Intelligent Techniques for Automatic Feature Recognition in CAD Models

A thesis

submitted to the

University of Wales, Cardiff

for the degree of

Doctor of Philosophy

by

Emmanuel Brousseau

Intelligent Systems Laboratory

Manufacturing Engineering Centre

School of Engineering

University of Wales, Cardiff

United Kingdom

December 2004

Summary

Automatic Feature Recognition (AFR) techniques are an important tool for achieving a true integration of design and manufacturing stages during the product development. In particular, AFR systems offer capabilities for recognising high-level geometrical entities in Computer-Aided Design (CAD) models. These entities represent the features that are semantically significant for downstream applications of engineering databases, for instance manufacturing. For the past twenty years, numerous AFR techniques have been proposed. However, most of them are domain specific. The research reported in this thesis presents a new AFR method that could be applied easily in different domains.

First, a method for automatic formation of feature recognition rules is developed. The method utilises inductive learning techniques to generate rules from a set of examples representing features in CAD models.

Next, a hybrid AFR method is proposed that employs the rule bases. In particular, this method combines the 'learning from examples' concept with the rule-based and hint-based approaches in order to benefit from their respective strengths. Also, a new technique is presented for automatic definition of feature hints that overcomes a major limitation of the hint-based AFR approach.

To extend the capabilities of the AFR method, a geometric reasoning algorithm is developed to tackle the problems associated with the recognition of interacting features.

The solutions suggested in this research are implemented in a prototype AFR system and its performance verified on commonly used benchmarking parts that are composed of machining features.

Acknowledgements

First of all, I would like to thank the director of the Manufacturing Engineering Centre, Professor D. T. Pham, for giving me the opportunity to study in the Intelligent Systems Laboratory and for his continuous support throughout my research.

I would like to express my gratitude to my first supervisor Dr. S. Dimov for his technical guidance and help in correcting this thesis and to my second supervisor Dr. R. Setchi for her constructive suggestions and remarks.

Many thanks to current and former members of the Intelligent Systems Laboratory for providing a friendly work environment. In particular, the members of the Intelligent Information Systems group: Dr. R. Setchi, Dr. B. Peat, Dr. A. Soroka, Miss D. Tsaneva, Mr. C. Pasantonopoulos, Mr. A. Huneiti, Mr. Q. Tang, Mr. N. Lagos, Mr. A. Noyvirt and Mr. V. Zlatanov, as well as Dr A. Afify and Dr. S. Bigot, are thanked for their help through technical discussions and collaborations.

I am very grateful to all of my friends for making my stay in Cardiff extremely pleasant. I would like to thank my dear friend Miss W. L. Wong for her patience and understanding. Finally, my deepest gratitude goes to my family who has given me constant support and encouragement.

Declaration

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed ....(E. B. Brousseau - Candidate)

Date ....21/12/04..........................

Statement 1

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by giving explicit references. A bibliography is appended.

Signed ....(E. B. Brousseau - Candidate)

Date ......21/12/04......................

Statement 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed ....(E. B. Brousseau - Candidate)

Date ......21/12/04......................

Table of contents

List of figures

List of tables

Abbreviations

AAG	Attributed Adjacency Graph
AIC	Application Interpreted Construct
AFR	Automatic Feature Recognition
AP	Application Protocol
B-Rep	Boundary Representation
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
cc	Concave
cl	Cylindrical
cv	Convex
CSG	Constructive Solid Geometry
FLG	F-Loops Graph
GMF	General Machining Feature
ho_bl	Blind Hole
ho_th	Through Hole
IS	International Standard
ISO	International Organization for Standardization
MCSG	Minimal Condition Sub-Graph
MRSEV	Material Removal Shape Element Volume
ne	Neutral
OVR	Overall Removable Volume
pl	Planar
po_ob	Obround Pocket

po_re	Rectangular Pocket
pr_ci	Circular Protrusion
pr_re	Rectangular Protrusion
SC	Sub-Committee
sl_nt	Non-Through Slot
sl_th	Through Slot
st	Step
TC	Technical Committee
STEP	STandard for the Exchange of Product model data
VLG	Virtual Link Graph

Nomenclature

ccAd	number of concave edges of the adjacent faces divided by the number of such faces
D_P^R	distance between a rule R and a plant P
e_i	an edge
E_1, E_2, M	heuristic measures
f	number of models covered by a condition for a feature class
F	number of models for a feature class
faceCv	face convexity
faceTy	face type
f_i	a face
f_i^{II}	a type II face
f_i^{III}	a type III face
f_i^{IV}	a type IV face
f_i^{V}	a type V face
f_i^{VI}	a type VI face
G	set of possible groupings of faces
G_i	set of possible combinations of i faces
m	number of faces for a feature
n	number of characteristic vectors covered by a condition and not belonging to the target feature class
	number of faces for a part

	number of feature classes of a taxonomy
nCcClFa	number of concave cylindrical feature faces
nCcEd	number of concave edges
	number of concave internal edges
nClAd	number of cylindrical adjacent faces
nCvClFa	number of convex cylindrical feature faces
nCvEd	number of convex edges
	number of convex internal edges
nEd	number of edges defining the face border
nFa	number of feature faces
nPlAd	number of planar adjacent faces
nPlFa	number of planar feature faces
nSccEd	number of smooth concave edges
nScvEd	number of smooth convex edges
p	number of characteristic vectors covered by a condition and belonging to the target feature class
P	a plant
R	a rule
S_i	a solid primitive
v_i	a vertex
V^i_{max}	maximum value of the ith attribute in the training data
V^i_{min}	minimum value of the ith attribute in the training data
V^i_P	value of the ith attribute for the characteristic vector of a plant P

$Vmax_R^i$ higher bound employed in a rule R to form a condition on the ith attribute

$Vmin_R^i$ lower bound employed in a rule R to form a condition on the ith attribute

Chapter 1 - Introduction

1.1 Motivation

Market pressures force companies to reduce the lead time from the conceptual design of products to their serial production. In order to stay competitive, the companies also have to manufacture the products up to their technical specification at a minimum cost. Such market pressures have led to the development of concurrent engineering practices that require the design of products and processes to be integrated and carried out simultaneously. To achieve this, complete and accurate information about products, production processes and manufacturing operations is essential. The introduction of formal techniques into the different product development phases contributes to such integration. In particular, these techniques allow data-rich engineering models to be created and thus, used as a communication medium between different design and manufacturing teams.

The realisation of a true integration between the product and process design stages is a challenging goal and it requires a consistent utilisation of product information at different levels of abstraction. One of the data representation schemes that is widely used to interface Computer-Aided Design (CAD) and Computer-Aided Manufacturing (CAM) processes is the Boundary Representation (B-Rep) scheme. However, in spite of its popularity, this scheme has some drawbacks (Dimov et al., 2004). In particular, the geometrical data stored using the B-Rep scheme cannot be utilised directly for process design because it lacks high-level geometrical entities that are meaningful from a manufacturing point of view.

To bridge this information gap between CAD and CAM systems, several approaches have been developed based on the concept of *features*. Features can be considered in a broad sense as "generic shapes useful in some computer-aided applications" (Shah et al, 2001). In the context of a specific engineering application, features represent particular shapes or characteristics of a product or a part that are significant for that application.

A feature-based model of a part can be created either by applying the design by feature approach or by conducting Automatic Feature Recognition (AFR) techniques. In the design by feature approach, designers conceive a product model by selecting features from a set of pre-defined geometrical entities that are stored in the CAD system database. These geometrical entities have a functional meaning and may also have some manufacturing information associated with them. This approach not only assumes that designers are aware of the manufacturing constraints of a particular production environment but it also tends to limit their creativity. On the contrary, if AFR techniques are applied, semantically significant geometrical entities, that are features in a CAD model, are identified automatically in the context of specific downstream manufacturing activities.

1.2 Objectives

This thesis concentrates on the problem of automatic feature recognition from CAD models. In particular, its main objective is to develop a new AFR method that could be applied easily in different domains. The development of such a domain-independent solution is very important because the recognition capabilities of most of the existing AFR systems are limited to the requirements of specific applications (Ji and Marefat, 1997).

To achieve this overall objective, it will be required:

1. To develop a method for creating automatically rule sets that form the knowledge base of AFR systems.

2. To define an AFR method that employs such rule bases. This method should also be able to perform an efficient search for features and, at the same time, it should not be constrained to any specific application domain.

3. To build, in the proposed AFR method, capabilities for recognising interacting features that are common in engineering models.

4. To verify the recognition capabilities of the proposed AFR method by applying it on benchmarking models from a particular application domain.

1.3 Outline of the thesis

Chapter 2 starts by introducing geometric modelling techniques and the main ideas associated with the feature concept. Then, existing approaches for AFR are reviewed and those utilising rule-based, hint-based and neural network-based techniques are critically analysed.

Chapter 3 discusses the application of inductive learning techniques for creation of rule sets that could be employed for AFR utilising data stored in CAD models. In particular, a method is proposed to meet the specific requirements imposed by the utilisation of these techniques for acquisition of feature recognition rules. Then, the method is illustrated by applying it in a particular application domain.

Chapter 4 presents an AFR method that combines the ‘learning from examples’ concept with the rule-based and hint-based feature recognition approaches. This method also utilises a novel technique for automatic definition of feature hints. Then, the method is implemented in an AFR system to verify its capabilities.

Chapter 5 discusses open issues associated with the recognition of interacting features with the proposed AFR method. Solutions are suggested to improve the recognition capabilities of the method when applied on such features. Then, these solutions are implemented in the developed AFR system to verify their performance.

Chapter 6 summarises the main contributions of this research, presents the most important conclusions, and also suggests directions for further work.

Appendix A describes the architecture of the STandard for the Exchange of Product model data (STEP), which is used in this research to store CAD models.

Appendix B shows an example of a STEP file used in this research.

Appendix C gives an example of a grammar file developed in this study, which is used for generating a Java™ program that can parse STEP files.

Appendix D and E show the results of the recognition process carried out on two parts studied in Chapter 5.

Chapter 2 - Literature review

2.1 Overview

This chapter reviews the background literature relevant to the research presented in this thesis. It starts by introducing geometric modelling since a link generally exists between features and the geometry of parts. Then, definitions for the concept of a feature are discussed together with some important aspects related to the application of this concept. Finally, the research in the field of Automatic Feature Recognition (AFR) is reviewed and a particular attention is paid to the rule-based, hint-based and neural network-based approaches.

2.2 Geometric modelling

Efforts towards the development of part modelling systems for computer-aided design date from the early sixties. These efforts were driven by the need to establish a computer representation of product data following the introduction of computer controlled machine tools (Shah and Mantyla, 1995). The early CAD systems provided only 2D functions to support the engineering drawing activity. Then, it was required in the early seventies to extend these 2D systems to the third dimension to represent 3D models. Such models fall in one of the following three categories:

- *Graphical models*. These models are also called *wireframe models*. They are made of graphical primitives such as points, lines and arcs that are defined in the 3D space. However, for representing 3D solids, graphical models are deficient. For example, they are ambiguous because the representation of a solid object can have several interpretations.

- □ *Surface models.* These models were developed to enhance the representation capability of graphical models for describing complex surfaces that are very common in the automobile and aircraft industries for instance. Like graphical models, they only store geometrical data but they are more complete and less ambiguous. However, a surface model does not necessarily define a solid object as a closed volume.

- □ *Solid models.* These models describe the volume enclosed by the surfaces of a physical object. They were developed to address the deficiencies of both graphical and surface models such as the ambiguity problem and the absence of interior and exterior notions. Various representation schemes exist for solid modelling. The most common and the best understood representations are the Constructive Solid Geometry (CSG) and Boundary Representation (B-Rep) schemes.

2.2.1 Constructive solid geometry representation

This scheme describes a physical object as a Boolean expression of solid primitives. The CSG standard primitives are the parallelepiped (block), the triangular prism, the sphere, the cylinder, the cone and the torus (Hoffman, 1989). A solid is generally represented by a tree whose leaves are the solid primitives and whose nodes are the Boolean operations and rigid motions on these primitives. Figure 2.1 shows

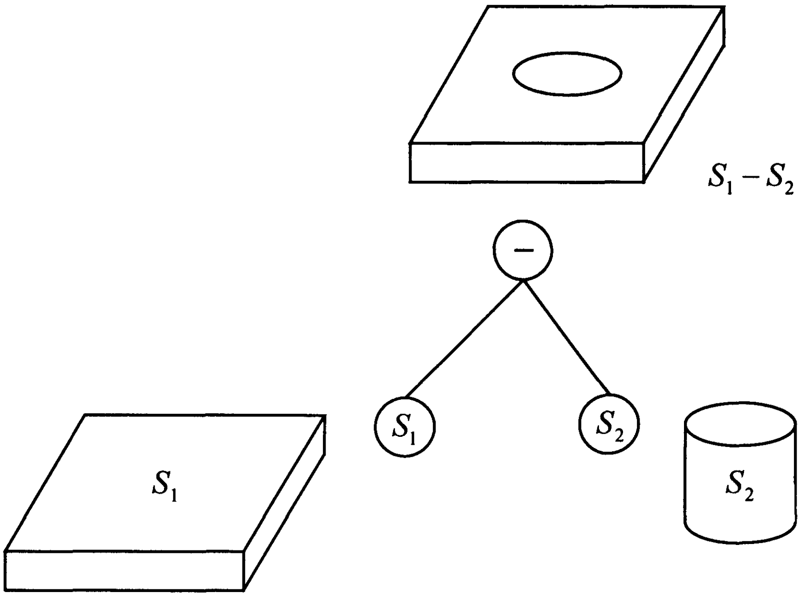


Figure 2.1 A CSG tree of a solid model

a simple example of a CSG tree. One drawback of this scheme is that a solid can generally have several CSG representations and thus it is not unique.

2.2.2 Boundary representation

A solid is represented by a set of boundary entities (faces, edges and vertices), their adjacency relationship (topology), and mathematical geometric descriptions that define the geometry associated with the boundary entities (Suh, 1995). The boundary of an object is segmented into a set of faces. Each face is described by its bounding edges and the surface on which it is embedded. Each edge is, in turn, represented by its associated vertices and the curve on which it lies. Vertices correspond to three-dimensional coordinate points. Figure 2.2 shows the boundary representation for the object shown in Figure 2.1. In comparison to CSG, the B-Rep scheme has the advantage that it is both unambiguous and unique.

2.2.3 Discussion

Due to the advantages of B-Rep models over CSG models, B-Rep has emerged as the dominant representation scheme for solid modelling. B-Rep models are also commonly used as input data for feature recognition systems. Thus, this scheme is adopted in this research to represent features and solid models of parts.

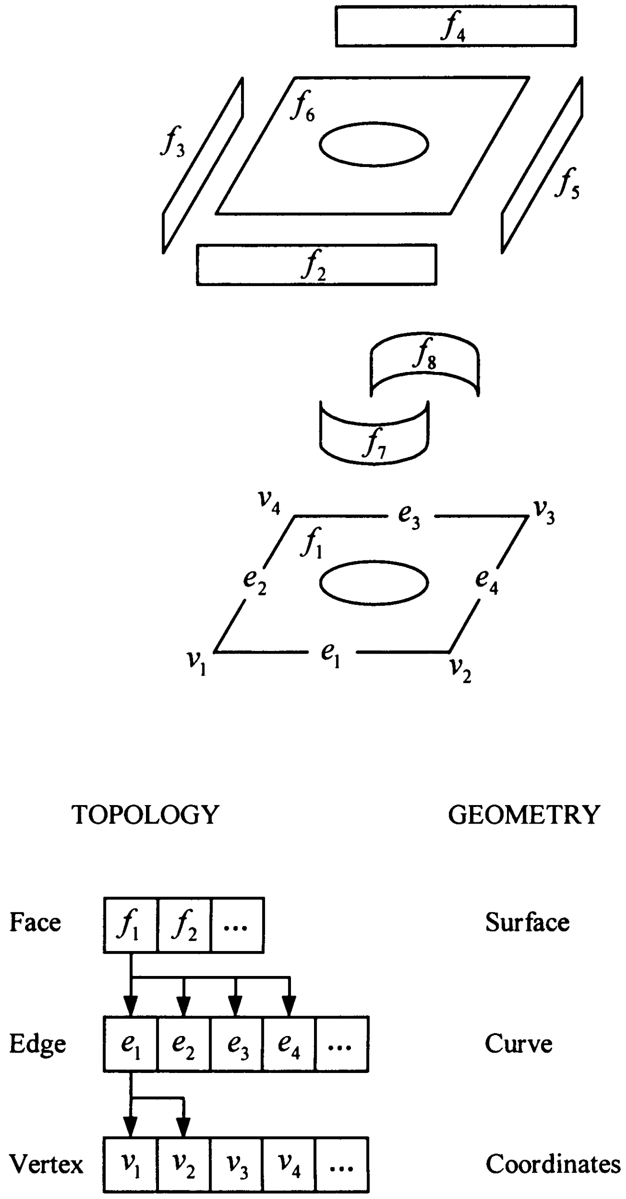


Figure 2.2 Boundary representation of a solid model

2.3 Feature concepts

2.3.1 Definitions

Shah and Mantyla (1995) define features from a cognitive point of view as "chunks of knowledge" used by engineers in performing certain tasks. Therefore, features are necessarily viewpoint dependent and application oriented. For this reason, there is no universally agreed definition for features (Ji and Marefat, 1997). However, some classification schemes consider only a subset of features, in particular those related to part geometry. It is generally accepted that such features represent "the *engineering meaning* or *significance of the geometry* of a part or assembly" (Shah and Mantyla, 1995). In general, these features can be classified as:

- Form features that describe portions of the nominal geometry of a part. This concept is used by many researchers in developing feature recognition tools such as those proposed by Sakurai and Gossard (1990), Brun (1994), Qamhiyah et al. (1996), Jha and Gurumoorthy (2000), Bhandarkar and Nagi (2000) and Ismail et al. (2002). Han (1996) further defines a form feature as "a shape macro constructed with little connection with function or manufacturing".

- Tolerance features that describe geometric variations from the nominal forms of a part (Zhang et al., 2000).

- Assembly features that describe relationships between parts in a mechanical assembly (van Hooland and Bronsvoort, 2000; Sung et al., 2001).

Shah and Mantyla (1995) propose that form features can be further classified according to the intended applications, for example:

- Design features (Suh, 1995; Han and Requicha, 1994).
- Fixturing features (Subrahmanyam, 2002).
- Manufacturing features (Vandenbrande and Requicha, 1993; Chan and Case, 1994; Regli, 1995; Chen and Lee, 1998; Gao and Shah, 1998; Cicirello and Regli, 2001; Han et al., 2001; Marquez et al., 2001; Li et al, 2003).

Manufacturing features, in particular those considered machining features, are studied by many researchers in the field of AFR. According to Regli (1995) and Ji and Marefat (1997), machining features are considered either collections of 2D patches on the boundary of a solid or 3D shapes bounded by a set of surfaces. These two different interpretations just illustrate the existing difficulties in agreeing on a common definition for features even when the application domain is well defined.

2.3.2 Taxonomies

Shah (1991) observes that although the number of conceivable features is not finite, it may be possible to categorise them into different classes that are based entirely on shape information, rather than on an application domain. For example, Wilson and Pratt (1988) propose a taxonomy based on the overall shapes of features. In particular, two feature types that can be represented by a solid modelling system are differentiated. The first type includes implicit features defined as those that do not have a detailed geometric description but whose representations contain sufficient information to derive it. The second type corresponds to explicit features for which all the geometrical details are fully defined. Also, these researchers suggest categorising explicit features based on their overall geometric form as through holes, protrusions, depressions and areas. Gindy (1989) proposes another classification framework for which form features are organised according to a hierarchy. In particular, at the top level of this classification, features are categorised in three generic groups: protrusions, depressions and surfaces.

Other researchers propose schemes that classify features according to their application domain. For example, a library of "Material Removal Shape Element Volumes" (MRSEVs) is developed by Kramer (1994) to group volumes that can be removed by machining operations on a three-axis machining centre. Another example of application-oriented taxonomy is suggested by Xu and Hinduja (1998) where features are also classified as volumes associated with different machining operations.

Another common approach is to consider only two generic feature types, protrusions and depressions, that are then further classified depending on the specific requirements of different application areas (Dong and Wozny, 1988; Han, 1996; Zhang et al., 1998; Jha and Gurumoorthy, 2000; Owodunni and Hinduja, 2002).

2.3.3 Feature interactions

An important problem that AFR systems should be able to address occurs when features in a part model do not exist independently from each other. In particular, such configurations are the consequence of feature interactions. For AFR systems, the successful recognition of each feature when they are interacting is a challenging task (Li et al., 2003).

Shah and Mantyla (1995) utilise the concept of *composite feature* to cover all the cases where it is desirable to treat a group of features as a single unit. A composite feature can be separated into two or more simple (or composite) features that can be recognised by an AFR system as features in their own right. Simple features are considered the lowest level features stored in a library and they cannot be further decomposed into other features present in this library. Two levels of relationships, recurring and non-recurring, are defined in order to describe the constraints imposed on a group of features. Recurring composite features are also referred to as *pattern features* because they are characterised by circular or linear pattern arrangements. Non-recurring composite features are made of simpler ones and are referred to as *compound features*.

Regli and Pratt (1996) notice the existence of different definitions with regard to the concept of interacting features. They argue that there is a need for a common definition that should be independent from a given feature representation scheme. In particular, they claim that it is important to establish a common conceptual framework in order to address effectively the problems associated with feature interactions. They suggest different interactions to be regarded as falling into one of the following three generic types:

- ❑ Interference interaction. This characterises an overlap between two or more features. It results in modifications affecting some of the faces that define each feature.

- ❑ Adjacency interaction. This is an interaction between features that share one or more boundary edges or faces in a part. Also, it is possible for two features belonging to different parts to be affected by such an interaction, in particular for mating features in assembly models.

- ❑ Remote interaction. This interaction does not concern any adjacencies or overlaps between features but refers to relationships that could be functional or significant to downstream applications of the design process.

2.3.4 Discussion

The main objective of this research is to develop an AFR method that could be applied in different domains. Therefore, the concept of form feature is considered the most appropriate for developing such generic tools because it does not refer to any particular application domain. In addition, the application-independent taxonomies that have been reviewed in this section could be used to define a suitable classification framework that covers all generic types of form features. Finally, it should be noted that the issues associated with the existence of feature interactions in part models require a special attention. Any AFR tools should have a built-in geometric reasoning mechanism in order to identify such interactions and their effects on simple form features.

2.4 Automatic feature recognition

2.4.1 Classification of existing approaches

Research in the field of automatic feature recognition started in the early eighties. Some classifications of the current techniques can be found in the reviews carried out by Henderson et al. (1994), Ji and Marefat (1997), Han et al. (2000) and Li et al. (2003). In general, the following categories are used to classify the existing approaches although there is sometimes an overlap between AFR techniques applied in some systems:

- Syntactic pattern recognition approach. The geometrical information about a feature is represented as an expression that defines a sequence of geometric primitives. During the recognition process, an expression based on these primitives is formed for the part studied. Then, this expression is parsed to identify its feature patterns. Most of the AFR systems applying syntactic pattern recognition techniques are developed for recognising features only in 2D shapes or 2D cross-sections of solids.

- Graph-based approach. The faces of a feature are represented by the nodes of a graph and the adjacency information between these faces is shown by the arcs connecting the nodes. Additional information can also be included into the graph such as face orientation. During the recognition process, the B-Rep model of a part is translated into a graph that is then searched using sub-graph isomorphism for matches with pre-defined feature graphs.

- Volumetric decomposition approach. First, a volume is obtained by subtracting a part from its convex hull or from its initial workpiece (stock) and then, this volume is decomposed into features.

- Rule-based approach. A set of necessary and sufficient conditions for the patterns found in features is defined. During the recognition process, these rules are applied on data stored in the solid model of a part.

- Hint-based approach. The 'generate-and-test' strategy is applied during the recognition process to form hypotheses/hints about the existence of features in a part. Then, a validation procedure based on additional geometric and topological constraints is carried out to confirm or discard the generated hints.

- Neural network-based approach. Using a set of feature examples, a neural network is trained to recognise the geometric and topological patterns that are specific for a given feature.

The following section reviews in more detail three groups of AFR techniques that are of interest to this research: the rule-based, hint-based and neural network-based approaches.

2.4.2 Rule-based approach

2.4.2.1 Description

This AFR approach was among the first to be investigated due to the success of expert systems in other application areas. Knowledge about a given domain can be represented as rules that are processed by the inference engine of an expert system in order to solve specific problems. In a similar way, information about feature patterns could be represented by rules stored in the knowledge base of an AFR system.

2.4.2.2 Methods and their applications

Henderson and Anderson (1984) define a feature as a production rule. Rules are written by determining the necessary and sufficient conditions for a given feature and by expressing them in a logic statement. For example, a rule for a simple cylindrical hole can be expressed as:

> IF a hole entrance exists,
>
> AND the face adjacent to the entrance is cylindrical,
>
> AND the face is convex,
>
> AND the next adjacent face is a plane,
>
> AND this plane is adjacent only to the cylinder,
>
> THEN the entrance face, cylindrical face and plane comprise a cylindrical hole.

This approach is implemented in a system that can recognise cavity features (holes, slots and pockets) from a B-Rep model of a part.

Hummel (1989) develops a system to classify pre-identified machining features. A feature is defined using an object-oriented language. A production rule is automatically generated for each feature by entering its definition into the system. Also, a hierarchical taxonomy is presented that starts with the description of generic features and then defines more specific ones. During the recognition process, a previously identified machining feature is first classified as a generic one and then incrementally reclassified along the hierarchy until no more rules can be activated.

Donaldson and Corney (1993) propose a system for recognising three-axis machining features. An algorithm is developed to extract potential features from a graph representation of a B-Rep part model. The set of potential features is stored in a Prolog database as predicates. Then, a set of production rules and a backward-chaining inference mechanism are used to classify and validate each potential feature.

Vosniakos and Davies (1993) develop a feature recognition framework for B-Rep models. This framework consists of two main parts. In particular, a feature definition part where features are described as Prolog predicates and a feature matching part to carry out the recognition task. Features are recognised by matching successively the feature definitions against the part description. If a match is found, a feature is recognised, otherwise the next feature definition from the knowledge base is selected and the matching process repeated. This framework is implemented only for the hole feature class.

Chan and Case (1994) integrate a solid modelling system with a rule-based system to implement a feature recognition method and a learning method. Recognition of machining features is performed by applying a set of rules on a B-Rep model representing a volume to be removed by machining operations. The authors argue that for recognising interacting machining features successfully, the system should also possess learning abilities because interactions can occur in an unpredictable way. Thus, when a particular feature cannot be recognised, the user of the system can input faces of this feature into a learning agent through a graphical interface. Once all the information required by the learning agent is provided, a new rule for the feature can be formed automatically and added to the rule base of the system.

In the work described by Dong and Vijayan (1997), the "Overall Removable Volume" (OVR) corresponding to the material to be removed from the stock is determined first. Then, the OVR is manually decomposed into "General Machining Features" (GMFs) while insuring that as much material as possible can be removed in each machine set-up. Finally, the shape of each GMF is analysed by applying a set of feature recognition rules that are embedded in an expert system.

2.4.2.3 Discussion

The rule-based AFR approach is a simple and successful method for recognising isolated and not very complex interacting features. Another important advantage is that rules can be easily understood by human experts for verification or development purposes. However, none of the techniques reported propose a formal mechanism for rule definition. Another limitation of these techniques is the exhaustive nature of the recognition process because repeated searches for features are carried out on the solid model of a part. In addition, it is difficult to define rules for all conceivable feature configurations or to expand an existing rule base while maintaining its consistency.

2.4.3 Hint-based approach

2.4.3.1 Description

This approach is also known as evidence-based (Ji and Marefat, 1997) or trace-based approach (Regli, 1995). It was introduced to tackle the problem of recognising interacting features by simulating the intuitive nature of the decisions made by humans when identifying such features. In particular, it is based on the assumption that certain feature patterns should exist in the solid model of a part in spite of the fact that some of their characteristics may be destroyed by the interactions. Therefore, such patterns could be used to generate hypotheses about the presence of features in a part model.

2.4.3.2 Methods and their applications

Vandenbrande and Requicha (1993) suggest the concept of hint for recognising machinable regions in a solid model when features interact. Hints are based on "feature presence rules" and correspond to combinations of faces that satisfy certain topological and geometrical relationships. The feature recognition process follows a generate-and-test strategy that is carried out in three steps: hint generation, feature completion and feature verification. In the first step, some production rules are executed when certain face patterns and geometrical conditions corresponding to hint definitions are detected in the B-Rep model of a part. Then, these hints are classified into three groups: promising, unpromising and rejected. In the second step, the promising hints are processed further in order to identify additional data about the potential features associated with them. For each hint, the largest feature volume that does not intrude into the part is also generated. Finally, the feature verification step checks whether the identified features are machinable.

This hint-based approach is further developed by Han (1996). Supplementary algorithms are employed to make the recognition process more robust. It is also suggested to apply Certainty Factor techniques from the field of uncertain reasoning to rank hints that could lead to valid features. In addition, the recognition process is carried out incrementally in order to be incorporated into a concurrent engineering environment.

To solve the problem of recognising depressions formed by interacting features, Marefat and Kashyap (1990) develop a method that employs a 'hypothesis generation-elimination' approach. This approach is similar to the generate-and-test strategy proposed by Vandenbrande and Requicha (1993). In particular, a graph representation called the cavity graph is suggested to describe the topology and geometry of depressions present in a B-Rep model of a part. Hypotheses are first generated by decomposing the graph of a part into cavity graphs corresponding to the patterns of the features to be recognised. These hypotheses are further processed by a rule-based system to eliminate the incorrect ones. In order to deal with interacting features, the authors introduce the concept of virtual links to augment the cavity graphs and thus, to generate additional candidate hypotheses. To determine correct virtual links, a hypothetical set of links possibly omitted from the cavity graph is first formed. Next, the Dempster-Shafer theory is applied to combine geometrical and topological evidences about each link. Then, a clustering technique is employed to add these links to the cavity graphs. Finally, new hypotheses about the presence of features are generated using these modified graphs.

This approach is further developed by Trika and Kashyap (1994) who introduce a geometric reasoning algorithm to determine the virtual links. In addition, the authors prove that all correct, and only correct, virtual links are generated and thus, the developed algorithm is both sound and complete. In other words, this means that the algorithm does not propose invalid features for a given part (soundness) and it recognises all the features present in it (completeness). However, the part domain that the system can handle is restricted to objects that do not have inclined faces.

Ji and Marefat (1995) also apply the approach proposed by Marefat and Kashyap (1990). The only difference is that the set of correct and necessary virtual links is found by exploiting Bayesian probabilistic propagations. First, a hypothesis space is constructed by obtaining a complete and minimal set of potential virtual links. This hypothesis space is further pruned to obtain a hierarchical singly connected belief network that serves as the medium for fusion and propagation of the evidences. The same authors (Marefat and Ji, 1997) further improve this approach by employing multi-connected belief networks in comparison with the previously adopted singly connected networks.

Ames (1991) introduces the concept of "featurettes" to develop a system that performs feature recognition on B-Rep models. A featurette is defined as a very low-level information about the CAD data such as a set of parallel edges or a set of faces that have similar attributes. In this way, a featurette acts like a hint that indicates the presence of a feature in a part. The recognition process follows the generate-and-test strategy. In particular, it is decomposed into small and simple steps, which are determined by a featurette hierarchy. The recognition proceeds along this hierarchy by searching and testing featurette hints until a correct feature is derived.

Regli (1995) utilises the concept of hints from Vandenbrande and Requicha (1993) but employs the term "trace" to describe it. A trace is defined as partial information produced by an instance of a feature that remains in the solid model of the part in spite of potential interactions. Also, definitions are presented for a class of volumetric features that describe material removal volumes produced by machining operations on a three-axis vertical machining centre. The basic components of his approach are a finite set of feature types and finite sets of traces. Each trace is associated with a geometric reasoning algorithm for constructing an instance of a feature from the B-Rep information of a part model and the stock material.

Gao and Shah (1998) present a hybrid approach for automatic recognition of machining features from B-rep solid models that combines graph-based and hint-based feature recognition techniques. First, a graph including different topological and geometric attributes of a part is constructed. Then, this graph is further decomposed into sub-graphs by deleting the nodes that represent either a stock face or a convex hull face. If a sub-graph does not match the graph of an isolated feature, it is assumed that it represents a group of interacting features. In this case, a graph is further decomposed in one or several Minimal Condition Sub-graphs (MCSGs). Each MCSG is considered to be a feature hint because it represents a trace left by an original feature. Next, the different MCSGs are completed in order to find the lost parts caused by feature interactions. This is achieved by generating all the virtual links of a MSCG using a geometric reasoning mechanism similar to that proposed by Trika and Kashyap (1994). Based on the classification of the virtual links, the corresponding feature for a MSCG can be retrieved.

Li et al. (2000) also propose a hybrid method based on hints, graph manipulations and an artificial neural network for recognising interacting machining features in a B-Rep model. A graph of the part studied is first constructed and some virtual links are generated following certain face conditions. The set of virtual links forms a Virtual Link Graph (VLG). The concept of F-Loop composed of a set of machining faces is introduced and considered a feature hint. Then, F-Loops Graphs (FLGs) corresponding to potential features are built based on the graph of a part and the VLG. Finally, the graph information of the FLGs is transformed into two-dimensional matrices and used as an input to a neural network that classifies the FLGs into six different types of features.

2.4.3.3 Discussion

The introduction of hint-based methods has represented a step forward in solving the problem of recognising interacting features. However, most of the proposed systems restrict the use of hints to the domain of machining features. In addition, the main difficulty in developing AFR systems based on this approach is the need to define an appropriate set of hints for each considered application domain. In particular, determining the characteristics of a hint and assessing its relevance in recognising a given feature is not a trivial task. To achieve this, system developers have to understand fully which feature patterns are still present in a part in spite of interactions. Thus, a major limitation of the current hint-based methods is that the hint definition task is always carried out manually. This could explain why the hint concept is mainly applied in the machining domain and not in other application areas as it would require an input from different experts.

