A Semantic Framework for Flexible Feature Validity Specification and Assessment

Rafael E. Bidarra
José C. Teixeira

Departamento de Matemática
Universidade de Coimbra
COIMBRA
PORTUGAL

ABSTRACT

Feature-based modeling is being given an increasing significance in the development of computer-aided design environments, due to its expressive power in capturing various aspects of designer intent. Integrity problems arise, however, when the specification of complex shapes requires several features to overlap or interact. In this paper we address the problem of form feature behavior representation for such situations, developing a semantic model that captures the actual contribution of each feature in the overall model shape. Within this semantic framework it becomes possible to establish naturally and flexibly the desired validity conditions for each feature class, enhancing a modeling system with the ability for permanently monitoring each feature's conformity with that original specification.

1. INTRODUCTION

Feature-based modeling promises an enormous potential improvement for the next generation of CAD systems, providing a high-level building block suitable for raising the so-called *level of intelligent assistance* they are expected to integrate. These are required to incorporate powerful reasoning mechanisms that further automate the task of creating and analyzing product designs. Form feature technology provides the ability to associate engineering significance to certain generic shapes intentionally introduced in the product model by the designer (Shah, 1991a).

Although considerable progress is being achieved in setting up the fundamentals of form feature concept and role (Pratt, 1988) (Shah, 1991b), as well as in developing for them suitable representation schemes and specific application prototypes (Luby et al., 1986), (van Emmerik and Jansen, 1989), (Shah et al., 1990), there is still a great deal to be done before real world, often complex shaped, products can be fully modeled in an integrated feature-based modeling environment (Bronsvoort and Jansen, 1993).

One of the most difficult issues faced by current design by features research, as noted by Dixon et al. (1990), is the ability to cope with complex and combined shapes, that directly express some more elaborated functionality intended by the designer. These are often modeled by means of overlapping form features, that are intentionally inserted to produce *feature interaction* phenomena (Pratt, 1987). Serious problems with explosive combinatorial possibilities disallow any extensive expansion of the modeling system's feature libraries; therefore, provision should be made in these systems to incorporate reasoning mechanisms that promptly recognize such situations and validate features involved in them (Bidarra and Teixeira, 1993a).

In this paper we are mainly concerned with presenting a semantic framework that captures the actual behavior of form features involved in arbitrary interactions. We first extend the definition of interaction among form features, introducing some basic concepts useful in this generalization (Section 2). A conceptual scheme for the representation of feature behavior is then presented, based on the notion of semantic entities, which provides a two-level structure for an object's form feature semantic model (Section 3). We then elaborate on the specification of feature validity conditions, based on the logical composition of a particular kind of constraints set on semantic entities, and suggest how they can be automatically monitored during the incremental modeling process, in order to permanently assess their conformity with the original designer intent (Section 4).

2. INTERACTIONS AMONG FORM FEATURES

As anticipated in the introduction, the functionality intended by the designer for some part of the model often requires complex shapes to be generated by overlapping several simpler form features. These require the use of modeling operations that create, insert or modify form features to yield *feature interactions*. On the other hand, if an interaction between two features may be intentionally introduced by the designer, it happens sometimes that, as an indirect, eventually unanticipated, side-effect interaction phenomena cause a particular feature to cease or

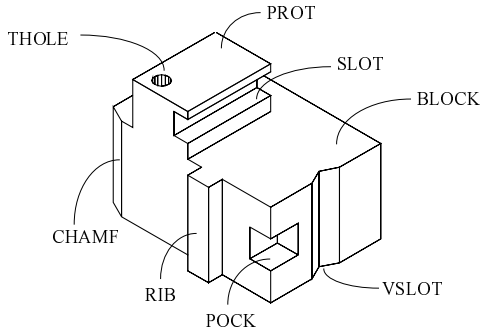

FIGURE 1. MODEL WITH FEATURES IN CANONICAL FORM

modify its original contribution to the shape of the model, overriding some intended behavior previously specified (Bidarra and Teixeira, 1993b). In order to further understand the nature of interaction phenomena, we introduce in this Section some basic concepts that will show of crucial importance in the subsequent development.

