Comput. & Graphics Vol 14, No 2, pp 149-172, 1990
Printed in Great Britain

0097-8493/90 $3 00 + .00

ISSUES ON FEATURE-BASED EDITING AND INTERROGATION OF SOLID MODELS

JAROSLAW R. ROSSIGNAC
IBM, Research Division, T. J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598

Abstract — Operations that create additive or subtractive volume features, such as bosses or slots, simplify the computer aided design of mechanical parts. Surface features, whether extracted automatically or selected interactively, group functionally related boundary elements, and thus provide an expedient interface between CAD systems and analysis or manufacturing applications. Despite much progress in CAD, design remains an iterative process and involves error-prone modifications of previous solutions. Features should in principle offer a high level vocabulary for characterizing errors and for specifying how they should be corrected. This paper points out the semantic ambiguities of simplistic feature-based commands for editing models. It recommends procedural models for editing volume features, and corrective volumes for editing surface features. It shows how space decomposition techniques and CSG expressions based on active zones reduce the cost of executing an editing command. Error detection may be automated by supporting intentional features, which correspond to the desired characteristics of the model, and by endowing them with domain dependent validity criteria expressed in terms of associated geometric elements. The paper demonstrates that validity may be tested by simply interrogating a mixed-dimensional geometric structure which is used to represent not only the model, but also the interactions between the geometric elements associated with intentional features.

1. INTRODUCTION

Solid modelling improves the efficiency of the design process for manufactured parts by supporting the geometric representations of these parts. It provides graphic feedback to the designer and offers interfaces to some analysis applications. Despite recent Progress in computer hardware and in interactive graphics, geometric design of three-dimensional shapes remains a complex and time consuming task. Since geometric representations of complex mechanical parts tend to be verbose, various abstractions, globally called *geometric features*, are often used to characterize certain types of shapes or to refer to Portions of a part model that may be important to the designer or to an application program.

Features are used in Computer Aided Design and Manufacturing for a variety of purposes:

- Geometric features provide a concise description of the parts characteristics[1]and thus simplify group technology and process planning. They also facilitate the communication between designers and solid modelling systems.
- Features provide a mechanism for attaching product or manufacturing information and various attributes to specific parts of a geometric model (see for example [2] for attaching tolerance information to features).
- The properties intuitively associated with common feature types define many convenient shape altering operations[3] that attempt to create features of specified types and dimensions.
- Access to the geometric elements that compose a feature simplifies the interrogation of shape by providing a naming scheme for sets of boundary elements [2], a convenient vocabulary for expressing relevant dimensions and positions [4] and for formulating validity checks that assess the compliance of the model with the designer's intent.
- Expressing and performing engineering changes or simply corrections of design errors may be eased by using geometric features [5]

This paper focuses on the last two issues, namely the use of geometric features to automate — or at least simplify — the editing of the solid model and the checking of the model's compliance with functional requirements (validity conditions).

Most CAD models of 3D manufactured parts are created by combining and incrementally modifying simple models. These combinations and modifications are often conveniently expressed in terms of Boolean operations. The primitive shapes from which the models are constructed are often restricted to arbitrarily positioned and sized solid primitives (blocks, cylinders, spheres ...), generic volume features (holes, slots, bosses ...), and linear or circular extrusions of 2D regions. The resulting part models can thus be represented by a CSG (Constructive Solid Geometry) tree [6], which leads to certain algorithmic advantages (see [7-9] for examples) and to an obvious archival conciseness.

CSG expressions may be complex, and the end-user often prefers to interact with a boundary model, which is algorithmically derived from CSG [10] and contains the list of faces and their adjacency graph [11]. Therefore, it is important to develop techniques for interactively specifying validity conditions and modifications in terms of boundary elements (faces, edges, and their incidence graphs) rather than in terms of CSG. Domain-dependent features provide a particularly convenient vocabulary for accessing relevant groups of boundary elements. On the other hand, direct boundary editing is error-prone, and editing operations, even if specified in terms of boundary information,

should be translated into mathematically well-defined (nonambiguous) operations, such as Boolean set operations or global rounding and filleting operations[12]. Besides, it is important to maintain a CSG representation of the model and to express validity conditions in terms of CSG so that the model can be parameterized, easily edited, and reused.

This paper studies the translation process, which takes validity conditions or model modifications expressed in terms of features (and thus of boundary elements) and performs the appropriate model modifications using CSG operations. Due to a lack of formalism of the semantics of feature-based specification, automatic translation remains a challenging research goal (some pitfalls of a "naive" translation process are pointed out in Section 3). Several new or recently developed techniques are discussed, which do not always provide the correct translation, but at least increase the designer's vocabulary or can automatically generate a tentative solution, which may have to be further adjusted by the designer. Furthermore, the paper addresses the issue of efficiently performing the model modifications or the validity tests by using informationally rich geometric structures and properties of Boolean expressions.

The paper is organized as follows:

- The basic concepts and terminology are introduced in Section 2.
- Section 3 points out some of the limitations of a straightforward use of features for editing and interrogating solid models. This section focuses on concepts and techniques rather than on their historical evolution or implementation, (For a more formal survey of the literature on features, the reader should refer to [13, 14].)
- The importance of procedural models for capturing the designer's intent into a flexible parameterized sequence is emphasized in Section 4. To edit a feature explicitly created by an operation, it may be simpler to "ask" the feature what operation created it, change the parameters of that operation, and reexecute the entire sequence.
- Reexecuting the entire sequence amounts to evaluating the boundary of a CSG representation, and may be very costly for large CSG models. A new approach that reduces the reevaluation cost is presented in Section 5. It derives a CSG expression for the regions that must be added to, or subtracted from, the solid model. Furthermore, Section 5 also presents a recently developed mixed-dimensional geometric representation called SGC (Selective Geometric Complex). Algorithms for SGCs generate a subdivision of space imposed by the features. This subdivision can be used to reduce considerably the amount of geometric calculations and of logic expression evaluations necessary to perform the feature-editing operations.
- Some features do not correspond to a single operation, and it may be too complicated to identify all the operations in the sequence that must be edited in order to rectify an "invalid" feature. Section 6 demonstrates on some simple examples how corrective volumes, obtained by extruding feature faces, can be used to perform simple feature alterations without reexecuting the design sequence. In more complicated situations, these corrective volumes must be trimmed before they can be added to—or subtracted from—the model of the part. Without the trimming step, side-effects may appear, especially when several features interact or when compound features incrementally created by successive operations, are edited. Trimming is best performed using Boolean operations, but producing a timming CSG expression may prove difficult and remains the designer's responsibility.
- Section 7 addresses the problem of feature validity. Specifically, it shows how features may be efficiently tested by interrogating the corresponding *SGC* representation.

2. BASIC CONCEPTS AND TERMINOLOGY

This section clarifies the distinction between intentional features and their geometric embodiment, and between volume features and surface features. It also introduces the CSG notation used in this paper.

2.1. *Intentional features and their geometric embodiment*

A distinction should be made between *geometric features* and *intentional features*. A geometric feature is a collection of geometric elements (for example, faces or volumes) that form a subset of the part's interior, boundary, and/or complement. An intentional feature [15] is an abstraction for accessing groups of geometric elements and for associating with them a type and consequently certain properties defined for all the features of this particular type. For example, an intentional feature of type *slot* may be associated with a part model. This association indicates that the designer intends to have a slot in the model, *i.e.*, a void bounded on three sides by faces of the model. The intentional feature contains references to the corresponding faces. However, due to model manipulations, the referred faces may have been modified or even deleted from the model's boundary. Whatever remains of them and of the associated void constitutes a geometric feature that may no longer exhibit the properties associated with a *slot*,

Inconsistencies between intentional features and the actual geometry of the part are avoided by treating intentional features only as hints and by relating them to geometric elements through collections of unevaluated references. It is acceptable that some, or all, of these references do not correspond to any geometric element of the model's boundary at some particular stage during the design process. Even if all geometric elements referenced in an intentional feature are present in a geometric model, their shapes and positions with respect to the rest of the model need not comply with the characteristics usually associated with the particular feature type. For example, an intentional

feature of type cylindrical hole could be associated with geometric elements (faces) that have been removed from the model's boundary by some Boolean operation, and thus do not correspond to a "valid" hole. In such situations, the intentional feature is said to be invalid, but should not be discarded, because the designer may have produced (intentionally or not) temporary situations, or instances of the model, where many previously defined intentional features are invalid, The overall validity may later be restored by repositioning a subsolid or adjusting a parameter. The designer should not be required to redefine all intentional features that went through an invalid transition stage.

Furthermore, feature validity is very subjective and in fact depends on the role the feature plays with respect to a particular application. To take a simplified example, a cylindrical hole is a valid "detail feature" to be discarded for analysis purposes only if its radius is sufficiently small; on the other hand, it is a valid "manufacturing feature" for process planning only if it is empty and accessible.Feature validity criteria may be expressed in terms of validity rules, which are logical predicates defined in terms of the referenced geometric elements and of their existence, shape, and relation to other geometric elements of the model. Evaluating the model's geometric references is therefore necessary to establish the validity of an intentional feature with respect to any one of the instances (or stages) through which a solid model evolves during the design process. Consequently, the interrogation of invalid features plays an essential role in correcting design errors [4]. Typically, the presence of an intentional feature of a certain type, valid or not, implies some intention that the designer has regarding the functionality of some portion of the part. Thus, intentional features may provide important hints for model alterations and manufacturing process planning.

2.2. *Constructive Solid Geometry (CSG)*

CSG (Constructive Solid Geometry)[6] refers to an unevaluated representation scheme for solids obtained by combining, in Boolean expressions, simple primitive shapes of arbitrary dimensions and positions. Solids specified in that way may conveniently be represented by a binary tree whose leaves correspond to primitive shapes, whose internal nodes correspond to Boolean operations and represent subvolumes, and whose root represents the final solid. Often, the primitive shapes are expressed as the intersection of closed half-spaces. Commonly used half-spaces (planar, cylindrical, spherical) are mathematically defined as the set of points for which the value of a simple linear or quadratic function is negative or null. For practical implementation reasons, solid models are often restricted to be r-sets (a subclass of closed, bounded three-dimensional sets with no dangling boundary elements and with a finite number of faces and edges) [16]. They are often represented in terms of their boundary, i.e., a list of their faces (in turn defined in term of their bounding edges) often structured in an adjacency graph [11]. To guarantee that results of Boolean operations are r-sets, a regularized version of these operations is used. It performs the standard operations and then removes the dangling and interior faces and edges and makes sure that a valid boundary is part of the pointset. Theoretically, this cleaning operation amounts to taking the topological interior of the pointsets produced by the Boolean operation (this eliminates the exterior dangling faces, edges, and vertices), and then the topological closure of the result (which amounts to putting a tight boundary around the pointset). These transformations are illustrated in Figure 1. In practice, the faces and edges of the model are classified using neighborhoods[10].Only the elements that play the appropriate role in the boundary of the solid are kept. Throughout this paper, all Boolean operations are regularized, unless explicitly specified otherwise.

The regularized Boolean operations will be denoted $\cup$ for the union, $\cap$ for the intersection, - for the difference, and $\oplus$ for the symmetric difference. Futhermore, the regularized complement of any solid X will be denoted $\tilde{X}$. For simplicity of notation, it is assumed that the Boolean operators in Boolean expression are ranked by decreasing priority as follows: complement, intersection, difference, symmetric difference, and finally union.

Throughout this paper it is assumed that the part models, or solids, are created by a sequence of operations that add or subtract material or move and combine subsolids through Boolean operations. Therefore, a CSG representation of such a model always exists, even though the explicit construction and use of the CSG tree may be avoided in certain cases.

2.3. *Volume and surface features*

An important distinction pointed out in[13] separates surfacefeatures, which are collections of faces of

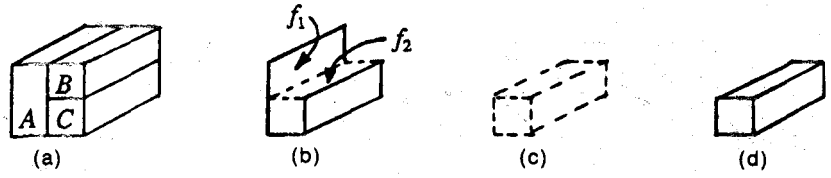

Fig. 1. Regularization: Three blocks, A, B, and C, shown in (a) are combined through a nonregularized Boolean expression $(A \cap B) \cup (C - B)$ to produce the pointset shown in (b), which has a dangling face f_1 and a missing face f_2. A regularized version of the pointset can be obtained by taking its interior, which is an open set depicted in (c) that does not contain any of its faces, and then taking the closure of the result and thus adding to the model all its faces, shown in (d).

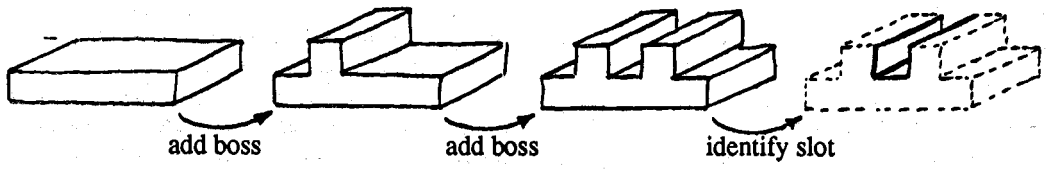

Fig. 2. Surface feature: Adding two bosses (volume features) to the part (left) creates a slot (geometric feature) that can be recognized by the user and interactively associated with an intentional surface feature for later references.

a part model[17–19], such as the walls and the floor of a slot, from volume features, which are full-dimensional pointsets of the part or of its complement, such as bosses and holes [20,21]. Very rarely are both types simultaneously supported in the same modelling system (as they are in the prototype system MAMOUR [4]). Pratt [13] provides a detailed discussion of the historical motivations, current merits, and drawbacks of both types of features and concludes that all surface features should be converted to volume features, which, although slightly more complex to support, offer greater flexibility for interactive editing and more information for driving analysis and application programs. The author believes that both volume and surface features are useful for editing and that a volume representation of a surface feature need not always be derived.

