Proceedings of DETC'99
1999 ASME Design Engineering Technical Conferences
September 12-15, 1999, Las Vegas, Nevada

DETC99/CIE-9122

HISTORY-INDEPENDENT BOUNDARY EVALUATION FOR FEATURE MODELING

Rafael Bidarra and Willem F. Bronsvoort
Faculty of Information Technology and Systems
Delft University of Technology
Zuidplantsoen 4, NL-2628 BZ Delft, The Netherlands
Email: (Bidarra/Bronsvoort)@cs.tudelft.nl

ABSTRACT

Most current feature modeling systems strongly rely on a history-based interpretation of the feature model, in order to maintain its evaluated boundary representation. This dependency on the model history is undesirable, as it forces the user of the modeling system to reason in terms of a strict chronological feature creation order. Moreover, re-evaluation of the boundary representation, as performed in such systems, has a computational cost proportional to the size of the model history. Such drawbacks suggest that current feature modeling systems are still too tied to conventional geometric modeling techniques.

In this paper, it is argued that to overcome the drawbacks mentioned above, a *declarative feature model* is required, whose structure is dynamically adapted as modeling operations create or modify relations among its features. Operations performed on this feature model can then be efficiently propagated to an evaluated *non-manifold* geometric representation, without invoking model history considerations. The paper describes how such a geometric model –the so-called *Cellular Model*– can be maintained throughout model evolution. For each modeling operation, this is achieved in two phases. First, the Cellular Model is incrementally re-evaluated. Second, the Cellular Model is interpreted, according to the feature information stored in its cellular entities and the current dependencies among the features. The advantages of the use of this history-independent boundary evaluation, implemented within the semantic feature modeling approach, are illustrated with some modeling examples.

1. INTRODUCTION

Boundary models have long been used in feature modeling for a variety of applications. For example, manufacturing and process planning applications use a boundary model for their specific analysis (Talwar and Manoochehri 1994, Gupta et al. 1995). Moreover, both visualization and interactive manipulation of the model geometry benefit from these evaluated boundary representations (Versluis et al. 1997). More recently, embedding feature information in boundary model entities has received increasing research attention. These proposals, surveyed in (Bidarra et al. 1998), are a valuable alternative to maintaining sets of references to those entities in separate data structures. In particular, such representations allow powerful schemes for the analysis and maintenance of feature semantics throughout the modeling process (Bidarra and Bronsvoort 1999a).

Almost all current feature modeling systems are parametric, history-based modeling systems, using a boundary representation as main geometric model. Examples of such systems are the commercial systems Pro/ENGINEER (Parametric 1998), MicroStation Modeler (Peters 1997), I-DEAS Master Series (SDRC 1998) and Autodesk Mechanical Desktop (Autodesk 1998), and the academic systems of Shah et al. (1990) and Chen and Hoffmann (1995).

History-based modeling systems are procedural systems which, together with the evaluated boundary representation, keep track of information about each modeling operation performed, e.g. the type of feature created, its parameter values, and its model references for positioning. Each new feature is positioned relative to boundary entities of the evaluated model, obtained from previously created features. The stored sequence of modeling operations, called the *model history*, completely determines the resulting boundary representation. Creation of a feature produces in the evaluated boundary model the shape imprint characteristic of its feature type.

Feature instances can be modified by specifying new values for their parameters, or be deleted from the model. This is done by modifying, or deleting, the respective feature creation operation in the model history, after which a new boundary model is evaluated by sequentially re-executing the operations in the

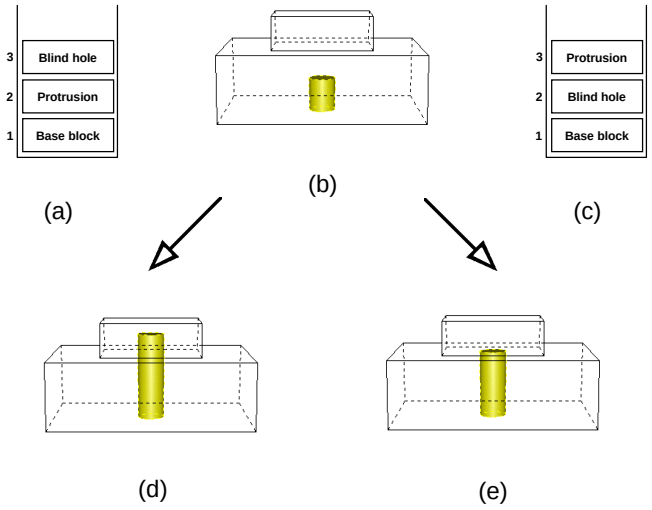


Figure 1 – Set operations problem in history-based boundary re-evaluation

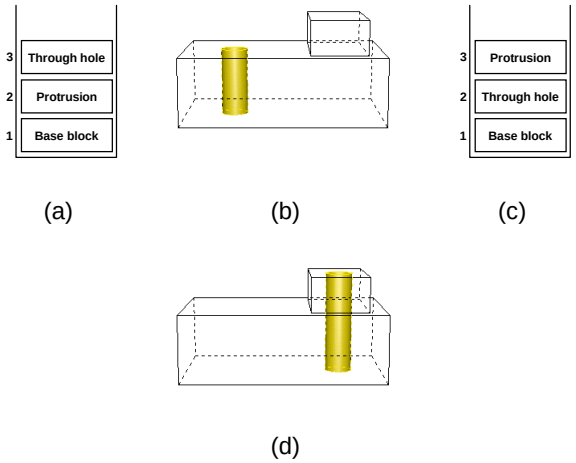


Figure 2 – Entity reference problem in history-based boundary re-evaluation

modified history. With this scheme, variants of a feature model can easily be created.

History-based feature modeling systems suffer from a number of shortcomings with regard to the modeling process. These have been surveyed in (Bidarra and Bronsvoort 1999b), where a new approach, called *semantic feature modeling*, has been proposed that overcomes them. In this, a *non-manifold* geometric representation, called the *Cellular Model*, is used.

