An Algebraic Approach to Feature Interactions

Raghu R. Karinthi and Dana Nau

Abstract—**Various approaches have been proposed to provide communication between CAD systems and process planning systems, including automated feature extraction, design by features, and human-supervised feature extraction. Regardless of which approach is used, a major problem is that due to geometric interactions among features, there may be several equally valid sets of manufacturable features describing the same part, and different sets of features may differ in their manufacturability. Thus, to produce a good process plan—or, in some cases, even to produce a process plan at all—it may be necessary to interpret the part as a different set of features than the one initially obtained from the CAD model. This paper proposes a way to address this problem, based on an algebra of features. Given a set of features describing a machinable part, other equally valid interpretations of the part can be produced by performing operations in the algebra. This will enable automated process planning systems (such as [39]) to examine these interpretations in order to see which one is most appropriate for use in manufacturing. The feature algebra has been implemented for a restricted domain and integrated with the Protosolid [42] solid modeling system and the EFHA process planning system [39].**

Index Terms—**Algebraic structures, automated manufacturing, concurrent engineering design, feature extraction, geometric reasoning, solid modeling.**

I. 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 noncompetitive 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 (or herself) to the traditional role of just developing products to meet specified functional requirements but should also actively consider the associated manufacturing implications. For example, consider the task of designing metal parts that will be produced using machining operations. 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. In addition, 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.

A. The CAD/CAM Integration Problem

The goals of concurrent engineering will require that CAD systems of the future interact extensively with modules such as process planning systems to evaluate the manufacturability of the design. It will be necessary for process planning systems to reason about geometric relationships among the various parts of an object while the design is underway. However, the achievement of such interactions presents several unresolved problems. One of the primary problems is that generative process planning systems (such as XCUT [2] or SIPS [24]) consider a machinable part to be a collection of machinable features, and it is necessary to obtain descriptions of such features from the CAD representation. Several approaches on how to do this have been proposed, as is discussed in the following:

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. Prominent among these are the ones by Kyprianou [20], Henderson [10], De Floriani [4], Kumar [18], Srinivasan [37], and Vandenbrande [41]. 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 that intersects or otherwise interacts with other features without disturbing those other features. For example, in

Manuscript received December 18, 1989; revised August 15, 1991. This work was supported 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-85-00108 to the University of Maryland Systems Research Center, NSF Equipment grant CDA-8 811952, NSF grant IRI-8907890, and by the Defense Advanced Research Projects Agency (DARPA) under contract MDA972-88-C-0047 for the DARPA Intiative in Concurrent Engineering (DICE). Recommended for acceptance by Associate Editor J. Mundy.

R. R. Karinthi is with the Department of Statistics and Computer Science and Concurrent Engineering Research Center, West Virginia University, Morgantown, WV 26506.

D. Nau is with the Computer Science Department, Systems Research Center and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742.

IEEE Log Number 9105005.

Henderson's feature extraction system [10], 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. In *design by features*, the user builds a solid model of an object by specifying directly various form features that translate directly into the relevant manufacturing features. Systems for this purpose have been built for designing injection-molded parts [40], aluminum castings, [21], and machined parts [17], [11].

2) 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 part for functionality, and a process engineer determines what the manufacturable features are. However, the design-by-features approach places the designer under the constraints of not merely having to design for functionality but also having to specify all of the manufacturable features as part of the geometry—a task the designer is not normally qualified to do. Another problem—that of alternate feature interpretations—is described in Section I-B.
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 and then requiring the process engineer to identify the machinable features of the part. Systems have been built for this purpose at General Motors Research Laboratories and at the National Bureau of Standards [3]. 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 interpretations.

B. Geometric Interactions Among Features

Regardless of what technique is used for obtaining machinable features from a CAD representation, geometric interactions among the features can create situations where there are several possible feature representations for 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. The importance of handling feature interactions has been stressed in the recent reports such as [35] and [33]. As an example, let us consider the part shown in Fig. 1. In this example, the part has been described as the part resulting from subtracting a hole h_1 and a slot s_1 and a slot s_2, in that order, out of a rectangular stock, but because of the interaction of h_1 with s_1 and s_2, we get $h_2 = h_1 -^* s_1$,[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 the holes h_1, h_2, h_3 and h_4. Which hole to use depends on issues such as cost

[1]The terminology of $-^*$ is explained later in the paper. For the present, the reader may treat this as the usual set subtraction.

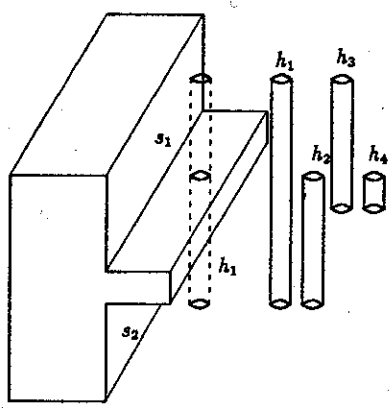

Fig. 1. Part with two slots s_1 and s_2 and a hole h_1.

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 (whether it is possible to fixture the part to prevent excessive vibration while making h_4 after s_1 and s_2 have been made), and machinability criteria (whether vibration during the machining of h_4 will allow acceptable machining tolerances to be achieved or whether h_1 can be machined with an acceptable straightness tolerance). Thus, one can see that there can be several possible interpretations such as $\{h_1, s_1, s_2\}$, $\{h_2, s_1, s_2\}$, $\{h_3, s_1, s_2\}$, or $\{h_4, s_1, s_2\}$ for the same part, and having only one of them can be a serious limitation in process planning. This paper discusses an approach for dealing with this problem based on an algebra of features. Alternative interpretations of features resulting from feature interactions are provided by means of operations in this algebra. Thus, this scheme provides several alternative feature representations of a machinable part, given one such representation.[2] We intend to use this algebra as the basis of a feature transformation system, which will be capable of examining feature descriptions, computing alternate feature representations for an object, and presenting them to a process planning system. A prototype version of the feature algebra has already been implemented. The reader is referred to Karinthi [16] for details of this implementation. This prototype serves as the communication interface between our Protosolid solid modeling system [42] and our EFHA process planning system [39] (EFHA is an extension of our previous process planning system called SIPS [24]) in the development of an integrated system for design and process planning. The rest of the paper is organized as follows: Section II introduces the mathematical concepts needed to understand the description of the feature algebra. Section III describes the feature algebra. Section IV describes the feature algebra in detail for a restricted set of features. Section V compares our work with related research in this area. Section VI summarizes this research and presents our conclusions and plans for future work.

II. Mathematical Preliminaries

This section summarizes some of the mathematical concepts required for the definition of a feature in the next section. For a

[2]The basic idea of an algebra of features was proposed in [14] and [15]. Since then, the scope of the feature algebra has increased significantly.

more detailed treatment of this material, the reader is referred to Requicha and Tilove [31], [30] and to Kuratowski [19], Mendelson [23], Simmons [36], and Agoston [1].

In the summary that follows, the symbol W denotes the *universal set* and $\varnothing$ the empty set. The symbols $\cup$, $\cap$, $-$, c, $\subseteq$, $\supseteq$, $\subset$, and $\supset$ are used to denote the operations, *union, intersection, subtraction, complement, subset, superset, proper subset,* and *proper superset* on sets.

A. Metric Spaces and Topological Spaces

If W is a nonempty set, a collection T of subsets of W is called a *topology* on W if it meets the following requirements:

1) The union of a finite number of sets of T belongs to T.
2) The intersection of a finite number of sets of T belongs to T.
3) $W \in T$.

An ordered pair (W, T) in which the first component W is a nonempty set and the second component T is a topology on W is called a *topological space.* A subset of W is said to be *open* if and only if it belongs to T.

A *metric space* is an ordered pair (W, f), where W is a set, $\Re$ is the set of all reals, and $f : W \times W \rightarrow \Re$ is a function called the *distance* or *metric* such that for all a, b, and c in W, the following applies:

1) $f(a, b) \geq 0$;
2) $f(a, b) = 0$ if and only if $a = b$;
3) $f(a, b) = f(b, a)$;
4) $f(a, c) \leq f(a, b) + f(b, c)$ *(the triangle inequality).*

As a simple example, the ordered pair (E^3, d), where d is the function giving the Euclidean distance between two points is a metric space. Let (W, f) be a metric space, and let x be a point of W. The *open ball* of radius $R > 0$ about x, denoted by $Ball(x, R)$, is the set of all points y in W that satisfy $f(x, y) < R$.

Given a metric space (W, f), a set $X \subseteq W$ is *open* if it contains an open ball about each of its points. For example, given the metric space $M = (\Re, abs)$, where $abs(x, y) = | x - y |$, the set $(0, 10)$ of all reals greater than 0 and less than 10 is an open set. Given a metric space (W, f), a set $X \subseteq W$ is *bounded* if it is a subset of a ball of finite radius.

Given a metric space (W, f), the set T of all (metric) open subsets of W form a topology on W. Thus, if (W, f) is a metric space, (W, T) is a topological space. Thus, the open sets of the metric space (W, f) and the topological space (W, T) are identical.

B. Closed Sets

A *neighborhood* of a point x in a topological space (W, T) is any subset of W that contains an open set containing x.

Given a topological space (W, T) and a set $X \subseteq W$, a point x is a *limit point* of the set X if each neighborhood of x contains at least one point of X different from x. Notice that the limit points of a set need not belong to the set. Given the metric space $M = (\Re, abs)$ seen earlier, there is a corresponding topological space $N = (\Re, T)$, where T is the set of all (metric) open subsets of $\Re$. Now, consider the open subset $(0, 10)$. 0 and 10 are the limit points of the set $(0, 10)$, and neither of them belongs to the set. The *closure* of a subset X, denoted by kX, is the union of X with the set of all its limit points. A set X is *closed* if and only if $X = kX$. For example, the set $[0, 10]$ of all reals greater than or equal to 0 and less than or equal to 10 is a closed set. Notice that some sets (the set $(0, 10]$, for example) are neither open nor closed.

C. Interior and Boundary

Given a topological space (W, T), a point x of W is an *interior point* of a set $X \subseteq W$ if X is a neighborhood of x, i.e., if X contains an open set that contains x.

