1994

A Boolean Algebra Approach To High-level Process Planning

Hugh Jack

A BOOLEAN ALGEBRA APPROACH TO HIGH-LEVEL PROCESS PLANNING

by

Hugh Jack

Department of Mechanical Engineering
Faculty of Engineering Science

Submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

Faculty of Graduate Studies
The University of Western Ontario
London, Ontario
November 1993

ISBN 0-315-90530-1

ABSTRACT

One of the most daunting challenges in Computer Integrated Manufacturing (CIM) is bridging the gap between Computer Aided Design (CAD) and Computer Aided Manufacturing (CAM). Many specific approaches have been developed for limited manufacturing domains. For example, a solid model may be directly converted into Numerical Control (NC) code for a computerized milling machine. But, when we try to extend the planning to cover a number of domains, there are no satisfactory techniques. This still leaves many unresolved problems when converting a product from design features to manufacturing features.

Computer Aided Process Planning (CAPP) is the umbrella term for automated approaches to determining production process parameters from a design. This thesis has a description of the various approaches and methods used when constructing these systems. For the most part, the existing research has focussed on limited production domains, while there is a definite need for a system that can combine a large number of different manufacturing processes. Such an approach is presented in this thesis.

This approach begins with a design stored in sets that are linked together with Boolean equations. The information in this representation can be collected using modern solid modeling systems. The primary difference is that current CAD systems perform operations and then discard the information provided by the user. When this information is kept, it provides a powerful tool for reasoning about manufacturing a product.

The design is loaded into a hierarchical, non-linear planning system. The planner first looks for assemblies and reused features. Each of the features and assem-

blies are examined separately. In a first pass, the equation for a part is examined for known forms in the equation using templates. If a template is matched to part of an equation, then corresponding production rules are examined. These rules are of the form 'if <conditions> then <actions>'. The planner will find a number of alternatives for each operation, and try to plan from beginning to end. If for some reason the planner is not able to complete the plan, it will backtrack and try another alternative. If the planner is unable to find a satisfactory plan, then it will simplify the Boolean equation and begin planning again.

The planner's capabilities are demonstrated through the use of examples. The results show that the planner is able to plan with mixed production technologies, select plans on the basis of cost, generate alternative operations, deal with new technologies, and recover from failures. The thesis is concluded with a discussion of the method in general, including some of the future developments, some of the shortcomings, and some of the outstanding research questions for the method.

AKNOWLEGEMENTS

Looking back there are so many people who must be thanked for their support during my studies. My parents must be thanked for their many years of support, through all of my studies. Also, my wife Alicia has provided years of encouragement, and this thesis would not have been possible without her patience and understanding.

Among the others to thank, my advisor, Professor W.H.ElMaraghy for his timely advice, Dan Corrin for his help in avoiding computational peril, and my academic colleagues for their discussions, and suggestions.

TABLE OF CONTENTS

LIST OF FIGURES

NOMENCLATURE

'AND, &' - These symbols are used for the binary operation known as Intersection. Only those parts of two sets that overlap will be preserved in the result. Equivalent notation found in other sources are '*, •, ∩'.

'OR, +' - These symbols are used for the binary operation known as Union. The total set membership of both sets will become the resultant. An equivalent notation found in other sources is, '∪'.

'NOT, ~' - These symbols are used for the unary operation known as Inverse, or Complement. The result of this operation will be an inverted geometry of the initial set. Where volume was before the operator was applied will become void, and vice versa for the previous void. An equivalent notation found in other texts is '$\overline{X}$'.

'-' - The CSG Difference, or Subtraction operator. When used as 'A - B', the set 'A' will have any overlap with set 'B' removed. This is not a simple Boolean operator, but actually a complex function of the form 'A & ~B'. The order is not commutative.

':' - This is the notation used in this thesis for the Assembly operator. While this is not a Boolean operator, it can be considered to define a list of Boolean equations. Some references use '+' for the assembly operator.

'+*, &*, -*, ~*' - The regularized set operators that ensure closed regular sets. Other references use the notations, '∪*, ∩*, -*, c*'.

'$+^{+}$, $\&^{+}$, $-^{+}$, $\sim^{+}$' - The regularized set operators for open regular sets. Other references use the notation, '$\cup^{+}$, $\cap^{+}$, $-^{+}$, c^{+}'.

';' - This operator is used in this thesis to append one set to another. For example 'set1;-set2' will append the members of 'set2' to the end of 'set1'. This operator would allow reuse of basis sets such as 'set1'.

GLOSSARY

B-Rep (Boundary Representation) - An object is represented with surfaces (often defined with splines), vertices and edges that will interconnect them.

BOM (Bill Of Materials) - A list of parts that are required for one manufacturing step of a product. This is most commonly used when dealing with assemblies.

CAD (Computer Aided Design) - Computers are used to enter and analyze designs. These systems commonly use graphics to allow interactive definitions of geometries, and properties.

CAM (Computer Aided Manufacturing) - Computers are used to control, and drive manufacturing processes.

CAPP (Computer Aided Process Planning) - The process of converting a design to a set of manufacturing processes, using computers.

CIM (Computer Integrated Manufacturing) - The use of computers to bring together automated systems throughout a manufacturing company. Most commonly the CIM system is used to connect CAD systems to CAM systems. Common CIM functions include CAPP, PPC, etc.

CSG (Constructive Solid Geometry) - A method of assembling objects out of simpler primitives using basic operations such as Intersection, Union and Subtraction.

Generative Process Planning - This sort of planner will create process plans without reference to existing process plans.

GT (Group Technology) - A method of assigning codes to parts so that they tend to fall into groups. The codes may subsequently be used for recalling similar designs, or associated information (such as process plans).

Nested Boolean - A representation adopted for this thesis that uses brackets for every operation when representing Boolean equations. This representation was developed for use in this thesis.

Nonmanifold - Defines cases where the geometry of a part is not rational mathematically. This includes infinitely thin sections, point contacts, and lines with no volume.

Operation Planning - The later stage of process planning where the detailed process parameters are determined. For example, in machining we might get speeds, feeds, number of passes, specific tools, etc.

PPC (Production Planning and Control) - This software will combine orders for products and their process plans to develop a schedule for manufacturing. Modern systems can deal with short term control problems such as machine failure.

Process Planning - This term is used to refer to the selection of processes for the production of a part. This term is also used at a higher-level to also include the selection of operation values. Both uses have been historically used, and all attempts have been made to avoid ambiguities in the text.

Set Theoretic - Another name for CSG.

Variant Process Planning - A Variant process planner will recall previously created process plans for new designs, and allow subsequent editing to resolve any differences. The important distinguishing feature is that the planner uses plans that are variations of other plans.

1.INTRODUCTION

1.1 WHY CAPP?

New technologies can have a revolutionary impact upon the manufacturing state, as was seen during the Industrial Revolution. Computers are in the process of reshaping the modern industry. Manufacturers are rushing to apply this technology to remain competitive. To date a majority of the methods have been limited to enhancing and/or replacing existing manufacturing functions with software.

Within the Manufacturing cycle, there are some fundamental functions (as shown in Figure 1). The figure depicts one particular view of manufacturing suggested by Rembold et. al. [1985]. It is easy to see the complexity of the interactions between functions. Most of these functions have been automated for Accounting, Monitoring and Control, Time Scheduling and Organizational Planning. However, planning for all aspects of manufacturing has been very difficult to computerize because of its highly abstract and time variant nature. The systems that can do so are limited to very small domains. All of the systems for limited domains create a need for a system that will make higher level decisions between production technologies.

In general terms, new product development is a function of; i) Design, ii) Process Planning, iii) Scheduling, iv) Control, and v) Manufacturing. The Design function has been supplemented with Computer Aided Design (CAD) and Computer Aided Engineering (CAE) tools that enable the modeling of geometry, analysis of stresses and

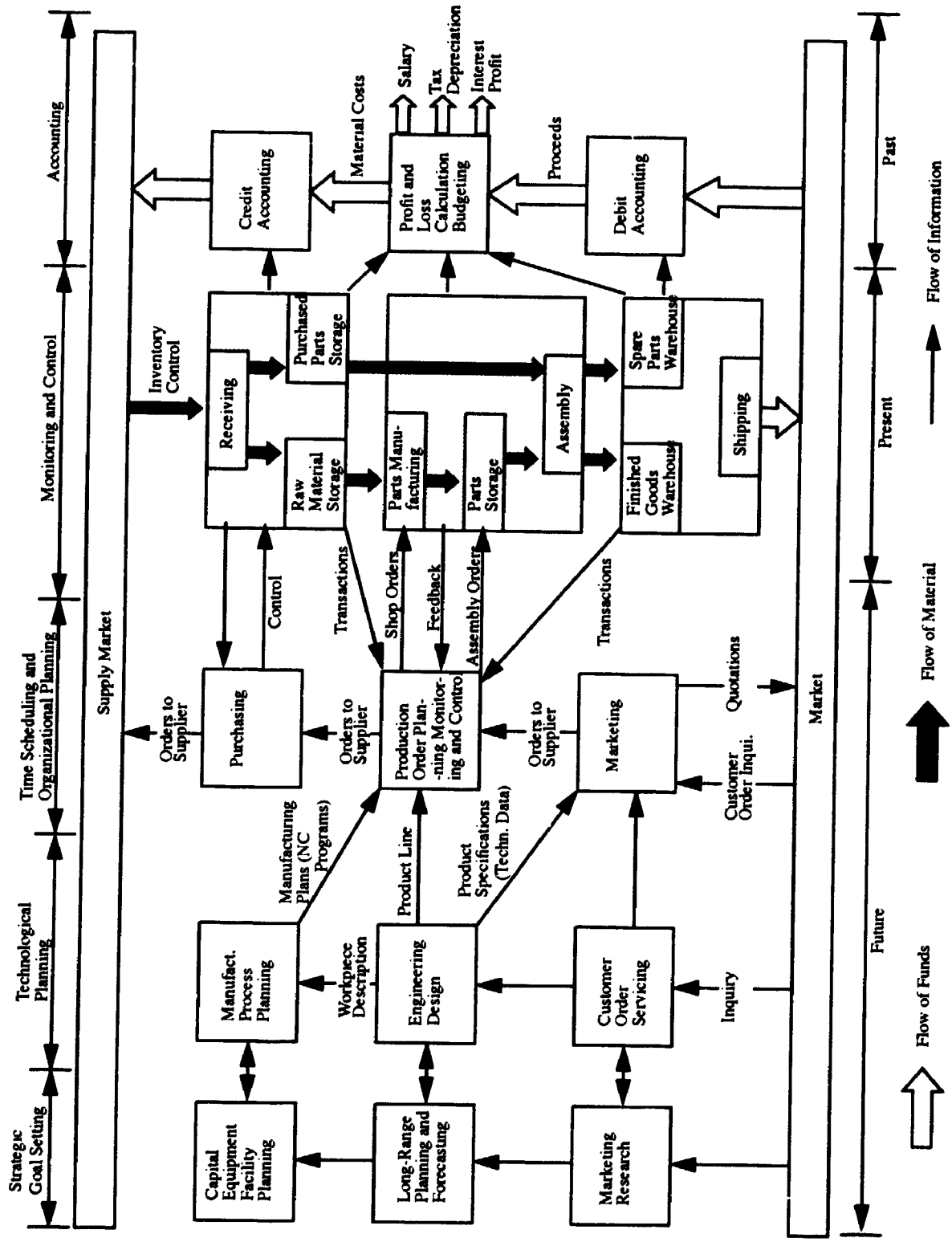


Figure 1 The Basic Manufacturing Cycle
(Adapted from Rembold et. al. [1985], pp.30)

heat flow, evaluation of tooling, and other sophisticated problems to be defined. During process planning, the design is converted into a list of operations and resources that are required to direct production. Up to this point products and parts are dealt with individually. During scheduling, the process plans for a number of products are considered together as required by the customer orders. Finally, the Production Control function ensures that the schedules are being adhered to in a reasonable manner.

The weakest link in the CIM (Computer Integrated Manufacturing) philosophy is Process Planning. The deficiencies of process planning have not resulted from a lack of effort by researchers. As early as 1965 the concept was discussed by Niebel [1965]. By 1988, Alting and Zhang [1988] listed 156 existing CAPP systems, of which 49 where alleged to be available commercially. Despite the large number of commercial packages, there are several reasons why CAPP systems have not been widely utilized to date:

- Very limited process planning domains.
- They require extensive setup.
- Input is difficult, and often not from a common CAD package.

One of the major hypotheses of this thesis is that the main reason for the shortcomings of process planning is a result of the method of product representation. This thesis aims to explore a new representation method which enhances the product representation using Boolean equations. The use of Boolean equations enables a new group of methods to be used to manipulate and analyze designs to eventually produce process plans.

1.2 PROCESS PLANNING

A simple job shop, with only a few machines, can do process planning using experience and intuition. But, as the size of the factory grows, and the permutations and combinations of options grows, so does the complexity. This creates a need for specialists called Process Planners who use their experience and detailed knowledge to select the method of production. Requicha and Vandenbrande [1988] describe the task of process planning as:

> "A process planner and a set-up planner (often the same person) examine a part's blueprint and consult various files and handbooks to produce a process plan. A plan contains process specifications and information on fixtures and clamping devices to be used, and on set-up of the work-piece on a machine tool. Set-up specifications are typically conveyed through annotated sketches or engineering drawings."

Requicha and Vandenbrande describe the process planning operation as a two step process, as shown in Figure 2.

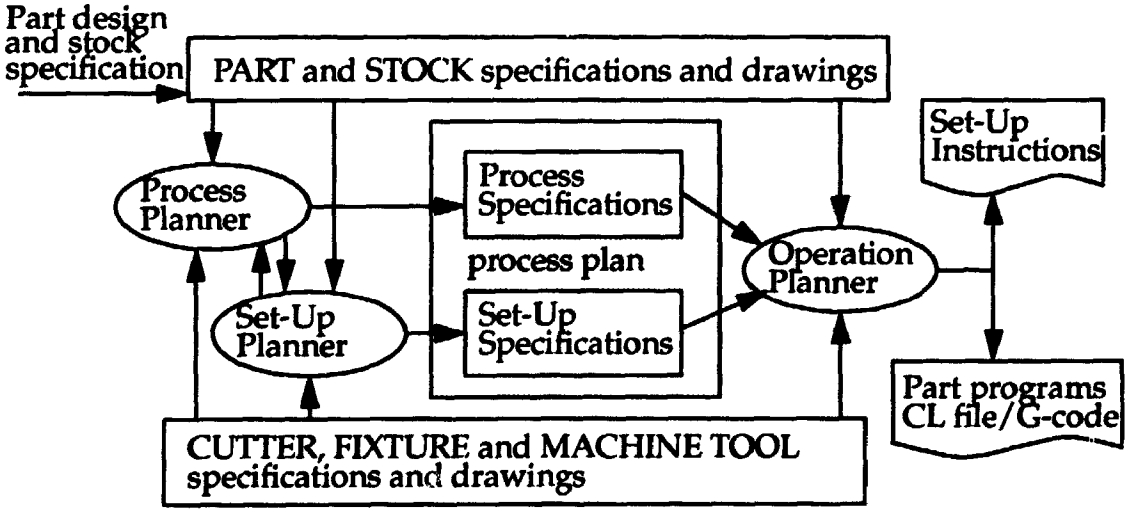


Figure 2 Traditional Two-Stage Approach To Process Planning
(Adapted from Requicha & Vandenbrande, 1988, pp.302)

The first step involves the abstract phases of Process Planning and Set-up Planning. In the second stage, Operation Planning is used to assign details to the operations in the process plan. The final result of process planning can be as shown in Figure 3. This sheet emphasizes the ad-hoc nature of data identification, and plan completeness. Many details are cryptic, or must be completed at the machine by the operator.

OPERATION SHEET

Part No. CLP023456-4-92-023 Material steel 1040
Part Name Widget
Orig. H.Jack Changes
Checked I.M. Yurboss Approved D.Corrin

No.	Operation	Machine	Setup	Time (hrs.)
0010	Saw off and slug 1.75 dia. hole	Dept 12. Saw 3		.3
0015	R'Turn 6.00 dia. stock to 5.210/5.190 R'Bore 1.75 dia to 2.00 F'Bore 2.00 to 2.005/2.015	G.E. Turn Lathe	Hold in counter centrifugal Chuck	1.2
0025	Deburr all edges			5 mins.

Figure 3 A Sample of a Process Plan Found in Industry

The details in a process plan can have a profound impact on the cost of a product and this makes many manufacturers justifiably cautious about change. Employees that do process planning are often long-term employees with a great deal of experience. Companies recognize that it is difficult to capture the knowledge of these individuals. One of the first types of Computer Aided Process Planning (CAPP) systems is generally classified as Variant. These systems 'look-up' plans created previously and allow them to be edited to suit new parts. These systems have not replaced the process planner, but they have increased productivity,

decreased costs, and reduced mistakes. To date these have been the most successful planners used in industry [Nolan, 1989].

Generative CAPP systems have more potential than the Variant systems, but are still undergoing development within the research community. At their simplest, Generative systems help a designer select operations. More complicated versions promise to fully automate planning. But, to replace the knowledge of a process planner is not a trivial task. Hence, the techniques of Artificial Intelligence (AI) have become very popular in CAPP.

Many of the Generative systems are limited to small domains of manufacturing, such as machining, assembly, etc. This is necessary because of the difficulty in capturing knowledge for performing tasks. Systems of restricted production domains are of limited use to a manufacturer. There is a need for a CAPP system is required that plans at a higher level and then, in turn, activates lower level CAPP systems for more detailed planning.

Within process planning the source of the plans is the key method of classification. The three basic classifications are Variant, Generative, and Automated [Chang, 1990]. A Variant CAPP system reuses or references standard process plans created previously. Old designs and their plans are recalled by matching abstract codes for the new designs to those of previous designs. The process plan from the old design is then customized for the new design [Nolan, 1989]. In this mode the CAPP system is essentially a plan editing system. This method can be fast to set up, but can be much slower when creating new process plans.

In a Generative CAPP system there are no stored plans. Each new plan is created using techniques including dedicated algorithms and Artificial Intelligence.

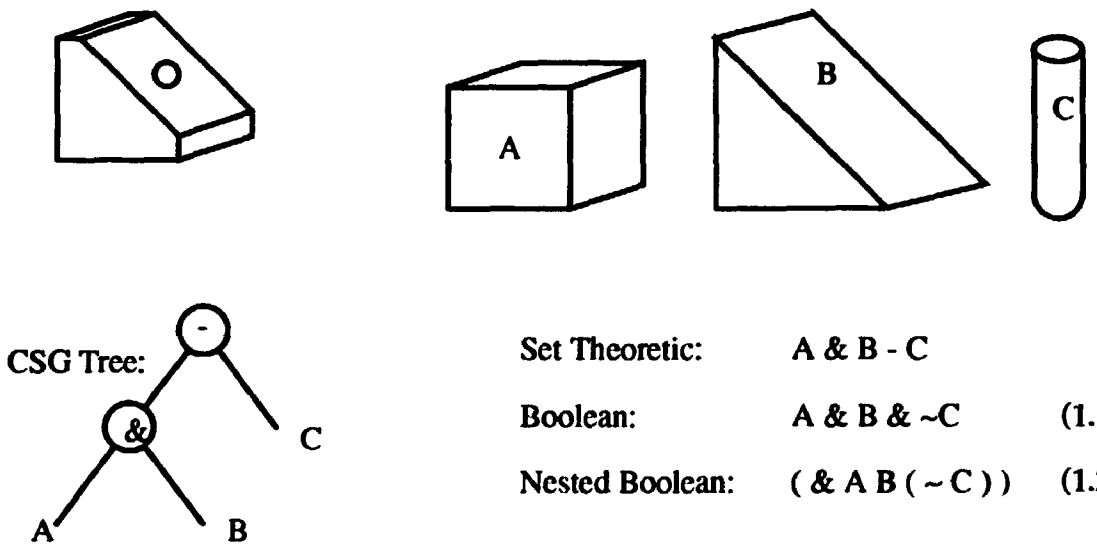
These systems take longer to set up, but are much faster when executing. Nolan [1989] states that these systems have only been successful for very limited cases with well defined planning problems. The third method, as suggested by Chang [1990], is Automated CAPP. This describes a generative CAPP system that runs without operator intervention. Naturally this system is more theoretically demanding, and must have extensive error checking and plan verification functions to replace those of the human planner.

This thesis describes a system called BCAPP (Boolean Computer Aided Process Planning) that is capable of planning at a high-level based upon design information. This system looks for clues about which rules (and consequently technologies) are useful. These rules operate on a design stored with Boolean equations of sets, where the sets describe primitive geometries. The planner is not intended to develop all of the detailed operation parameters for the production steps, but it will easily develop high-level plans with mixed production domains. A CAPP module for detailed plans of operations was also created for verification.

The interested reader looking for introductory material on Process Planning is directed to Muchnicki [1987], Wang and Wysk [1988] and Nolan [1989].

1.3 AN INTUITIVE EXAMPLE

To aid the reader, a summary of the approach is given here before the details are presented. To begin with we have a design for a part as shown below with the final geometry, the primitives, the Boolean expression that describes it, and some equivalent representations.



Set Theoretic:	A & B - C	
Boolean:	A & B & ~C	(1.1)
Nested Boolean:	(& A B (~ C))	(1.2)

Figure 4 A Sample Part and Some Theoretical Representations

Although the Nested Boolean representation is used in this thesis for BCAPP, and is equivalent to the other models, the Boolean equation will be used for this descriptive example.

First the reader is asked to look at the equation (1.1), and notice that certain equation forms suggest certain operations. If we look at the '~C' term ANDed with the other primitives, this tends to suggest machining out of some internal feature such as the a drilling operation. In this case the two other primitives that are ANDed suggest that one is a piece of stock and the other primitive defines some external features to be machined off (as well as other operations). The following steps show how a simple planner would use this approach.

1. a) Start with equation 'A & B & ~C'.
2. a) Perform a drilling operation.
   b) Remove '~C' to get remaining equation of 'A & B'.
3. a) Perform a milling operation to remove material outside wedge, but in the block volume.
   b) Remove 'B' from the equation to leave only 'A'.
4. a) Cut block of dimensions for 'A' from stock.
   b) Remove 'A' from the equation to leave an 'NULL' expression.
5. a) Quit because there is no design equation left to interpret.

A previous criticism of trying a method like this is non-uniqueness, but this has been used to advantage. For example, if we take the previous design equation (1.2), and rearrange it using the Laws of Boolean Algebra, we can get the equivalent expression (1.3).

$$\sim(\sim(A \;\&\; B \;\&\; \sim C)) \qquad (1.3)$$

The double negative can suggest a molding or casting type process. In effect the term,

$$\sim(A \;\&\; B \;\&\; \sim C) \qquad (1.4)$$

describes the molding cavity (everything around the part 'A & B & ~C' is solid, and the space for the part is now the only void location), and the second negation indicates an injection of material into the void location. We can also observe that the primitives 'A' and 'B' would generally enlarge the cavity volume, whereas '~C' would create an obstruction when milling the cavities. It is these sort of clues that make this method well suited to the abstract nature of Artificial Intelligence (AI) planning. A similar example to this sort of reasoning can be seen in Shpitalni, et. al. [1991].

1.4 RESEARCH OBJECTIVES

The main objective of this thesis is to develop a computerized process planning system that generates near optimal process plans for mixed production technologies. The system will be based on product models described by Boolean equations.

As directed by the main objective, a more specific set of objectives were selected to give the research some direction and criteria for evaluating the final success of the approach. These are given in the short list below:

- Be able to recognize alternative production technologies for manufacture of product features. This requires the planner to be able to consider multiple planning domains.
- Be able to produce alternative operations for each feature.
- Allow some degree of innovation in the process plan,.
- Permit a structure that allows feedback of production problems to the process planner.
- To allow the computer to reduce the knowledge barrier between process planning and production.
- Minimize human effort and intervention when process planning.
- Be capable of accepting new manufacturing technologies without fundamental changes in the process planner.
- Be able to optimize process plans.
- Handle all products.
- Simplify the problem of recognizing features from the design.

Many authors have used some or all of these goals when developing process planning systems, as will be seen in the literature review.

1.5 STRATEGY OF THESIS

An overview of the theory and software is presented here to help the reader keep the organization of the thesis in perspective. At present, BCAPP uses information from two different sources to find process plans. One input is the design definition. This definition is in the form of Boolean equations that act upon sets, as

seen in the intuitive example. The approach to product modeling is described in detail in chapter 3. The second source of information is the manufacturing resources definitions. These are in the form of rules that describe available processes and requirements. Chapter 5 covers the details of modelling and selection of the resources.

Both the design and rule files are used by the process planner. The planner is described over 3 chapters. The fundamentals of CAPP systems are covered in Chapter 2. In Chapter 4 the basics of Artificial Intelligence Planning systems are covered. Finally, in Chapter 5 the implementation for the BCAPP planning system is presented. The major contribution of the thesis is; the design representation in Chapter 3 and the rules and planner in Chapter 5. Chapter 6 describes a system that acts as a post processor for the high level planner. The material covered in Chapter 6 represents the role of other process planning systems that are designed for specific manufacturing technologies. Finally, Chapters 7, 8 and 9 present examples of the planners successes and failures, discuss the results, and suggest future work.

The layout for the software is shown in Figure 5 below. The product description file is at present provided by the user in the format described in Chapter 3. Eventually this could be provided by a commercial CAD package with minor modifications. The technology rule file is also provided by the user. This rule file must be developed so as to describe all resources in a factory. At present this file would have to be created by a trained Knowledge Engineer. Eventually the rule file could be entered through some inductive learning rule generation software.

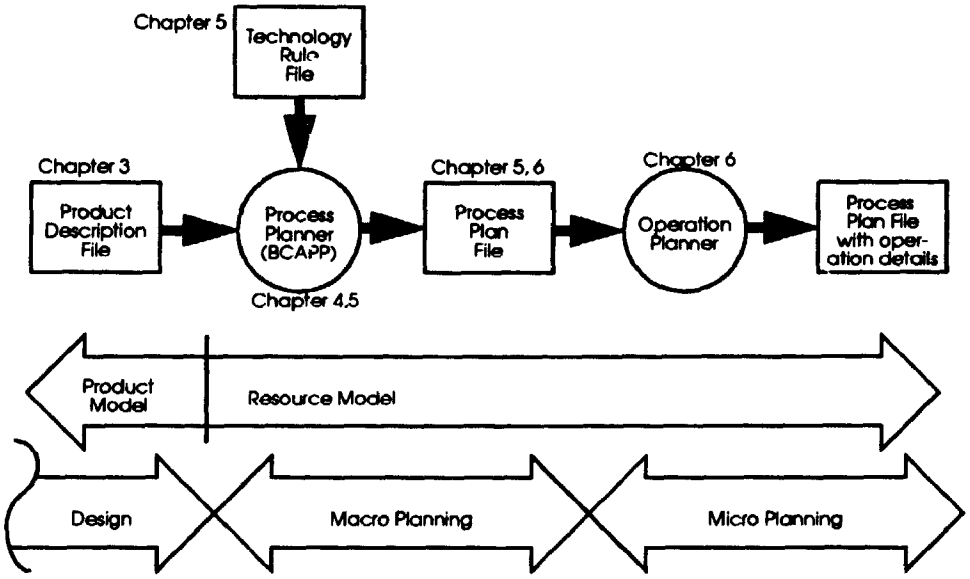


Figure 5 The BCAPP Approach to Process Planning

The process planning software (BCAPP) will accept the design file and the rule file, and attempt to find a process plan. The process plan steps will be added to the design file. At this point the process plan contains suggested operations and alternates, but it does not contain many details. (Please note that the contribution of this thesis is in the stages up to and including BCAPP, subsequent sections are provided for support only).

After the high-level process plan is passed on, it is picked up by the operation planner (as described in chapter 6). This software will use existing CAPP systems to provide details necessary for production. In addition to this, the software must determine the operation sequences and deal with production failures. If, for some reason, a process plan cannot be executed then it is possible to restart BCAPP to provide more alternatives.

1.6 THESIS ORGANIZATION

The following sections of this thesis begin with a review of process planning research that will be of particular use to readers not familiar with the current state of the topic (Chapter 2). After this, the area of solid modeling is reviewed briefly and then the modelling method used in this thesis is reviewed (Chapter 3). There is a subsequent discussion of the rules used for the design (Chapter 4). Next there is a discussion of the AI planning system that applies the rules to produce the process plans (Chapter 5). A short section is presented on operation planning to show how plans are detailed for factory use.

After the background is laid out for the thesis, a set of demonstration examples are given to show the strengths and weaknesses of the approach (Chapter 7). As the thesis concludes, a discussion of problems and future work is presented.

2.PROCESS PLANNING

2.1 INTRODUCTION

This chapter is intended to give the reader a perspective for the research issues in CAPP. It will begin with a discussion of modeling process plans and designs. Then it will discuss a number of approaches to process planning. The chapter will finally conclude with brief descriptions of other CAPP systems. If the reader desires more extensive descriptions than presented here, then they are directed to other publications [Chang and Wysk, 1985][Wang and Wysk, 1988][Chang, 1990][Alting and Zhang, 1988][Requicha and Vandenbrande, 1988]. More advanced research directions in CAPP were discussed by Ham and Lu [1988, 1989], and ElMaraghy [1993].

2.2 THE OBJECTIVES OF CAPP SYSTEMS

Eversheim and Shulz [1985] have done an extensive survey on the use of CAPP systems by industry. Their paper identifies a number of the popular factors considered in CAPP systems. The majority of the systems they surveyed included,

- product costs,
- processing time,
- setup time,
- text strings,
- cutting data,
- machine groups,
- searching similar process plans (Variant),
- determination of raw materials,
- selection of operations,
- segmentation of cut,
- rate settings.

There were other less popular features found in some of the other systems,

- machine adjustment data,
- partial operations,

- wage groups,
- fixture information,
- accounting information,
- plan editing.

While these do not represent all possibilities considered within a CAPP system, they do indicate the features that are of importance to industry. Eversheim and Schulz also identify the major domains of CAPP systems. The most popular is rotational parts, followed by prismatic parts, sheet metal, assembly, etc. Their figures indicate a definite lack of multi-domain CAPP systems. Their paper also indicates that 70% of CAPP systems are used in facilities that focus on single part and batch production.

A classification scheme for process planning was also suggested in Eversheim and Schulz [1985]. They divided CAPP functions into three major divisions, with sub-divisions;

- Process Planning,
  - determination of the raw material,
  - selection of operations,
  - determination of machining groups.
- Operation Planning,
  - selection of partial operations,
  - determination of technical data,
  - time and cost calculation.
- Management Functions,
  - retrieval of similar process plans,
  - duplication of process plans,
  - modification of process plan data.

They present a table of the surveyed systems using the classifications given above. A larger table was presented by Alting and Zhang [1988].

2.3 INPUTS AND OUTPUTS FOR PROCESS PLANNERS

Before discussing the details of how process planners operate, it is worth a

short consideration of the planning context. There are a number of inputs to planning systems as reported in Lenau [1992]. These vary widely, and can have drastic effects on the abilities of the planner. At the output there are a number of factors required for complete process plans, but previously established manual methods have provided a thorough understanding of the requirements.

2.3.1 Process Plan Modeling

It is necessary to specify what information will be requested in a process plan that is modelled in a particular system. While the decision seems obvious there are a number of fundamental factors to be considered. First, there is the process description that varies greatly between each technology and manufacturing facility. Second, and more challenging, is the relationship (such as operation precedence) between operations. When dealing with a simple linear plan all we have is a list of operations. If the process plan is non-linear it can have parallel or alternative operations and we must look to more sophisticated representations.

Assuming the non-linear process plan is stored in a graph, then the nodes and arcs could be assigned in a number of ways.

- Operations
- Operation Steps
- Resources (typically machines or capacity groups)
- State of the Work in Process
- Routings

When dealing with assembly process plans this can be represented with numerous Bill Of Material (BOM) and operation sheets. The state of Work In Process (WIP) is well suited to inspection planning for parts. Routings and Work orders are better suited to batch orders where the production conditions may vary. All of these methods represent various manifestations of the fundamental

data. The data itself has a few important data structure requirements,

- plans for various workparts,
- alternate plans for a workpart,
- a list of operations in a plan (with precedences),
- alternate operations for an operation in a plan,
- a list of steps for an operation (with precedences),
- alternate steps for each step,
- detailed data about each of the above levels of plan hierarchy.

Many methods to date limit these requirements for practical reasons. For example XPS-2 [Nolan, 1989] uses a process plan structure which is shown in Figure 6.

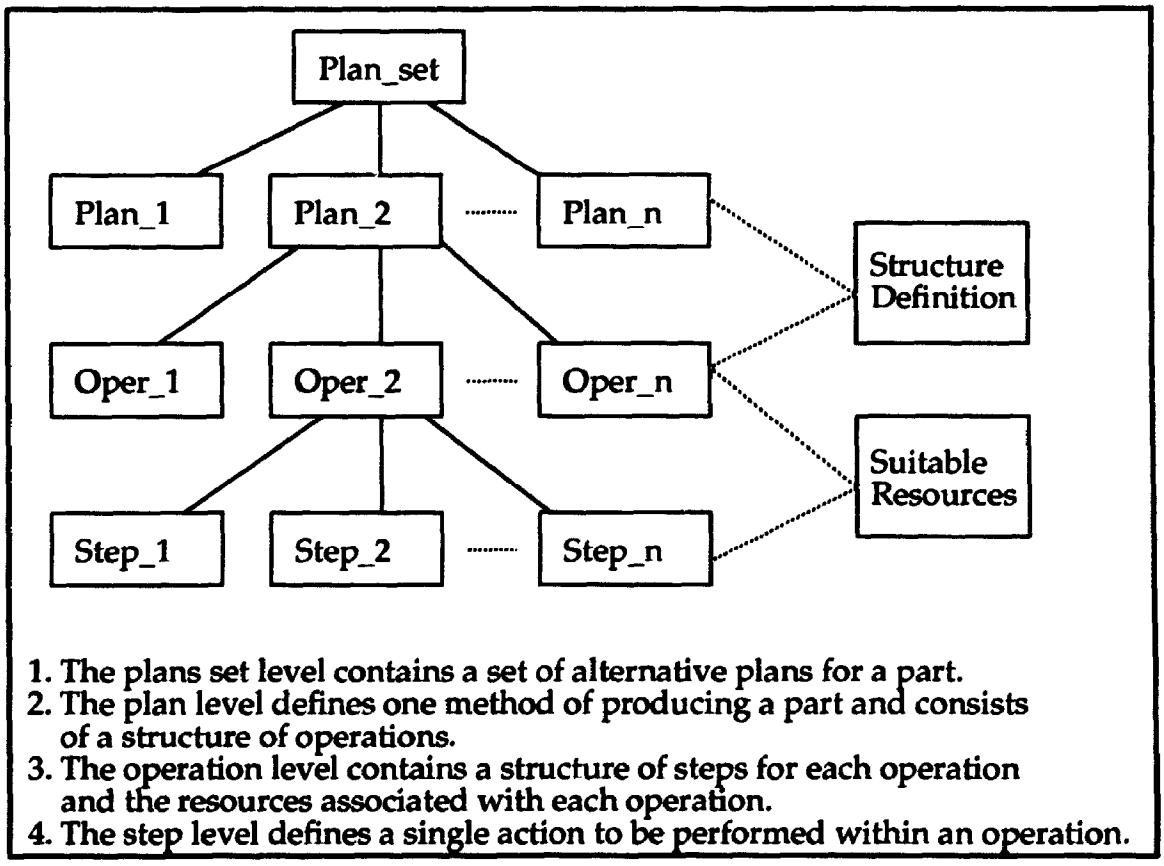


Figure 6 Process Plan Data Interface Specification (PPIS) Hierarchy

This structure is further augmented with sequencing graphs to determine the precedence of the various plan steps.

A method to indicate parallel, and alternative operations was reported by Tonshoff et. al. [1989]. They opted for a plan representation using Petri nets. Using this structure they were able to create plans that could be easily rescheduled when problems occurred on the factory floor that invalidated a process plan operation. A similar approach was adopted by Kruth and Detand [1992]. Mantyla [1993], suggested a process plan structure capable of defining process plans for the relationships between steps and alternatives for operations.

2.3.2 Product Modeling

The geometrical and non-geometrical knowledge about a product design can be difficult to represent formally. As a result many methods have been developed for representing a product design.

- Frames/Features
- B-Rep (Boundary Representation)
- CSG (Constructive Solid Geometry)
- Decision tree
- 2D drawings
- Group Technology (GT)

Frames can be used to represent data that is in a fixed format, such as machining features. The example in Figure 7 was taken from Kusiak [1991]. In the example the feature is a rectangular pocket, and dimensions are given with tolerances. Other information such as surface finishes, and wall thicknesses are also given for planning purposes. Finally, the connectivity of the feature to other faces on the part is defined.

```
(Pocket-1
     (Type Rectangular_pocket)
     (Length 2.5 Tol 0.01)
     (Width 1.4 Tol 0.01)
     (Depth 0.75 Tol 0.01)
     (Corner_radius 0.25)
     (Side_wall_thickness 0.250)
     (Bottom_wall_thickness 0.250)
     (Side_surface_finish 125)
     (Bottom_surface_finish 125)
     (Axial_clearance 0.75)
     (Radial_clearance 0.25)
     (Connecting_face f1)
     (Bottom_face f2)
     (Side_faces f3f5f7f9)
     (Fillet_faces f4f6f8f10)
)
```

Figure 7 Part of a Frame Based Feature Model

Frames are fairly rigid in their form, and are therefore commonly used with feature based models. The reader should be aware that there are also examples where frames and features are not used together. These models experience difficulties when a nonstandard (not available in the modeler) feature is required [Karinthi and Nau, 1989][Shpitalni, 1991]. Many systems have been developed with feature based representations, such as Kjellberg et. al. [1990] and Hummel and Brooks [1986]. There are also CAD systems capable of doing design by features, such as those by H.A. ElMaraghy [1991] and Arikan and Totuk [1992]. More recent work has implemented features in an object oriented framework [Marefat et. al., 1993] [McNeilly, 1993]. For some extensive lists of features see Wilson [1983] and ESPRIT [1990].

Boundary Representation (B-Rep) has become the most popular method of representation for solid modelers and, as a result, for new CAPP systems also. B-Rep models explicitly define all of the faces on a solid, as well as the connectivity to

other faces. But one problem with these models is that after the model is created, features must be recognized by examining sets of faces. It is difficult to associate feature type information with these models, because the feature is not present in the stored model. But, by far it is most difficult to extract a single feature when multiple features intersect.

Constructive Solid Geometry (CSG) is a useful approach that carries high level information, and can be easily interpreted into B-Rep easily at any time, whereas conversion from B-Rep to CSG is very difficult [Brun, 1989]. Early work in CAPP has seen a simple CSG model used by Woo [1977], but there have not been many other uses since then. This method has a problem dealing with non-primitive shapes. At present there are some approaches to incorporating splined surfaces into CSG models, such as Ling et. al. [1987] who modelled sheet metal bends with CSG. Varady [1986] had investigated incorporating splined surfaces into the Build solid modeler (an early research solid modeler). He supplied a normal vector to identify outside surfaces, and he used the surfaces for CSG cuts.

Some older methods rely upon input of data from 2D drawings. When used for rotational parts these can be very effective (turned parts can be considered 2D). But, when dealing with 3D objects, the model becomes ambiguous. Another older approach is based upon querying the user about the part in question. In effect the user answers questions from the computer and, as a result, a decision tree is traversed.

Group Technology (GT) is a very popular abstract method for representing designs. It is used to identify subsets or families of similar parts for the purpose of realizing common features. This improves the design and process efficiency through standardization. The GT codes can be made to represent products by

using any combination of geometry, manufacturing process, or function. The advantages of using such a coding system are standardization, re ''ıcing design proliferation, and it provides a basis for value engineering. To reduce the proliferation of new designs, GT can be used to perform searches for existing designs. The standardized plans also provide a basis for standard values for time and cost. In most cases GT codes are used in Variant systems, but in one case a Generative system was developed using a GT code as the basis for generating plan steps [Iwata et. al., 1987].

GT methods are highly sensitive to the specific GT code selected. Despite the attempt to create a standard code, the results always have to leave options for customization. Also, because GT codes are an abstract representation of a part, they do not carry enough detailed information for representing the part.

2.4 PROCESS PLANNING PHILOSOPHIES

There are a number of methods for planning. At the lower level we must consider decisions about databases, Artificial Intelligence, algorithms, etc. But, at the higher level we require decisions about the approach to planning, whether it is secretarial or creative. These decisions affect what the system will be capable of doing, and how it will be able to do it. This section starts off with a general discussion of the Variant and Generative approaches. A lengthy discussion of various methods for Generative planning follows this section.

2.4.1 Variant CAPP Systems

A Variant CAPP system is suited to dealing with a set of designs that tend to be variants of other standard designs. The basic concept is that new process plans are not completely redone when a new design is received. An existing process plan for a similar design is used as the basis for the new process plan. The old plan is

edited to compensate for the differences. This reduction of human intervention provides advantages in terms of efficiency, reliability and standardization. But, human intervention is still required for adapting the plan for the new part.

Nolan [1989] presents a thorough overview of Variant planning as well as making a good case for its use. There are also a number of papers discussing Variant planning systems [Emerson and Ham, 1982] [Carringer, 1984] [Mehta et. al., 1990].

Variant systems are probably best explained with reference to Group Technology (GT) codes. The decision to use GT should be determined by product variety. There must be a large number of parts that can be divided into groups, based on geometry, function, or production. The first implementation stage requires the development of a GT code. This code must then be verified. To verify the code, a sample of parts (10-20% of all parts is suggested) must be coded, then the results examined critically. If there is too much duplication, or too few similarities between codes, then the code should be corrected. After the code has been verified, the remainder of the parts should be coded. After all parts are coded, the GT system can be used for referencing stored designs and various associated information, like standard process plans. If standard process plans are also stored in the system, the GT code for a new part can be used to find a similar design and, consequently, a similar process plan.

The GT code is made up of a string of digits or letters that identify specific features of a part. The entire string of digits may be related or unrelated. If they are unrelated, that is a polycode, then the meanings of the code are found using a list of features for each digit individually. If they are related, referred to as a monocode, they can be represented with a decision tree. In the decision tree, each digit would represent a branch in the tree. This allows the code to take on a wider vari-

ety of meanings. Hybrid codes are also used that are a combination of monocodes and polycodes. One example is the first GT system developed by the Dutch, called MICLASS [Nolan, 1989] which uses a four digit monocode followed by a polycode. An example of a GT code is seen in Figure 8.

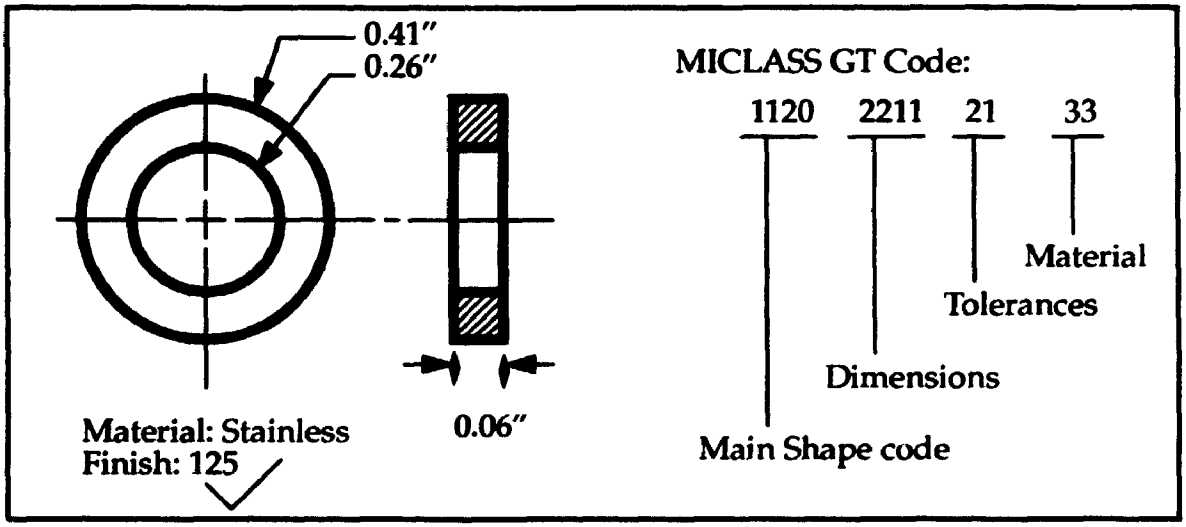


Figure 8 A Group Technology Example for the MICLASS GT Code

There are some suggested guidelines for selecting the digits in a new GT code,

- they must differentiate products,
- must represent non-trivial features,
- only critical features should be encoded,
- function should be encoded,
- every digit should be significant.

Parts can be encoded using process flow, tool axis, tolerance, function, material, and shape. If a GT code is poorly chosen, there may be problems with too many or too few matches for the new GT codes, or the code might be inflexible to technological change. If the Group Technology code has been well implemented, it is easier to identify standardized routings, estimate work content, assure quality, and maintain the database integrity.

There have been a number of approaches explored for applying GT to process planning. One of the common approaches is to follow the procedure for coding parts in a factory. After the parts are coded, standard part families and companion standard process plans are identified and stored using the GT codes. When a new design is developed a GT code is found for it. The GT code is then used to find the closest part family and, thus, a standard plan. Finally, the standard plan is edited to suit the new design.

2.4.2 Generative CAPP Systems

While the Variant method relies upon existing plans for the basis of process planning, Generative systems rely upon planning knowledge. Rules, algorithms, and heuristics about planning are used to recreate new process plans for each design. This allows process plans to be created with reduced human intervention, but at a large cost for the knowledge capture during implementation. The variety of approaches to this method have varied greatly.

This introduction has been kept short because the remainder of this chapter will deal with the aspects of Generative systems, and the issues related to the implementation of such systems.

2.5 METHODS FOR REASONING ABOUT PRODUCT MODELS

A product model can be represented many ways. These range from basic geometric entities (such as lines) to abstract semantic networks of surface transitions. The representation affects how the model may be interpreted. Joshi and Chang [1990] give a list of methods.

- Expert Systems
- Syntactic Pattern Recognition
- State Transition Diagrams

- Graph-Based Approaches
- Constructive Solids Approaches
- Decomposition Approaches

Methods for process planning can be complicated, and techniques range from simple algorithms to Artificial Intelligence (AI). There is a great deal of work making a case for AI techniques, but there are also papers dealing with the advantages of algorithms [Halevi and Weill, 1992]. The work in all areas suggest that a judicious combination of both can yield a very successful CAPP system.

2.5.1 Expert Systems

The most popular approach to recognition is using rules that are based upon human experience. The rules for an expert system are derived by a Knowledge Engineer who consults human experts. In general the rules are of the 'if <conditions> then <action>' form. As these rules are applied to a known set of facts, new facts are found. Most researchers use these rules to examine geometrical and manufacturing facts, in order to recognize manufacturing features. Some of the classic systems are excellent examples of rule based reasoning, such as GARI [Descotte and Latombe, 1984], TOM [Matsushima et. al., 1982], EXCAP [Davies and Darbyshire, 1984], XPLAN [Alting, et. al., 1988], etc.

The expert system approach is well suited to designs stored symbolically. Other expert system approaches use Frame based reasoning or Decision Tables. It is quite common to augment data in a frame with connectivity (derived from B-Rep models) to other faces and features to facilitate symbolic condition testing. A good overview of expert system use in CAPP is available in Alting and Zhang [1988].

2.5.2 Syntactic Pattern Recognition

By encoding the geometry of a part using a set of grammatical symbols, an abstract expression may be formulated. A matching process may then be performed between the syntax of the derived grammatical expressions and grammatical syntaxes of known features. An example of this could be a representation of a sheet metal part. The part is described with line geometry, as pictured in Figure 9. A basic set of grammars are define for the shape. These are then applied to the shape to obtain a code for use in pattern matching. Here we see code for the part begins at the top left corner, and works around in a clockwise direction.

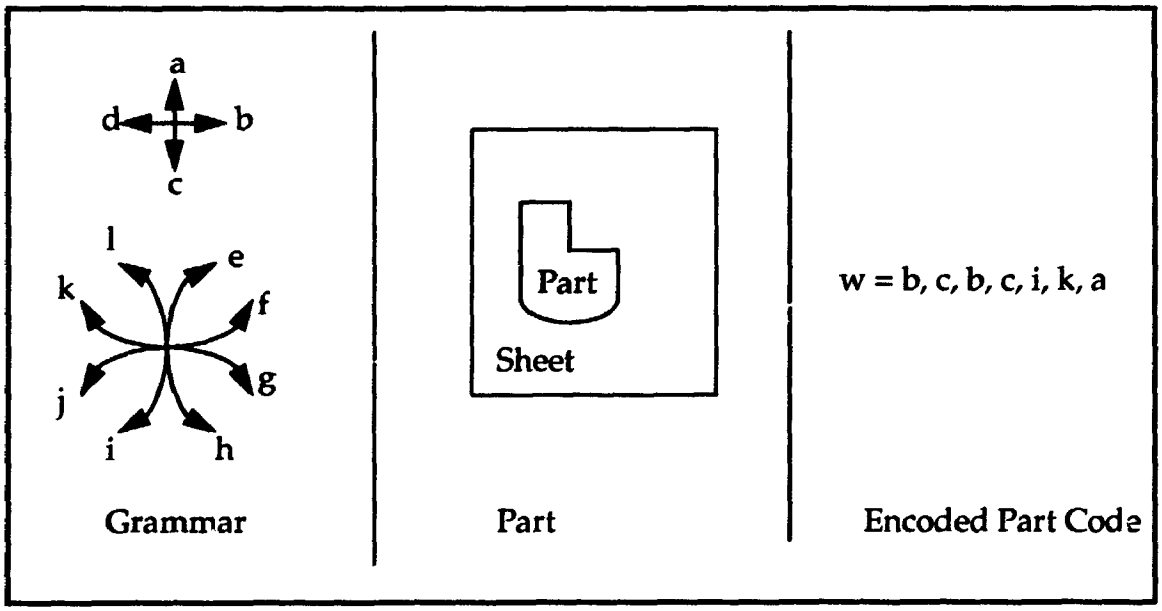


Figure 9 Syntactic Pattern Representation of a Sheet Metal Design

Kusiak [1991] summarizes the process of Syntactic Pattern Recognition as three distinct stages. Initially, an input string ($w = a_1, a_2, ..., a_n$) and an extended context-free grammar ($G = (V_n, V_t, P, S)$), form the input to a schema. The context-free grammar is composed of,

V_n	set of input symbols described in the input string
V_t	set of symbols which will comprise the solution
P	a set of production rules which will transform the input string S (containing symbols V_n) into an output string G which contains symbols only in V_t
S	starting set, as w above.