In this paper, we concentrate on the specific problems related to the evaluation and maintenance of a boundary representation for feature models (Section 2). A declarative feature model is introduced, in which these problems are overcome (Section 3). The Cellular Model and its incremental re-evaluation process are presented (Section 4), and the computational cost of this process is discussed (Section 5). The scheme for interpretation of the Cellular Model is discussed, and history-independent precedence criteria presented (Sections 6 and 7). Finally, a number of examples illustrates the usefulness of this scheme in practice (Section 8).

2. PROBLEMS WITH HISTORY-BASED BOUNDARY RE-EVALUATION OF FEATURE MODELS

Boundary re-evaluation in history-based modeling systems has, at least, three major shortcomings that will now be identified and illustrated with typical examples. They all have a common cause: a strong dependency on the chronological order of feature creation.

Computational cost

The first shortcoming is that re-executing the whole model history after modifying or deleting a feature has a computational cost that is proportional to the number of features in the model. Several methods have been devised to improve this, e.g. storing the intermediate evaluated model between each history step. Then, only the history steps after the modified, or deleted, operation need to be re-executed. However, storing intermediate models between all history steps requires a considerable amount of storage space, proportional to the square of the model history size. An alternative improvement is to store only the deltas between history steps, and to rollback to the state from which the model needs to be re-evaluated. This requires less storage space, but more computation time again. In any case, the sequence of history steps re-executed almost always includes more features than those actually modified by the operation in question.

Non-associative set operations

The second shortcoming is that history-based re-evaluation of the boundary model does not always guarantee that the evaluated model matches the specified parameters of features that overlap. This is illustrated in the model of Figure 1.b, which consists of a base block, a blind hole and a protrusion. Because the blind hole and the protrusion do not overlap, the history of this model could be either that in Figure 1.a or that in Figure 1.c. However, if the blind hole depth is increased, so that it now overlaps with the protrusion, different models will result for the two histories: in case (a), re-execution of the history produces a blind hole with the expected depth (Figure 1.d), whereas, in case (c), the blind hole will be "truncated" by the protrusion, its depth becoming equal to the block height (Figure 1.e). The problem is caused here by the static precedence order upon which model re-evaluation is based: the *chronological feature creation order*. The resulting models are different because the evaluation process uses two non-associative set operations according to the nature of a feature being processed: union for additive features, and difference for subtractive features. The order in which these are executed determines the result: performing the union of the protrusion as last operation, prevents

the blind hole to exhibit its nominal depth in the model of Figure 1.e.

Entity references in the model history

The third shortcoming of history-based re-evaluation of the model is that it cannot always process feature modification operations such as, for example, feature re-attachment or re-positioning relative to other model entities. This is illustrated in the example of Figure 2. The model consists of a block, a through hole and a protrusion, see Figure 2.b. The history of this model could be either that in Figure 2.a or that in Figure 2.c. In the first case, re-attachment of the through hole to the top of the protrusion and the bottom of the block, see Figure 2.d, can be achieved by modifying the corresponding attach reference of the through hole in the history, and re-executing that history step. However, if this reference modification would have been made in the model history of Figure 2.c, re-evaluation of the model would not be possible, because the through hole creation cannot be re-executed with a reference to a face (the top of the protrusion) that will be created in the model at a later stage of its history. Evaluation of the boundary model by step-wise re-executing a sequence of operations allows each of them to refer *only* to those boundary entities left there by the previous operation. Therefore, modification of the references in a modeling operation, e.g. when re-attaching a feature to other model entities, is not always possible, because the entities concerned may be *tied to a posterior stage* of the model history.

3. A DECLARATIVE FEATURE MODEL

In this section, a declarative feature model is presented that sets the basis for overcoming the drawbacks identified above. This model was developed within the semantic feature modeling approach, whose main goal is to raise the level of assistance provided to the user of a feature modeling system by maintaining the semantics of all features throughout model evolution (Bidarra 1999).

In this approach, each feature has a well-defined meaning, or semantics, which is specified in the respective feature class by means of a variety of constraint types. The geometry of each feature instance, designated the feature's *shape extent*, accounts for the bounded region of space comprised by its volumetric shape. Moreover, its boundary is decomposed into functionally meaningful subsets, the *shape faces*, each one labeled with its own generic name. Associated to each feature instance is the notion of *feature nature*, indicating whether it represents material added to or removed from the model (respectively *additive* and *subtractive* natures).

Characteristic of semantic feature modeling is that the whole modeling process is uniformly carried out in terms of features and their entities (e.g. faces and parameters), and of constraints among these (the so-called *model constraints*). All modeling actions performed by the user are, thus, effectively

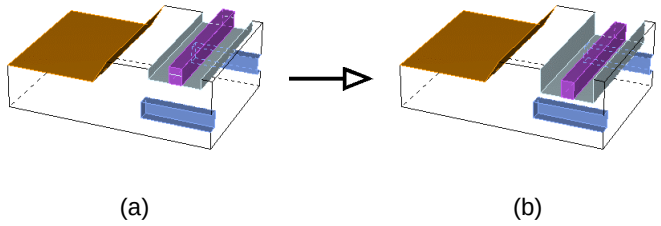


Figure 3 – Example of a dependency relation between features

feature-based, and the same applies to all output, both graphical and textual, generated by the modeling system.

The semantic feature model has a two-level structure, clearly distinguishing *modeling entities* from *entities in the evaluated geometric model*. The former, i.e. the entities on which all modeling operations are performed, are kept in the first level of the model –the so-called *Feature Dependency Graph*–, which contains all feature and model constraint instances, interrelated by *dependency relations*. The second level contains the evaluated geometric representation of the product in the so-called *Cellular Model*. Its entities are kept internal, being only required to "reflect" the geometry that results from the modeling operations performed on the first level. The semantic feature model disposes of mechanisms for maintaining the consistency between the two levels. The Feature Dependency Graph is briefly described in this section; the Cellular Model and its maintenance are elaborated in subsequent sections.

