.

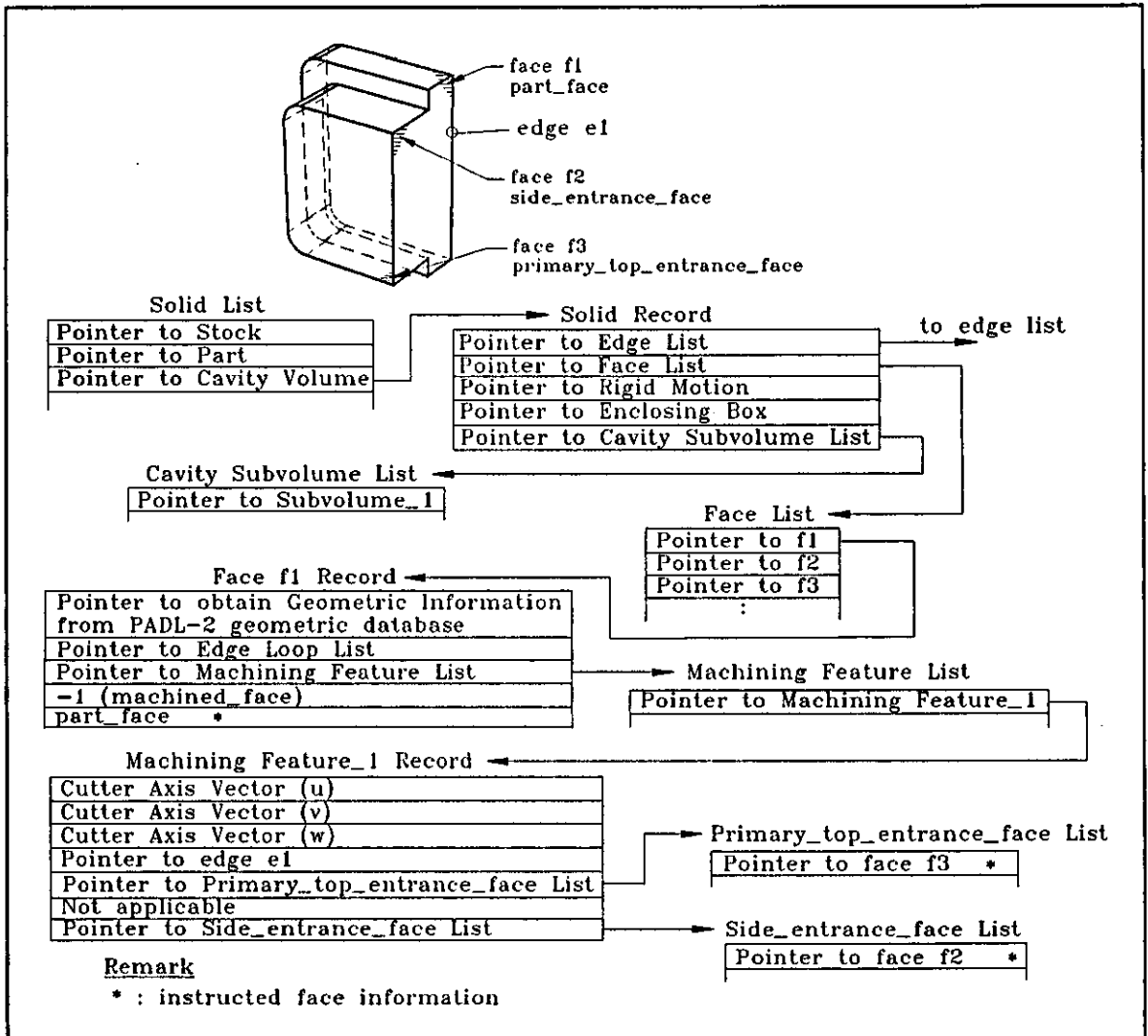


Figure 7.12 : Storing the instructed machining face information in the B-rep database.

### 7.5.2 Memorizing the Taught Feature

The user is then asked to enter a unique name, say 'TSLOT', for the cavity volume. The acquired name is stored in a default file called 'LEARN.NAM' which is used to maintain the names of all the features previously taught by user. The system then traverses the winged-edge B-rep database of the cavity volume and collects relevant B-rep data to automatically code a set of OPS5 rules according to the format and structure as described below. The user given name 'TSLOT' is used as a basis for naming the coded rules, and the integer identities of the faces and edges are used as binding variables in the rules.

Ideally, the complete boundary characteristics of the cavity volume would have been coded in a single rule as a matching template. However, this cannot be implemented because an OPS5 production rule only allows a maximum of 32 positive condition elements. Thus the boundary characteristics of a cavity volume are described separately in a number of rules. The first rule 'TSLOT-1' shown below can be considered as the header of the entire set of new rules. It matches the utility frame 'goal ^context TSLOT' and then generates the utility frame 'goal ^context TSLOT-2' as a message to invoke the second rule.

```
(p TSLOT-1
  {<recollect> (goal ^context TSLOT)}
-->
  (remove <recollect>)
  (make goal ^context TSLOT-2))
```

As shown below, the second rule 'TSLOT-2' defines the characteristics of the three instructed faces and the two edges (e1 and e2) shared between the three faces (Fig. 7.13).



For the current example, since the cavity volume has 14 faces (Fig. 6.4), while the second rule has already matched the 3 instructed machining faces, so there are 11 faces remaining. For each of the 11 faces, a rule is coded to match the face's characteristics. For instance, for matching the face f10 (Fig. 7.13), the rule would be:

```
(p TSLOT-3
  {<tslot> (goal ^context TSLOT-3)}
  {<v1> (face ^edgcnt 4 ^clascd mcfac ^factyp pln ^status nil ^identity
<f10>)}
  (edge ^rhtfac <f10> ^edgtyp lin ^clascd convex) ; e4
  (edge ^rhtfac <f10> ^edgtyp lin ^clascd convex) ; e5
  (edge ^lftfac <f10> ^edgtyp lin ^clascd smooth) ; e6
  (edge ^lftfac <f10> ^edgtyp lin ^clascd concav) ; e7
-->
  (remove <tslot>)
  (modify <v1> ^status marked)
  (make goal ^context TSLOT-4))
```

So for the current example, there are altogether 12 rules used (from the second to the thirteenth rules) to memorize the boundary characteristics of the cavity volume. Each rule activates its succeeding one by means of generating a utility frame. In effect, the 12 rules are virtually linked together as a single rule that describes the boundary shape of the cavity volume as represented by the face/edge graph in Fig. 6.4. The number of rules coded by the system for memorizing the boundary characteristics of a cavity volume is equal to  $n-2$ , where  $n$  is the total number of boundary faces of the cavity volume.

To close the rule set, an additional rule is coded :

```
(p TSLOT-14
  {<tslot> (goal ^context TSLOT-14)}
  - (face ^status nil)
  (face ^status part_face ^identity <f1>)
  (face ^status side_entrance_face ^identity <f2>)
  (face ^status primary_top_entrance_face ^identity <f3>)
-->
  (remove <tslot>)
  (call xmkreg <f1> <f2> <f3>))
```

As the previous rules are virtually linked together serially as a single rule, it means that one of the conditions for the above last rule to fire is that all the previous rules should have fired. In other words, the first matching condition ensures that the boundary faces of the cavity volume have been successfully matched. The second matching condition requires that the global database does not contain a face whose status is nil. Thus the second matching condition essentially ensures that the matched cavity volume has the correct number of boundary faces. The remaining matching conditions in the rule have the effect of retrieving and passing the three instructed machining faces as input arguments to the external subroutine 'xmkreg' in the action part of the rule. The external subroutine is used to add the machining face information in the B-rep database.

The coded rules are written to a file with 'TSLOT.OPS' as the file name. To incorporate the new rules in the system, the 'TSLOT.OPS' source file is compiled into an object code file which is linked with the old object code files of the system to produce a new binary executable image.

### 7.5.3 Recollecting the Learnt Feature

Assuming that a similarly shaped cavity volume 'Y' as shown in Fig. 6.5 is subsequently encountered and it cannot be recognized by the feature recognizer as its shape is similar to that of cavity volume 'X'. At this stage, the user can retrieve the previously learnt features by issuing the following commands in sequence :

```
'flush'
'bframe/mv_1'
'recollect'
```

The first command is used to clear the global database so as to ensure that any old cavity volume description in the global database is removed. The second command loads the frame-based description of the cavity volume 'Y' in the global database. The

third command is used to activate the sets of rules that have been created in previous learning exercises for matching with the frame-based description of the cavity volume 'Y'.

More specifically, the corresponding command procedure opens the previously mentioned file 'LEARN.NAM' and uses each of the names stored in the file to generate a corresponding utility frame in the global database. Recalling that the names stored in the file are actually the user given names of the previously learnt cavity volumes, and since the name 'TSLOT' is in the file, the utility frame 'goal context TSLOT' is inserted in the global database. This utility frame acts as a message to invoke the first rule 'TSLOT' described in the previous section. As the cavity volumes 'X' and 'Y' have identical boundary characteristics, the entire set of 'TSLOT' rules will fire. The last rule identifies the corresponding three machining faces (fa, fb and fc in Fig. 7.14) of the cavity volume 'Y' that have geometrical and topological characteristics similar to those of the three faces defined in the set of 'TSLOT' rules.

The 'xmkreg' external routine adds the three machining faces as machining feature information in the B-rep database of the cavity volume. The enhanced B-rep database of the cavity volume 'Y' is illustrated in Fig. 7.14 which is basically the same as that of the cavity volume 'X' shown in Fig. 7.12. Also by comparing Figs. 7.10 and 7.14, it can be appreciated that the machining feature information obtained by using the feature recognition approach and the feature learning approach is basically the same and is represented by the same data structure in the B-rep database.

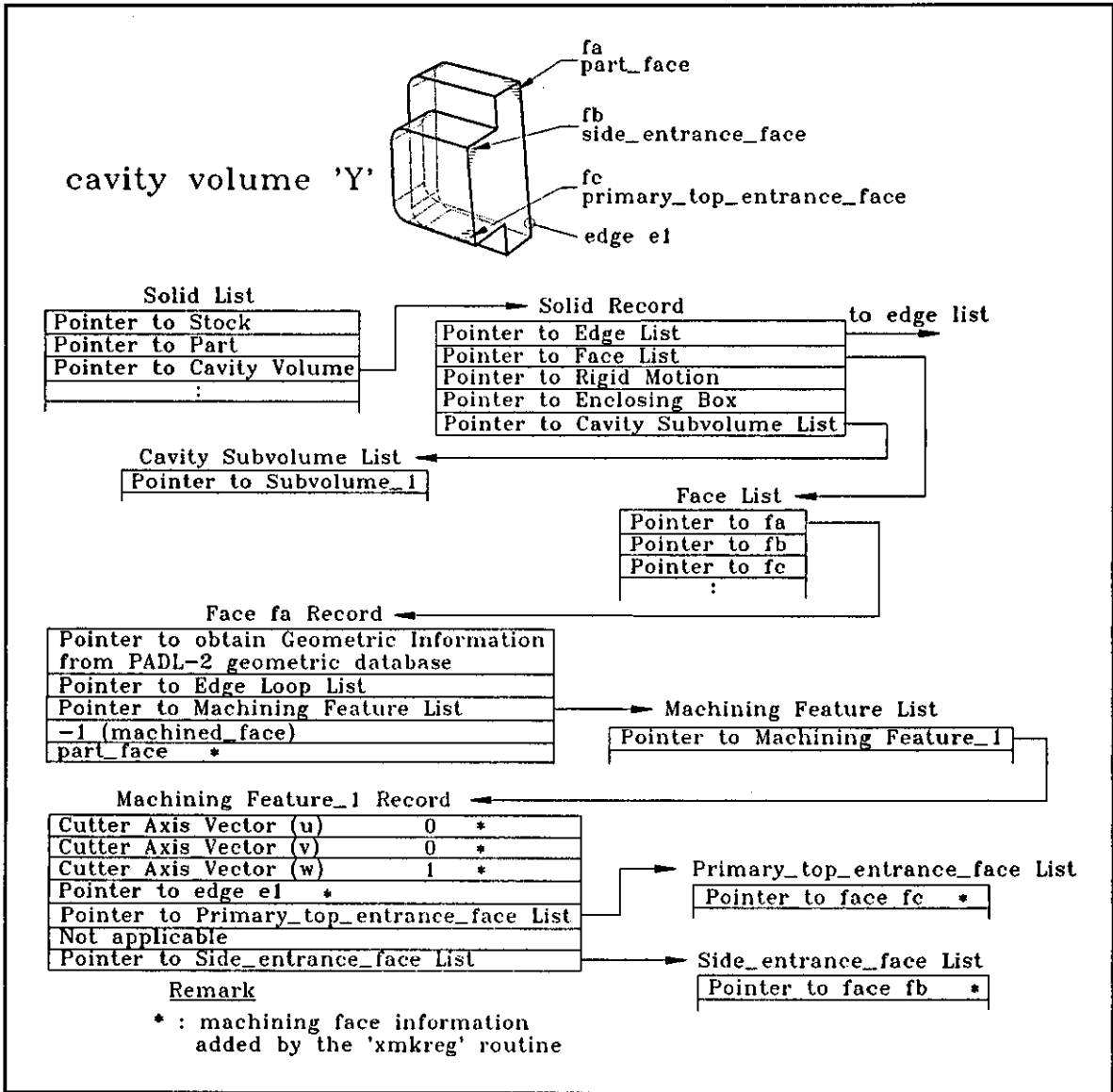


Figure 7.14 : Instructed machining faces added in the B-rep.



It can be seen that in the current prototype system, the activation of the above three commands is done manually. The reason of such an implementation is mainly for the purpose of distinguishing the two approaches more clearly. In fact, the two approaches can be easily coupled together by incorporating the three commands in the feature recognition algorithm so that when the feature recognition approach fails, it can automatically retrieve the previously learnt feature templates for matching.

## 7.6 Discussion

Having described the methodology and implementation aspects, the advantages and disadvantages of the two adopted approaches are now discussed.

### 7.6.1 The Feature Recognition Approach

Many feature recognition methods [Henderson84, Joshi88, etc.] are designed for recognizing general form features only. As form features are different from machining features, these methods need to have a separate, post-recognition tool accessibility analysis for validating the machinability of the recognized form features. The feature recognition algorithm used in this thesis is unique in the sense that it virtually simulates the human behaviour of recognizing machining features first by focusing on a potential part\_face and then assessing the tool accessibility of the potential part\_face. The author has the following two arguments for the incorporation of machining heuristics and tool accessibility analysis in a machining feature recognition algorithm :

- (a) It has been an accepted principle that features are application specific, and hence a form feature is considered as a machining feature based on machining application considerations. For instance, as illustrated by Pratt's [Pratt87] example in section 1.4, a depression in a part can be interpreted either as a web space formed by reinforcing ribs (viewpoint based on casting, welding, etc.) or as a

machining pocket (viewpoint based on machining). Thus the author feels that for recognizing machining features (and not general form features), it would be beneficial to exploit as much machining related considerations as possible in the recognition process so that any candidate features that have invalid machining properties such as a 'pocket' without a tool entrance face or without round corners can be detected as soon as possible in the recognition process. These important recognizing results could be made known to the user during the recognition process for further actions or at least for further contemplation, and not after the recognition process.

- (b) The difference between the approach that exploits machining related knowledge in the recognition process for identifying machining features and the approach that performs tool accessibility as a post-recognition process is not only a matter of time difference. This implies that without the ingredient of machining technology in the recognition process, the latter recognition strategy will likely find difficulty in resolving machining feature interactions since the recognition mechanism will have to rely mainly on general form feature (geometric and topological) reasoning or pattern matching (assuming that other feature information such as tolerance information is not available). Shah [Shah91a] used an abstract term called 'conjugate feature' to refer to complex features that are formed due to feature spatial interactions or due to alternative feature interpretations based on diverse application considerations. He also postulated that sophisticated, application specific conjugate feature transformation (feature recognition) would be required for obtaining a correct and comprehensive feature interpretation. The use of an aspect vector (cutter approach direction) as discussed in Corney's [Corney91a, Corney91b] work can also be considered as another example of utilizing machining knowledge very early in the machining feature recognition process. Thus the use of the machining heuristics and ray-casting accessibility analysis in the algorithm has the crucial effect of assisting form feature reasoning, and as a result the following significant benefits have been realized :

- (i) can handle rather complicated feature situations including non-orthogonal feature interaction,
- (ii) the machinability of the recognized machining features is ensured, and
- (iv) the ray-casting process can detect alternative `primary_top_entrance_faces` and `secondary_top_entrance_faces` of a machining feature which are very useful for process planning.

The ray-casting technique has been employed as a less rigorous and less computational expensive analysis for tool accessibility. Other more precise and costly methods can be used : surface oriented and volume oriented.

In the former method, the half-spaces of the `check_faces` (wall faces) of the potential `part_face` will intersect with the other half-spaces of the cavity volume (as well as with the half-spaces of the part for global accessibility test) for determining an intersection boundary 'X'. This intersection boundary 'X' will be compared with the boundary edges 'Y' of the potential `part_face`. If 'X' is identical to 'Y' or 'X' totally encloses 'Y', then the potential `part_face` is obstruction free, otherwise the reverse will be true.

In the latter method, the boundary edges 'Y' of the potential `part_face` will be swept linearly along a cutter axis vector to create a sufficiently long virtual object. This very long virtual object is basically a simulated image of the tool swept volume above the potential `part_face`. Its very long length can be determined by using information that is related to the dimensions of the starting stock or the finished part. This simulated tool swept volume can be Boolean subtracted (or intersected) with the finished part. If the resultant intersecting volume is null then tool accessibility is satisfied. However, it is anticipated that these two methods would require very computational expensive processes such as edge/edge comparison, area calculation and boundary evaluation, and hence these two methods are not adopted in the algorithm.

The efficiency of the feature recognition algorithm now depends very much on the efficiency of the ray-casting algorithm which however, can be improved by using more efficient ray-casting algorithms [Weghorst84]. Moreover, as the ray-casting algorithm is basically a version of the better known 'clipping' algorithm commonly used in the CAD/CAM community, many modern CAD/CAM development systems also provide the facility of calling a ray-casting utility procedure that has been firmwared in their electronic circuitry. Hence, the use of the ray-casting technique for tool accessibility analysis should not be a serious concern for enhancing the efficiency of the feature recognition algorithm.

Although the CSG based PADL-2 solid modeller is used in the prototype system, the algorithm actually works with the boundary representation database, and hence the algorithm can be easily adopted in boundary representation systems. In addition, the algorithm could also be embedded in a feature based design systems as a procedure for checking the machinability of a designing part.

The combination of the knowledge based environment with the solid modeller also offers significant advantages for implementing the algorithm. The production rule programming paradigm allows a concise and symbolic embodiment of the feature recognizing knowledge in the system, and hence the development and maintenance of the algorithm are much facilitated. The inference engine (recognize/act cycle) of the KBS is designed for symbolic manipulation, and is therefore exploited to simulate the human function of recognizing feature characteristic conditions. Whenever, numerical computation or database communication is required, the algorithm will switch to the use of procedural routines. In this way, a good match of jobs with the correct types of working tool is maintained. With the command interpreter utilities of the KBS, the recognition process can also be performed in a more interactive manner. In the current prototype implementation, the user can interactively perform various activities such as inspecting the status of the working memory elements in the global database and tracing the rules that have been fired or that will be fired during the recognition process. With a more sophisticated implementation, the system could be made more interactive such

as using previously fired rules to explain why a former decision has been made.

The machining features extracted by the algorithm are essentially generic 2.5D machining regions that have not been differentiated clearly into different feature types. Moreover, there is only one cutter axis vector associated with a machining region because the ray-casting accessibility analysis is not repeated on the other faces of the machining feature once a valid `part_face` is located. This is a significant shortcoming as it precludes other alternative interpretations of a machining feature. Further work needs to be done to enhance the extracted feature content with more meaningful feature information so that other manufacturing activities such as process planning can be fully automated. The refinement work on this part will be discussed in more detail in chapter 9.

### 7.6.2 The Feature Learning Approach

This approach can be considered as a remedy of the first method that can only deal with 2.5D machining features. Besides, this approach itself also represents a novel means of extending the recognition ability of the system to adapt to diverse manufacturing conditions. The author has deliberately used the feature representation scheme adopted in the first approach for representing custom features in this second approach so as to maintain a uniformity of feature representation in the system. In addition, the process of instructing new custom features has also been designed to be interactive and without the use of a programming language. This is important as the teaching of custom features to the system is supposed to be done in an on-line mode by practical engineering personnel of a factory rather than in an off-line mode by a software knowledge engineer.

In the current implementation, the method only permits the user to teach one machining feature in a cavity volume. Moreover, the description of a feature based on the instruction of the three machining faces (`part_face`, `side_entrance_face` and

primary\_top\_entrance\_face) may not be sufficiently general to account for all possible situations. For instance, a non-2.5D machining feature may have alternative side\_entrance\_faces, and an erroneous instruction of an alternative side\_entrance\_face may lead to undesirable results such as generation of a faulty cutter path that will collide with the machined part.

The representation of cavity volume shape is by means of a defined set of face and edge conditions, and testing of shape similarity is by matching the face/edge conditions of a previously learnt feature with the corresponding face/edge conditions of a new feature. However, it is not clear whether there exists a theoretical, adequate set of conditions for governing a reliable testing of shape similarity. Intuitively, it is postulated that the more matching conditions used (provided that the conditions are not redundant), the more stringent will be the shape matching process, and the more reliable will be the shape similarity testing.