2.4.4 Neural network-based approach

2.4.4.1 Description

An artificial neural network is a computational model inspired by the structure and activity of the brain. It generally consists of a number of interconnected processing elements or neurones. Information processing takes place through the interaction of the neurones, each sending excitatory or inhibitory messages to other neurones. The structure of a neural net is determined by the arrangement and the nature of the connections between the neurones. A learning algorithm governs how the strengths or weights of these connections are adjusted to achieve a desirable overall behaviour of the network. In particular, two main types of learning algorithms are distinguished, supervised and unsupervised. During the training of a network, a supervised learning algorithm adjusts the weights of the connections according to the difference between the desired and actual network outputs corresponding to a given input. An unsupervised learning algorithm does not require the desired outputs to be known. During training, only input patterns are presented to the neural network, which automatically adapts the weights of its connections to cluster the input patterns into groups with similar characteristics (Pham and Liu, 1995).

2.4.4.2 Methods and their applications

Peters (1992) identifies neural networks as promising components to support the development of a generalised feature recognition system where the user can define his/her own features. The network presented is trained to recognise four different classes of two-dimensional profiles (square, rectangle, parallelograms and slots). During the training stage, some values are extracted from each feature examples and stored in a vector that is input to the neural net. During the feature recognition stage, candidate geometric subsets corresponding to potential features are first extracted from the representation of a part. Then, these potential features are coded into vectors that are input to the network for classifying them into one of the four feature classes.

Prabhakar and Henderson (1992) also discuss the application of neural networks for AFR. They suggest constructing a network for each feature example belonging to a given library. Each network is then trained to recognise a pattern defined by rules specifying conditions for the presence of a feature. During the recognition process, topological and geometrical information is extracted from a B-Rep part model in order to construct its adjacency matrix. Each element of this matrix represents the relationships between faces of a part. Finally, this matrix is fed one row at a time to each neural net to recognise the features. This system can also tackle certain cases of feature interactions.

Hwang and Henderson (1992) propose an approach for recognising features in a B-Rep model of a part by applying a single layer perceptron network. During the training stage, the input data to the perceptron have the format of a "face score vector" composed of eight elements. Each element is a measure that takes into account geometrical information about a face, its edges, its vertices, and its adjacent faces. The network is trained with the face score vectors of feature examples. During the recognition stage, first, the score of each face in a part is assessed. Then, the inner product between a face score vector and a vector composed of the weight values obtained during training for a given feature is calculated. The output is a parameter that measures the confidence factor about the presence of a feature. If the value of the confidence factor is within a pre-defined tolerance, a feature is recognised.

Lankalappalli et al. (1997) notice that for neural networks that require a supervised learning algorithm, the training set utilised should be representative of the entire domain studied. However, for problems such as AFR, it may be difficult to include in the training set all the possible features in a given application domain. Consequently, if a particular pattern cannot be classified by a network, the training process has to be executed again. To address this issue, the authors suggest employing a self-organising neural network based on adaptive resonance theory (ART-2) to cluster similar features together without supervision. The benefit from using such a neural network is that it can create a new cluster if a pattern cannot be classified with an existing one. The scheme used to code features is similar to that suggested by Hwang and Henderson (1992). The proposed method is implemented and tested with nine different types of machining features.

Nezis and Vosniakos (1997) also present an AFR system utilising a neural network. During the training stage, examples of machining features are defined and then, for each example, topological information is extracted from its B-Rep model to construct an Attributed Adjacency Graph (AAG). A graph is further translated into a representation vector containing twenty elements that are used for training the network. During the recognition process, the AAG of a B-Rep part model is constructed and then, by using a set of heuristics, it is further broken down into sub-graphs that are considered potential features. Then, the representation vector of each sub-graph is formed and used as an input to the trained neural network to identify its corresponding feature class.

Similarly to Peters (1992), Chen and Lee (1998) identify neural networks as promising components to support the creation of generalised feature recognition systems. Users of such systems could define their own features via a graphical user interface. The training set utilised by the authors consists of shapes that are representative of six different types of two-dimensional features relevant to sheet metal manufacturing. During both the training and feature recognition stages, the input to the network is a vector that codes information about the line segments that form a two-dimensional feature.

Zulkifli and Meeran (1999) report a technique for recognising interacting features employing two different neural networks. The recognition process starts by searching a B-Rep part model for volumes that correspond to interacting features. Then, a Kohonen neural network is applied to cluster the vertices of these volumes and based on this information, the interacting features are broken down into primitive ones. Finally, data about the edges and vertices of theses primitive features is used as an input to a multilayer feedforward neural network for recognising the classes to which the primitive features belong.

Marquez et al. (2001) integrate an AFR system employing a feedforward neural network into a system that performs a manufacturability analysis on B-Rep models of reinforced plastic parts. The scheme adopted by the authors to code features is again similar to that used by Hwang and Henderson (1992) but some modifications are introduced for the face score calculation and the vector formation. They tackle the problem associated with supervised learning, highlighted by Lankalappalli et al. (1997), by creating one neural network for each feature class. In this way, although it is necessary to train every neural net to recognise a specific feature, the system can easily be expanded for recognising new features.

As mentioned in the previous section, the work on the recognition of interacting machining features in a B-Rep model presented by Li et al. (2000) combines the use of hints, graph manipulations and an artificial neural network. The same authors (Li et al., 2003) develop this approach further by employing an ART-2 network. During the training stage, eight different examples of machining features are defined and then, each of them is coded into a vector containing nine elements. The training does not require any supervision and it stops after a certain number of iterations, when the vectors used as inputs are distinguished by the network into eight different types. Then, during the recognition stage, the ART-2 neural net is utilised to classify extracted features into one of these eight categories.

2.4.4.3 Discussion

The utilisation of neural networks for AFR has attracted a significant interest in the last decade. Their learning capability is beneficial for solving AFR problems because they can be trained to recognise the characteristic patterns of a pre-defined set of feature classes. Thus, system developers do not need to design recognition procedures and also, the capability of an AFR system to recognise new or user-defined features can be easily extended. However, neural networks can only deal with numerical inputs that are not always suitable to represent geometrical and topological data stored in CAD models. In addition, during the recognition stage, a neural network acts as a 'black box' and consequently, the classification model created during the training stage is not easily interpretable by domain experts for validation or interpretation purposes.

2.5 Summary

This chapter has reviewed different 3D modelling techniques together with two commonly used representation schemes for solid modelling. The main notions associated with the concept of feature have also been discussed. A classification of existing AFR approaches has been presented and three of them, the rule-based, the hint-based and the neural network-based approaches have been analysed in detail. The main conclusion is that AFR systems implementing these three approaches could be applied only to recognise features that are domain-specific. Thus, the main knowledge gap that this research should address is the development of AFR methods that are domain independent.

Chapter 3 - Extraction of feature patterns

3.1 Introduction

The main problem addressed in this chapter is the knowledge acquisition associated with the development of rule-based systems for feature recognition. A method is proposed for automatic formation of feature recognition rules. This method employs the 'learning from examples' concept for creation of rules that define the characteristic patterns for the existence of features in CAD models. In particular, these rules are formed by applying an inductive learning algorithm on training data consisting of feature examples. Thus, the creation of a rule base for AFR systems could be automated.

This chapter starts with the definition of a feature in the context of this research. Then, basic concepts of inductive learning together with the algorithm employed in this study are presented. In the following section, the specific requirements imposed by the utilisation of this machine learning technique for acquisition of rules for feature recognition are discussed. Finally, one possible implementation of the proposed method is described and its application demonstrated on an illustrative example.

3.2 Feature definition

The proposed method for automatic formation of feature recognition rules should not be limited to a particular domain. Thus, a feature should be considered a *form feature*, a generic geometrical shape that does not relate to any specific application.

Also, it is important to determine how features could be described using Boundary Representation (B-Rep) entities because this representation scheme is employed in this research for storing geometrical and topological data about solid models. For this, the feature concept proposed by Sakurai and Gossard (1990) is adopted. A feature is defined by these researchers as a single face or a set of contiguous faces, called a *face set*. Thus, a feature composed of *m* faces is represented by the notation $\{f_1,...,f_m\}$. In addition, the B-Rep entities used to describe a feature are given specific names. In particular:

- A face belonging to a feature is called a *feature face*.
- An edge shared by two feature faces is considered an *internal edge*.
- A face adjacent to a feature face, but not included in the topological structure of a feature, is called a *boundary face*.
- An edge shared by a feature face and a boundary face is called a *boundary edge*.

3.3 Inductive learning

Inductive learning algorithms are a subset of machine learning algorithms. A common characteristic of machine learning techniques is that they identify hidden patterns in training data in order to automatically build classification models for a given application domain. The inductive learning algorithms create models that are represented as rule sets or decision trees. The rule sets include IF-THEN rules that can be readily interpreted by humans and can be used for automatic generation of rule bases for expert systems.

In this chapter, the objective is to develop a method for automatic formation of feature recognition rules. Thus, an inductive learning algorithm that forms classification models represented as rule sets is adopted. In particular, the algorithm utilised in this study is DynaSpace (Bigot, 2002). It belongs to the RULES family of inductive learning algorithms (Pham and Dimov, 1997). Like all algorithms for inductive learning, DynaSpace requires the input data to be in a specific format. In particular, the data files presented to such algorithms should contain a collection of *objects,* each belonging to one of a number of given *classes*. Each object is described by its *class value* and by a set of *attribute values* represented as a vector. Each attribute value in this vector can be either discrete or continuous. Table 3.1 gives an example of a training set that can be used for inductive learning. By applying the DynaSpace algorithm on this data set, the IF-THEN rules shown in Table 3.2 are generated.

The next section discusses the requirements to form training sets for acquisition of feature recognition rules that are suitable for inductive learning.

Object	Attr_1	Attr_2	Attr_3	Class
1	0	-1	0	1
2	1	0	0	1
3	1	-1	1	2
4	1	0	1	1
5	0	0	1	1
6	1	1	1	2
7	1	-1	0	1
8	0	-1	1	2

Table 3.1 An example of a training set

Rule	Rule description
1	IF Attr_3 = 0 THEN Class = 1
2	IF Attr_2 = -1 AND Attr_3 = 1 THEN Class = 2
3	IF Attr_2 = 0 THEN Class = 1
4	IF Attr_2 = 1 THEN Class = 2

Table 3.2 Rule set for the data in Table 3.1

3.4 Training data creation

3.4.1 Proposed approach

To generate the required training data in the context of this research, the following three steps are proposed:

- First, a taxonomy that represents the *feature classes* for a given application domain is defined. For example, the proposed method could be applied to generate rules for recognising machining features that are associated with particular manufacturing methods/machining strategies. In such a case, a taxonomy reflecting the specific requirements of this application should be adopted. Such a domain specific classification groups the data available for each feature class and guides the search for pattern recognition rules. However, the proposed method should not be limited to any particular application. For this reason, only the most generic part of a feature taxonomy that could be considered application-independent is discussed in this research. In this context, the top-level classification of features into protrusions, depressions or surfaces proposed by Gindy (1989) is adopted in this study because it satisfies this requirement.

- Second, a set of B-Rep models representing examples of features is designed for each class of a given taxonomy. For example, in the machining domain, a feature class could cover all B-Rep models for slot features. A systematic approach is adopted in designing the B-Rep models of features. In particular, this approach is applied to balance the representation of the different feature classes in the training set. As a result, the weight of all feature classes during the induction process will be the same. The systematic approach adopted in this research for designing the B-Rep models of features is described in more detail in Section 3.5.1.1 and an example to illustrate it is provided.

- Third, the B-Rep models of features are converted into data files that are suitable for inductive learning. This conversion is necessary because these models cannot be used for inductive learning directly. In particular, the input data for such algorithms should be in a specific format. The next section discusses the requirements for converting the B-Rep models of features into training sets that are suitable for inductive learning.

3.4.2 Data format

The data files for inductive learning should contain a collection of objects. In this research, these objects are called *characteristic vectors*. Each characteristic vector is composed of attributes that store information about a given B-Rep feature model. Thus, each vector also belongs to the feature class of the considered model. The feature classes determine the taxonomy that is applied to classify features in a particular application domain. Thus, to classify all characteristic vectors belonging to a given class, a coding scheme for storing the information contained in B-Rep feature models should be implemented very carefully.

In general, the definition of a coding/representation scheme to encapsulate meaningful data about a specific engineering domain is not a trivial task. It is beneficial that such a scheme includes as many attributes as possible (Nezis and Vosniakos, 1997). However, if a representation scheme includes irrelevant attributes, this would have a detrimental effect on the algorithm classification performance (Liu and Motoda, 1998). One possible solution could be to rely on domain experts in identifying attributes that should be considered (Liu and Motoda, 1998).

In the field of AFR, a systematic approach for defining a representation scheme that codes feature information from B-Rep models does not exist. Therefore, in this research the following general guidelines are adopted in designing such a scheme:

- The specific characteristics of the application domain should be taken into account in deciding which attributes should be selected for inclusion in a characteristic vector. For example, if machining features are considered, the characteristic vector should be composed of attributes that represent geometrical forms associated with removal volumes.

- The attributes included in a characteristic vector should provide sufficient information to solve a recognition task at a particular level of abstraction. For example, if an AFR system needs to reason about local properties of features, the characteristic vector should include attributes that represent low-level feature information such as data associated with a feature face.

- A characteristic vector should include as many attributes as possible whilst avoiding the inclusion of misleading attributes. In this research, an example of such an attribute is 'the date at which a feature has been created'. Such information is obviously not relevant to feature recognition tasks. However, this does not necessarily mean dismissing attributes that are considered irrelevant by a domain expert because an interesting aspect of inductive learning techniques is their capability to generalise and create rules that are not ‘obvious’ at first for experts.

Thus, to design a representation scheme that is suitable for solving a particular feature recognition task, it would be useful to consider initially a broad range of attributes. Therefore, in the next section, different types of attributes that could be used to code B-Rep models of features are identified in order to decide what scheme to be adopted in this research.

3.4.3 Feature attributes

Most of the representation schemes for coding B-Rep feature models are developed for creating training data for neural network-based AFR systems. Thus, these schemes are a valuable source of information about attributes that are important in AFR. Other coding schemes are implemented in the feature recognition techniques presented by Yuen and Venuvinod (1999) and Dereli and Filiz (2002). These techniques rely on such schemes for comparing previously extracted groups of faces in a part against a finite feature database where all the features classes of interest are listed along with their feature codes.

The study of these different schemes shows that the attributes considered for coding B-Rep models of features are very diverse and represent information about B-Rep entities at different levels of abstraction. These could be a face set, a single face, a loop of edges, a single edge or a vertex. In addition, according to Yuen and

Venuvinod (1999), the data associated with these B-Rep entities falls into the following categories:

- Topological data. This high-level data provides information about the adjacency of faces, edges and vertices that compose a feature.

- Coarse geometrical data. This category describes data such as the geometric form of a feature face and the concavity or convexity of its edges.

- Fine geometrical data. This low-level data includes information such as dimensions, analytic geometric equations and angular orientations.

Table 3.3 attempts to classify attributes that can be derived from or attached to B-Rep data and hence utilised for defining a representation scheme for coding features. In this table, different B-Rep entities are categorised into five levels of abstraction to help in designing an appropriate coding scheme for inductive learning. The following section describes the approach adopted in this research for designing such a scheme for extraction of feature recognition rules.

3.4.4 Proposed strategy for feature coding

In this research, two levels of abstraction are considered in identifying the feature patterns that are representative for a given application domain. Thus, two sets of training data are formed in order to extract two different sets of rules. The first level of abstraction considers feature faces as single entities. Then, at the second level, the face set that defines a feature is analysed.

	Topological data	Coarse geometrical data	Fine geometrical data
Face set	• Number of faces that compose a feature • Number of faces that bound a feature	• Parallelism • Perpendicularity • Coaxiality • Number of concave or convex edges	
Single face	• Number of adjacent faces	• Face type (geometry) • Face concavity • Number of concave or convex edges • Number of loops	• Face area • Surface representation
Loop	• Number of edges	• Loop concavity • Inner or outer loop	
Single edge		• Edge geometry (curve) • Edge concavity	• Curve representation • Edge length • Angular value
Vertex	• Number of incident edges		• Coordinate points

Table 3.3 Feature attributes

In the first training set, each characteristic vector stores information about a single feature face. Thus, several vectors are extracted from each B-Rep feature model. The attributes belonging to single faces (see Table 3.3) are of a particular interest in designing a coding scheme at this level of abstraction. The rules generated from such training data are referred to as the *first set of rules*. They define feature patterns that are extracted from partial representations of a feature. Such rule bases could be applied in AFR systems that search for patterns suggesting the existence of a feature, such as hint-based systems.

However, such a coding scheme has some limitations. In particular, it is difficult to identify a set of attributes that does not lead to code duplications, i.e. characteristic vectors that are completely identical, having the same values for all their attributes and, at the same time, belonging to different feature classes. In the machine learning domain, such code duplications are called noise. Thus, such vectors in the training set should be avoided because they introduce ambiguity and could prevent inductive learning algorithms from generating valid rules for some feature classes. This coding approach is criticised by other researchers, i.e. Nezis and Vosniakos (1997). In particular, they claim that it is difficult to solve the code duplication problem even by introducing more complex coding schemes for single faces.

Another problem with such a coding scheme is associated with the intrinsic nature of a feature. In a B-Rep model, any individual feature face represents a low level of information about a feature. Although a feature has been defined previously as a single face or a set of contiguous faces, it is generally the case that a feature is made of more than one face. Thus, this low-level representation scheme should be complemented by another scheme of a higher level of abstraction. The training data generated at this second level of abstraction should contain characteristic vectors that include information about the face set defining a feature. Thus, in this second training set, only one vector is created for each B-Rep feature model. In this research, the rules generated from such training data are referred to as the *second set of rules*. The attributes associated with a face set should be used to design this coding scheme (see Table 3.3).

3.5 Illustrative example

This section discusses a possible implementation of the proposed method for automatic formation of feature recognition rules that represent patterns identified in B-Rep feature models. First, the feature taxonomy that is adopted in this implementation is outlined and then, two representation schemes are described for coding feature information at the defined two levels of abstraction. The DynaSpace inductive learning algorithm is then applied on the training data created using theses two representation schemes to generate two sets of rules. Finally, some issues associated with the implementation of this method are discussed.

3.5.1 Training data creation

3.5.1.1 Feature taxonomy

In this implementation, the adopted taxonomy categorises features belonging to the machining domain (see Table 3.4). It is inspired by the classification of machining features proposed by Pham and Dimov (1998). The top level of this taxonomy groups machining features into two generic types, depression or protrusion, that are

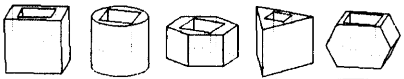
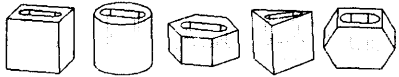
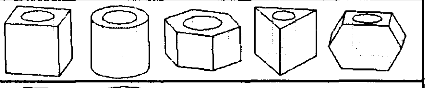
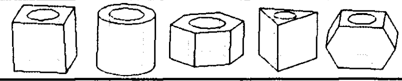
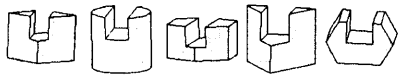
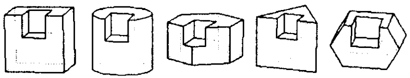
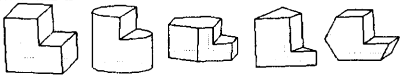
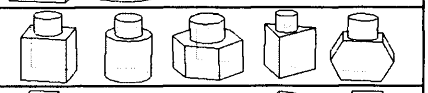
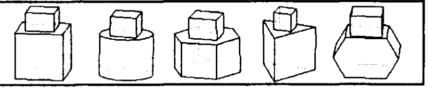
Generic feature type	Feature group	Feature class	Symbol	B-Rep feature models
Depression	Pocket	Rectangular	po_re	
		Obround	po_ob	
	Hole	Blind hole	ho_bl	
		Through hole	ho_th	
	Slot	Through slot	sl_th	
		Non-through slot	sl_nt	
	Step	Step	st	
Protrusion	Protrusion	Circular	pr_ci	
		Rectangular	pr_re	

Table 3.4 Taxonomy of machining features

considered application-independent. The second level clusters the features into five sub-groups that correspond to the machining strategies that should be employed for their manufacture. Hence, this intermediate level is not anymore application-independent. The third level defines the feature classes depending on their geometrical profiles. The B-Rep feature models considered in this study are also represented in Table 3.4. They are constructed using a solid modelling system and thus, they define closed volumes. However, a single face or a face set forming a feature does not necessarily define a closed volume. For this reason, a B-Rep feature model is composed of feature faces and also faces that belong to a *base protrusion.*

A systematic approach is implemented to construct the B-Rep feature models shown in Table 3.4 applying the following three guiding principles:

- First, to balance the importance of each feature class in the training set, the same number of models is created for each class. The importance of this was already highlighted in Section 3.4.1.

- Second, the same base protrusions are used for each class in order to minimise any influence that they could have on the inductive learning process. Thus, if a given base protrusion is utilised for creating a model for one feature class, it is also utilised for the other classes.

- Third, these base protrusions are different in order to vary the topological and geometrical neighbouring configurations of features. This is required because the number and the type of boundary edges of a feature could differ.

For simplicity, only planar and cylindrical faces are used to construct the feature models shown in the table. However, the proposed method is not restricted to these types of geometric entities only. Also, it should be noted that not all faces included in a B-Rep feature model have to be considered in designing a coding scheme. For example, the faces defining the base protrusions in these models should not be taken into account during the coding because they will not provide additional information for the feature recognition process. Even, the vectors created for these faces could introduce a noise in the training sets. That is why the relevant faces in the B-Rep models are selected by end-users to form the characteristic vectors for each feature. The process of designing a feature coding scheme at both levels of abstraction is discussed in the next section.

3.5.1.2 Feature coding schemes

Characteristic vector for a feature face

At this level of abstraction, information about individual feature faces (see Table 3.3) is used to define a coding scheme. In particular, topological and coarse geometrical data about a single face are utilised in designing this scheme. In this example, the following attributes are considered to be of importance during the feature recognition process:

- Attribute 1 (faceTy): the face type. This attribute describes the geometry of the surface defining a face. As mentioned earlier, the face types considered in this example are either planar or cylindrical.

- Attribute 2 (nEd): the number of edges defining the face border.

- Attribute 3 (faceCv): the face convexity. There are three possible values for this attribute, neutral (i.e. planar), concave or convex. According to the definition given by Marquez et al. (2001), a face is convex if a straight line between two points on the face is enclosed inside a solid model, otherwise it is concave. This definition is extended to cover also the case when a straight line between two points on a face lies completely on it. Such a face is considered neutral.

- Attributes 4 & 5 (nPlAd & nClAd): the number of adjacent faces that are planar and cylindrical, respectively.

- Attributes 6 to 9 (nCcEd, nCvEd, nSccEd and nScvEd): the number of edges of a face that fall into one of the following four edge categories (Sandiford and Hinduja, 2001). An edge shared by faces *a* and *b* is considered:
  - Concave, if the solid angle between faces *a* and *b* is between 180° and 360°;
  - Convex, if the solid angle between faces *a* and *b* is between 0 and 180°;
  - Smooth concave, if faces *a* and *b* are tangential to each other, and if both are concave or one is concave and the other is neutral;
  - Smooth convex, if faces *a* and *b* are tangential to each other, and if both are convex or one is convex and the other neutral.

  Figure 3.1 illustrates these four cases.

- Attribute 10 (ccAd): a measure obtained by dividing the number of concave edges of the adjacent faces by the total number of such faces. This attribute was added to the list of attributes because the other nine attributes were not sufficient to represent all different types of faces in the training data with unique characteristic

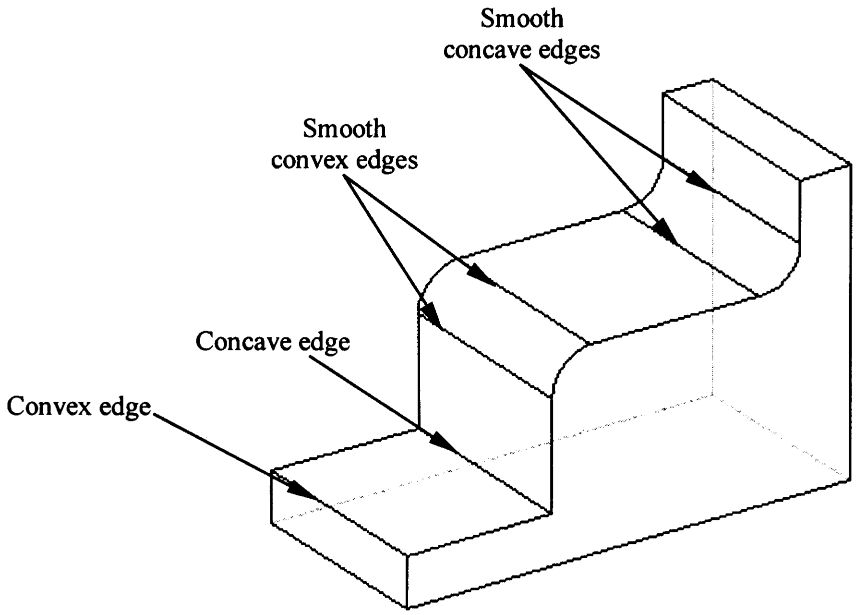


Figure 3.1 Edge categories

vectors. Thus, this attribute was included in the list to bring additional information about geometrical and topological characteristics of the feature faces.

Other attributes could also be included in the representation scheme at this level of abstraction to satisfy the specific requirements of any particular application. It is not necessary to restrict the number of attributes that are considered initially for a given application as long as they provide additional information to distinguish one face from another. The inductive learning algorithms could be used to assess the 'information content' of each attribute and after a few induction cycles, the most important of them for a given feature recognition task could be selected and in this way, the total number of attributes reduced.

Characteristic vector for a face set

Attributes belonging to a face set defining a feature are utilised to design a coding scheme at this level of abstraction. The attributes that should be considered in designing such a scheme are listed in Table 3.3. These include topological and coarse geometrical data used to define a given feature. The following attributes are identified to be of importance in distinguishing one feature from another at this level of abstraction:

- □ Attribute 1 (nFa): the number of feature faces.

- □ Attribute 2 (nPlFa): the number of planar feature faces.

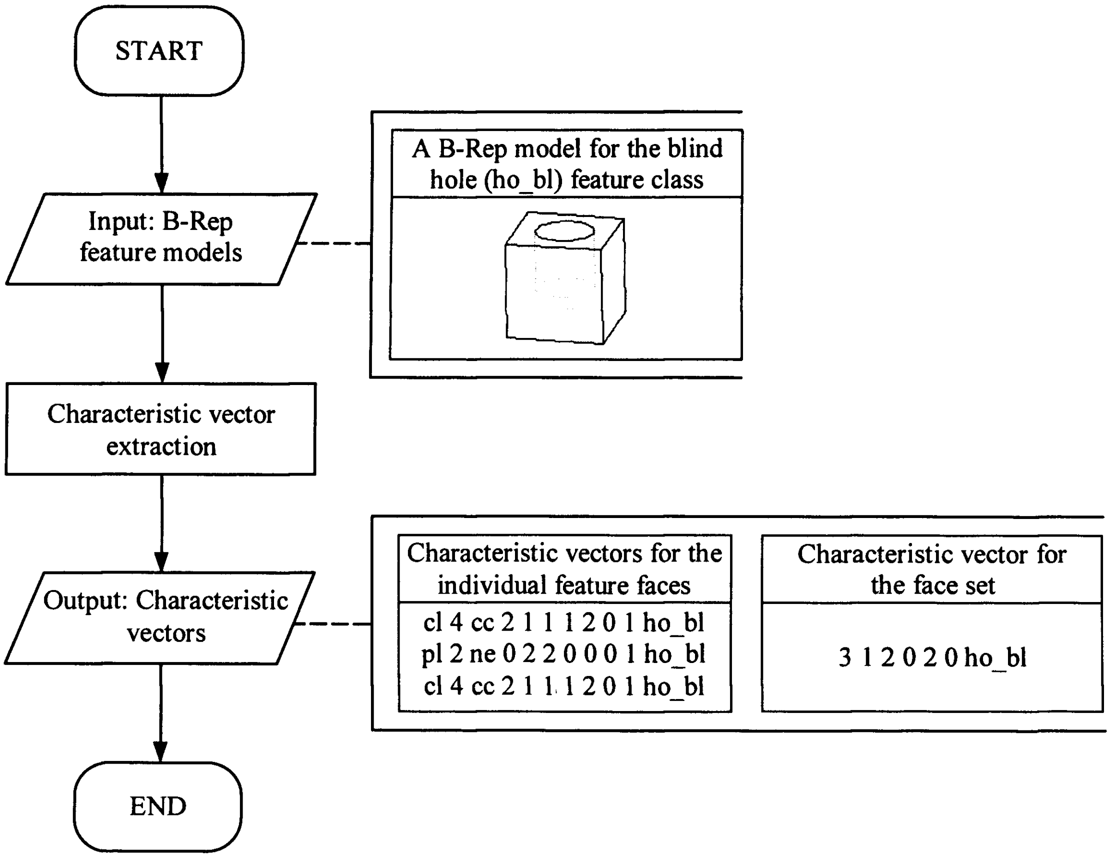
- Attribute 3 & 4 (nCcClFa & nCvClFa): the number of cylindrical feature faces that are concave and convex, respectively.

- Attribute 5 & 6 (nCcEd & nCvEd): the number of concave and convex internal edges.

Again, other attributes could also be considered if required in designing a coding scheme at this level of abstraction.

3.5.1.3 Extraction of characteristic vectors

Each of the 45 B-Rep feature models in Table 3.4 was analysed using an automated procedure to extract the attribute values included in the characteristic vectors at both levels of abstraction. Figure 3.2 illustrates this procedure and shows the resulting vectors for a model belonging to the blind hole feature class (ho_bl). In total, 160 characteristic vectors were created when individual feature faces were considered. The encoding at the second level of abstraction resulted in 45 vectors. For both cases, the characteristic vectors formed from the B-Rep models are stored in a text file for further processing by the DynaSpace inductive learning algorithm. In this way, two different training sets are created. Before the algorithm is executed on this data, it is pre-processed to eliminate the noisy vectors. In particular, this pre-processing removes vectors that are identical but, at the same time, represent features belonging to different classes. In this example, the attributes considered proved to be sufficient to create unique vectors for each feature class at both levels of abstraction.


START
Input: B-Rep feature models
A B-Rep model for the blind hole (ho_bl) feature class
Characteristic vector extraction
Output: Characteristic vectors
END

Characteristic vectors for the individual feature faces	Characteristic vector for the face set
cl 4 cc 2 1 1 1 2 0 1 ho_bl pl 2 ne 0 2 2 0 0 0 1 ho_bl cl 4 cc 2 1 1 1 2 0 1 ho_bl	3 1 2 0 2 0 ho_bl

Figure 3.2 Extraction of characteristic vectors

3.5.2 Rule formation

The DynaSpace algorithm adopted in this research is applied successively on both training sets to extract rules that depict feature patterns at both levels of abstraction. In particular, DynaSpace created 15 rules from the first training set that encapsulates information about individual feature faces. This set of rules is shown in Table 3.5. For example, Rule 6 in this set is:

IF nEd = 2 **AND** nCcEd = 2 **THEN** featureClass = ho_bl

This means that a face with two edges that are both concave indicates the existence of a feature face belonging to a blind hole in a B-Rep model.

The application of the same algorithm on the second training set that includes vectors representing face sets resulted in 9 rules (see Table 3.6). For example, Rule 4 in this set states:

IF nPlFa = 1 **AND** nCcEd = 2 **THEN** featureClass = ho_bl

This means that a face set with one planar face and two concave edges represents a blind hole feature in a B-Rep model.

Rule	Rule description
1	**IF** ccAd=3 **THEN** featureClass = po_re
2	**IF** 2<=nPlAd<=4 **AND** nCvEd=1 **AND** nSccEd=2 **AND** ccAd=2 **THEN** featureClass = po_ob
3	**IF** nClAd=2 **AND** nCcEd=4 **THEN** featureClass = po_ob
4	**IF** faceCv=cc **AND** nCcEd=0 **THEN** featureClass = ho_th
5	**IF** faceCv=cc **AND** nPlAd=2 **AND** nCvEd=1 **THEN** featureClass = ho_bl
6	**IF** nEd=2 **AND** nCcEd=2 **THEN** featureClass = ho_bl
7	**IF** faceCv=cv **THEN** featureClass = pr_ci
8	**IF** nEd=2 **AND** nCcEd=0 **THEN** featureClass = pr_ci
9	**IF** nCcEd=1 **AND** nCvEd=3 **AND** ccAd=2 **THEN** featureClass = pr_re
10	**IF** nClAd=0 **AND** nCcEd=0 **AND** nSccEd=0 **THEN** featureClass = pr_re
11	**IF** 2<=nCcEd<=3 **AND** ccAd=2 **THEN** featureClass = sl_nt
12	**IF** nEd=5 **AND** nCvEd=2 **THEN** featureClass = sl_nt
13	**IF** 5<=nEd<=6 **AND** 3<=nCvEd<=4 **THEN** featureClass = sl_th
14	**IF** faceCv=ne **AND** 2<=nPlAd<=4 **AND** 1<=nCcEd<=2 **AND** ccAd=1 **THEN** featureClass = sl_th
15	**IF** faceTy=pl **AND** 1<=nPlAd<=4 **AND** ccAd=0 **THEN** featureClass = st

Table 3.5 First set of rules

Rule	**Rule description**
1	**IF** nCcEd=8 **THEN** featureClass = po_re
2	**IF** nPlFa=3 **AND** nCcClFa=2 **THEN** featureClass = po_ob
3	**IF** nFa = 2 **AND** nCcClFa=2 **THEN** featureClass = ho_th
4	**IF** nPlFa=1 **AND** nCcEd=2 **THEN** featureClass = ho_bl
5	**IF** nCvClFa=2 **THEN** featureClass = pr_ci
6	**IF** nPlFa=5 **AND** nCcEd=0 **THEN** featureClass = pr_re
7	**IF** nPlFa=4 **THEN** featureClass = sl_nt
8	**IF** nPlFa=3 **AND** nCcClFa=0 **THEN** featureClass = sl_th
9	**IF** nPlFa=2 **THEN** featureClass = st

Table 3.6 Second set of rules

3.5.3 Implementation approach

3.5.3.1 B-Rep data formation

The feature models in Table 3.4 were designed using the Pro/Engineer™ CAD system (PTC, 2001). Because the Pro/Engineer™ native data could not be used directly to form the required training sets, each feature model was then exported into a STEP file. The architecture of the STEP standard is briefly explained in Appendix A. The Application Protocol 203 (AP 203) developed in this standard for the exchange of mechanical parts and assembly data was used in this research to generate STEP files. A benefit from exporting the feature models designed with Pro/Engineer™ into STEP files is that B-Rep data could then be extracted from such files. Another benefit is that AP 203 is supported by most commercially available CAD packages and thus, the feature models could be created using other CAD packages and not only Pro/Engineer™.

