A Relational Model for Interactive Manipulation of Form Features based on Algebraic Geometry

G. Brunetti*, T. De Martino*,**, B. Falcidieno**, S. Haßinger*,**

(*) Fraunhofer-Institut für Graphische Datenverarbeitung
Wilhelminenstrasse 7, 64283 Darmstadt, Germany
(**) Istituto per la Matematica Applicata, C.N.R.
Via de Marini, 6, 16149 Genova, Italy

Abstract

One of the main problems in feature-based modeling is to represent and to handle the feature semantics as well as the geometric and topological information. In this paper an algebraic representation structure which maps feature shape semantics into a geometric model and provides facilities for interactive feature manipulation is presented. In this structure, called the Feature Entity Relation Graph (FERG), the parameters defining a form feature are expressed as algebraically formulated relationships between feature entities. In the same way it is possible to represent dimensions and geometric constraints to express the functional shape requirements of the part and to preserve the feature shape semantics. To enable local as well as global interactions and manipulations on FERG, entities and relationships are structured at different levels of detail.

Keywords: *Design by Features, Geometric Constraints, Algebraic Geometry, Interactive Feature Manipulation, User-defined Features, Integrated Feature-Based Systems*

1 Introduction

In CAD systems based on the design-by-features approach, features are used as semantic entities which are provided to the user in order to express the design intent. Examples of design features are slots, holes and pockets. The relationships between features are kept in a feature-based model which then is evaluated in order to derive the geometric 3D representation of the final object. Traditionally, parametric systems perform this evaluation following a procedural approach, where every pre-defined design feature is hardcoded in a procedure driving the solid modeler. The problem with this approach is that no link between the feature description and the solid representation is preserved. For detailed discussion of the procedural approach see [Sha93, Sha94]. The necessity to overcome these problem leads us to the definition of a parametric design feature model, which is able to represent the parametric description of design features independently of the functionality and type of the underlying geometric modeler. It allows to represent and maintain the design semantics of pre-defined design features and arbitrary user-defined features. Furthermore, it can achieve the consistency between the solid model and the design feature-based model. Interactive changes on the two models such as dragging a side face of a slot or editing a design feature parameter should be automatically reflected in both representations.

This paper introduces a parametric representation, the Feature Entity Relation Graph (FERG), which provides functionalities to create and to maintain user-defined and pre-defined design features. The FERG enables to represent the feature entities and their parametric dependencies. The entities are real constituents of the part model, such as faces and edges, or are virtual entities representing construction entities such as axis of symmetry or coordinate frames. Dimensions and positions are expressed as relationships between entities. Within FERG, design features are described by their entities and intrinsic and extrinsic relationships. Intrinsic relations are used to represent the internal structure of the feature shape, while extrinsic relations represent the inter-relationships among features. These relations cover the relative position and orientation of features with respect to real and virtual entities.

The paper is organized as it follows: The FERG model is outlined in paragraph 2. The properties of FERG and its major benefits are discussed and shown by an example in paragraph 3. In paragraph 4 the representation of pre-defined and user-defined design features by FERG is described.

2 Feature Entity Relation Graph

The theoretic foundations to define a model able to represent user-defined design features are derived from the parametric and algebraic geometry [Abh87, Buc85, Chu90, Hof89, Hof93] as well as from feature-based modeling [KoP93, Laa93, Sha94]. These foundations are applied for the specification of a *Feature Entity Relation Graph* (FERG) [Bru94], which

Solid Modeling '95, Salt Lake City, Utah USA
© 1995 ACM 0-89791-672-7/95/0005...$3.50

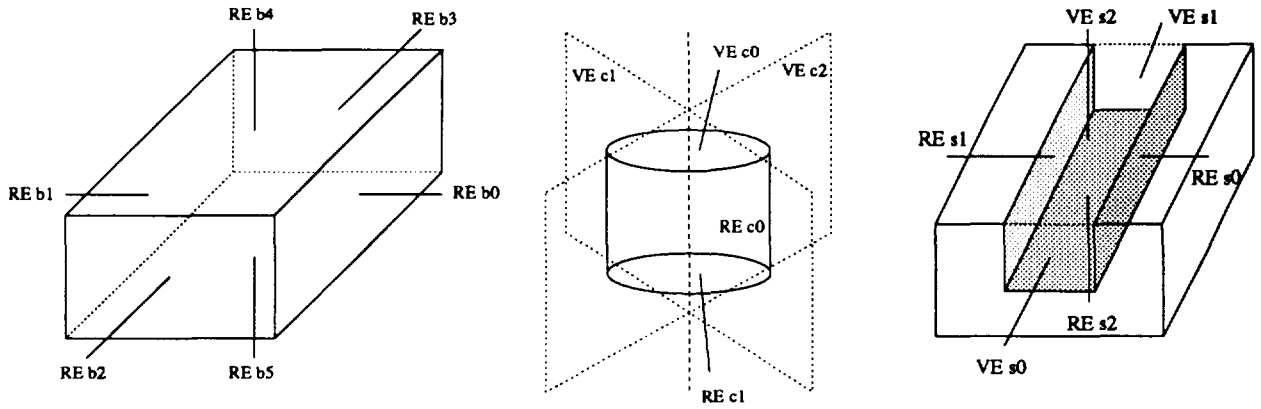

Figure 1: *Real entities (RE) and virtual entities (VE) composing a form feature.*

is introduced in this paragraph. The basic entities and the relationships between these entities are further described. Applications from design-by-features using FERG and the major benefits of FERG are also outlined.