2.3.1. *Volume features.* Design is often done in an incremental manner, by first laying out the overall shape, and then adding or modifying details by creating or editing features. The creation of a geometric feature is necessarily accompanied by a modification of the volume occupied by the part and io practice always corresponds to either an addition or a subtraction of material. This transformation may be expressed as the union or difference between the part and the volume feature. Because the volume feature may be viewed as a sophisticated parameterized primitive shape, this approach is particularly effective in dual modellers which derive a boundary representation from a CSG tree.

The computational expense of explicitly deriving the effect of a Boolean operation [10] has discouraged certain developers of solid modellers from evaluating the boundary of the part obtained by subtracting or adding a volume feature. Instead, implicit features (also called unevaluated) have been recommended [19].

A boundary representation of the feature is directly derived from the designer's specification without checking if this representation is geometrically correct. For example, an implicit feature of type slot may have been defined by mistake as hanging in air, away from the part, or buried inside the part and not accessible from any side. This incompatibility problem does not occur when intentional features are used instead of implicit features because intentional features, although unevaluated, carry no assumption as to their corresponding geometry embodiment.

2.3.2. *Surface features.* The volume features resulting from shape modifying operations do not always provide a sufficient set of abstraction tools for interacting with the model. For example, a slot feature of interest for manufacturing applications may have been created as a side effect of adding two parallel boss features (Fig. 2). The slot may provide a convenient abstraction for expressing engineering changes (which, for example, modify its width) and thus should be made accessible to the designer through an intentional feature. The use of such a posteriori identified features requires the association of intentional features with a subset of an existing geometry. Often such association is done by interactively selecting a collection of faces of the part model and treating it as a surface feature.

Information provided by surface features may be sufficient for some applications, such as planning for surface finishing operations or as specifying and analyzing dimensions and tolerances [2]. Other applications, such as assembly or manufacturing planning, heavily rely on the manipulation of volume features [22]. Except for simple cases, the derivation of a volume feature that corresponds to a surface feature remains an open issue [13], because there is no unique mapping from surface features to volume features. Typically a selected set of closing faces is added to a surface feature in order to produce a valid two-dimensional shell that unambiguously defines a volume (Fig. 3). Desirable, or even correct, closing faces may not always be obtained by extending existing adjacent faces (Fig. 4). Methods or heuristics for automatically con-

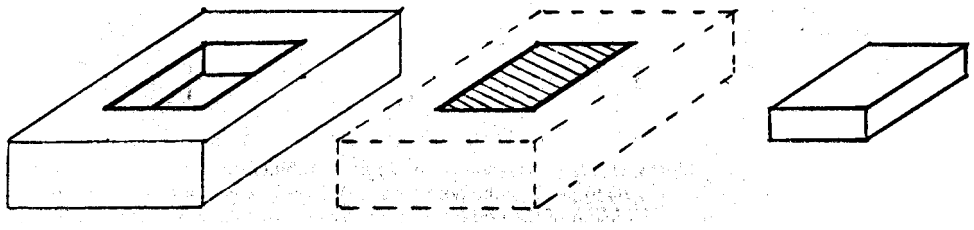

Fig. 3. The volume of a surface feature: By adding a closing face (center) to the faces of a surface feature (left), one obtains a valid boundary of a corresponding volume feature (right).

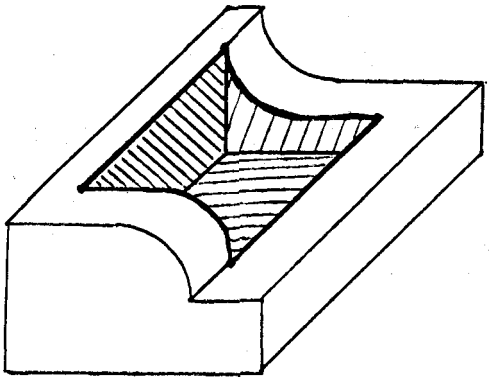

Fig. 4. Complex closing faces: No simple set of closing faces exists for the pocket surface feature.

structing such faces are currently limited to simple situations, and the designer's intervention may sometimes be required to generate acceptable solutions.

2.4. *Compound features*

Several volume or surface features may overlap. For example, two orthogonal slots (volume features) may have a common intersection volume (Fig. 5).

Similarly, a boss and the adjacent slot may share a common vertical wall (Fig. 6). Furthermore, interior features, such as a boss on the floor of a slot (Fig. 7), may be used as modifiers of other features. It is not always necessary to capture such feature interactions explicitly in the data structure. For example, a geometric element (face or volume) may be shared by several intentional features that are independently used by different applications. On the other hand, a hierarchical organization of intentional features may be useful to represent explicitly compound features (Fig. 8) and also patterns of features (Fig. 9) when such situations reflect the designer's intentions or are important for applications such as process planning. The nature of the geometric and topological interaction between the individual features of a compound feature should be derived, when needed, from the actual geometry of the faces referenced by the individual features.

3. *Pitfalls*

Because they provide an intuitive, domain dependent, high-level vocabulary, both volume and surface features are good candidates for facilitating the specification of shape modifying operations and the expression of validity Conditions, provided that one can make the specification convenient and unambiguous.

For the designer's convenience, these specifications have to be unambiguous, so that the effect of shape editing commands can be clearly understood and easily predictable, and the validity rules must precisely characterize invalid situations independently of the verification procedures employed. They also must be convenient, so that the specifications correspond to powerful high-level operations that produce the desired effect and so that validity rules are simple to formulate and powerful enough to trap common design errors. Furthermore, procedures for executing the shape modifying commands and for evaluating validity rules must be available.

This paper shows how extensions of several techniques may be integrated to improve the specification and the execution of unambiguous shape modifications using compound or isolated volume or surface features. It also shows how a rich geometric representation scheme can be used to simplify the expression and evaluation of validity rules. Most of these techniques have been made possible by recent developments in geometric modelling, which must now be integrated. These developments will be briefly summarized, and their potential applications to feature-based editing of solid models will be demonstrated.

3.1. *Limitations of simple shape modifying techniques*

To stress the need for the approaches such as those proposed in this paper, this section discusses the limitations of several simple schemes that come to mind as possible ways of using features to modify and test solid models.

3.1.1. *Implicit features.* Formerly mentioned implicit features may be trivially edited by modifying their parameters. For example, an implicit slot can be moved and enlarged by changing its position and its width. However, as pointed out earlier, the existence of an implicit feature with specified dimensions and position does not guarantee that the corresponding geometric feature with the expected characteristics is to be found on the part. Thus, to produce a reliable description of a part, implicit features should be treated as intentional features, and the corresponding geometric features should be constructed (if possible) and their validity (i.e., compliance with functional requirements) assessed.

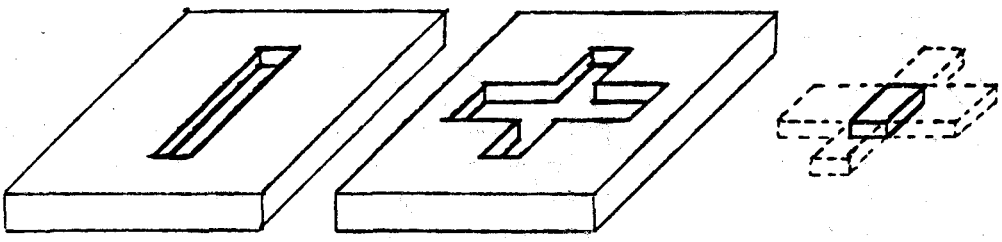

Fig. 5. Two interfering features: Subtracting a slot (volume feature) from the model (left) that already has a slot feature creates a model (center) in which the volumes of the two features interfere (right).

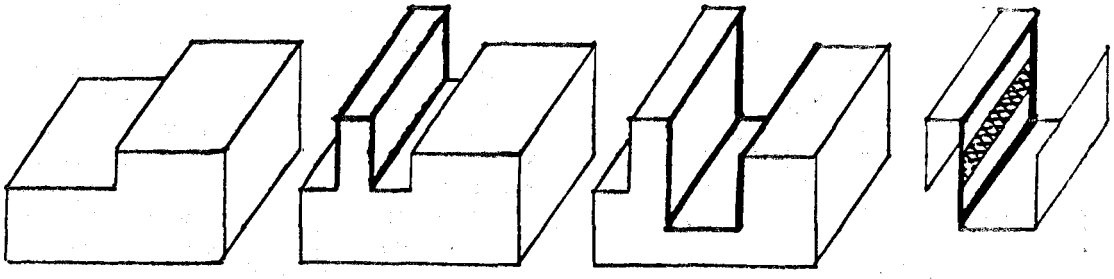

Fig. 6. Shared face: Adding a boss (volume feature) to the model (left) and then subtracting a slot feature (center) creates a model in which the boundaries of two adjacent features overlap along a portion of a face (right).

3.1.2. Procedural models. To further automate the generation of models and allow the designer to concentrate on high-level design decisions, it is suitable to support the automatic derivation of CAD models from a set of functional constraints specified by the designer. The functional requirements specifying the geometric characteristics of intentional features could be considered as constraints and combined with the geometric constraints describing the part. A constraint solving system would converge to a valid solution, if such a solution exists, or declare that the specification (i.e., set of constraints) is invalid. Such a scheme would have the considerable advantage of supporting incomplete specifications of features. For example, an intentional feature of type slot could be defined and its dimensions specified, but its position would not be provided by the designer, except for one constraint: The slot should be abutting on a given face of the object.

Such an automated approach has been given a serious consideration [23,24], but it carries a high computational cost and requires that designers provide a complete set of consistent constraints before the CAD system can create a model and return some useful feedback. Deriving and maintaining such systems of constraints, especially when additive (bosses) and subtractive (pockets, slots, holes) features interfere is a considerable endeavor, as pointed out in [25]. Furthermore, a simple indication that the set of constraints is incompatible does not provide useful hints as to what part of the specification should be modified to produce the correct result. A more practical approach is to build models incrementally by transforming and combining simpler models. A procedural rather than declarative approach, in which the designers specify a sequence of operations that transform a model in attempt to satisfy constraints, has been described by Rossignac in [26]. It relies on the designer's ability to decompose the problem into an ordered set of subproblems that can be solved one at a time. The procedural specification (i.e., the sequence of operations that solve the individual problems) is saved and can be edited by the designer and reexecuted on demand. This technique could be used for feature-based editing by considering the implicit features as intentional features to be created in a predefined order. Reexecuting the specification would attempt to create the geometric counterpart of the intentional features at specified positions and would report whether the creation was successful (i.e., whether valid features have been produced).

Considering that a procedural model can be obtained simply by storing the designer's commands into a log file and making the file available for modification and reexecution fails to address three important problems:

1. Features successfully created during an execution of the procedural model can be invalidated by the subsequent creation of other features. Therefore, to assess the validity of a design, intentional features created at an early stage of the specification must be preserved and methods for accessing the corresponding geometric elements (whenever they exist) and for testing the compliance of these elements with feature validity rules should be available.
2. Feature parameters may be defined in terms of other features. For example, a boss may be intended to lie at the center of the floor of a rectangular pocket. Although this relation may have been established at the creation of the boss, subsequent editing of

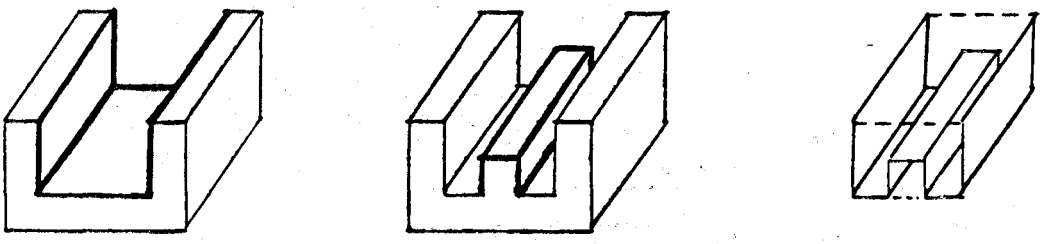

Fig. 7. Nested features: Adding a boss feature in the middle of a slot feature (left) creates a model (center) with a nested feature (right): The boss is inside the volume of the slot.

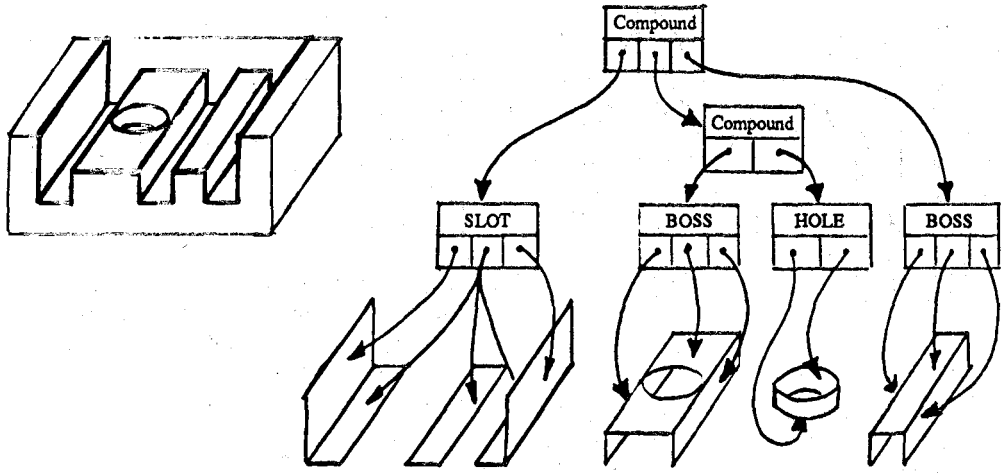

Fig. 8. Compound features: A hierarchical organization of features may be used to represent a compound feature made of a slot that contains two bosses, one of which has a hole (left). These relations can be captured by creating additional intentional features of type compound that refer to other features instead of referring to geometric elements (right).

the procedural model may alter the position of the pocket. To preserve the relation between the positions of the two features, this relation must be captured in the procedural model and used to position the boss at each execution.

3. Surface features are typically selected by the user in an interactive mode, preferably using a graphic cursor to pick the appropriate geometric elements from a picture of the model on a computer screen. Surface features in a model may be used to attach tolerances or other manufacturing attributes to particular portions of the model and to serve as clues for process planning [5], or simply to provide constraints on the position and size of volume features to be created later by the procedural model. It is therefore essential that surface features, once selected by the designer on one instance of a model, be selected automatically when a new instance of the model is created. To support automatic reselection, operations that select surface features must be captured in the procedural model in such a manner that their execution produces the desired result, even when earlier parts of the procedural model are modified.