3.5.3.2 B-Rep data processing

An example of a STEP file generated using Pro/Engineer™ for a blind hole feature model is shown in Appendix B. Such a file should be processed further in order to extract the data required for forming the characteristic vectors. This processing was carried out automatically by the successive application of different parsers, one for each B-Rep entity of interest. These parsers were created utilising the Java Compiler Compiler™ (JavaCC, 2003) tool. JavaCC™ is a parser generator that converts a given grammar specification into a Java™ program in order to recognise structures/definitions satisfying that grammar. In particular, the JavaCC™ grammar specification for STEP physical files presented by Ma (2003) was extended to develop the different parsers. An example of such an extended grammar file developed for this research is given in Appendix C. When a parser progresses through the STEP file of a feature model and finds a B-Rep entity of interest, Java™ procedures specially implemented to extract information about this entity are triggered. Thus, these procedures automatically generated the characteristic vectors associated with each feature model.

3.6 Summary

This chapter has described an original method for automatic formation of feature recognition rules by applying inductive learning techniques on feature examples. Furthermore, the utilisation of geometrical and topological data at two levels of abstraction has been proposed to generate a comprehensive rule base for feature recognition. These rules define feature patterns either for individual faces or face sets for each considered feature class. In addition, a possible implementation of this method has been presented that includes specially developed procedures for automatic extraction of characteristic vectors from B-Rep feature models.

The method developed is generic as it could be employed to generate feature recognition rules for different application domains. The performance of AFR systems that utilise the proposed method depends on two factors. The first factor is the capability of the designed representation scheme to encapsulate relevant information about features in a given domain. More specifically, the number and information content of the attributes used to define the characteristic vectors at the selected levels of abstraction influence their representation power. The second factor is the adopted feature taxonomy and the feature models employed to create the necessary training data. These two aspects determine the quality of the data used to represent a particular domain. Thus, it is very important to verify the feature recognition capabilities of the generated rule sets. This issue together with different strategies for carrying out feature recognition are discussed in the next chapter.

Chapter 4 - A hybrid feature recognition method

4.1 Introduction

The previous chapter introduced a method for generating feature recognition rules at two levels of abstraction. The rules are extracted from a training set containing B-Rep models of feature examples by applying inductive learning techniques. This chapter presents an AFR method that benefits from the rule extraction techniques described in Chapter 3. The method provides a formal reasoning mechanism for feature recognition by combining the 'learning from examples' concept with the rule-based and hint-based feature recognition approaches. In particular, the generate-and-test strategy applied in the hint-based approaches is employed to simplify and speed up the search for features in B-Rep part models. This addresses one of the main limitations of rule-based techniques, the need to carry out a computationally expensive exhaustive search for features. Thus, the proposed AFR method combines the advantages offered by inductive and deductive techniques and therefore is called hybrid.

To apply efficiently this method in different application domains, the main difficulty is associated with the acquisition/definition of hints. Traditional techniques for hint definition rely on inputs from system developers. These techniques imply a good understanding of the patterns that indicate the existence of a specific feature in a CAD model. Thus, new techniques are required to automate the definition of feature hints.

This chapter starts with an overview of the proposed feature recognition method. Then, a technique to automate the definition of feature hints is presented. Finally, a procedure for recognising features in B-Rep part models is discussed and its application is demonstrated on an illustrative example.

4.2 Overview

The proposed AFR method includes two main processing stages, learning and feature recognition. During the learning stage, rules and feature hints are extracted from training data. Then, these hints and rule bases are utilised in the feature recognition stage to analyse B-Rep part models and identify their feature-based internal structure. In this section, a brief overview of these two processing stages is provided. Then, the hint definition and feature recognition processes are discussed in detail in Sections 4.3 and 4.4, respectively.

4.2.1 Learning process

The main process is composed of three consecutive sub-processes (Figure 4.1):

- Training data creation. This sub-process was described in the previous chapter. It includes the design of B-Rep feature models in accordance to a given feature taxonomy and then the extraction of characteristic vectors from each of them. As a result of this process, two different data sets are created that represent features at two levels of abstraction.

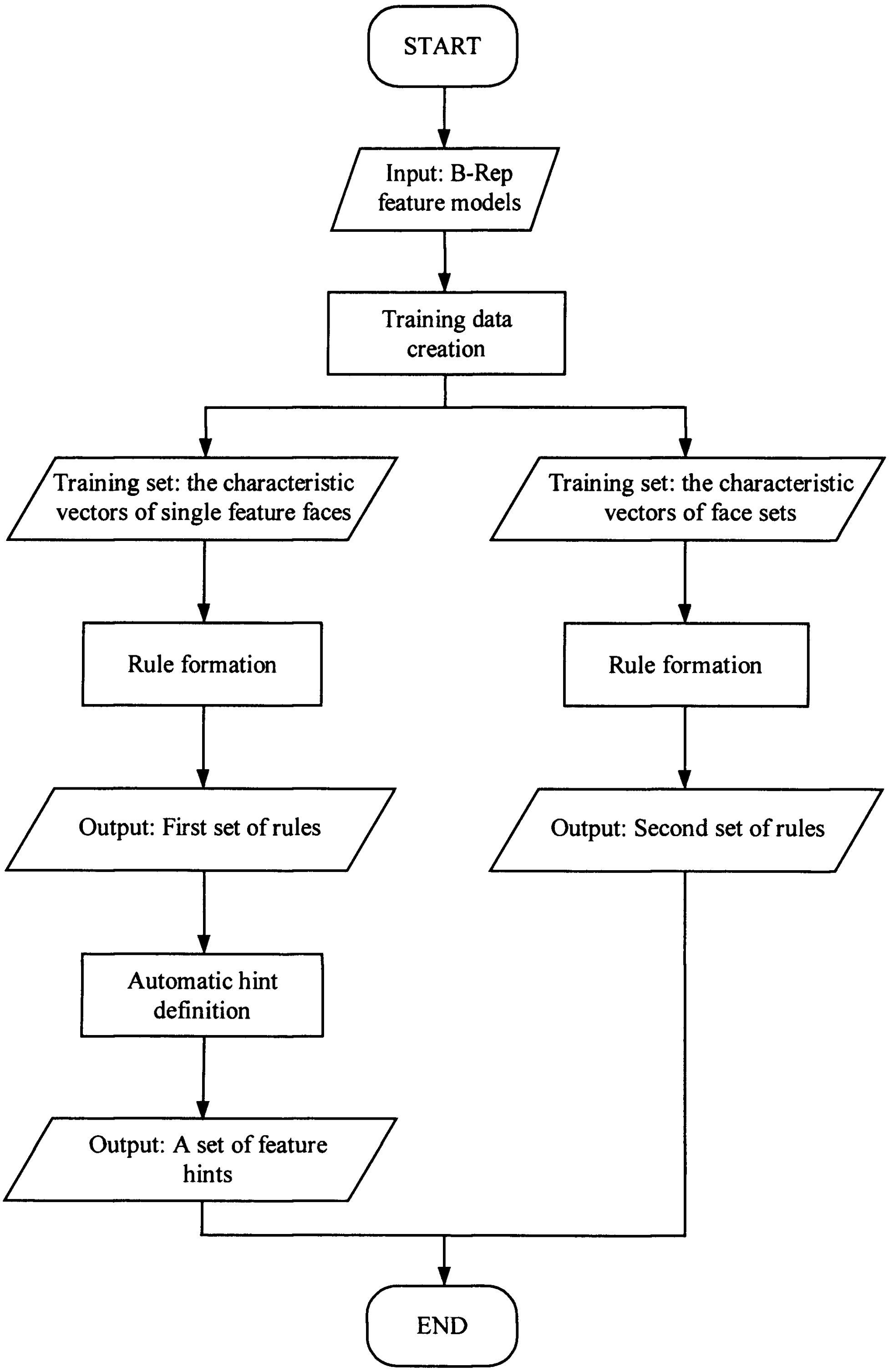

START
Input: B-Rep feature models
Training data creation
Training set: the characteristic vectors of single feature faces
Training set: the characteristic vectors of face sets
Rule formation
Rule formation
Output: First set of rules
Output: Second set of rules
Automatic hint definition
Output: A set of feature hints
END

Figure 4.1 Learning process

- Rule formation. This sub-process was also described in the previous chapter. The DynaSpace algorithm is applied on each of the two training sets of characteristic vectors to generate two sets of rules. These two rule bases define feature patterns found in the B-Rep models of feature examples.

- Automatic hint definition. In this research, it is proposed to extract the hints from the rules generated for every feature class present in a given taxonomy. Conceptually, a hint is a suggestion that a specific feature is present in a part model and also, it is an incomplete representation of a feature from an implementation point of view (Han, 1996). Therefore, the rules that define feature patterns at the first level of abstraction (partial geometrical representation of features) are utilised to define the hints.

It would be also possible to define feature hints by applying the same approach on the second set of rules that represents patterns based on geometrical or topological relations between feature faces. For example, a parallelism between a pair of planar opposing faces could be used as a hint for a slot feature as suggested by Vandenbrande and Requicha (1993). However, in this research, the focus is on identifying hints that utilise data belonging to individual faces. The feature hints defined in this way are computationally less expensive to apply because it is required to analyse only the geometrical or topological properties of single faces. This is very important taking into account that the hints are utilised for a quick search for faces indicating the existence of specific features in B-Rep part models.

4.2.2 Feature recognition process

The results of the learning process are two sets of rules and a set of feature hints. These rules together with the hints are employed in this research to recognise features in B-Rep part models. In particular, the proposed feature recognition method 'reconstructs' features in stages relying initially on an indicative information in the form of a hint. This could be interpreted as a process of 'entity growing'. Shah (1991) first proposed the entity growing terminology for feature recognition. In particular, it is described as a process in which, once a feature has been recognised it is removed from a part by adding or subtracting a volumetric shape that corresponds to this feature.

In this research, the idea of entity growing is defined differently. An analogy is made between the feature recognition process and a simplified process of vegetal plant growing. The structure of a vegetal plant could be described as a stem with foliage attached to it. Accordingly, the simplified plant growth process should include the following two steps. First, the development of a seed into a stem and then the growth of the leaves from this stem. Similarly to this, the feature recognition process includes the following four sub-processes (Figure 4.2):

□ Seed detection. Individual faces matching the definition of feature hints are detected. A face identified in this way is an incomplete representation of a feature and it is considered only a *seed* from which a feature might be constructed.

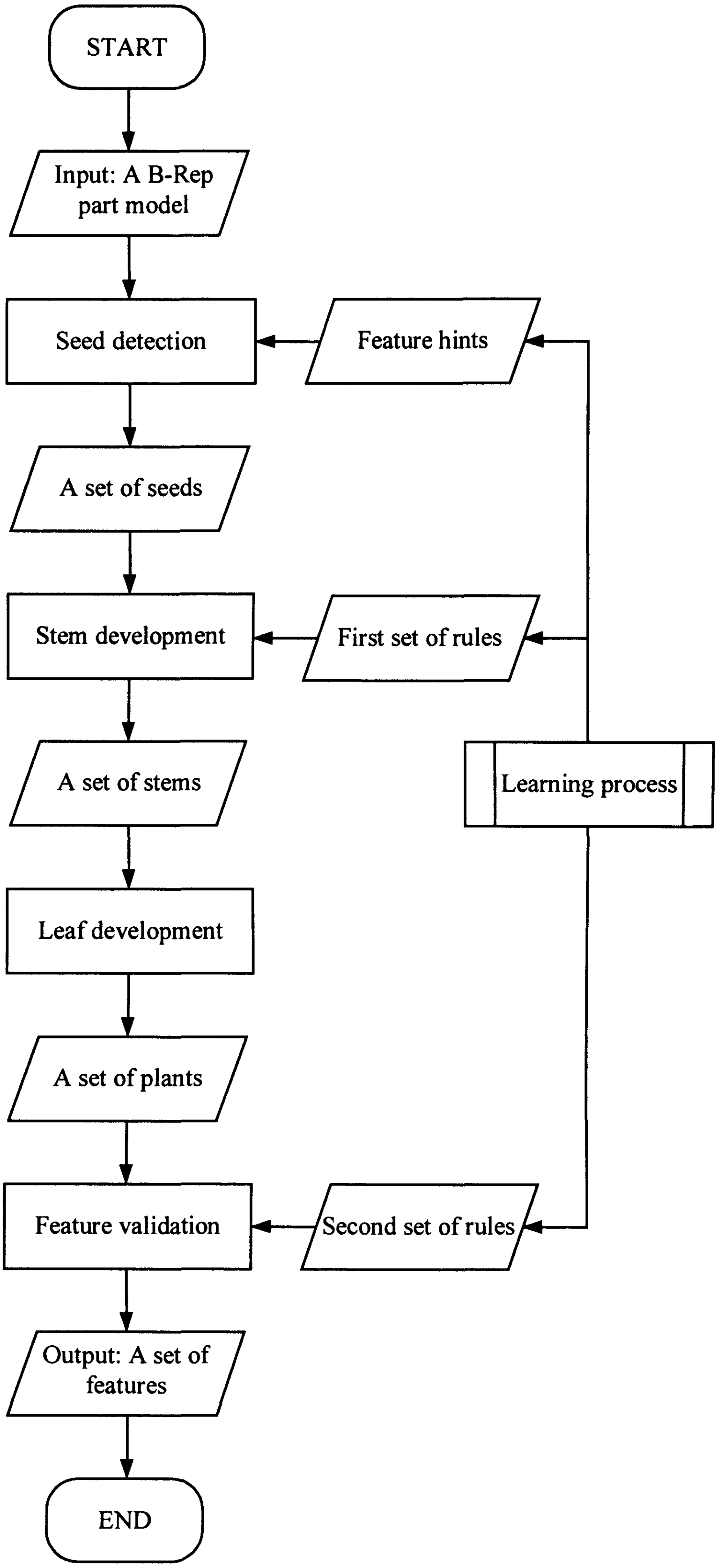


Figure 4.2 Feature recognition process

- Stem development. A seed constitutes only a hypothesis for the presence of a feature in a B-Rep part model. Such hypotheses have to be analysed further in order to validate them. The first set of rules generated during the learning stage is employed to carry out this validation because it includes patterns associated with individual feature faces. A seed that satisfies the geometrical and topological constraints defined by the rules for a particular feature class is considered one of the possible *stems* for that class. These stems constitute the starting point from which the search continues for other faces in order to complete the reconstruction of a given feature. Also, it should be noted that a stem again represents only a hypothesis for the existence of a feature in a B-Rep model. This is due to the fact that a stem corresponds to an individual face in a part and as such, to an incomplete representation of a feature. However, any hypothesis associated with a stem has a higher probability to result in a successful reconstruction of a feature in comparison with a hypothesis represented by a seed.

- Leaf development. This sub-process is analogous to the development of the foliage from a vegetal plant stem. In particular, the faces surrounding a face labelled as a stem are analysed to decide whether they could be aggregated together to form a face set representing a potential feature. A face that is selected in this way is called a *leaf*. The output of this sub-process is a face set, composed of faces labelled either as stems or leaves, that is called a *plant*.

- Feature validation. A plant needs to be checked against the second set of rules to verify whether it represents a valid feature. This rule set is employed because its rules represent geometrical and topological relations between faces in valid features. This validation is necessary because a plant is still only a hypothesis, a potential feature generated from a stem. If the output of this process is positive, a feature is recognised.

4.3 Automatic hint definition

4.3.1 Motivation

Several feature recognition strategies could be implemented by applying the two rule sets formed during the learning process. For example, one strategy could employ the first set of rules only. In this case, the rules would be applied to individual faces in a B-Rep part model to group them in clusters of adjacent faces that could form a feature. The feature patterns defined in this rule set represent the geometrical and topological properties of single faces, which are low-level geometrical entities. Thus, it is possible for a face, which is not a 'building block' of a feature but whose properties are similar to those of a feature face, to match some of these patterns. This suggests that the sole reliance on the first set of rules does not provide an adequate solution to most feature recognition tasks.

Another possible feature recognition strategy could rely only on the second set of rules. In this case, an exhaustive search for features would be performed by applying these rules to all possible groupings of faces in a B-Rep model of a part. Unfortunately, the number of such groupings in a part composed of n faces increases exponentially with the increase of n and is equal to $2^n - 1$ (Owodunni and Hinduja, 2002). The set of possible groupings G can be formally specified as follows:

$$G = \{G_1, G_2, ..., G_i, ..., G_n\} \qquad (4.1)$$

where G_i represents all the combinations of i faces. It is obvious that such an exhaustive search would be computationally very expensive. As mentioned in Chapter 2, this is one of the main limitations of rule-based approaches.

In order to overcome the shortcomings of the above strategies, it is proposed to apply the concept of hints and to employ both sets of rules during the feature recognition process. In particular, it is suggested initially to search only for faces that are considered hints for the existence of features in the part model. Then, faces matching the definitions of such hints are validated against the first set of rules to identify those from which face sets can be formed. Finally, the second set of rules is employed to verify if such face sets represent valid features. This strategy reduces the initial search space significantly because the seeds only, rather than all the faces in a given B-Rep model, are considered in forming these face sets. The main difference between this implementation of the hint concept and others is the proposed technique for automatic definition of hints. In particular, the set of hints is extracted from the first set of rules that represents patterns based on the geometrical and topological properties of individual feature faces (see Figure 4.1).

4.3.2 Methodology

4.3.2.1 Heuristic measure

All inductive learning algorithms require a measure for assessing the quality of the generated rules. This is usually a statistical measure that is utilised in these algorithms as a search heuristic. In this research, the hints are identified by analysing the conditions in each rule. In particular, a measure is utilised to assess the importance of each condition for a given feature class and thus to select which of them are to be used as hints for that class. This assessment is done by using the data available in the training set that is comprised of the characteristic vectors of individual feature faces and does not require any input from system developers.

The objective for such a statistical measure is to identify conditions that cover a maximum number of characteristic vectors for a given target feature class, while their coverage of those not associated with that class is minimised. To address this requirement, the consistency metric reported in Bigot (2002) is employed:

$$Consistency = \frac{p}{p+n} \tag{4.2}$$

where p is the number of characteristic vectors covered by a condition and belonging to the target feature class and n is the number of characteristic vectors covered by a condition and not belonging to the target feature class.

The measure adopted in this research should also take into account some other considerations in assessing the importance of a given condition. In particular, in the training set studied, it is possible several vectors to be generated from each feature model. Consequently, a condition covering one vector per every model of a given feature class would be preferable to a condition covering a higher number of vectors and, at the same time, not covering at least one vector for each model of this class. The ratio E_1 between f, the number of feature models for a given feature class covered by a condition, and F, the total number of feature models for this class, is used to make this assessment:

$$E_1 = \left(\frac{f}{F}\right) \tag{4.3}$$

Equation 4.3 defines a linear function that evaluates the condition importance in the context of their coverage of the feature models. In particular, this function helps to identify such conditions that cover at least one vector per model for a given feature class. To give E_1 a higher weight in measuring this condition performance, the function is redefined to specify an exponential increase of E_1 by raising it to the power of F:

$$E_2 = (E_1)^F = \left(\frac{f}{F}\right)^F \tag{4.4}$$

as shown graphically in Figure 4.3.

To benefit from both metrics, *Consistency* and E_2, a new heuristic measure M that combines them is designed:

$$M = \frac{p}{p+n} \cdot \left(\frac{f}{F}\right)^F \tag{4.5}$$

4.3.2.2 Hint extraction

The hint extraction requires initially the rules for each feature class in the first set of rules to be grouped together. Then, each condition in this subset of rules is analysed using M. In this way, the importance of each condition is assessed and then they are ranked according to this measure.

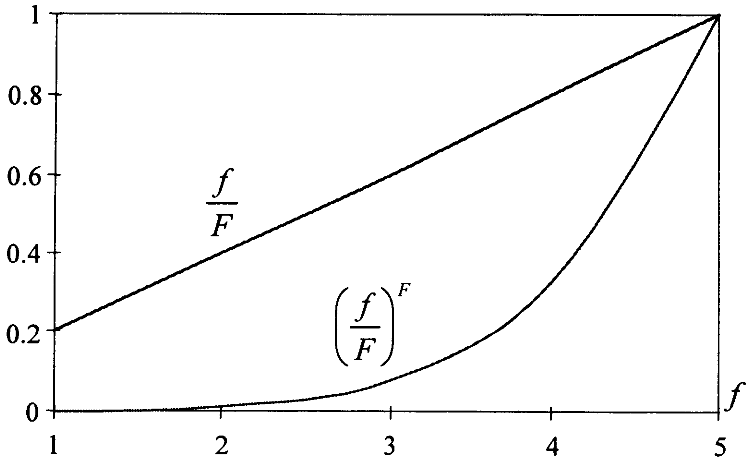


Figure 4.3 The graph of E_1 and E_2 for $F = 5$

The application of the hint concept for feature recognition requires one or several hints for each feature class to be defined. For example, in the approach proposed by Regli (1995), a finite set of hints associated with each feature for a given application domain is identified. Accordingly, the technique applied in this research should allow several hints per feature class to be defined.

The technique proposed in this study allows a *primary feature hint* and a set of *secondary feature hints* to be identified for each feature class. The highest ranked condition for each subset of rules of a given class is considered the primary feature hint for that class. Ideally, the value of E_2 (Equation 4.4) for such a hint should be equal to 1. This ensures that the condition defined by this hint is satisfied by every feature model for that class in the training set. Any other condition ranked below the primary feature hint and, at the same time, whose value of E_2 is equal to the one obtained for the primary feature hint can also be considered a hint. Such conditions form the set of secondary feature hints. It is also possible to define a threshold value for E_2 above which conditions could be used to form the set of secondary feature hints. The adoption of a threshold value is not discussed in this research because this is a parameter that should be defined by a user depending on the specific requirements of a given application domain.

4.3.3 Illustrative example

In the previous chapter, a method for extracting rules from B-Rep feature models was described. The method was implemented by applying the DynaSpace algorithm on training data that include machining features belonging to the classes defined in Table 3.4. The first set of rules extracted from this data (see Table 3.5) is used in this example to illustrate the proposed hint definition technique. This technique was implemented again using the Java™ programming language.

The primary feature hints obtained for all the feature classes covered by these rules are shown in Table 4.1. In addition, Figure 4.4 depicts the technique when it was applied on rules defined for the blind hole feature class (ho_bl). The primary feature hint for a blind hole is defined as 'a face whose number of edges equals two', 'nEd=2'. The value of E_2 for this hint is equal to 1. Thus, every feature model of that class in the training set has a face satisfying this condition. The set of secondary feature hints is also formed and it includes all the remaining conditions because E_2 for each of them is also equal to 1. The hints in this set are ranked according to the value of M as it is shown in Figure 4.4.

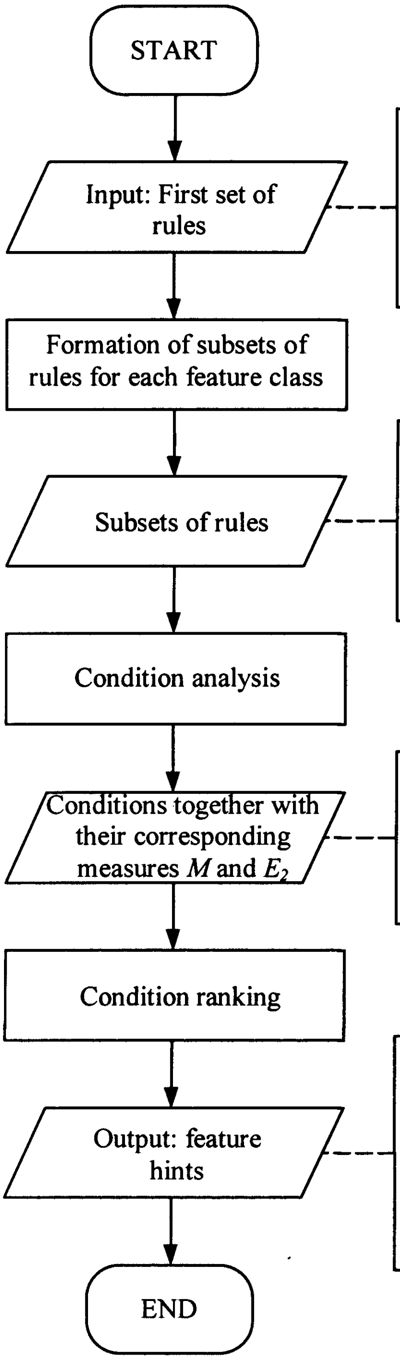
4.4 Feature recognition process

4.4.1 Methodology

An overview of the feature recognition process was provided in Section 4.2.2. The recognition of a particular feature in a B-Rep part model is considered an 'entity growing' process analogous to the growth of a vegetal plant. This section discusses in detail each of the four sub-processes that take place during the feature recognition process. They are defined as the seed detection, stem development, leaf development and feature validation sub-processes.

Feature class	**Primary feature hint**	
	Condition	**Description**
po_re	ccAd = 3	A face whose ccAd measure equals three.
po_ob	nSccEd = 2	A face with two smooth concave edges.
ho_bl	nEd = 2	A face whose number of edges equals two.
ho_th	nCcEd = 0	A face with no concave edge.
pr_ci	faceCv = cv	A convex face.
pr_re	nCvEd = 3	A face whose number of convex edges equals three.
sl_th	nCvEd = 3	A face whose number of convex edges equals three.
sl_nt	nCcEd = 2	A face with two concave edges.
st	ccAd = 0	A face whose ccAd measure equals zero.

Table 4.1 Primary feature hints


START
Input: First set of rules
Formation of subsets of rules for each feature class
Subsets of rules
Condition analysis
Conditions together with their corresponding measures M and E_2
Condition ranking
Output: feature hints
END

Rule	Rule description
1	**IF** ccAd=3 **THEN** featureClass = po_re
...	...
15	**IF** faceTy=pl **AND** 1<=nPlAd<=4 **AND** ccAd=0 **THEN** featureClass = st

Subset of rules for the ho_bl feature class	
Rule	Rule description
5	**IF** faceCv=cc **AND** nPlAd=2 **AND** nCvEd=1 **THEN** featureClass = ho_bl
6	**IF** nEd=2 **AND** nCcEd=2 **THEN** featureClass = ho_bl

Condition analysis for the ho_bl feature class					
Condition	faceCv=cc	nPlAd=2	nCvEd=1	nEd=2	nCcEd=2
E_2	E_2= 1	E_2= 1	E_2= 1	E_2= 1	E_2= 1
M	M = 0.33	M = 0.20	M = 0.14	M = 0.50	M = 0.25

Hints for the ho_bl feature class		
Primary feature hint	nEd=2	
Secondary feature hints	1	faceCv=cc
	2	nCcEd=2
	3	nPlAd=2
	4	nCvEd=1

Figure 4.4 Automatic hint definition process illustrated with the blind hole (ho_bl) feature class

4.4.1.1 Seed detection

The aim of this sub-process is to detect individual faces in a B-Rep model that are considered hints for the existence of features in a part. Every face identified in this way is called a seed. As a result, the initial search space for features is divided into a set of smaller search spaces. Each of these sub-spaces contains all of the faces in a solid model that match the description of a given hint. Figure 4.5 illustrates the seed detection sub-process. It is possible for a face to match the descriptions of the hints of more than one feature class. For example, in this figure, f_3 was detected as a seed for two feature classes. This is due to the fact that a hint represents only a hypothesis for the presence of a feature in a part model. In particular, each hint in this research is only a partial definition of a given feature class. Therefore, a face could satisfy the conditions of more than one hint, hence this face could be considered a seed for more than one feature class.

4.4.1.2 Stem development

A validation step is necessary to confirm or discard any hypothesis associated with a seed. This implies that a seed may or may not be considered further for constructing from it a feature. During the learning process, the feature hints are derived from the first set of rules and therefore, this rule base is employed in validating each seed during the stem development sub-process. When a set of seeds for a particular feature class is analysed, first, all the rules of that class are identified and then applied to validate each seed. When a seed satisfies the conditions of a rule, this means that it meets the geometrical and topological constraints associated with a feature face and thus, it could be utilised to construct a feature of a given class. Such a seed is then

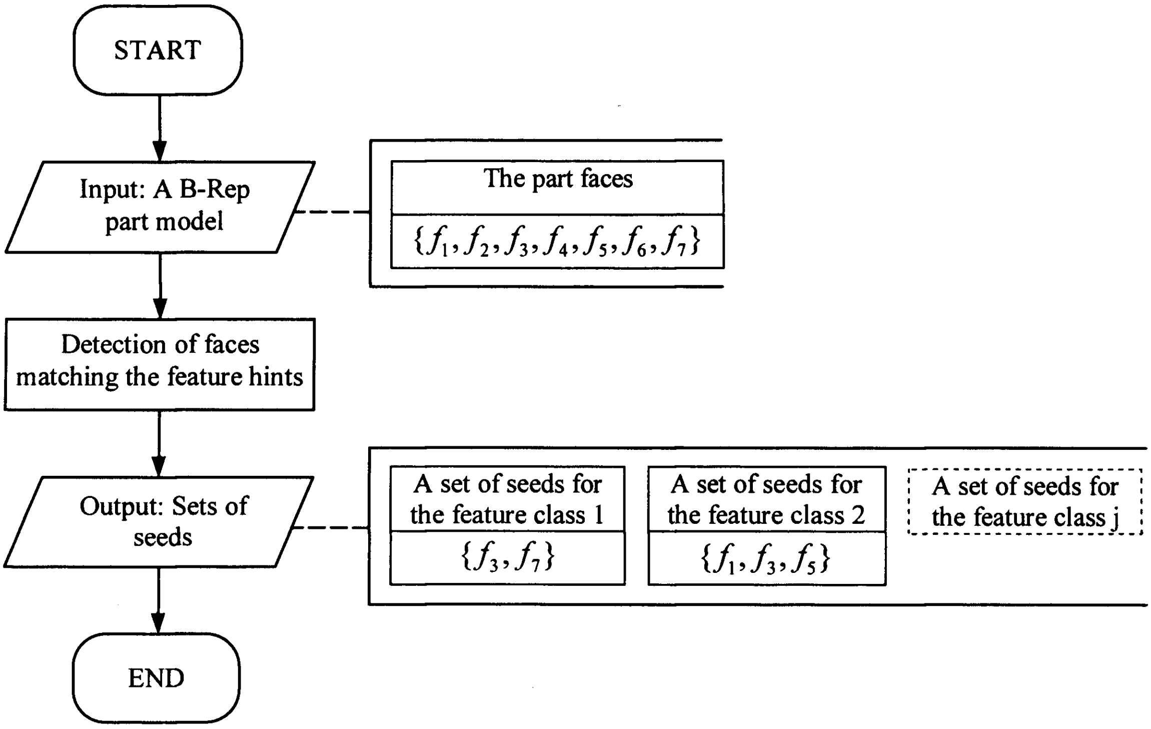

START
Input: A B-Rep part model
The part faces
{f1,f2,f3,f4,f5,f6,f7}
Detection of faces matching the feature hints
Output: Sets of seeds
A set of seeds for the feature class 1
{f3,f7}
A set of seeds for the feature class 2
{f1,f3,f5}
A set of seeds for the feature class j
END

Figure 4.5 Seed detection sub-process illustrated with an example

called a stem because it is used to identify its surrounding faces and to form a potential feature by aggregating these faces together.

4.4.1.3 Leaf development

Once a stem has been identified, it is then utilised to build a face set that could form a feature. Such a face set is called a plant and is composed of faces labelled either as stems or leaves. Figure 4.6 illustrates the stem and leaf development sub-processes.

The leaf development sub-process is performed by a geometric reasoning algorithm that uses a stem as input. In Figure 4.6, two stems, f_1 and f_5, for a feature class are successively analysed by the algorithm. For each of these stems, one leaf, a face, is identified and thus, two plants, $\{f_1, f_2\}$ and $\{f_5, f_6\}$, are formed. This analysis is carried out by verifying whether some of the surrounding faces to a stem could be used to construct a plant. The algorithm stops when pre-defined *termination conditions* are reached. Such conditions should be defined by taking into account only the top level, the most generic part, of the feature taxonomy discussed in Chapter 3. Thus, the proposed geometrical reasoning mechanism would not be restricted to any specific taxonomy.

