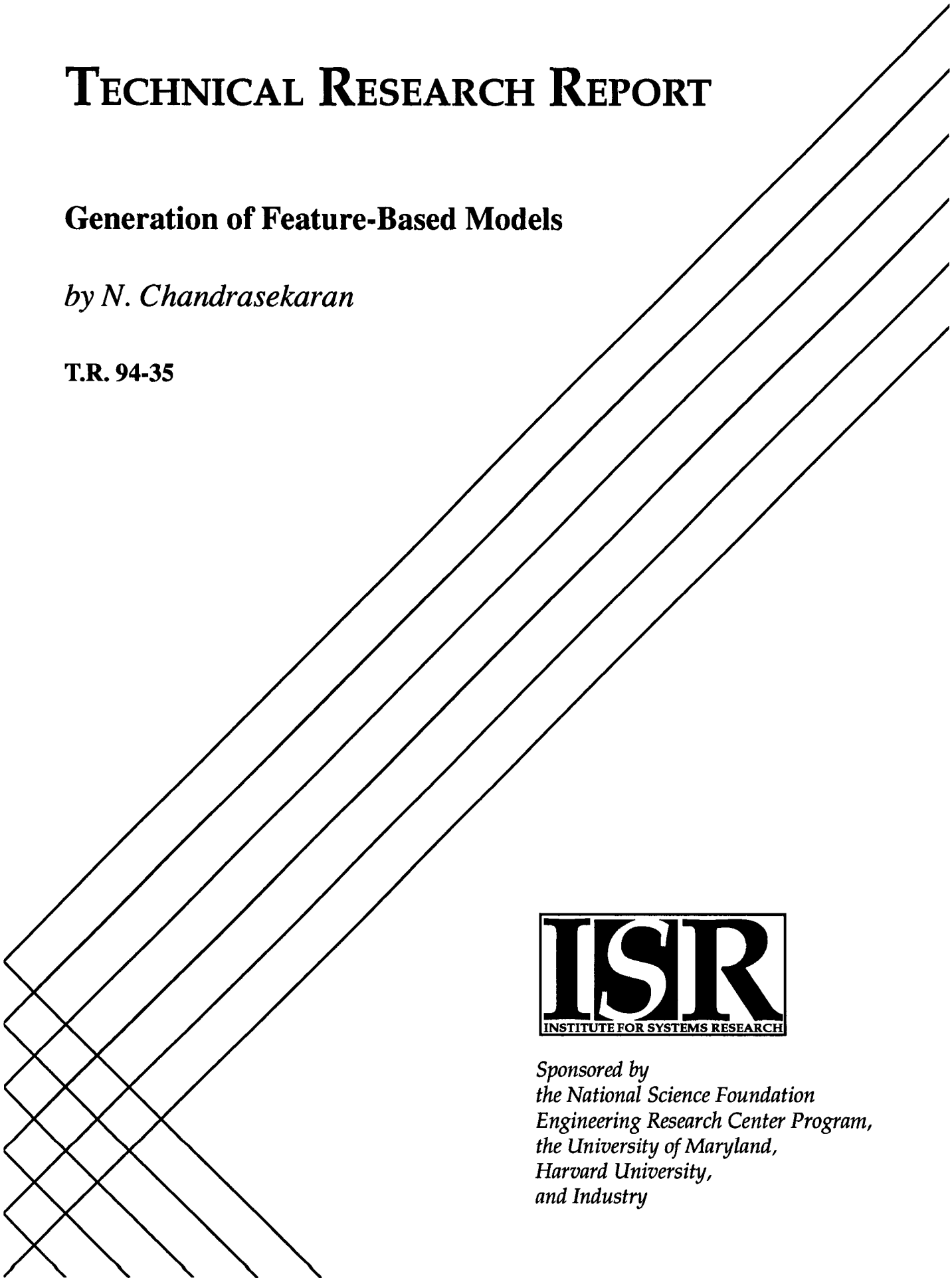
TECHNICAL RESEARCH REPORT

Generation of Feature-Based Models

by N. Chandrasekaran

T.R. 94-35



Sponsored by
the National Science Foundation
Engineering Research Center Program,
the University of Maryland,
Harvard University,
and Industry

Generation of Feature-Based Models

Honors Paper

Nirupama Chandrasekaran*

University of Maryland
College Park, MD 20742 USA
May 4, 1994

*Department of Computer Science and Institute for Systems Research. Email: nirupama@cs.umd.edu

1 Introduction

Machining a part involves performing various operations on a piece of stock to transform it into a desired form. Each of the machining operations act on the stock and change its structure. A part is produced by performing a sequence of machining operations on the stock. Given the design of a part, there are often several different sequence of machining operations, each of which when applied on the stock would produce the part. A *feature-based model* is one such sequence. Thus a feature-based model is a set of features that constitute the entire part. The purpose of this project is to generate all the feature-based models with all possible sequences of machining operations for parts that can be manufactured on a lathe machine. From the feature-based models generated in this project, the designer can choose the best sequence of machining operations by evaluating each sequence with respect to parameters such as fixturing the workpiece, feed rates, cutting speeds etc.

A part could be prismatic, sculptured or rotational, but the scope of this project is restricted to rotational parts (i.e. parts that are rotationally symmetric about the horizontal axis). A rotational part can be reduced to two dimensions, without loss of information, by taking a cross section going through its rotational axis. The purpose of the project was to develop an algorithm that operates on the two-dimensional representation of a rotational part and generates feature-based models.

For this project I have implemented three modules. The first module provides a user interface, which lets the user give the description of the part that is to be machined. The second module of the system lets the user specify the features to be extracted. It then performs validity checks for those features and verifies if the part can been be machined with the set of features specified by the user. The third module generates the feature-based models by reducing the problem to a simple irredundant set-cover problem.

There are drafting packages which have provisions for describing three-dimensional parts and some of them also support tolerance and surface finish descriptions. Most of the packages however, do not support feature extraction and generation of feature-based models (Modules II and III of this project respectively). The system developed in this project provides a simple X-window user interface for describing a part and specifying the features, and generates the feature-based models for the part.

Section 2 of this paper gives a detailed description of data structure used and the implementation of Module I. Specification of features and feature extraction are explained in Section 3, and Section 4 provides an algorithm to generate the feature-based models.

2 Definition of terms

A *part* is the piece of metallic solid that is obtained after performing a sequence of machining operations on the *stock*. In order to machine the part from the stock, some volume of material has to be removed from the stock. Such a volume is referred to as the *removal volume*. Various machining tools are used to remove the removal volume from the stock. The volume occupied by the tool minus the volume occupied by that part of the tool corresponding to the removal volume is referred to as the *accessibility volume*. When the removal volume is removed from the stock by the machining tool, a *feature* is created in the part. The type of features supported in this project are holes, recesses, bores, turns and faces (Figure 9).

The scope of this project is limited to parts that are rotationally symmetric about the x-axis, the horizontal axis. Such a part is said to be *axisymmetric* and is obtained by sweeping a two-

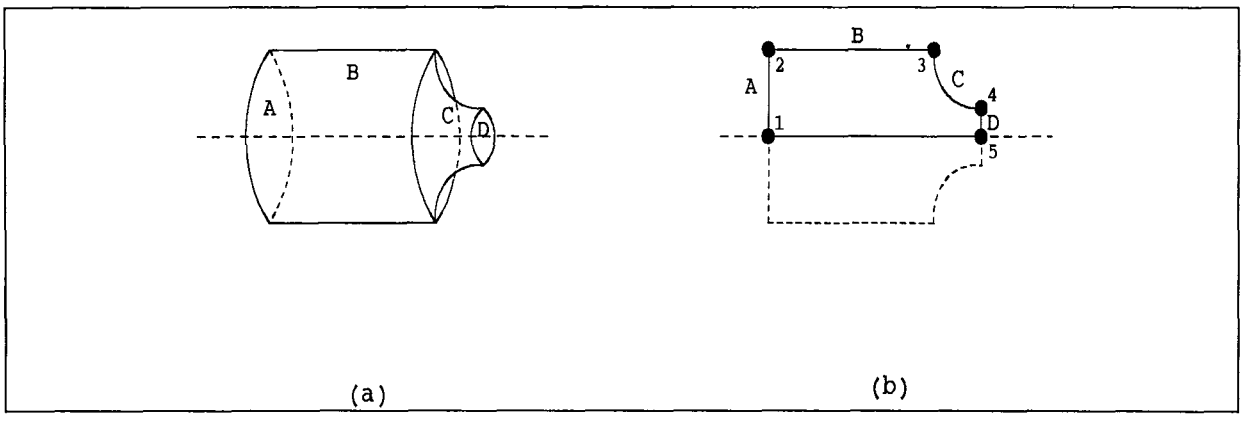


Figure 1: (a) a cylindrical solid, and (b) its representation in two dimensions

dimensional object through 360 degrees about the x-axis. Thus, throughout the implementation of this project, the part itself is represented in two dimensions and only the data for this two-dimensional representation is maintained (Figures1 and 2). The two-dimensional representation of a part is obtained by taking the cross section of the part going through the x-axis, the axis of rotational symmetry, parallel to the xy plane and taking only the part of the section above the x-axis. Such a cross section is hereafter referred in this paper as the *vertical section* (Figure 1).

