To be presented at Manufacturing International'90, March 25-28, 1990 Atlanta, Georgia.

Handling Feature Interactions in Concurrent Design and Manufacturing [‡]

Dana S. Nau Raghu R. Karinthi

Department of Computer Science
University of Maryland

ABSTRACT

To achieve the goals of concurrent engineering in the design and manufacture of machinable parts will require extensive communication between CAD systems such as solid modelers, and CAM systems such as generative process planning systems. One of the key steps in communicating between a process planning system and a solid modeler is the derivation of machinable features from the solid model. Regardless of whether the features are derived using feature extraction, design by features, or some other approach, geometric interactions among the features can create situations where there are several possible feature representations for the same part. This presents a problem for generative process planning, since some of these representations may lead to feasible process plans and some may not. To address this issue, we are developing an algebra of feature interactions. The algebra is used as the basis of a feature transformation system which is being implemented. This will be integrated with our Protosolid solid modeler [22] with our SIPS process planning system [14].

NOMENCLATURE

$\cup^*$	Regularized Union
$\cap^*$	Regularized Intersection
$-^*$	Regularized Subtraction
c^*	Regularized Complement
$\mathcal{S}$	Shortened
$\mathcal{I}_p$	Infinite Extension
$\mathcal{I}_f$	Face Infinite Extension
$\mathcal{M}_p$	Maximal Extension
$\mathcal{M}_f$	Face Maximal Extension

[‡]This work supported in part by NSF Presidential Young Investigator award DCR-83-51463, with matching funds provided by Texas Instruments and General Motors Research Laboratories, and by NSF Engineering Research Centers Program Grant NSFD CDR-88003012 to the University of Maryland Systems Research Center.

Introduction

Many of the problems faced by modern industry are related to a lack of coordination between design and manufacturing. Typical problems include inconsistencies among process plans for similar designs, large discrepancies from optimal shop utilization, poor product quality, and non-competitive costs. Recently, there has been increasing awareness of the importance of taking manufacturing considerations into account during the design of the part, rather than afterwards. This concept is known by a variety of terms, such as "concurrent engineering", "concurrent design and manufacturing," and "design for manufacturability."

Design for manufacturability requires that the part designer should not confine himself to the traditional role of just developing products to meet specified functional requirements, but should also actively consider the associated manufacturing implications. Thus, as the part design is being developed, issues such as availability of resources including machine tools, cutting tools, jigs and fixtures, and labor, as well as their particular capabilities and costs should be considered. Also, required manufacturing, assembly and inspection operations should all be considered at the design level. This calls for a detailed knowledge of the capabilities of the manufacturing shop, which normally resides with the process planning department.

If design for manufacturability is to be achieved, CAD systems of the future must incorporate interfaces to modules such as process planning systems, to evaluate the manufacturability of the design. Furthermore, it will be necessary for such process planning systems to reason about geometric relationships among the various parts of an object while the design is underway. To carry out such reasoning automatically will require extensive interaction between the process planning system and the CAD system (presumably a solid modeler) during design and process planning.

One of the primary problems in communicating between a solid modeler and a process planning system is the derivation of machinable features from the solid model. Regardless of whether the features are derived using feature extraction, design by features, or some other approach, geometric interactions among the features can create situations where there are several possible feature representations for the same part. This presents a problem for generative process planning, since some of these representations may be feasible for manufacturing and some may not.

To address this issue, we have developed an algebra of feature interactions. Given one valid interpretation of a machinable part as a collection of machinable features, all other valid interpretations of it as other collections of machinable features can be derived through operations in the feature algebra. We intend to use this algebra as the basis of a feature transformation system (currently being implemented), which will be capable of examining the feature descriptions, computing alternate feature representations for an object, and presenting them to a process planning system. Once the implementation has been completed, it will be used as the communication interface between our Protosolid solid modeling system [22] and our SIPS process planning system [14], in the development of an integrated system for design and process planning.

Feature-Based Design and Analysis

CAD-generated objects can be defined in terms of the complete geometry of the part. The descriptions contain solids, faces, edges and vertices making up the part. For CAM descriptions of the objects, the geometry and topology are the same, but the meaning associated with this geometric structure is different, and dictates a change in the description. A form feature which the designer may think of as a tab sticking out of a block of metal will be considered by the manufacturing engineer to be the material left after some shoulders have been cut out of a larger block of metal.

There have been various solutions proposed to solve this incompatibility; several of them are discussed below.

1. *Automatic feature extraction* consists of automating (algorithmically?) the task of determining the manufacturing features of a part from existing CAD databases such as IGES files, BReps, etc. The general algorithms developed are of the recognise-feature/extract-feature genre. Prominent among these are the ones by Henderson [5], Kyprianou [12], De Floriani [3], and Srinivasan [20].