Given a topological space (W, T), the *interior* of a set $X \subseteq W$, denoted by iX, is the set of all interior points of X.

$X = iX$ if and only if X is open. Given a topological space (W, T), a point x of W is a *boundary point* of a set $X \subseteq W$ if each neighborhood of x intersects both X and cX.

The *boundary* of X, denoted by $b(X)$, is the set of all boundary points of X.

The reader may observe that the definitions of interior and boundary correspond to the intuitive notions of interior and boundary for solid objects.

D. Compactness

It can easily be seen that the ordered pair (E^n, f), where E^n is the Euclidean n space and f is the *Euclidean distance,* is a metric space and that there is a corresponding topological space (E^n, T), where T is the set of all open sets of E^n. Henceforth, we talk of the properties of a subset of E^n, where the corresponding metric and topological spaces are the ones just mentioned. A subset of Euclidean n space is *compact* if and only if it is closed and bounded [23].

E. Regular Sets

The *regularization* of a subset X of W, denoted rX, is the set $rX = kiX$. A set X is *closed regular* if $X = rX$, i.e., if $X = kiX$. Note that $rrX = rX$. From now on, we simply refer to a closed regular set as a regular set. In intuitive terms, a regular set in E^3 cannot have any dangling faces, dangling edges, or isolated points. It is well known that when set-theoretic operations such as union, intersection, and subtraction are applied to two valid n-dimensional objects, the result is not necessarily a valid n-dimensional object. For example, if two squares touch on one side, their intersection is a single line segment, which is not a valid object. Requicha and Voelcker [32] have shown that this difficulty can be overcome by using regularized set operations instead of ordinary set operations. The symbols $\cup^*$, $\cap^*$, $-^*$, and c^* are used to denote regularized union, intersection, subtraction, and complementation, respectively. They are defined below:

$$X \cup^* Y = r(X \cup Y)$$
$$X \cap^* Y = r(X \cap Y)$$
$$X -^* Y = r(X - Y)$$
$$c^* X = rcX$$

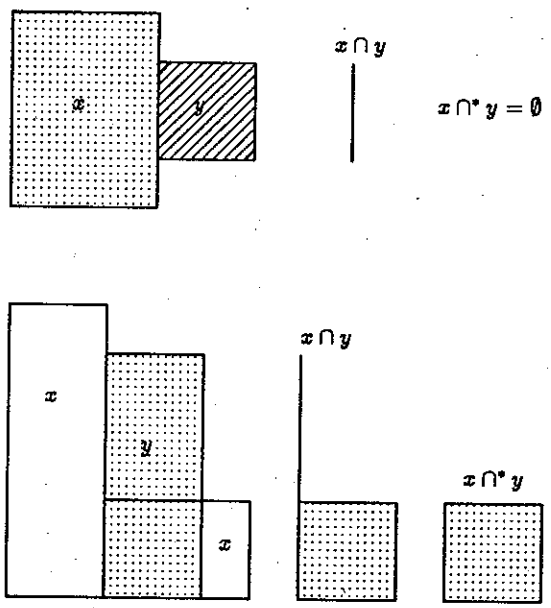

Fig. 2. Ordinary and regularized set operations.

Fig. 2 shows the difference between ordinary and regularized set operations.

F. Semianalytic Sets

A function $f : E^3 \rightarrow \Re$ is said to be *analytic* [6] throughout its domain if it can be expanded in a power series in x, y, and z about every point in its domain. A necessary condition for a (real) function to be analytic is that it be infinitely differentiable. A subset of E^3 is said to be *semianalytic* if it is a finite combination, via the set operations union, intersection, and complement, of sets X_i of the form

$$X_i = \{p \in E^3 : f_i(p) \geq 0\}$$

where f_i is any analytic function on E^3.

It can be shown that the interior, boundary, and closure of a semianalytic set is also semianalytic and that class of compact, regular, semianalytic sets is closed under regularized set operations.

G. Convexity and Concavity

Given two distinct points p_1 and p_2 in Euclidean n space, the *convex combination* of p_1 and p_2 is the set

$$\{p : p = \alpha p_1 + (1 - \alpha)p_2, \alpha \in \Re, 0 \leq \alpha \leq 1\}.$$

The convex combination normally describes the *straight line segment* $\overline{p_1 p_2}$ (unordered pair). A subset x of E^3 is said to be *convex* if and only if for any two points p_1 and p_2 belonging to x, the segment $\overline{p_1 p_2}$ is entirely contained in x. In this paper, a subset x of E^3 that is not convex is said to be *concave*.

It can be shown that the intersection of two convex sets is a convex set [29]. It can easily be seen that the union and difference of two convex sets is not necessarily a convex set.

III. The Algebra of Features

An algebraic structure [27] 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 that assigns to each ordered pair $\langle x, y\rangle$ of elements of A exactly one element $x * y$ in A. There are three aspects of the definition that need to be stressed:

1) $x * y$ is defined for every ordered pair $\langle x, y\rangle$ of elements of A;
2) $x * y$ must be uniquely defined;
3) A is closed under the operation $*$.

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 $*$

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

A. Feature Definition

A feature (other than INVALID) is given by a pair $x = \langle \text{ps}, \text{patches}\rangle$ where the two entries in the pair satisfy the following conditions: The first entry $(\text{ps}(x))$ is the set of all points in a feature and is a subset of E^3 that is compact, regular, and semianalytic.

1) The second entry $(\text{patches}(x))$ is a partition of the boundary of $\text{ps}(x)$ into labeled patches (as defined below).
2) Henceforth, the boundary of $\text{ps}(x)$ and the patches in $\text{patches}(x)$ will be referred to as the boundary and patches of x.

A *patch* is a regular, semianalytic subset of the boundary of a feature. A labeled patch is a patch p with a label $\text{label}(p)$ whose value is either BLOCKED or UNBLOCKED. The patches and their labels are intended to describe what the boundaries of the feature mean in the real world. In particular, suppose x is a feature of some manufacturable object X. If some patch p of x separates metal from air (i.e., p lies on the boundary of X), then p is BLOCKED, and if p separates air from air (i.e., p does not lie on the boundary of X), then p is UNBLOCKED. Fig. 3 shows the BLOCKED and UNBLOCKED patches for the features h_1, s_1, and s_2 shown in Fig. 1. Fig. 4 illustrates some examples of patches.

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

It is worthwhile now to examine the scope and significance of the above definitions. Regularity restricts a feature to be homogeneously 3-D. In other words, a feature may not have dangling faces, edges, and vertices. Fig. 5 shows examples of valid and invalid features. Even parts with sheet metal components have a finite thickness; therefore, 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 semianalytic sets covers practically all the shapes of interest to manufacturing. The reader may note that all planar polyhedra, cylinders, cones, spheres, tori,

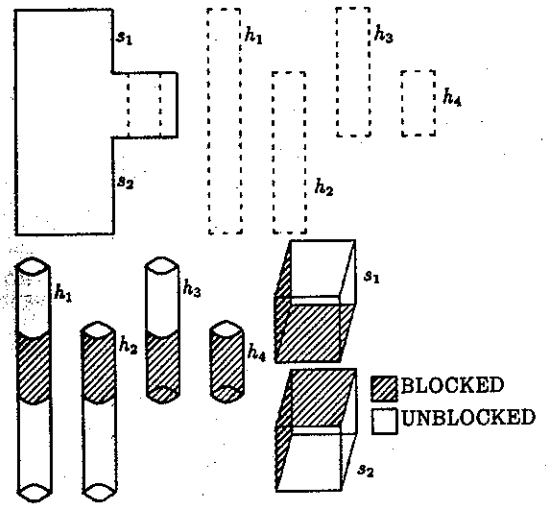

Fig. 3. Patches and patch labels for the features shown in Fig. 1.

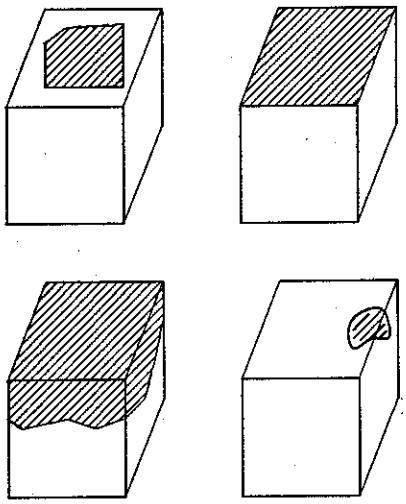

Fig. 4. Some examples of patches.

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 Fig. 6).

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

Proof: The proof is a direct consequence of the closure properties of compact, regular, and semianalytic sets stated earlier.

1) Neighborhood: The notion of a neighborhood of a point on the boundary of a feature will be useful later on. If $\delta > 0$, the δ neighborhood $N(p, \delta)$ of a point p on the boundary of a feature x is the set of all points on the boundary of x that are at a distance $\leq \delta$ from p.

2) Orthogonal Projection: In this section, a function known as *orthogonal projection* will be defined. It will be used later in describing the operations in the algebra. Given a planar patch s and a point p not in the plane of s, the orthogonal projection of p in s, denoted $\mathcal{O}(p, s)$, is the point p' (if it exists) on s such that the line through p and p' is perpendicular to the plane of s.

Given two planar patches s_1 and s_2, the orthogonal projection of s_1 in s_2 (if it exists) is denoted $\mathcal{O}(s_1, s_2) = \{\mathcal{O}(p, s_2) \mid p \in s_1\}$ if $\mathcal{O}(p, s_2)$ is defined for all $p \in s_1$.

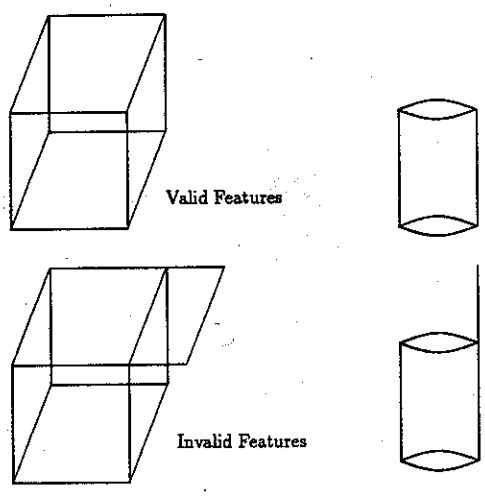

Fig. 5. Valid and invalid features.