Techniques for supporting procedural models together with intentional surface features and for capturing relations between such features have been developed and implemented by Borrel, Nackman, and the author, and are described in [4]. They will be briefly reviewed in Section 4 and their applications to feature-based editing will be discussed.

3.1.3. *Local boundary modifications.* If no procedural model is available for editing and reexecuting, or if the execution of the procedural model is costly, the part model may sometimes have to be directly edited, or "patched." Since a valid solid model is unambiguously described by its boundary, boundary "tweaking" seems an attractive technique for editing. For example, the four vertices of the floor of the slot in Fig. 10 could be raised to change the depth of the slot. A new boundary representation would be readily available if the floor and the adjacent faces were implicitly defined by their vertices. However, such boundary tweaking techniques may require major alterations of the boundary structure of the object. If polyhedral modellers, if the geometry of the edges and faces of the model are implicitly represented in terms of vertex coordinates, a face (for example, the floor of

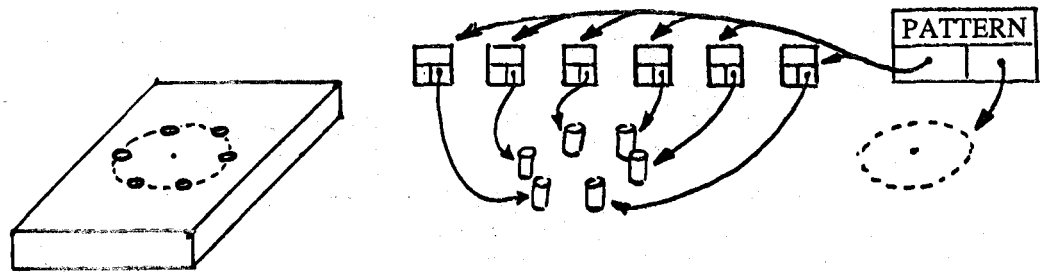

Fig. 9. Patterns of features: A pattern of hole features (left) may be represented by a compound feature referencing the intentional features of each hole (center). The compound feature has a description of the pattern parameters (right) and can be used to access and interrogate the entire pattern.

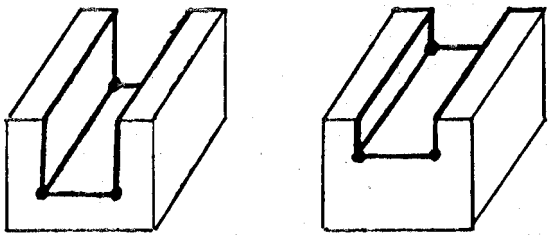

Fig. 10. Boundary tweaking: The floor of a slot (left) may be raised by moving up its vertices (right).

a pocket) may be moved simply by displacing its vertices. However, the vertices of each face must remain coplanar. Coplanarity is guaranteed by forcing the vertices of the face being moved to remain coplanar and to move along extensions of the edges they bound.

The situation is more complicated when vertices correspond to intersections of curved surfaces. Suppose that the designer wishes to move only one face of the feature and wants to express this modification in terms of vertex displacement. Then vertices are constrained to move along extensions of abutting (curved) edges while simultaneously remaining on the edited surface as it evolves. The number of degrees of freedom, which specifies how many of the vertices may be moved independently, is dictated by the nature of that surface and its acceptable deformations (scaling, rotations ...).

In addition, the validity domain, which specifies the maximum amount of vertex displacement along each edge so that the boundary structure (adjacency graph between faces) remains constant, is defined by the geometries and relative positions of the faces. Restricting the displacement of each vertex along the extension of an edge so that it does not exceed the intersection of this extension with other faces is not sufficient to guarantee that the resulting boundary will not be self-intersecting (as is the case in Fig. 11).

An alternate approach is to modify the boundary structure or to alter several faces simultaneously. A simple example may be found in Fig. 11, where a vertical left wall is added to connect the lowered floor to the abutting cylindrical face, but in general it is difficult to devise procedures for specifying these modifications. Furthermore, detecting self-intersecting boundaries is computationally difficult, and no unambiguous and useful semantics has been defined for specifying how self-intersecting boundary representations should be corrected.

In conclusion, the semantics of boundary tweaking operations is not well defined and the results may be invalid. This paper thus proposes techniques that use mathematically well-defined set theoretic operations to alter features.

A possible approach would be to convert incorrect surface features into volume features (by the insertion of closing faces) and then delete these volume features and create new correct ones. As pointed out earlier, the set of closing faces is not unique for any surface feature, and even a single suitable set may be hard to produce. To overcome the "closing face problem," a technique based on corrective volumes, previously employed by Requicha and the author for local blending operations[12], will be proposed in Section 6. Here, it uses a generalization of sweeps or extrusions to create solids that can be added to—or subtracted from—the part in order to change the dimensions or positions of geometric features,

3.1.4. *Order dependency of volumetric alterations*
A volume feature, whether directly created by a Boolean operation or derived by closing a surface feature, could in principle be modified by deleting it and, if necessary, by creating a new volume feature with the desired characteristics (Fig. 12).

Deletion of a volume feature may be obtained by using the inverse of the Boolean operation that could have been used to create it. For example, a subtractive feature of type slot could have been created by subtracting the corresponding slot volume from some previous representation of the part. Therefore, adding the volume back, using a Boolean union, would delete the feature. Unfortunately, this approach suffers from three major problems, which could be qualified as "undesirable side effects."

1. As demonstrated in Fig. 13, the lack of associativity properties of set theoretic Boolean operations do not in general permit to undo the effect of subtracting (respectively adding) a feature by adding (respectively subtracting) it back. Specifically: $(A - B) \cup B \neq A$, unless B C A, and similarly $(A \cup B) - B \neq A$, unless $B \subset \bar{A}$.

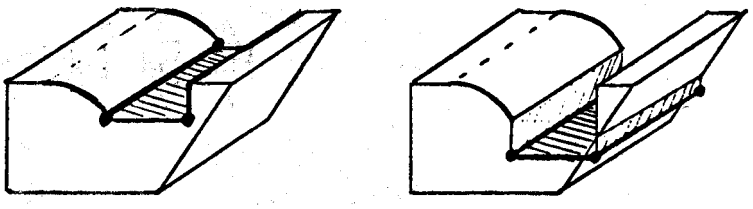

Fig. 11. Validity problems: The depth of the slot (left) is altered by lowering its floor face. In the resulting model, the boundary is disconnected and self-intersecting. To connect the boundary on the right side of the slot, the right wall can naturally be extended to follow the vertices. On the left side, however, the wall is cylindrical and the lowered vertices do not lie on its extension. They either have to be moved horizontally, or a new vertical face must be created. In any case, the boundary is self-intersecting on the right side of the slot, and therefore portions of it must be eliminated, which involves geometric calculations.

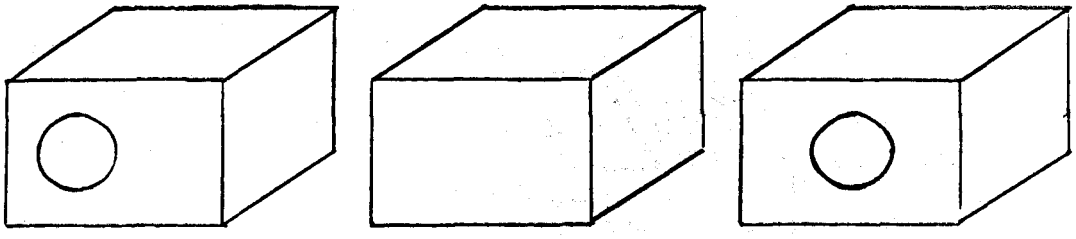

Fig. 12. Editing by deletion and creation: A hole volume feature (left) may be moved by deleting it (center) and by creating a new hole in the right position (right).

2. When additive and subtractive features interfere, the order in which they have been created is important and in general cannot be altered without producing undesirable side effects, such as the one illustrated in Fig. 14.
3. Modifying features by deleting them and creating new ones could modify or destroy other features (see Fig. 15).

For solids created by combining simpler solids and volume features through Boolean operations, correct results may be obtained, without reevaluating the whole Boolean expression, by confining the deletion of the old feature and the creation of the new correct one to the active zone of the old feature. Active zones have been introduced by Voelcker and the author in [27], and would correspond here to the portion of space where the shape of the feature is important. A formal definition and relevant properties of active zones will be summarized in Section 5, and new properties for applications to feature-based editing will be derived.

Instead of computing the active portions of all features (i.e., their intersection with their active zones), space decomposition techniques [28–30] may be used to precompute and store the geometry of the active portions of all features simultaneously, and thus to improve the performance of algorithms that execute the editing operations. Then, each editing operation may be confined to the appropriate regions without repeating expensive geometric calculations each time. Such an approach has been proposed by Pratt [13] using an extension of the radial edge structure developed by Weiler [28]. Section 5 briefly presents a different, more general, and slightly simpler structure called Selective Geometric Complex (abbreviated SGC) described by O'Connor and the author in [29]. Algorithms for subdividing SGCs may be used to decompose space into cells of dimension three or less, such that given any cell C and any volume (or even surface) feature F, C either lies entirely in the interior of F, entirely in its complement, or entirely in the boundary of F (when C is a face and F is a volume feature). Each cell of an SGC "remembers" what features it belongs to and is associated with an attribute which defines whether the cell is part of the represented solid or not. Fig. 16 illustrates such a decomposition. The applications of the SGC structure to the deletion and modification of features within their active zones will be demonstrated.

3.2. *Limitations of simple interrogation techniques*

Detecting whether the geometry associated with an intentional feature satisfies the validity requirements explicitly associated with that particular feature-type is extremely convenient for validating a design and automatically verifying that changes produced by adding new features or modifying old ones did not create undesirable side effects. However, the validity of each individual feature may not be sufficient to assess the validity of a complex part, and sometimes, a relation between several features is also important. Most of these validity requirements may be explicitly attached as rules to single or compound features [4], and automatic procedures for checking feature validity may be invoked, as described in Section 7.

Some validity rules may be characterized in terms of topological relations between volume features and can be expressed in terms of Boolean operations between a volume feature and the part it is supposed to modify. For instance, when a slot B is to be subtracted from a part model A (Fig. 17 center), one can detect situations where the slot is too deep and where its floor

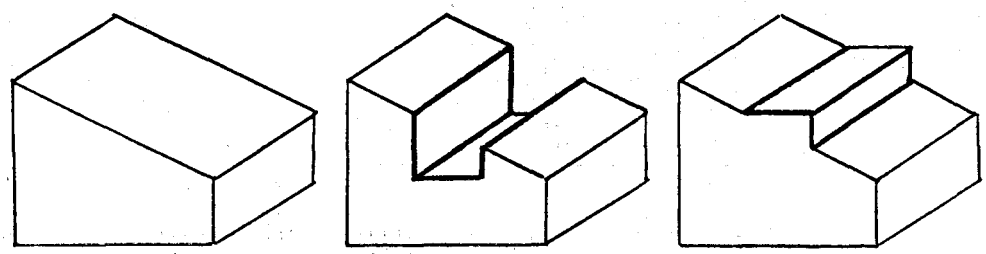

Fig. 13. Side effects: In the slanted top face of the model (left), a slot is created (center) by subtracting a block. Adding the block back (right) does not restore the original model.

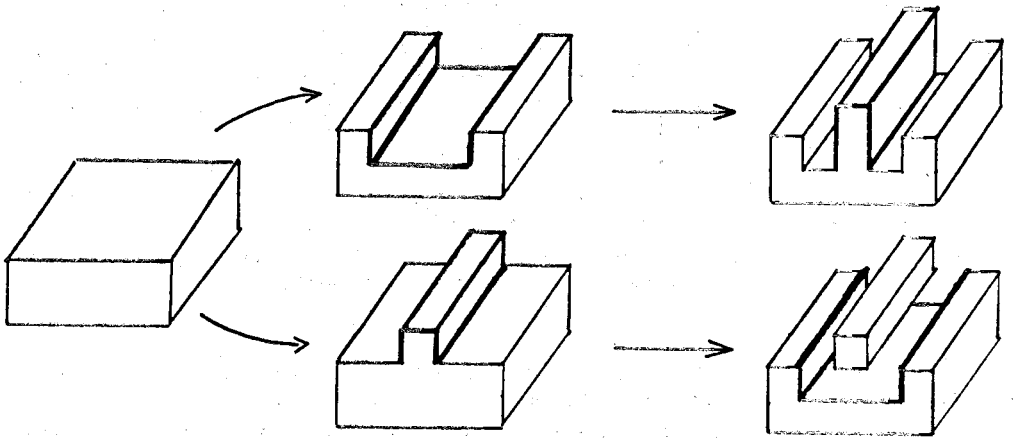

Fig. 14. Order dependency: Modifying a solid (left) by first creating a slot (top center) and then a boss (top right) produces a different result than first making the boss (bottom center) and then the slot (bottom right).

is missing in the boundary $A - B$ (Fig. 17 right) by checking whether $B - A$ represents the empty set or not (this works only if the floor is not coincident with the bottom face of A).

When regularized Boolean operations are used, such simple tests are insufficient to distinguish between significantly different situations. For example, a distinction between the correct situation and the unacceptable configuration in Fig. 18 may not be obtained as previously suggested by considering $B - A$, because B − A is empty in both cases. Clearly, in most cases, a more complex test involving Boolean combinations of auxiliary solids may be provided, but each situation and each test may require different types of auxiliary solids. For example, a "roof," thin block B' over B may be used: $B' \cap A$ is empty† in the correct situation of Fig. 19, but is not empty in the invalid case of the same figure. These tests are expensive to perform and may require constructing and intersecting complicated auxiliary volumes. Furthermore, this approach is limited to volume features. An alternate approach is proposed in Section 7 which demonstrates that most common validity criteria may be tested by querying the existence of lower-dimensional parts in an SGC representation of the space decomposition defined in terms of features.

†Testing whether a Boolean combination of two or more solids is empty requires evaluating its boundary or performing a Null Object Detection test on a CSG representation[31, 27].