   Some of the more significant problems with feature extraction are as follows:

   (a) Some attributes of a machined part cannot be made without reference to a particular feature (for example, the surface finish, corner radius, and machining tolerances of a pocket). When an object is designed without making reference to these features explicitly, it is unclear how to associate the machining specifications with the proper features.

   (b) It is difficult to extract a feature which intersects or otherwise interacts with other features, without disturbing those other features. For example, in Henderson's feature extraction system, once a feature volume has been recognized it is subtracted from the overall cavity volume—making it impossible to obtain multiple feature interpretations for the same cavity volume.

2. In *design by features*, the user builds a solid model of an object by specifying directly various form features which translate directly into the relevant manufacturing features. Systems for this purpose have been built for designing injection-molded parts [21], aluminum castings, [13], and machined parts [10, 6, 1, 11].

   In the case of machined parts, one problem with design by features is that it requires a significant change in the way a feature is designed. Traditionally, a designer designs a designs a part for functionality, and a process engineer determines what the manufacturable features are. However, design by features places the designer under the constraints of not merely having to design for functionality, but at the same time specify all of the manufacturable features as part of the geometry—a task which the designer is not normally qualified to do. Another problem—that of alternate feature interpretations—is described in the next paragraph.

3. *Human-supervised feature extraction* overcomes one of the problems of design by features, by allowing the designer to design the part in whatever way is most convenient,

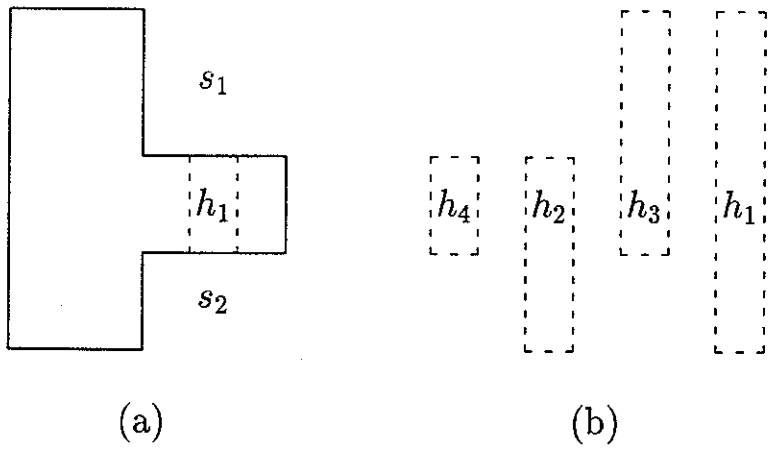

Figure 1: Several possible interpretations of a hole in an object.

and then requiring the process engineer to identify the machinable features of the part. A system for this purpose was built at General Motors Research Laboratories, and another is being built at the National Bureau of Standards [2] using Unicad/Romulus [19].

Human-supervised feature extraction provides a way for a qualified manufacturing engineer to identify the machinable features–but it still does not handle the problem of alternate feature interperetations. Sometimes a machinable part can be specified as a set of machinable features in more than one way—and unless one is very careful to identify all such possible interpretations, this can lead to significant problems in process planning.

Consideration of the above approaches makes it evident that no matter what approach is used for designing a part, a major problem to be solved is how to handle feature interactions. Geometric interactions among two or more features may lead to several alternative interpretations as to the identity and dimensions of those features—and these interpretations will dictate restrictions on the order in which the features are to be made.

Geometric Interactions and Planning

Due to geometric interactions among the features, there may be several equally valid sets of features describing the same part. To produce a good process plan—or, in some cases, even to produce a process plan at all—it may be necessary to use a different interpretation of the part than the one obtained from the CAD model. The problem is how to find these alternate interpretations. A machinist solves this problem because of the human ability to visualize an object in three dimensions and by doing geometric reasoning to figure out the interactions.

To illustrate the role of feature interactions, consider the part depicted in Fig. 1. In this example, the part has been described as the part resulting from subtracting a hole h_1, a slot s_1 and a slot s_2, in that order, from a rectangular stock. Because of the interaction of h_1

with s_1 and s_2, we get $h_2 = h_1 -^* s_1$, $h_3 = h_1 -^* s_2$, and $h_4 = h_2 -^* s_2 = h_3 -^* s_1$. The final set of features used for machining would be s_1, s_2 and one of h_1, h_2, h_3 and h_4. Which hole to use depends on issues such as cost criteria (whether it is cheaper to make a deep hole or a small hole), feasibility (availability of proper tools to make a deep hole), fixturing criteria (can the part be properly fixtured to make h_4 after s_1 and s_2 have been made, or will it vibrate). Thus, one can see that there can be several possible interpretations for the same part, such as $\{h_1, s_1, s_2\}$, $\{h_2, s_1, s_2\}$, $\{h_3, s_1, s_2\}$ and $\{h_4, s_1, s_2\}$. Having only one of them can be a serious limitation in process planning.