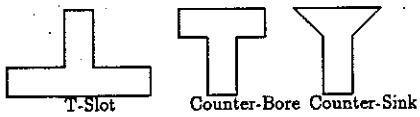

Fig. 6. Cross-sectional views of three concave features.

B. Operations on Features

This section describes the operations on features. In order to describe them, one needs to first understand the methodology for classifying the patches of the boundary of one solid with respect to another solid. This methodology has been developed by Vaněček [42], who used it in the context of performing set operations on planar polyhedra.

1) Patch Classification Scheme: 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.

Fig. 7 illustrates the patch classification scheme described above. For example, the patch (also face) v_1 of the solid v is not homogeneous with respect to w. However, it can be partitioned into two patches, each of which is homogeneous with respect to w.

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**IN**y, x**OUT**y, x**WITH**y, and x**ANTI**y given by the following definitions:

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

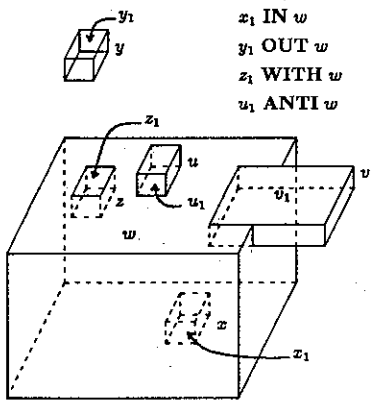

Fig. 7. Classification of patches of solids x, y, z, u, and v with respect to the solid w.

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.

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

2) Truncation: Given two features x and y, the truncation operation ($\mathcal{T}$) 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 $p \subseteq p_1$.

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

The rationale for the above labeling is as follows: If a patch p of $x\mathcal{T}y$ belongs to $\text{ps}(x)\mathbf{OUT}\text{ps}(y) \cup \text{ps}(x)\mathbf{ANTI}\text{ps}(y)$, then $p \subset b(\text{ps}(x))$, and hence, the patch p must have the same label as a patch p_1 of x that contains p. If there is no single patch p_1 that contains p, then we can either split p or combine some patches of x to satisfy this property. If a patch p of $x\mathcal{T}y$ belongs to $(\text{ps}(y)\mathbf{IN}\text{ps}(x))^{-1}$, then this patch is **IN** with respect to $\text{ps}(x)$, and hence, the interior of the patch p cannot belong to the boundary of the final part. Therefore, p should be labeled UNBLOCKED.

3) 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.*

Let us consider a patch p on a feature x.

At any point $p_i = \langle x_i, y_i, z_i\rangle$ on the boundary of p (b(p)), consider the neighborhood $N(p_i, \delta_i)$ for some arbitrarily small $\delta_i > 0$. Suppose there exists an analytic function f such that for every point $p' \in N(p_i, \delta_i) - p$, $f(p') = 0$ and the following limits exist:

$$\lim_{\delta_i \to 0} f_x(p') = f_{ix}$$
$$\lim_{\delta_i \to 0} f_y(p') = f_{iy}$$
$$\lim_{\delta_i \to 0} f_z(p') = f_{iz}$$

where $f_x(p')$, $f_y(p')$, and $f_z(p')$ are the partial derivatives of f with respect to X, Y, and Z.

Now, we define what is called the *partial-tangent-plane* $\bar{\mathcal{T}}_p(p_i, x) = 0$ as follows:

$$\bar{\mathcal{T}}_p(p_i, x) = \begin{cases} f_{ix}(X - x_i) + f_{iy}(Y - y_i) \\ \quad + f_{iz}(Z - z_i) & \text{if } f_{ix},\ f_{iy} \text{ and} \\ & f_{iz} \text{ exist} \\ \text{INVALID} & \text{otherwise.} \end{cases}$$

$\bar{\mathcal{T}}_p(p_i, x) = 0$ divides E^3 into two planar half spaces, and the one containing the inward-pointing normal to the feature x at p_i is denoted by $\mathcal{G}_p(p_i, x)$.

Now, we define the function $\mathcal{C}_p$ as follows:

$$\mathcal{C}_p(x) = \bigcap_{p_i} \mathcal{G}_p(p_i, x)\ \forall p_i \in b(p) \text{ wherever } \mathcal{G}_p(p_i, x) \text{ is defined.}$$

At any point p_i on p, let us denote the tangent plane by $\mathcal{T}_p(p_i, x) = 0$. $\mathcal{T}_p(p_i, x) = 0$ divides E^3 into two planar half spaces, and the one containing the inward-pointing normal to the feature x at p_i is denoted by $\mathcal{H}_p(p_i, x)$.

Now, we define the function $\mathcal{A}_p$ as follows:

$$\mathcal{A}_p(x) = \bigcap_{p_i} \mathcal{H}_p(p_i, x)\ \forall p_i \in p \text{ wherever } \mathcal{H}_p(p_i, x) \text{ is defined.}$$

Let us denote $c^*\mathcal{A}_p(x)$ by $\mathcal{B}_p(x)$.

Now, $\mathcal{I}_p(x)$ is defined as

$$\mathcal{I}_p(x) = (\mathcal{B}_p(x) \cap \mathcal{C}_p(x)) \cup \text{ps}(x).$$

Now, we will illustrate the notion of infinite extension through some examples. Consider the feature f showed in Fig. 8. Fig. 8(a) shows the patch p that has been identified on the feature. To compute $\mathcal{C}_p(f)$, we first construct tangent planes on the boundary of the patch and consider the intersection of the half spaces. The $\mathcal{C}_p(f)$ thus obtained is shown in Fig. 8(b). To compute $\mathcal{A}_p(f)$, we construct the tangent planes at each point on the patch p. The resulting $\mathcal{A}_p(f)$ is shown in Fig. 8(c). $\mathcal{B}_p(f)$, which is the regularized complement of $\mathcal{A}_p(f)$, is shown in Fig. 8(d). Finally, $\mathcal{I}_p(f)$ is shown in Fig. 8(e).

Figs. 9(a) through (e) illustrate the same ideas for a second example.

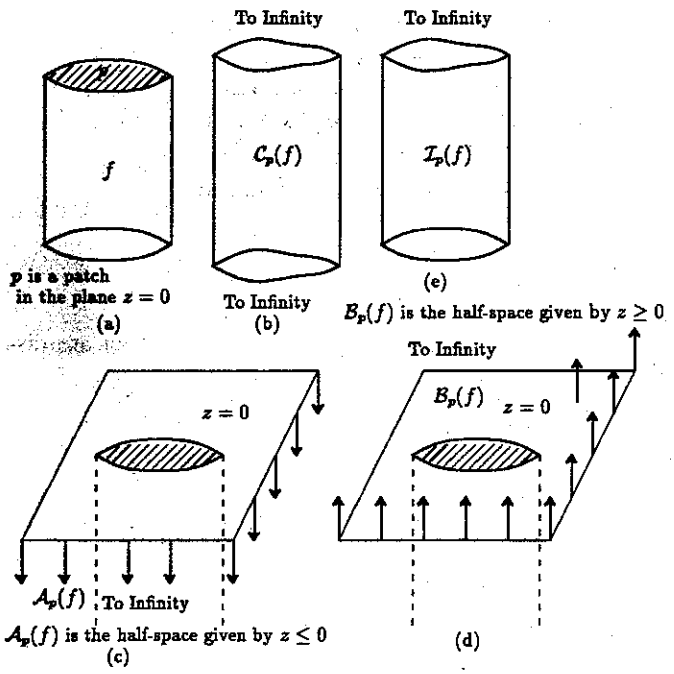

Fig. 8. Example illustrating infinite extension.

4) 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:

$$x\mathcal{M}_p y = \langle u, v \rangle$$

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 $p \subseteq p_1$.

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

The rationale for the above scheme of labeling the patches is as follows: If a patch p of $x\mathcal{M}_p y$ belongs to $(\text{ps}(x) \cup^* \text{ps}(y))\mathbf{IN}\mathcal{I}_p(x) \cup (\text{ps}(x) \cup^* \text{ps}(y))\mathbf{WITH}\mathcal{I}_p(x)$, then $p \subset b(\text{ps}(x) \cup^* \text{ps}(y))$, and hence, the patch p must have the same label as a patch p_1 of $\text{ps}(x) \cup^* \text{ps}(y)$ that contains p. If there is no single patch p_1 that contains p, then we can either split p or combine some patches of $b(\text{ps}(x) \cup^* \text{ps}(y))$ to satisfy this property. If a patch p of $x\mathcal{M}_p y$ belongs to $\mathcal{I}_p(x)\mathbf{IN}(\text{ps}(x) \cup^* \text{ps}(y))$, then this patch is **IN** with respect to $\text{ps}(x) \cup^* \text{ps}(y)$, and hence, the interior of the patch p cannot belong to the boundary of the final part. Therefore, p should be labeled UNBLOCKED.