4. PROCEDURAL MODELS

As pointed out in the introduction, procedural models are particularly convenient for trial-and-error geometric design, because they capture the designer's specifications in a form that can be easily understood, edited, and repeatedly executed. This section briefly presents a prototype system for procedural modelling, called MAMOUR, which has been implemented in the object-oriented language AML/X [32] and described in more details in [4, 5]. MAMOUR supports intentional features with validity rules and can keep track of relations between different features.

4.1. *Sequences*

Such a system offers an adequate platform for integrating the techniques presented in this paper. As the user of MAMOUR creates and transforms a part model, the system automatically constructs a procedural model composed of an ordered set of operations that measure geometric (or other) properties, create or move primitive or subsolids, define intentional features, or perform Boolean operations between objects. The

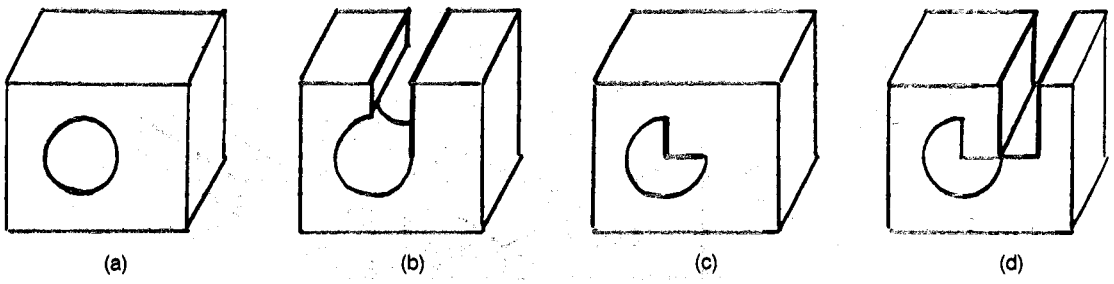

Fig. 15. Unintentional feature destruction: In a model with a through-hole (a), a vertical slot (b) is created in a wrong position. Deleting the slot by a union creates a solid (c) in which the through-hole feature is partly destroyed. Making a new slot in the right position (d) produces a valid slot, but leaves the through-hole invalid.

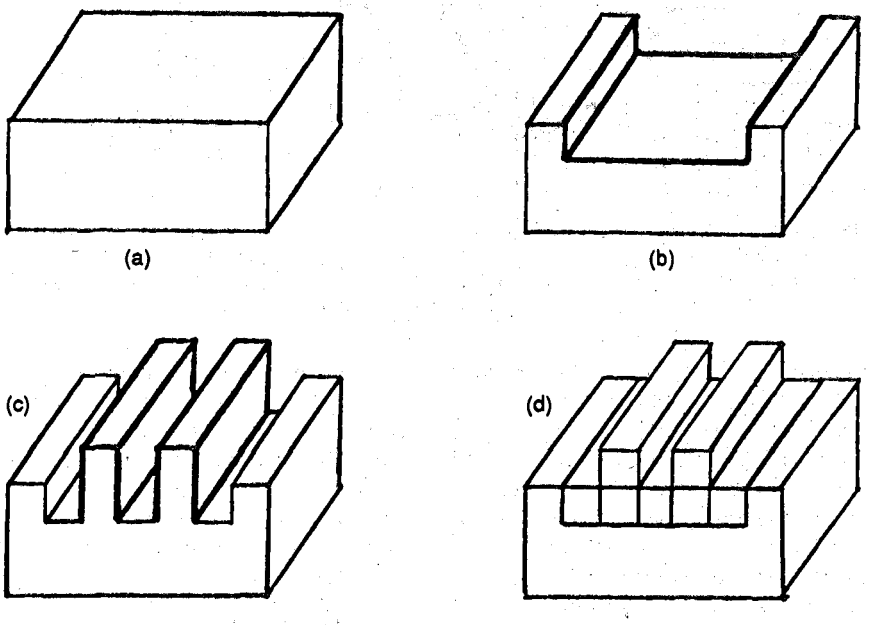

Fig. 16. Space decomposition: In the model (a) a slot is made (b). Then two bosses are created inside the slot (c). Space is decomposed into 3D cells (d), so that each cell is either inside or outside the original model and any one of the features.

execution of the procedural model constructs a geometric model and the associated set of intentional features. The procedural model can be edited by the user without interrupting the design session, because it is stored in memory as an aggregate (list) of AML/X objects.‡Each object corresponds to an operation, or design step. Fig. 20 top illustrates one such sequence. Each operation has its own internal variables and execution methods. A whole sequence of operations may be captured in another AML/X object of type SEQUENCE and can be edited, executed, and included as a parameterized macro operation into other sequences.

4.2. *Unevaluated parameter expressions*

To provide a greater flexibility and multiple applications of the same sequence, each operation and therefore each sequence is parameterized. When several sequences are pieced together in different ways or when an early part of a sequence is edited, an individual operation may be executed on a model different from the model that has been used to specify that operation. For example, a make-rib operation may have been used to make a rib in the center of the boss in the original model. The parameters locating the rib were derived from the position of the boss. If operations that construct or modify the boss have been edited so that the shape of the boss is changed, the execution of the make-rib operation will produce a rib that is no longer in the center of a boss, unless make-rib can adapt its execution to the new geometry. This adaptability is possible in MAMOUR, because parameters of operations may be stored in an unevaluated form, thus capturing the designer's intents (for example, to position the rib in the center of the boss). To simplify the formulation of this constraint, an intentional feature is associated with the boss. Intentional features in MAMOUR are also AML/X objects with internal variables and methods. The internal variables typically contain sets of unevaluated references to boundary elements of the model (for ex-

‡ An object is an entity that has a type (for example, "Drill operation"), a set of internal variables (for example, expressions that define the radius and position of the hole), and a set of procedures, called methods (for example, the one that modifies a CSG representation of a part by subtracting a cylinder of the appropriate radius).

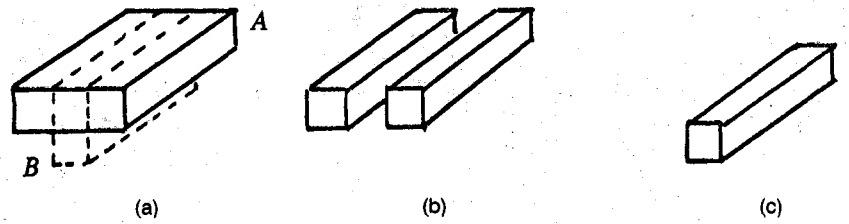

Fig. 17. Validity test: The position of a volume feature B in relation to the part A is shown in (a). Subtracting B from A fails to produce the desired slot feature (b) because the depth of B is too large. Sometimes, such situations may be detected by testing if the difference $B - A$ is an empty solid or not (c).

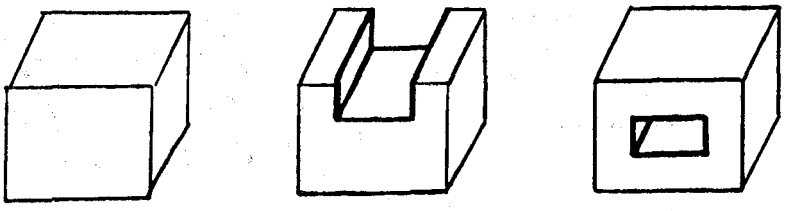

Fig. 18. Inadequate validity test: Subtracting from a part A (left) a correctly positioned slot volume feature B produces a valid geometric feature of type slot (center). Subtracting an ill-positioned slot B creates an invalid geometric feature (right). The difference may not be detected by considering a regularized Boolean combination of A and B.

ample, references to the faces of the boss). The methods can be used to evaluate these references and obtain a geometric description of the appropriate elements (for instance, of the floor face of the boss). These geometric elements are also AML/X objects with methods for computing their geometric properties (for example, the center of a face). Thus, if boss1 is the name of the intentional feature that corresponds to the geometric feature of type boss, the position of the rib may have been specified in make-rib using boss1.floor().center(), which defines the correct position, as long as the geometry referenced in boss1 is a valid boss, or at least has a floor face.

Until now, MAMOUR has been interfaced with only a 2D geometric modeller, and therefore intentional features contain only edge-references. Such an edge-reference is an unevaluated AML/X expression defined in terms of:

- the structure of a CSG representation of the particular model (if available),
- the adjacency information typically available in a boundary representation, and
- the names of operations that created or modified the edge.

Tools for automating the construction of expression that consistently identify faces in various versions of a model are currently under ivestigation by Paul Borrel and by the author.

4.3. *Validity tests*

The validity of intentional features, and thus of the produced model, may be tested automatically using validity rules attached to features. To a particular feature may be associated several rules (predicates), which measure geometric and topological properties of boundary elements. If, for example, any rule evaluates to FALSE, the feature is considered invalid. Note that the same feature may be valid for another application with different validity criteria. A rule can be any expression† in AML/X that returns a Boolean value. Typically such expressions involve references to intentional features, and therefore indirectly to corresponding geometric entities, and also calls to methods applied to these entities for computing their properties or deriving specific measures.

4.4. *Editing features*

4.4.1. *Volume features.* Intentional features can be created in MAMOUR by executing a shape-modifying operation, which attempts to produce a geometric feature through a Boolean operation and at the same time creates an object that represents the intentional feature and contains, in its internal variable, references to the corresponding geometric entities. Note that these entities need not always exist in the boundary representation of the model as it evolves. Since a volume rep resentation of the geometric feature is used to the Boolean operation, these features correspond to the volumefeatures discussed above. Such volume features may easily be used for editing the. model. To each boundary element and to each intentional feature in MAMOUR is associated a history attribute, which indicates what operation created the entity. After a face is graphically selected, its history may be accessed by the designer and the parameters* of the corresponding

† In particular, it can combine through an OR operation several Boolean subexpressions, and thus provides a simple mechanism for expressing validity rules as conjunctive forms or even more complex Boolean combinations.

‡ It is the designer's responsibility to decide which parameters should be edited and how. The systems could provide some help by maintaining a dependency graph that relates faces to parameter expression. Often these relations are simple, and we have not investigated such facilities for further assisting the designer.

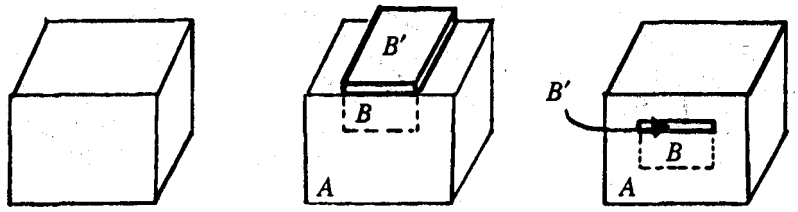

Fig. 19. Auxiliary volumes: A thin block B' over the slot B may be used to test whether the slot B is made in the block A (left) is accessible from the top. In the valid configuration (center) the set $B' \cap A$ is empty. In the incorrect configuration (right) $B' \cap A$ is not empty.

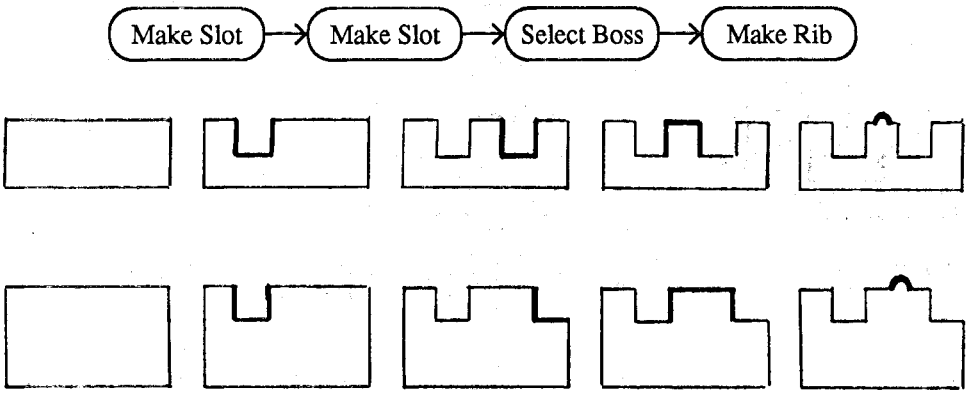

Fig. 20. Sequence of operations: The sequence (top) contains two slot-making operations, which subtract two volume features, followed by a feature-selection operation which identifies a surface feature of type boss used in the last operation to position a rib volume feature. The execution of the sequence produces a model (center). Editing one of the slot-making operations and reexecuting the sequence applying it to a different starting object produces a different result (bottom), which still reflects the designer's intent to center the rib on the boss.

operation edited so that the correct feature is created by reexecuting the whole sequence. This reevaluation may be computationally expensive. Therefore, alternate techniques for editing the model are discussed below.

4.4.2. *Surface features*. Intentional features may also be created without geometric modifications by a select operation which takes as parameters a feature type and a list of references to existing boundary elements.§ Typically there is no single shape modifying operation responsible for creating such a surface feature and thus no explicit volume representation. Techniques for editing such surface features by adding another operation to the sequence are addressed in the next section. They complement facilities for editing operations already in the sequence, which are well suited for modifying volume features when the cost of reevaluating the entire sequence is not prohibitive.

5. EFFICIENT EDITING OF VOLUME FEATURES

Active zones were introduced by Voelcker and the author in [27] to speed up certain geometric computations over Constructive Solid Geometry (CSG) representations. The active zone associated with a CSG node is the region in which the shape of the set represented by the node is important. An active zone is defined algebraically as the intersection of certain nodes of the whole CSG tree, and thus its CSG expression is always available. The concept of active zones is applied in this paper to deal with editing feature. In this section, from the properties of active zones, a CSG expression is derived, which, in conjunction with spatial localization techniques, may be used to improve the performance of algorithms that update the boundary representation of a model when a volume feature is altered. These improvements are particularly interesting when a spatial decomposition scheme is used for the boundary representation.

5.1. *Active zones*

Let S be a CSG representation of a solid encoded in a binary tree. (For simplicity, we shall use the same symbol for the solid and its CSG tree.) Let A be any primitive (or any internal node) of S. The active zone of A in S, denoted Z_A^S, is equal to the Boolean difference $I_A^S - U_A^S$, where I_A^S and U_A^S are respectively the I-zone and the U-zone of A in S. Let A be a primitive or internal node of a subtree N in S, and let F be the parent node of N of another node B. The following properties provide a recursive formulation for incrementally constructing I_A^S, U_A^S, and Z_A^S.