The Algebra of Features

Earlier we illustrated the importance of having alternative feature interpretations through Figure. 1. This means one needs to be able to translate from one set of features that describe a part to equivalent sets that describe the same part. We have developed an algebraic approach to do this, which is described below.

An algebraic structure [16] in its simplest form is a set, with a rule (or rules) for combining its elements. Let A be any set. An *operation* $*$ on A is a rule which assigns to each ordered pair $\langle x, y \rangle$ of elements of A exactly one element $x * y$ in A.

In particular, the feature algebra is characterized by a set of features (denoted by $\mathcal{D}$), and binary operations on the features. Since these operations give meaningful values only for certain pairs of features, we include an element called INVALID in the set of features, to be used in cases where the operations do not produce meaningful values. By definition, for any operation $*$,

$$\forall x,\ x * \text{INVALID} = \text{INVALID} * x = \text{INVALID}$$

The essential features of the algebra are the operations on features and the algebraic properties of the operations. Originally, we developed the algebra for a small set of features [8, 9] and we have subsequently generalized it to cover practically all features of interest to manufacturing. The generalized feature algebra is described briefly in this paper. The reader is referred to [15] for a detailed treatment of the material and underlying mathematical concepts.

Feature Definition

A feature (other than INVALID) is given by a pair $x = \langle \text{ps}, \text{binfo} \rangle$ where the two entries in the pair satisfy the following conditions:

1. The first entry (ps(x)) is the set of all points in a feature and is a subset of E^3 that is compact, regular and semi-analytic.

2. The second entry (binfo(x)) is a partition of the boundary of ps(x) into regular semi-analytic components, each of which is labeled as BLOCKED or UNBLOCKED.

It is worthwhile now to examine the scope and significance of the above definition. Regularity restricts a feature to be homogeneously three dimensional. Even parts with sheet metal components have a finite thickness, so this appears to be a very reasonable restriction. Since the features that are considered are of finite dimensions, they are bounded and hence

compact. The domain of semi-analyic sets covers practically all the shapes of interest to manufacturing. The reader may note that all planar polyhedra, cylinders, cones, spheres, tori and a variety of sculptured surfaces are encompassed by this set. It also includes concave features such as T-slots, counter-bores and counter-sinks (see Figure 3).

A *patch* is a regular, semi-analytic subset of the boundary of a feature. Figure 2 illustrates some examples of patches. Thus, binfo(x) is a partition of the boundary of ps(x) into a collection of patches each of which labeled BLOCKED or UNBLOCKED. Henceforth, the boundary and the patches of ps(x) will be referred to as the boundary and patches of x. For any feature x, we denote a set of patches of x by patches(x). For any patch p, we denote the label of the patch by label(p).

The labeling of patches of a feature as BLOCKED or UNBLOCKED is an aid to the process planning system in determining the faces of a feature that are reachable by a cutting tool. Let S_i be the part after a feature x_i has been created. The patches of x_i that belong to the boundary of S_i are labeled BLOCKED. The rest of the patches of x_i are labeled UNBLOCKED.

Given a feature x and a patch p of x, the regularized complement of p is defined as $c^*(p, x) = b(\text{ps}(x)) -^* p$.

Proposition 1 *The set of all patches of a feature is closed under regularized union, intersection, complement and difference.*

Proof: It can be shown that the interior, boundary, and closure of a semi-analytic set is also semi-analytic, and that class of compact, regular, semi-analytic sets is closed under regularized set operations (see [15]). This proposition a direct consequence of the closure properties of compact, regular, semi-analytic sets.

Operations on Features

This section describes the operations on features. In order to describe the propagation of patch labels we need to define additional concepts related to classification of a patch with respect to a solid. Due to lack of space, we describe only the first component: the set of all points of a feature.

Shortened

Given two features x and y, the operation shortened ($\mathcal{S}$) is defined as follows: $z = x\mathcal{S}y = \langle u, v\rangle$, where $u = \text{ps}(x) -^* \text{ps}(y)$. The second component is omitted from the discussion.

Infinite Extension

In this section the *infinite extension* of a feature with respect to a patch will be defined. This is used subsequently in defining an operation called *maximal extension*. For simplicity we will present here the definiton of infinite extension only for convex solids. Figure 4 illustrates some examples of infinite extension.