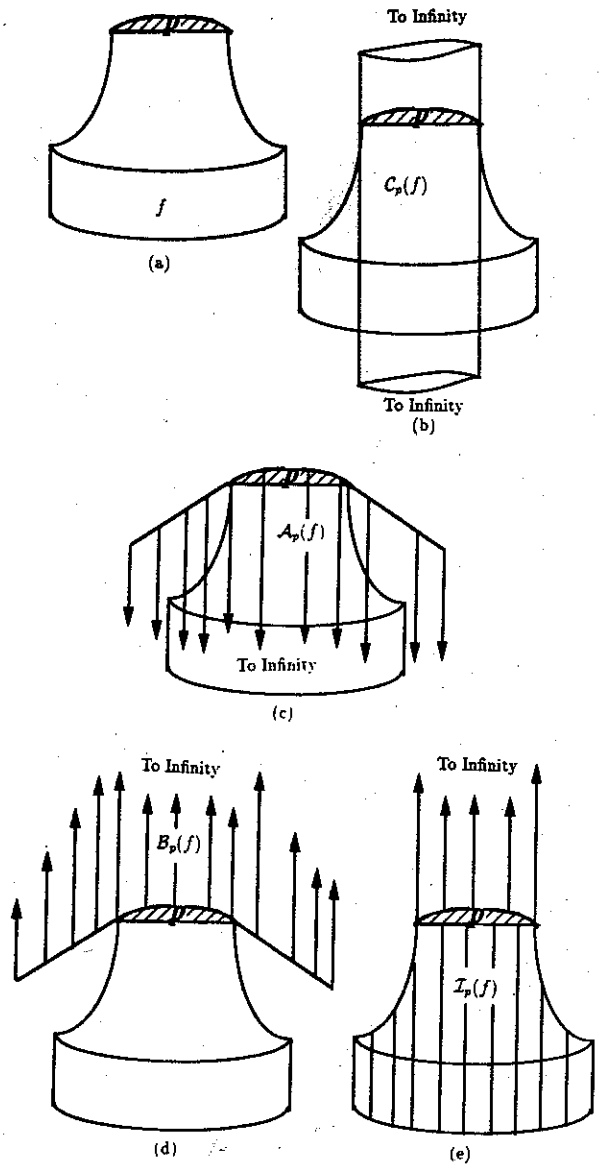

Fig. 9. Example illustrating infinite extension.

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

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

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

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

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

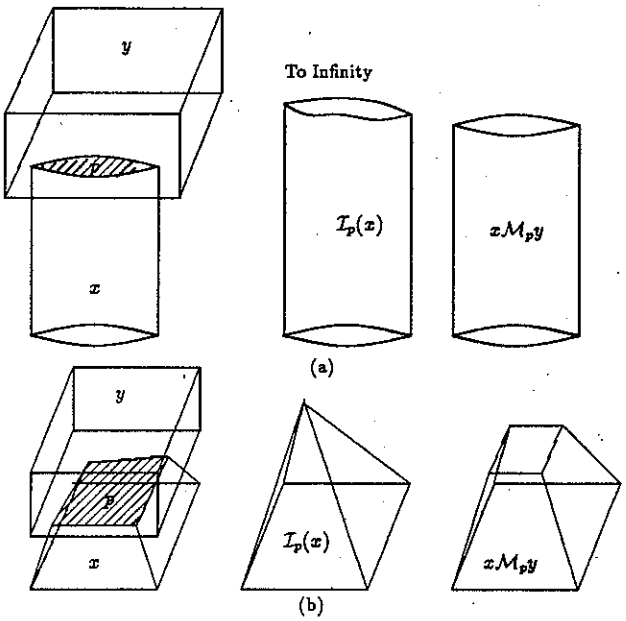

Fig. 10. Examples of maximal extension operation.

The rationale for the above labeling scheme is as follows: If a patch p of $\text{ps}(x) \cup^* \text{ps}(y)$ belongs to $\text{ps}(x)$**OUT**$\text{ps}(y)$, then $p \subset b(\text{ps}(x))$, and hence, the patch p must have the same label as a patch p_1 of x that contains p. If a patch p of $\text{ps}(x) \cup^* \text{ps}(y)$ belongs to $\text{ps}(y)$**OUT**$\text{ps}(x)$, then $p \subset b(\text{ps}(y))$, and hence, the patch p must have the same label as a patch p_2 of y that contains p. If a patch p of $\text{ps}(x) \cup^* \text{ps}(y)$ belongs to $\text{ps}(x)$**WITH**$\text{ps}(y)$, then $p \subset b(\text{ps}(x))$ and $p \subset b(\text{ps}(y))$, and hence, the patch p must have the same label as a patch p_1 of x that contains p or the patch p_2 of y that contains p. (Both p_1 and p_2 will have the same label.) In the above cases, if there is no single patch p_1 that contains p, then we can either split p or combine some patches of $b(\text{ps}(x))$ or $b(\text{ps}(y))$ to satisfy the above properties. Fig. 10 (a) and (b) illustrate some examples of maximal extension.

C. Properties of the Algebra of Features

In this section, certain properties of the algebra of features are discussed. 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 with the same set of points 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.

Let us now try to prove some properties of the features and their interactions. We begin by stating a few results for regular sets, (which in our case apply to features) from Kuratowski [19] and Requicha and Tilove [31]

Theorem 1 (Requicha and Tilove): The regular sets are a *boolean algebra* with operations $\cup^*$, $\cap^*$, and c^*, i.e., they satisfy the properties stated below. Let X, Y, and Z be arbitrary regular sets, let W be the universal set, and let $\varnothing$ be the empty set.

1) Union and intersection are commutative:

$$X \cup^* Y = Y \cup^* X;$$
$$X \cap^* Y = Y \cap^* X.$$

2) Each of the operations union and intersection are distributive over the other:

$$X \cup^* (Y \cap^* Z) = (X \cup^* Y) \cap^* (X \cup^* Z);$$
$$X \cap^* (Y \cup^* Z) = (X \cap^* Y) \cup^* (X \cap^* Z).$$

3) The empty set $\varnothing$ and the universal set W are identity elements for the union and intersection operations:

$$X \cup^* \varnothing = X;$$
$$X \cap^* W = X.$$

4) The complement satisfies the following properties:

$$X \cup^* cX = W;$$
$$X \cap^* cX = \varnothing.$$

Proposition 2 (Requicha and Tilove): If X and Y are regular sets, then

$$X \cup^* Y = X \cup Y.$$

Proposition 3 (Requicha and Tilove): If X and Y are regular

$$X -^* Y = X \cap^* cY = X \cap^* c^* Y.$$

The next two results are from Kuratowski [19] (with slight modification).

Proposition 4 (Kuratowski): The regularized union operation on regular sets is associative.

Proposition 5 (Kuratowski): The regularized intersection operation on regular sets is associative.

Proposition 6: Given a feature x and a patch p of x

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

Proof:

$$\mathcal{I}_p(x) = (\mathcal{B}_p(x) \cap \mathcal{C}_p(x)) \cup \text{ps}(x).$$

Therefore

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

□

Proposition 7: For any three features x, y, and z, the following result holds:

$$(\text{ps}(x) -^* \text{ps}(y)) -^* \text{ps}(z) = (\text{ps}(x) -^* \text{ps}(z)) -^* \text{ps}(y).$$

Proof:

$$(\text{ps}(x) -^* \text{ps}(y)) -^* \text{ps}(z)$$
$$= (\text{ps}(x) \cap^* c^*\text{ps}(y)) \cap^* c^*\text{ps}(z) \quad \text{(from Proposition 3)}$$
$$= \text{ps}(x) \cap^* c^*\text{ps}(y) \cap^* c^*\text{ps}(z) \quad \text{(from Proposition 5).}$$

Similarly

$$(\text{ps}(x) -^* \text{ps}(z)) -^* \text{ps}(y) = \text{ps}(x) \cap^* c^*\text{ps}(y) \cap^* c^*\text{ps}(z).$$

□

Proposition 8: Given a feature x and a patch p of x, if $\mathcal{I}_p(x) = \text{ps}(x)$, then $\text{ps}(x\mathcal{M}_p y) = \text{ps}(x)$, for any feature y.

Proof: Since $\mathcal{I}_p(x) = \text{ps}(x)$, $\mathcal{I}_p(x) \neq$ INVALID. Therefore

$$x\mathcal{M}_p y = \langle \text{ps}(x\mathcal{M}_p y), v \rangle$$

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

$$\begin{aligned}\text{ps}(x\mathcal{M}_p y) &= \mathcal{I}_p(x) \cap^* (\text{ps}(x) \cup^* \text{ps}(y)) \\ &= \text{ps}(x) \cap^* (\text{ps}(x) \cup^* \text{ps}(y)) \\ &= \text{ps}(x) \cap^* (\text{ps}(x) \cup \text{ps}(y)) \quad \text{(from Proposition 2)} \\ &= ki(\text{ps}(x) \cap (\text{ps}(x) \cup \text{ps}(y))) \\ &= ki(\text{ps}(x)) \\ &= \text{ps}(x).\end{aligned}$$

□

Since the infinite extension for most features and patches is an infinite solid, one might doubt the applicability of the above proposition. However, one might note that in any implementation, we delimit the infinite extension at the boundaries of the stock, and therefore, infinite extension will be a finite solid. Thus, there will be a lot of cases where $\mathcal{I}_p(x) = \text{ps}(x)$. In all such cases, we never need to compute $x\mathcal{M}_p y$ for any feature y or any patch p because $\text{ps}(x\mathcal{M}_p y) = \text{ps}(x)$.

The following proposition is used in proving Proposition 10. Proposition 10 provides an easy way to detect the cases where $\text{ps}(x\mathcal{T} y) = \text{ps}(x)$ in the domain of convex features. In such cases, we need not compute $x\mathcal{T} y$ since it will not result in a new feature. The proposition states that if we can find a patch p of x that is **ANTI** with respect to y, then we can infer $\text{ps}(x) -^* \text{ps}(y) = \text{ps}(x)$ and $\text{ps}(y) -^* \text{ps}(x) = \text{ps}(y)$.