- I_A^A is the universe (Euclidean three-dimensional space) and U_A^A is the empty set.
- When $F = N \cup B$ or $F = B \cup N$, then $I_A^F = I_A^N$, $U_A^F = U_A^N \cup B$, and $Z_A^F = Z_A^N - B$.
- When $F = N \cap B$ or $F = B \cap N$, then $I_A^F = I_A^N \cap B$, $U_A^F = U_A^N$, and $Z_A^F = Z_A^N \cap B$.
- When $F = B - N$, then $I_A^F = B - U_A^N$, $U_A^F = I_A^N$, and $Z_A^F = Z_A^N \cap B$.
- When $F = N - B$, then $I_A^F = I_A^N \cap \bar{B}$, $U_A^F = U_A^N$, and $Z_A^F = Z_A^N - B$.

It follows that CSG expressions for I_A^S and U_A^S, and thus for Z_A^S, may be computed by traversing the path¶ in the tree from A to S, and constructing the Boolean expressions for I-zones and U-zones using intersections

§ Automatic extraction of existing 3D features is a difficult and expensive process [33], especially when loose conditions are used to identify features. For example, a slot with a filleted bottom edge may still be a slot, although the floor face is not directly connected to the wall. In MAMOUR, references to edges of features may be specified in a semiautomatic way using graphic interaction. These references are then integrated into unevaluated expressions, stored with the model, that will identify the corresponding feature in a family of part models generated by reexecuting a modified version of the sequence.

¶ The path is defined as the set of nodes traversed by moving from S to A in the CSG tree, following the parent-to-child links.

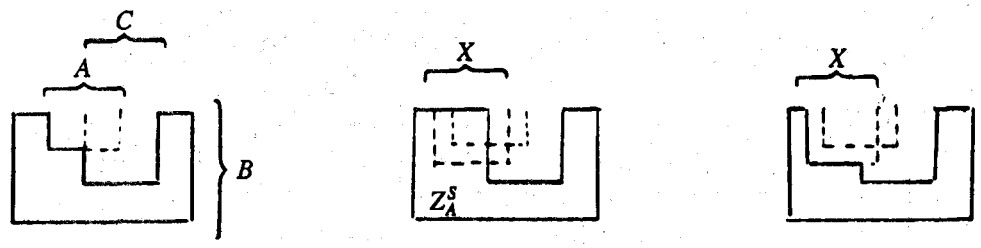

Fig. 21. Changes are confined to the active zone: Replacing the primitive A by X in the CSG definition of S (left) can only affect S in the active zone of A (center). The result is show (right).

with other branching nodes" or with their complements. On the other hand, these expressions need not be explicitly constructed, nor stored, for most applications, since efficient algorithms presented in [27] can perform calculations with respect to I_A^S and U_A^S using the appropriate branching nodes in the original tree S.

Several applications of Active Zones are discussed in details in [27]. For instance, it is shown that if a node A does not intersect its active zone, it can be replaced by the empty set. Similarly, if a node A contains its active zone, it can be replaced by the universe without affecting the set represented by the whole tree. (These results provide a new algorithm for testing whether any particular node is redundant.) Algorithms for the detection of interferences among solids defined in CSG and for the generation of shaded pictures directly from CSG by ray-casting[34] or depth-buffering[9] are based on facilities for classifying subsets of the boundary of primitives against the CSG tree, The performance of these algorithms may be improved using only the I-zones of the primitives, and not the whole CSG tree for the classification.

5.2. *Editing volumefeatures*

5.2.1. *Localization to the active zone.* Because a volume feature corresponds to an internal node of the CSG tree, it has an active zone, an I-zone, and a **U**-zone. It was established in [27] that all changes to a primitive or node A inside its active zone Z_A^S in Swill affect the shape of S and conversely that changes to A outside of Z_A^S will leave S unchanged. This property and related properties are used in this section to produce trimming expressions for localizing changes that implement a volume feature modification. For example, let A be a volume feature subtracted in the sequence (B − A) − C defining a solid **S** (see Fig. 21); its active zone Z_A^S is B − C. Only changes to A inside B − C need to be taken into account.

5.2.2. *Localization to the altered portion of the feature.* When a volume feature A is edited and replaced by X in a CSG representation of the solid S, a new boundary may be obtained without reevaluating the entire boundary of the solid because changes are restricted to $A \oplus X$, the symmetric difference† between A and X [35], Consequently, in the example Fig. 21, only alterations in $(A \cap X) \cap (B - C)$ are needed in order to compute the boundary of the solid obtained by replacing A by X in the CSG tree of S.

5.2.3. *Combined localization.* Let o and **w** represent respectively the empty set and the universe. Let S_X denote the set represented by the tree of S in which A was replaced by X. The two previous results may be combined to yield: $S_X = S \oplus Z_A^S \cap (X \oplus A)$. A proof of this equation may be derived from Equation (4) in **[36]**, given that $(X \oplus Y = Z) \Leftrightarrow (X = Y \oplus Z)$ holds for any three sets, X, Y, and **Z**.

5.2.4. *Classifying additive and subtractive parts.* The symmetric difference $A \oplus X$ may be decomposed into two disjoint sets,‡ A − X and X − A, which can be treated separately. For example, when a negative (subtractive) volume feature A is replaced by **X**, a subset of A − X must be added to S and a subset of X − A mustbe subtracted (see Fig. 22 for an example). This section derives new CSG expressions that characterize these subsets and are simpler than the CSG expression of s§.

Although Z_A^S is the smallest region where changes to A affect S, the classification of A − X and X − A against the CSG expression of Z_A^S in general contains unnecessary steps which can be eliminated using the following property;

$$S_X = S - (A_X^- - U_A^S) \cup (A_X^+ \cap I_A^S),$$

where A_X^+ is the portion of material added to A and where A_X^- is the portion of material subtracted from A during the same operation.

First consider the case of an additive feature A. We have: $A_X^+ = X - A$ and $A_X^- = A - X$. For a proof, first we establish the following:

‖ The symbol **B** in the above properties refers to what is formally called a branching node of A in S, i.e., a node that does not lie in the path from A to S, but whose parent node does. I_A^S and Z_A^S are defined as intersections of such branching nodes or of their complements.

† The symmetric difference between A and X is defined as: $A \oplus X = (A - X) \cup (X - A)$.

‡ The term disjoint is used here for three-dimensional volumes whose intersection is empty or of lower dimension.

§ Note that each one of these CSG expressions may be further trimmed down by removing all primitives that are disjoint from A − X (respectively X − A) by using techniques discussed in **[37]**.

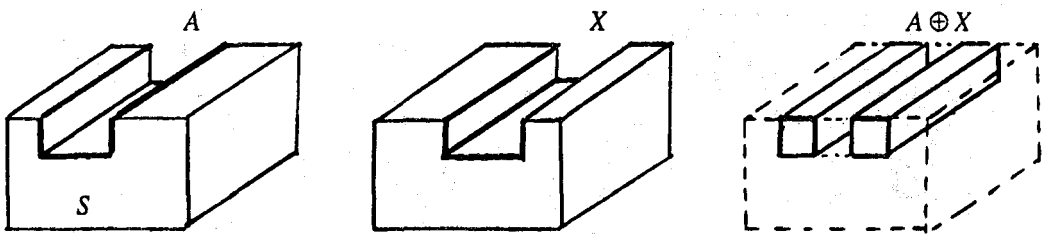

Fig. 22. Changes are confined to the symmetric difference: The solid S (left) is represented by a CSG tree defined in terms of a subtractive volume feature A. Replacing A by X in the CSG tree of S may change the shape of S (center) only in the symmetric difference A ⊖ X (right). Note that A − X is added and that X − A is subtracted.

$$I_A^S \cap U_A^S \subset I_A^S \cap U_A^S \cup S_o \cap U_A^S,$$

$$I_A^S \cap U_A^S \subset (I_A^S \cup S_o) \cap U_A^S,$$

$$I_A^S \cap U_A^S \subset S_w \cap U_A^S, \quad \text{by Proposition 5 of [27]},$$

$$I_A^S \cap U_A^S \subset S_o, \quad \text{by Proposition 6 of [27]}.$$

Now the proof may be constructed as follows:

$$S_X = S_o \cup (X \cap I_A^S), \quad \text{by Proposition 1 of [27]},$$

$$S_X = S_o \cup (X \cap I_A^S) \cup (A \cap I_A^S \cap U_A^S), \quad \text{since } I_A^S \cap U_A^S \subset S_o \text{ (see above)},$$

$$S_X = (S_o \cap \bar{A}) \cup (S_o \cap X) \cup S_o \cup (X \cap I_A^S) \cup (A \cap I_A^S \cap U_A^S), \quad \text{since } S_o \cap \bar{A} \cup S_o \cap X \subset S_o,$$

$$S_X = (S_o \cap \bar{A}) \cup (S_o \cap X) \cup (S_o \cap U_A^S) \cup (X \cap I_A^S) \cup (A \cap I_A^S \cap U_A^S), \quad \text{since } S_o \subset U_A^S \text{ (see [27])},$$

$$S_X = (S_o \cap \bar{A}) \cup (S_o \cap X) \cup (S \cap U_A^S) \cup (A \cap X \cap I_A^S \cap \bar{A} \cap X \cap I_A^S) \cup (A \cap I_A^S \cap U_A^S), \quad \text{since } A \cup \bar{A} = w,$$

$$S_X = (S_o \cap \bar{A}) \cup (S_o \cap X) \cup (S_o \cap U_A^S) \cup (A \cap I_A^S \cap \bar{A}) \cup (A \cap X \cap I_A^S) \cup (A \cap I_A^S \cap U_A^S) \cup (\bar{A} \cap X \cap I_A^S),$$

$$S_X = (S_o \cup A \cap I_A^S) \cap (\bar{A} \cup X \cup U_A^S) \cup (\bar{A} \cap X \cap I_A^S), \quad \text{factoring out } S_o \text{ and } \bar{A} \cup X,$$

$$S_X = S \cap (\bar{A} \cup X \cup U_A^S) \cup (\bar{A} \cap X \cap I_A^S), \quad \text{replacing } X \text{ with } A \text{ in Proposition 1 of [27]},$$

$$S_X = S - ((A - X) - U_A^S) \cup ((X - A) \cap I_A^S), \quad \text{using complementation}.$$

Now note that for a subtractive feature, one may consider the complements of A and X to be additive features, and the same proof holds.

Since the interiors of A_X^+ and of A_X^- are disjoint, the interiors of $A_X^+ - U_A^S$ and $A_X^- \cap I_A^S$ are also disjoint. Therefore, since ∪ and − are regularized, we also have:

$$S_X = S \cup (A_X^- \cap I_A^S) - (A_X^+ - U_A^S).$$

Consequently, subsets of A_X^+ need only be classified against I_A^S; subsets inside I_A^S should be added to S, while subsets outside may be discarded.' Similarly, subsets of A_X^- need only be classified against U_A^S; subsets outside U_A^S should be subtracted from S, while subsets inside U_A^S may be discarded."

The CSG expression of Z_A^S, defined as $I_A^S - U_A^S$, is a complex as the combination of the CSG expressions for I_A^S and for U_A^S, and classifying† a point with respect to Z_A^S may often require classifying it against I," and against U_A^S and then combining the result. The above result suggests that A_X^+ needs only be classified against I_A^S and not against both I_A^S and U_A^S. Similarly, A_X^- needs only be classified against U_A^S and not against both I_A^S and U_A^S. Significant improvements for algorithms that perform feature modification result. To further improve the performance of these algorithms, tree pruning and other space subdivision techniques[39, 40, 35, 41] may be used in conjunction with classification against I_A^S and U_A^S. Note, however, that this speed-up may suggest to add to S portions of A_X^+ that are already included in S, or to subtract from S portions of A_X^- that are not in S. These unnecessary additions do not invalidate the result but may correspond to redundant geometric calculations unless the representation scheme presented below is used.

5.3. *Mixed-dimensional geometric model*

To represent a solid part together with its additive and subtractive volume features, one needs to extend the boundary representation scheme and to provide support for representing decompositions of solids and of their complements in terms of boundaries of potentially overlapping features (Fig. 16). Furthermore, the construction of closing faces for surface features may

¶ See Property 13 in [27].
‖ See Property 12 in [27].
† "Classifying a point against a set consists of determining whether the point lies inside or outside of the set[38].

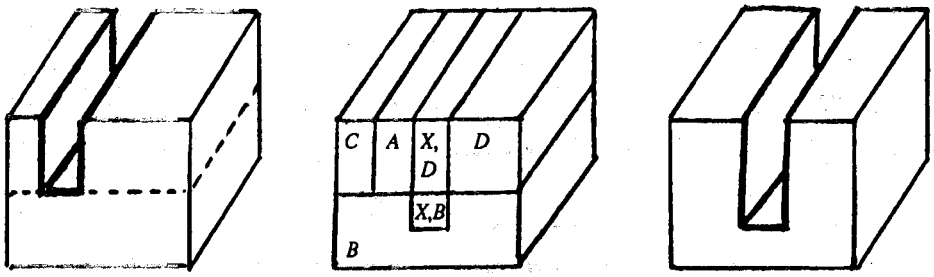

Fig. 23. Space decomposition for editing a surface feature: The slot surface feature in the solid B U C U D (left) may be modified by creating the corresponding volume feature A and its modified version X, by constructing an SGC representation of the resulting space subdivision (center) and by setting the status of cells of A − X to IN, and of cells of X − A to OUT (right).

involve splitting faces of the part model and inserting internal faces in the part or creating external faces in its complement. Objects with internal structures and external dangling edges and faces are not supported by conventional solid modellers, and thus richer schemes for geometric representations must be used.

