Using a Feature Algebra for Reasoning about Geometric Feature Interactions*

Raghu R. Karinthi
Department of Computer Science
University of Maryland
College Park, MD 20742 U.S.A
Email: raghu@brillig.umd.edu

Dana S. Nau
Department of Computer Science,
Institute for Advanced Computer Studies,
and Systems Research Center
University of Maryland
College Park, MD 20742 U.S.A
Email:@nau@*mi*msy.umd.edu

Abstract

(AD generated objects can be defined in terms of the complete geometry of the part. In automated process planning, the geometry is the same, but the meaning associated with this geometric structure is different, and dictates a change in the description. Various solutions have been proposed to solve this incompatibility, including automated feature extraction, design by features, and human-supervised feature extraction.

An important problem to be addressed here is handling feature interactions. Due to geometric interactions among the machinable features, there may be several equally valid feature interpret ations for the same part. This paper describes an automatic way of computing alternate feature interpretations, using an algebra of feature interactions. This will enable automated process planning systems to decide whether various interpretations of a part as a collection of machinable features are feasible for manufacturing, and among the feasible ones, which is the most appropriate one for manufacturing.

Alternate feature interpretations are computed by performing operations in the algebra. Furthermore, various provable properties oi the feature algebra aid in resolving a majority of the feat ure interactions without even applying the operations in the algebra, thereby improving the efficiency of the algebraic approach.

1 Introduction

Machining a metal part consists of taking a piece of stock and performing various *processes* or operations on it to transform it into the desired part. Each machining process acts on a piece of stock and changes its structure in a unique way. Civen the ('AD description of the object, one would like to derive how it can he manufactured or machined (since we talk only of machining processes here). This means arriving at a sequence of processes which when applied to the stock would result in the final machinable part. Notice that such a sequence is not unique. Traditionally the designer comes up with the CAD description of the object to be machined, and the process engineer and the NC programmer produce the precise process plan.

There has been a lot of progress in deriving process plans from manufacturable features and we have several working *(jtnnativt* process planners such as XCUT [Brooks et *al* , 1987] and SIPS [Nan, 1987]. Several approaches have been proposed for bridging the gap between Computer Aided Design (CAD) and Computer Aided Manufacturing (('AM) and automatic feature extraction, design by features, and human supervised feature extraction are among the prominent ones. One of the limitations of these approaches is of multiple feature interpretations, which is discussed in the next paragraph.

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

To illustrate the role of feature interactions, let us consider the part described 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. Because of the interaction of h_1 with s_1 and s_2, we get $h_2 = h_1 -^* s_1$, $h_3 = h_1 -^* s_2$, and $h_4 = h_2 -^* s_2 = h_3 -^* s_1$. The final set of features used for machining would be s_1, s_2 and one of h_1, h_2, h_3 and h_4. Which hole to use depends on issues such as

*This work has been supported in part by NSF Presidential Young Investigator award I)('R-83-51463, with matching funds provided by Texas Instruments and General Motors Research Laboratories, and by NSF Engineering Research ('enters Program Grant. NSFD CDK-88003012 to the I'nivrrsily of Maryland Systems Research ('ent.er.