The approach has taken the view that every feature example presented to the system to be learnt is a totally new feature that has no connection with the previously learnt features in terms of shape similarity. Consequently, each set of new rules added to the system is completely independent, and the system has no control on the possibility of generating redundant rules. Hence, the number of new rules incorporated in the system can easily grow to an impractical size. At the same time, the efficiency of the recognize/act cycle will also decrease to an inadmissible level. A possible method for improving this shortcoming will be discussed in chapter 9.

## CHAPTER 8

# VERIFICATION OF WORK

This chapter presents a practical elucidation of using the machining feature information produced by the two approaches for downstream manufacturing planning operations. For this purpose, two simple software modules are developed to post-process the machining feature information established in the B-rep database. The first module is basically a simple machining operation sequencer which puts machining features of identical cutter axis vector together in a group, and sequences the machining features in each group for machining. The result of the first module is stored in a machining operation file as a machining operation agenda to generate NC cutter paths for the machining features.

It is emphasized that the two simple modules are mainly developed and used for the purpose of verifying the practical usefulness of the feature recognition and learning software. They do not represent a formal study of the various process planning activities such as set-up planning and process planning. For more substantial work in these areas, the reader can refer to other publications such as [Murray86, Gindy91, and Sakuari91].

### 8.1 Grouping and Ordering Machining Features

The first module is activated by inputting the command '`oplan/<fname>`', where `<fname>` is the user given name for the machining operation file that is going to be output. The corresponding command procedure manipulates the feature information established in the B-rep data base as described below.

### 8.1.1 Retrieving Machining Features

The extracted machining features of the cavity volume represented in the B-rep database are identified by examining the fifth field of the face record. If the fifth field indicates that the face is a `part_face`, then the machining feature list pointer stored in the third field of the face record is used to retrieve the information of the machining features associated with the face. For instance, for the hypothetical part used in chapter 5, the machining feature information represented in the B-rep database of the cavity volume (subvolume\_1 and subvolume\_2) is depicted in Figs. 8.1 and 8.2. In summary, the following information about a machining feature can be obtained directly from the B-rep database :

- (1) `part_face`,
- (2) cutter axis vector (i.e. cutter approach direction),
- (3) an edge belonging to the concerned edge loop of the `part_face`,
- (4) `primary_top_entrance_face`,
- (5) `secondary_top_entrance_face` (if there are any), and
- (6) `side_entrance_face` (if there are any).

### 8.1.2 Grouping the Machining Features

Machining features that have identical cutter axis vector are put together in a group. This grouping is based simply on the notion that machining features with the same cutter axis vector can potentially be machined in the same machining set-up. For example, the subvolume\_1 shown in Fig. 8.1 has two machining feature groups. The first group has two machining features, while the second group has one machining feature as summarized in the following table :

Machining Feature Group	Cutter Axis Vector			Machining Feature	Part_face	Primary_top_entrance_face
	u	v	w			
1	0	0	1	1	f10	f1
1	0	0	1	2	f8	f1
2	-1	0	0	1	f6	f12

Table 8.1 : Machining features of subvolume\_1.



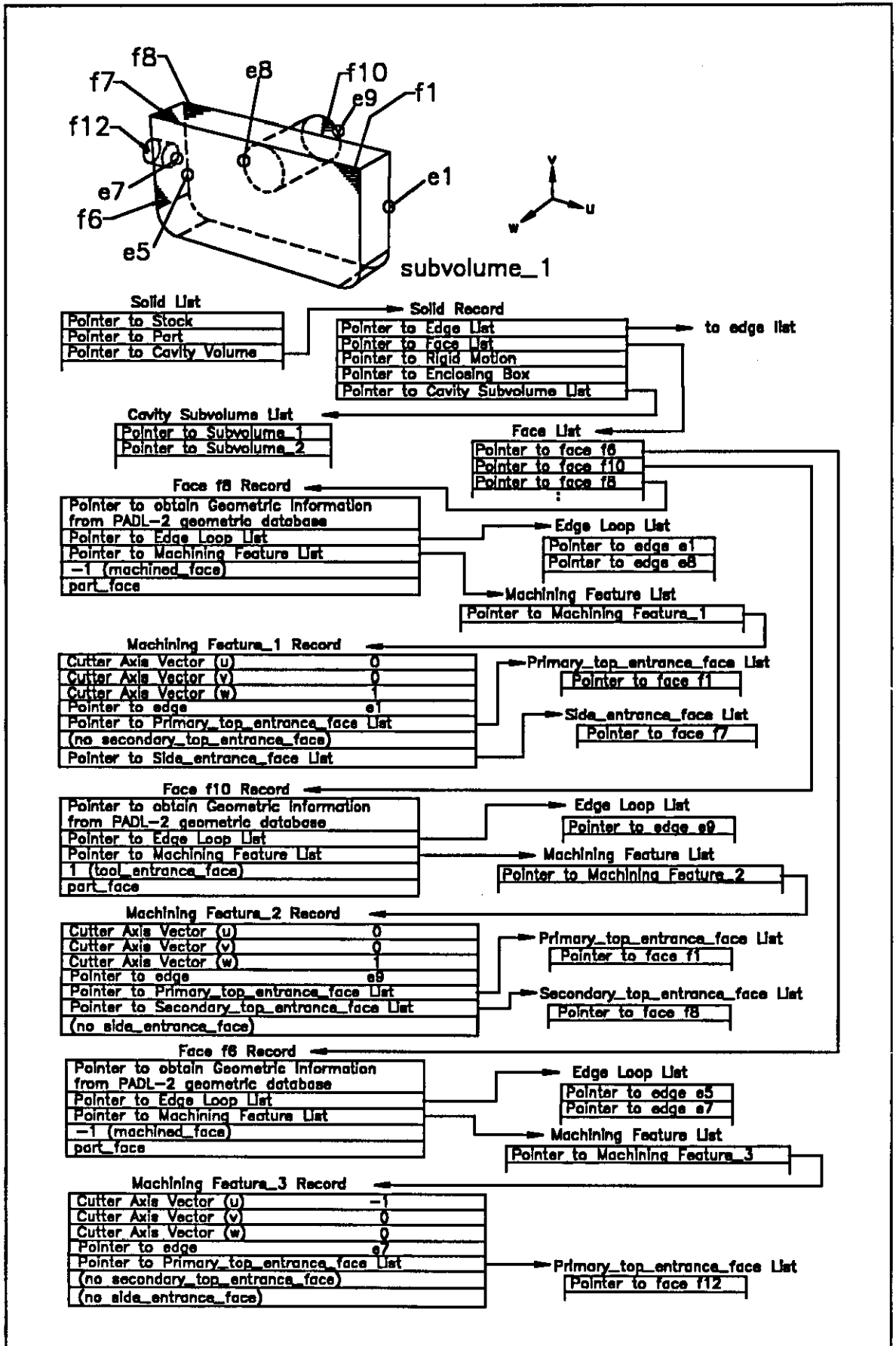


Figure 8.1 : The B-rep of subvolume\_1 enhanced with feature information.

Similarly, the grouping of machining features of subvolume\_2 (Fig. 8.2) is summarized in the following table :

Machining Feature Group	Cutter Axis Vector			Machining Feature	Part_face	Primary_top_entrance_face
	u	v	w			
1	0	0	1	1	f13	f15
2	1	0	0	1	f14	f17

Table 8.2 : Machining features of subvolume\_2.

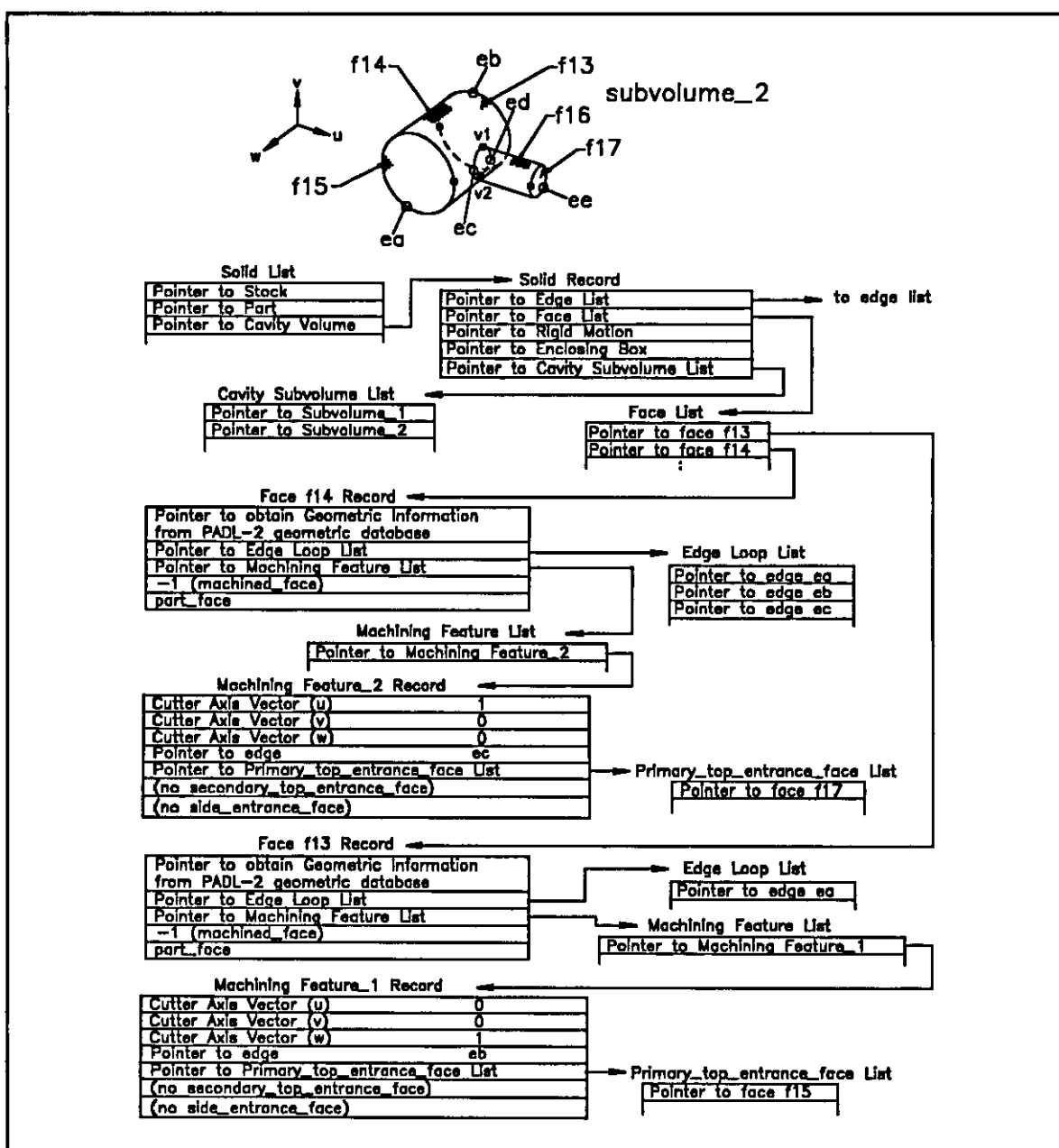


Figure 8.2 : The B-rep of subvolume\_2 enhanced with feature information.

### 8.1.3 Resolving Identical Features Condition

As discussed in section 4.4, machining features that have condition type 1 part\_face, such as the through hole illustrated in Figure 8.3, will be extracted by the feature recognizer as two machining features. The presence of this kind of machining feature condition is identified by checking whether the following conditions exist :

- (1) the cutter axis vector of two machining feature groups 'A' and 'B' are opposite to each other,
- (2) the part\_face of a machining feature 'i' in group 'A' is the primary\_top\_entrance\_face of a machining feature 'j' in group 'B', and vice versa, and
- (3) machining features 'i' and 'j' have the same set of check\_face(s) surrounding the part\_face.

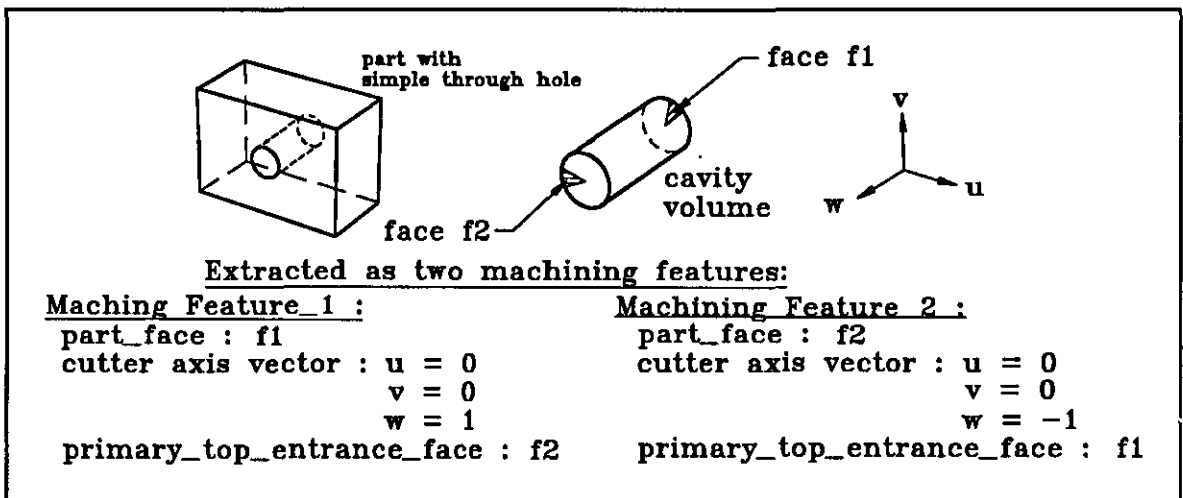


Figure 8.3 : Machining feature with part\_face condition type 1.

If the above conditions exist then either machining feature 'i' or machining feature 'j' is deleted since they can be machined by a single machining operation. The rule used for deletion is that if the number of machining features in groups 'A' and 'B' is different, then the machining feature belonging to the smaller group is deleted, otherwise the choice is made arbitrarily. For instance, if group 'A' has more machining features than group 'B', then machining feature 'j' is deleted. On the other hand, if

groups 'A' and 'B' have the same number of machining features, then either machining feature 'i' or machining feature 'j' can be deleted. The motive for using this decision rule for deleting redundant machining features is to reduce the group size of smaller machining feature groups so that if the size can be reduced to zero, the total number of machining feature groups (or machining set-ups) can also be reduced.

#### 8.1.4 Sequencing Machining Features

The machining features in a machining feature group are then sequenced according to the secondary\_top\_entrance\_face dependency relationship. For instance, for the first machining feature group of the subvolume\_1, the machining feature with part\_face 'f8' is ordered before the machining feature with part\_face 'f10' because the latter machining feature can use face 'f8' as its secondary\_top\_entrance\_face. This means that the former machining feature will be machined before the latter one (Table 8.3).

Machining Feature Group	Cutter Axis Vector			Machining Feature	Part face	Primary top entrance face	Secondary top entrance face
	u	v	w				
1	0	0	1	1	f8	f1	nil
1	0	0	1	2	f10	f1	f8

Table 8.3 : Sequencing of machining features in group 1 of subvolume\_1.

After the above sequencing process, those machining features that have a single cylindrical check\_face are grouped together in a subgroup. This grouping is based on the assumption that the machining features within the subgroup can be machined by using simple cylindrical hole drilling operations. Factors such as size, tolerance and surface finish of the machining features are not considered in this thesis.

Finally, the machining features within a subgroup are further divided into smaller groups of equal cylindrical diameter. The objective of this grouping is to machine equal sized holes together with the same cutting tool so that tool change and tool travelling time can be minimized. Also if the group of machining features form a higher level feature pattern, such as a pattern of holes on a pitch circle diameter, they can be machined in a more sensible manner. The method of grouping machining features here is necessarily simple. For a more substantial study on process capability modelling, references such as [Gindy90] can be pursued.

The post-processed machining feature information is then written to a machining operation file. Each record in the file represents a machining operation. For example, for the subvolume\_1 and subvolume\_2, the machining operation file would contain information as shown below :

Record No.	Cutter Axis Vector			Part face Identity	Edge Identity	Primary top entrance face Identity	Secondary top entrance face Identity	Side entrance face Identity
	u	v	w					
1	0	0	1	f8	e1	f1	nil	f7
2	0	0	1	f10	e9	f1	f8	nil
3	0	0	1	f13	eb	f15	nil	nil
4	-1	0	0	f6	e7	f12	nil	nil
5	1	0	0	f14	ec	f17	nil	nil

(Please refer to Figures 8.1 and 8.2 for the face and edge notations)

Table 8.4 : Machining operation file content of subvolume\_1 and subvolume\_2.

## 8.2 Cutter Path Generation

Having produced the machining operation file, the cutter path generation module is activated by issuing the command 'ncpath/<fname>', where <fname> is the name of the machining operation file just produced. The module opens the machining operation file and processes the file records sequentially.

The cutter path generation module computes cutter paths using the B-rep of the cavity subvolumes rather than the B-rep of the finished part since the machining feature information is represented with reference to the boundary of the cavity subvolumes. The cutter axis vector in each machining operation record represents the cutter approach direction, and hence it is used to determine a rotational transformation matrix (Appendix F) for transforming the orientation of the corresponding cavity subvolume in such a way that the cutter axis vector aligns with the system's z-axis. The z-axis is taken as the machine spindle axis in the cutter path generation module. For example, for the five machining operation records shown in Table 8.4, the corresponding orientation of the two cavity subvolumes are illustrated in Figure 8.4.

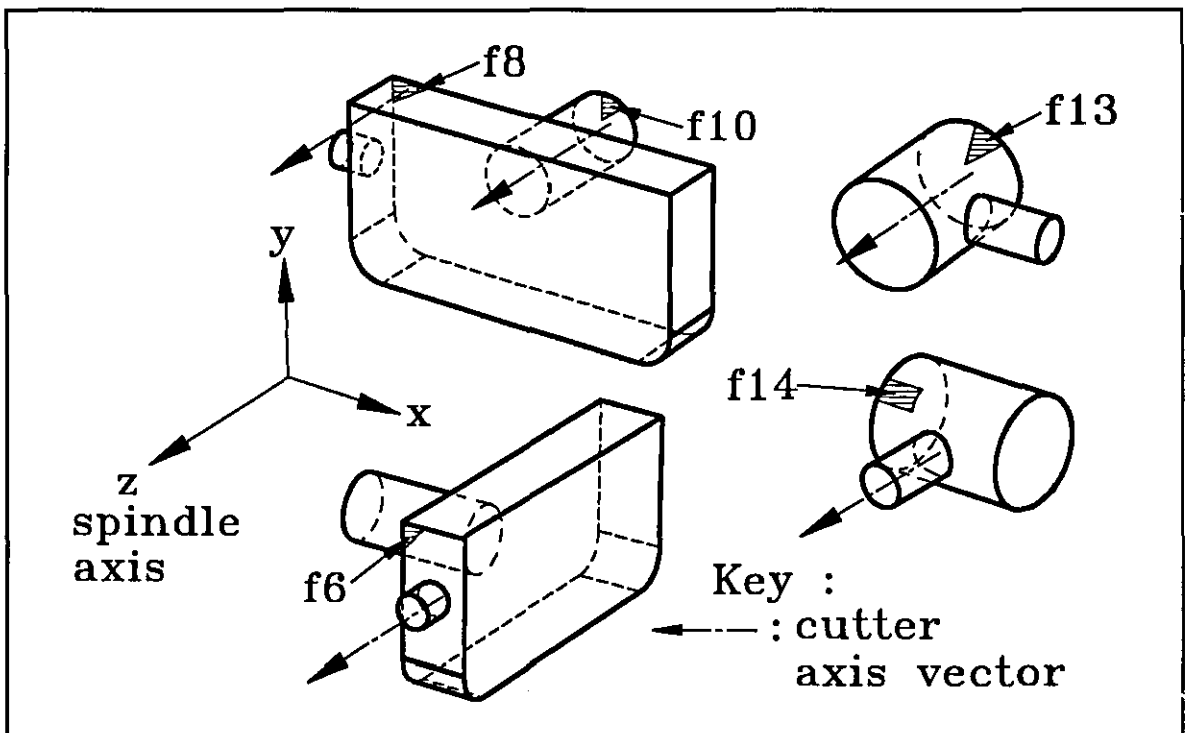


Figure 8.4 : Orientation of the machining features for machining.

Two types of cutter paths are generated depending on the part\_face conditions. The first type is essentially a drilling operation, and is used when the part\_face is surrounded by a single cylindrical check\_face. The cutter is directed to enter through the primary\_top\_entrance\_face (and secondary\_top\_entrance\_faces if there are any) and travel along the axis of the cylindrical check\_face to the part\_face. The total axial depth of cut is determined according to the part\_face condition types as discussed in section 4.4. For instance, if the part\_face is of condition type 1, the cutter path goes through the part\_face by an amount as described in section 4.4. The number 2 machining record shown in Table 8.4 is an example of such a through hole condition. The corresponding cutter path is illustrated in Figure 8.5.