Several data structures that support representations of unions of quasi-disjoint subvolumes have been proposed (see [28–30, 42] for examples). Such schemes may be used to decompose a volume feature B into its active and inactive portions with respect to various Boolean combinations of other features. The Selective Geometric Complex (abbreviated SGC) data structure permits representation of such decompositions together with the modelled part, whether the feature has been added or subtracted [29],

SGCs provide a common framework for representing objects of mixed dimensionality, possibly with internal structures and incomplete boundaries. SGCs are composed of finite collections of mutually disjoint cells, which are open connected subsets of n-dimensional manifolds and generalize the concepts of edges, faces, and vertices used in most solid modellers. The connectivity between such cells is captured in a very simple incidence graph, whose links indicate "is-a-boundary-of" relations between cells. By setting the status (attribute) of certain cells to IN and others to OUT, one can associate various pointsets with a single collection of cells. When a cell's status is IN, the cell is called "active," and the pointset spanned by the cell is considered as part of the object; otherwise it is considered as part of the complement of the object. Consequently, the pointset of an SGC object needs not be homogeneous in dimension, nor even be closed or bounded. To support useful operations on SGCs. Boolean and other set-theoretical operations (closure, interior, boundary) have been decomposed into combinations of three fundamental steps for which dimension-independent algorithms have been developed [29]:

- a subdivision step, which makes two objects "compatible" by subdividing the cells of each object at their intersections with cells of the other object;
- a selection step, which defines "active" cells, i.e., cells whose status is IN; and
- a simplification step, which, by deleting or merging certain cells, reduces the complexity of an object's representation without changing the represented pointset and without destroying decompositions that are marked as important for applications.

Furthermore, combinations of these steps may produce a variety of special-purpose operations whose effect is controlled by simple predicates, or filters, for cell selection.

The subdivision step may be used to create a space decomposition that reflects the geometry of the part and of all its volume and surface features. Then, feature modifications may be performed by modifying the status of cells that belong to the appropriate volume features and the associated I or U zones. Fig. 23 illustrates how the result of a sequence of operations may be modified by selecting a surface feature and modifying it by replacing the volume feature A with X.

5.3.1. *Application to model updating.* A space decomposition technique may be used to split A ⊕ X into connected cells that are entirely inside or entirely outside of both I_A^S and U_A^S. Adding these cells to S or subtracting them from S only requires setting their status to IN our OUT. If such an approach is used, the main cost lies in the insertion of X in the space decomposition and in the classification of each cell of A ⊕ X. The classification may simply be obtained by evaluating a Boolean expression. Using I_A^S or U_A^S instead of the expression for Z_A^S considerably reduces the cost of updating the model to reflect a feature modification (see Fig. 24).

5.3.2. *Volumefeature deletion.* A particular case of editing is deletion. A volume feature could be deleted by altering a specification stored in a procedural model and by reexecuting the procedural model to create a new geometric model without the undesirable feature The same result could be obtained directly by changing the status of all cells that lie in the intersection of that volume feature with its active zone.

For example, to delete a feature A in (B U A) − C. it suffices to change the status of cells in A − (B U C). the active portion of A (Fig. 25). These changes can be done by traversing† all the cells in A and classifying

† We assume that each feature has a reference to all the cells that it includes or that these cells may be efficiently identified in the SGC. If no provision is made for such a direct access, all cells of specific dimensions may have to be traversed to see if they belong to the appropriate feature.

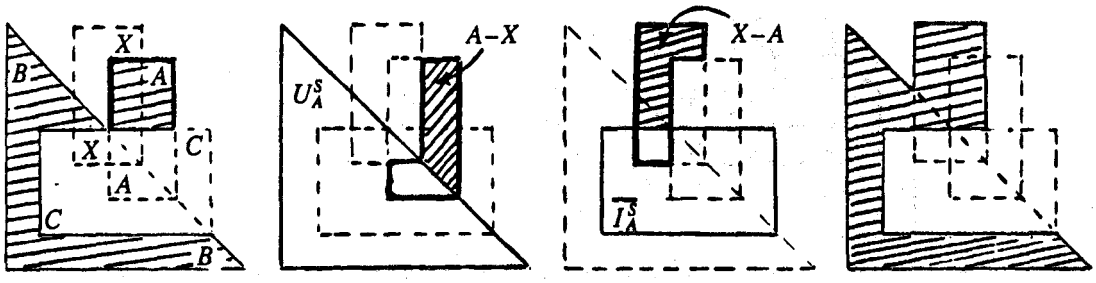

Fig. 24. Active zone in feature modification: S defined as (B U A) − C is shown (left) superimposed on the space decomposition obtained by using the boundaries of the primitives A, B, C, and X. The primitive A is a misplaced additive volume feature in S and must be replaced with a similar feature X slightly shifted to the left and upwards. To obtain the correct object, first cells that belong to A − X are classified against U_A^S (here U_A^S = B). The status of cells of A − X outside of U_A^S is set to OUT. Note that one of these cells was already OUT (center left). Then, cells of X − A are classified against I," (here $I_A^S = \bar{C}$). The status of the cells of X − A inside I_A^S is set to IN (center right). Note that one of these cells was already IN. The result (cells whose attributes are IN) is shown (right).

each one against B U C. Since each cell contains (in its history) a list of the features to which it belongs, this classification amounts simply to evaluating a Boolean expression.

As discussed earlier, instead of changing the status of only the cells of A that lie in its active zone, one could simply set to OUT the status of cells of A that lie in its U-zone when A is an additive volume feature, or set to IN the status of cells of A that lie in its I-zone when A is a subtractive volume feature.

For example, in the solid (B U A) − C (Fig. **25),** the U-zone, U_A^S = B, is simpler than the active zone, $Z_A^S = \overline{B \cup C}$, and thus a saving in classification time is achieved. This simplistic example does not reflect the importance of such savings, and one should consider that classification has to be performed for a large number of cells and that it may involve large Boolean expressions for both U_A^S and I_A^S.

6. CORRECTIVE VOLUMES

If a CSG tree representing the model is available, surface features could in principle be corrected by editing the tree. Suppose that no single node of the tree represents the appropriate volume feature. Each face of the feature may still be associated with one or more half-spaces of the CSG tree, so that modifying the position or shape of these half-spaces will affect the face, and thus the feature.

Editing half-spaces directly in the original CSG tree has the following drawbacks:

1. It may be difficult to recognize which half-space should be edited. Typically the boundary of more than one half-space coincides with each face, and techniques proposed in [27] for classifying half-spaces against their active zones (to predict which half-spaces will affect which portion of a particular face) may not be sufficient.
2. Editing half-spaces in a CSG tree to alter a particular portion of space will often produce undesired side-effects in other locations.
3. Editing half-spaces and not solid primitives can produce unbounded subsolids and makes it very difficult to implement popular performance-improving techniques for CSG algorithms that use bounding boxes around CSG primitives.
4. Designers' intentions expressed in a posteriori alterations of half-spaces are difficult to specify and to capture in the sequence of a procedural model. Consequently, the edited specification is not well suited for reparameterization or reuse in a different context. It is not even suited for further editing that involves a reevaluation of the sequence.

Reorganizing the CSG tree to regroup the relevant half-spaces of the feature into a single node that can be edited as a feature is not always possible since the result of evaluating a general Boolean expression is order-dependent. Consequently, this section proposes to use Boolean operations to edit surface features (the previous section dealt with volume features), These

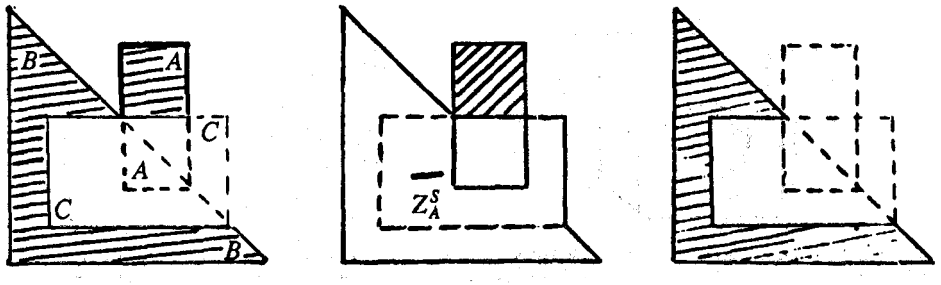

Fig. 25. Deletion using the Active Zone: Given the set S defined by (B U A) − C (left) the additive volume feature A may be deleted by changing the active attribute of all the cells that lie in the active portion of A (center), which is A − (B U C). The result is shown (right).

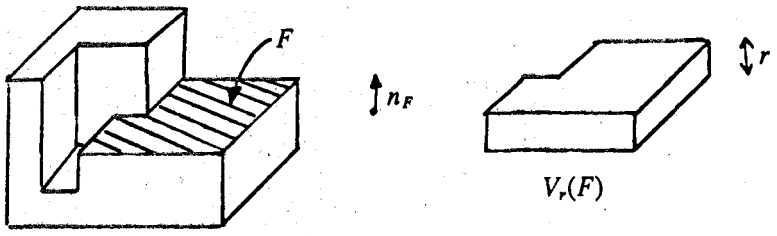

Fig. 26. Face extrusions: Extruding a planar base face (left) produces a volume (right) that is bounded by the base face and its offset by r, and by side faces obtained by sweeping the edges of the base face.

operations involve corrective volumes which are added to—or subtracted from—a solid model to alter a particular feature. These corrective volumes could be derived from volume features themselves derived from surface features by generating closing faces, but, as pointed out in Section 2, the derivation of closing faces remains a research issue. Furthermore, once a volume feature A is derived from the surface feature, it is not easy to produce a modified volume feature X so that $A - X$ and $X - A$ may be used as corrective volumes and added to—or subtracted from—the model to alter the surface feature without side-effects. Instead, the author proposes to construct corrective volumes by extruding appropriate faces of the surface feature, Examples of these extrusions are provided in the following part of this section.

Once corrective volumes are computed, whether through closing faces and volume features or through face extrusions, they must often be trimmed to avoid undesirable side-effects before they can be combined with the solid. Automatically computing the correct trimming expression is impossible unless the expression "undesirable side-effect"is formally defined. Indeed, the correctness of a feature editing operation depends on the function of the edited feature and on the function of its geometric relation with other features.

6.1. *Extrusion of faces*

A simplest corrective volume may be obtained by extruding a planar face called the base face along the normal to its supporting plane (see Fig. 26 for an example). Such a corrective volume may often be adequate to change the width or depth of features that have orthogonally oriented planar faces.

The extrusion $V_r(F)$ of a planar face F by a distance r is a volume formally defined by

$$V_r(F) = \{\mathbf{p} + t\mathbf{n}_F \text{ with } 0 \le t \le r \text{ and } \mathbf{p} \in F\},$$

where $\mathbf{n}_F$ is the normal unit vector to the surface containing F.

For faces on curved surfaces, a normal extrusion will be used (Fig. 27). It is simple extension of the above extrusion. The normal extrusion $V_r(F)$ of a curved base face F by a distance r is a volume formally defined as:

$$V_r(F) = \{\mathbf{p} + t\mathbf{n}_F(\mathbf{p}) \text{ with } 0 \le t \le r \text{ and } \mathbf{p} \in iF\},$$

where $\mathbf{n}_F(\mathbf{p})$ is the normal unit vector to F at point p and where iF is the relative interior† of F with respect to its supporting surface. The orientation of the normal is chosen in a consistent manner throughout F. Note that, for the semialgebraic surfaces popular in solid modelling, $\mathbf{n}_F(\mathbf{p})$ is well defined for the smooth portions of F (the relative interior of the base face), but needs not be well defined for the bounding edges of F, its cusps, or singulanties.‡ Therefore, $V_r(F)$ is obtained by extruding a nonclosed face and thus does not necessarily contain all its boundary; it needs to be regularized. $\mathbf{n}_F(\mathbf{p})$ can often be computed from the cross product of partial derivatives when F is defined in parametric form, or from a gradient when F is defined by an implicit equation. Such extrusion volumes are very simple to obtain for natural surfaces (see [43]).

Applications of extrusions and normal extrusions to feature modification are illustrated in Figs. 28 and 29, where the extrusions are used as corrective volumes and added to the part to modify a feature.

Such applications are clearly limited to simple cases where, for example, the extruded base face and the abutting faces meet at a right angle. Fig. 30 shows a

† The interior of a face is the face minus its bounding edges, cusps, or singular points.

‡ The normal to a cone is not defined as its apex.

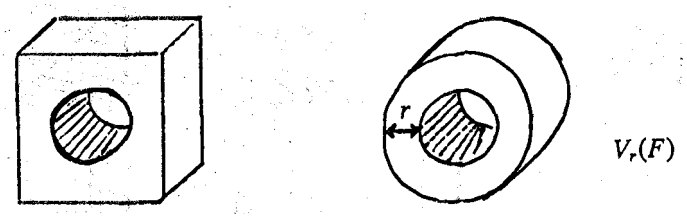

Fig. 27. Normal extrusion for curved base faces: A normal extrusion of a curved face that lies on a cylinder (left) is a volume (right) bounded by the original face, its offset by r, and by side faces that are subsets of ruled surfaces sustained by the edges of the original face. Note that the side faces are in the closure of $V_r(F)$ but are not in $V_r(F)$.

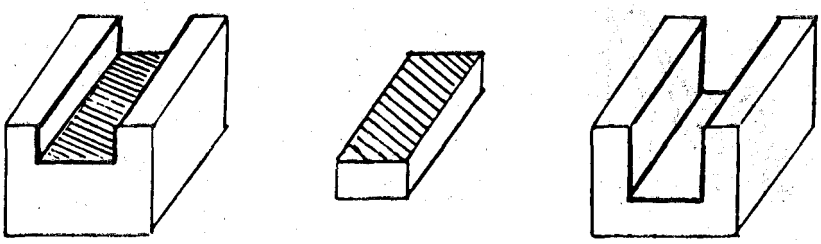

Fig. 28. Application of extrusion: To change the depth of the slot feature (left), a corrective volume (center) may be generated by extruding the floor of the slot and subtracted from the part (right).

counterexample for which a corrective volume generated by extruding a base face is inadequate.

To circumvent such limitations, an extended corrective volume may be generated and then trimmed using the abutting faces (see Fig. 31).