Proposition 9: Given two features x and y, if $\text{ps}(x)$ and $\text{ps}(y)$ are convex sets and if a patch p of x is **ANTI**$\text{ps}(y)$, then

$$\text{ps}(x) \cap^* \text{ps}(y) = \varnothing.$$

Proof: At every point p_i on the patch p, consider the tangent planes to $\text{ps}(x)$ and $\text{ps}(y)$ and the half spaces that contain $\text{ps}(x)(G_i)$ and $\text{ps}(y)(H_i)$. Since the patch p is **ANTI**$\text{ps}(y)$, at any point $p_i \in p$ $G_i \cap^* H_i = \varnothing$. Therefore

$$\bigcap_{p_i} G_i \cap^* \bigcap_{p_i} H_i = \varnothing$$

for all $p_i \in p$. Since each $G_i \supseteq \text{ps}(x)$ and $H_i \supseteq \text{ps}(y)$, $\text{ps}(x) \cap^* \text{ps}(y) = \varnothing$. □

Proposition 10: Given two features x and y, if $\text{ps}(x)$ and $\text{ps}(y)$ are convex sets, and if a patch p of x is **ANTI**$\text{ps}(y)$, then

$$\text{ps}(x) -^* \text{ps}(y) = \text{ps}(x) \textit{ and } \text{ps}(y) -^* \text{ps}(x) = \text{ps}(y).$$

Proof: The result follows directly from Proposition 9. □

IV. Restricted Feature Algebras

The previous section described an algebra of features, viz., the domain, the operations, and the properties. The algebra was defined in a very general way in an effort to include every feature that could ever be of interest to manufacturing. However, the definition of the algebra does not specify any algorithms for performing the operations in the algebra—nor would this be possible since the features are described by collections of arbitrary analytic functions. In order to develop algorithms implementing the algebraic operations, we will need to restrict ourselves to some subset of the algebra by placing restrictions on the features and operators. Ideally, the subset would include all features and interactions of interest in manufacturing—but this seems infeasible since there is no general agreement on what features and interactions these might be. What is considered to be a machinable feature may vary from one manufacturing domain to another and from one shop floor to another. Rather than trying to enumerate all features of interest in manufacturing, we instead present a simple example of how a subset of the algebra can be formulated and implemented computationally. For this subset, the domain consists of rectangular solids and cylinders that have their planar faces parallel to the faces of the stock. Rectangular solids 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 *cutout*, a *notch*, etc.[3] The common manufacturing feature that is cylindrical in shape is a *hole*.

A prototype of the feature algebra encompassing the restricted feature algebra described here has been implemented in Lisp on a TI-Explorer. The details of the implementation are beyond the scope of this paper and are discussed in [16]. We are currently working on a feature algebra encompassing a much larger collection of features than described here [7]. This algebra covers a number of features that occur in machining.

A. Computing the Operations

For this restricted class, the operation $\mathcal{T}$ will be illustrated. For this set of features, $\text{ps}(x) -^* \text{ps}(y)$ could be the point set of a single feature or a pair of features or it may be a meaningless object as far as this domain is concerned. Fig. 11 shows examples of the cases that are not of interest because $\text{ps}(x) -^* \text{ps}(y)$ is neither a rectangular solid nor a cylinder.

Propagation of labels is straightforward once $\text{ps}(x) -^* \text{ps}(y)$ is computed. One way to compute $x\mathcal{T} y$ would be to compute $\text{ps}(x) -^* \text{ps}(y)$ using a solid modeler and then test if it can be

[3] For manufacturing purposes, many of these features should be modeled as rectangular solids with rounded corners, and we are currently specifying and implementing a subset of the algebra that will allow such features. However, we disallow rounded corners in this paper because they simplify the mathematical treatment while still illustrating the basic concept.

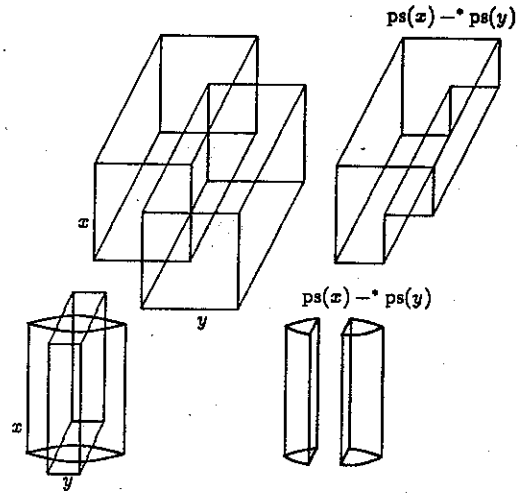

Fig. 11. Examples of cases where $\text{ps}(x) -^* \text{ps}(y)$ does not result in a object of interest.

considered to be the point set of a feature or a pair of features. (We have thus extended $x\mathcal{T}y$ to return either one feature or a pair of features.) However, this would be quite inefficient and would not take advantage of the restricted set of features and interactions that we have here. More efficient methods are described in the next section.

1) Truncation: If x is a rectangular solid, let $\{x_i | i = 1, \ldots, 6\}$ be the faces of x, and if x is a cylinder, let $\{x_i | i = 1, \ldots, 3\}$ be the faces of x (where two of them are planar faces and one is a cylindrical face). We will also number the faces such that $x_1 || x_2$. Thus, x_1 and x_2 are the end faces of the cylinder. If x_i is a face, then let x_i' denote the maximal subpatch of x_i for which $x_i'\textbf{OUT}y$ is true.

For the cases we have below, the equation for the labels (equation (1)) of the patches of $x\mathcal{T}y = \langle u, v \rangle$ can be simplified as follows:

For every patch p in v, if $p \in \text{ps}(x)\textbf{OUT}\text{ps}(y)$, let p_1 be the patch that contains p

$$\text{label}(p) = \begin{cases} \text{label}(p_1) & \text{if } p \in \text{ps}(x)\textbf{OUT}\text{ps}(y) \\ \text{UNBLOCKED} & \text{otherwise.} \end{cases} \quad (2)$$

The following is a procedure for computing $x\mathcal{T}y$:

Case 1: x is a rectangular solid.

Case a) *$ps(x\mathcal{T}y)$ is two rectangular solids:* (see Fig. 12). If $x_1\textbf{OUT}y$ and $x_2\textbf{OUT}y$ and $\forall i \in \{3, \ldots, 6\} x_i'$ consists of two disjoint rectangles, then $\text{ps}(x\mathcal{T}y)$ is two rectangular solids. Let the two rectangles of x_i' be x_i^1 and x_i^2, where x_i^1 is adjacent to x_1, and x_i^2 is adjacent to x_2. Thus, the superscripts 1 and 2 are used to denote the two rectangles of x_i'. The faces x_1, x_3^1, x_4^1, x_5^1, and x_6^1 determine one rectangular solid. The faces x_2, x_3^2, x_4^2, x_5^2, and x_6^2 determine another rectangular solid. Thus, the two rectangular solids in $ps(x\mathcal{T}y)$ are determined.

Case b) *$ps(x\mathcal{T}y)$ is one rectangular solid:* (see Fig. 13). If $x_1' = \emptyset$ and $x_2\textbf{OUT}y$, and $\forall i \in \{3 \ldots 6\} x_i'$ consists of a single rectangle, then $ps(x\mathcal{T}y)$ is one rectangular solid. The rectangular solid is determined by the faces x_2, x_3', x_4', x_5', and x_6'.

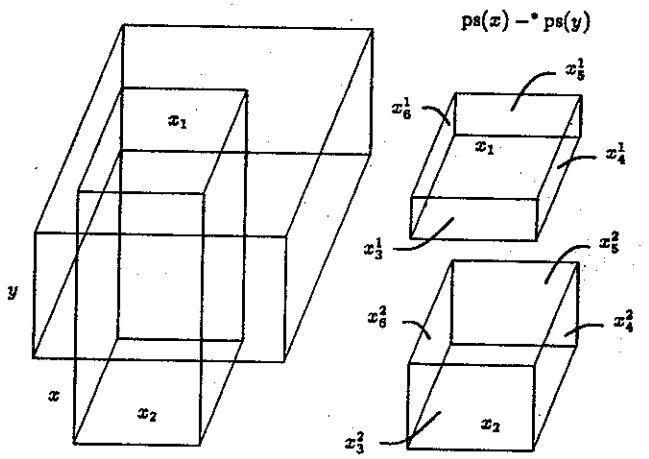

Fig. 12. Example of the case where $\text{ps}(x\mathcal{T}y)$ is two rectangular solids.

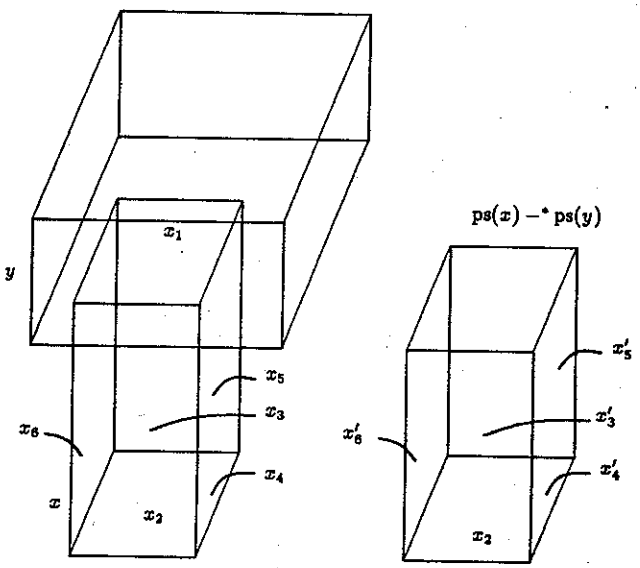

Fig. 13. Example of the case where $\text{ps}(x\mathcal{T}y)$ is one rectangular solid.

Case 2: x is a cylinder.

Case a) *$ps(x\mathcal{T}y)$ is two disjoint cylinders:* (see Fig. 14). If $x_1\textbf{OUT}y$ and $x_2\textbf{OUT}y$, and x_3' consists of two disjoint cylindrical patches, then $ps(x\mathcal{T}y)$ is two cylinders. Let the two patches of x_3' be x_3^1 and x_3^2, where x_3^1 is adjacent to x_1, and x_3^2 is adjacent to x_2. The faces x_1 and x_3^1 determine one cylinder. The faces x_2 and x_3^2 determine the another cylinder. Thus, the two cylinders in $ps(x\mathcal{T}y)$ are determined.

Case b) *$ps(x\mathcal{T}y)$ is one cylinder:* (see Fig. 15). If $x_1' = \emptyset$ and $x_2\textbf{OUT}y$, and x_3' consists of a single cylindrical patch, then $ps(x\mathcal{T}y)$ is one cylinder. The cylinder is determined by the faces x_2 and x_3'.

Case 3: If the conditions for the previous cases are not met, then $x\mathcal{T}y = \text{INVALID}$.