2.1 Feature Entities

Form features are described by their semantics and a set of parameter. The shape model of a form feature is derived from this description and it is mapped to a set of *feature entities*. Feature entities are specified at two levels of abstraction: 1. A lower level, which specifies the shape of a feature entity using *algebraic entities*. 2. A higher level, which allows a point of view to feature entities closer to their semantical signification by separating them into *real entities* and *virtual entities*. These feature entities are represented as implicit algebraic equations. They build up the halfspace decomposition of the form feature.

Algebraic entities are the containers able to hold the representation of algebraically specified feature entities. They are restricted to shapes that can be expressed by implicit algebraic surface or curve equations generated by irreducible polynomials of first or second order. The following categories of entity primitives are supported: planar surfaces, quadric surfaces, space curves, linear plane curves, conic plane curves, and points. The surface and curve points have to be regular to avoid self-intersecting or degenerated shapes.

Surfaces are interpreted as algebraic halfspaces; i.e., point sets of the form: $\{(x,y,z)|f(x,y,z) \leq 0\}$. The interior of an halfspace is defined by the opposite direction of the surface normal. The implicit algebraic equation of a *planar surface* is a polynomial of degree one:

$$a \cdot x + b \cdot y + c \cdot z + d = 0 \qquad (1)$$

An alternative representation is $N^T \cdot P = 0$, where $N = [a\ b\ c\ d]^T$ is the column vector of the plane-equation coefficients and $P = [xyz1]^T$ the plane points in homogeneous coordinates.

The implicit algebraic equation of a *quadric surface* is a polynomial of degree two. It can be represented by a symmetric matrix A containing the coefficients of the generating polynomial. Then the quadric surface is defined by all points P such that $P^T \cdot A \cdot P = 0$. Provided that a quadric surface is transformed to the origin, its general equation is limited to the following form:

$$[x\ y\ z\ 1] \cdot \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & k \end{pmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = 0 \qquad (2)$$

Based on this equation, the following classes of quadrics are allowed ($a > 0$ can always be obtained by multiplying the equation by -1):

a	b	c	k	
> 0	> 0	> 0	< 0	ellipsoid
> 0	> 0	< 0	< 0	hyperboloid
> 0	> 0	$= 0$	< 0	elliptical cylinder

Curves can be defined in two different ways: first, as a space curve derived by the intersection of two surfaces, and second, as linear or conic plane curves on a surface with generating implicit polynomials similar to those of the surfaces. Points are represented as the intersection of three surfaces. However, the most common representation for a point is the representation via homogeneous coordinates. Even if homogeneous points are not algebraic, they are integrated for the sake of computational reasons and due to the compatibility with other components of the system environment.

Feature entities are classified in two categories: *real entities* and *virtual entities*. They express the semantics of a feature entity with respect to its role in the part model or in the definition of a form feature. *Real entities* (RE) correspond to elements of the solid representation. Examples are the faces of a block and the bottom face of a blind hole. *Virtual entities* (VE) are construction primitives and closure faces of a volumetric form feature. Construction primitives are, for instance, the principal axis of a cylinder (usually defined as the intersection of two planes), or the center of a circle. An example for a closure face is the top face of a blind hole or a pocket. Virtual entities have no permanent representation in the solid model. They can be temporarily evaluated and visualized to the user, for example to support interactive manipulation on features by editing and transforming construction primitives. Further examples of real and virtual entities are illustrated in Figure 1.