The extended corrective volume may be obtained by extruding an extension of the base face. A good candidate to use as base face extension is the entire surface that contains the base face, but in certain cases, to avoid side-effects in distant areas, it may be preferable to consider only a simple subset of that surface. This technique has been proposed by Requicha and the author in [12] for generating local fillets and blends by (1) growing and shrinking Boolean combinations of appropriate half-spaces, by (2) subtracting the result from the original combination to obtain an extended blend, by (3) trimming the blend to the desired shape near its ends, and by (4) adding the resulting corrective volume to the part or subtracting it.

Extended corrective volumes derived from a single base face are convenient for modifying a single dimension of a simpler feature (e.g., the width of a slot or the radius of a hole). In general, however a corrective volume involves more than one base face (for example, when the depth of a pocket with an uneven floor is to be changed). For such cases, instead of combining several corrective volumes, a single extended corrective volume may be obtained by extruding several faces along a common direction (Fig. 32). Such a generalized extrusion, $V_{\mathbf{u}}(F)$ of a set of faces F along a direction **u**, may be formally defined as the Minkowski sum [44] of F with the line segment joining the origin **O** with the point **O** + **u**. And thus:

$$V_{\mathbf{u}}(F) = \{\mathbf{p} + t\mathbf{u}, \text{ with } \mathbf{p} \in F \text{ for } 0 \le t \le r\}.$$

6.2. *Automatic derivation of a default trimming expression*

Picking the base face and providing an offset distance does not in general provide an unambiguous specification for the corrective volume. It is thus necessary to provide facilities for automatically creating supersets of the desired corrective volumes and trimming expressions that produce the desired subsets. Trimming operations may involve nontrivial Boolean combinations of the half-spaces bounded by the abutting faces. This Boolean combination may, in principle, be derived from the entire CSG tree, if available, but this derivation may be difficult and will often require human intervention. A possible approach to assist the designer and suggest a default trimming CSG expression is to consider only half-spaces bounded by the faces of the solid that share edges with the base face and to eliminate other half-spaces from the tree. The correct Boolean combination of these half-spaces may be hard to derive, and the results may still be incorrect (see Fig. 33). The designer may have to provide an auxiliary trimming expression, but the system should be able to suggest a good default trimming expression.

Clearly, corrective volumes should be confined to a region where they do not destroy the effects of other features. For example, they may be trimmed by some other (but not necessarily all other) volume features.

Unfortunately, a surface feature, F, typically does not correspond to any individual node in the CSG tree and therefore is not associated with an active zone; the results derived in Section 5 may not be directly applied here.

To provide an often reasonable default trimming expression, the concept of a virtual active zone may be used, The virtual active zone of a surface feature F is defined as the active zone of the lowest node T in the

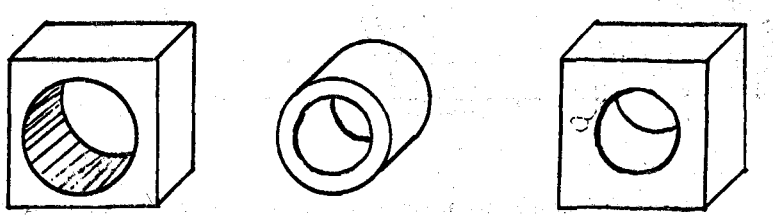

Fig. 29. Application of normal extrusion: To change the radius of a cylindrical hole (left), a corrective volume (center) may be generated through a normal extrusion of the cylindrical base face and added to the part (right).

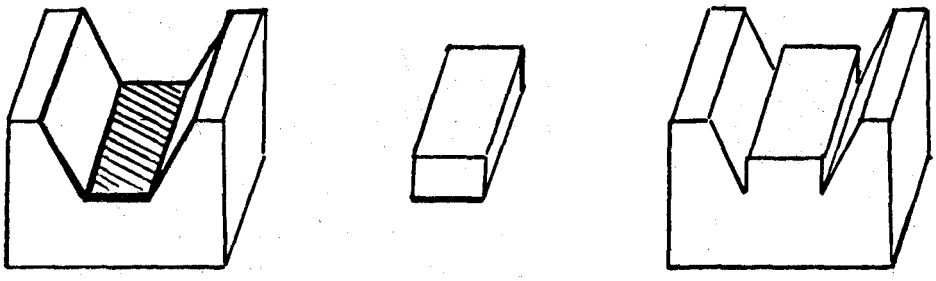

Fig. 30. Limitation of face extrusion: The geometric feature (left) has been edited by adding to the solid the corrective volume generated by extruding one of its faces (center). The result (right) is not correct.

CSG tree of ***S*** at which the surface feature F appears. This technique may be automated, but is only effective if at T one could delete and create again the corresponding volume feature without changing the final result. In other cases, fine tuning "by hand" the automatically generated virtual active zone may be necessary.

7. VALIDITY TESTS

Previous sections dealt with techniques for assisting the designer in performing object modifications using a feature-oriented syntax. It has been assumed that no automated solution exists and that human intervention is necessary to correct the side-effect of these editing operations. To further assist the designer, the system should support facilities for interrogating important properties of features. We shall refer to these properties using the global term of validity.

The validity of a feature or of a compound feature greatly depends on the nature of the feature and on the function played by the feature in a particular application. Addressed here are only the qualitative (or discrete) validity issues that can be expressed id terms of the presence or absence of specific cells in SGC structures. (Many other quantitative criteria may be easily addressed through geometric measures, derived from features, as described in [4].) Cells of in terest have a specified dimension and belong to the appropriate combination of features. Note that this approach is not based on matching subsets of the adjacency graph of a boundary representation, but on the interogation of a rich data structure that captures various geometric entities and their relations.

Filters for cell selection in SGCs may be used for validity testing. The next section introduces the query operators, which provide the vocabulary necessary for expressing the filters. The subsequent section demonstrates their application on a few simple examples.

7.1. *Interrogation operators for SGCs*

Filters will be expressed in terms of the following queries that can be made to an SGC, O, or to a particular cell, C, of O. A list of typical filters, expressed as methods of cell or SGC objects, follows.

- O.cells(k) returns the collection of cells of dimension k in ***O.***
- O.skeleton(k) returns the collection of cells of dimension less or equal to k in O.
- C.boundary returns the collection of cells that bound C.
- C.star returns the collection of cells bounded by C.
- C.dimension returns the dimension of C.
- Given a cell D of C.star such that D.dimension = C.dimension + 1, C.nbhd(D) returns leftdir, rightdir, or bothdirs. The value leftdir means that, given the definition of "left" and "right" with respect to an orientation of C in the manifold containing D as an open subset, D lies on the "left" of C. Similarly, when C.nbhd(D) returns rightdir, C bounds D on the "right." bothdirs means that C is an interior boundary "surrounded" by D, or more precisely that

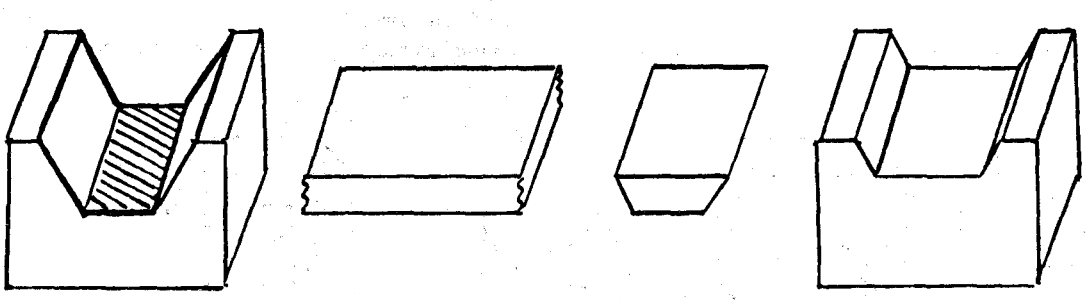

Fig. 31. Extended corrective volumes: To raise the floor of the slot (left) without modifying the position of its walls, an extended corrective volume is generated and trimmed (center). The result is added to the part (right).

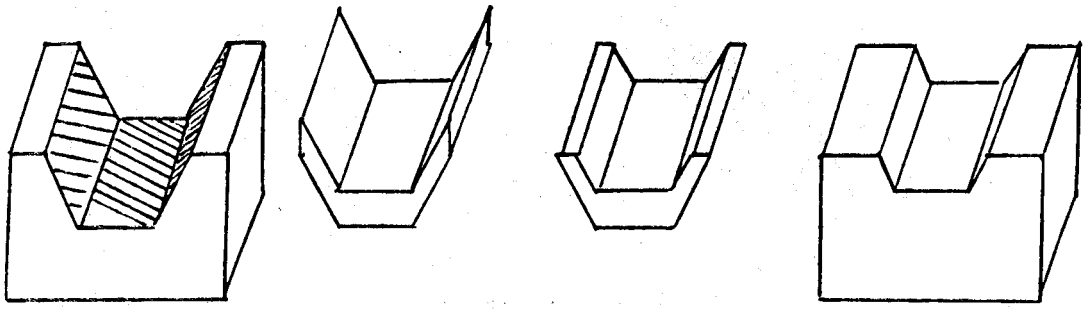

Fig. 32. Corrective swept volumes: To raise a slot in the model (left) without changing the shape of its floor, a corrective volume is obtained by sweeping upwards all the faces of the slot (center left), and then by trimming the result (center right) and adding it to the model (right). The dimensions of the floor remain unchanged and therefore the swept volumes of the different faces have different thickness.

C is contained in the topological interior (with respect to the manifold containing D) of the topological closure of D.

- C.history returns the set of features to which C belongs.
- Finally, C.status returns IN or OUT depending whether the cell is considered as actively contributing to the pointset of O or not.

For more formal definitions of these operators, the reader should consult [29].

7.2. *Examples of application*

Validity criteria are domain dependent, and the goal of this paper is not to derive them but to illustrate a technique for expressing them. Three simple examples involving a block of material and two volume features will be used. Simple tests that characterize each situation are proposed. These tests do not involve any geometric calculations, but simply search the SGC structure for cells that satisfy appropriate selection criteria. In fact, to improve the interrogation performance, references to cells of the SGC could be organized in a data base and accessed using their history, dimension, or other characteristics by standard data base queries.

The local inaccessibility from the top of a slot volume feature A in a part B may be detected by checking whether the "roof" face, F1, of A is connected on the outside (with respect to A) to a full-dimensional cell of B (Fig. 34). The test may be performed by selecting cells D of F1.star that contains B in their history and such that F1.nbhd(D) = leftdir. (For simplicity, we assume that F1 is oriented so that the leftdir direction points toward the outside of A.)

Of course, the foregoing filter is not adequate for global accessibility, which may require performing Boolean operations on auxiliary volumes, such as the volume swept by the portion of the feature that is visible from the direction from which the feature is to be accessed.

To find whether two slots A and C are adjacent along a common face or not (Fig. 35), it suffices to inquire whether a 2D cell, or face, F2 exists such that it bounds a 3D cell of A on one side and a 3D cell of C on the other side.

The search may be confined to the common 2D cells of the boundaries of A and C, which may be accessed directly if a directory of cells that belong to each feature is maintained.

To test whether a slot C is contained in a slot A (Fig. 36), or more generally whether A and C interfere, it suffices to query if there is a three-dimensional cell whose history contains both A and C as well as B. Again, the selection may be efficiently performed by standard data base queries on the directories of cells that belong to A, B, and C organized by dimension.

8. CONCLUSION

Interactive editing of CAD models may be simplified by the use of volume and surface features. Surface fea-

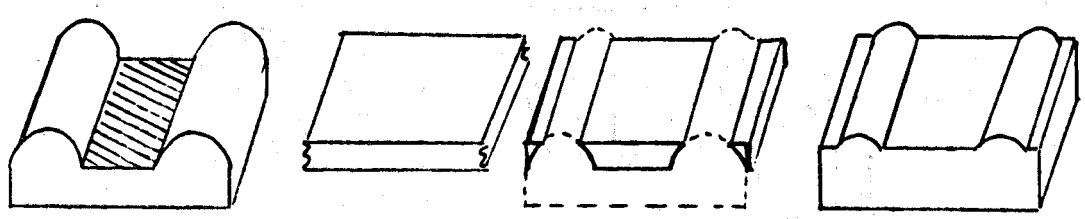

Fig. 33. Incorrect trimming CSG: To raise the floor of the slit in the model (left), an extended corrective volume is computed (center left). CSG expressions defined only in terms of half-spaces bounded by the faces of the original model are not adequate for trimming the corrective volume. An example of an incorrectly trimmed corrective volume is shown (center right), and the resulting model is shown (right).

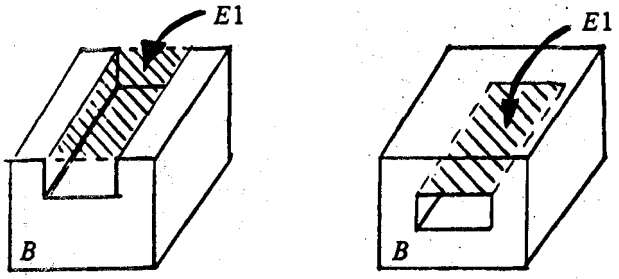

Fig. 34. Testing top accessibility: To distinguish between the correct positioning of the slot feature (left) and the incorrect one (right), one can simply inquire whether, in the corresponding SGC that represents the space decomposition, the roof face E1 of slot A is connected on its outside to full-dimensional cells of **B**.

tures are interactively selected by the user from existing faces of a model. Volume features may be created by addition or subtraction of material, or derived from surface features by defining the necessary closing faces. This paper addresses the issue of interrogating such volume and surface features as to their validity and their relation to each other and to the final part. It also proposes techniques for editing the model by modifying or deleting its features.

Interrogation techniques are based on a scheme for representing mixed-dimensional pointsets with internal structures. In that scheme, space, i.e., the modelled part and its complement, is subdivided into connected cells (volumes, faces, edges, and vertices) such that all points of any given cell belong to the same set of features. Feature validity and particular relations among features may be easily characterized by the existence or the absence of cells of specific dimensions associated with specific sets of features.

Model editing to alter or delete a volume feature may be performed by using a procedural representation of the designer's specification, locating in it the command that created a particular volume feature to be modified, editing this command, and reexecuting the entire procedural model. Reexecuting means computing the boundary of the geometric model, which may be an expensive process. When a CSG expression for the part is available, the reexecution may be limited to a particular domain defined by the intersection of the volume feature with its active zone. This technique may be adapted to surface features by providing closing faces which define corresponding volume features and by considering the role these volume features play with respect to the CSG tree. On the other hand, surface features may be directly modified by constructing corrective volumes and by combining them to the part model through addition or subtraction. The use of extended geometric representations of sets with internal and external structures permits calculation of the effect of feature deletion without further geometric calculations.