The top level of the taxonomy adopted in this research classifies features as either protrusions, depressions or surfaces. However, the termination conditions are restricted to protrusion and depression features only. Surface features are not considered and thus, the proposed algorithm is not valid for them. Two termination conditions are defined for stopping the algorithm. The utilisation of one or the other condition depends on whether the feature class of the considered face set corresponds

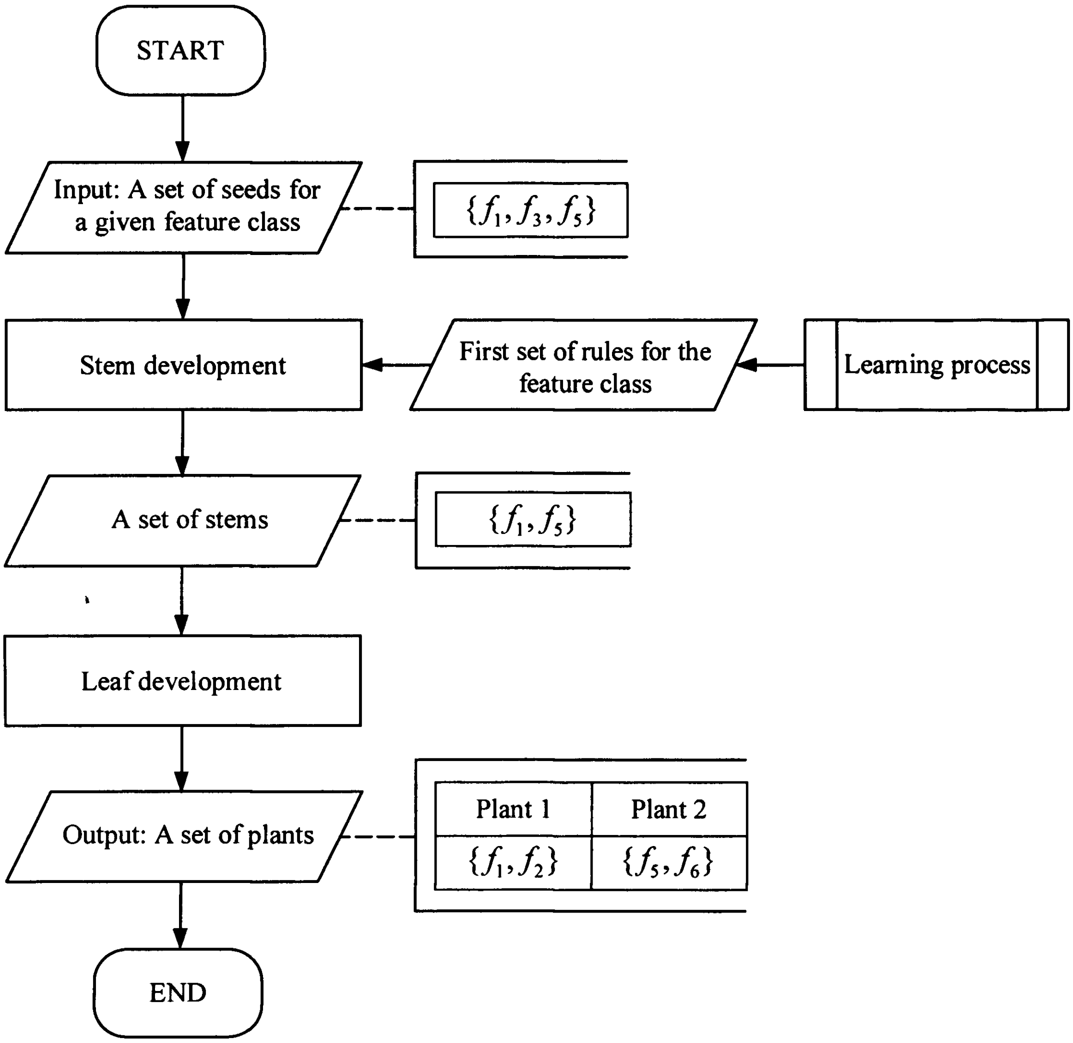


Figure 4.6 Overview of the stem and leaf development sub-processes illustrated with an example

to a protrusion or depression. Similarly to Sakurai and Gossard (1990), the boundary edges of a depression feature are defined as a closed sequence of convex edges. Inversely, the boundary edges of a protrusion feature are defined as a closed sequence of concave edges. Following these definitions, a termination condition is reached when the constructed face set is bounded by one of these two closed sequences of edges.

Figure 4.7 illustrates the geometrical reasoning algorithm that is utilised during the leaf development sub-process. The first step tests if the stem input to the algorithm is already included in an existing plant for the considered feature class. If the output of this test is negative, a new plant composed of the stem is created. Next, an iterative process takes place for aggregating more faces into this plant. In this iterative process, the faces that are adjacent to the faces in the plant and, at the same time, that are not included in that plant, are analysed. This analysis determines if the edges shared between such adjacent faces and faces in the plant correspond to the internal edges for a feature. For each adjacent face considered, if the output of this analysis is positive, it is called a leaf and is appended to the plant (the face set formed so far). Conversely, if the output is negative, it means that the face shares a boundary edge with one of the faces in the plant. Therefore, this face cannot be appended to the face set and it is considered a boundary face to the plant. This procedure is recursively applied to all adjacent faces that are considered potential leaves for a given plant. When no more faces can be appended to the plant, the termination condition for the created face set is satisfied and the leaf development process stops.

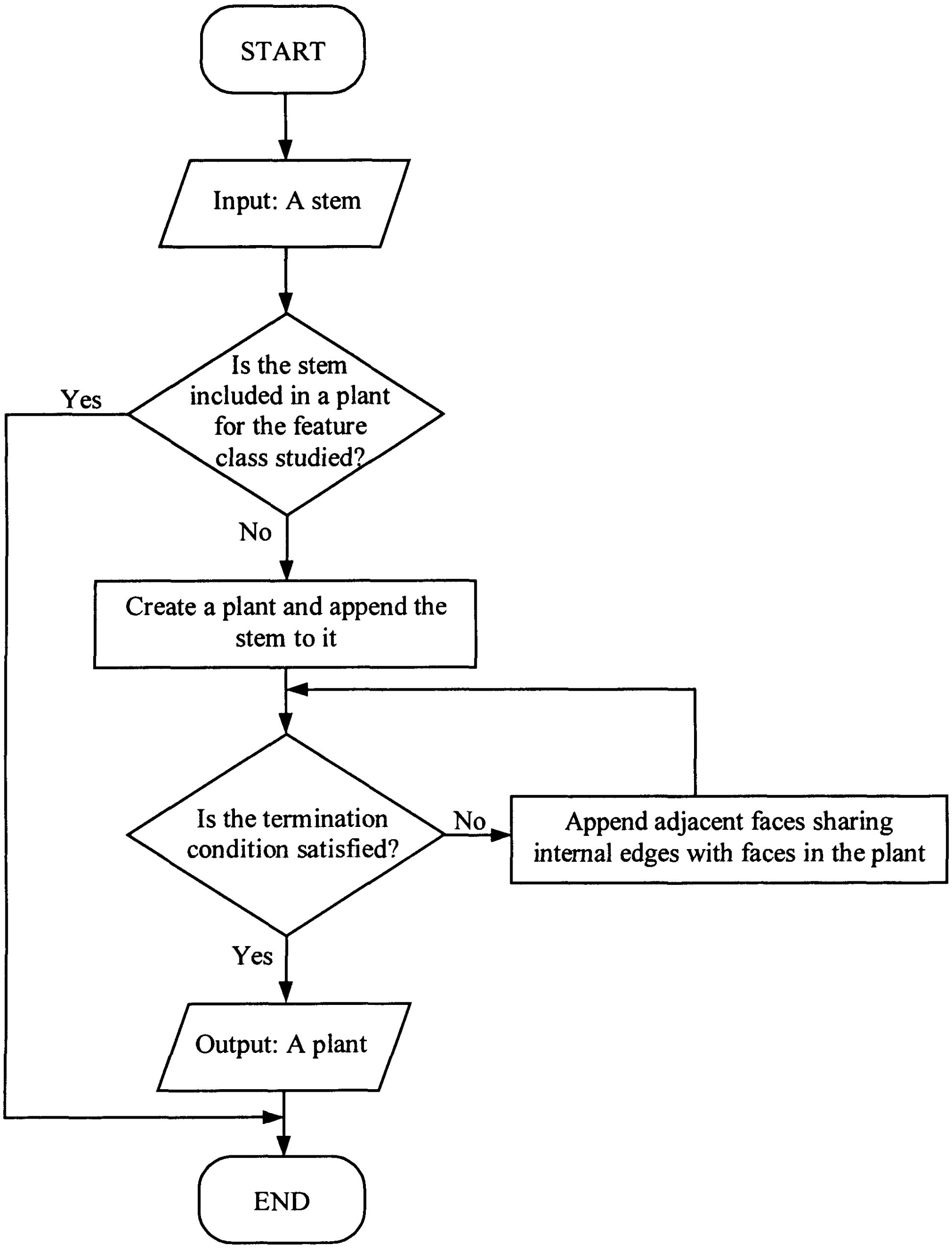

START
Input: A stem
Is the stem included in a plant for the feature class studied?
Yes
No
Create a plant and append the stem to it
Is the termination condition satisfied?
No
Append adjacent faces sharing internal edges with faces in the plant
Yes
Output: A plant
END

Figure 4.7 Geometrical reasoning algorithm for the leaf development sub-process

4.4.1.4 Feature validation

The second set of rules, that includes patterns defining geometrical and topological relations between feature faces, is applied to validate the plants formed during the leaf development sub-process. One possible outcome of this validation could be that the face set defining a given plant satisfies all the conditions in a rule and thus, the feature constructed with the faces in this plant is considered recognised. This means that the plant meets all geometrical and topological constraints that are associated with a face set defining a valid feature of a given class. The other possible outcome could be that the plant is not validated by the rules due to three possible reasons. The first is that each plant is only a potential feature, just a hypothesis constructed around a stem and thus, a plant may fail to be validated. The second reason could be that the plant constitutes a valid feature whose class is not included in the taxonomy adopted for a given application. Therefore, this taxonomy should be extended. Finally, it is possible that the plant constitutes a valid feature for one of the classes of a given taxonomy but, at the same time, the topological and geometrical configuration of this plant is not covered by the existing rules for that class. In such a case, the coverage of these rules should be extended.

The coverage of an existing rule set could be increased by considering the plants that are not validated to belong to the feature class of the closest rule. Figure 4.8 provides an example showing how the distance, D_P^R, between a rule R and a plant P could be computed in a two-dimensional space. In this figure, each cross corresponds to a characteristic vector in a training set from which the second set of rules is extracted. Also, all the vectors shown belong to the same feature class. The rectangle represents the area covered by a given rule. The dashed line illustrates the distance between the

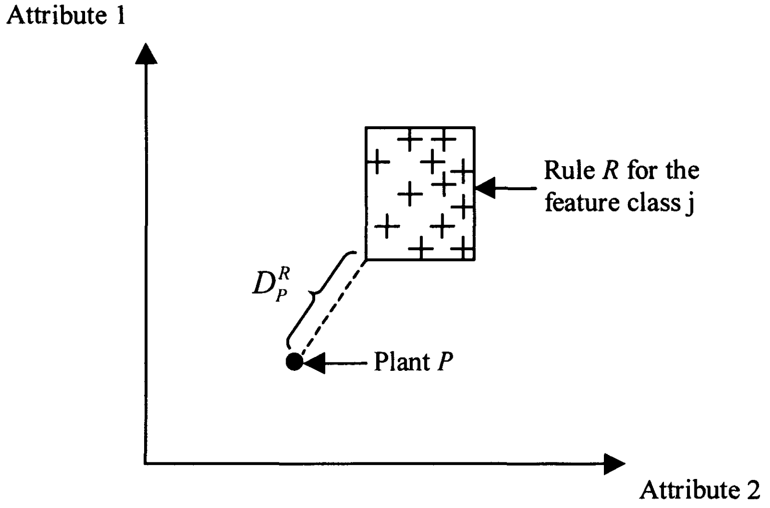


Figure 4.8 The distance D_P^R between a rule R and a plant P

rule and the plant not validated by it. This distance indicates the likelihood of a given plant to be a valid feature of a given class. Thus, the user of the system is provided with two options: to reject or accept the plant studied as a valid feature. In the second case, a new feature model should be added to the training set of the considered feature class. Then, by executing the learning process again, a new rule base can be automatically generated.

In this research, the distance measurement between a rule, *R*, and a plant, *P*, is defined as follows (Bigot, 2002):

$$D_P^R = \sqrt{\sum_c d_c + \sum_d d_d} \tag{4.6}$$

where $\sum_c$ is the sum of the continuous attributes and $\sum_d$ is the sum of the discrete attributes of the characteristic vector of *P*. The value d_c is defined for each continuous attribute as follows:

- □ If the value of the i^th attribute in the characteristic vector is outside the condition range for this attribute in a rule:

$$d_c = \left(\frac{\min\left(\left| V_P^i - Vmax_R^i \right|, \left| V_P^i - Vmin_R^i \right| \right)}{V_{max}^i - V_{min}^i} \right)^2 \tag{4.7}$$

- □ Else:

$$d_c = 0 \tag{4.8}$$

Where: V_P^i is the value of the i^th attribute of the characteristic vector; $Vmax_R^i$ and $Vmin_R^i$ represent the range defined by the condition for the i^th attribute in rule *R*; V_{max}^i

and V^i_{min} are the maximum and the minimum values of the ith attribute among all the vectors present in the training data. Finally, the value d_d is defined for each discrete attribute as follows:

- If $V^i_P = V^i_R$ then:

$$d_d = 0 \qquad (4.9)$$

- Else:

$$d_d = 1 \qquad (4.10)$$

4.4.2 Illustrative example

The proposed feature recognition method was implemented using the Java™ programming language. Both sets of rules obtained in Chapter 3 (see Table 3.5 and Table 3.6) and the hints defined in Section 4.3.3 are utilised in this illustrative example by the developed prototype system.

4.4.2.1 Test part

A test part (see Figure 4.9) which has been used by other researchers (Marquez et al., 2001) is utilised to validate the capabilities of the prototype system. According to the taxonomy of machining features defined in Chapter 3, this part contains one circular protrusion, one non-through slot, three through holes, one through slot, one step, one blind hole and two rectangular pockets. A 3D model of this part was created using Pro/Engineer™ and then exported into a STEP AP203 file. The B-Rep data were extracted from this file by applying the parsers described in Chapter 3. Each entity in

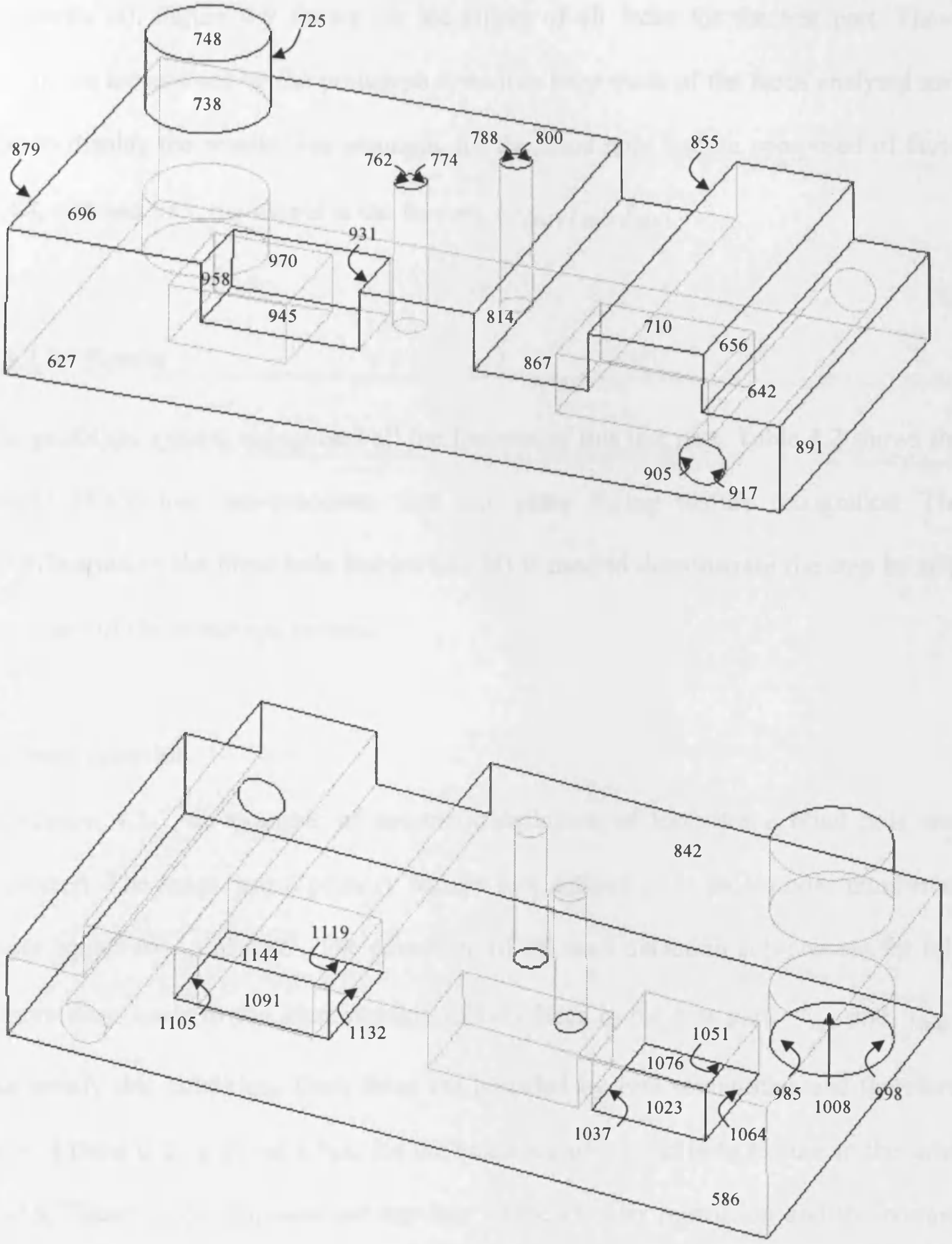


Figure 4.9 Two views of the test part (Marquez et al., 2001)

a STEP file has a unique numerical identifier, an integer, associated with it (see Appendix B). Figure 4.9 shows the identifiers of all faces for the test part. These identifiers are utilised by the prototype system to keep track of the faces analysed and also to display the results. For example, for the blind hole feature composed of faces 1008, 998 and 985, the output is the face set $\{f_{1008}, f_{998}, f_{985}\}$.

4.4.2.2 Results

The prototype system recognised all the features of this test part. Table 4.2 shows the results of the four sub-processes that take place during feature recognition. The identification of the blind hole feature (ho_bl) is used to demonstrate the step by step execution of the prototype system.

□ Seed detection

In Section 4.3.3, an example of automatic definition of hints for a blind hole was presented. The result was a primary feature hint defined as ‘a face whose number of edges equals two’, ‘nEd=2’. The execution of the seed detection sub-process for this feature class leads to the identification of two faces in the test part, f_{748} and f_{1008}, that satisfy this condition. Both faces are bounded by two semicircles and therefore each of them is considered a hint for the existence of a blind hole feature in the solid model. These seeds represent the top face of the circular protrusion and the bottom face of the blind hole respectively.

Feature class	Set of seeds detected	Set of stems developed	Plant developed	Feature validated
po_re	$\{f_{1132}, f_{1076}, f_{1119}, f_{1037}, f_{1023}, f_{1064}, f_{1091}, f_{1144}, f_{1051}, f_{1105}\}$	$\{f_{1132}, f_{1076}, f_{1119}, f_{1037}, f_{1023}, f_{1064}, f_{1091}, f_{1144}, f_{1051}, f_{1105}\}$	$\{f_{1132}, f_{1119}, f_{1091}, f_{1144}, f_{1105}\}$	**yes**
			$\{f_{1076}, f_{1037}, f_{1023}, f_{1064}, f_{1051}\}$	**yes**
po_ob	$\{f_{998}, f_{985}, f_{774}, f_{905}, f_{788}, f_{800}, f_{917}, f_{762}\}$	$\{\emptyset\}$		
ho_bl	$\{f_{748}, f_{1008}\}$	$\{f_{1008}\}$	$\{f_{1008}, f_{998}, f_{985}\}$	**yes**
ho_th	$\{f_{627}, f_{842}, f_{586}, f_{710}, f_{774}, f_{905}, f_{748}, f_{788}, f_{800}, f_{891}, f_{879}, f_{917}, f_{762}\}$	$\{f_{774}, f_{905}, f_{788}, f_{800}, f_{917}, f_{762}\}$	$\{f_{762}, f_{774}\}$	**yes**
			$\{f_{905}, f_{917}\}$	**yes**
			$\{f_{788}, f_{800}\}$	**yes**
pr_ci	$\{f_{725}, f_{738}\}$	$\{f_{725}, f_{738}\}$	$\{f_{725}, f_{738}, f_{748}\}$	**yes**
pr_re	$\{f_{855}, f_{656}, f_{814}, f_{642}\}$	$\{\emptyset\}$		
sl_th	$\{f_{855}, f_{656}, f_{814}, f_{642}\}$	$\{f_{855}, f_{814}\}$	$\{f_{855}, f_{814}, f_{867}\}$	**yes**
sl_nt	$\{f_{1008}, f_{958}, f_{867}, f_{931}, f_{696}\}$	$\{f_{958}, f_{931}\}$	$\{f_{958}, f_{931}, f_{970}, f_{945}\}$	**yes**
st	$\{f_{905}, f_{656}, f_{891}, f_{917}, f_{642}\}$	$\{f_{656}, f_{891}, f_{642}\}$	$\{f_{656}, f_{642}\}$	**yes**
			$\{f_{891}\}$	no

Table 4.2 The results of the feature recognition for the test part in Figure 4.9

□ Stem development

Next, these two seeds are validated against the first set of rules defined for the blind hole feature class. Seed f_{748}, the top face of the circular protrusion, fails this validation because there is no rule that covers the characteristic vector of this face. The other seed, f_{1008}, the bottom face of the blind hole, is successfully validated as a stem for forming a plant.

□ Leaf development

This sub-process adds two leaves to the stem, f_{1008}, to form a plant composed of three faces, f_{1008}, f_{998} and f_{985}. Figure 4.10 illustrates the steps involved in identifying the leaves belonging to this plant.

□ Feature validation

The plant formed during the leaf development sub-process is successfully validated using the second set of rules for the ho_bl feature class. In particular, the characteristic vector of this plant is covered by the rules of that class. As a result, the blind hole composed of faces f_{1008}, f_{998} and f_{985} is recognised.

4.4.2.3 Discussion

It should be noted that the results presented in Table 4.2 were obtained using only the primary feature hints. Thus, these hints were sufficient to recognise all the features present in the test part. In addition, as already mentioned, theoretically, a face could match the feature hints of more than one class. For example, the face f_{1008} of the test part matches not only the hint for a blind hole but also that for a non-through slot.

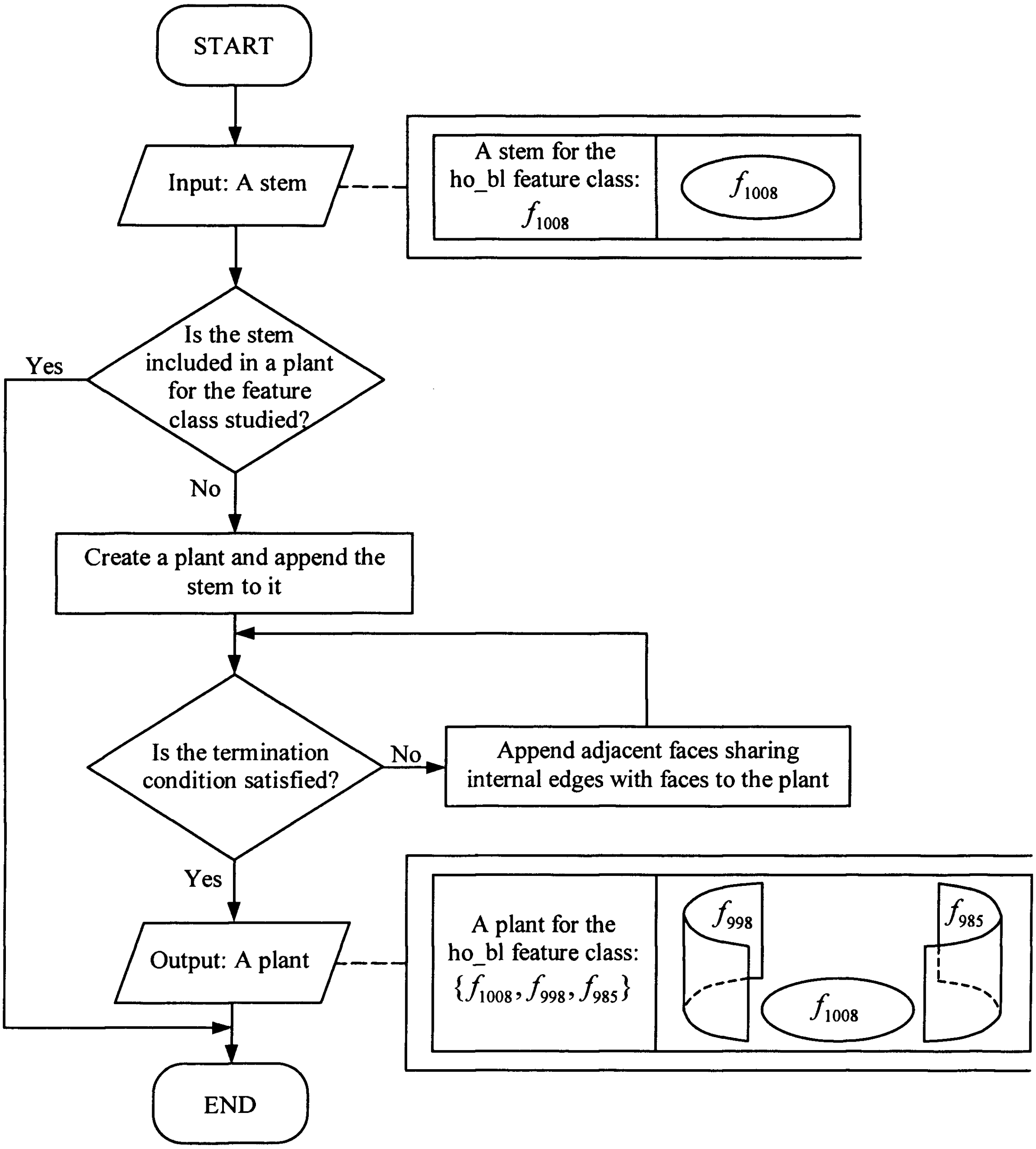

START
Input: A stem
A stem for the ho_bl feature class: f_{1008}
f_{1008}
Is the stem included in a plant for the feature class studied?
Yes
No
Create a plant and append the stem to it
Is the termination condition satisfied?
No
Append adjacent faces sharing internal edges with faces to the plant
Yes
Output: A plant
A plant for the ho_bl feature class: $\{f_{1008}, f_{998}, f_{985}\}$
f_{998}
f_{985}
f_{1008}
END

Figure 4.10 An example of leaf development for a ho_bl feature

This demonstrates once more that a seed is just a hypothesis for the presence of a given feature in a part model and that further analysis is required to verify this. In this example, the hypothesis that f_{1008} indicates the existence of a non-through slot in the model was not validated during the stem development sub-process. As stated before, the leaf development sub-process is limited to protrusion and depression feature types and does not cover surface features. The test part considered in this study does not include surface features and therefore all machining features in it were recognised according to the taxonomy adopted in this illustrative example.

The plant, $\{f_{891}\}$, was rejected as a step during the feature validation sub-process. Thus, the distances between the characteristic vector of this plant and the rules in Table 3.6 were computed and the results are shown in Table 4.3. The closest rule to this plant is rule R9 for the step feature class. Based on this information, the user of the system should decide if this particular plant could be considered a valid step feature. If the decision is yes, the feature model corresponding to this plant is added to the training set of the step feature class and two new rule sets are generated by executing the learning process. Thus, in case such a plant is encountered again by the system, it will be recognised automatically as a step feature.

4.5 Summary

This chapter has presented a new hybrid AFR method that employs the 'learning from examples' concept with the rule-based and hint-based feature recognition approaches. The method applies the rule extraction techniques proposed in Chapter 3. A technique has also been devised for automatic definition of feature hints for the classes of a

Rule number	R9	R4	R8	R7	R6	R1	R5	R3	R2
Feature class	st	ho_bl	sl_th	sl_nt	pr_re	po_re	pr_ci	ho_th	po_ob
$\boldsymbol{D_P^R}$	0.20	0.25	0.40	0.60	0.80	1.0	1.0	1.05	1.08

Table 4.3 The distances between the plant $\{f_{891}\}$ and the rules in Table 3.6

given taxonomy. This technique overcomes one of the main limitations of other existing implementations of the hint-based methods for which the hint definition task is always carried out manually.

Another important characteristic of the proposed AFR method is that it is not tied to a particular application domain as most of the feature recognition approaches reviewed in Chapter 2. Rules and also feature hints could be defined automatically for any application as long as the features of interest are represented in the considered taxonomy. Thus, new rules and hints could be added easily to the knowledge base of the system to extend its application area. Moreover, given the fact that hints are derived from IF – THEN rules, they are readily understandable by the users. This is beneficial to system developers because this offers them a new insight into the hint definition process.

Patterns indicating the existence of a particular feature in a model are extracted automatically and reflect the training data available at any particular moment. Thus, the rule sets cover only those areas in the feature space that are represented with characteristic vectors in the training set. Consequently, a new feature could be recognised if its characteristic vector falls in one of the areas covered by the existing rule sets. These generalisation capabilities of the inductive learning techniques allow unseen features, i.e. features that are not included in the training set, to be recognised.

Finally, if the characteristic vector of a plant is not covered by any existing rule, the likelihood of such a plant being a valid feature could be estimated by computing the distance between this plant and all available rules in the knowledge base.

Unfortunately, this distance measurement may not be sufficient to recognise unseen features that result from feature interactions. Some possible solutions to this problem will be discussed in the next chapter.

Chapter 5 - Recognition of interacting features

5.1 Introduction

The previous chapter introduced a hybrid AFR method that combines different feature recognition approaches. The method was implemented and then tested on a B-Rep part model composed of features that do not intersect with each other. However, the recognition of features when they interact is very important for developing robust AFR systems and, at the same time, it constitutes a major challenge in feature recognition research (Li et al., 2003). Thus, the problem of recognising interacting features by applying the proposed AFR method is discussed in this chapter. In particular, the objective of this research is to suggest, to implement and to test solutions for recognising such interacting features. Also, it is important that these solutions are built upon the main idea implemented in the proposed AFR method, in particular, the application of the 'learning from examples' concept with the rule-based and hint-based feature recognition approaches.

First, the chapter discusses a definition for interacting features together with a classification of their different types. Then, the shortcomings of the proposed AFR method for recognising such features are analysed and some solutions are suggested to overcome these limitations. Following this analysis, a geometric reasoning mechanism is described for extending the capabilities of this method to tackle the recognition problems associated with interacting features. Finally, different parts with such features are studied in order to validate the proposed geometric reasoning mechanism.

5.2 Feature interactions

5.2.1 Definition

It is important to define the concept of interacting features in the context of this research because in the literature, there is no consensus about its meaning. For instance, this could describe features that do not belong to the same part and that mate or connect to each other during an assembly operation. This issue is also illustrated by the fact that several terms are used to define this concept, such as intersecting, interacting, compound and complex features.

In this research, the concept of interacting features falls into the category proposed by Regli and Pratt (1996) that characterises interactions by an overlap between two or more features resulting in modifications affecting some of their faces. In addition, the geometric form created by such interactions corresponds to the term of a compound feature introduced by Shah and Mantyla (1995). A compound feature represents a group of features that is not arranged in a circular or linear pattern and that can be decomposed into two or more *simple* features. In this research, the set of considered simple features is defined by the taxonomy adopted for a given application and they cannot be decomposed further into other features present in this taxonomy.

The interaction between two or more simple features results in modifications that affect the geometry of their faces and also their topology. As a result, essential information for the feature recognition process of the proposed AFR method could be altered or even removed. Thus, it is important to carry out a systematic analysis of the possible feature modifications resulting from such interactions.

5.2.2 Types of feature interactions

This section discusses different approaches for categorising possible types of feature interactions and based on this, the classification adopted in this research is described.

Joshi and Chang (1988) consider two types of interactions that depend on the B-Rep entities shared between the interacting features. For the first type, the features only have common edges and the faces of one of them could be split up. For the other type, they share a common face and the interaction also splits one of the feature faces. A classification with two types of interactions is also proposed by Nezis and Vosniakos (1997). It considers the faces involved in the interaction and groups them into two different types: *internal* and *external.* An internal face represents a feature face and a boundary face is considered external. Thus, the first type of interaction in this classification represents compound features having a face that is at the same time an internal and an external face of different simple features. The second type represents compound features having a face that is internal for more than one simple feature. Zhang et al. (1998) consider two other common face modifications that take part when features interact, in particular, when a feature face is partly removed or when it is completely divided.

The most explicit and comprehensive description of the possible types of feature interactions is provided by Gao and Shah (1998). They defined six categories according to the following three topology variations caused by the interactions: merging of faces, loss of concave edges and splitting of faces. Merged faces are defined as those that are shared by more than one simple feature. To cover all combinations of topology variations, eight types of feature interactions should be considered. However, two variations are not possible as both the splitting of a face and the loss of a concave edge cannot be present simultaneously. Table 5.1 describes these six types of interaction and they are illustrated in Figure 5.1.

In this research, the classification proposed by Gao and Shah (1998) is adopted because it constitutes a comprehensive framework for studying the effects of different types of interactions in regard to the applied AFR method. The use of this classification is justified also by the fact that Li et al. (2003) recently applied it for comparing different AFR approaches.

5.3 AFR method analysis

In this section, the two main processes of the proposed AFR method, learning and feature recognition, are discussed in order to understand their sensitivity to the interaction types considered in this research. Also, solutions are suggested to overcome the shortcomings of these processes when applied to such interacting features.