2.1. Feature Associated Volume (FAV) and Boundary (FAB)

Provided that we are dealing with volumetric form features, we often need to refer to the region of the space comprised within each feature instance; hereafter, we will designate it as the *Feature Associated Volume*, FAV. Intuitively, the FAV expresses the volume of material to be removed from or added to the model with the insertion of that feature (with subtractive or additive signal, respectively). Analogously, we define the *Feature Associated Boundary*, denoted FAB, as the set of boundary points of the FAV. Clearly, not all the points in a FAB belong to the model boundary; for example, the two open ends of a cylindrical through hole are not part of the model boundary. A partition can be established on the FAB that classifies points according to this property:

$$\mathbf{FAB} = \mathbf{FAB}^{+} \cup \mathbf{FAB}^{-}$$

where $\mathbf{FAB}^{+}$ denotes the point subset that contributes to the model boundary.

2.2. Feature Canonical Form

Every feature class should always provide a standard specification for the default behavior of each instance feature in the model. This primary configuration is termed *feature canonical form* and it can be illustrated by the features in the model depicted in Figure 1.

This situation can be characterized by observing that there are no two features of the same signal whose FAV overlap. Indeed, this is a necessary condition for the occurrence of the canonical form.

2.3. Types of Interaction

A feature canonical form occurs, most often, when its FAV intersects only that of one other feature, as is the case of the SLOT, the POCKET and the RIB features in Figure 1: this is designated as *single interaction*. In some situations, however, a feature may exhibit a FAV that intersects a few others while still keeping its canonical form, as exemplified by the THROUGH HOLE in Figure 1; such situations are here termed *multiple interactions*.

2.4. Interaction Extent (IE)

The *interaction extent* of any two features F_1 and F_2 is the region of the space (whatever its dimensionality) defined by the intersection of the respective FAVs; we shall denote it by $IE(F_1,F_2)$.

The notion of interaction extent is so fundamental that we can ground on it our definition of interaction between two features:

> ***Definition**: given any two features F_1 and F_2, we say they are in interaction, and denote it by $F_1 \leftrightarrow F_2$, iff their interaction extent is non empty, i.e.,*

$$\mathbf{F_1} \leftrightarrow \mathbf{F_2} \quad \Leftrightarrow \quad \mathbf{IE(F_1,F_2)} \neq \varnothing$$

2.5. Interaction Mode

Another useful distinction can be established according to the dimensionality of the interaction extent between any two interacting features, F_1 and F_2. We have a *boundary interaction* whenever

$$\mathbf{IE(F_1,F_2)} \subset \mathbf{FAB(F_1)} \wedge \mathbf{IE(F_1,F_2)} \subset \mathbf{FAB(F_2)}$$

and a *volume interaction* otherwise.

2.6. Feature Interaction Graph (FIG)

The above definition of interaction generalizes and unifies the notions of *adjacency* and *interaction* identified by Pratt (1988) and Shah and Rogers (1988), respectively. Therefore, the interaction relationship provides a powerful basis for a high-level graph representation of a feature model; we call it the *Feature Interaction Gaph* (FIG) and it is defined as

$$\mathbf{FIG} = \langle \mathbf{F}, \mathbf{I} \rangle$$

where

F is the set of features in the model, and

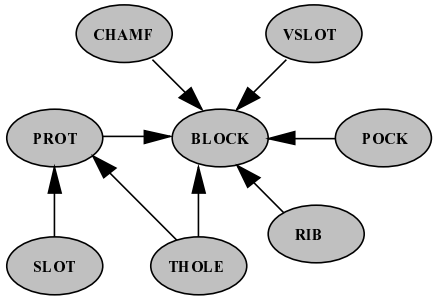

FIGURE 1. FEATURE INTERACTION GRAPH FOR THE MODEL OF FIGURE 1

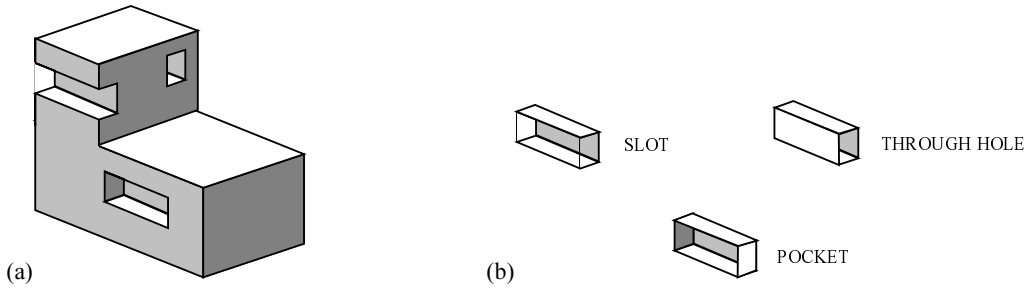

FIGURE 3. MODEL WITH THREE CONGRUENT SUBTRACTIVE FEATURES (a) AND THEIR DISTINCTIVE FAB CONFIGURATIONS (b)

I is the subset of $\mathbf{F}^2$ that contains all the pairs of features of **F** that actually interact in the model.