In the two-dimensional representation, a straight edge corresponds to either a cylindrical surface or a conical surface or a planar surface, and a curved edge corresponds to a part of a toroidal surface. For example, in Figure 1(b), the straight edge A corresponds to the planar surface A in Figure 1(a), the straight edge B corresponds to the cylindrical surface B, the curved edge C corresponds to the part of the toroidal surface C, and the straight edge D corresponds to the planar surface D.

The two-dimensional representation consists of a sequence of edges placed such that each edge, except possibly the first and the last edges in the sequence, shares its end points with the neighboring edges on either side (Figures 1(b) and 2). If the first and the last edges of the sequence have a common end point, i.e. the starting point of the first edge coincides with the finishing point of the last edge, we will say in this paper that the part is *completely defined* and that the edges of the two-dimensional representation form a *closed loop.* It is possible that the two-dimensional representation involves several disjoint areas (encountered in Module II). In this case a list of sequence of edges, i.e. a list of closed loops are used to represent the disjoint areas and such a structure will be referred in this paper as a *set of closed loops.* For example, in Figure 14(b), the difference $M - S$ would be represented as a set of closed loops consisting of four closed loops, one for each of the four disjoint shaded regions.

3 Part Description

In the system described in this paper, the first step in designing a part is to specify its geometry and the tolerance values attached to the its surfaces. The geometry of a part is its contour and the dimensions such as lengths and angles associated with the contour. The tolerance values are the values attached to the surfaces of a part indicating the permissible amount of deviations from the

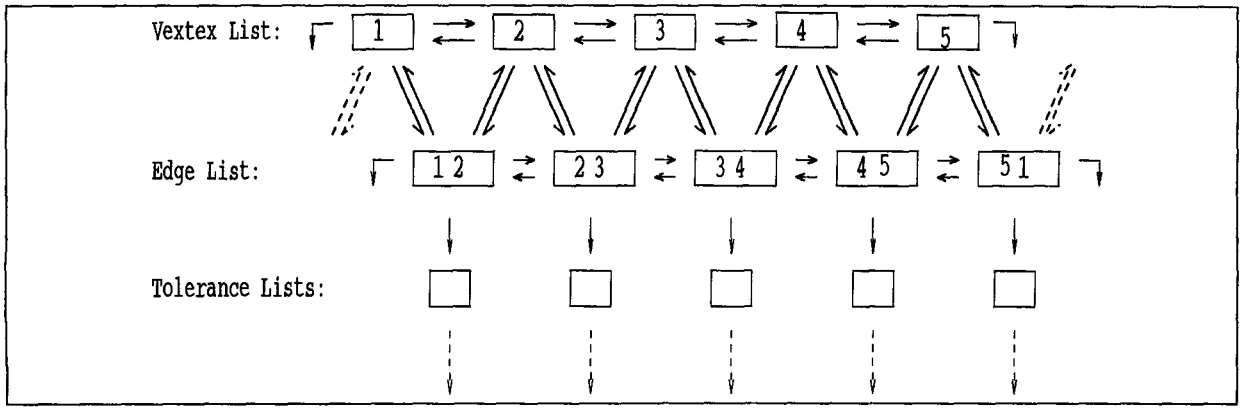


Figure 2: Data structure for the part in Figure 1

"ideal part" the designer has in mind. In reality, it is not possible to machine the ideal part because of the imperfections in the machining process, such as vibrations produced by the machining tools during the machining operations. The amount of permissible deviation from the ideal part would depend on the functional requirements of the part. The type of tolerance values that can be attached by the user in the system described in this paper are (see Figure 8):

1. *Surface Finish* to indicate how rough a surface is allowed to be.
2. *Diameter Tolerance* to indicate the amount of deviation permissible from the nominal diameter.
3. *Length Tolerance* to indicate the amount of deviation permissible from the nominal length between two surfaces.
4. *Concentricity* to indicate the amount of deviation permissible between two coaxial surfaces.
5. *Cylindricity* to indicate the amount of deviation permissible for a cylindrical surface.
6. *Perpendicularity* to indicate the amount of deviation permissible between two perpendicular surfaces.

The Surface Finish, the Diameter tolerance and the Cylindricity can be specified simply as numbers. However the Length tolerance, the Concentricity and the Perpendicularity require both a number and a datum, i.e. a reference surface elsewhere on the part. For example, Figure 8 shows how all three of these tolerances might be specified with respect to a datum surface named A.

A software system used to specify the geometry and tolerance values of a three-dimensional part must be easy to use and follow conventions that are simple to understand. Module I of the project provides an implementation of such a system for axisymmetric parts. As explained in the previous section, an axisymmetric part is a part that is rotationally symmetric about the x-axis and is obtained by sweeping a two-dimensional object through 360 degrees about the x-axis. By reducing the three-dimensional part to a two-dimensional representation (refer previous section), the process of specifying the geometry of the part is greatly simplified because the geometry for the third dimension is eliminated.

3.1 Description of the Data Structure

In this project, the two-dimensional representation of a part consists of a list of vertices with edges extending between each pair of adjacent vertices. A vertex in the two-dimensional representation corresponds to an edge of the three-dimensional part and an edge in the two-dimensional representation corresponds to a face of three dimensional part (Figure 1). A vertex is represented by its x and y coordinates, and the parameters of an edge include:

- type of edge: straight or curved.
- if the edge is curved, additional parameters to describe it.
- a list of attributes of the tolerance values associated with the edge.

The data is maintained as three sets of entities, with links among them as illustrated in Figure 2.

1. *the vertex list* - a doubly linked list of consecutive vertices (x and y coordinates).
2. *the edge list* - a doubly linked list of edges with each edge having pointers to nodes in the vertex list to specify the two end points (for reasons that will be explained later, these are referred to as the *from point* and the *to point* of an edge).
3. *tolerance lists* - a linked list attached to each edge, holding the tolerance values associated with the face of the part corresponding to that edge. These tolerance values can be surface finishes, diameter tolerances, length tolerances, concentricities, cylindricities or perpendicularities.

With this data structure, the edges are separated from the vertices and while also separating the tolerance values from the specification of the geometry for the two-dimensional representation, the data structure still associates the tolerance values with the corresponding faces of the part. Separating the geometry from the tolerances, and the vertices from the edges make it easier and faster to manipulate data related to the geometry of the part for Modules II and III becomes faster and easier. The double links between nodes in the vertex list and the edge list, and the cross links between them help in speeding up of queries such as the following:

- *Given an edge, what are the coordinates of its two end points?* This query is needed, for example, in Module II of the system (human supervised feature extraction), where each edge is classified (according to conventions explained in Section 4), for which information on its two end points is required.
- *Give two edges that intersect at a point, what are the coordinates of the two end points of the two edges?* This query is needed, for example, in Module II of the system (human supervised feature extraction), where each edge is classified (according to conventions explained in Section 4), for which information about the edges that meet at the end points of the edge being classified is required.
- *Given an edge, what are the coordinates of the end points the edges adjacent to this edge?* This query is needed, for example, in the edit geometry and assign tolerance portions of Module I, and in the classification of edges in Module II, where information on neighboring edges is used.

