AN APPROACH TO ADDRESSING GEOMETRIC FEATURE INTERACTIONS IN CONCURRENT DESIGN

Raghu Karinthi and Dana Nau
University of Maryland
College Park, Maryland

ABSTRACT

Concurrent engineering design requires the manufacturability aspects to be considered during the process of designing a part. One of the issues of interest in this context is that of geometric feature interactions. Due to geometric interactions among the various features of a machinable part, a machiable part could have several interpretations as a collection of features and having only one of them could be a drawback in generative process planning. To address this problem, we have developed an algebra of features. By performing operations in the algebra one can obtain several feature interpretations of a machinable part given one feature interpretation. A feature transformation system based on the algebra has been implemented and integrated with our Protosolid solid modeler [16] and with our EFHA process planning system [15].

NOMENCLATURE

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

This work was supported in part by an NSF Presidential Young Investigator award for Dr. Nau with matching funds from Texas Instruments and General Motors Research Laboratories, NSF Grant NSFD CDR-88003012 to the University of Maryland Systems Research Center, NSF Equipment grant CDA-8811952, and NSF grant IRI-8907890.

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."

Such considerations will require CAD systems of the future to 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. The problem is how to find these alternate interpretations.

For example, consider [1] the part depicted in Figure 3. In this example, the part has been described as the part resulting from subtracting a rectangular pocket p_1, a rectangular pocket p_2 and a hole h_1, in that order, from a rectangular stock. Because of the interaction of the hole h_1 with the pocket p_2, the hole h_1 can be extended into the pocket p_2. Let h_2 be the extended hole. Let the extension of the hole h_2 into the pocket p_1 be h_3. Let the extension of the pocket p_2 into the pocket p_1 be p_3. The various possible feature interpretations of the part are $\{p_1, p_2, h_1\}$, $\{p_1, p_3, h_1\}$, $\{p_1, p_2, h_2\}$, $\{p_1, p_3, h_3\}$, $\{p_1, p_2, h_3\}$ and $\{p_1, p_3, h_2\}$. Of the above feature interpretations $\{p_1, p_3, h_2\}$ is not feasible from geometric considerations. From machining considerations, the larger pocket is made first, and hence the interpretations $\{p_1, p_3, h_1\}$ and $\{p_1, p_3, h_3\}$ are not good choices. Hence, the final set of features used for machining would be the pocket p_1, the pocket p_2 and one of the holes h_1, h_2 and h_3. The choice of h_1 or h_2 or h_3 cannot be be made without considering the parameters of the holes. The holes h_1, h_2 and h_3 have the same diameter but different depths. The holes h_1 and h_2 are located at the bottom of a pocket, and hence are more difficult to position than h_3 which is located on a face of the stock. However, they since their depth is less than that of h_3 they are less expensive to machine than h_3.

To address problems such as the one described above, 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 the part as other collections of machinable features can be derived through operations in the feature algebra. We have implemented a subset of this feature algebra dealing with rectangular solids and cylinders, as the basis of a geometric reasoning system for use in communicating between a solid modeler and a process planning system.

In some previous work [6, 7], we developed a rather limited feature algebra for a small set of features, with the different kinds of features and their interactions all described as special cases. Since then, we have developed a unified mathematical way to describe features and feature interactions, so that our current feature algebra covers practically all features of interest to manufacturing. The current paper presents the highlights of this generalized feature algebra; reader is referred to [10] for a more detailed presentation.

The Algebra of Features and Its Properties

An algebraic structure [11] is a set, with one or more operations defined on the elements of that set. The feature algebra involves the set of all possible features (where *feature* is as defined below), and operations such as truncation and maximal extension (also defined below).

[1] We wish to thank Dr. Guangming Zhang of the Mechanical Engineering Department at the University of Maryland for some insightful discussions on this and other examples.

Domain

A *solid* is a compact, regular and semi-analytic subset of E^3. Let us see the scope and significance of the above definition. Regularity restricts a solid 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 solids that are considered are of finite dimensions, they are bounded and hence compact. The domain of semi-analytic 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 includes concave features such as T-slots, counter-bores and counter-sinks.