The edges in the FIG can be oriented so that for each interaction the active feature points to the passive one, i.e., the feature that suffers the interaction. As an example, the FIG of the model in Figure 1 is shown in Figure 2.

3. THE REPRESENTATION OF FEATURE BEHAVIOR

The functional and technological conotation of every form feature class is one of the most promising skills for the development of product modeling systems (Shah, 1991a). Its sound capture in the product model, however, is crucial if the designer intent is to be kept consistent throughout the design process. This poses considerable demands on the system's ability to express and represent the actual behavior of each feature present in the model.

3.1. Semantic Entities

The usual primary classification of features according to its additive or subtractive nature proves insufficient to distinguish the specific behavior of different feature classes, even among their canonical instance features. This is clearly shown by the three subtractive features in the model of Figure 3.(a), which also present congruent FAVs.

This distinction, however, can be accomplished if we carefully analyse the FAB of each feature, identifying on it the various subsets with specific roles or contributions to the feature global behavior: Figure 3.(b). In particular, we should notice the presence in the model boundary of only some of them, said to exhibit a *positive status*, while the others, with a closure character, are said to have a *negative status*.

We designate each of these subsets a *semantic entity*, according to the following

> ***Definition**: a semantic entity of a feature is a collection of sectors of its boundary that plays a specific and individualized role in the behavior desired for the respective feature class, as expressed by its canonical form.*

Any feature class can, thus, describe its specific behavior through a set of semantic entities. In Figure 4 we show an example of these sets for the rectangular SLOT (a) and rectangular BLOCK (b) features.

From this example, two observations can be made:

i) the generation of feature interactions always produces a new further decomposition of the involved FABs, eventually changing the status of some resulting elements (or sectors); as a consequence of this process, a semantic entity may become composed of sectors of both status, as shown in (b), for instance, by the **top(BLOCK)** semantic entity, which exhibits two positive status elements (forming the so-called positive subentity $\mathbf{top}^{+}\mathbf{(BLOCK)}$) and one negative element (that integrates the negative subentity $\mathbf{top}^{-}\mathbf{(BLOCK)}$);

ii) due to the particular configuration of the interaction extent, it may occur that some sectors of a FAB are shared

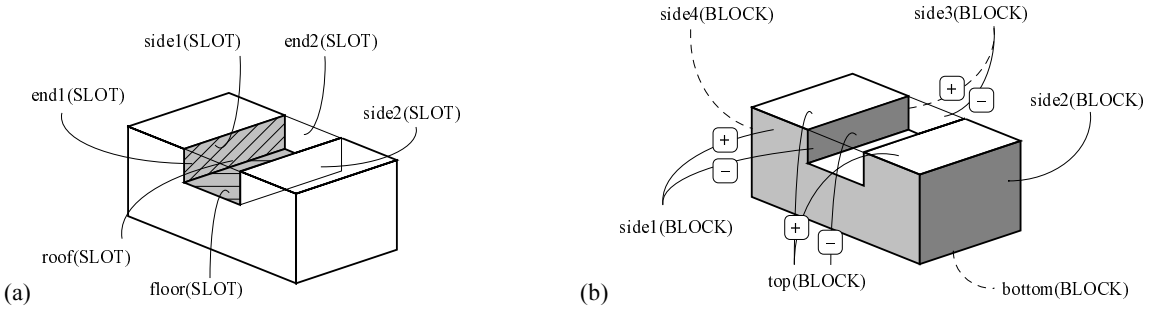

FIGURE 3. SEMANTIC ENTITIES OF A RECTANGULAR SLOT (a) AND A RECTANGULAR BLOCK (b)