The dependency relation

Instantiation of a new feature requires the user to supply a set of parameter values, aimed at initializing all feature constraints and parameters. Some of these values consist of references to elements of other features (e.g. faces), and are meant to specify how the new feature should be attached and positioned relative to the features already present in the model. Such references are persistent, in the sense that they remain valid as long as the features referred to remain in the model.

Moreover, these references establish a clear dependency among the features in the model. Thus, for example, if a rib is attached to the bottom of a slot, see Figure 3.a, it will be displaced when the depth parameter of the slot is increased, see Figure 3.b. Also, the rib attachment has to be readjusted, or, alternatively, the rib itself also removed, when the slot is removed from the model.

A dependency between two features is unidirectional: one can always distinguish the feature that is determined from the feature that determines. In this sense, again referring to the example of Figure 3, removal of the rib from the model does not present any problem to the slot.

We can therefore state that feature f_1 *directly depends* on feature f_2 whenever f_1 is attached, positioned or, in some other way, constrained relative to f_2. Stated differently, f_1 directly de-

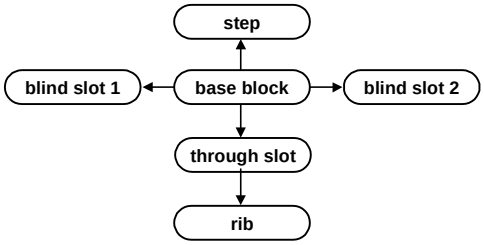


Figure 4 – Feature Dependency Graph of the model in Figure 3

pends on f_2 if some feature constraint of f_1 has a reference to an entity of feature f_2.

By extension, a feature is considered to depend on another feature if the above definition recursively applies between them: feature f_1 *depends* on feature f_2 whenever f_1 directly depends on some feature f_3 that *depends* on f_2. Finally, two features are said to be *independent* if and only if none of them depends on the other.

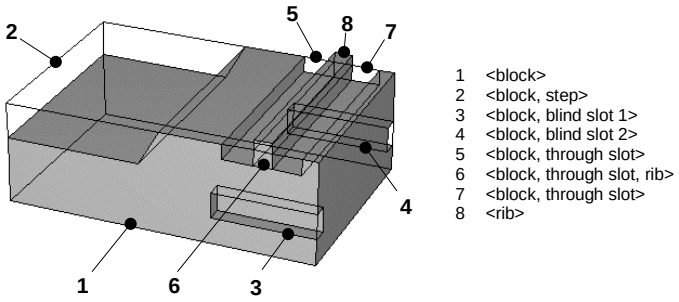
The notion of dependency plays a crucial role in the semantic feature model. It is a dynamic relation among modeling entities, and can thus evolve as these entities are modified, in contrast to the static chronological feature creation order used in most history-based feature modeling systems.

The Feature Dependency Graph

The *Feature Dependency Graph* contains all feature instances in the product model, each of them with its own set of entities (e.g. shape elements, parameters and constraints) (Bidarra and Bronsvoort 1999b). These instances are interrelated by the dependency relation introduced above, yielding a directed acyclic graph structure, consisting of the set of all model entities (feature instances and model constraint instances), and the set of dependency relations among these entities. Each edge represents one dependency relation, and is oriented towards the dependent feature or model constraint. As an example, Figure 4 depicts the Feature Dependency Graph of the model in Figure 3.

The Feature Dependency Graph provides a high-level structure of the feature model. In fact, it contains *all* entities and information required for model manipulation, in a structured way. Interaction between the user of the modeling system and the model takes place in terms of the features and model constraints in the Feature Dependency Graph. Each entity in the Feature Dependency Graph may be queried about its current parameter values and dependencies. Furthermore, each feature node in the graph "knows" about its current global position, as well as its geometry.

All modeling computations are also primarily carried out at this level. For example, an essential step for all modeling operations (except for feature removal operations) is the internal geometric and algebraic constraint solving process, which acts upon entities at this level. When this process is successfully



(b) Cellular Model (c) cell owner lists

Figure 5 – Cellular Model of the part in Figure 3.a

performed, all feature instances in the Feature Dependency Graph have their parameters, position and (shape extent) geometry updated. The solving process also records which features have actually been geometrically modified by a modeling operation. For example, by the slot modification operation in Figure 3, the rib is also displaced, whereas all other features maintain all their parameters and their position.

The Feature Dependency Graph contains no evaluated model geometry, but instead all information necessary to generate and maintain this in the Cellular Model, as will be described in the next sections. For each modeling operation, this process is carried out in two phases. First, the Cellular Model is incrementally re-evaluated. Second, the Cellular Model is interpreted, according to the feature information stored in its cellular entities and the current dependencies among the features. These two phases are now separately discussed.

4. INCREMENTAL BOUNDARY EVALUATION

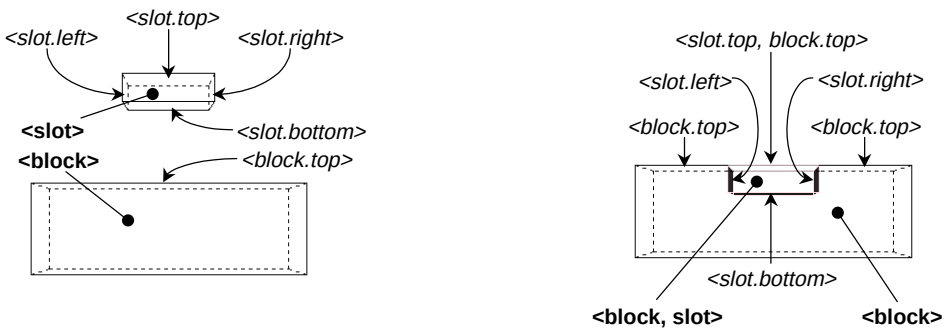
In this section, the structure of the Cellular Model is first briefly presented, after which its re-evaluation process is described, for each modeling operation.