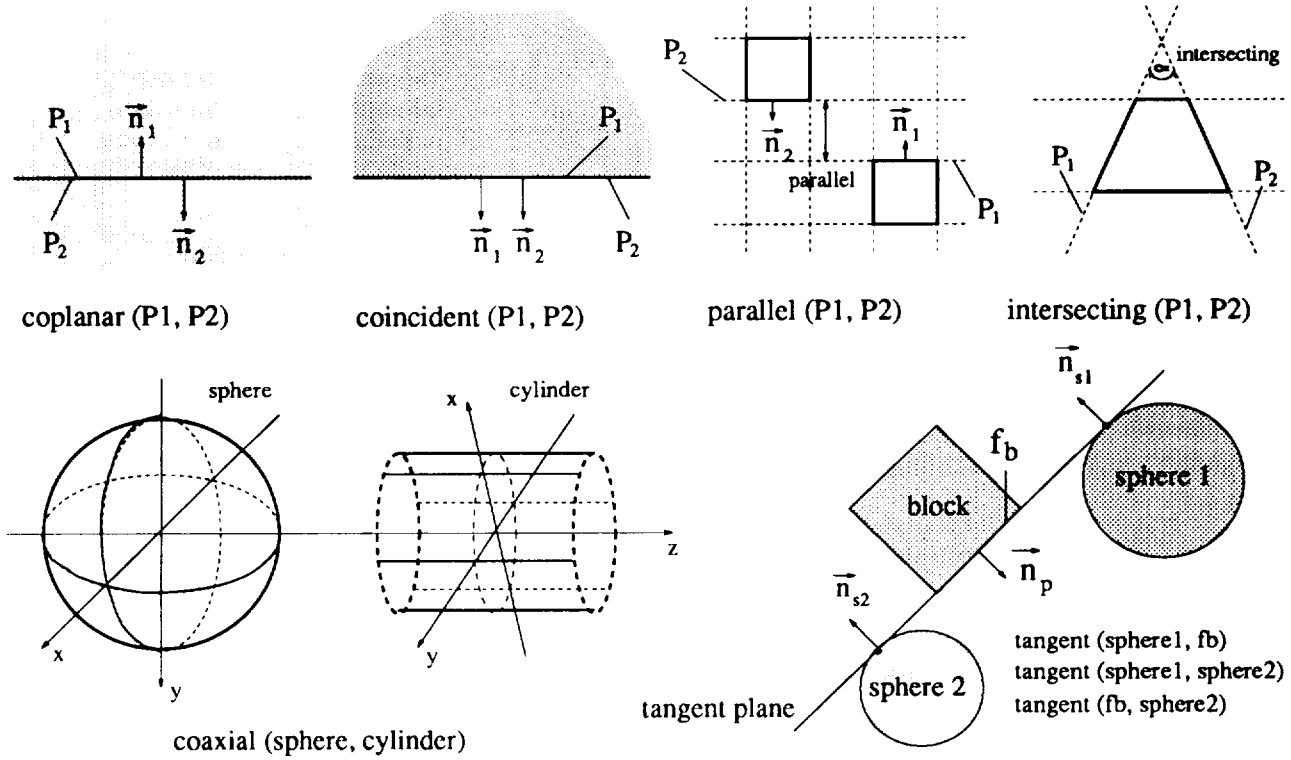

Figure 2: *Examples of spatial relations.*

2.2 Entity Relations

Entity relations (ER) are used to attach parameters and geometric constraints to feature entities and to control the spatial relations and dimensions between form features. Spatial relations represented by the ERs are algebraically formulated. The goal is to represent all spatial relations between feature entities that are significant in the design process. ERs are not just passive objects that represent and store spatial relations. They are active since they are able to determine the spatial relation between feature entities and know how to create a feature entity having a specific spatial relation to a reference entity. The measure associated to an ER depends on the type of the represented spatial relation. It is interpreted as a distance or an angle.
The supported spatial relations and the feature entities they are applied to, are defined as follows (see Figure 2):

Coincident Two feature entities are *coincident*, if they have the same type of generating polynomials, and if they have the same position and orientation in 3D space. A coincident relationship can be established between surfaces, between curves, and between points.

Coplanar *Coplanarity* is similar to the coincident relation. The difference is that the reference and target entity must have opposite orientation. Coplanar relations are defined between the following types of feature entities: plane-plane, plane-curve, plane-point, curve-curve, curve-point.

Parallel A *parallelism* relation can be established between planes and between lines.

Perpendicular *Perpendicularity* is a relation restricted to planes and lines.

Intersecting Also the *intersection* relationship is defined only between planes and lines. Two intersecting entities do not need to intersect on the border of the object (see Figure 2).

Tangent A quadric is *tangent* to another quadric, if they have the same tangent plane for a surface point. A plane is tangent to a quadric, if the tangent plane of one of the surface points of the quadric is coincident or coplanar with the plane. The case of tangent curves is similar.

Concentric Only feature entities with an algebraic specification of degree two can be *concentric*: quadric - quadric, conic - conic. Quadrics or conics are concentric, if their center-point has the same position and if their algebraic specification is equal with the exception of the radii.

Coaxial Quadrics are *coaxial*, if the z-axis of their local coordinate systems have the same orientation within the reference coordinates system. The coaxial spatial relation corresponds to the parallel relationship of planes.

Solving spatial relations is relevant in three contexts:

1. When instantiating an ER between existing feature entities. In this case their spatial relation has to be determined.

2. When a feature entity has to be created with a specific spatial relation to an existing reference entity. In this case the evaluation is performed in two steps: First, the new feature entity has to be represented algebraically. Second, an entity relation has to be instantiated, which represents and stores the spatial relation between the reference entity and the new feature entity. The new feature entity is called the target entity.