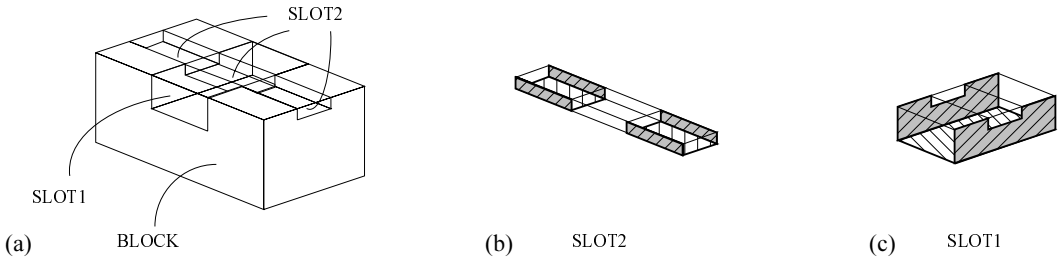

FIGURE 5. INTERNAL DECOMPOSITION OF SEMANTIC ENTITIES OF FEATURES IN MULTIPLE INTERACTION

by (or belong to) semantic entities of distinct features (e.g., the **top_(BLOCK)** and the **roof_(SLOT)** in Figure 4 share the same boundary element).

The internal decomposition of each semantic entity and the classification of its sectors as positive and negative elements provide us with the appropriate descriptive tools to overcome the difficulties resulting from the virtually unlimited number of configurations that arise from generalized feature interactions. These skills can be stressed with the example model of Figure 5.(a), obtained from the previous one inserting a new feature, SLOT2, in double interaction.

As shown in (b) and (c), the non-standard topology of both slots (one of which is even disconnected), is captured and structured within the respective semantic entities (whose positive elements are shown dashed). These will, therefore, form the basis for the development of validity condition for every feature class, in Section 4.

3.2. Form Feature Semantic Model

As mentioned in Subsection 2.6, we can describe a feature model, from a high level of abstraction, as a graph (the FIG), expressing the interaction relationship among the form features that integrate the model. Although surprisingly simple, the FIG provides already a global perception of the overall structure of the feature model, as depicted in Figure 6.

On the other hand, we stressed the crucial role played by the notions of feature associated volume and boundary, interaction extent and semantic entity, which are so closely related. These concepts can be conveniently described in a second lower level of abstraction, by means of a structured layer of cellular entities. This level emerges as a common basis for the unification of the FAV (3-cells), the semantic entities (2-cells) and the interaction extent (potentially, cells of all dimensionalities). Therefore, it should not be confused with any type of cellular representation scheme used in solid modeling, although some analogy can be found through the *boundary of* relationship between cells of dimension n and n+1 (with n=0, 1, 2).

The global Form Feature Semantic Model integrates both the FIG (at the semantic level) and the structured cellular level, as depicted in the diagram of Figure 7.

For each interaction relationship recorded at the semantic level there is one interaction extent, whose counterpart at the cellular level consists of a collection of n-cells (with n=3 for volumetric interaction and n<3 for boundary interaction). On the other hand, the semantic entities and the FAV of each feature have their corresponding expression in the cellular level as collections of 2- and 3-cells, respectively. The hierachical structure of this level provides a unified basis for the consistent access and reference to the actual situation of both interaction extents and semantic entites, essential for the on-line monitoring of feature manipulation operations and the effective detection of critical interactions. Thus, for example, when a 3-cell of a FAV has to be decomposed to yield one or more 3-cells for interaction extents, its boundary (made up of 2-cells) is consistently decomposed, automatically updating the contents of the respective semantic entities.

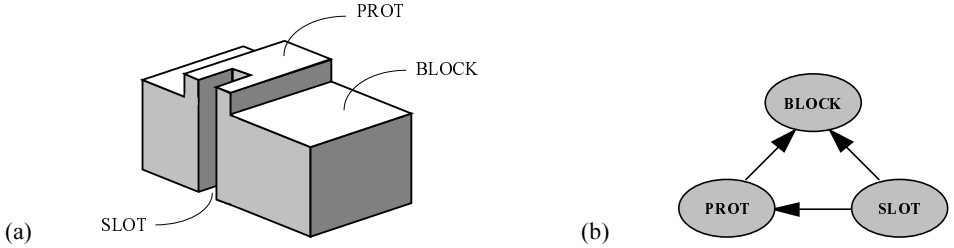

FIGURE 6. FORM FEATURE MODEL (a) AND THE CORRESPONDING FIG (b)