The Cellular Model

The *Cellular Model* is a non-manifold representation of the feature model geometry, integrating the contributions from all features in the Feature Dependency Graph. The Cellular Model is presented in detail in (Bidarra et al. 1998).

The Cellular Model represents a part's geometry as a connected set of volumetric quasi-disjoint *cells*, in such a way that each one either lies entirely *inside* a shape extent or entirely *outside* it. The cells represent the point sets of the shape extents of all features in the model. Each shape extent is, thus, represented in the Cellular Model by a connected subset of cells.

Furthermore, the cellular decomposition is interaction-driven, i.e. for any two overlapping shape extents, some of their cells lie in both shape extents (and are called *interaction cells*), whereas the remaining cells lie in either of them. As a consequence of this, two cells can never volumetrically overlap. They



(a) before the operation

(b) after the operation

Figure 6 – Propagation of owner data in a cellular union operation

may, however, be adjacent, in which case there is an interior face of the Cellular Model separating them. Such a face can be regarded as having two "sides", designated as partner *cell faces*. A face that lies on the boundary of the Cellular Model has only one cell face (one "side"), that of the only cell it bounds. In either case, a cell face always bounds one and only one cell. Each shape face is, thus, represented by a connected set of cell faces.

In order to be able to search and analyze features and their faces in the Cellular Model, each cell has an attribute –called *owner list*– indicating which shape extents it belongs to, see Figure 5. Similarly, each cell face has also an owner list, indicating which shape faces it belongs to.

Just like for features, the *nature of a cell* expresses whether its volume represents "material" of the part or not. Its determination will be precisely described in Section 6. For example, in the model of Figure 5, cells 1, 6 and 8 have additive nature (i.e. the nature of either the block or the rib), whereas all other cells have subtractive nature (i.e. that of a subtractive feature in their owner lists). Similarly, the *nature of a cell face* expresses whether it lies on the boundary of a part or not.

The Cellular Model, including its attribute mechanism to maintain and propagate the owner lists of cells and cell faces, was implemented using the Cellular Topology husk of the Acis Geometric Modeler (Spatial 1998).

Cellular Model re-evaluation

An important step in each modeling operation consists of updating the Cellular Model, so that changes in the Feature Dependency Graph are also reflected in the geometric model.

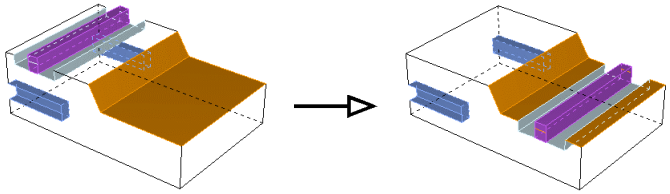
In contrast with history-based systems, which use two non-associative set operations (union and difference) to evaluate the geometric model (see example in Section 2), in the semantic feature modeling approach *only one set operation* is used to evaluate the Cellular Model: it is computed by performing the *non-regular cellular union* of the shape extents of all features. Because it is a union operation, the order in which the shape extents are processed is irrelevant for the final Cellular Model obtained. By these non-regular cellular operations, between (the single cell representing) each shape extent and the other cells generated so far, the cellular decomposition described above is computed. Essential in this process is the correct propagation of the owner lists of each cell and cell face when these are further decomposed, so that each entity "knows" precisely which shape extents, or shape faces, it belongs to.

A simple example of a non-regular cellular union operation is given in Figure 6, where a rectangular slot is inserted into a Cellular Model consisting of a single block. Before the cellular union, the owner lists of both cells are as shown in Figure 6.a (for the sake of legibility, only some face owner lists of both shapes are depicted). After the operation, the block cell is decomposed into two cells, of which one is shared with the slot, as depicted in Figure 6.b. The owner lists of the cell faces in Figure 6.a are also propagated, when these faces are split, as shown in Figure 6.b.

Re-evaluation of the Cellular Model after each modeling operation makes extensive use of the ability to process the cellular topology. Detailed Cellular Model processing algorithms can be found in (Bidarra et al. 1998). According to the particular feature operation, these can be summarized as follows:

Adding a new feature instance to the model The shape extent of the new feature is added to the current Cellular Model. For this, the nonregular cellular union operation is used, which computes the cellular decomposition described above, and propagates the owner list attributes among the relevant cells and cell faces in the Cellular Model.

Removing a feature instance from the model This is carried out in three steps: (i) all references to that feature are removed from the owner lists of Cellular Model entities; (ii) cells with an empty owner list are removed from the Cellular Model; and (iii) adjacent cells and cell faces with the same owner list are merged.



(a) before the operation (b) after the operation

Figure 7 – Incremental re-evaluation of the Cellular Model for a model modification

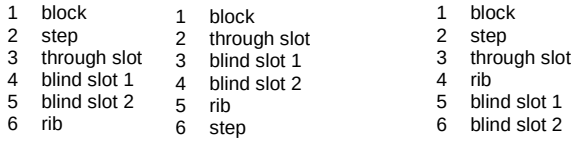
Editing a feature instance in the model In this case, *only* the edited feature, and all its dependent features that are also modified by the operation, need to be taken into account. These are removed from the Cellular Model and then re-added with their new parameters, using the *add* and *remove* operations just described.

Figure 7 gives a simple example of a feature modification operation: after moving the through slot, by changing its top attach from the top of the block to the bottom of the step, *only* the through slot and its dependent rib need to be updated in the Cellular Model (all other features keep the parameters and position they had before the operation). This is carried out by removing (the cells of) the through slot and the rib from their original position (i.e. cells 5, 6, 7 and 8 in Figure 5), and adding their shape extents (with a cellular union operation) in the new position.