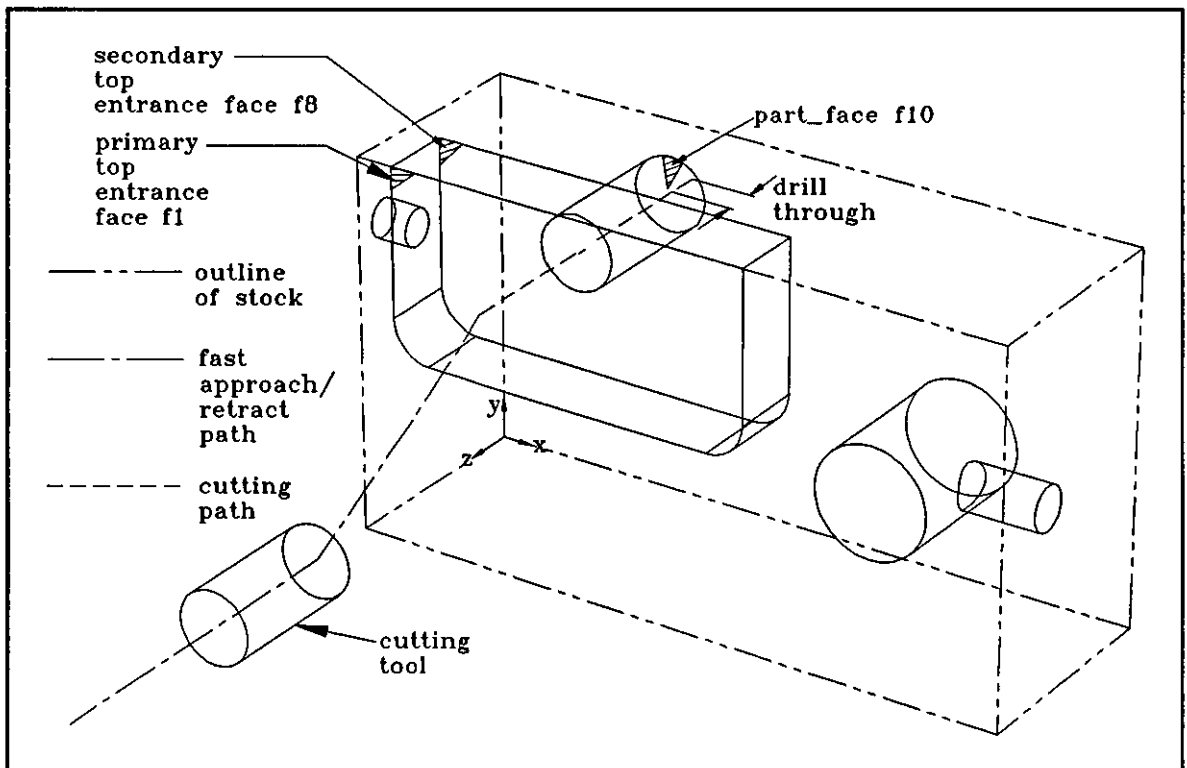


Figure 8.5 : Cutter path for machining record no. 2.

If the part\_face is of condition type 2 which represents a blind hole situation, the cutter stops right on the surface of the part\_face. Machining record number 3 is an example of such a blind hole condition, and the generated cutter path is shown in Fig. 8.6.

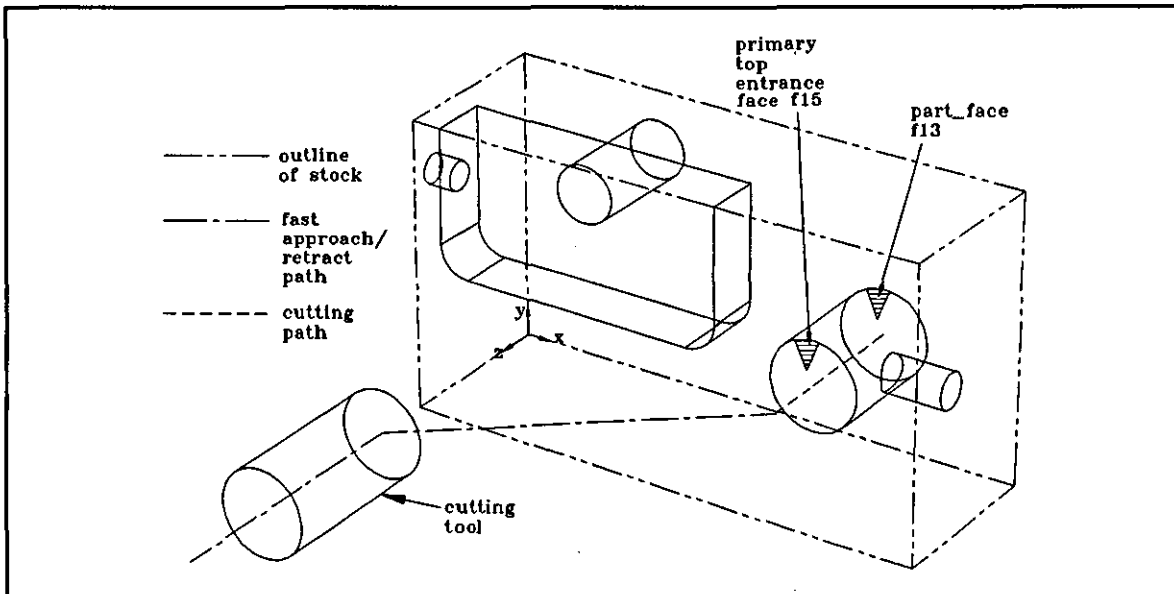


Figure 8.6 : Cutter path for machining record no. 3.

On the other hand, if the part\_face is of condition type 5 which represents an inner edge loop feature interaction, the cutter path also overshoots the surface of the part\_face by an amount as described in section 4.4. Examples of this condition are the machining records number 4 and 5 shown in Table 8.4. The corresponding generated cutter path is shown in Figs. 8.7 and 8.8.

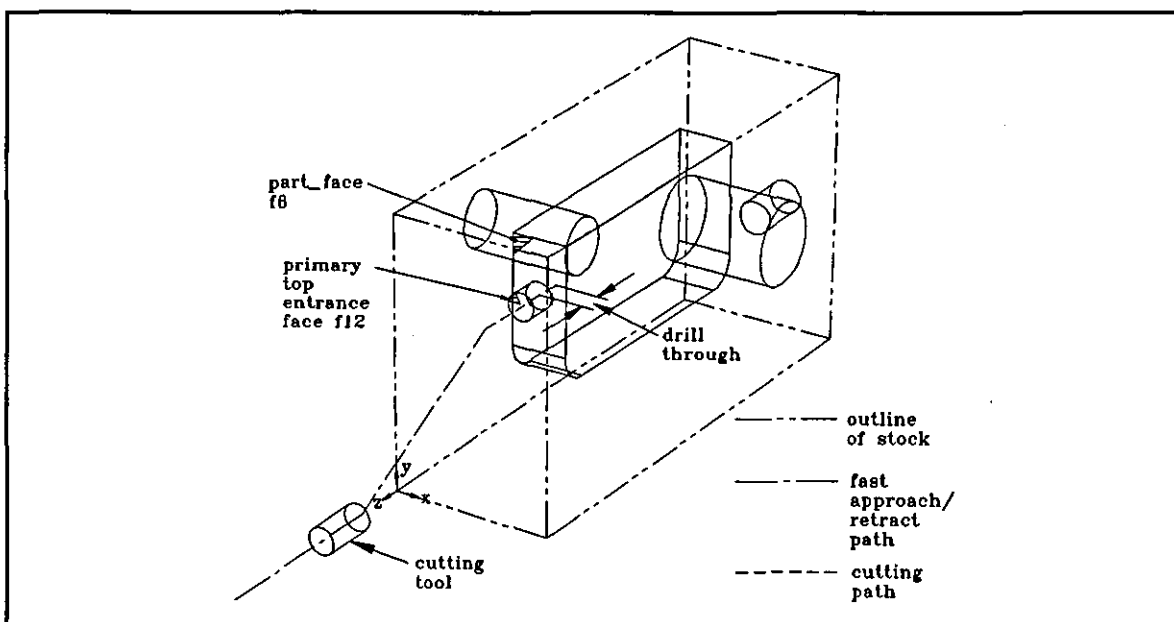


Figure 8.7 : Cutter path for machining record no. 4.



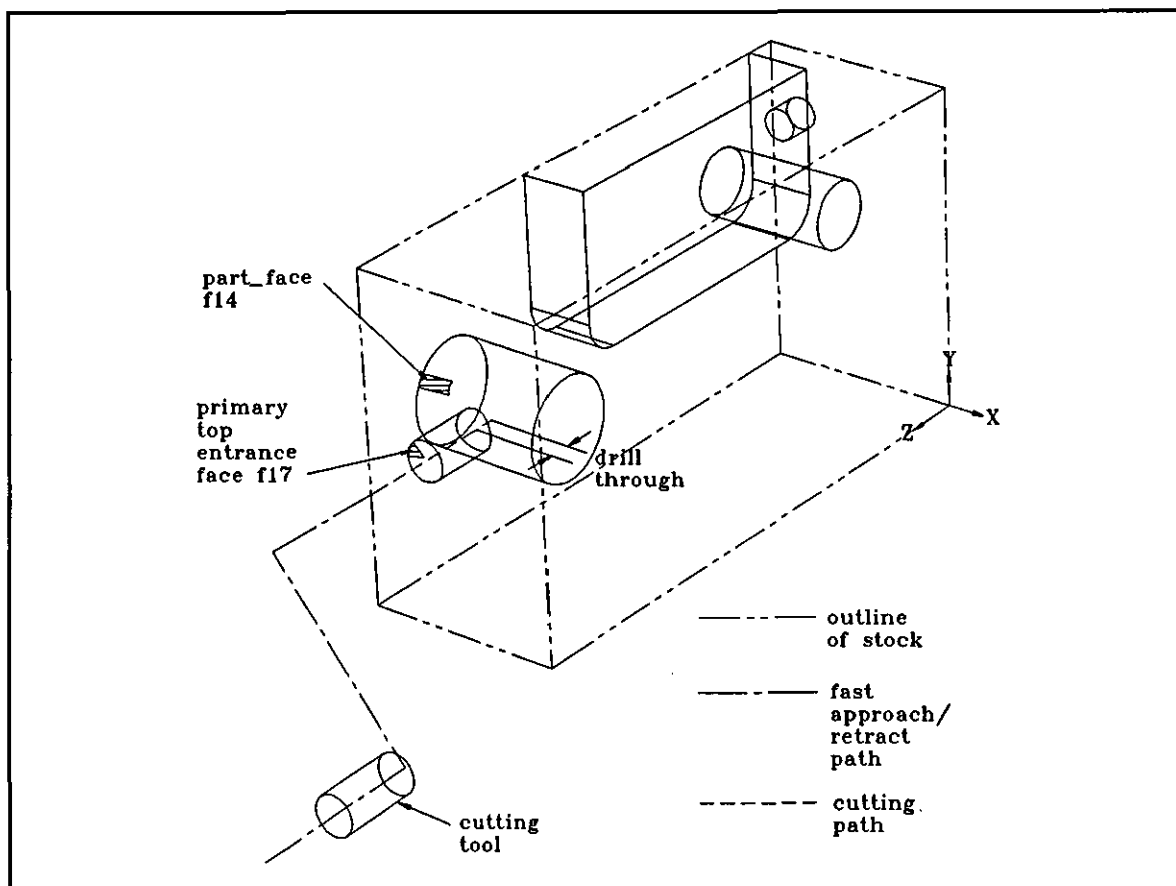


Figure 8.8 : Cutter path for machining record no. 5.

The second type of cutter path is used when the `part_face` is surrounded by more than one `check_faces`. At the outset, the boundary edges of the `part_face` are virtually offset [Tiller84, Saeed88] by an amount equal to the cutter radius which is specified by the user in the prototype system. The actual offset direction depends on the edge convexity. For convex boundary edges the offset is towards the inside of the cavity volume, while for concave boundary edges the offset is outside the cavity volume. The offset edges are trimmed or extended to form a polygon. A pattern of zig-zag cutter path is then generated within the polygon. The zig-zag cutter paths are used for clearing the material within the bounded region of the `part_face`. This zig-zag cutter path is essentially based on a fixed direction-parallel milling method. A contour-parallel milling method would be a better choice for milling profiles or pockets with an arbitrary contour shape. A good discussion of cutter path generation methods can be found in [Persson78, Held91].

The offsetting and zig-zag cutter path generation procedures are repeated on successive levels between the `primary_top_entrance_face` (or the last `secondary_top_entrance_face`) and the `part_face`. Each level of zig-zag cutter paths represents a layer of machining. The gap between two layers represents the axial depth of cut. Currently, the increment of axial depth of cut is implemented as a hard-coded value. If `side_entrance_face` is present, the cutter will enter and exit the machining region laterally through the first `side_entrance_face` represented in the `side_entrance_face` list, otherwise the cutter will enter vertically through the centre of an imaginary rectangle that bounds the `primary_top_entrance_face`. For example, the generated rough milling cutter paths for the machining record number 1 is shown in Fig. 8.9.

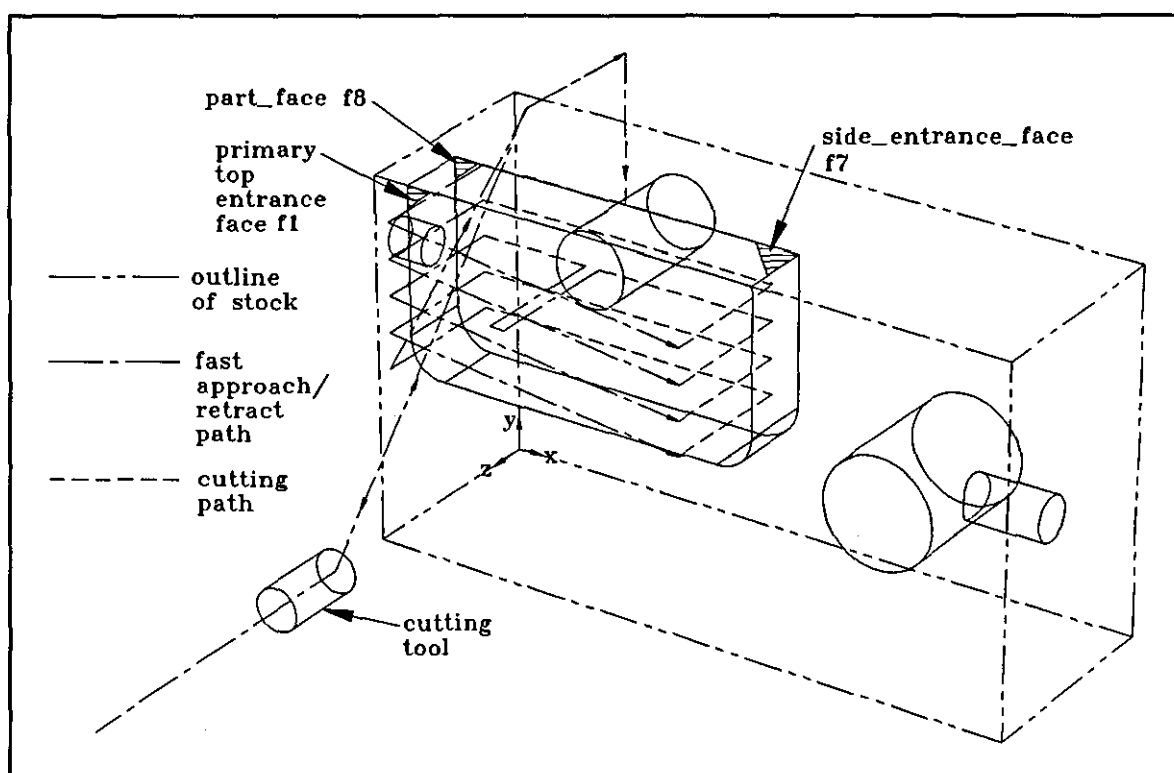


Figure 8.9 : Rough milling for machining record no. 1.

When the `part_face` has convex inner edge loops, it means that the `part_face` contains inner protrusions or islands. To avoid collision, the cutter is raised to a safe height when moving across an inner edge loop. The safe height is determined according

to the height of the `primary_top_entrance_face`. Following the rough milling, a fine milling cutter path is generated by driving the cutter to move around the perimeter of the polygon as shown in Fig. 8.10. This is equivalent to perform a profile milling around the vertical 'walls' of the machining feature.

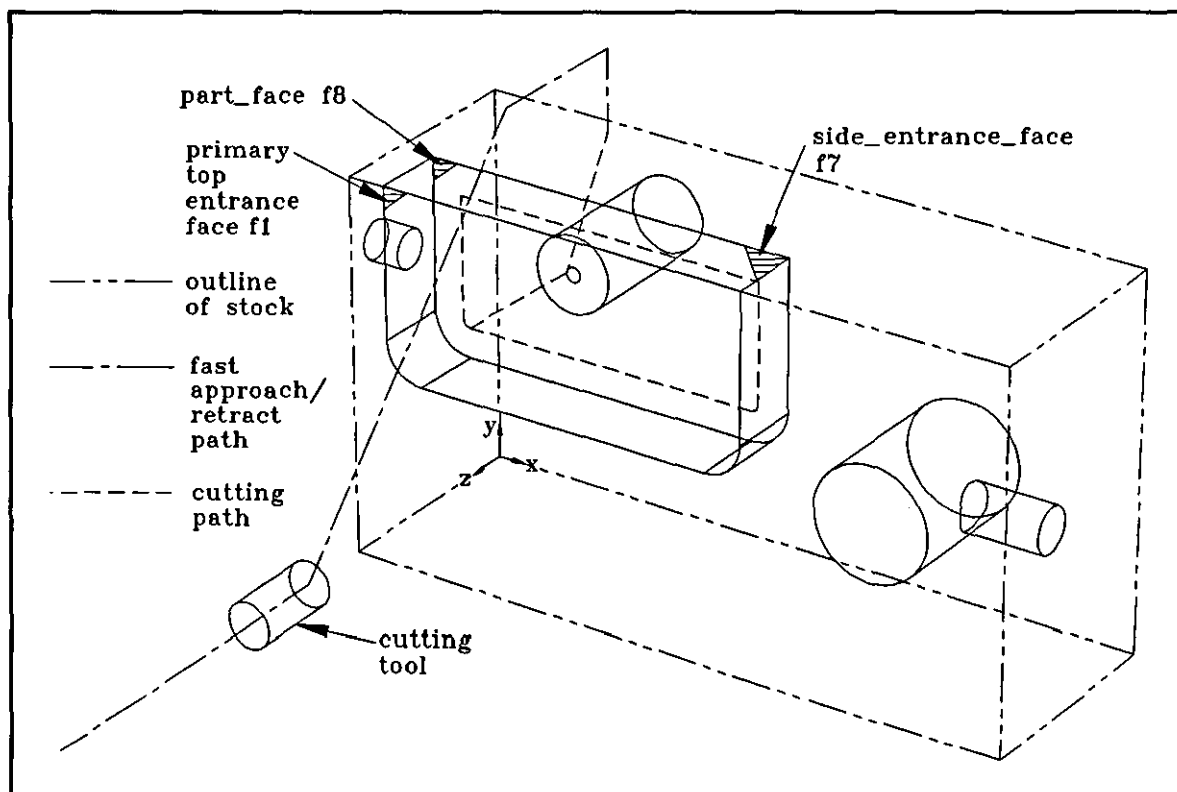


Figure 8.10 : Fine milling for machining record no. 1.

The generated cutter path is maintained internally in a linear list. It can be output to an intermediate cutter location data file [BS5110] which can be post-processed to produce the NC programs.

### 8.3 Examples

Figure 8.11 illustrates a reasonably complicated sample part together with the machining features that can be extracted by the feature recognition algorithm. For convenience of illustration, the extracted machining features are represented by means

of highlighting their corresponding part\_faces. The generated cutter paths for the machining features are displayed in Figs. 8.12a and 8.12b. The image of the original stock is shown in the figure for visual credibility.

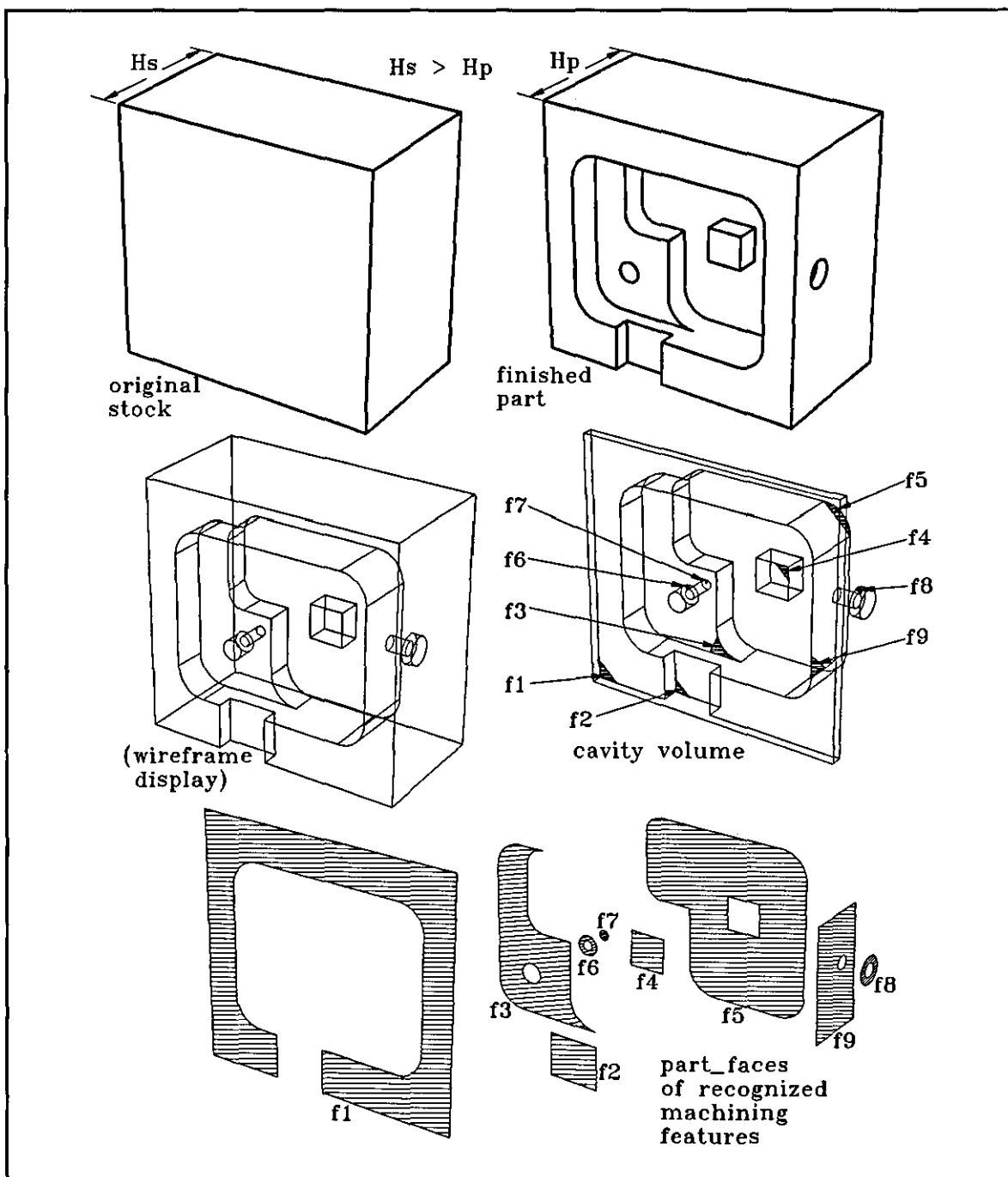


Figure 8.11 : Sample part no. 1.