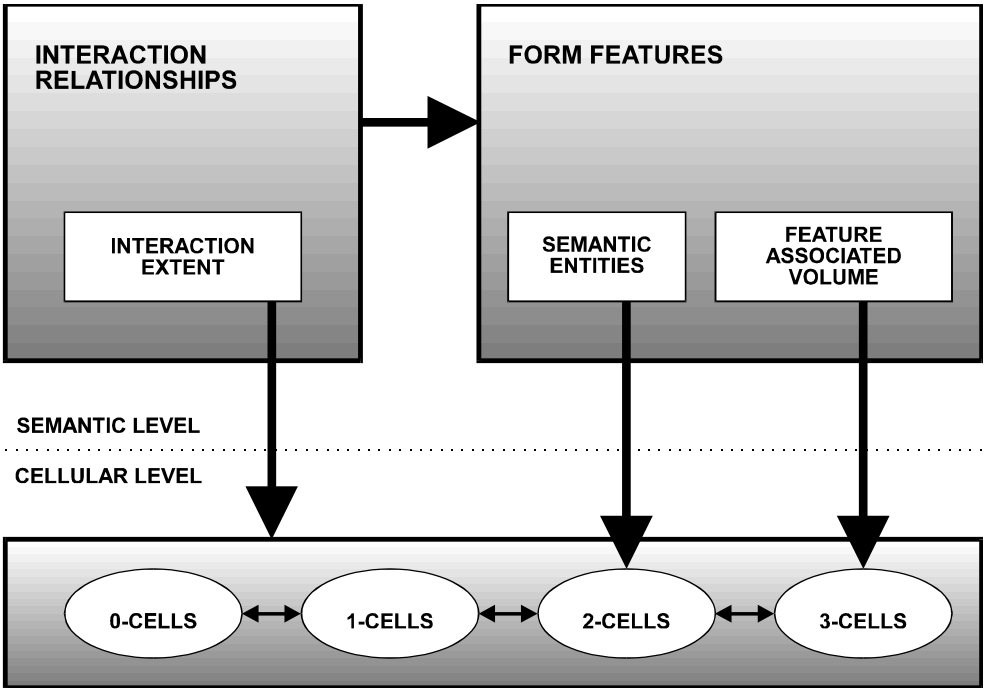

FIGURE 7. GLOBAL FORM FEATURE SEMANTIC MODEL

In Figure 8 we exemplify the semantic model structure for the object of Figure 6.

The interaction extent for the boundary interaction between the BLOCK and the PROT is highlighted, at the cellular level, by dashed 2-cells. The correspondence of semantic entities in the cellular level, intentionally omitted in Figure 8 for clarity, is presented in