Global access to the geometric elements of a particular feature is provided through intentional features, to which may be associated validity rules and methods for evaluating these rules or for measuring important properties of the feature. Intentional features may be created automatically when feature-based shape-modifying operations are used or by interactively selecting existing faces. Intentional features do not directly point to any geometric entity, but carry an unevaluated expression, constructed at feature creation or selection, which, when evaluated, returns appropriate geometric elements, if they exist. This indirect approach prevents inconsistencies between an abstract list of assumed features that could characterize some important aspects of a part and the actual presence and geometry of these features in the part. This point is essential if further shape modifications, done either by editing and replaying the designer's specification or by creating new features, can alter the geometry of a previously defined

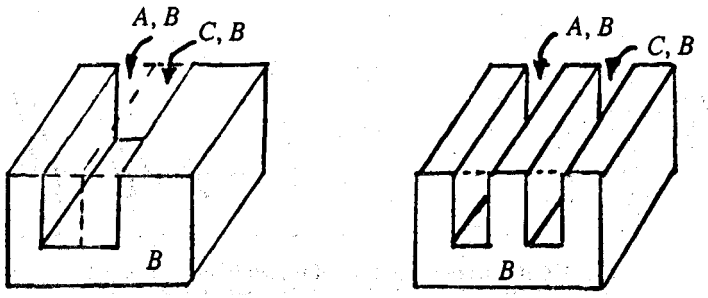

Fig. 35. Testing adjacency: For process planning, the two adjacent slots A and C (left) must be treated differently from the two disconnected slots (right).

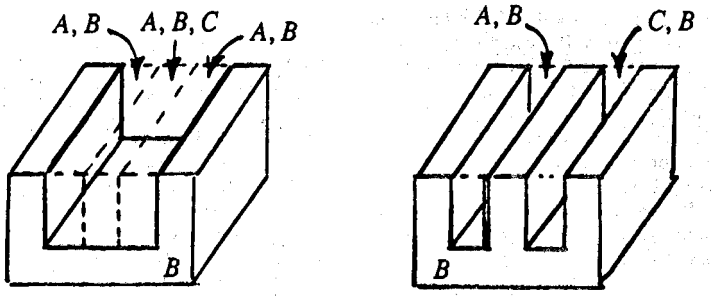

Fig. 36. Testing containment: Machining a slot C that lies inside a slot A may result in suboptimal manufacturing processes. Therefore, such a situation (left) must be distinguished from a normal situation where the two slots are disjoint (right).

feature to the point that it no longer exhibits the expected geometric characteristics.

Acknowledgments—The original motivation for a significant part of this work has been provided by Professor Michael Pratt'ssuggestion to use nonmanifold geometry for representing volume features and their interrelations. It became clear that the concepts and data structures for Selective Geometric Complexes, which have been designed at IBM by Michael O'Connorand the author, can be used to facilitate the explicit representation and interrogation of the geometric relations between features. The use of intentional features as a means for interrogating and editing a design has been investigated by Paul Borrel, Lee Nackman, and the author, as part of the development of the MAMOUR prototype system, and also by Franklin Gracer and the author, while attempting to incorporate "persistent" features in the GDP solid modeller. Furthermore, techniques for using in intentional features the unevaluated references to boundary elements and for ensuring that these referencesremain meaningful, even when the specification has been edited, have been studied by Paul Borrel and the author. The idea of using trimmed corrective volumes to edit solids defined in CSG was proposed by Professor Aristides Requicha and the author at the Universityof Rochester. The original concepts of Active Zones in CSG were introduced by Professor Herbert Voelcker and the author at the University of Rochester. The author thus wishes to thank Michael Pratt for his inspiration and Paul Borrel, Franklin Gracer, Michael O'Conner,Aristides Requicha, and Herbert Voelcker for their contributions to certain aspects of this work. The author is also grateful to Vijay Srinivasan and Michael Wesley for supporting this work, to Professors Fahrad Arbab and Tetsuo Tomiyama for encouragingits publication, and to Erik Jansen, Martti Mäntylä,and several other colleagues and friends for their constructive comments.

REFERENCES

1. T. Tomiyama and H. Yoshikawa, Extended general design theory, In Design Theory for CAD, H. Yoshikawa and E. A. Warman (Eds.), North-Holland, The Netherlands, 95–130 (1987).
2. A. A. G. Requicha and S. C. Chan, Representation of geometric features, tolerances and attributes in solid modellers based on constructive geometry. IEEE Journal of Robotics and Automation 2(3), 156–166 (September 1986).
3. J. R. Dixon, J. C. Cunningham and M. K. Simmons, Research in designing with features, IFIP Working Group 5.2 Conference on Intelligent CAD, Boston, October 5–8, 1987.
4. J. R. Rossignac, P. Borrel and L. R. Nackman, Interactive design with sequences of parameterized transformations. Proceedings ofthe Second Eurographics Workshop on Intelligent CAD Systems: Implementation Issues, April 11–15, 1988, Veldhoven, The Netherlands, 97–127. Also available as IBM Research Report RC 13740, IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY, May 1988.
5. J. R. Rossignac, P. Borrel and L. R. Nackman, Procedural models for design and fabrication. Proceedings of the MIT Sea Grant Symposium, Cambridge, MA, October 24–26, 1988. Also available as IBM Research Report RC 14056, IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY, October 1988.
6. A. A. G. Requicha and H. B. Voelcker, Constructive solid geometry. Tech. Memo. No. 25, Production Automation Project, Univ. of Rochester, November 1977. (Reports of the Production Automation Project are no longer available from the University of Rochester, but may be obtained from CPA, 304 Kimball Hall, Cornell University, Ithaca, NY 14853.)
7. Y. T. Lee and A. A. G. Requicha, Algorithms for computing the volume and other integral properties of solids: I—Known methods and open issues, II–A family of algorithms based on representation conversion and cellular approximation. Comm ACM25(9), 635–641 (September 1982)
8. J. E. Bobrow, NC machine tool path generations from CSG part representations. Computer Aided Design 17(2), 69–76 (March 1985).
9. J. R. Rossignac and A. A. G. Requicha, Depth buffering display techniques for constructive solid geometry. IEEE, Computer Graphics and Applications 6(9), 29–39 (September 1986).
10. A. A. G. Requicha and H. B. Voelcker, Boolean operations in solid modelling: Boundary evaluation and merging algorithms. Proceedings ofthe IEEE 73(1), 30–44 (January 1985).
11. K. J. Weiler, Edge-based data structure for solid modelling in curved surface environments. IEEE Computer Graphics andApplications 5(1), 21-40 (January 1985).
12. J. R. Rossignac and A. A. G. Requicha, Constant radius blending in Solid Modelling. Computers in Mechanical Engineering 3(1),65–73 (July 1984).
13. M. J. Pratt, Synthesis of an optimal approach to form feature modelling. ASME Computer and Engineering Conference, San Francisco, California, August 1–3, 1988.
14. J. J. Shah, P. Sreevalsan, M. T. Rogers, R. Billo and A. Mathew, Current Status of Features Technology, Report R-88-GM-04.1, CAM-I Inc., Arlington, TX, November 1988.
15. T. Tomiyama and P. J. W. ten Hagen, Representing Knowledge in Two Distinct Descriptions, Extensional vs Intensional. Report CS-R8728, Center for Mathematics and Computer Science, P.O. Box 4079, 1009 AB Amsterdam, The Netherlands, June 1987.
16. A. A. G. Requicha, Mathematical models of rigid solid objects. Tech. Memo. No. 28, Production Automation

Project, Univ. of Rochester, November 1977. (Reports of the Production Automation Project are no longer available from the University of Rochester, but may be obtained from CPA, 304 Kimball Hall, Cornell University, Ithaca, NY 14853.)
17. L. K. Kyprianou, Shape classification in computer aided design Ph.D. dissertation, Christ's College, University of Cambridge, U.K., July 1980.
18. A. A. G. Requicha, Representation of tolerances in solid modeling: Issues and alternative approaches. In Solid Modelling by Computers, M. S. Pickett and J. W. Boyse (Eds.), Plenum Press, New York, 3–22 (1984).
19. M. J. Pratt and P. R. Wilson, Requirements for the support of Form Features in a Solid Modelling System. Report No. R-85-ASPP-01, CAM-I Inc., Arlington, TX, 1985.
20. M. R. Henderson, Extraction of feature information from three dimensional CAD data. Ph.D. Dissertation, Purdue University, May 1984.
21. J. R. Dixon and C. L. Dym, Artificial intelligence and geometric reasoning in manufacturing technology. Applied Mechanics Reviews 39(10), (October 1986).
22. A. A. G. Requicha and J. Vandenbrande, Automatic process planning and part programming. Institute for Robotics and Intelligent Systems, Report IRIS 217, University of Southern California, Los Angeles, April 1987.
23. V. C. Lin, D. C. Gossard, and R. A. Light, Variational geometry in computer aided design. ACM Computer Graphics 15(3), 171–177, (August 1981).
24. A. P. Ambler and R. J. Poppelstone, Inferring the positions of bodies from specified spatial relationships. Artificial Intelligence 6, 157–174 (1975).
25. D. Gossard, R. P. Zuffante and H. Sakurai, Representing dimensions, tolerances, and features in MCAE systems. IEEE Computer Graphics and Applications 8(2), 51–59 (March 1988).
26. J. R. Rossignac, Constraints in Constructive Solid Geometry. Proceedings 1986 Workshop on Interactive 3D Graphics, University of North Carolina, Chapel Hill, NC 27514, F. Crow and S. M. Pizer, Eds., ACM Press, pp. 93–110, October 23–24, 1986. Also available as IBM Research Report RC 12356, IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY, September 1986.
27. J. R. Rossignac and H. B. Voelcker, Active zones in CSG for accelerating boundary evaluations, redundancy elimination, interference detection, and shading algorithms. ACM Transactions on Graphics 8(1), 51–87 (January 1989).
28. K. J. Weiler, The radial edge structure: A topological representation for non-manifold geometric modeling. In Geometric Modeling for CAD Applications, M. Wozny, H. McLaughlin and J. Encarnacao (Eds.), Springer-Verlag, 37–68, (May 1986).
29. J. R. Rossignac and M. A. O'Connor SGC: A dimension-independent model for pointsets with internal structures and incomplete boundaries, IBM Research Report RC14340, IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY, January 89. Also in "Geometric Modeling for Product Engineering," Proceedings of the 1988 IFIP/NSF Workshop on Geometric Modelling, Rensselaerville, NY, September 18–22, 1988. M. Wozny, J. Turner and K. Preiss, (Eds.), North-Holland, The Netherlands, 145–180 (1989).
30. G. Vanecek and D. Nau, Non-regular decomposition: An efficient approach for solving the polygon intersecting problem. Proc. Symposium on Integrated and Intelligent Manufacturing at the ASME Winter Annual Meeting, 271–279, 1987.
31. R. B. Tilove, A null-object detection algorithm for Constructive Solid Geometry. COMM ACM 27(7), 684–694, (July 1984).
32. L. R. Nackman, M. A. Lavin, R. H. Taylor, W. C. Dietrich and D. D. Grossman, AML/X: a programming language for design and manufacturing. Proceedings of the IEEE Fall Joint Computer Conference, Dallas, TX, pp, 145–159, November 2–6, 1986.
33. A. A. G. Requicha and J. Vandenbrande, Form features for mechanical design and manufacturing. Report IRIS#244, Computer Science Department, University of Southern California, Los Angeles, CA 90089-0782, October, 1988.
34. S. D. Roth, Ray casting for modeling solids. Computer Graphics and Image Processing 18(2), 109–144 (February 1982).
35. R. B. Tilove, A. A. G. Requicha and M. R. Hopkins, Efficient editing of solid models by exploiting structural and spatial locality. Tech. Memo. N. 46, Production Automation Project, Univ. of Rochester, May 1984. (Reports of the Production Automation Project are no longer available from the University of Rochester, but may be obtained from CPA, 304 Kimball Hall, Cornell University, Ithaca, NY 14853.)
36. A. R. Halbert, S. J. P. Todd, and J. R. Woodwark, Generalizing active zones for set-theoretic solid models. Computer Journal (UK) 32(1), 86–89 (February 1989).
37. R. B. Tilove, Exploiting spatial and structural locality in geometric modelling. Tech. Memo. No. 38, Production Automation Project, University of Rochester, October 1981. (Reports of the Production Automation Project are no longer available from the University of Rochester, but may be obtained from CPA, 304 Kimball Hall, Cornell University, Ithaca, NY 14853.)
38. R. B. Tilove, Set membership classification: A unified approach to geometric intersection problems. IEEE Trans. on Computers C-29(10), 874–883 (October 1980).
39. J. R. Woodwark and K. M. Quinlan, Reducing the effect of complexity on volume model evaluation. Computer-Aided Design 14(2), 89–95 (1982).
40. J. R. Woodwark, Eliminating redundant primitives from set-theoretic solid models by a consideration of constituents. IEEE CG&A, 38–47 (May 1988).
41. S. A, Cameron and J. R. Rossignac, Relationship between S-bounds and Active Zones In Constructive Solid Geometry. IBM Research Report PC14246, T. J. Watson Research Center, Yorktown Heights, NY, December 1988. To appear in the proceedings of the International Conference on the Theory and Practice of Geometric Modelling, FRG, October 3-7, 1988.
42. F. Arbab. Set models and Boolean operations for solids and assemblies. Technical Report CS-88-52, Computer Science Department, University of Southern California, Los Angeles, CA, 90089-0782, July 1985.
43. J. R. Rossignac, Blending and Offsetting Solid Models. Tech. Memo. No. 54 (Ph.D. Dissertation), Production Automation Project, Univ. of Rochester, June 1985. (Reports of the Production Automation Project are no longer available from the University of Rochester, but may be obtained from CPA, 304 Kimball Hall, Cornell University, Ithaca, NY 14853.)
44. J. Serra, Image analysis and mathematical morphology Academic Press, New York (1982).