The syntactic pattern recognition approach is used by Liu and Srinivasan [1984] to select machine tools for manufacturing parts.

2.5.3 State Transition Diagrams

Whereas the Syntactic Pattern Recognition was based upon a large set of grammar, the State Transition approach uses a binary operator. When moving between previous features there are relative changes from concave to convex. These changes are marked with a state of 1 or 0 respectively. The profile of a part may be converted to a string, similar to the method shown in Figure 9. This is then fed to an automata which uses state transition diagrams to classify features. A State Transition Diagram may be seen in Figure 10. This example begins at the top left-hand corner, then proceeds in a clockwise direction. The only '1' in the sequence represents the concave transition near the top right-hand corner.

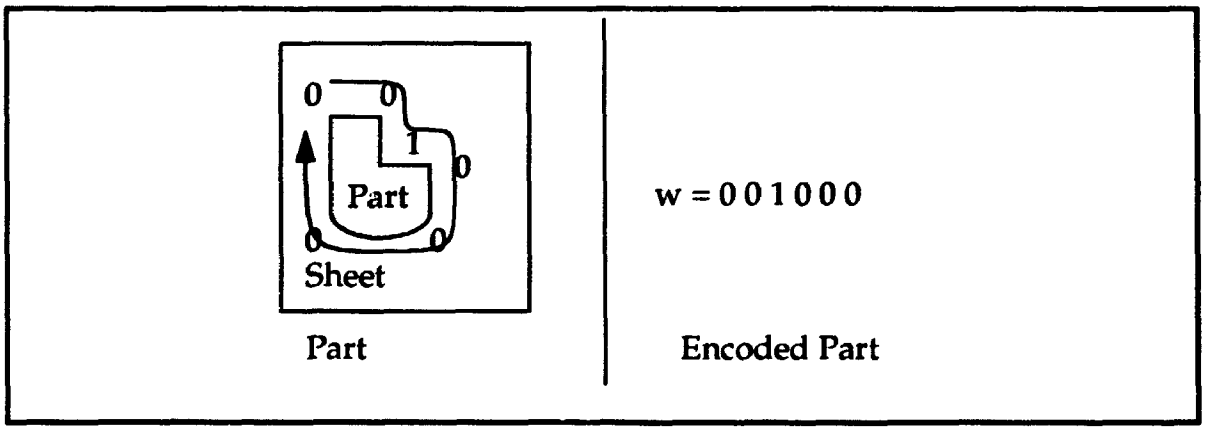


Figure 10 A State Transition Diagram for Feature Recognition

The code for a part is then examined for recognized sequences (sub-strings). For example, if the shape encoded is for a power screw thread, the code would contain a string such as '.....0011001100110011...' (This is because of the square thread). Therefore, recognizing the string contains a power screw thread, the planner would be able to select the appropriate tooling, and cutting operation.

2.5.4 Graph-Based Approach

While Syntactic Pattern Recognition and State Transition Diagrams are best suited to planar parts and rotated profiles, graph-based approaches are suited to full three dimensional parts. Kusiak [1991] describes the form of the graph as $G = (N, A, T)$ where,

N	represents nodes in the graph
A	represents arcs between nodes
T	represents attributes assigned to the arcs

This basic method involves capturing the design geometry in the form of a graph. The graph is then examined for patterns that match known features.

Joshi and Chang [1988] and Joshi, Vissa and Chang [1988] developed the Attribute Adjacency Graph (AAG). Each face is represented as a node. The arcs are the edges, or adjacency. The attribute of the arc is assigned a value of 0 or 1 depending upon a concave or convex transition, respectively. They then recognize features with algorithms and rules. The algorithms find candidates for indented features by examining the concavity of points. When potential features have been identified, the AAG is examined with production rules to compare features to feature graphs. They also deal with interacting features and there are three cases they discuss. The first is nested features, such as a pocket milled inside a pocket. In this mode, features are identified and then cut out of the AAG. The next case is inter-

secting features, such as two slots cut to cross each other. To detect this condition, virtual surfaces are created, then the features reexamined. Finally, when the forms of features interact, such as two stepped slots, the feature is dealt with by using virtual pockets. The important feature of these methods is the creation (or destruction) of new subvolumes to clarify feature details, and therefore simplify recognition. Their work was implemented for machined parts with indented features.

Sakurai and Gossard [1988] have an interactive system which the user uses to identify significant faces in a feature by selecting faces on a B-Rep model. They can search a B-Rep model to identify subsets that correspond to known features. deFloriani[1987] categorizes recognized features into depressions, protrusions, and holes from a B-Rep Model. The simple features are then grouped into compound features. The final model is a generalized edge face graph.

In general the syntactic pattern recognition, state transition diagrams, and graph based approaches are all suited to models that have some form of 2D or 3D B-Rep model. These methods can recognize features in an object, unless they overlap, but they are not able to reason past that stage.

2.5.5 Constructive Solids Approach

A very limited number of methods have examined the use of Constructive Solids Geometry (CSG), also known as the Set Theoretic method. The CSG method uses primitive volumes, and then performs additions, joins, and removes between them to form more complex objects. Eventually complex geometries can be built up.

In some early work in the area of solid modelling, Woo [1977] developed a CSG

based process planning method. His system allows design, process planning and then milling of cavities into parts. At the front end of the system is a solid modeler called CARVD. The system has two operations, ADD and REMOVE (a sub-set of the full CSG operations). The system puts the design into a nested algebraic expression, which is searched for cavity patterns. This search progresses from the most nested term to the least. The search involves determining 'hooks' between the terms in the algebraic expression. The system then uses interpretive rules (in the form of semantic nets) to match these to features (see Figure 11).

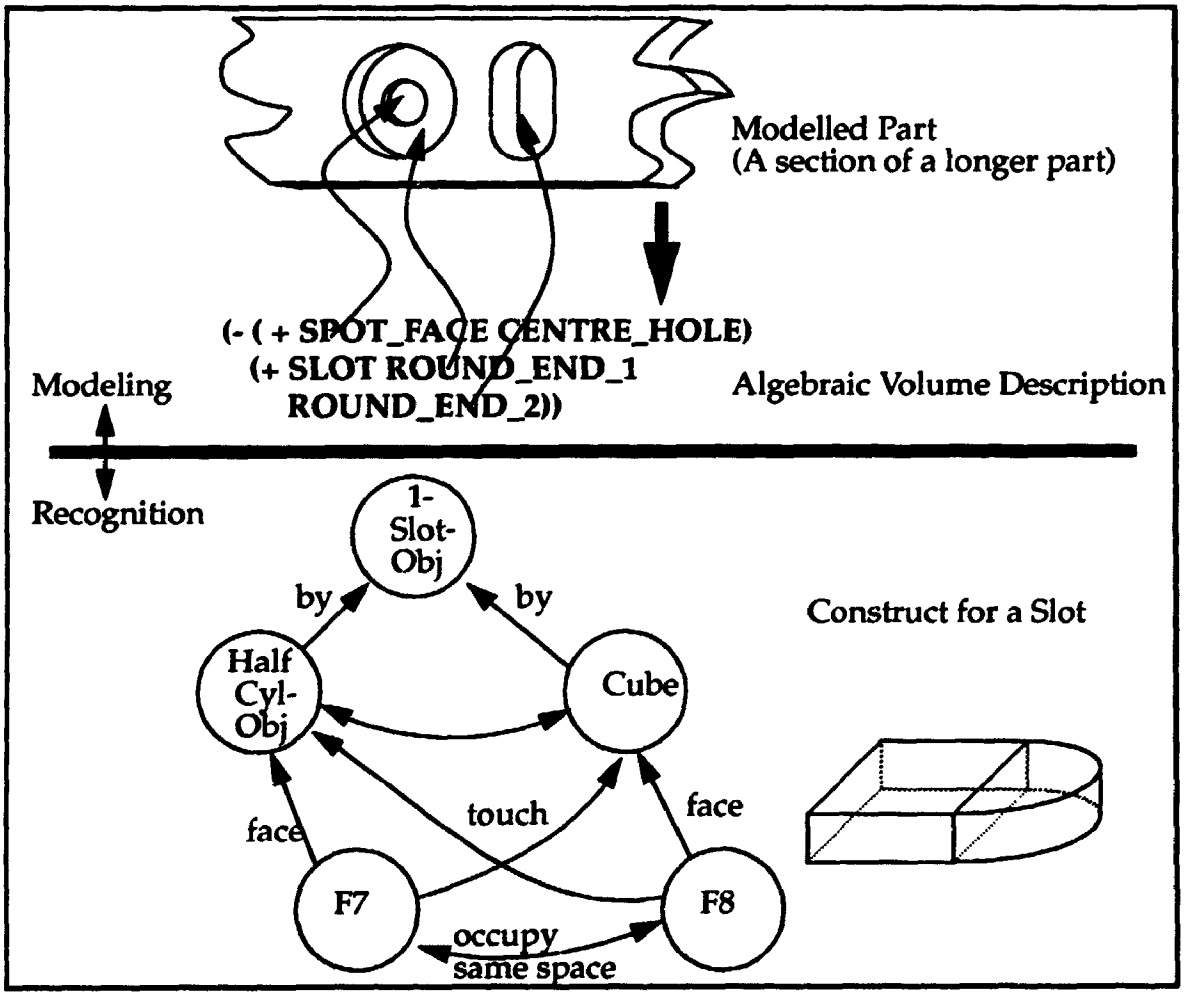


Figure 11 The CARVD Method of Feature Recognition
(Adapted from [Woo, 1977])

After a feature is chosen, the NC code is generated. The procedure of selecting an NC tool path involves an approach direction for the feature. The method of NC code generation is highly specific to cavity milling. This method of casting the expression in a nested equation form is similar to the representation used in this thesis.

Woodwark [1988] philosophizes about CSG approaches to Feature Recognition. He identifies the non-uniqueness of the CSG model as one of the faults of the method. He argued that the non-uniqueness of the method makes search spaces much larger, and therefore harder to search and prune. As a result, he suggested three ways to overcome these problems.

- Restrict the domain of the model, by restricting the range of primitives and orientations that they may assume (used in the work by Woo [1977]).
- To restrict the ways in which primitives may interact spatially.
- To restrict the allowable forms of set theoretic expressions which define the model.

By limiting the complexity of the model, it is possible to limit the complexity of the designs is also limited. Woodwark also suggests matching to shape templates. This method involves using feature templates that are applied to local parts of the CSG model. This is similar to traditional pattern matching, except that it is localized for specific features to save search time. He claims this method can fail when two or more features interact, thus causing the feature matching to be inconclusive.

The final approach proposed by Woodwark is a hybrid CSG and Boundary Representation. He discusses adding geometrical boundary information to the set theoretic model. In this way the benefits of CSG are maintained, while making the actual geometry available for geometrical reasoning.

Work has been done by a number of authors to recognize features using CSG pattern recognition. Kakazu and Okino [1984] developed a method for converting a set theoretic representation of a part into a canonical form, and then a pattern recognition approach was used to find a Group Technology code. Other work for manipulating CSG trees was done by Goldfeather et. al. [1986] who developed the CSG tree into a normal form for faster computer graphics. Lee and Jea [1988] also developed a method for moving single nodes up through a CSG tree for the purpose of eventually developing a system that recognizes CSG features recognizing features. They only provided a brief description of how a feature recognition algorithm would work.

2.5.6 Decomposition

The method of the previous section is successful, but assumes that the CSG operations are all subtractions of machined features. But, this is not always the case, sometimes a larger stock shape must be selected before features can be recognized. After the stock is selected, solid models can be decomposed into smaller subvolumes through iterative methods. These subvolumes may then be examined by a feature recognition stage. There are common methods often used for this technique. These methods are often based on some sort of differencing operation to derive sub-volumes.

2.5.6.1 Geometric Difference

Ruf and Jablonski [1990] discuss their integrated CAPP and PPC system. One of the modules for feature recognition is called Feature Recognition Extraction, Decomposition and Organization System (FREDOS). This module will take a design represented in B-Rep, and select a piece of stock to machine from. The boolean difference between the stock and the part is calculated. This remainder is

then decomposed into individual features. The features then specify a generic process plan, without machines specified.

2.5.6.2 Alternating Sum of Volumes (ASV)

A method of overcoming the non-unique properties of CSG representations has been suggested by Tang and Woo [1991a,b]. Their papers discuss a method of decomposing a solid into a set of convex solids. This is done using an iterative method of covering an existing part with a convex hull. The hull then has the basis part subtracted. The same operation is continued on the remaining solid until there is no solid left. The result is a set of volumes that may be resolved into a set of removable volumes. An example from their paper is shown in Figure 12. In the example, the part goes through a number of iterations to identify the primitive geometry components. At the bottom of the figure the primitives are collected and then simplified to material removal only. The very bottom of the figure shows a set of primitive shapes that may be removed (H_1' and H_3) from the stock (H_0) to produce the part.

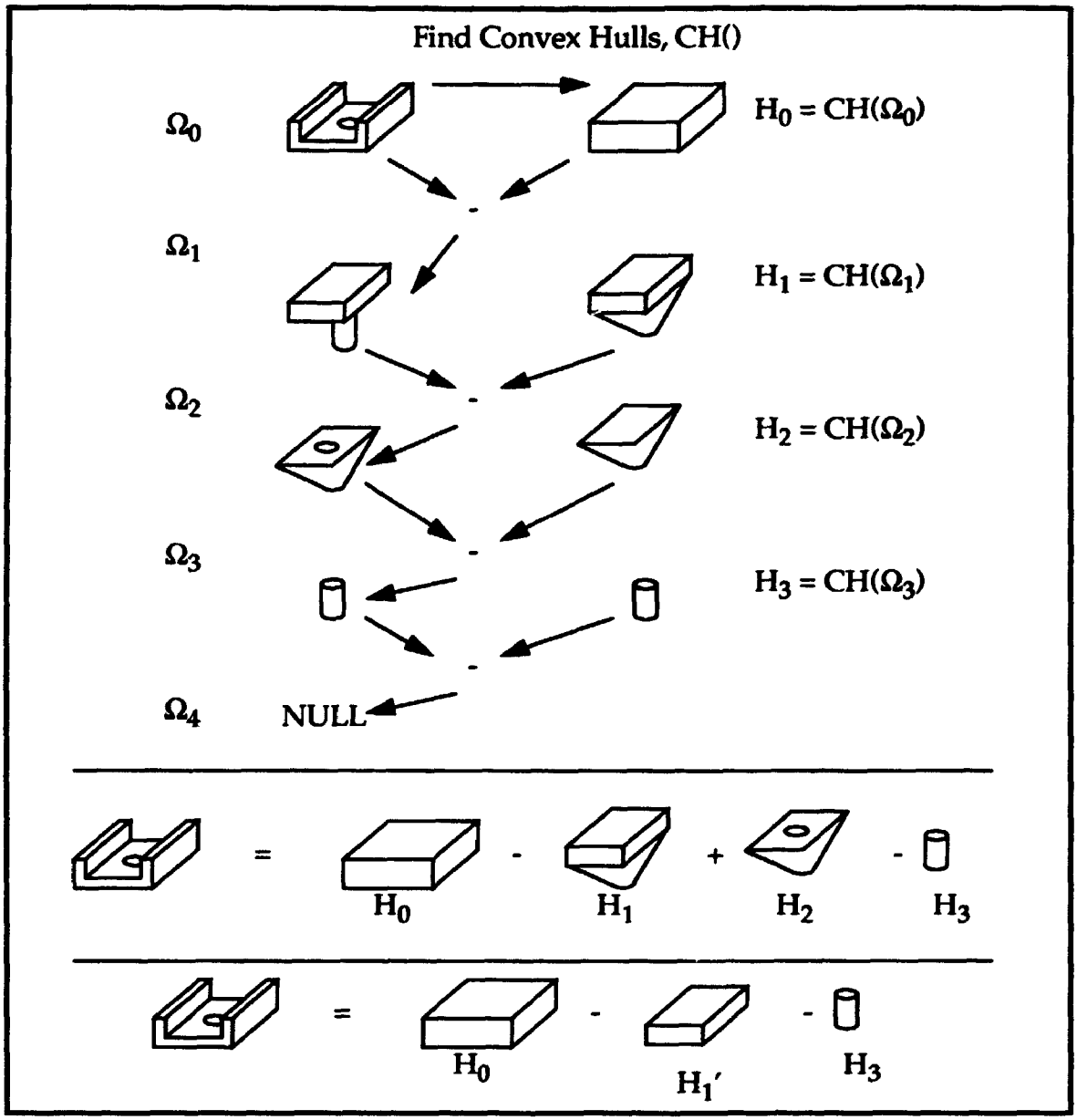


Figure 12 The Alternating Sum of Volumes (ASV) (the figures depicted here are not to scale for clarity) [Tang and Woo, 1991a,b]

Although this method will determine a set of machinable features, it has some problems which the authors also identified. Tang and Woo [1991b] refer to these problems as nonconvergence. And, considering that this method is iterative, the algorithm would continue indefinitely, as shown in Figure 13. The have proposed

an algorithm to check for nonconvergence based upon strong, weak, and internal hull vertices. They present a set of algorithms, in pseudocode, that deal with a Boundary Representation. These algorithms have also been developed to deal with non-manifold edges and vertices.

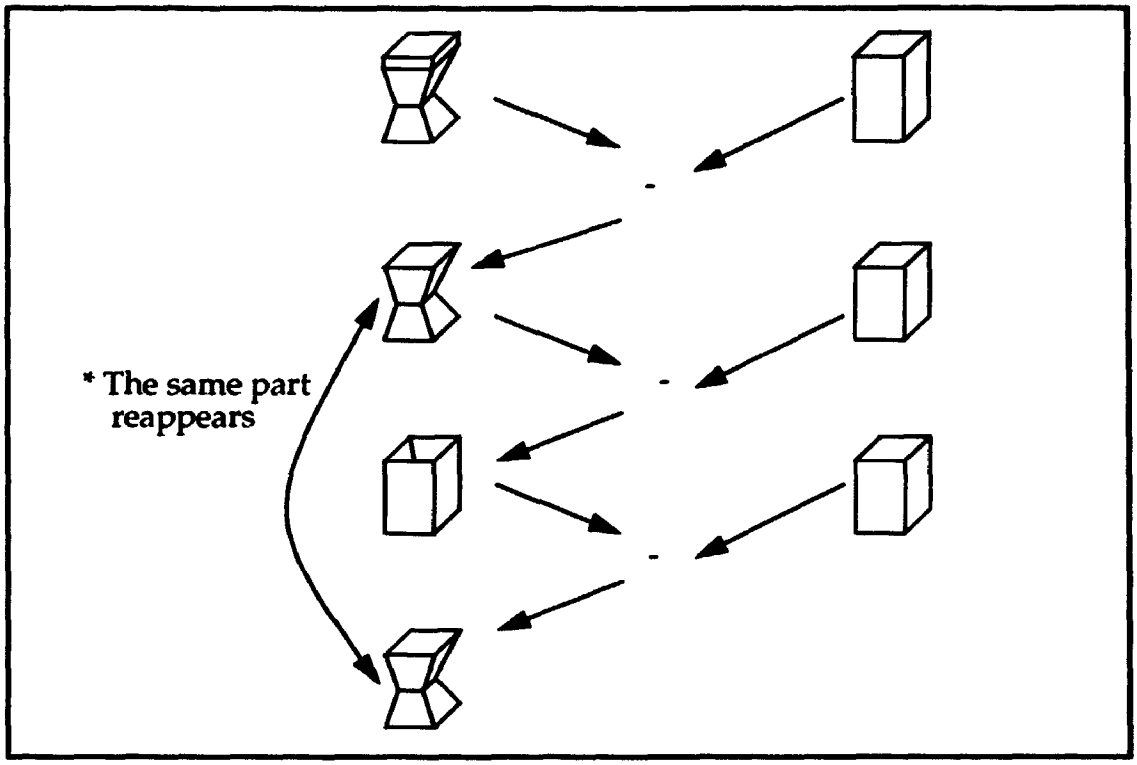


Figure 13 An Example of a Nonconvergent ASV

2.6 A PROCESS PLANNING EXAMPLE

In general, a decision must be made about the method of producing a part. Almost all previous CAPP research has limited itself to fixed domains, such as machining, turning, injection mold making, etc. This has simplified the planning process, but it does not allow for parts that are produced using a number of different manufacturing technologies. For the purpose of illustration this section will discuss a system for machining techniques. At the end of this section the reader should have an appreciation for what parameters are normally covered when doing operation planning.

2.6.1 Selection of Machining Processes

Machining is distinguished by the successive removal of material. The order of removal, the tools and fixtures chosen, and other factors all have a profound impact on the cost. As discussed before, a good mix of AI and algorithms will result in a more successful system, and this will be obvious throughout this section.

2.6.1.1 Machinable Volumes

A machinable volume is the amount of material that may be removed in one tool pass. If we consider a volume of material to be removed, it may then be divided into a number of machinable volumes. Processes are selected for removing the machinable volumes. Criteria which may be considered when selecting machinable volumes are; shape, size, dimensional and geometric tolerances, surface finish, etc. Many processes are also multi-step, and may require preparatory processes. Process selection is often done with Rules, Frames, and other methods (see the examples in Figure 14).

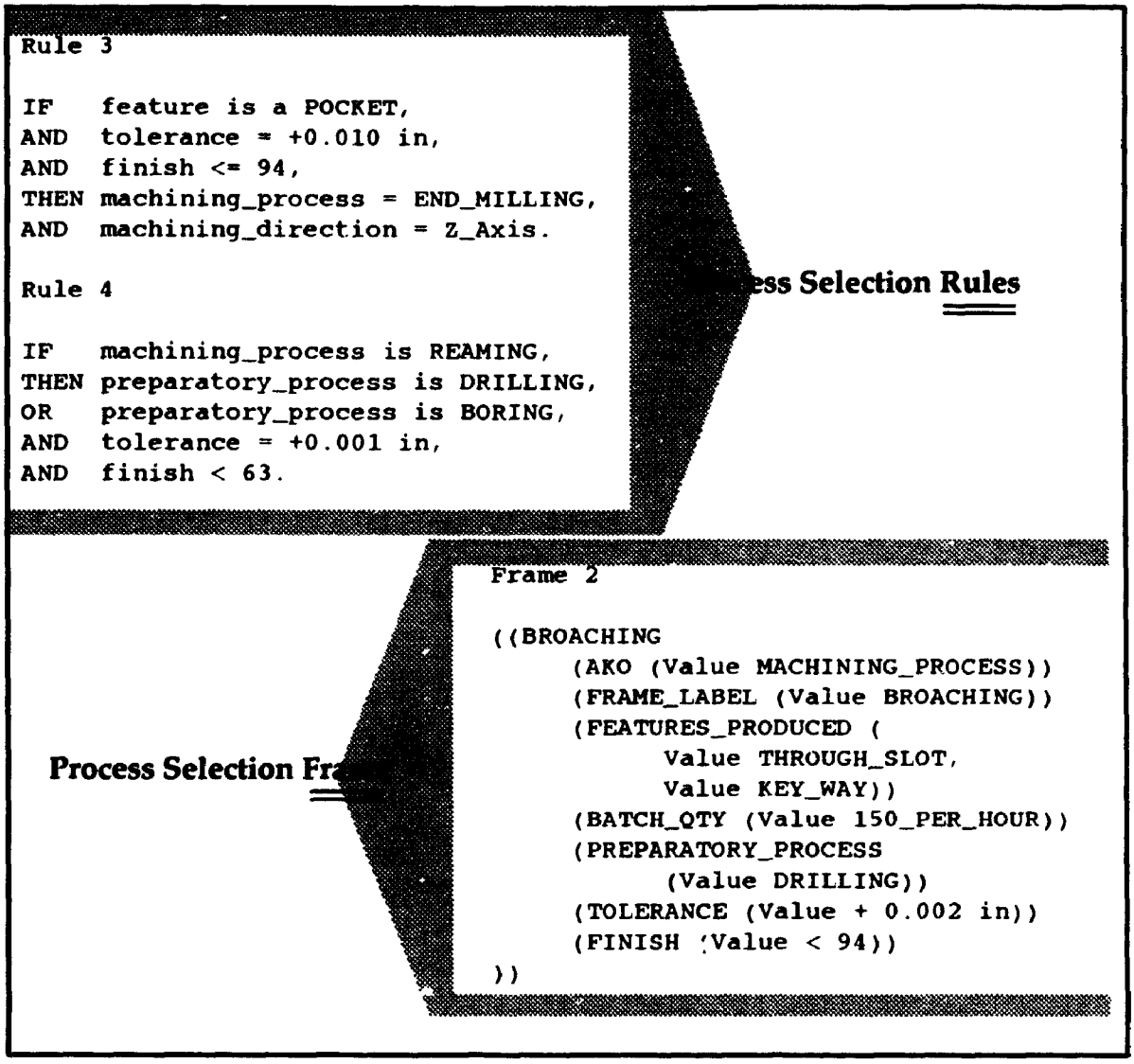


Figure 14 Rules or Frames for Process Selection
(Adapted from [Kusiak, 1991])

Similar rules are also used in XPS-2 [Nolan, 1989], GARI [Descotte and Latmobe, 1984], etc.

2.6.2 Selection of Machine Tools

There are many ways to choose machining details such as: tools, coolants, fixtures, etc. Methods for selecting include rules, frames, etc. (see Figure 15).

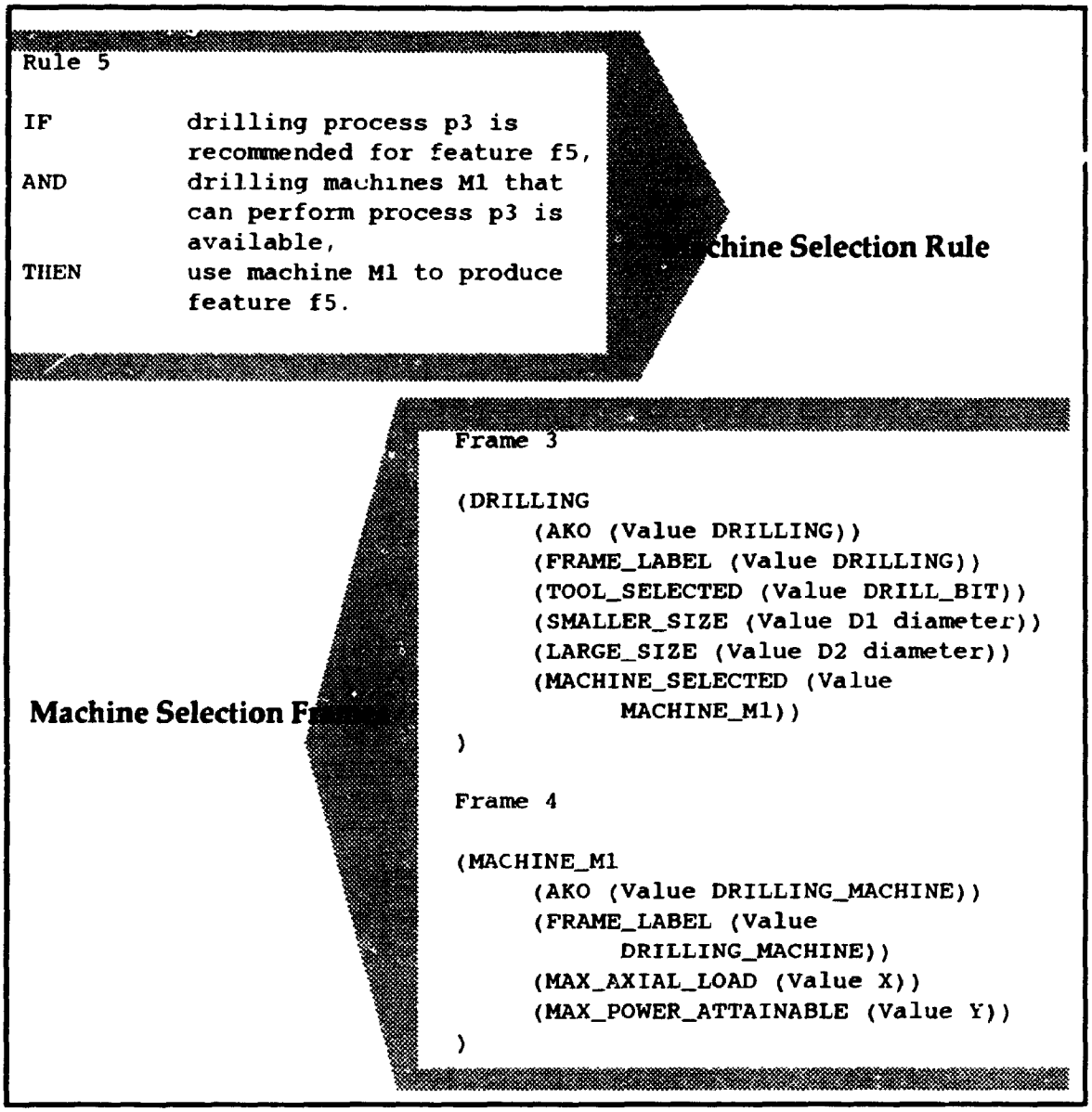


Figure 15 Rules and Frames for Machine Selection
(Adapted from [Kusiak, 1991])

2.6.3 Machining Optimization

The very nature of manufacturing is such that a manufacturer must survive through outperforming the competition. To do this there are a number of objectives the manufacturer will try to meet through optimizing some aspects of production. These objectives vary, and alter how decisions are made within the

factory. Process planning has a large effect on the cost of production. By considering optimization of processes during the planning there can be considerable savings.

The machining optimization problem has some fundamental elements found in all optimization problems. Cost tends to rule all decisions. This results from a number of interrelated factors,

- tool life,
- machining time,
- tool cost,
- etc.

Each of these can be estimated using approximations. For example, tool wear may be estimated using Taylor's tool life equation,

$$t = \frac{\lambda C}{V^{\alpha} f^{\beta} a_p^{\gamma}} \tag{2.1}$$

where,

t	tool life
λ, C	constants for tool/part
α, β, γ	constant exponents for tool/part
V	cutting speed
f	feed rate
a_p	depth of cut

Machining time is greatly affected by the number of passes. The single pass model is fairly simple, and can be formulated as a constrained optimization problem. The multi-pass model is similar, but has added parameters to account for the multiple passes. Kusiak [1991] discusses the model shown in Figure 16. Please note that the single-pass model is ignored because it may be easily derived from the multi-pass model.

t_{pr}	Total time for a prismatic part - machining, handling, and tool charge.
t_m	Machining time.
t_h	Material handling time.
t_t	Tool changing time.
t	Tool life.
C_{pr}	Production cost per part.
C_b	Setup cost for a batch.
C_m	Total machine and operator rate.
C_r	Tool cost.
N_b	Batch size. ▲ Single Pass Model
n	Number of machining passes. ▼ Multi-Pass Model
a_p^i	Depth of cut in pass i.
t_m^i	Time required for pass i.
C_{pr}^i	Production cost per part for the machining pass i.

Figure 16 Parameters of the Multi-pass Machining Model
(Adapted from Kusiak [1991, pp. 261-262])

Time Objective Function:

$$min\,(t_{pr}) = t_h + \sum_{i=1}^{n} (t_m^i + t_t(t_m^i/t)) \quad (2.2)$$

Cost Objective Function:

$$min\,(C_{pr}) = (C_b/N_b) + C_m t_h + \sum_{i=1}^{n} C_{pr}^i \quad (2.3)$$

$$C_{pr}^i = C_m [t_m^i + (t_m^i/t)\,(t_t + (C_r/C_m))] \quad (2.4)$$

Spindle speed constraint (for part and tool):

$$n_{w_{min}} < n_w < n_{w_{max}} \quad (2.5)$$

$$n_{t_{min}} < n_t < n_{t_{max}} \quad (2.6)$$

Feed constraint:

$$f_{min} < f < f_{max} \quad (2.7)$$

Cutting force constraint:

$$F_C < F_{c_{max}} \quad (2.8)$$

Power constraint:

$$P_m < P_{max} \quad (2.9)$$

Surface-finish constraint:

$$R_a < R_{a_{max}} \quad (2.10)$$

Cutting depth constraint:

$$a_{p_{min}} < a_p^i < a_{p_{max}} \quad (2.11)$$

Figure 17 Objective and Constraints of Multi-pass Machining Model

There are many approaches used for optimization. Challa and Berra[1976] used a gradient search method. Chang et. al. [1982] used dynamic programming, after putting some variables in the discrete domain. Kusiak [1987] presents a hybrid system for knowledge based, and numerical optimization of the problem.

2.6.4 Optimal Selection of Machinable Volumes

A large machinable volume will require multiple passes of a tool. To determine the sub-volumes, it should be decomposed using some geometrical methods. The example in Figure 18 shows how a part has been decomposed by extending the planes describing the surfaces.

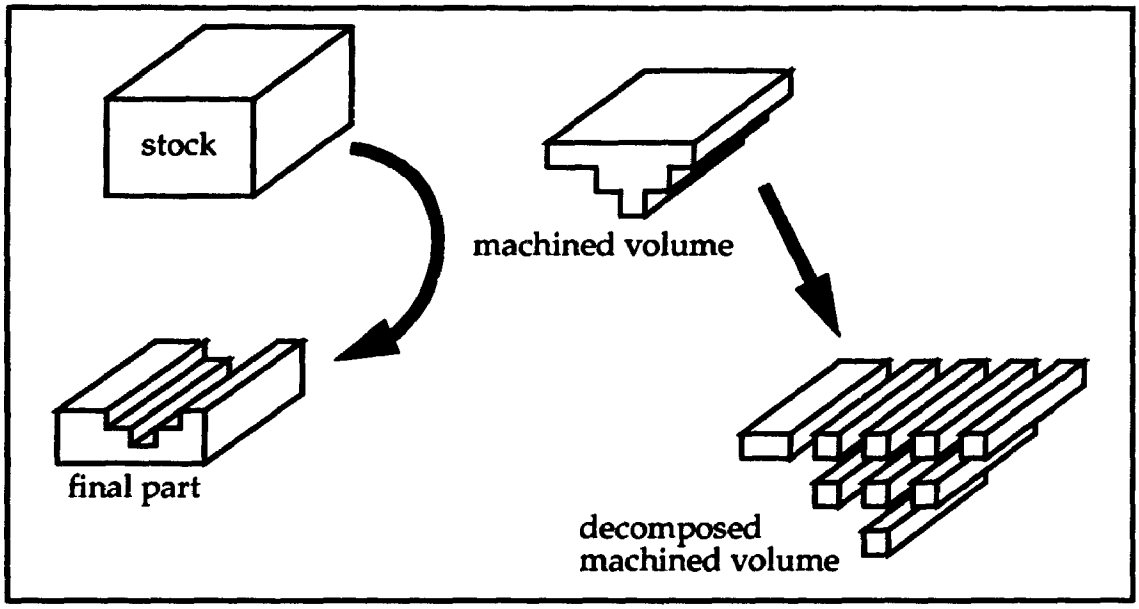


Figure 18 Decomposition of a Machinable Volume

This decomposition is easily accomplished if the planes existing in the part, are extended, and used as cutting planes, for the machined volume.

When a machinable volume has been decomposed, each of the volumes may be further sub-divided into elemental volumes. The fully decomposed volume is shown in Figure 19.

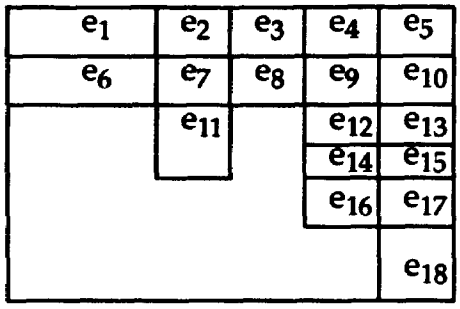


Figure 19 A Machinable Volume Sub-Divided into Elementary Volumes (Adapted from [Kusiak, 1991])

After dividing the machinable volume, the problem may be approached as an optimization problem. The list of variables in Figure 20 defines the decision variables involved with the problem. The Objective function, and the constraints are shown in Figure 21. This formulation is taken from Kusiak [1991]. He states that this formulation should be complete. He also states that some of the constraints and variables may be ignored in some problems.

I	Set of elementary volumes to be machined
J	Set of all machinable volumes
a_{ij}	1 if elementary volume e_i corresponds to machinable volume V_j, otherwise is 0
T	Set of available tools for machining current part
F	Set of available fixtures
N_t	Upper limit on number of tools to be used
J_t	A subset of machinable volumes, which tool t has been used for
J_f	A subset of machinable volumes, which fixture f has been used for
c_j	Cost of removing volume V_j
p_t	Usage cost for tool t
k_f	Usage cost for fixture f
N_f	Upper number of fixtures which may be used
x_j	1 if machinable volume V_j is selected, otherwise 0
y_t	1 if tool t is selected, otherwise 0
z_f	1 if fixture f is selected, otherwise 0

Figure 20 Decision Variables for Optimal Machinable Volume Selection (Adapted from [Kusiak, 1991])

$$min\,(Z) = \sum_{j \in J} c_j x_j + \sum_{t \in T} p_t y_t + \sum_{f \in F} k_f z_f \qquad (2.12)$$

such that,

$$\sum_{j \in J} a_{ij} x_j \geq 1 \quad \text{for all} \quad i \in I \qquad (2.13)$$

$$\sum_{j \in J_t} x_j \leq |J_t| y_t \quad \text{for all} \quad t \in T \qquad (2.14)$$

$$\sum_{j \in J_{tf}} x_j \leq |J_f| z_f \quad \text{for all} \quad f \in F \qquad (2.15)$$

$$x_j = 0, 1 \quad \text{for all} \quad j \in J \qquad (2.16)$$

$$y_t = 0, 1 \quad \text{for all} \quad t \in T \qquad (2.17)$$

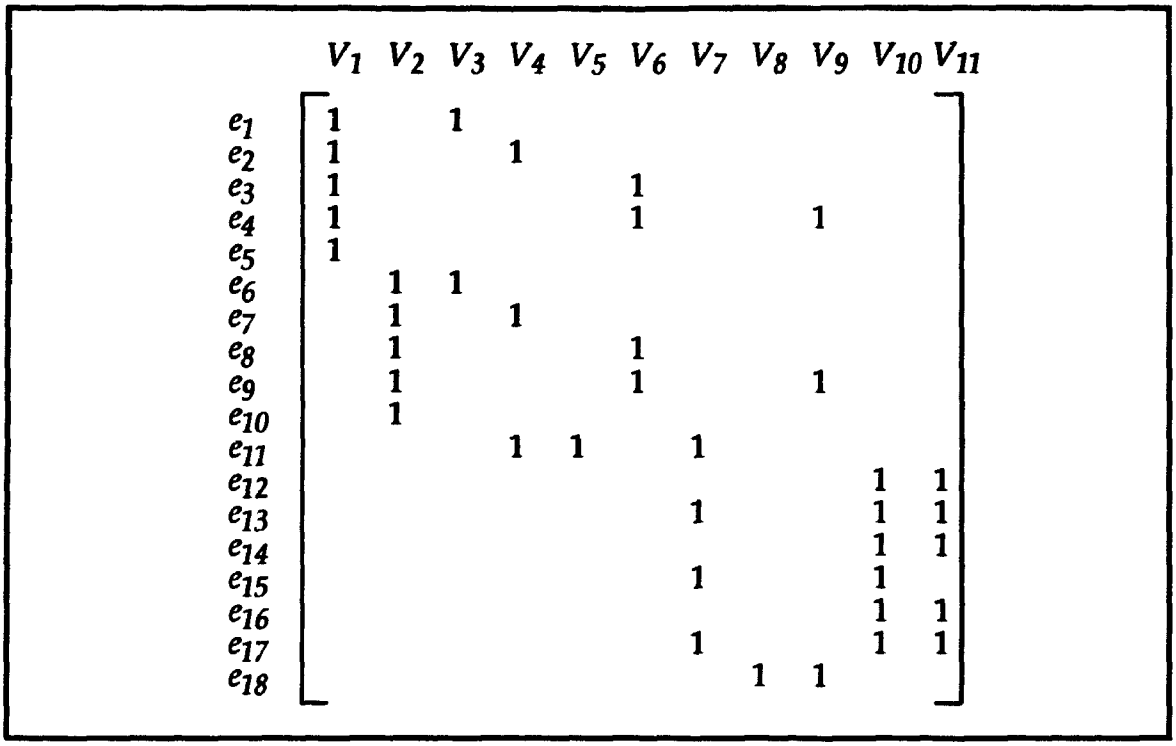
$$z_t = 0, 1 \quad \text{for all} \quad f \in F \qquad (2.18)$$

$$\sum_{t \in T} y_t \leq N_t \qquad (2.19)$$

$$\sum_{f \in F} z_f \leq N_f \qquad (2.20)$$

Figure 21 Objective and Constraints for Selection of Machinable Volumes (Adapted from [Kusiak, 1991])

The example in Kusiak [1991] goes on to show a machinable volume matrix, shown here in Figure 22. The matrix shows that there are a number of possible ways to machine the volumes (not all must be used). These volumes also correspond to costs, tools, and fixtures (shown in Figure 23). These also generate a precedence graph (shown in Figure 25).



	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9	V_{10}	V_{11}
e_1	1		1								
e_2	1			1							
e_3	1					1					
e_4	1					1			1		
e_5	1										
e_6		1	1								
e_7		1		1							
e_8		1				1					
e_9		1				1			1		
e_{10}		1									
e_{11}				1	1		1				
e_{12}										1	1
e_{13}							1			1	1
e_{14}										1	1
e_{15}							1			1	
e_{16}										1	1
e_{17}							1			1	1
e_{18}								1	1		

Figure 22 A Machinable Volume Matrix
(Adapted from [Kusiak, 1991])

$$C = [5, 5, 2, 3, 1, 4, 5, 1, 5, 6, 4]$$

$$T = [t_1, t_1, t_3, t_2, t_2, t_3, t_1, t_1, t_2, t_4, t_2]$$

$$F = [f_1, f_1, f_2, f_2, f_2, f_1, f_1, f_1, f_3, f_2, f_1]$$

Figure 23 Example Sets for Machinable Volume Matrix

In addition to this, the constraints that $N_t = 3$, and $N_f = 2$ are added for this example. The resulting solution is developed using unspecified optimization techniques, and is shown below in Figure 24.

The volumes selected are,

$V_1, V_2, V_5, V_8, V_{10}$

therefore,

$x_1 = x_2 = x_5 = x_8 = x_{10} = 1$

where the volumes are, (these include all volumes to be machined)

$V_1 = \{e_1, e_2, e_3, e_4, e_5\}$

$V_2 = \{e_6, e_7, e_8, e_9, e_{10}\}$

$V_5 = \{e_{11}\}$

$V_8 = \{e_{18}\}$

$V_{10} = \{e_{12}, e_{13}, e_{14}, e_{15}, e_{16}, e_{17}\}$

and, tools t_1, t_2 and t_4 are selected, thus,

$y_1 = y_2 = y_4 = 1$

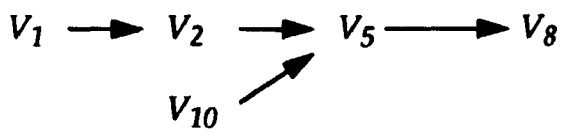
also, fixtures f_1 and f_2 are selected, thus,

$z_1 = z_2 = 1$

Figure 24 Solution for Example Problem

The solution produces an objective function value of $Z = 23$. Kusiak claims that this solution reduces cutting costs by as much as 8%, and reduces the tool and fixture count over traditional methods.

To continue this example, consider that the sequence of operations is not yet determined. This may be done using simple rules about volume relations. The relations are mainly based on i) accessibility of volume, and ii) datums for tolerances. Some examples of rules are given by Kusiak [1991], and shown here in Figure 25.



Rule Accessible Volume

IF machinable volume V_i can be accessed only after machinable volume V_k is removed,
THEN generate a precedence constraint V_k -> V_i.

Rule Tolerance Order

IF a plane p is an element of set V_k is a tolerance base for volume V_i,
THEN generate a precedence constraint V_k -> V_i.

Figure 25 Precedence Graph and Rules for Machinable Volumes (to Produce the Precedence Relations)

After the precedence constraints have been determined, the operations must be sequenced. This is done by first checking to see that there are some start nodes, with no predecessors, to ensure that the precedence graph is solvable. If it is then starting from the left the graph is expanded. The two possible sequences are $\{V_1, V_2, V_{10}, V_5, V_8\}$, and $\{V_1, V_{10}, V_2, V_5, V_8\}$. In his paper, Kusiak [1991] does not suggest how to resolve this dilemma, but does mention setup costs are one possible method.

The work of Kusiak is of importance to this thesis when considering what BCAPP must do. By itself Kusiak's work describes a complete operation planner for some forms of machining. But, without first selecting machining, and then specifying features to machined, it has reduced value. Therefore, after the process plan is completed by BCAPP, a system like the one described here should be used to select tools, cuts, speeds, feeds, etc.

2.7 CONNECTING PROCESS PLANNING TO SCHEDULING

As CAPP becomes more accepted in the factory environment, it will become necessary to integrate it with other functions. According to Ham[1988, 89] the advantages of such a development would be,

- improved efficiency in the information flow,
- improved quality of the process planning,
- reduction of the human errors,
- functional integration of process planning and scheduling, enabling a quick search for alternative solutions for optimization in the use of equipment and production control, and,
- flexible use of the different functions.

Ham [1988, 89] also asserts that there are certain key elements to a successful implementation of such a planner,

- a uniform product description based on proper features,
- the use of different modules for different functions,
- the use of a uniform user interface for each module,
- the use of a uniform data base for each module,
- the possibility of facilitating user interaction at the request of the operator.

Finally Ham [1988, 89] describes the issues involved in integrating CAPP with the related manufacturing functions.

- Planning Knowledge - A mixture of physical properties, and knowledge heuristics.
- Planning Activities - Must be integrated with Production Planning, and with Operation Planning, including physical process models.
- Planning Techniques - A collection of many process planning techniques (like Group Technology, Rules, etc.) must be used to avoid problems with each, and find the best plan possible.
- Planning Constraints - Technological, and other constraints, should be considered during planning, not simply used after planning to eliminate plans.
- Planning Feedback - Information which results from previous process plans must be used to issue new plans, and avoid similar mistakes in the future.

Some work has been done by various authors to address integrating CAPP

with PPC. For example, Ruf and Jablonski [1990] discuss their system called FIPS. The system contains three modules, FREDOS (Feature Recognition, Extraction, Decomposition, and Organization System), SSM (Static System Manager), and DRS (Dynamic Resource Scheduler). Essentially FREDOS will identify the machinable features in the part. The SSM will then assign all possible combinations of machines to the process plan. The DRS module will dynamically schedule the jobs into various machines, depending upon the current status of all machines. Although this system is for a limited domain, it displays the important concept of delaying resource assignment until required.

The CAM-I group began developing an interface specification for connecting CAPP to their factory management programs [a section by Sack, in Nolan, 1989]. The project began in 1986, at the UTRC at MIT, where they defined and reconciled terms, and developed a conceptual view of the standard database structure. They produced a Process Plan data Interface Specification (PPIS), which indicated interface subroutines used by the various computer programs. The diagram in Figure 26 shows how their system is used to integrate XPS-2 and a factory control system called MADEMA.

Other work has also been done by other groups on the general nature of integrated scheduling and CAPP. ElMaraghy [1992] examines bridging the data and functional gaps. Feedback of information about the resources was discussed by ElMaraghy, as well as Krause et. al. [1991] and Chryssolouris et. al. [1984].

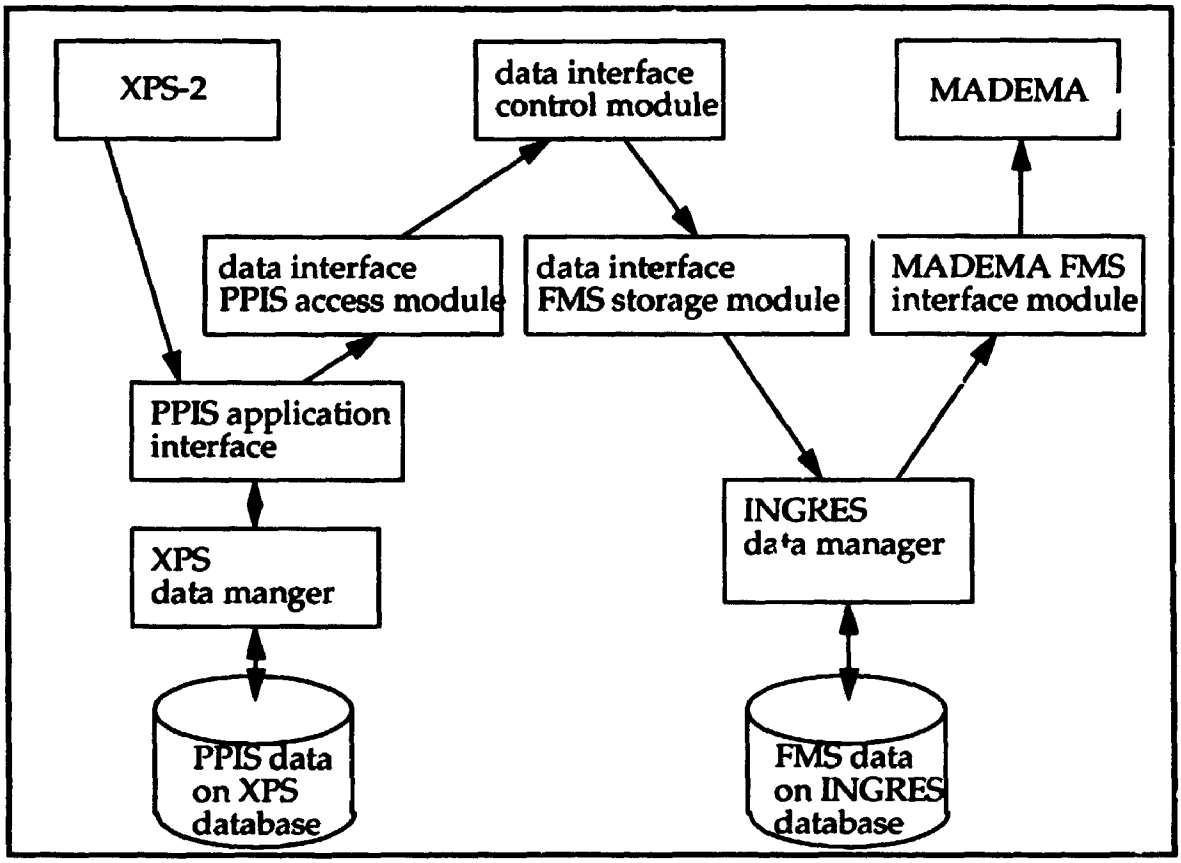


Figure 26 PPIS Interface Between CAPP and PPC

2.8 SOME WELL KNOWN PLANNING SYSTEMS

Most CAPP systems have been through many revisions which can be illustrated through the simple grouping shown in Figure 27. The order and relationships of these systems were found or deduced through the review of the literature discussed throughout this chapter.