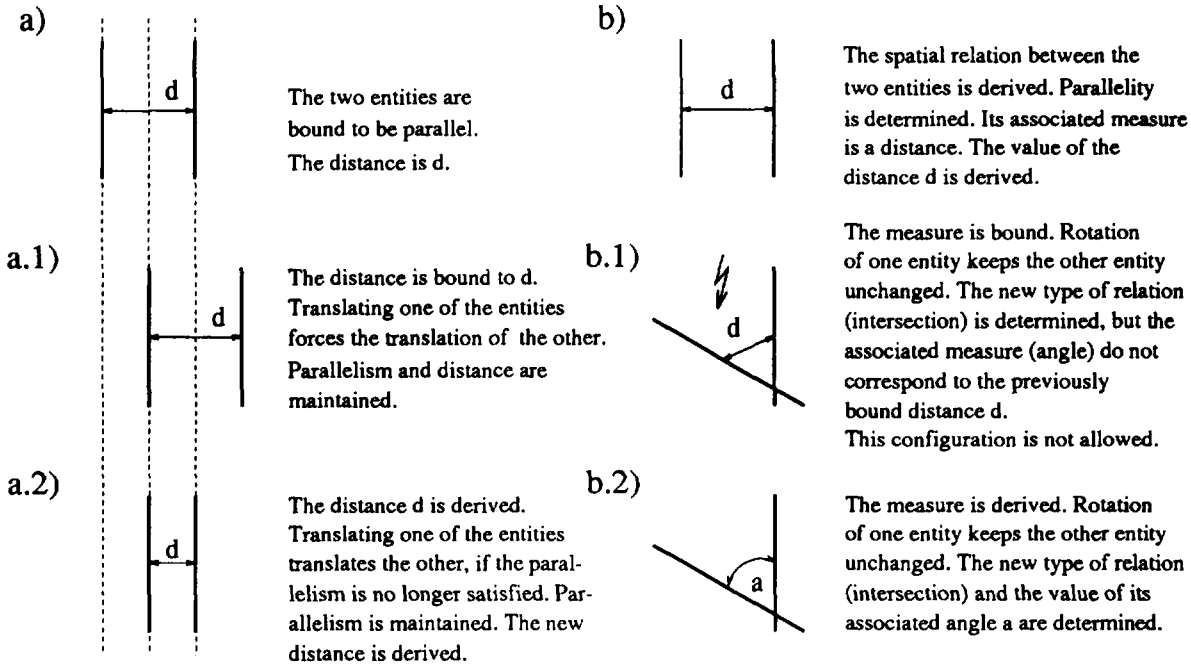

Figure 3: *Bound and derived entity relations.*

3. When modifying the part model, for example, by a user interaction. If the user modifies the model by transforming a plane then all entity relations referring to this plane have to be activated to perform the following operations: first, the spatial relation between the transformed reference plane and the target plane has to be determined. This is similar to the situation 1. Second, if the entity relation represents a geometric constraint, then it is tried to modify the position of the target entity accordingly, otherwise the new spatial relation is stored in the entity relation. This is similar to the situation 2 previously described. Of course, the modification of a target entity implies that all entity relations referring to it have to perform the same operations, until all feature entities affected by the transformation of the plane are modified. This is called the propagation effect.

The actual value of a spatial relation can be *bound* to a specific value or it can be *derived* from the current situation. Also the ER itself can be bound or derived. Three significant combinations are possible:

1. The spatial relation and the associated measure are bound. This configuration is used to define a geometric constraint between feature entities or to represent bound parameters. The defined type of spatial relation and the value of the associated measure have always to be maintained. Transforming one feature entity causes all other feature entities associated by a bound relation to be transformed as well (see Figure 3(a.1)).

2. The spatial relation is bound, but the value of the associated measure is derived. This configuration defines a geometric constraint between feature entities that is less strong than the first one. Only the defined type of spatial relation has to be maintained. The value of the measure is determined from the actual situation. Entity relations of this type are used to represent derived parameters. Transforming one feature entity causes the modification of other entities, if and only if a bound spatial relation between them is no longer verified. The value of the associated measure is not considered, but it is updated according to the new configuration (see Figure 3(a.2)).

3. The spatial relation and the associated measure are derived. This configuration do not define any constraint, but it sets a notifier which informs about every change of a spatial relation between the referenced feature entities (see Figure 3(b.2)).

ERs are divided in different types depending on the feature entities they refer to, and on the type of relation they represent. They are hierarchically classified as illustrated in Figure 4. The ERs expressing the shape of a form feature by geometric constraints and parameters are called *intrinsic entity relations* (IER). IERs are *binary*, if they refer to a pair of feature entities. They are *hyper IERs*, if they define spatial relations between a set of entities belonging to the same feature. Hyper IERs, for instance, are used to constrain and to parameterize pattern features. IERs are always bound to specific spatial relation. The value of the associated measure might be derived. Spatial relations between pairs of entities of different form features are represented by *extrinsic entity relations* (EER). They define a relative positioning between form features. EERs can be *unidirectional* or *bidirectional*. *Unidirectional EERs* define a parent-child relation between form features. The position of a child feature is relative to the position of the parent feature. *Bidirectional EERs*, in contrast to the unidirectional, define an equivalent entity relation between form features. The relative position is a bilateral constraint for both features, for example, to control the invariant perpendicular intersection of two slots as illustrated in Figure 5. The initial set of IERs describing the shape characteristics of a form feature are pre-defined,

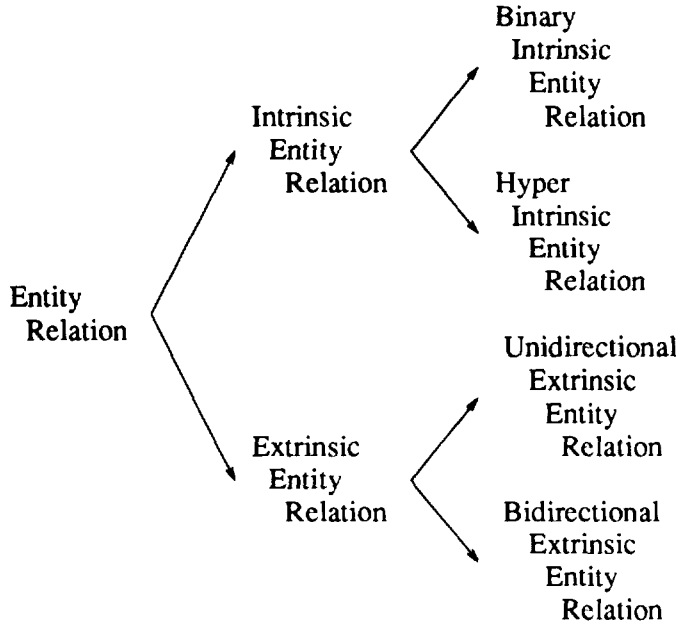

Figure 4: *The entity relation class hierarchy.*

while the EERs depend on the design process. However, it is possible to define, to erase, to convert, or to filter ERs explicitly.