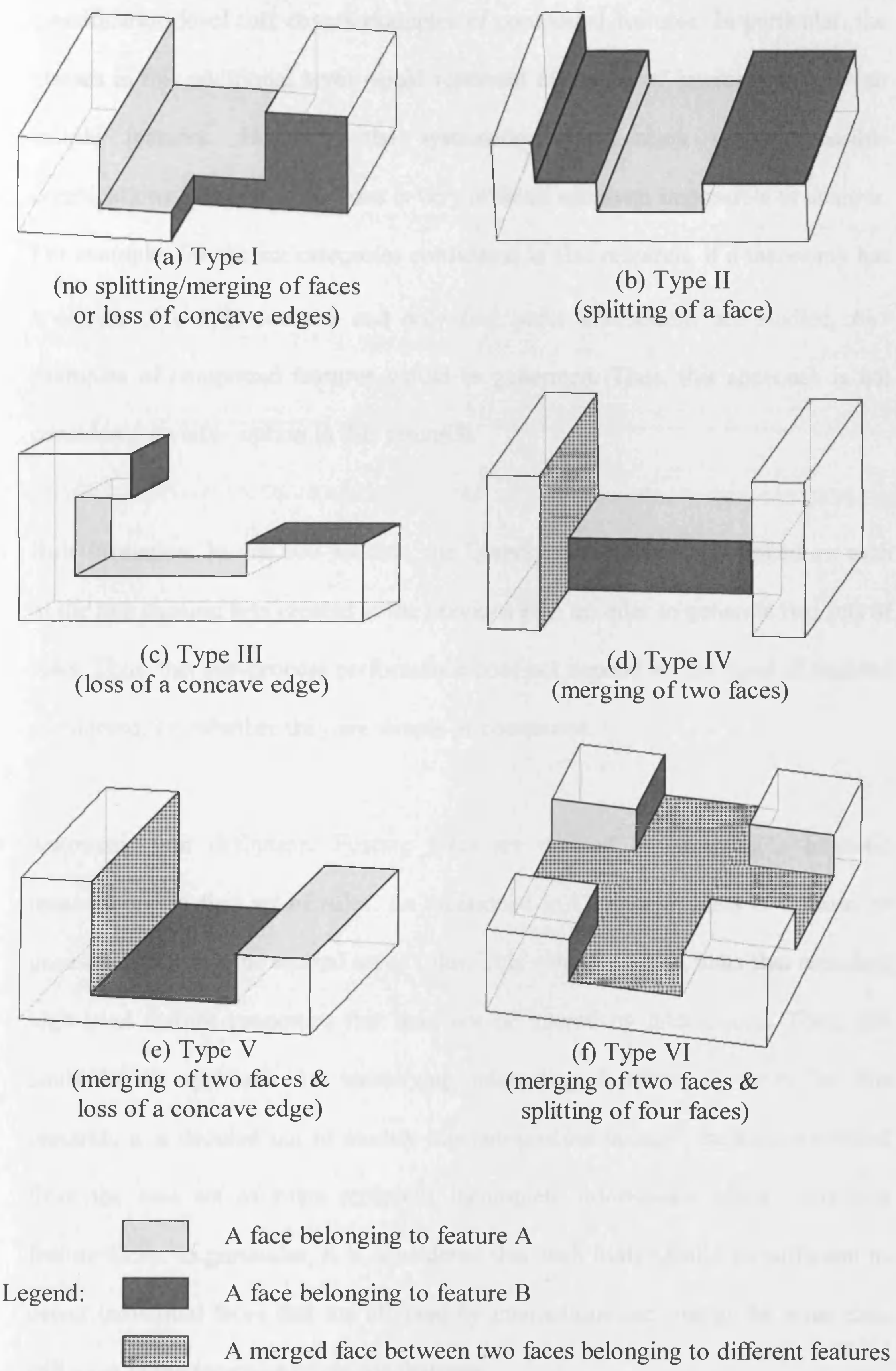
5.3.1 Learning process

This process is composed of three consecutive sub-processes:

- □ Training data creation. B-Rep feature models are created for all classes in a given taxonomy and then, characteristic vectors at two levels of abstraction are extracted from these models. To apply this sub-process for recognising interacting features, one solution could be to modify this taxonomy by including an additional

Interaction type	Merged faces	Lost concave edges	Split faces
I	No	No	No
II	No	No	Yes
III	No	Yes	No
IV	Yes	No	No
V	Yes	Yes	No
VI	Yes	No	Yes

Table 5.1 Classification of feature interactions (Gao and Shah, 1998)



(a) Type I
(no splitting/merging of faces
or loss of concave edges)

(b) Type II
(splitting of a face)

(c) Type III
(loss of a concave edge)

(d) Type IV
(merging of two faces)

(e) Type V
(merging of two faces &
loss of a concave edge)

(f) Type VI
(merging of two faces &
splitting of four faces)

Legend:
░ A face belonging to feature A
▓ A face belonging to feature B
▒ A merged face between two faces belonging to different features

Figure 5.1 Examples of feature interactions (adapted from Gao and Shah, 1998)

classification level that covers examples of compound features. In particular, the classes in this additional level would represent examples of interactions between simple features. However, the systematic identification of all possible combinations of such interactions is very difficult and even impossible to achieve. For example, for the six categories considered in this research, if a taxonomy has n classes of simple features and only first order interactions are studied, $6n^2$ examples of compound features would be generated. Thus, this approach is not considered a viable option in this research.

- □ Rule formation. In this sub-process, the DynaSpace algorithm is applied on each of the two training sets created at the previous step in order to generate two sets of rules. Thus, this sub-process performance does not depend on the types of features considered, i.e. whether they are simple or compound.

- □ Automatic hint definition. Feature hints are defined by applying a heuristic measure on the first set of rules. As mentioned in Chapter 4, hints could also be generated by using the second set of rules. This would result in hints that represent high-level feature properties that may not be altered by interactions. Thus, this could be an approach for identifying interacting features. However, in this research, it is decided not to modify this sub-process because the hints extracted from the first set of rules represent incomplete information about individual feature faces. In particular, it is considered that such hints should be sufficient to detect individual faces that are affected by interactions and that at the same time still exist in the structure of simple features.

The outcome of this brief analysis is that it is not appropriate to modify any of the learning sub-processes in order to apply the proposed AFR method for recognising interacting features. Thus, the feature recognition process should be adapted for identifying such features.

5.3.2 Feature recognition process

The recognition of a particular feature in a B-Rep part model is considered an 'entity growing' process analogous to the growth of a vegetal plant. In particular, four consecutive sub-processes take place during the feature recognition process: seed detection, stem development, leaf development and feature validation.

- Seed detection. Individual faces matching the definition of feature hints are detected during this sub-process. The introduction of hint-based approaches for AFR resulted in the first significant progress towards solving the problem of recognising interacting features (Marefat and Kashyap, 1990; Vandenbrande and Requicha, 1993). The hint concept is already applied in this research and therefore, this sub-process does not require any modification.

- Stem development. Any hypothesis associated with a seed is validated or rejected by employing the first set of rules that define patterns based on the geometrical and topological properties of individual feature faces. However, this sub-process is not suitable for recognising interacting features because the rules utilised are formed from examples of features that do not interact. Thus, it is possible that a seed could fail this validation stage although it represents a true hypothesis about the existence of a feature. This is explained by the fact that, as a result of the interactions, the geometry and topology of such a seed do not match the patterns defined in the first set of rules. Thus, it is suggested that this sub-process should be bypassed and that the seeds should be used, directly, as inputs to the next step - the leaf development.

- Leaf development. A geometric reasoning algorithm is employed to generate a face set (a plant) from a seed instead of using as an input, a stem. It is very important that this algorithm retrieves a face set that potentially represents a simple feature. Thus, it would be possible to use the rules that define patterns of simple features for the validation of this plant. The algorithm utilised in this sub-process stops when the face set considered is bounded by a closed sequence of concave or convex edges. For this reason, there are two possible outcomes when some of the faces in a plant are affected by interactions:

1. The algorithm could form a plant that does not include all faces belonging to a simple feature. This could result from interaction types II, III, V and VI because they lead to either the splitting of faces or the loss of concave edges.

2. The algorithm could retrieve a plant that includes faces belonging to different simple features. This could occur due to interaction types IV, V and VI because they result in face merging.

Figure 5.2 shows an example of the first case where a step feature is composed of the face set $\{f_1, f_2, f_3\}$. In this example, if either f_1 or f_2 is detected as a seed, then the plant $\{f_1, f_2\}$ bounded by a closed sequence of convex edges would be

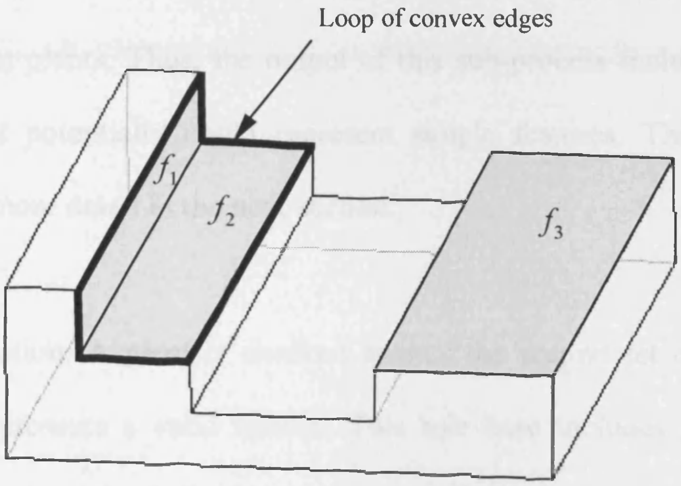


Figure 5.2 A feature face outside a closed loop of convex edges

formed without taking f_3 into account. Thus, a sub-process should be added to verify that a given plant includes all relevant faces, and only those faces forming a simple feature. The sub-process that is introduced follows the leaf development idea and is called *plant modification.*

- □ Plant modification. Faces in a plant are analysed in order to detect if they are affected by feature interactions. Depending on the type of interactions identified, a plant could either be extended to include more faces or divided to form two or more different plants. Thus, the output of this sub-process includes one or more face sets that potentially could represent simple features. This sub-process is described in more detail in the next section.

- □ Feature validation. A plant is checked against the second set of rules to verify whether it represents a valid feature. This rule base includes patterns defining geometrical and topological relations between faces that are not affected by any feature interactions. It is not required that this sub-process is modified because the plant modification carried out at the previous step results in face sets that potentially could represent simple features. For a given B-Rep part model, it is expected that the number of plants rejected by this sub-process will be greater than the number of plants rejected by the AFR method discussed in Chapter 4. This is due to the fact that a seed, instead of a stem, is utilised to generate a plant.

Figure 5.3 illustrates the proposed feature recognition process together with the modifications introduced to address the problems associated with the recognition of interacting features.

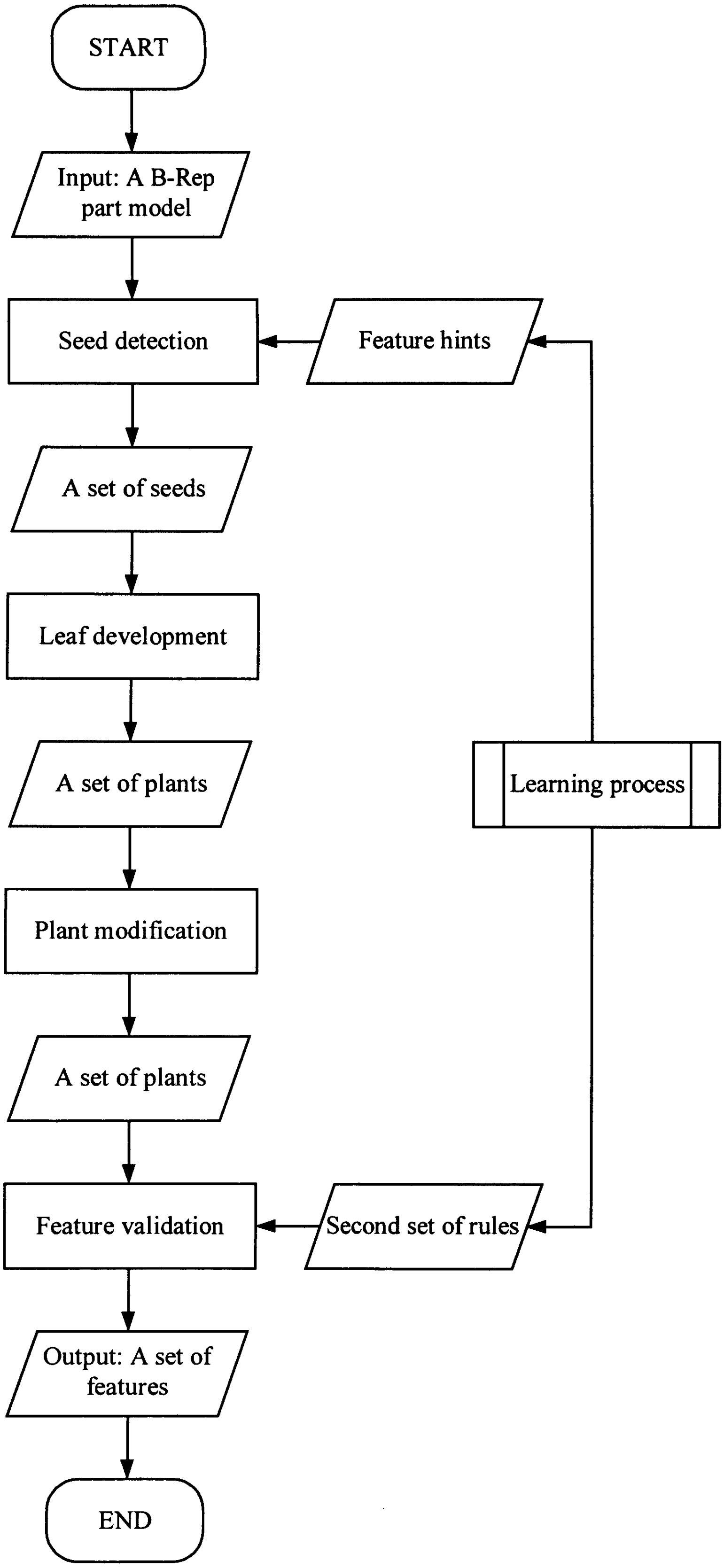


Figure 5.3 The modified feature recognition process for interacting features

5.4 Plant modification

A plant is analysed in order to check whether it includes all relevant faces and also whether it is comprised only of faces defining a simple feature. If this is not the case, further processing is required to generate from a plant one or more different face sets that meet this requirement. In particular, a geometric reasoning algorithm is employed to detect face properties that could be associated with interaction types II to VI. Only faces affected by interaction type I are not considered because such a feature interaction does not lead to any merged or split faces or to the loss of concave edges. Thus, such faces should not prevent the leaf development sub-process from effectively reconstructing the face set defining a potential simple feature from a seed.

The following face properties indicate the existence of one of the considered five interactions in a plant:

- □ Type II interaction (face splitting). Faces f_a and f_b could be affected by an interaction of this type if they lie on the same plane and their normal vectors have the same orientation, and one can be extended to merge with the other face without intersecting any other face in the part model. Such faces are called *type II faces* and are represented by the notation f_a^{II} and f_b^{II}.

- □ Type III interaction (loss of concave edges). The effect of this type of interaction on two planar faces f_a and f_b is that both are completely in the *positive halfspace* of each other and their extensions intersect each other without colliding with any other face in the part model. The positive halfspace of a face is

$\{P(x_P, y_P, z_P) \mid ax_P + by_P + cz_P \geq d\}$ where $ax + by + cz = d$ is the equation of the plane associated with the face and (a,b,c) is its outward pointing normal vector. Such faces are called *type III faces*, f_a^{III} and f_b^{III}.

- Type IV interaction (face merging). This type of interaction can be identified if a planar face, f_a, shares a concave edge with two other planar faces, f_b and f_c, and if these two faces share a convex edge between them. Such a merged face is called a *type IV face*, f_a^{IV}.

- Type V (face merging and concave edge loss). The planar faces f_a and f_b are indicative of this type of interaction if f_a is completely in the positive halfspace of f_b, while f_b is partly in the positive halfspace of f_a. In addition, the extension of f_a should intersect f_b and divide it completely without colliding with any other face in the part model. A face that matches the description of f_b corresponds to a merged face and is called a *type V face*, f_b^{V}.

- Type VI (face merging and splitting). A type IV face, f_a^{IV}, with adjacent faces matching the description of type II faces, could be the result of this type of interaction. Such a merged face is called a *type VI face*, f_a^{VI}.

For simplicity, these face properties are defined only for planar faces, however they could be extended to include cylindrical and other face types.

Figure 5.4 illustrates the geometric reasoning algorithm used in the plant modification sub-process. First, the algorithm tests if a particular face type exists in a given plant. If this is the case, the structure of the plant is then modified according to the interaction type detected. The presence of type VI faces is checked first because their properties are similar to those of the type II and type IV faces. Next, the algorithm searches for type II to V faces in the plant. The outcome of this sub-process could be:

- The plant is not affected by any feature interactions.

- The initial plant is modified if one or more type II or III faces are identified. Such a modified plant includes all its original faces plus some others resulting from the feature interactions. In particular, these additional faces could be a result of type II interactions that lead to face splitting. Thus, when the characteristic vector for such a plant is extracted, it includes not only information about its original faces but also about their corresponding type II faces. These additional faces could also be a result of type III interactions. In this case, when the characteristic vector of such a face set is extracted, these faces are considered adjacent to their corresponding type III faces in the plant.

- The plant is divided into two or more plants if faces of type IV, V or VI are detected. Such faces result from the merger of faces belonging to different simple features. Therefore, they should be divided into two or more separate faces depending on the number of mergers resulting from the feature interactions. These new faces should then be used to form new plants. Thus, two or more face sets could be created from the original plant.

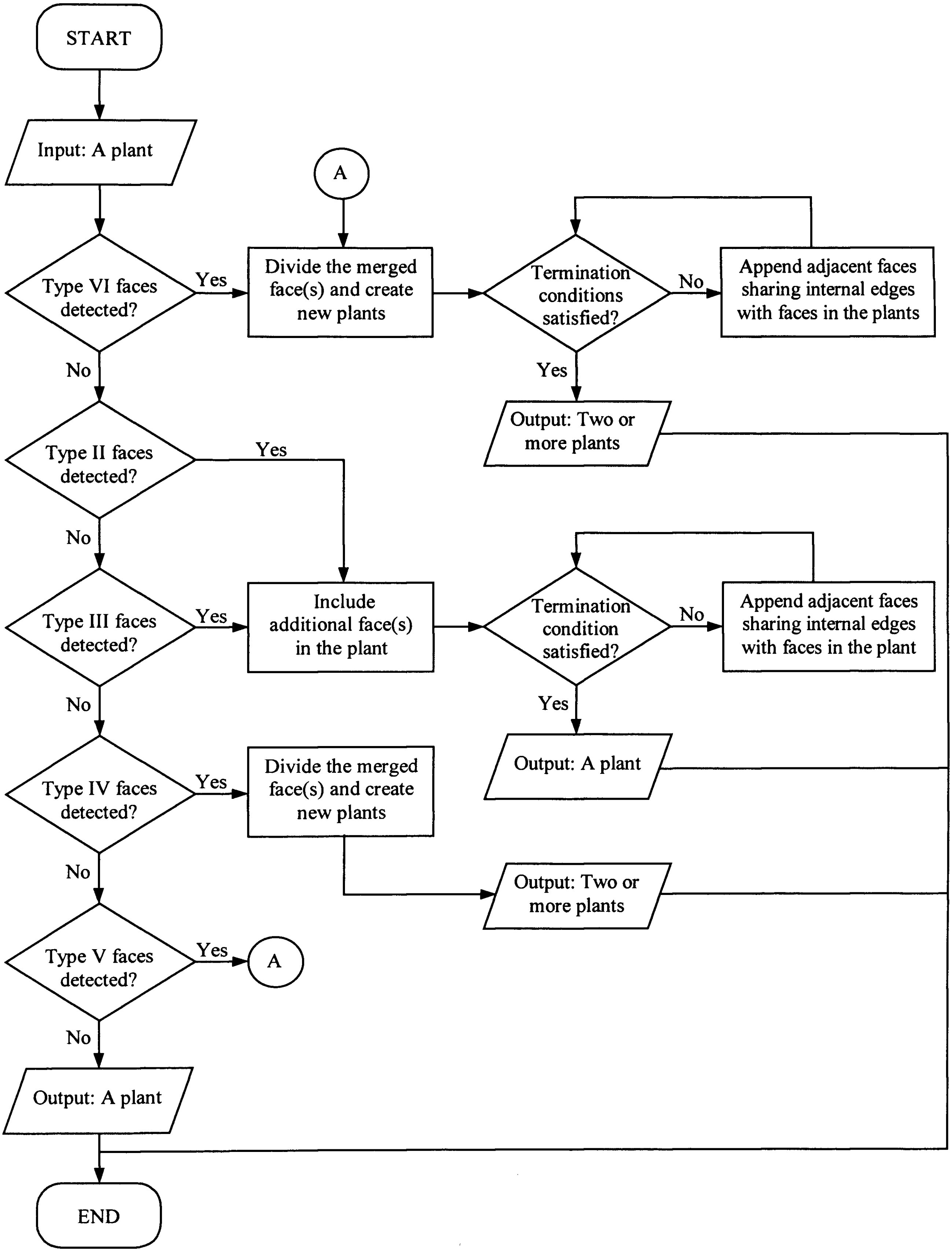

START
Input: A plant
Type VI faces detected?
Yes
No
Divide the merged face(s) and create new plants
A
Termination conditions satisfied?
No
Yes
Append adjacent faces sharing internal edges with faces in the plants
Output: Two or more plants
Type II faces detected?
Yes
No
Type III faces detected?
Yes
No
Include additional face(s) in the plant
Termination condition satisfied?
No
Yes
Append adjacent faces sharing internal edges with faces in the plant
Output: A plant
Type IV faces detected?
Yes
No
Divide the merged face(s) and create new plants
Output: Two or more plants
Type V faces detected?
Yes
A
No
Output: A plant
END

Figure 5.4 Geometric reasoning algorithm for the plant modification sub-process

The proposed plant modification sub-process was implemented within the prototype system described in Chapter 4 using the Java™ programming language. Only the detection of the extension of a face colliding with other faces represented in the part is carried out manually. In the next section, six parts illustrating the different types of interactions studied in this research together with two benchmarking parts are used to verify the proposed extensions to the AFR method.

5.5 Illustrative examples

In this section, validation studies are carried out on different test parts that include interacting simple features as defined in Chapter 3, Table 3.4. Thus, it is possible in these studies to apply the rule sets obtained in Chapter 3 and the feature hints generated in Chapter 4. The 3D models of the test parts were created using the Pro/Engineer™ CAD system and then, the B-Rep data were extracted from their STEP files by applying the parsers described in Chapter 3.

The following sub-sections present six case studies, each of which corresponds to one of the six interaction types discussed in Section 5.2.2, together with two studies carried out on benchmark parts. For each of these case studies, the solid model of the test part is presented and the feature interactions existing in the model are described. Then, the results of the recognition process are discussed.

5.5.1 Case study I

The test part utilised in this case study is shown in Figure 5.5 and, according to the adopted taxonomy, a type I interaction takes place between a through slot $\{f_{199}, f_{217}, f_{346}\}$ and a step feature $\{f_{322}, f_{236}\}$. This interaction does not change the number of feature faces and internal edges of either feature. Therefore, their corresponding plants would be formed directly during the leaf development sub-process.

The results of the recognition process are shown in Table 5.2. The through slot and the step features were successfully recognised. It should be noted that for the step, the highest ranked secondary feature hint defined as 'a face whose surface is planar', 'faceTy = pl' was used because the primary feature hint for that class, 'ccAd = 0', was not satisfied.

The hint 'ccAd = 0' relies on a feature attribute that represents the ratio between the number of concave edges of the adjacent faces and the total number of such faces. Hints defined using this attribute show some limitations when they are applied to interacting features. This is due to the fact that the number of concave edges of the adjacent faces changes as a result of the interaction. Thus, when interacting features are present in the part model, it would be better not to use hints defined with this attribute.

It should be noted that some seeds were detected for the rectangular protrusion (pr_re) feature class. However, the plant generated from them includes all faces of the part model since no loops of concave edges exist in this part. In this special case, the plant

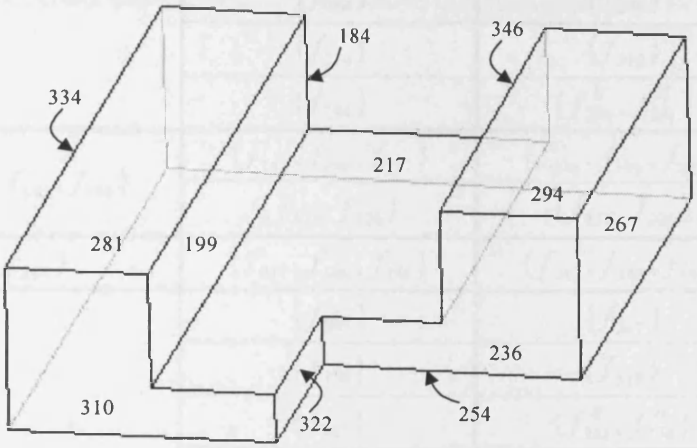


Figure 5.5 The solid model for case study I

Feature class	Set of seeds detected	Plant developed	Plant modified	Feature validated
ho_th	$\{f_{281}, f_{267}, f_{334}, f_{184}, f_{254}, f_{310}, f_{294}\}$	$\{f_{281}\}$	$\{f^{II}_{281}, f^{II}_{294}\}$	no
		$\{f_{267}\}$	$\{f_{267}\}$	no
		$\{f_{334}\}$	$\{f_{334}\}$	no
		$\{f_{184}\}$	$\{f_{184}\}$	no
		$\{f_{254}\}$	$\{f_{254}\}$	no
		$\{f_{310}\}$	$\{f_{310}\}$	no
		$\{f_{294}\}$	$\{f^{II}_{294}, f^{II}_{281}\}$	no
sl_th	$\{f_{322}, f_{346}, f_{199}\}$	$\{f_{346}, f_{199}, f_{217}\}$	$\{f_{346}, f_{199}, f_{217}\}$	yes
		$\{f_{322}, f_{236}\}$	$\{f_{322}, f_{236}\}$	no
sl_nt	$\{f_{217}\}$	$\{f_{217}, f_{346}, f_{199}\}$	$\{f_{217}, f_{346}, f_{199}\}$	no
st	$\{f_{281}, f_{217}, f_{322}, f_{236}, f_{267}, f_{334}, f_{184}, f_{346}, f_{254}, f_{199}, f_{310}, f_{294}\}$	$\{f_{267}\}$	$\{f_{267}\}$	no
		$\{f_{310}\}$	$\{f_{310}\}$	no
		$\{f_{281}\}$	$\{f^{II}_{281}, f^{II}_{294}\}$	no
		$\{f_{254}\}$	$\{f_{254}\}$	no
		$\{f_{334}\}$	$\{f_{334}\}$	no
		$\{f_{184}\}$	$\{f_{184}\}$	no
		$\{f_{294}\}$	$\{f^{II}_{294}, f^{II}_{281}\}$	no
		$\{f_{322}, f_{236}\}$	$\{f_{322}, f_{236}\}$	yes
		$\{f_{199}, f_{346}, f_{217}\}$	$\{f_{199}, f_{346}, f_{217}\}$	no

Table 5.2 Results for case study I

is automatically discarded before the feature validation sub-process starts because the part model does not correspond to a feature. When such a case occurs, the seeds detected are not reported.

5.5.2 Case study II

This case study illustrates a type II interaction that takes place between a through slot $\{f_{321}, f_{334}, f_{346}\}$ and a step $\{f_{203}, f^{II}_{232}, f^{II}_{218}\}$ (see Figure 5.6). This interaction affects the step feature by splitting one of its original faces into f^{II}_{232} and f^{II}_{218}.

Table 5.3 shows the results of the recognition process. Both features, the through slot and the step, were recognised successfully. During the plant modification sub-process, f^{II}_{218} and f^{II}_{232} were added into the plants $\{f_{203}, f_{232}\}$ and $\{f_{218}\}$ respectively. Then, these two faces were regarded as a single entity during the feature validation sub-process.

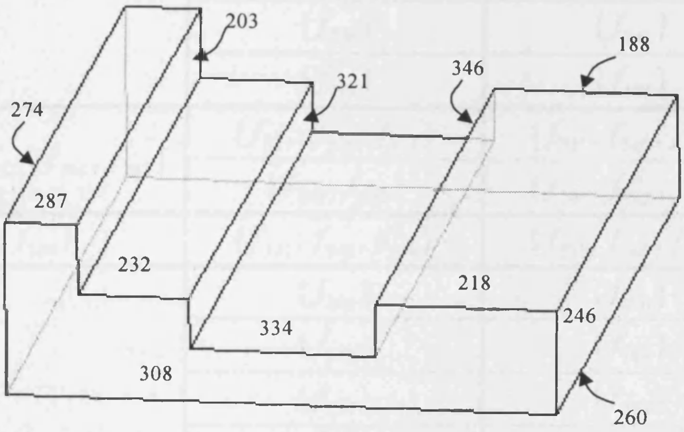


Figure 5.6 The solid model for case study II

Feature class	Set of seeds detected	Plant developed	Plant modified	Feature validated
ho_th	$\{f_{308}, f_{274}, f_{218}, f_{287}, f_{260}, f_{246}, f_{188}\}$	$\{f_{308}\}$	$\{f_{308}\}$	no
		$\{f_{274}\}$	$\{f_{274}\}$	no
		$\{f_{218}\}$	$\{f_{203}, f^{II}_{232}, f^{II}_{218}\}$	no
		$\{f_{287}\}$	$\{f_{287}\}$	no
		$\{f_{260}\}$	$\{f_{260}\}$	no
		$\{f_{246}\}$	$\{f_{246}\}$	no
		$\{f_{188}\}$	$\{f_{188}\}$	no
sl_th	$\{f_{203}, f_{321}, f_{346}, f_{232}\}$	$\{f_{321}, f_{346}, f_{334}\}$	$\{f_{321}, f_{346}, f_{334}\}$	**yes**
		$\{f_{203}, f_{232}\}$	$\{f_{203}, f^{II}_{232}, f^{II}_{218}\}$	no
sl_nt	$\{f_{334}\}$	$\{f_{321}, f_{346}, f_{334}\}$	$\{f_{321}, f_{346}, f_{334}\}$	no
st	$\{f_{308}, f_{274}, f_{218}, f_{203}, f_{287}, f_{334}, f_{260}, f_{321}, f_{346}, f_{246}, f_{232}, f_{188}\}$	$\{f_{246}\}$	$\{f_{246}\}$	no
		$\{f_{188}\}$	$\{f_{188}\}$	no
		$\{f_{274}\}$	$\{f_{274}\}$	no
		$\{f_{308}\}$	$\{f_{308}\}$	no
		$\{f_{287}\}$	$\{f_{287}\}$	no
		$\{f_{203}, f_{232}\}$	$\{f_{203}, f^{II}_{232}, f^{II}_{218}\}$	**yes**
		$\{f_{218}\}$	$\{f_{203}, f^{II}_{232}, f^{II}_{218}\}$	**yes**
		$\{f_{321}, f_{346}, f_{334}\}$	$\{f_{321}, f_{346}, f_{334}\}$	no
		$\{f_{260}\}$	$\{f_{260}\}$	no

Table 5.3 Results for case study II

5.5.3 Case study III

Figure 5.7 shows the solid model used in this case study to illustrate an interaction of type III between a step $\{f_{203}^{III}, f_{273}^{III}\}$ and a feature that is not present in the adopted taxonomy (Table 3.4). This new feature that is composed of faces, f_{231}, f_{245}, f_{259}^{III} and f_{217}^{III}, could be called a *passage*. The interaction between the step and the passage results in the loss of a concave edge for both features, in particular the edges between the original faces f_{203} and f_{273} and f_{259} and f_{217} respectively. Thus, each of them matches the characteristic of a type III face.

The results of the feature recognition process are shown in Table 5.4. The type III faces of the step feature were identified during the plant modification sub-process. Then, during the feature validation, the step was recognised by taking into account that both faces in the modified plant $\{f_{273}^{III}, f_{203}^{III}\}$ should share a concave edge. Furthermore, the face set $\{f_{231}, f_{245}, f_{259}^{III}, f_{217}^{III}\}$ that defines the passage was used to form a plant that was considered to belong to two different classes, sl_th and sl_nt. During the feature validation sub-process, this plant was even validated as a feature for one of these classes. This result could be explained by the fact that such a feature class does not exist in the taxonomy adopted in this research.

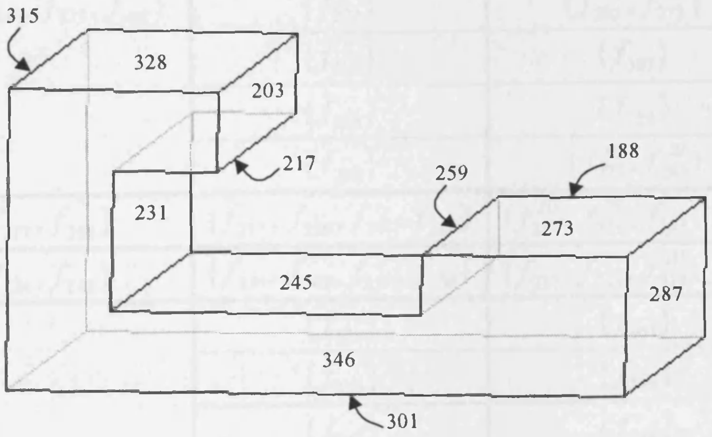


Figure 5.7 The solid model for case study III

Feature class	Set of seeds detected	Plant developed	Plant modified	Feature validated
ho_th	$\{f_{315}, f_{203}, f_{287}, f_{328}$ $f_{301}, f_{346}, f_{273}, f_{188}\}$	$\{f_{346}\}$	$\{f_{346}\}$	no
		$\{f_{287}\}$	$\{f_{287}\}$	no
		$\{f_{315}\}$	$\{f_{315}\}$	no
		$\{f_{188}\}$	$\{f_{188}\}$	no
		$\{f_{203}\}$	$\{f_{203}^{III}, f_{273}^{III}\}$	no
		$\{f_{301}\}$	$\{f_{301}\}$	no
		$\{f_{328}\}$	$\{f_{328}\}$	no
		$\{f_{273}\}$	$\{f_{273}^{III}, f_{203}^{III}\}$	no
sl_th	$\{f_{217}, f_{259}\}$	$\{f_{217}, f_{259}, f_{245}, f_{231}\}$	$\{f_{217}^{III}, f_{259}^{III}, f_{245}, f_{231}\}$	no
sl_nt	$\{f_{231}, f_{245}\}$	$\{f_{231}, f_{245}, f_{217}, f_{259}\}$	$\{f_{231}, f_{245}, f_{217}^{III}, f_{259}^{III}\}$	yes
st	$\{f_{217}, f_{315}, f_{231}, f_{203},$ $f_{287}, f_{328}, f_{301}, f_{346},$ $f_{273}, f_{245}, f_{259}, f_{188}\}$	$\{f_{301}\}$	$\{f_{301}\}$	no
		$\{f_{315}\}$	$\{f_{315}\}$	no
		$\{f_{346}\}$	$\{f_{346}\}$	no
		$\{f_{273}\}$	$\{f_{273}^{III}, f_{203}^{III}\}$	yes
		$\{f_{203}\}$	$\{f_{203}^{III}, f_{273}^{III}\}$	yes
		$\{f_{188}\}$	$\{f_{188}\}$	no
		$\{f_{287}\}$	$\{f_{287}\}$	no
		$\{f_{328}\}$	$\{f_{328}\}$	no
		$\{f_{217}, f_{259}, f_{231}, f_{245}\}$	$\{f_{217}^{III}, f_{259}^{III}, f_{231}, f_{245}\}$	no

Table 5.4 Results for case study III

5.5.4 Case study IV

This case study focuses on the interaction of type IV and uses the solid model shown in Figure 5.8. It represents an interaction occurring between two through slots, $\{f_{203}^{IV}, f_{217}, f_{268}^{IV}\}$ and $\{f_{203}^{IV}, f_{231}, f_{268}^{IV}\}$. In this solid model, faces f_{203}^{IV} and f_{268}^{IV} are the result of the merger of two faces, each of them belonging to one of the considered two through slots.