This example also illustrates that re-evaluation of the Cellular Model is independent of the chronological order of feature creation: the process is the same, regardless of whether the through slot was the first feature attached to the block or not (see Figure 8.a). In contrast with this, in the history-based approach, after the through slot displacement operation, the whole model history (at least since the slot creation) is re-executed, including features whose imprint remains unaltered, e.g. blind slot 1 and blind slot 2 (see model history at the left-hand of Figure 8.a). Even worse, a history-based modeling system would not be able to perform this operation if the model history were that at the right-hand of Figure 8.a, because the step is there more recent than the through slot (see discussion of this drawback in Section 2, Figure 2).

5. COMPUTATIONAL COST OF CELLULAR MODEL RE-EVALUATION

An important issue is the efficiency of operations on the Cellular Model, because boundary evaluation is still a bottleneck in many modeling systems. The structure of the Cellular Model is certainly more complex than that of a manifold boundary representation, normally used in history-based feature modeling sys-



(a) before the operation (b) after the operation

Figure 8 – Feature precedence examples for the models of Figure 7

tems. In addition, attribute storing and propagation mechanisms demand some additional processing not required by set operations on a conventional manifold boundary representation. However, this is far outweighed by the performance improvement of *incremental re-evaluation* of the Cellular Model.

In history-based feature modeling, evolution of the model is, by definition, dependent on the re-execution of sequences of modeling operations from the model history. As discussed in Section 2, it is impossible to always avoid including in those sequences operations on unmodified features, although their re-execution is superfluous. The computational cost of a modeling operation, such as the modification or removal of a feature, is therefore proportional to the total number of features in the model if no intermediate evaluated models are stored, or to the number of features created after the modified or deleted feature, if intermediate models or deltas are stored.

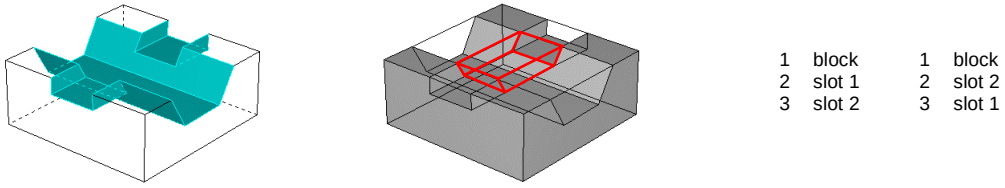
Building the whole Cellular Model from scratch has also a computational cost that is proportional to the number of features in the model. Fortunately, this is only required when the Cellular Model needs to be built in one step, e.g. when starting a modeling session with a previously created model file.

Once this has been done, the computational cost of re-evaluating the Cellular Model after a modeling operation is kept limited, i.e. it is independent of the total number of features present in the model, because, as described in the previous section, *only* the imprint of the shape extents whose geometry has been affected by a modeling operation is updated. This scope is easily obtained from the geometric and algebraic constraint solving process, which keeps track of which features it actually modifies during the operation (see Section 3).

In conclusion, the computational cost of Cellular Model re-evaluation is only dependent on the number of features whose geometry is affected by the operation. Usually, this number is very limited, so computational cost is minimized.

6. HISTORY-INDEPENDENT INTERPRETATION OF THE CELLULAR MODEL

Interpretation of the Cellular Model consists of determining whether the point set represented by each cell does belong to (or represent “material” of) the product, i.e. the nature of that



(a) (b) (c)

Figure 9 – Precedence permutation among independent features with the same nature

cell. This requires deciding which of the features in its owner list "prevails", either as additive or as subtractive. It is only at this point that the precedence among features needs to be taken into account.

Determination of cell natures

If, based on some precedence criteria, a *global ordering* can be defined on the set *F* of all features in the model (say assigning to them unique, increasing precedence numbers), then every cell owner list (a subset of *F*) can be sorted according to these precedence numbers. The nature of a cell becomes, then, the nature of the last feature in its owner list (i.e. the feature with the highest precedence number). It is obvious that such a global ordering is always possible, as the set of features in the model is discrete and finite, and thus numerable.

Considering that the nature of a cell, whose owner list has *n* elements, is exclusively determined by the n^{th} element (the last feature) in the owner list, we can derive the following properties:

1. The nature of a cell is independent of (the precedence numbers of) features that do not occur in its owner list.

   For example, referring to the models in Figure 7, the nature of the cell of the blind slot 1 is independent of whether the precedence numbers of, say, the step, the blind slot 2 and the through slot are higher or lower than its own precedence number.

2. The nature of a cell is preserved under permutations of the *n* elements of its owner list, provided that the nature of the n^{th} element is kept the same. In particular, the cell nature remains invariant under permutations of the first *n-1* elements of its owner list.

   This is illustrated by the example model with two crossed slots, see Figure 9.a: the nature of the interaction cell shared by both slots (see Figure 9.b) is not affected by the relative precedence of these two subtractive features, and thus either precedence sequence in Figure 9.c yields the same interpretation.

From these two properties, we can conclude that, in general, different feature precedence sequences can result in the same nature for each cell. For the interpretation of the model, it is enough to have a procedure that is always able to generate *one* such sequence. We now discuss appropriate precedence criteria to achieve this goal.

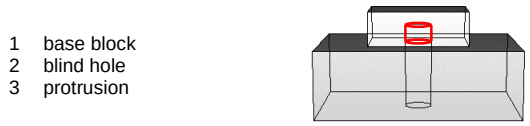
Precedence criteria

The example in Figures 7 and 8 suggests that sorting the precedence sequence of features according to the *static* chronological feature creation order, is not a good criterion for the interpretation of the Cellular Model. In fact, whatever the sequence of precedence numbers before the operation, changing the slot's attachment requires the step to precede the slot and the rib after the operation. Otherwise, the precedence number of the rib would be lower than that of the step, and the former would appear truncated by the latter. We can conclude from this example that the precedence sequence of features should be *dynamic*, i.e. subject to revision after each modeling operation.

