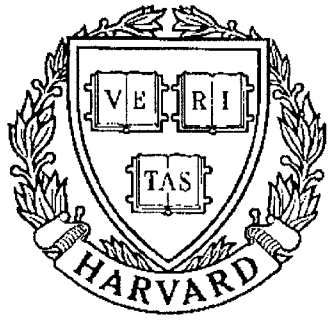
TECHNICAL RESEARCH REPORT

SYSTEMS RESEARCH CENTER



Supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), Industry and the University

An Algebraic Approach to Feature Interactions

by D.S. Nau and R.R. Karinthi

TR 89-101

CS-TR-2363
SRC-TR-89-101

November, 1989

An Algebraic Approach to Feature Interactions*

Dana S. Nau†
Raghu Karinthi‡
University of Maryland
College Park, MD 20742

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 [Nau87]) to examine these interpretations in order to see which one is most appropriate for use in manufacturing.

*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.

†Computer Science Department, Systems Research Center, and Institute for Advanced Computer Studies. Email: nau@cs.umd.edu.

‡Computer Science Department. Email: raghu@cs.umd.edu

1 Introduction

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

Design for manufacturability requires that the part designer should not confine himself to the traditional role of just developing products to meet specified functional requirements, but should also actively consider the associated manufacturing implications. For example, consider the task of designing metal parts which 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. Also, required manufacturing, assembly, and inspection operations should all be considered at the design level. This calls for a detailed knowledge of the capabilities of the manufacturing shop, which normally resides with the process planning department.

1.1 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. But the achievement of such interactions presents several unresolved problems. One of the primary problems is that generative process planning systems (such as XCUT [BH87] or SIPS [Nau87]) 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 have been proposed for how to do this, as discussed below.

1. *Automatic feature extraction* consists of automating (algorithmically?) the task of determining the manufacturing features of a part from existing CAD databases such as IGES files, Breps, etc. Prominent among these are the ones by Kyprianou [Kyp80], Henderson [Hen84], De Floriani [DF89], Kumar [Kum88] and Srinivasan [SL87].

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

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

   (b) It is difficult to extract a feature which intersects or otherwise interacts with other features, without disturbing those other features. For example, in Henderson's

feature extraction system [Hen84], once a feature volume has been recognized, it is subtracted from the overall cavity volume—making it impossible to obtain multiple feature interpretations for the same cavity volume.

2. In *design by features*, the user builds a solid model of an object by specifying directly various form features which translate directly into the relevant manufacturing features. Systems for this purpose have been built for designing injection-molded parts [VDZS85], aluminum castings, [LDS86], and machined parts [KJ87, Hum89].

   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 at the same time specify all of the manufacturable features as part of the geometry—a task which the designer is not normally qualified to do. Another problem—that of alternate feature interpretations—is described in Section 1.2.

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 [BR87]. 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.

1.2 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 [SRSM89, Rog89].

As an example, let us consider the part shown in Figure 1. In this example, the part has been described as the part resulting from subtracting a hole h_1, a slot s_1 and a slot s_2, in that order out of a rectangular stock. But because of the interaction of h_1 with s_1 and s_2, we get $h_2 = h_1 -^* s_1$, $h_3 = h_1 -^* s_2$, 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 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

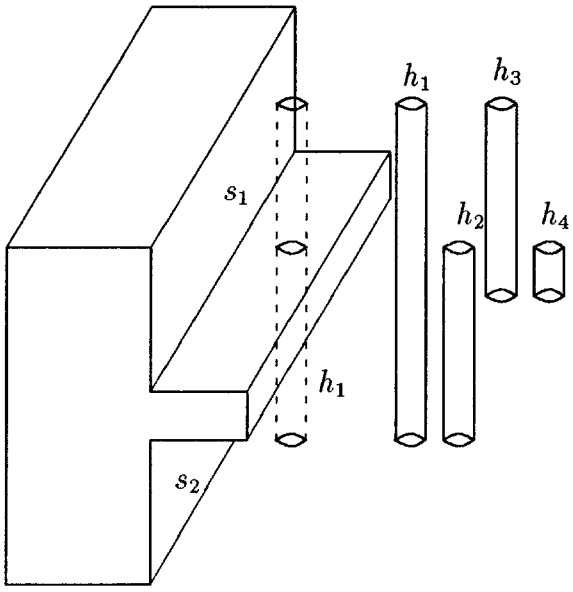


Figure 1: A part with two slots s_1 and s_2 and a hole h_1

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.[1]

We intend to use this algebra as the basis of a feature transformation system (currently being implemented), which will be capable of examining feature descriptions, computing alternate feature representations for an object, and presenting them to a process planning system. Once the implementation has been completed, it will be used as the communication interface between our Protosolid solid modeling system [Van89] and our SIPS process planning system [Nau87], in the development of an integrated system for design and process planning.

The rest of the paper is organized as follows: Section 2 introduces the mathematical concepts needed to understand the description of the feature algebra. Section 3 describes the feature algebra. Section 4 describes the feature algebra in detail for a restricted set of features. Section 5 compares our work with related research in this area. Section 6 summarizes of this research, and presents our conclusions and plans for future work.

[1]The basic idea of an algebra of features was proposed in [KN89a, KN89b]. Since then, the scope of the feature algebra has increased significantly.

2 Mathematical Preliminaries

This section summarizes some of the mathematical concepts required for the definition of a feature in the next section. For a more detailed treatment of this material the reader is referred to Requicha and Tilove [RT78, Req77] and to Kuratowski [KM76], Mendelson [Men75], Simmons [Sim63] and Agoston [Ago76].

In the summary that follows, the symbol W denotes the *universal set* and $\emptyset$ 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.

2.1 Metric Spaces and Topological Spaces

If W is a non-empty 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 non-empty 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:

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 x 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 which 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 $abs(x, y) = \mid x - y \mid$, 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.

2.2 Closed Sets

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

2.3 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 which 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.

2.4 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. Hence forth, 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 [Men75].

2.5 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 when 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 two dimensional object. Requicha and Voelcker[RV85] 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:

$$X \cup^* Y = r(X \cup Y)$$

$$X \cap^* Y = r(X \cap Y)$$

$$X -^* Y = r(X - Y)$$

$$c^*X = rcX$$

2.6 Semi-Analytic Sets

A function $f : E^3 \rightarrow \Re$ is said to be *analytic* [Ful69] 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 *semi-analytic* 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 semi-analytic set is also semi-analytic, and that class of compact, regular, semi-analytic sets is closed under regularized set operations.

2.7 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. A subset x of E^3 is said to be *concave* if it is not convex.

It can be shown that the intersection of two convex sets is a convex set [PS85]. It can easily be seen that the union and difference of two convex sets is not necessarily a convex set.

3 The Algebra of Features

An algebraic structure [Pin82] in its simplest form is a set, with a rule (or rules) for combining its elements. Let A be any set. An *operation* $*$ on A is a rule which assigns to each ordered pair $\langle x, y\rangle$ of elements of A exactly one element $x * y$ in A. 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 $*$,

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

3.1 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:

1. 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 semi-analytic.
2. The second entry ($\text{patches}(x)$) is a partition of the boundary of $\text{ps}(x)$ into labeled patches (as defined below).

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, semi-analytic 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. Figure 7 shows the BLOCKED and UNBLOCKED patches for the features h_1, s_1 and s_2 shown in Figure 1. Figure 8 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 three dimensional. Even parts with sheet metal components have a finite thickness, so this appears to be a very reasonable restriction.

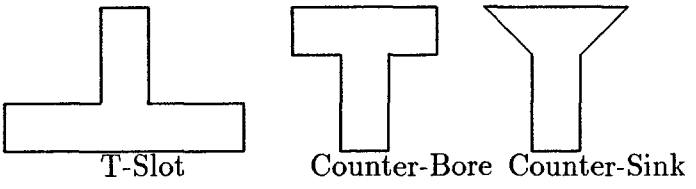


Figure 2: Cross-sectional Views of three concave features.