A *patch* is a regular, semi-analytic subset of the boundary of a solid. Figure 4 illustrates some examples of patches. Given a solid x and a patch p of x, the regularized complement of p is defined as $c^*(p, x) = b(\text{ps}(x)) -^* p$. From these definitions, we can prove the following lemma.

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

A *feature* x is any pair $x = \langle \text{ps}(x), \text{patches}(x) \rangle$ such that $\text{ps}(x)$ is a solid and $\text{patches}(x)$ is a partition of the boundary of $\text{ps}(x)$ into one or more patches, each of which is labeled as BLOCKED or UNBLOCKED. For any patch p, we denote the label of the patch by $\text{label}(p)$. The intended interpretation of a feature is as a solid $\text{ps}(x)$ that is subtracted from a larger solid, with the blocked and unblocked patches indicating boundaries of the patch that separate metal from air or air from air, respectively. For example, Figure 5 shows the BLOCKED and UNBLOCKED patches for the features h_1, h_2 and h_3 shown in Figure 3.

Operations

Patch Classification Scheme

In order to describe the operations on features, one first needs to understand the methodology for classifying the patches of the boundary of one solid with respect to another solid. This methodology was developed by Vaněček[16], who used it in the context of performing set operations on planar polyhedra.

A patch x_i of the boundary of a solid x is *homogeneous* with respect to solid y if one or more of the classification relationships holds:

1. x_i **IN** y; i.e., the interior of x_i lies in the interior of y.
2. x_i **OUT** y; i.e., the interior of x_i is outside of y.
3. x_i **WITH** y; i.e., x_i lies on the boundary of y, and both x and y are on the same side of the boundary.
4. x_i **ANTI** y; i.e., x_i lies on the boundary of y, and x and y are on the opposite sides of the boundary.

Given the above patch classification scheme, the boundary of a solid x can be partitioned into a set of patches ($\mathcal{P}$), such that each patch in $\mathcal{P}$ is homogeneous with respect to y. Thus, one can obtain collections of patches $x\mathbf{IN}y$, $x\mathbf{OUT}y$, $x\mathbf{WITH}y$ and $x\mathbf{ANTI}y$ given by the following definitions:

$$\begin{aligned} x\mathbf{IN}y &= \{p \in \mathcal{P} | p\mathbf{IN}y\} \\ x\mathbf{OUT}y &= \{p \in \mathcal{P} | p\mathbf{OUT}y\} \\ x\mathbf{WITH}y &= \{p \in \mathcal{P} | p\mathbf{WITH}y\} \\ x\mathbf{ANTI}y &= \{p \in \mathcal{P} | p\mathbf{ANTI}y\}. \end{aligned}$$

The operations in the feature algebra are defined in terms of set operations on solids. Using the above classification sets, the boundaries of $x \cup^* y$, $x \cap^* y$, $x -^* y$ and $y -^* x$ can be computed as given below. For any patch p, p^{-1} is the same as p with the sign of the normal to the patch at every point reversed.

$$\begin{aligned} b(x \cup^* y) &= x\mathbf{OUT}y \cup y\mathbf{OUT}x \cup x\mathbf{WITH}y \\ b(x \cap^* y) &= x\mathbf{IN}y \cup y\mathbf{IN}x \cup y\mathbf{WITH}x \\ b(x -^* y) &= x\mathbf{OUT}y \cup (y\mathbf{IN}x)^{-1} \cup x\mathbf{ANTI}y \\ b(y -^* x) &= y\mathbf{OUT}x \cup (x\mathbf{IN}y)^{-1} \cup y\mathbf{ANTI}x. \end{aligned}$$

Truncation

Given two features x and y, the truncation operation (T) is defined as follows: $z = x\mathcal{T}y = \langle u, v \rangle$, where $u = \text{ps}(x) -^* \text{ps}(y)$ and v is a collection of patches satisfying the following properties:

1. the union of all the patches in v is $b(u)$.
2. every patch in v that belongs to $\text{ps}(x)\mathbf{OUT}\text{ps}(y)$ or $\text{ps}(x)\mathbf{ANTI}\text{ps}(y)$ is a subset of some patch of x.