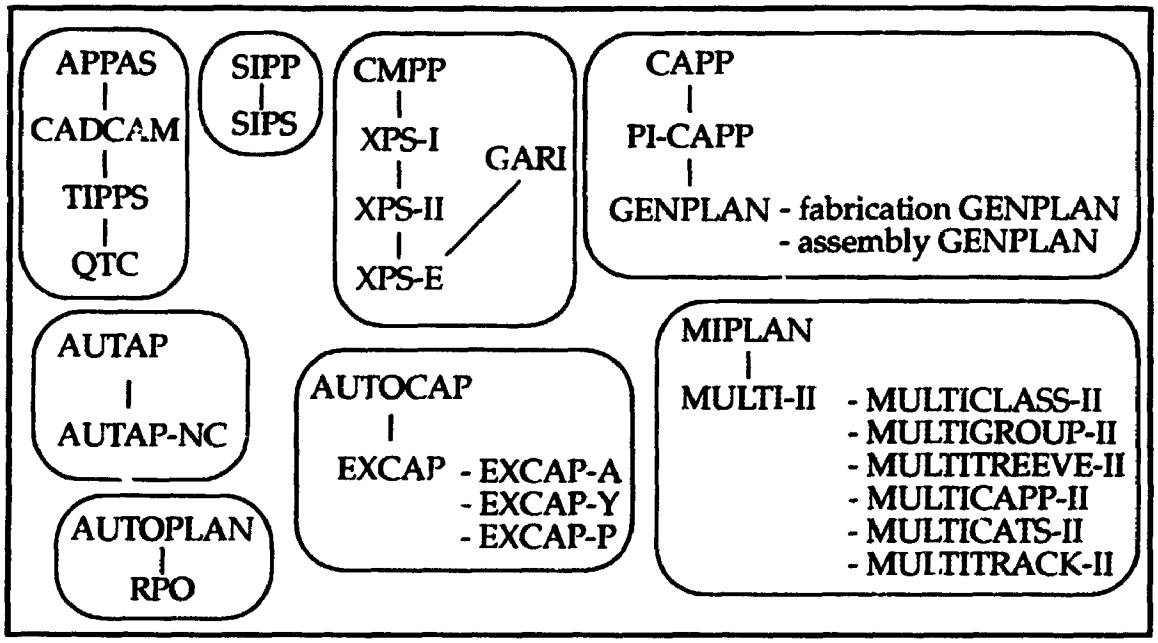


Figure 27 Genealogy of Some Families of CAPP Systems

The list below summarizes some systems for Process Planning. Longer descriptions are given for some of the more well known systems. Systems without names are referred to by the authors name. The summary of each system should provide an overview to many of the planning approaches taken to date.

APPAS, CADCAM [Chang et. al., 1982], TIPS [Chang and Wysk, 1985], QTC [Kanumurz et. al., 1988] [Anderson and Chang, 1989] [Chang, 1991] - APPAS (Automated Process Planning and Selection) models machined surfaces with special codes. Decision trees are used to direct the planner. It can handle multiple passes and operations for each surface. APPAS was extended into the CADCAM system to allow the use of an interactive graphic interface for design entry for direct use in process planning. This example was taken even farther in TIPPS that added a full graphical design system to allow the user to guide the CAPP system through a point-and-click interface. And, subsequent development lead to QTC (Quick Turnaround Cell). This system is based on a CSG type recognition system that plans from design to production.

AUTAP, AUTAP-NC [Eversheim et. al., 1980] - AUTAP will first find a stock shape, then develop a set of operations to produce rotational parts. Operations are sequenced, time estimates are made and tools are selected. AUTAP-

NC is a subsequent system capable of producing NC part programs to produce the turned components.

CADCAPP [Kamvar and Melkanoff, 1986] - Describes a system that accepts a drawing of a turned part from a commercial CAD system, then uses an algorithm to assign a GT code.

CAPES [Fujita and Oochi, 1989] - A generative system for prismatic parts. Uses a solid modeler.

CIMS [Iwata and Sugimura, 1985] - Uses interactive solids modeling with associated technological information. A process plan is generated using the finished part, and a blank. The plan is generated using production rules in a knowledge base. The planning is done in a five step process: machined surfaces are identified, appropriate tools are selected, machined surfaces are grouped, precedence constraints are generated, and finally, a process plan is generated. Sequencing is done using matrix methods.

CMPP [Dunn, in Nolan [1989]] [Parks et. al., 1989], XPS-I [Pavey et. al, 1986] [Groppetti, 1986] [Austin, 1986], XPS-II - For rotational, and other domains such as grinding, based upon representation. The planner analyses the machinability, then produces plans. This system contains the language COPPL described earlier in the chapter for manufacturing knowledge. A variety of research has been done on applying this system, such as accepting IGES files as input to eventually generate NC code [Knutilla and Park, 1990].

CUTTECH [Barkocy and Zdeblick, 1984] - Uses information from the user about features, machines, materials, etc. Generates a GT code for each feature, selects machine tools, cutting tools, etc. using rules from a knowledge base. A second knowledge base is used for selecting operation parameters like speeds and feeds.

[Delbressine, 1989] - A system to develop a part using stock that is transformed using manufacturing objects based on swept tool geometries.

EXCAP [Davies and Darbyshire, 1984] [Joseph and Davies, 1990] [Davies et. al., 1985] [Darbyshire and Davies, 1984] - Uses a limited description for rotational parts. Planning is divided into two steps, sequence planning, then operation planning. Rules are backward chained to prove hypothesis. If accepted the geometry is modified, and planning continues. This stage constructs a tree where the root is the finished part, and the leaves are blanks. The final tree has branches with certainties associated with each. Plans can be easily generated from the tree, and if rejected, alternate plans are easily generated. A further exploration with an application was done by Joseph [1989].

EXPLAIN [Prabhu et. al., 1990] - A feature based system that generates process plans for turning using rules.

EXPLAN [Warnecke et. al., 1989] - Describes a generative planner using a graphical interface.

FEXCAPP [Lee et. al., 1989] - Uses connectivity graphs and then does pattern matching to find features.

FLEXPLAN [Lampkemeyer et. al, 1991] [Tonshoff et. al., 1989] - Uses Petri nets to represent non-linear process plans. FLEXPLAN can store, generate, and schedule for non-linear process plans.

FRAPP [Henderson and Chang, 1988] - A system that does B-Rep based feature recognition, then some process planning based on the features.

GAPP [Laperriere and ElMaraghy, 1992] - An assembly planner using part connectivity and an A* search.

GARI [Descotte and Latombe, 1984] - Parts are described with feature descriptions, dimensional locations and tolerances of features, and other information like materials and finishes. An initial loosely constrained plan is developed, then a hierarchical planning system is applied in an iterative manner to suggest operations, propagate constraints, and backtrack.

GCAPPS [Pande and Palsule, 1988] - A system for feature based design of rotational parts.

Genoa [Held and Juttner, 1991] - Uses rules to generate plans in a limited domain.

[Han et. al., 1987] - A semi-generative system.

[Henderson, 1986] - A method for recognizing holes, slots and pockets. Faces are identified, then their relationships are determined, and converted to solid features with a connectivity graph.

Hi-Mapp [Berenji et. al., 1986] - Parts are represented with feature descriptions, locations, tolerances, and other manufacturing information. An abstract, but correct plan is generated. A hierarchical planning system is applied, which operates using priorities. The knowledge base contains rules for selecting processes, recommending cuts and machines, and recommending other user defined options.

KAPPS [Iwata, 1988] - An expert process planner for machining using over 600 rules.

KCAPPS [Wei et. al., 1990] - An AI generative system with an interactive interface for defining manufacturing attributes. Uses CSG subtractions of features from various sources of stock.

[Lee and Wang, 1990] - Does robotic assembly planning. They use features and rules to get a) largest feature stability, and b) heaviest part, and then rank the assembly order. It then generates robot path parameters for a complete off-line programming system.

MicroCAPP/GEPPCAPP [Wang and Wysk, 1985 a b] [Wang, 1984] [Cachia and Vajpayee, 1986] - Complementary systems that have MicroCAPP, a code based variant planning system, and GEPPCAPP, a decision tree based generative planner.

MiniCIM [Shyu et. al., 1987] - A complete design to manufacture system for turning.

PART [vanHouten et. al., 1984] [Jonkers and Kals, 1992], PART-S [deVin et. al., 1992] - A generative system that does both process and production planning. Another implementation is PART-S for doing sheet metal planning.

PC-CAPP [Pande and Walvekar, 1989] - A feature based CAPP system for prismatic parts for a specific corporation. Primarily a rule based data-base lookup package.

[Phillips et. al., 1985] - Discusses a profile based planner.

[Preiss and Kaplansky, 1984] - A system that goes directly from design to NC milling.

PROPLAN [Phillips, 1984] [Phillips and Mouleeswaran, 1985] - Primitive lines and arcs, describes a symmetrical rotational part. Production rules are stored in a database and applied using a graph search. The graph nodes are the parts geometry, the arcs are the transformation operations.

PWA_Planner [Chang and Terwilliger, 1987] - Does assembly planning for circuit boards.

ROUND [vanHouten and Kals, 1984] - A generative system for turned parts.

SAPT [Milacic, 1985] [Milacic and Vrosevic, 1984] [Milacic et. al., 1988] - Three modules are used, the first is technological pattern recognition, which defines detailed structure of process planning. Secondly, the manufacturing

process logic module defines logical relationships between the part, machine, fixture, and operation sequence. The rules for these tasks are stored in a knowledge-base, along with rules for the strategy of planning.

[Schneider et. al., 1989] - Their system attempts to match rules for rotational parts by examining the 2D profiles.

SIPP [Nau and Chang, 1985] - Hierarchical frames are used to represent parts, where a frame will represent a feature, and other frames define the properties of the feature. A search strategy using least cost first, is used in conjunction with a knowledge base relating features, to machining processes. The nodes of the graph represent parts, and the arcs are the operations that transforms them. Process restrictions are considered in generation of the graph.

TOM [Matsushima et. al, 1982] [Iwata and Sugimura, 1984] - Technostructure Of Machining uses a final design, a collection of holes, and backward chains a set of production rules to develop an optimal process plan. When conflicting rules are found heuristics are used, including frequency of application.

TOPS [Pinte, 1987] - A system that uses frames, constraints and rules with a truth maintenance system for planning.

[Willis et. al., 1989] - A system that uses a solid modeler based on an expert system. Non-linear planning is supported.

XCUT [Hummel, 1989] [Hummel and Brooks, 1988] - Uses a hierarchical layering of rules Even more important is the use of Meta-knowledge to select between modules of rules.

XMAPP [Inui, 1986] - Does forward planning using a product model like GARI.

XPLAN [Alting, Zhang and Lenau, 1988], XPLAN-R [Zhang and Alting, 1988 a, b] - A planning system for rotational parts based on DCLASS. The successor is XPLAN-R.

XPS-1 [Groppetti and Semeraro, 1986], XPS-2 [Nolan, 1989], XPS-E [Chryssolouris et. al., 1986] - A interesting line of generative CAPP systems have been developed from the CMPP project, eventually evolving into the XPS-E system. CMPP (Computer-Managed Process Planning) was originally developed at United Technologies [Mark Dunn in Nolan, 1989]. It is a process planning system which is Generative, and Automatic, and capable of handling rotational parts. After the data entry module is done the Process Planning module will begin. Its four functions are; generate a summary of operations, select tolerancing datums, determine dimensions and tolerances, output process documentation. The Computer Aided Manufacturing-Inter-

national (CAM-I) group has been developing a process planning system. This has been done through a set of projects dubbed XPS-N [Sack authored a section in Nolan, 1989]. The first system in this line was XPS-1, which was completed in 1984. The system provides an environment for process planning, using feature based product data, and manufacturing resource data. This information is stored in a relational database. Rules are used to provide the logic for process planning. The second prototype system, XPS-2 was completed in 1987. Other CAM-I design projects were used as an interactive front end for collecting feature information including, geometry, dimensioning, and tolerancing. An attempt to use Artificial Intelligence was proposed with XPS-E. This was done using a report commissioned from two French organizations, Institut National Polytechnique de Grenoble (INPG) and Industrie et Technologie de la Machine Intelligente (ITMI). Their method conformed to the ideal of the XPS-N Philosophy, but augmented it with Artificial Intelligence techniques.

2.9 CONCLUSION

This chapter has discussed the fundamentals of CAPP research, including product modelling, process plan modelling, Variant and Generative systems. It then went on to discuss techniques used for interpreting the product model. A detailed look at a CAPP system for machining is then presented, including operation planning. Finally, a list of many of the systems reviewed is given, with a brief description of each.

At this point the reader will have some appreciation for the implementation of CAPP systems, and the environments they operate in. Most of the techniques discussed in this section can be used when planning for specific technologies, but there is a notable absence of techniques for planning across multiple manufacturing domains. In terms specific to the thesis, process planners described in this section are all best suited to the Operation Planning aspects of CAPP, as will be covered in Chapter 6.

3.PRODUCT MODELLING

3.1 INTRODUCTION

The role of this chapter is to provide a description of how products will be modelled for use in subsequent sections of the thesis. The representation is similar to other approaches, but has some subtle differences to support the subsequent planning process. This is not to say that the method is a replacement of previous methods, in fact it may be used to imitate many of the other methods available.

The description begins with a review of solid modelling as it applies to product modelling. This is followed by some discussion of how to incorporate not only solid modelling, but other aspects of product design, into an algebraic form. Software implementations were used for verification of the method. These include capabilities for storage of arbitrary sets, and storage and manipulation of Boolean equations.

If the reader requires more background in solid modeling, they are directed to Requicha and Rossignac [1992] and Stewart [1990].

3.2 THE BASIC TYPES OF SOLID MODELS

Requicha [1992] asserts that there are three fundamental methods for representing solid objects: Decomposition, Boundaries, and Construction. Decomposition uses representations which attempt to divide space in some discrete manner. These approaches include thin 2 dimensional slices, voxels, ray casting, oct-trees, and binary space partition trees. Boundary methods use the faces, edges, and vertices of the object. Various data structures differentiate much of the work in this field. Requicha [1992] also states that additional information is required with the boundaries to establish solid properties. Finally, Constructive approaches build up the data representation from other geometries. The most popular constructive

techniques are the CSG (Constructive Solids Geometry) and Sweeping methods.

There are clear differentiations between each of the three main solid modelling approaches. The Constructive methods are well defined volumes, whereas the other methods are subject to spatial resolution, volume location and data abstraction problems. For example the CSG method is described by Requicha as "rooted, directed, acyclic, binary graphs, where the internal nodes correspond to Boolean operations or rigid-body transformations and where leaves are primitive bounded solids or half spaces. CSG representations are concise, always valid in the r-set modeling domain, and easily parameterized and edited."

Higher level methods for representing solids have also been examined in recent research. One particular method often cited is feature modelling. In this approach the part is described with features. The features are treated as solids, and operated upon as solids [ElMaraghy, 1991]. In addition to problems mentioned in the last chapter, a problem it causes for Solid Modeling is that when features overlap there is no method for determining priority [Requicha, 1992]. For example, if a fillet overlaps a hole, does the fillet cover the hole, or is it removed?

The CSG representation has been chosen for this thesis because it is both abstract and unambiguous. This makes it well suited to serve as the basis for this reasoning system. More specifically the representation will be done with the CSG expressions in Boolean form.

3.3 BOOLEAN BASED SOLID GEOMETRY

An analytic expression of geometry can provide an exact representation of a part. As the geometry of the part is analyzed, it may be decomposed into boundaries, then surfaces, and finally lines. At each interpretation some information is

lost. Boolean operations on sets are the most abstract representation of solids, with the least detail. By its nature the CSG representation is informationally complete, concise, and the method of representation ensures syntactic correctness.

Constructive Solid Geometry (CSG) is an explicit mathematical representation. The geometry may be expressed as a Boolean equation with sets as the variables. This means there are a number of basic properties that are of relevance to any work using these equations. This section of the chapter will address the basic properties and relationships of interest.

3.3.1 Basic Concepts

CSG models are based upon Boolean operations performed on primitive solids. Many names have been given to this method, for example Constructive Solids Geometry (CSG) and the Set Theoretic method. There are also many basic representations that may be used. One is the CSG tree, which is a combination of operations and primitives as seen in Figure 28. The nodes of the tree are operations, and the leaves of the tree are primitives. There are only two children for each node. The top of the tree is referred to as the root. Another method for representation is in expression form. The expression form uses operators applied to the variables, which represent the primitives (see Lee and Jea [1988] for more examples). Other deviations on the expressions have been developed by various researchers, such as Woo [1977]. He uses a similar expression structure, but each term has the same operator applied to all of the variables within the set of brackets.

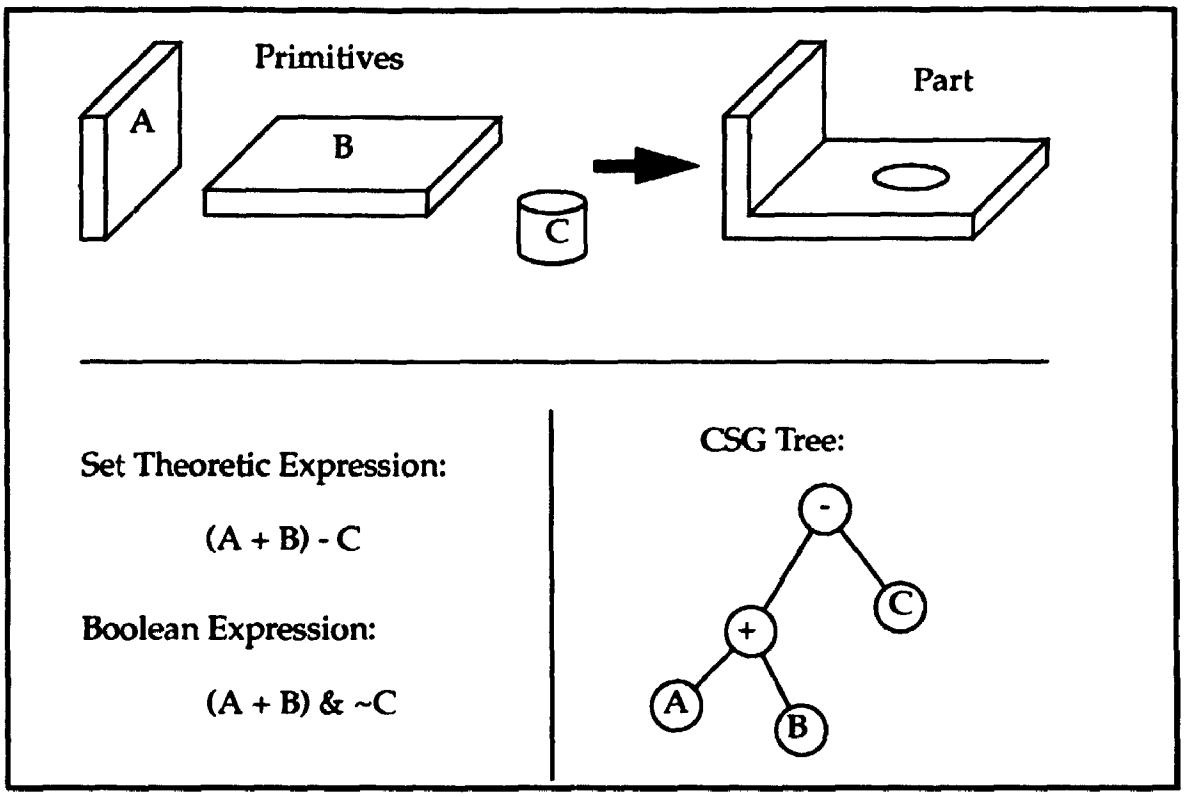


Figure 28 CSG Representation in Tree and Expression Form

Some of the distinctive features of CSG models are globalness, non-uniqueness, and redundancy. The term 'global' refers to the possible distribution of solids over a tree, or an expression, whatever the case. There have been attempts to overcome this by using normal [Goldfeather et. al., 1986] and canonical [Kakaza and Okino, 1984] forms. Although these methods may still leave distributed terms, they do reduce the distribution. The non-uniqueness property refers to the infinite number of ways in which specific solid geometries may be composed. This is still an ongoing research problem. Some authors [Woodwark, 1988] view this as a major problem for CSG when finding a single set of manufacturing features for a part. But, as the reader will see, I have exploited this to find many alternative manufacturing features for production. The redundancy problem builds upon the non-uniqueness. In this case there are two primitives that cancel each other. Tilove

[1984] gives as an example a CSG tree where the designer has used a cylinder to fill a previously cut hole. To solve this problem Tilove [1984] developed algorithms for Null Object Detection (NOD), to determine when there are redundant primitives.

3.3.2 Set Topology (Unary Operators)

A set topology is considered to be a set of points in space. By definition, these points do not include a boundary. The standard topological operators are complement, interior, closure and boundary. These unary operators differ from the binary operators in that they only operate on one set.

3.3.3 Binary Operators

When dealing with two interacting topologies, binary operators are required to describe the interaction. The commonly used operators are union (OR), intersection (AND) and difference (AND with NOT). If these operators are applied with ignorance to the point sets there are possible overlap and singular contact point problems. When these binary operators are used in combination with the unary operators the result is a theoretically correct and complete set of operators. These are described in the next section.

3.3.4 Set Theoretic Operators

The basic operators were described before, and there are other enhanced operators such as Assemble [Arbab, 1990] [Hartquist and Marisa, 1985]. These operators work on basic sets of topological primitives. The primitives can be represented in a few different forms. The traditional approach uses r-sets [Requicha, 1980] and a newer approach uses s-sets [Arbab, 1990].

An r-set [Tilove and Requicha, 1980] is used to represent a homogenous solid in

three dimensional Euclidean space. The face of the solids is assumed to be rational, therefore there are no unassociated vertices, edges, faces. This also implies no infinitely thin holes or cracks. The volumes must be enclosed by a set of boundaries. The set is closed, which means that it includes its own boundary. The set is also regular which means it is equal to the closure of its interior. And, the set is semianalytic, which means it can be expressed as a finite boolean combination of analytic sets. When operations are done they may not result in a set which is closed and regular. Therefore, a special set of operators have been developed which force a closed regular object. These are the Regularized operators. When applied these operators will eliminate dangling planes, vertices, edges, infinitely small holes and cracks. The definitions of these operators is given in Figure 29. The r-set approach is not always desirable. When dealing with assemblies, nonmanifold contact r-sets would call for physically connected parts. But, it may be desired to have parts that are physically separate, such as press fits. [Aside: manifold solids have edges bounding exactly two faces, and vertices acting as a point of a cone. Nonmanifold parts touch with one or both parts along points, or edges only, therefore not sharing any volume at the point of contact. In more practical terms, nonmanifold objects can erroneously suggest that part of an object exists without any volume].

$$X +^{*} Y = \text{closure}(\text{interior}(X + Y))$$

$$X \&^{*} Y = \text{closure}(\text{interior}(X \& Y))$$

$$\sim^{*} X = \text{closure}(\text{interior}(\sim X))$$

Figure 29 Axioms of r-sets
(These ensure that all sets are closed, and regular)

One simple method of allowing touching parts is to use an assembly operator, this allows aggregates of r-sets (each part in the assembly is stored separately). PADL2 [Hartquist and Marisa, 1985] uses these sets to represent assemblies. Extended CSG trees are used to represent Assembly operators, along with Regularized Boolean operators. The axioms in Figure 30 are for assembly operators, and will permit us to manipulate the individual assemblies, without forcing them to join. Arbab [1990] discusses the use of the Assembly operator, and likens it to a placeholder in the CSG trees. This method cannot derive a single r-set, thus a re-evaluation is required whenever a new operator is applied. Also, because the sets are stored separately, mathematical errors may accumulated differently and result in slight variations in object positions after transformations.

$$(X : Y) +^* Z = X +^* Y +^* Z$$

$$(X : Y) \&^* Z = (X \&^* Z) : (Y \&^* Z)$$

$$(X : Y) -^* Z = (X -^* Z) : (Y -^* Z)$$

$$X -^* (Y : Z) = (X -^* Y) -^* Z$$

$$(X : Y) : Z = X : (Y : Z)$$

$$X : Y = Y : X$$

$$X : NULL = X$$

Figure 30 Axioms for Assembly Operators

When dealing with assemblies, an alternative to r-sets and Assembly operators are open regular sets. These sets have all of the properties of the r-sets, except that points on the boundary are not considered part of the volume. This allows surfaces to make contact, without having to become a single solid [Arbab, 1991]. This method does not allow internal nonmanifold boundaries. To allow internal

boundaries, hypersets may be used [Requicha, 1992]. A volume set is decomposed into volume subsets (hypersets) which are the result of previous set operations that have resulted in the nonhomogenous solid.

Arbab [1990] proposes s-sets as an alternative to r-sets, Assembly operators, and open regular sets. The essential feature of this method is that separate internal boundaries are allowed within the solids. An assembly operator is an inherent part of s-sets. The basic s-set operators are shown in Figure 31. Note that the assembly operator only acts when the components do not intersect in the first intersection axiom. A preferred approach to the assembly axiom is also provided that will split the assembled volume into three components. The internal boundaries are not preserved by all operations, only the assembly operation may create new internal boundaries. Arbab also poses a set of axioms specifically for a B-Rep of s-sets.

$R +^{+} S = \text{interior}(\text{closure}(R + S)$

$R : S = R + S$ if $(R \,\&\, S) = \text{NULL}$ --- Basic Axiom

$R : S = (R -^{+} S) + (S -^{+} R) + (R \,\&\, S)$ ----- Preferred Axiom

$R \,\&^{+} S = \text{interior}(\text{closure}(R \,\&\, S))$

$\sim^{+}R = \text{interior}(\text{closure}(\sim R))$

$R -^{+} S = R \,\&\, (\sim^{+} S)$

Figure 31 Open Regular Axioms for s-sets

The methods previously described are concerned with exact representations of solids. In actuality, solids never conform exactly to a geometrical form. Variational classes have been introduced as a method for dealing with sets of acceptable toler-

ances for solids [Requicha, 1992].

In the authors opinion the most flexible approach which has been developed is Selective Geometry Complexes (SGC). This method uses aggregates of mutually disjoint cells that are connected in a continuous geometry. The sets can be open or closed, and the geometry expressed in equation form. For example, a union of two shapes could result in one solid, containing three separate volume regions. Where the two solids touched, an unknown volume would now exist hence the three regions are required. Rossignac and Requicha [1991] developed a new representation scheme called Constructive Non-Regularized Geometry (CNRG) which takes advantage of previous set operations, including SGC. In total, their representation scheme allows all of the previous regularized operations, as well as nonhomogenous solids (materials with varying properties), and assemblies. This method is very exciting, and would be excellent for representing the solids commonly found in design. Unfortunately, at the time of writing, this work was still under development, and not in a suitable state for use.

The previous discussion indicates that there are many methods for modelling solids in a set theoretic approach. It also indicates that there are no 'perfect' methods yet. As a result it was decided that the design representation should be as abstract as possible, to ensure conformity to whatever method is developed. In practical terms this means that the Assembly operator will be maintained as an independent operator keeping all of the geometries separate. This is most similar to the r-set approach. Eventually, new operators for assembly can be added when the tools to support them are available.

3.4 MANIPULATING SOLID MODELS OF DESIGNS

It was decided to use the non-uniqueness and global nature of the CSG representation to drive the logic of the BCAPP planning system. But, in addition to this the CSG expression will be represented with Boolean expressions to allow use of Boolean algebra. This section will describe my particular representation. The following sections will discuss implementation of this representation in a Boolean Algebra program, and representing product designs with Boolean equations of sets.

3.5 A COMPUTER BASED BOOLEAN DESIGN REPRESENTATION

First, while the scope of operators is clearly defined in Boolean algebra, they can be somewhat confusing to comprehend. The problem of changing operator scope will require that any computer program understand the rules for their application. This would require some form of complex algorithm to determine operator scope before the simplest operation could be performed. To simplify this problem a Nested Boolean form is adopted which is very scope specific, similar to that used in Woo[1977]. This form puts each operator in a set of brackets. As in normal algebra, nested terms in brackets must be performed first. This method does not in any way restrict the use of any of the Boolean operators, but removes any question of ambiguity in interpretation. A detailed discussion will follow, but an example of this is,

$$A * B + A * \sim C \qquad (3.1)$$

becomes,

$$(+(*\,A\,B)(*\,A(\sim C))) \qquad (3.2)$$

While the first equation above would require some interpretation as to order,

the second form specifies that the NOT operation, and the AND operations must be performed before the OR operation. The only addition is the generous use of brackets, and the collection of an operator over a list of terms. In a more logical form the equality of both representations may be shown by first considering the premise that precedence of operations in an expression is directly based upon either conventions of priority for operations, or precedence that is clearly indicated by bracketed terms. Therefore, when considering the conventional precedence of operators, a high priority operator would be forced to occur first, as if it were in brackets. When brackets are used anyway, they duplicate the brackets in the Nested Boolean representation.

Secondly, because of the nature of design, we would attempt to economize on effort by reusing terms, and sets. As is seen in the expression above (3.2), A is reused in the same equation. Say for example that A is a common screw. In the equation above it would be used twice. The problem arises in that the set that defines the geometry of the screw, also defines attributes such as position. In the representation above, both of the screws would overlap in space. This assertion is obviously nonsense, so to overcome it, enhancements were added to the equations. These come in the form of attribute sets that may be appended to any symbol, or term in an equation. An example is seen below,

(+ (* A;move_to_B B) (* A;Fixture (~ C);Move_to_A);Move_to_spot)

In this case some terms and symbols have been augmented with additional property sets that effectively become part of the original set. Therefore in this case we have defined only one A, but after adding 'move_to_B' to one, and 'Move_to_A' to the other, they now become separate, and individual sets. This starts to explore the subtle differences of equations of sets to be discussed later.

Thirdly, the exclusive use of Boolean operators is not realistic here because of the use of sets, and the irrationality of product representation. Therefore the Boolean operators have been supplemented with some other operators and constraints. As discussed before, assemblies are difficult to model within one solid model. As a result we use the assembly operator used in many solid modelers. This allows parts which may overlap in a geometrical model (e.g., a press fit) to coexist separately. This also allows the designer to specify when an assembly is actually desired.

3.5.1 A File Format for Product Description

While the mathematical descriptions in the previous section outlined the theoretical model of the product, it was necessary to describe the part in some file format for verification of the method, and for other stages of testing.

The file format chosen was a text format, which would allow easy editing and verification. A sample of the file format is shown in Figure 32. Within the file the sets are defined by giving the set name, and then a list of properties, delimited by the '{' and '}' brackets. All of the properties are arbitrary, except for the 'EQUATION' property. The purpose of the 'EQUATION' property is to indicate how other sets combine into this one.

This format has been left particularly loose, so as to not restrict future development. The problems with fixed fields are emphasized by the number of special cases required for small differences, as can be seen in vonRimscha [1989]. One potential problem that could result from the free form representation is the inadvertent creation of a recursive reference for a child to a parent, which is physically impossible. Within my thesis I will use a few properties that can be considered standard, such as 'translate_y'. Most of the attributes will be easy to identify from

their name only. The selection of properties should be done on an application specific basis. A readily available example of this is, that by choosing 'translate_y' we assume all coordinates are Cartesian, whereas we may have a robotics based design which considers cylindrical coordinates the norm, such as 'translate_r'. A number of properties have been identified in other work that must be stored in addition to the basic solid model for completeness. Some of these other properties are tolerances and material [Braid, 1986].

The file below is one of the test files used to verify the software written. It depicts a nut and bolt assembly (as indicated by the 'main' classifier). Notably, this set only contains an equation. In a practical use this would probably contain some information from the designer. The expression contains a Boolean equation that refers to 'nut', and to 'bolt'. And, the set 'bolt' is modified by adding 'move_to_hole'. The 'move_to_hole' set is now appended onto the 'bolt' set for the assembly operation.

```
main nut_and_bolt {
     EQUATION: ( : nut bolt )
}
nut {
     EQUATION: ( & A ( ~ B ) )
}
bolt {
     EQUATION: ( + C ( & D;test1 ( ~ E ) )
}
test1 {
     color =SILVER
}
A {
     form = BLOCK
     width = 10.0
     depth = 10.0
     height = 4.0
     name = Nut_Blank
}
```

Figure 32 A Sample File Defining a Part

```
B {
      form = CYLINDER
      radius = 2.0
      height = 5.0
      name = tapped_radius_hole
      finish = threaded
      diameter_tolerance = 0.1
}
C {
      form = CYLINDER
      radius = 2.0
      height = 5.0
      name = bolt_shaft
      finish = threaded
      translate_x = 0.0
      translate_y = 0.0
      translate_z = 0.0
      rotate_x = 0.0
      rotate_y = 0.0
      rotate_z = 0.0
}
D {
      form = CYLINDER
      radius = 3.0
      height = 1.0
      name = bolt_head
      finish = polish
      diameter_tolerance = .1
      translate_x = 0.0
      translate_y = 0.0
      translate_z = 3.0
      rotate_x = 0.0
      rotate_y = 0.0
      rotate_z = 0.0
}
E {
      form = BLOCK
      width = 9.0
      depth = .5
      height = .5
      translate_x = 0.0
      translate_y = 0.0
      translate_z = 3.25
      rotate_x = 0.0
      rotate_y = 0.0
      rotate_z = 0.0
}
move_to_hole {
      translate_x = 3.0
}
```

Figure 32 A Sample File Defining a Part (cont'd)

This particular structure can be of use because the process planner in not just

receiving geometry, but also some insight to the designers methodology. For example, the assembly of the nut and bolt in a traditional solid model would just be two separate non-overlapping geometries. In this model we can clearly (and easily) identify the two separate parts that are assembled. The file format not only represents the final geometry, but the designer's perceptions of the design.

When the file format is loaded it can be easily interpreted into a Boolean equation. The example below uses substitution to expand the equation in the main set to a final geometric description.

```
( : nut bolt;move_to_hole )                                    (3.3)
( : ( & A ( ~ B ) ) bolt;move_to_hole )                        (3.4)
( : ( & A ( ~ B ) ) ( + C ( & D;test1 ( ~ E ) ) );move_to_hole )  (3.5)
```

The final equation is of the type that can be used to drive a solid modeler. The figure below was automatically generated from the 'nut_and_bolt' file above, and coincidently also helped verify the correctness of the low level boolean algebra routines.

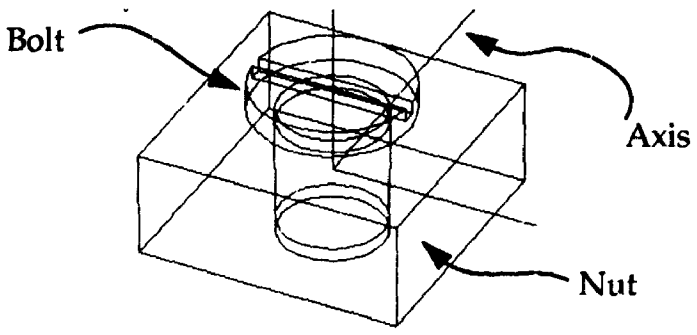


Figure 33 The 'nut_and_bolt' Example Interpreted into Geometry (modelled rendered using ACIS [Spatial Technology, 1991])

The representation here was developed to facilitate testing of the system. In review it is safe to say that this system could eventually benefit by the incorporation of a more formal language structure, such as that of PDL [H. A. ElMaraghy, 1991], or ALPS [Mantyla, 1993].

3.6 A DATA STRUCTURE FOR STORING BOOLEAN ALGEBRA

The Boolean algebra methods described in this thesis are supported by a set of Boolean algebra data structures. These structures were written in C++, thus there is some use of inheritance, and functions are associated with the Boolean data ucture [Strousap, 1986].

3.6.1 Data Structures for Boolean Equation Storage

As shown before, each term is represented in a LISP like fashion with an operator being first in the list, followed by a list of operands. An example of this equation form is shown in Figure 34, along with a data structure diagram that is used to store the equation (only part of the equation is shown for the purpose of clarity). The Operation List may be thought of as storing the start of each nested term, it coincides with the open bracket '(' and the operator. Therefore, each operator is assigned a row. The row points to a list of terms. The other field in the row is for a symbol pointer. The Symbol Pointer allows a text string (and other symbol information) to be associated with each operation. The append list field allows lists of appended properties, such as 'test', to be appended to symbols and terms.

The Term Lists array is simply for storing lists of arguments for the operations. The reader will note that the pointer refers to both the Operation List, and the Symbol List. This allows both sets, like 'A' and sub-terms '(+ D G)' to exist in common. The 'type' field specifies which type of pointer is in the 'pointer' field. The 'point to next' field provides a linked list of the arguments, and terminates with a 'last' marker.

Finally, the Symbol list is used to store textual information for recall by the user. These also contain two fields for use by the calling routines. A 'pointer' field is provided so that the user may store a reference to an outside object (like a geo-

metric primitive), whereas a 'type' field provides information about the pointer type. There are additional types of data associated with each list for the purposes of maintenance, as is described on the next paragraph.

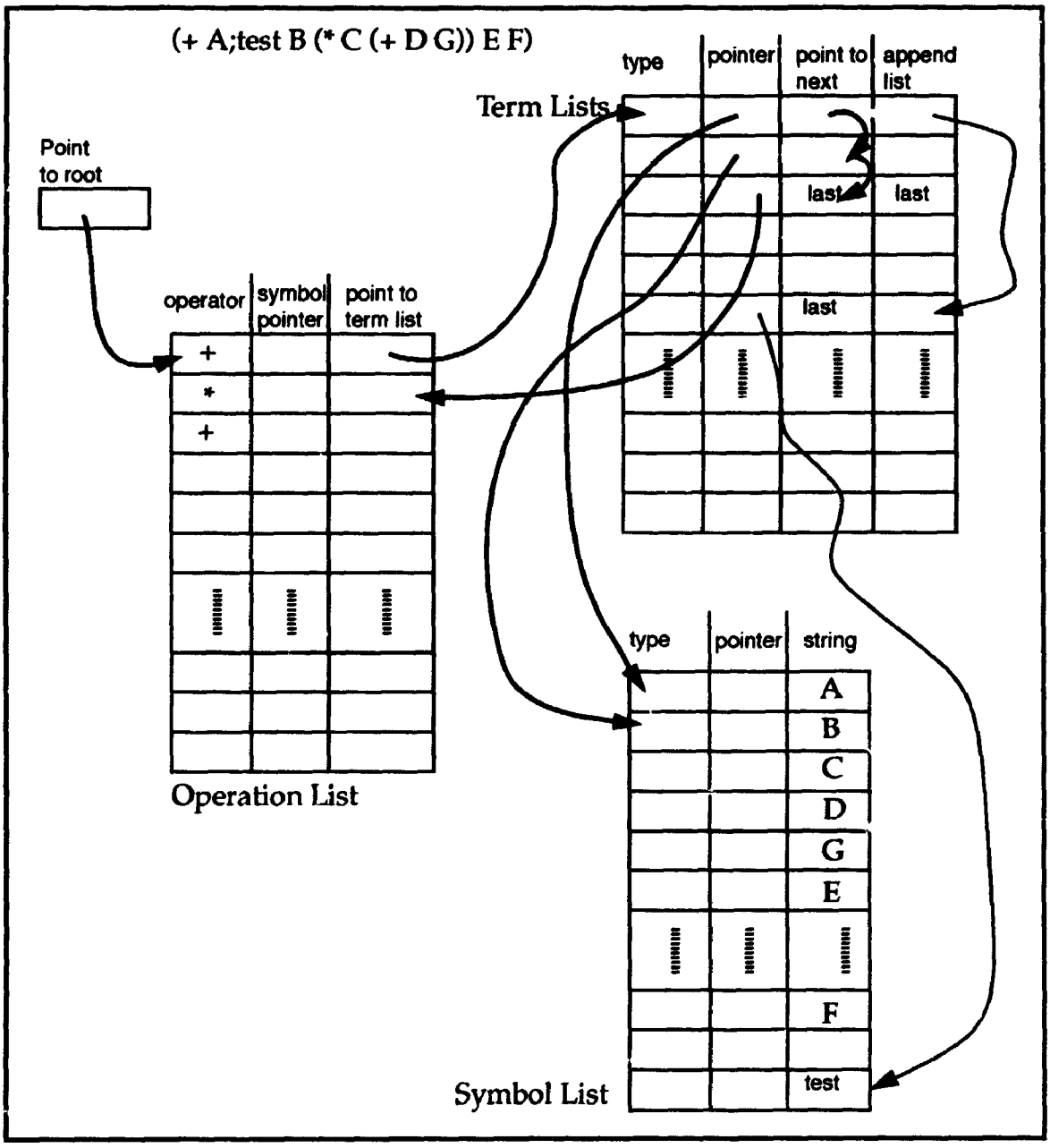


Figure 34 Nested Boolean Algebra Data Structure
(Only a partial example is given for clarity)

The reader should note that the data structures in Figure 34 do not show the method for dynamic memory allocation. The lists shown in the data structure have many rows which will be added and deleted in a semi-random fashion. As a result, a reuse mechanism structure has developed for each of the lists. One example of this structure is shown in Figure 35, and may be viewed as a generic approach for all of the lists above. In Figure 34 both 'used' and 'point to next' are added data fields. This structure makes allocation of a new row, simply a matter of deleting from the head of empty linked list, and pushing onto the head of the used list. Deletion of an existing term is a case of modifying next pointers to point over the defunct term, then pushing the now empty term onto the empty list. The used and unused flags are updated to make validity checking easier, without having to retrace lists. The end of list pointer makes range checking easy, when the array has been dynamically allocated (as was done using C++).

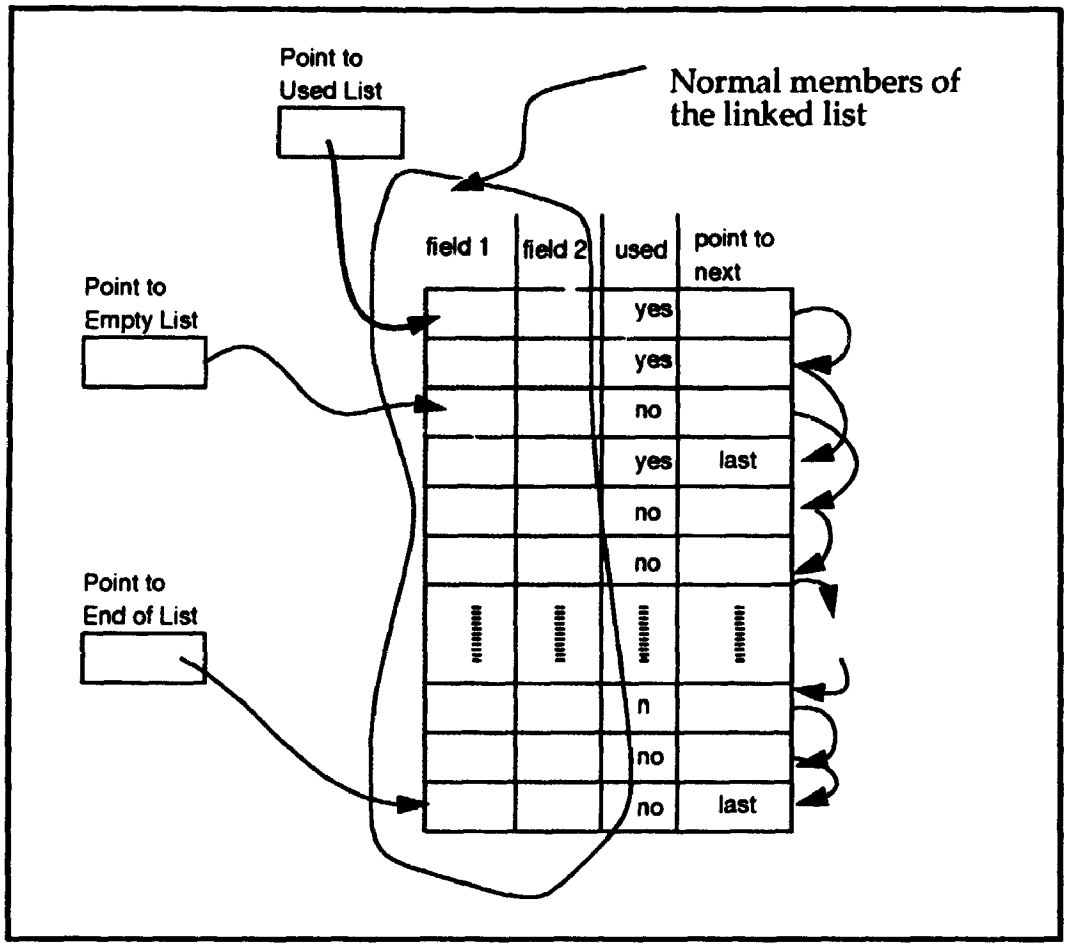


Figure 35 Dynamic Memory Structures Added to Lists

Public functions have been created to manipulate each object. These are structured so that the functionalities are nested. Some of the functions are listed below.

Symbol List:

- Initialize a new symbol.
- Find a defined symbol by name.
- Set/Get a pointer value for the symbol.
- Set/Get a pointer type for the symbol.
- Set/Get a pointer name for the symbol.
- Remove an unused symbol.

Terms Lists:

- Create a new term list.
- Append/Insert a term to a list.
- Remove a term from a list.

- Remove a term list.
- Copy a term list.
- Get an individual term reference from a list (e.g. 3rd term in a list).
- Get/Set a term type (symbol, or operation).
- Get/Set a term pointer (e.g. pointer to symbol list).
- List/Count all terms in a list.

Operation List:

- Create a new operation.
- Remove an unused operation.
- Find a term/symbol in the equation.
- Address term list operations with respect to equation.
- Copy a branch in the equation.
- Dump/Import a string which represents the equation.
- List/Count terms/symbols in an equation.

This list has many functions, but as a group they enable storage, and manipulation of boolean expressions. When referring to individual terms/symbols in the expression their location may be addressed by a simple scheme. For example, the address ':2:2:2' in the equation '(+ A B (* C D (+ E F G H)' would refer to the symbol 'G'. For the same equation, address ':2' (or ':2:') would refer to the term '(* C D (+ E F G H))', and address ':' refers to the entire equation. This lays the ground work for a systematic addressing of the equation for boolean algebra.

3.7 MANIPULATION OF THE BOOLEAN EXPRESSIONS

Once an equation has been stored in the Boolean Equation data structures, it need to be manipulated. This can be done by examining the rules of Boolean algebra, and then casting these in the form of algorithmic operations.

3.7.1 Previous Work in Manipulation of Boolean Equations

Lee and Jea [1988] discuss a method for manipulating the structure of a CSG tree by moving branches up and down the tree structure. They limit the operators to union and subtraction, and assume the operators are regularized. They argue that for feature recognition using CSG, it is necessary to move nodes up the CSG

tree to be compared. Although they do not pose a feature recognition algorithm, they do suggest that various nodes may be moved up the CSG to partially overcome the global, and uniqueness problems of the CSG representation. Their algorithm is based on transformation tables for CSG branch forms. These tables are pictured in Figure 36. The reader should note that 16 basic input states have been given by Lee and Jea, and there are a number of possible output states associated by form. In the tables given, each of the 16 states are listed in each of the four tables. There are four tables to indicate the four possible states for parents of the subtree containing A, B, and C. The work is of particular interest to this thesis because of the use of equation forms to manipulate tree structures. Their work also supports the idea of manipulating the CSG structure to achieve various interpretations of a part.

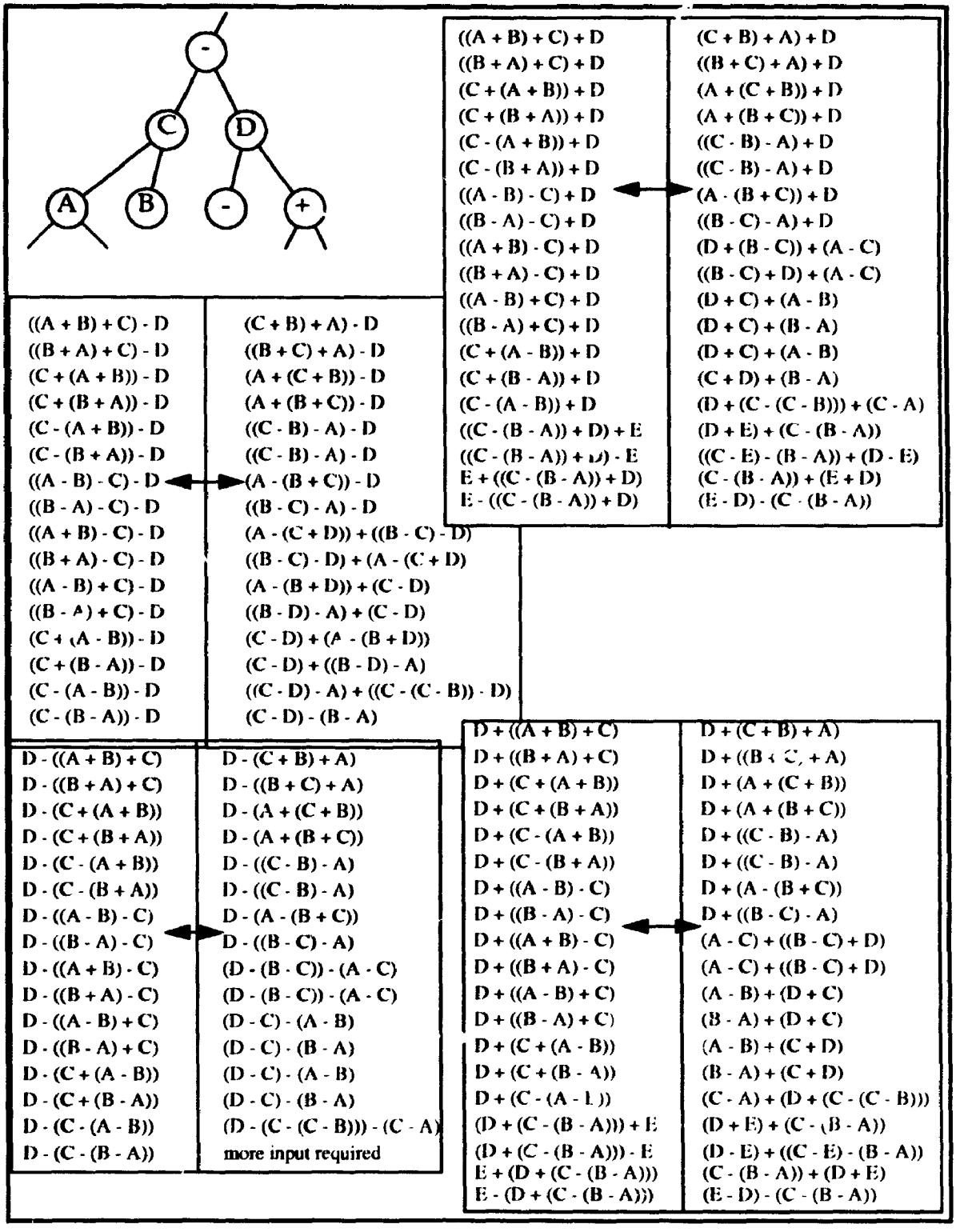


Figure 36 Transformation Tables for Moving Node A up CSG Trees
(Read the tables as a form in the left hand column is equivalent to that in the right hand column)

When attempting to eliminate redundant primitives in a CSG tree, a Null Object Detection (NOD) algorithm may be used. (Please note that the assumption with this technique is that unused and redundant primitives are undesirable). The algorithm attempts to locate primitives that cancel each other out. Tilove [1984] describes his use of Boolean algebra properties to design algorithms to detect null objects. This is done through a set of five algorithms he presents, which look for positive, and negative redundancy by direct examination of the tree. Tilove also describes previous algorithms that were not based on CSG methods and required greater access to computational resources. As an aside, he also states that the NOD algorithm can be used as the basis for Same Object Detection (SOD). This is useful when trying to ascertain if two objects are indeed similar.