Given a patch p on a feature x, let the tangent plane (if the tangent plane is defined) at a point $p_i \in c^*(p, x)$ be given by the equation $\mathcal{T}(p_i, x) = 0$. $\mathcal{T}(p_i, x) = 0$ divides E^3 into two half-spaces, one of which includes ps(x). We will denote this half-space by $H(p_i, x)$. The infinite extension of x with respect to the patch p denoted by $\mathcal{I}_p(x)$ is defined as follows:

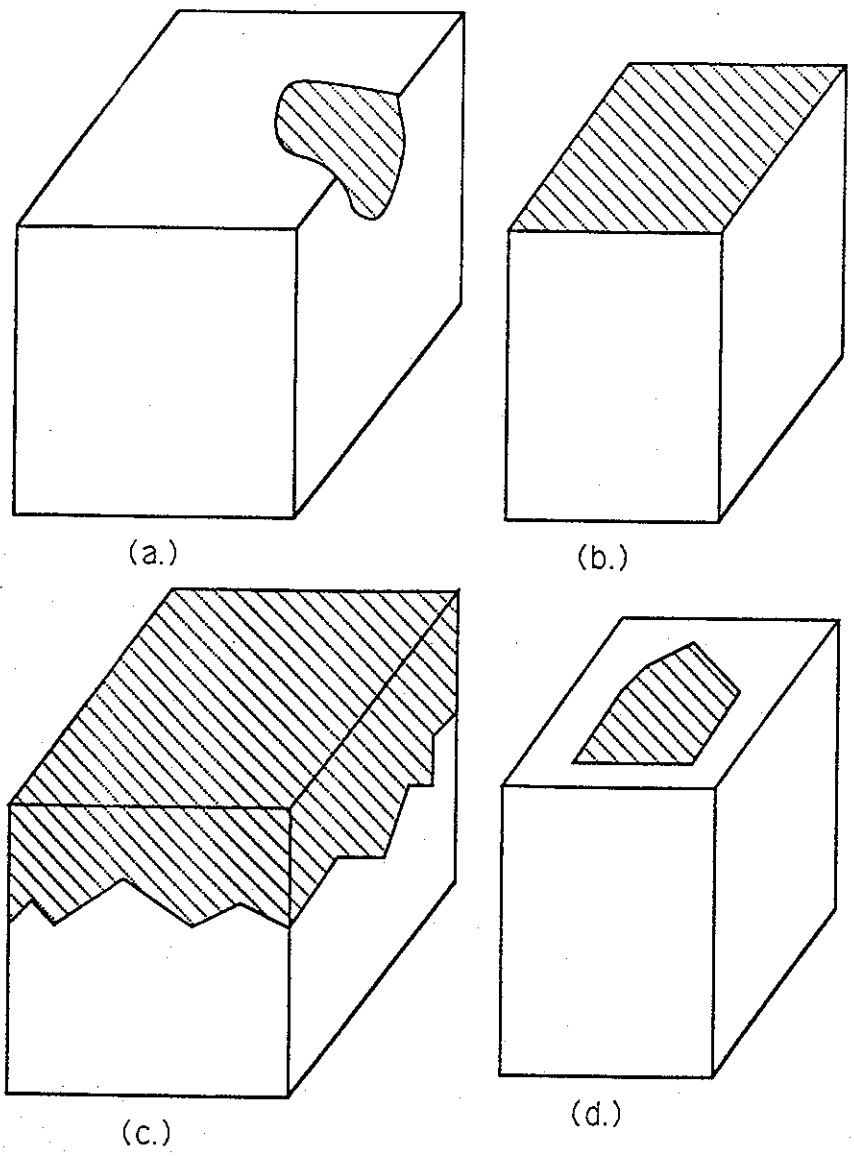

Figure 2: Some examples of patches (shaded).

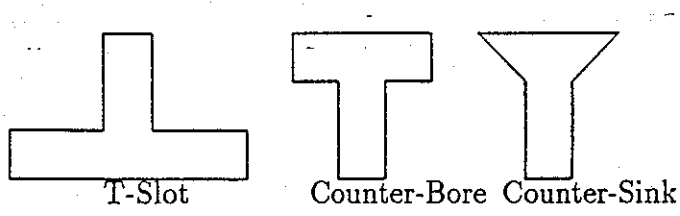

Figure 3: Cross-sectional views of three concave features.

