Automatic program synthesis—From CAD to CAM*

by ROBERT T. CHIEN and TONY C. WOO
University of Illinois at Urbana-Champaign
Urbana, Illinois

INTRODUCTION

The technology gap between design and manufacturing has stimulated a number of research projects whose common objective is to bring design specification into some form of procedure that specifies the manufacturing sequence and tool path. Most notably, there are two related areas of study—sculptured surfaces and process planning. The former is concerned with representing non-analytic surfaces in parametric forms so that points on a surface can be generated from the parametric equations as cutting tool locations. The latter is concerned with grouping conventional machine parts and describing the part in special programming languages or representing them in codes to arrive at operation sequences, choice of machine tools, cutting data, and other information.

It is recognized that differential geometry is a powerful tool for representing sculptured surfaces. Coons' and Bezier's methods[1,2] have led to many developments and successful implementations of surface modeling systems with the use of computer graphics.[3,4] It is noted that there are logical operations in manufacturing, such as deciding the machining operations, or sequencing the operations, that cannot be represented in the framework of differential geometry alone.

On the other hand, studies in the area of process planning are just as successful. There are currently many systems in existence that accept special format inputs and generate work orders in batch or interactive modes.[5,6,7] Most of these systems deal with turned components by exploiting the symmetry of the part. Curiously enough, very little attention is given to the possibility of using designs done on computers as input to integrate the design and planning stages.

We see the problem of computer aided manufacturing as that of transforming geometric information into procedural information—from how something looks like into how it can be manufactured. We believe that there is enough information in a design to be processed automatically and to produce procedures for manufacturing purposes, if a person (a process planner, a part programmer, or a machinist) can do so with the same amount of information given.

In order to have a firm grasp of the problem, we have chosen three-dimensional machine parts as the problem domain. We feel there is sufficient structure in parts geometry and in numerical control machine tools for us to understand the problem and to develop the solutions.

To state our objective, we wish to associate the cavities in a given machine part design with the appropriate machining operations, sequence them, and expand the operations into numerical control programs.

DEVELOPMENT AND APPROACH

The basic issue addressed in this paper is that the shape of an object suggests a procedure. The handle of a tea kettle, a hammer, or a door knob suggest their functions in relation to the object they are attached to. Similarly, a familiar shape such as a hole in a machine part suggests drilling, a slot or a pocket milling. Our task is to attempt the construction of a manufacturing procedure from the shape information in geometry.

We understand that during the design stage, an operation such as "side mill roughing" can be associated with a particular cavity being designed. Consider the simple case of a hole. It may seem that attaching a label "drill" to a graphical representation would solve our problem. Not exactly. There are other key information that must be available before such a "drill" subroutine can be called. For example, is the hole all the way through? If not, how should the workpiece be set up, or from what direction should the hole be drilled? Should the hole be drilled together with other holes? Are there other holes of the same size? These are some of the specific questions normally asked by a part programmer when examining a design.

It would seem very convenient if a program can manipulate the representations of cavities at the level of bodies. As we learned in engineering graphics, a part can be visualized as a composition of simpler objects in an exploded view. This idea has been implemented in a computer aided design system by I. C. Braid.[8] Consequently, we assume an algebraic description of a machine part as obtained from Braid's system, i.e., in terms of a set of

* This work was supported under the Joint Services Electronics program (U.S. Army, U.S. Navy, U.S. Air Force) under contract DAAB-07-72-C-0259.