Since the features 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 also includes concave features such as T-slots, counter-bores and counter-sinks (see Figure 2).

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, semi-analytic sets stated earlier.

3.1.1 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.

3.1.2 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), 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 $p \in s_1$.

3.2 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[Van89] who used it in the context of performing set operations on planar polyhedra.

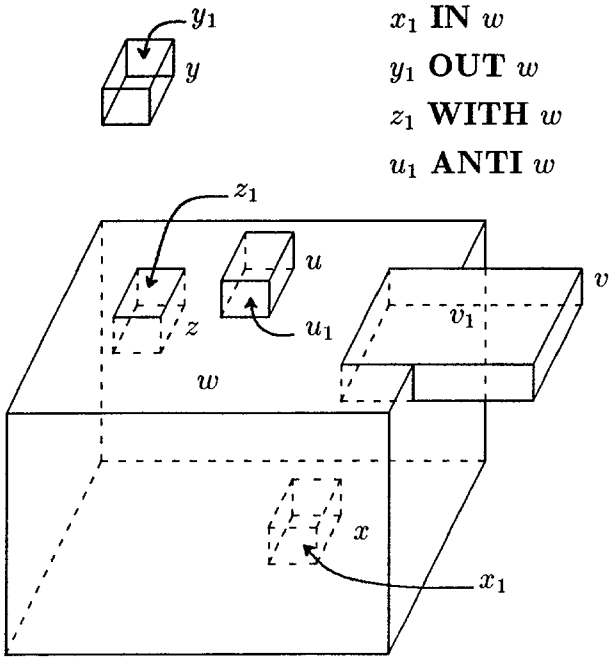


Figure 3: The classification of patches of solids x, y, z, u and v with respect to the solid w

3.2.1 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.

Figure 3 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 $x\mathbf{IN}y$, $x\mathbf{OUT}y$, $x\mathbf{WITH}y$ 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.

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

3.2.2 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 $p \subseteq p_1$.

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

3.2.3 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:

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

$\bar{\mathcal{T}}_p(p_i, x) = 0$ 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:

$$\mathcal{C}_p(x) = \begin{cases} \bigcap_{p_i} \mathcal{G}_p(p_i, x)\ \forall p_i \in b(p) \text{ wherever } \mathcal{G}_p(p_i, x) \text{ is defined} & \text{if} \\ \text{the tangent plane at all points } p_i \in b(p) \text{ does not intersect} & \\ \text{the interior of the patch } \textit{and} & \\ \text{for any pair of points } p_1 \text{ and } p_2 \in p, \text{all points in } \overline{p_1 p_2} \in \text{ps}(x) & \\ \text{INVALID} & \text{otherwise} \end{cases}$$

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:

$$\mathcal{C}_p(x) = \begin{cases} (\mathcal{B}_p(x) \cap \mathcal{C}_p(x)) \cup \text{ps}(x) & \text{if } \mathcal{C}_p(x) \neq \text{INVALID} \\ \text{INVALID} & \text{otherwise} \end{cases}$$

Figures 9 and Figure 10 illustrate some examples of infinite extension for convex and concave features respectively.

3.2.4 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:

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

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,

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

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 $p \subseteq p_2$.

Let an arbitrary patch of $b(\text{ps}(x) \cup^* \text{ps}(y))$ 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 $b(\text{ps}(x) \cup^* \text{ps}(y))$ are defined below:

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

3.3 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[KM76], Requicha and Tilove[RT78]

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,* W *the universal set and* $\emptyset$ *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* $\emptyset$ *and the universal set* W *are identity elements for the union and intersection operations:*

$$X \cup^* \emptyset = X;$$

$$X \cap^* W = X.$$

4. *The complement satisfies the following properties:*

$$X \cup^* cX = W;$$

$$X \cap^* cX = \emptyset.$$

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 [KM76] (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,*

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

Proof:

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

Therefore,

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

□

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

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

Proof:

$$\begin{aligned}
&(\mathrm{ps}(x) -^* \mathrm{ps}(y)) -^* \mathrm{ps}(z) \\
&= (\mathrm{ps}(x) \cap^* c^*\mathrm{ps}(y)) \cap^* c^*\mathrm{ps}(z) \quad \text{(from Proposition 3)} \\
&= \mathrm{ps}(x) \cap^* c^*\mathrm{ps}(y) \cap^* c^*\mathrm{ps}(z) \quad \text{(from Proposition 5)}
\end{aligned}$$

Similarly,

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

□

Proposition 8 *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.*

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

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

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

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

□

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

$$ps(x) \cap^* ps(y) = \emptyset$$

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

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

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

□

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

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

Proof: The result follows directly from Proposition 9.

□

4 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 which 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 *cut-out*, a *notch*, etc.[2] The common manufacturing feature that is cylindrical in shape is a *hole*.

[2]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 which will allow such features. However, we disallow rounded corners in this paper, because it simplifies the mathematical treatment while still illustrating the basic concept.

4.1 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. Figure 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 considered as 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.

4.1.1 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 cylindrical face). We will also number the faces such that $x_1 \| x_2$. If x_i is a face, then let x_i' denote the maximal sub-patch of x_i for which $x_i'\mathbf{OUT}y$ 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 $p \in \text{ps}(x)\mathbf{OUT}\text{ps}(y)$, let p_1 be the patch that contains p

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

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 Figure 12) If $x_1\mathbf{OUT}y$ and $x_2\mathbf{OUT}y$ 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. 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 Figure 13) If $x_1' = \emptyset$ and $x_2\mathbf{OUT}y$ 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'.

Case 2: x is a cylinder.

Case a. *$ps(x\mathcal{T}y)$ is two disjoint cylinders:* (see Figure 14) If $x_1\mathbf{OUT}y$ and $x_2\mathbf{OUT}y$ 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 Figure 15) If $x'_1 = \emptyset$ and $x_2\mathbf{OUT}y$ 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 $x\mathcal{T}y =$ INVALID .

4.1.2 Infinite Extension

The shapes of interest for $\mathcal{I}_p(x)$ are shown in Figure 16. These shapes can be generated by considering a planar face of a rectangular solid or a cylinder as a patch. Let us call this restricted form of infinite extension as *face infinite extension* ($\mathcal{I}_f$) and the corresponding maximal extension as *face maximal extension* ($\mathcal{M}_f$). Since there are six planar faces in a rectangular solid and two in a cylinder, there are six possible face infinite extensions for a rectangular solid and two for a cylinder. Let us denote the set of all possible face infinite extensions by Σ_I and the set of all possible face maximal extensions by Σ_M.

Property 1 *If x is a rectangular solid then the faces of $x\mathcal{T}y$ 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, $\mathcal{I}_{xs_1}(x\mathcal{T}y) = X_2 \cap X_3 \cap X_4 \cap X_5 \cap X_6 = \mathcal{I}_{x_1}(x)$. If x is a cylinder, then the faces of $x\mathcal{T}y$ 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, $\mathcal{I}_{xs_1}(x\mathcal{T}y) = X_2 \cap X_3 = \mathcal{I}_{x_1}(x)$.*

4.1.3 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, $\{i = 3\ldots6\}$ 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 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 ps($x\mathcal{M}_f y$). This procedure is illustrated in Figure 17. Determining the labels of the faces (or portions of the faces) once the boundary of $x\mathcal{M}_f y$ is known is straight forward. The details are omitted in this paper.

4.2 Properties of the Algebra

This section presents some additional properties that hold for the restricted feature algebra that we are dealing with. 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.

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*

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

Proof:

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

□

Proposition 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 $\mathcal{I}_{x_1}(x) = ps(x)$ and $ps(w) \cap^* ps(y) = \emptyset$, then*

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

Proof:

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

Therefore,

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

This result will be used later on in the proof.

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

□

4.3 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 Figures 4 and 5), $\bar{F}$ 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 $\bar{F} = \{\text{F}\}$. There are two additional variables called CURRENT and NEW used in this algorithm. Let Σ be the set of applicable binary operations. Thus, $\Sigma = \{\mathcal{T}\} \cup \Sigma_M$.