Note that,

$$b(\text{ps}(x) -^* \text{ps}(y)) = \text{ps}(x)\mathbf{OUT}\text{ps}(y) \cup (\text{ps}(y)\mathbf{IN}\text{ps}(x))^{-1} \cup \text{ps}(x)\mathbf{ANTI}\text{ps}(y)$$

The labels of the patches of v are determined as follows:
For every patch p of v, if $p \in (\text{ps}(x)\mathbf{OUT}\text{ps}(y))$ or $p \in (\text{ps}(x)\mathbf{ANTI}\text{ps}(y))$, let p_1 be the patch of x such that $p \subseteq p_1$.

$$\text{label}(p) = \begin{cases} \text{label}(p_1) & \text{if } p \in \text{ps}(x)\mathbf{OUT}\text{ps}(y) \text{ or} \\ & p \in \text{ps}(x)\mathbf{ANTI}\text{ps}(y) \\ \text{UNBLOCKED} & \text{otherwise} \end{cases} \quad (1)$$

Maximal Extension

In order to define the maximal extension operation, we need to first define the infinite extension of a feature with respect to a patch. To keep the discussion simple, we present here the definiton of infinite extension only for convex solids. For a more general definition, the reader is referred to [10]. Figure 6 illustrates some examples of infinite extension.

Let x be any feature and p_i be any point on x for which there exists a plane tangent to x at p_i. Then $H(p_i, x)$ is the closed half-space tangent to x at p_i that contains x.

The infinite extension of x with respect to the patch p is

$$\mathcal{I}_p(x) = \bigcap \{H(p_i, x) | p_i \in c^*(p, x).$$

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:

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

where $u = \mathcal{I}_p(x) \cap^* (\text{ps}(x) \cup^* \text{ps}(y))$ and v is a collection of patches satisfying the following properties:

1. the union of all patches in v is $b(u)$.
2. every patch in v that belongs to $(\text{ps}(x) \cup^* \text{ps}(y))\mathbf{IN}\mathcal{I}_p(x)$ or $(\text{ps}(x) \cup^* \text{ps}(y))\mathbf{WITH}\mathcal{I}_p(x)$ is a subset of some patch of $\text{ps}(x) \cup^* \text{ps}(y)$.

Note that

$$b(u) = \mathcal{I}_p(x)\mathbf{IN}(\text{ps}(x) \cup^* \text{ps}(y)) \cup (\text{ps}(x) \cup^* \text{ps}(y))\mathbf{IN}\mathcal{I}_p(x) \cup (\text{ps}(x) \cup^* \text{ps}(y))\mathbf{WITH}\mathcal{I}_p(x)$$

The labels of the patches of v are determined as follows: For every patch p of v, if $p \in (\text{ps}(x) \cup^* \text{ps}(y))\mathbf{IN}\mathcal{I}_p(x)$ or $p \in (\text{ps}(x) \cup^* \text{ps}(y))\mathbf{WITH}\mathcal{I}_p(x)$ let p_1 be the patch of $\text{ps}(x) \cup^* \text{ps}(y)$ such that $p \subseteq p_1$.

$$\text{label}(p) = \begin{cases} \text{label}(p_1) & \text{if } p \in (\text{ps}(x) \cup^* \text{ps}(y))\mathbf{IN}\mathcal{I}_p(x) \text{ or} \\ & p \in (\text{ps}(x) \cup^* \text{ps}(y))\mathbf{WITH}\mathcal{I}_p(x) \\ \text{UNBLOCKED} & \text{otherwise} \end{cases}$$

For the definition to be complete, the labels of the patches of $\text{ps}(x) \cup^* \text{ps}(y)$ must be defined, given the labels of the patches of x and y. As stated earlier,

$$b(\mathrm{ps}(x) \cup^* \mathrm{ps}(y)) \;=\; \mathrm{ps}(x)\mathbf{OUT}\mathrm{ps}(y) \cup \mathrm{ps}(y)\mathbf{OUT}\mathrm{ps}(x) \cup \mathrm{ps}(x)\mathbf{WITH}\mathrm{ps}(y)$$