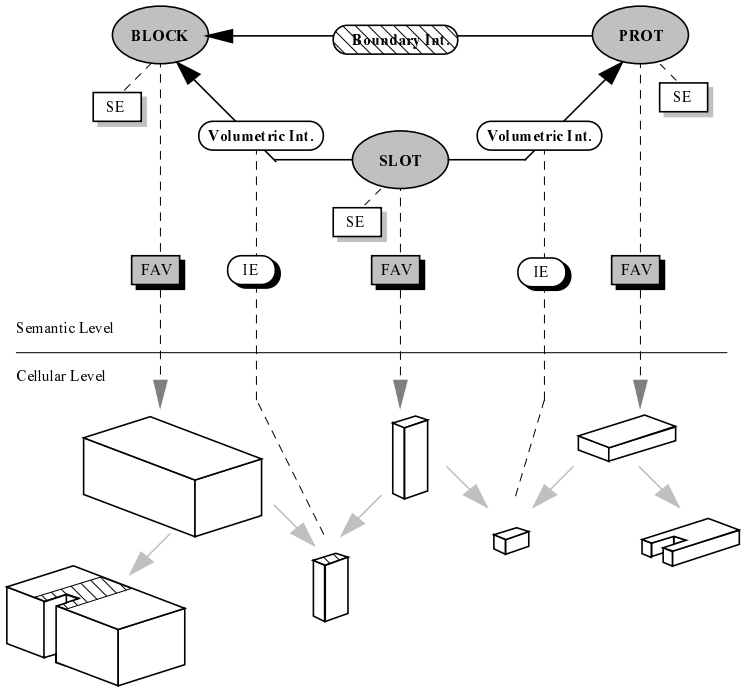

FIGURE 8. SEMANTIC MODEL FOR THE OBJECT OF FIGURE 6

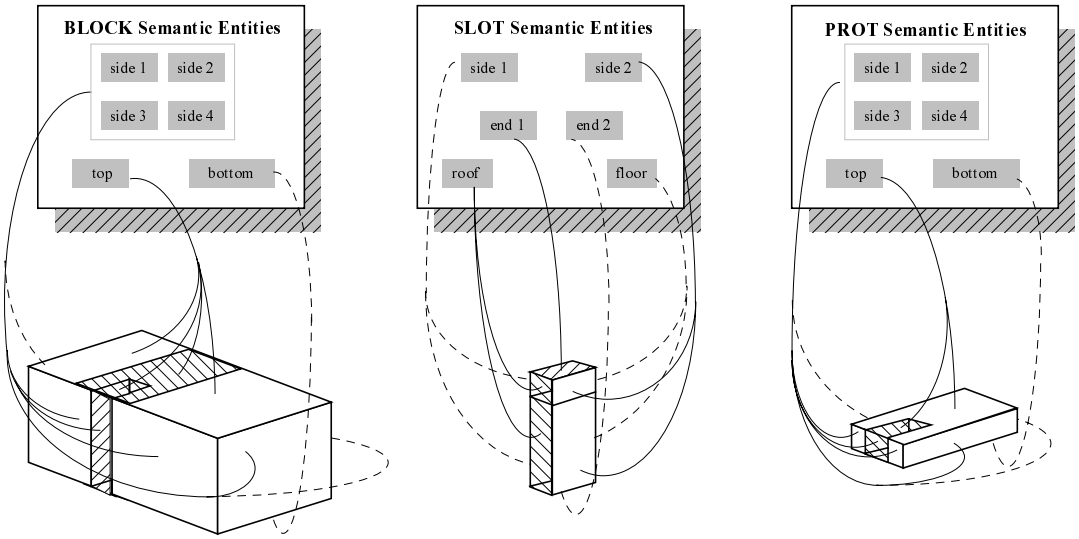

FIGURE 10. SEMANTIC ENTITIES AND THEIR CORRESPONDENCE IN THE CELLULAR LEVEL

Figure 9, where we can distinguish their internal partition in subentities (the negative ones being dashed).

Although, for the sake of clarity, cells appear in the previous two Figures as completely disjoint, one should keep in mind that they are actually all *glued* together, some of them (those that integrate an interaction extent) being shared by more than one FAV. Correspondingly, some 2-cells are shared by more than one semantic entity, as, for example the one shaded in Figure 10, which is shared by both the **roof(**SLOT**)** and one **side(**BLOCK**)**.

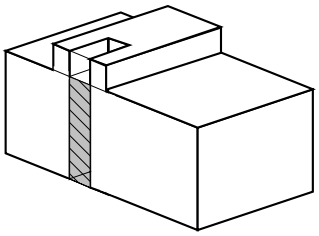

FIGURE 10. SHARED CELLS AT THE CELLULAR LEVEL

4. THE SPECIFICATION OF FEATURE VALIDITY

Form feature validity checking is a key issue in any feature-based modeling environment. This problem is particularly important when we want the system to automatically detect and signal situations that violate the integrity of the previous specification of the designer intent. With this objective, we propose the system to monitor every feature operation (insertion, removal, modification, etc.) to assess the conformity of the result with the original semantic specification provided in the respective feature class.

All instance features of a particular feature class, although significantly distinct in their FAB configurations due to interactions, are expected to preserve always some essential behavior of that class. This semantic description can be established as a set of validity conditions, to be verifyed and obeyed permanently by each instance feature present in the model.

4.1. Semantic Constraints

The key for specifying feature validity conditions lays in settling a set of constraints - here called *semantic constraints* -, each of which requires a semantic entity to exhibit a particular behavior, as a consequence of the type of internal decomposition it allows.

The first step in this goal consists of specifying a default or *canonical status* for each semantic entity in a feature class description; in this way, for instance, a rectangular slot class should assign a positive canonical status for the **sides** and **floor** semantic entities, as well as a negative one for the **ends** and **roof** semantic entities (see Figure 4.(a)). The role of this specification, however, is essentially indicative, in that it makes no assumption on the actual partition of the semantic entity. In turn, this can actually be constrained by the introduction of a predicate that specifies *how* the semantic entity is allowed to be decomposed.

Let us first introduce the generic Prolog-like predicate

s_entity(Elements,Entity,Status,Feature)

that provides the access to the **Elements** of **Status** in the semantic **Entity** of an instance **Feature**. The **Status** specified in a query, if constrained, can be either absolute (positive, **+**, or negative, **-**) or relative to the **Feature** semantics (i.e., canonical, **c**, or improper, **i**).