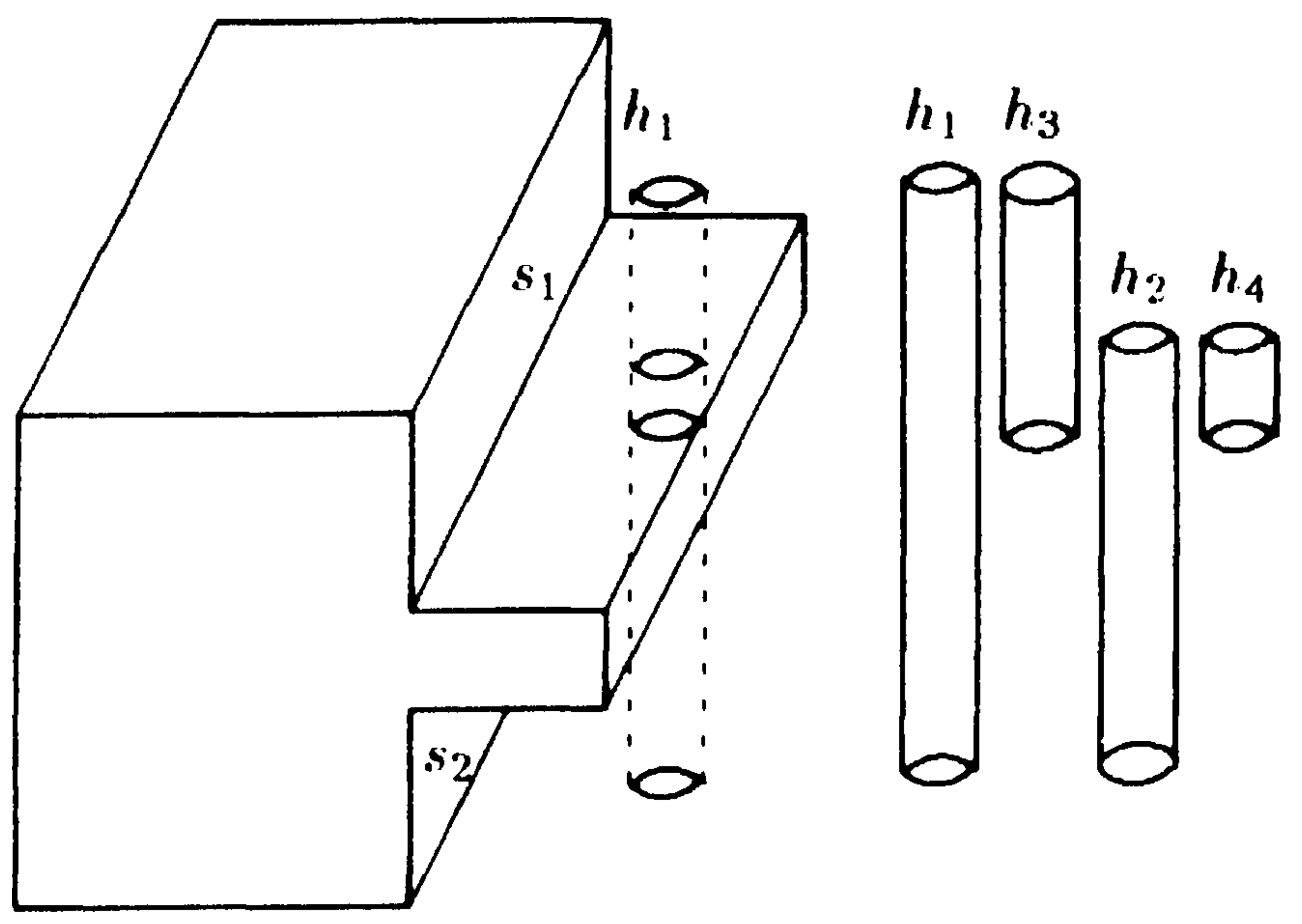

Figure 1:

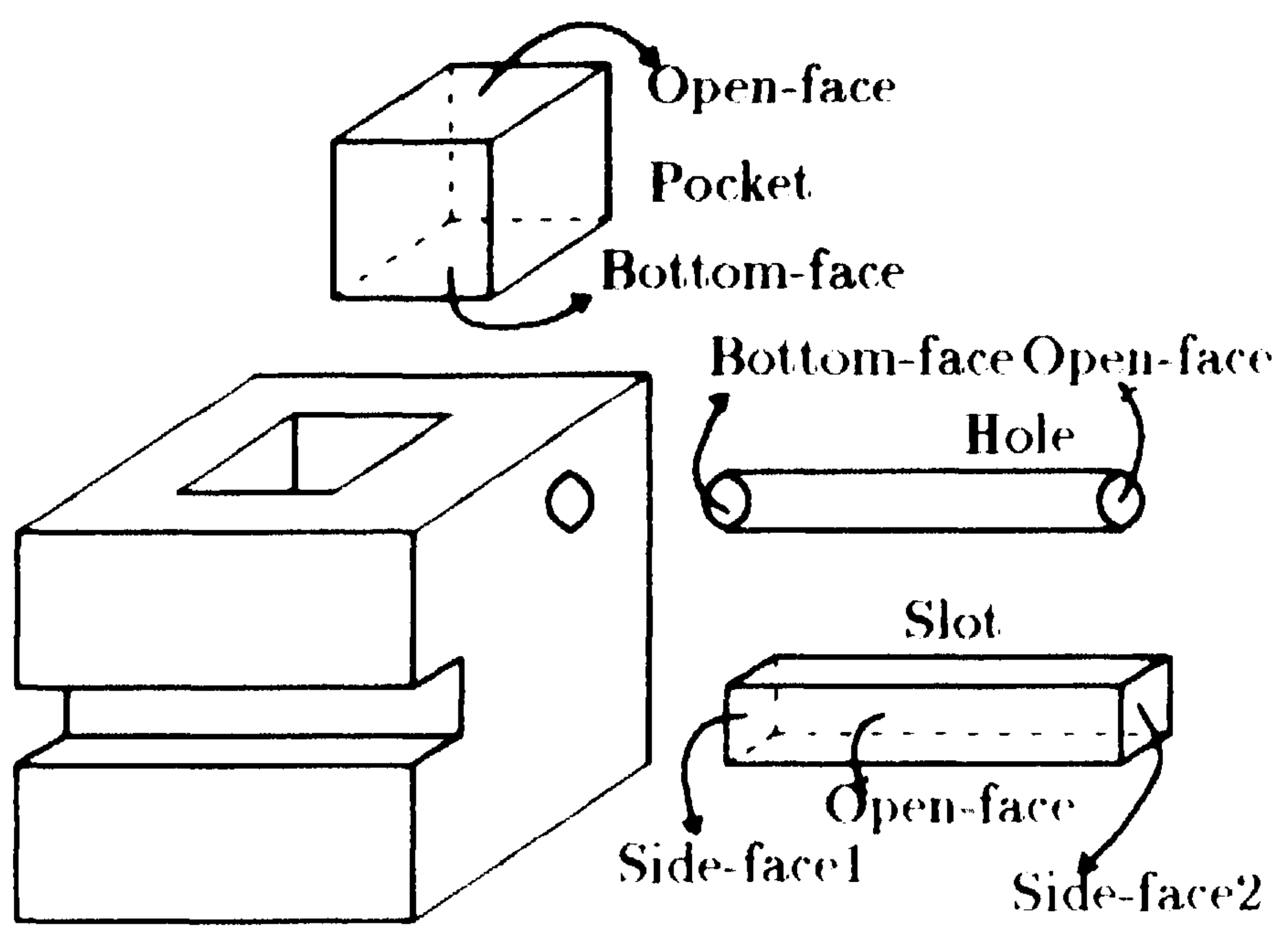

Figure 2:

cost criteria (whether it is cheaper to make a deep hole or a small hole), feasibility (availability of proper tools to make a deep hole), fixturing criteria (can the part be properly fixtured to make h_4 after S_1 and $.s_2$ have been made, or will it vibrate). Thus one can see that there can be several possible interpretations 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 an' provided by means of operations in this algebra. This scheme provides several alternative feature representations of a machinable part, given one such represent at ion.

The rest of the paper is organized as follows: In Section 2 we describe the feature algebra. In Section 3 we compare our work with related research in this area. Section 4 is a discussion of the issues addressed by this research. In Section 5 present the current status of the research and the work we plan to do in the future.

2 The Algebra of Features

In this section we discuss the domain of features we work with, the operations on features, some properties of the feature algebra and an algorithm that produces multiple feature interpretations of a part, given one such representation.

An algebraic structure [Pinter, 1982] in its simplest form is any 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 $< a, b >$ of elements of *A* exactly one element. $a * b$ in A.

2.1 Feature Definitions

For the purposes of this paper, the only features considered are a rectangular pocket, a slot and a hole. Further, in this paper we restrict the features to have the normals of their planar faces, to be parallel to the base vectors of our co-ordinate system. In addition to the features, there is a special object called the *slock* on which all the features are made.