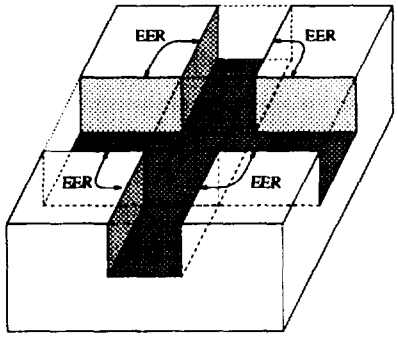

Figure 5: *The bidirectional EER controlling the invariant perpendicular intersection of two slots. The four EERs illustrate the possible alternatives.*

2.3 The FERG Model

When creating a pre-defined design feature, its correspondent form feature REs and VEs as well as the IERs are instantiated and stored. Furthermore, the EERs are established in order to connect feature entities of different form features of a part model. The result is a graph whose nodes represent the feature entities and whose arcs represent either intrinsic or extrinsic ERs. The graph is called *feature entity relation graph* (FERG) and represents the FERG model of a part. Since the FERG model is a low level representation of a part model, which considers just the feature entities, it is necessary to abstract from FERG by an hierarchical organization of its elements.

The first abstraction is done by representing the FERG model of a single form feature with a separate structure. This structure is called *intrinsic feature entity relation graph* (I-FERG). In contrast to the general FERG, the I-FERG is not directed. It is an hyper graph because of the hyper IERs. Because the arcs of an I-FERG are restricted to IERs, its family is a proper subset of the general FERG family: $I-FERG^* \subset FERG^*$. In the following all sets are denoted by a * to distinguish them from their elements.

When a pre-defined design feature is created its I-FERG representation is instantiated as well. So it is not necessary for the user to define the intrinsic geometric constraints of a feature explicitly. Only the bound parameter values have to be specified. They are obtained during the interaction with the user. This is done either by the direct input of a parameter value or by deriving it through the construction method of the pre-defined design feature. Some examples for the I-FERG representation of pre-defined design features are illustrated in Figure 7.

Formally, the *intrinsic feature entity relation graph* $I-FERG^*$ is defined by an hyper graph (E^*, IER^*) with nodes E^* and hyper-arcs $IER^* \subset \mathcal{P}(E^*)$ (here, $\mathcal{P}(E^*)$ denotes the set of sub-sets of E^*) on which the following functions are defined:

$$\sigma_E : E^* \longrightarrow \{RealEntity, VirtualEntity\};$$
$$\sigma_{AE} : E^* \longrightarrow AlgebraicE^*;$$
$$\sigma_{GC} : IER^* \longrightarrow GC^*;$$

where:

- σ_E defines the entity type of the node;
- σ_{AE} associates a node with its algebraic representation, i.e. its implicit equation (here, $AlgebraicE^*$ is the set of supported algebraic entities);
- σ_{GC} determines and controls the spatial relation and geometric constraint of an IER (GC^* is the set of supported geometric constraints).

The FERG model of a part always contains a set of I-FERGs, each representing one form feature. These I-FERGs are connected by EERs. Hence, it is possible to identify each I-FERG within a FERG model as a single node of a primary graph structure. The obtained graph is called *extrinsic feature entity relation graph* (E-FERG). Its nodes represent I-FERGs. The arcs are sets containing the EERs defined between the feature entities of two I-FERGs. They are called *extrinsic feature relations* (EFR) to express that they reference features rather then feature entities. The EFRs do not contain any geometric information, but document that there are some EERs defined between I-FERGs. EFRs are directed. Their direction depends on the direction of their underlying EERs, hence they can be bidirectional. An E-FERG representation of a part is shown in Figure 6. It corresponds to the block with a slot and a cylindrical blind hole illustrated in the same Figure. A detailed illustration of the nodes, which represent I-FERGs, and the EFRs, which are sets of EERs, is shown in Figure 7.

The *extrinsic feature entity relation graph* $E-FERG^*$ is formally defined by (F^*, EFR_u^*, EFR_b^*) with nodes F^*, unidirectional arcs $EFR_u^* \subset F^* \times F^*$, and bidirectional arcs $EFR_b^* \subset \mathcal{P}^2(F^*)$ (i.e. the set of subsets containing exactly two elements of F^*). The bijective function $\Sigma_{Feature} : F^* \longrightarrow Feature^*$ links a node with the form feature it represents and vice versa.