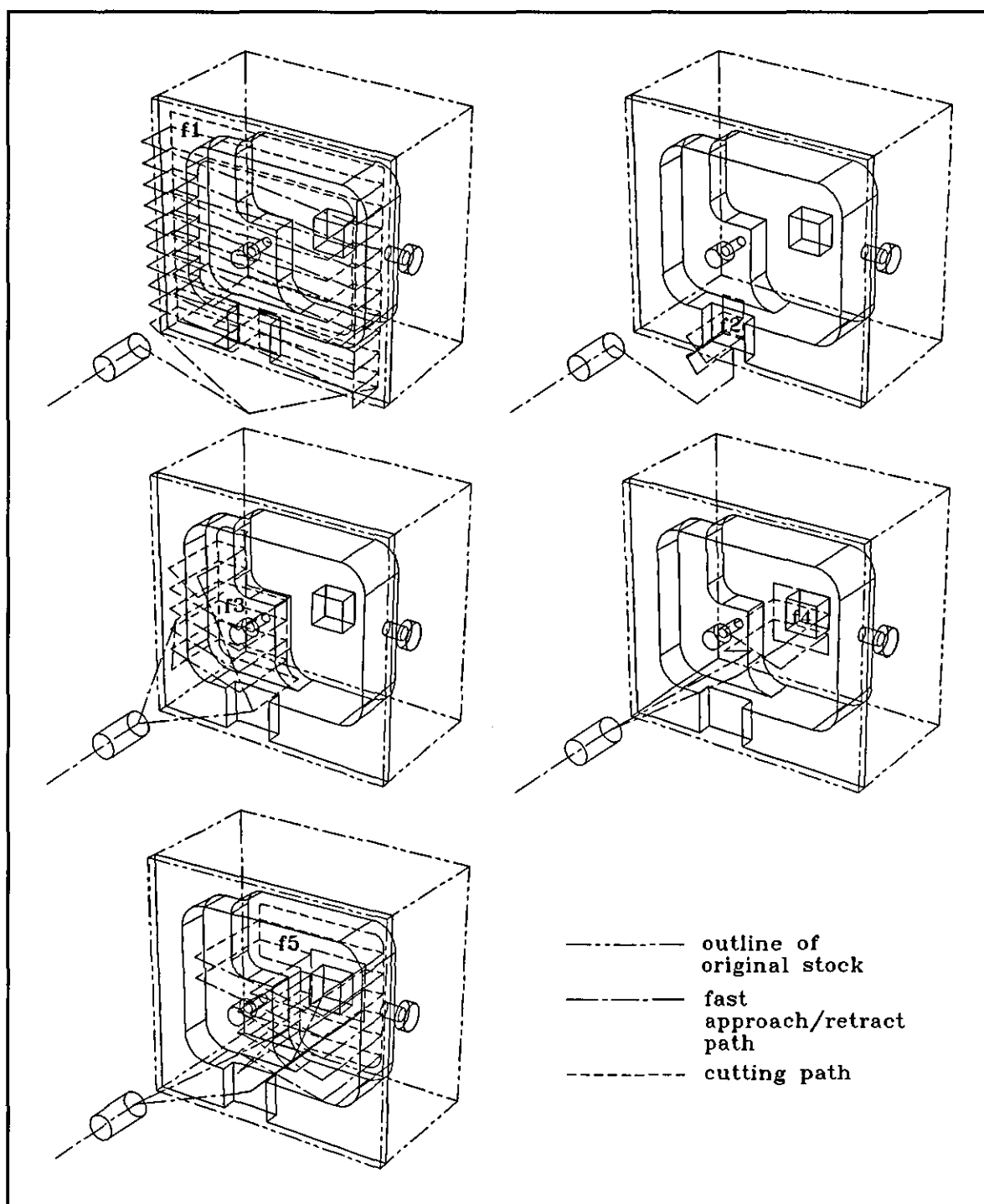


Figure 8.12a : Generated cutter path for sample part no. 1.

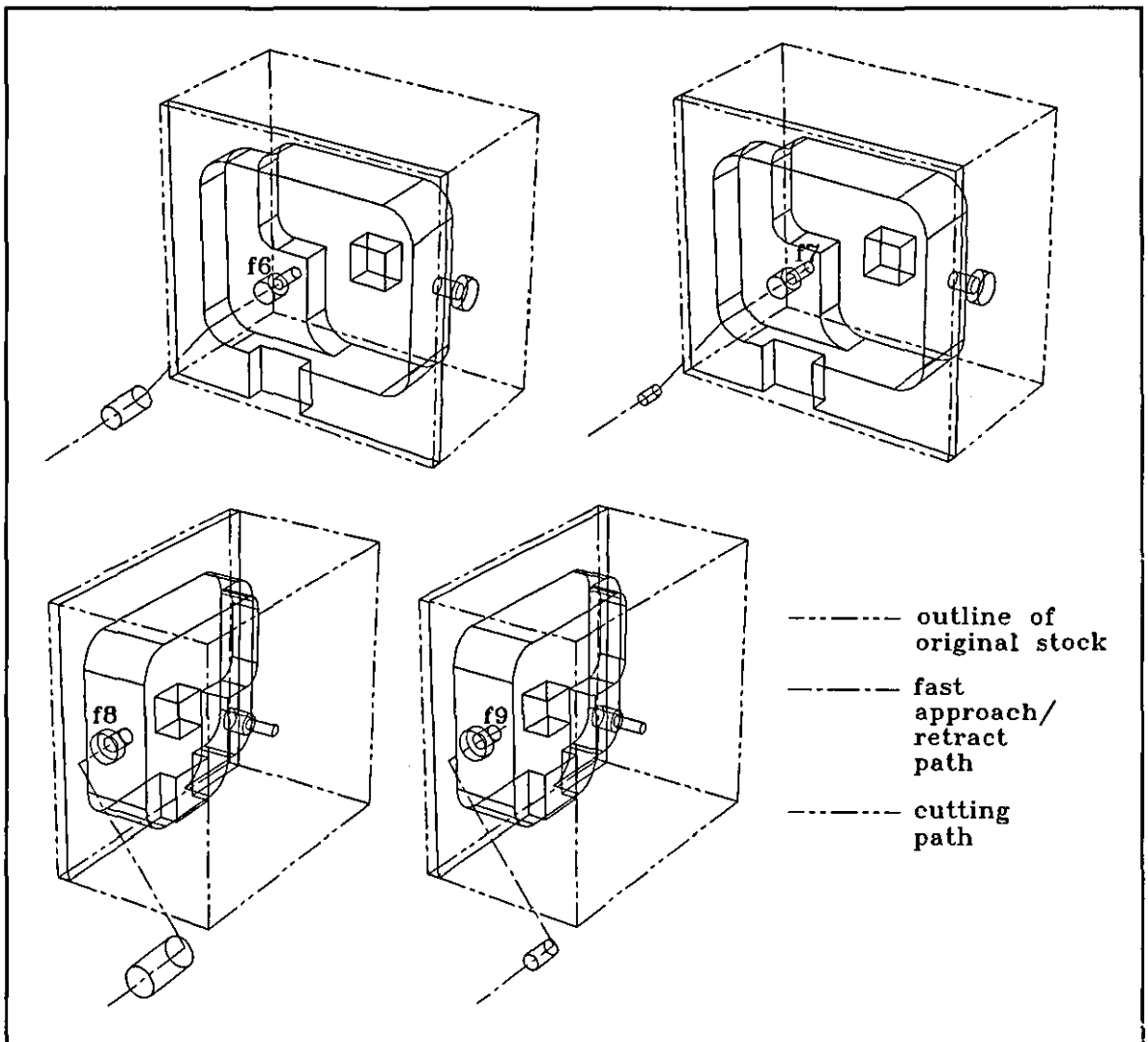


Figure 8.12b : Generated cutter path for sample part no.1.

Figure 8.13 shows a rectangular pocket that has its four corners recessed by means of drilling holes. As a result, the interaction between the holes and the pocket becomes rather complicated for feature recognition. The feature recognition algorithm can recognize the pocket and the four holes in terms of determining their `part_faces` and `primary_top_entrance_faces`. The `part_face` (i.e. bottom face) of the pocket is also recognized as the `secondary_top_entrance_face` of the four holes. The generated cutter path for the part is also illustrated in Figure 8.13.

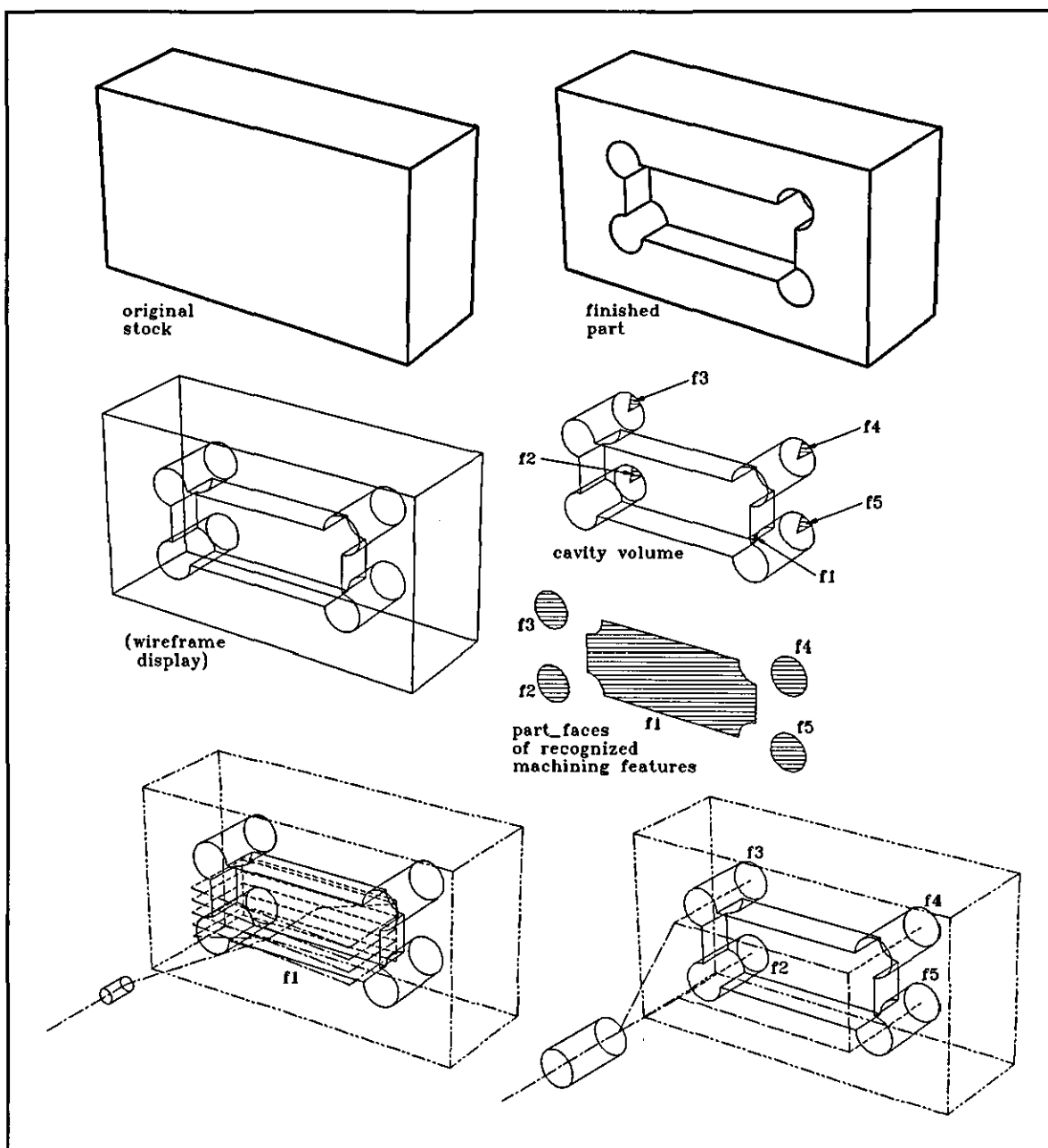


Figure 8.13 : Sample part no. 2.

The part depicted in Figure 8.14 is a mould platen used in a plastic injection moulding machine which is manufactured by a local factory.

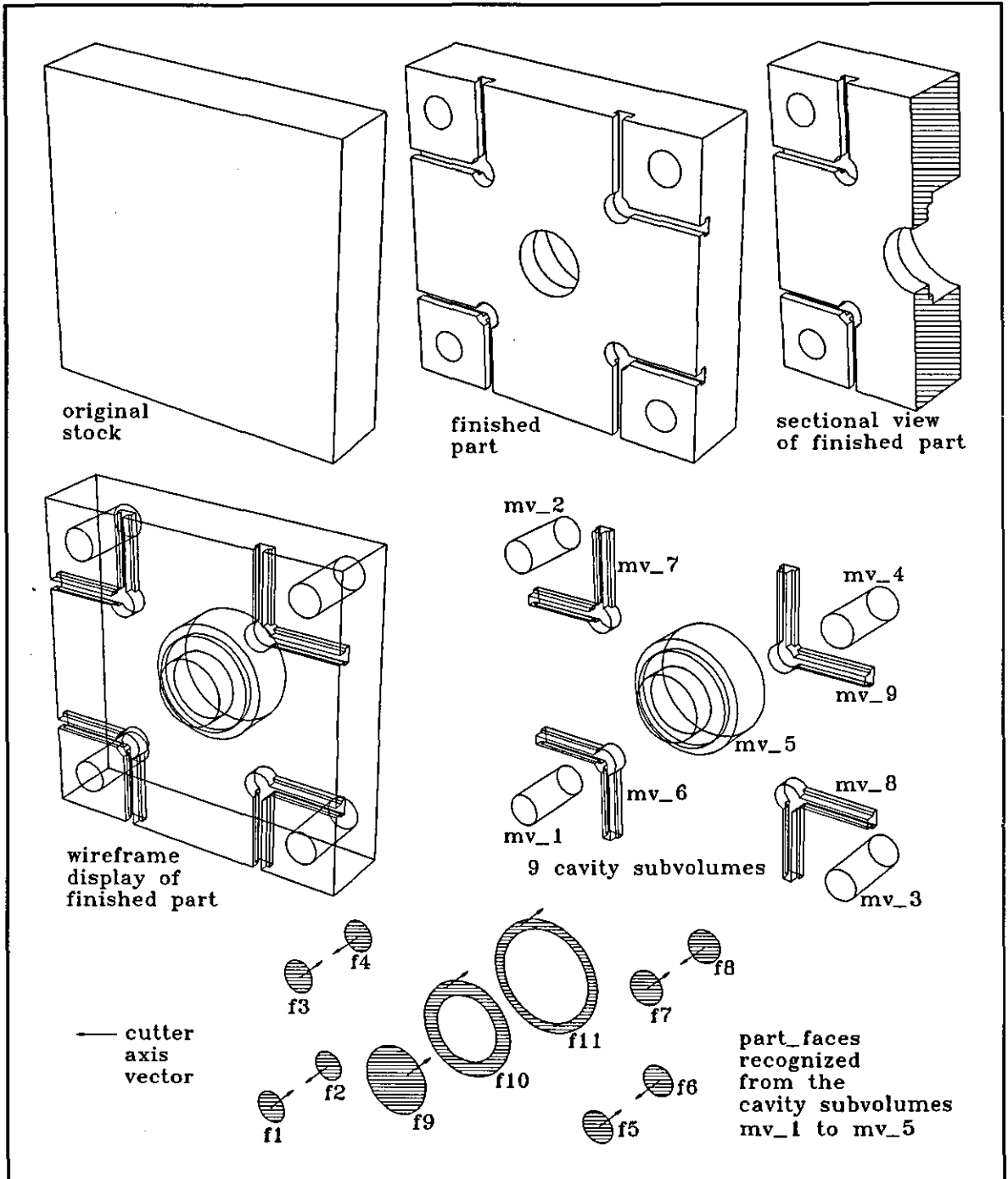


Figure 8.14 : Sample part no. 3.



The T-slots are designed to facilitate the clamping of moulds of variable sizes on the mould platen surface. The design of the actual dimensions of the T-slots depends on the size or capacity of the injection moulding machine, but the T-slot shape remains basically the same irrespective of the moulding machine capacity. The central stepped hole provides a space for adapting the frontal portion (injection nozzle) of the plastics extruder. The sliding movement of the mould platen is guided by cylindrical tie rods which pass through the four holes near the corners of the mould platen.

There are altogether 9 cavity subvolumes as shown in the figure. From cavity subvolumes mv\_1 to mv\_5, the feature recognition algorithm extracts 11 machining features whose corresponding part\_faces are highlighted in the figure. It can be seen that for each of the four corner holes, two part\_faces are extracted as the hole can be machined from two cutter approach directions. For instance, for cavity subvolume mv\_1, part\_faces 'f1' and 'f2' are extracted. The 11 machining features will be classified by the operation sequencing module into two groups based on the similarity of their cutter axis vectors. One group will consist of features represented by part\_faces f1, f3, f5, f7, f9, f10 and f11, while the other group will consist of features represented by part\_faces f2, f4, f6 and f8. As described in section 8.1.1, the operation sequencing module will resolve the situation of dual approach directions by deleting part\_faces f2, f4, f6, and f8 in the latter group because the latter group size is smaller than that of the former group. As a result, only the former group remains and thus the central stepped hole together with the four corner holes will be machined in the same set-up as illustrated in Figure 8.15.

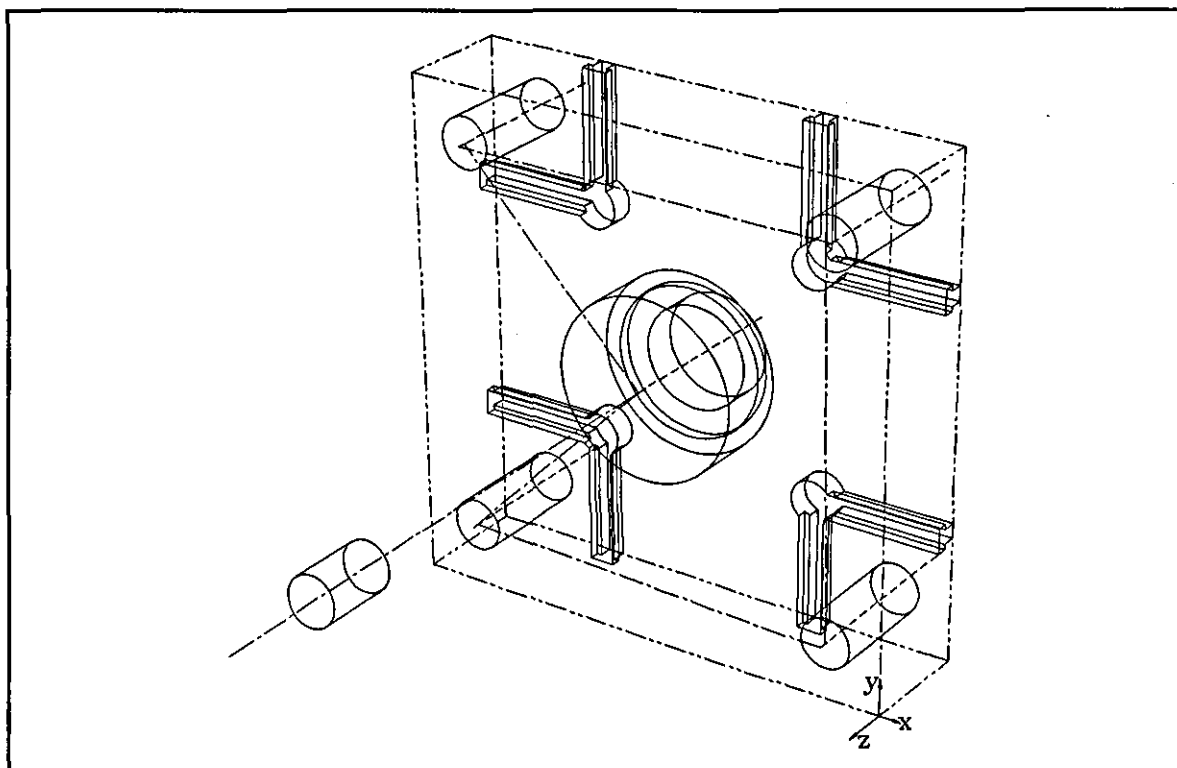


Figure 8.15 : Machining of the central stepped holes and the four side holes.