Let us denote a arbitrary patch of $b(\mathrm{ps}(x) \cup^* \mathrm{ps}(y))$ by p. If p is a patch of $\mathrm{ps}(x)\mathbf{OUT}\mathrm{ps}(y)$ or $\mathrm{ps}(x)\mathbf{WITH}\mathrm{ps}(y)$, let p_1 be a patch of x such that $p \subseteq p_1$. If p is a patch of $\mathrm{ps}(y)\mathbf{OUT}\mathrm{ps}(x)$ or $\mathrm{ps}(x)\mathbf{WITH}\mathrm{ps}(y)$, let p_2 be the patch of y such that $p \subseteq p_2$.

Let an arbitrary patch of $b(\mathrm{ps}(x) \cup^* \mathrm{ps}(y))$ be denoted by p and the patch of x (if applicable) that is a superset of p by p_1 and the patch of y (if applicable) that is a superset of p by p_2.

The labels of the patches of $b(\mathrm{ps}(x) \cup^* \mathrm{ps}(y))$ are defined below:

$$\mathrm{label}(p) = \begin{cases} \mathrm{label}(p_1) & \text{if } p \in \mathrm{ps}(x)\mathbf{OUT}\mathrm{ps}(y) \\ \mathrm{label}(p_2) & \text{if } p \in \mathrm{ps}(y)\mathbf{OUT}\mathrm{ps}(x) \\ \mathrm{BLOCKED} & \text{if } p \in \mathrm{ps}(x)\mathbf{WITH}\mathrm{ps}(y) \text{ and} \\ & \mathrm{label}(p_1) = \mathrm{label}(p_2) = \mathrm{BLOCKED} \\ \mathrm{UNBLOCKED} & \text{otherwise} \end{cases}$$

Properties

From the above definitions, a number of useful properties can be proved. Below are a few examples.

Proposition 2 *Given a feature x and a patch p of x,*

$$ps(x) \subseteq \mathcal{I}_p(x).$$

Proposition 3 *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 4 *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.*