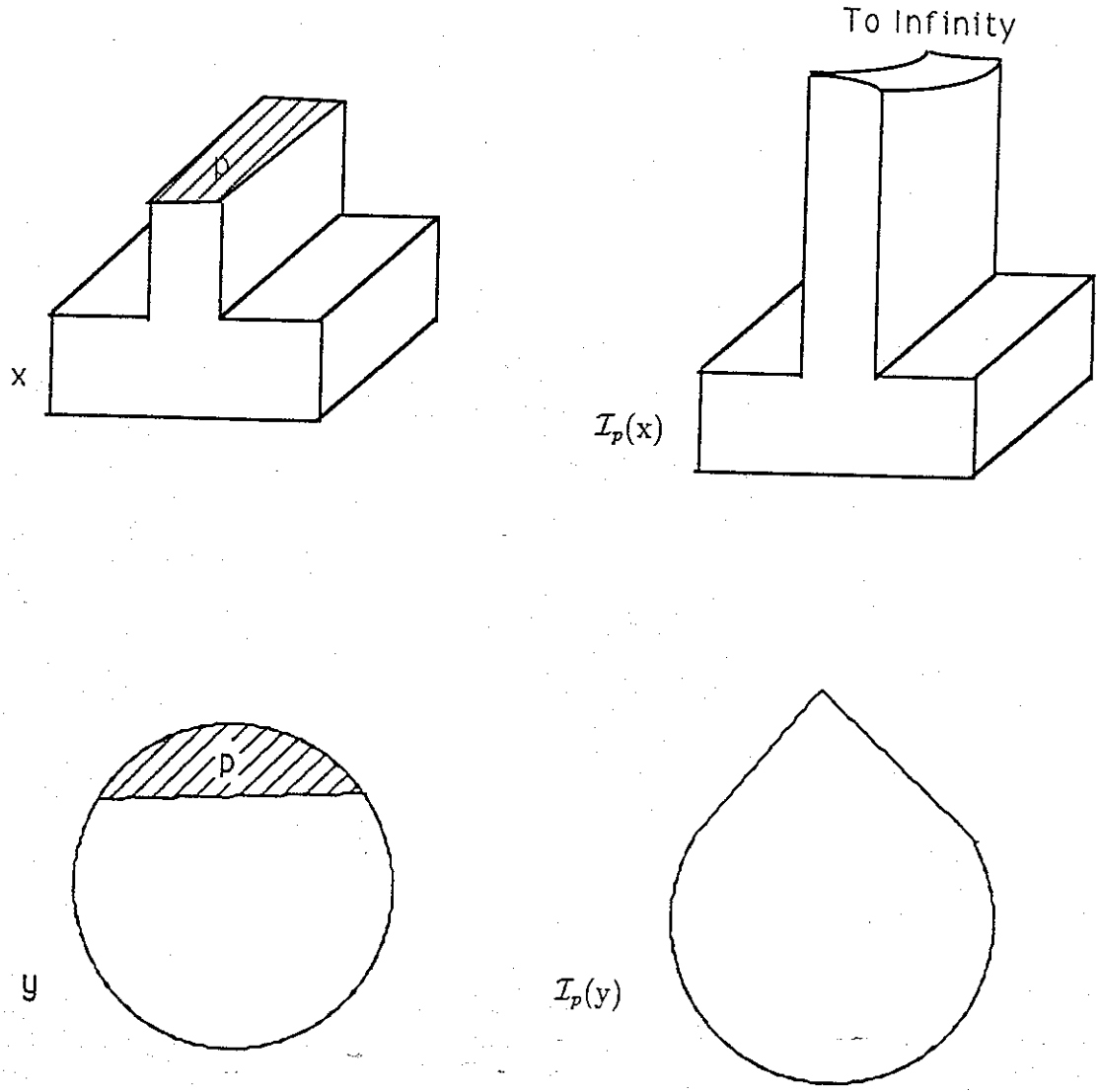

Figure 4: Examples of patches and infinite extension.

$$\mathcal{I}_p(x) = \bigcap_{p_i} H(p_i, x)$$

where $p_i \in c^*(p, x)$ and $\mathcal{T}(p_i, x) = 0$ (and hence $H(p_i, x)$) is defined at p_i.

Maximal Extension

Given two features x and y, and a patch p on x, the maximal extension of x in y with respect to a patch p (denoted by $x\mathcal{M}_p y$) is defined as follows:

$$x\mathcal{M}_p y = \begin{cases} \langle u, v \rangle & \text{if } \mathcal{I}_p(x) \neq INVALID \\ INVALID & \text{otherwise} \end{cases}$$

where $u = \mathcal{I}_p(x) \cap^* (\text{ps}(x) \cup^* \text{ps}(y))$.