2) Infinite Extension: In computing infinite extension, it should be stated that in any implementation, the original

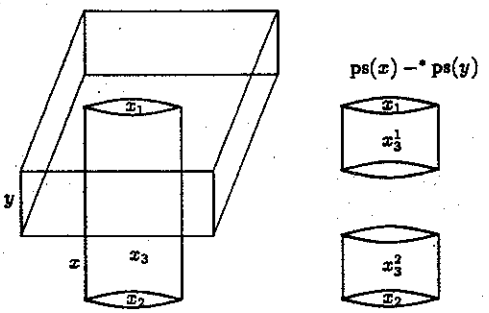

Fig. 14. Example of the case where ps(xTy) is two cylinders.

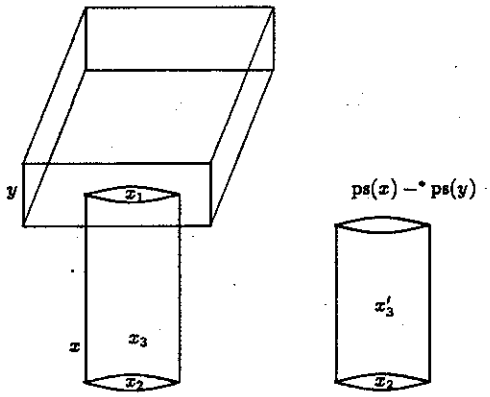

Fig. 15. Example of the case where ps(xTy) is one cylinder.

definition can be modified to delimit infinite extension to the boundaries of the stock. Since all the features are cut out starting with the stock, this does not compromise the generality or the scope of the definition in any way. Given the domain of features we have identified, an important task is to identify the patches of interest for computing infinite extension. We must have as many patches as it takes to include all the interesting shapes but not too many as to result in repeated computation of the same shape. We should also avoid patches that result in a shape for infinite extension outside our domain. For the restricted domain of rectangular solids and cylindrical holes, this task is easy. Typical examples of shapes of interest for infinite extension are shown in Fig. 16. These shapes can be generated by considering the planar faces of rectangular solids or cylinders as patches. Several other possibilities for patches can be considered, but they do not result in shapes of interest. The reader is encouraged to try out various possibilities and become convinced that the above claim is true. In the case of rectangular solids, a proof can be constructed by enumerating all the possible categories of patches. In Fig. 17, we illustrate two examples of patches that result in infinite extension that is not of interest. In Fig. 17(a), the infinite extension is identical to the feature itself, and in Fig. 17(b), it is too complex to be encompassed by our domain. Let us call this restricted form of infinite extension *face infinite extension* ($\mathcal{I}_f$) and the corresponding maximal extension *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

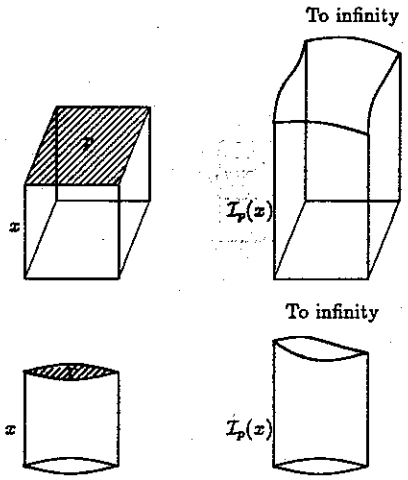

Fig. 16. Interesting shapes for infinite extension for rectangular solids and cylinders.

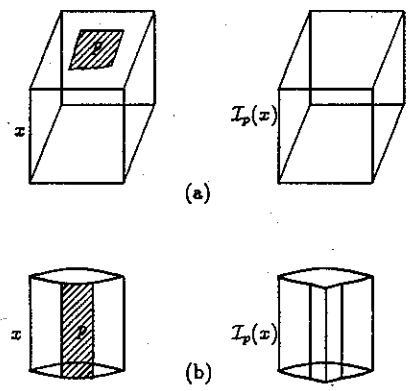

Fig. 17. Shapes not of interest for infinite extension for rectangular solids and cylinders.

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. Let us denote the set of applicable operations by Σ. Therefore, $\Sigma = \{T\} \cup \Sigma_M$. In the following discussion, given a feature x and a face x_i of x, we will denote the half space determined by the equation of x_i and containing x as X_i.

Property 1: If x is a rectangular solid, then the faces of xTy can be written as $\{xs_i | i = 1, \ldots, 6\}$, where $\forall i \neq 1$ x_i is in the same plane as xs_i. Therefore, $\mathcal{I}_{xs_1}(xTy) = X_2 \cap X_3 \cap X_4 \cap X_5 \cap X_6 = \mathcal{I}_{x_1}(x)$. If x is a cylinder, then the faces of xTy can be written as $\{xs_i | i = 1, \ldots, 3\}$, where x_2 is in the same plane as xs_2, and x_3 is in the same cylindrical surface as xs_3. Therefore, $\mathcal{I}_{xs_1}(xTy) = X_2 \cap X_3 = \mathcal{I}_{x_1}(x)$.

3) Maximal Extension: **Property 2**: If the feature x is a rectangular solid and if $x\mathcal{M}_f y$ is a valid feature (where $f = x_1$), its faces are f', x_2, x'_3, x'_4, x'_5, and x'_6, where $f' || x_2$ and x'_i, $\{i = 3 \ldots 6\}$ is in the same plane as x_i. If the feature x is a cylinder and if $x\mathcal{M}_f y$ is a valid feature, its faces are f', x_2, x'_3, where $f' || x_2$ and x'_3 is in the same cylindrical surface as x_3. Therefore, $\mathcal{I}_{f'}(x\mathcal{M}_f y) = \mathcal{I}_f(x)$.

Let x be a feature whose maximal extension with respect to a planar face $f = x_1$ into a feature y is to be computed. Let f be **IN** or **ANTI** with respect to ps(y). Otherwise, $x\mathcal{M}_f y$ is

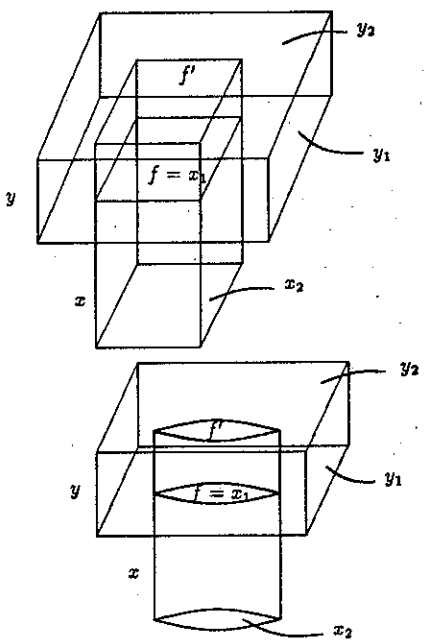

Fig. 18. Computing maximal extension.

INVALID. Let x_2 be the face of x parallel to f. Let y_1 and y_2 be the faces of y parallel to f. Let y_1 be closer to x_2 than y_2. Let $f' = \mathcal{O}(f, y_2)$. The faces f' and x_2 define a solid (a rectangular solid or a cylinder), which determines $\text{ps}(x\mathcal{M}_f y)$. This procedure is illustrated in Fig. 18. Determining the labels of the faces (or portions of the faces) once the boundary of $x\mathcal{M}_f y$ is known is straightforward. The details are omitted in this paper.

B. Properties of the Algebra

This section presents some additional properties that hold for the restricted feature algebra with which we are dealing. This algebra deals only with cylinders and rectangular solids as features, whose planar faces are parallel to the faces of the stock. When different kinds of restricted feature algebras are considered, different sets of properties hold, and these properties can be used in reducing the computations to be performed in computing new features from the old ones. The following propositions are discussed in greater detail in [16].

Proposition 11: Given features x, y, and $x\mathcal{T}y$, if the face xs_1 of $x\mathcal{T}y$ is not in the same plane as any face of x, then

$$\text{ps}((x\mathcal{T}y)\mathcal{M}_{xs_1} y) = \text{ps}(x\mathcal{M}_{x_1} y).$$

Proof:

$$\begin{aligned}
&\text{ps}((x\mathcal{T}y)\mathcal{M}_{xs_1} y) \\
&\quad = \mathcal{I}_{xs_1}(x\mathcal{T}y) \cap^* (\text{ps}(x\mathcal{T}y) \cup^* \text{ps}(y)) \\
&\quad = \mathcal{I}_{x_1}(x) \cap^* ((\text{ps}(x) -^* \text{ps}(y)) \cup^* \text{ps}(y)) \\
&\qquad \text{(from Property 1)} \\
&\quad = \mathcal{I}_{x_1}(x) \cap^* ((\text{ps}(x) \cap^* c^*\text{ps}(y)) \cup^* \text{ps}(y)) \\
&\qquad \text{(from Proposition 3)} \\
&\quad = \mathcal{I}_{x_1}(x) \cap^* ((\text{ps}(x) \cup^* \text{ps}(y)) \cap^* (\text{ps}(y) \cup^* c^*\text{ps}(y))) \\
&\qquad \text{(from Thm. 1)} \\
&\quad = \mathcal{I}_{x_1}(x) \cap^* (\text{ps}(x) \cup^* \text{ps}(y)) \\
&\qquad \text{(from Thm. 1)} \\
&\quad = \text{ps}(x\mathcal{M}_{x_1} y).
\end{aligned}$$

□

Property 12: Given features x, y, $x\mathcal{T}y$, and w, if the face xs_1 of $x\mathcal{T}y$ is not in the same plane as any face of x if $\mathcal{I}_{x_1}(x) = \text{ps}(x)$ and $\text{ps}(w) \cap^* \text{ps}(y) = \emptyset$, then

$$\text{ps}((x\mathcal{T}y)\mathcal{M}_{xs_1} w) = \text{ps}(x\mathcal{T}y).$$

Proof:

$$\begin{aligned}
\text{ps}(w) &= \text{ps}(w) \cap^* (\text{ps}(y) \cup^* c^*\text{ps}(y)) \quad \text{(from Thm. 1)} \\
&= (\text{ps}(w) \cap^* \text{ps}(y)) \cup^* (\text{ps}(w) \cap^* c^*\text{ps}(y)) \quad \text{(from Thm. 1)} \\
&= \emptyset \cup^* (\text{ps}(w) \cap^* c^*\text{ps}(y)) \\
&= \text{ps}(w) \cap^* c^*\text{ps}(y)
\end{aligned}$$