Using the Algebra

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 (see Figure 1), $\bar{F}$ 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 and Σ_I the set of possible infinite extension operations.

```
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 1: The Features Algorithm

In the algorithm shown in Figure 1 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 (such as the Propositions 2, 3 and 4) 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 for the operations 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.

Implementation

The features algorithm has been implemented on a restricted domain consisting of rectangular solids and cylinders that have their planar faces parallel to the faces of the stock, such as holes, slots, shoulders, pockets, etc. Input to the algorithm is taken from a feature-based design system built on top of our Protosolid[16] solid modeler.

The implementation was done on a Texas Instruments Explorer II. The operations in the algebra are implemented as procedures in Lisp, and the algebraic properties (such as Propositions 2, 3, and 4) as rules in Prolog. Whenever necessary, the features algorithm either asserts facts into the Prolog data base or queries it to determine the result of an operation on two features. If the query cannot be answered, then the procedures for the operations are invoked directly. Once the features algorithm has terminated, we have several possible feature interpretations of the part.

After these features have been produced, EFHA [15] (a successor to our SIPS process planning system[9]) can be invoked to make process plans for the features. These results give us an indication of which interpretation of the part is likely to be the easiest to manufacture.

Example

Let us illustrate the workings of the algorithm with the example discussed in Figure 3. In the beginning, we have exactly one feature interpretation of the part, viz. $\{p_1, p_2, h_1\}$. Due to the interaction between the features, in the first iteration we compute two additional features $h_2 = h_1\mathcal{M}_a p_2$ and $p_3 = p_2\mathcal{M}_b p_1$, where a and b are the top faces of h_1 and p_2 respectively. The additional feature interpretations added in this iteration are $\{p_1, p_2, h_2\}$ and $\{p_1, p_3, h_1\}$. During the second iteration we compute one more feature $h_3 = h_2\mathcal{M}_c p_1$, where c is the top face of h_2. The additional feature interpretations added are $\{p_1, p_2, h_3\}$ and $\{p_1, p_3, h_3\}$. During the third iteration no new features are generated and hence the algorithm terminates after the third iteration. However, a new feature interpretation $\{p_1, p_3, h_2\}$ is added due to the interaction between the features h_1 and p_1. As discussed earlier, this combination is not feasible from geometric considerations.

For the example shown in Figure 3, the total processing time for computing the alternative feature interpretations was 6.35 seconds. In this example, the total number of interactions (the number of times an operator was applied) was 118, but only three of the interactions resulted in new features. For this example, 9 of these interactions were resolved via queries to the Prolog implementation of the algebraic properties, without invoking the procedures to compute the operations.

When the same example was run without access to the algebraic properties, the processing time decreased to 2.6 seconds. One reason for this result could be that the Lisp code is compiled and the Prolog runs interpretively. The other reason could be that since the features and interactions considered right now are rather simple, the procedures for the operations do not take a significant amount of time. With more complex features and interactions, access to the algebraic properties may turn out to be more useful.

The procedures for computing the operations take advantage of the nature of the features and interactions. Declaratively, the operations for this sub-algebra can be expressed in terms of set operations on solids. Therefore, another way to compute the operations would be to convert them into set operations on solids (computed by the solid modeler) and then test if the result of the operation is a new feature. Using this method, the time taken for computing alternate feature interpretations for the example in Figure 3 is 138 seconds. This shows that using the algebra is much more efficient than translating the operations into equivalent set operations. The reason is that if the result of an operation is not a valid feature there is no reason to compute the shape of the solid and then realize it is not a valid feature. Currently, we are working on a mathematical analysis of the computational time taken by both methods.