Properties of the Algebra of Features

The goal is to generate new features from the features one already has, using the operators discussed earlier. During this process, one would not like to generate a feature anew if it can be shown that a feature equivalent to it has already been generated. To compute new features from existing ones, one must perform set operations on solids. Existing algorithms for performing set operations have an average case complexity of $O(n \log n)$, where n is the number of topological entities in a solid (such as vertices, edges and faces). Thus, set operations on solids are expensive computations and should be minimized or substituted by cheaper operations. We have proved several properties of the operations on features. Below, we illustrate two of our results:

Proposition 2 *For any three features x, y and z, the following result holds:*

$$(ps(x) -^* ps(y)) -^* ps(z) = (ps(x) -^* ps(z)) -^* ps(y).$$

Proposition 3 *Given a feature x and a patch p of x, if $\mathcal{I}_p(x) = ps(x)$, then $ps(x\mathcal{M}_p y) = ps(x)$, for any feature y.*

Restricted Feature Algebras

The previous section described an algebra of features, viz. the domain, the operations and the properties. The domain includes almost any feature of interest to manufacturing. A crucial question regarding the operations in the feature algebra is their computability; whether they can be computed and if so, the efficiency. As defined, the operation $\mathcal{M}_p$ involves intersecting an infinite number of half-spaces. This definition itself does not specify any algorithm, and algorithms for computing $\mathcal{M}_p$ for features and patches of various shapes must be developed. One must also identify the subset of the patches are of interest in computing infinite extension and maximal extension.

We have addressed these issues for a restricted domain consisting of rectangular solids and cylinders that have their planar faces parallel to the faces of the stock *. Rectangular solids

*This scheme will be implemented with rectangular solids modeled with rounded corners. In this paper, the corners are taken to be pointed, as it simplifies the discussion.

occur as manufacturing features known by a variety of names, such as a *slot* (which in turn could be *single-ended* or *through*), a *shoulder*, a *pocket*, a *cut-out*, a *notch* etc. The common manufacturing feature that is cylindrical in shape is a *hole*.

For the this domain, the patches whose $\mathcal{I}_p$ is computed are chosen to be the the planar faces of a feature. Let us call this restricted form of infinte extension as *face infinite extension* ($\mathcal{I}_f$) and the corresponding maximal extension as *face maximal extension* ($\mathcal{M}_f$). Since there are six planar faces in a rectangular solid and two in a cylinder, there are six possible face infinite extensions for a rectangular solid and two for a cylinder. Let us denote the set of all possible face infinite extensions by Σ_I and the set of all possible face maximal extensions by Σ_M. The procedures for computing the operations are described in [15].

The Features Algorithm

Given a set of features that describe a part, one would like to generate alternate sets of features that describe the part. Any set of features describing the part under consideration is called a feature set. This section describes an algorithm for generating alternate feature sets given one feature set. In this algorithm, $\bar{F}$ is the set of features generated at any stage, and F is the union of all the features in $\bar{F}$. Initially, F is the starting set of features and $\bar{F} = \{\mathrm{F}\}$. There are two additional variables called CURRENT and NEW used in this algorithm. Let Σ be the set of applicable binary operations. Thus, $\Sigma = \{\mathcal{S}\} \cup \Sigma_M$.

In the algorithm shown in Figure 5 the function new-ps-member (x, y) returns *true* if $\forall z \in y$ $\mathrm{ps}(x) \neq \mathrm{ps}(z)$ and *false* otherwise. In this algorithm we have to determine if if $x\eta y$ is a valid feature and if so, whether new-ps-member$(x\eta y, F)$ is true or false. The properties of the feature algebra are used in these two steps. If it is possible to determine if $x\eta y$ is INVALID or if new-ps-member$(x\eta y, F)$ is true using the properties of the feature algebra, then one need not do any further computations. Otherwise, one has to compute $x\eta y$ using the procedures described in [15] and then determine if it is a new feature.

At first glance, the worst case complexity of the algorithm might appear to be exponential, because of the possibility of combinatorial explosion if there are several mutually-interacting features. However, geometric locality dictates that each feature will interact with only a few of its neighbors, so there is no reason to believe that significant exponential blowup would ever occur.

Illustrative Example

This section illustrates the working of the algebra of features (and the features algorithm), through the example shown in Figure. 1. The starting features are h_1, s_1 and s_2. In this section, the example is illustrated, by showing only the changes that occur from one iteration to the next of the outermost **while** loop (without tracing every step of the algorithm). A complete trace of the algorithm for this example is left as an exercise to the reader.