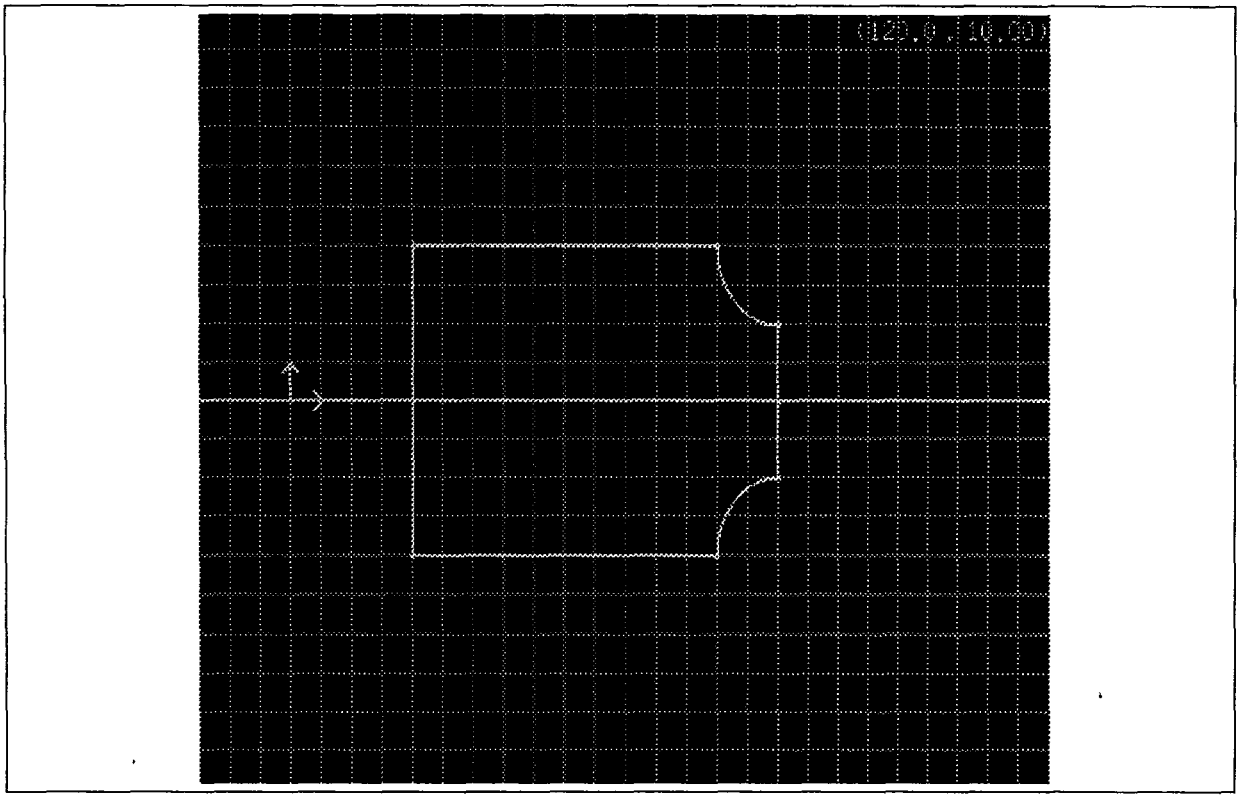


Figure 3: A completed part on the *Graphics Display* Window

- *Given a vertex, what are the coordinates of the vertices adjacent to this vertex?* This query is needed, for example, in the edit geometry and assign tolerance portions of Module I, and in the classification of edges in Module II, where information on neighboring vertices is used.

The tolerance values attached to faces of a part are used while performing the manufacturability analysis for the feature-based models generated in Module III of this project. By making the tolerance values attributes of the faces of the part, the data structure allows for easier manipulation of data during the manufacturability analysis.

3.2 Part description - Execution

The program's user interface consists of the following three windows, as shown in Figures 3, 4 and 5.

1. The *Graphics Display* window is the window where the part to be machined is displayed. Since this project deals only with parts that have rotational symmetry, the part is represented in two dimensions by taking its *vertical section* (as described in Section 2 and shown in Figure 1). Figure 3 shows the graphics display window with the completed part of Figure 1.
2. The *Command* window is used when input is given through the keyboard or when a comment is issued by the program to the user. Figure 5 shows the command window with the comment "Part is completed" indicating that the part has been completely defined.

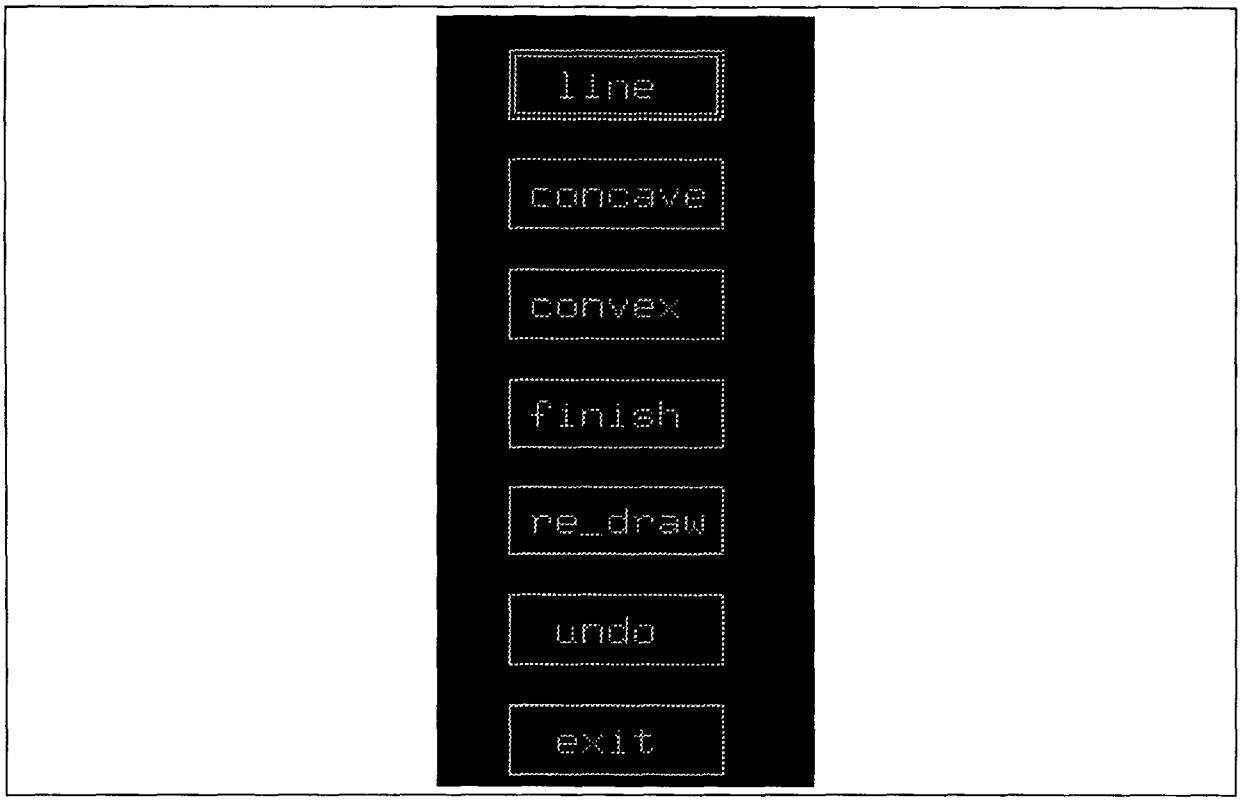


Figure 4: *Menu* window showing the options available during part description

3. The *Menu* window displays the menu options available to the user at a particular point of execution. Figure 4 shows the menu options available during the execution of the "Define Geometry" section of Module I.

3.2.1 Define Geometry

In order to maintain correspondence between the two-dimensional representation of the three-dimensional part and the data structure, the system enforces the following rules during the definition of the geometry of two-dimensional representation.

- The user is allowed to create edges only in a sequential manner, following the outline of the vertical section of the part, starting from any point and following any direction. For example, in Figure 6, starting from vertex labeled '1', and following the clockwise direction, edges A, B, C, D and E are created sequentially in that order.
- The user is allowed to make changes to the two-dimensional representation of the part only if it is completely defined, i.e. the two-dimensional representation forms a closed loop. For the example in Figure 6, changes can be made only after creating the edge E.
- The user can attach tolerance values to an edge only if the two-dimensional representation of the part is completely defined, i.e. the two-dimensional representation forms a closed loop.

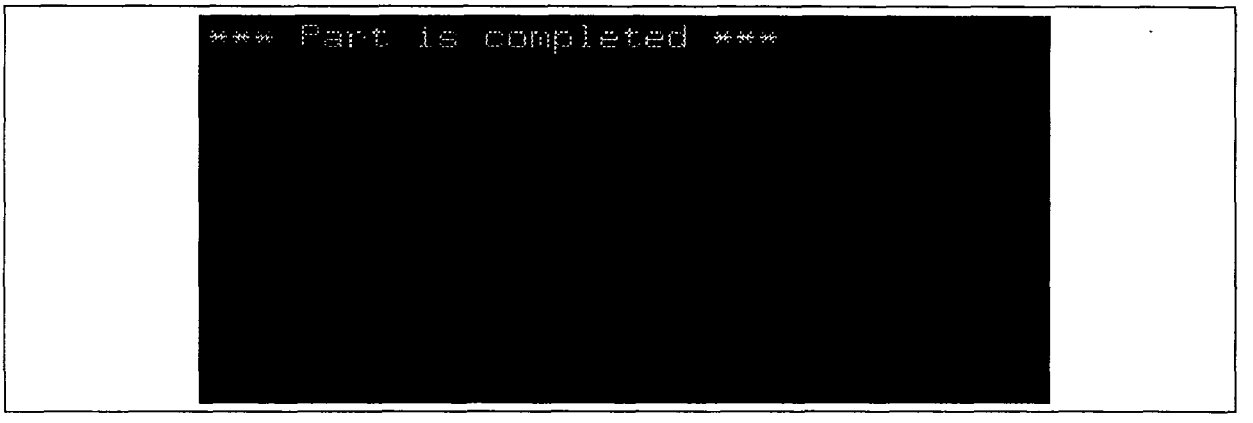


Figure 5: Comment on the *Command* window indicating that the part is completely defined

For the example in Figure 6, changes can be made only after creating the edge E.