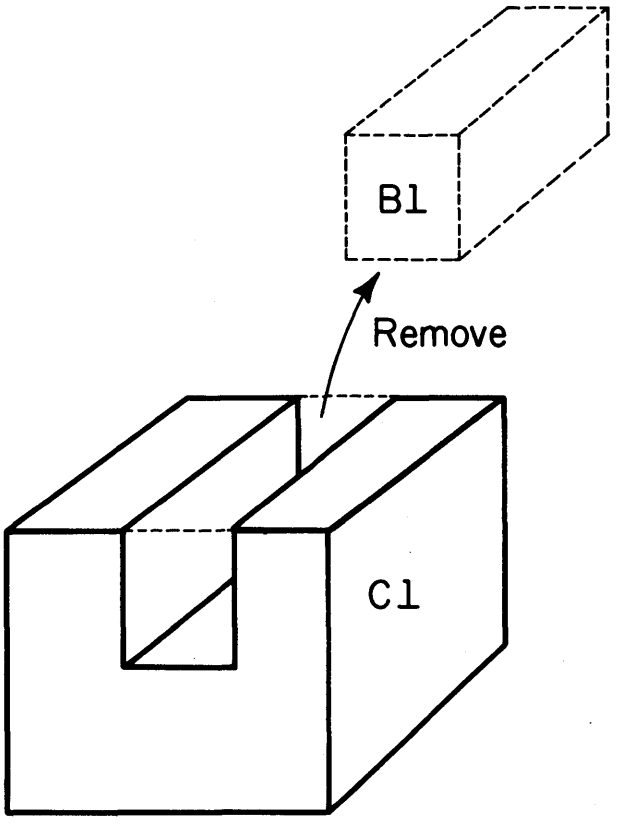

Figure 1—Create a slot by removing

primitive bodies such as cubes and cylinders, and relations such as adding and removing.

We see such a description as a very good way of telling to the computer how a machine part looks like, in a language of bodies. It is interesting to note, however, that there are many ways in which the same machine part can be described using the same set of primitives and operations. This is not particularly surprising because we humans can express an idea in natural language in just as many, if not more, ways using different components such as words, phrases, and clauses, and different arrangements of the components. Our problem here is in interpreting a body description of a machine in such a way that different descriptions of the same object should have the same interpretation.

To understand the problem of interpreting the geometry of a design, let us consider an example of a slot. A "slot" can be represented, as in Braid's system, by a negative body B1 initially transformed from a primitive cube (by scaling in the X direction, for example). If this negative body B1 is removed from a larger cube C1, a "slot" is formed. See Figure 1. The same object can be described by building up on both sides of the slot (yet to be formed). As in Figure 2, one could add two long blocks, B2 and B3, on a flat block B4.

We have just seen two of the many ways of syntactically representing a cavity in a computer. In order for the computer to carry out automatic operations such as using the right cutting tool, approaching the workpiece in the right direction, move the cutting tool a proper distance, it must first understand what it is going to cut. In other words, a program must first relate the geometry of a "slot" to information such as part surface, tool diameter, tool locations.

PARSING THE DESCRIPTION

A major concern in dealing with part descriptions is that it is a many-to-one geometry to machining concept mapping problem. In order to capture the intent of a design, we have constructed a grammar that transforms a design description into an internal representation in terms of machining operations. The grammar handles the six primitive bodies (cube, cylinder, fillet, sector, tetrahedron, and wedge) and commands (translate, rotate, scale, copy, negate, combine, and intersect) used in Braid's system.

Let us first analyze the structure of a design. We define an object (OBJ) as a concatenation of a modifier (MOD) and an object, where an object may be a primitive object or a modified object, and a modifier may be any combination of the three transformations—translation, rotation, and scaling.

OBJ = MOD OBJ
OBJ = primitive body/OBJ
MOD = translate/rotate/scale/MOD

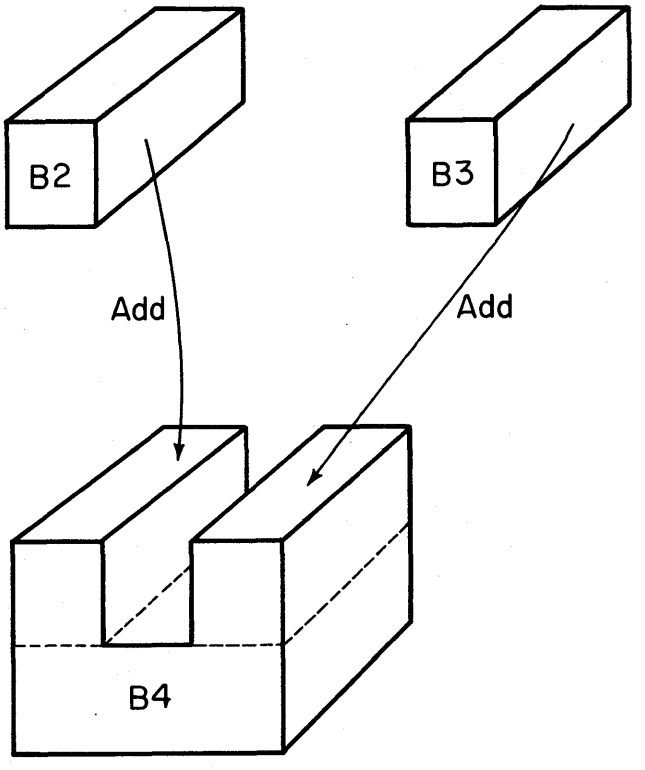

Figure 2—Create a slot by adding

An object can be operated on by copying, or negating it. We call the concatenation of an operation on a body an object group (OG).