The feature recognition algorithm cannot recognize any feature from the cavity subvolumes  $mv_6$  to  $mv_9$  because their T-slot-like feature shape violates the ray-casting test of the algorithm. As only the parametric dimensions of the T-slot vary while the shape of the T-slot remains the same for different models of mould platens, it is therefore worthwhile to use the feature learning approach to remember its generic shape and the associated machining method so that after the learning process the system will be able to recognize other T-slots of different mould platen models, and at the same time, determine their corresponding machining faces.

As the shapes of cavity subvolumes  $mv_6$  to  $mv_9$  are identical to each other, any one of them can be used as a teaching example. For instance, the cavity subvolume  $mv_9$  is used for instructing the three machining faces to the system as illustrated in Figure 8.16.

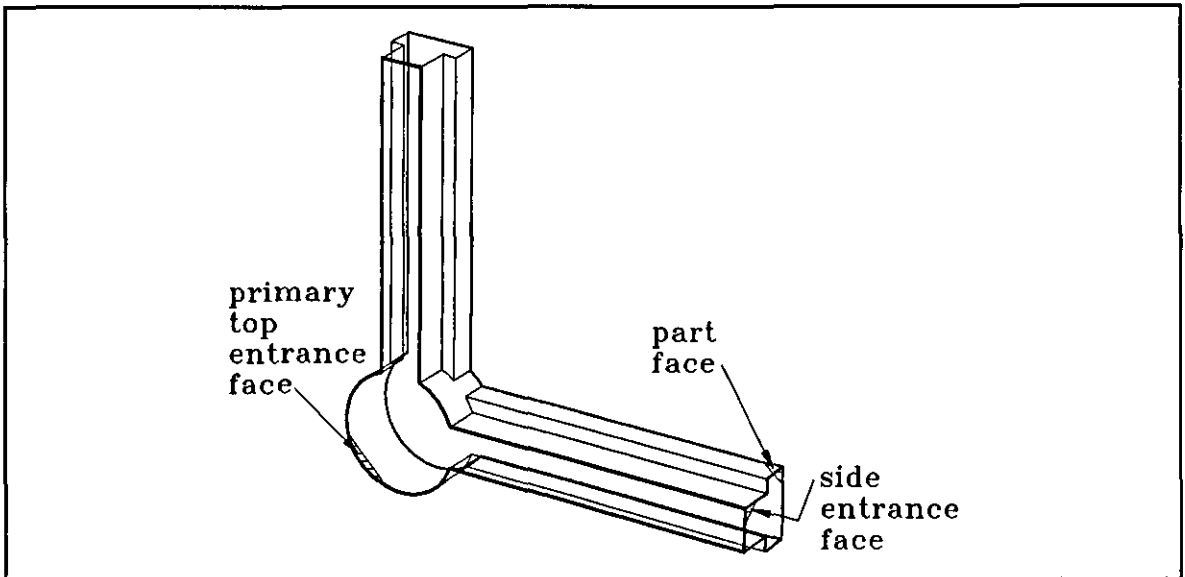


Figure 8.16 : Generated cutter path for milling the T-slots.

As shown in the figure, the T-slot can have two `side_entrance_faces` which are different only in the direction of their surface normals. The surface normal orientation of the machining faces are not included in the new rules created during the learning process, and therefore either one of the two `side_entrance_faces` can be defined as the `side_entrance_face`. It is understood that T-slot machining is a rather special machining process that normally requires several steps of machining operations such as milling a simple rectangular slot first so as to provide a spatial clearance for a T-slot cutter to do the T-slot milling afterwards. These technical details of machining operation are not considered in the machining face instruction process. When instructing the `part_face`, attention is focused mainly on the final T-slot milling operation during which the bottom face of the T-slot will be used as `part_face` as illustrated in Fig. 8.17. The instruction of the `primary_top_entrance_face` is rather obvious as the top face of the T-slot must not cause obstruction to the cutter shank when the cutter is machining the T-slot.

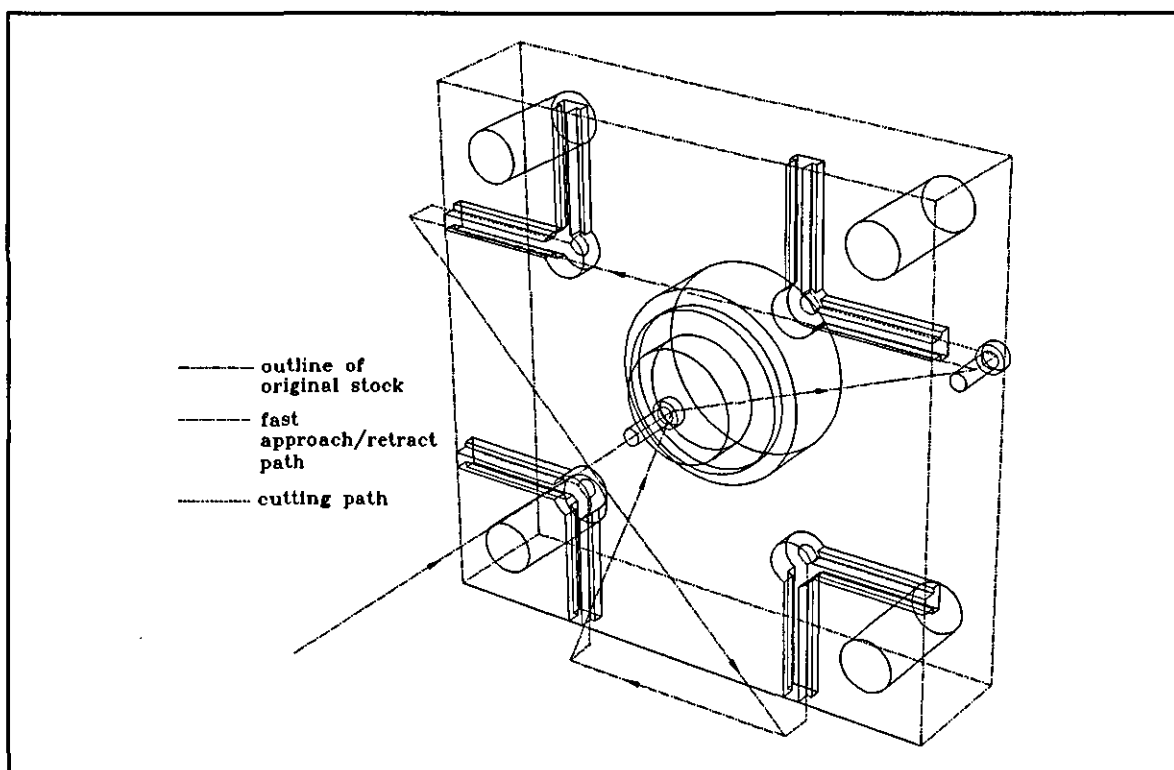


Figure 8.17 : Generated cutter path for milling the T-slots.

After the machining face instruction process, the system remembers the shape of the example T-slot and the instructed machining method in the form of a set of new rules as described in the last chapter. With the incorporation of the new rules, the system is able to recognize a T-slot of shape identical to the example one and, at the same time, augment the B-rep database of the T-slot with the three machining faces information. The operation sequencing module and the cutter path generation module can then process the enhanced B-rep database in the manner as described before. The cutter paths generated for the T-slots are illustrated in Figure 8.17.

## 8.4 Concluding Remarks

From the above extensive description and illustration of the series of steps starting from the extraction of machining features to the final production of NC cutter path, it can be appreciated that the two approaches have basically satisfied the original research objectives of enhancing the communication link between CAD and CAM.

The two approaches do not require a special feature based design environment as the feature modelling strategy employed in both approaches is fundamentally based on post-design boundary data manipulation.

The first approach can recognize rather complicated 2.5D machining features, while the second approach provides a remedial backup to the first approach for handling non-2.5D or custom features. After the application of the two approaches, the boundary data of the design model is enriched with specific and practical machining feature information. In other words, the solid model is virtually transformed into a feature model which contains not only geometric and topological information, but also important manufacturing oriented information such as machining region, tool\_entrance\_face and cutter approach direction. Despite the rather simplistic implementation of the set up determination and cutter path software, they still serve the purpose of demonstrating the practical usefulness of the two approaches which are capable of extracting valid manufacturing information directly from the design database.

## CHAPTER 9

# CONCLUSIONS AND FUTURE WORK

### 9.1 Conclusions

Features have been widely accepted as a basis for integrating CAD and CAM because they can embody design intent along with part geometry, and thus provide the necessary information for various manufacturing applications. Two distinct approaches have been used by researchers for modelling features : design by features and automatic feature recognition. The research work presented in this thesis has focused mainly on using the latter approach for recognizing machining features.

Machining features in other research have been represented either as faces or as volumes, and each representation has its own advantages and disadvantages. This thesis has taken the advantages of both representations by recognizing machining features as groups of machining faces from a cavity volume model that is obtained by Boolean subtraction between the starting stock and finished part models. The merit of using the cavity volume is that it not only reveals the volumes of material actually required to be removed but also provides a comprehensive boundary description of the machining volume which is useful in an automatic process planning context.

The core of this thesis discusses two methods of recognizing machining features directly from a CAD database, and their implementation in a prototype software system. The first method is designed to recognize 2.5D machining features in the corresponding cavity volume model of a finished part. The cavity volume is primarily represented in a winged-edge based B-rep data structure and secondarily represented as frames in the working memory of a knowledge-based system. Recognition is achieved by analyzing the cutter accessibility of cavity volumes whose boundary faces are classified either as `machined_faces` or as `tool_entrance_faces`. The accessibility analysis is essentially a simulation of the cutting action when a cylindrical cutter is used to machine a selected

face of the cavity volume, and is implemented by projecting semi-infinite, imaginary lines from the surface of the selected face. If these cast lines are free of obstruction, a machining feature is considered to have been recognized. This ray-casting feature recognition algorithm has several advantages : it avoids complicated searches for boundary shape elements during recognition, allows the identification of reasonably complex machining features that interact with each other, and ensures cutter accessibility of the recognized features. Representation of a recognized machining feature is in terms of the identified machining faces and the projection direction of the cast lines. The extracted feature information is stored in appropriately linked data records in a boundary representation database.

This feature recognition method has the drawback that it can only recognize 2.5D machining features that satisfy the cutter accessibility analysis. Consequently, a second method of machine learning has been implemented and this allows the user to use a cavity volume that is not recognizable by the first method to be used as a positive teaching example to interactively instruct the corresponding machining faces to the system. These instructed faces together with the boundary description of the cavity volume are then compiled into a group of production rules which are then added into the rule base of the system. When a similarly shaped cavity volume is subsequently encountered the system will be able to make use of the new rules to recognize and generate appropriate machining faces as machining feature information in the B-rep database.

## 9.2 Contributions to Knowledge

The major contributions of this thesis are summarised below :

- (1) A scheme has been designed and implemented for representing machining features. Based on this scheme, a ray-casting based algorithm has been devised for recognizing 2.5D machining features. Besides being a useful tool for exposing potential 2.5D machining features, the algorithm can also be used as a general method for dealing with machining feature interactions in a design by features system. For instance, with slight modifications, the algorithm could be embedded in a feature based design system as a procedure for validating the machinability of a particular portion of a part after each step of design construction. Although the implementation described in this thesis uses a CSG-based system, the method actually works with the B-rep database, and hence the algorithm can also be adopted in B-rep modellers.
- (2) A machine learning based procedure has been designed and implemented for adding custom machining features to the system for subsequent recognition. The representation of the custom features is compatible with the feature representation scheme adopted in the first method. The process of instructing new custom features is interactive and does not require the knowledge of a programming language. This provides a novel means of extending the recognition ability of the system to adapt to diverse manufacturing conditions.
- (3) As a proof-of-concept implementation, a simple machining operation sequencing program and a NC cutter path generation program have been developed to study and elucidate the chain of steps leading from the extracted feature information to the ultimate production of NC part programs.
- (4) The practical demonstration of how AI and solid modelling techniques can be combined together to accomplish the automatic extraction and organization of CAD



information for manufacturing has been produced. In particular, a suite of programs have been implemented for constructing/managing a comprehensive B-rep database which is an improved version of the original PADL-2's boundary file, and for transforming the B-rep data into a frame representation for feature recognition. In addition, a prototype software system using a combination of knowledge-based and solid modeller architectures has been developed. The testbed system is a flexible research platform for pursuing future explorations on the automation and integration of design and manufacturing.

### **9.3 Future Work**

Major recommendations for future work are discussed based on refinement of some of the ideas and their implementations in this research. Some general research avenues opened up by this thesis are also briefly outlined.

#### **9.3.1 Feature Classification**

The extracted machining features have not been differentiated into specific feature classes or types, and thus the cutter path generation module can only treat all extracted machining features as a general pocket. This results in the use of a rather general cutter path generation strategy. If the extracted features could be classified into more meaningful types such as slots and steps, then many of the process planning decisions, such as operation selection and cutter path planning, can be made more intelligently.

This shortcoming can be ameliorated by adding a feature classification module to enhance the semantic content of the extracted machining features with feature type information. In this connection, a feature classification scheme would need to be

established for defining the feature type domain. The form feature taxonomy proposed by Gindy [Gindy89] is recommended for this purpose, not only because the taxonomy is designed for generic features, but also because the classification attributes used in the structure are very similar to those used in the feature representation scheme of this work. For instance, Gindy's entry/exit boundaries can be related to the `primary_top_entrance_face/part_face`, while the external access directions can be seen as equivalent to the cutter axis vector described in this thesis.

The feature type classification module could be implemented by specifying the classification attribute conditions of each defined feature type in production rules so that successful firing of a set of rules would lead to confirmation of a particular feature type. The comprehensive B-rep database established in this research work would still be a primary source for providing necessary information to the classification process. However, the major information input to the feature type classification process would not be raw CAD data but extracted machining features that have practical manufacturing meanings already attached to their bounding faces.

### 9.3.2 Alternative Cutter Axis Vectors

The cutter axis vector, which is determined directly from the projection direction of the cast rays, basically represents the cutter axial approach direction. In effect, the cutter axis vector defines how a machining feature is orientated for machining, and hence it is a very useful piece of information for setup planning, machining path planning, etc.. Currently, only one cutter axis vector is represented in an extracted machining feature due to the fact that the ray-casting accessibility analysis is not repeated on the other faces of the machining feature once a valid `part_face` is found. This limitation could be removed if the accessibility test were also performed on the other faces of the recognized machining feature so that the total number of alternative cutter axis vectors can be explored and ascertained. This would also facilitate the feature type classification process as recommended above since the number of

alternative cutter axis vectors could be used as a clue for discriminating between different classes of features.

### **9.3.3 Instructing Multiple Features**

The machine learning method only allows the user to teach one machining feature in a cavity volume. As a cavity volume may contain several machining features, it is desirable to improve the method so that multiple features could be learnt by the system.

### **9.3.4 Learning Technique Enhancement**

The learning method adopted in this thesis is essentially based on the learning by rote strategy which involves the conversion of the boundary characteristics of a feature example into an independent set of rigidly linked production rules. Each set of production rules is used as a unique feature template for subsequent feature recognition. An undesirable result is that the system considers every feature example as a completely new and unique case that bears no relationship or similarity with previously learnt features. This has an adverse effect on the system performance since the number of new rules added to the system can easily increase to an unmanageable size.

Perhaps a more positive approach for enhancing the learning process would be to use the learning by example strategy described in chapter 3. With such an approach, the system would regard the features to be learnt as a continuous supply of training examples. With the feature classification structure mentioned above, the system would be able to discern structural shape similarities or differences between a given feature example and the previously learnt features. The user would need to teach the system by providing more specific information or instructions so that the system could enhance its knowledge about feature shapes through one or a combination of the following three

major actions :

- (1) Generalize a previously learnt feature shape description into a more generic description that could be used as a basis for recognizing the feature example. For example, generalization could be done by relaxing shape classification constraints, such as using convexity classification for representing edge angle rather than specifying an exact angular measure in degrees;
- (2) Specialize a previously learnt feature shape description into a more stringent or discriminative description that has a high discerning power for recognizing the feature example. For instance, specialization could be effected by defining precisely a classification attribute value, such as stating the exact number of cutter axis vectors of a feature;
- (3) Create a new feature shape description for the feature example when the feature example is found to be a genuinely new feature instance. This action is essentially the approach currently used in this thesis.

With these methods, the problem of managing a large rule base would be alleviated since the acquisition of new feature shape knowledge would not always be by creating new feature shape description rules but also by modifying existing rules. However, the alternate generalization and specialization of rules may make the system's performance in recognizing features become unstable or inconsistent. For example, the system may 'forget' some previously learnt features after an improper specialization process. Exploration in using such a feature learning approach is a challenging research task.

### 9.3.5 General Research Directions

Further investigations can be pursued along two major directions :

- (1) Extension of the part domain in terms of complexity and variety;
- (2) Comprehensive exploitation of feature technology for integrating various automated activities in the product life cycle.

The parts considered in this research are machined components that contain only planar and cylindrical faces. Extension of the feature finding methods to machined parts that are constructed by using additional surface types is essential. Methods for representing variational geometric information in CAD models are lacking as they embody much design and manufacturing meanings that are very beneficial to feature reasoning. While PADL-2 can still be used to fulfil this extension, the use of an advanced B-rep modeller that offers high extensibility and manipulation flexibility of part geometry would be a more desirable and long-term choice. The knowledge and experience gained from this research could also be applied to study features associated with other product types such as sheet-metal parts and moulded parts.

The use of the extracted feature information can be extended for diverse design and manufacturing activities such as finite-element analysis, process capability modelling, setup planning, assembly planning and inspection planning. Investigations using a feature model as a central database for supporting manufacturing logistics oriented activities such as material requirements planning and product costing are of crucial importance in a computer integrated manufacturing environment. However, so far these kinds of research activities on feature applications seem to be severely lacking.

## REFERENCES

- [Akagi91] Akagi, S., "Expert system for engineering design based on object-oriented knowledge representation concept", *Artificial Intelligence in Design*, Pham, D.T., (ed.), Springer-Verlag, pp. 61-95, 1991.
- [Alting89] Alting, L. and Zhang, H.C., "Computer aided process planning : the state-of-the-art survey", *International Journal of Production Research*, Vol. 27, No. 4, pp. 553-586, April 1989.
- [Anderson79] Anderson, J.R., Kline, P. and Beasley, C., "A general learning theory and its implications to schema abstraction", *The Psychology of Learning and Motivation*, Bower, G.H., (ed.), Academic Press, New York, Vol. 13, pp. 277-318, 1979.
- [Arbab82] Arbab, F., "Requirements and architecture of a CAM-Oriented CAD system for design and manufacture of mechanical parts," Ph.D. Thesis, Computer Science Dept., University of California, Los Angeles, 1982.
- [Baer79] Baer, A., Eastman, C. and Henrion, M., "Geometric modelling : a survey", *Computer-Aided Design*, Vol. 11, No. 5, pp. 255-272, September 1979.
- [Baumgart74] Baumgart, B.G., "Geometric modeling for computer vision", Ph.D. thesis, Department of Computer Science, Stanford University, August 1974.
- [Berenji86] Berenji, H.R. and Khoshnevis, B., "Use of artificial intelligence in automated process planning", *Computers in Mechanical Engineering*, pp. 47-55, September 1986.
- [Bono87] Bono, P.R. and Herman I., (eds.), "GKS theory and practice", Springer-Verlag, Berlin, 1987.
- [Brown82] Brown, C.M., "PADL-2: A technical summary", *IEEE Computer Graphics and Applications*, Vol. 2, No. 2, pp. 68-84, March 1982.
- [Brownston85] Brownston, L., Farrell, R., Kant, E. and Martin, N., "Programming Expert Systems in OPS5", Addison-Wesley Publishing Company Inc., 1985.

- [Bruzzzone91] Bruzzzone, E. and Floriani, L. De., "Extracting adjacency relationships from a modular boundary model", *Computer-aided Design*, Vol. 23, No. 5, pp. 344-356, June 1991.
- [BS3635] BS3635 : Part 1 : control input data, "Specification for the numerical control of machines", November 1972.
- [BS5110] BS5110 : Part 1 : 1979, "Logical structure and major words of processor output (CLDATA), Specification for programming languages for the numerical control of machines", British Standards Institution.
- [Buller80] Buller, W.I., Nof, S.Y. and Whinston, A.B., "Artificial Intelligence in Manufacturing Planning and Control", *AIIE Transactions*, pp. 351-363, December 1980.
- [Butterfield87] Butterfield, W.R., Green, M.K., Scott, D.C. and Stoker, W.J., "Part features for process planning", CAM-I Report C-85-PPP-03, CAM-I Inc., Arlington, Texas, 1987.
- [CAMI-ANC85] General Dynamics Corporation, "Volume decomposition algorithm final report", CAM-I Report R-85-ANC-01, 1985.
- [Case92] Case, K., "Feature technology : an integration methodology for CAD and CAM", Keynote paper to the International Conference on Manufacturing Automation, Ko, N.W.M. and Tan, S.T., (eds.), University of Hong Kong, Hong Kong, pp. 613-624, 10-12 August 1992.
- [Chan82] Chan, B.T.F., "ROMAPT : a new link between CAD and CAM", *Computer-aided Design*, Vol. 14, No. 5, pp. 261-266, September 1982.
- [Chan86] Chan, S. and Voelcker, H.B., "An introduction to MPL - a new machining process programming language", *IEEE Int. Conf. on Robotics and Automation*, San Francisco, April 1986.
- [Chan88] Chan, K.C. and Tan, S.T., "Hierarchical structure to winged-edge structure : a conversion algorithm", *The Visual Computer*, 4:133-141, Springer-Verlag, 1988.
- [Chang81] Chang, T.C. and Wysk, R.A., "An integrated CAD/Automated process planning system", *AIIE Transactions*, Vol. 13, No.3, pp. 223-233, September 1981.