In all of the cases above, the parameters of the features were communicated to the EFHA process planning system which was used to generate the process plans for the individual features. The process plans and the (relative) costs of the cheapest process plans for each feature are shown in Table 2.

Related Work

The popular approaches to CAD/CAM integration are auto-

Feature	Process Plan	Cost
p_1	Rough Face Mill	1.0
p_2	Rough Face Mill	1.0
p_3	Rough Face Mill	1.0
h_1	Twist Drill → Rough Bore → Finish Bore	9.0
h_2	Twist Drill → Rough Ream → Finish Ream	11.0
h_3	Gun Drill → Rough Ream → Finish Ream	25.0

Figure 2: The Process Plans for the features in Figure 3

matic 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[14] and Joshi and Chang [5] accounts for certain kinds of feature interactions. Elaborate systems ([8, 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.

Nicholas Ide [4] has developed a system for feature based design using the PADL–2 solid modeler [2]. Ide's system bridges the PADL–2 solid modeler with a process planner developed at the University of Maryland called SIPS [9]. In addition to providing a design-features-interface this system also does two types of checking, for local constraints (consistency in feature parameters) and for geometric constraints (constraints requiring geometric reasoning). A majority of the feature interactions are either not allowed or not checked for by this system (eg., a hole cannot intersect any other feature). Hayes [3] 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 [13] have proposed the notion of engineering environments (analogous to programming environments) which are tools useful for design and manufacturing engineers. They 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[12] 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 truncation 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.

Summary and Conclusions

Motivation

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. One of the primary motivations of the feature algebra is to allow consideration of geometric objects as several possible alternative collections of features. Based on our discussions with machinists, it appears that a machinable part cannot be interpreted as a unique set of features. What seems more appropriate is to consider alternative interpretations, and generate plans to see which is better (or feasible).

The kinds of reasoning done in the Machinist [3] system represent significant steps in the development of ways to handle feature interactions. However, this system does not use an unambiguous representation of solid objects. For example, if the Machinist program decides that some hole h needs to be made before some slot s, it does not recognize that this requires machining a hole of different dimensions than if h were machined after s—and yet such information may be necessary in order to know whether it is possible to machine h. In the feature algebra described in this paper, a feature includes a complete representation of a physical solid. Thus, it might be possible to add additional sophistication to the operation of the Machinist program by rewriting some of its rules in terms of operations in the feature algebra.

Properties

As defined in this paper, the algebra is general enough to encompass practically all shapes of features encountered in the real world. It is not computationally feasible to implement the operations in the algebra on this entire set of features. But by restricting the algebra in different ways, one can obtain different sub-algebras which satisfy the properties of the general feature algebra, allow the algebraic operations to be computed efficiently, and include features of interest in practical problems. In this way, the operations in the algebra can be made more efficient than set operations on solids, because they take advantage of the special properties of the shapes and their interactions.

Concluding Remarks

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.

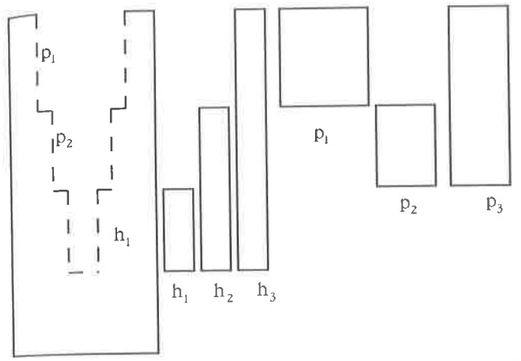

Figure 3. A Sample Part with two pockets and a hole

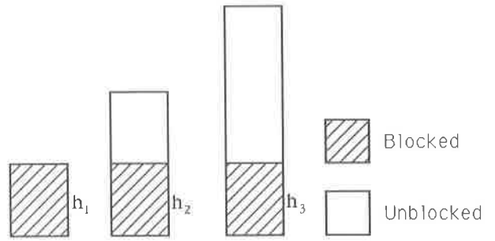

Figure 5. BLOCKED and UNBLOCKED patches.