OG=OP OBJ
OP=copy/negate

A machine part can be composed by joining several object groups together either by combining and merging the surfaces of the bodies involved, or by intersecting and creating new surfaces. A part is therefore a string of object groups joined together.

PART=OG JOIN OG
JOIN=combine/intersect

Having analyzed the structure of constructing a part from primitive bodies, we need to define the grammar for machining terminologies such as holes, slots, and pockets. Our grammar parses the design description by looking for possible constructs of such cavities. Since they are in the form of programs, they are best explained as decision procedures.

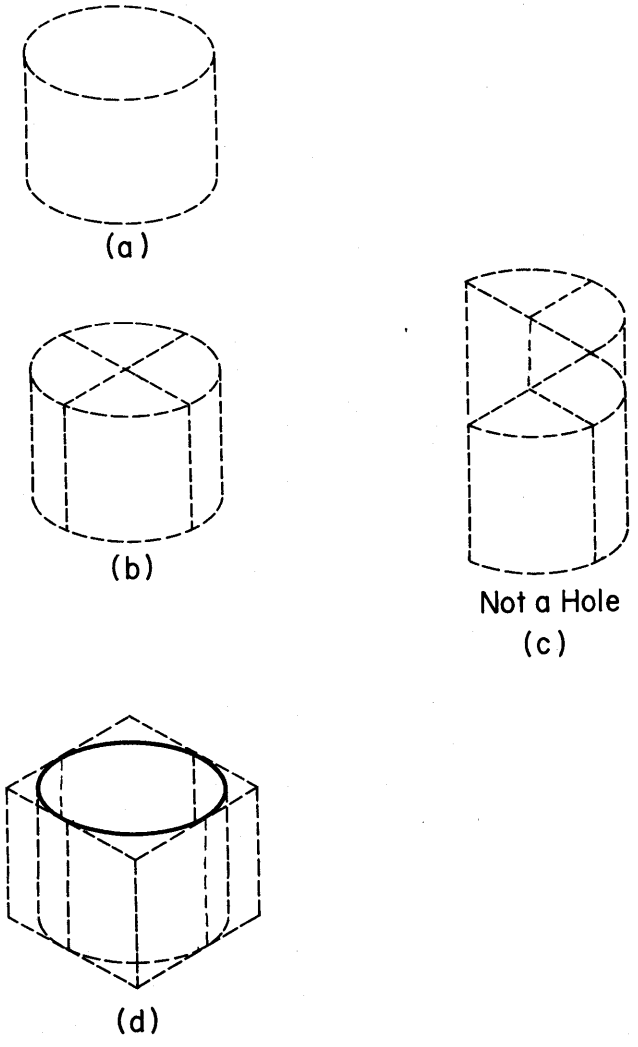

Figure 3—A hole procedure

We define a hole as a negative cylindrical object. A cylindrical object may be a modified primitive cylinder, e.g., a rotated and scaled cylinder. It could also be four sectors joined together forming a cylinder. See Figure 3b. Since joining four sectors together arbitrarily does not necessarily form a cylinder, as shown in Figure 3c, we look at the modifier for each sector involved to make sure that they are of the same size (the same scale modifier) first. We further note that rotating a symmetrical object with respect to a certain axis is a modulo operation. For instance, rotating a cube, scaled in the X-axis, 180 degrees around any of the three axes does not cause any change. Our program takes this into account and eliminates redundant transformations that may occur in the design. We also check the translation of each sector involved to make sure that the surfaces "mate" in the right manner. When these conditions are met, our program concludes that there are four bodies in the description of a machine part that forms a hole. A third possibility of a hole is the intersection of a cube with four negative fillets, i.e., taking away the corners of a cube. See Figure 3d. A similar procedure for checking each body involved is carried out.

Our program is context sensitive in that it not only checks the constituents in an algebraic expression of bodies and relations for a possible cavity, it also checks the relationship between the cavity and the body in which the cavity is supposed to reside. An example of this context sensitive aspect of our grammar is finding a rectangular open slot created in the manner shown in Figure 2. Our program first checks the scale, translate, and rotate modifiers of the two objects to make sure that there are indeed two parallel surfaces belonging to two different bodies flanking the "slot." It then concludes a cavity and checks for the size of the formed cavity with respect to the size of the two objects creating it. See Figure 4b. If the cavity is too wide, program perceives the two blocks as some sort of "walls" and does not treat the cavity as a slot. Our program also recognizes the importance of the relative sizes and orientations of the two bodies creating the slot and the body on which the slot is intended. If the two bodies are too small, the possibility of a slot is rejected. Similarly, if the two bodies are not oriented correctly, as in Figure 4c, the result is not interpreted as a slot.