3.7.2 Boolean Algebra Manipulation of Data Structures

As discussed before, we may store and manipulate the Boolean expressions. The data structures discussed in the previous section did not discuss the addition of Boolean algebra whereas this section will. A set of routines were written to apply the rules of Boolean algebra to manipulate the equations. The fundamental operations are given in Figure 37. All but Identity and Duality have been implemented in the software.

Idempotent

$A + A = A$ $\qquad$ $A \& A = A$

Associative

$(A + B) + C = A + (B + C)$ $\qquad$ $(A \& B) \& C = A \& (B \& C)$

Commutative

$A + B = B + A$ $\qquad$ $A \& B = B \& A$

Distributive

$A + (B \& C) = (A + B) \& (A + C)$ $\qquad$ $A \& A = A$

Identity

$A + NULL = A$ $\qquad$ $A + UNIVERSAL = UNIVERSAL$

$A \& NULL = NULL$ $\qquad$ $A \& UNIVERSAL = A$

Complement

$A + \sim A = UNIVERSAL$ $\qquad$ $\sim(\sim A) = A$

$A \& \sim A = NULL$ $\qquad$ $\sim UNIVERSAL = NULL$

DeMorgan's

$\sim(A + B) = \sim A \& \sim B$ $\qquad$ $\sim(A \& B) = \sim A + \sim B$

Duality

interchange union and intersection operators, as well as all Universal, and Null sets. The resulting equation is equivalent to the original.

Figure 37 Fundamental Boolean Operations

These operations are set up so that they may be applied in a parametric manner (see Mendelson [1989], for more information on Boolean algebra). In other words, there may be a term '(+ A B C)'. The law of Commutivity says that the expression

may have up to six (3! = 3*2*1) forms; '(+ A B C)', '(+ A C B)', '(+ C B A)', '(+ B A C)', '(+ B C A)', and '(+ C A B)'. Therefore the law of Commutivity can be applied to some term, a finite number of ways. By expressing the manipulation in this manner, the method of manipulating equations becomes similar to methods used in tree searches (i.e., the operator is a simple finite description). This becomes the first step to deriving an exhaustive method for equation manipulation. The list below shows all of the operations used, and gives examples of how they are applied.

- Idempotent
  - Simplify - A + A = A, A & A = A
    - Example equation (& A B A C A A;e)
      - In this expression there are n=6 terms, and therefore there are (n-1)(n-2)...(1) possible combinations for application. Of this there are only 3 that can be successfully performed in the example equation. This is between the first three 'A's. The last 'A' is considered different because it has a ';e' appended.
      - Possible results,
        (& A B C A A;e)
        (& A B A C A;e)
        (& B A C A A;e)
  - Complicate - A = A + A, A = A & A
    - Example equation (+ A B C)
      - In this expression there are n=3 possible terms to apply the members to.
      - Possible results,
        (+ A A B C)
        (+ A B B C)
        (+ A B C C)
- Associative
  - Simplify - A + (B + C) = A + B + C, A & (B & C) = A & B & C
    - Example equation (& A B (+ C D) (& E;x F);y)
      - There are n=4 possible candidates for this operation of which two are bracketed, and one of those two has the same sign. The '(& E F)' term can be 'popped' because it has the same operation as the parent term.
      - Possible result,
        (& A B (+ C D) E;x;y F;y)
  - Complicate - A + B + C = A + (B + C), A & B & C = A & (B & C)
    - Example equation (& A;x B (~ C))

- For this operation it was chosen to allow each possible commutation (see commutative operator). This gives a total n!(n!) possible permutations that are acceptable for this operation.
- Possible results,
  (& (& A;x B) (~ C))
  (& (& B A;x) (~ C))
  (& (~ C) (& A;x B))
  (& (~ C) (& B A;x))
  (& (& A;x (~ C)) B)
  (& (& (~ C) A;x) B)
  (& B (& A;x (~ C)))
  (& B (& (~ C) A;x))
  (& (& B (~ C)) A;x)
  (& (& (~ C) B) A;x)
  (& A;x (& B (~ C)))
  (& A;x (& (~ C) B))

- Commutative
  - Does not simplify or complicate - A + B = B + A, A & B = B & A
    - Example equation (+ A (& B C) D)
      - There are n! possible acceptable permutations for this operation. In simple words this operation only shuffles the order of the terms in the equation.
      - Possible results,
        (+ A (& B C) D)
        (+ A D (& B C))
        (+ D A (& B C))
        (+ D (& B C) A)
        (+ (& B C) A D)
        (+ (& B C) D A)
- Distributive
  - Simplify - (& A (+ B C)) = (+ (& A B) (& A C)),
    (+ A (& B C)) = (+ (& A B) (+ A C))
    - Example equation (+ A (+ B;e C);f (+ D E) (& F G))
      - There is only one possible permutation with this operation.
      - Possible result,
        (& (+ A F (+ B;e C);f (+ D E)) (+ A G (+ B;e C);f (+ D E)))
  - Complicate - (+ (& A B) (& A C)) = (& A (+ B C)),
    (& (+ A B) (+ A C)) = (+ A (& B C))
    - Example equation (& A B (+ C D) (+ D E) (& F G))
      - Again, this operation will only have one possible permutation for the purpose of simplicity.
      - Possible result,
        (& A B (+ D (& C E)) (& F G))
- Complement
  - Simplify - (~ (~ A)) = A
    - Example equation (~ (~ (& A B);x);y); z

- Again there is only one possible permutation possible here. If a double negative exists, then it is replaced with the doubly negated term
- Possible results,
  (& A B);x;y;z

- Complicate - A = (~ (~ A))
  - Example equation (+ A B C;x D)
    - In this example there are n=4 possible permutations allowable.
    - Possible results,
      (+ (~ (~ A)) B C;x D)
      (+ A (~ (~ B)) C;x D)
      (+ A B (~ (~ C;x)) D)
      (+ A B C;x (~ (~ D)))
- DeMorgan's
  - Simplify - (~ (& A B)) = (+ (~ A) (~ B)),
    (~ (+ A B)) = (& (~ A) (~ B))
    - Example equation (+ A B (~ (~ (& C;x D);y);z))
      - There are n=3 possible permutations possible, but only one of these three has the double negative required for this operation to be successful.
      - Possible result,
        (+ A B (& C;x D);y;z)
  - Complicate - (+ (~ A) (~ B)) = (~ (& A B)),
    (& (~ A) (~ B)) = (~ (+ A B))
    - Example equation (+ A B (~ C) (~ D;x);y)
      - There is only one possible permutation.
      - Possible result,
        (~ (& (~ A) (~ B) C D;x;y)

Obviously these methods can be used to simplify, or complicate an expression. In the next section the application of these operators will be addressed.

3.7.3 Automatically Generating Boolean Equations

The operators described in the last section were described using permutation parameters. To manipulate the equation the user must specify the i) operation of interest, ii) simplification or complication, and iii) the permutation of the operation desired. As these are specified, the equation may be changed. If they are applied blindly using an exhaustive search, the possible variations are enormous (actually infinite). A directed searches is one possible method for reducing the

search space.

Exhaustive methods are the most dependable methods for finding any given term. But this reliability comes with a high cost on efficiency. In the case of simplifying the equation to a disjunctive normal form, the method is slow, but very effective. To reduce an equation to its simplest form, we need only to apply the operators discussed in the previous section, until all possible alternatives have been exhausted. It was found that to start with the most nested terms resulted in a more direct solution. The basic algorithm is described below.

1. Get a list of all terms in the equation, in hierarchical order. That is the last item on the list of terms, would be one of the first operations performed mathematically.
2. Pick the last term on the list, and begin testing to see if any simplifying operators can be applied. If no operators are fired, then remove the last item from the list, and begin step 2 again. If the list is empty, the equation is simplified, so quit.
3. If an operator was fired successfully, then return to 1.

For the software written, this Exhaustive Simplify operation takes a few seconds for an example with between 10 and 20 terms. If the code were optimized, the simplification would take less than a second.

Exhaustive Simplification has a goal, therefore making the algorithm easy. Exhaustive complication has no goals, but it continually expands an infinite tree systematically. More importantly, when simplifying, there is only one term at all times. But, when complicating, each new term must now be kept. This is obviously not very practical, but useful to note.

The exhaustive equation manipulations focus on complicating, and simplifying expressions. But, if we merely want to generate alternatives, we may use a more interesting algorithm. A Random Walk through equation space may be of

use for obtaining novel equation forms. In this case a term in the equation is chosen at random, and one operator is randomly selected, and the operators permutation is also randomly selected. This would allow the equation to be made simpler in some cases, more complex in others, and change the topology of the equation. While this is sort of a blind technique, it does lay the groundwork for a more complex heuristic search.

At present the simplification technique is the only one used, but the future potential for this method is excellent. The advanced use of equation manipulation will be discussed more later with reference to Meta-knowledge.

3.8 NON-BOOLEAN DESIGN DATA

While the equation forms presented before concentrate on high level representation of the product, they do not consider representation of set properties. To allow set properties to accompany the equations, a symbolic form of representation has been chosen. Each symbol in the equation has a list of symbols associated with it. This allows a set of arbitrary properties to be associated with each set. For example, if we have a Boolean expression '(* A (~ B))', it is possible for 'B' to have a list of properties such as,

B.form = CYLINDER
B.height = 2.0
B.radius = 1.0
B.height_max_tolerance = +0.02
B.material = 1040Steel

The reason for this representation is that it is not necessary to be more specific for this method. The reader will notice in the next chapter that the CAPP planning rules are made to use arbitrary symbols. This allows more freedom in the CAD system, where geometry is represented because it is essential, but the other prop-

erties are often ignored because they are a nuisance for the CAD software designer. The set 'B' above can be used to illustrate some common problems related to tightly specified data components. The form of 'B' is a cylinder. When this is interpreted in the solid modeler, the location of the origin for the primitive is critical. But, various solid modelers set the origin either at the centre of one end or at the centre of mass. Obviously this would lead to the definition of two different geometries.

The data structure to store symbol lists is given in Figure 38. In the diagram there are two lists, and a pointer into the lists. There are two separate types of lists stored. The first list is for all sets (not including members). The second list is specifically for the set members (also called properties). In the diagram there is a pointer to the set list. This serves as a reference to the position in the set list. The 'Term List' is used specifically for lists of objects (and is similar to the 'Term List' used for storing equations, and it uses the same C++ class). The 'pointer' field refers to symbols containing the name of the set. The 'type' field is filled in to indicate that it is a set pointer, but this is only useful for checking data integrity. Obviously the 'point to next' field is used for linking the lists. The list of symbols are treated in a similar manner.

The symbol list is used to store the set names and the symbol names. Pointers are also used for the set names. These pointers refer back to the term list, so as to give the list of symbols. The type field is used to differentiate between numeric and symbolic properties. This reason for this becomes apparent when trying to compare fields, and field values. Numeric values must be treated differently than character strings.

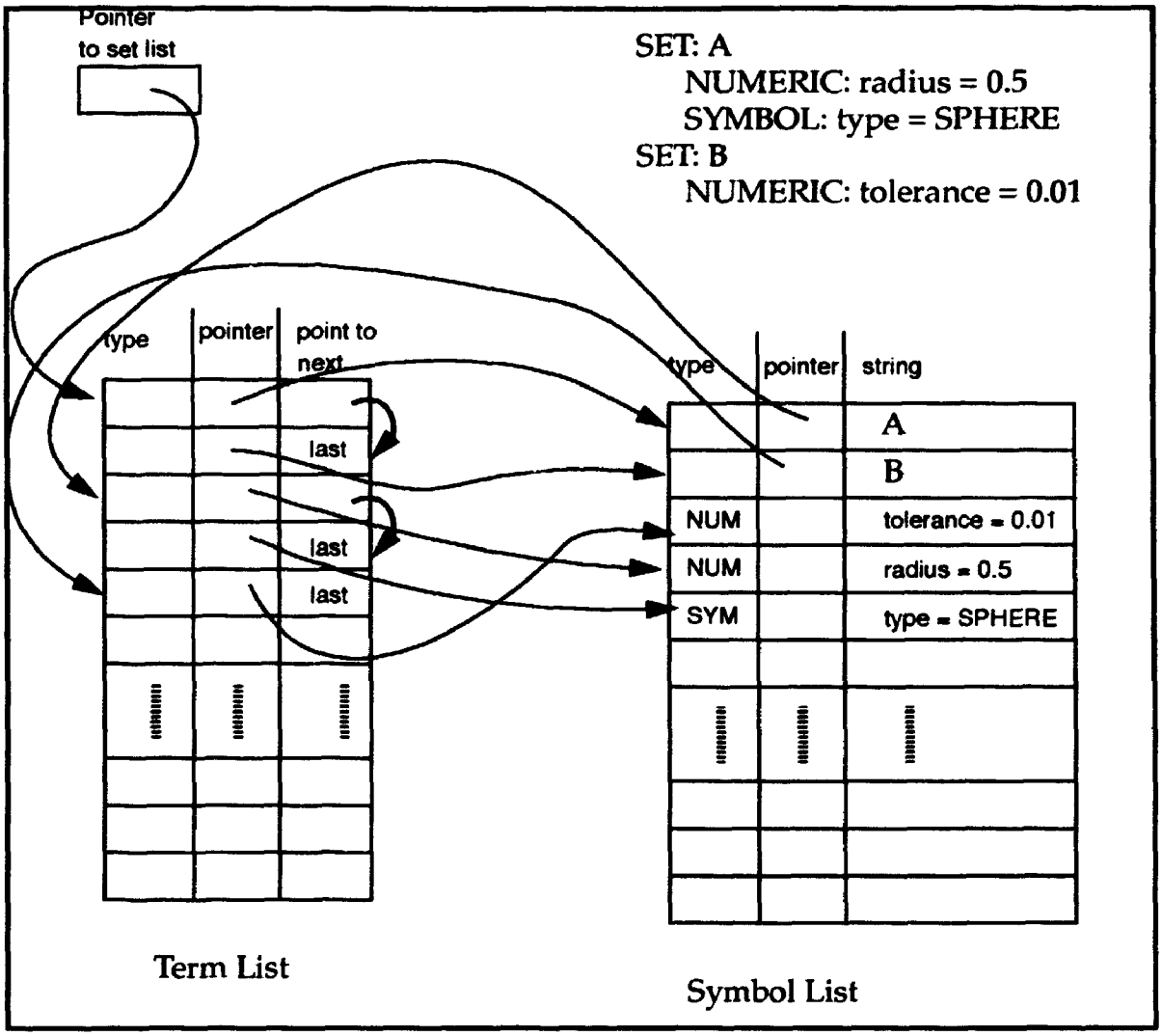


Figure 38 Set Data Structure (With Example)

A set of functions has been developed to manipulate the data structures. Some of these functions are,

- add a new set,
- add a new property symbol,
- parse a property symbol,
- find a reference to a set/property,
- remove a property,
- edit a property.

3.9 CONCLUSION

The basic theory behind solid modelling was reviewed, including the fundamental operators. Of particular interest was the Assembly operator. While it was shown that there are currently no satisfactory techniques for assembly modelling, it was made clear that techniques are evolving.

The product representation was presented, including a basic structure, and details necessary for a computer implementation. The representation includes data structures capable of storing and manipulating Boolean equations. These equations also refer to sets of properties. The final data structure was selected in such a way that the method of representation (r-sets/s-set/CNRG/etc.) did not have to be chosen. This can be said because at no point is the data structure interpreted physically, thereby forcing a selection of representation.

There are a number of highlights from various sections of this chapter.

- The attributes of sets, such as tolerances, have been left undefined to allow for the greatest flexibility in future use. (Only a few very fundamental properties have been defined, such as position).
- The computer was able to generate a graphical representation of the part based upon the part file. This indicates that the method is at least sufficient for geometrical representation.
- The nested Boolean form has been used to eliminate ambiguities in operator scope.
- Properties may be appended to sets and primitives. This allows reuse of basic parts as would be desired in an Engineering design environment.
- Most of the common Boolean algebra axioms are included in the equation manipulation subroutines.

4.A REVIEW OF AI PLANNING TECHNIQUES

4.1 INTRODUCTION

One of the fundamental skills of intelligence is planning. Generally, planning is described as finding sequences of actions to satisfy one or more goals. Despite its importance, the process of planning is not well understood. This has lead to a number of attempts to mimic the planning process with computer based experiments. These Artificial Intelligence planners use various algorithms to approach a fundamental set of planning problems.

By their nature, planning problems are defined using goal states. If there is a single goal, the problem is often simple. Planning for multiple goals is also simple if the goals are completely independent. Some goals must be satisfied in a particular order; this occurs when they are independent but ordered. The most complex case arises when a number of goals must be satisfied simultaneously (this is non-linear planning; all other cases are linear).

To move between a start and goal state, operators are applied. The choices and orders of operators are both major concerns. A great deal of work has been done for selecting operators and choosing their order, but problems of complexity still tend to overwhelm most planning methods.

4.2 PLANNING PROBLEMS

Researchers have embodied the planning problems with a symbolic approach. Thus, the problem is easily described in terms of lists, rules and expressions. This approach was first described by Green [1969] who discussed a planner based on resolution of logical statements.

4.2.1 Representation of Problem Space

The representation of problem space is primarily split into two main divisions. Initially the state of the world must be described by a model. As a plan is executed, the world model will evolve through time. If a world model is not used (for simulation) then the operators must be applied to the world directly. This will cause problems when backtracking operations to recover from a failed plan.

4.2.1.1 The World Model

A list of conditions (predicates) are stored in a list. As conditions in the world change, conditions are added and removed from the list.

4.2.1.2 The Solution (Plan) Model

Operators are applied that may update a world model. As the model changes, it will form a linear sequence of states over time. This is equivalent to a path through a graph [Korf, 1987]. As a result, the description of planning often takes on the terminology used for search.

4.2.2 Representation of Goals

At any time the world model contains a set of conditions. When some goals are introduced, they must describe the desired state of the world model. Thus, goals are described with a conjunctive expression of all the desired conditions in the world model. Each condition is considered a separate goal.

4.2.3 Operators

To describe actions, operators are used. Traditionally, an operator has a set of preconditions which must exist in the world model before an action can be performed. If the preconditions of an action are satisfied, the results of the action are

simulated by deleting and adding conditions to the world model.

It is important to note that this may result in one operator violating the preconditions of other operators.

4.2.4 Plans

The representation of plans is important (we will avoid the subject of planning until section 4.3). Ultimately, all plans must be stored as sequences of serial, concurrent or parallel preconditions and actions. It is also possible to store states after each operation. Including a state representation allows the ability to track plan progress, and to check for failures during execution. These advantages do not come without costs. It is easy to store sequential plans with all of their preconditions. Non-linear plans are much harder to store, because of the non-linear graph structure caused by inherent parallelism.

Korf [1987] described the various plan types and how they may be described with graph search theory. His work proved formally that sequential plan steps are preferable, although not always possible.

4.2.4.1 Sequential Plans

In its most simple form, a sequential plan is a list of operations. A more sophisticated plan also describes the state of the world model after each operation. One such method for doing this is triangle tables [Nilsson, 1980, pp. 282-286]. These tables store the result of each operation in (selected rows of) the column beneath. The rows selected are determined by which subsequent operations the results are preconditions for.

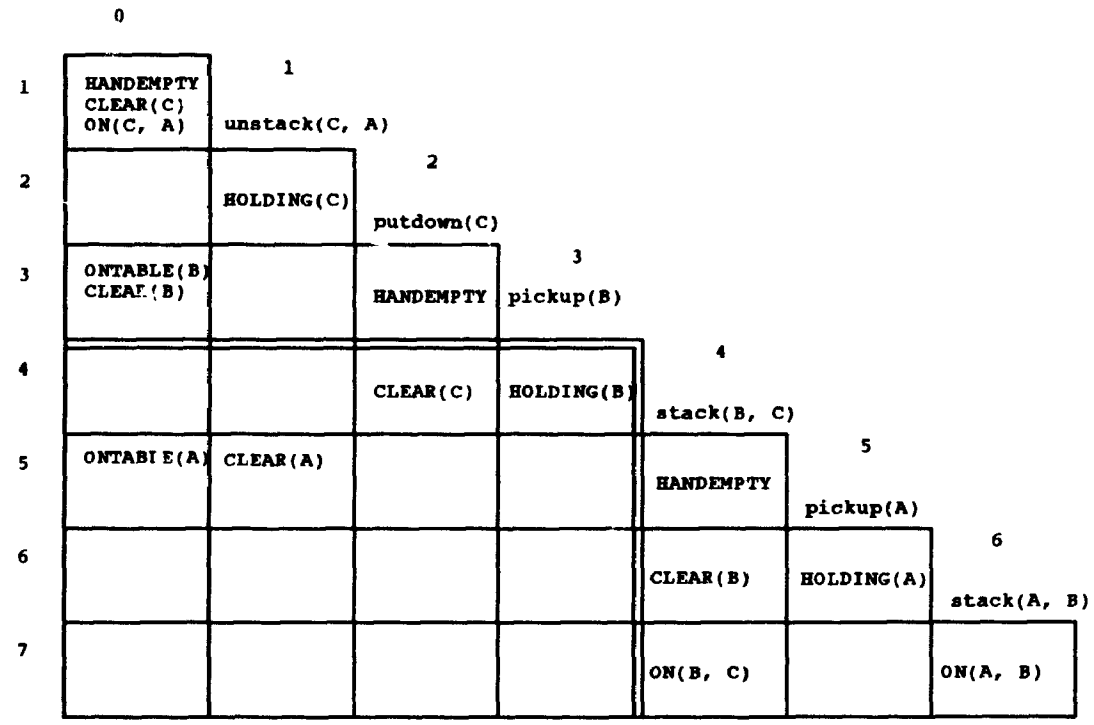


Figure 39 A Triangle Table
(Adapted from [Nilsson, 1980, pg. 285])

The plan's progress may be verified by examining the kernel. As shown in Figure 39, the kernel (outlined in double lines) for STACK(B, C) is a matrix of conditions. If these conditions match the world model, then the plan is progressing normally at STACK(B, C).

Nilsson also mentions the complete expansion of state space in a tree form [1980, pp. 282-284]. Each node becomes an intermediate plan state, and each arc becomes an operation. This approach is both memory intensive and inefficient, but recovery from plan execution failure is easier. An example of this approach applied to a blocks world problem is shown in Figure 40.

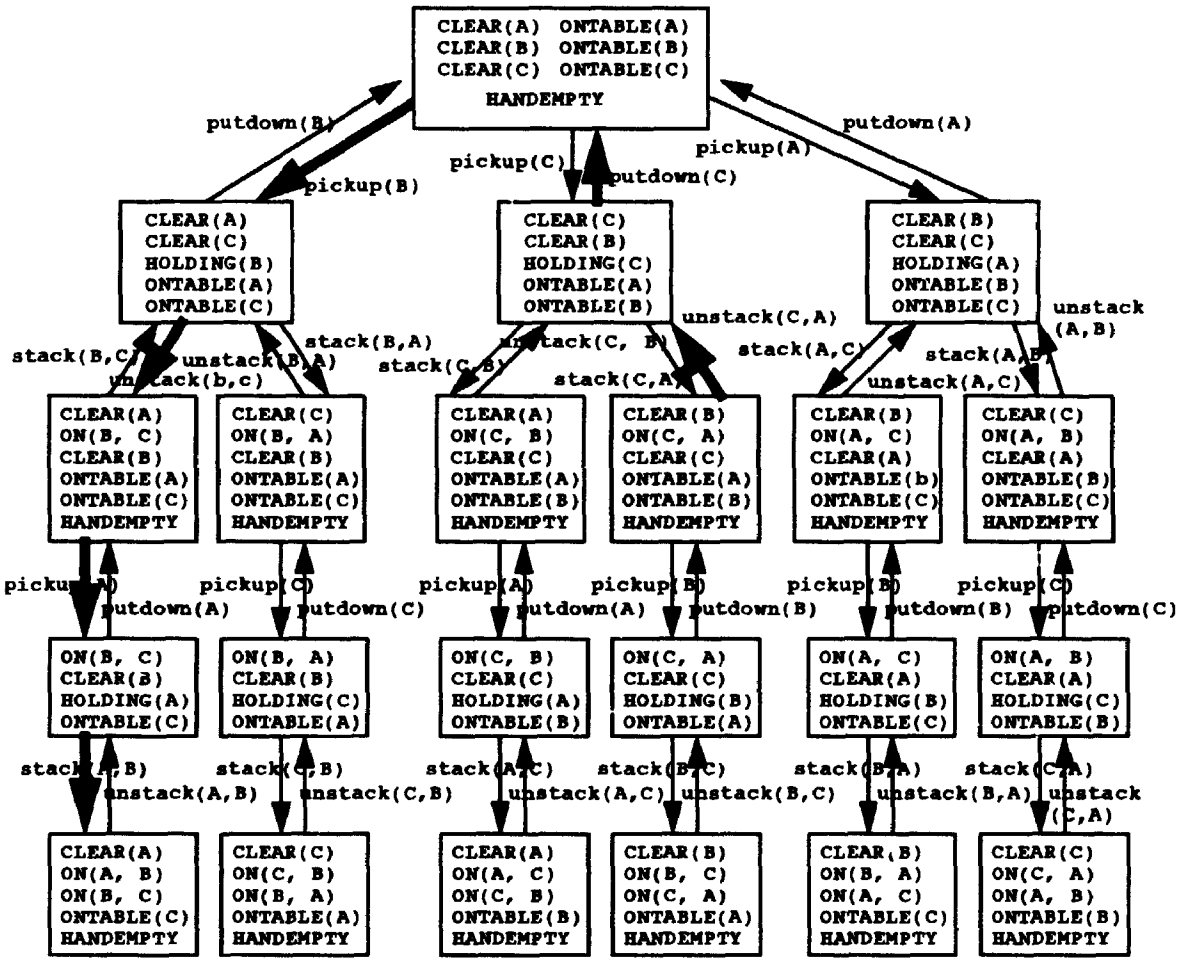


Figure 40 A Search Tree in Plan Space
Larger Arrows indicate path followed by planner
(Adapted from [Nilsson, 1980, pg. 283])

4.2.4.2 Non-Sequential Plans

By treating all plans as sequential, we have assumed that all operations are ordered, and two events may not happen simultaneously. In actuality, many operations are parallel in nature, and therefore it may be desirable to have plans which are concurrent and only partially ordered. The main disadvantages of unordered plans are that progress checking (and error recovery) will be more difficult, and the scheduling of events must still be specified.

An early planner called NOAH [Sacerdoti, 1975, 1977] used a representation called procedural nets. The nets had a starting point at the left hand side, from which plan execution would begin. During execution concurrent paths could branch and rejoin. Preconditions and operations were located at various positions along the paths. The only drawback to this representation is that conditions that were previously set can not be undone. Tate [1977] discussed an approach based on project networks, and claimed that he overcame the problem with procedural nets. These nets are similar to procedural nets except the goals and preconditions are not separated.

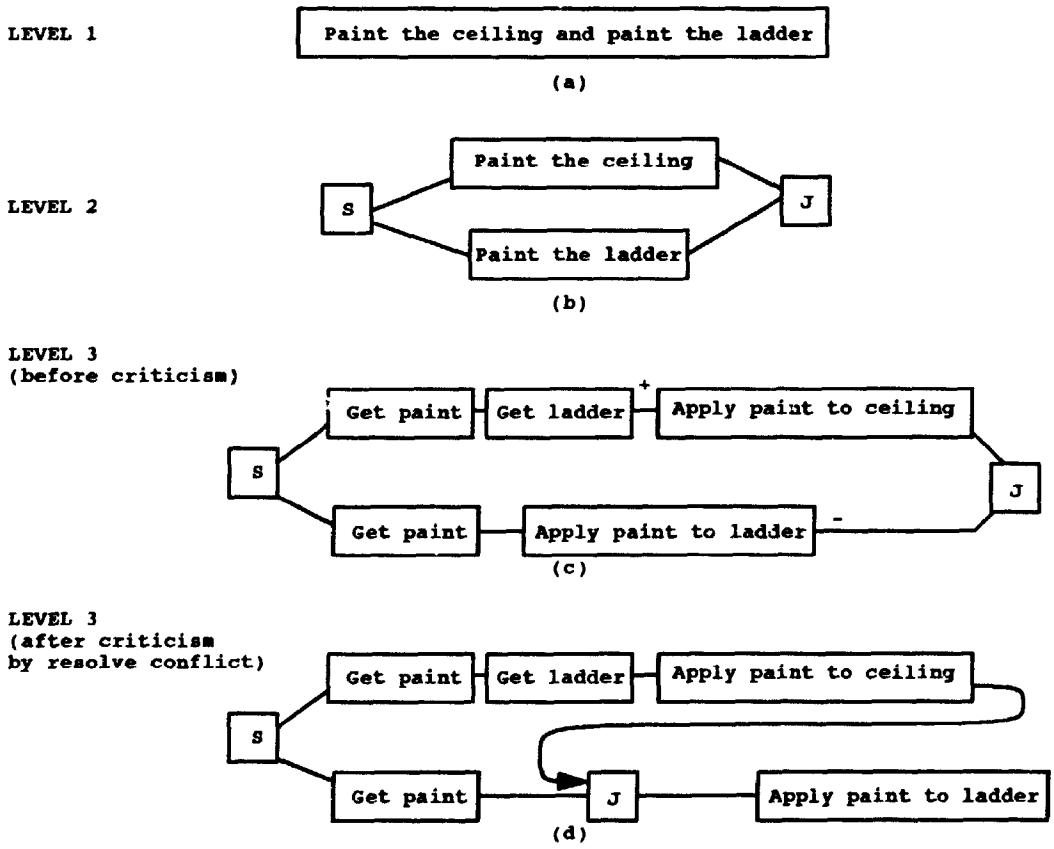


Figure 41 A Procedural Net (for painting)
(Adapted from [Sacerdoti, 1977, pg. 10])

Tate discussed finding an order for an unordered plan using Operations

Research techniques on a project network. DEVISER [Vere, 1983] was able to generate plan representations with non-sequential operations, using an approach based on Tate's project networks.

4.2.5 Planning Objectives

A real problem rarely has a unique solution. Typically there are either multiple solutions, or none at all. When there are multiple solutions, a value analysis must be performed to find the greatest valued plan. If there are no solutions, goals must be compromised to reach a solution. A diagram by Gevarter [1985] which he

derived from the book by Wilensky [1983] is shown below in Figure 42.

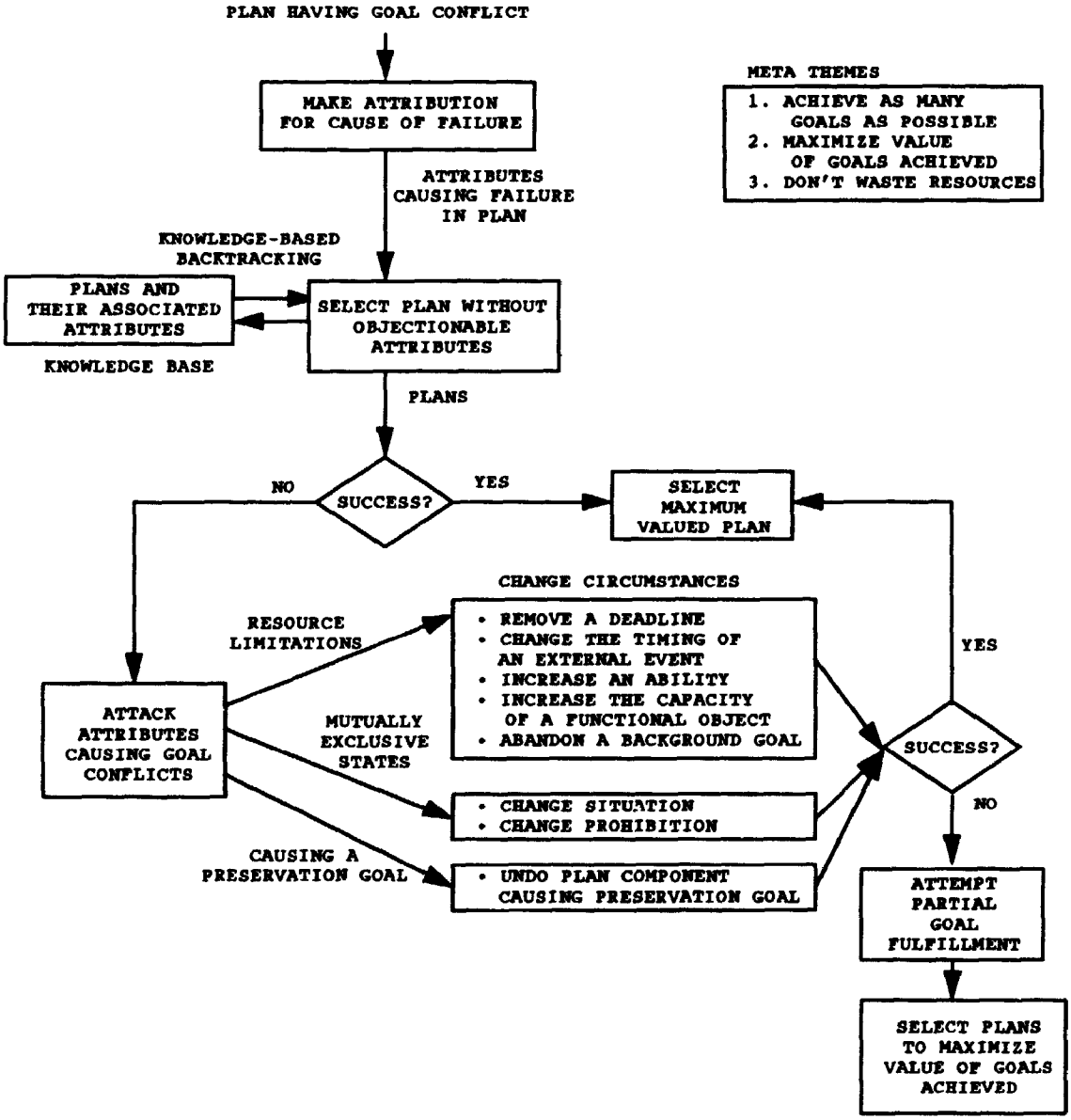


Figure 42 Objective Evaluation
(Adapted from [Gevarter 1985, pg. 71])

Although, this figure illustrates a comprehensive planning approach, planners commonly find the first solution possible, or quit if none is found.

4.3 PLANNERS

Early planners used structured environments that could result in a single plan. Later planners started to deal with the complexity of real world problems. This section discusses a number of approaches to planning, but first, an outline of planning issues is covered.

4.3.1 Planning Issues

Exhaustive solutions to planning problems are quite unfeasible for any large problem. For example, the large search tree for the simple blocks world problem makes this obvious in Figure 40. Thus, approaches have been formulated to exploit the nature of the planning process, and prune the search tree.

4.3.1.1 Means-End Analysis

A search for a plan may be extensive when working from the start node. Starting at the goal state and reducing the difference between the goal and the start makes the problem more tractable. This technique is referred to as Means-End Analysis and was originally proposed for GPS (General Problem Solver) by Newell and Simon [1972]. This is the most common method used in planning systems.

4.3.1.2 Inductive Plans

The reversibility of an action is important when generating a plan. If actions can be undone, then plans may be forward searched in the real world. This is because reversible actions make it possible to backtrack from a failed solution. If actions are not reversible, a world model must be used to simulate the effects of actions. Planning in the real world is not desirable, but may become necessary when the world model has randomly changing conditions.

4.3.1.3 General Comments on Plan Completeness

A search for a plan may be an overwhelming task. Thus, a number of simplifying assumptions are often made in planning,

- there is only one operation to perform at each task,
- operators are costless,
- operator order is costless,
- the world models are limited to the problems only,
- time is not a factor,
- plans are sequential,
- operations are concurrent (not parallel).

These factors tend to simplify the search space. But, even when searching the simplified problem spaces, there are a number of arbitrary decisions that have to be made. These usually arise because of the arbitrary sequence of operator execution, arbitrary choice of alternative operators, and lack of information. In the presence of arbitrary decisions the best plan can be found with the aid of cost estimates [e.g., Tate, 1977]. Lack of information is often dealt with by a least commitment strategy, where decisions are made as late as possible, or not at all [Sacerdoti, 1975, 1977 and Stefik, 1981 a, b].

4.3.2 Complexity

The complexity of planning is related to the number of viable plans in the plan space. Alternate plans can be created by unordered operators, and equivalent operators (they cause branches in the plan search space). A number of approaches have been developed to deal with the complexity issue. In general, these methods focus on dividing or constraining the search, so that the problem becomes tractable.

4.3.2.1 Non-Hierarchical Planners

When goals are used to directly find operators, the system is referred to as non-hierarchical. This name definition is commonly used to describe all planners which do not fall into the other categories, namely hierarchical, skeletal, and opportunistic planning. One attribute of these planners is that since they consider all of the search space at once, the search must cover a larger number of possible solutions. The common approach is to use a linear search through space. Some descriptions of the more famous planners follow. This includes descriptions of the most popular non-hierarchical planners, NOAH and STRIPS.

STRIPS - described in [Nilsson, 1980], is based on a goal stack. To begin, the goals are pushed on the goal stack. An Operator is found which satisfies the top goal on the stack, and it will also be pushed on the top of the stack. The preconditions for the operator are then also pushed on the stack. The basic process continues as the condition currently at the top of the stack is examined. If the condition of the top of the stack is satisfied (by the world model), it is removed. If an operator is at the top of the stack all its preconditions have been satisfied, the operator is added to the plan, and used to update the world model. If the condition at the top of the stack is unsatisfied, it will be replaced with an operator which will satisfy it (then all of the operators preconditions will also be pushed on the stack). This process continues until the stack is empty, thus indicating a complete plan.

It is possible for strips to become caught in certain situations that involve non-linear plans, and fail to produce a solution. In some cases STRIPS is able to find solutions to hard problems, but still results in inefficient solutions.

HACKER - Sussman's work on HACKER [1974] used a plan and repair approach. If an existing plan satisfied the goals presented, it was used. Otherwise, a plan was generated by subdividing the goals until they could be matched to operators. The plan could then be examined by a set of daemons (monitor processes) that can detect known problem patterns, and fix them. Afterwards, the plan was simulated, and any existing errors were found. The problems that caused the errors were identified, and fixed with general debugging techniques. The problem patterns were then stored as new daemons for later problem pattern matching. This process continued until the plan was error free.

Sussman's [1974] examination of this method has become well known because of the failure of HACKER (a linear planner) to perform a certain three block problem. Dubbed the Sussman Anomaly, the problem involved blocks stacked in such a way that two goals had to be combined. This served as an excellent example of the limitations often encountered by the simpler linear planning schemes.

Goal Regression - Waldinger [1977] proposed a planner that would satisfy one goal at a time. The planner would arbitrarily choose one goal as the first and find a plan for it. A second goal would be incorporated into the plan by trying to expand the second goal after the last step in the first plan. If the preconditions of the second goal plan were violated then the attempt would fail and the second goal would be reconsidered before the last step in the first plan. If the second goal could not be expanded there, then it would be moved though the first plan by another step. This regression will continue until the goal is successful, and is expanded, or it fails by reaching the start of the other plan.

Waldingers approach did not discuss many of the detailed issues, and was lacking an implementation to verify his ideas. The planner could fail in some situ-

ations because of its inability to violate the precondition constraints. But, he did discuss some of the anticipated side-effects which would occur if the preconditions could be violated (and goals expanded prematurely). He asserted that some previously unsolvable problems could be solved, but he also stated that some problems could arise. For example, it is possible to have a solution which repeatedly does an action and then undoes it. It is also possible to generate inefficient solutions having needless operations.

4.3.2.2 Hierarchical Planners

A more efficient approach to planning involves least commitment examination of the various hierarchies of detail. In other words, the planning should begin at an abstract high level [abstractions are discussed in Korf, 1987]. As planning progresses, the abstract goals should be expanded into more detailed sub-goals. Planning becomes a task of examining a small search space, pruning the search, then expanding the new pruned search space. This makes problems with a large search space tractable (e.g., non-linear plans).

ABSTRIPS - a simple approach to hierarchical planning was based on STRIPS [Nilsson, 1980]. This new planner was called ABSTRIPS [described in Cohen and Feigenbaum, 1982], and only involved the addition of priorities to the preconditions already used in STRIPS. The planning techniques were similar to STRIPS, except that all of the first stage of planning was done with the highest priority conditions. After a complete high level plan was pushed on the stack, it was expanded at a lower priority level. The effect was a greatly reduced search space which could allow solutions of much greater complexity.

In terms of abilities, ABSTRIPS has all the features of STRIPS, but is much more efficient at solving large problems.

NOAH, NONLIN and DEVISER - the procedural nets mentioned earlier are important to NOAH [Sacerdoti, 1975, 1977]. If the reader refers to Figure 41 they will note the square nodes. These are used to represent both goals and sub-goals. At each iteration of the search, the square nodes are expanded into sub-plans. After this, three constructive critics are applied to resolve conflicts, eliminate redundant preconditions, and deal with unbound variables. Resolving conflicts involves ordering unordered operations to avoid precondition violations. If preconditions are specified more than once, they should be recognized and the redundant ones eliminated. Unbound variables can occur because of the general form of operation definitions. This can lead to problems assigning them. The critics approach is to bind them to a formal variable or to an instance. The least commitment strategy dictates that the formal variables (or unbound variables) should only be bound to an instance when no other alternatives exist.

Tate [1977] indicates that Sacerdoti's approach has problems because it is unable to backtrack and undo failed plan steps. Therefore, Tate developed NONLIN, which is like NOAH except that it keeps lists of all decisions made (and all goals achieved) so that backtracking is possible in planning (as well as execution checking). If a plan cannot be found, NONLIN backtracks and tries a new solution.

Vere [1983] states that both NOAH and NONLIN assume that tasks are concurrent, thus trivializing the time planning problem. DEVISER [Vere, 1983] was based upon NONLIN, and can consider parallel tasks with critical execution times. The planner was able to plan device operation plans for a fly by of an interplanetary satellite. The significant difference between NONLIN and DEVISER is the addition of time windows in the consideration of conflict resolution.

MOLGEN [Stefik, 1981a, b] - uses a less general approach to planning. Three layers of abstraction are used: the strategy, design and planning spaces. Goal relations are considered and manipulated in strategy space. Details of the plan are generated in the planning space. In the design space, constraints are considered. The interesting feature of MOLGEN is its use of constraints. Details of each operation include constraints. As operations are specified, the constraints are formulated, and propagated to be considered against constraints from other parts of the plan. MOLGEN then chooses objects to satisfy the constraints of the operations. If a plan is formulated at the strategy level, but the constraints cannot be satisfied at the design level, then the strategy level will replan.

This planner was successful with the complex problem of planning molecular genetics experiments, and shows promise for other more sophisticated problems.

4.3.2.3 Skeletal Planners

The technique of skeletal planning is variant, not generative as the previous methods are. If a successful plan template is available from earlier planning sessions (or elsewhere) it is stored in a plan database. Before planning begins, the goals are compared against the skeletal plans. If plans are found to satisfy the goal pattern, they are then checked for their success with the current world model. Any remaining plans will be examined for suitability, and the best will be chosen. A successful implementation of this technique was demonstrated with MOLGEN [as described in Cohen and Feigenbaum, 1982, pp. 557-562].

4.3.2.4 Opportunistic Planning

In his book Gevarter [1985] describes an approach to planning which is opportunistic. This only refers to a technical report (by Hayes-Roth and Hayes-Roth), but suggests that plans may be developed in two stages. First, parts of a plan may

be devised with backtracking. Secondly, the parts may be liked together, added to, and enlarged as opportunities become available. Gevarter asserts that this is a more human like approach to planning.

4.4 CONCLUSION

This chapter has reviewed AI planning techniques as they apply in general. Process planning can involve some, or all of the techniques described here. The next chapter will discuss planning within the domain of manufacturing processes.

The reviewed planning literature makes it obvious that work needs to be done in planning. In particular, planners need to be developed that will deal with complex problems, that have process costs. It is the author's opinion that as existing planners are applied to more applications, the existing planning techniques will be refined to meet the needs of the real world problems. A few features are listed below that would be desired in a planning system.

- Be able to deal with linear and non-linear plans.
- Produce a plan in a shorter time with a slightly higher cost.
- Produce plans with low costs, having greater computation time.
- Deal with plans involving millions of steps.
- Deal with equivalent operators.
- Learn from experience.
- Be able to backtrack in the event of plan execution failures.
- Be able to perform inductive and deductive reasoning.
- Deal with probabilistic events.

The hierarchical planning approach has performed well on many of these points, suggesting that is an excellent method to consider for future planning research and application.

5.PLANNING TO FIND MANUFACTURING PROCESSES

5.1 INTRODUCTION

There are fundamental concepts when manufacturing a part. For example, a part may be machined from a block, or cast from molten metal. Even though the results may be the same, the processes have an inherent difference. Recognizing the geometrical impact of a process is essential to this thesis. Thus, converting the design geometries of the part directly into manufacturing geometries requires some understanding about the formation of a part.

5.2 MANUFACTURING PROCESSES

Before discussing the rules and planning strategy used, it is important to outline the manufacturing context they are used in.

5.2.1 Basic Factors Considered in Process Selection

There are a number of factors commonly considered when selecting processes. Some of these can be quickly listed, as below (not in order).

- Tolerances
- Material
- Quantity
- Feature Accessibility
- Safety
- Finish
- Cost

The priority of these factors vary depending upon the industry, and the product. The process selected also determines which of these factors are relevant.

5.2.2 Conceptual Process Groups

At present, processes are often described in loose terms. The classifications vary between applications, and therefore it is hard to develop a unified model of

all manufacturing processes. A list of processing concepts is given which represents what the author feels is a reasonable division of some modern manufacturing processes. Each of these process groups is considered to be independent, and complementary.

- Molding - Forming amorphous material into a solid shape through the use of a form, or mold.
- Joining - Causing a permanent kinematic bonding between parts through some sort of connection on a molecular level.
- Assembly - A group of parts which form a new part through couplings, fasteners, screws, etc. An assembly requires no bonding.
- Cutting - Creating new parts by severing an existing part.
- Machining - Deliberate removal of material to develop a newer, smaller geometry.
- Finishing - Processes which have imperceptible effect on the geometry.
- Forming - An intentional deformation to produce a new geometry.
- Handling - The physical movement of parts between locations and orientations.

By grouping the processes in this way, they can be matched to geometric functions (eventually the Boolean equation templates will be used). A list is presented below that shows how various processes match the classifications above. While this is not complete, it does indicate the intention of the author. Please note that the classification items vary for each of the following process groups.

```
• Molding
    • Casting
        - Gravity Fed Casting
            - Hollow Molds
                - Green Sand
                - Dry Sand
                - Core-sand molding
                - Cement bonded sand molding
                - Carbon Dioxide Process
                - Plaster Molds
                  Loam Molds
                - Shell Molding
                - Ceramic Molds
                - Graphite
                - Permanent Molds
            - Consumable Cores
                - Precision Molding (Investment Casting)
```

- Forced Fed Casting
    - Die Casting
        - Hot Chamber Machines
        - Cold Chamber Machines
    - Centrifugal Casting
        - True Centrifugal Casting
        - Semi-Centrifugal Casting
        - Centrifuging
    - Continuous Casting
    - The V-Process (Vacuum Casting)

• Powdered Metals
- Pressure Forming
    - Conventional Compaction
    - Vibratory Compaction
    - Powder Extrusion
    - Powder Rolling
    - Hot Isostatic Pressing
    - Cold Isostatic Pressing
    - Explosive
    - Forming with Binders
- Pressureless
    - Loose Sintering
    - Slip Casting
    - Slurry Casting

• Plastics
- Casting
- Injection molding
- Compression Molding
- Transfer Molding
- Extrusion
- Rotational Molding
- Open-Mold Processing
- Pultrusion
- Filament Winding

• Joining
- Mechanical Joining
    - Riveting
- Welding
    - Cold Pressure Welding
        - Cold pressure Welding of sheets and wires
        - Ultrasonic
        - Explosive
    - Hot Pressure Welding (Molten Metal)
        - Percussion
        - Resistance Flash
        - Resistance Spot
        - Resistance Seam
        - Resistance Projection
        - Thermit
    - Hot Pressure Welding (Solid State)
        - Diffusion Bonding
        - Friction Welding
        - Inertia Welding
        - Induction Welding

- Resistance Upset (Butt)
- Fusion Welding
- Arc Welding
- Shielded metal arc welding
- Carbon Arc Welding
- Flux-cored arc Welding
- Stud Welding
- Submerged-arc Welding
- Gas Metal Arc Welding
- Gas-tungsten Arc Welding
- Plasma-arc Welding
- Electroslag Welding
- Gas Welding
- Electron Beam Welding
- Laser Welding

- Sticking
- Brazing and Soldering
- Soldering
- Brazing
- Torch brazing
- Furnace Brazing
- Induction Brazing
- Dip Brazing
- Salt Bath Brazing
- Resistance Brazing

- Adhesives
- Epoxies
- Phenolics
- Polyamide
- Silicones

- Plastics
- Thermal Bonding
- Ultrasonic

• Finishing
- Surfacing
- Hardfacing
- Powdered Metals
- Sizing
- Machining
- Oil Impregnation
- Infiltration
Heat Treatment
- Steam Oxidizing
- Plating

• Cutting
- Thermal Cutting
- Oxy-fuel cutting
- Arc Cutting
- Conventional Arc
- Air-arc
- Oxygen-arc
- Carbon-arc
- Tungsten-arc
- Air-carbon-arc

- Plasma-arc
- Laser Cutting

- Mechanical Cutting
  - Shearing
  - Bar Cropping
  - Blanking
  - Cutting Die Construction
  - Sawing

- Forming
  - Working
    - Rolling
    - Metal Drawing
    - Wire Drawing
    - Tube Drawing
  - Extrusion
    - Direct
    - Indirect
    - Hydrostatic
    - Impact
  - Forging
    - Open-die
    - Closed-die
    - Drop-Forging
    - Press-Forging
    - Horizontal Forging
  - Cold Forming
    - Sizing
    - Swaging
    - Coining
    - Cold Heading
  - Press operations
    - Punching
    - Flanging
    - Forming
    - Rubber forming
    - Embossing
      - Beading
      - Offsetting
    - Hydroform
  - Bending
    - Hemming
    - Corrugating
    - Wiring
    - False Wiring
    - Roll Bending
    - Rotary Bending
  - Drawing
    - Deep Drawing
    - Ironing
  - High Energy Rate
    - Explosive Forming
    - Electrohydraulic Forming
    - Electromagnetic Forming
  - Spinning

- Plastics
    - Blow Molding
    - Thermoforming
    - Calandering
- Machining
    - Rotational
        - Turning
            - Lathe
                - Cylindrical Turning
                - Facing
                - Groove Cutting
                - Boring and Internal Turning
                - Taper Turning
                - Thread Turning
        - Drilling
            - Conventional Drilling
            - Boring
            - Counterboring
            - Spot Facing
            - Countersinking
            - Reaming
            - Tapping
    - Prismatic
        - Shaping and Plaining
            - Horizontal Push-Cut Shapers
            - Vertical Shaper
            - Planer
        - Milling
            - Face Milling
            - Peripheral Milling
        - Grinding
            - Surface Grinding
            - Cylindrical Grinding
            - Internal Grinding
            - Centreless Grinding
        - Broaching
    - Miscellaneous
        - Ultrasonic
        - Abrasive Jet
        - Chemical Machining
        - Electrochemical
        - Electrodischarge
- Assembly
    - Packaging
    - Screwing and Bolting
        - Manual
        - Automatic
        - Robotic
    - Press Fitting
- Handling
    - Small Parts
        - Robots

This list is too long, but it is also far from complete. (See Chang [1990], and Dieter [1991] for more information on processes, and their limitations).