In the algorithm shown in Figure 4 the function new-ps-member(x, y) returns *true* if $\forall z \in y$ ps$(x) \neq$ ps(z) and *false* otherwise. In this algorithm we have to determine if if $x\eta y$ is a valid feature and if so, whether new-ps-member$(x\eta y, F)$ is true or false. The properties of the feature algebra are used in these two steps. If it is possible to determine if $x\eta y$ is INVALID or if new-ps-member$(x\eta y, F)$ is true using the properties of the feature algebra, then one need not do any further computations. Otherwise, one has to compute $x\eta y$ using the procedures described in Section 4.1 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.

4.4 Illustrative Example

This section illustrates the working of the algebra of features (and the features algorithm), through the example discussed in Section 1.2. The starting features are h_1, s_1 and s_2. In this section, the example is illustrated, by showing only the changes that occur from one iteration to the next of the outermost **while** loop (without tracing every step of the algorithm). A complete trace of the algorithm for this example is left as an exercise to the reader.

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

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

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

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

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

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

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

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

```
F = Starting set of features;
F̄ = {F};
CURRENT = F;
NEW = ∅
for each x in CURRENT do
  for each I_f ∈ Σ_I do
    compute I_f(x) ;
  end for ;
end for ;
while CURRENT ≠ ∅ do
  for each FS in F̄ do
    for each ⟨x, y⟩ such that 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
            manipulate-feature (z);
            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
              manipulate-feature (u);
            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 4: The Features Algorithm