Stated differently, and according to property 2 above, for the interpretation of the structure of the feature model at any moment, the chronological order in which its features were originally created is, in general, not determinative. Instead, *the actual dependencies* among them *at that stage* do provide the key for this precedence analysis.

For the model of Figure 7.a, for example, one can draw the following two precedence relations, based on an attachments' analysis: (i) the through slot feature precedes the rib feature (i.e., the latter is dependent on the former), and (ii) the base block feature precedes all other features (i.e., they are all dependent on it). Relative precedence among all other features is irrelevant when it comes to interpret this model; so, for example, both feature precedence sequences of Figure 8.a produce the same model interpretation of Figure 7.a. Figure 8.b, on the other hand, shows a possible sequence of precedence numbers for the modified model in Figure 7.b. Whatever the sequence of precedence numbers before the operation, it can be remarked that the step now should precede the through slot (i.e. should have a lower precedence number), as required by the new attachment of the latter.

In short, the dynamic dependency relation of the Feature Dependency Graph permanently "reflects" the current structure of the feature model. Therefore, it makes up the first precedence criterion in our goal of generating a global precedence sequence:



(a) (b)

Figure 10 – The precedence sequence (a) for the model of Figure 1.e yields an incorrect nature for the cell highlighted in (b)

Criterion I Each edge in the Feature Dependency Graph represents a precedence relation between two features in the model: if feature f_2 depends on feature f_1, then f_1 *precedes* f_2.

By definition, the above criterion is able to define a precedence relation between dependent features only. In the modeling operation described in Figure 1, however, a precedence problem was pointed out between two independent features, the blind hole and the protrusion: if the precedence numbers were kept as shown in Figure 10.a, i.e. following the sequence of the history in Figure 1.c, the top interaction cell of the blind hole (highlighted in Figure 10.b) would be additive, i.e. have the nature of the protrusion. This nature is incorrect, because it is not in accordance with the semantics of the modeling operation performed: the nominal depth of the blind hole, which has been increased, does not match the actual depth it exhibits in the model.

What is characteristic of the situation described in Figures 1 and 10, is that the modeling operation in question causes an *overlap* between two *independent features of different natures*. To avoid incorrect interpretations of a model such as shown in Figure 10, an explicit precedence relation should be established when, as a result of a modeling operation, two independent features with different natures come to overlap. The question arises then *which* orientation should be assigned to this precedence relation, considering that none of the two features depends on the other. As mentioned above for the example of Figure 1, to preserve the semantics of a modeling operation, a feature f that is modified by the operation should "prevail" in the determination of the nature of its interaction cells. Stated differently, other overlapping independent features with different nature should precede f in the precedence sequence. After the operation in Figure 1, thus, the protrusion should precede the blind hole. Hence the following:

Criterion II To each *new* overlap between independent features f_1 and f_2 of different natures, caused by some modeling operation on f_2, corresponds a precedence relation f_1 *precedes* f_2.

With the two criteria above, based on the dependency relation and on possible overlap between independent features, a global sorting of all features in the model can be achieved. In the next section, we show how such precedence criteria are used to produce a correct interpretation of the Cellular Model, which is unambiguously determined without invoking model history considerations.

7. COMPUTATION OF FEATURE PRECEDENCE RELATIONS

The precedence relation, defined in the previous section, is an example of a so-called *partial ordering* relation, i.e. a relation that defines an ordering between *some* pairs of elements in a set *S*, but not among all of them. In general, partial ordering relations satisfy the following three properties for any distinct elements *x*, *y* and *z* of the set *S*:

1. Transitivity

   `if` ***x*** `precedes` ***y*** `and` ***y*** `precedes` ***z***`,`
   `then` ***x*** `precedes` ***z***

2. Asymmetry

   `if` ***x*** `precedes` ***y***`,`
   `then` ***y*** `does not precede` ***x***

3. Irreflexivity

   x `does not precede` ***x***

The dependency relation used in Criterion I is permanently maintained in the Feature Dependency Graph, and is therefore always explicitly available for use in the model interpretation process.

Criterion II states that an explicit precedence relation should also be established when a modeling operation causes an overlap between two independent features with different natures. To detect such occurrences and determine the orientation of the relation, the set of features involved in the modeling operation, i.e. those actually processed in the incremental re-evaluation of the Cellular Model, see Section 4, is analyzed according to the algorithm in Figure 11. Basically, the algorithm checks whether any of these features, f_i, has acquired a new overlap with an independent feature f_j; if this is the case, and the features have different natures, then the relation "f_j precedes f_i" is recorded.

In detecting a new overlap, the algorithm uses the notion of *overlapping set* of a feature *f*, i.e. the set of all features that overlap with feature *f*, denoted *OS(f)*. Determination of the *OS(f)* is straightforward and requires no geometric computations: it is simply computed as the union of the owner lists of all cells of feature *f*. The overlapping set of each feature f_i involved in the operation is computed and stored before the Cellular Model is re-evaluated, and compared with the $OS(f_i)$ determined after the re-evaluation, in order to detect new overlaps.

```
FeaturesInvolved = {features involved in the modeling operation}
For each f_i in FeaturesInvolved
   NewOverlappings = OS_after(f_i)\OS_before(f_i)
   for each f_j in NewOverlappings
      if f_i independent of f_j and f_i.nature ≠ f_j.nature
      then record relation "f_j precedes f_i"
```

Figure 11 – Precedence detection algorithm for overlapping independent features

```
NewSequence = <>
OldSequence = <current precedence sequence>
While OldSequence is not empty do
   find in OldSequence the next feature f such that
           all precedents of f are already in NewSequence
   move f from OldSequence to NewSequence
   assign new dependency number to f
```

Figure 12 – Topological sorting algorithm for assigning feature precedence numbers

Once the precedence relations have been established, using the two criteria described, the global sorting of features can be easily performed by a classical topological sorting algorithm, whose goal is precisely to generate a linear ordering of a partially ordered set of elements (Wirth 1976). The algorithm, shown in Figure 12, builds a new sorted sequence by iteratively selecting (and removing) from the old sorted sequence a feature whose precedents are all already sorted.