EER^* is the set of EERs that can be established between the feature entities of different I-FERGs. It is defined as the

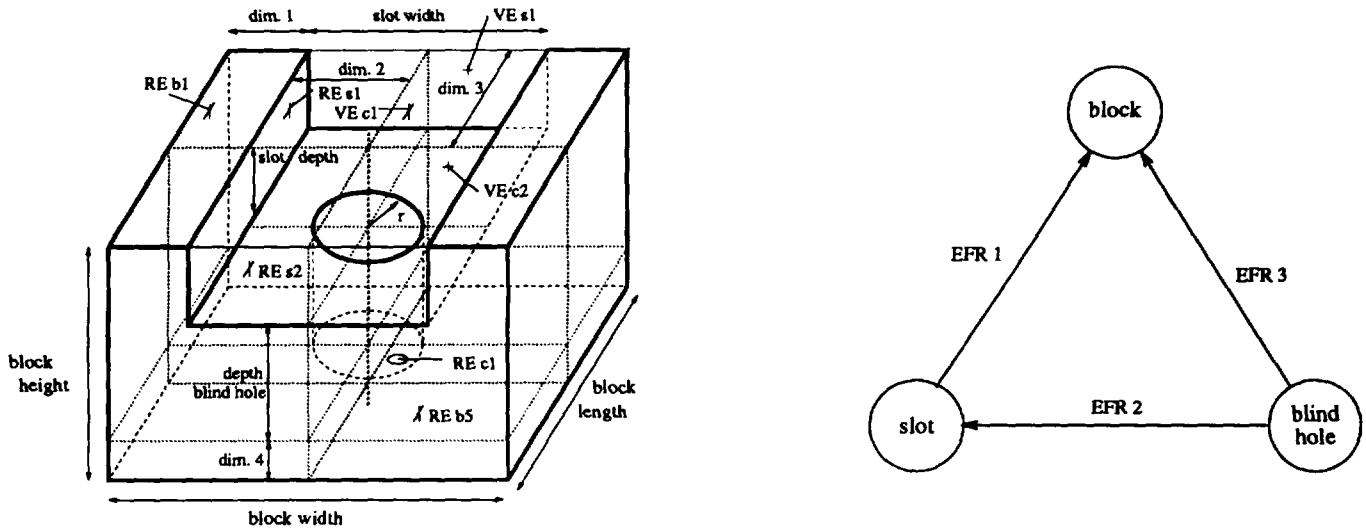

Figure 6: *A slot on a block with a cylindrical blind hole and its corresponding E-FERG representation.*

union of the unidirectional arcs $EER_u^* \subset E^* \times E^*$ and the bidirectional arcs $EER_b^* \subset \mathcal{P}^2(E^*)$ (where $\mathcal{P}^2(E^*)$ the set of subsets containing exactly two elements of E^*). The spatial relation and geometric constraint of an EER is determined and controled by the function $\Sigma_{GC} : EER^* \longrightarrow GC^*$ (GC^* is the set of supported geometric constraints).

The link between elements of the I-FERGs and the E-FERGs are maintained by the following functions:

$$\Sigma_{I-FERG} : F^* \longrightarrow I - FERG^* \text{ (bijective)};$$
$$\Sigma_F : E^* \longrightarrow F^*;$$
$$\Sigma_{EFR} : EER^* \longrightarrow EFR^*;$$
$$\Sigma_{EER} : EFR^* \longrightarrow \mathcal{P}(EER^*);$$

where:

- Σ_{I-FERG} associates a node of an E-FERG with its defining connected component (an I-FERG) and vice versa;
- Σ_F links a feature entity (an I-FERG node) with the corresponding E-FERG node;
- Σ_{EFR} associates an EER with its EFR.
- Σ_{EER} is the "inverse" of Σ_{EFR} and allows the access to the EERs contained within an EFR;

Let $f \in F^*$, such that (E_f^*, IER_f^*) denotes the I-FERG associated with f, i.e. $\Sigma_{I-FERG}(f)$. Then the FERG model is valid, if the following conditions are fulfilled:

- All E-FERG nodes are associated with one I-FERG such that all feature entities of an I-FERG are linked to the same feature:

  $$\forall f \in F^* : \forall e \in E_f^* : \Sigma_F(e) = f;$$

- Feature entities connected by an IER belong to the same feature:

  $$\forall IER \in IER^* : \forall \{e, e'\} \subset IER : \Sigma_F(e) = \Sigma_F(e');$$

- Feature entities connected by an EER belong to different features:

  $$\forall (e, e') \in EER_u^* : \Sigma_F(e) \neq \Sigma_F(e'),$$
  $$\forall \{e, e'\} \in EER_b^* : \Sigma_F(e) \neq \Sigma_F(e');$$

- Two related feature entities of different I-FERGs are connected by exactly one EER:

  $$\forall (e, e') \in EER_u^* : (e', e) \notin EER_u^* \wedge \{e, e'\} \notin EER_b^*;$$
  $$\forall \{e, e'\} \in EER_b^* : (e, e') \notin EER_u^* \wedge (e', e) \notin EER_u^*;$$

- Two related features are connected by exactly one EFR:

  $$\forall (f, f') \in EFR_u^* : (f', f) \notin EFR_u^* \wedge \{f, f'\} \notin EFR_b^*,$$
  $$\forall \{f, f'\} \in EFR_b^* : (f, f') \notin EFR_u^* \wedge (f', f) \notin EFR_u^*;$$

- If all EERs between two I-FERGs have the same direction, then the corresponding EFR has this direction, too. Otherwise, the EFR is bidirectional:

  $$\forall EFR \in EFR^* : \Sigma_{EER}(EFR) \neq \emptyset,$$
  $$\forall EFR \in EFR^* : EFR \in EFR_u^* \iff$$
  $$\Sigma_{EER}(EFR) \subset EER_u^*$$
  $$\wedge \forall (e, e') \in \Sigma_{EER}(EFR) : e \in E_f^* \wedge e' \in E_{f'}^*$$

  where $EER = (f, f')$;

Finally, all geometric constraints of part have to satisfied. Local modifications of feature entities or entity relations are recursively propagated through the model as described in Section 2.3.