The flexibility of this predicate becomes clearly illustrated with a few examples:

s_entity(Elements,floor,+,SLOT**)**

- obtains the **Elements** of **floor**$^{+}$**(**SLOT**)**;

s_entity(Elements,_,-,POCKET**)**

- obtains all negative **Elements** of all semantic entities of POCKET, i.e., the **FAB**$^{-}$ **(**POCKET**)**;

s_entity([],end1,i,THOLE)

- verifies if the semantic entity **end1** of THOLE has no improper elements, i.e., if it is only composed of canonical elements.

This last example serves to show a first kind of semantic constraint, namely that of requiring a semantic entity to be intact or **complete**, i.e., integrating only elements in canonical status. This means that if the feature undergoes some decomposition due to an interaction, that semantic entity is required to keep active and completely contributing to the global feature behavior, with no missing (or improper) fragments. The definition of the **complete** semantic constraint is, thus,

```
complete(Entity,Feature)  ←
          s_entity([],Entity,i,Feature).
```

Another, more loose, constraint requires the semantic entity to be only **present** in the model boundary, i.e., integrating some canonic elements, according to the definition

```
present(Entity,Feature)  ←
    s_entity([_|_],Entity,c,Feature).
```

This allows for arbitrary interactions to partially corrupt a semantic entity, provided that it still keeps some original canonical subset contributing to the overall feature behavior. Naturally, this semantic constraints could be established in conjunction with some kind of quantification for the actual canonical subset, e.g., based on geometric criteria.

Some other constraint predicates can be defined, based on the topological notions of adjacency and accessibility. Among these, the requirement for the connectivity of a semantic entity may be used to prevent a form feature from being split into several canonical components due to an interaction with some subtractive feature, as shown by the through hole of the model in Figure 11.

The definition of such a constraint can be stated as follows:

```
connected(Entity,Feature)  ←
   s_entity(Elements,Entity,c,Feature),
          forall(
                 (element(X,Elements),
                  element(Y,Elements)),
                 accessible(X,Y)
                 ).
```

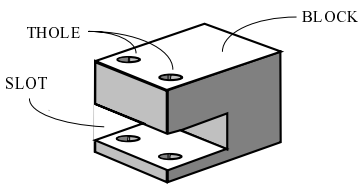

FIGURE 11. THROUGH HOLE DISCONNECTED BY AN INTERACTION WITH A SUBTRACTIVE FEATURE

4.2. Validity Conditions

If each semantic constraint establishes some local partial behavior for a feature, it seems both natural and powerful to associate to each feature class a set of semantic constraints, combined to form a logical expression that describes the overall behavior for all its instance features. This expression is called the *feature class validity conditions*. Whenever an instance feature satisfies these conditions in the model, we say it presents semantic completeness; otherwise, it diverges in some fashion from the original class specification and must, therefore, be considered as an invalid feature.

Due to the incremental nature of the design process, semantically invalid features can emerge as an indirect, eventually unanticipated, result of feature interactions. Such situations should always be detected and signaled by the system, although it is the designer responsibility to allow their maintenance in the model. In this case, they should be considered as intentional features, just reflecting some designer intention on the respective functionality, temporarily overriden, which he should restore latter on (Rossignac, 1990).

It should be remarked that the notion of form feature validity here developed is closely tied with the exact expression of the intentional behavior that form features actually exhibit in a model. Naturally, the choice for some particular semantic specification is, most often, domain dependent and it is not, therefore, our purpose to elaborate here on a taxonomy for such a scheme, if it ever exists. Instead, we propose a flexible framework for the semantic expression of any validity conditions, suitable for the effective configuration of the system's feature library according to designer specific requirements.

We now illustrate with a simple example the specification of a set of validity conditions for a particular SLOT subclass. The set of conditions

```
complete(floor,SLOT) ∧

present(side1,SLOT)∨present(side2,SLOT) ∧

complete(end1,SLOT) ∧

complete(end2,SLOT) ∧

complete(roof,SLOT)
```

would accept as valid instances the SLOT features in the models of Figure 12.(a), whereas those of (b) exhibit some interaction that makes incomplete either the **floor** or the **end** semantic entities, yielding invalid situations.