MODEL OF NUMERICAL CONTROL MACHINE TOOL

After the cavities are interpreted, the results are passed onto another set of programs that produces an internal representation of the cavities in terms of cutting tool, tool diameter, approaching and cutting surfaces, and tool locations in space. Very often, our program returns more than one way of machining a particular cavity. This information is kept and saved for later processing.

At present, we have a model of a numerical control ma-

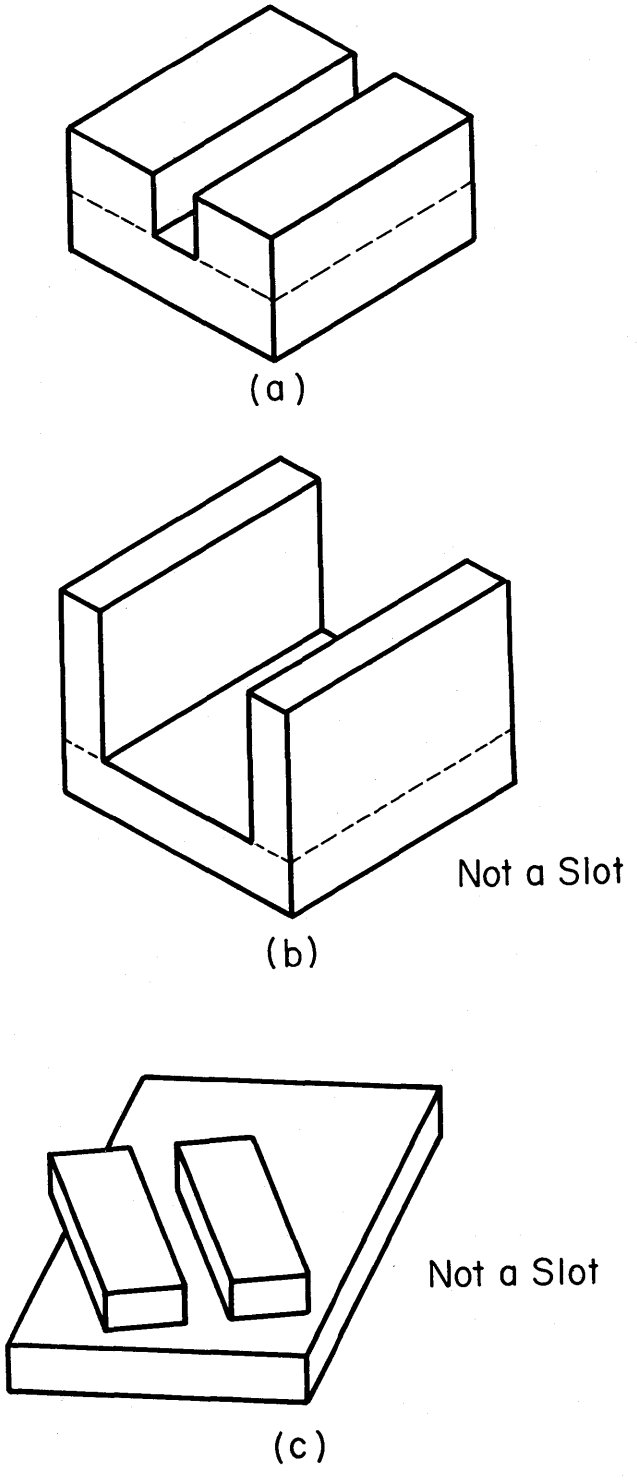

Figure 4—A slot procedure

chine tool which has the capability of 3-axis drilling and milling. Our model is built around the idea of a negative cylinder propagating in space. There are three such cylinders corresponding to the shapes of the most elementary cavities created by a drill, a side mill, and an end mill. Strictly speaking, the cylinders differ only in the bases, i.e., they are pointed, flat, and spherical, respectively.

The model is a collection of programs that simulates the cutting motions of the three kinds of tools. A slot, for example, is modeled as a two dimensional propagation of a flat-based cylinder moving in a straight line or following an arc. The result is a list of tool locations.