3 An Example

In this paragraph an example of a FERG model is discussed. Its intension is to clarify the previously introduced theory on FERGs.

The underlying part is a slot on a block with a blind hole as it is illustrated in Figure 6. As shown, the part is composed by the three pre-defined design features represented in Figure 1. The main shape is a block. It consists of six planar faces, which are real entities. They are labeled as in

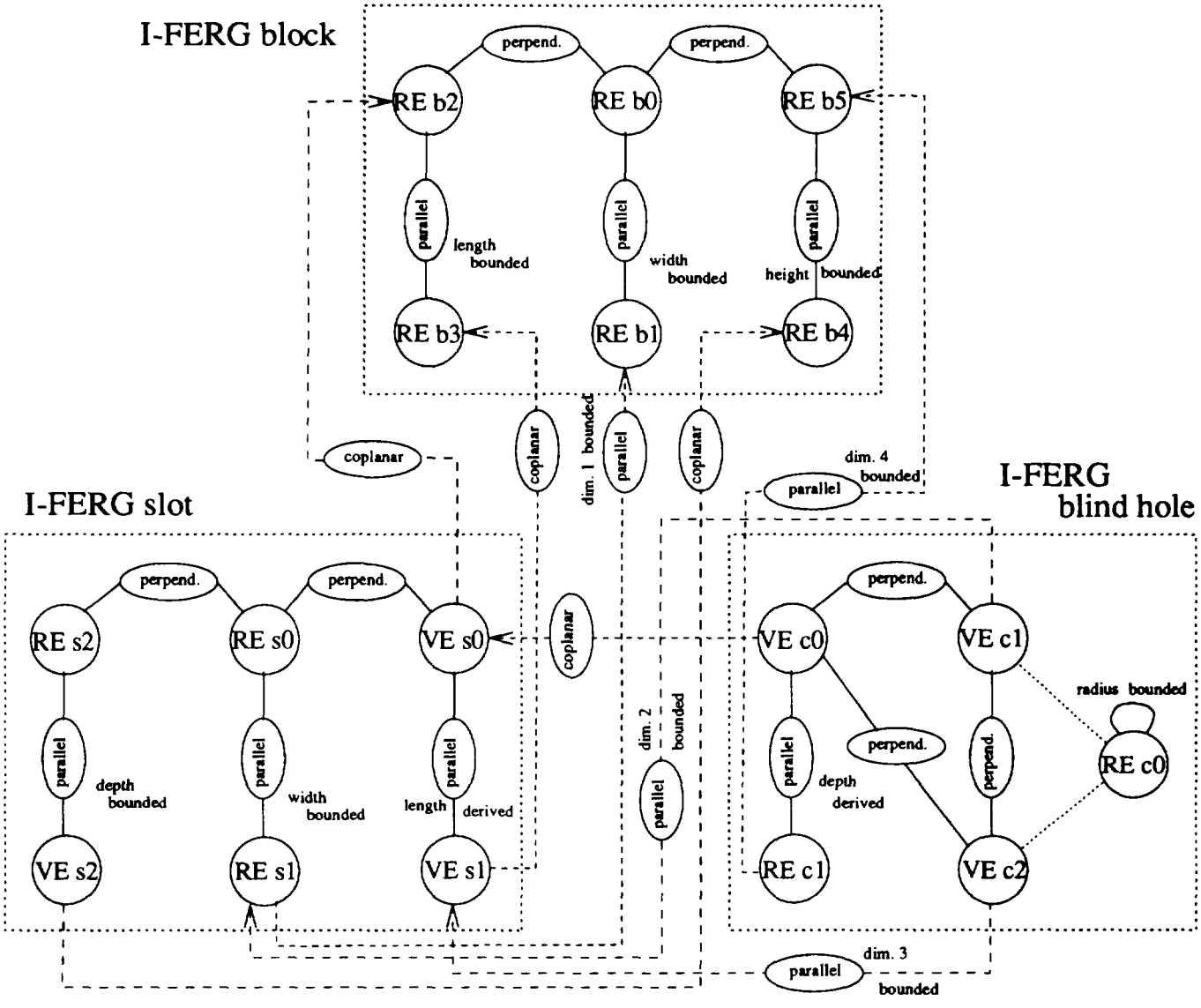

Figure 7: *The complete FERG representation of the part in Figure 6.*

Figure 1. The I-FERG representation of the block is shown in Figure 7. Its interactive creation may be performed by the following sequence of operations. First, the user has to pick two points of the bottom face, which specify two corners and the width of the block. A third pick on the same plane defines the length of the block. With this information five of the six faces of the block are already specified. These are the bottom face, which is algebraically equal to the plane on which the corners are picked, the two side faces, and the front and back face. Their algebraic representation is easily obtained by the three point equation for planes. The top face and the associated height finally can be obtained, for example, by a linear sweep operation. Since the block is the main shape of the part, the resulting parameter values associated to the IERs have to be bound.