```
procedure manipulate-feature ( x );
  if new-ps-member(x, F) then
    for each I_f ∈ Σ_I do
      compute I_f(x) ;
    end for ;
    F = F ∪ {x};
    NEW = NEW ∪ {x};
  end if ;
end manipulate-feature;
```

Figure 5: Procedure *manipulate feature* used in the features algorithm.

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

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

After the second iteration:

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

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

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

In this iteration, let us say $h_2 \mathcal{T} s_2$ was considered before considering $h_3 \mathcal{T} s_1$. Let us denote $h_2 \mathcal{T} s_2$ by h_4. Later on, when $h_3 \mathcal{T} s_1$ is considered, using Proposition 7 we infer $\mathrm{ps}(h_3 \mathcal{T} s_1) = \mathrm{ps}(h_2 \mathcal{T} s_2) = \mathrm{ps}(h_4)$. Therefore, new-ps-member$((h_3 \mathcal{T} s_1), F) = \mathit{false}$. So, $h_3 \mathcal{T} s_1$ is not considered as a new feature. Several other algebraic properties are applied at various stages.

After the third iteration :

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

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

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

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

5 Related Work

5.1 Image Processing Research

From a mathematical perspective, the research in mathematical morphology [Ser82] 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.

5.2 Manufacturing Research

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

5.2.1 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 [SL87, JC88] 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 [Woo82] who developed a volume based approach for feature extraction.

Henderson's [Hen84] 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 [Kyp80] used syntactic pattern recognition techniques to extract features from the boundary representation of a part. Dong and Wozny [DW88] have improved on the techniques used by Kyprianou to develop algorithms that can recognize a wider class of features. Leila De Floriani [DF89] 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[JC88] for feature extraction. Joshi and Chang have also addressed certain kinds feature interactions by performing geometric checks during feature recognition. Srinivasan and Liu[SL87] 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.