In order to set the scale, the precision and the datum point (a fixed point serving as a reference point for all other points on the window) for the two-dimensional representation of the part on the *Graphics Display* window, the following parameters should be set by the user before proceeding with the definition of the geometry.

1. the maximum x and y ranges: to set the scale on the *Graphics Display* window.
2. the grid increment: to set the precision value (when the user clicks with the mouse at a point on the Graphics Display window, in order to maintain the required precision, the system shifts that point to the lower left corner of the grid).
3. the datum point as a reference point (the origin (0, 0)).

Figure 6 shows the steps involved in creating a part with five edges: four straight edges (A, B, D and E) and a curved edge (C). Since the two-dimensional representation of the part does not include any point below the x-axis, the system takes input only from the part of the *Graphics Display* window above the horizontal axis. In order for the user to visualize the part that is he/she is creating, the system displays the entire cross section of the part, by reflecting the input on the part of the *Graphics Display* window below the horizontal axis (Figure 6).

In order for a part to be completely defined, the *to point* of the last edge created in the two-dimensional representation must be the same as the *from point* of the first edge that was created. The *from point* of an edge corresponds to the end point that was defined first during the creation of the edge and the *to point* refers to the end point that was defined second, indicating that the edge extends between the *from point* and the *to point*. For the edge B in Figure 6, vertex 2 is the *from point* and vertex 3 is the *to point* because vertex 2 was created before vertex 3. The finish button, in the menu at this level, helps in defining the part completely, by automatically choosing the first point in the vertex list as the *to point* for the last edge. For the example in Figure 6, the *to point* for edge E, the last edge created, is vertex 1 which is also the *from point* for edge A, the first edge that was created. The finish button also sets the finish flag to indicate that the part is now completely defined.

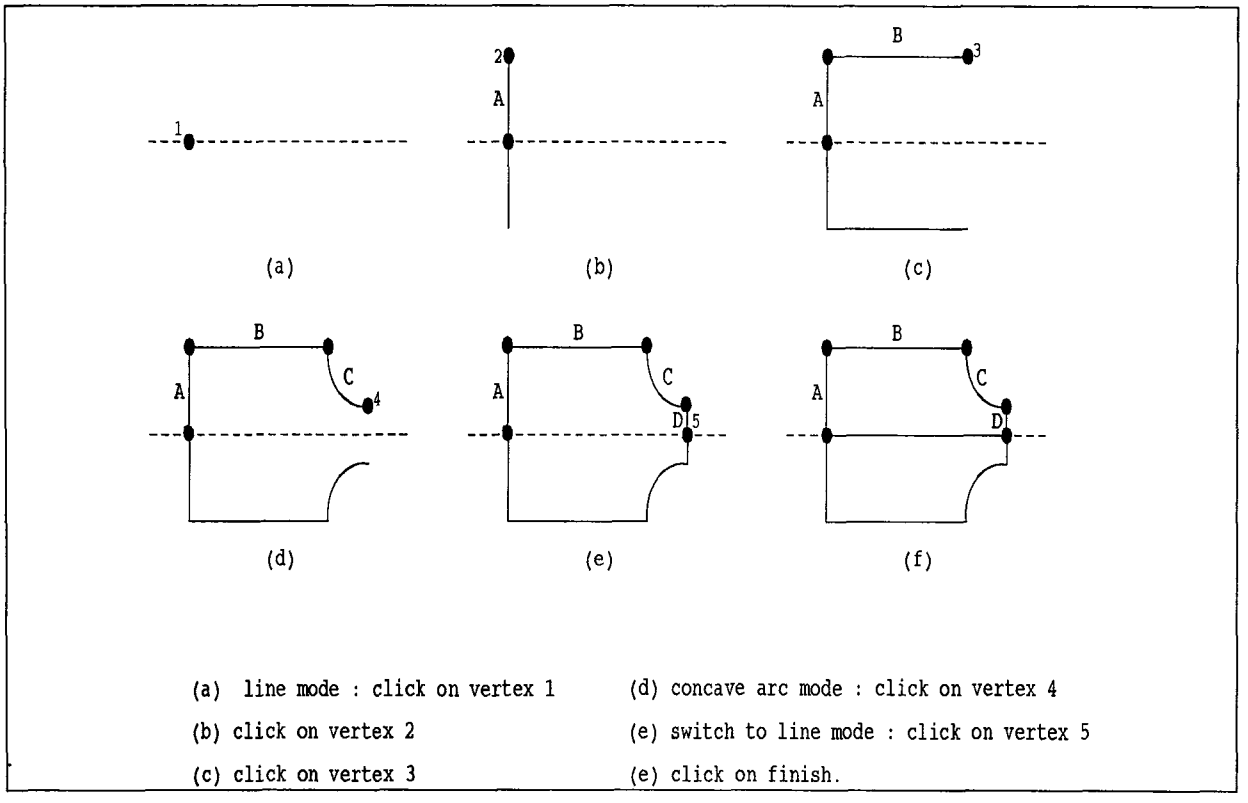


Figure 6: Step by step creation a part with five vertices

3.2.2 Edit Geometry

The constraints imposed by the system in the define geometry module, give the user very little freedom in making changes while defining the two-dimensional representation of a part. In order to allow for changes to be made after the two-dimensional representation has been defined, Module I contains an "Edit Geometry" option. The set of operations available to the user are:

1. insert vertex;
2. delete vertex;
3. change straight edge to curved edge;
4. change curved edge to straight edge.

With this set of operations, it is possible to make any modifications necessary to the two-dimensional representation of the part defined in the "Define Geometry" module. This becomes especially useful when the user has to define the geometry for a new part whose two-dimensional representation is very similar to that of the two-dimensional representation of a part that was created earlier (Figure 7).

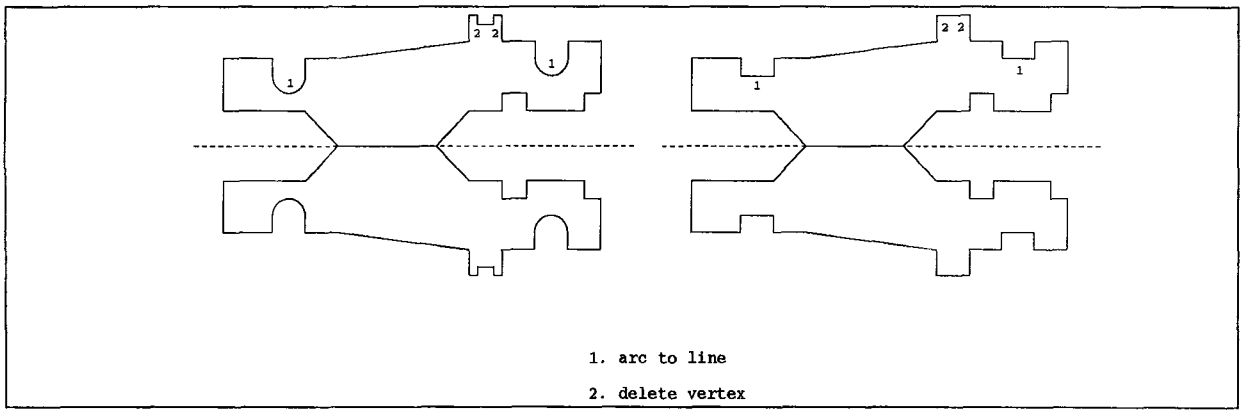


Figure 7: The part on the right can be created by editing the part on the left

3.2.3 Tolerance

Once the part has been completely defined, Surface finish, Diameter tolerance, Length tolerance, Concentricity, Cylindricity and Perpendicularity can be attached to any edge. These values become attributes of that edge as shown in Figure 2. Figure 8 is a sample part with tolerance values attached to its faces.

3.2.4 Load File

The "Load File" option lets the user load the geometry and the tolerance values of the two-dimensional representation of a part that has already been defined. It is not necessary for the part to be completely defined. The name of the file that is to be loaded should end with the characters ".part". A part that has been created using other software can be loaded provided it has been translated to the format compatible to this system.

3.2.5 Save File

The "Save File" option lets the user save the geometry and tolerance values of the two-dimensional representation of a part that the user described in this module. Before saving, the system concatenates the characters ".part" onto the filename given by the user. It is not necessary for a part to be completely defined for it to be saved.

3.2.6 Other Functions Supported

The following function are added to Module I to aid the user while defining the part.