Therefore

$$\text{ps}(w) = \text{ps}(w) \cap^* c^*\text{ps}(y). \tag{3}$$

This result will be used later on in the proof.

$$\begin{aligned}
&\text{ps}((x\mathcal{T}y)\mathcal{M}_{xs_1} w) \\
&\quad = \mathcal{I}_{xs_1}(x\mathcal{T}y) \cap^* (\text{ps}(x\mathcal{T}y) \cup^* \text{ps}(w)) \\
&\quad = \mathcal{I}_{x_1}(x) \cap^* (\text{ps}(x\mathcal{T}y) \cup^* \text{ps}(w)) \\
&\qquad \text{(from Property 1)} \\
&\quad = \text{ps}(x) \cap^* ((\text{ps}(x) \cap^* c^*\text{ps}(y)) \cup^* \text{ps}(w)) \\
&\qquad \text{(from Prop. 3)} \\
&\quad = \text{ps}(x) \cap^* (\text{ps}(x) \cup^* \text{ps}(w)) \cap^* (\text{ps}(w) \cup^* c^*\text{ps}(y)) \\
&\qquad \text{(from Thm. 1)} \\
&\quad = \text{ps}(x) \cap^* (\text{ps}(x) \cup \text{ps}(w)) \cap^* (\text{ps}(w) \cup^* c^*\text{ps}(y)) \\
&\qquad \text{(from Prop. 2)} \\
&\quad = k(i(\text{ps}(x) \cap (\text{ps}(x) \cup \text{ps}(w)))) \cap^* (\text{ps}(w) \cup^* c^*\text{ps}(y)) \\
&\quad = kip\text{s}(x) \cap^* (\text{ps}(w) \cup^* c^*\text{ps}(y)) \\
&\quad = \text{ps}(x) \cap^* ((\text{ps}(w) \cap^* c^*\text{ps}(y)) \cup^* c^*\text{ps}(y)) \\
&\qquad \text{(from Eqn. 3)} \\
&\quad = \text{ps}(x) \cap^* ((\text{ps}(w) \cup^* c^*\text{ps}(y)) \cap^* c^*\text{ps}(y)) \\
&\qquad \text{(from Thm. 1)} \\
&\quad = \text{ps}(x) \cap^* c^*\text{ps}(y) \\
&\quad = \text{ps}(x) -^* \text{ps}(y) \quad \text{(from Prop. 3)} \\
&\quad = \text{ps}(x\mathcal{T}y)
\end{aligned}$$

□

C. 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. The terms feature interpretation and feature set are used synonymously in this paper. The task of generating all the feature interpretations, given one feature interpretation, can be given a state-space [26] formulation. In this formulation, a state is a feature interpretation of the part. A state is a vertex in what is known as the state-space graph. Given two elements

x and y in a state s and an operator η, if $x\eta y$ is a valid feature or a pair of valid features, then we can obtain a new feature interpretation s' as follows: If $x\eta y$ is one feature, then $s' = (s - \{x\}) \cup \{x\eta y\}$, and if $x\eta y$ is a pair of features, then $s' = (s - \{x\}) \cup (x\eta y)$. After deriving s' from s, we can draw a directed edge labeled $x\eta y$ from s to s'. By repeated application of the above step, we can derive all the possible feature interpretations of the part linked to each other by directed edges. Such a graph is known as the state-space graph. We have developed an algorithm called generate-features for generating alternate feature sets given one feature set. This algorithm essentially produces the state space in a breadth-first manner.

D. Complexity

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; therefore, it is unlikely that exponential blowup would occur in *real-world* parts. Further, since the interactions occur only among a spatially connected set of features, one can reduce the number of interactions that need to be considered by partitioning the *delta volume* (the union of all the features) into as many disjoint volumes as possible. Now, we consider an example where a combinatorial explosion of the number of feature interactions occurs. Consider a part that is obtained by removing a slab of material from a piece of stock. A slab is a layer of material of uniform thickness removed from an entire face. Suppose this material has been described by the designer as composed of n slots of uniform thickness. The total number of feature interpretations for this case will be 2^{n-1}.

Lemma 1: The worst-case number of feature interpretations for a part with n features is $\Omega(2^{n-1})$.

Proof: For the example being considered, let $f_1, f_2, \ldots f_n$ be the n contiguous slots given by the designer. All the possible feature interpretations can be obtained as follows: One can think of the material of the slab as being separated into n slots by drawing $n-1$ hypothetical planes on the slab. One can derive a feature interpretation by choosing k, $0 \leq k < n$ of these $n-1$ planes. Therefore, the total number of feature interpretations is

$$\sum_{k=0}^{n-1} \binom{n-1}{k} = 2^{n-1}.$$

□

E. Illustrative Example

In Fig. 19, we show the state space for the the example discussed in Section I-B. The generate-features algorithm produces only the states and the relationships between the states. The explicit state space is not produced by generate-features.

The state-space graph has four nodes: one for each feature interpretation. The nodes are marked n_0, n_1, n_2, and n_3. Notice the labeled directed edges between states that show how one state is related to another.

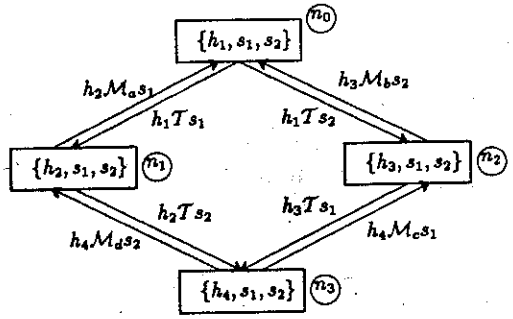

Fig. 19. State space for the example given in Section I-B.

V. Related Work

A. Image Processing Research

From a mathematical perspective, the research in mathematical morphology [34] for image processing uses a basic approach that is similar to ours. In particular, there are are several operators that can be applied to transform one image to another, such as dilation, erosion, opening, and closing, and the algebraic properties of these operators are studied so that one can replace one sequence of operators by another to obtain significant savings in computation. However, there are also significant differences. For example, the operators in our feature algebra are different from the ones used in image processing in their nature, the mode of computation, and, thus, their properties.

B. Manufacturing Research

This section summarizes the research on automatic feature extraction, design by features, and work addressing feature interactions.

1) Feature Extraction: There appear to be two dominant paradigms in the area of feature extraction: the rule-based approach and syntactic pattern recognition. A recent development [37], [13] is to combine both rules and pattern recognition techniques. The rules are mainly used for improving the computational efficiency and for handling feature interactions. Most of the researchers have used the boundary representation as the CAD model. A notable exception is Woo [43], who developed a volume-based approach for feature extraction.

Henderson's [10] work is an example of the rule-based approach. Henderson used rules to extract features from the boundary representation of the part. His system did not provide multiple feature interpretations for a part. Henderson pointed out the need for doing this in the *Future Work* section of his thesis. Kyprianou [20] used syntactic pattern recognition techniques to extract features from the boundary representation of a part. Dong and Wozny [5] have improved on the techniques used by Kyprianou to develop algorithms that can recognize a wider class of features. De Floriani [4] has developed algorithms for feature extraction using the connectivity information between vertices, edges, and faces of a solid. Similar information has been used by Joshi and Chang [13] for feature extraction. Joshi and Chang have also addressed certain kinds feature interactions by performing

geometric checks during feature recognition. Srinivasan and Liu [37] have developed a tree grammar (very similar to an AND/OR graph) for translating from the boundary representation of a part to the process plan. They have also addressed feature interactions within a process model.

2) Design by Features: In recent years, design-by-features systems have been developed for a variety of domains. Two of the design-by-features systems developed for the machining domain are the VWS2 [17] system developed at the National Institute of Standards and Technology (NIST) and XCUT [2], which was developed at Bendix Kansas City Division. These two systems are fairly complete in the sense that they translate from part specifications to low-level process plans, but they require the designer to specify the precise manufacturing features to be planned for. The design-by-features approach is also used by an integrated process planning system called FIRSTCUT [38], which is being developed at Stanford University.

3) Work Addressing Feature Interactions: Ide [12] has developed a system for feature-based design using the PADL-2 solid modeler [8]. Ide's system bridges the PADL-2 solid modeler with a process planner developed at the University of Maryland called SIPS [24]. 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 (e.g., a hole cannot intersect any other feature).

Maeda and Shinohara [22] have written an expert system called ESPER that performs geometric manipulations in generating cutter areas. This system addresses some of the nongeometric issues in process planning, but its geometric reasoning is based on feature parameters and is not integrated with a solid modeler. Hayes [9] 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. Vandenbrande [41] has developed a system that combines the principles of artificial intelligence and solid modeling. The program uses hints or clues to identify potential features in the boundary representation of a part (obtained from the PADL-2 solid modeler). The clues are generated by production rules and posted on a blackboard. The clues are assessed, and the promising ones are pursued to recognize and extract the features. This system is capable of identifying interacting features (e.g., two crossing slots). This program also produces alternative feature interpretations in certain cases. Since there is no formalization available regarding the kinds of interactions it handles, it is hard to determine all the interpretations it produces. Pratt [28] 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.

VI. Discussion

A. 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 ESPER [22] and Machinist [9] systems represent significant steps in the development of ways to handle feature interactions. However, these systems do 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.

B. 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 subalgebras that satisfy the properties of the general feature algebra, allow the algebraic operations to be computed efficiently, and include features of interest in practical problems. Provided that an appropriate subalgebra is chosen, 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. Furthermore, the properties of the feature algebra allow us to resolve many of the interactions without invoking the algebraic operators. This has the potential to result in further computational savings.