For bodies with complicated boundaries, such as a pocket with islands, a more general algorithm is needed to generate the tool paths. The basic ideas are partitioning the area enclosed and sequencing the partitions. This method is particularly useful in dealing with pockets with a concave boundary and with islands inside the pocket. Since the boundary is composed of line segments and arcs, local concavity points are first computed. They are next sorted according to their locations. The area enclosed is then partitioned into several regions with parallel line segments connecting the local concavities to the boundary. These regions are then traced and labeled. Since cutting is assumed in an increasing X, increasing Y manner, the regions form a lattice. They are partially ordered in the sense that a region cannot be cut unless the ones below it have all been cut. Lattice traversing algorithms have been developed for this purpose. The entire boundary including those for the islands is then digitized using scan-line conversion technique.[9] The digitized boundary is then offset and sorted in the increasing X, increasing Y order; the points provide a zig-zag tool path that runs between the boundary of the pocket and that of the islands.

In general, before these programs are called, more information on the cavity should be obtained. A special program for a specific kind of cavity is called to analyze the tool required. At present, there are ten sizes for each of the three types of cutting tools available. If, for instance, a hole is too large for drilling, an alternative machining method, milling in this case, is taken. In addition, the program checks what surfaces of the cavity in question are external, hence accessible to the cutting tool, and what surface the tool should reside on (the part surface in APT). If there is more than one possibility in which the tool could approach the cavity and does not penetrate other parts of the workpiece, they are reported to programs to be described in the next section.

GROUPING AND SEQUENCING

With the cutting tools selected, and the possible approach surfaces available, our program next sorts the cavities into groups. Basically, the program attempts to use the same cutter as much as possible without another set-up.

The algorithm proceeds by first grouping all cavities according to the plane on which their approach surfaces lies. This implies the number of set-ups. Very often, a cavity occurs in different groups because it may be cut from a number of directions. The multiple occurrence is first

partially eliminated by considering the number of identical operations within a group. If there are less than three of the same kind, and if the group is not the last one in the list, the occurrences are deleted from the group. There may still be cavities occurring in more than one group after this operation. A semi-circular cut-out is an example. It can be interpreted as a hole and as a slot, thus it can be machined in more than one way provided the required cutting tools are available. The groups are next divided into subgroups according to the type of cutting tool needed. An elimination of multiple occurrence may happen if there are less than three cavities using the same kind of tool. If multiple occurrence still exists, drilling operation is given the preference to milling.

CODE SYNTHESIS

Our code synthesizer is quite simple. It is APT-like and has two kinds of information. If a list of points is given to it, the result is a series of statements of the form:

GOTO/Pi

If a list of line segments and arcs are supplied, it produces statements of the form:

GO*/Li%Li+1

where the modifier, *, is to be replaced by LFT or RGT, and % is to be replaced by TO, ON, or PAST, depending on the angular relationships between Li and the segment preceding it Li-1, and the one following it, Li+1. A computer graphics package is written that translates these statements into graphics commands, thus enabling one to visually examine the tool paths on a machine part.

CONCLUSION

We view our system as a pilot study of a totally integrated manufacturing automation system. Our objective, as we indicated earlier, is to create an environment in which design information can be utilized to produce manufacturing procedures directly and automatically.

We have shown that the problem of manufacturing is one of transforming graphical descriptions into program descriptions. We have divided the problem into two stages—obtaining from the many ways of describing a machine part in primitive bodies a kernel description of a machine part in terms of cavities, and interpreting the cavities in terms of the capabilities of a numerical control machine tool. It is hoped that our work provides the basis for bridging the automation gap between design and fabrication.

REFERENCES

1. Coons, S. A., *Surfaces for Computer-Aided Design of Space Forms*, MIT MAC-TR41, June, 1967.
2. Bezier, P., *Numerical Control—Mathematics and Application*, John Wiley Sons, 1970.
3. *POLYSURF User Manual*, Computer Aided Design Centre, Cambridge, England, 1974.
4. *SSX4 Sculptured Surfaces Project*, Computer Aided Manufacturing International, Arlington, Texas, 1974.
5. Bockholts, P., "TNO Miturn Programming System for Lathes," *Proceedings of PROLAMAT* '73, Budapest.
6. Hellstrom, P., "An Interactive System for Operations Planning for Turning on Centre Lathes," *Proceedings of PROLAMAT* '73, Budapest.
7. Sohlenius, G., "CAM-PRAUTO-DRILLING," *Proceedings of CAM-I Congress*, Hamilton, Ontario, Canada, May, 1974.
8. Braid, I. C., *Designing with Volumes*, Cantab Press, Cambridge, England, 1973.
9. Metzger, R. A., "Computer Generated Graphic Segments in a Raster Display," *Proceedings of SJCC*, 1969, pp. 161-172.