5.2.3 Process Data Parameters

Four common process selection parameters listed are Geometry, Material, Tool, and Machine. The machine specified will allow examination of factors like specific tolerances for a machine, horsepower of the machine, pressure of the machine, or other parameters of interest. The tool information will allow access to things such as tool material, tool geometry (including chip breakers, shape of tool, methods of fixation, etc.). Using this information along with workpart geometry and material, the machining process may be selected in terms of tolerance capabilities, time, and cost. Workpart geometry and material also give information about possible processes for the part. The Geometry of the feature (including dimensions, tolerances, and geometric form) to be produced will allow, or disallow a process as a candidate. The material can also be a design factor in selecting a process.

5.2.4 Manufacturing Materials and Process Groups

In some cases materials can determine a process. The list below shows some common processes and a couple of materials that they are suited to.

- Casting
  - Cast Steels
  - Cast Grey Iron
  - White Cast Iron
  - Ductile Cast Iron
  - Compacted Graphite Cast Iron
  - Malleable Cast Iron
  - Cast Aluminum and Alloys
  - Cast Copper Alloys
  - Magnesium Alloys
- Sheet Metal

- Shearing
- Blanking

• etc

More information about processes, and materials is available in Dieter [1991].

5.3 SELECTING PROCESSES FROM BOOLEAN EQUATIONS

The number of manufacturing details can be enormous. As discussed before, there is only one complete CAPP system that has tried to reason with a CSG model alone [Woo, 1977]. When Woo did his work he was attempting to recognize features by looking for geometrical connections between primitives. This by itself would be very limiting, and undoubtedly would invoke the problems arising from the non-uniqueness of CSG models. If his work had continued, he would probably have done reasoning based on the equation form.

The rules discussed in the following pages should be viewed as manufacturing transformations, much like those discussed in Delbressine and van der Wolf [1990], Delbressine and Hijink [1991], and Delbressine et. al. [1993]. While the concept of design with features is not different, the features they use are tool geometries, and their CSG transformation of the workpiece.

5.3.1 Equation Templates

There is a structural relationship between the 8 process groups, and the Boolean form of equations. The list below shows some examples of the Boolean equation templates, and the processes they are associated with (templates are covered in more detail later).

1) ... (~ (~ A)) ...	Molding shape A (molding)
2) ... (& ... (~ ... A ...) ...) ...	Cutting off shape A (milling)
3) ... (& ... (~ ... A ...) ...) ...	Machining out cavity of shape A (drilling)
4) ... (& A B ...) ...	From shape A cut off shape B (sawing)

5) ... (+ A B C) ...	Join shapes B and C to A (welding)
6) ... (: A B C ...) ...	Assemble parts B, C, ... to base part A (robot)
7) ... A ...	Finish part A (part A has color=red, so paint)
8) ... A ...	Part A has a feature that calls for forming
9) ... A;move ...	Move part A (robot)

The form used in this list indicates where other arguments may exist by using '...'. Also note that the interpretation is sensitive to the content of the sets. For example, if 2) and 3) are compared, if 'A' is a cylinder then 3) would be the obvious choice. If 'A' is a box, or a half-space, then 2) would be the logical choice. Using these cases a set of manufacturing rules may be developed. These rules will use the form of the equation, and the (somewhat arbitrary) set properties to select a manufacturing process. As can be seen this list is limited, but still suggests how the templates can be matched up to various manufacturing processes. The reader is also directed to 7) and 8) above. Both of these are identical, and are only differentiated by the properties in the sets.

5.3.2 Data Structures for Rule Storage

The rules are stored in a hierarchical manner. The list contains,

1. a group of equation templates, where each refers to rules they may fire,
2. conditions (to be used by rules),
3. rules, including conditions to be satisfied, and results,
4. results to be performed when a rule is satisfied.

This approach requires a large degree of refinement in the future, but for now it is sufficient to provide the rule matching information. The rule structure has been chosen to be intentionally segmented, so that results, conditions, and rules may be reused when possible, therefore reducing the number of comparisons. The structure can be logically discussed within the context of the example in Figure 43 below (keep in mind that this example is in no way complete).

```
equation_form EQN_1 {
     EQUATION: ( & ...( - VAR:V:0 ):LABEL:REF
     RULE: drill_hole
     RULE: chamfer_drill
}
equation_form EQN_9 {
     EQUATION: (> : VAR:V:0 VAR:V:1 ...)
     RULE: assemble_parts
}
rule assemble_parts {
     EQUATION: ( & PART_0_EXISTS PART_1_EXISTS )
     RESULT: ASSEMBLE_PARTS
}
rule drill_hole {
     EQUATION: ( & CYLINDER_SHAPE DRILL_SIZE )
     RESULT: DRILL_HOLE
}
rule chamfer_drill {
     EQUATION: ( & CONE_SHAPE DRILL_SIZE )
     RESULT: DRILL_CHAMFER
}
condition PART_0_EXISTS {
     COMPARE ( V0.form $ )
}
condition PART_1_EXISTS {
     COMPARE ( V1.form $ )
}
condition CYLINDER_SHAPE {
     COMPARE ( V0.form == CYLINDER )
}
condition CONE_SHAPE {
     COMPARE ( V0.form == CONE )
}
condition DRILL_SIZE {
     MATH ( V0.ratio = V0.height / V0.radius )
     ASSIGN ( V0.minimum = 0.5 )
     COMPARE ( V0.ratio > V0.minimum )
     COMPARE ( V0.ratio < 20 )
     COMPARE ( V0.radius > 0.05 )
     COMPARE ( V0.radius < 2.5 )
     COMPARE ( V0.height > 0.1 )
     COMPARE ( V0.height < 15.0 )
}
```

Figure 43 Part of a Rule File

```
result DRILL_HOLE {
     EQUATION_DELETE_VARIABLE_TERM ( REF )
     ADD_SET ( DRILL_HOLE )
     ADD_PROPERTY ( DRILL_HOLE a1 = 0.5 )
     ADD_PROPERTY ( DRILL_HOLE a2 = 1.0 )
     ADD_PROPERTY ( DRILL_HOLE height = 0.0 )
     PROPERTY_FUNCTION_VARIABLE ( DRILL_HOLE height + V0 height )
     PROPERTY_FUNCTION_NUMBER ( DRILL_HOLE height / 20.0 )
     PLAN_PUSH_TEXT ( ' DESCRIPTION ( drill hole ) ' )
     PLAN_PUSH_TEXT ( ' CUTTING ( FEATURE single diameter hole ) ' )
     PLAN_PUSH_TEXT ( ' CUTTING ( MACHINE qa1000 ) ' )
     PLAN_PUSH_TEXT ( ' CUTTING ( MATERIAL t-7075 ) ' )
     PLAN_PUSH_FORMAT ( ' CUTTING ( PARAMETERS ' DRILL_HOLE.height
      DRILL_HOLE.a1 DRILL_HOLE.a2 ' ) ' )
     // A costing section is added here for a trial run
     ADD_PROPERTY ( DRILL_HOLE cost = 0.0 )
     PROPERTY_FUNCTION_VARIABLE ( DRILL_HOLE cost + V0 height )
     PROPERTY_FUNCTION_VARIABLE ( DRILL_HOLE cost * V0 radius )
     PROPERTY_FUNCTION_NUMBER ( DRILL_HOLE cost * 1250.25 )
     PROPERTY_FUNCTION_NUMBER ( DRILL_HOLE cost + 0.25 )
     DECLARE_COST ( DRILL_HOLE cost )
}
result DRILL_CHAMFER {
     EQUATION_DELETE_VARIABLE_TERM ( REF )
     ADD_SET ( CHAMFER_HOLE )
     ADD_PROPERTY ( CHAMFER_HOLE a1 = 0.5 )
     ADD_PROPERTY ( CHAMFER_HOLE a2 = 1.0 )
     ADD_PROPERTY ( CHAMFER_HOLE height = 0.0 )
     PROPERTY_FUNCTION_VARIABLE ( CHAMFER_HOLE height + V0 height )
     PROPERTY_FUNCTION_NUMBER ( CHAMFER_HOLE height / 20.0 )
     PLAN_PUSH_TEXT ( ' DESCRIPTION ( drill chamfer ) ' )
     PLAN_PUSH_TEXT ( ' CUTTING ( FEATURE Front Countersunk Hole ) ' )
     PLAN_PUSH_TEXT ( ' CUTTING ( MACHINE qa1000 ) ' )
     PLAN_PUSH_TEXT ( ' CUTTING ( MATERIAL t-7075 ) ' )
     PLAN_PUSH_FORMAT ( ' CUTTING ( PARAMETERS ' CHAMFER_HOLE.height
      CHAMFER_HOLE.a1 CHAMFER_HOLE.a2 ' ) ' )
}
result ASSEMBLE_PARTS {
     EQUATION_DELETE_TERM ( :1 )
     PLAN_PUSH_TEXT ( ' DESCRIPTION ( add part to fixtured part ) ' )
}
```

Figure 43 Part of a Rule File (cont'd)

The 'equation_form' sets identify possible templates that the equation may be matched to. This is obviously to drive a pattern matching system that identifies possible candidates from each expression. This pattern matching is likely to turn up a good number of candidates from each Boolean expression. After these candidates have been matched, all 'RULE:'s identified in each matching 'equation_-

form' are checked. The conditions of the 'rules' are determined by the 'EQUATION:' field. The equation is interpreted for Boolean truth using the conditions as the logical statements. If the equation yields a Boolean truth, then all of the 'RESULTS:' are performed. The conditions are simple, they allow some simple calculations using 'MATH' and 'ASSIGN', and comparisons using the 'COMPARE' statements. If any of these items fail, then the condition fails.

The most structured part of the rules are the results. The various operations here will be described in detail later, but for now it is useful to point out that the results section can create new sets, alter their properties, alter the equation structure, and report on costs of the results. The rules sections also add steps to the process plan. The system internally keeps the effects of rules isolated so that the effects of a rule firing will only affect rules performing subsequent operations. This allows the process planning options to be done in a tree structure. If we wish to return to a previous state, then the complete state information is stored and no backwards reconstruction is required.

5.3.3 Equation Templates

If patterns are to be found within the equation structure, then we must have a method for describing these patterns. A method was developed for describing these patterns. There are two requirements of templates for doing this matching. First the templates should be able to represent any number of fixed and variable forms that the equation could match. Secondly, if variable equation forms are to be used, then variables must be available, so that the rules may refer to equation members through abstractions. The template form has a number of basic elements,

(X Start of a term, where 'X' is '?' for any operator, or '+, &, :, ~' for a specific operator.

) End of a term.

VAR:name:i
Refers to a variable list of all sets within a term (not including sets).

VAR:name:x
As above, but where the variable is 'x' position from the start.

(> This refers to the start of the expression only.

...(Allows a list of variables to be ignored before the start of a term.

...) Allows a list of variables to be ignored before the end of the term.

):LABEL:X
Allow a variable 'X' name to be bound to a term.

):VAR:X Assigns all of the properties of a term to be given a variable name.

VAR:name:i:prop
Same as above, but now it must have properties.

This list of elements can then be strung together to set up a template for matching. The author has written a simple algorithm for matching these templates to given equation segments, using a piecewise comparison. An example of this is given below.

EQUATION: (& (~ B) (~ C) (& A E) (~ D)) (5.1)
TEMPLATE: (& ...(~ VAR:V:0):LABEL:REF (5.2)

The template given could be for a hole drilling operation. To begin, the algorithm examines the equation, and template one token at a time from left to right. In this example, both the template, and equation begin with a '(', and so the comparison continues. Next, the '&' operation matches for both, and the comparison continues. The template string then has a '...(' condition that is fulfilled because the equation has a '(', but if there were variables before it, they would have been skipped. Next, the '~' operator is matched. Next, the 'VAR:V:0' operator is detected, and this is matched to the 'B'. In terms of the rules, 'B' is now referred to as 'V0'. This notation adds the order value to that variable name to avoid potential ambiguities. Both the template, and the equation now match for the ')', except the template also has a ':LABEL:REF'. This will allow the term '(~ B)' to be refer-

enced using the description 'REF' within the remainder of the rule execution. At this point, the pattern matcher notices that the template string is exhausted, and as a result concludes there is a match, and this will subsequently lead to the checking of conditions in the rule stage.

The template matcher is not very sophisticated, and is applied to each and every term in the given expression to find all possible matches (See Figure 44 below). As this method is explored more deeply in the future, the author hopes that this method of template matching can be improved. At present some cases are difficult to consider, such as if we attempted to refer to the third '(~ D)' term as being the third of that type, not the fourth in the sequence.

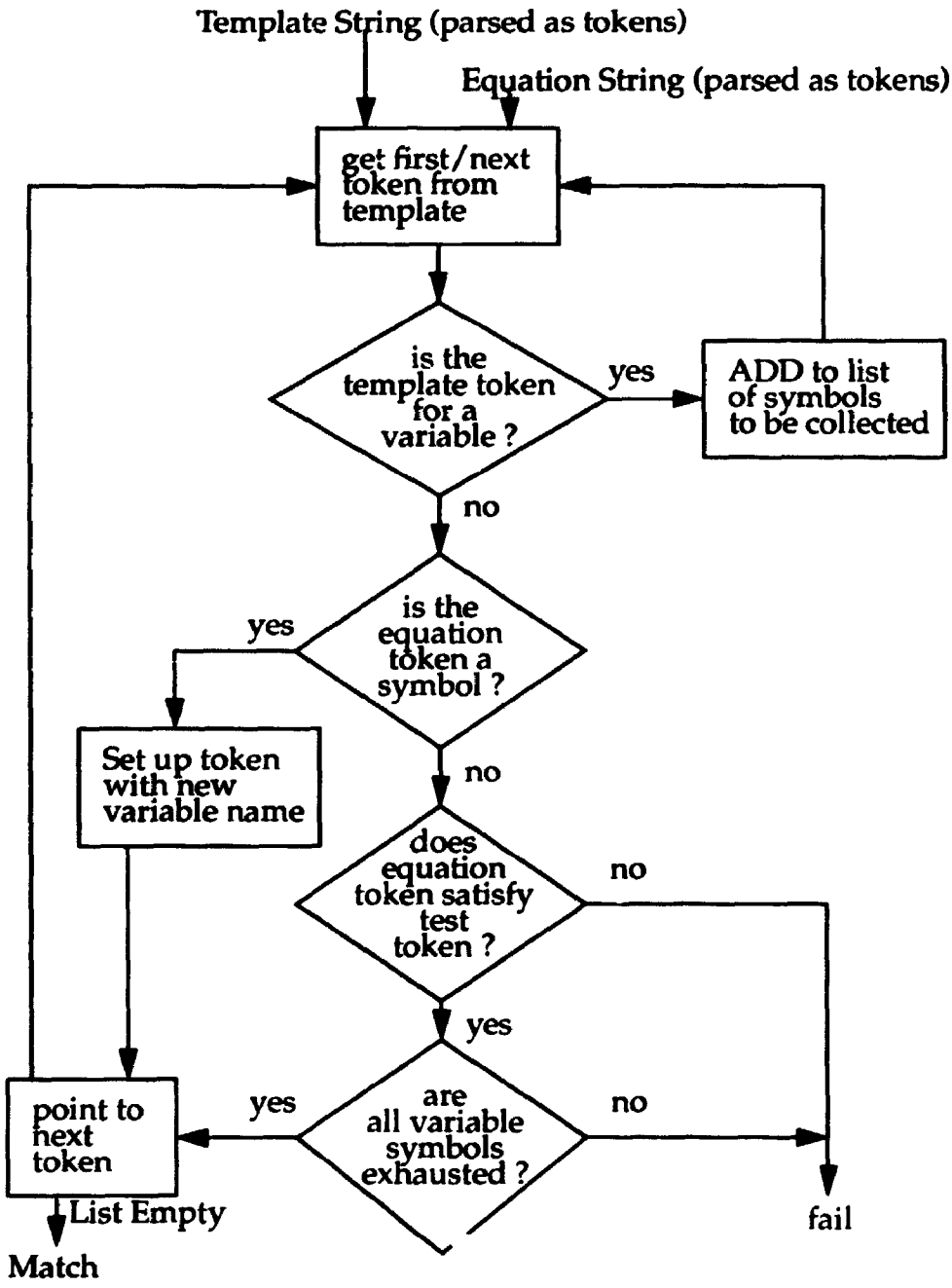


Figure 44 Flow Chart for Equation Template Matching

After the templates are matched, respective rules are tested for condition satisfaction, then actions are performed if successful.

5.3.4 Equation Conditions

It is natural to require some sort of conditions when rules are involved. The question is how to construct these to allow a variety of cases. The direction of this research initially was fairly simplistic, trying to just establish some true or false conditions. This was found to be limiting and as a result there have been a number of functions allowed within the condition matching implementation. These functions are still rather limited, because they do not include some fundamental constructs such as, if, loop, and while statements. Obviously these would be required in a complete system, but for proof of concept, the ramifications of not having these are logically redundant and repetitive condition definitions.

The reader should recall that in the rule file shown in Figure 43, the conditions had a few basic types of functions. These include,

COMPARE	This directs a comparison of the arguments. If the comparison is false, a false is returned.
MATH	This performs a math operation. Obviously this is false if the operation cannot be performed.
ASSIGN	This will create a variable, and assign a value. There are no cases at present that make this false.
FIND	This will do a complicated lookup from a database or do some geometrical calculations. This will also act as a catch-all for all complicated algorithmic calculations. This will be false if the required 'FIND' is not valid, or cannot be done.

These basic operators are some simple ones to start. Eventually these could be expanded to include more elaborate choices, such as loops, ifs, or other valid logical operators. The method for analyzing the condition set is to begin at the top of the set. If any of the conditions become false, then the condition is ruled false. How the false is interpreted is determined by the boolean expression in the rule.

Some details of interest should be identified in the comparison statement.

There are a number of basic operators that are intuitively clear, such as <=, >=, <, >. There are other operators that are very specific, for example = is for assignment, whereas == is for a logical equals, and '$' is for existence. There are other operators used in other arguments, but these are considered basic to any reader versed in any computer language.

5.3.5 Rules

These rules are basically production rules. If the conditions are satisfied, then the results are performed. When referring to previous examples the reader will note that a rule only contains two types of entries; 'EQUATION:' and 'RESULT'. The equation section describes the boolean expression that determines if a rule is valid or not. The form of the equation is identical to that given earlier for boolean designs, except that it is limited in operators to AND, OR, and NOT. The arguments of the boolean expression obviously refer to the conditions discussed in the previous section. If the rule passes, then the results indicated in the 'RESULT:' lines are executed. If there is more than one, they are applied in sequence from beginning to end. This rules structure is still fairly crude, but it is sufficient to produce a working system. Eventually these could also be augmented with some of the constructs of a modern computer language, to allow compact, and sophisticated result selection.

5.3.6 Results

When setting up the test software originally, the number of result operations was quite conservative. But, as the application grew, it became obvious that many operations were not only desired, but necessary.

When a rule is fired there are a number of expected effects. The first is a change in the 'state' of the part. This state change is performed by manipulating the Bool-

ean description of the product, and the sets that the representation refers to. Secondly, these changes must also be accompanied by the addition of steps to the process plan that will eventually be reflected in the final process plan. The manipulation of the equation was fairly straight forward, having only a few fundamental operations.

EQUATION_INSERT_SYMBOL
: This will insert a symbol (a set name) at a specified position in an active expression.

EQUATION_DELETE_SYMBOL
: This function will delete a set name that is at a specified location.

EQUATION_DELETE_TERM
: A term at a specified location will be deleted.

EQUATION_DELETE_VARIABLE_TERM
: This will delete a term that has been given a label using the 'LABEL' operator in the template matching string.

EQUATION_INSERT_TERM
: A given term will be inserted at the given position in the current expression.

At present, these few operators seem to perform all operations required for the current template matching methodology. Some notes about the equation manipulation routines are required. First, after a rule result manipulates an equation, the result is stored separately for that operation only. Second, when the result is fired, it refers to a particular term in an expression. This term becomes a sort of root term for reference. So when a rule refers to ':' for example, it is not referring to the main root of the equation, but to the first element in the matched equation. While this may seem somewhat confusing in words, observation of the examples should make the role perfectly clear.

If the equations are being manipulated, it is understandable that the sets the equations represent should also undergo change. This is done through the application of a number of other operators.

COPY_SET
: This simply copies one set to a new set.

ADD_SET
: This function creates a new set.

ADD_PROPERTY
: This function will add a property to a set.

PROPERTY_FUNCTION_VARIABLE
: Performs a mathematical operation based on set members.

PROPERTY_FUNCTION_NUMBER
: Performs a mathematical operation based on a set member, and a number.

DELETE_PROPERTY
: A specified property will be deleted.

The reader is reminded that these operators are driven by the template matching results in the previous sections. Thus there are references to variable set names, as well as new set names in these functions.

Other functions also exist for a variety of other tasks. These tend to be a bit more complex in requirements. In general these consist of math operations, operation cost reporting, process plan additions, and basic function lookups.

PLAN_PUSH_TEXT
: Adds a line of test to the process plan.

PLAN_PUSH_FORMAT
: Adds a line of text to the process plan, but using a formatted output. This allows the use of variables in plan output.

DECLARE_COST
: A numerical cost is declared, so that the system may perform some process selection based upon the cost of operations.

PROPERTY_FUNCTION_VARIABLE
: Perform a math function based upon two variable names.

PROPERTY_FUNCTION_NUMBER
: Perform a math function on a variable with a number.

FIND
: This function allows complicated operations to be performed, such as the lookup of the cost of a material, or a geometric property. Many complex functions can be added through this device.

All of the process plan declarations made are added to a new set. This set is

incorporated into the process plan structure using the cost declared as one of the selection functions. Therefore, when a rule result is fired it will declare as much information as possible about the operation that will occur. It can also calculated an estimated cost, and then let the planner know so that it may use the information during selection of the least expensive operation.

5.3.7 Other Considerations for the Rules

These rules are quite specific to the template matching scheme proposed here. More work is required to make this scheme more flexible. Subject to changes that arise from this additional work, it is possible these operators may change slightly or drastically to accommodate the changes.

During the implementation it was noticed that many conditions and other sets were redundant. This was mainly due to the 'inflexible' definition of conditions, such as one condition for checking for the existence of the first argument, and a second one for checking for the existence of the second argument. These are identical, except for the variable 'V0' as opposed to 'V1'. This could be easily handled by a variable list for condition calls, to allow one condition check that would easily interchange two arguments in the call stack. This idea may also be expanded to the definition of rules, and results as well.

5.3.8 Structure of the Process Plan

Traditional manufacturing methods tend to separate design and process planning. But, considering the link between design and process planning, it is logical to use a combined representation. In this case the process plan can be viewed as an addition of information to the original design. Most importantly this approach allows the original design to be retained intact, while allowing the process plan to refer to information directly. And, eventually this can be used so that changes to

the design are only propagated to the process plan steps that they effect.

A Bill of Material (BOM) list was used as the focus of the process plan. This naturally splits up the design into sub-assemblies and redundant features. So when looking for the design information we look for the 'main_product' set type, and when looking for process plans, we look for the 'MAIN_BOM' set. In this set there is a list of required parts, and pertinent information, as shown in the example below (This part is covered in detail in Chapter 7, and can be seen in section 7.4).

```
MAIN_BOM {
     ASSEMBLY ( 2.000000 Clip_half Clip_half_OP1 Clip_half_PART )
     ASSEMBLY ( 1.000000 0:Logo_Side 0:Logo_Side_OP7 0:Logo_Side_PART)
     ASSEMBLY ( 1.000000 Logo_Side Logo_Side_OP13 Logo_Side_PART )
     ASSEMBLY ( 1.000000 Spring Spring_OP16 Spring_PART )
     ASSEMBLY ( 1.000000 Big_Clothes_Pin Big_Clothes_Pin_OP19
      Big_Clothes_Pin_PART )
}
```

This particular entry contains a total of 5 different parts. At the top of the list are the parts that must be made first. 'ASSEMBLY' identifies the parts as members of an assembly, whereas in some cases it may say 'FEATURE' to identify the member as a reused feature. The first number in the arguments of these records indicate the quantity of parts required (a real number is used because it is possible to require only a fraction of something). The next item is the name of the part to be produced in the original design. The reader will notice that the second line claims that '0:Logo_Side' is the part to be produced. This notation is used when an assembly is indicated in the design, but it is part of a large equation, and therefore it does not have a distinct name. This particular part is from the part 'Logo_Side', and is given the equation location of ':0' in the equation,

```
( : ( & Clip_half ( - Screw_hole );position_screw_hole );add_logo )
```

And, therefore '0:Logo_Side' is,

```
( & Clip_half ( - Screw_hole );position_screw_hole );add_logo
```

This notation is not inherently clear, but is necessary to preserve the intent of the designer. The next argument in the list refers to the first operation in the operations list for the part, and the last argument points to a final part state set. For example 'Spring_OP16' points to the sets below.

```
Spring_OP16 {
     EQUATION: ( : Spring )
     OPERATION ( AND RULE 17:0:0 0.000000 Spring_OP17 spring_ends )
     ACTIVE: 0
     EXPANDED ( 0 2 3 2 )
}
Spring_OP17 {
     DESCRIPTION ( bend spring ends )
     DESCRIPTION ( L offset on spring = 0.0 )
     DESCRIPTION ( length of free end = 3.400000 )
     EQUATION: ( : BEND_SPRING_ENDS_18 )
     OPERATION ( AND RULE 18:0:0 0.000000 Spring_OP18 spring_roll )
     ACTIVE: 0
     EXPANDED ( 0 11 1 0 )
}
BEND_SPRING_ENDS_18 {
     form = SPRING
     inside_radius = 0.12
     outside_radius = 0.2
     length = 3.400000
     turns = 14.5
     rotate_y = 90
     offset = 0.0
     length = 0.0
}
Spring_OP18 {
     DESCRIPTION ( coil spring from wire )
     DESCRIPTION ( bend coil spring )
     DESCRIPTION ( wire dia. = 0.080000 )
     DESCRIPTION ( turns. = 14.500000 )
     EQUATION: NULL
}
COIL_SPRING_19 {
     wire_dia = 0.080000
     turns = 14.500000
}
Spring_PART {
     form = COMPLEX
     description = Spring
     equation = ( : Spring )
}
```

The reader will notice that there is an equation term in some sets. These equations represent the form of the product after the operation is performed. The possible operations are indicated in the operation fields. The field is easy to interpret. First, the 'AND' shown indicates all operations listed are to be done, an 'OR' would indicate only one has to be done. The next section is done in anticipation of future development. At present only rules is applied, in the future this would be expanded to include geometrical transformations. 'RULE' indicates that the last change was induced by a simple rule selection, and the next field '17:0:0' indicates the applied permutation. The author has noticed that if any search (and planning) methods are to be used, the exact firing conditions must be traceable. Three obvious methods for doing this are by rule number, also by simple geometry matching by equation form (with different variable names), or by matching by analyzing the part fully, and comparing geometry, and other properties. These tasks are without doubt lengthy, but seem to fit cleanly into the proposed framework. The next argument is the cost (estimated) of the operations in question. This slot is filled by the 'DECLARE_COST' operator discussed in the last section. These are then used when selecting operations based upon cost.

The end of the 'OPERATION' arguments is followed by a pair of arguments. The first one is the subsequent operation, in this case 'Spring_OP17' is one of these. Also included is the rule that called for this operation. By also identifying the rule, it can simplify feedback in the case of process plan failure. Although these lists only have one pair of operation/rule, an unlimited list is allowed, this will give the 'AND' and 'OR' operators some use when selecting alternate and parallel operations.

The reader should note that in 'Spring_OP17' there are some 'DESCRIPTION' fields. These are used to convey information to the operation planner. In this case

'DESCRIPTION' will simply cause the given text to be written directly on the process plan. There are other operators that will be discussed in the operation planning section. The important thing to notice is that all lines in this set are examined by the operation planner, but some are ignored, or not used, such as the 'EQUATION' description. The planner keeps track of where it stopped, the operation selected, and failed operations. The 'ACTIVE' field will point to the operation that is in use, based upon their order in the set (0 being the first). 'EXPANDED' keeps track of the exact state of the planner when it stopped last. This allow the planner to restart from the middle of a search. If for some reason the planner finds an operation that always leads to failure, it will indicate this by adding a 'FAIL' to the operation list. Although it is not pictured here, it can be seen in the results section.

As rules are applied and the product description is altered, there are new equations and sets created to accommodate this. One such set is 'BEND_SPRING_-ENDS_18'. For all effective purposes this is equivalent to a design set. And when attempting to backtrack, or examine product geometry at various stages of completion, this provides easy access to all relevant information.

There are some points of interest about this representation. First, the set and operation names are all augmented with numbers (from a global counter) to prevent confusion when set and operation names are reused as the result of rules that are used more than once. This structure is very conducive to replanning and viewing the status of work in process. For example, if a rule for drilling is fired twice, it will result in a duplicate set to keep the drilling operation information in. This obviously causes problems, but by adding a unique number to the name the two sets become different and distinct, while still maintaining a recognizable name.

Finally, a note about the representation in general. The structure described previously can be a bit confusing to deal with, but as the reader will have noticed in the discussion of rule results, and will see in the planning section, these details are buried in the system. The user is not required to deal with these aspects, and therefore the structure given is realistic, and should support planning requirements, without overwhelming the user with complicated data structures.

5.4 USING RULES TO FIND A PROCESS PLAN

When planning for a typical product it is common to consider the product in terms of components and reused features. (For a review of planning techniques in manufacturing refer to [Tsatsoulis and Kashyap, 1988]) Then each of these is separately considered for production methods, and finally details are planned. All of this can be illustrated as a hierarchy of tasks.

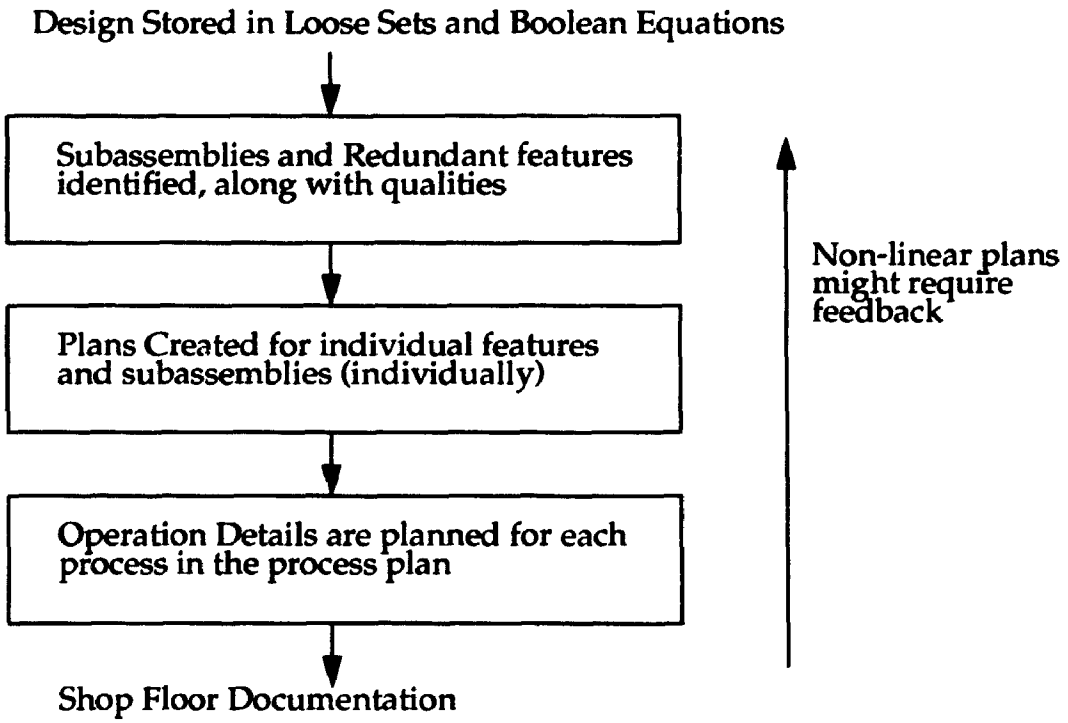


Figure 45 Hierarchical Decomposition of Planning Tasks

Dividing the tasks this way provides for an orderly division of macro tasks in

the planning system. Within each of these tasks there are further subdivisions. The non-linear nature of planning may mean that some planning decisions must be undone. This is in itself an extensive subject [ElMaraghy, 1992], but some attempts have been made to address this in this thesis. For example, if operation details cannot be calculated in Operation Planning, then the failure of the rule can be reported to Process Planning for Parts. If there are alternate operations not already considered, then Process Planning can continue planning, or undo operations until a suitable alternative is found. If none can be found, then feedback of problems to the previous stage is expected.

The following sections will explain details of each of the planning modules.

5.4.1 Preparation for Planning

The basic form of the design may give clues about the structure of a product. One of these examples is the use of the Assemble operator. When this is used, it is obvious that the parts in the list are sub-assemblies, or basic parts. Another feature of interest is the presence of redundant features. These features are recognized by the system when redundant part names are used in the design. There are a number of cases not recognized by the system at this point. For example, if two parts are exactly identical, but have different names, they are not recognized as similar. A similar case exists if two parts are identical, but have different CSG representations. Eventually this can be solved by the addition of some geometric reasoning software, and a Same Object Detection algorithm.

The basic procedure for finding the hierarchy of subassemblies and features can be described using a sequence of operations.

1. Beginning with the main product, start examining the equation, and the other sets (and their equations). As the entire product definition tree is

defined a list of terms, and subterms can be defined. A sample list is seen below, based upon the Big Clothes Pin in the examples chapter (Chapter 7). The derivation of this list is considered obvious, so in the interest of space the method is not included.

part name	priority	parent name	quant.	parent operation	child operation
Big_Clothes_Pin	0	NULL	1	ASSEM.	ASSEM.
Spring	1	Big_Clothes_Pin	1	ASSEM.	NULL
Logo_side	1	Big_Clothes_Pin	1	ASSEM.	AND
Clip_half	3	Big_Clothes_Pin, 0:Logo_side	2	ASSEM., AND	AND
0:Logo_side	2	Logo_Side	1	ASSEM.	AND
0:Clip_half	4	Clip_half	1	AND	NOT
1:Clip_half	4	Clip_half	1	AND	NOT
2:Clip_half	4	Clip_half	1	AND	AND
3:Clip_half	4	Clip_half	1	AND	NOT
1:0:Logo_Side	3	0:Logo_side	1	AND	NOT
B	5	0:Clip_half	1	NOT	NULL
C	5	1:Clip_half	1	NOT	NULL
A	5	2:Clip_half	1	AND	NULL
E	5	2:Clip_half	1	AND	NULL
D	5	3:Clip_half	1	NOT	NULL
Screw_hole	4	1:0:Logo_Side	1	NOT	OR
HOLES	5	Screw_hole	1	OR	NULL
CHAMFER	5	Screw_hole	1	OR	NULL

Figure 46 A Product Hierarchy

2. The list is reduced to terms that have assemblies as parents and/or have a quantity that is greater than 1. The result is a list of subassemblies, and features. A sample of this reduced list is shown below.

part name	priority	parent name	quant.	parent operation	child operation
Big_Clothes_Pin	0	NULL	1	ASSEM.	ASSEM.
Spring	1	Big_Clothes_Pin	1	ASSEM.	NULL
Logo_side	1	Big_Clothes_Pin	1	ASSEM.	AND
0:Logo_side	2	Logo_Side	1	ASSEM.	AND
Clip_half	3	Big_Clothes_Pin, 0:Logo_side	2	ASSEM., AND	AND

Figure 47 A Feature/Sub-Assembly Decomposition

3. The normal planning techniques now begin from the bottom of the list, and plan to the top.

As is described in the list above, a set of parts is identified with an algorithm. This step reduces the planning complexity, and identifies reused portions of a design.

5.4.2 Determination of Plans for an Individual Part

A simple planning method has been developed for finding process plans for individual parts. The rules were discussed earlier, but there are many approaches to applying them to the design. As described earlier, the method used here is a backtracking approach, using a hierarchical decomposition. The equation templates are matched, and then they call up production rules. Although this seems to be straight forward, there are a number of variations possible. The basic flow chart for the method for rule application is shown below (a higher level flow chart will be given later).

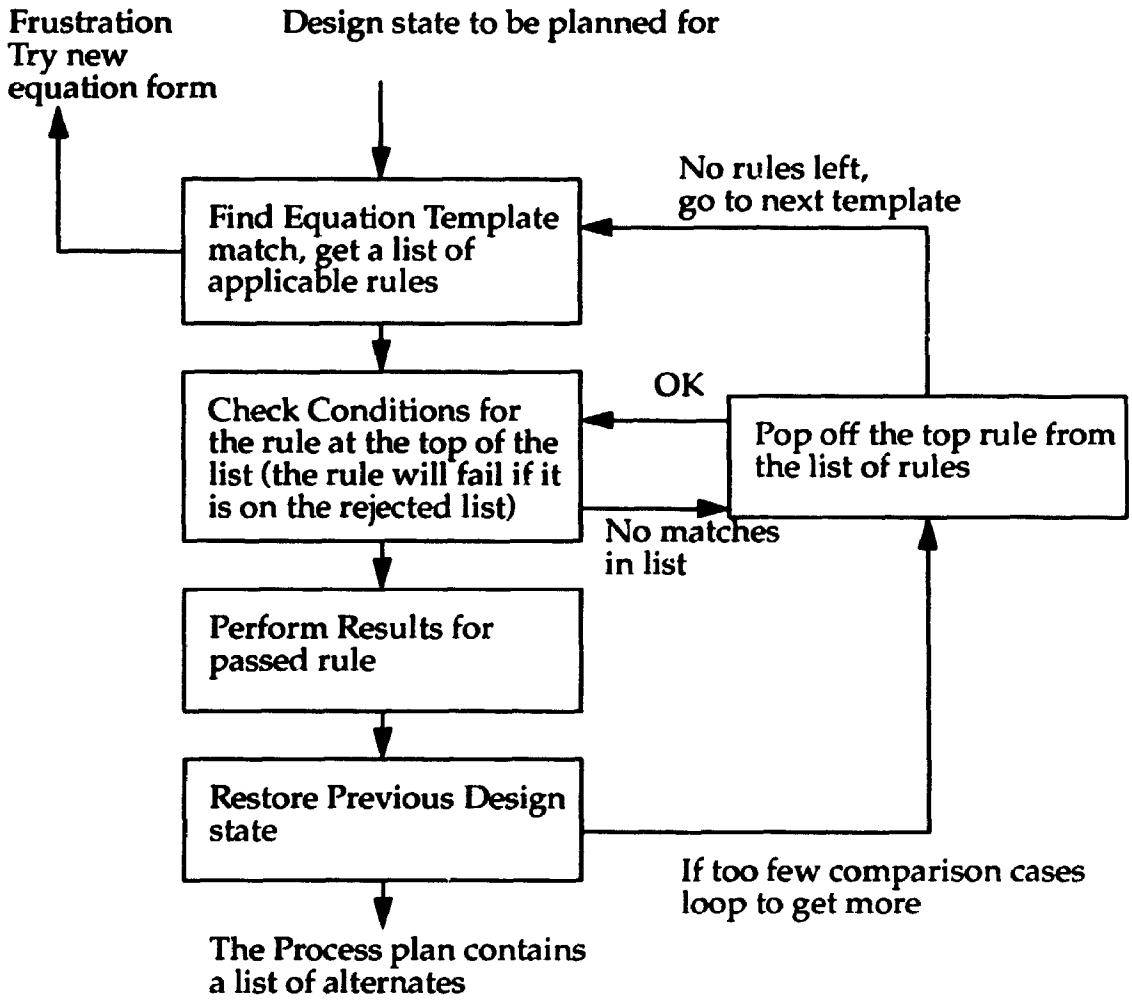


Figure 48 A Flow Chart Showing Low Level Rule Matching

This flow chart depicts the basic steps used for matching rules. In this case the routine starts with the equation for the part stored in equation form the equation refers to sets stored in a common area. Each of the templates are applied in turn until one matches. At this point the rules listed with the template are passed to the next stage. The list of rules are examined one at a time from top to bottom to check for condition matches. If a rule is fired it is applied, and the results are stored in other sets. After this the equation at the start of the routine is restored, and another rule is applied. If this fails a new template match is found. Eventually the

system will have found a minimum number of alternatives, or run out of choices, When this minimum number of choices is found the routine quits so that selection of an optimal cost operation can be selected at a higher level.

The results from the planning method described are only for one plan step. It is of use for the reader to notice that there is one feedback in the case that the equation form does not yield any success. In this case a higher level routine can perform operations such as equation manipulation. The flowchart below describes a higher level planning routine. This routine is intended to plan for a complete part.

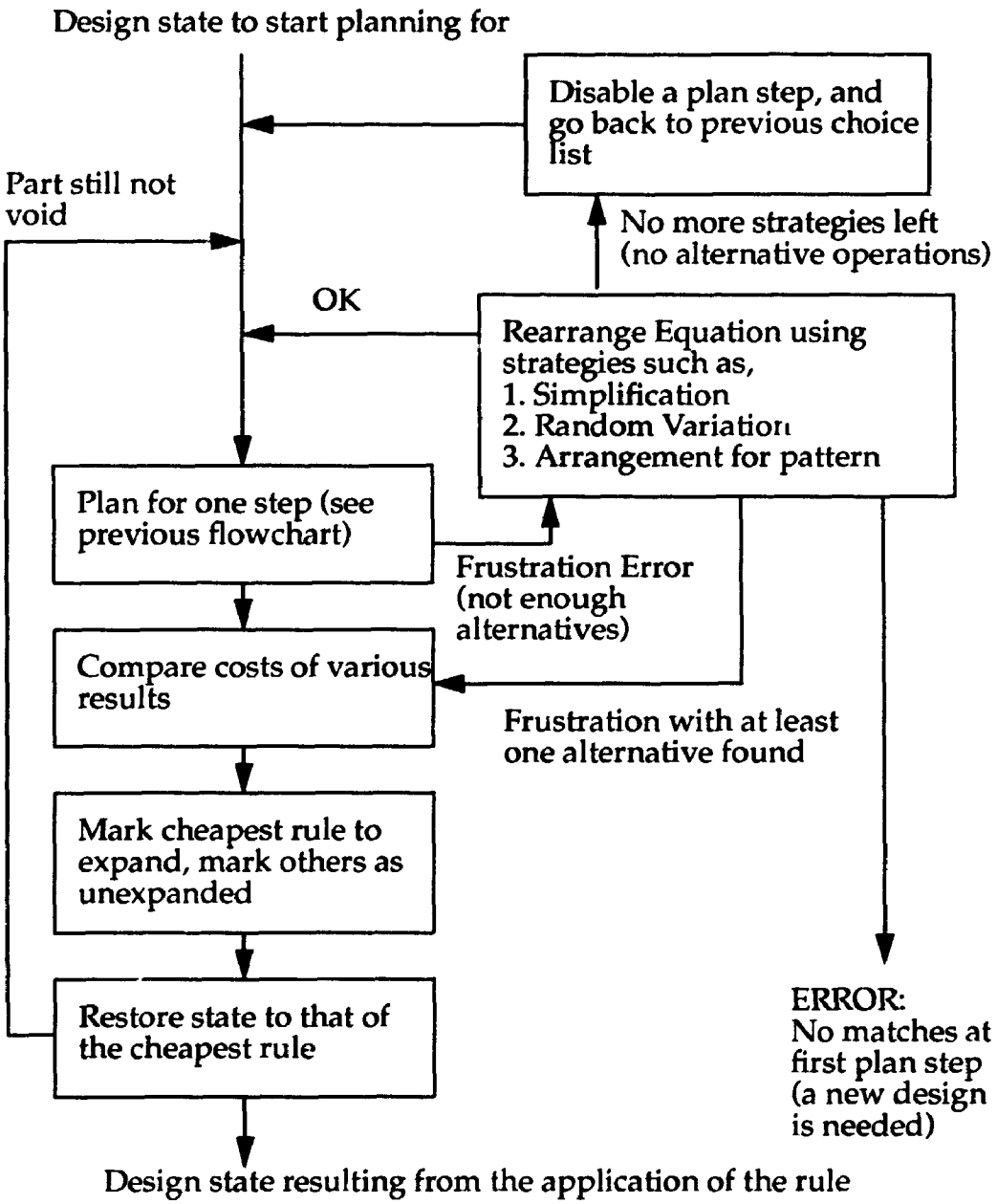


Figure 49 Macro Planning Level for Part

This routine will do some backtracking when a problem occurs, thus incorporating the non-linear planning techniques. The use of strategies to rearrange the equation also incorporates the concept of macro planning. The method of opera-

tion is that the routine will begin with the plan state to start planning for (This also allows replanning when outside failures occur). The first step involves planning for a single step. If enough alternatives are found, then the best alternative is found, and applied. Planning continues until the part is a void, indicating plan completion. In the event that too few alternatives are found for a single plan step the routine will attempt to work around it. First, the system will try to rearrange the equation. If this does not work, then the system will accept a limited number of alternatives. If no alternatives are found the system will backtrack to a previous operation, and rule out that alternative. This way, if a plan cannot continue, the planner will then look for viable alternatives. It is also possible that the planner cannot find a process plan. In this case there is simply an error report. This error report could be used in a system like that of Kimura [1989] that has design and CAPP tightly coupled for fast feedback of design features poorly designed for manufacture.

5.5 CONCLUSION

This chapter has discussed a number of manufacturing processes, and they were grouped into 8 general divisions. After that examples of templates were given for equation forms corresponding to various manufacturing processes.

After the discussion of the templates, the rules that they could activate were discussed. This discussion included conditions, rules, and results. Theses rules had a simple 'if-then' approach. Finally a planning strategy for the rules was presented. The strategy showed how the rule templates would first be used to prune the search tree. After the number of candidate rules was reduced they were examined one at a time until a number of alternative operations were found. Then the least expensive operation was selected. This continued until the process plan was

complete, or the planner was forced to backtrack and try a new solution.

A point of interest is that before the planner starts, the design equation is broken down into assemblies, and duplicate parts. At this point the Assembly operators become a valuable reference for separation into subassemblies for manufacturing.

In terms of planning capabilities, it is best to summarize them on an AI basis. First the planner is capable of dealing with non-linear plans where goals (part features) can only be produced in certain sequences. This is done with backtracking. At present the planner will only deal with one goal at once. Therefore there are some problems that it is not capable of solving. The planner is by definition Hierarchical. This is a result of the use of templates to prune the search tree before the rules are applied.

Some aspects of the planner are only discussed in brief, but deserve consideration, such as the use of Meta-goals in the system proposed by Nylund and Novak [1992] for turned parts. Meta-knowledge is also considered in Hummel and Brooks [1988].

6.OPERATION PLANNING

6.1 INTRODUCTION

The previous chapter described the fundamental components of the BCAPP system. It is also essential to describe what the BCAPP system will be providing process plans to. Basically once the process domain has been chosen (e.g. machining, assembly, etc.) there are many excellent commercial and research software packages that support this function of a CAPP system. In this section the conversion of the process plan to operation details, and preparation of documentation is reviewed. An interface to one commercial system for machining operation planning, called MetCAPP, will be described, as well as the requirements for an operation planner for injection molding.

6.2 CONVERTING PROCESS PLANS TO OPERATION SHEETS

In the previous chapter the planning system was described, including how the process plan is represented. In the simplest of approaches, this plan can be directly converted to lists of processes. We can enhance these encoded descriptions by using additional process, and operation planning software. A sample of part of a process plan is shown in Figure 50 for discussion.