C. Future Work

This work constitutes a portion of a larger project whose goal is to develop an integrated system for design and process planning [25]. Other components of this project include the Protosolid solid modeler [42] and the EFHA [39] process planning system. Protosolid and EFHA are written in Lisp and run on a Texas Instruments Explorer. Based on the feature algebra described in this paper, we have implemented a system for feature analysis that interfaces to both Protosolid and EFHA and provides a means for communication between them. We would also like to extend the feature algebra to include some additional kinds of feature interactions. For

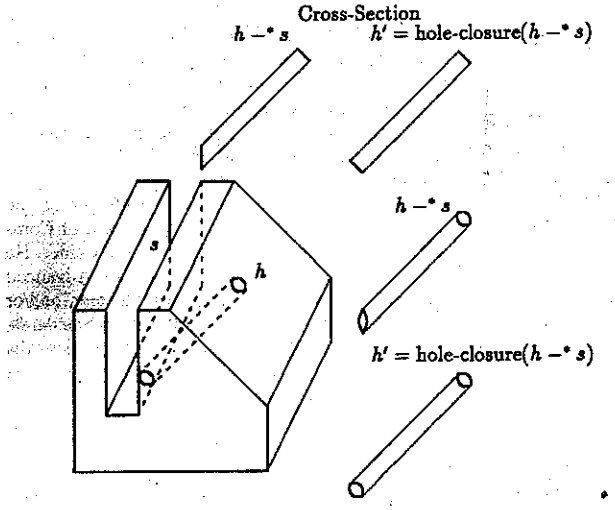

Fig. 20. Hole-closure operator.

example, in the part shown in Fig. 20, the feature $h\mathcal{T}s$ is not a useful feature for manufacturing purposes. It would be more useful to have an operator that would produce the hole h' or some extension of h'. We are currently working on extending the feature algebra to include such operators [7].

D. 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 3-D 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] M. K. Agoston, *Algebraic Topology: A First Course*. New York: Marcel Dekker, 1976.

[2] S. L. Brooks and K. E. Hummel, "XCUT: A rule-based expert system for the automated process planning of machined parts," Techn. Rep. BDX-613-3768, Bendix Kansas City Div., 1987.

[3] P. Brown and S. Ray, "Research issues in process planning at the National Bureau of Standards," in *Proc. 19th CIRP Int. Seminar Manuf. Syst.*, June 1987, pp. 111–119.

[4] L. De Floriani, "Feature extraction from boundary models of three-dimensional objects," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. 11, no. 8, pp. 785–798, Aug. 1989.

[5] X. Dong and M. Wozny, "Feature extraction for computer aided process planning," in *Proc. Third Int. Conf. Comput.-Aided Production Eng.* (Ann Arbor, MI), June 1988.

[6] W. Fulks, *Advanced Calculus, An Introduction to Analysis*. New York: Wiley, 1969.

[7] S. Gupta, R. Karinthi, D. Nau, and G. Zhang, "Multiple feature interpretations and time orderings in machining," to be published.

[8] E. E. Hartquist and A. Marisa. *PADL-2 Users Manual*. Rochester, NY: Univ. of Rochester, 1985.

[9] C. Hayes, "Using goal interactions to guide planning," in *Proc. AAAI-87 Sixth Nat. Conf. Artificial Intell.*, 1987, pp. 224–228.

[10] M. Henderson, "Extraction of feature information from three dimensional CAD data," Ph.D. thesis, Purdue Univ., West Lafayette, IN, 1984.

[11] K. E. Hummel, "Coupling rule-based and object-oriented programming for the classification of machined features," in *ASME Int. Comput. Eng. Conf.*, July 1989.

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

[13] S. Joshi and T. C. Chang, "Graph-based heuristics for recognition of machined features from a 3D solid model," *Computer-Aided Design*, vol. 20, no. 2, pp. 58–66, Mar. 1988.

[14] R. R. Karinthi and D. S. Nau, "Geometric reasoning as a guide to process planning," in *Proc. ASME Int. Comput. Eng. Conf.*, July 1989.

[15] ———, "Using a feature algebra for reasoning about geometric feature interactions," in *Proc. Eleventh Int. Joint Conf. Artificial Intell.*, Aug. 1989. pp. 1219–1224.

[16] R. Karinthi, "An algebraic approach to feature interactions," Ph.D. thesis, Univ. of Maryland, College Park, Dec. 1990.

[17] T. Kramer and J. Jun, "The design protocol, part editor, and geometry library on the vertical workstation of the automated manufacturing research facility at the National Bureau of Standards, internal rep., 1987.

[18] B. Kumar, "Feature extraction and validation within a flexible manufacturing protocol," Ph.D. thesis, Univ. of Maryland, College Park, 1988.

[19] K. Kuratowski and A. Mostowski, *Set Theory*. Amsterdam: North Holland, 1976.

[20] L. K. Kyprianou, "Shape classification in computer-aided design," Ph.D. thesis, Univ. of Cambridge, 1980.

[21] 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," *Comput. Mech. Eng.*, vol. 5, no. 3, pp. 25–33, 1986.

[22] Y. Maeda and K. Shinohara, "Geometric reasoning and organized optimization for automated process planning," in *Proc. Seventh Nat. Conf. Artificial Intell.*, Aug. 1988, p. 105–110.

[23] B. Mendelson, *Introduction to Topology*. Boston: Allyn and Bacon, 1975.

[24] D. S. Nau, "Automated process planning using hierarchical abstraction," *Texas Instrum. Tech. J.*, pp. 39–46, Winter 1987.

[25] D. S. Nau, N. Ide, R. Karinthi, G. Vanecek, and Q. Yang, "Solid modeling and geometric reasoning for design and process planning, in *Proc. Amer. Assoc. Artificial Intell. Workshop Production Planning Scheduling*, Aug. 1988, pp. 1–9.

[26] N. J. Nilsson, *Principles of Artificial Intelligence*. Palo Alto, CA: Tioga, 1980.

[27] C. Pinter, *A Book of Abstract Algebra*. New York: McGraw-Hill, 1982.

[28] M. J. Pratt, "Form features and their applications in solid modeling," in *Tutorial paper Adv. Topics Solid Modeling SIGGRAPH*, July 1987.

[29] F. P. Preparata and M. I. Shamos, *Computational Geometry, An Introduction*. New York: Springer-Verlag, 1985.

[30] A. G. Requicha, "Mathematical models of rigid solid objects," Tech. Rep. TM-28, Univ. of Rochester, Nov. 1977.

[31] A. G. Requicha and R. Tilove, "Mathematical foundations of constructive solid geometry: General topology of closed regular sets," Tech. Rep. TM-27a, Univ. of Rochester, June 1978.

[32] A. G. Requicha and H. B. Voelcker, "Boolean operations in solid modeling boundary evaluation and merging algorithms," *Proc. IEEE*, vol. 73, no. 1, pp. 30–44, 1985.

[33] M. Rogers, "Comparison of Task1 functional requirements to Task0 features technology state of the art," Tech. Rep. R-89-GM-02, CAM-I Inc., 1989.

[34] J. Serra, *Image Analysis and Mathematical Morphology*. London: Academic, 1982.

[35] J. Shah, M. Rogers, P. Sreevalsan, and A. Mathew, "Functional requirements for feature based modeling systems," Tech. Rep. R-89-GM-01, CAM-I Inc., 1989.

[36] G. Simmons, *Introduction to Topology and Modern Analysis*. New York: McGraw-Hill, 1963.

[37] R. Srinivasan and C. R. Liu, "On some important geometric issues in generative process planning," in *Proc. Winter Ann. Mtg. Amer. Soc. Mech. Eng.* (Boston, MA), Dec. 1987, pp. 229–244, 1987.

[38] J. M. Tenenbaum and M. R. Cutkosky, "First-cut: A computational framework for rapid prototyping and team design," in *Proc. AAAI Spring Symp. AI Manuf.*, Mar. 1989.

[39] S. Thompson, "Environment for hierarchical abstraction: A user guide," Master's Scholarly paper, Univ. of Maryland, May 1989.

[40] 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 *Proc. 1985 ASME Conf. Comput. Eng.*, 1985.

[41] J. Vandenbrande, "Automatic recognition of machinable features in solid models," Ph.D. thesis, Univ. of Rochester, 1990.

[42] G. Vanecek, Jr., "Set operations on volumes using decomposition methods," Ph.D. thesis, Univ. of Maryland, College Park, 1989.

[43] T C. Woo, "Feature extraction by volume decomposition," in *Proc. Conf. CAD/CAM Technol. Mech. Eng.*, 1982.

Raghu Karinthi received the bachelors degree in electrical engineering in 1984 from the Indian Institute of Technology (IIT), Madras, India. He received the M.S. and Ph.D. degrees in computer science from the University of Maryland, College Park, in 1988 and 1990, respectively.

He is currently an Assistant Professor in the Department of Statistics and Computer Science and the Concurrent Engineering Research Center (CERC) at West Virginia University, Morgantown. At CERC, he is currently involved in a project on enterprise integration. His current research interests include solid modeling, geometric reasoning, automatic feature extraction, and concurrent engineering.

Dr. Karinthi is a member of ACM SIGGRAPH, ACM SIGART, and AAAI.

Dana Nau received the B. S. degree in applied mathematics from the University of Missouri, Rolla, in 1974. He received the A. M. and Ph.D. degrees in computer science from Duke University, Durham, NC, in 1976 and 1979, respectively, where he was supported by a National Science Foundation graduate fellowship and a James B. Duke graduate fellowship.

He is currently an Associate Professor at the University of Maryland in the Department of Computer Science and the Systems Research Center. He is also affiliated with the University of Maryland Institute for Advanced Computer Studies. His research experience has included summer and/or sabbatical appointments at IBM Research, the National Bureau of Standards, the University of Rochester, and General Motors Research Laboratories. His current research interests include AI techniques for searching, reasoning, planning, and representing knowledge and applications of AI to automated manufacturing.

Dr. Nau has received an IBM faculty development award and a National Science Foundation Presidential Young Investigator Award. He has been on the program committees for several symposia and workshops on AI for manufacturing, on several review panels for NSF, NRC, and DARPA, is the Academic Co-Director for the AAAI Special Interest Group on Automated Manufacturing (SIGMAN), and is on the editorial board for several journal and book series. He has published more than 70 technical papers.