- [Chang88] Chang, T.C., Anderson, D.C. and Mitchell, O.R., "QTC - an integrated design/manufacturing inspection system for prismatic parts", Proceedings of the 1988 ASME International Computers in Engineering Conference and Expositions, Vol. 1, pp. 417-426, 1988.
- [Choi84] Choi, B.K., Barash, M.M. and Anderson, D.C., "Automatic recognition of machined surfaces from a 3D solid model", Computer-aided Design, Vol. 16, No. 2, pp. 81-86, March 1984.
- [Choi88] Choi, B.K., Lee, C.S., Hwang, J.S. and Jun, C.S., "Compound surface modelling and machining", Computer-aided Design, Vol. 20, No. 3, pp. 127-136, April 1988.
- [Chung88] Chung, J.C.H., Cook, R.L., Patel, D. and Simmons, M.K., "Feature-based geometry construction for geometric reasoning", Proceedings of the 1988 ASME International Computers in Engineering Conference and Expositions, Vol. 1, pp. 497-504, 1988.
- [Cohen83] Cohen, P.R. and Feigenbaum, E.A., (eds.), "Learning and inductive inference", The Handbook of Artificial Intelligence, Vol. 3, Chapter XIV, pp. 324-511, Pitman Books Limited, 1983.
- [Corney91a] Corney, J. and Clark, D.E.R., "A feature recognition algorithm for multiply connected depressions and protrusions in 2.5D objects", Proceedings of Symposium in Solid Modelling Foundations and CAD/CAM Applications, ACM, USA, 1991.
- [Corney91b] Corney, J. and Clark, D.E.R., "Method for finding holes and pockets that connect multiple faces in 2.5D objects", Computer-aided Design, Vol. 23, No. 10, pp. 658-668, December 1991.
- [Crawford87] Crawford, T.M. and Marton, V., "A machine learning approach to expert systems for fault diagnosis in communications equipment", Computer-aided Engineering Journal, pp. 31-38, 1987.
- [Cunningham88] Cunningham, J.J. and Dixon, J.R., "Designing with features : the origin of features", Proceedings of the 1988 ASME International Computers in Engineering Conference and Expositions, Vol. 1, pp. 237-243, August 1988.
- [Cutkosky88] Cutkosky, M., Tenenbaum, J.M. and Muller, D., "Features in process based design", Proceedings of the 1988 ASME International Computers in Engineering Conference and Expositions, Vol. 1, pp. 557-562, 1988.



- [Dahl73] Dahl, O.J. and Hoare, C.R., "Hierarchical program structures", Structured Programming, Academic Press, London, 1973.
- [Davies84] Davies, B.J. and Darbyshire, I.L., "The use of expert systems in process planning", Annals of the CIRP, Vol. 33/1, pp. 303-306, 1984.
- [Descotte84] Descotte, Y. and Latombe, J., "GARI : an expert system for process planning", Solid Modelling by Computers : from theory to applications, Pickett, M.S. and Boyse, J.W., (ed.), Plenum Press, pp. 329-346, 1984.
- [Digital85] Digital Equipment Corporation, "VAX OPS5 reference manual (AA-EZ19A-TE) and VAX OPS5 user's guide (AA-EZ18A-TE)", 1985.
- [Dong88a] Dong, X. and Wozny, M.J., "FRAFES : a frame-based feature extraction system", Proceedings of the International Conference on Computer Integrated Manufacturing, Troy, New York, pp. 296-305, May 1988.
- [Dong88b] Dong, X., "Geometric feature extraction for computer-aided process planning", Ph.D. Thesis, Center for Interactive Computer Graphics, Rensselaer Polytechnic Institute, Troy, New York, May 1988.
- [Eastman79] Eastman, C. and Weiler, K., "Geometric modelling using the Euler operators", Proceedings of the First Annual Conference on Computer Graphics in CAD/CAM Systems, MIT, Cambridge, Mass., pp. 248-259, April 1979.
- [Falcidieno87] Falcidieno, B. and Giannini, F., "Extraction and organization of form features into a structures boundary model", EUROGRAPHICS '87, Marechal, G., (ed.), Elsevier Science Publishers B.V. (North-Holland), pp. 249-259, 1987.
- [Floriani88] Floriani, L.De and Falcidieno, B., "A hierarchical boundary model for solid object representation", ACM Transactions on Graphics, Vol. 7, No. 1, pp. 42-60, January 1988.
- [Floriani89] Floriani, L.De and Bruzzone, E., "Building a feature-based object description from a boundary model", Computer-aided Design, Vol. 21, No. 10, pp. 602-610, December 1989.

- [Forgy77] Forgy, C. and McDermott, J., "OPS, a domain-independent production system language", Proceedings of the Fifth International Joint Conference on Artificial Intelligence, pp. 933-939, 1977.
- [Gallagher73] Gallagher, C.C. and Knight, W.A., "Group technology", Butterworths, London, 1973.
- [Gandhi89] Gandhi, A. and A. Myklebust, "A natural language approach to feature based modeling", Advances in Design Automation-1989, Vol. 1, Computer-Aided and Computational Design, Ravani, B., (ed.), pp. 69-77, 1989.
- [Gindy89] Gindy, N.N.Z., "A hierarchical structure for form features", International Journal of Production Research, Vol. 27, No. 12, pp. 2089-2103, 1989.
- [Gindy90] Gindy, N.N.Z. and Case, K., "Process capability models for design and selection of processing equipment", Research Proposal, Department of Manufacturing, Loughborough University of Technology, Loughborough, Leicestershire, UK, July 1990.
- [Gindy91] Gindy, N.N.Z. and Ratchev, T.M., "Product and machine tools data models for CAPP systems", 4th IFIP Conference, CAPE91, Bordeaux, France, September 1991.
- [Goldberg83] Goldberg, A. and Robson, P., "Smalltalk80 : the language and its implementation", Addison-Wesley, Reading, 1983.
- [Grayer77] Grayer, A.R., "The automatic production of machined components starting from a stored geometric description", Advances in Computer-Aided Manufacture, McPherson, D., (ed.), North-Holland Publishing Company, pp. 137-151, 1977.
- [Harmon88] Harmon, P., Maus, R. and Morrissey, W., "Expert systems tools and applications", John Wiley & Sons, Inc., 1988.
- [Hartquist81] Hartquist, E.E. Peterson, D.P., and Voelcker, H.B., "BFILE/2 : a boundary file for PADL-2", Computational Geometry Group Memo. No. 20, Production Automation Project, University of Rochester, 1982.
- [Hart86] Hart, N. and Bennaton, J., "A CAD engineering language to aid manufacture", Proceedings of International Conference on Computer Aided Production in Engineering, Edinburgh, April 1986.

- [Held91] Held, H., "On the computational geometry of pocket machining", Springer-Verlag, Berlin Heidelberg, 1991.
- [Henderson84] Henderson, M.R., "Extraction of feature information from three dimensional CAD data", Ph.D. thesis, Purdue University, May 1984.
- [Hillyard82] Hillyard, R., "The BUILD group of solid modelers", IEEE Computer Graphics & Applications, pp. 43-52, March 1982.
- [Houten90] van Houten, F.J.A.M., van 't Erve, A.H., Boogert, R.M., Nauta, J., and Kals H.J.J., "Part, selection of methods and tools", 22nd CIRP International Seminar on Manufacturing Systems, Enschede, 1990.
- [Hummel89] Hummel, K.E., "Coupling rule-based and object-oriented programming for the classification of machined features", Proceedings of the 1989 ASME International Computers in Engineering Conference and Expositions, Vol. 1, pp. 409-418, 1989.
- [Hummel90] Hummel, K.E. and Wolf, M.L., "Integrating expert systems with solid modeling through interprocess communications and the applications interface specifications", Proceedings of the 1990 ASME International Computers in Engineering Conference and Expositions, Vol. 1, pp. 355-360, 1990.
- [IIT67] IIT Research Institute, "APT Part Programming", McGraw-Hill, N.Y., 1967.
- [Ishii89] Ishii, K., Goel, A. and Adler, R.E., "A model of simultaneous engineering design", Artificial Intelligence in Design, Gero, J.S.,(ed.), Proceedings of the 4th International Conference on the Applications of Artificial Intelligence in Engineering, Cambridge, UK, July, 1989, Computational Mechanics Publications, Springer-Verlag, pp. 483-501, 1989.
- [Iwata90] Iwata, K., Fukuda, Y. and Sugimura, N., "A proposal of knowledge base structure for integrated process planning system", Proceedings of the 22th CIRP international seminar on manufacturing systems, Enschede, 1990.
- [Joseph90] Joseph, A.T., Kalta, M. and Davies, B.J., "Automatic generation of NC programs for turned components from CAD product models", Proceedings of the 28th MATADOR, UMIST, 18-19 April 1990.

- [Joshi88] Joshi, S. and Chang, T.C., "Graph-based heuristics for recognition of machined features from a 3D solid model", *Computer-aided Design*, Vol. 20, No. 2, pp. 58-66, March 1988.
- [Juster92] Juster, N.P., "Modelling and representation of dimensions and tolerances : a survey", *Computer-aided Design*, Vol. 24, No. 1, pp. 3-17, January 1992.
- [Kochan86] Kochan, D., (ed.), "Developments in computer-integrated manufacturing", Springer-Verlag, Berlin, Heidelberg, 1986.
- [Kroll91] Kroll, E., Lenz, E. and Wolberg, J.R., "Intelligent analysis and synthesis tools for assembly-oriented design", *Artificial Intelligence in Design*, Pham, D.T., (ed.), Springer-Verlag, pp.125-145, 1991.
- [Kuratowski76] Kuratowski, K. and Mostowski, A., "Set theory", Amsterdam : North-Holland Publishing Co., 1976.
- [Kusiak91] Kusiak, A. and Heragu, S.S., "Knowledge-based programs for manufacturing system design" *Artificial Intelligence in Design*, Pham, D.T., (ed.), Springer-Verlag, pp. 437-492, 1991.
- [Kyprianou80] Kyprianou, L., "Shape classification in computer aided design", Ph.D. Thesis, University of Cambridge, July 1980.
- [Lee82a] Lee, Y.T. and Requicha, A.A.G., "Algorithms for computing the volume and other integral properties of solids : I. known methods and open issues", *Comm. of the ACM*, Vol. 25, No. 9, pp. 635-641, September 1982.
- [Lee82b] Lee, Y.T. and Requicha, A.A.G., "Algorithms for computing the volume and other integral properties of solids : II. a family of algorithms based on representation conversion and cellular approximation", *Comm. of the ACM*, Vol. 25, No. 9, pp. 642-650, September 1982.
- [Lee87] Lee, Y.C. and Fu, K.S., "Machine understanding of CSG : extraction and unification of manufacturing features", *IEEE Computer Graphics & Applications*, pp. 20-32, January 1987.
- [Lee88] Lee, Y.C. and Jea, K.F.J., "A new CSG tree reconstruction algorithm for feature representation", *Proceedings of the 1988 ASME International Computers in Engineering Conference and Expositions*, Vol. 1, pp. 521-528, 1988.

- [Li91] Li, R.K. and Tzieng, S.D., "Machinable volumes extraction-non-block type raw material input", *International Journal of Computer Integrated Manufacturing*, Vol. 4, No. 4, pp. 241-252, 1991.
- [Luby86] Luby, S.D., Dixon, J.R. and Simmons, M.K., "Creating and using a features data base", *Computers in Mechanical Engineering*, Vol. 5, No. 3., pp. 25-33, November 1986.
- [Mendelson75] Mendelson, B., "Introduction to topology", Boston : Allyn and Bacon, Inc., 3rd edition, 1975.
- [McDermott78] McDermott, J. and Forgy, C., "Production system conflict resolution strategies", in *Pattern-directed inference systems*, Waterman, D.A. and Hayes-Roth, F., (eds.), Academic Press, Inc., pp. 177-199, 1978.
- [Minsky75] Minsky, M., "A framework for representing knowledge", *The Psychology of Computer Vision*, Winston, P., (ed.), McGraw-Hill, N.Y., 1975.
- [Mostow83] Mostow, D.J., "Machine transformation of advice into a heuristic search procedure", *Machine Learning, An Artificial Intelligence Approach*, Michalski, R.S., Carbonell, J.G. and Mitchell, T.M., (eds.), Tioga Publishing Co., Palo Alto, CA, pp. 368-403, 1983.
- [Murray86] Murray, J.L. and Linton, H., "The development of an automatic system for the generation of planning and control data for milled components", *Proc. Inter. Workshop on Expert Systems in Production Engineering*, Spa, Belgium, Springer Verlag, Heidelberg, pp. 231-245, 1986.
- [Murray89] Murray, J.L. and Williams, M.H., "Knowledge-based systems in process planning and assembly design", *Geometric Reasoning*, Woodwark, J. (ed.), Clarendon Press, Oxford, pp. 217-236, 1989.
- [Newell72] Newell, A. and Simon, H.A., "Human problem solving", Prentice-Hall, Cliffs, N.J., 1972.
- [Nilsson71] Nilsson, N.J., "Problem-solving methods in artificial intelligence", McGraw-Hill, New York, 1971.
- [Okino73] Okino, N., Kakazu, Y. and Kubo, H., "TIPS-1 : technical information processing system for computer-aided design, drawing and manufacturing", *Computer Languages for Numerical Control*, Hatvany, J. , (ed.), North-Holland Publishing Co., Amsterdam, pp. 141-150, 1973.

- [Perng90] Perng, D.B., Chen, Z. and Li, R.K., "Automatic 3D machining feature extraction from 3D CSG solid input", *Computer-aided Design*, Vol. 22, No. 5, pp. 285-295, June 1990.
- [Persson78] Persson, H., "NC machining of arbitrarily shaped pockets", *Computer-aided Design*, Vol. 10, No. 3, pp. 169-174, May 1978.
- [Pratt87] Pratt, M.J., "Form features and their applications in solid modelling", *SIGGRAPH '87 Course Notes*, Vol. 26, July 1987.
- [Pratt88] Pratt, M.J., "Synthesis of an optimal approach to form feature modelling", *Proceedings of the 1988 ASME International Computers in Engineering Conference and Expositions*, Vol. 1, pp. 263-274, 1988.
- [Pressman77] Pressman, R.S. and Williams, J.E., "Numerical control and computer-aided manufacturing", Chapter 1, pp. 1-17, John Wiley & Sons, Inc., 1977.
- [Quillian68] Quillian, M.R., "Semantic memory", *Semantic Information Processing*, Minsky, M., (ed.), MIT Press, Cambridge, Mass. pp. 216-270, 1968.
- [Requicha78] Requicha, A.A.G. and Tilove, R.B., "Mathematical foundations of constructive solid geometry : general topology of regular closed sets", *Tech. Memo. No. TM-27a*, Production Automation Project, University of Rochester, Rochester, N.Y., June 1978.
- [Requicha80] Requicha, A.A.G., "Representations for rigid solids : theory, methods, and systems", *Computing Surveys*, Vol. 12, No. 4, pp. 437-464, December 1980.
- [Requicha82] Requicha, A.A.G. and Voelcker, H.B., "Solid modeling : a historical summary and contemporary assessment", *IEEE Computer Graphics & Applications*, pp. 9-24, March 1982.
- [Requicha83] Requicha, A.A.G., "Toward a theory of geometric tolerancing", *International Journal of Robotics Research*, Vol. 2, No. 4, 1983.
- [Requicha85a] Requicha, A.A.G. and Voelcker, H.B., "Boolean operations in solid modelling : boundary evaluation and merging algorithms", *Proc. IEEE*, Vol. 73, No. 1, pp. 30-44, January 1985.
- [Requicha85b] Requicha, A.A.G. and Chan, S., "Representation of geometric features, tolerances and attributes in solid modelers based on CSG", *Tech. Memo. No. 48*, Production Automation Project, University of Rochester, Rochester, N.Y., October 1985.

- [Rogers89] Rogers, D.F. and Adams, J.A., "Mathematical elements for computer graphics", McGraw-Hill Publishing Company, second international edition, 1989.
- [Roth82] Roth, S.D., "Ray casting for modeling solids", *Computer Graphics and Image Processing*, 18, pp. 109-144, 1982.
- [Saeed88] Saeed, S.E.O., de Pennington, A. and Dodsworth, J.R., "Offsetting in geometric modelling", *Computer-aided Design*, Vol. 20, No. 2, March 1988.
- [Sakurai88] Sakurai, H. and Gossard, D.C., "Shape feature recognition from 3D solid models", *Proceedings of the 1988 ASME International Computers in Engineering Conference and Expositions*, Vol. 1, pp. 515-519, 1988.
- [Sakurai91] Sakuari, H. and Gossard, D.C., "Geometric modelling in setup planning and fixture design", *Product Modelling for Computer-Aided Design and Manufacturing*, Turner, J., Pegna, J. and Wozny, M. (eds.), Elsevier Science Publishers B.V. (North-Holland), IFIP, pp. 299-313, 1991.
- [Samuel63] Samuel, A.L., "Some studies in machine learning using the game of checkers", *Computer and Thought*, Figenbaum, E.A. and Feldman, J., (eds.), McGraw-Hill, New York, pp. 71-105, 1963.
- [Samuel76] Samuel, N.M., Requicha, A.A.G. and Elkind, S.A., "Methodology and results of an industrial part survey", *Tech. Memo. No. 21, Production Automation Project, University of Rochester, Rochester, N.Y., July 1976.*
- [Shah88a] Shah, J.J., Sreevalsan, P.C., Rogers, M., Billo, R., and Mathew, A., "Current status of features technology", *Computer Aided Manufacturing-International, Inc., Revised Report, R-88-GM-04.1, Arlington, Texas, November 1988.*
- [Shah88b] Shah, J.J. and Rogers, M.T., "Feature base modelling shell : design and implementation", *Proceedings of the 1988 ASME International Computers in Engineering Conference and Expositions*, Vol. 1, pp. 255-261, 1988.
- [Shah90] Shah, J.J., Rogers, M.T., Sreevalsan, P.C., Hsiao, D.W., Mathew, A., Bhatnagar, A., Liou, B.B., and Miller, D.W., "The A.S.U. features testbed : an overview", *Proceedings of the 1990 ASME International Computers in Engineering Conference and Expositions*, pp. 233-241, 1990.

- [Shah91a] Shah, J.J., "Assessment of features technology", *Computer-aided Design*, Vol. 23, No. 5, pp. 331-343, June 1991.
- [Shah91b] Shah, J.J., "Conceptual development of form features and feature modelers", *Research in Engineering Design*, Springer-Verlag, New York Inc., Vol. 2, pp. 93-108, 1991.
- [Shapiro91] Shapiro, V. and Vossler, D.L., "Construction and optimization of CSG representations", *Computer-aided Design*, Vol. 23, No. 1, pp. 4-20, January/February 1991.
- [Simon83] Simon, H.A., "Why should machines learn ?", *Machine Learning, An Artificial Intelligence Approach*, Michalski, R.S., Carbonell, J.G. and Mitchell, T.M., (eds.), Tioga Publishing Co., Palo Alto, CA, pp. 25-37, 1983.
- [Simon88] Simon, W., Ersuu, E., Gose, H., and Zoll, M., "A real-time knowledge scheme for sensory-controlled robot assembly tasks, *Proceedings of the Symposium on Robot Control*, Karlsruhe, FRG, October, 1988.
- [Silva81] Silva, C.E., "Alternative definitions of faces in boundary representations of solid objects", *Tech. Memo. No. 36*, Production Automation Project, University of Rochester, 1981.
- [Smithers89] Smithers, T., Conkie, A., Doheny, J., Logan, B., and Millington, K., "Design as intelligent behaviour : an AI in design research programme", *Artificial Intelligence in Design*, Gero, J.S., (ed.), *Proceedings of the 4th International Conference on the Applications of Artificial Intelligence in Engineering*, Cambridge, UK, July, 1989, Computational Mechanics Publications, Springer-Verlag, pp. 293-334, 1989.
- [Tang91] Tang, K. and Woo, T., "Algorithmic aspects of alternating sum of volumes. Part 2 : nonconvergence and its remedy", *Computer-aided Design*, Vol. 23, No. 6, pp. 435-443, July/August 1991.
- [Thomson86] Thomson, V. and Graefe, U., "CIM - A manufacturing paradigm", *Division of Mechanical Engineering Report, DM-6*, NRC No. 26198, National Research Council Canada, 1986/7.
- [Tiller84] Tiller, W. and Hanson, E.G. "Offsets of two-dimensional profiles", *IEEE Computer Graphics and Applications*, pp. 36-46, September 1984.



- [Tilove80] Tilove, R.B., "Set membership classification : a unified approach to geometric intersection problems", IEEE Transactions on Computers, Vol. C-29, No. 10, pp. 874-883, October 1980.
- [Tilove81] Tilove, R.B., "Line/Polygon classification : a study of the complexity of geometric computation", IEEE Computer Graphics & Applications, Vol. 1, No. 2, pp. 75-83, April 1981.
- [Tilove84] Tilove, R.B., Requicha, A.A.G. and Hopkins, M.R., "Efficient editing of solid models by exploiting structural and spatial locality", Tech. Memo. No. 46, Production Automation Project, University of Rochester, Rochester, N.Y., May 1984.
- [Tsang89] Tsang, J.P. and Brissaud, D., "A feature-based approach to process planning", Proceedings of the 1989 ASME International Computers in Engineering Conference and Expositions, Vol. 1, pp. 419-430, 1989.
- [Turner88] Turner, G.P. and Anderson, D.C., "An object-oriented approach to interactive, feature-based design for quick turnaround manufacturing", Proceedings of the 1988 ASME International Computers in Engineering Conference and Expositions, Vol. 1, pp. 551-555, 1988.
- [Unger88] Unger, M.B. and Ray, S.R., "Feature-based process planning in the AMRF", Proceedings of the 1988 ASME International Computers in Engineering Conference and Expositions, Vol. 1, pp. 563-569, 1988.
- [Voelcker81] Voelcker, H.B. and Requicha, A.A.G., "Boundary evaluation procedures for objects defined via constructive solid geometry", Tech. Memo. No. 26, Production Automation Project, University of Rochester, 1981.
- [Weghorst8] Weghorst, H., Hooper, G., and Greenberg, D.P., "Improved computational methods for ray tracing", ACM Transactions on Graphics, Vol. 3, No. 1, pp. 52-69, January 1984.
- [Weiler85] Weiler, K., "Edge-based data structures for solid modelling in curved-surface environments", IEEE Computer Graphics & Applications, pp. 21-40, January 1985.
- [Weiler88] Weiler, K., "Boundary graph operators for non-manifold geometric modeling topology representations", Geometric Modeling for CAD Applications, Wozny, M.J., McLaughlin, H.W. and Encarnacao, J.L., (eds.), Elsevier Science Publications B.V., North-Holland, pp. 37-66, 1988.