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. Requieha and Voelcker[Hequicha and Voelcker, 1985)I have shown that this difficultv can be overcome I>v 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.

Let us denote the initial stock by S_0. Let *J\, f-j, . . f„ be* the features subtracted out of N_n, in that order. The part, after / features have been defined is denoted by *S,.*

$$S_i = S_0 -^* \bigcup_{j=1}^{i} f_j$$

and

$$S_n = S_0 -^* \bigcup_{j=1}^{n} f_j$$

A face *j-* of a feature /, is said to be *opt n,* if J* belongs to the boundary of $S_v \cup^* f$, where S_b denotes the final part. A pocket is a rectangular cavity with at least one of its faces open. A slot is a rectangular cavity with at least one of its faces and two faces orthogonal to it also open. These two faces must be parallel to each other. A hole is simply a cylindrical hole. Figure 2 shows a part with a pocket, a slot and a hole.

Every feature must have at least one of its faces in common with of a face of the stock or some other feature. These two parameters are called the *containing fact* and the *containing featurt.* In other words, the con taining feature is the feature having the containing face. At the time of defining any feature the containing face must belong to the boundary of the part at that step. It cannot be interior to the part or outside the part. This condition is more restrictive than the ones given in the previous paragraph, and hence supercedes them.

Further, for slots, two faces orthogonal to the containing face are required to be part of the part boundary at that step. As mentioned earlier, these two faces must be parallel to each other.

1. The initial stock is specified by its location in space and the dimensions along the x, *y* and *z* axes
2. A pocket is specified by four parameters, viz. the containing face, the containing feature, the *open fare* and the depth. The containing face and the containing feature specify the face and the feature in which the pocket is contained. The open face of the pocket is a set of four points in the containing face specifying the face of the pocket that is open and the direction of the normal pointing into the feature. We will call the face of the pocket parallel to the open face as the *bottom fact.* The depth parameter gives the dimension normal to the open face.
3. A slot has the same parameters as a pocket. The two open faces of the slot that are orthogonal to the open face of the slot are called the *side faces.*
4. A hole is specified by the containing face, the containing feature, the open face and the depth. The containing face, the containing feature and the depth parameters are analogous to those of a pocket. The open face parameter in case of a hole is specified by the center and the diameter of the circle that constitutes the open face of the hole and the direction of the normal pointing into the feature. As in the case of a pocket, the open face of a hole must lie on the containing face. The face parallel to the open face is known as the *bottom fan.*

2.2 Operations

As the operations on features are defined, one will notice that the operations give meaningful values only for certain pairs of features, thereby causing a conflict with the definition of an operation. This problem is solved by adding a special element called *INVALID* to the set of features. Further, by definition, for any operation *,

$$\forall x, x * INVALID = INVALID * x = INVALID.$$

Let D be the domain consisting of all the features and the element *INVALID*. We will now define the operations *shortened* (denoted by $\mathcal{S}$), *first maximal-extension* (denoted by $\mathcal{M}_1$) and *second maximal-extension* (denoted by $\mathcal{M}_2$) and a *bisection* function $\mathcal{B} : D \rightarrow 2^D$. These are based on regularized boolean operations on solids. The mathematical definitions of $\mathcal{S}$, $\mathcal{B}$, $\mathcal{M}_1$ and $\mathcal{M}_2$ are based on the characterization of features as *semi-algebraic* sets. The definitions and the procedures for determining the operations are described in [Karinthi and Nau, 1989]