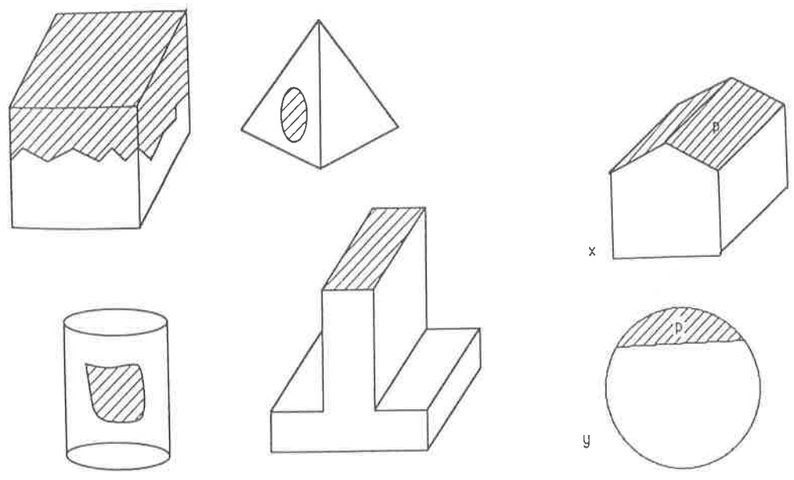

Figure 4. Some examples of patches (shaded).

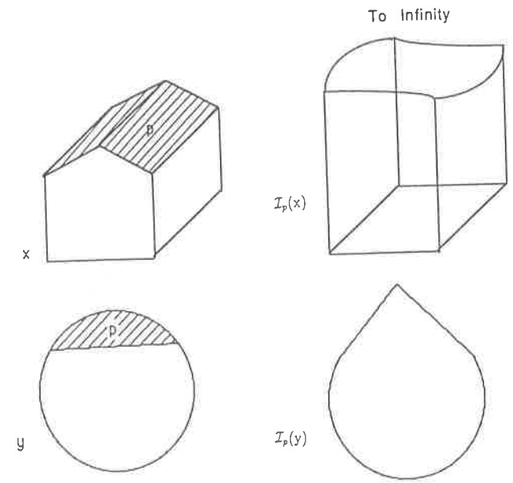

Figure 6. Some examples of patches and infinite extension.

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] E. E. Hartquist and A. Marisa. *PADL–2 Users Manual.* Production Automation Project, University of Rochester, Rochester, New York, 1985.

[3] 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.

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

[5] 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.

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

[7] 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.

[8] 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.

[9] 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.

[10] D. S. Nau and R. R. Karinthi. An algebraic approach to feature interactions. Technical Report TR-89-101, Systems Research Center, University of Maryland, College Park, nov 1989. Submitted for journal publication.

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

[12] 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.

[13] 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.

[14] 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.

[15] Scott Thompson. Environment for hierarchical abstraction: A user guide, May 1989. Master's Scholarly paper.

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

SPATIAL REASONING FOR AUTOMATIC RECOGNITION OF INTERACTING FORM FEATURES

Jan H. Vandenbrande and Aristides A. G. Requicha
Programmable Automation Laboratory
Computer Science Department and
Institute for Robotics and Intelligent Systems
University of Southern California
Los Angeles, California

ABSTRACT

Recognition of machining features such as holes, slots and pockets is essential for the fully automatic manufacture of mechanical parts. This paper discusses an experimental feature recognizer that uses a blend of artificial intelligence (AI) and computational geometry techniques. The recognizer is implemented in a rapid prototyping test bed consisting of the KnowledgeCraft™ AI environment tightly coupled with the PADL-2 solid modeler. It is capable of finding features with interacting volumes (e.g., two crossing slots), and takes into account nominal shape information as well as tolerancing and other available data. A generate-and-test strategy is used. Production rules generate hints or clues for the existence of features, and post them on a blackboard. The clues are assessed, and those judged promising are tested through geometric computations. The process continues until it produces a complete decomposition of the volume to be machined in terms of volumetric features that correspond to material removal operations.

INTRODUCTION

This paper discusses on-going research on the automatic recognition of *machinable* form features in solid models of mechanical parts. Machinable *volumetric features* (or simply "features") in our work are solids removable by the operations typically performed in 3-axis machining centers [Requicha & Vandenbrande 1989]. Our ultimate goal is to build the system shown in Figure 1. A geometric model, created either by a design-by-features system or by other methods, is analyzed by a feature finder, which produces a decomposition of the part into volumetric machining features. These are passed to a process planner which selects processes and tools for machining each feature, and generates setup and fixturing information. The operations prescribed in a process plan are expanded by an operation planner, whose output is a sequence of instructions executable by Numerically Controlled (NC) machine tools and robots. (Some authors argue that feature recognition is not necessary in design-by-features systems, but we and others think otherwise, because design (or functional) features often are

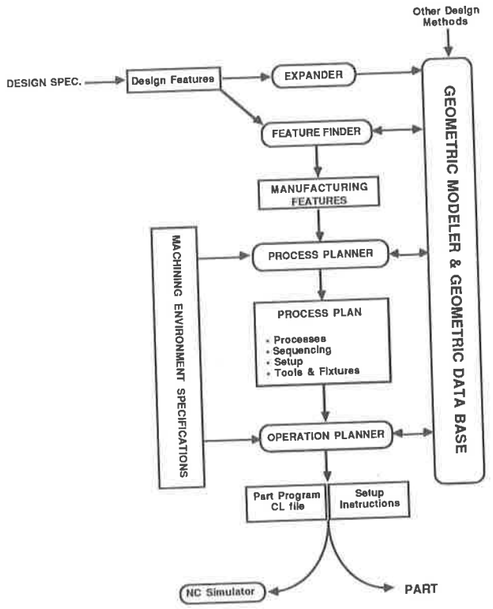

Fig. 1 Integrated Design and Manufacturing System

distinct from manufacturing features [Requicha & Vandenbrande 1989].) Reasoning about spatial data is crucial for many of the modules shown in Figure 1, and constitutes one of the major difficulties in the development of intelligent systems for mechanical design and manufacturing. Feature finding is a research area in which progress has been relatively slow, and also is an area intimately associated with spatial reasoning.

Most of the previous work on feature recognition is based on searching Boundary Representations (BReps) for certain