1. *Tolerance Switch:* Acts as a toggle switch for either displaying the tolerance values on the screen or not displaying them on the screen.
2. *Create Picture File:* To create a Postscript file of any of the windows, the system takes a window dump and saves the dump in a file named by the user after concatenating the file name with the characters ".ps".

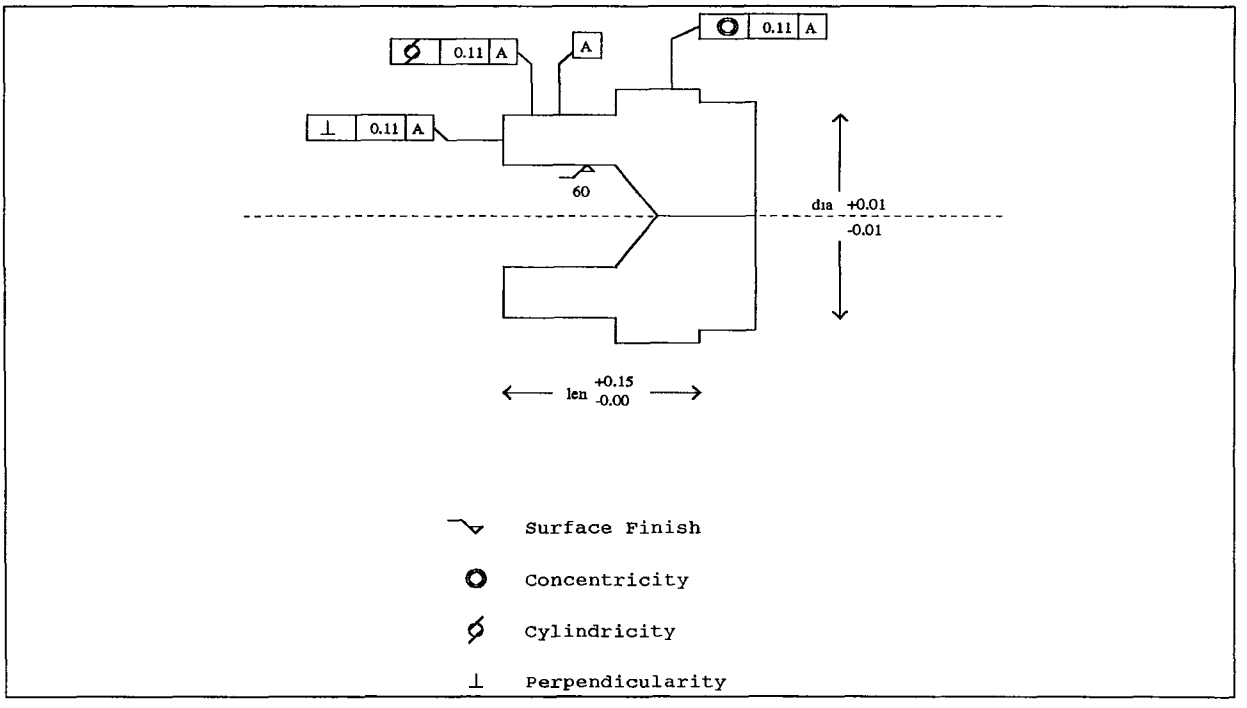


Figure 8: Sample part with tolerance values attached to it

3. *Grid Display Switch:* Acts as a toggle switch for either displaying the grid on the screen or not displaying them on the screen.

3.2.7 Quit

Exit from the system.

4 Human Supervised Feature Extraction

Having specified the design of a part that has to be manufactured, it is necessary to identify the features that constitute the part in order to apply the required machining operations to the stock to produce the part. For example, suppose that using Module I, the user represented a three-dimensional part in two dimensions and the system built a data structure for the representation. Then using Module II, the user can define the features for the part described in Module I and the system does validation checks for the features and performs the operation of feature extraction using the data structure built in Module I to see if the part can be machined with the set of features defined. Feature extraction is the removal of material from the stock so as to create a feature of the part. The set of features provided in this system are holes, recesses, bores, turns and faces (Figure 9). The system has provisions to add more features to this list. In order to be able to define features, the system requires that the part for which the features are defined be completely defined (see Section 3). The operations involved in this module are:

1. load the geometry of the two-dimensional representation created in Module I for the part.
2. load the two-dimensional representation of the stock from which this part is to be machined. This can be done in two ways.
   - create the stock using methods described in Module I.
   - take the smallest possible cylindrical solid that can encapsule the entire part.
3. define the features to be extracted from the set of features provided (see Section 4.1).
4. extract the features to determine if the specified set of features can produce the part.
5. save all the features defined, so that they can be used as input to Module III.

From the set of features defined by the user in this module, Module III generates the feature-based models for the part.

4.1 Feature Definition

In order to define a feature, the user has to provide certain parameters for the feature. These parameters are the length of the feature (L), the inner and outer radii (R and r) and the coordinates of the reference point (x, y) as indicated in Figure 9.

A feature with these parameters is valid and can be extracted provided the following conditions are satisfied:

1. the removal volume does not intersect with part;
2. the accessibility volume does not intersect with part;
3. the removal volume intersects with some portion of the stock.

For a given feature, if the above conditions are satisfied, the feature is added to the *feature list* - a list holding all the features defined by the user for the part.

Figures 11(a) and (b) show an example stock and part and the parameters associated with them. Figure 11(c) shows the volume to be extracted (shaded region) from the stock to produce the part in 11(b), and Figure 12 shows four features (three holes and a recess) defined for the part in Figure 11(b).

4.2 Feature Extraction

To extract a feature from the stock means that the volume represented by the feature is removed from the stock. In the two-dimensional representation, this reduces to deleting the area represented by the feature from the area represented by the stock. The difference routine accomplishes this by taking two areas, A and B, as input and giving as output the area $A - B$. If after all the user specified features have been subtracted from the stock, the remaining volume exceeds the volume of the part, then the part can not be manufactured with the given set of features.

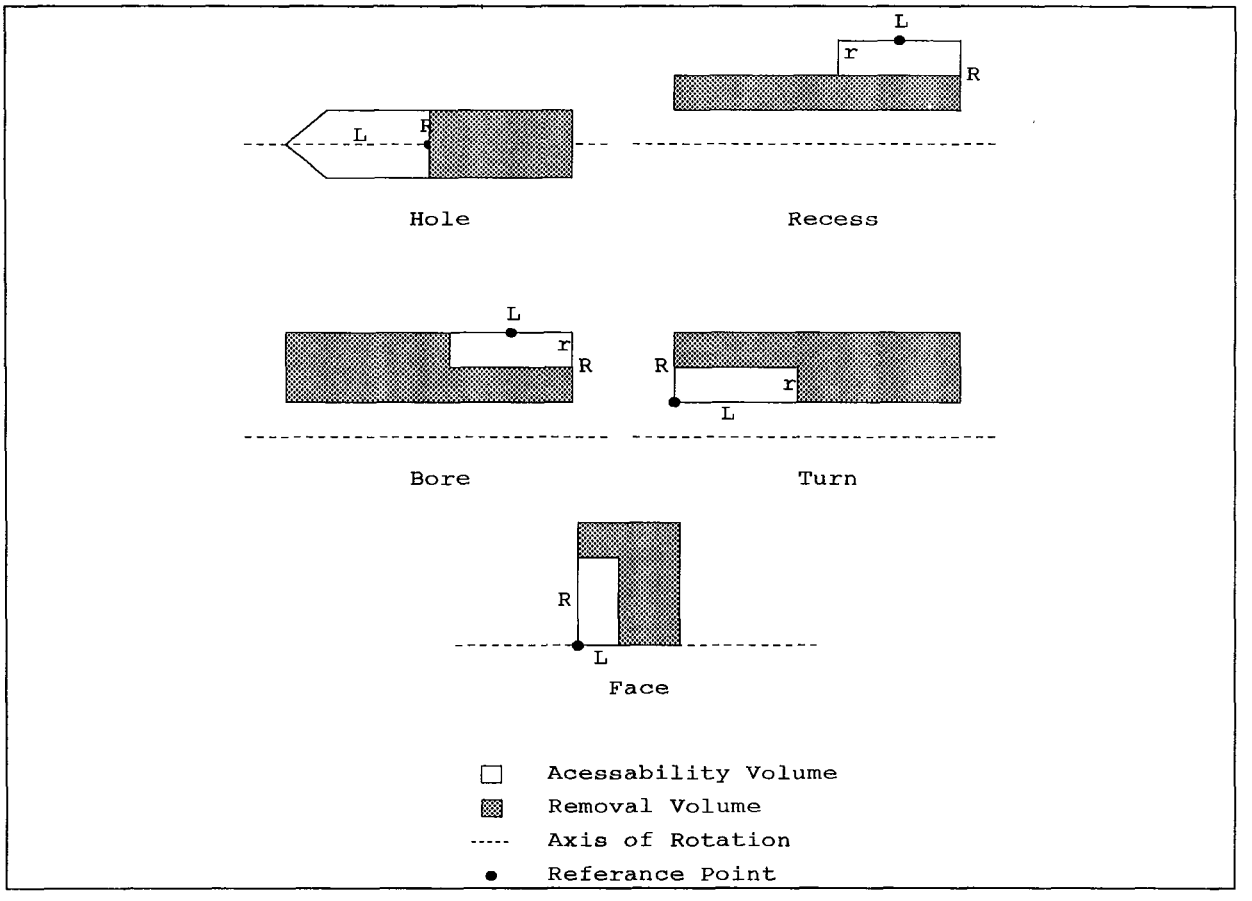


Figure 9: Feature set

4.2.1 The Difference routine

This routine takes as input a set of closed loops M, defining disjoint areas ($Stock - Part$ = volume to be extracted) and a closed loop S, defining a single area ($feature$) and produces as output a set of closed loops representing the area $M - S$ (Figure 13). This difference is calculated through the following steps.