- [Wilson88] Wilson, P.R. and Pratt, M.J., "A taxonomy of features for solid modeling", *Geometric Modeling for CAD Applications*, Wozny, M.J., McLaughlin, H.W. and Encarnacao, J.L., (eds.), Elsevier Science Publications B.V., North-Holland, pp. 125-136, 1988.
- [Winston75] Winston, P.H., "Learning structural descriptions from examples", *The Psychology of Computer Vision*, Winston, P.H., (ed.), McGraw-Hill, New York, pp.157-209, 1975.
- [Woo75] Woo, T.C., "Computer aided recognition of volumetric designs", *Advances in Computer-Aided Manufacture*, McPherson, D., (ed.), North-Holland Publishing Company, pp. 121-136, 1977.
- [Woo82] Woo, T.C., "Feature extraction by volume decomposition", *Proceedings of the Conference on CAD/CAM Technology in Mechanical Engineering*, MIT, Cambridge, MA, pp. 76-94, March 1982.
- [Wysk77] Wysk, R.A, "Automatic process planning and selection - APPAS", Ph.D. Thesis, Purdue University, May 1977.
- [Yuen88] Yuen, M.M.F., Chan K.W., Sze, W.S. and Tan, S.T., "A CIM implementation feasibility study", *Proceedings of the 4th International Conference on Computer-aided Production Engineering*, University of Edinburgh, pp. 265-269, November 1988.

## APPENDIX A

### DERIVATION OF THE CAVITY VOLUME BOUNDARY EXPRESSION

Before explaining the derivation, some symbols and their corresponding definitions are introduced first :

- $\in$  = a subset of
- $W$  = world set of 3D Euclidean space
- $d(x_1, x_2)$  = metric distance between points  $x_1$  and  $x_2$  in  $W$
- $B(x, r)$  = a set in the form of an open ball of radius  $r$  about a point  $x$  of a subset  $X$  in  $W$  that satisfies  $d(x, y) < r$ ,  
such that  $y \in B(x, r) \in W$ , and  $x \in X \in W$
- $ix$  = an interior point of a subset  $X$  in  $W$  which contains  $B(ix, r)$
- $cX$  = complement of a subset  $X$  in  $W$   
=  $W - X$
- $\wedge$  = set intersection
- $bx$  = a boundary point of a subset  $X$  in  $W$  such that  
 $B(bx, r) \wedge X$ , and  $B(bx, r) \wedge cX$
- $iX$  = a set of all the interior points  $ix$  of subset  $X$  in  $W$
- $bX$  = a set of all the boundary points  $bx$  of a subset  $X$  in  $W$
- $\cup$  = set union
- $kX$  = the closure of  $X$   
=  $iX \cup bX$
- $rX$  = a regular set  $X$   
=  $kiX$
- $\langle c \rangle X$  = regular complement of a subset of  $X$   
=  $rcX$
- $\langle - \rangle$  = regularized set subtraction
- $\langle \wedge \rangle$  = regularized set intersection

The following regular point-set properties are also needed for explanation of the derivation process :

Property A1 :

If  $X$  and  $Y$  are regular point sets then  $X \leftrightarrow Y = X \wedge \langle c \rangle Y$

Property A2 :

If  $X$  and  $Y$  are regular point sets then

$$b(X \wedge Y) = (bX \wedge iY) \cup (iX \wedge bY) \cup [bX \wedge bY \wedge k(iX \wedge iY)]$$

Property A3 :

If  $X$  is a regular point set then  $i \langle c \rangle X = cX$

Property A4 :

If  $X$  is a regular point set then  $b \langle c \rangle X = bX$

The proof of the above properties is very laborious and requires a rigorous and fundamental discussion of the regularized point-set theory. Hence, the proof is not included in this thesis, and interested readers are recommended to study the references [Kuratoswski76, Mendelson75, Requicha78].

As defined in section 4.2 (chapter 4), the cavity volume model  $V$  is the total volume of material machined from  $S$  to produce  $P$ , which can be expressed as :

$$V = S \leftrightarrow P$$

By using the boundary point set operator  $b$  defined above, the surface boundary of the cavity volume  $V$  can be expressed as :

$$bV = b(S \leftrightarrow P)$$

$$= b(S \wedge \langle c \rangle P) \tag{i}$$

$$= (bS \wedge i \langle c \rangle P) \cup (iS \wedge b \langle c \rangle P) \cup [bS \wedge b \langle c \rangle P \wedge k(iS \wedge i \langle c \rangle P)] \tag{ii}$$

$$= (bS \wedge cP) \cup (iS \wedge b \langle c \rangle P) \cup [bS \wedge b \langle c \rangle P \wedge k(iS \wedge cP)] \tag{iii}$$

$$= (bS \wedge cP) \cup (iS \wedge bP) \cup [bS \wedge bP \wedge k(iS \wedge cP)] \tag{iv}$$

**Remarks :**

- Expression (i) is obtained by applying Property A1  
 ,, (ii) is obtained by applying Property A2  
 ,, (iii) is obtained by applying Property A3  
 ,, (iv) is obtained by applying Property A4

The meaning of expression (iv) is illustrated in Fig. A1. It can be seen that the last term,  $[bS \wedge bP \wedge k(iS \wedge cP)]$ , basically represents the edges formed by the intersection of the cavity volume boundary faces. As the focus of interest is on the cavity volume boundary faces, the last term is ignored.

$$\text{Hence, } bV = (bS \wedge cP) \vee (iS \wedge bP)$$

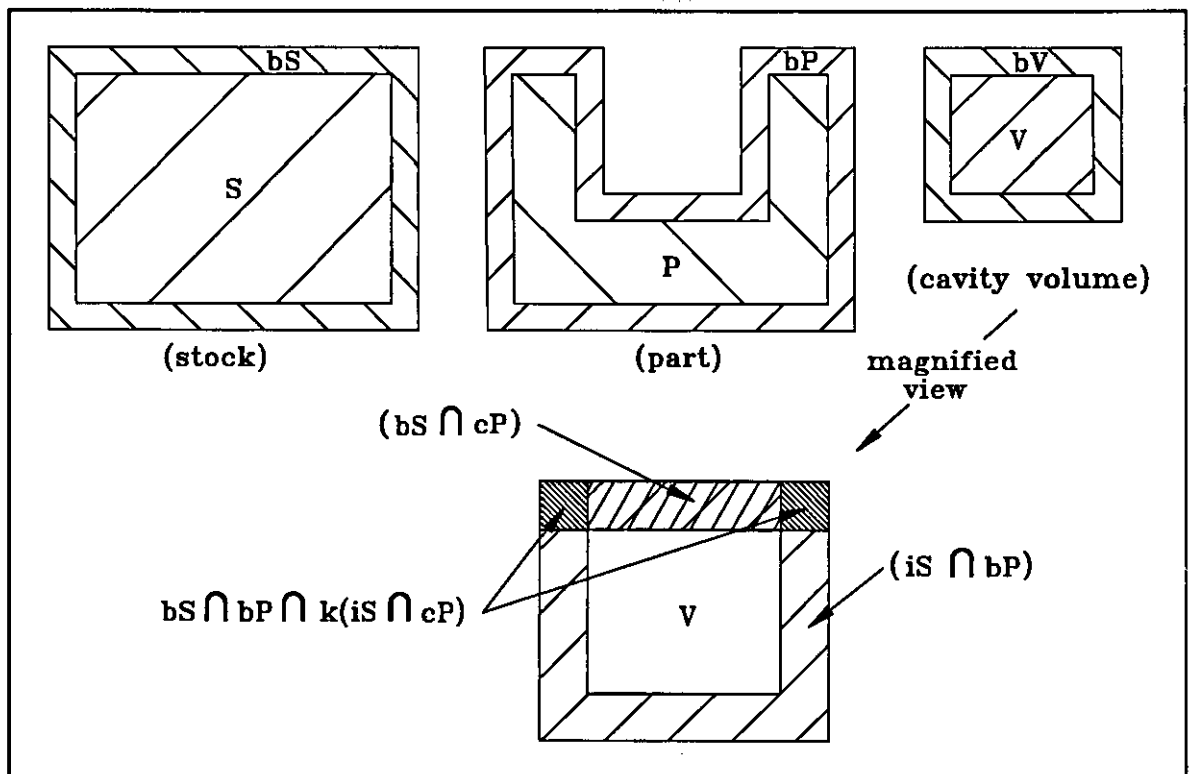


Figure A1 : Illustration of the cavity volume boundary expression.

## APPENDIX B

### FEATURE REPRESENTATION - ILLUSTRATED EXAMPLES

Figure B1 illustrates a stepped blind hole. The representation of the lower and upper holes together with some of their geometric and topological information that can be deduced from the cavity volume boundary database are summarized in Tables B1, B2, B3, B4, and B5.

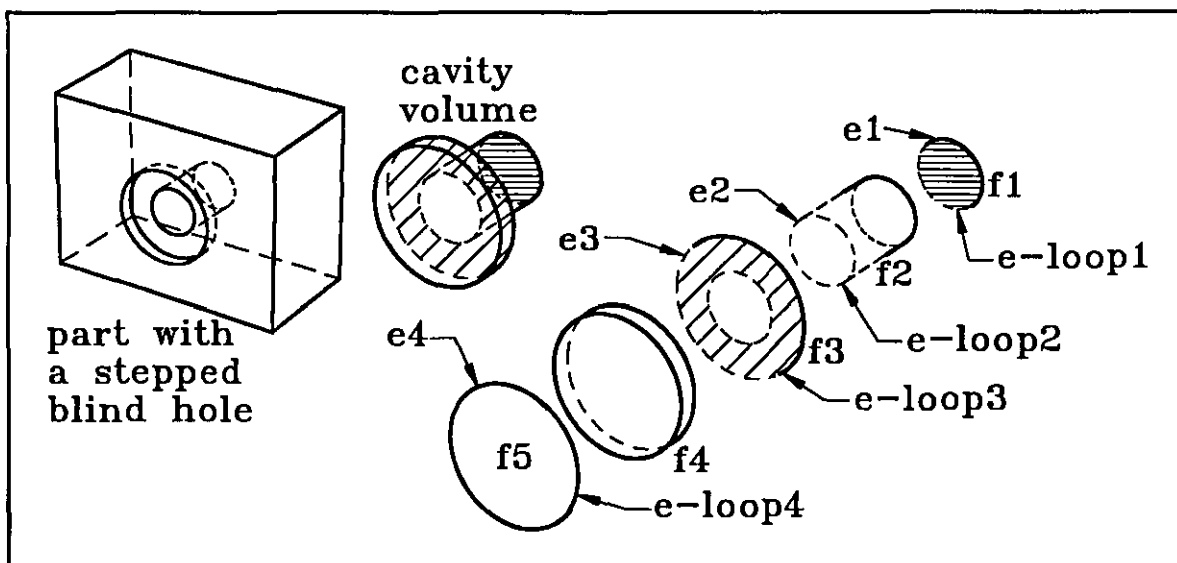


Figure B1 : An example part with a stepped hole.

Feature : the lower hole	part face	check face	primary top entrance face	secondary top entrance face	side entrance face
cutter axis vector: axis of f2					
number of face	1	1	1	1	nil
face id.	f1	f2	f5	f3	nil

Table B1 : Feature representation of the lower hole shown in Fig. B1.

<b>Feature : the upper hole</b>	<b>part face</b>	<b>check face</b>	<b>primary top entrance face</b>	<b>secondary top entrance face</b>	<b>side entrance face</b>
<b>cutter axis vector: axis of f4</b>					
<b>number of face</b>	1	1	1	nil	nil
<b>face id.</b>	f3	f4	f5	nil	nil

Table B2 : Feature representation of the upper hole shown in Fig. B1.

<b>face id.</b>	<b>surface type</b>	<b>nature</b>	<b>edge loop id.</b>
f1	planar	machined_face	e-loop1
f2	cylindrical	machined_face	e-loop1, e-loop2
f3	planar	machined_face	e-loop2, e-loop3
f4	cylindrical	machined_face	e-loop3, e-loop4
f5	planar	tool_entrance_face	e-loop4

Table B3 : Face information.

<b>e-loop id.</b>	<b>constituent edge</b>	<b>inner/outer loop</b>
e-loop1	e1	outer
e-loop2	e2	inner
e-loop3	e3	outer
e-loop4	e4	outer

Table B4 : Edge loop information.

<b>edge id.</b>	<b>curve type</b>	<b>convexity</b>
e1, e3, e4	ellipse	convex
e2	ellipse	concave

Table B5 : Edge information.

Figure B2 shows a part with a rectangular boss that is assumed to be produced by removing its surrounding and upper part material by means of two surface milling operations. Similarly, the representation of the two machining features is summarized in Tables B6, B7, B8, B9 and B10.

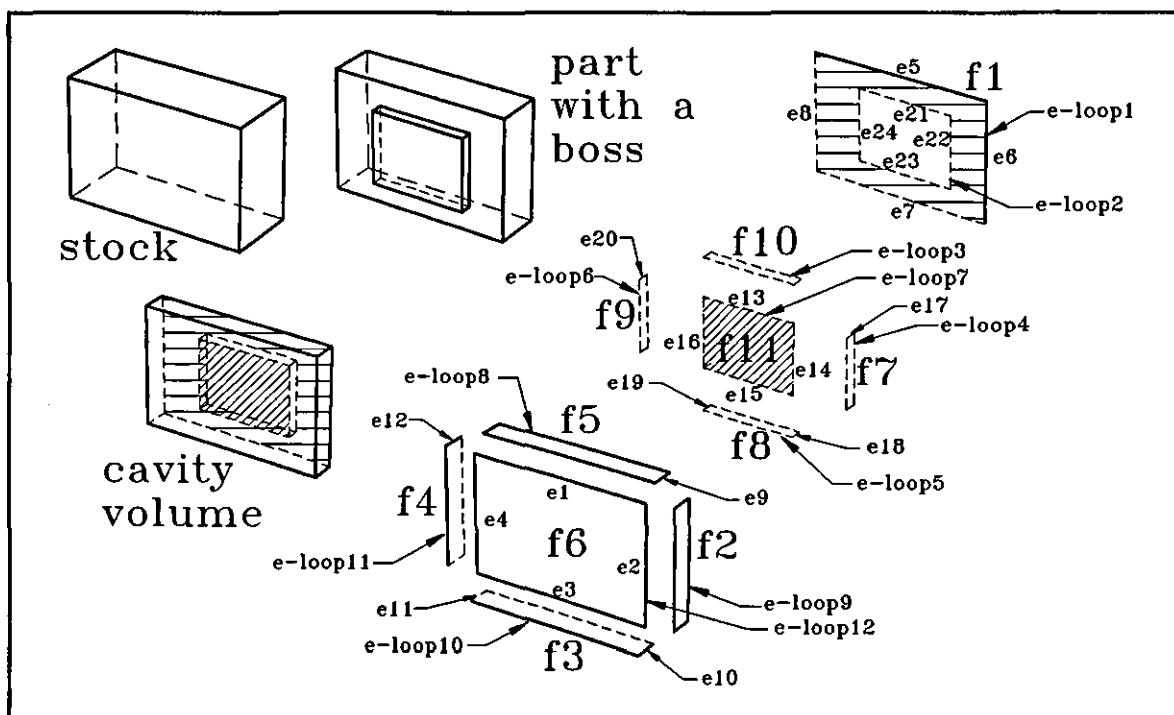


Figure B2 : An example part with a rectangular boss.

Feature : surface milling	part face	check_face	primary top entrance face	secondary top entrance face	side entrance face
<b>cutter axis vector:</b> for example, using the curve vector of edge e9 towards f6					
<b>number of face</b>	1	8	1	1	4
<b>face id.</b>	f1	f2, f3, f4, f5, f7, f8, f9, f10	f6	f11	f2, f3, f4, f5

Table B6 : Feature representation of the surface milling shown in Fig. B2.



<b>Feature : boss top milling</b>	<b>part face</b>	<b>check_face</b>	<b>primary top entrance face</b>	<b>secondary top entrance face</b>	<b>side entrance face</b>
<b>cutter axis vector:</b> for example, using the curve vector of edge e17 towards f6					
<b>number of face</b>	1	4	1	nil	nil
<b>face id.</b>	f11	f7, f8, f9, f10	f6	nil	nil

Table B7 : Feature representation of the boss top milling shown in Fig. B2.

<b>face id.</b>	<b>surface type</b>	<b>nature</b>	<b>edge loop id.</b>
<b>f1</b>	planar	machined_face	e-loop1, e-loop2
<b>f2</b>	planar	tool_entrance_face	e-loop9
<b>f3</b>	planar	tool_entrance_face	e-loop10
<b>f4</b>	planar	tool_entrance_face	e-loop11
<b>f5</b>	planar	tool_entrance_face	e-loop8
<b>f6</b>	planar	tool_entrance_face	e-loop12
<b>f7</b>	planar	machined_face	e-loop4
<b>f8</b>	planar	machined_face	e-loop5
<b>f9</b>	planar	machined_face	e-loop6
<b>f10</b>	planar	machined_face	e-loop3
<b>f11</b>	planar	machined_face	e-loop7

Table B8 : Face information.

e-loop id.	constituent edge	inner/outer loop
e-loop1	e5, e6, e7, e8	outer
e-loop2	e21, e22, e23, e24	inner
e-loop3	e13, e17, e20, e21	outer
e-loop4	e14, e17, e18, e22	outer
e-loop5	e15, e18, e19, e23	outer
e-loop6	e16, e19, e20, e24	outer
e-loop7	e13, e14, e15, e16	outer
e-loop8	e1, e5, e9, e12	outer
e-loop9	e2, e6, e9, e10	outer
e-loop10	e3, e7, e10, e11	outer
e-loop11	e4, e8, e11, e12	outer
e-loop12	e1, e2, e3, e4	outer

Table B9 : Edge loop information.

edge id.	curve type	convexity
e1, e2, e3, e4, e5, e6, e7, e8, e9, e10, e11, e12, e20, e21, e22, e23, e24	line	convex
e13, e14, e15, e16, e17, e18, e19, e20	line	concave

Table B10 : Edge information.

## APPENDIX C

### LINE/SURFACE INTERSECTION

As the part models are assumed to contain planar and cylindrical faces, two types of line/surface intersection need to be considered : line/plane and line/cylinder.

#### Line/Plane Intersection

The problem is to find the coordinates of the point of intersection between a line and a plane. Let a line (or a ray) be defined in a parametric form as a point  $(x_0, y_0, z_0)$  and a direction vector  $(dx, dy, dz)$ . For a parameter  $t$ , any point  $(x, y, z)$  on the line is given by

$$x = x_0 + t * dx$$

$$y = y_0 + t * dy$$

$$z = z_0 + t * dz$$

For simplicity, consider the intersection of the parametrized line

$$[ (x_0, y_0, z_0) (dx, dy, dz) ] \text{ with the XY plane,}$$

solving the two simultaneous equations :

$$z = 0$$

$$z = z_0 + t * dz$$

gives  $t = -z_0/dz$

Having found the parameter value  $t$ , the point of intersection can be found as :

$$[x_0 + (-z_0/dz)dx, y_0 + (-z_0/dz)dy, 0]$$

If  $dz$  is zero, the line is parallel to the plane, so they do not intersect.

**Line/Cylinder Intersection**

For simplicity, consider a cylindrical surface  $P = \{(x, y, z) : x^2 + y^2 + z^2 < R^2\}$ , and  
 $-\infty < z < +\infty$

Substituting the x and y components of the line's equation yields

$$(x_0 + t * dx)^2 + (y_0 + t * dy)^2 = R^2$$

Rearranging gives

$$t^2 [(dx)^2 + (dy)^2] + 2 t (x_0 * dx + y_0 * dy) + x_0^2 + y_0^2 - R^2 = 0$$

Using the quadratic formula, parameter t can be found as :

$$t = [ - B +/- \text{Sqrt} (B^2 - 4AC) ] / 2A$$

$$\text{where } A = (dx)^2 + (dy)^2$$

$$B = 2 (x_0 * dx + y_0 * dy)$$

$$C = x_0^2 + y_0^2 - R^2$$

The line will intersect the cylindrical surface only if A is not equal to zero and  $(B^2 - 4AC)$  is greater than or equal to zero.

Having found t, the intersection point can be found as in the line/plane case.

## APPENDIX D

### LINE/POLYGON INTERSECTION

In the recognition algorithm, it is required to determine whether a point  $P$  lies inside or outside the boundary of a potential part face. The boundary edges of the potential part face can be considered as a polygon of line segments since non-linear edge segments can be approximated with line segments. When a line (or ray) is projected from a starting point  $P$  to a destination point  $Q$  such that line  $PQ$  cut across the polygon boundary, a number of intersection points will be created (Fig. D1). If the number of intersection point  $N$  is even then the original point  $P$  is outside the polygon, whereas if  $N$  is odd then  $P$  is inside the polygon. Thus the problem is virtually reduced to finding Line/Line intersection between the projected line  $PQ$  and the line segments of the polygon.