The method for expanding the process plan information is quite straight forward. The documentation produced is in the form of Operation Sheets that include operation information. If referring to the example above, there will be a separate Operation Sheet generated for each of the line entries in the 'MAIN_BOM' set. This set is ordered such that the parts at the top of the list should be produced before the following parts. This does not mean that operations cannot be changed in order, but it does specify one of the possible manufacturing sequences. Each line in the 'MAIN_BOM' set specifies whether a part is an

'ASSEMBLY', or a 'FEATURE'. The first argument represents how many of the

```
MAIN_BOM {
     ASSEMBLY ( 2.000000 Clip_half Clip_half_OP1 Clip_half_PART )
     ASSEMBLY ( 1.000000 0:Logo_Side 0:Logo_Side_OP13
      0:Logo_Side_PART )
     ASSEMBLY ( 1.000000 Logo_Side Logo_Side_OP26 Logo_Side_PART )
     ASSEMBLY ( 1.000000 Spring Spring_OP33 Spring_PART )
     ASSEMBLY ( 1.000000 Big_Clothes_Pin Big_Clothes_Pin_OP42
      Big_Clothes_Pin_PART )
}
_GLOBAL {
     plan_count = 64
}
Clip_half_OP1 {
     EQUATION: ( & ( ~ B ) ( ~ C ) ( & A E ) ( ~ D ) )
     OPERATION ( AND RULE 2:0:0 507.851500 Clip_half_OP2 drill_hole )
     OPERATION ( AND RULE 3:0:0 0.000000 Clip_half_OP3 mill_wedge )
     OPERATION ( AND RULE 4:0:0 507.851500 Clip_half_OP4 drill_hole )
     ACTIVE: 1
     EXPANDED ( 1 14 0 0 )
}
Clip_half_OP2 {
     DESCRIPTION ( drill hole )
     CUTTING ( FEATURE single diameter hole )
     CUTTING ( MACHINE qa1000 )
     CUTTING ( MATERIAL polypropylene )
     CUTTING ( PARAMETERS 0.072500 0.5 1.0 )
     EQUATION: ( & ( ~ C ) ( & A E ) ( ~ D ) )
}
DRILL_HOLE_3 {
     a1 = 0.5
     a2 = 1.0
     height = 0.072500
     cost = 507.851500
}
Clip_half_OP3 {
     DESCRIPTION ( mill out block shape )
     CUTTING ( FEATURE Flat Rectangular Surface - Open,
      Cutter Axis Perpend. )
     CUTTING ( MACHINE qa1000 )
     CUTTING ( MATERIAL polypropylene )
     CUTTING ( PARAMETERS 10 10 10 )
     EQUATION: ( & ( ~ B ) ( ~ C ) A ( ~ D ) )
     OPERATION ( AND RULE 5:0:0 507.851500 Clip_half_OP5 drill_hole )
     OPERATION ( AND RULE 6:0:0 507.851500 Clip_half_OP6 drill_hole )
     ACTIVE: 0
     EXPANDED ( 1 14 0 0 )
}
```

Figure 50 Output from Process Planning for Operation Planning

parts are required in the design. The second argument is the name of the part in the original design. The fourth argument is a reference to a set that contains a single geometry expression for display purposes. And, finally the third field points to the head of a tree of operation sets.

If we follow the plan direction for the 'Clip_half_OP1' we find a set that has some 'OPERATION' information, an 'EQUATION', and an 'ACTIVE' pointer that says operation '1' is active, therefore the second 'OPERATION' is selected (0 would be the first 'OPERATION'). The 'EXPANDED' operator is only used by the planner for restarting planning that has been stopped by keeping track of where the planner stopped. The first operation pointed to from the 'MAIN_BOM' (in this case 'Clip_half_OP1') is a special case in that it contains no plan operation information, but points to a number of drastically different plans.

When deciding which operation comes next we must follow the trail of the 'ACTIVE' 'OPERATIONS'. At this point in the discussion the second 'OPERATION' is active, and it points to 'Clip_half_OP2'. Looking at this operation reveals that there is much more information than in the previous operation. The reader should note that the 'EQUATION' in this operation is now changed from the previous operation. At this point operation planning would be performed, and there would be details added to the operation lists. The operator 'DESCRIPTION' will cause the arguments to be directly echoed to the operation lists. On the other hand 'CUTTING' will result in calls to MetCAPP, which will eventually suggest tools, cutting sequences, speeds, feeds, and other cutting information. The interpretation of these will be discussed in more detail later. After all of the planning fields are completed, then the next 'ACTIVE' 'OPERATION' will be selected, and planning will continue until no 'ACTIVE' pointer is found. This should also coincide with an 'EQUATION: NULL' if planning was completed.

The plan steps are actually reversed in the sequence described above. This is a result of the backwards planning approach. Therefore the plan steps are reversed when printed on the plan. There are also operation numbers assigned to each operation step. This is done to maintain consistency with the typical industrial practice. A flowchart of the operation planning section is shown below.

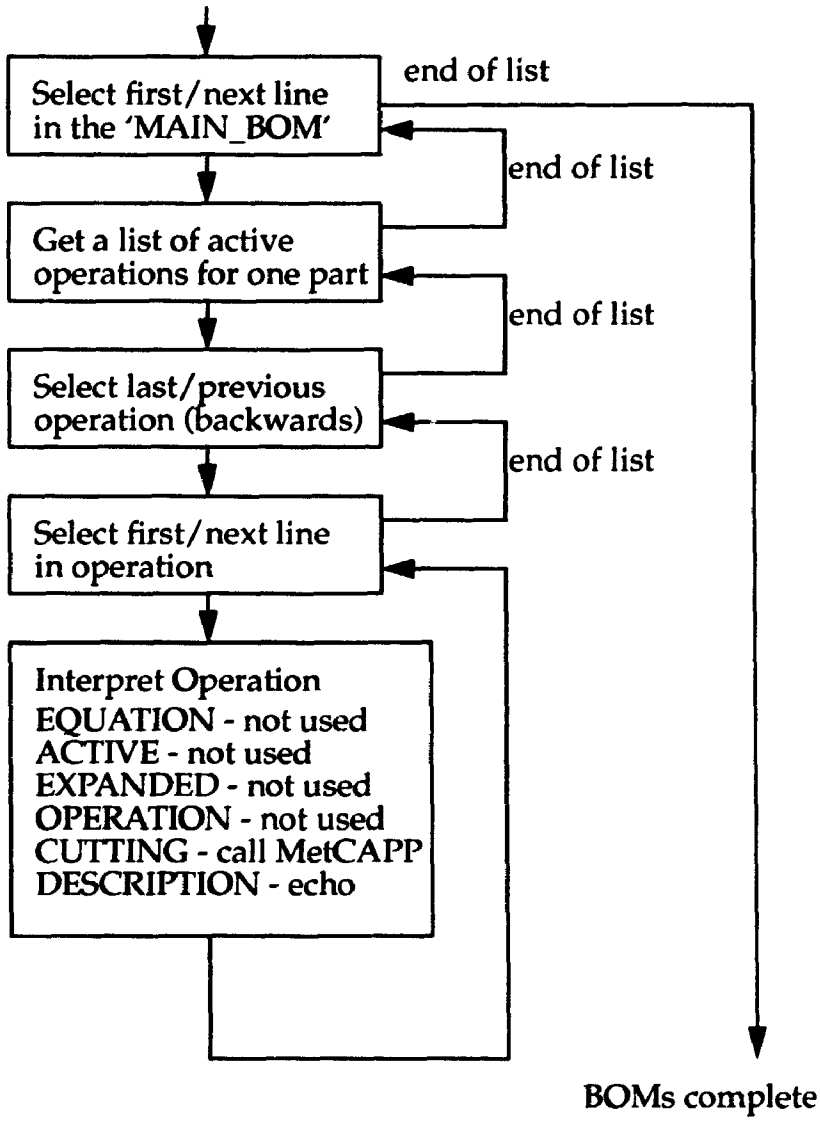


Figure 51 A Flowchart for Process Plan to Operation Sheets

As can be seen the process of developing operation sheets is just a case of looping iteratively. The place for other CAPP technologies and software is in the 'Interpret Operation' block. As is described, MetCAPP is used here, as other CAPP systems could be as well.

Also of interest is a sample of the operation sheet produced from the lines above. A sample is given below.

```
OPERATION SUMMARY_SHEET: Clip_half_PART - Quantity 2.000000
 OP# Operation Description
----------------------------------------
 0    cut a block from stock with hot wire
      width = 1.4
      depth = 5.9
      height = 0.55
 10   drill hole
 20   drill hole
 30   drill hole
 40   mill out block shape
```

Here we see that the list ends with 'mill out block shape'. This was the first operation in the list, and therefore it is the last in practice. Each of the operations has a separate number to identify it as one process step. The reader will notice the 'DESCRIPTION' in the initial operation description echoed its text to the BOM. The reader should also note that a total of two parts are specified at the top of this BOM. When the entire plan is done, there will be a list of similar sheets for all parts of the product.

6.3 RECOVERING FROM OPERATION FAILURE

When an error occurs in the execution of a plan, it is necessary to request a replanning. At this point the method is very simple and straight forward. It involves indicating which operation has failed, and restarting the planning algorithm of the previous section. The assignment of operation numbers to the plan the planner uses requires that a lookup table be used for converting operation

numbers to operation names, so that the planner would know which operation failed. It is also of use to point out that in this case backward planning causes some problems when it tries to replan. This is based on the fact that when it is selecting rules to 'undo' features the planner has started from the complete part, backtracking would suggest backtracking toward the completed part (in the opposite direction from the direction needed for replanning). For example, we might say a drilling operation has failed half way through production. The planner would note that the operation has failed, and select another replacement operation, and another feature is undone, and so on down another path in the operation tree until all features are undone. In effect we are finding another path to the start point. But, what we need for practical purposes is the exact opposite, a path to the start state (the final product).

6.4 OPERATION PLANNING WITH METCAPP

MetCAPP [Institute for Advances Manufacturing Science, 1991] [Zdeblick, 1985] [Barocky and Zdeblick, 1984] is the result of many years of development of machining data. Initially the work began as a handbook containing many machining recommendations. Eventually it evolved to become a computerized database containing the information (CutTech). After that it was enhanced with a cutting sequence planner (CutPlan), and eventually a front end was added to allow specification of machining features (MetCAPP). This software includes an interface library to allow it to be used by other computer programs. In this thesis, CutTech was provided with some manufacturing feature information, as well as material, and a machine. From this the software was able to suggest multistep machining operations to produce machined features. This software is used to emphasize the fact that other CAPP systems play an important role in complementing BCAPP.

There are a number of functions that the operation planner recognizes for MetCAPP. These are,

CUTTING (MACHINE string) - defines a machine with the name 'string'.
CUTTING (MATERIAL string) - defines a material with the name 'string'.
CUTTING (FEATURE string) - defines a feature with the name 'string'.
CUTTING (PARAMETERS x0...xn) - defines a variable list of parameters'.

These are the minimum definition requirements for CutTech to make a recommendation. CutTech contains a list of common machines, and materials found in modern machining, and it will look up the properties of these. There are also a limited number of features (an extensive list may be found in the MetCAPP manuals). The parameter list is required so that required information about the features can be provided. This list is variable in length, because the required parameter list varies in size for each feature. The values for the parameter list should also be found through reference to the MetCAPP manuals. An example of a MetCAPP operation plan is given below for a hole.

```
Seq.#      Machine                          Passes       Time  Operation Description

SUB_OP 5   Center Drill Non-Insertable          1   0.008000  Center Drill Non-Insertabl

 I  !  Center Drill  !  0.7500  !  No  !  3.250  !  M10
 1  !  DLS-007  !  0.7500  !  No  !  0 750  !  M7
 2  !  DLS-008  !  0.7500  !  No  !  0.750  !  M7
 3  !  DLS-009  !  1.0000  !  No  !  1.000  !  M7
 4  !  DLS-010  !  1.0000  !  No  !  1 000  !  M7
 5  !  DLS-005  !  0.5000  !  No  !  0.500  !  M7

SUB_OP 10  Drill Non-Insertable                1   0.031000  Drill Non-Insertable

 I  !  Non Insert Drill  !  0.4844  !  No  !  1.484  !  M10
 1  !  DLS-118  !  0.4688  !  No  !  4.313  !  M7
 2  !  DLS-119  !  0.4844  !  No  !  4.375  !  M7
 3  !  DLS-279  !  0.4688  !  No  !  4.313  !  M42
 4  !  DLS-280  !  0.4844  !  No  !  4.375  !  M42

SUB_OP 15  Bore Non-Insertable                 1   0.017000  Bore Non-Insertable

 I  !  Non Insert Bore  !  0.500  !  15.00  !  0.015  !  C2
 1  !  BRS-004  !  0.500  !  0.00  !  0.008  !  C7
 2  !  BRS-031  !  0.406  !  0.00  !  0.008  !  C7
```

Figure 52 A CutTech Operation Plan for a Drilled Hole

This plan has three steps for rough drilling, then fine boring. Each operation is also followed by a list of tools. The first tool in the list is the first choice. The other tools are acceptable alternates. In the list given the first of three operations (#5) is 'Center Drill Non-Insertable'. The operation will be done in one pass, in .008 hours. The tool parameters are listed below, with the tool at the top specifying the best tool choice.

6.5 CONCLUSION

This section has outlined the general process of operation planning used in this thesis. The discussion includes converting process plans to operation plans, and representing those operation plans in Operation Sheets. This section illustrates the role of other CAPP systems, and demonstrates the addition of other CAPP systems in the operation planning stage. Some of the techniques and limitations of replanning for failed operations were discussed.

The output from MetCAPP is stored in separate files on disk, and can be printed when the job is sent to production. The details are not shown with the normal process plans because of the extra bulk they would add.

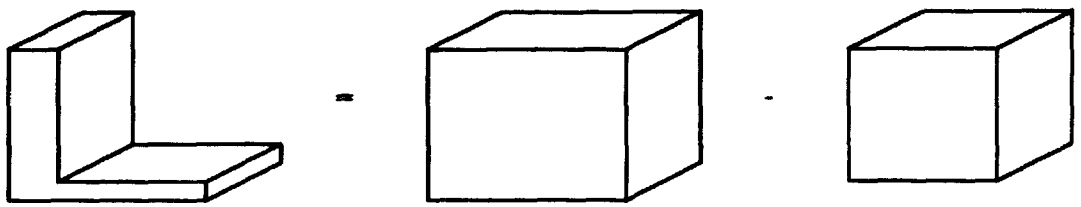
7.TEST CASES

7.1 INTRODUCTION

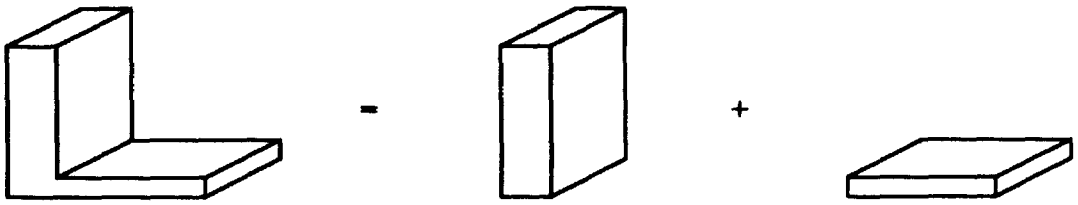
Previous chapters have outlined the functional aspects of BCAPP. At this point it is the intention of the author to demonstrate the validity of the Boolean approach to planning, as well as discussing some outstanding problems. A few examples have been chosen to illustrate the method in general. The first examples will be simple machined parts. These are selected as a simple introduction to the results. After these a plan will be presented for a desk toy, in the form of a scaled up 'clothes pin'. This is customized with an advertisement, and can be used to hold a stack of paper. Finally, a novel application will be displayed by using the planner to select electrical components for a digital circuit.

7.2 A SIMPLE MACHINED PART

Machined parts come from a very limited domain. When machining, most parts involve selecting a piece of stock, and then removing metal to get to a finished part shape. Generally there will be two ways a designer would approach this problem. The first would be when he selects a basic piece of stock, then subtracts volumes until the final part shape is reached. At present this approach is easily handled by the system. On the other hand, if the designer decides to build up a part from primitives, then a piece of stock must be identified for machining. Since stock identification requires geometrical reasoning, and this has not been added to the method yet, it is not possible to deal with this case for production. It has been stated in the past [Requicha and VandenBrande, 1988] that the alternate representations are a problem. However, if we use the knowledge about what the alternate representations are, and what they mean, then it is possible to convert between them. There are two equivalent cases shown below in Figure 53.



A Construction by Stock Removal (Machining)



A Construction by Part Joining (Welding)

Figure 53 Alternate Representations for the Same Part

If a designer has chosen one representation or another it would suggest a particular process. In the top representation, the designer would obviously be suggesting machining. In the bottom, the designer would be suggesting some sort of joining process. At present BCAPP can plan for both of these processes, but until geometrical reasoning is added to allow transformation between the two representations, it will be difficult for BCAPP to suggest welding for the top part representations, and likewise machining for the bottom part.

One of the first parts used for development is seen below. It is a block with an angles face cut off and a through hole drilled.

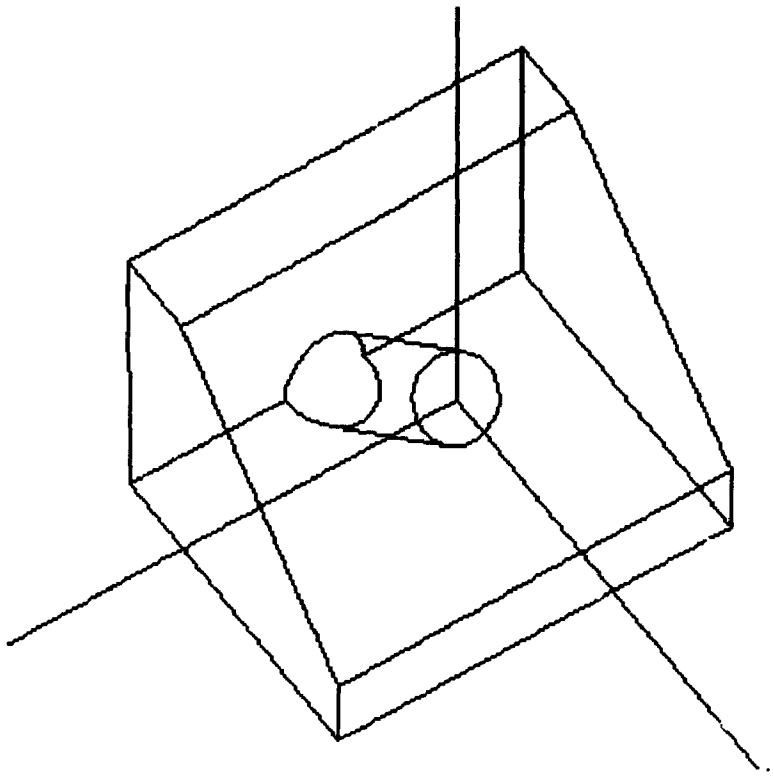


Figure 54 An Angled Part for Demonstration

The number of possible machining operations that can be used for this part is limited. In fact the number of reasonable CSG models for such a part are also limited. The design file for this part is given below.

```
//
// An angles part for test purposes
//
main Block_with_Hole {
 EQUATION: ( & BLOCK ( - HOLE ) WEDGE )
}
WEDGE {
 form = WEDGE
 width = 2
 depth = 2
 height = 2
 name = cut_out_wedge
}
HOLE {
 form = CYLINDER
 radius = 0.2
 height = 4
```

```
 rotate_x = -45
}
BLOCK {
 form = BLOCK
 width = 2
 depth = 2
 height = 2
 material = 1050steel
 translate_y = -.4
 translate_z = -.4
}
```

The part file only uses a single design equation to represent the design. As can be seen, I have constructed the basic form of the part with block and wedge primitives. While the basic form can be created a number of ways, the hole through the block can only rationally be made with a '(~ CYLINDER)' term, any other would be excessive, or redundant. When it comes to the formation of the block/wedge pair they exemplify the non-unique nature of CSG. This could be made by,

- cutting a square block, and cutting off the parts where it doesn't overlap the wedge,
  (& BLOCK WEDGE)
- cutting a wedge shape, and removing the sections where it doesn't overlap the block shape,
  (& WEDGE BLOCK)
- cutting a square block, and removing the material on one half of a half-space plane,
  (& BLOCK HALFSPACE)
- etc.

As mentioned before, there are two clear manipulations at work here. The first is simple equation manipulation, as in the commutation of the 'WEDGE' and 'BLOCK' above. In the second case, there is the selection of new equations and primitives, as in replacing the 'WEDGE' with the 'HALFSPACE'. As mentioned before, only equation manipulation is used here. Therefore the discussion will centre about the first two cases listed above.

Both of the cases above can be recognized with rules, and both are quite valid.

But, it is not desirable to cut stock with a triangular cross section, as would be required in the first case. Recognizing this there are only a few rules for cutting stock in standard shapes, such as square and round. In addition to rules for recognizing stock, there are also rules for looking for certain shapes to cut externally, such as the wedge. It is known that this shape is external because it is ANDed with the stock. Whereas the cylinder is probably an internal feature because is in the form '(AND ..... (NOT CLYINDER) .... '. In simple terms the basic planning process will follow the steps below.

1. a) Setup equation '(& BLOCK (~ CYLINDER) WEDGE)'
2. a) Recognize the CYLINDER as a drill operation
   b) Add a drill operation to the process plan
   c) Alter the equation to remove the hole '(& BLOCK WEDGE)'
3. a) Recognize the WEDGE as an external milling operation
   b) Add a milling operation to the process plan
   c) Alter the equation to remove the angled face '(& BLOCK)'
4. a) Recognize the lone primitive as a stock selection
   b) Add a call for stock to the process plan
   c) Delete the last term in the equation to leave 'NULL'
5. a) Quit, because the 'NULL' equation means planning is done

This plan is quite straight forward, and does get planned without problems. If the wedge and the block were reversed the plan would continue until it was time to get a triangular stock. No rule would be found for getting triangular stock, and the system would have to backtrack until it finds an alternative plan. Also of interest is the naive suggestions of machining volumes. When suggesting milling parts of the block outside the wedge shape, some geometrical calculations are required to determine the volumes to be milled. Geometrical information is not available, but if it was a CSG model of the volume as '(& BLOCK (~ WEDGE)' would be the machined volume.

The plan developed for this part is shown below.

```
-------- Work Order Sheets ------------
Product: Block_with_Hole


OPERATION SUMMARY_SHEET: Block_with_Hole_PART - Quantity 1.000000
 OP# Operation Description
-----------------------------------------
 0    cut a block from stock with band saw
      width = 2
      depth = 2
      height = 2
 10   Mill Off Wedge Shape
 20   drill hole : HOLE
```

Figure 55 Operation Sheet for Angled Part

This plan is the direct output from the operation planner, with detailed machining information not given. At the top we see that the Product name is identified. Second we see an operation summary sheet, that identifies the part, the quantity to be manufactured, and a number of operations. The first operation requires that stock be cut. Secondly, a milling operation is called for to eliminate the excess wedge. Finally, the hole is -illed.

To be critical, there are a number of data details that would eventually have to be added to the rules to make them complete, such as a knowledge of available stocks sizes and shapes. Also important is that the call for milling naively calls for milling of a wedge shape. At its worst this would work, but it would be very inefficient. However, computer graphics tools could be used to show a picture of the part before and after for the benefit of human users. The final call for a drilled hole is fairly straight forward, and should work well. Detailed locations, etc. are not given here for most operations. This is for clarity, but the indication of block dimensions shows that the system is indeed capable of reflecting the detailed information. Another detail to point out is that it is possible that the drilling oper-

ation would be listed before the milling operation. This is not acceptable because both of the entrance faces for the hole would not be at a normal to the axis of drill insertion (i.e. the drill would slip). This sort of information requires some geometrical interpretation to allow rule conditions to be checked. This does not mean that future work cannot incorporate this feature, but it does mean in the present state some geometrically absurd operations will be suggested.

7.3 A MORE COMPLEX MACHINED PART

The previous section showed a very simple machined part that was quite simple to plan. Also, the rules for planning were overlooked. In this section a more complex part will be suggested, and the complete rule set will be presented.

The part shown below is a rod support mount that was randomly selected from a drafting textbook. It contains many similar operations in the previous example, but is more extensive.

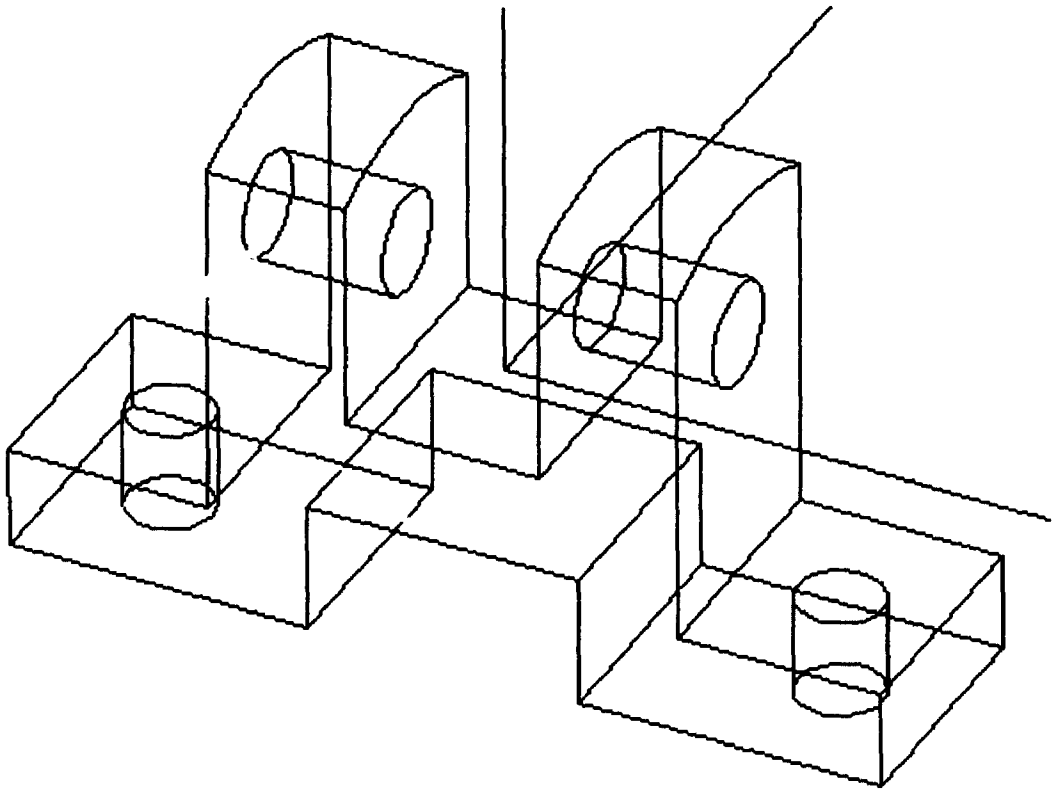


Figure 56 A Rod Support Mount

```
//
// A sample mechanical part to be machined.
//
main Rod_Support {
     EQUATION: ( & Stock ( - Rod_Hole ) ( ~ Mount_Feature;Mount_Hole_1
      ) ( ~ Mount_Feature;Mount_Hole_2 ) ( ~ Top_Channel )
      ( ~ Bottom_Channel ) Top_Round )
}
Stock {
     material = brass
     form = BLOCK
     width = 11.3
     depth = 3.2
     height = 6.4
}
```

Figure 57 A File to Describe the Rod Support Mount

```
Rod_Hole {
      form = CYLINDER
      radius = .65
      height = 11.3
      rotate_y = -90
      translate_x = 1.4
}
Mount_Feature {
      EQUATION: ( + Mount_Hole Mount_Gouge )
}
Mount_Gouge {
      form = BLOCK
      width = 2.6
      depth = 3.4
      height = 6.4
      translate_z = 1.3
}
Mount_Hole {
      form = CYLINDER
      radius = .55
      height = 6.5
}
Mount_Hole_1 {
      translate_x = -4.35
}
Mount_Hole_2 {
      translate_x = 4.35
}
Top_Channel {
      form = BLOCK
      width = 2.5
      depth = 3.4
      height = 4.0
      translate_z = 1.8
}
Bottom_Channel {
      form = BLOCK
      width = 3.5
      depth = 3.4
      height = 1.8
      translate_z = -2.4
}
Top_Round {
      form = CYLINDER
      radius = 3.4
      height = 12.0
      translate_x = 0.2
      rotate_y = 90
}
```

Figure 57 A File to Describe the Rod Support Mount (cont'd)

The file above is more sophisticated than the previous design file, and should

give the reader an appreciation of the power of this representation technique. For example there is a reused feature in the design called the 'Mount_Feature'. This feature is used twice in the design.

The rule file below will only be shown once because of size, although the same file is used for the first three examples in this chapter (i.e. the angles part, the rod support mount, and in the next section the big clothes pin).

```
//
// This set of equation forms sets up search patterns in the equation,
// and suggests rules which can be fired by a match
equation_form EQN_A {
     EQUATION: ( ? VAR:V:0;PROP ...)
     RULE: translate_part
     RULE: add_logo_to_part
}
equation_form EQN_C {
     EQUATION: ( ? VAR:V:1;PROP ...)
     RULE: translate_second_part
}
equation_form EQN_D {
     EQUATION: ( ? VAR:V:1;PROP ...)
     RULE: translate_second_part
}
equation_form EQN_0 {
     EQUATION: ( & ...( - VAR:V:0 ):VAR:PEEP
     RULE: before_drill_hole
}
equation_form EQN_1 {
     EQUATION: ( & ...( - VAR:V:0 ):LABEL:REF
     RULE: drill_hole
     RULE: chamfer_drill
     RULE: mill_out_block
}
equation_form EQN_2 {
     EQUATION: ( & VAR:V:1 )
     RULE: cut_angled_chunk1
     RULE: cut_angled_chunk2
}
equation_form EQN_3 {
     EQUATION: ( : ...( & VAR:V:0 ):LABEL:REF
     RULE: get_stock
     RULE: cut_plastic_stock
}
```

Figure 58 The General Rule File Developed for Discrete Manufacturing

```
equation_form EQN_4 {
     EQUATION: ( : ( & VAR:V:1 ):LABEL:REF
     RULE: cut_bar_stock
     RULE: cut_plastic_stock
}
equation_form EQN_5 {
     EQUATION: ( : ( & VAR:V:i ) )
// RULE: cut_bar_stock
}
equation_form EQN_6 {
     EQUATION: ( : VAR:V:0 )
     RULE: spring_roll
     RULE: spring_ends
}
equation_form EQN_7 {
     EQUATION: ( & VAR:V:0 VAR:V:1 ...)
     RULE: mill_off_wedge
     RULE: mill_off_round
     RULE· mill_off_block
}
equation_form EQN_8 {
     EQUATION: (> : VAR:V:0 )
     RULE: fixture_part
}
equation_form EQN_9 {
     EQUATION: (> : VAR:V:0 VAR:V:1 ...)
     RULE: assemble_parts
}
equation_form EQN_10 {
     EQUATION: (> : VAR:V:0;PROP VAR:V:1 ...)
     RULE: assemble_parts
}
equation_form EQN_11 {
     EQUATION: (> ? VAR:V:0 )
     RULE: get_cut_block_stock
     RULE: get_cut_plastic_stock
     RULE: get_cut_round_stock
     RULE: get_WIP
}
equation_form EQN_12 {
// Assume if here there are no other options, so squash in terms
     EQUATION: ( ? VAR:V:0;PROP ...)
     RULE: append_properties
}
equation_form EQN_13 {
     EQUATION: ( ? VAR:V:1;PROP ...)
     RULE: append_properties_2
}
```

Figure 58 The General Rule File Developed for Discrete Manufacturing (cont'd)

```
// This section lists the possible rules which may be fired. Each rule
// references a set of conditions which are in the following section
rule mill_off_round {
     EQUATION: ( & ROUND_SHAPE_2 )
     RESULT: MILL_OFF_ROUND
}
rule mill_off_wedge {
     EQUATION: ( & WEDGE_SHAPE_2 )
     RESULT: MILL_OFF_WEDGE
}
rule mill_out_block {
     EQUATION: ( & BLOCK_SHAPE )
     RESULT: MILL_OUT_BLOCK
}
rule translate_part {
     EQUATION: ( + TRANSPORT_X TRANSPORT_Y TRANSPORT_Z )
     RESULT: TRANSPORT
}
rule translate_second_part {
     EQUATION: ( + TRANSPORT2_X TRANSPORT2_Y TRANSPORT2_Z )
     RESULT: TRANSPORT2
}
rule add_logo_to_part {
     EQUATION: ( & IS_LOGO )
     RESULT: ADD_LOGO
}
rule fixture_part {
     EQUATION: ( & PART_0_EXISTS )
     RESULT: FIXTURE_PART
}
rule assemble_parts {
     EQUATION: ( & PART_0_EXISTS PART_1_EXISTS )
     RESULT: ASSEMBLE_PARTS
}
rule spring_roll {
     EQUATION: ( & SPRING_SHAPE ( ~ SPRING_ENDS ) )
     RESULT: MAKE_SPRING
}
rule spring_ends {
     EQUATION: ( & SPRING_SHAPE L_SPRING_ENDS )
     RESULT: SPRING_BEND_ENDS_L
}
rule before_drill_hole {
     EQUATION: ( & CYLINDER_SHAPE DRILL_SIZE )
     RESULT: ADD_PROPS_TO_PRIMITIVE
}
rule drill_hole {
     EQUATION: ( & CYLINDER_SHAPE DRILL_SIZE )
     RESULT: DRILL_HOLE
}
```

Figure 58 The General Rule File Developed for Discrete Manufacturing (cont'd)

```
rule chamfer_drill {
     EQUATION: ( & CONE_SHAPE DRILL_SIZE )
     RESULT: DRILL_CHAMFER
}
rule cut_angled_chunk1 {
     EQUATION: ( & WEDGE_SHAPE )
     RESULT: CUT_ANGLED_SURFACE
}
rule cut_angled_chunk2 {
     EQUATION: ( & WEDGE_SHAPE )
     RESULT: CUT_ANGLED_SURFACE2
}
rule get_stock {
     EQUATION: ( & BLOCK_SHAPE METAL )
     RESULT: GET_BLOCK_STOCK
}
rule cut_bar_stock {
     EQUATION: ( & BLOCK_SHAPE BLOCK_STOCK METAL )
     RESULT: CUT_BLOCK_STOCK
}
rule cut_plastic_stock {
     EQUATION: ( & BLOCK_SHAPE BLOCK_STOCK THERMO_PLASTIC )
     RESULT: CUT_BLOCK_STOCK_PLASTI
}
rule mill_off_block {
     EQUATION: ( & BLOCK_AND_BLOCK )
     RESULT: MILL_BLOCK_SHAPE
}
rule get_cut_block_stock {
     EQUATION: ( & BLOCK_SHAPE METAL )
     RESULT: CUT_BLOCK_STOCK
}
rule get_cut_plastic_stock {
     EQUATION: ( & BLOCK_SHAPE THERMO_PLASTIC )
     RESULT: CUT_BLOCK_STOCK_PLASTIC
}
rule get_cut_round_stock {
     EQUATION: ( & CYLINDER_SHAPE METAL )
     RESULT: CUT_ROUND_STOCK
}
rule get_WIP {
     EQUATION: ( & COMPLEX_SHAPE )
     RESULT: GET_WIP_STOCK
}
rule append_properties {
     EQUATION: ( ~ SHAPE )
     RESULT: APPEND_PROPERTIES_TO_SET
}
```

Figure 58 The General Rule File Developed for Discrete Manufacturing (cont'd)

```
rule append_properties_2 {
     EQUATION: ( - SHAPE2 )
     RESULT: APPEND_PPOPERTIES_TO_SET_2
}
//
// Basic conditions for reasoning in the plan
condition SHAPE {
     COMPARE ( PROP0.form $ )
}
condition SHAPE2 {
     COMPARE ( PROP1.form $ )
}
condition TRANSPORT_X {
     COMPARE ( PROP0.translate_x $ )
     COMPARE ( PROP0.translate_x > 20 )
}
condition TRANSPORT_Y {
     COMPARE ( PROP0.translate_y $ )
     COMPARE ( PROP0.translate_y > 20 )
}
condition TRANSPORT_Z {
     COMPARE ( PROP0.translate_z $ )
     COMPARE ( PROP0.translate_z > 20 )
}
condition TRANSPORT2_X {
     COMPARE ( PROP1.translate_x $ )
     COMPARE ( PROP1.translate_x > 20 )
}
condition TRANSPORT2_Y {
     COMPARE ( PROP1.translate_y $ )
     COMPARE ( PROP1.translate_y > 20 )
}
condition TRANSPORT2_Z {
     COMPARE ( PROP1.translate_z $ )
     COMPARE ( PROP1.translate_z > 20 )
}
condition IS_LOGO {
     COMPARE ( PROP0.operation == emboss )
}
condition PART_0_EXISTS {
     COMPARE ( V0.form $ )
}
condition PART_1_EXISTS {
     COMPARE ( V1.form $ )
}
condition BLOCK_AND_BLOCK {
     COMPARE ( V0.form == BLOCK )
     COMPARE ( V1.form == BLOCK )
}
```

Figure 58 The General Rule File Developed for Discrete Manufacturing (cont'd)

```
condition CYLINDER_SHAPE {
     COMPARE ( V0.form == CYLINDER )
}
condition ROUND_SHAPE_2 {
     COMPARE ( V1.form == CYLINDER )
}
condition WEDGE_SHAPE_2 {
     COMPARE ( V1.form == WEDGE )
}
condition COMPLEX_SHAPE {
     COMPARE ( V0.form == COMPLEX )
}
condition CONE_SHAPE {
     COMPARE ( V0.form == CONE )
}
condition DRILL_SIZE {
     MATH ( V0.ratio = V0.height / V0.radius )
     ASSIGN ( V0.minimum = 0.5 )
     COMPARE ( V0.ratio > V0.minimum )
     COMPARE ( V0.ratio < 25 )
     COMPARE ( V0.radius > 0.05 )
     COMPARE ( V0.radius < 2.5 )
     COMPARE ( V0.height > 0.1 )
     COMPARE ( V0.height < 15.0 )
}
condition WEDGE_SHAPE {
     COMPARE ( V0.form == WEDGE )
}
condition BLOCK_SHAPE {
     COMPARE ( V0.form == BLOCK )
}
condition BLOCK_SHAPE_2 {
     COMPARE ( V1.form == BLOCK )
}
condition BLOCK_STOCK {
     COMPARE ( V1.form == BLOCK )
}
condition SPRING_SHAPE {
     COMPARE ( V0.form == SPRING )
}
condition SPRING_ENDS {
     COMPARE ( V0.ends $ )
}
condition L_SPRING_ENDS {
     COMPARE ( V0.ends == L )
}
condition COCENTRIC {
     FIND ( V0.axis = ROTATIONAL_AXIS V0 )
     FIND ( V1.axis = ROTATIONAL_AXIS V1 )
     COMPARE ( V0.axis == V1.axis )
}
```

Figure 58 The General Rule File Developed for Discrete Manufacturing (cont'd)

```
condition THERMO_PLASTIC {
     FIND ( V0.material_class = MATERIAL_CLASS V0.material )
     COMPARE ( V0.material_class == THERMO_PLASTIC )
}
condition METAL {
     FIND ( V0.material_class = MATERIAL_CLASS V0.material )
     COMPARE ( V0.material_class == METAL )
}
//
// Various Results to build the plan, and alter the equation.
//
result ADD_PROPS_TO_PRIMITIVE {
     COPY_SET ( V0 COMBINED_SET )
     COPY_SET ( PEEP COMBINED_SET )
     EQUATION_INSERT_SYMBOL ( :0 COMBINED_SET )
     FIND ( COMBINED_SET name SET_NAME V0 )
     EQUATION_DELETE_SYMBOL ( :1 )
}
result APPEND_PROPERTIES_TO_SET {
     COPY_SET ( V0 COMBINED_SET )
     APPEND_SET ( PROP0 COMBINED_SET )
     EQUATION_INSERT_SYMBOL ( :0 COMBINED_SET )
     FIND ( COMBINED_SET name SET_NAME V0 )
     EQUATION_DELETE_SYMBOL ( :1 )
}
result APPEND_PROPERTIES_TO_SET_2 {
     COPY_SET ( V1 COMBINED_SET )
     APPEND_SET ( PROP1 COMBINED_SET )
     EQUATION_INSERT_SYMBOL ( :1 COMBINED_SET )
     FIND ( COMBINED_SET name SET_NAME V1 )
     EQUATION_DELETE_SYMBOL ( :2 )
}
result DRILL_HOLE {
     EQUATION_DELETE_VARIABLE_TERM ( REF )
     ADD_SET ( DRILL_HOLE )
     ADD_PROPERTY ( DRILL_HOLE a1 = 0.5 )
     ADD_PROPERTY ( DRILL_HOLE a2 = 1.0 )
     ADD_PROPERTY ( DRILL_HOLE height = 0.0 )
     PROPERTY_FUNCTION_VARIABLE ( DRILL_HOLE height + V0 height )
     PROPERTY_FUNCTION_NUMBER ( DRILL_HOLE height / 20.0 )
     FIND ( DRILL_HOLE name SET_NAME V0 )
     PLAN_PUSH_FORMAT ( ' DESCRIPTION ( drill hole : '
      DRILL_HOLE.name ' ; ' )
     DELETE_PROPERTY ( DRILL_HOLE name )
     PLAN_PUSH_TEXT ( ' CUTTING ( FEATURE single diameter hole ) ' )
     PLAN_PUS _TEXT ( ' CUTTING ( MACHINE qa1000 ) ' )
     PLAN_PUSH_TEXT ( ' CUTTING ( MATERIAL t-7075 ) ' )
     PLAN_PUSH_FORMAT ( ' CUTTING ( PARAMETERS ' DRILL_HOLE.height
      DRILL_HOLE.a1 DRILL_HOLE.a2 ' ) ' )
     // A costing section is added here for a trial run
```

Figure 58 The General Rule File Developed for Discrete Manufacturing (cont'd)

```
      ADD_PROPERTY ( DRILL_HOLE cost = 0.0 )
      PROPERTY_FUNCTION_VARIABLE ( DRILL_HOLE cost + V0 height )
      PROPERTY_FUNCTION_VARIABLE ( DRILL_HOLE cost * V0 radius )
      PROPERTY_FUNCTION_NUMBER ( DRILL_HOLE cost * 1250.25 )
      PROPERTY_FUNCTION_NUMBER ( DRILL_HOLE cost + 0 25 )
      DECLARE_COST ( DRILL_HOLE cost )
}
result DRILL_CHAMFER {
      EQUATION_DELETE_VARIABLE_TERM ( REF )
      ADD_SET ( CHAMFER_HOLE )
      ADD_PROPERTY ( CHAMFER_HOLE a1 = 0.5 )
      ADD_PROPERTY ( CHAMFER_HOLE a2 = 1.0 )
      ADD_PROPERTY ( CHAMFER_HOLE height = 0.0 )
      PROPERTY_FUNCTION_VARIABLE ( CHAMFER_HOLE height + V0 height )
      PROPERTY_FUNCTION_NUMBER ( CHAMFER_HOLE height / 20.0 )
      PLAN_PUSH_TEXT ( ' DESCRIPTION   drill chamfer ) ' )
      PLAN_PUSH_TEXT ( ' CUTTING ( FEATURE front Countersunk Hole ) ' )
      PLAN_PUSH_TEXT ( ' CUTTING   MACHINE qa1000 ) ' )
      PLAN_PUSH_TEXT ( ' CUTTING ( MATERIAL t-7075 ) ' )
      PLAN_PUSH_FORMAT ( ' CUTTING ( PARAMETERS ' CHAMFER_HOLE.height
       CHAMFER_HOLE.a1 CHAMFER_HOLE.a2 ' ) ' )
}
result CUT_ANGLED_SURFACE {
      EQUATION_DELETE_TERM ( :0 )
      PLAN_PUSH_TEXT ( ' DESCRIPTION ( mill surface at angle ) ' )
}
result CUT_ANGLED_SURFACE2 {
      EQUATION_DELETE_SYMBOL ( :0: )
      PLAN_PUSH_TEXT ( ' DESCRIPTION ( mill surface at angle ) ' )
}
result GET_BLOCK_STOCK {
      EQUATION_DELETE_VARIABLE_TERM ( REF )
      PLAN_PUSH_TEXT ( ' DESCRIPTION ( get block from stock room ) ' )
}
result MAKE_SPRING {
      PLAN_PUSH_TEXT ( ' DESCRIPTION ( coil spring from wire ) ' )
      ADD_SET ( COIL_SPRING )
      ADD_PROPERTY ( COIL_SPRING wire_dia = 0.0 )
      PROPERTY_FUNCTION_VARIABLE ( COIL_SPRING wire_dia + V0
       outside_radius )
      PROPERTY_FUNCTION_VARIABLE ( COIL_SPRING wire_dia - V0
       inside_radius )
      ADD_PROPERTY ( COIL_SPRING turns = 0.0 )
      PROPERTY_FUNCTION_VARIABLE ( COIL_SPRING turns + V0 turns )
      PLAN_PUSH_TEXT ( ' DESCRIPTION ( bend coil spring ) ' )
      PLAN_PUSH_FORMAT ( ' DESCRIPTION ( wire dia. =
       ' COIL_SPRING.wire_dia ' ) ' )
      PLAN_PUSH_FORMAT ( ' DESCRIPTION ( turns. = ' COIL_SPRING.turns
       ' ) ' )
      EQUATION_DELETE_SYMBOL ( :0 )
}
```

Figure 58 The General Rule File Developed for Discrete Manufacturing (cont'd)

```
result SPRING_BEND_ENDS_L {
     COPY_SET ( V0 BEND_SPRING_ENDS )
     DELETE_PROPERTY ( BEND_SPRING_ENDS ends )
     ADD_PROPERTY ( BEND_SPRING_ENDS offset = 0.0 )
     ADD_PROPERTY ( BEND_SPRING_ENDS length = 0.0 )
     PROPERTY_FUNCTION_VARIABLE ( BEND_SPRING_ENDS length + V0
      outside_radius )
     PROPERTY_FUNCTION_NUMBER ( BEND_SPRING_ENDS length * 2.0 )
     PLAN_PUSH_TEXT ( ' DESCRIPTION ( bend spring ends ) ' )
     PLAN_PUSH_FORMAT ( ' DESCRIPTION ( L offset on spring = '
      BEND_SPRING_ENDS.offset ' ) ' )
     PLAN_PUSH_FORMAT ( ' DESCRIPTION ( length of free end = '
      BEND_SPRING_ENDS.length ' ) ' )
     EQUATION_INSERT_SYMBOL ( :0: BEND_SPRING_ENDS )
     EQUATION_DELETE_SYMBOL ( :1: )
}
result MILL_BLOCK_SHAPE {
     EQUATION_DELETE_TERM ( :1 )
     COPY_SET ( V1 MILL_OUT_BLOCK )
     COPY_SET ( V0 BASE_BLOCK )
     FIND ( MILL_OUT_BLOCK name SET_NAME V1 )
     PLAN_PUSH_FORMAT ( ' DESCRIPTION ( mill out block shape : '
      MILL_OUT_BLOCK.name ' ) ' )
     DELETE_PROPERTY ( MILL_OUT_BLOCK name )
     PLAN_PUSH_TEXT ( ' CUTTING ( FEATURE Flat Rectangular Surface -
      Open, Cutter Axis Perpend. ) ' )
     PLAN_PUSH_TEXT ( ' CUTTING ( MACHINE qa1000 ) ' )
     PLAN_PUSH_FORMAT ( ' CUTTING ( MATERIAL ' BASE_BLOCK.material
      ' ) ' )
     PLAN_PUSH_FORMAT ( ' CUTTING ( PAPAMETERS ' MILL_OUT_BLOCK.width
      MILL_OUT_BLOCK.depth MILL_OUT_BLOCK.height ' ) ' )
}
result CUT_BLOCK_STOCK {
     COPY_SET ( V0 BLOCK_STOCK )
     EQUATION_DELETE_TERM ( :0 )
     PLAN_PUSH_TEXT ( ' DESCRIPTION ( cut a block from stock with band
      saw ) ' )
     PLAN_PUSH_FORMAT ( ' DESCRIPTION ( width = ' BLOCK_STOCK.width
      ' ) ' )
     PLAN_PUSH_FORMAT ( ' DESCRIPTION ( depth = ' BLOCK_STOCK.depth
      ' ) ' )
     PLAN_PUSH_FORMAT ( ' DESCRIPTION ( height = ' BLOCK_STOCK.height
      ' ) ' )
     // Calculate Cost
     ADD_PROPERTY ( BLOCK_STOCK cost = 0.0 )
     PROPERTY_FUNCTION_VARIABLE ( BLOCK_STOCK cost + V0 width )
     PROPERTY_FUNCTION_VARIABLE ( BLOCK_STOCK cost * V0 depth )
     PROPERTY_FUNCTION_VARIABLE ( BLOCK_STOCK cost * V0 height )
     FIND ( BLOCK_STOCK matl_cost MATERIAL_COST V0 material )
     PROPERTY_FUNCTION_VARIABLE ( BLOCK_STOCK cost * BLOCK_STOCK
      matl_cost )
```

Figure 58 The General Rule File Developed for Discrete Manufacturing (cont'd)

```
      PROPERTY_FUNCTION_NUMBER ( BLOCK_STOCK cost + 0.50 )
      DECLARE_COST ( BLOCK_STOCK cost )
}
result CUT_BLOCK_STOCK_PLASTIC {
      COPY_SET ( V0 BLOCK_STOCK )
      EQUATION_DELETE_TERM ( :0 )
      PLAN_PUSH_TEXT ( ' DESCRIPTION ( cut a block from stock with hot
       wire ) ' )
      PLAN_PUSH_FORMAT ( ' DESCRIPTION ( width = ' BLOCK_STOCK.width
       ' ) ' )
      PLAN_PUSH_FORMAT ( ' DESCRIPTION ( depth = ' BLOCK_STOCK.depth
       ' ) ' )
      PLAN_PUSH_FORMAT ( ' DESCRIPTION ( height = ' BLOCK_STOCK.height
       ' ) ' )
      // Calculate Cost
      ADD_PROPERTY ( BLOCK_STOCK cost = 0.0 )
      PROPERTY_FUNCTION_VARIABLE ( BLOCK_STOCK cost + V0 width )
      PROPERTY_FUNCTION_VARIABLE ( BLOCK_STOCK cost * V0 depth )
      PROPERTY_FUNCTION_VARIABLE ( BLOCK_STOCK cost * V0 height )
      FIND ( BLOCK_STOCK matl_cost MATERIAL_COST V0 material )
      PROPERTY_FUNCTION_VARIABLE ( BLOCK_STOCK cost * BLOCK_STOCK
       matl_cost )
      PROPERTY_FUNCTION_NUMBER ( BLOCK_STOCK cost + 0.30 )
      DECLARE_COST ( BLOCK_STOCK cost )
}
result CUT_ROUND_STOCK {
      COPY_SET ( V0 ROUND_STOCK )
      EQUATION_DELETE_TERM ( :0 )
      PLAN_PUSH_TEXT ( ' DESCRIPTION ( cut round stock ) ' )
      PLAN_PUSH_FORMAT ( ' DESCRIPTION ( length = ' ROUND_STOCK.length
       ' ) ' )
      PLAN_PUSH_FORMAT ( ' DESCRIPTION ( radius = ' ROUND_STOCK.radius
       ' ) ' )
}
result GET_WIP_STOCK {
      COPY_SET ( V0 WIP_STOCK )
      PLAN_PUSH_TEXT ( ' DESCRIPTION ( get WIP from inventory ) ' )
      PLAN_PUSH_FORMAT ( ' DESCRIPTION ( name = ' WIP_STOCK.description
       ' ) ' )
      EQUATION_DELETE_TERM ( :0 )
}
result FIXTURE_PART {
      EQUATION_DELETE_TERM ( :0 )
      COPY_SET ( V0 FIXTURED_PART )
      FIND ( FIXTURED_PART name SET_NAME V0 )
      PLAN_PUSH_FORMAT ( ' DESCRIPTION ( fixture part : '
       FIXTURED_PART.name ' ) ' )
      DELETE_PROPERTY ( FIXTURED_PART name )
}
```