Given two features x and y, $x\mathcal{S}y$ and $x\mathcal{B}y$ are computed from $x -^* y$, provided $x -^* y$ can be regarded as a feature or a pair of features of the same type as x. If $x -^* y$ is a single feature, then $x -^* y = x\mathcal{S}y$. If $x -^* y$ is a pair of features, then $x -^* y = x\mathcal{B}y$. For example, in Figure 1, $h_1\mathcal{S}s_1 = h_2$. The first maximal-extension of x in y, denoted as $x\mathcal{M}_1 y$ and the second maximal-extension of x in y denoted by $x\mathcal{M}_2 y$ are determined from $\mathcal{I}_1(x) \cap^* (x \cup^* y)$ and $\mathcal{I}_2(x) \cap^* (x \cup^* y)$, subject to certain conditions. Informally, $\mathcal{I}_1(x)$ and $\mathcal{I}_2(x)$] are two ways of extending a feature till the boundaries of the stock. Pockets and holes are extended only in the direction normal to their open and bottom faces. This is done using the $\mathcal{I}_1$ function. On the other hand, slots can be extended both in the direction normal to their open and bottom faces as well as in the direction normal to their side faces. Thus, for any slot r both $\mathcal{I}_1(x)$ and $\mathcal{I}_2(x)$ have meaningful values. Referring back to the example in Figure 1, $\mathcal{I}_1(h_1) = \mathcal{I}_1(h_2) = h_1$ and $h_2\mathcal{M}_1 s_1 = h_1$

2.3 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 the same feature has already been generated. Fsing the alg"hraic properties is a significant saving particularly in this domain, because the boolean operations on solids are quite expensive as compared to symbolic comparisons.

Several properties regarding the features and their interactions have been proved. Some properties of the features are stated below. The interested reader is referred to [Karinthi and Nan, 1989] for more details on the properties and the proofs.

Proposition 1 *For any three features x, y and z, if $(x -^* y)$ and $(x -^* z)$ are also features, then*

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

Proposition 2 *If x, y, z and w are features such that x is a hole, $z = x\mathcal{S}y$, $\mathcal{I}_1(x) = x$ and $w \cap^* y = \emptyset$, then the following result holds:*

$$z\mathcal{M}_1 w = z$$

Proposition 3 *If x and y are features and if $\mathcal{I}_1(x) = x$, then*

$$x\mathcal{M}_1 y = x.$$

2.4 The Features Algorithm

In this section, an algorithm which generates various possible feature sets corresponding to a machinable part given one set of features corresponding to the part is described. In describing the algorithm and an example subsequently, we will use the notation $\emptyset$ to denote the empty set. In this algorithm, the variable F contains the set of feature sets generated at any stage, while F has the union of all the features in F. Before the algorithm is executed, F is initialized to the starting set of features and F is initialized to be a set with only one element, viz. the set F. There are also two additional variables called $CURRENT$ and NEW. Before entering the algorithm, $CURRENT$ and NEW are initialized to the sets F and $\emptyset$, respectively.

```
while $CURRENT \neq \emptyset$ do
    for each feature set do
        for each operator applicable to a pair of
            features in the feature set atleast
            one of which is in $CURRENT$ do
        if the resulting feature(s) have NOT
            already been generated
            then begin
                add the feature to F
                add the feature to NEW
            end
            if the feature set is distinct
                then add the feature set(s) to F.
        end for
    end for
    CURRENT= NEW
    NEW = ∅
end while
```

At first glance, the worst case performance of the algorithm might appear to be of exponential complexity, 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.

2.5 Illustrative Example

In this section, we will illustrate the working of the algebra of features (and the features algorithm) through an example.

This example is the same as the example one discussed in Figure 1. The starting set. of features are h_1, S_1 and s_2. We compute, $\mathcal{I}_1(h_1)$, $\mathcal{I}_1(s_1)$, $\mathcal{I}_1(s_2)$, $\mathcal{I}_2(h_1)$, $\mathcal{I}_2(s_1)$ and $\mathcal{I}_2(s_2)$. Therefore, we have,

$$\mathcal{I}_1(h_1) = h_1 \quad (1)$$
$$\mathcal{I}_1(s_1) = s_1 \quad (2)$$
$$\mathcal{I}_1(s_2) = s_2 \quad (3)$$
$$\mathcal{I}_2(h_1) = INVALID \quad (4)$$
$$\mathcal{I}_2(s_1) = s_3 \;\; (\textit{not shown in Figure } 1) \quad (5)$$
$$\mathcal{I}_2(s_2) = s_4 \;\; (\textit{not shown in Figure } 1) \quad (6)$$