5.2.2 Design by Features

In the 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 [KJ87] system developed at the National Institute of Standards and Technology (NIST), and XCUT [BH87] 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[TC89] being developed at Stanford University.

5.2.3 Work Addressing Feature Interactions

Nicholas Ide [Ide87] has developed a system for feature based design using the PADL–2 solid modeler [HM85]. Ide's system bridges the PADL–2 solid modeler with a process planner developed at the University of Maryland called SIPS [Nau87]. 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).

Maeda and Shinohara[MS88] have written an expert system called ESPER that performs geometric manipulations in generating cutter areas. This system addresses some of the non-geometric issues in process planning, but its geometric reasoning is based on feature parameters and is not integrated with a solid modeler. Hayes [Hay87] 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 [RV88] 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[Pra87] 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.

6 Discussion

6.1 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 [MS88] and Machinist [Hay87] 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.

6.2 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.

Provided that an appropriate sub-algebra 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 results in further computational savings.

6.3 Future Work

This work constitutes a portion of a larger project whose goal is to develop an integrated system for design and process planning [NIK+88]. Other components of this project include the Protosolid solid modeler [Van89] and the SIPS [Nau87] process planning system. Protosolid and SIPS are written in Lisp and run on a Texas Instruments Explorer. Based on the feature algebra described in this paper, we are implementing a system for feature analysis which will interface to both SIPS and Protosolid and provide a means for communication between them. For this purpose, we intend to extend the feature algebra to incorporate tolerances and propagate them among features.