The number of tests in this selection is minimized if candidate features are sought in the order of the old sorted sequence, because most modeling operations have a fairly local effect, affecting only the precedence of a few other features, if any at all. For example, adding a new feature to the model typically maintains the whole precedence sequence, the new feature being just attached at its end.

Eventually, the features in the resulting sorted sequence have new precedence numbers assigned, and the nature of all cells becomes thus automatically determined.

Summarizing, precedence numbers are revised after every modeling operation. For this, the precedence relations are updated in the model, and a new sorting is performed among all its features. These get then new precedence numbers assigned, reflecting the new model structure, as has been illustrated for the modeling operation of Figures 7 and 8.

8. APPLICATION EXAMPLES

In this section, a number of examples is presented to illustrate both the incremental evaluation and the interpretation of the Cellular Model. In each example, a modeling operation is performed that involves changes in one (or more) feature(s). The Cellular Model corresponding to the final situation is also shown, together with the graph of precedence relations used in its interpretation. In this graph, the feature nodes that are actually modified by the operation are highlighted (in black). Moreover, additional precedence relations between independent features, established by the precedence detection algorithm of Figure 11, are drawn with a dotted line, to distinguish them from the other precedence relations, derived from the dependencies in the Feature Dependency Graph.

Example I

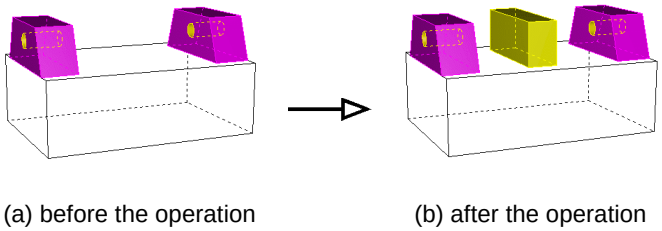
The initial model consists of a base block, two ribs and a through hole, attached between the two ribs, see Figure 13.a. A protrusion is then inserted between the ribs, so that it overlaps with the through hole, see Figure 13.b. Considering that the through hole and the protrusion are overlapping independent features, the precedence detection algorithm of Figure 11 prescribes that the through hole should precede the protrusion, as indicated by the dotted edge in Figure 13.c. Thus the protrusion receives the highest precedence number in the sorting algorithm and, consequently, the nature of the interaction cell highlighted in Figure 13.d is additive, i.e. that of the protrusion.

In a way, this is comparable to what history-based modeling systems correctly assume when a *new* feature is *added* to the model: it becomes the last feature in the model history and, thus, it is the last to leave its shape imprint on the model boundary. This strategy is, in fact, a particular case of Criterion I (a *new* feature is always made dependent on *existing* features), and possibly also of Criterion II (the *new* feature overlaps with *existing* independent features, as in the example of Figure 13).

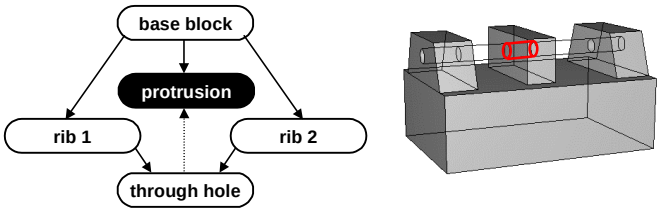
However, history-based boundary re-evaluation often fails when some *existing* feature is *modified* in the model, as discussed in Section 2. The approach presented above, on the other hand, remains applicable for all modeling operations, as will be illustrated with the next two examples.

Example II

In this example, the model has two crossing slots of different depths attached to a base block, and a rib on the bottom face of

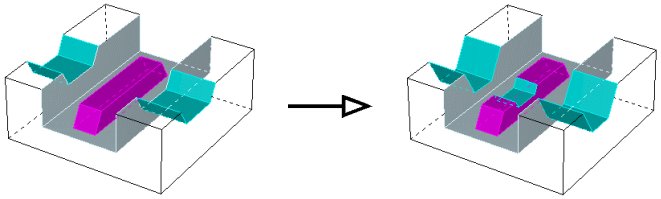


(a) before the operation (b) after the operation

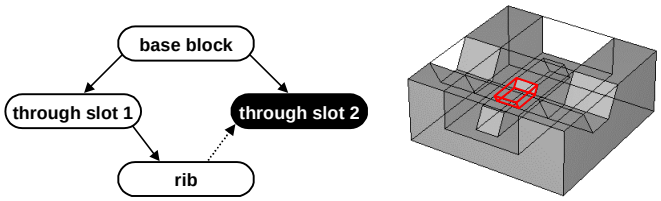


(c) Precedence relations (d) Cellular Model

Figure 13 – Cellular Model interpretation after adding a new feature



(a) before the operation (b) after the operation



(c) Precedence relations (d) Cellular Model

Figure 14 – Cellular Model interpretation after editing a subtractive feature

the deeper slot, through slot 1, see Figure 14.a. The depth of the split through slot 2 is then increased, so that it overlaps with the rib, see Figure 14.b. Again, as these two features are independent, their overlap leads to a precedence relation being established between them, see Figure 14.c. As a consequence, the rib receives a precedence number lower than the through slot 2, and their interaction cell is thus subtractive, as shown in Figure 14.d.

With history-based boundary re-evaluation, the resulting model of Figure 14.b, would not be achievable if the through slot 2 had been created before the rib.