Figure 58 The General Rule File Developed for Discrete Manufacturing (cont'd)

```
result ASSEMBLE_PARTS {
     EQUATION_DELETE_TERM ( :1 )
     PLAN_PUSH_TEXT ( ' DESCRIPTION ( add part to fixtured part ) ' )
     COPY_SET ( V1 PART )
     FIND ( PART name SET_NAME V1 )
     FIND ( PART name2 SET_NAME V0 )
     PLAN_PUSH_FORMAT ( ' DESCRIPTION ( add ' PART.name ' to '
      PART.name2 ' ) ' )
     DELETE_PROPERTY ( PART name )
     DELETE_PROPERTY ( PART name2 )
}
result ADD_LOGO {
     COPY_SET ( V0 LOGO_PART )
     APPEND_SET ( PROP0 LOGO_PART )
     EQUATION_INSERT_SYMBOL ( :0 LOGO_PART )
     DELETE_PROPERTY ( LOGO_PART operation )
     EQUATION_DELETE_SYMBOL ( :1 )
     PLAN_PUSH_TEXT ( ' DESCRIPTION ( Add Logo to part ) ' )
}
result TRANSPORT {
     COPY_SET ( V0 MOVED_PART )
     APPEND_SET ( PROP0 MOVED_PART )
     EQUATION_INSERT_SYMBOL ( :0 MOVED_PART )
     EQUATION_DELETE_SYMBOL ( :1 )
     PLAN_PUSH_FORMAT ( ' DESCRIPTION ( Move (x,y,z) by '
      MOVED_PART.translate_x ' , ' MOVED_PART.translate_y ' , '
      MOVED_PART.translate_z ' ) ' )
     DELETE_PROPERTY ( MOVED_PART translate_x )
     DELETE_PROPERTY ( MOVED_PART translate_y )
     DELETE_PROPERTY ( MOVED_PART translate_z )
}
result TRANSPORT2 {
     COPY_SET ( V1 MOVED_PART )
     APPEND_SET ( PROP1 MOVED_PART )
     EQUATION_INSERT_SYMBOL ( :1 MOVED_PART )
     EQUATION_DELETE_SYMBOL ( :2 )
     PLAN_PUSH_FORMAT ( ' DESCRIPTION ( Move (x,y,z) by '
      MOVED_PART.translate_x ' , ' MOVED_PART.translate_y ' , '
      MOVED_PART.translate_z ' ) ' )
     DELETE_PROPERTY ( MOVED_PART translate_x )
     DELETE_PROPERTY ( MOVED_PART translate_y )
     DELETE_PROPERTY ( MOVED_PART translate_z )
}
result MILL_OUT_BLOCK {
     COPY_SET ( V0 MILLED_CHUNK )
     PLAN_PUSH_TEXT ( ' DESCRIPTION ( Mill Out Block Shape ) ' )
     PLAN_PUSH_TEXT ( ' CUTTING ( FEATURE Blind Pocket ) ' )
     PLAN_PUSH_TEXT ( ' CUTTING ( MACHINE qa1000 ) ' )
     PLAN_PUSH_TEXT ( ' CUTTING ( MATERIAL t-7075 ) ' )
     PLAN_PUSH_FORMAT ( ' CUTTING ( PARAMETERS ' MILLED_CHUNK.width
```

Figure 58 The General Rule File Developed for Discrete Manufacturing (cont'd)

```
      MILLED_CHUNK.depth MILLED_CHUNK.height ' ) ' )
     EQUATION_DELETE_VARIABLE_TERM ( REF )
     ADD_PROPERTY ( MILLED_CHUNK cost = 0.0 )
     PROPERTY_FUNCTION_VARIABLE ( MILLED_CHUNK cost + V0 height )
     PROPERTY_FUNCTION_VARIABLE ( MILLED_CHUNK cost * V0 width )
     PROPERTY_FUNCTION_VARIABLE ( MILLED_CHUNK cost * V0 depth )
     PROPERTY_FUNCTION_NUMBER ( MILLED_CHUNK cost * 1250.25 )
     PROPERTY_FUNCTION_NUMBER ( MILLED_CHUNK cost + 0.25 )
     DECLARE_COST ( MILLED_CHUNK cost )
}
result MILL_OFF_ROUND {
     COPY_SET ( V1 MILLED_ROUND )
     PLAN_PUSH_TEXT ( ' DESCRIPTION ( Mill Off Round Shape ) ' )
     PLAN_PUSH_TEXT ( ' CUTTING ( FEATURE External Round ) ' )
     PLAN_PUSH_TEXT ( ' CUTTING ( MACHINE qa1000 ) ' )
     PLAN_PUSH_TEXT ( ' CUTTING ( MATERIAL t-7075 ) ' )
     PLAN_PUSH_FORMAT ( ' CUTTING ( PARAMETERS ' MILLED_ROUND.radius
      MILLED_ROUND.height ' ) ' )
     EQUATION_DELETE_SYMBOL ( :1 )
     ADD_PROPERTY ( MILLED_ROUND cost = 0.0 )
     PROPERTY_FUNCTION_VARIABLE ( MILLED_ROUND cost + V0 radius )
     PROPERTY_FUNCTION_VARIABLE ( MILLED_ROUND cost * V0 height )
     PROPERTY_FUNCTION_NUMBER ( MILLED_ROUND cost * 900.25 )
     PROPERTY_FUNCTION_NUMBER ( MILLED_ROUND cost + 0.25 )
     DECLARE_COST ( MILLED_ROUND cost )
}
result MILL_OFF_WEDGE {
     COPY_SET ( V1 MILLED_WEDGE )
     PLAN_PUSH_TEXT ( ' DESCRIPTION ( Mill Off Wedge Shape ) ' )
     PLAN_PUSH_TEXT ( ' CUTTING ( FEATURE External Planes ) ' )
     PLAN_PUSH_TEXT ( ' CUTTING ( MACHINE qa1000 ) ' )
     PLAN_PUSH_TEXT ( ' CUTTING ( MATERIAL t-7075 ) ' )
     PLAN_PUSH_FORMAT ( ' CUTTING ( PARAMETERS ' MILLED_WEDGE.width
      MILLED_WEDGE.depth MILLED_WEDGE.height ' ) ' )
     EQUATION_DELETE_SYMBOL ( :1 )
     ADD_PROPERTY ( MILLED_WEDGE cost = 0.0 )
     PROPERTY_FUNCTION_VARIABLE ( MILLED_WEDGE cost + V0 width )
     PROPERTY_FUNCTION_VARIABLE ( MILLED_WEDGE cost * V0 depth )
     PROPERTY_FUNCTION_VARIABLE ( MILLED_WEDGE cost * V0 height )
     PROPERTY_FUNCTION_NUMBER ( MILLED_WEDGE cost * 400.25 )
     PROPERTY_FUNCTION_NUMBER ( MILLED_WEDGE cost + 0.25 )
     DECLARE_COST ( MILLED_WEDGE cost )
}
```

Figure 58 The General Rule File Developed for Discrete Manufacturing (cont'd)

When this rule file is applied to the part we obtain an enlarged design file, as shown below.

```
main Rod_Support {
     EQUATION: ( & Stock ( ~ Rod_Hole ) ( ~ Mount_Feature;Mount_Hole_1
      ) ( ~ Mount_Feature;Mount_Hole_2 ) ( ~ Top_Channel ) ( ~
      Bottom_Channel ) Top_Round )
}
Stock {
     material = brass
     form = BLOCK
     width = 11.3
     depth = 3.2
     height = 6.4
}
Rod_Hole {
     form = CYLINDER
     radius = .65
     height = 11.3
     rotate_y = -90
     translate_x = 1.4
}
Mount_Feature {
     EQUATION: ( + Mount_Hole Mount_Gouge )
}
Mount_Gouge {
     form = BLOCK
     width = 2.6
     depth = 3.4
     height = 6.4
     translate_z = 1.3
}
Mount_Hole {
     form = CYLINDER
     radius = .55
     height = 6.5
}
Mount_Hole_1 {
     translate_x = -4.35
}
Mount_Hole_2 {
     translate_x = 4.35
}
Top_Channel {
     form = BLOCK
     width = 2.5
     depth = 3.4
     height = 4.0
     translate_z = 1.8
}
```

Figure 59 The Plan File for the Rod Support Mount

```
Bottom_Channel {
      form = BLOCK
      width = 3.5
      depth = 3.4
      height = 1.8
      translate_z = -2.4
}
Top_Round {
      form = CYLINDER
      radius = 3.4
      height = 12.0
      translate_x = 0.2
      rotate_y = 90
}
1:Rod_Support {
      EQUATION: ( - Rod_Hole )
}
2:Rod_Support {
      EQUATION: ( - Mount_Feature;Mount_Hole_1 )
}
3:Rod_Support {
      EQUATION: ( - Mount_Feature;Mount_Hole_2 )
}
4:Rod_Support {
      EQUATION: ( - Top_Channel )
}
5:Rod_Support {
      EQUATION: ( - Bottom_Channel )
}
MAIN_BOM {
      FEATURE ( 1.000000 Rod_Support Rod_Support_OP1 Rod_Support_PART )
}
_GLOBAL {
      plan_count = 20
}
Rod_Support_OP1 {
      EQUATION: ( & Stock ( - Rod_Hole ) ( - ( + Mount_Hole Mount_Gouge
       );Mount_Hole_1 ) ( - ( + Mount_Hole Mount_Gouge );Mount_Hole_2 )
       ( - Top_Channel ) ( - Bottom_Channel ) Top_Round )
      OPERATION ( AND RULE 2:0:0 9183.336250 Rod_Support_OP2
       drill_hole )
      OPERATION ( AND RULE 3:0:0 9183.336250 Rod_Support_OP3
       drill_hole )
      ACTIVE: 0
      EXPANDED ( 1 4 1 0 )
}
Rod_Support_OP2 {
      DESCRIPTION ( drill hole : Rod_Hole )
      CUTTING ( FEATURE single diameter hole )
      CUTTING ( MACHINE qa1000 )
      CUTTING ( MATERIAL t-7075 )
      CUTTING ( PARAMETERS 0.565000 0.5 1.0 )
```

Figure 59 The Plan File for the Rod Support Mount (cont'd)

```
      EQUATION: ( & Stock ( - ( + Mount_Hole Mount_Gouge );Mount_Hole_1
       ) ( - ( + Mount_Hole Mount_Gouge );Mount_Hole_2 ) ( -
       Top_Channel ) ( - Bottom_Channel ) Top_Round )
      OPERATION ( AND RULE 4:0:0 0.000000 Rod_Support_OP4
       append_properties )
      OPERATION ( AND RULE 5:0:0 0.000000 Rod_Support_OP5
       append_properties )
      ACTIVE: 0
      EXPANDED ( 1 15 2 0 )
}
DRILL_HOLE_3 {
      a1 = 0.5
      a2 = 1.0
      height = 0.565000
      cost = 9183.336250
}
Rod_Support_OP3 {
      DESCRIPTION ( drill hole : Rod_Hole )
      CUTTING ( FEATURE single diameter hole )
      CUTTING ( MACHINE qa1000 )
      CUTTING ( MATERIAL t-7075 )
      CUTTING ( PARAMETERS 0.565000 0.5 1.0 )
      EQUATION: ( & Stock ( - Mount_Hole;Mount_Hole_1 ) ( -
       Mount_Gouge;Mount_Hole_1 ) ( - Mount_Hole;Mount_Hole_2 ) ( -
       Mount_Gouge;Mount_Hole_2 ) ( - Top_Channel ) ( - Bottom_Channel
       ) Top_Round )
}
DRILL_HOLE_4 {
      a1 = 0.5
      a2 = 1.0
      height = 0.565000
      cost = 9183.336250
}
Rod_Support_OP4 {
      EQUATION: ( & Stock ( - COMBINED_SET_5 ) ( -
       Mount_Gouge;Mount_Hole_1 ) ( - Mount_Hole;Mount_Hole_2 ) ( -
       Mount_Gouge;Mount_Hole_2 ) ( - Top_Channel ) ( - Bottom_Channel
       ) Top_Round )
      OPERATION ( AND RULE 6:0:0 4469 893750 Rod_Support_OP6
       drill_hole )
      OPERATION ( AND RULE 7:0:0 0.000000 Rod_Support_OP7
       append_properties )
      ACTIVE: 1
      EXPANDED ( 0 15 1 0 )
}
COMBINED_SET_5 {
      form = CYLINDER
      radius = .55
      height = 6.5
      translate_x = -4.35
      translate_x = -4.35
      name = Mount_Hole;Mount_Hole_1
}
```

Figure 59 The Plan File for the Rod Support Mount (cont'd)

```
Rod_Support_OP5 {
      EQUATION: ( & Stock ( - Mount_Hole;Mount_Hole_1 ) ( -
       COMBINED_SET_6 ) ( - Mount_Hole;Mount_Hole_2 ) ( -
       Mount_Gouge;Mount_Hole_2 ) ( - Top_Channel ) ( - Bottom_Channel
       ) Top_Round )
}
COMBINED_SET_6 {
      form = BLOCK
      width = 2.6
      depth = 3.4
      height = 6.4
      translate_z = 1.3
      translate_x = -4.35
      translate_x = -4.35
      name = Mount_Gouge;Mount_Hole_1
}
Rod_Support_OP6 {
      DESCRIPTION ( drill hole : COMBINED_SET_5 )
      CUTTING ( FEATURE single diameter hole )
      CUTTING ( MACHINE qa1000 )
      CUTTING ( MATERIAL t-7075 )
      CUTTING ( PARAMETERS 0.325000 0.5 1.0 )
      EQUATION: ( & Stock ( - Mount_Gouge;Mount_Hole_1 ) ( -
       Mount_Hole;Mount_Hole_2 ) ( - Mount_Gouge;Mount_Hole_2 ) ( -
       Top_Channel ) ( - Bottom_Channel ) Top_Round )
}
DRILL_HOLE_7 {
      a1 = 0.5
      a2 = 1.0
      height = 0.325000
      cost = 4469.893750
}
Rod_Support_OP7 {
      EQUATION: ( & Stock ( - COMBINED_SET_5 ) ( - COMBINED_SET_8 ) ( -
       Mount_Hole;Mount_Hole_2 ) ( - Mount_Gouge;Mount_Hole_2 ) ( -
       Top_Channel ) ( - Bottom_Channel ) Top_Round )
      OPERATION ( AND RULE 8:0:0 4469.893750 Rod_Support_OP8
       drill_hole )
      OPERATION ( AND RULE 9:0:0 0.000000 Rod_Support_OP9
       append_properties )
      ACTIVE: 1
      EXPANDED ( 0 15 1 0 )
}
COMBINED_SET_8 {
      form = BLOCK
      width = 2.6
      depth = 3.4
      height = 6.4
      translate_z = 1.3
      translate_x = -4.35
      translate_x = -4.35
      name = Mount_Gouge;Mount_Hole_1
}
```

Figure 59 The Plan File for the Rod Support Mount (cont'd)

```
Rod_Support_OP8 {
     DESCRIPTION ( drill hole : COMBINED_SET_5 )
     CUTTING ( FEATURE single diameter hole )
     CUTTING ( MACHINE qa1000 )
     CUTTING ( MATERIAL t-7075 )
     CUTTING ( PARAMETERS 0.325000 0.5 1.0 )
     EQUATION: ( & Stock ( ~ COMBINED_SET_8 ) ( ~
      Mount_Hole;Mount_Hole_2 ) ( ~ Mount_Gouge;Mount_Hole_2 ) ( ~
      Top_Channel ) ( ~ Bottom_Channel ) Top_Round )
}
DRILL_HOLE_9 {
     a1 = 0.5
     a2 = 1.0
     height = 0.325000
     cost = 4469.893750
}
Rod_Support_OP9 {
     EQUATION: ( & Stock ( ~ COMBINED_SET_5 ) ( ~ COMBINED_SET_8 ) ( ~
      COMBINED_SET_10 ) ( ~ Mount_Gouge;Mount_Hole_2 ) ( ~ Top_Channel
      ) ( ~ Bottom_Channel ) Top_Round )
     OPERATION ( AND RULE 10:0:0 4469.893750 Rod_Support_OP10
      drill_hole )
     OPERATION ( AND RULE 11:0:0 0.000000 Rod_Support_OP11
      append_properties )
     ACTIVE: 1
     EXPANDED ( 0 15 1 0 )
}
COMBINED_SET_10 {
     form = CYLINDER
     radius = .55
     height = 6.5
     translate_x = 4.35
     translate_x = 4.35
     name = Mount_Hole;Mount_Hole_2
}
Rod_Support_OP10 {
     DESCRIPTION ( drill hole : COMBINED_SET_5 )
     CUTTING ( FEATURE single diameter hole )
     CUTTING ( MACHINE qa1000 )
     CUTTING ( MATERIAL t-7075 )
     CUTTING ( PARAMETERS 0.325000 0.5 1.0 )
     EQUATION: ( & Stock ( ~ COMBINED_SET_8 ) ( ~ COMBINED_SET_10 ) (
      ~ Mount_Gouge;Mount_Hole_2 ) ( ~ Top_Channel ) ( ~
      Bottom_Channel ) Top_Round )
}
DRILL_HOLE_11 {
     a1 = 0.5
     a2 = 1.0
     height = 0.325000
     cost = 4469.893750
}
```

Figure 59 The Plan File for the Rod Support Mount (cont'd)

```
Rod_Support_OP11 {
     EQUATION: ( & Stock ( ~ COMBINED_SET_5 ) ( ~ COMBINED_SET_8 ) ( ~
      COMBINED_SET_10 ) ( ~ COMBINED_SET_12 ) ( ~ Top_Channel ) ( ~
      Bottom_Channel ) Top_Round )
     OPERATION ( AND RULE 12:0:0 4469.893750 Rod_Support_OP12
      drill_hole )
     ACTIVE: 0
     EXPANDED ( 2 0 0 0 )
}
COMBINED_SET_12 {
     form = BLOCK
     width = 2.6
     depth = 3.4
     height = 6.4
     translate_z = 1.3
     translate_x = 4.35
     translate_x = 4.35
     name = Mount_Gouge;Mount_Hole_2
}
Rod_Support_OP12 {
     DESCRIPTION ( drill hole : COMBINED_SET_5 )
     CUTTING ( FEATURE single diameter hole )
     CUTTING ( MACHINE qa1000 )
     CUTTING ( MATERIAL t-7075 )
     CUTTING ( PARAMETERS 0.325000 0.5 1.0 )
     EQUATION: ( & Stock ( ~ COMBINED_SET_8 ) ( ~ COMBINED_SET_10 ) (
      ~ COMBINED_SET_12 ) ( ~ Top_Channel ) ( ~ Bottom_Channel )
      Top_Round )
     OPERATION ( AND RULE 13:0:0 70734.394000 Rod_Support_OP13
      mill_out_block )
     ACTIVE: 0
     EXPANDED ( 2 0 0 0 )
}
DRILL_HOLE_13 {
     a1 = 0.5
     a2 = 1.0
     height = 0.325000
     cost = 4469.893750
}
Rod_Support_OP13 {
     DESCRIPTION ( Mill Out Block Shape )
     CUTTING ( FEATURE Blind Pocket )
     CUTTING ( MACHINE qa1000 )
     CUTTING ( MATERIAL t-7075 )
     CUTTING ( PARAMETERS 2.6 3.4 6.4 )
     EQUATION: ( & Stock ( ~ COMBINED_SET_10 ) ( ~ COMBINED_SET_12 ) (
      ~ Top_Channel ) ( ~ Bottom_Channel ) Top_Round )
     OPERATION ( AND RULE 14:0:0 4469.893750 Rod_Support_OP14
      drill_hole )
     ACTIVE: 0
     EXPANDED ( 2 0 0 0 )
}
```

Figure 59 The Plan File for the Rod Support Mount (cont'd)

```
MILLED_CHUNK_14 {
      form = BLOCK
      width = 2.6
      depth = 3.4
      height = 6.4
      translate_z = 1.3
      translate_x = -4.35
      translate_x = -4.35
      name = Mount_Gouge;Mount_Hole_1
      cost = 70734.394000
}
Rod_Support_OP14 {
      DESCRIPTION ( drill hole : COMBINED_SET_10 )
      CUTTING ( FEATURE single diameter hole )
      CUTTING ( MACHINE qa1000 )
      CUTTING ( MATERIAL t-7075 )
      CUTTING ( PARAMETERS 0.325000 0.5 1.0 )
      EQUATION: ( & Stock ( ~ COMBINED_SET_12 ) ( ~ Top_Channel ) ( ~
       Bottom_Channel ) Top_Round )
      OPERATION ( AND RULE 15:0:0 70734.394000 Rod_Support_OP15
       mill_out_block )
      ACTIVE: 0
      EXPANDED ( 2 0 0 0 )
}
DRILL_HOLE_15 {
      a1 = 0.5
      a2 = 1.0
      height = 0.325000
      cost = 4469.893750
}
Rod_Support_OP15 {
      DESCRIPTION ( Mill Out Block Shape )
      CUTTING ( FEATURE Blind Pocket )
      CUTTING ( MACHINE qa1000 )
      CUTTING ( MATERIAL t-7075 )
      CUTTING ( PARAMETERS 2.6 3.4 6.4 )
      EQUATION: ( & Stock ( ~ Top_Channel ) ( ~ Bottom_Channel )
       Top_Round )
      OPERATION ( AND RULE 16:0:0 42508.750000 Rod_Support_OP16
       mill_out_block )
      ACTIVE: 0
      EXPANDED ( 2 0 0 0 )
}
MILLED_CHUNK_16 {
      form = BLOCK
      width = 2.6
      depth = 3.4
      height = 6.4
      translate_z = 1.3
      translate_x = 4.35
      translate_x = 4.35
```

Figure 59 The Plan File for the Rod Support Mount (cont'd)

```
      name = Mount_Gouge;Mount_Hole_2
      cost = 70734.394000
}
Rod_Support_OP16 {
      DESCRIPTION ( Mill Out Block Shape )
      CUTTING ( FEATURE Blind Pocket )
      CUTTING ( MACHINE qa1000 )
      CUTTING ( MATERIAL t-7075 )
      CUTTING ( PARAMETERS 2.5 3.4 4.0 )
      EQUATION: ( & Stock ( - Bottom_Channel ) Top_Round )
      OPERATION ( AND RULE 17:0:0 26780.605000 Rod_Support_OP17
       mill_out_block )
      ACTIVE: 0
      EXPANDED ( 2 0 0 0 )
}
MILLED_CHUNK_17 {
      form = BLOCK
      width = 2.5
      depth = 3.4
      height = 4.0
      translate_z = 1.8
      cost = 42508.750000
}
Rod_Support_OP17 {
      DESCRIPTION ( Mill Out Block Shape )
      CUTTING ( FEATURE Blind Pocket )
      CUTTING ( MACHINE qa1000 )
      CUTTING ( MATERIAL t-7075 )
      CUTTING ( PARAMETERS 3.5 3.4 1.8 )
      EQUATION: ( & Stock Top_Round )
      OPERATION ( AND RULE 18:0:0 0.250000 Rod_Support_OP18
       mill_off_round )
      ACTIVE: 0
      EXPANDED ( 2 0 0 0 )
}
MILLED_CHUNK_18 {
      form = BLOCK
      width = 3.5
      depth = 3.4
      height = 1.8
      translate_z = -2.4
      cost = 26780.605000
}
Rod_Support_OP18 {
      DESCRIPTION ( Mill Off Round Shape )
      CUTTING ( FEATURE External Round )
      CUTTING ( MACHINE qa1000 )
      CUTTING ( MATERIAL t-7075 )
      CUTTING ( PARAMETERS 3.4 12.0 )
      EQUATION: ( & Stock )
      OPERATION ( AND RULE 19:0:0 34714.100000 Rod_Support_OP19
       get_cut_block_stock )
```

Figure 59 The Plan File for the Rod Support Mount (cont'd)

```
      ACTIVE: 0
      EXPANDED ( 2 0 0 0 )
}
MILLED_ROUND_19 {
      form = CYLINDER
      radius = 3.4
      height = 12.0
      translate_x = 0.2
      rotate_y = 90
      cost = 0.250000
}
Rod_Support_OP19 {
      DESCRIPTION ( cut a block from stock with band saw )
      DESCRIPTION ( width = 11.3 )
      DESCRIPTION ( depth = 3.2 )
      DESCRIPTION ( height = 6.4 )
      EQUATION: NULL
}
BLOCK_STOCK_20 {
      material = brass
      form = BLOCK
      width = 11.3
      depth = 3.2
      height = 6.4
      cost = 34714.100000
      matl_cost = 150.000000
}
Rod_Support_PART {
      form = COMPLEX
      description = Rod_Support
      equation = ( & Stock ( - Rod_Hole ) ( - ( + Mount_Hole
       Mount_Gouge );Mount_Hole_1 ) ( - ( + Mount_Hole Mount_Gouge
       );Mount_Hole_2 ) ( - Top_Channel ) ( - Bottom_Channel )
       Top_Round )
}
```

Figure 59 The Plan File for the Rod Support Mount (cont'd)

The file listed above lists not only the design file information, but also the process plan information, as discussed in the previous chapters. One point of interest is that for some of the operations there are multiple choices. Only one of these has been selected, but the system is at least choosing between a number of alternatives. When this file is interpreted through to the final design, the operation plan results are listed below.

```
-------- Work Order Sheets ------------
Product: Rod_Support


OPERATION SUMMARY_SHEET: Rod_Support_PART - Quantity 1.000000
 OP# Operation Description
-----------------------------------------
 0    cut a block from stock with band saw
      width = 11.3
      depth = 3.2
      height = 6.4
 10   Mill Off Round Shape
 20   Mill Out Block Shape
 30   Mill Out Block Shape
 40   Mill Out Block Shape
 50   drill hole : COMBINED_SET_10
 60   Mill Out Block Shape
 70   drill hole : COMBINED_SET_5
 120  drill hole : Rod_Hole
```

Figure 60 An Operation Plan for the Rod Support Mount

Here again we can see that many of the operations may be alternated in order. For example, BCAPP would be quite likely to schedule the drilling of the 'Mount_feature' holes before the 'Mount_gouges'. This would not make sense, but without the solid modeling capabilities, it is difficult to detect the excess volume that would have to be drilled for the reversed sequence.

7.4 MULTI-DOMAIN PLANNING

In both examples so far the process domain has been limited to machining. It is very unusual to find a part that is manufactured using one type of an industrial process, such as machining only. Therefore another important test of the planner was planning for a product with mixed operation types and parts. The part selected is shown below. The product is a paper clip in the form of a 'clothes pin'. The clip has two halves and a connecting spring. On one of the sides there is an advertising logo embossed on. The same side also has a hole drilled into it so that the clip may be mounted on a wall. In total there are a number of possible technol-

ogies that may be applied to the part. First, the spring is quite straight forward, and only involves the forming of some wire. In the second case, either of the clip halves could be made by injection molding (this was used in the actual production of the sample product used). But, in other cases, it is possible to machine the entire part from a piece of polypropylene. The decision can be based on the quantities of parts to be produced. For example, to machine two halves of a clip for a prototype might only cost a few hundred dollars, as opposed to $50,000 to $100,000 for molds for injection molding (the figure are based on my personal experience). A graphical representation of the clip is shown below.

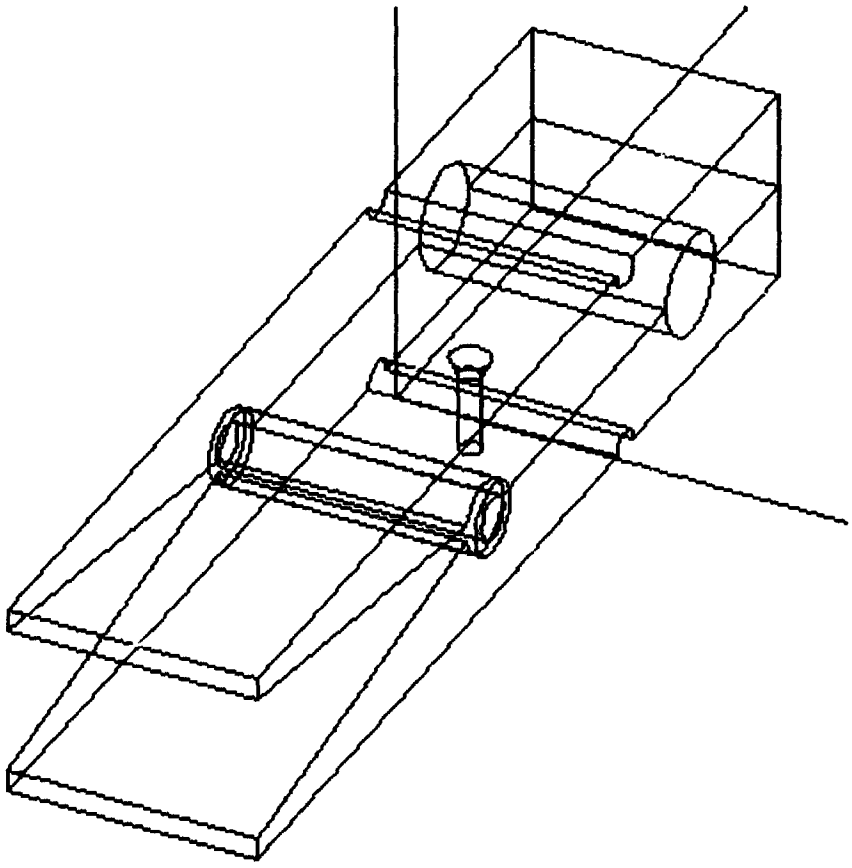


Figure 61 A Test Part - A Large Plastic Clothes Pin

```
// This file contains a test part for the BCAPP system. The part is a
// large Plastic Clothes peg which is intended for use as an paper
// holder. It is also has an embossed logo on the side.
// -------------------- Main Geometry Definition --------------------
//

main Big_Clothes_Pin {
     EQUATION: ( : Spring;attach_spring Logo_Side;position_logo_side
      Clip_half )
}
// ------------------ Assembly and positioning information ----------
Package {
     quantity = 20
}
attach_spring {
     translate_y = -0.4
     translate_z = -0.275
}
position_logo_side {
     translate_z = 0.55
     rotate_y = 180
}
// -------------------- Defintion of Logo Side of Clip -------------
Logo_Side {
     EQUATION: ( : ( & Clip_half ( - Screw_hole );position_screw_hole
      );add_logo )
}
Screw_hole {
     EQUATION: ( + HOLE CHAMFER )
}
HOLE {
     form = CYLINDER
     radius = 0.06
     height = 0.6
}
CHAMFER {
     form = CONE
     radius = 0.125
     height = 0.25
     translate_z = -0.18
}
position_screw_hole {
     translate_y = 0.85
}
add_logo {
     operation = emboss
//   description = ' add CBS logo '
}
// ---------- This is the box to put the pins into ------------------
Box {
     description = #10_standard_box
     EQUATION: ( & OUTSIDE ( ~ INSIDE ) )
}
```

Figure 62 Big Clothes Peg Product Description File

```
OUTSIDE {
      form = BLOCK
      width = 15.2
      depth = 6.1
      height = 1.3
      translate_z = -0.1001
}
INSIDE {
      form = BLOCK
      width = 15
      depth = 5.9
      height = 1.1
}
// ---------- This is the clip spring (This is a tough definition) ---
Spring {
      form = SPRING
      ends = L
      inside_radius = 0.12
      outside_radius = 0.2
      length = 1.50
      turns = 14.5
      rotate_y = 90
}
// ------------ The geometry for one half of the clip is defined here
Clip_half {
      EQUATION: ( & ( ~ B ) ( ~ C ) ( & A E ) ( ~ D ) )
      material = polypropylene
      color = hong_kong_red
}
A {
      form = BLOCK
      width = 1.4
      depth = 5.9
      height = 0.55
      material = polypropylene
}
B {
      form = CYLINDER
      radius = 0.28
      height = 1.45
      translate_x = 0.275
      translate_y = 1.95
      rotate_y = 90
}
C {
      form = CYLINDER
      radius = 0.1
      height = 1.45
      translate_x = -0.275
      translate_y = 1.2
      rotate_y = 90
}
```

Figure 62 Big Clothes Peg Product Description File (cont'd)

```
D {
      form = CYLINDER
      radius = .2
      height = 1.45
      translate_x = 0.275
      translate_y = -0.4
      rotate_y = 90
}
E {
      form = BLOCK
      height = 10
      width = 10
      depth = 10
      translate_z = 4.68
      rotate_x = -9
}
// ---------------------------------------------------------------------
```

Figure 62 Big Clothes Peg Product Description File (cont'd)

After the rule file listed previously (in Figure 58) was applied, the following output file was obtained.

```
-------- Work Order Sheets ------------
Product: Big_Clothes_Pin


OPERATION SUMMARY_SHEET: Clip_half_PART - Quantity 2.000000
 OP# Operation Description
-----------------------------------------
0     cut a block from stock with hot wire
      width = 1.4
      depth = 5.9
      height = 0.55
10    drill hole : D
20    drill hole : C
30    drill hole : B
40    mill out block shape : E
```

Figure 63 Operation Plan for the Big Clothes Pin

```
OPERATION SUMMARY_SHEET: 0:Logo_Side_PART - Quantity 1.000000
 OP# Operation Description
----------------------------------------
1000 get WIP from inventory
     name = Clip_half
1010 drill chamfer
1020 drill hole : COMBINED_SET_10


OPERATION SUMMARY_SHEET· Logo_Side_PART - Quantity 1.000000
 OP# Operation Description
----------------------------------------
2000 fixture part : LOGO_PART_18
2010 Add Logo to part


OPERATION SUMMARY_SHEET: Spring_PART - Quantity 1.000000
 OP# Operation Description
----------------------------------------
3000 coil spring from wire
     bend coil spring
     wire dia. = 0.080000
     turns. = 14.500000
3010 bend spring ends
     L offset on spring = 0.0
     length of free end = 3.400000


OPERATION SUMMARY_SHEET: Big_Clothes_Pin_PART - Quantity 1.000000
 OP# Operation Description
----------------------------------------
4000 fixture part : Spring_PART;attach_spring
4010 add part to fixtured part
     add Clip_half_PART to COMBINED_SET_28
4020 add part to fixtured part
     add Logo_Side_PART;position_logo_side to COMBINED_SET_28
```

Figure 63 Operation Plan for the Big Clothes Pin (cont'd)

The plan above exhibits a few very important factors. First, the plan contains a range of operations, from forming to cutting. Thus showing the independence from a single processing domain. Also shown is the ability for the planner to break the product into natural components, one of these being the clip half with the logo added. In this case the planner separated out the addition of the logo,

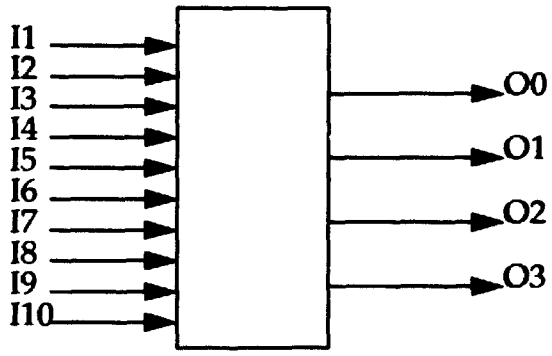
even though the design does not indicate it as an individual assembly. This is in keeping with what would happen in an industrial environment, where a number of clip halves would be made in a large economical batch, then the logos would be added a later time to smaller batches, as requested by a customer.

The reader is also asked to consider the flexible nature of the description within the plan. This indicates that the plan can be easily customized to suit the varying needs and abilities of the manufacturer.

7.5 SELECTION OF ELECTRICAL COMPONENTS

Up to this point the examples have been mainly limited to discrete manufacturing of mechanical parts. In this section a somewhat novel approach of the planner will be suggested. This is given for some illustration of the implications of this planning approach. The example chosen was a digital circuit design for an encoder that would take ten input lines, and derive a four bit binary output number that would indicate which of the input lines is active. The design was done in a format similar to the previous designs, except that obvious features such as geometrical features are no longer needed.

The digital-to-binary encoder is a simple logical circuit using combinatorial logic expressions. The truth table for such a system is shown below.



Input Line Number		Output Binary			
Digit	Input Name	O0	O1	O2	O3 (Least Significant)
1	I1	0	0	0	1
2	I2	0	0	1	0
3	I3	0	0	1	1
4	I4	0	1	0	0
5	I5	0	1	0	1
6	I6	0	1	1	0
7	I7	0	1	1	1
8	I8	1	0	0	0
9	I9	1	0	0	1
10	I10	1	0	1	0

Figure 64 Truth Table for Digital to Binary Encoder

This can be directly converted to as design file, as shown below.

```
// This is a description of a digital circuit that does 10 bit digital
// input to 4 bit binary output conversion
main decade_to_binary {
     EQUATION: ( : A B C D )
}
A {
     // The least significant bit
     EQUATION: ( + I1 I3 I5 I7 I9 )
}
B {
     // The second least significant bit
     EQUATION: ( + I2 I3 I6 I7 I10 )
}
```

Figure 65 A Product Description File for a Digital to Binary Encoder

```
C {
      // The second most significant bit
      EQUATION: ( + I4 I5 I6 I7 )
}
D {
      // The most significant bit
      EQUATION: ( + I8 I9 I10 )
}
I1 {
      // input for 0
      form = INPUT
}
I2 {
      // input for 1
      form = INPUT
}
I3 {
      // input for 2
      form = INPUT
}
I4 {
      // input for 3
      form = INPUT
}
I5 {
      // input for 4
      form = INPUT
}
I6 {
      // input for 5
      form = INPUT
}
I7 {
      // input for 6
      form = INPUT
}
I8 {
      // input for 7
      form = INPUT
}
I9 {
      // input for 8
      form = INPUT
}
I10 {
      // input for 9
      form = INPUT
}
```

Figure 65 A Product Description File for a Digital to Binary Encoder (cont'd)

The file above is sufficient to describe the function of the circuit. Interesting is

the representation of the parallel Boolean equations using the assembly operator that is not often used in standard switching circuit design theory. This design was then planned with a set of rules shown below.

```
// Rules for selecting chips for digital circuits

// The equation templates are listed here
equation_form 2_INPUT_OR_GATES {
     EQUATION: ( + VAR:V:0 VAR:V:1 ...)
     RULE: 2_input_or
}
equation_form 3_INPUT_OR_GATES {
     EQUATION: ( + VAR:V:0 VAR:V:1 VAR:V:2 ...)
     RULE: 3_input_or
}
equation_form 4_INPUT_OR_GATES {
     EQUATION: ( + VAR:V:0 VAR:V:1 VAR:V:2 VAR:V:3 ...)
     RULE: 4_input_or
}
equation_form FIND_SINGLE_TERM_OR {
     EQUATION: ( + VAR:V:0 )
     RULE: Eliminate_single_term
}
equation_form OUTPUT_LIST {
     EQUATION: (> : VAR:V:0 ...)
     RULE: List_output_connection
}

// The rules are listed here
rule 2_input_or {
     EQUATION: ( & ( + 1_INPUT 1_RESULTANT ) ( + 2_INPUT
      2_RESULTANT ) )
     RESULT: USE_2_INPUT_OR_GATE
}
rule 3_input_or {
     EQUATION: ( & ( + 1_INPUT 1_RESULTANT ) ( + 2_INPUT 2_RESULTANT )
      ( + 3_INPUT 3_RESULTANT ) )
     RESULT: USE_3_INPUT_OR_GATE
}
rule 4_input_or {
     EQUATION: ( & ( + 1_INPUT 1_RESULTANT ) ( + 2_INPUT 2_RESULTANT )
      ( + 3_INPUT 3_RESULTANT ) ( + 4_INPUT 4_RESULTANT ) )
     RESULT: USE_4_INPUT_OR_GATE
}
```

Figure 66 Rules For Converting a Digital Design to Gate Embodiment

```
rule List_output_connection {
     EQUATION: ( & 1_EXISTS )
     RESULT: ADD_OUTPUT_TO_LIST
}
rule Eliminate_single_term {
     EQUATION: ( & 1_EXISTS )
     RESULT: KILL_SINGLE_TERM
}

// The rule conditions are used here
condition 1_EXISTS {
     COMPARE ( V0.form $ )
}
condition 1_INPUT {
     COMPARE ( V0.form $ )
     COMPARE ( V0.form == INPUT )
}
condition 1_RESULTANT {
     COMPARE ( V0.form $ )
     COMPARE ( V0.form == CALCULATED )
}
condition 2_INPUT {
     COMPARE ( V1.form $ )
     COMPARE ( V1.form == INPUT )
}
condition 2_RESULTANT {
     COMPARE ( V1.form $ )
     COMPARE ( V1.form == CALCULATED )
}
condition 3_INPUT {
     COMPARE ( V2.form $ )
     COMPARE ( V2.form == INPUT )
}
condition 3_RESULTANT {
     COMPARE ( V2.form $ )
     COMPARE ( V2.form == CALCULATED )
}
condition 4_INPUT {
     COMPARE ( V3.form $ )
     COMPARE ( V3.form == INPUT )
}
condition 4_RESULTANT {
     COMPARE ( V3.form $ )
     COMPARE ( V3.form == CALCULATED )
}

// The rule results are defined here
result USE_2_INPUT_OR_GATE {
     COPY_SET ( V0 2_INPUT_GATE )
     FIND ( 2_INPUT_GATE input_0 SET_NAME V0 )
```

Figure 66 Rules For Converting a Digital Design to Gate Embodiment (cont'd)

```
      FIND ( 2_INPUT_GATE input_1 SET_NAME V1 )
      EQUATION_INSERT_SYMBOL ( :0 2_INPUT_GATE )
      EQUATION_DELETE_SYMBOL ( :1 )
      EQUATION_DELETE_SYMBOL ( :1 )
      DELETE_PROPERTY ( 2_INPUT_GATE form )
      ADD_PROPERTY ( 2_INPUT_GATE form = CALCULATED )
      ADD_PROPERTY ( 2_INPUT_GATE cost = 0.50 )
      DECLARE_COST ( 2_INPUT_GATE cost )
      PLAN_PUSH_FORMAT ( ' DESCRIPTION ( Use 2 input OR gate for '
       2_INPUT_GATE.input_0 2_INPUT_GATE.input_1 ' ) ' )
      DELETE_PROPERTY ( 2_INPUT_GATE input_0 )
      DELETE_PROPERTY ( 2_INPUT_GATE input_1 )
      FIND ( 2_INPUT_GATE output THIS_NAME 2_INPUT_GATE )
      PLAN_PUSH_FORMAT ( ' DESCRIPTION ( Output is '
       2_INPUT_GATE.output ' ) ' )
      DELETE_PROPERTY ( 2_INPUT_GATE output )
}
result USE_3_INPUT_OR_GATE {
      COPY_SET ( V0 3_INPUT_GATE )
      FIND ( 3_INPUT_GATE input_0 SET_NAME V0 )
      FIND ( 3_INPUT_GATE input_1 SET_NAME V1 )
      FIND ( 3_INPUT_GATE input_2 SET_NAME V2 )
      EQUATION_INSERT_SYMBOL ( :0 3_INPUT_GATE )
      EQUATION_DELETE_SYMBOL ( :1 )
      EQUATION_DELETE_SYMBOL ( :1 )
      EQUATION_DELETE_SYMBOL ( :1 )
      DELETE_PROPERTY ( 3_INPUT_GATE form )
      ADD_PROPERTY ( 3_INPUT_GATE form = CALCULATED )
      ADD_PROPERTY ( 3_INPUT_GATE cost = 0.65 )
      DECLARE_COST ( 3_INPUT_GATE cost )
      PLAN_PUSH_FORMAT ( ' DESCRIPTION ( Use 3 input OR gate for '
       3_INPUT_GATE.input_0 3_INPUT_GATE.input_1 3_INPUT_GATE.input_2
       ' ) ' )
      DELETE_PROPERTY ( 3_INPUT_GATE input_0 )
      DELETE_PROPERTY ( 3_INPUT_GATE input_1 )
      DELETE_PROPERTY ( 3_INPUT_GATE input_2 )
      FIND ( 3_INPUT_GATE output THIS_NAME 3_INPUT_GATE )
      PLAN_PUSH_FORMAT ( ' DESCRIPTION ( Output is '
       3_INPUT_GATE.output ' ) ' )
      DELETE_PROPERTY ( 3_INPUT_GATE output )
}
result USE_4_INPUT_OR_GATE {
      COPY_SET ( V0 4_INPUT_GATE )
      FIND ( 4_INPUT_GATE input_0 SET_NAME V0 )
      FIND ( 4_INPUT_GATE input_1 SET_NAME V1 )
      FIND ( 4_INPUT_GATE input_2 SET_NAME V2 )
      FIND ( 4_INPUT_GATE input_3 SET_NAME V3 )
      EQUATION_INSERT_SYMBOL ( :0 4_INPUT_GATE )
      EQUATION_DELETE_SYMBOL ( :1 )
      EQUATION_DELETE_SYMBOL ( :1 )
```

Figure 66 Rules For Converting a Digital Design to Gate Embodiment (cont'd)

```
      EQUATION_DELETE_SYMBOL ( :1 )
      DELETE_PROPERTY ( 4_INPUT_GATE form )
      ADD_PROPERTY ( 4_INPUT_GATE form = CALCULATED )
      ADD_PROPERTY ( 4_INPUT_GATE cost = 0.75 )
      DECLARE_COST ( 4_INPUT_GATE cost )
      PLAN_PUSH_FORMAT ( ' DESCRIPTION ( Use 4 input OR gate for '
       4_INPUT_GATE.input_0 4_INPUT_GATE.input_1 4_INPUT_GATE.input_2
       4_INPUT_GATE.input_3 ' ) ' )
      DELETE_PROPERTY ( 4_INPUT_GATE input_0 )
      DELETE_PROPERTY ( 4_INPUT_GATE input_1 )
      DELETE_PROPERTY ( 4_INPUT_GATE input_2 )
      DELETE_PROPERTY ( 4_INPUT_GATE input_3 )
}

result ADD_OUTPUT_TO_LIST {
      COPY_SET ( V0 OUTPUT )
      FIND ( OUTPUT output SET_NAME V0 )
      EQUATION_DELETE_SYMBOL ( :0 )
      ADD_PROPERTY ( OUTPUT form = CALCULATED )
      ADD_PROPERTY ( OUTPUT cost = 0.05 )
      DECLARE_COST ( OUTPUT cost )
      PLAN_PUSH_FORMAT ( ' DESCRIPTION ( Connect output for '
       OUTPUT.output ' ) ' )
}
result KILL_SINGLE_TERM {
      COPY_SET ( V0 OUTPUT )
      FIND ( OUTPUT output SET_NAME V0 )
      FIND ( OUTPUT terminal THIS_PART )
      EQUATION_DELETE_SYMBOL ( :0 )
      PLAN_PUSH_FORMAT ( ' DESCRIPTION ( Connect gate output '
       OUTPUT.output   to output ' OUTPUT.terminal ' ) ' )
}
```

Figure 66 Rules For Converting a Digital Design to Gate Embodiment (cont'd)

For the sake of brevity, the rules only used OR gates, but these are sufficient to implement the design. These rules were applied to the design to get the following process plan for connecting the circuit.

```
-------- Work Order Sheets ------------
Product: decade_to_binary
OPERATION SUMMARY_SHEET: D_PART - Quantity 1.000000
 OP#        Operation Description
-----------------------------------------
 0          Connect gate output 2_INPUT_GATE_5 to output D_OP5
 10         Use 2 input OR gate for 2_INPUT_GATE_3 I10
            Output is 2_INPUT_GATE_5
 20         Use 2 input OR gate for I8 I9
            Output is 2_INPUT_GATE_3
OPERATION SUMMARY_SHEET: C_PART - Quantity 1.000000
 OP#        Operation Description
-----------------------------------------
 1000       Connect gate output 2_INPUT_GATE_12 to output C_OP12
 1010       Use 2 input OR gate for 2_INPUT_GATE_10 I7
            Output is 2_INPUT_GATE_12
 1020       Use 2 input OR gate for 2_INPUT_GATE_8 I6
            Output is 2_INPUT_GATE_10
 1030       Use 2 input OR gate for I4 I5
            Output is 2_INPUT_GATE_8
OPERATION SUMMARY_SHEET: B_PART - Quantity 1.000000
 OP#        Operation Description
-----------------------------------------
 2000       Connect gate output 2_INPUT_GATE_21 to output B_OP21
 2010       Use 2 input OR gate for 2_INPUT_GATE_19 I10
            Output is 2_INPUT_GATE_21
 2020       Use 2 input OR gate for 2_INPUT_GATE_17 I7
            Output is 2_INPUT_GATE_19
 2030       Use 2 input OR gate for 2_INPUT_GATE_15 I6
            Output is 2_INPUT_GATE_17
 2040       Use 2 input OR gate for I2 I3
            Output is 2_INPUT_GATE_15
OPERATION SUMMARY_SHEET: A_PART - Quantity 1.000000
 OP#        Operation Description
-----------------------------------------
 3000       Connect gate output 2_INPUT_GATE_30 to output A_OP30
 3010       Use 2 input OR gate for 2_INPUT_GATE_28 I9
            Output is 2_INPUT_GATE_30
 3020       Use 2 input OR gate for 2_INPUT_GATE_26 I7
            Output is 2_INPUT_GATE_28
 3030       Use 2 input OR gate for 2_INPUT_GATE_24 I5
            Output is 2_INPUT_GATE_26
 3040       Use 2 input OR gate for I1 I3
            Output is 2_INPUT_GATE_24
OPERATION SUMMARY_SHEET: decade_to_binary_PART - Quantity 1.000000
 OP#        Operation Description
-----------------------------------------
 4000       Connect output for D_PART
 4010       Connect output for C_PART
 4020       Connect output for B_PART
 4030       Connect output for A_PART
```

Figure 67 A Process Plan for Circuit Construction

This process plan is a bit naive, but it does allow construction of the part automatically. Some points of interest in this case are,

- the operations to produce the part are reversed. This is not a large problem, but indicates that some problems are not necessarily suited to backwards planning,
- two input OR gates have been chosen, as opposed to 3 or 4 input gates. This is the result of a search routine that only considers the cheapest operation first. A more extensive search routine would do a more cost based approach to planning,
- the system was easily able to track inputs and outputs to gates, and generate unique identifiers to minimize confusion.

This example illustrates that the BCAPP approach is truly multi-domain, and is able to be extended to a number of problems.

7.6 CONCLUSION

This section outlined a number of successful examples for process planning, including machining, assembly, embossing, forming, and electrical assembly. Some problems were indicated in terms of design representation, sequencing, planning approach, cost optimization, and the need for geometrical modeling.