The algorithm classifies each vertex in M and S with respect to S and M respectively as:

- *in* : the point lies inside the loop with respect to which it is being classified.
- *out* : the point lies outside the loop with respect to which it is being classified.
- *on* : the point lies on the perimeter of the loop with respect to which it is being classified.

Each edge of M and S can now be classified with respect to S and M respectively using the above vertex classification, as follows (Figure 13):

- if both end points are *in* then the edge is *in*.

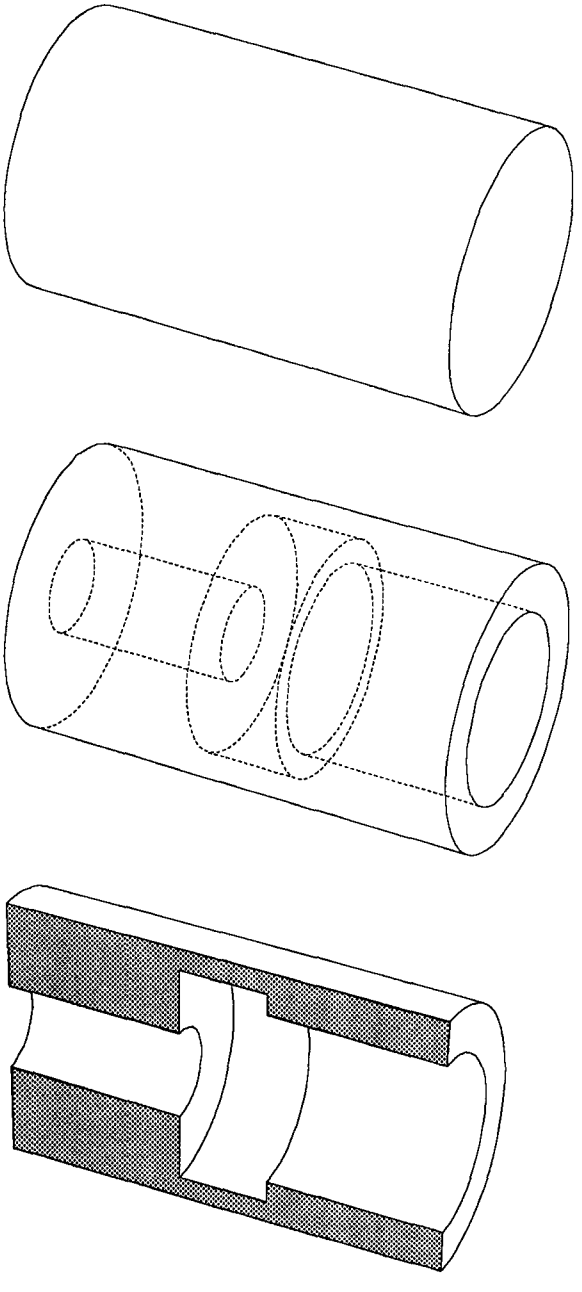
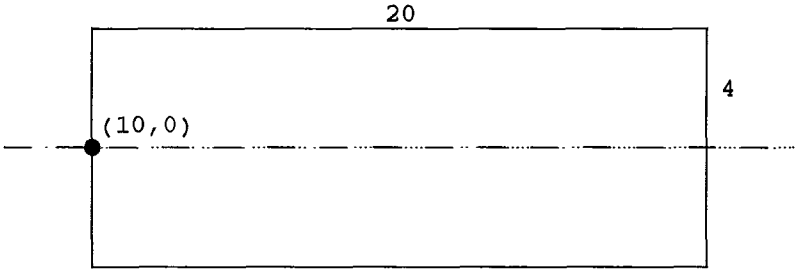
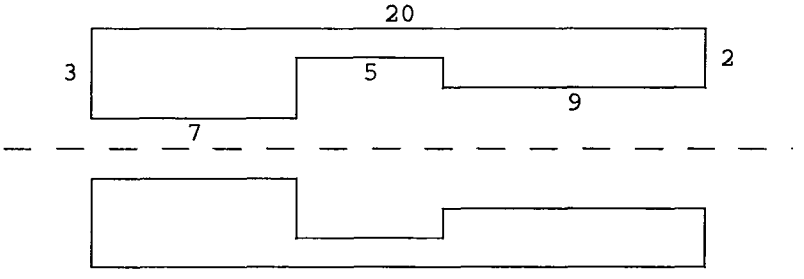


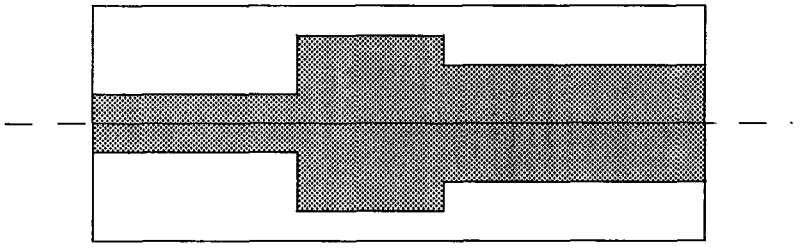
Figure 10: Stock and part in three dimensions