We would also like to extend the feature algebra to include some additional kinds of feature interactions. For example, in the part shown in Figure 6 the feature $h\mathcal{T}s$ is not a useful feature for manufacturing purposes. It would be more useful to have an operator which would produce the hole h' or some extension of h'. We intend to extend the feature algebra to include such operators.

6.4 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.

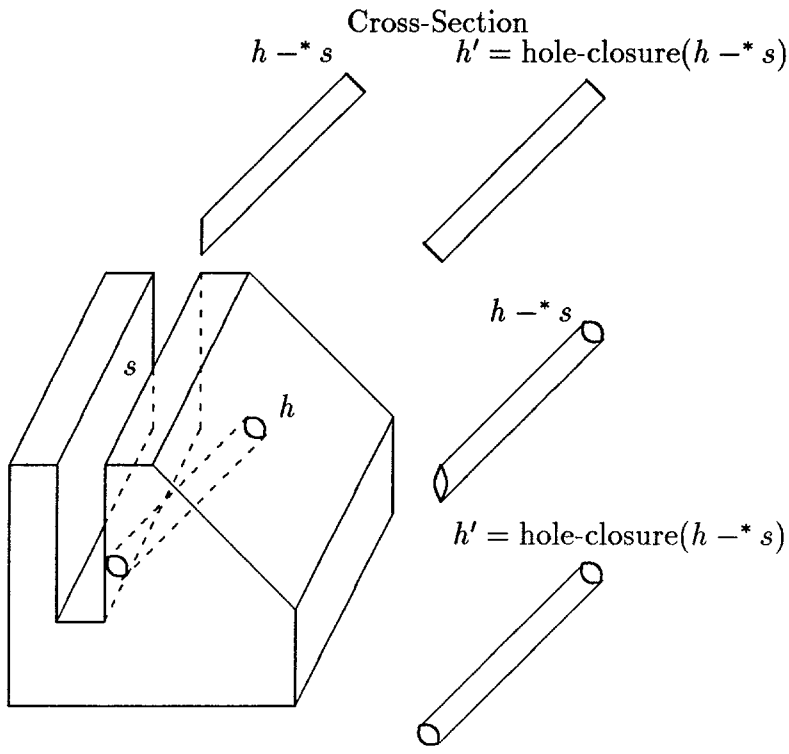


Figure 6: The hole-closure operator

References

[Ago76] M. K. Agoston. *Algebraic Topology: A First Course.* New York: Marcel Dekker, Inc., 1976.

[BH87] 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.

[BR87] P. Brown and S. Ray. Research issues in process planning at the national bureau of standards. In *Proceedings of the 19th CIRP International Seminar on Manufacturing Systems*, pages 111–119, June 1987.

[DF89] L. De Floriani. Feature extraction from boundary models of three-dimensional objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(8):785–798, Aug 1989.

[DW88] Xin Dong and Michael Wozny. Feature extraction for computer aided process planning. In *Proceedings of the Third International Conference on Computer-Aided Production Engineering, Ann Arbor, Michigan*, June 1988.

[Ful69] Walton Fulks. *Advanced Calculus, An Introduction to Analysis.* John Wiley and Sons, Inc., New York, 1969.

[Hay87] 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.

[Hen84] M. Henderson. *Extraction of Feature Information from Three Dimensional CAD Data.* PhD thesis, Purdue University, 1984.

[HM85] E. E. Hartquist and A. Marisa. *PADL–2 Users Manual.* Production Automation Project, University of Rochester, Rochester, New York, 1985.

[Hum89] K. E. Hummel. Coupling rule–based and object–oriented programming for the classification of machined features. In *ASME International Computers in Engineering Conference*, July 1989.

[Ide87] N. C. Ide. Integration of process planning and solid modeling through design by features. Master's thesis, University of Maryland, College Park, 1987.

[JC88] 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.

[KJ87] 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.

[KM76] K. Kuratowski and A. Mostowski. *Set Theory.* North Holland Publishing Company, 1976.