In this section, we will study the example. by look ing only at the changes that occur from one iteration to the next, without examining all the interactions (like t he ones which result in already existing features). Obviously, all the interactions need to be considered because apriori, one does not know, which interactions result in new features and which do not. At the start of the features algorithm the values of the variables are:

$$F = \{h_1, s_1, s_2\}$$
$$CURRENT = \{h_1, s_1, s_2\}$$
$$NEW = \emptyset$$
$$\bar{F} = \{\{h_1, s_1, s_2\}\}$$

Let us examine F and $\bar{F}$ each time before the outermost loop is entered. The value of NEW is each time before the outermost loop is entered. After the first iteration :

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

In this iteration, h2 and /13 were generated using the S operation, where /12 = *h1Ss1* and h3 = *h1Ss2*- Furthermore, using Equation 1, and and Proposition 3 we conclude that $h_1 M_1 S_1$ — *h\,* $h_1 M_1 S2$ = *h\.* Other interactions are considered similarly.

After the second iteration :

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

In this iteration, let us say h4 was generated as h_4 = h_2Ss2. But notice that h_3Ss1 was not generated as a new feature, say h_5, This is because, from Proposition 1 h_3Ss_1 — h_2Ss_2 — $h4$, which has already been generated Similarly, while considering h2M1s2 using Proposition 2 we infer *h24\s<>* — *h2.*

After the third iteration :

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

At this point t h e l o o p terminates, because[1] *(7 ■■ RRF N J* = ∅ For this example, it can be shown that summed over the three iterations, a total of 54 possible feature interactions are considered. Of these interact ions, only three result in new features (h_2, h_3 and h_4).

3 Related Work

There has been considerable research in automatic feature extraction and feat ure-based design over the last ten years. But as mentioned earlier, even though this work is pertinent from the point of view of CAD/C'AM integration, very little of it has actually addressed feature interactions. The remainder of the section describes some of the recent work that addresses feature interactions.

Nicholas Ide [ble, 1987] at the University of Mary land developed a feature based design system integrating the SIPS process planning system [Nan, 1987] with the PADb 2 solid modeler. In Ide s system, the designer starts out with a representation of a piece of stock, and designs his part by subtracting various machinable features from the stock. Ide s system points out certain geometric interactions that make a feature impossible to machine. In this system, the kinds of feature interaclions that, can lead to multiple feature interpretations were either not considered or not allowed (for example, a hole was not allowed to intersect any other feature).

Maeda and ShinoharaJMaeda and Shinohara, 1988] have developed an rule-based system called KSPKR (Kxpert System for Product ion KngineeRs) that does certain kinds of geometric reasoning to account for feature interactions. The rules in KSPKR can be used not only to recognize features in geometry, but to manipulate the geometry directly. 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[Hayes, 1987], has developed a rule-based system called *Machinist* that detects feature interactions using rules obtained from human experts. This system is written in OPS5. The system determines time-ordering relations among features by applying the rules. Thus, if certain conditions are satisfied, a rule would tell us that one feature must be machined before another.

Requicha and Vandenbrande [Requicha and Vanden brande, 1988] 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 has geometric reasoning and feature recognition modules combined with a solid modeling and computational geometry modules. This system was not completed as of the above citation, so we do not know of its capabilities in detail, but it appears to us that their approach of combining solid modeling with geo metric reasoning is one of the most promising directions for future research in CAD/CAM integration

4 Discussion

A methodology for addressing feature interactions using an algebra of features has been described. We believe the following are the significant aspects of the algebraic approach:

1. One of the primary purposes of the feature algebra is to allow consideration of several possible feature alternatives. 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 alternatives and generate plans to see which is better (or leasible). The popular approaches addressing the CAD/CAM integration problem (as described in Section 1) have ignored the issue of multiple feature interpretations.