(a): Stock



(b): Part



(c): Stock - Part

Figure 11: Stock and part in two dimensions

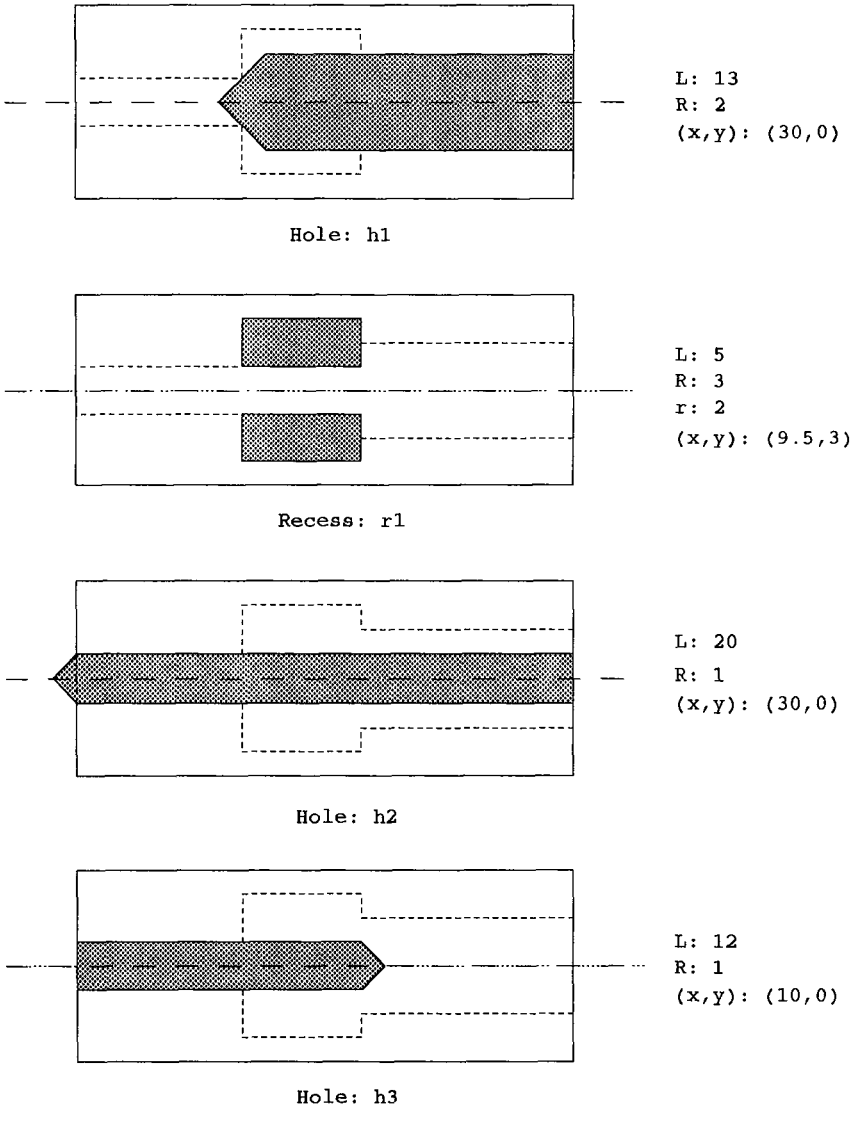


Figure 12: Features defined for the part in Figure 11

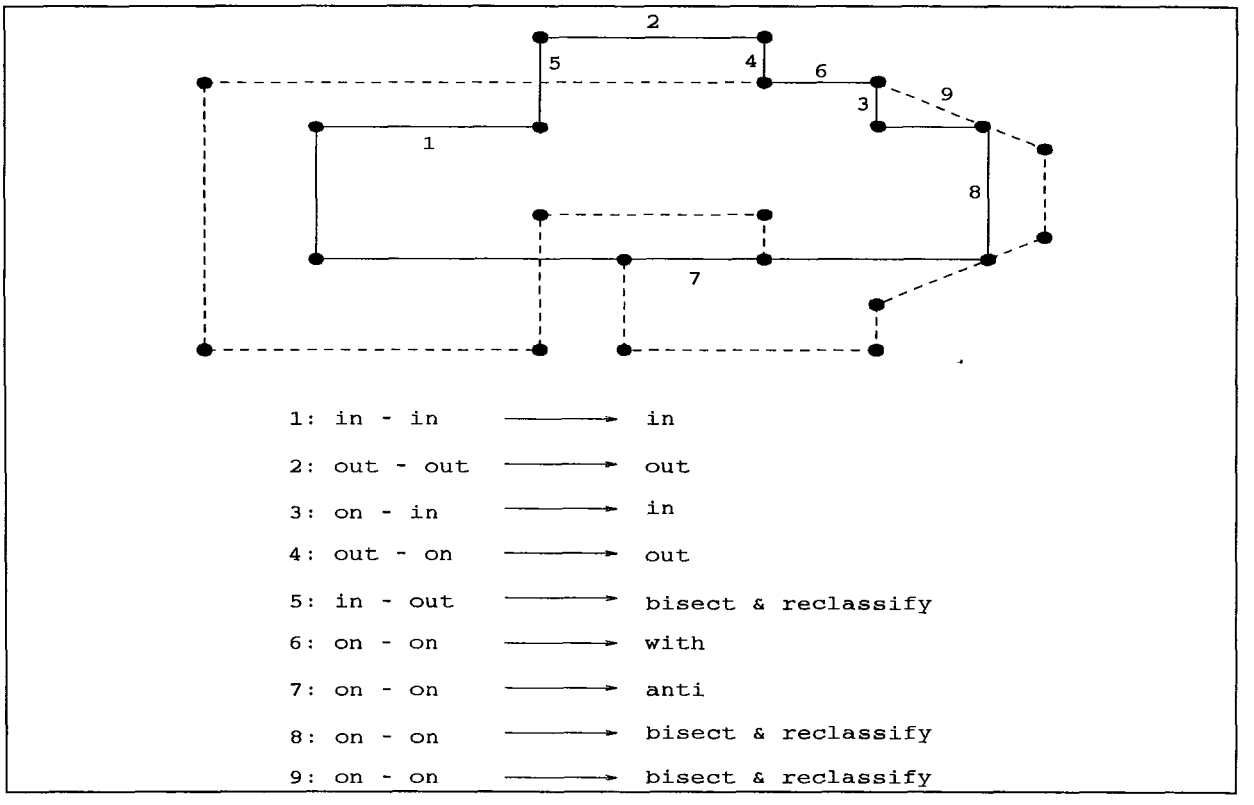


Figure 13: Various classification of edges

- if both end points are *out* then the edge is *out.*
- if one end point is *in* and the other end point is *on* then the edge is *in.*
- if one end point is *out* and the other end point is *on* then the edge is *out.*
- if one end point is *in* and the other end point is *out*, then bisect the edge at the point where it intersects with loop with respect to which it is being classified, and recursively classify each segment.
- if both end points are *on* and the edge lies on an edge of the loop with respect to which it is being classified then it is either *with* or *anti*, depending on its orientation with respect to the orientation of the edge on which it lies. If both the edges have similar orientation, then both are classified as *with* else if the two edges have opposite orientation then both the edges are classified at *anti.*
- if both end points are on but the edge does not fall on an edge of the loop with respect to which it is being classified, then bisect the edge at any point and recursively classify each segment.

Now $M - S$ is constituted by the following edges (Figure 14):

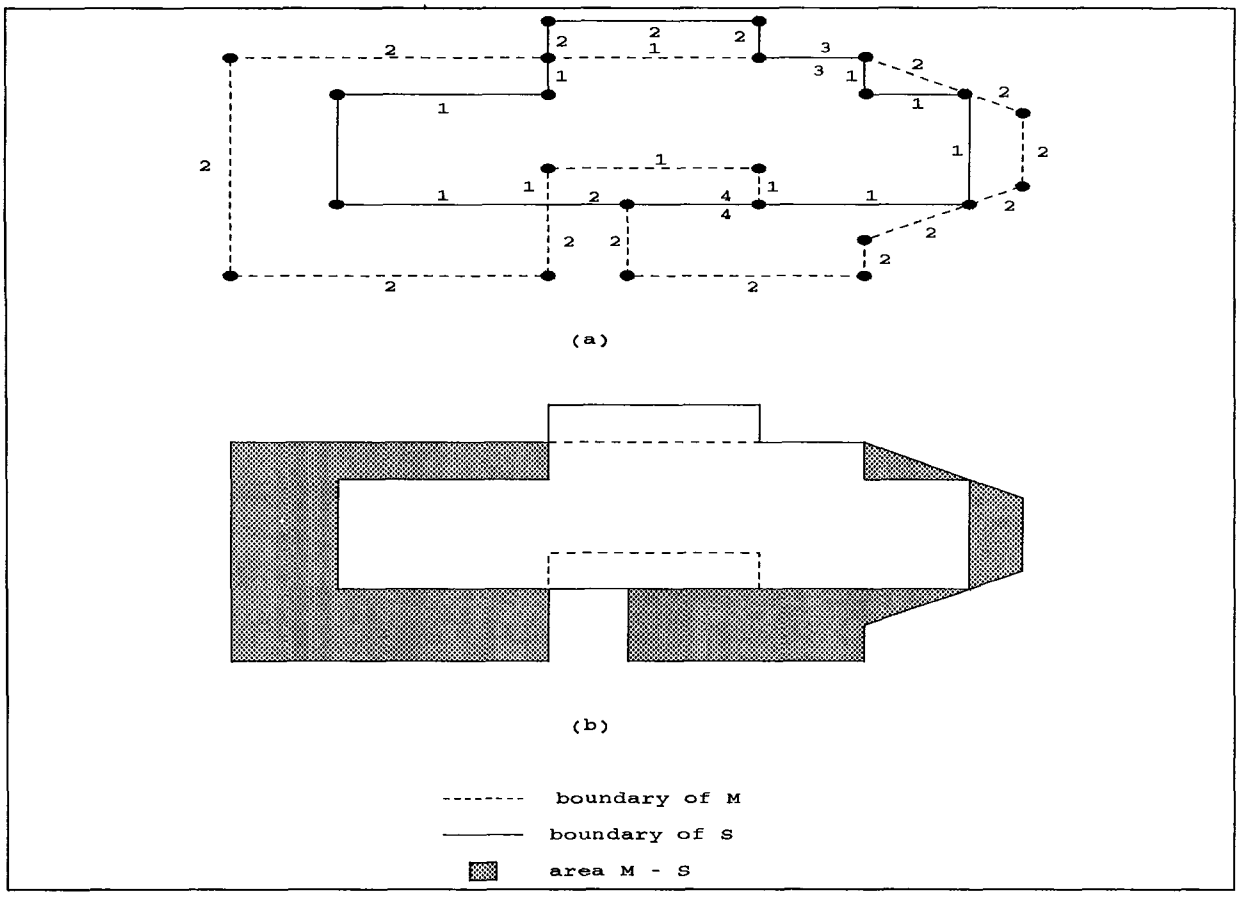


Figure 14: (a) Two closed loops and (b) their difference, a set of four closed loops, each of which encloses one of the four shaded areas

- edges of M that are *out* and *anti*
- edges of S that are *in*

These edges are then sorted to form closed loops which are then put together to form a set of closed loops representing the region $M - S$.