Table 5.5 shows the results of the recognition process. One plant only, $\{f_{217}, f_{268}, f_{231}, f_{203}\}$, was first constructed for the through slot feature class. Next, f_{203} and f_{268} in this plant were identified as type IV faces. The convex edge between the adjacent faces with which each of them shares a concave edge indicates that they should be considered divided into two. Thus, the original plant was modified to construct two new plants $\{f_{203}^{IV}, f_{217}, f_{268}^{IV}\}$ and $\{f_{203}^{IV}, f_{231}, f_{268}^{IV}\}$ that were then confirmed as valid through slot features. It should also be noted that this result was obtained by using the highest secondary feature hint for a through slot that is defined as ‘a face whose number of concave edges equals two’, ‘nCcEd=2’.

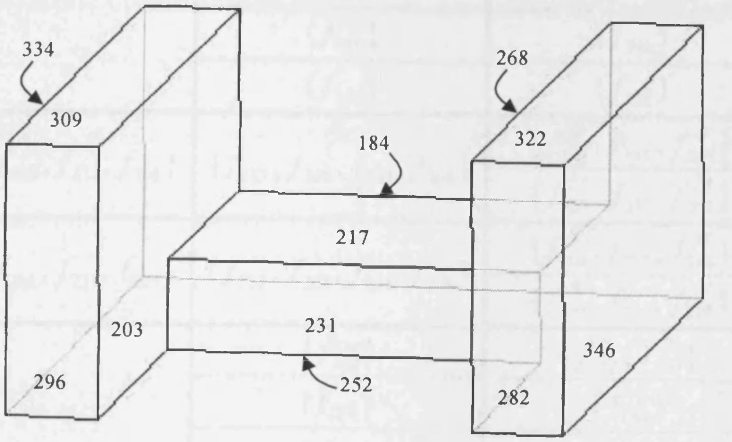


Figure 5.8 The solid model for case study IV

Feature class	Set of seeds detected	Plant developed	Plant modified	Feature validated
ho_th	$\{f_{322}, f_{282}, f_{309}, f_{334}, f_{184}, f_{346}, f_{252}, f_{296}\}$	$\{f_{334}\}$	$\{f_{334}\}$	no
		$\{f_{184}\}$	$\{f_{184}\}$	no
		$\{f_{322}\}$	$\{f^{II}_{322}, f^{II}_{309}\}$	no
		$\{f_{296}\}$	$\{f^{II}_{296}, f^{II}_{282}\}$	no
		$\{f_{282}\}$	$\{f^{II}_{282}, f^{II}_{296}\}$	no
		$\{f_{309}\}$	$\{f^{II}_{309}, f^{II}_{322}\}$	no
		$\{f_{346}\}$	$\{f_{346}\}$	no
		$\{f_{252}\}$	$\{f_{252}\}$	no
sl_th	$\{f_{217}, f_{268}, f_{231}, f_{203}\}$	$\{f_{217}, f_{268}, f_{231}, f_{203}\}$	$\{f^{IV}_{203}, f_{217}, f^{IV}_{268}\}$	**yes**
			$\{f^{IV}_{203}, f_{231}, f^{IV}_{268}\}$	**yes**
sl_nt	$\{f_{217}, f_{268}, f_{231}, f_{203}\}$	$\{f_{217}, f_{268}, f_{231}, f_{203}\}$	$\{f^{IV}_{203}, f_{217}, f^{IV}_{268}\}$	no
			$\{f^{IV}_{203}, f_{231}, f^{IV}_{268}\}$	no
st	$\{f_{217}, f_{322}, f_{268}, f_{231}, f_{203}, f_{282}, f_{309}, f_{334}, f_{184}, f_{346}, f_{252}, f_{296}\}$	$\{f_{322}\}$	$\{f^{II}_{322}, f^{II}_{309}\}$	no
		$\{f_{184}\}$	$\{f_{184}\}$	no
		$\{f_{309}\}$	$\{f^{II}_{309}, f^{II}_{322}\}$	no
		$\{f_{296}\}$	$\{f^{II}_{296}, f^{II}_{282}\}$	no
		$\{f_{217}, f_{268}, f_{231}, f_{203}\}$	$\{f^{IV}_{203}, f_{217}, f^{IV}_{268}\}$	no
			$\{f^{IV}_{203}, f_{231}, f^{IV}_{268}\}$	no
		$\{f_{334}\}$	$\{f_{334}\}$	no
		$\{f_{282}\}$	$\{f^{II}_{282}, f^{II}_{296}\}$	no
		$\{f_{252}\}$	$\{f_{252}\}$	no
		$\{f_{346}\}$	$\{f_{346}\}$	no

Table 5.5 Results for case study IV

5.5.5 Case study V

The solid model used in this case study illustrates a type V interaction between a through slot $\{f_{237}^{V}, f_{251}, f_{264}\}$ and a step $\{f_{167}, f_{237}^{V}\}$ (see Figure 5.9). This interaction causes the merger of two faces belonging to these two features into f_{237}^{V}. This also leads to the loss of the internal concave edge for the step feature that was shared between f_{167} and f_{237} before the interaction.

The results of the recognition process are shown in Table 5.6. Both the step and the through slot features were recognised as a result of the changes made to the face set, $\{f_{237}, f_{264}, f_{251}\}$, during the plant modification sub-process. In particular, f_{237} was identified as a type V face and then, divided into two faces since f_{167} is the only face that could split it. Thus, two plants $\{f_{237}^{V}, f_{264}, f_{251}\}$ and $\{f_{237}^{V}, f_{167}\}$ were created and then confirmed as valid through slot and step features respectively. The faces in the plant $\{f_{237}^{V}, f_{167}\}$ were considered adjacent and sharing a concave edge during the feature validation sub-process.

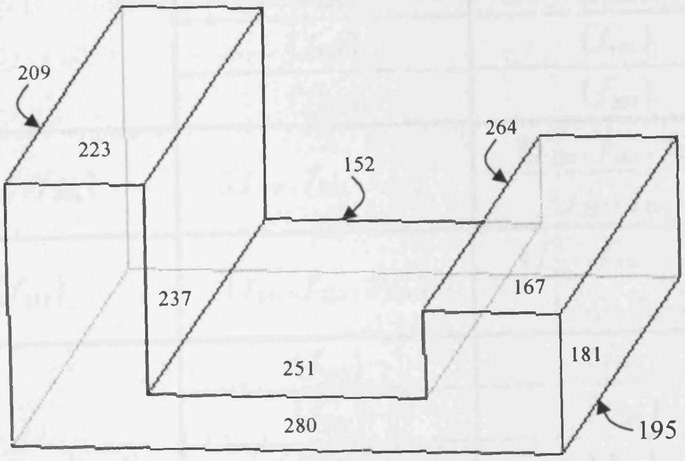


Figure 5.9 The solid model for case study V

Feature class	Set of seeds detected	Plant developed	Plant modified	Feature validated
ho_th	$\{f_{195}, f_{167}, f_{209}, f_{223}, f_{152}, f_{280}, f_{181}\}$	$\{f_{152}\}$	$\{f_{152}\}$	no
		$\{f_{280}\}$	$\{f_{280}\}$	no
		$\{f_{181}\}$	$\{f_{181}\}$	no
		$\{f_{167}\}$	$\{f_{167}\}$	no
		$\{f_{223}\}$	$\{f_{223}\}$	no
		$\{f_{195}\}$	$\{f_{195}\}$	no
		$\{f_{209}\}$	$\{f_{209}\}$	no
sl_th	$\{f_{237}, f_{264}\}$	$\{f_{237}, f_{264}, f_{251}\}$	$\{f_{237}^{V}, f_{264}, f_{251}\}$	yes
			$\{f_{237}^{V}, f_{167}\}$	no
sl_nt	$\{f_{251}\}$	$\{f_{251}, f_{237}, f_{264}\}$	$\{f_{237}^{V}, f_{264}, f_{251}\}$	no
			$\{f_{237}^{V}, f_{167}\}$	no
st	$\{f_{195}, f_{167}, f_{209}, f_{251}, f_{223}, f_{152}, f_{237}, f_{280}, f_{264}, f_{181}\}$	$\{f_{152}\}$	$\{f_{152}\}$	no
		$\{f_{280}\}$	$\{f_{280}\}$	no
		$\{f_{181}\}$	$\{f_{181}\}$	no
		$\{f_{195}\}$	$\{f_{195}\}$	no
		$\{f_{237}, f_{264}, f_{251}\}$	$\{f_{237}^{V}, f_{264}, f_{251}\}$	no
			$\{f_{237}^{V}, f_{167}\}$	yes
		$\{f_{223}\}$	$\{f_{223}\}$	no
		$\{f_{167}\}$	$\{f_{167}\}$	no
		$\{f_{209}\}$	$\{f_{209}\}$	no

Table 5.6 Results for case study V

5.5.6 Case study VI

This case study focuses on a type VI interaction, which is illustrated in Figure 5.10. The solid model includes two through slots, $\{f_{412}^{II}, f_{532}^{II}, f_{502}^{II}, f_{444}^{II}, f_{311}^{VI}\}$ and $\{f_{399}^{II}, f_{457}^{II}, f_{489}^{II}, f_{325}^{II}, f_{311}^{VI}\}$. The original side faces of both simple features are split by the interaction. In addition, the original bottom faces of the simple features are merged into one face, f_{311}^{VI}.

Table 5.7 shows the results of the recognition process. For the through slot feature class, one plant was constructed and then, f_{311} was identified in it as a type VI face. Next, the type II faces adjacent to f_{311} that lie on the same surface and that have the same orientation were grouped together. It was assumed that the opposing and parallel faces should be part of the same face set and this led to the division of the original plant into two different face sets $\{f_{412}^{II}, f_{532}^{II}, f_{502}^{II}, f_{444}^{II}, f_{311}^{VI}\}$ and $\{f_{399}^{II}, f_{457}^{II}, f_{489}^{II}, f_{325}^{II}, f_{311}^{VI}\}$. During the feature validation sub-process, each pair of type II faces was regarded as a single entity and thus, both through slots were recognised.

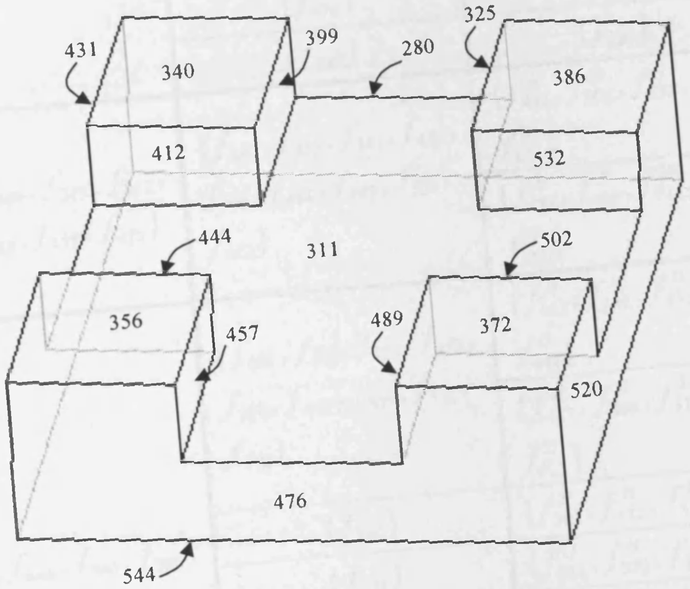


Figure 5.10 The solid model for case study VI

Feature class	Set of seeds detected	Plant developed	Plant modified	Feature validated
ho_th	$\{f_{340}, f_{386}, f_{280}, f_{476}, f_{431}, f_{544}, f_{356}, f_{372}, f_{520}\}$	$\{f_{520}\}$	$\{f_{520}\}$	no
		$\{f_{431}\}$	$\{f_{431}\}$	no
		$\{f_{544}\}$	$\{f_{544}\}$	no
		$\{f_{386}\}$	$\{f^{II}_{386}, f^{II}_{372}, f^{II}_{356}, f^{II}_{340}\}$	no
		$\{f_{340}\}$	$\{f^{II}_{386}, f^{II}_{372}, f^{II}_{356}, f^{II}_{340}\}$	no
		$\{f_{356}\}$	$\{f^{II}_{386}, f^{II}_{372}, f^{II}_{356}, f^{II}_{340}\}$	no
		$\{f_{372}\}$	$\{f^{II}_{386}, f^{II}_{372}, f^{II}_{356}, f^{II}_{340}\}$	no
		$\{f_{476}\}$	$\{f_{476}\}$	no
		$\{f_{280}\}$	$\{f_{280}\}$	no
sl_th	$\{f_{444}, f_{489}, f_{325}, f_{532}, f_{399}, f_{502}, f_{457}, f_{412}\}$	$\{f_{489}, f_{325}, f_{311}, f_{457}, f_{399}, f_{444}, f_{502}, f_{532}, f_{412}\}$	$\{f^{II}_{412}, f^{II}_{502}, f^{II}_{532}, f^{VI}_{311}, f^{II}_{444}\}$	**yes**
			$\{f^{II}_{457}, f^{II}_{489}, f^{VI}_{311}, f^{II}_{399}, f^{II}_{325}\}$	**yes**
st	$\{f_{311}, f_{444}, f_{340}, f_{386}, f_{489}, f_{325}, f_{532}, f_{280}, f_{476}, f_{431}, f_{544}, f_{399}, f_{502}, f_{457}, f_{356}, f_{372}, f_{412}, f_{520}\}$	$\{f_{489}, f_{311}, f_{325}, f_{457}, f_{399}, f_{444}, f_{502}, f_{532}, f_{412}\}$	$\{f^{II}_{412}, f^{II}_{502}, f^{II}_{532}, f^{VI}_{311}, f^{II}_{444}\}$	no
			$\{f^{II}_{457}, f^{II}_{489}, f^{VI}_{311}, f^{II}_{399}, f^{II}_{325}\}$	no
		$\{f_{340}\}$	$\{f^{II}_{386}, f^{II}_{372}, f^{II}_{356}, f^{II}_{340}\}$	no
		$\{f_{372}\}$	$\{f^{II}_{386}, f^{II}_{372}, f^{II}_{356}, f^{II}_{340}\}$	no
		$\{f_{386}\}$	$\{f^{II}_{386}, f^{II}_{372}, f^{II}_{356}, f^{II}_{340}\}$	no
		$\{f_{280}\}$	$\{f_{280}\}$	no
		$\{f_{544}\}$	$\{f_{544}\}$	no
		$\{f_{356}\}$	$\{f^{II}_{386}, f^{II}_{372}, f^{II}_{356}, f^{II}_{340}\}$	no
		$\{f_{431}\}$	$\{f_{431}\}$	no
		$\{f_{476}\}$	$\{f_{476}\}$	no
		$\{f_{520}\}$	$\{f_{520}\}$	no

Table 5.7 Results for case study VI

5.5.7 Case study VII

The test part shown in Figure 5.11 has been used by Nezis and Vosniakos (1997) to validate their AFR method. In this study, the same part is employed for benchmarking purposes. This test part is composed of eleven features that can be described according to the type of interaction affecting them as follows:

- □ No interaction. This is the case with one through slot $\{f_{1384}, f_{1372}, f_{1359}\}$, one rectangular protrusion $\{f_{1022}, f_{1036}, f_{1049}, f_{1008}, f_{1061}\}$, one non-through slot $\{f_{1300}, f_{1287}, f_{1273}, f_{1312}\}$, and two other features that are not present in the adopted taxonomy. These latter two features can be identified as one passage $\{f_{1675}, f_{1688}, f_{1701}, f_{1713}, f_{1662}\}$ and one corner $\{f_{1636}, f_{1623}, f_{1648}\}$.

- □ Type I interaction. Such an interaction affects two through slots, $\{f_{1206}, f_{1187}, f_{1174}\}$ and $\{f_{875}, f_{1082}, f_{1134}\}$ and one non-through slot $\{f_{1220}, f_{1234}, f_{1247}, f_{1259}\}$.

- □ Type II interaction. The through slot $\{f^{II}_{1593}, f^{II}_{1581}, f^{II}_{1556}, f^{II}_{1569}, f^{II}_{1412}, f^{II}_{1398}\}$ is altered as a result of such an interaction.

- □ Type IV interaction. Such an interaction affects two rectangular pockets $\{f^{IV}_{1475}, f_{1543}, f_{1433}, f^{IV}_{1451}, f_{1492}\}$ and $\{f^{IV}_{1475}, f_{1531}, f_{1506}, f^{IV}_{1451}, f_{1519}\}$.

The results of the recognition process obtained in regard to these features are shown in Table 5.8. A complete report of the results for this test part is given in Appendix D.

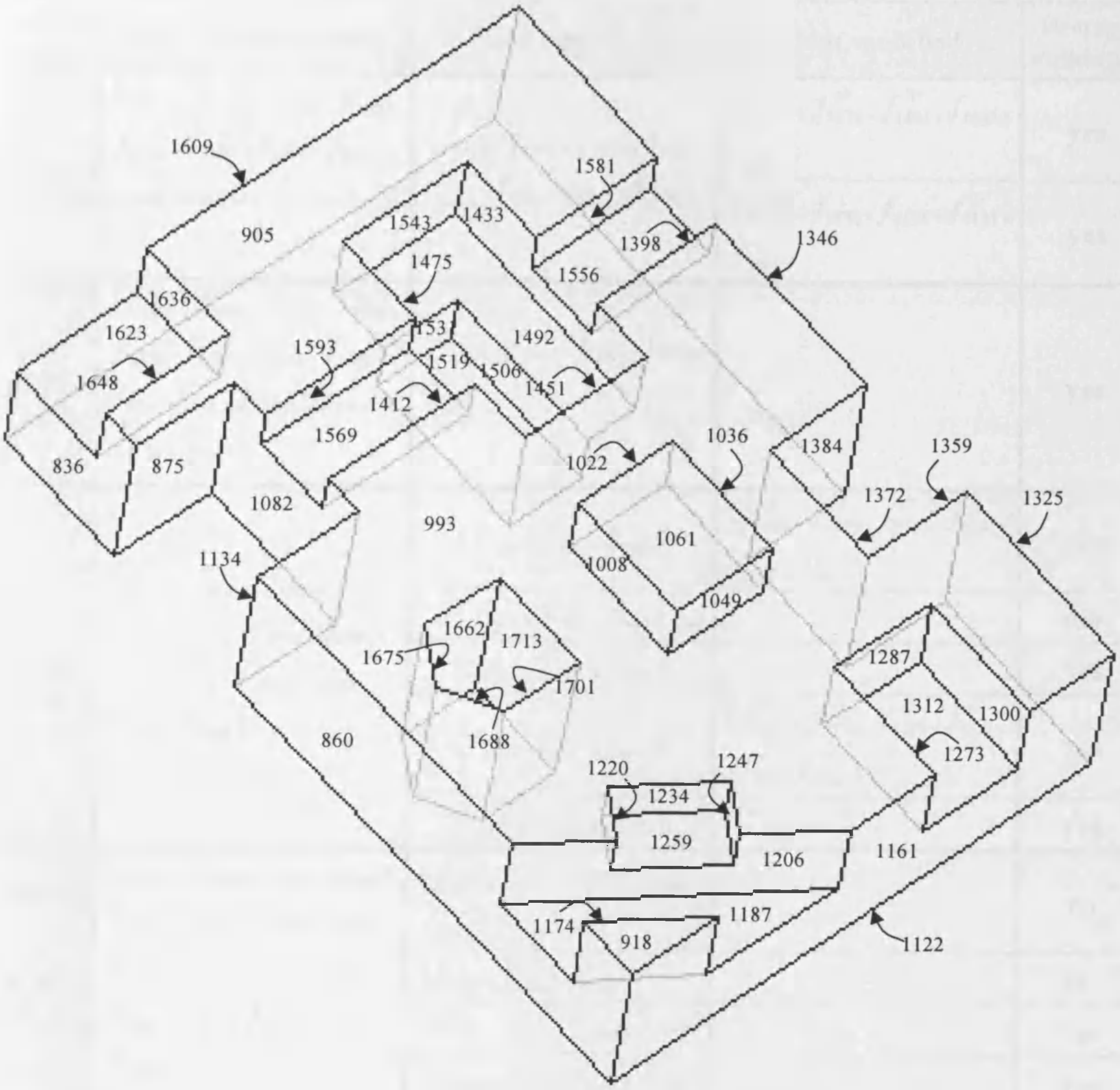


Figure 5.11 The solid model for the case study VII (Nezis and Vosniakos, 1997)

Feature class	Set of seeds detected	Plant developed	Plant modified	Feature validated
po_re	$\{f_{1234}, f_{1412}, f_{1273}, f_{1300}, f_{1220}, f_{1287}, f_{1543}, f_{1247}, f_{1433}, f_{1475}\}$	$\{f_{1475}, f_{1492}, f_{1543}, f_{1531}, f_{1451}, f_{1506}, f_{1519}, f_{1433}\}$	$\{f_{1492}, f_{1475}^{IV}, f_{1451}^{IV}, f_{1543}, f_{1433}\}$	yes
			$\{f_{1475}^{IV}, f_{1531}, f_{1506}, f_{1451}^{IV}, f_{1519}\}$	yes
pr_re	$\{f_{1412}, f_{1022}, f_{1593}, f_{1008}, f_{1398}, f_{1174}, f_{1581}, f_{1384}, f_{875}, f_{918}, f_{1359}, f_{1036}, f_{1134}, f_{1049}\}$	$\{f_{1036}, f_{1049}, f_{1061}, f_{1008}, f_{1022}\}$		yes
sl_th	$\{f_{1412}, f_{1022}, f_{1593}, f_{1008}, f_{1398}, f_{1174}, f_{1581}, f_{1384}, f_{875}, f_{918}, f_{1359}, f_{1036}, f_{1134}, f_{1049}\}$	$\{f_{1581}, f_{1556}, f_{1398}\}$	$\{f_{1569}^{II}, f_{1593}^{II}, f_{1412}^{II}, f_{1581}^{II}, f_{1556}^{II}, f_{1398}^{II}\}$	yes
		$\{f_{1134}, f_{1082}, f_{875}\}$		yes
		$\{f_{1384}, f_{1372}, f_{1359}\}$		yes
		$\{f_{1569}, f_{1593}, f_{1412}\}$	$\{f_{1569}^{II}, f_{1593}^{II}, f_{1412}^{II}, f_{1581}^{II}, f_{1556}^{II}, f_{1398}^{II}\}$	yes
		$\{f_{1187}, f_{1206}, f_{1174}\}$		yes
sl_nt	$\{f_{1713}, f_{1688}, f_{1273}, f_{1556}, f_{1623}, f_{1636}, f_{1300}, f_{1701}, f_{1372}, f_{1220}, f_{1187}, f_{1247}, f_{1648}, f_{1675}, f_{1082}, f_{1662}, f_{1569}\}$	$\{f_{1675}, f_{1713}, f_{1701}, f_{1688}, f_{1662}\}$		no
		$\{f_{1623}, f_{1648}, f_{1636}\}$		no
		$\{f_{1287}, f_{1273}, f_{1300}, f_{1312}\}$		yes
		$\{f_{1234}, f_{1259}, f_{1247}, f_{1220}\}$		yes

Table 5.8 Results for case study VII

It is important to note that nine of the eleven features in this part could be recognised. For the features that were not recognised, the corner and the passage, plants were constructed but failed to be validated because such feature classes were not defined in the taxonomy used in this study.

5.5.8 Case study VIII

The test part used in this case study, shown in Figure 5.12, has also been used by other researchers for validation purposes (Gupta et al., 1994; Regli, 1995). The STEP file of this part was downloaded from the National Design Repository (2004). However, some of the entity names used in this file are not supported by the STEP parsers employed to process the data. These parsers were developed in compliance with the STEP AP203 format. Thus, to avoid modifying the parsers, the part was redesigned using Pro/Engineer™ and another version of this STEP file was generated.

This test part is composed of twelve features that can be described according to the type of interaction affecting them as follows:

- □ No interaction. This is the case for two steps, $\{f_{819}, f_{795}\}$ and $\{f_{603}, f_{807}\}$, and two through holes, $\{f_{697}, f_{682}\}$ and $\{f_{1098}, f_{1100}\}$.
- □ Type I interaction. Such an interaction exists between one step feature $\{f_{981}, f_{940}\}$ and two non-through slots, $\{f_{1019}, f_{968}, f_{953}, f_{741}\}$ and $\{f_{1007}, f_{995}, f_{725}, f_{758}\}$.

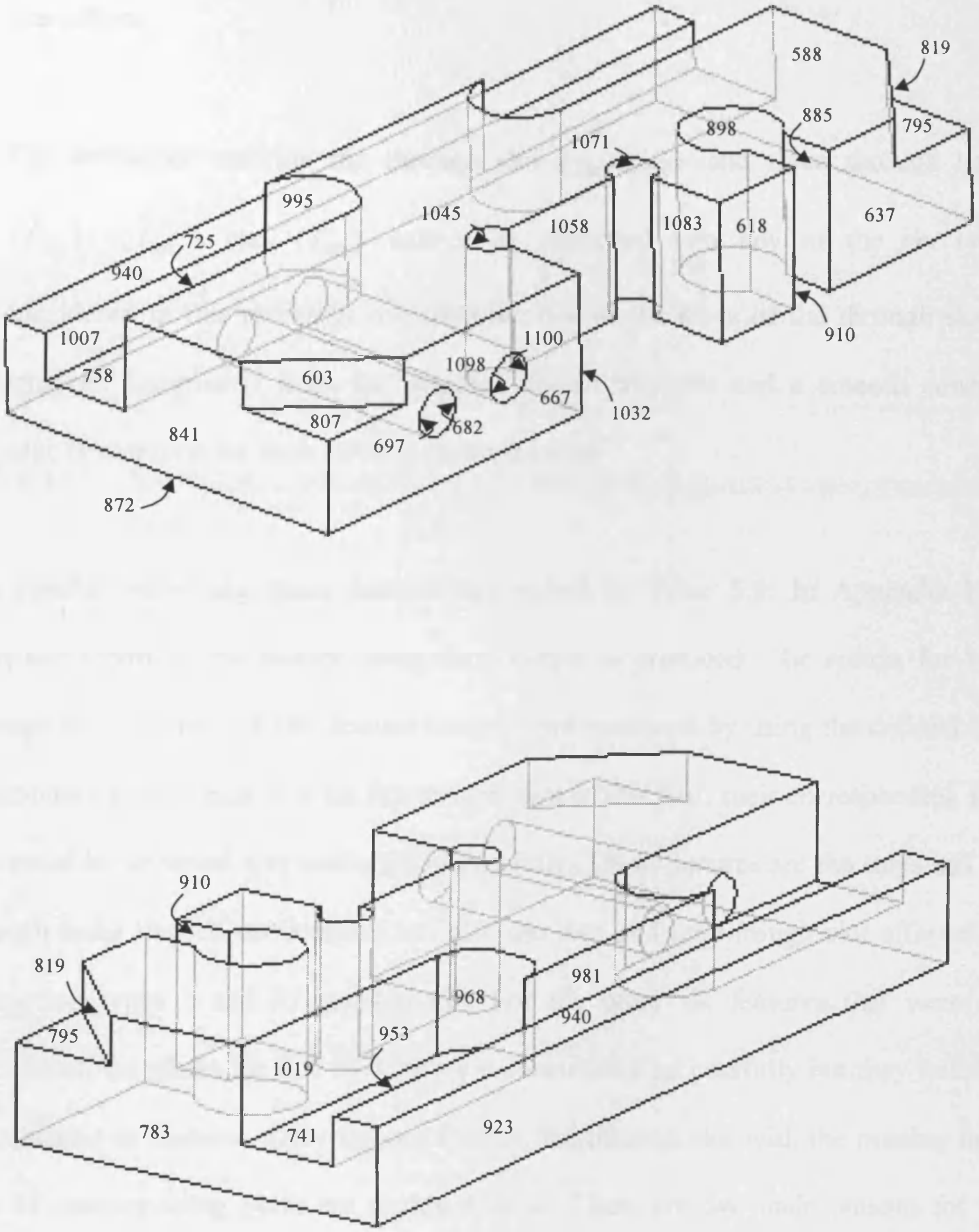


Figure 5.12 Two views of the solid model for case study VIII (Gupta et al., 1994; Regli, 1995)

- Type III interaction. The through slot $\{f_{1083}^{III}, f_{1058}^{III}, f_{1032}^{III}\}$ is affected by such an interaction.

- The interaction altering the through slot $\{f_{885}, f_{910}\}$ and three through holes $\{f_{1071}\}$, $\{f_{1045}\}$ and $\{f_{898}\}$ cannot be classified into any of the six types considered in this research. In particular, one of the faces of the through slot is removed completely from the original feature structure and a smooth concave edge is removed for each of these through holes.

The results concerning these features are shown in Table 5.9. In Appendix E, a complete report of the feature recognition output is provided. The results for both through and non-through slot feature classes were produced by using the definition of a secondary feature hint. For six features present in this part, their corresponding face sets could be retrieved and validated successfully. These features are the steps and the through holes that do not interact and also the step and the through slot affected by interaction types I and III respectively. For the other six features that were not recognised, the plants for five of them were constructed successfully but they failed to be validated as features. Only for one feature, the through slot with the missing face, was its corresponding plant not retrieved at all. There are two main reasons for the results obtained for the features that were not recognised:

- A plant is not covered by the rule set. This is the case with both non-through slots that have a cylindrical face in their feature structure. This can be easily addressed by adding examples representing such features in the training set.

Feature class	Set of seeds detected	Plant developed	Plant modified	Feature validated
ho_th	$\{f_{1110}, f_{783}, f_{1032}, f_{673}, f_{1071}, f_{1045}, f_{841}, f_{618}, f_{682}, f_{697}, f_{923}, f_{1098}, f_{872}, f_{885}, f_{667}, f_{588}, f_{1058}, f_{910}, f_{898}, f_{1083}\}$	$\{f_{1045}\}$		no
		$\{f_{898}\}$		no
		$\{f_{1100}, f_{1098}\}$		yes
		$\{f_{682}, f_{697}\}$		yes
		$\{f_{1071}\}$		no
sl_th	$\{f_{783}, f_{741}, f_{807}, f_{725}, f_{1032}, f_{637}, f_{819}, f_{1007}, f_{603}, f_{758}, f_{841}, f_{618}, f_{923}, f_{795}, f_{872}, f_{885}, f_{667}, f_{588}, f_{1058}, f_{910}, f_{940}, f_{981}, f_{1083}, f_{1019}, f_{953}\}$	$\{f_{1032}\}$	$\{f_{1032}^{III}, f_{1058}^{III}, f_{1083}^{III}\}$	yes
		$\{f_{1058}\}$	$\{f_{1032}^{III}, f_{1058}^{III}, f_{1083}^{III}\}$	yes
		$\{f_{1083}\}$	$\{f_{1032}^{III}, f_{1058}^{III}, f_{1083}^{III}\}$	yes
		$\{f_{885}\}$		no
		$\{f_{910}\}$		no
sl_nt	$\{f_{1110}, f_{807}, f_{1007}, f_{682}, f_{795}, f_{1019}, f_{953}\}$	$\{f_{725}, f_{1007}, f_{758}, f_{995}\}$		no
		$\{f_{953}, f_{741}, f_{1019}, f_{968}\}$		no
st	$\{f_{783}, f_{741}, f_{807}, f_{725}, f_{1032}, f_{637}, f_{819}, f_{1007}, f_{603}, f_{758}, f_{841}, f_{618}, f_{923}, f_{795}, f_{872}, f_{885}, f_{667}, f_{588}, f_{1058}, f_{910}, f_{940}, f_{981}, f_{1083}, f_{1019}, f_{953}\}$	$\{f_{981}, f_{940}\}$		yes
		$\{f_{807}, f_{603}\}$		yes
		$\{f_{795}, f_{819}\}$		yes

Table 5.9 Results for case study VIII

- The faces affected by interactions are not identified. This is the case with three through holes, each of which is altered by the loss a smooth concave edge. This problem also arises for a through slot as one of its faces is removed completely. To address this issue, it is necessary to extend the range of considered interaction types. The proposed AFR system has an open architecture and easily could accommodate additional cases of feature interactions.

5.6 Summary

This chapter has presented some solutions to extend the capabilities of the AFR method introduced in Chapter 4. They overcome some of its limitations when it is applied for recognising interacting features. The proposed solutions are the result of a critical analysis of the method sub-processes when they are used for recognising simple features that interact. In particular, the capability of these sub-processes to perform adequately when different types of interactions are present has been assessed. In addition, the proposed modifications to the AFR method have been implemented and verified on eight case studies representing different types of interactions.