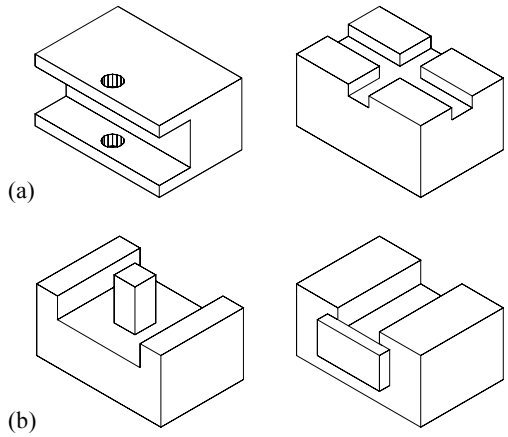

FIGURE 12. MODELS WITH (a) VALID AND (b) INVALID SLOT INSTANCES IN MULTIPLE INTERACTION

5. CONCLUSIONS

Complex-shaped solid objects present strong demands on a feature-based modeler's ability to cope with form feature interaction phenomena. The nature of feature interactions, the semantic interpretation of interacting features, and the specification of non-canonic feature validity are some of the current open issues addressed in this paper.

A semantic framework for the dynamic interpretation of form feature behavior was developed, based on the notion of semantic entity. This has shown to bring together the semantic expressive power of form features and the flexibility of their behavior throughout interaction phenomena.

A mechanism for feature validity maintenance was proposed, that relies on the logic composition of semantic constraints. The need for enhancing the conventional manipulation operations in any feature-based modeling system with validity monitoring and assessment is crucial if one wishes to achieve a finer tuning with the designer throughout the design process.

Our current research goals include the development of reasoning mechanisms to cope with such requirements for feature insertion, removal and geometric modification operations.

REFERENCES

Bidarra, R., Teixeira, J.C., 1993a, "Mecanismos de Raciocínio na Manipulação de Modelos Baseados em Características de Forma", *Proceedings of V Encontro Português de Computação Gráfica*, Aveiro, Portugal.

Bidarra, R., Teixeira, J.C., 1993b, "Intelligent Form Feature Interaction Management in a Cellular Modeling Scheme", *Proceedings of the II ACM/IEEE Symposium on Solid Modeling and Applications*, Montreal, Canada.

Bronsvoort, W.F., Jansen, F.W., 1993, "Feature Modelling and Conversion - Key Concepts to Concurrent Engineering", *Computers in Industry*, no. 21, pp. 61-86.

Dixon, J.R., Libardi, E.C., Nielsen, E.H., 1990, "Unresolved Research Issues in Development of Design-with-Features Systems", *Geometric Modeling for Product Engineering*, Wozny, M., Turner, J., and Preiss, K. (Eds.), North-Holland, The Netherlands.

van Emmerik, M.J.G.M., Jansen, F.W., 1989, "User Interface for Feature Modelling", *Computer Applications in Production and Engineering*, Kimura, F. and Rolstadas, A. (Eds.), Elsevier Science Publishers, B.V. (North-Holland), IFIP, pp. 625-632.

Luby, S.C., Dixon, J.R., Simmons, M.K., 1986, "Designing with Features: Creating and Using a Features Data Base for Evaluation of Manufacturability in Castings", *Proceedings of the ASME International Computers in Engineering Conference and Exhibition*, Chicago, IL.

Pratt, M.J., 1987, "Recent Research in Form Features", *Siggraph Course Notes: Advanced Topics in Solid Modeling*, Course #26, Siggraph'87 Conference , Anaheim, CA.

Pratt, M.J., 1988, "Synthesis of an Optimal Approach to Form Feature Modelling", *Proceedings of the ASME International Computers in Engineering Conference and Exhibition*, San Francisco, CA, Vol. 1, pp. 263-274.

Rossignac, J.R., 1990, "Issues on Feature-Based Editing and Interrogation of Solid Models", *Computers & Graphics*, Vol. 14, no. 2, pp. 149-172.

Shah, J.J., et al., 1990, "The ASU Features Testbed: An Overview", *Proceedings of the ASME International Computers in Engineering Conference and Exhibition*, Boston, MA, Vol. 1, pp. 233-241.

Shah, J.J., 1991, "Assessment of features technology", *Computer-Aided Design*, Vol. 23, no. 5, pp. 331-343.

Shah, J.J., 1991, "Conceptual Development of Form Features and Feature Modelers", *Research in Engineering Design*, Vol. 2, pp. 93-108.

Shah, J.J., Rogers, M.T., 1988, "Expert Form Feature Modelling Shell", *Computer-Aided Design*, Vol. 20, no. 9, pp. 515-524.

Shah, J.J., Wilson, P.R., 1989, "Analysis of Design Abstraction, Representation and Inferencing Requirements for Computer-Aided Design", *Journal of Design Studies*, Vol. 10, no. 3, pp. 169-178.