At the start of the features algorithm the values of the variables are:

$$\mathrm{F} = \{h_1, s_1, s_2\} \ ;$$

$$\mathrm{CURRENT} = \{h_1, s_1, s_2\} \ ;$$

$$\mathrm{NEW} = \emptyset \ ;$$

$$\bar{F} = \{\{h_1, s_1, s_2\}\} \ .$$

```
F = Starting set of features;
F̄ = {F};
CURRENT = F;
NEW = ∅
for each x in CURRENT for each I_f ∈ Σ_I compute I_f(x) ;
while CURRENT ≠ ∅ do
  for each FS in F̄ do
    for each ⟨x, y⟩ : x ∈ FS and y ∈ FS and x ≠ y do
      if (x ∈ CURRENT) or (y ∈ CURRENT) then
        for each η ∈ Σ do
          if xηy = z then
            if new-ps-member(z, F) then
              for each I_f ∈ Σ_I compute I_f(z) ;
              F = F ∪ {z};
              NEW = NEW ∪ {z};
            end if ;
            if ((FS − {x}) ∪ {z}) ∉ F̄ then
              F̄ = F̄ ∪ ((FS − {x}) ∪ {z});
            end if ;
          end if ;
          if xηy = {z1, z2} then
            for each u ∈ {z1, z2} do
              if new-ps-member(u, F) then
                for each I_f ∈ Σ_I compute I_f(u);
                F = F ∪ {u};
                NEW = NEW ∪ {u};
              end if ;
            end for ;
            if ((FS − {x}) ∪ {z1, z2}) ∉ F̄ then
              F̄ = F̄ ∪ ((FS − {x}) ∪ {z1, z2});
            end if ;
          end if ;
        end for ;
      end if ;
    end for ;
  end for ;
  CURRENT = NEW;
  NEW = ∅;
end while .
```

Figure 5: The Features Algorithm

The value of NEW is $\emptyset$ each time before the outermost loop is entered. After the first iteration:

$$\mathrm{F} = \{h_1, s_1, s_2, h_2, h_3\} \ ;$$

$$\mathrm{CURRENT} = \{h_2, h_3\} \ ;$$

$$\bar{F} = \{\{h_1, s_1, s_2\}, \{h_2, s_1, s_2\}, \{h_3, s_1, s_2\}\} \ .$$

In this iteration, h_2 and h_3 were generated using the $\mathcal{S}$ operation, where $h_2 = h_1 \mathcal{S} s_1$ and $h_3 = h_1 \mathcal{S} s_2$.

After the second iteration:

$$\mathrm{F} = \{h_1, s_1, s_2, h_2, h_3, h_4\} \ ;$$

$$\mathrm{CURRENT} = \{h_4\} \ ;$$

$$\bar{F} = \{\{h_1, s_1, s_2\}, \{h_2, s_1, s_2\}, \{h_3, s_1, s_2\}, \{h_4, s_1, s_2\}\} \ .$$

In this iteration, let us say $h_2 \mathcal{S} s_2$ was considered before considering $h_3 \mathcal{S} s_1$. Let us denote $h_2 \mathcal{S} s_2$ by h_4. Later on, when $h_3 \mathcal{S} s_1$ is considered, using Proposition 2 we infer $\mathrm{ps}(h_3 \mathcal{S} s_1) = \mathrm{ps}(h_2 \mathcal{S} s_2) = \mathrm{ps}(h_4)$. Therefore, new-ps-member$((h_3 \mathcal{S} s_1), F) = \mathit{false}$. So, $h_3 \mathcal{S} s_1$ is not considered as a new feature. Several other algebraic properties are applied at various stages.

After the third iteration :

$$\mathrm{F} = \{h_1, s_1, s_2, h_2, h_3, h_4\} \ ;$$

$$\mathrm{CURRENT} = \emptyset \ ;$$

$$\bar{F} = \{\{h_1, s_1, s_2\}, \{h_2, s_1, s_2\}, \{h_3, s_1, s_2\}, \{h_4, s_1, s_2\}\} \ .$$

At this point the loop terminates, because $\mathit{CURRENT} = \emptyset$.

Related Work

The popular approaches to CAD/CAM integration are automatic feature extraction and design by features. Most of the research in feature extraction has not addressed the issue of feature interactions. Some of the recent work such as that of Srinivasan and Liu[20] and Joshi and Chang [7] accounts for certain kinds of feature interactions. Elaborate systems ([10, 1]) based on the design by features paradigm have been built for process planning that can translate from manufacturing features to NC code. But this paradigm requires the designer to account for all the interactions among the features. The next paragraph summarizes research that has specifically addressed the issue of feature interactions.

Hayes [4] has addressed the problem of feature interactions in her Master's thesis. Her program uses feature interactions to determine precedence relations among features. The system is written in OPS5 and uses rules to detect feature interactions of interest. Requicha and Vanderbrande [18] are building a system known as the AI/SM test bed, which performs certain kinds of geometric reasoning, combining the principles of solid modeling, computational geometry and rule-based systems. This system, however, is not complete as of this writing; so we do not know of its capabilities in detail. Michael Pratt[17] has addressed several issues pertaining to feature interactions. He has developed the notions of effective volume of interaction and actual volume of interaction which are equivalent to the shortened operation. He has developed a graph showing the relationships among features. His work

addresses interactions among protrusions and depressions. There are no equivalents of the maximal extension and infinite extension in his frame work.