The various machined parts were planned successfully, and indicated that the planner would work well for a single process domain. The addition of multiple process domains increased the challenge to the planner. This example clearly showed the shortcoming of the method by suggesting that a drilling operation be used for a hole that is only half embedded in the workpart (and would cause drill bit deflection). In addition, this example demonstrated the planners ability to deal with complex shapes (the spring). The planner was also capable of recognizing the symmetry of two parts, and it produced them in batches of two, instead of as two separate parts. Finally the process plan included a printing operation, that would effectively have no geometrical significance, but be of great importance to

the value of the final product.

The digital circuit example illustrated that the planner was also capable of doing innovative planning problems. But, the plan for the selection and assembly of the circuit was reversed in order. This in itself is not significant, but it does indicate that the planner requires more study.

8.CONCLUSIONS

8.1 THE OBJECTIVES

It is obvious from the previous chapters that the BCAPP approach was successful in generating process plans. But this alone is not the only measure of fitness in the method. The objectives initially stated in the first chapter are an important guide in evaluating the success of the method. These will be addressed one at a time.

"Simplify the problem of recognizing features from design" - By using the Boolean equations and sets, I have sidestepped the problems that arise when defining geometry by surfaces, vertices, and points, but have maintained the power of feature based systems. In fact the definition of the Spring in one of the test cases illustrates that the design representation can encompass features also.

"Be able to recognize alternative production technologies to produce product features. This requires the planner be able to consider multiple planning domains" - The examples clearly illustrate the ease with which the system combines multiple planning domains. For example one product is manufactured using metal forming, machining, embossing, stock retrieval and assembly.

"Be able to produce alternative operations for each feature" - This is shown through examination of the process plans with multiple operations listed.

"Allow some degree of innovation in the process plan" - Innovation will naturally occur in these process plans. This is mostly because the system is not intelligent enough to ignore some uncommon approaches.

"Permit a structure that allows feedback of production problems to the process planner" - As the thesis has illustrated, process planning feedback can be used by

the system, and if enhanced with a complete manufacturing database, this information could be used by the process planner immediately.

"To allow the computer to reduce the knowledge barrier between process planning and production" - By using manufacturing rules, to capture some production knowledge, it reduces the requirements on the process planner to immediately know all detailed cases that may occur in production.

"Minimize human effort and intervention when process planning" - This item has two aspects. At best the system will work unguided, and produce good process plans. At worst, the system will get stuck, but still provide the planner with partial process plans. Also, since all plans are generated from rules at present, this system is easily classified as generative, but in the future the system could also make use of a variant approach (also known as semi-generative).

"Be capable of accepting new manufacturing technologies without fundamental changes in the process planner" - The use of Boolean expressions for mathematical combinations allows easy incorporation of new technologies through the addition of new rules. This statement can be supported by the argument that since Boolean algebra is a rigorous form of mathematics, and it can generate all possible mappings of space, it can represent any physical product. Since the BCAPP rules are based on these methods, they can be formed to accommodate any physical configuration.

"Be able to optimize process plans" - The ability to generate alternates, as BCAPP does, allows the system to iteratively try to reduce the costs of manufacturing. This does not mean the plan will have a minimum cost, but the system can reduce the cost, with an increase in computational time.

"Handle all products" - As can be seen by the variety of examples, the method is capable of handling many possible products. But, at present there are still outstanding research questions about modelling some products with CSG models. For example the spring in the Large Clothes Pin has an unusual shape that is hard to model with CSG.

8.2 SOFTWARE IMPLEMENTATION

The software that was written for this thesis has been developed in a modular structure. All of the functions are isolated whenever possible. For example the Boolean algebra representation is stored in one C++ class, while the Boolean algebra manipulation subroutines refers to that class. The sets that the equations refer to are stored in yet another class. This hierarchy continues up, and beyond the planning level.

All software was written in a portable style, and it can be compiled and run under UNIX and MS-DOS at the present time (Excluding the MetCAPP component that is only available on the UNIX platform). To accommodate the wide variety of platforms, the software utilizes dynamic memory allocation. For the test cases the memory usage grew anywhere from less than 100 Kilobytes to over 200 Kilobytes for the examples in this thesis. The base memory usage is about 200 Kilobytes for the executable programs. Computation time also varied between 1 to 30 seconds on both the UNIX and MS-DOS platforms.

When this software is actually used in an industrial setting it will probably require about 50 rules for a small job shop, and hundreds of rules for large corporations. At present the relationship is hard to evaluate, and will remain so until an industrial implementation is tried.

To date the entire planning process is automatic. In the best of cases this is ideal, but for industrial use, it would be advisable to allow human direction, and verification for all plan steps. This would require the addition of an extensive user interface that does not exist at the present time.

The design files were all created by hand. Eventually the designs should be obtained as the output from a CAD system. Even more useful would be a CAD and CAPP system that would share a common database, and not need to encode the design in ASCII format for transfer between software. This tight coupling would allow realtime interaction between CAD and CAPP. The same can also be said about an interface to Scheduling.

The rule files are constructed by hand, and would have to be produced by a skilled Knowledge Engineer. At present this will lead to difficulties in locating personnel capable of developing the rule set. Eventually other methods and techniques could be developed to simplify the collection of new rules.

The final product of BCAPP is a design file that has been expanded to include process information. This format has benefits for replanning, and also referring back to the original design after process planning. At present I have provided a simple utility to convert this file into a set of Operation Sheets, and provide some operation planning using MetCAPP. Eventually this should be replaced by specific planning modules for the specific manufacturing technologies.

8.3 SUMMARY

By using Boolean equations, and sets to represent designs, it has been shown to be possible to generate process plans. The process plans have a number of features of interest,

- multiple manufacturing domains are considered,
- cost can be considered,
- sequencing is somewhat inherent, although not required,
- even without interpreting the product geometry, the system is able to suggest process plans.

The input form used by the planner does not contain unreasonable knowledge that would not normally be available from a CAD system. This information is processed by a non-linear, hierarchical process planner. The hierarchical representation of knowledge is based upon high-level examination of design equations for recognized equation forms and then subsequent application of more traditional production rules. The process planner can handle non-linear cases by backtracking from failed states. The example cases showed a number of successful applications to process planning situations. In the next section some of the outstanding issues will be discussed in terms of future work.

9.FUTURE WORK

The exploratory nature of this work makes it appropriate to end the thesis with a discussion of future work. In trying to develop methods for testing the theory, there were a number of problems that were exposed. Some of the problems were specific to the implementation, and can be considered negligible in terms of judging the success of the method. On the other hand, there were some problems recognized that had no obvious solutions, and will require more research. Both of these classes of problems will be covered in separate sections. Finally a description of some of the novel research possibilities of this system will be discussed.

9.1 EASY PROBLEMS

One of the basic problems relates to the templates used for finding known forms in the Boolean equations. The templates used here are rather rigid in form, whereas it is recognized that better ways exist, but require more investigation. An example of the problem is if we have an equation of the form '(& X Y (~ B) ...etc... (~ C))', it becomes very difficult to refer to the '(~ C)' term if the form of the expression in front is unknown. One possible approach to this would be a constraint based approach where the term is asked for using a list of conditions such as,

```
OPERATION(:) == &
FIND_TERM(:2), OPERATION(:2) == ~
FIND_TERM(:x), OPERATION(:x) == ~
x > 2
```

These conditions could find both terms in the example given. Another useful enhancement, would involve adding conditionals, loops, and argument lists to the rules. As the rules were being developed, it was noted that there was a large amount of repetition. This could be handled easily by allowing conditions to use

argument lists, instead of relying upon a specific variable name for a template match. This would also allow some rule results to be combined when there are only minor differences.

A database of available manufacturing resources (such as machines) would allow the planner to select operations based upon available machines. Subsequently this would allow replanning if one of the resources in the factory became unavailable.

As demonstrated by the test cases there is a definite need for solid modeling based geometrical reasoning. This would support rules that could be more accurate about process selection and operation parameters. It might also be useful to construct a CAD system that also uses the design structure presented in this thesis. The system could eventually become an integral design and process planning system for concurrent engineering.

The Boolean algebra software written does some simplification, and also random manipulation of the equations. This is sufficient for now, but eventually techniques will have to be developed for manipulating the equation into forms suitable for different technologies. Macros could then be added that direct the planner to arrange the equation for a certain form, and fire the more specific rules sets (to reduce planning time). Some work has been done on the manipulation of CSG designs by Shapiro and Vossler [1991]. They discuss finding a canonical CSG form that is the most basic. This is done through examining half spaces and eliminating those not used, or redundant. Work has also been done by Tilove [1984] on algorithms for Null Object Detection and Same Object Detection (NOD and SOD). Other work by Schpitalni and Fisher [1991] attempts to find elements in a CSG tree that interact and group them into features. Their approach could be the basis

for developing strategies for manipulating the equations.

Future work must involve the examination of more test cases from a variety of sources. These test cases would hopefully expose other features required within the system. Test cases would also help determine many of the equation forms that are associated with various technologies. One example that would be ideal is the use of this system to select between machining and casting for parts. As batch sizes and finishes change, the system would be required to select between, and possibly, produce a plan for making a core used to make sand casting molds.

9.2 HARD PROBLEMS

While backwards planning suits some designs, it is not suited to others. This was demonstrated by the process plan created for the digital circuit. Another problem that is caused by backward planning is recovery from a plan error. An instance of this would be when a plan is being executed, and one step fails. A plan that has come from a backwards planner is executed from the last step to the first. This means that in the worst case the planner would have to start planning from the first step, and try to find an intermediate plan state. This problem can become very difficult at this stage. It might be possible to incorporate forward planning strategies as well.

Most solid modelling systems have difficulty defining relationships between primitives. Tolerancing is an excellent example of this. Tolerances can exist between parts of a single primitive, but because of the nature of tolerances they are often stacked up over one or more primitives. There are methods for defining form and dimensional tolerances for a single primitive, but there is an outstanding question of the definition of datums and tolerances through a CSG type representation. Some work has been done by Roy and Liu [1988] when they

incorporated tolerances into a hybrid CSG and B-Rep modeler. Shah and Miller [1990] also incorporate tolerancing into a solid modeling scheme. Gupta and Turner [1993] describe an approach to tolerancing using a B-Rep modeler that subdivides a surface. They can vary the surface to simulate tolerance effects.

Sequencing is somewhat inherent to the nested form of the design model, so it was not considered paramount to proving the thesis. But, eventually the system should be set up so that it may produce non-linear process plans, with sequencing constraints added. Flexplan [Tonshoff, 1989] has proposed a novel approach to modelling non-linear process plans using Petri nets. In addition to these methods, research has also been done by Rho et. al. [1992], Sluga et. al. [1988], ElMaraghy and ElMaraghy [1992].

There are some general problems not approached in this thesis that should be considered in future planning work. One of these is the joining of different jobs for economic reasons. An example of this is the nesting of many sheet metal parts, for different jobs, in one sheet of metal to reduce scrap for a job.

When a solid modeler is incorporated, it is also possible to try a number of other verification procedures. For example Delbressine and Hijink [1991] verify the validity of tool paths by doing a sweep of a tool profile, then doing a simple CSG check for overlap that is in excess of the desired machinable volume.

The reader will recall that the initial design representation uses the r-set Assembly operator. This is done for want of a better operator(s). As research progresses it will become useful to evaluate new representations that emerge. For example, press fits can be modeled by my method, but it does not have a true sense for the implications of material overlap, as would be obtained with a CNRG

operator.

At present the only optimization criteria considered for operation selection is cost. This could easily be expanded to consider other objectives separately, or combinations of objectives. This would require the addition of other declared values such as setup time, run time, environmental impact, etc.

9.3 NOVEL RESEARCH

The well formed nature of this system leads to some potential developments. One of the most exciting would have to be the possibility of adding user interaction. If a plan step is rejected, or the planner is unable to complete a plan, a human user could construct a plan step manually (like decision table rule entry in Hullenkremer [1985]). A new rule could be induced from this plan step, or old rules could be altered. This would allow a system that could be operated without the use of a knowledge engineer to develop the rules. In a slightly different approach, Lu and Zhang [1989] have developed a system that will induce rules from detailed process models. This could also be a novel approach to developing rule sets.

Variant planning does not always receive the recognition it deserves. The benefits of Variant planning are the fast setup times, ease of use, and the ability to deal with plans that are highly specific to a factory or technology. There has been research in the past trying to match CSG representations to GT codes [Kakazu and Okino, 1984]. Using this approach GT codes could be developed for products and parts, then products and parts could be found with similar codes, and their plans recalled and edited. There is also work being done to categorize parts to drive a generative planner [Iwata et. al., 1987].

Developing a CAD system for the design structure would create some possibilities for the addition of some other techniques, such as,

- automatic dimensioning as is done in Yuen et. al. [1988] who chain up dimension dependencies. Work has also been done by Minagawa et. al. [1986],
- addition of functional modelling as is found in Kiriyama et. al. [1991],
- use of CAPP as a concurrent engineering tool [Domazet and Lu, 1991,92]

Other systems may also be added for process plan verification, and operation planning. There is work about the topic in high level terms [Peklenik, 1990] but in more specific terms some possible approaches are listed below,

- assembly sequence planning is of great value. deMello and Sanderson [1991 a, b], Heemskerk [1989], Laperriere and ElMaraghy [1992], Baldwin et. al. [1991], Tonshoff et. al. [1992] suggest such a system. Systems for Disassembly planning have also been done by Lee et. al. [1992].
- automatic generation of NC tool paths. Tsai et. al. [1991], Kao et. al. [1990], Cavendish and Marin [1992], Parkinson [1986], Su and Mukerjee [1991] describe systems to do this,
- automatic generation of NC code for machine forging dies [Eversheim and Cobanoglu, 1989].
- fixture planning, such as the work of Nee et. al. [1992],
- database lookup - such as that for metal forming by Raj and Kumar[1990],
- models of specific processes to verify operations like non-linear optimization for Electro-Discharge Machining (EDM),
- modules for sheet metal bending [Huwiler and Reissner, 1992],
- planning for connecting joints [Neiminen et. al., 1989].
- The problem of mapping from design to production has not been examined formally but it might reveal some interesting results in the context of this thesis.

REFERENCES

Alting, L., Zhang, H., and Lenau, T., "XPLAN - An Expert Process Planning System and its Further Development", 27th Int'l Matador Conf. Proceedings, Manchester, UK, 1988, pp. 155-163.

Alting, L., and Zhang, H., "Computer Aided Process Planning: The State-of-the-Art Survey", IIE Integrated Systems Conf. Proceedings, St. Louis, Missouri, 1988. Also appears in the International Journal of Production Research, Vol.27, 1989, pp. 533-85.

Anderson, D. C., and Chang, T., "Automated Process Planning Using Object-Oriented Feature Based Design", Int'l Symposium on Advanced Geometric Modelling for Engineering Applications, Berlin, FRG, 1989, pp. 233-246.

Arbab, F., "Set Models and Boolean Operations for Solids and Assemblies", IEEE Computer Graphics and Applications, November 1990, pp. 76-86.

Arikan, M.A.S., and Totuk, O.H., "Design by Using Machining Operations", Annals of the CIRP, Vc.. 41, 1992, pp. 185-188.

Austin, B. L., "Computer Aided Process Planning for Mach'ned Cylindrical Parts", Proceedings of ASME Winter Annual Meeting, Anaheim, CA, 1987, pp. 359-366.

Baldwin, D., Abell, T.E., Lui, M.-C.M., DeFazio, T.L., Whitney, D.E., "An Integrated Computer Aid for Generating and Evaluating Assembly Sequences for Mechanical Products", IEEE Transactions on Robotics and Automation, Vol. 7, No. 1, February, 1991, pp. 78-94.

Barkocy, B. E., and Zdeblick, W.J., "A Knowledge-Based System for Machining Operation Processing", Autofact 6 Conf. Proceedings, Anaheim, CA, 1984, pp. 2-11 to 2-25.

Braid, I. C., "From Geometric to Product Modelling", in Software for Discrete Manufacturing, editors J. P. Crestin, J. F. McWaters, Elsevier Science Publishers, 1986, pp. 123-133.

Brun, J. M., "B-Rep to CSG Conversion as a Feature Recognition Process", in Advanced Geometric Modelling for Engineering Applications, editors. F.-L. Krause, H. Jansen, International GI-IFIP Symposium 89, 1989, pp. 201-214.

Cachia, T., and Vajpayee, S., "Process Planning with a Micro-Computer", Proceedings of the CASA/RI/SME 5th Canadian CAD/CAM and Robotics Conference, June 1986, pp. 14-26.

Carringer, R., "On the Implementation of Group Technology Support Software for Process Planning", Autofact 6 Conference Proceedings, Anaheim, CA, 1984, pp. 14-10 to 14-30.

Cavendish, J. C., and Marin, S. P., "Feature-Based Surface Design and Machining", IEEE Computer Graphics and Applications, September 1992, pp. 61-68.

Chang, T-C, Expert Process Planning for Manufacturing, Addison-Wesley Publishing Co., 1990.

Chang, T., and Wysk, R. A., An Introduction to Automated Process Planning Systems, Prentice-Hall Inc., New Jersey, 1985.

Chang T. C., Wysk, R. A., and Davis, R. P., "Interfacing CAD and CAM - A Study in Hole Design", Computing and Industrial Engineering, Vol 6., No. 2, 1982, pp. 91-102.

Chang, T., and Terwilliger, J., "A Rule Based System for Printed Wiring Assembly Process Planning", International Journal of Production Research, Vol. 25, No. 10, 1987, pp. 1465-1482.

Chryssolouris, G., Chan, S., and Cobb, W., "Decision Making on the Factory Floor: An Integrated Approach to Process Planning and Scheduling", Robotics & Computer-Integrated Manufacturing, Vol. 1, No. 3/4, 1984, pp. 315-319.

Chryssolouris, G., and Wright, K., "Knowledge-Based Systems in Manufacturing", Annals of the CIRP, Vol. 35, No. 2, 1986.

Cohen, P.R., and Feigenbaum, E.A., The Handbook of Artificial Intelligence, William Kaufmann, Inc., Vol. 3, 1982, pp. 523-562.

Darbyshire, I., and Davies, B., "EXCAP: An Expert System s' Approach to Recursive Process Planning", 16th CIRP Int'l Seminar on Manufacturing Systems, Tokyo, Japan, 1984, pp. 73-82.

Davies, B., Darbyshire, I.and Wright, A., "The Integration of Process Planning with CAD CAM Including the Use of Expert Systems", Proceedings of Int'l Conf. on Computer Aided Production Engineering, Edinburgh, Scotland, 1986, pp. 35-40.

Davies, B.J., Darbyshire, I.L., Wright, A.J., and Zhang, K.F., "IKBS Process Planning System for Rotational Parts", Intelligent Manufacturing Systems II, Elsevier Science Publishers, Netherlands, 1988, pp. 27-38.

Delbressine, F. L. M., and van der Wolf, A. C. H., "Integrated Design and Manufacturing", Annals of the CIRP, Vol 39, No. 1, 1990, pp. 149-152.

Delbressine, F. L. M., and Hijink, J. A. W., "Discrete Part Design by Taking Manufacturing Restrictions into Account", Annals of the CIRP, Vol. 40, No. 1, 1991, pp. 171-173.

Delbressine, F. L. M., On The Integration of Design and Manufacturing, a doctoral thesis for the University of Eindhoven, Germany, 1989.

Delbressine, F.L.M., deGroot, R., and van der Wolf, A.C.H., "On the Automatic Generation of Set-Ups Given a Feature-Based Design Representation", Annals of the CIRP, 1993, pp. 527-530.

de Mello, L., and Sanderson, A., "A Correct and Complete Algorithm for the Generation of Mechanical Assembly Sequences", IEEE Transactions on Robotics and Automation, Vol. 7, No. 2, April, 1991, pp. 228-240.

de Mello, L., and Sanderson, A., "Representations of Mechanical Assembly Sequences", IEEE Transactions on Robotics and Automation, Vol. 7, No. 2, April, 1991, pp. 211-227.

Descotte, Y., and Latombe, J., "GARI: An Expert System for Process Planning", in Solid Modelling by Computers, edited by M.S. Pickett and J.W. Boyse, Plenum Press, 1984, pp. 329-344.

deVin, L.J., deVries, J., Streppel, A.H., and Kals, H.J.J., "PART-S, a CAPP System for Small Batch Manufacturing of Sheet Metal Components", 24th CIRP International Seminar on Manufacturing Systems, June 11-12, 1992, pp. 171-182.

Dieter, G. E., Engineering Design: A Materials and Processing Approach, McGraw-Hill, 1991.

Dini, G., and Santochi, M., "Automated Sequencing and Subassembly Detection in Assembly Planning", Annals of the CIRP, Vol. 41, 1992, pp. 1-4.

Domazet, D. S., and Lu, S, C.-Y., "Integration of Product Design and Process Planning for Manufacturability Analysis", Knowledge-Based Engineering Systems Research Laboratory Annual Report, 1991, University of Illinois. pp. 33-40.

Domazet, D.S., and Lu, S.C., "Concurrent Design and Process Planning of Rotational Parts", Annals of the CIRP, Vol. 41, 1992, pp. 181-184.

Dunn, M. S., and Mann, W.S., "Computerized Production Process Planning for Machined Cylindrical Parts", Proceedings of the 15th Numerical Control Society Annual Meeting and Technical Conference, 1978, pp. 288-303.

ElMaraghy, H. A., "Intelligent Product Design and Manufacture", in Artificial Intelligence in Design, edited by T. Pham, Addison-Wesley, 1991, pp. 147-168.

ElMaraghy, H.A., and ElMaraghy, W.H., "Bridging the Gap Between Process Planning and Production Planning and Control", Proceedings of the 24th CIRP International Seminar on Manufacturing Systems, Copenhagen, 1992, pp. 1-10.

ElMaraghy, H.A., "Evolution and Future Perspectives of CAPP", Keynote paper in the Annals of the CIRP, Vol. 42, No. 2, 1993.

ElMaraghy, W.H., "Integrating Assembly Planning and Scheduling - CAPP-Related Issues", Annals of the CIRP, Vol. 41, 1992, pp. 11-14.

ElMaraghy, W.H., and Jack, H., "Facilitating Rapid Product Realization with an Intelligent CAPP System", Keynote paper Proceedings of IFIP Conference on World Class Manufacturing, Phoenix, Arizona, September, 1993.

Emerson, C., and Ham, I., "An Automated Coding and Process Planning System Using a DEC PDP-10", Computing and Industrial Engineering, Vol. 6, No. 2, 1982, pp. 159-168.

Esprit Project, "Feature Modelling as a link for Part and Process Modelling", Document # R.FTFG-I.IR1/001, Version 1.0, 1990.

Eversheim, W., Fuchs, H., and Zons, J.H., "Automatic Process Planning with Regard to Production by Application of the System AUTAP for Control Problems", Proceedings of the 12th CIRP International Seminar on Manufacturing Systems, Belgrade, 1980.

Eversheim, W., and Schulz, J.C., "Survey of Computer Aided Process Planning Systems", Proc. of the 1st CIRP Workshop on CAPP, Vol.34, No. 2, 1985.

Eversheim, W., and Cobanoglu, M., "Integrated Process Planning and Part Programming System for Machining Forging Dies", CIRP Int'l Workshop on Computer Aided Process Planning, Hanover, FRG, 1989, pp. 37-53.

Fujita, S., Okamoto, T., Oouchi, S., Oshima, M., "Study of Practical Computer Aided Process Planning Based on an Expert System", Proceedings of the 7th Int'l IFIP/ IFAC Conf. on Software for CIM, Dresden, GDR, 1988, pp. 589-599.

Gammons, R. A., "ESKIMO: An Expert System for KODAK Injection Molding Operations", in Artificial Intelligence in Design, edited by C. Tong, D. Sriram, Vol. 3, 1992, pp. 81-104.

Gevarter, W.B., Intelligent Machines: An Introductory Perspective of Artificial Intelligence and Robotics, Prentice Hall, Inc., 1985, pp. 67-81

Goldfeather, J., Hultquist, J.P.M., and Fuchs, H., "Fast Constructive Solid Geometry Display in the Pixel-Powers Graphics System", Computer Graphics, Vol. 20, No. 4, Aug. 1986, pp. 107-116.

Green, C., "The Application of Theorem-Proving to Question Answering Systems", IJCAI-1, 1969, pp. 219-237.

Groppetti, R., and Semeraro, Q., "Generative Approach to Computer Aided Process Planning", Proceedings of Int'l Conf. on Computer Aided Production Engineering, Edinburgh, Scotland, 1986, pp. 179-189.

Gupta, S., and Turner, J. U., "Variational Solid Modelling for Tolerance Analysis", IEEE Computer Graphics and Applications, May 1993, pp. 64-74.

Halevi, G., and Weill, R., "Algorithmic Systems versus Expert Systems for CAPP", 24th CIRP International Seminar on Manufacturing Systems, June 11-12, 1992, pp. 11-20.

Ham, I., and Lu, S., "New Developments of CAPP in U.S.A. and Japan", CIRP Int'l Workshop on Computer Aided Process Planning, Hanover, FRG, 1989, pp. 1-23.

Ham, I., and Lu, S., "Computer Aided Process Planning: The Present and the Future", Annals of the CIRP, Vol. 37, January, 1988, pp. 591-602.

Han, C., Li, J., and Ham, I., "Development of an In-House Computer Automated Process Planning System Based on Group Technology Concept", Proceedings of the 15th North American Research Conference, May 1987, pp. 41-46.

Hartquist, E.E., and Marisa, H.A., "PADL-2 Users Manual", The Production Automation Project, The University of Rochester, Rochester, New York, 1985.

Held, H., and Juttner, G., "GENOA: Feature Based Generation and Optimization of Process Plans", in Complex Machining and AI-Methods, edited by D.Kochan, Elsevier Science Publishers, 1991, pp. 85-97.

Henderson, M., and Chang, G., "FRAPP: Automated Feature Recognition and Process Planning from Solid Model Data", ASME Int'l Conf. on Computers in Engineering, San Francisco, CA, 1988, pp. 529-536.

Henderson, M. R., "Extraction and Organization of Form Features", in Software for Discrete Manufacturing, editors J. P. Crestin, J. F. McWaters, Elsevier Science Publishers, 1986, pp. 547-557.

Hullenkremer, M., "Computer Aided Process Planning with Help of a Decision table Generator", Proceedings of the 8th Int'l Conf. on Production Research, Stuttgart, FRG, 1985, pp. 43-46.

Hummel, K., "Coupling Rule-Based and Object-Oriented Programming for the Classification of Machine Features", ASME Conf. on Computers in Engineering, Anaheim, CA, 1989, pp. 409-418.

Hummel, K., and Brooks, S., "Symbolic Representation of Manufacturing Features for an Automated Process Planning System", Proceedings of ASME Winter Annual Meeting, Anaheim, CA, 1986, pp. 233-243.

Hummel, K.E., and Brooks, S.L., "Using Hierarchically Structured Problem-Solving Knowledge in a Rule-Based Process Planning System", in Expert Systems and intelligent Manufacturing, edited by Oliff, M.D., Elsevier Science Publishing Co., 1988, pp. 120-137.

Huwiler, B., and Reissner, J., "CAPP Based on Rules and Algorithms in the Manufacturing of Bent Sheet Parts", 24th CIRP International Seminar on Manufacturing Systems, June 11-12, 1992, pp. 189-198.

Institute of Advanced Manufacturing Sciences, MetCAPP: User's Guide, UNIX Version, Cincinnati, Ohio, 1990.

Inui, M., Suzuki, H., Kimura, F., and Sata, T., "Generation and Verification of Process Plans Using Dedicated Models of Products in Computers", Proceedings of ASME Winter Annual Meeting, Anaheim, CA, 1986, pp. 275-286.

Iwata, K., and Sugimura, N., "A Knowledge Based Computer Aided Process Planning System for Machine Parts", 16th CIRP Int'l Seminar on Manufacturing Systems, Tokyo, Japan, 1984, pp. 83-92.

Iwata, K., and Sugimura, N., "An Integrated CAD/CAPP System with Know-hows on machining Accuracies of Parts", Transactions of the ASME, Vol. 109, May, 1987, pp. 128-133. Also appears in the Proceedings of ASME Winter Annual Meeting, Miami Beach FL, 1985, pp. 121-130.

Iwata, K., Sugimura, N., and Fukada, Y, "Knowledge-Based Flexible Part Classification System for CAD/CAM", Annals of the CIRP, Vol. 36, 1987, pp. 317-320.

Iwata, K., "Knowledge Based Computer Aided Process Planning", Intelligent Manufacturing Systems II, Elsevier Science Publishers, Netherlands, 1988, pp. 3-25.

Jonkers, F.J.C.M., and Kals, H.J.J., "The Development of the Software Architecture for PART, A Computer Aided Process Planning System", 24th CIRP International Seminar on Manufacturing Systems, June 11-12, 1992, pp. 71-80.

Joseph, A. T., "Knowledge Acquisition and Implementation in an Expert Process Planning System", PhD Thesis, Dept. of Mech. Engineering, University of Manchester Institute of Science and Technology, 1989.

Joseph, A.T., and Davies, B.J., "Implementation of a 'Backwards Planning' Technique in the Automation of Process Planning", International Journal of Advanced Manufacturing Technologies, Springer-Verlag, 1990, pp. 321-344.

Joshi, S., Vissa, N. and Chang, T., "Expert Process Planning System with Solid Model Interface", Int'l Journal of Production Research, Vol. 26, No. 5, 1988, pp. 863-885.

Joshi, S., and Chang, T.C., "Graph-Based Heuristics for Recognition of Machined Features from a 3D Solid Model", Computer-Aided Design, Vol.20, No. 2, March 1988, pp. 59-66.

Kamvar, E., and Melkanoff, M. A., "Automatic Generation of GT Codes for Rotational Parts From CAD Drawings", in Software for Discrete Manufacturing, editors J. P. Crestin, J. F. McWaters, Elsevier Science Publishers, 1986, pp. 259-272.

Kanumury, M., Shah, J., and Chang, T., "An Automated Process Planning System for a Quick Turnaround Cell - An Integrated CAD and CAM System", ASME Proceedings of the USA-Japan Symposium on Flexible Automation, Minneapolis, Minnesota, 1988, pp. 861-868.

Kao, C., Shy, L., Hseih, S.-T., Chen, C.-M.., "Post Processor for CAPP", CSME Mechanical Engineering Forum, Toronto, Ont., 1990, pp. 7-12.

Karinthi, R., and Nau, D., "Geometric Reasoning as a Guide to Process Planning", ASME Int'l Conf. on Computers in Engineering, Anaheim, CA, 1989, pp. 609-616.

Kimura, F., "High Quality Product Realization Through the Tight Integration of Product Design and Process Planning", CIRP Int'l Workshop on Computer Aided Process Planning, Hanover, FRG, 1989, pp. 1-10.

Kiriyama, T., Tomiyama, T., and Yoshikawa, H., "The Use of Qualitative Physics for Integrated Design Object Modelling", Design Theory and Methodology, ASME, Vol.31, 1991, pp. 53-60.

Kjellberg, T., "Product Modelling - a Tool for Design and Manufacturing Systems", the Royal Institute of Technology, Stockholm Sweden, 1990.

Knutilla, A. J., and Park, B. C., "Computer Managed Process Planning for Cylindrical Parts", CAD/CAM Robotics and Factories of the Future '90, Springer Verlag, Vol. 2, 1990, pp. 153-165.

Korf, R.E., "Planning as Search: A Quantitative Approach", Artificial Intelligence, 1987, pp. 65-88.

Krause, F.L., Ulbrich, A., and Woll, R., "Feedback for Process Planning as an Approach to Intelligent Quality Assurance", Complex Machining and AI-Methods, edited by D.Kochan, Elsevier Science Publishers, 1991, pp. 147-159.

Kruth, J.P., and Detand, J., "A CAPP System for Nonlinear Process Plans", Annals of the CIRP, Vol. 41, 1992, pp. 489-492.

Kusiak, A., "Process Planning: A Knowledge-Based and Optimization Perspective", IEEE Transactions on Robotics and Automation, Vol. 7, No. 3, June, 1991, pp. 257 - 266.

Lamkemeyer, U., Schmidt, B., and Detand, J., "FLEXPLAN: Description of the Main Modules", CIM-Fabrik, Hannover, Germany, March, 1991, pp. 1-14.

Laperriere, L., and ElMaraghy, H.A., "Planning of Products Assembly and Disassembly", Annals of the CIRP, Vol. 41, 1992, pp. 5-9.

Laperriere, L., and ElMaraghy, H.A., "Generation and Evaluation of Assembly Sequences in Concurrent Engineering", 24th CIRP International Seminar on Manufacturing Systems, June 11-12, 1992, pp. 159-170.

Lee, K.I., Lee, J.W., and Lee, J.M., "Pattern Recognition and Process Planning Prismatic Workpieces by Knowledge Based Approach", Annals of the CIRP, Vol. 38, No. 1, 1989, pp. 485-488.

Lee, S., and Wang, H. P., "Optimal Process Planning for Robotic Assembly", 5th International Conference on CAD/CAM, Robotics and Factories of the future (CARS and FOF'90) Proceedings, Springer Verlag, 1990, pp. 179-184.

Lee, Y.C., and Fu, K. S., "Machine Understanding of CSG: Extraction and Unification of Manufacturing Features", IEEE Computer Graphics and Applications, Vol. 7, No. 1, 1987, pp. 20-32.

Lee, Y., and Jea, K., "A New CSG Tree Construction Algorithm for Feature Representation", ASME Int'l Conf. on Computers in Engineering, San Francisco, CA, 1988, pp. 521-528.

Lee, Y., Kumara, S.R.T., and Ham, I., "Individual and Group Disassembly Plan Sequence Generation: Theory and Implementation", 24th CIRP International Seminar on Manufacturing Systems, June 11-12, 1992, pp. 147-157.

Lenau, T., "Integrating Process Planning with Product Design (Design / CAPP Integration)", 24th CIRP International Seminar on Manufacturing Systems, June 11-12, 1992, pp. 21-34.

Ling, Z. K., and Chase, T. R., "Simulation of Pressbrake Bending Using Constructive Solid Geometry", Proceedings of the ASME Computers in Engineering Conference, 1987, pp. 321-326.

Lu, S., A. Herman and Arnold, K., "TOPS: An Intelligent Associate for Turning Operation Planning", the Knowledge Bases Systems Laboratory, the Department of Mechanical and Industrial Engineering, the University of Illinois at Urbana-Champagne, 1991.

Lu, S., and Zhang, G., "A Combined Inductive Learning and Experimental Design Approach to Manufacturing Operation Planning", Journal of Intelligent Manufacturing Systems, August 1989.

Mantyla, M., "Representation of Process Planning Knowledge for Part Families", Annals of the CIRP, Vol. 43, No. 1, 1993, pp. 561-564

Marefat, M., Malhotra, S., and Kashyap, R. L., "Object-Oriented Intelligent Computer-Integrated Design, Process Planning and Inspection", Computer, March 1993, pp. 54-65.

McNeilly, B., An Object-Oriented Approach to Computer Aided Process Planning Integration, a thesis presented at the University of Western Ontario, January 1993.

Mehta, N. K., Pandey, P. C., and Mohan, L., "Development of Variant Computer Aided Process Planning System for Rotational Parts", Proceedings of the 14th All India Machine Tool Design and Research Conference, 1990, pp. 507-512.

Mendelson, E., Shaum's Outline of Theory and Problems of Boolean Algebra and Switching Circuits, McGraw-Hill Book Company, 1989.

Milacic, V., and Urosevic, M., "SAPT- Knowledge based CAPP System", Robotics and Computer Integrated Manufacturing, Vol. 4, No. 1/2, 1988, pp. 69-76.

Milacic, V.R., Urosevic, M., Veljovic, A., Miler, A., and Race, I., "SAPT - EXAPT System Based on Hybrid Concept of Group Technology", Intelligent Manufacturing Systems II, Elsevier Science Publishers, Netherlands, 1988, pp. 39-51.

Minagawa, M., Okino, N., Kakazu, Y., "Development of Full Automatic Dimensioning System Based on 3D Solid Geometry", in Software for Discrete Manufacturing, editors J. P. Crestin, J. F. McWaters, Elsevier Science Publishers, 1986, pp. 145-154.

Muchnicki, P. F., "Process Planning", in Organizing Data for CIM Applications, edited by C. S. Knox, Marcel Dekker Inc., 1987, pp. 265-288.

Nau, D., and Chang, T., "A Knowledge Based Approach to Generative Process Planning", Proceedings of ASME Winter Annual Meeting, Miami Beach FL, 1985, pp. 65-71.

Nau, D., and Grey, M., "SIPS: An Application of Hierarchical Knowledge Clustering to Process Planning", Proceedings of ASME Winter Annual Meeting, Anaheim, CA, 1986, pp. 219-225.

Nee, A.Y.C., Kumar, A.S., Prombanpong, S., and Puah, K.Y., "A Feature-Based Classification Scheme for Fixtures", Annals of the CIRP, Vol. 41, 1992, pp. 189-192.

Newell, A., and Simon, H.A., Human Problem Solving, Prentice Hall, Englewood-Cliffs, NJ, 1972.

Niebel, B.W., "Mechanized Process Selection for Planning New Designs", ASME Paper, No. 737, 1965.

Nieminen, J., Kanerva, J., and Mantyla, M., "Feature-Based Design of Joints", Int'l Symposium on Advanced Geometric Modelling for Engineering Applications, Berlin, FRG, 1989, pp. 332-341.

Nilsson, N.J., Principles of Artificial Intelligence, Tioga Publishing Co., 1980, pp. 282-287.

Nolen, J. Computer-Automated Process Planning for World-Class Manufacturing, Marcel Dekker Inc., New York, 1989.

Nylund, K., and Novak, A., "Feature Recognition in Process Planning for Turning", 24th CIRP International Seminar on Manufacturing Systems, June 11-12, 1992, pp. 91-100.

Okino, N., Kakazu, Y., and Kubo, H., "TIPS-1: Technical Information Processing System, for Computer-Aided Design, Drawing and Manufacturing", in Computer Languages for Numerical Control, North-Holland, 1973, pp. 141-150.

Pande, S., and Palsule, N., "GCAPPS - A Computer Assisted Generative Process Planning System for Turned Components", Computer-Aided Engineering Journal, August, 1988, pp. 163-168.

Pande, S., and Walvekar, M., "PC-CAPP - A Computer Assisted Process Planning System for Prismatic Components", Computer-Aided Engineering Journal, August, 1989, pp. 133-138.

Parkinson, A., "The Use of Solid Models in Build as a Database for NC Machining", in Software for Discrete Manufacturing, editors J. P. Crestin, J. F. McWaters, Elsevier Science Publishers, 1986, pp. 175-183.

Parks, C., Graves, G. and Koonce, D., "Computer Aided Process Planning with CMPP", Proceedings of 11th Annual Conference on Computers and Industrial Engineering, Vol. 17, Nos. 1-4, 1989, pp. 246-251.

Pavey, S., Hailstone, S., and Pratt, M., "An Automated Interface between CAD and Process Planning", Int'l Conf. on Computer Aided Production Engineering, Edinburgh, Scotland, 1986, pp. 191-194.

Peklenik, J., "An Analysis of the Manufacturing Process Topology and the Process Features as basis for the CAPP", 22nd CIRP Int'l Seminar on Manufacturing Systems, Enschede, Netherlands, 1990, pp. 153-156.

Phillips, R., Arunthavanathan, V., and Zhou, X., "Symbolic Representation of CAD Data for Artificial Intelligence Based Process Planning", Proceedings of ASME Winter Annual Meeting, Miami Beach FL, 1985, pp. 31-42.

Phillips, R. H., and Mouleeswaran, C.B., "A Knowledge-Based Approach to Generative Process Planning", Proceedings of the CASA/SME Autofact '85 Conference, 1985.

Pinte, J., "Computer Aided Process Planning", Proceedings of CAD/CAM 87, Teknologisk Institut, Denmark, Oct., 1987.

Prabhu, B. S., Pande, S. S., and Biswas, S., "EXPLAIN - Expert System for Computer Assisted Process Planning of Turned Components", Proceedings of the 14th All India Machine Tool Design and Research Conference, 1990, pp. 477-481.

Preiss, K., and Kaplansky, E., "Automated CNC Milling by Artificial Intelligence Methods", Autofact 6 Conference Proceedings, Anaheim, CA, 1984, pp. 2-41 to 2-55.

Raj, K. H., and Kumar, V. M., "An Expert System for Metalforming", 5th International Conference on CAD/CAM, Robotics and Factories of the Future (CARS and FOF'90) Proceedings, Springer-Verlag, Vol. 2, 1990, pp. 173-178.

Ramulu, M., See, H. W., and Wang, D. H., "An Application of Non-linear Goal Programming in Electrodischarge Machining of Composite Material", 5th International Conference on CAD/CAM, Robotics and Factories of the Future (CARS and FOF'90) Proceedings, Springer-Verlag, Vol. 2, 1990, pp. 166-172.

Rembold, U., Blume, C., and Dillman R., Computer-Integrated Manufacturing Technology and Systems, Marcel Dekker Inc., New York, 1985.

Requicha, A.A.G., and Vandenbrande, J., "Automated Systems for Process Planning and Part Programming", Artificial Intelligence: implications for computer integrated manufacturing, IFS Publications Ltd., 1988, pp. 301-326.

Requicha, A. A. G., and Rossignac, J. R., "Solid Modelling and Beyond", IEEE Computer Graphics and Applications, September 1992, pp. 31-44.

Rho, H.M., Geelink, R., van't Erve, A.H., and Kals, H.J.J., "An Integrated Cutting Tool Selection and Operation Sequencing Method", Annals of the CIRP, Vol. 41, 1992, pp. 517-520.

Rich, E., Artificial Intelligence, McGraw-Hill Book Company, 1983, pp. 247-277.

Rossignac, J.R., and Requicha, A.A.G., "Constructive Non-Regularized Geometry", Computer-Aided Design, Vol.23, No. 1, February 1991, pp. 21-32.

Roy, U., and Liu, C., "Feature-Based Representational Scheme of a Solid Modeler for Providing Dimensioning and Tolerancing Information", Robotics and Computer Integrated Manufacturing, Vol. 4, No. 3/4, 1988, pp. 335-345.

Ruf, T., and Jablonski, S., "Flexible and Reactive Integration of CIM Systems: A Feature-Based Approach", CSME Mechanical Engineering Forum, Toronto, Ont., 1990, pp. 31-36.

Sacerdoti, E., "The Nonlinear Nature of Plans", International Joint Conference on Artificial Intelligence, 1975, pp. 206-214.

Sacerdoti, E., A Structure for Plans and Behavior, Elsevier North-Holland Inc., 1977.

Schneider, R., Kriegel, H. P., Seeger, B., and Heep, S., "Geometry-based similarity retrieval of rotational parts", IEEE Computer, 1989, pp. 150-160.

Shah, J. J., and Miller, D. W., "A Structure for Supporting Geometric Tolerances in Product Definition Systems for CIM", Manufacturing Review, Vol. 3, No. 1, March 1990, pp. 23-31.

Shanbhogue, H. B. V., Ravichandran, C., and Prabhu, V., "Design of a Manufacturing Database for CAPP", Proceedings of the 14th All India Machine Tool Design and Research Conference, 1990, pp. 513-518.

Shapiro, V., and Vossler, D.L., "Construction and Optimization of CSG Representations", Computer-Aided Design, Vol.23, No. 1, February 1991, pp. 5-20.

Shpitalni, M., and Fischer, A., "CSG Representation as a Basis for Extraction of Machining Features", Annals of the CIRP, Vol. 40, No. 1, 1991, pp. 157-160.

Shyu, J., and Chen, Y.W., "A Mini CIM System for Turning", Annals of the CIRP, Vol. 36, 1987, pp. 277-280.

Sluga, A., Butala, P., Lavrac, N., and Gams, M., "An Attempt to Implement Expert System Techniques in CAPP", Robotics and Computer Integrated Manufacturing, Vol. 4, No. 1/2, 1988, pp. 77-82.

Spatial Technology Inc., ACIS Geometric Modeling: Interface Guide, Boulder Colorado, 1991.

Stefik, M., "Planning with Constraints (MOLGEN: Part 1)", Artificial Intelligence, 1981a, pp. 111-140.

Stefik, M., "Planning and Meta-Planning (MOLGEN: Part 2)", Artificial Intelligence, 1981b, pp. 141-170.

Stewart, N. F., 'Solid Modelling", in Scientific Visualization and Graphics Simulation, editor D. Thalmann, John Wiley and Sons Ltd., 1990, pp. 43-60.

Strousap, B., "An Overview of C++", ACM Sigplan Notices, October 1986, pp. 7-18.

Su, C-J, and Mukerjee, A., "Automated Machinability Checking for CAD/CAM", IEEE Transactions on Robotics and Automation, Vol. 7, No. 5, October 1991, pp. 691-699.

Sussman, G.J., "The Virtuous Nature of Bugs", Proceedings of the AISB Summer Conference, 1974, pp. 224-237.

Tang, K., and Woo, T., "Algorithmic Aspects of Alternating Sum of Volumes. Part 1: Data Structure and Difference Equations", Computer-Aided Design, Vol 23, No. 5, June 1991, pp. 357-366.

Tang, K., and Woo, T., "Algorithmic Aspects of Alternating Sum of Volumes. Part 2: Nonconvergence and its Remedy", Computer-Aided Design, Vc! 23, No. 6, August 1991, pp. 435-443.

Tate, A., "Generating Project Networks", IJCAI, 1977, pp. 888-893

Tilove, R. B., and Requicha, A. A. G., "Closure of Boolean Operations on Geometric Entities", Computer-Aided Design, September 1980, pp. 217-220.

Tilove, R.B., "A Null-Object Detection Algorithm for Constructive Solid Geometry", Communications of the ACM, Vol. 27, No. 7, July 1984, pp. 684-694.

Tonshoff, H. K., Beckendorff, U., and Anders, N., "FLEXPLAN - A Concept for Intelligent Process Planning and Scheduling", CIRP International Workshop on Computer Aided Process Planning, Hanover, FRG, 1989, pp. 87-106.

Tonshoff, H.K., Menzel, E., and Park, H.S., "A Knowledge-Based System for Automated Assembly Planning", Annals of the CIRP, Vol. 41, 1992, pp. 19-24.

Tsai, M., Takata, S., Inui, M., Kimura, F., Sata, T., "Operation Planning Based on Cutting Process Models", Annals of the CIRP, Vol. 40, January, 1991, pp. 95-98.

Tsatsoulis, C, and Kashyap, R.L., "Planning and Its Applications to Manufacturing", in Artificial Intelligence: Manufacturing Theory and Practice, Institute of Industrial Engineers, 1988, pp. 193-223.

van Houten, F., and Kals, H., "Round, A Flexible Technology Based Process and Operations Planning System for NC-Lathes", 16th CIRP Int'l Seminar on Manufacturing Systems, Tokyo, Japan, 1984, pp. 102-111.

van Houten, F.J.A.M., van't Erve, A.H., Jonkers, F.J.C.M., and Kals, H.J.J., "PART, a CAPP System With a Flexible Architecture", Proc. of CIRP Int. Workshop on CAPP, 1989, pp.57-69.

van 't Erve, A., and Kals, H., "XPLANE, A Knowledge Base Driven Process Planning Expert System", Int'l Conf. on Computer Aided Production Engineering, Edinburgh, Scotland, 1986, pp. 41-46.

Varady, T., "Operations to Integrate Free-form Surfaces into the 'BUILD' Volumetric Modeller", in Software for Discrete Manufacturing, editors J. P. Crestin, J. F. McWaters, Elsevier Science Publishers, 1986, pp. 185-196.

Vere, S.A., "Planning in Time: Windows and Durations for Activities and Goals", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No.3, 1983, pp. 246-267.

von Rimscha, M., "Feature Modelling and Assembly Modelling - A Unified Approach", in Advanced Geometric Modelling for Engineering Applications, editors, F. L. Krause, H. Jansen, 1989, pp. 190-200.

Waldinger, R., "Achieving Several Goals Simultaneously", Machine Intelligence, 1977, pp. 94 136.

Wang, H., and Wysk, R.A., "Applications of Microcomputers in Automated Process Planning", Proceedings of the CASA/SME Journal of Manufacturing Systems, Volume 5, No. 2, 1984-1985, pp. 47-55.

Wang, H., and Wysk, R., "Microcomputer-Based Process Planning Systems", IIE Fall Industrial Engineering Conf., Chicago, IL, 1985, pp. 145-153.

Wang, H., and Wysk, R., "MICRO-GEPPS - A Microcomputer-Based Process Planning System", Proceedings of ASME Winter Annual Meeting, Miami Beach, FL, 1985, pp. 139-149.

Wang, H., and Wysk, R.A., "Expert Systems Methods for Process Planning", Artificial Intelligence: Manufacturing Theory and Practice, Institute of Industrial Engineers, 1988, pp. 535-564.

Warnecke, H., Mayer, C., and Muthsam, H., "New Tools in CAPP", CIRP International Workshop on Computer Aided Process Planning, Hanover, FRG, 1989, pp. 169-180.

Wei, Y., Fischer, G., and Santos, J., "A Concurrent Engineering Design Environment for Generative Process Planning Using Knowledge Based Decisions", ASME 16th Design Automation Conf., Chicago, IL, 1990, pp. 35-45.

Wilensky, R., Planning and Understanding, Addison-Wesley Publishing Co, 1983.

Willis, D., Donaldson, I.A., Ramage, A.D., Murray, J.L., and Williams, M.H., "A Knowledge-Based System for Process Planning Based on a Solid Modeler", Computer Aided Engineering Journal, February, 1989, pp. 21-26.

Wilson, P., Features Interim Report III, Lucas Group Services Ltd., Lucas Research Centre, West Midlands, B90 4JJ, England, October, 1983.

Woo, T.C., "Computer Aided Recognition of Volumetric Designs", in Advances in Computer-Aided Manufacture, edited by D. McPherson, North-Holland Publishing Co., 1977, pp. 121-136.

Woodwark, J.R., "Some Speculations on Feature Recognition", Computer-Aided Design, Vol.20, No. 4, May 1988, pp. 189-196.

Yuen, M.M.F., Tan, S.T., and Yu, K.M., "Scheme for Automatic Dimensioning of CSG Defined Parts", Computer-Aided Design, Vol.20, No. 3, April 1988, pp. 151-159.

Zdeblick, W.J., "CUTPLAN/CUTTECH: A Hybrid Computer-Aided Process and Operation Planning System", Proceedings of the 1st CIRP Seminar on CAPP, Paris, Jan. 22-23, 1985, pp.61-64.

Zhang, H., and Alting, L., "XPLAN-R: An Expert Process Planning System for Rotational Components", Proc. of IEE 1988 Integrated Systems Conference, St. Louis, Missouri.

Zhang, H., and Alting, L., "Introduction to an Intelligent Process Planning System for Rotational Parts", Proc. of the Advances in Manufacturing Systems Engineering Symposium, 1988.