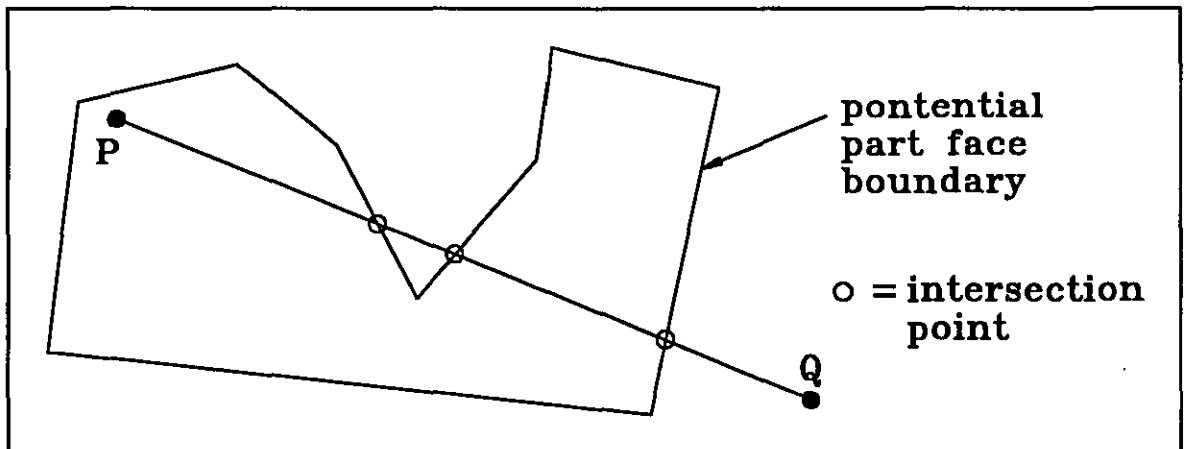


Figure D1 : The notion of line/polygon intersection.

#### Line/Line Intersection

Let the line  $PQ$  (or ray) be expressed parametrically as a starting point  $P_0$  and a unit direction vector  $V_p$ , i.e.  $PQ(t) = P_0 + t * V_p$

Similarly, let a line segment AB of the potential part face boundary be defined as a starting point  $M_o$  and a unit direction vector  $W_m$ , i.e.  $AB(s) = M_o + s * W_m$

The intersection occurs when  $PQ(t) = AB(s)$  or equivalently when

$$P_o + t * V_p = M_o + s * W_m$$

Subtracting  $P_o$  from both sides and vector cross multiplying with  $W_m$  yields

$$(V_p \times W_m) * t = (M_o - P_o) \times W_m$$

where  $\times$  denotes vector cross multiplication

Hence  $t = \{(M_o - P_o) \times W_m\} / (V_p \times W_m)$ .

Having found  $t$ , the intersection point can be found from :

$$PQ(t) = P_o + t * V_p$$

If the intersection point lies between the endpoints of AB then a valid intersection is counted, otherwise there is no intersection counted.

## APPENDIX E

# THE MODIFIED WINGED-EDGE DATA STRUCTURE

The modified winged-edge data structure was proposed by Weiler [Weiler85]. It is an enhanced version of the winged-edge data structure originally proposed by Baumgart [Baumgart74] for representing the adjacency relationships of topological entities (i.e. faces, edges and vertices) of a polyhedral object in a computer.

As can be seen in Fig. E1, the winged-edge structure is an edge-based structure since an edge is used as a reference to access its adjacent entities : two faces, four edges and two ending vertices. The clockwise and counter-clockwise names used in the figure refer to their use in determining the cycle of edges surrounding a face, as viewed from outside the solid looking towards the reference edge. However, as there is no explicit indication of which side of the edge pointed at is intended, an extra test has to be performed in data structure traversal routines to ensure that the desired topological entity is consistently retrieved.

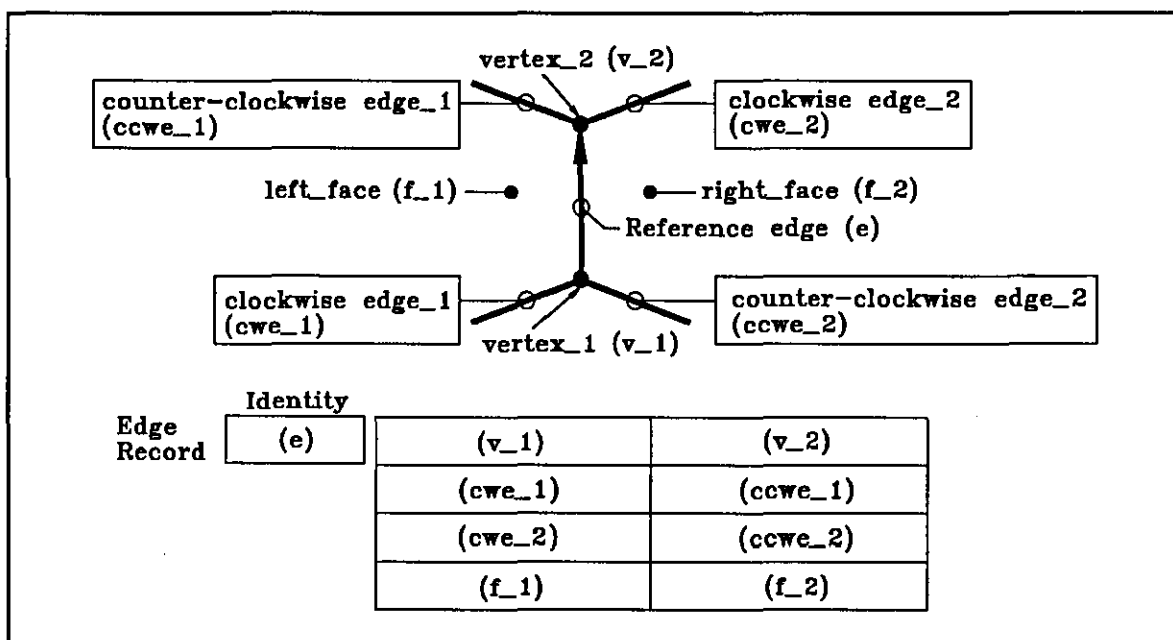


Figure E1 : The original winged-edge data structure.

For instance, as shown in Fig. E2, given the left\_face  $f_1$  and the reference edge  $e$ , it is desired to find the boundary edges of  $f_1$ . Starting from the reference edge  $e$ , the next edge to be retrieved around the sequence of edges of  $f_1$  can be either  $ec$  (i.e. the  $cwe\_1$  of  $e$ ) or  $ea$  (i.e. the  $ccwe\_1$  of  $e$ ). Assuming that  $ea$  is taken, the next edge to be retrieved around the edge cycle should be the new edge  $eb$ . However, as the pointing side of edge  $ea$  is not known, the next edge to be retrieved can be the reference edge  $e$  or the new edge  $eb$ . In order to correctly select  $eb$ , a test of the edge's pointing or traversing direction is necessary. For instance, the test can be done by comparing the ending vertices of the edges.

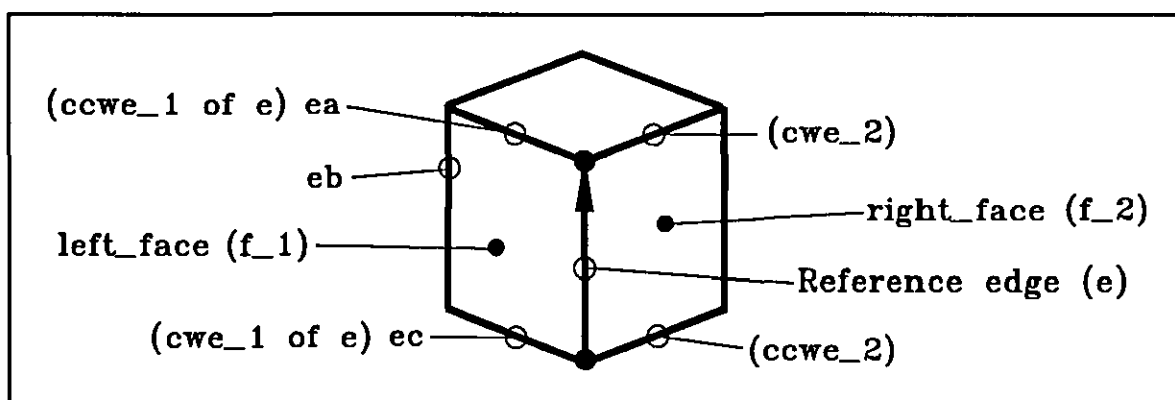


Figure E2 : Edge retrieval with the original winged-edge data structure.

In view of this drawback, Weiler [Weiler85] improved the original winged-edge structure by introducing an additional field at each of the four adjacent edges of a reference edge as shown in Fig. E3.

The additional field is called an edge half (or edge side) field which indicates explicitly which side of the edge pointed at is intended. In the prototype system, the edge half field is implemented as an integer value of either 1 or 2 that represents the pointing direction of the reference edge as illustrated in Fig. E4.



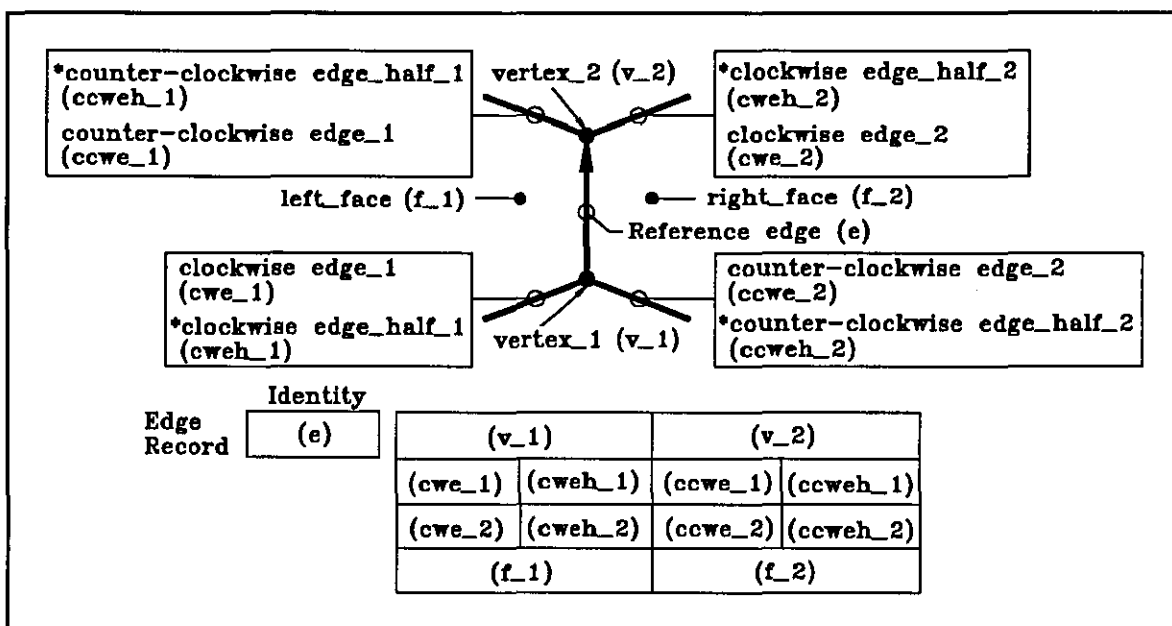


Figure E3 : The modified winged-edge data structure.

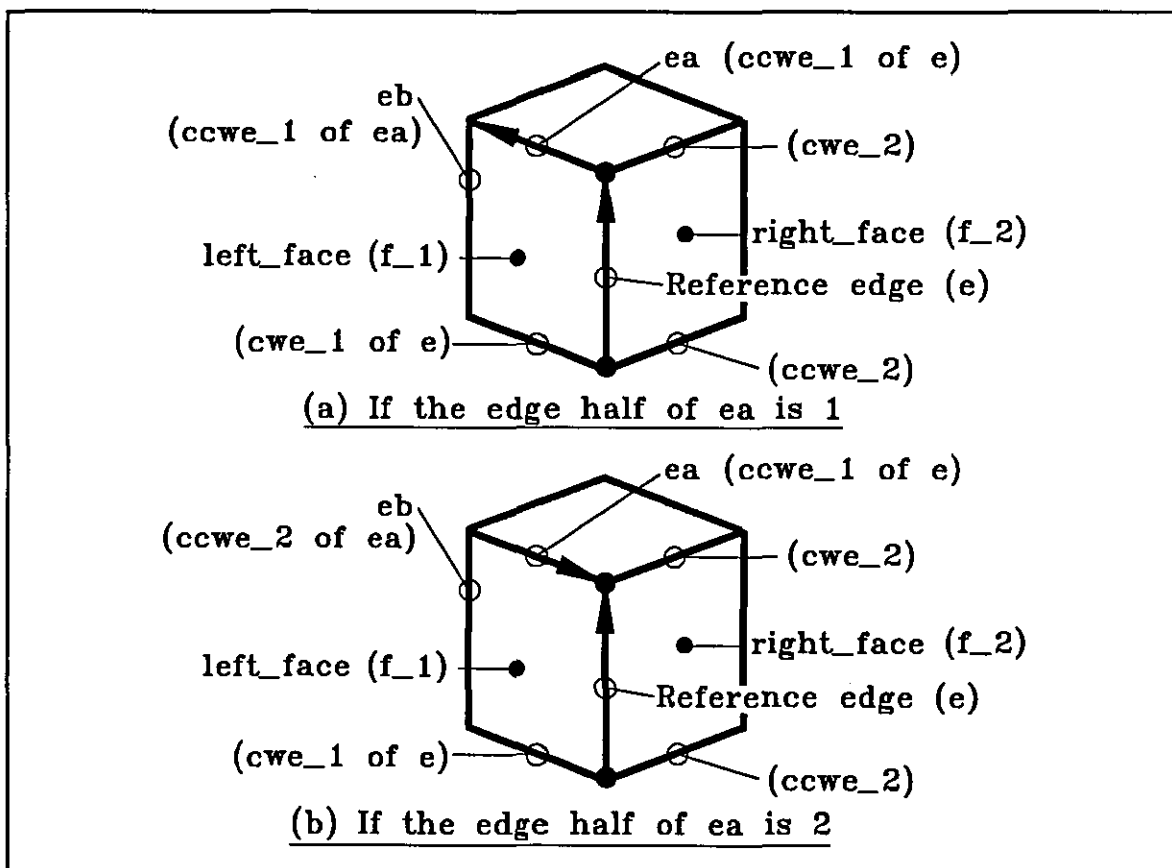


Figure E4 : Edge-half configuration of the modified winged-edge data structure.

Using the above example again, and assuming that the edge half of the starting reference edge is 1, the next edge  $ea$  (i.e. the  $ccwe\_1$  of  $e$ ) is the next edge to be retrieved from the edge record of  $e$ . At the same time, the edge half integer value of  $ea$  (i.e. the  $ccweh\_1$  of  $e$ ) is also retrieved from the edge record of  $e$ . If the edge half integer value of  $ea$  is 1, the next edge to be selected is  $eb$  which is also the  $ccwe\_1$  of  $ea$  as shown in Fig. E5(a). If the edge half integer value of  $ea$  is 2, the next edge to be selected is the  $ccwe\_2$  of  $ea$  which is also  $eb$  as shown in Fig. E5(b). Hence, there is no need to perform expensive test to decide on the next edge to be retrieved despite the fact that the implementation of the modified winged-edge structure is also at the expense of more system memory.

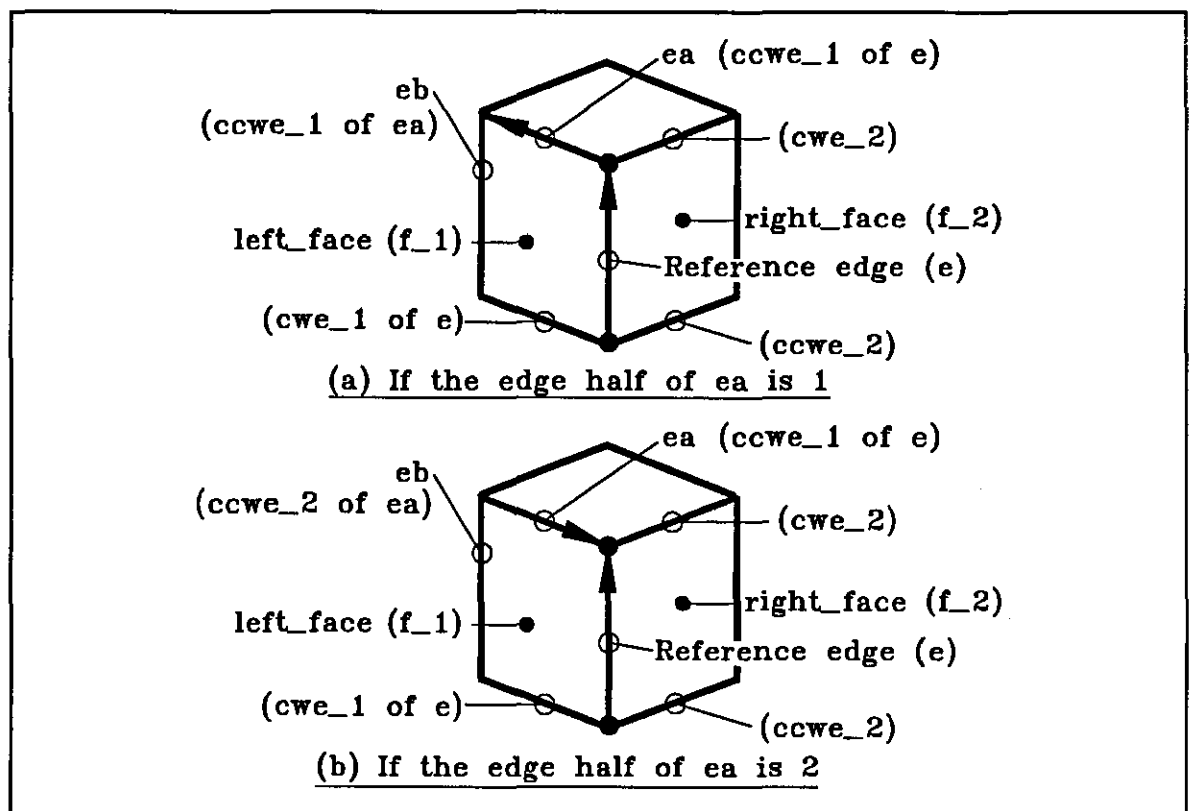


Figure E5 : Edge retrieval with the modified winged-edge data structure.

A set of data structure traversal routines are developed to support the management of the modified winged-edge data structure. For instance, given a reference edge as input, a routine can return its two adjacent faces.

## APPENDIX F

ALIGNING THE CUTTER AXIS VECTOR  
WITH THE Z-AXIS

As shown in Fig. F1, the basic problem is to align the cutter axis vector  $C$  with the z-axis of the system so as to become  $C^*$ . This can be done first by rotating  $C$  about the y-axis an angle  $-a$  so that  $AB$  is collinear with the z-axis. Following the above rotation,  $C$  is then rotated about the x-axis an angle  $b$  so that  $C$  is collinear with the z-axis. Angles  $a$  and  $b$  can be found by the following equations :

$$\begin{aligned} r &= |C| = \sqrt{u^2 + v^2 + w^2} \\ \sin b &= v / r \\ \tan a &= u / w \end{aligned}$$

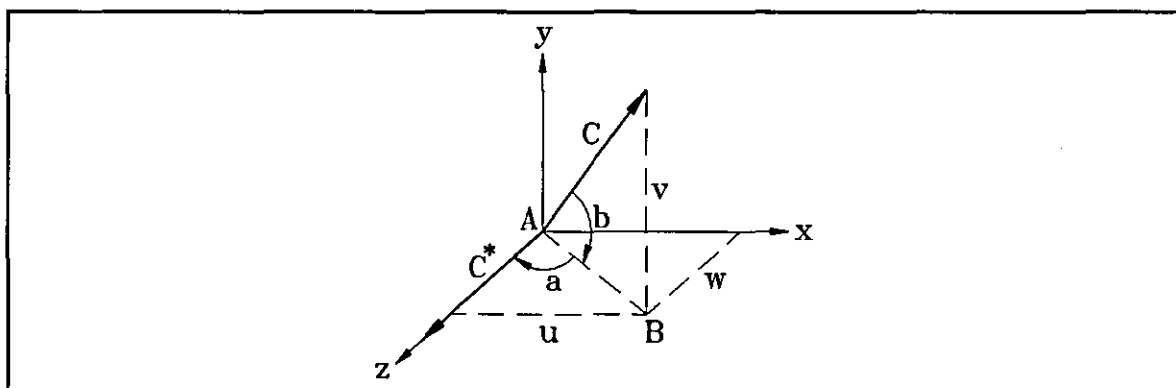


Figure F1 : Aligning the cutter axis vector with the z-axis.

In matrix form, the rotation can be expressed as :

$$C^* = [Ry] [Rx] [C]$$

where

$$[Ry] = \begin{bmatrix} \cos -a & 0 & \sin -a \\ 0 & 1 & 0 \\ -\sin -a & 0 & \cos -a \end{bmatrix} \quad (\text{rotation about y-axis})$$

$$[Rx] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos b & -\sin b \\ 0 & \sin b & \cos b \end{bmatrix} \quad (\text{rotation about x-axis})$$

$$[C] = \begin{bmatrix} u \\ v \\ w \end{bmatrix}$$

Thus the required rotational transformation matrix is  $[Ry][Rx]$ .