Conclusions

The primary issue addressed in this paper is the development of a way to reason about geometric interactions among features, via an algebra of feature interactions. We believe this algebra will have utility for a number of different approaches to design, including automated feature extraction, design-by-features, and human-supervised feature extraction. The feature algebra is being implented on Texas Instruments Explorer, and it will serve as the communication between the Protosolid [22] solid modeler and SIPS process planner [14].

This work is being done with two long-term goals in mind: the development of a practical integrated system for designing metal parts and planning their manufacture, and the investigation of fundamental issues in representing and reasoning about three-dimensional objects. We believe this work will have utility not only for automated manufacturing, but also for other problems in geometric modeling and geometric reasoning.

References

[1] S. L. Brooks and K. E. Hummel. Xcut: A rule-based expert system for the automated process planning of machined parts. Technical Report BDX–613–3768, Bendix Kansas City Division, 1987.

[2] P. Brown and S. Ray. Research issues in process planning at the national bureau of standards. In *Proceedings of the 19th CIRP International Seminar on Manufacturing Systems*, pages 111–119, June 1987.

[3] L. De Floriani. Feature extraction from boundary models of three-dimensional objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(8):785–798, Aug 1989.

[4] C. Hayes. Using goal interactions to guide planning. In *Proceedings of the AAAI-87; the Sixth National Conference on Artificial Intelligence*, pages 224–228, 1987.

[5] M. Henderson. *Extraction of Feature Information from Three Dimensional CAD Data.* PhD thesis, Purdue University, 1984.

[6] N. C. Ide. Integration of process planning and solid modeling through design by features. Master's thesis, University of Maryland, College Park, 1987.

[7] S. Joshi and T. C. Chang. Graph-based heuristics for recognition of machined features from a 3d solid model. *Computer-Aided Design*, 20(2):58–66, Mar 1988.

[8] R. R. Karinthi and D. S. Nau. Geometric reasoning as a guide to process planning. In *ASME International Computers in Engineering Conference*, July 1989.

[9] R. R. Karinthi and D. S. Nau. Using a feature algebra for reasoning about geometric feature interactions. In *Eleventh International Joint Conference on Artificial Intelligence*, August 1989.

[10] T. Kramer and J. Jun. The design protocol, part editor, and geometry library on the vertical workstation of the automated manuacturing research facility at the national bureau of standards, 1987. Internal Report.

[11] B. Kumar, D. K. Anand, and J. A. Kirk. Integration and testing of an intelligent feature extractor within a flexible manufacturing protocol. In *Sixteenth North American Manufacturing Research Conference Proceedings*, 1988.

[12] L. K. Kyprinaou. *Shape Classification in Computer-Aided Design.* PhD thesis, Christ's College, University of Cambridge, 1980.

[13] S. C. Luby, J. R. Dixon, and M. K. Simmons. Design with features: Creating and using a feature data base for evaluation of manufacturability of castings. *Computers in Mechanical Engineering*, 5(3):25–33, 1986.

[14] D. S. Nau. Automated process planning using hierarchical abstraction. *Texas Instruments Technical Journal*, pages 39–46, Winter 1987. Award Winner, Texas Instruments 1987 Call for papers on Industrial Automation.

[15] D. S. Nau and R. R. Karinthi. An algebraic approach to feature interactions, 1989. Paper in preparation.

[16] C. Pinter. *A Book of Abstract Algebra.* McGraw–Hill Book Company, 1982.

[17] M. J. Pratt. Form features and their applications in solid modelling. In *Tutorial paper on Advanced Topics in Solid Modelling at SIGGRAPH 1987*, July 1987.

[18] A. G. Requicha and Jan H. Vandenbrande. Form features for mechanical design and manufacturing. Technical Report IRIS 244, Institute for Robotics and Intelligent Systems, Univerisity of Southern California, Los Angeles, October 1988.

[19] Shape Data Ltd. *The Romulus Solid Modeling System, V6.0 User's Reference Manual Version 1*, 1985.

[20] R. Srinivasan and C. R. Liu. On some important geometric issues in generative process planning. In *Proceedings of the Winter Annual Meeting of the American Society of Mechanical Engineers, Boston, MA, December 1987*, pages 229–244, 1987.

[21] M. Vaghul, J. R. Dixon, G. E. Zinmeister, and M. K. Simmons. Expert systems in a cad environment : Injection molding part design as an example. In *Proceedings of the 1985 ASME Conference on Computers in Engineering*, 1985.

[22] G. Vanecek Jr. *Set Operations on Volumes Using Decomposition Methods.* PhD thesis, University of Maryland, College Park, 1989.