The contribution of this research lies in the development of a geometric reasoning mechanism to tackle the recognition problems associated with interacting features by applying the proposed AFR method. In particular, the faces in a plant that could be affected by such interactions are detected and this triggers modifications in the plant structure that lead to the formation of new face sets corresponding to potential simple features. It is important to note that the solutions suggested in this chapter develop further the main idea behind the proposed AFR method by combining the 'learning from examples' concept with the rule-based and hint-based feature recognition approaches.

The reported results are restricted to recognising interactions between planar faces. However, they prove the feasibility of the proposed approach and suggest that the method could be extended to include other types of face geometry. These results also show that the feature hints, defined by applying the method proposed in Chapter 4, are suitable for detecting seeds that indicate the existence of simple features despite the face alterations caused by possible interactions between them. Finally, the recognition results obtained for both benchmarking parts are similar to those achieved by other researchers. However, the proposed AFR method has some advantages over other techniques. In particular, the method could be deployed in different application domains and the knowledge base that determines the performance of the developed AFR systems could be easily updated to broaden their application areas.

Chapter 6 - Contributions, conclusions and future work

6.1 Overview

This chapter discusses the main contributions of this research, presents the most important conclusions and suggests directions for future work.

6.2 Contributions

Following the review of existing AFR approaches in Chapter 2, it was concluded that the main knowledge gap that this research should address is the development of AFR methods that are domain independent. In this context, this work is an original contribution to the field of automatic feature recognition. A new AFR method that could be applied in different application domains is proposed. In particular, to achieve this, the following contributions are made to the current state-of-the-art in this field:

1. A method for automatic generation of feature recognition rules that is applicable in different application domains is proposed. This is a new method for creating knowledge bases of AFR systems that eliminates a major deficiency of rules-based AFR techniques. The two most important characteristics of this method are:

   - The application of inductive learning techniques for identification of hidden patterns in sets of feature examples.

- □ The utilisation of two representation schemes that code feature information at different levels of abstraction to complement and extend the learning capabilities of this method.

2. A hybrid AFR method that combines the 'learning from examples' concept with the rule-based and hint-based AFR approaches is developed. In particular:

   - □ A technique for defining feature hints automatically is proposed to address one of the main deficiencies of hint-based approaches. This technique makes possible the effective application of the hint concept in different application domains.

   - □ The feature recognition process is considered analogous to the growth of a vegetal plant. A face set defining a feature is constructed in stages from a seed, a face, representing a hint for the existence of this feature. This process employs sequentially, a set of hints, two rule sets and a geometric reasoning algorithm to construct valid features from the geometrical and topological information stored in B-Rep part models.

3. A geometric reasoning mechanism is developed to extend the capabilities of the proposed AFR method for recognising interacting features. It is a technique for detecting faces in a potential feature that could be affected by interactions. Then, depending on the type of the identified interaction, a face set is modified or split into more face sets that potentially could represent simple features.

The capabilities of the proposed AFR method to recognise simple and interacting machining features are verified. This is achieved by:

- Implementing the learning and feature recognition processes of this method into a prototype system.

- Defining a taxonomy of machining features and two schemes for coding geometrical and topological information required to generate feature recognition rules and hints.

- Testing the recognition performance of the developed prototype AFR system on three benchmark parts.

6.3 Conclusions

The application of inductive learning techniques for AFR has several advantages:

- It elevates the knowledge acquisition issues associated with the development of rule-based AFR systems. The application of the 'learning from examples' concept provides a formal and automatic mechanism for rule definition and also assures the consistency of the generated rule sets.

- The development of AFR systems for different application domains requires only representative training sets to be formed for each of them. This is a major advantage of the proposed approach due to the domain-dependent nature of features.

- Due to the generalisation capabilities of inductive learning techniques, AFR systems could recognise features that are not present in the training sets.

- The knowledge base of such systems could be extended easily to cover new or user-defined features.

In addition, the following conclusions are also drawn from this research:

- The generation of rule sets at two levels of abstraction offers flexibility in adopting different recognition strategies in AFR systems. The search for features could be carried out by utilising data present in individual faces or face sets.

- The creation of a rule set that represents patterns associated with individual feature faces is particularly suitable for the application of the hint concept. This is due to the fact that each hint derived from such rules represents only a hypothesis for the existence of a feature in a part.

- The utilisation of the hint concept is initially suggested in this research to speed up the exhaustive search for features that is carried out by rule-based AFR systems. It could be argued that given the computing power available today, the application of this concept does not bring an important advantage to the proposed AFR method. However, such an argument is not valid any more when the problems associated with the recognition of interacting features are considered. In particular, the application of this concept is very important to detect the existence of features that are altered as a result of feature interactions.

- □ Traditional rule-based AFR systems cannot handle feature interactions adequately because it may be necessary to define rules for all possible types of feature interactions. This research shows that by combining different AFR approaches and, at the same time, by complementing them with geometric reasoning mechanisms, rules can be successfully employed for recognising interacting features.

6.4 Future work

The proposed AFR method was implemented only for a taxonomy of machining features. More work is required to verify its recognition capabilities in other domains such as layer-based manufacturing and injection moulding for example.

Together with the automatically generated rule sets and feature hints, the geometric reasoning algorithms are an important component of the proposed AFR method. Thus, to apply it successfully in different domains, it is required generic geometric reasoning algorithms to be developed. The algorithm implemented in Chapter 4 can be used to recognise generic features such as protrusions or depressions, however further work is required to handle free-form surface features.

Finally, a comprehensive description of all possible types of feature interactions in a given domain is required in order to apply the proposed method successfully for such features. The different face alterations caused by feature interactions have to be studied further in order to develop a geometric reasoning mechanism for identifying the constituent simple features in solid models of parts.

Appendix A - Structure of the STEP standard

This appendix provides a concise description of the structure of the STEP Standard for the Exchange of Product Model Data. It is based on texts including (Owen, 1993), (Pratt, 2001) and (Nell, 2004).

STEP, developed by ISO TC184/SC4, is the familiar name for ISO 10303. It is an international effort towards the definition of a standard for describing product data throughout the life cycle of a product that is independent of any particular computer system. An interesting characteristic of STEP is that it provides not only a representation of product-related information but also the mechanisms and definitions to enable product data exchange and sharing. Its development started in 1984 when the need for producing a single international standard was recognised due to the identification of deficiencies in the existing product data standards.

Early efforts have led to the division of the standard into a number of classes of parts. Each part in the different classes has its own status, which can vary from the ISO preliminary stage status to the acceptance as an international standard. The different parts of STEP fall into one of the following classes:

- □ Description methods (Part 1-14). This class provides the standardised methods to describe the STEP entities. The part 11 is the EXPRESS language reference manual, which describes the data-modelling language that is employed in STEP.

- Implementation methods (Parts 21-29). The parts in this class support the development of software implementation of the standard. This class contains the part 21 (Clear text encoding of the exchange structure), that specifies how physical files should be written. The syntax of a physical file is formally defined and it has a specified alphabet and tokens, which enable it to be parsed. Part 21 also contains a formal mapping from EXPRESS to the file structure, which dictates how an instance of any EXPRESS schema will appear in a physical file.

- Conformance testing methodology and framework (Parts 31-35). The parts in this class specify the standard procedures and tools required in testing an implementation of ISO 10303 for conformance to the standard.

- Integrated generic resources (Parts 41-58). This class provides information models of general applicability that are used to build the application protocols (see Parts 201-240).

- Integrated application resources (Parts 101-110). The resources described in this class are slightly more specialised than the integrated generic resources. They can support a single application or a range of similar applications.

- Application protocols (APs) (Parts 201-240). This class specifies the information needs in specific engineering applications. APs give context and constraints to the information resources to represent a particular data model of some stages of a product life. The application protocol AP 203 (Configuration-controlled design) used in this research is concerned with the transfer of product shape models, assembly structure and configuration control information (e.g. part versioning, etc.).

- Abstract test suites (Parts 301-336). Each application protocol has an associated abstract test suite, which consists of test data and criteria to be used in assessing the conformance of a software implementation of an AP.

- Application interpreted constructs (AICs) (Parts 501-523). AICs are built from the integrated resources. They are reusable groups of information that are common in several APs.

- Application modules (Parts 1001-1414). Application modules have the same functionality as AICs. They are also designed to standardise the interpretation of the integrated resources but they extend the capabilities of the AICs.

Appendix B - An example of a STEP physical file

This appendix shows the STEP AP 203 file of one of the blind hole feature models used in Chapter 3. The file starts with the keyword ISO-10303-21 and is divided into a HEADER and a DATA section as follows:

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION((''),'2;1');
FILE_NAME('HO_BL1','2003-06-24T',('sceeb'),(''),
'PRO/ENGINEER BY PARAMETRIC TECHNOLOGY CORPORATION, 2001150',
'PRO/ENGINEER BY PARAMETRIC TECHNOLOGY CORPORATION,
2001150','');
FILE_SCHEMA(('CONFIG_CONTROL_DESIGN'));
ENDSEC;
DATA;
#1=DIRECTION('',(1.E0,0.E0,0.E0));
#2=VECTOR('',#1,5.E1);
#3=CARTESIAN_POINT('',(0.E0,0.E0,0.E0));
#4=LINE('',#3,#2);
#5=DIRECTION('',(0.E0,0.E0,-1.E0));
#6=VECTOR('',#5,5.E1);
#7=CARTESIAN_POINT('',(5.E1,0.E0,0.E0));
#8=LINE('',#7,#6);
#9=DIRECTION('',(-1.E0,0.E0,0.E0));
#10=VECTOR('',#9,5.E1);
#11=CARTESIAN_POINT('',(5.E1,0.E0,-5.E1));
#12=LINE('',#11,#10);
#13=DIRECTION('',(0.E0,0.E0,1.E0));
#14=VECTOR('',#13,5.E1);
#15=CARTESIAN_POINT('',(0.E0,0.E0,-5.E1));
#16=LINE('',#15,#14);
#17=DIRECTION('',(0.E0,1.E0,0.E0));
#18=VECTOR('',#17,5.E1);
#19=CARTESIAN_POINT('',(0.E0,0.E0,0.E0));
#20=LINE('',#19,#18);
#21=DIRECTION('',(0.E0,1.E0,0.E0));
#22=VECTOR('',#21,5.E1);
#23=CARTESIAN_POINT('',(0.E0,0.E0,-5.E1));
#24=LINE('',#23,#22);
#25=DIRECTION('',(0.E0,1.E0,0.E0));
#26=VECTOR('',#25,5.E1);
#27=CARTESIAN_POINT('',(5.E1,0.E0,-5.E1));
```

```
#28=LINE('',#27,#26);
#29=DIRECTION('',(0.E0,1.E0,0.E0));
#30=VECTOR('',#29,5.E1);
#31=CARTESIAN_POINT('',(5.E1,0.E0,0.E0));
#32=LINE('',#31,#30);
#33=DIRECTION('',(1.E0,0.E0,0.E0));
#34=VECTOR('',#33,5.E1);
#35=CARTESIAN_POINT('',(0.E0,5.E1,0.E0));
#36=LINE('',#35,#34);
#37=DIRECTION('',(0.E0,0.E0,1.E0));
#38=VECTOR('',#37,5.E1);
#39=CARTESIAN_POINT('',(0.E0,5.E1,-5.E1));
#40=LINE('',#39,#38);
#41=DIRECTION('',(-1.E0,0.E0,0.E0));
#42=VECTOR('',#41,5.E1);
#43=CARTESIAN_POINT('',(5.E1,5.E1,-5.E1));
#44=LINE('',#43,#42);
#45=DIRECTION('',(0.E0,0.E0,-1.E0));
#46=VECTOR('',#45,5.E1);
#47=CARTESIAN_POINT('',(5.E1,5.E1,0.E0));
#48=LINE('',#47,#46);
#49=CARTESIAN_POINT('',(2.5E1,5.E1,-2.5E1));
#50=DIRECTION('',(0.E0,-1.E0,0.E0));
#51=DIRECTION('',(1.E0,0.E0,0.E0));
#52=AXIS2_PLACEMENT_3D('',#49,#50,#51);
#54=CARTESIAN_POINT('',(2.5E1,5.E1,-2.5E1));
#55=DIRECTION('',(0.E0,-1.E0,0.E0));
#56=DIRECTION('',(-1.E0,0.E0,0.E0));
#57=AXIS2_PLACEMENT_3D('',#54,#55,#56);
#59=DIRECTION('',(0.E0,-1.E0,0.E0));
#60=VECTOR('',#59,3.E1);
#61=CARTESIAN_POINT('',(4.E1,5.E1,-2.5E1));
#62=LINE('',#61,#60);
#63=DIRECTION('',(0.E0,-1.E0,0.E0));
#64=VECTOR('',#63,3.E1);
#65=CARTESIAN_POINT('',(1.E1,5.E1,-2.5E1));
#66=LINE('',#65,#64);
#67=CARTESIAN_POINT('',(2.5E1,2.E1,-2.5E1));
#68=DIRECTION('',(0.E0,-1.E0,0.E0));
#69=DIRECTION('',(1.E0,0.E0,0.E0));
#70=AXIS2_PLACEMENT_3D('',#67,#68,#69);
#72=CARTESIAN_POINT('',(2.5E1,2.E1,-2.5E1));
#73=DIRECTION('',(0.E0,-1.E0,0.E0));
#74=DIRECTION('',(-1.E0,0.E0,0.E0));
#75=AXIS2_PLACEMENT_3D('',#72,#73,#74);
#77=CARTESIAN_POINT('',(0.E0,0.E0,0.E0));
#78=CARTESIAN_POINT('',(5.E1,0.E0,0.E0));
#79=VERTEX_POINT('',#77);
#80=VERTEX_POINT('',#78);
#81=CARTESIAN_POINT('',(5.E1,0.E0,-5.E1));
```

```
#82=VERTEX_POINT(",#81);
#83=CARTESIAN_POINT(",(0.E0,0.E0,-5.E1));
#84=VERTEX_POINT(",#83);
#85=CARTESIAN_POINT(",(0.E0,5.E1,0.E0));
#86=CARTESIAN_POINT(",(5.E1,5.E1,0.E0));
#87=VERTEX_POINT(",#85);
#88=VERTEX_POINT(",#86);
#89=CARTESIAN_POINT(",(5.E1,5.E1,-5.E1));
#90=VERTEX_POINT(",#89);
#91=CARTESIAN_POINT(",(0.E0,5.E1,-5.E1));
#92=VERTEX_POINT(",#91);
#93=CARTESIAN_POINT(",(4.E1,2.E1,-2.5E1));
#94=CARTESIAN_POINT(",(1.E1,2.E1,-2.5E1));
#95=VERTEX_POINT(",#93);
#96=VERTEX_POINT(",#94);
#97=CARTESIAN_POINT(",(4.E1,5.E1,-2.5E1));
#98=CARTESIAN_POINT(",(1.E1,5.E1,-2.5E1));
#99=VERTEX_POINT(",#97);
#100=VERTEX_POINT(",#98);
#101=CARTESIAN_POINT(",(0.E0,0.E0,0.E0));
#102=DIRECTION(",(0.E0,1.E0,0.E0));
#103=DIRECTION(",(1.E0,0.E0,0.E0));
#104=AXIS2_PLACEMENT_3D(",#101,#102,#103);
#105=PLANE(",#104);
#107=ORIENTED_EDGE(",*,*,#106,.T.);
#109=ORIENTED_EDGE(",*,*,#108,.T.);
#111=ORIENTED_EDGE(",*,*,#110,.T.);
#113=ORIENTED_EDGE(",*,*,#112,.T.);
#114=EDGE_LOOP(",(#107,#109,#111,#113));
#115=FACE_OUTER_BOUND(",#114,.F.);
#116=ADVANCED_FACE(",(#115),#105,.F.);
#117=CARTESIAN_POINT(",(0.E0,0.E0,0.E0));
#118=DIRECTION(",(0.E0,0.E0,1.E0));
#119=DIRECTION(",(1.E0,0.E0,0.E0));
#120=AXIS2_PLACEMENT_3D(",#117,#118,#119);
#121=PLANE(",#120);
#122=ORIENTED_EDGE(",*,*,#106,.F.);
#124=ORIENTED_EDGE(",*,*,#123,.T.);
#126=ORIENTED_EDGE(",*,*,#125,.T.);
#128=ORIENTED_EDGE(",*,*,#127,.F.);
#129=EDGE_LOOP(",(#122,#124,#126,#128));
#130=FACE_OUTER_BOUND(",#129,.F.);
#131=ADVANCED_FACE(",(#130),#121,.T.);
#132=CARTESIAN_POINT(",(0.E0,0.E0,-5.E1));
#133=DIRECTION(",(-1.E0,0.E0,0.E0));
#134=DIRECTION(",(0.E0,0.E0,1.E0));
#135=AXIS2_PLACEMENT_3D(",#132,#133,#134);
#136=PLANE(",#135);
#137=ORIENTED_EDGE(",*,*,#112,.F.);
#139=ORIENTED_EDGE(",*,*,#138,.T.);
```

```
#141=ORIENTED_EDGE('',*,*,#140,.T.);
#142=ORIENTED_EDGE('',*,*,#123,.F.);
#143=EDGE_LOOP('',(#137,#139,#141,#142));
#144=FACE_OUTER_BOUND('',#143,.F.);
#145=ADVANCED_FACE('',(#144),#136,.T.);
#146=CARTESIAN_POINT('',(5.E1,0.E0,-5.E1));
#147=DIRECTION('',(0.E0,0.E0,-1.E0));
#148=DIRECTION('',(-1.E0,0.E0,0.E0));
#149=AXIS2_PLACEMENT_3D('',#146,#147,#148);
#150=PLANE('',#149);
#151=ORIENTED_EDGE('',*,*,#110,.F.);
#153=ORIENTED_EDGE('',*,*,#152,.T.);
#155=ORIENTED_EDGE('',*,*,#154,.T.);
#156=ORIENTED_EDGE('',*,*,#138,.F.);
#157=EDGE_LOOP('',(#151,#153,#155,#156));
#158=FACE_OUTER_BOUND('',#157,.F.);
#159=ADVANCED_FACE('',(#158),#150,.T.);
#160=CARTESIAN_POINT('',(5.E1,0.E0,0.E0));
#161=DIRECTION('',(1.E0,0.E0,0.E0));
#162=DIRECTION('',(0.E0,0.E0,-1.E0));
#163=AXIS2_PLACEMENT_3D('',#160,#161,#162);
#164=PLANE('',#163);
#165=ORIENTED_EDGE('',*,*,#108,.F.);
#166=ORIENTED_EDGE('',*,*,#127,.T.);
#168=ORIENTED_EDGE('',*,*,#167,.T.);
#169=ORIENTED_EDGE('',*,*,#152,.F.);
#170=EDGE_LOOP('',(#165,#166,#168,#169));
#171=FACE_OUTER_BOUND('',#170,.F.);
#172=ADVANCED_FACE('',(#171),#164,.T.);
#173=CARTESIAN_POINT('',(0.E0,5.E1,0.E0));
#174=DIRECTION('',(0.E0,1.E0,0.E0));
#175=DIRECTION('',(1.E0,0.E0,0.E0));
#176=AXIS2_PLACEMENT_3D('',#173,#174,#175);
#177=PLANE('',#176);
#178=ORIENTED_EDGE('',*,*,#125,.F.);
#179=ORIENTED_EDGE('',*,*,#140,.F.);
#180=ORIENTED_EDGE('',*,*,#154,.F.);
#181=ORIENTED_EDGE('',*,*,#167,.F.);
#182=EDGE_LOOP('',(#178,#179,#180,#181));
#183=FACE_OUTER_BOUND('',#182,.F.);
#185=ORIENTED_EDGE('',*,*,#184,.F.);
#187=ORIENTED_EDGE('',*,*,#186,.F.);
#188=EDGE_LOOP('',(#185,#187));
#189=FACE_BOUND('',#188,.F.);
#190=ADVANCED_FACE('',(#183,#189),#177,.T.);
#191=CARTESIAN_POINT('',(2.5E1,5.E1,-2.5E1));
#192=DIRECTION('',(0.E0,-1.E0,0.E0));
#193=DIRECTION('',(1.E0,0.E0,0.E0));
#194=AXIS2_PLACEMENT_3D('',#191,#192,#193);
#195=CYLINDRICAL_SURFACE('',#194,1.5E1);
```

```
#196=ORIENTED_EDGE(",*,*,#184,.T.);
#198=ORIENTED_EDGE(",*,*,#197,.T.);
#200=ORIENTED_EDGE(",*,*,#199,.F.);
#202=ORIENTED_EDGE(",*,*,#201,.F.);
#203=EDGE_LOOP(",(#196,#198,#200,#202));
#204=FACE_OUTER_BOUND(",#203,.F.);
#205=ADVANCED_FACE(",(#204),#195,.F.);
#206=CARTESIAN_POINT(",(2.5E1,5.E1,-2.5E1));
#207=DIRECTION(",(0.E0,-1.E0,0.E0));
#208=DIRECTION(",(1.E0,0.E0,0.E0));
#209=AXIS2_PLACEMENT_3D(",#206,#207,#208);
#210=CYLINDRICAL_SURFACE(",#209,1.5E1);
#211=ORIENTED_EDGE(",*,*,#186,.T.);
#212=ORIENTED_EDGE(",*,*,#201,.T.);
#214=ORIENTED_EDGE(",*,*,#213,.F.);
#215=ORIENTED_EDGE(",*,*,#197,.F.);
#216=EDGE_LOOP(",(#211,#212,#214,#215));
#217=FACE_OUTER_BOUND(",#216,.F.);
#218=ADVANCED_FACE(",(#217),#210,.F.);
#219=CARTESIAN_POINT(",(2.5E1,2.E1,-2.5E1));
#220=DIRECTION(",(0.E0,-1.E0,0.E0));
#221=DIRECTION(",(1.E0,0.E0,0.E0));
#222=AXIS2_PLACEMENT_3D(",#219,#220,#221);
#223=PLANE(",#222);
#224=ORIENTED_EDGE(",*,*,#199,.T.);
#225=ORIENTED_EDGE(",*,*,#213,.T.);
#226=EDGE_LOOP(",(#224,#225));
#227=FACE_OUTER_BOUND(",#226,.F.);
#228=ADVANCED_FACE(",(#227),#223,.F.);
#229=CLOSED_SHELL(",(#116,#131,#145,#159,#172,#190,#205,#218,#228));
#230=MANIFOLD_SOLID_BREP(",#229);
#231=DIMENSIONAL_EXPONENTS(1.E0,0.E0,0.E0,0.E0,0.E0,0.E0,0.E0);
#232=(LENGTH_UNIT()NAMED_UNIT(*)SI_UNIT(.MILLI.,.METRE.));
#233=LENGTH_MEASURE_WITH_UNIT(LENGTH_MEASURE(2.54E1),#232);
#234=(CONVERSION_BASED_UNIT('INCH',#233)LENGTH_UNIT()NAMED_U
NIT(#231));
#235=DIMENSIONAL_EXPONENTS(0.E0,0.E0,0.E0,0.E0,0.E0,0.E0,0.E0);
#236=(NAMED_UNIT(*)PLANE_ANGLE_UNIT()SI_UNIT($,.RADIAN.));
#237=PLANE_ANGLE_MEASURE_WITH_UNIT(PLANE_ANGLE_MEASURE(1
.745329251994E-2),#236);
#238=(CONVERSION_BASED_UNIT('DEGREE',#237)NAMED_UNIT(#235)PLA
NE_ANGLE_UNIT());
#239=(NAMED_UNIT(*)SI_UNIT($,.STERADIAN.)SOLID_ANGLE_UNIT());
#240=UNCERTAINTY_MEASURE_WITH_UNIT(LENGTH_MEASURE(8.65990
7627683E-3),#234,
'closure',
'Maximum model space distance between geometric entities at asserted
connectivities');
#241=(GEOMETRIC_REPRESENTATION_CONTEXT(3)GLOBAL_UNCERTAIN
TY_ASSIGNED_CONTEXT((
#240))GLOBAL_UNIT_ASSIGNED_CONTEXT((#234,#238,#239))REPRESENTA
TION_CONTEXT
('ID1','3'));
#243=APPLICATION_CONTEXT(
'CONFIGURATION CONTROLLED 3D DESIGNS OF MECHANICAL PARTS
AND ASSEMBLIES');
#244=APPLICATION_PROTOCOL_DEFINITION('international standard',
'config_control_design',1994,#243);
#245=DESIGN_CONTEXT(",#243,'design');
#246=MECHANICAL_CONTEXT(",#243,'mechanical');
#247=PRODUCT('HO_BL1','HO_BL1','NOT SPECIFIED',(#246));
#248=PRODUCT_DEFINITION_FORMATION_WITH_SPECIFIED_SOURCE('1','
LAST_VERSION',#247,
.MADE.);
#252=PRODUCT_CATEGORY('part',");
#253=PRODUCT_RELATED_PRODUCT_CATEGORY('detail',",(#247));
#254=PRODUCT_CATEGORY_RELATIONSHIP(",",#252,#253);
#255=SECURITY_CLASSIFICATION_LEVEL('unclassified');
#256=SECURITY_CLASSIFICATION(",",#255);
#257=CC_DESIGN_SECURITY_CLASSIFICATION(#256,(#248));
#258=APPROVAL_STATUS('approved');
#259=APPROVAL(#258,");
#260=CC_DESIGN_APPROVAL(#259,(#256,#248,#249));
#261=CALENDAR_DATE(103,24,6);
#262=COORDINATED_UNIVERSAL_TIME_OFFSET(2,0,.AHEAD.);
#263=LOCAL_TIME(18,34,2.8E1,#262);
#264=DATE_AND_TIME(#261,#263);
#265=APPROVAL_DATE_TIME(#264,#259);
#266=DATE_TIME_ROLE('creation_date');
#267=CC_DESIGN_DATE_AND_TIME_ASSIGNMENT(#264,#266,(#249));
#268=DATE_TIME_ROLE('classification_date');
#269=CC_DESIGN_DATE_AND_TIME_ASSIGNMENT(#264,#268,(#256));
#270=PERSON('UNSPECIFIED','UNSPECIFIED',$,$,$,$);
#271=ORGANIZATION('UNSPECIFIED','UNSPECIFIED','UNSPECIFIED');
#272=PERSON_AND_ORGANIZATION(#270,#271);
#273=APPROVAL_ROLE('approver');
#274=APPROVAL_PERSON_ORGANIZATION(#272,#259,#273);
#275=PERSON_AND_ORGANIZATION_ROLE('creator');
#276=CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT(#272,#27
5,(#248,#249));
#277=PERSON_AND_ORGANIZATION_ROLE('design_supplier');
#278=CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT(#272,#27
7,(#248));
#279=PERSON_AND_ORGANIZATION_ROLE('classification_officer');
#280=CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT(#272,#27
9,(#256));
#281=PERSON_AND_ORGANIZATION_ROLE('design_owner');
#282=CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT(#272,#28
1,(#247));
#53=CIRCLE(",#52,1.5E1);
```

```
#58=CIRCLE(",#57,1.5E1);
#71=CIRCLE(",#70,1.5E1);
#76=CIRCLE(",#75,1.5E1);
#106=EDGE_CURVE(",#79,#80,#4,.T.);
#108=EDGE_CURVE(",#80,#82,#8,.T.);
#110=EDGE_CURVE(",#82,#84,#12,.T.);
#112=EDGE_CURVE(",#84,#79,#16,.T.);
#123=EDGE_CURVE(",#79,#87,#20,.T.);
#125=EDGE_CURVE(",#87,#88,#36,.T.);
#127=EDGE_CURVE(",#80,#88,#32,.T.);
#138=EDGE_CURVE(",#84,#92,#24,.T.);
#140=EDGE_CURVE(",#92,#87,#40,.T.);
#152=EDGE_CURVE(",#82,#90,#28,.T.);
#154=EDGE_CURVE(",#90,#92,#44,.T.);
#167=EDGE_CURVE(",#88,#90,#48,.T.);
#184=EDGE_CURVE(",#99,#100,#53,.T.);
#186=EDGE_CURVE(",#100,#99,#58,.T.);
#197=EDGE_CURVE(",#100,#96,#66,.T.);
#199=EDGE_CURVE(",#95,#96,#71,.T.);
#201=EDGE_CURVE(",#99,#95,#62,.T.);
#213=EDGE_CURVE(",#96,#95,#76,.T.);
#242=ADVANCED_BREP_SHAPE_REPRESENTATION(",(#230),#241);
#249=PRODUCT_DEFINITION('design',",#248,#245);
#250=PRODUCT_DEFINITION_SHAPE(",'SHAPE FOR HO_BL1.',#249);
#251=SHAPE_DEFINITION_REPRESENTATION(#250,#242);
ENDSEC;
END-ISO-10303-21;
```

Appendix C - An example of a JavaCC™ grammar file

This appendix shows one of the grammar files developed in this research. Such a file is processed by JavacCC™ to generate a Java™ program that can parse a STEP physical file. In this example, Java™ code was developed for the purpose of extracting information about the entities ADVANCED_FACE within a STEP file (see Appendix B).

```
/*******************************************************************
 *
 * contents:     STEP Part 21 - clear text encoding parser to be used with JavaCC
 *
 * history :
 * --------
 *   18 Aug 1999: Creation. Singva Ma <Singva.Ma@leg.ensieg.inpg.fr>.
 *
 *   20 Jan 2004: Addition of Java™ code to extract information about the STEP
 *   entities ADVANCED_FACE. Emmanuel Brousseau<BrousseauE@cf.ac.uk>
 *
 *******************************************************************/

options {
        LOOKAHEAD = 3;
        STATIC = true;
        DEBUG_PARSER = false;
}

PARSER_BEGIN(AdvancedFaceParser)

package stepFileParsing.advancedFaceParser;

import java.io.*;
import java.util.*;
import topology.*;
import utilities.*;

public class AdvancedFaceParser{

        private static boolean foundAdvancedFace = false;
        private static boolean listFaceBound = false;
        private static AdvancedFace advancedFace;
        private static SolidModel sm;
        private static Token tag;
        private static Map afMap = new HashMap();
}

PARSER_END(AdvancedFaceParser)



/*************************************************
**************************************************
** Tokens
**************************************************
*************************************************/

SKIP : /* WHITE SPACE */
{
  " "
| "\t"
| "\n"
| "\r"
| "\f"
}

SPECIAL_TOKEN : /* COMMENTS */
{
 <EMBEDDED_REMARK: "/*" (~["*"])* "*" ("*" | (~["*",")"] (~["*"])* "*"))* "/">
}

TOKEN :
{
  < LPAREN: "(" >
| < RPAREN: ")" >
| < LBRACE: "{" >
| < RBRACE: "}" >
| < LBRACKET: "[" >
| < RBRACKET: "]" >
| < SEMICOLON: ";" >
| < COLON: ":" >
| < COMMA: "," >
| < DOT: "."  >
| < EQ: "=" >
| < DOLLAR: "$" >
| < STAR: "*">
| < SLASH: "/">
}

TOKEN : {
  <INTEGER: (<SIGN>)? <DIGIT> (<DIGIT>)*>
|  <KEYWORD: <USER_DEFINED_KEYWORD> | <STANDARD_KEYWORD>>
| <USER_DEFINED_KEYWORD: "!" <UPPER> (<UPPER> | <DIGIT>)*>
| <STANDARD_KEYWORD: <UPPER> (<UPPER> | <DIGIT>)*>
| <#SIGN: ["+", "-"]>
| <REAL: (<SIGN>)? <DIGIT> (<DIGIT>)* <DOT> (<DIGIT>)* ("E" (<SIGN>)?
<DIGIT> (<DIGIT>)* )?>
| <NON_Q_CHAR: <SPECIAL> | <DIGIT> | " " | <LOWER> | <UPPER>>
| <STRING: """ (<NON_Q_CHAR> | <APOSTROPHE><APOSTROPHE> |
<REVERSE_SOLIDUS><REVERSE_SOLIDUS> | <CONTROL_DIRECTIVE>)*
"""">
| <ENTITY_INSTANCE_NAME: "#" <DIGIT> (<DIGIT>)*>
| <ENUMERATION: <DOT> <UPPER> (<UPPER> | <DIGIT>)* <DOT>>
| <#HEX: ["0"-"9", "A"-"F"]>
| <BINARY: "\"" ( "0" | "1" | "2" | "3" ) (<HEX>)* "\"">
}

TOKEN : {
  <#DIGIT: ["0"-"9"]>
| <#LOWER: ["a"-"z"]>
| <#UPPER: ["A"-"Z", "_"]>
| <SPECIAL: "!" | "\"" | <STAR> | <DOLLAR> | "%" | "&" | <DOT> | "#" | "+" |
<COMMA> | "-" | <LPAREN> | <RPAREN> | "?" | <SLASH> | <COLON> |
<SEMICOLON> | "<" | <EQ> | ">" | "@" | <LBRACKET> | <RBRACKET> |
<LBRACE> | "|" | <RBRACE> | "^" | "`">
| <REVERSE_SOLIDUS: "\\">
| <APOSTROPHE: """>
| <CHARACTER: " " | <DIGIT> | <LOWER> | <UPPER> | <SPECIAL> |
<REVERSE_SOLIDUS> | <APOSTROPHE>>
}


TOKEN : {
  <CONTROL_DIRECTIVE: <PAGE> | <ALPHABET> | <EXTENDED2> |
<EXTENDED4> | <ARBITRARY> >
| <PAGE: <REVERSE_SOLIDUS> "S" <REVERSE_SOLIDUS>
<CHARACTER>>
| <ALPHABET: <REVERSE_SOLIDUS> "P" <UPPER> <REVERSE_SOLIDUS>>
| <EXTENDED2: <REVERSE_SOLIDUS> "X2" <REVERSE_SOLIDUS>
<HEX_TWO> (<HEX_TWO>)* <END_EXTENDED>>
| <EXTENDED4: <REVERSE_SOLIDUS> "X4" <REVERSE_SOLIDUS>
<HEX_FOUR> (<HEX_FOUR>)* <END_EXTENDED>>
| <END_EXTENDED: <REVERSE_SOLIDUS> "X0" <REVERSE_SOLIDUS>>
| <ARBITRARY: <REVERSE_SOLIDUS> "X" <REVERSE_SOLIDUS>
<HEX_ONE>>
| <HEX_ONE: <HEX> <HEX>>
| <HEX_TWO: <HEX_ONE> <HEX_ONE>>
| <HEX_FOUR: <HEX_TWO> <HEX_TWO>>
}
```