2. The kinds of reasoning done in the FSPFR [Maeda and Shinohara, 1988] and Machinist [Hayes, 1987] systems represent significant steps in the development of wavs to. handle feature interactions. How ever, problems arise in trying to extend these systems into complete computer-aided manufacturing systems, because the representation of solid objects is not sufficient to resolve ambiguities, lor 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 differ eut dimensions than if/; were machined after *s* and vet such information is necessarv in order to know whether it is possible to machine //. In the feature algebra described in this paper, a feat tire is an unambiguous and complete representation of a physi cal solid. Thus, the information missing from Machinist's rules could be restored by rewriting these rules in terms of operations in the feature algebra.

/\. The operations in the algebra are efficient as com pared to boolean operations on solids, because, they take advantage of the nature of the features (simple 2-D shapes) and the nature of the interactions (orthogonal 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 savings in computations.

5 Current Status and Future Work

This work constitutes a portion of a larger project whose goal is to develop an integrated system for design and process planning [Nau *et ai* , 1988]. Other components of this project include the Protosolid solid modeler [Vanecek and Nau, 1988] and the SIPS [Nau and Cray, 1986, Nau, 1987] process planning system. Protosolid and SIPS are written in Lisp and run on a Texas Instruments Explorer-FI. Having developed the feature algebra, our next step will be to use it as the basis of a feature analysis and translation system to facilitate communication between Protosolid and SFPS. For example, this will allow SIPS to decide dynamically whether various interpretations of a part as a collection of manufacturing features are feasible for manufacturing, and among the feasible ones, which is the most appropriate one for manufacturing.

Currently, a theoretical framework for the algebra is complete, and work is beginning on the feature analysis and translation system. Future work on the algebra includes incorporating a wider set of features and interactions than what we have now. In particular, we would like to extend it to model interactions among features that are not at a right angle to the faces of the stock or other features. This requires new operations and functions to be incorporated into the algebra. We also intend to extend the algebra to model the rounded corners of pockets and slots, and tapered holes.

References

[Brooks et a/. , 1987] ST. Brooks, k E. Hummel, and ML. Wolf. XCUT: a rule-based expert system for the automatic process planning of machined piece parts. In *Symposium on Int(grattd and Intelhgent Manufacturing. ASME Winter Annual Mating,* Dec 1987.

[Hayes, 1987] C. Hayes. (sing goal interactions to guide planning. In *Proaectdrugs of the AAAI-87; the Suth Sational Conft r< na on Artjicial Intelligi net.* pages 221 228, 1987.

[ide, 1987] N. C. Ide. Integration of Process Planning and Solid Modeling through Design by Features. Mas ters thesis, I'niversity of* Maryland, College Park, 1987.

[Karinthi and Nau, 1989] H. Karinthi and I). S. Nau. 1989. Paper in preparation.

[Maeda and Shinohara, 1988] Y. Maeda and K. Shinohara. Ceometric reasoning and organized optimization for automated process planning. In *Proacdings of the Siventh Sat tonal Confer* na on Artificial In-Itlligcna,* pages 101) 110, Aug 1988.

[Nau, 1987] D. S. Nau. Automated process planning using hierarchical abstraction, *liras Jnstrumi nts 1\ clinical Journal,* 39 4<>, Winter 1987 Award Winner, Texas Instruments 1987 Call for papers on Industrial Automation.

[Nau and Gray, 1986] D. S. Nau and M. Gray. Sips: an application of hierarchical knowledge clustering to process planning. In *Proceedings of the Winter Annual Meeting of ASME, Anaheim, CA,* pages 219 225, 1986.

[Nau *et ai* , 1988] I). 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.

[Pinter, 1982] C. Pinter. *A Book of Abstract Algebra.* McGrawHill Book Company, 1982.

[Requicha and Voelcker, 1985] 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

[Requicha and Vandenbrande, 1988] A. G. Requictia and Jan H. Vandenbrande. Form Features for Mechanical Design and Manufacturing. Technical Report IRIS 244, Institute for Robotics and Intelligent Systems. University of Southern California, Los Angeles, October 1988.

[Vanecek and Nau, 1988] G. Vanecek Jr. and I). S. Nau. A General Method for Performing Set Operations on Polyhedra. Technical Report CS TR 2057, University of Maryland, College Park, 1988.