5 Generation of Feature-Based Models (FBMs)

With the set of features defined using Module II for the part described using Module I, the system generates the feature-based models in Module III. The problem of generating the feature-based models can be reduced to the well known combinatorial problem of finding irredundant set covers [2]. An instance (W, X) of the *irredundant set cover problem* consists of a finite set W and a family X of subsets of W, such that every element of W belongs to at least one subset in X. The problem is to find a minimum-size subset $C \subseteq X$ whose members cover all of W. Therefore, C is an irredundant cover for W if $\exists f, f \in C$ and $C - f$ does not cover all of W. To generate the feature-based models,

if the volume $(Stock - Part)$ is taken to be W and each feature taken to be an element in X, then each cover generated by the irredundant set cover algorithm will correspond to a feature-based model. By generating all the set covers, L, the set of all feature-based models are generated.

5.1 Reducing the Problem of Generating the FBMs to an Irredundant Set Cover Problem

Let X be the set of all features and W be $(Stock - Part)$. The problem of generating the feature-based models can be reduced to the problem of generating a subset of X that covers W as follows. In the two dimensional representation, the area $W = (Stock - Part)$ is divided into disjoint areas, $a_1 \ldots a_n$, such that for any feature, its two-dimensional representation is formed by the union of one or more of these areas. More formally, $W = \bigcup(a_i), 1 \leq i \leq n$, and for every $g \in X$, there exist $a_{i_1}, \ldots, a_{i_k}$ such that $a_{i_1} \cup \cdots \cup a_{i_k} = g$. Now, each irredundant cover C, generated by the irredundant set cover algorithm (Figure 15), represents a minimal set of features that are required to produce the part; and L contains all of the irredundant set covers.

For example, for the part and stock in Figure 11, let four features (three holes and a recess) as shown in Figure 12 be defined. Figure 16 shows the break up of the area $Stock - Part$ into disjoint areas $a_1 \ldots a_n$. Therefore,

hole h1 is formed by $\bigcup(a_3, a_4, a_5, a_7, a_8, a_9)$,
hole h2 is formed by $\bigcup(a_1, a_4, a_8)$,
hole h3 is formed by $\bigcup(a_1, a_4)$, and
recess r1 is formed by $\bigcup(a_2, a_3, a_5, a_6)$.

More formally, W and X are as follows:

$$\begin{aligned} W &= \bigcup(a_1 \ldots a_9) \\ X &= \{h1, h2, h3, r1\} \\ &= \{\bigcup(a_3, a_4, a_5, a_7, a_8, a_9),\ \bigcup(a_1, a_4, a_8),\ \bigcup(a_1, a_4),\ \bigcup(a_2, a_3, a_5, a_6)\} \end{aligned}$$

5.2 Algorithm for Generating FBMs

Because of the above reduction, we can write an algorithm that uses the set covering technique to generate the FBMs. The algorithm works as follows. If $\bigcup(X)$ does not include all elements of W, then the problem cannot be solved. Otherwise, divide X into two sets, F and $X - F$ where $F = \{f : f \in X$ and f has a node which does not belong to any $g \in X$ other than f itself $\}$. Therefore, F has elements, all of which must be in any cover for W and $X - F$ has elements which may or may not be in a cover for W. The function generate_cover performs three checks to pick out elements from $R = X - F$ required in a cover for W in addition to the elements in F. The three checks are:

1. *Redundancy Check*: $f \in F$, is a redundant element in the cover if $\bigcup(F - f) = \bigcup(F)$.
2. *FBM Check*: if $\bigcup(F) = W$, then $\bigcup(F)$ covers W and hence forms a cover.
3. *Feasibility Check*: if $\bigcup(F \cup R) \neq W$, then W cannot be covered.

```
procedure generate_fbm();
    if (⋃(X) ≠ W)
        "unsolvable problem";
        exit;
    end if;
    F = ∅;
    for each f ∈ X
        if f − ⋃_{g∈X−f}(g) ≠ ∅
            F = F ⋃ f;
        end if;
    end for;
    generate_cover(F, X − F);
end generate_fbm;

procedure generate_cover(F, R)
    if redundancy_check(F)
        return;
    end if;
    if fbm_check(F)
        return;
    end if;
    if feasibility_check
        return;
    end if;
    Let g ∈ R;
    generate_cover(F ∪ g, R − g);
    generate_cover(F, R − g);
end generate_cover;

procedure redundancy_check(F, R)
    for each (f ∈ F)
        if ⋃(F − f) = ⋃(F)
            return(failure);
        end if;
    end for;
end redundancy_check;

procedure fbm_check(F, R)
    if ⋃(F) = W
        return(found a cover);
    end if;
end fbm_check;

procedure feasibility_check(F, R)
    if ⋃(F ∪ R) ≠ W
        return(failure);
    end if;
end feasibility_check;
```

Figure 15: Algorithm to generate the feature-based models

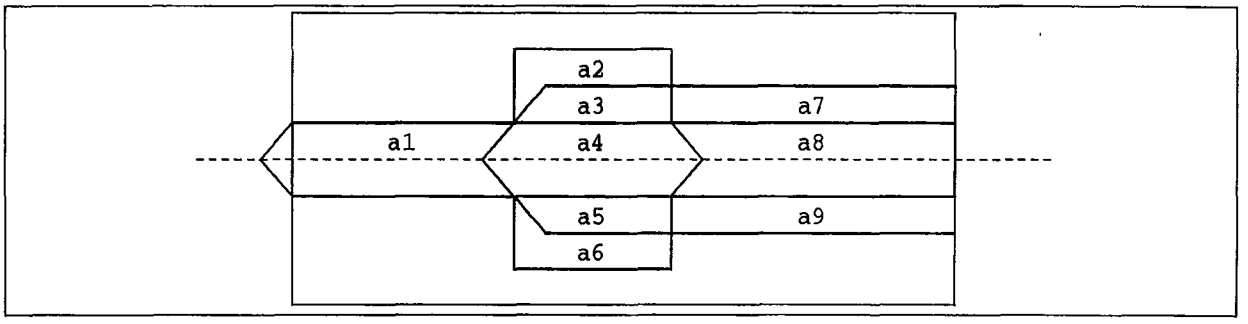


Figure 16: Forming the set cover for *Stock – Part*

The algorithm to generate all the irredundant set covers is outlined in Figure 15. For the example in Section 5.1, the algorithm in Figure 15 computes L as having two feature-based models. That is,

$$\begin{aligned} L &= \{\{h1, h3, r1\},\ \{h3, r1\}\} \\ &= \{\{\bigcup(a_3, a_4, a_5, a_7, a_8, a_9), \bigcup(a_2, a_3, a_5, a_6), \bigcup(a_1, a_4)\},\ \{\bigcup(a_1, a_4, a_8), \bigcup(a_2, a_3, a_4, a_5, a_6)\}\} \end{aligned}$$

6 Conclusion

Often there can be a number of ways to perform machining operations on a stock to produce a part, and only some of them are cost- and quality-effective. To determine the best machining sequence, it is necessary that each sequence of machining operations be evaluated and the best sequence be chosen. A feature-based model is a sequence of machining operations that when performed on a stock produces a part. This paper describes a way to generate all the feature-based models for a design of a part so that the designer can choose the best sequence of machining operations by evaluating each of the feature-based models.

By limiting the scope of the project to rotational parts, it has been possible to represent three-dimensional parts in two dimensions without loss of information. Throughout the project we develop algorithms that work on two dimensions, and finally yield the feature-based models for the three-dimensional part.

The project is functionally divided into three modules. The first and second modules respectively, provide interfaces for the user to describe the part that is to be machined, and to identify the features of the part. In addition, the second module also performs validity checks for the features defined by the user. With the input given by the user in the first two modules, the third module generates the feature-based models by applying the irreduntant set cover algorithm.

This project can be further extended to generate alternate machining plans by imposing precedence constraints on the machining sequence for each of the feature-based models generated by the third module. By performing manufacturability analysis on the alternate machining plans, the designer can choose the best plan to manufacture the part.

The ideas developed in this project helped provide the motivation for the ongoing development of a system called the Integrated Manufacturability Analysis and Critiquing System (IMACS) at

the University of Maryland, College Park. Limiting the scope of the project discussed in this paper to rotational parts and working in two dimensions, helped to focus on the underlying issues involved in developing a system for concurrent engineering, rather than on the specific issues involved in dealing with different kinds of parts. This was helpful during the initial phases of the development of IMACS which supports parts other than rotational parts and had support systems such as the ACIS and Hoops geometric modelers to handle specific issues related to different kinds of parts.

References

[1] S. K. Gupta and D. S. Nau. A systematic approach for analyzing the manufacturability of machined parts. Technical Report TR 93-76, The University of Maryland, Institute for Systems Research, College Park, MD 20742, USA, 1993.

[2] C. E. Leiserson T. H. Cormen and R. L. Rivest. *Introduction to Algorithms.* MIT Press/McGraw Hill Book Company, 1990.