[KN89a] R. R. Karinthi and D. S. Nau. Geometric reasoning as a guide to process planning. In *ASME International Computers in Engineering Conference*, July 1989.

[KN89b] 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.

[Kum88] B. Kumar. *Feature Extraction and Validation within a Flexible Manufacturing Protocol.* PhD thesis, University of Maryland, College Park, 1988.

[Kyp80] L. K. Kyprinaou. *Shape Classification in Computer-Aided Design.* PhD thesis, Christ's College, University of Cambridge, 1980.

[LDS86] S. C. Luby, J. R. Dixon, and M. K. Simmons. Design with features: Creating and using a feature data base for evaluation of manufacturability of castings. *Computers in Mechanical Engineering*, 5(3):25–33, 1986.

[Men75] B. Mendelson. *Introduction to Topology.* Allyn and Bacon, Inc., 1975.

[MS88] Y. Maeda and K. Shinohara. Geometric reasoning and organized optimization for automated process planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 105–110, Aug 1988.

[Nau87] 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.

[NIK+88] D. S. Nau, N. Ide, R. Karinthi, G. Vanecek, and Q. Yang. Solid modeling and geometric reasoning for design and process planning. In *Proceedings of the American Association for Artificial Intelligence Workshop on Production Planning and Scheduling*, pages 1–9, August 1988.

[Pin82] C. Pinter. *A Book of Abstract Algebra.* McGraw–Hill Book Company, 1982.

[Pra87] 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.

[PS85] Franco P. Preparata and Ian Shamos, Michael. *Computational Geometry, An Introduction.* Springer-Verlag, New York, 1985.

[Req77] A. G. Requicha. Mathematical models of rigid solid objects. Technical Report TM–28, Production Automation Project, University of Rochester, Nov 1977.

[Rog89] Mary Rogers. Comparison of task1 functional requirements to task0 features technology state of the art. Technical Report R–89–GM–02, CAM–I Inc., 1989.

[RT78] A. G. Requicha and R. Tilove. Mathematical foundations of constructive solid geometry : General topology of closed regular sets. Technical Report TM–27a, Production Automation Project, University of Rochester, June 1978.

[RV85] A. G. Requicha and H. B. Voelcker. Boolean operations in solid modeling boundary evaluation and merging algorithms. *Proceedings of the IEEE*, 73(1):30–44, 1985.

[RV88] 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.