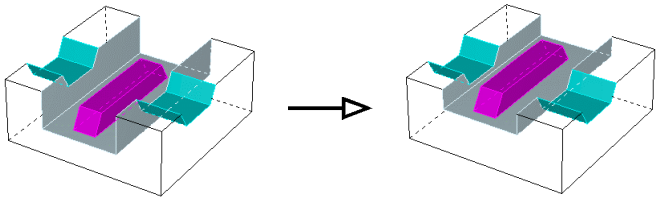
Example III

The third and last example is based on the same model of example II, see Figure 15.a. However, the modeling operation now consists of decreasing the depth of the deeper slot, through slot 1, such that its dependent rib becomes in interaction with the other slot, through slot 2, see Figure 15.b. In this case, from the analysis of the precedence detection algorithm, the (indirectly) modified rib is preceded by the independent through slot 2, see Figure 15.c, resulting in an additive nature for their interaction cell, highlighted in Figure 15.d.

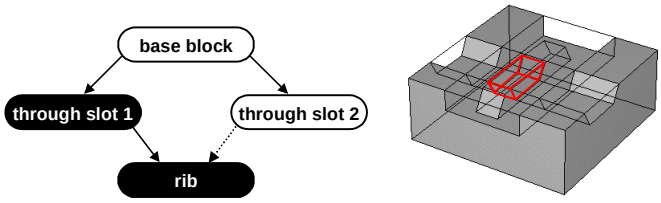
Again, the detection of the new overlap, and the precedence relation established, yields a model interpretation in which the nature of the modified features prevails over that of the other overlapping features. To achieve the model of Figure 15.b using a history-based modeling system, the through slot 2 should be created before the rib (which is exactly the history sequence that would make the resulting model of example II unfeasible).

9. CONCLUSIONS

This paper addresses several problems related to the maintenance of a boundary representation for feature models. To overcome the shortcomings exhibited in this respect by most current history-based feature modeling systems, a declarative feature model has been proposed, which integrates two levels, and de-



(a) before the operation (b) after the operation



(c) Precedence relations (d) Cellular Model

Figure 15 – Cellular Model interpretation after (indirectly) editing an additive feature

ploys mechanisms for automatically maintaining consistency between these levels.

At the first level, a high-level representation of the product –the Feature Dependency Graph– is maintained, with only those entities that are relevant for its manipulation: features and model constraints. This facilitates user interaction, providing a natural dialog in terms of features, and hiding from the user many unnecessary details of the geometric model.

At the second level, a non-manifold Cellular Model is maintained as evaluated geometric representation of the product. This representation has two important properties. First, its re-evaluation after each modeling operation has a lower computational cost, compared to that of boundary representations maintained in history-based modeling systems. Second, its evaluation and interpretation are independent of the chronological order of feature creation in the model. The latter solves several problems inherent to history-based modeling.

ACKNOWLEDGMENTS

We thank Klaas Jan de Kraker for his constructive comments in several discussions that motivated this research.

REFERENCES

Autodesk (1998) Autodesk Mechanical Desktop 2.0 User's Guide, Autodesk, Inc., San Rafael, CA, USA

Bidarra, R. (1999) Validity maintenance in semantic feature modeling. PhD Thesis, Delft University of Technology, The Netherlands

Bidarra, R. and Bronsvoort, W.F. (1999a) Validity maintenance of semantic feature models. In: *Proceedings of Solid Modeling '99 – Fifth Symposium on Solid Modeling and Applications, 9–11 June, Ann Arbor, MI, USA*, Bronsvoort, W.F. and Anderson, D.C. (Eds.), ACM Press, New York, pp. 85–96

Bidarra, R. and Bronsvoort, W.F. (1999b) Semantic feature modelling. Submitted for publication

Bidarra, R., de Kraker, K.J. and Bronsvoort, W.F. (1998) Representation and management of feature information in a cellular model. *Computer-Aided Design* **30**(4): 301–313

Chen, X. and Hoffmann, C.M. (1995) On editability of feature based design. *Computer-Aided Design* **27**(12): 905–914

Gupta, S.K., Regli, W.C. and Nau, D.S. (1995) Manufacturing feature instances: which ones to recognize? In: *Proceedings Solid Modeling '95 – Third Symposium on Solid Modeling and Applications, 17–19 May, Salt Lake City, UT, USA*, Hoffmann, C.M. and Rossignac, J.R. (Eds.), ACM Press, New York, pp. 141–152

Parametric (1998) Pro/ENGINEER Modeling User's Guide, Version 19, Parametric Technology Corporation, Waltham, MA, USA

Peters, B.F. (1997) MicroStation Modeler: the design and implementation of an extensible solid modeling system, In: *Geometric Modeling: Theory and Practice – The State of the Art*, Strasser, W., Klein, R. and Rau, R. (Eds.), Springer, Berlin, pp. 361–378

SDRC (1998) I-DEAS Master Series 6 User's Guide, Structural Dynamics Research Corporation, Milford, OH, USA

Shah, J.J., Rogers, M.T., Sreevalsan, P.C., Hsiao, D.W., Mathew, A., Bhatnagar, A., Liou, B.B. and Miller, D.W. (1990) The ASU Features Testbed: an overview. In: *Proceedings of the 1990 ASME Computers in Engineering Conference, August, Boston, MA, USA*, ASME, New York, Vol. 1, pp. 233–241

Spatial (1998) Acis 3D Modeling Kernel, Version 4.1, Spatial Technology Inc., Boulder, CO, USA

Talwar, R. and Manoochehri, S. (1994) Algorithms to detect geometric interactions in a feature-based design system. In: *Proceedings of the 1994 ASME Design Engineering Technical Conferences, September, Minneapolis, MN, USA*, ASME, New York

Versluis, J.W., Bronsvoort, W.F., de Kraker, K.J. and Seebregts K. (1997) Feature visualization. In: *CD-ROM Proceedings of the ASME 1997 Design Engineering Technical Conferences, 14-17 September, Sacramento, CA, USA*, ASME, New York

Wirth, N. (1976) Algorithms + Data Structures = Programs. Prentice-Hall, Englewood Cliffs, NJ, USA