```
/**********************************************
 * Grammar
 **********************************************/

void exchange_file() : {}
{
  "ISO-10303-21;"
        header_section()
        data_section()
        "END-ISO-10303-21"
}


void header_section() : {}
{
  "HEADER;"
        header_entity() header_entity() header_entity()
        [header_entity_list()]
        "ENDSEC;"
}


void header_entity_list() : {}
{
  header_entity() (header_entity())*
}


void header_entity() : {}
{
 <KEYWORD>
 <LPAREN>
 [parameter_list()]
 <RPAREN>
 <SEMICOLON>
}


void parameter_list() : {}
{
  parameter() (<COMMA> parameter())*
}


void parameter() : {}
{
   typed_parameter()
        | untyped_parameter()
        | omitted_parameter()
}
```

```
void typed_parameter() : {}
{
 <KEYWORD> <LPAREN> parameter() <RPAREN>
}


void untyped_parameter() : { Token x = null; int i = 0; }
{
   <DOLLAR>
  | <INTEGER>
  | <REAL>
  | <STRING>
  | x = <ENTITY_INSTANCE_NAME>
        { if (foundAdvancedFace == true && listFaceBound == true) {
                /*
                *The first reference stored in this advancedFace object will be to a
                *FACE_OUTER_BOUND
                *The next reference, if present, will be to one or more FACE_BOUND
                *entity
                */
                advancedFace.setFaceBoundRef(x.image);
          }
          /*
        *If the next condition is true, it means that the parser is reading the reference
        *which points towards the geometry information for the ADVANCED_FACE
        *considered.
        */
          if (foundAdvancedFace == true && listFaceBound == false) {
                advancedFace.setSurfaceRef(x.image);
          }
        }
  | x = <ENUMERATION>
        { if (foundAdvancedFace == true) {
                /*
                *Here we get the information about the flag of the
                *ADVANCED_FACE entity.
                * x.image returns ".T." or ".F."
                */
                advancedFace.setSameSense((x.image).substring(1,2));
          }
        }
  | <BINARY>
  | list()
}


void omitted_parameter() : {}
{
 <STAR>
}


void list() : {}
{
 <LPAREN> parameter() (<COMMA> parameter())* <RPAREN>
 { listFaceBound = false; }

}


void data_section() : {}
{
 "DATA;" entity_instance_list() "ENDSEC;"
}


void entity_instance_list() : {}
{
  entity_instance() (entity_instance())*
}


void entity_instance() : {}
{
 simple_entity_instance() | complex_entity_instance()
}


void simple_entity_instance() : {}
{
 tag=<ENTITY_INSTANCE_NAME> <EQ> [scope()] simple_record()
<SEMICOLON>
}


void complex_entity_instance() :
{
  Token t;
}
{
 <ENTITY_INSTANCE_NAME> <EQ> [scope()] subsuper_record()
<SEMICOLON>
}


void scope() : {}
{
 "&SCOPE" entity_instance_list() "ENDSCOPE" [export_list()]
```

```
}


void export_list() : {}
{
 <SLASH> <ENTITY_INSTANCE_NAME> (<COMMA>
<ENTITY_INSTANCE_NAME>)* <SLASH>
}


void simple_record() : { Token x; }
{
 x = <KEYWORD>
 { if (x.image.equals("ADVANCED_FACE")) {
               foundAdvancedFace = true;
               advancedFace = new AdvancedFace(sm);
               advancedFace.setTag(tag.image);
               afMap.put(tag.image, advancedFace);
               listFaceBound = true;
       }
 }
 { if (x.image.equals("FACE_SURFACE")) {
               foundAdvancedFace = true;
               advancedFace = new AdvancedFace(sm);
               advancedFace.setTag(tag.image);
               afMap.put(tag.image, advancedFace);
               listFaceBound = true;
       }
 }

 <LPAREN> [ parameter_list() ] <RPAREN>

 {
       /*
       *The parser has reached the end of the line, we have to set the
       *boolean attribute foundAdvancedFace to false.
       */
       foundAdvancedFace = false;
 }
}


void subsuper_record() : {}
{
  <LPAREN> simple_record_list() <RPAREN>
}


void simple_record_list() : {}
{
  simple_record() (simple_record())*
}


Map parseAdvancedFace(SolidModel solidModel) : {}
{
 {
        sm = solidModel;
        afMap.clear();
 }
 exchange_file()
 {
        return afMap;
 }
}
```

Appendix D - Results for case study VII in Chapter 5

This appendix is composed of three tables that show the results of the recognition process carried out on the part used for case study VII in Chapter 5.

Feature class	Set of seeds detected	Plant developed	Plant modified	Feature validated
po_re	$\{f_{1234}, f_{1412}, f_{1273}, f_{1300}, f_{1220}, f_{1287}, f_{1543}, f_{1247}, f_{1433}, f_{1475}\}$	$\{f_{1234}, f_{1259}, f_{1247}, f_{1220}\}$		no
		$\{f_{1569}, f_{1593}, f_{1412}\}$	$\{f_{1569}^{II}, f_{1593}^{II}, f_{1412}^{II}, f_{1581}^{II}, f_{1556}^{II}, f_{1398}^{II}\}$	no
		$\{f_{1475}, f_{1492}, f_{1543}, f_{1531}, f_{1451}, f_{1506}, f_{1519}, f_{1433}\}$	$\{f_{1492}, f_{1475}^{IV}, f_{1451}^{IV}, f_{1543}, f_{1433}\}$	**yes**
			$\{f_{1475}^{IV}, f_{1531}, f_{1506}, f_{1451}^{IV}, f_{1519}\}$	**yes**
		$\{f_{1287}, f_{1273}, f_{1300}, f_{1312}\}$		no
po_ob	$\{\varnothing\}$			
ho_bl	$\{\varnothing\}$			
ho_th	$\{f_{1346}, f_{860}, f_{836}, f_{1161}, f_{1325}, f_{905}, f_{918}, f_{1122}, f_{1609}, f_{1061}\}$	$\{f_{918}\}$	$\{f_{1036}, f_{1049}, f_{905}^{II}, f_{1008}, f_{918}^{II}, f_{1022}, f_{993}^{II}\}$	no
		$\{f_{860}\}$	$\{f_{860}^{II}, f_{836}^{II}\}$	no
		$\{f_{836}\}$	$\{f_{860}^{II}, f_{836}^{II}\}$	no
		$\{f_{1161}\}$		no
		$\{f_{905}\}$	$\{f_{1036}, f_{1049}, f_{905}^{II}, f_{1008}, f_{918}^{II}, f_{1022}, f_{993}^{II}\}$	no
		$\{f_{1061}\}$		no
		$\{f_{1122}\}$		no
		$\{f_{1325}\}$	$\{f_{1346}^{II}, f_{1325}^{II}\}$	no
		$\{f_{1346}\}$	$\{f_{1346}^{II}, f_{1325}^{II}\}$	no
pr_ci	$\{\varnothing\}$			

Table D.1 Results for the feature classes po_re, po_ob, ho_bl, ho_th and pr_ci

Feature class	Set of seeds detected	Plant developed	Plant modified	Feature validated
pr_re	$\{f_{1412}, f_{1022}, f_{1593}, f_{1008}, f_{1398}, f_{1174}, f_{1581}, f_{1384}, f_{875}, f_{918}, f_{1359}, f_{1036}, f_{1134}, f_{1049}\}$	$\{f_{1036}, f_{1049}, f_{1061}, f_{1008}, f_{1022}\}$		yes
sl_th	$\{f_{1412}, f_{1022}, f_{1593}, f_{1008}, f_{1398}, f_{1174}, f_{1581}, f_{1384}, f_{875}, f_{918}, f_{1359}, f_{1036}, f_{1134}, f_{1049}\}$	$\{f_{1581}, f_{1556}, f_{1398}\}$	$\{f^{II}_{1569}, f^{II}_{1593}, f^{II}_{1412}, f^{II}_{1581}, f^{II}_{1556}, f^{II}_{1398}\}$	yes
		$\{f_{1134}, f_{1082}, f_{875}\}$		yes
		$\{f_{1036}, f_{1049}, f_{1008}, f_{1022}, f_{993}\}$	$\{f_{1036}, f_{1049}, f_{1008}, f_{1022}, f^{II}_{993}, f^{II}_{905}, f^{II}_{918}\}$	no
		$\{f_{918}\}$	$\{f_{1036}, f_{1049}, f^{II}_{905}, f_{1008}, f^{II}_{918}, f_{1022}, f^{II}_{993}\}$	no
		$\{f_{1384}, f_{1372}, f_{1359}\}$		yes
		$\{f_{1569}, f_{1593}, f_{1412}\}$	$\{f^{II}_{1569}, f^{II}_{1593}, f^{II}_{1412}, f^{II}_{1581}, f^{II}_{1556}, f^{II}_{1398}\}$	yes
		$\{f_{1187}, f_{1206}, f_{1174}\}$		yes
sl_nt	$\{f_{1713}, f_{1688}, f_{1273}, f_{1556}, f_{1623}, f_{1636}, f_{1300}, f_{1701}, f_{1372}, f_{1220}, f_{1187}, f_{1247}, f_{1648}, f_{1675}, f_{1082}, f_{1662}, f_{1569}\}$	$\{f_{1569}, f_{1593}, f_{1412}\}$	$\{f^{II}_{1569}, f^{II}_{1593}, f^{II}_{1412}, f^{II}_{1581}, f^{II}_{1556}, f^{II}_{1398}\}$	no
		$\{f_{1675}, f_{1713}, f_{1701}, f_{1688}, f_{1662}\}$		no
		$\{f_{1623}, f_{1648}, f_{1636}\}$		no
		$\{f_{1187}, f_{1206}, f_{1174}\}$		no
		$\{f_{1134}, f_{1082}, f_{875}\}$		no
		$\{f_{1581}, f_{1556}, f_{1398}\}$	$\{f^{II}_{1569}, f^{II}_{1593}, f^{II}_{1412}, f^{II}_{1581}, f^{II}_{1556}, f^{II}_{1398}\}$	no
		$\{f_{1384}, f_{1372}, f_{1359}\}$		no
		$\{f_{1287}, f_{1273}, f_{1300}, f_{1312}\}$		yes
		$\{f_{1234}, f_{1259}, f_{1247}, f_{1220}\}$		yes

Table D.2 Results for the feature classes pr_re, sl_th and sl_nt

Feature class	Set of seeds detected	Plant developed	Plant modified	Feature validated
st	$\{f_{1234}, f_{1713}, f_{1412}, f_{1022}, f_{1312}, f_{1593}, f_{1688}, f_{1008}, f_{1346}, f_{1273}, f_{860}, f_{1556}, f_{1623}, f_{836}, f_{1398}, f_{1636}, f_{1300}, f_{1174}, f_{1701}, f_{1581}, f_{1161}, f_{1372}, f_{1220}, f_{1287}, f_{1187}, f_{1384}, f_{1325}, f_{1531}, f_{875}, f_{905}, f_{1543}, f_{1247}, f_{918}, f_{1433}, f_{1122}, f_{1451}, f_{1359}, f_{1609}, f_{1648}, f_{1259}, f_{1506}, f_{1675}, f_{1036}, f_{1134}, f_{993}, f_{1475}, f_{1061}, f_{1492}, f_{1082}, f_{1662}, f_{1569}, f_{1519}, f_{1049}, f_{1206}\}$	$\{f_{1134}, f_{1082}, f_{875}\}$		no
		$\{f_{1234}, f_{1259}, f_{1247}, f_{1220}\}$		no
		$\{f_{1569}, f_{1593}, f_{1412}\}$	$\{f^{II}_{1569}, f^{II}_{1593}, f^{II}_{1412}, f^{II}_{1581}, f^{II}_{1556}, f^{II}_{1398}\}$	no
		$\{f_{1187}, f_{1206}, f_{1174}\}$		no
		$\{f_{1384}, f_{1372}, f_{1359}\}$		no
		$\{f_{860}\}$	$\{f^{II}_{860}, f^{II}_{836}\}$	no
		$\{f_{836}\}$	$\{f^{II}_{860}, f^{II}_{836}\}$	no
		$\{f_{1287}, f_{1273}, f_{1300}, f_{1312}\}$		no
		$\{f_{1325}\}$	$\{f^{II}_{1346}, f^{II}_{1325}\}$	no
		$\{f_{905}\}$	$\{f_{1036}, f_{1049}, f^{II}_{905}, f_{1008}, f^{II}_{918}, f_{1022}, f^{II}_{993}\}$	no
		$\{f_{1675}, f_{1713}, f_{1701}, f_{1688}, f_{1662}\}$		no
		$\{f_{1161}\}$		no
		$\{f_{1623}, f_{1648}, f_{1636}\}$		no
		$\{f_{1581}, f_{1556}, f_{1398}\}$	$\{f^{II}_{1569}, f^{II}_{1593}, f^{II}_{1412}, f^{II}_{1581}, f^{II}_{1556}, f^{II}_{1398}\}$	no
		$\{f_{1346}\}$	$\{f^{II}_{1346}, f^{II}_{1325}\}$	no
		$\{f_{1475}, f_{1492}, f_{1543}, f_{1531}, f_{1451}, f_{1506}, f_{1519}, f_{1433}\}$	$\{f_{1492}, f^{IV}_{1475}, f^{IV}_{1451}, f_{1543}, f_{1433}\}$	no
			$\{f^{IV}_{1475}, f_{1531}, f_{1506}, f^{IV}_{1451}, f_{1519}\}$	no
		$\{f_{1122}\}$		no
		$\{f_{1609}\}$		no
		$\{f_{1061}\}$		no
		$\{f_{1036}, f_{1049}, f_{1008}, f_{1022}, f_{993}\}$	$\{f_{1036}, f_{1049}, f_{1008}, f_{1022}, f^{II}_{993}, f^{II}_{905}, f^{II}_{918}\}$	no
		$\{f_{918}\}$	$\{f_{1036}, f_{1049}, f^{II}_{905}, f_{1008}, f^{II}_{918}, f_{1022}, f^{II}_{993}\}$	no

Table D.3 Results for the feature class st

Appendix E - Results for case study VIII in Chapter 5

This appendix consists of three tables that report the results of the recognition process carried out on the part used for case study VIII in Chapter 5.

Feature class	Set of seeds detected	Plant developed	Plant modified	Feature validated
po_re	{Ø}			
po_ob	$\{f_{1110}, f_{682}, f_{697}, f_{1098}, f_{968}, f_{995}\}$	$\{f_{741}, f_{1019}, f_{953}, f_{968}\}$		no
		$\{f_{682}, f_{697}\}$		no
		$\{f_{1098}, f_{1110}\}$		no
		$\{f_{725}, f_{1007}, f_{758}, f_{995}\}$		no
ho_bl	{Ø}			
ho_th	$\{f_{1110}, f_{783}, f_{1032}, f_{637}, f_{1071}, f_{1045}, f_{841}, f_{618}, f_{682}, f_{697}, f_{923}, f_{1098}, f_{872}, f_{885}, f_{667}, f_{588}, f_{1058}, f_{910}, f_{898}, f_{1083},\}$	$\{f_{637}\}$	$\{f^{II}_{637}, f^{II}_{618}, f^{II}_{667}\}$	no
		$\{f_{910}\}$		no
		$\{f_{1045}\}$		no
		$\{f_{898}\}$		no
		$\{f_{1110}, f_{1098}\}$		yes
		$\{f_{682}, f_{697}\}$		yes
		$\{f_{1032}\}$	$\{f^{III}_{1032}, f^{III}_{1058}, f^{III}_{1083}\}$	no
		$\{f_{558}\}$		no
		$\{f_{783}\}$		no
		$\{f_{841}\}$		no
		$\{f_{618}\}$	$\{f^{II}_{637}, f^{II}_{618}, f^{II}_{667}\}$	no
		$\{f_{667}\}$	$\{f^{II}_{637}, f^{II}_{618}, f^{II}_{667}\}$	no
		$\{f_{1058}\}$	$\{f^{III}_{1032}, f^{III}_{1058}, f^{III}_{1083}\}$	no
		$\{f_{1083}\}$	$\{f^{III}_{1032}, f^{III}_{1058}, f^{III}_{1083}\}$	no
		$\{f_{923}\}$		no
		$\{f_{872}\}$		no
		$\{f_{885}\}$		no
		$\{f_{1071}\}$		no

Table E.1 Results for the feature classes po_re, po_ob, ho_bl and ho_th

Feature class	Set of seeds detected	Plant developed	Plant modified	Feature validated
pr_ci	$\{\emptyset\}$			
pr_re	$\{\emptyset\}$			
sl_th	$\{f_{783}, f_{741}, f_{807}, f_{725}, f_{1032}, f_{637}, f_{819}, f_{1007}, f_{603}, f_{758}, f_{841}, f_{618}, f_{923}, f_{795}, f_{872}, f_{885}, f_{667}, f_{588}, f_{1058}, f_{910}, f_{940}, f_{981}, f_{1083}, f_{1019}, f_{953}\}$	$\{f_{1032}\}$	$\{f_{1032}^{III}, f_{1058}^{III}, f_{1083}^{III}\}$	yes
		$\{f_{618}\}$	$\{f_{637}^{II}, f_{618}^{II}, f_{667}^{II}\}$	no
		$\{f_{841}\}$		no
		$\{f_{819}, f_{795}\}$		no
		$\{f_{885}\}$		no
		$\{f_{588}\}$		no
		$\{f_{872}\}$		no
		$\{f_{667}\}$	$\{f_{637}^{II}, f_{618}^{II}, f_{667}^{II}\}$	no
		$\{f_{1058}\}$	$\{f_{1032}^{III}, f_{1058}^{III}, f_{1083}^{III}\}$	yes
		$\{f_{953}, f_{741}, f_{1019}, f_{968}\}$		no
		$\{f_{637}\}$	$\{f_{637}^{II}, f_{618}^{II}, f_{667}^{II}\}$	no
		$\{f_{923}\}$		no
		$\{f_{725}, f_{1007}, f_{758}, f_{995}\}$		no
		$\{f_{981}, f_{940}\}$		no
		$\{f_{807}, f_{603}\}$		no
		$\{f_{783}\}$		no
		$\{f_{910}\}$		no
		$\{f_{1083}\}$	$\{f_{1032}^{III}, f_{1058}^{III}, f_{1083}^{III}\}$	yes
sl_nt	$\{f_{1110}, f_{807}, f_{1007}, f_{682}, f_{795}, f_{1019}, f_{953}\}$	$\{f_{807}, f_{603}\}$		no
		$\{f_{819}, f_{795}\}$		no
		$\{f_{682}, f_{697}\}$		no
		$\{f_{1110}, f_{1098}\}$		no
		$\{f_{725}, f_{1007}, f_{758}, f_{995}\}$		no
		$\{f_{953}, f_{741}, f_{1019}, f_{968}\}$		no

Table E.2 Results for the feature classes pr_ci, pr_re, sl_th and sl_nt

Feature class	Set of seeds detected	Plant developed	Plant modified	Feature validated
st	$\{f_{783}, f_{741}, f_{807}, f_{725}, f_{1032}, f_{637}, f_{819}, f_{1007}, f_{603}, f_{758}, f_{841}, f_{618}, f_{923}, f_{795}, f_{872}, f_{885}, f_{667}, f_{588}, f_{1058}, f_{910}, f_{940}, f_{981}, f_{1083}, f_{1019}, f_{953}\}$	$\{f_{953}, f_{741}, f_{1019}, f_{968}\}$		no
		$\{f_{841}\}$		no
		$\{f_{1032}\}$	$\{f_{1032}^{III}, f_{1058}^{III}, f_{1083}^{III}\}$	no
		$\{f_{885}\}$		no
		$\{f_{923}\}$		no
		$\{f_{981}, f_{940}\}$		yes
		$\{f_{872}\}$		no
		$\{f_{637}\}$	$\{f_{637}^{II}, f_{618}^{II}, f_{667}^{II}\}$	no
		$\{f_{783}\}$		no
		$\{f_{667}\}$	$\{f_{637}^{II}, f_{618}^{II}, f_{667}^{II}\}$	no
		$\{f_{807}, f_{603}\}$		yes
		$\{f_{618}\}$	$\{f_{637}^{II}, f_{618}^{II}, f_{667}^{II}\}$	no
		$\{f_{910}\}$		no
		$\{f_{795}, f_{819}\}$		yes
		$\{f_{1058}\}$	$\{f_{1032}^{III}, f_{1058}^{III}, f_{1083}^{III}\}$	no
		$\{f_{588}\}$		no
		$\{f_{1083}\}$	$\{f_{1032}^{III}, f_{1058}^{III}, f_{1083}^{III}\}$	no
		$\{f_{725}, f_{1007}, f_{758}, f_{995}\}$		no

Table E.3 Results for the feature class st

References

Ames, A.L., 1991. Production ready feature recognition based automatic group technology part coding. *Proceedings of the First ACM Symposium on Solid Modelling Foundations and CAD/CAM Applications*, Austin, Texas, US, June 5-7, pp. 161-169.

Bhandarkar, M.P. and Nagi, R., 2000. STEP-based feature extraction from STEP geometry for agile manufacturing. *Computers in Industry*, Vol. 41, N°1, pp. 3-24.

Bigot, S., 2002. New techniques for handling continuous values in inductive learning. Ph.D. Thesis, University of Wales, Cardiff, UK.

Brun, J.M., 1994. From characteristic shapes to form features: a recognition strategy. *Proceedings of IFIP International Conference on Feature Modeling and Recognition In Advanced CAD/CAM Systems*, Valenciennes, France, Vol. 1, pp. 315-326.

Chan, A.K.W. and Case, K., 1994. Process planning by recognizing and learning machining features. *International Journal of Computer Integrated Manufacturing*, Vol. 7, N°2, pp. 77-99.

Chen, Y.H. and Lee, H.M., 1998. A neural network system for two-dimensional feature recognition. *International Journal of Computer Integrated Manufacturing*, Vol. 11, N°2, pp. 111-117.

Cicirello, V. and Regli, W.C., 2001. Machining feature-based comparisons of mechanical parts. *International Conference on Shape Modelling and Applications*, Genova, Italy, May 7-11, pp. 176-185.

Dereli, T. and Filiz, H., 2002. A note on the use of STEP for interfacing design to process planning. *Computer-Aided Design*, Vol. 34, N°14, pp. 1075-1085.

Dimov, S.S., Setchi, R.M. and Brousseau, E.B., 2004. Automatic feature recognition – a hybrid approach. *Fifth International Conference on Integrated Design and Manufacturing in Mechanical Engineering, IDMME'04*, Bath, UK, April 5-7.

Donaldson, I.A. and Corney, J.R., 1993. Rule-based feature recognition for 2.5D machined components. *International Journal of Computer Integrated Manufacturing*, Vol. 6, N°1&2, pp. 51-64.

Dong, J. and Vijayan, S., 1997. Feature extraction with the consideration of manufacturing processes. *International Journal of Production Research*, Vol. 35, N°8, pp. 2135-2155.

Dong, X. and Wozny, M., 1988. FRAFES, a frame-based feature extraction system. *Proceedings of the International Conference on Computer Integrated Manufacturing*, Troy, NY, US, May 23-25, pp. 296-305.

Gao, S. and Shah, J.J., 1998. Automatic recognition of interacting machining features based on minimal condition subgraph. *Computer-Aided Design*, Vol. 30, N°9, pp. 727-739.

Gindy, N.N.Z., 1989. A hierarchical structure for form features. *International Journal of Production Research*, Vol. 27, N°12, pp. 2089-2103.

Gupta, S.K., Kramer, T.R., Nau, D.S., Regli, W.C. and Zhang, G., 1994. Building MRSEV models for CAM applications. *Advances in Engineering Software*, Vol. 20, N°2-3, pp. 121-139.

Han, J., 1996. 3D geometric reasoning algorithms for feature recognition. Ph.D. Thesis, University of Southern California, USA.

Han, J. and Requicha, A.A.G., 1994. Incremental recognition of machining features. *Proceedings of ASME International Conference on Computers in Engineering,* Minneapolis, MN, Vol. 1, September 11-14, pp. 143-149.

Han, J., Kang, M. and Choi, H., 2001. STEP-based feature recognition for manufacturing cost optimization. *Computer-Aided Design*, Vol. 33, N°9, pp 671-686.

Han, J., Pratt, M. and Regli, W.C., 2000. Manufacturing feature recognition from solid models: a status report. *IEEE Transactions on Robotics and Automation*, Vol. 16, N°6, pp. 782-796.

Henderson, M.R. and Anderson, D.C., 1984. Computer recognition and extraction of form features: a CAD/CAM link. *Computers in Industry*, Vol. 5, N°4, pp. 329-339.

Henderson, M.R., Srinath, G., Stage, R., Walker, K. and Regli, W., 1994. Boundary representation-based feature identification. In *Advances in feature based manufacturing*, Shah, J.J., Mantyla, M. and Nau, D.S. (Eds.), Elsevier Science Publishers, pp. 15-38.

Hoffmann, C.M., 1989. Geometric and solid modeling, an introduction. Morgan Kaufmann Publishers, San Mateo, CA, USA.

Hummel, K.E., 1989. Coupling rule-based and object-oriented programming for the classification of machined features. *Proceedings of the ASME International Computers in Engineering Conference*, Anheim, CA, USA, pp. 409-418.

Hwang, J-L. and Henderson, M.R., 1992. Applying the perceptron to three-dimensional feature recognition. *Journal of Design and Manufacturing*, Vol. 2, N°4, pp. 187-198.

Ismail, N., Abu Bakar, N. and Juri, A.H., 2002. Feature recognition patterns for form features using boundary representation models. *International Journal of Advanced Manufacturing Technology*, Vol. 20, N°8, pp. 553-556.

Java Compiler Compiler™ Version 3.1, 2003. [WWW] <URL: https://javacc.dev.java.net/> [Accessed 16 September 2003].

Jha, K. and Gurumoorthy, B., 2000. Multiple feature interpretation across domains. *Computers in Industry*, Vol. 42, N°1, pp. 13-32.

Ji, Q. and Marefat, M.M., 1995. Bayesian approach for extracting and identifying features. *Computer-Aided Design*, Vol. 27, N°6, pp. 435-454.

Ji, Q. and Marefat, M.M., 1997. Machine interpretation of CAD data for manufacturing applications. *ACM Computing Survey*, Vol. 24, N°3, pp. 264-311.

Joshi, S. and Chang, T.C., 1988. Graph-based heuristic for recognition of machined features from a 3D solid model. *Computer-Aided Design*, Vol. 20, N°2, pp. 58-66.

Kramer, T.R., 1994. A library of material removal shape element volumes (MRSEVs). *Technical Report NISTIR 4809*, The National Institute of Standards and Technology, Gaithersburg, USA.

Lankalapalli, K., Chatterjee, S. and Chang T.C., 1997. Feature recognition using ART2: a self-organising neural network. *Journal of Intelligent Manufacturing*, Vol. 8, N°3, pp. 203-214.

Li, W.D., Ong, S.K. and Nee, A.Y.C., 2000. Recognition of overlapping machining features based on hybrid artificial intelligent techniques. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, Vol. 214, N°8, pp. 739-744.

Li, W.D., Ong, S.K. and Nee, A.Y.C., 2003. A hybrid method for recognizing interacting machining features. *International Journal of Production Research*, Vol. 41, N°9, pp. 1887-1908.

Liu, H. and Motoda, H., 1998. Feature selection for knowledge discovery and data mining. Kluwer Academic Publishers, MA, USA.

Ma, S., 2003. STEP Part 21 - clear text encoding parser to be used with JavaCC. [WWW] <URL: http://www.cobase.cs.ucla.edu/pub/javacc/#Ssection> [Accessed 16 September 2003]

Marefat, M.M. and Ji, Q., 1997. Hierarchical Bayesian methods for recognition and extraction of 3-D shape features from CAD solid models. *IEEE Transactions on Systems, Man and Cybernetics – Part A: Systems and Humans*, Vol. 27, N°6, pp. 705-727.

Marefat, M. and Kashyap, R.L., 1990. Geometric reasoning for recognition of three-dimensional object features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12, N°10, pp. 949-965.

Marquez, M., White, A. and Gill, R., 2001. A hybrid neural network – feature-based manufacturability analysis of mould reinforced plastic parts. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, Vol. 215, N°8, pp. 1065-1079.

National Design Repository, 2004. [WWW] <URL: http://edge.mcs.drexel.edu/repository > [Accessed 16 October 2004]

Nell, J., 2004. STEP on a Page. [WWW] <URL: http://www.mel.nist.gov/sc5/soap/> [Accessed 25 November 2004].

Nezis, K. and Vosniakos, G., 1997. Recognising $2\frac{1}{2}$D shape features using a neural network and heuristics. *Computer-Aided Design*, Vol. 29, N°7, pp. 523-539.

Owen, J., 1993. STEP an introduction. Information Geometers Ltd.

Owodunni, O. and Hinduja, S., 2002. Evaluation of existing and new feature recognition algorithms. Part 1: theory and implementation. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, Vol. 216, N°6, pp. 839-851.

Peters, T.J., 1992. Encoding mechanical design features for recognition via neural nets. *Research in Engineering Design*, Vol. 4, N°1, pp. 67-74.

Pham, D.T. and Dimov, S.S., 1997. An efficient algorithm for automatic knowledge acquisition. *Pattern Recognition*, Vol. 30, N°7, pp. 1137-1143.

Pham, D.T. and Dimov, S.S., 1998. An approach to concurrent engineering. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, Vol. 212, N°1, pp. 13-27.

Pham, D.T. and Liu, X., 1995. Neural networks for identification, prediction and control. Springer-Verlag London Limited, Great Britain.

Prabhakar, S. and Henderson, M.R., 1992. Automatic form-feature recognition using neural-network-based techniques on boundary representations of solid models. *Computer-Aided Design*, Vol. 24, N°7, pp. 381-393.

Pratt, M.J., 2001. Introduction to ISO 10303 – the STEP Standard for Product Data Exchange. *Journal of Computing and Information Science in Engineering*, Vol. 1, N°1, pp. 102-103.

PTC, 2001. Pro/ENGINEER 2001 Release Notes. *Doc-RN100-EN-320,* Parametric Technology Corporation, Waltham, MA, USA.

Qamhiyah, A.Z., Venter, R.D. and Benhabib, B., 1996. Geometric reasoning for the extraction of form features. *Computer-Aided Design*, Vol. 28, N°11, pp. 887-903.

Regli, W.C., 1995. Geometric algorithms for recognition of features from solid models. Ph.D. Thesis, University of Maryland, USA.

Regli, W.C. and Pratt, M.J., 1996. What are feature interactions? *Proceedings of the ASME Computers in Engineering Conference*, August 18-22, Irvine, California, USA.

Sakurai, H. and Gossard, D.C., 1990. Recognizing shape features in solid models. *IEEE Computer Graphics and Applications*, Vol. 10, N°5, pp. 22-32.

Sandiford, D. and Hinduja, S., 2001. Construction of feature volumes using intersection of adjacent surfaces. *Computer-Aided Design*, Vol. 33, N°6, pp. 455-473.

Shah, J.J., 1991. Assessment of features technology. *Computer-Aided Design*, Vol. 23, N°5, pp.331-343.

Shah, J.J. and Mantyla, M., 1995. Parametric and feature-based CAD/CAM. Wiley, New York.

Shah, J.J., Anderson, D., Kim, Y.S. and Joshi, S., 2001. A discourse on geometric feature recognition from CAD models. *Journal of Computing and Information Science in Engineering*, Vol. 1, N°1, pp. 41-51.

Subrahmanyam, S.R., 2002. Fixturing features selection in feature-based systems. *Computers in Industry*, Vol. 48, N°2, pp. 99-108.

Suh, Y.S., 1995. A feature conversion CAD system for the concurrent engineering environment. Ph.D. Thesis, Rensselear Polytechnic Institute, Troy, New York.

Sung, R.C.W., Corney, J.R. and Clark, D.E.R., 2001. Automatic assembly feature recognition and disassembly sequence generation. *Journal of Computing and Information Science in Engineering*, Vol. 1, N°4, pp. 291-299.

Trika, S.N. and Kashyap, R.L., 1994. Geometric reasoning for extraction of manufacturing features in iso-oriented polyhedrons. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16, N°11, pp. 1087-1110.

van Holland, W. and Bronsvoort, W.F., 2000. Assembly features in modeling and planning. *Robotics and Computer Integrated Manufacturing*, Vol. 16, N°4, pp. 277-294.

Vandenbrande, J.H. and Requicha, A.A.G., 1993. Spatial reasoning for the automatic recognition of machinable features in solid models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, N°12, pp.1269-1285.

Vosniakos, G.C. and Davies, B.J., 1993. A shape feature recognition framework and its application to holes in prismatic parts. *International Journal of Advanced Manufacturing Technology*, Vol. 8, N°5, pp. 345-351.

Wilson, P.R. and Pratt, M.J., 1988. A taxonomy of features for solid modeling. In *Geometric modeling for CAD applications*, Wozny, M.J., Encarnacao, J.L. (Eds.), Elsevier Science Publishers, Holland, pp.125-135.

Xu, X. and Hinduja, S., 1998. Recognition of rough machining features in $2\frac{1}{2}$D components. *Computer-Aided Design*, Vol. 30, N°7, pp. 503-516.

Yuen, C.F. and Venuvinod, P.K., 1999. Geometric feature recognition: coping with the complexity and infinite variety of features. *International Journal of Computer Integrated Manufacturing*, Vol. 12, N°5, pp. 439-452.

Zhang, C., Chan, K.W. and Chen, Y.H., 1998. A hybrid method for recognising feature interactions. *Integrated Manufacturing Systems*, Vol. 9, N°2, pp. 120-128.

Zhang, S.G., Ajmal, A., Wootton, J. and Chisholm, A., 2000. A feature-based inspection process planning system for co-ordinate measuring machine (CMM). *Journal of Material Processing Technology*, Vol. 107, N°1-3, pp. 111-118.

Zulkifli, A.H. and Meeran, S., 1999. Decomposition of interacting features using a Kohonen self-organizing feature map neural network. *Engineering Applications of Artificial Intelligence*, Vol. 12, N°1, pp. 59-78.