[Ser82] J. Serra. *Image Analysis and Mathematical Morphology.* Academic Press, London, 1982.

[Sim63] G. Simmons. *Introduction to Topology and Modern Analysis.* McGraw–Hill Book Company, 1963.

[SL87] 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.

[SRSM89] Jami Shah, Mary Rogers, Palat Sreevalsan, and Abraham Mathew. Functional requirements for feature based modeling systems. Technical Report R–89–GM–01, CAM–I Inc., 1989.

[TC89] Jay M. Tenenbaum and Mark R. Cutkosky. First-cut: A computational framework for rapid prototyping and team design. In *Proceedings of the AAAI Spring Symposium on AI in Manufacturing*, March 1989.

[Van89] G. Vanecek Jr. *Set Operations on Volumes Using Decomposition Methods.* PhD thesis, University of Maryland, College Park, 1989.

[VDZS85] M. Vaghul, J. R. Dixon, G. E. Zinmeister, and M. K. Simmons. Expert systems in a cad environment : Injection molding part design as an example. In *Proceedings of the 1985 ASME Conference on Computers in Engineering*, 1985.

[Woo82] T C. Woo. Feature extraction by volume decomposition. In *Proceedings of the Conference on CAD/CAM technology in Mechanical Engineering*, 1982.

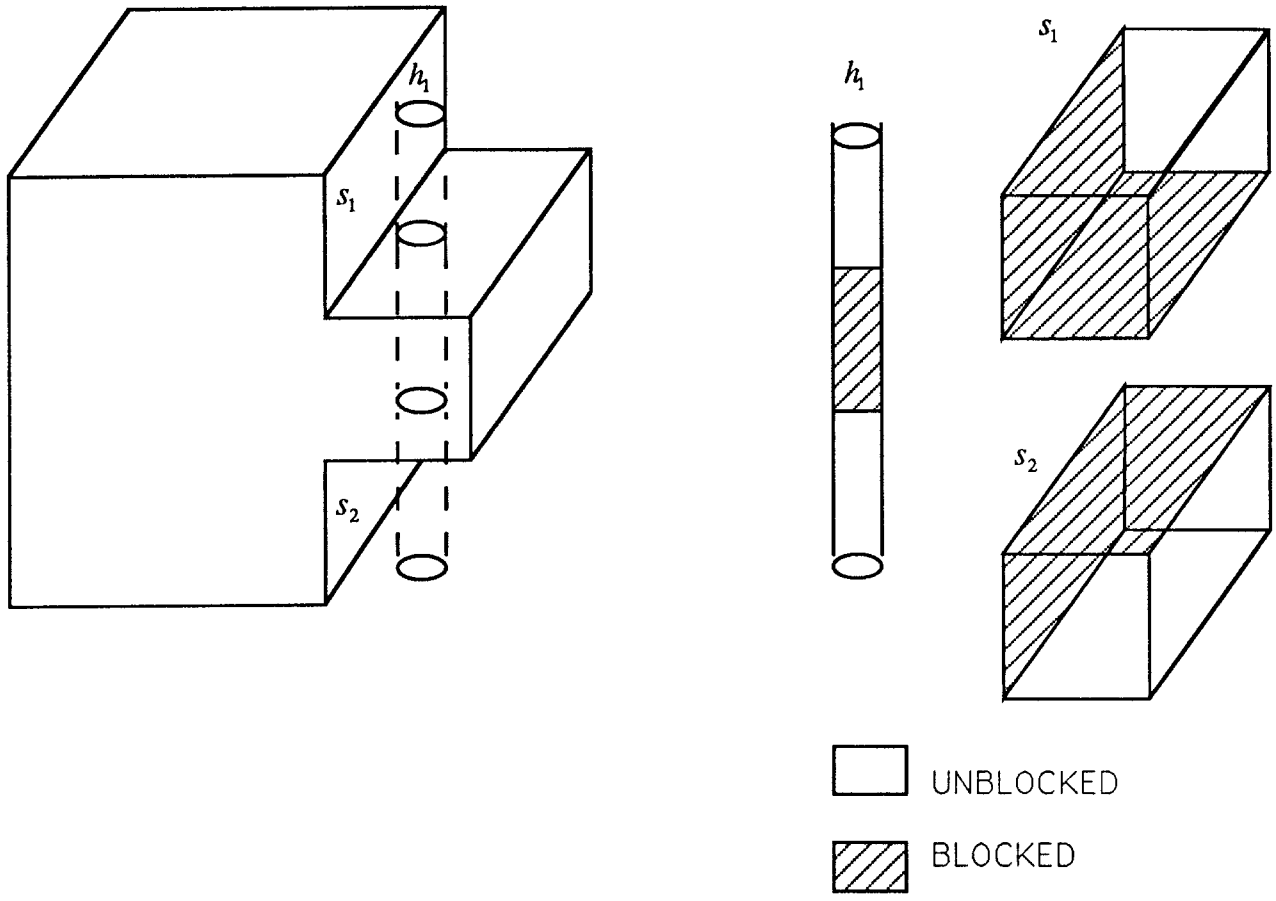


Figure 7. BLOCKED and UNBLOCKED patches of features.

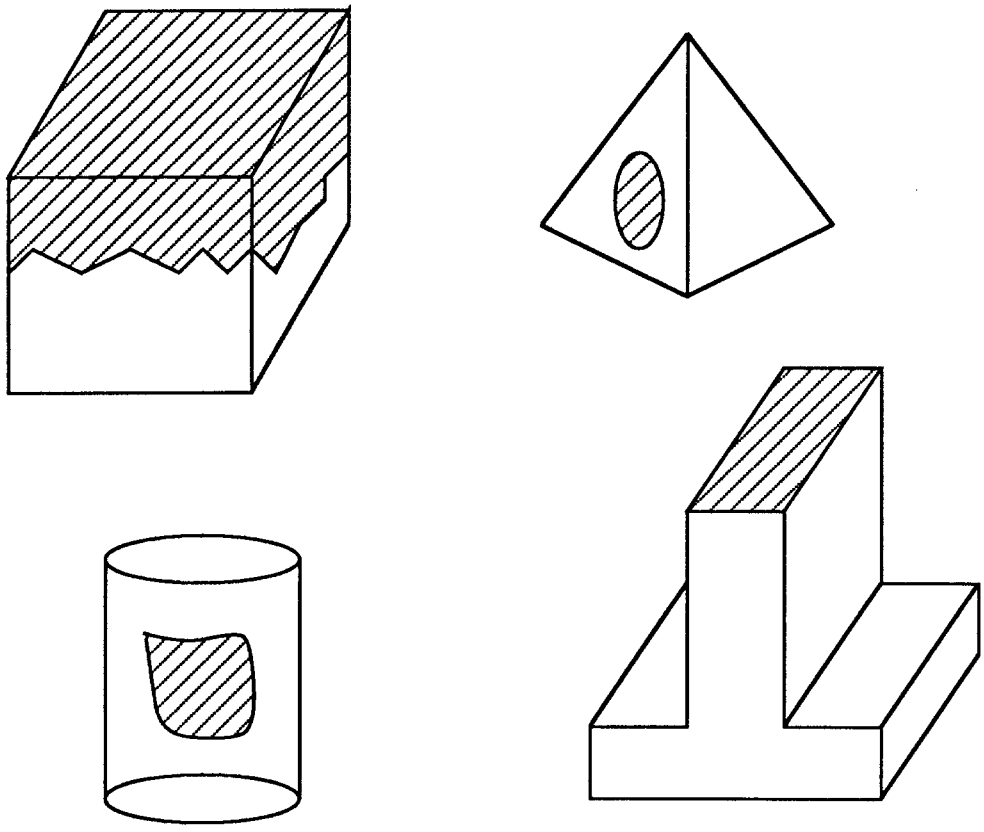


Figure 8. Some examples of patches (shaded).

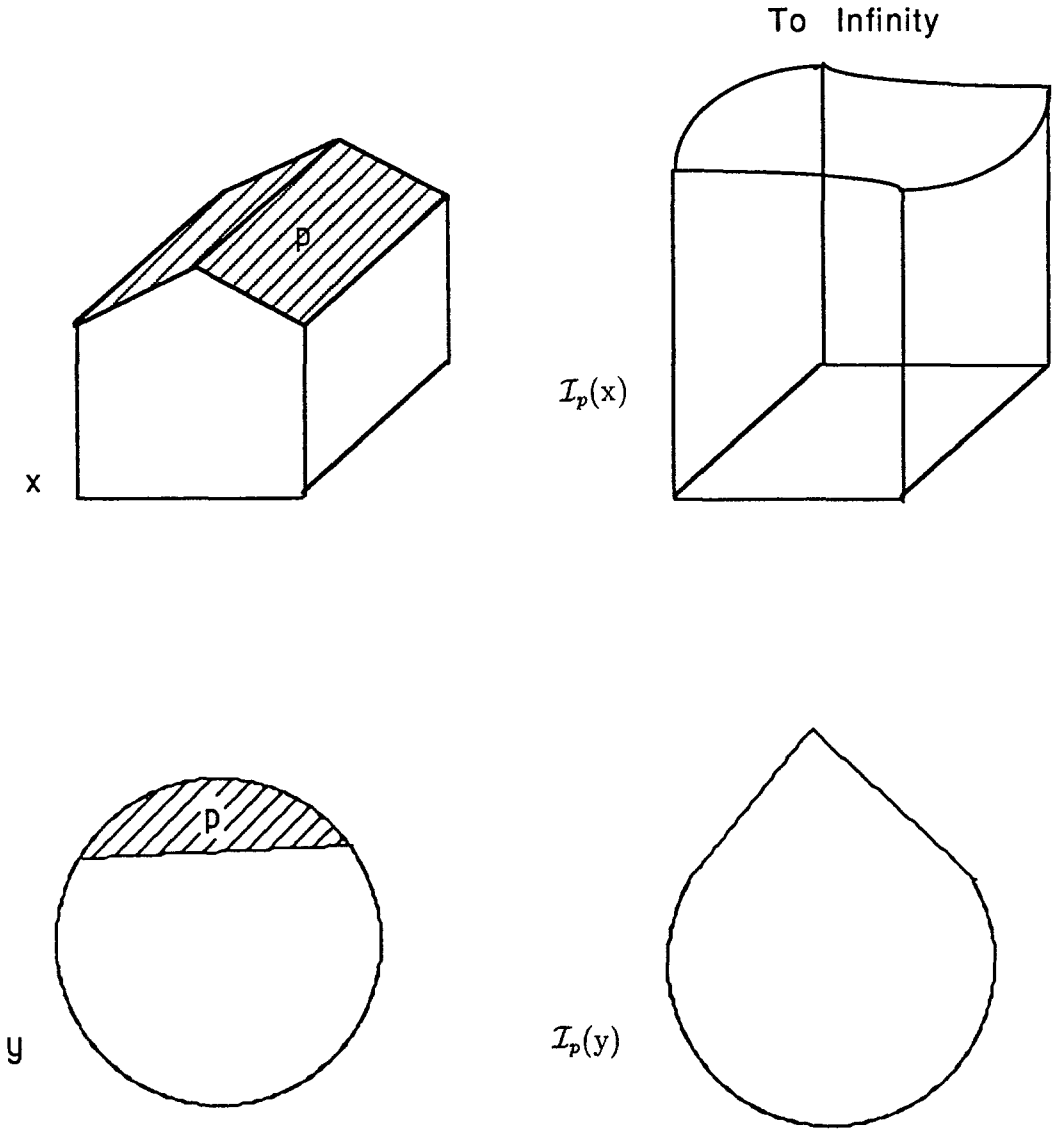


Figure 9. Examples of Infinte Extension for convex features.

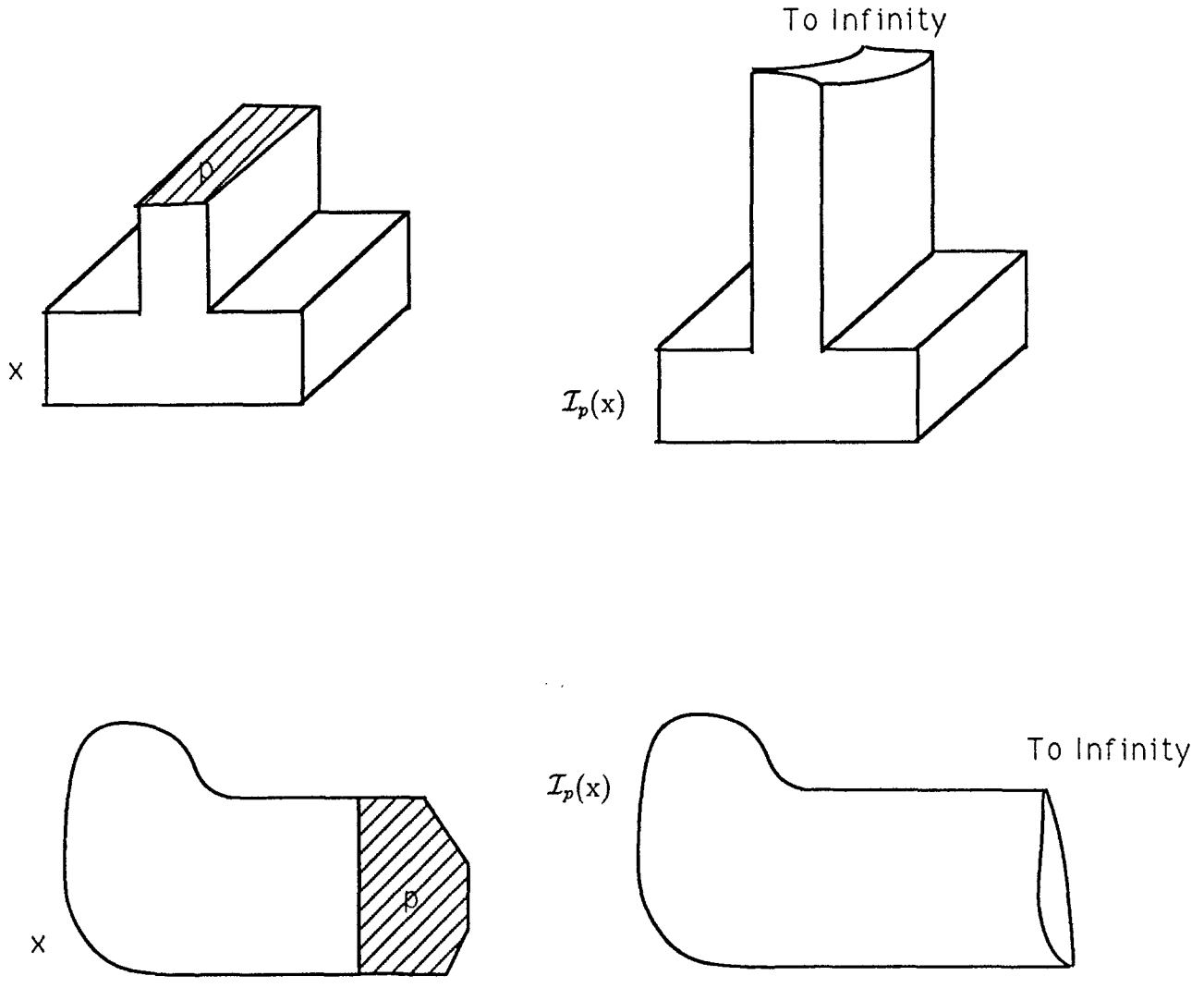


Figure 10. Examples of Infinite Extension for concave features.

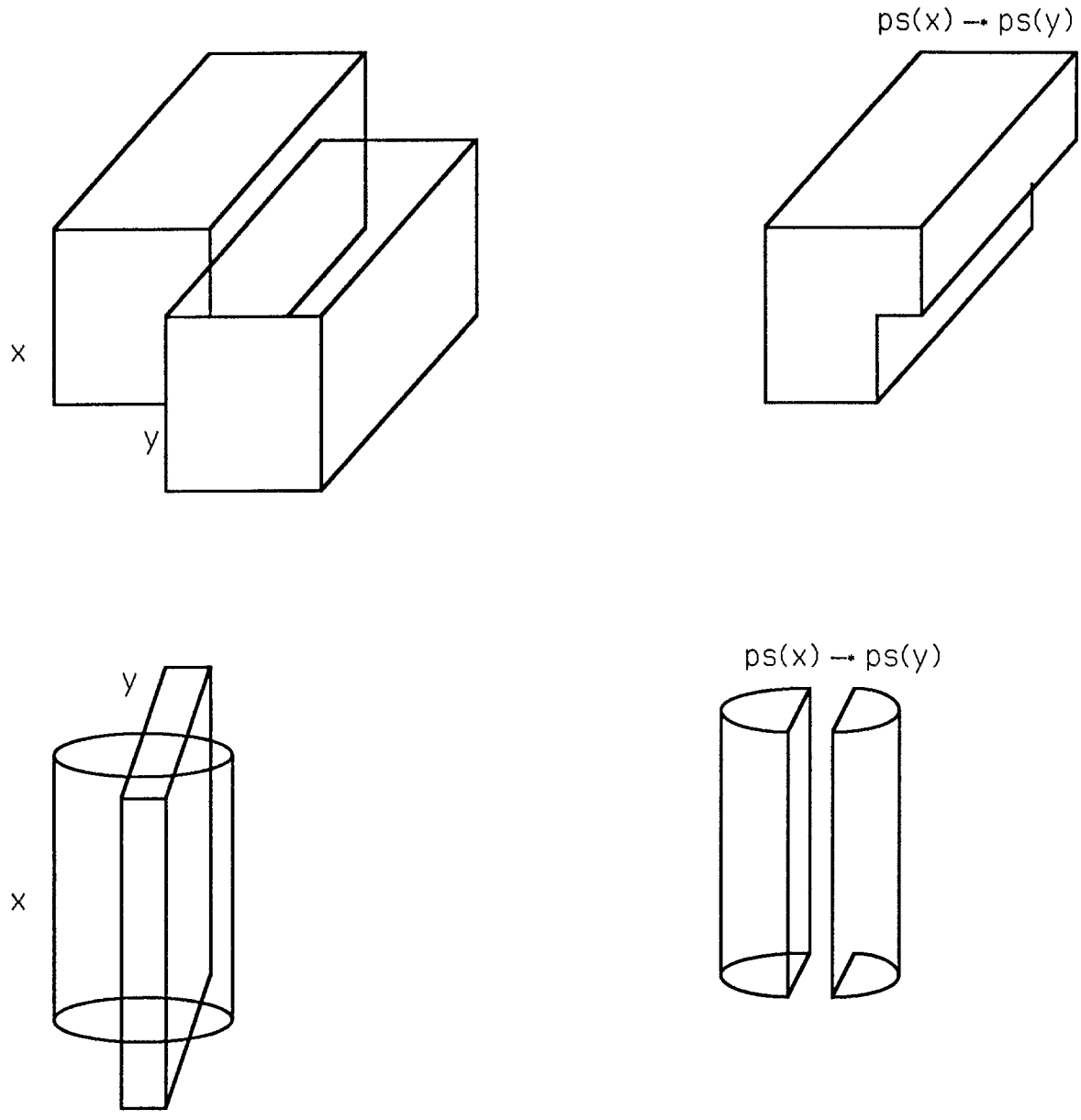


Figure 11. Cases where ps(x) → ps(y) is not a shape of interest.

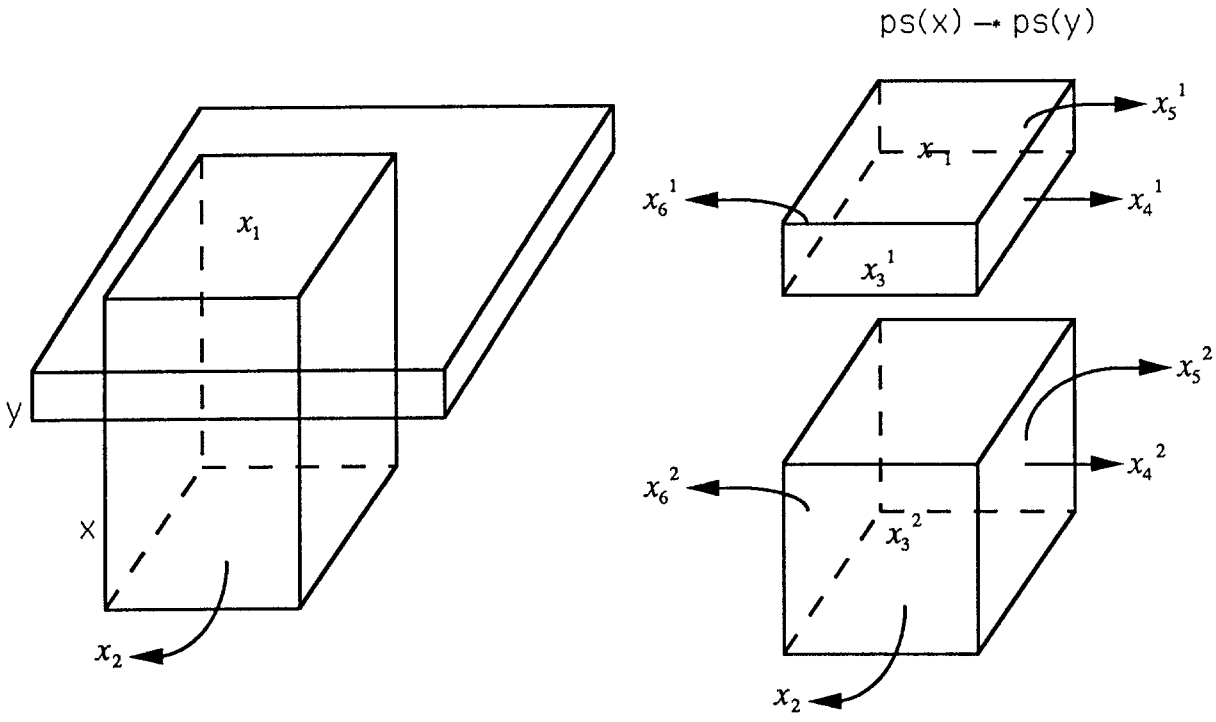


Figure 12. Case where ps(x) → ps(y) is two rectangular solids.

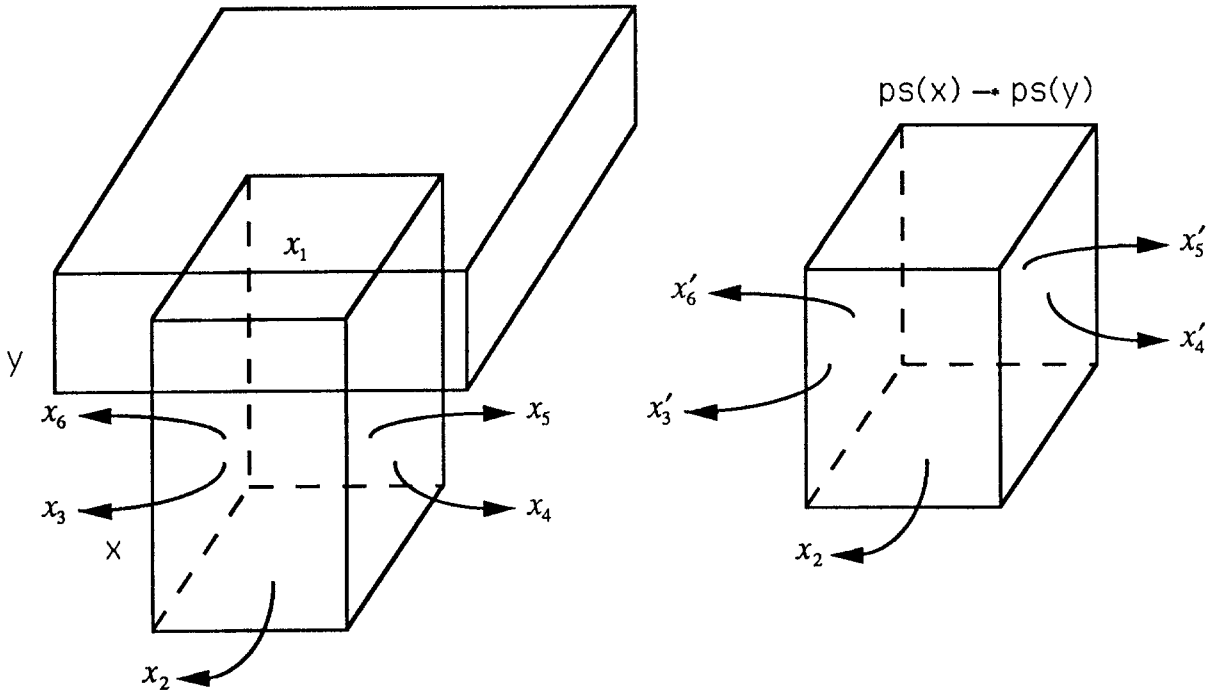


Figure 13. Case where ps(x) → ps(y) is a single reactangular solid.

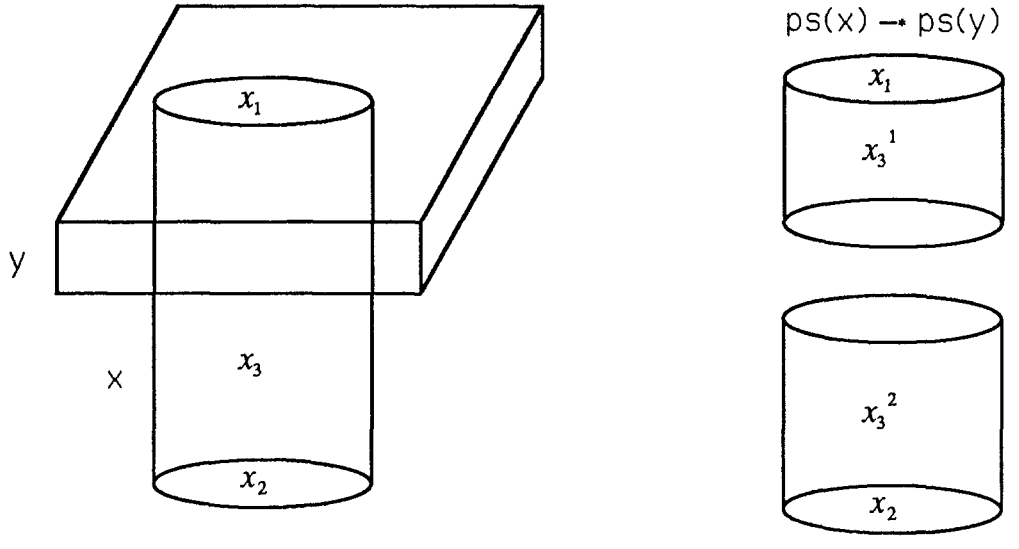


Figure 14. Case where ps(x) → ps(y) is two cylinders.

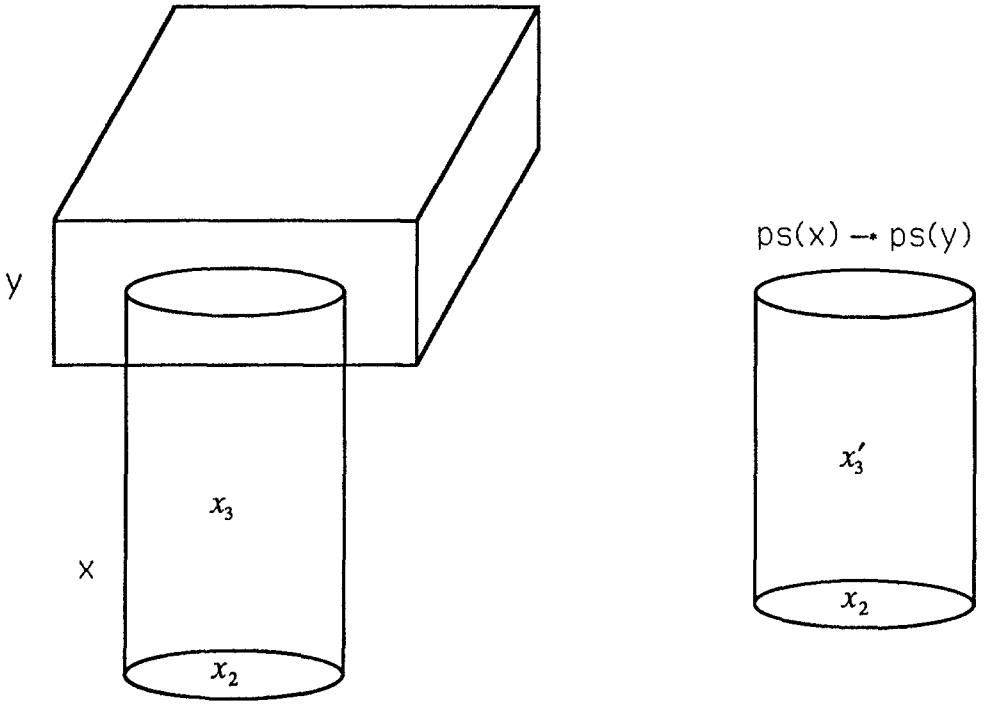


Figure 15. Case where ps(x) → ps(y) is a single cylinder.

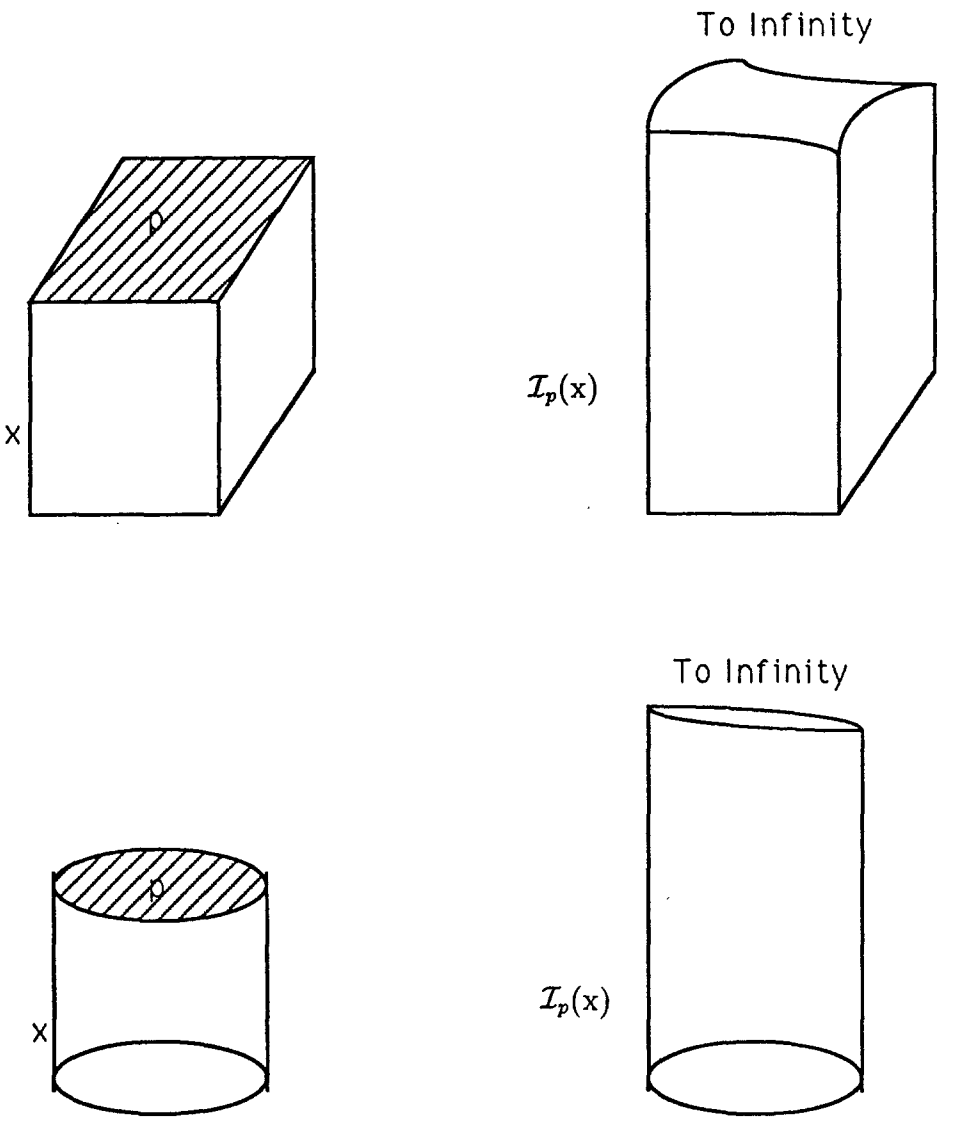


Figure 16. Interesting shapes for infinite extension for rectangular solids and cylinders.

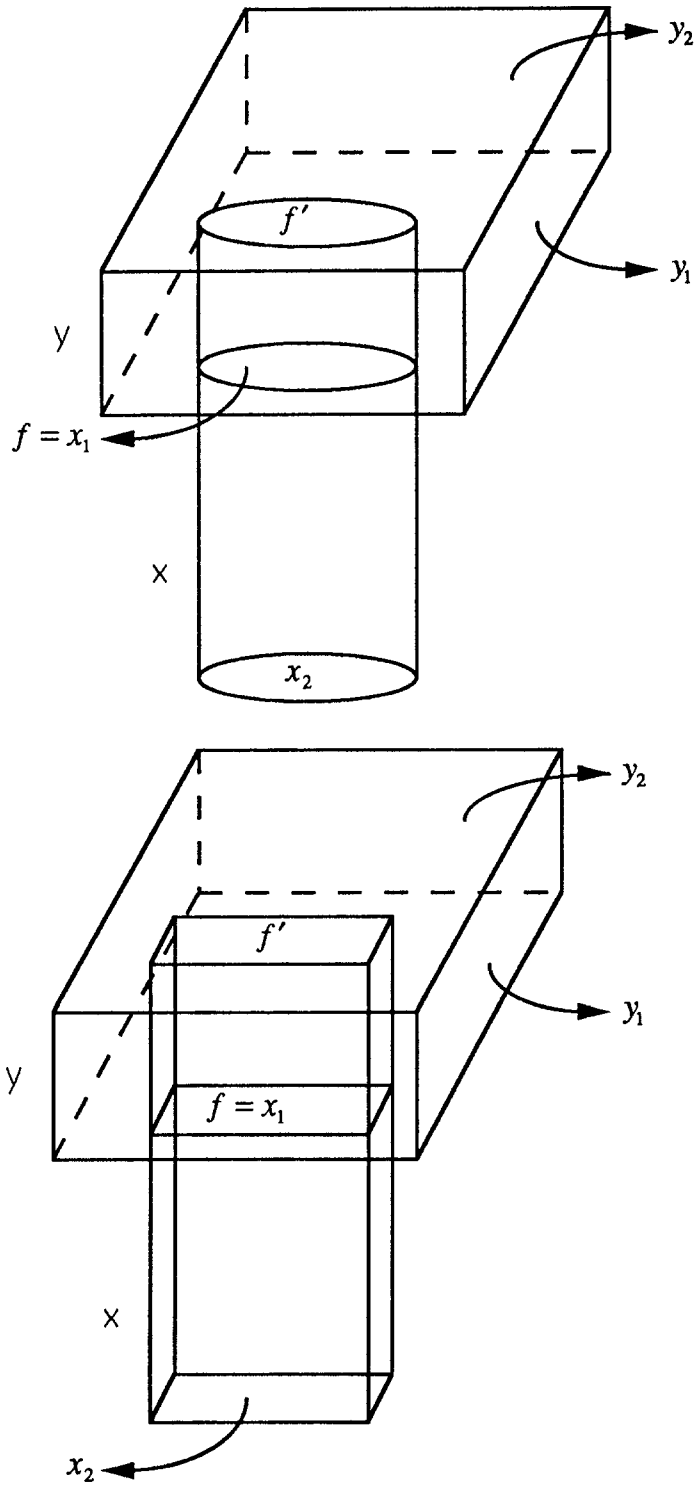


Figure 17. Computing Maximal Extension.