Next, the slot is created. Its construction starts with the instantiation of the I-FERG representation, which is similar to that of a block (see Figure 7). The differences are that all planar entities have an opposite orientation with respect to the corresponding faces of the block, and that the closure faces are virtual entities. The reason is that the halfspace decomposition of the slot must be a subtractive volume. To position the slot, the user has to pick one of the block faces, which is the reference face for the slot. This pick already defines a coplanar relation between the top face of the block and the top face of the slot. To define the direction of the slot, the user may pick two additional faces (which have to be adjacent to the firstly picked face). Also this defines implicitly two further EERs, which are the front and back face of the slot coplanar with the picked faces. From that the first parameter of the slot can be derived. It is the slot length. Other two picks on one of the already defined faces specify the position of the left and the right face of the slot, and its width. Finally, the user may decide that the position of the slot on the block has to be constrained, for instance, by an EER which bounds the left face of the block to a parallel relationship with the left face of the slot. The associated distance may be derived, bound to the actual value, or bound to a specific value. The latter may cause the transformation of the slot relative to the block. Here the

distance is defined to be bound to a given dimension (dim. 1 in Figure 6).

Finally, the cylindrical blind hole has to be created. As before, first the corresponding I-FERG is instantiated. In comparison with the previously instantiated I-FERGs, the I-FERG of a cylindrical volume has a particularity (see Figure 7), which is the representation of the parameter radius and the orientation of the cylindrical surface. Because the radius is directly encoded in the algebraic representation of the cylinder, it has to be represented by a self-relation. The orientation of the cylinder, in this example, is defined by the intersection of two virtual planes. They define the z-axis of the local coordinate system of the cylinder. Hence, the relation between these planes and the cylinder is not represented by IERs, but it is directly encoded in the algebraic representation of the cylindrical entity.

The I-FERG of the blind hole is updated to the actual values by performing a sequence of operations similar to those before. To position the blind hole, two bound EERs are instantiated. They control the parallelism between the virtual planes of the blind hole and the faces of the slot. The values are bound to the dimensions dim. 2 and dim. 3. At the end, the user decides that the depth of the blind hole has to be derived by defining the distance dim. 4 between the bottom face of the blind hole and the bottom face of the block. To specify this dependency, the user instantiates an EER, which represents a parallel relationship, and he connects the two bottom faces by picking them.

4 Mapping Design Features into FERG

Design features are classified as pre-defined and user-defined features [Ovt92]. In the following paragraph the mapping of design features into FERG is discussed. Then it is shown how user-defined features are obtained from a FERG model.

4.1 Pre-Defined Design Features

The pre-defined design features (PDF) are represented by I-FERGs, which are stored in the design feature library of the application. Before a pre-defined design feature is created interactively, its I-FERG representation is instantiated. It is not necessary for the user to define the intrinsic geometric constraints of a feature explicitly, but bound parameter values have to be specified. They are obtained from the interaction with the user. This is done either by the direct input of the parameter value or by deriving it through the construction method of the pre-defined design feature.

4.2 User-Defined Design Features

User-defined design features (UDF) can be either compound features or arbitrary user-defined design features. The FERG structure supplies the methods for creating and maintaining user-defined design features. They can be defined by adding some nodes and/or arcs to the I-FERG representation of a pre-defined design feature. Adding an arc, i.e. an IER, defines an UDF with less degrees of freedom. Adding feature entities and intrinsic entity relations to the I-FERG defines a new design feature.

The added feature entities can be, for instance, solids created with the solid modeler which then are transformed into their algebraic representation. After their implicitization, for instance, by using the methods described in [Man90, Kal90, Kap92], they have to be connected to feature entities of the I-FERG by instantiating IERs representing the spatial relations. A second possibility to create user-defined arbitrary design features is guided by the interactive design process. The user instantiates feature entities by defining a specific spatial relation to a reference entity. In this case, the IER is created together with the new entity.

A property of FERG is the ease of building compound features. The generation of those features is based on a simple transformation operation. Consider a FERG model created during a design session, for instance, the block with a slot and a cylindrical blind hole in Figure 7. The difference between this FERG model, which represents a part model composed of several features, and the representation of the same part expressed as a compound feature, is the type of ERs. This means that, if the EERs between feature entities of the different features are modified into IERs, then the E-FERG is transformed into the I-FERG of the corresponding compound feature. The relevant design parameters of the obtained compound feature are all bound ERs of the original graph. For example, the positioning of the slot relative to the left face of the block, which was previously expressed by an EER, transforms to an IER, which now represents a parameter of the new UDF. The obtained I-FERG can be stored in the design feature library in order to reinstantiate it.

5 Conclusions

In this paper the FERG (Feature Entity Relation Graph) model is introduced. It is a relational model, which allows to maintain generic (pre-defined) and arbitrary (user-defined) design feature representations is introduced. FERG is based on entities, which constitute the existing (real) topologic feature entities as well as non-existing (virtual) entities. Virtual entities allow to access construction geometry such as coordinate frames or axis of symmetry as well as entities, which are removed after boolean operations such as closure faces. The parametric relations between those entities store spatial as well as positional information derived from the interactive design process. The parametric relations are solved using algebraic equations, which determine the values bound to those relations.

The FERG model constitutes a central part of an integrated feature modeling system [DeM94a]. This system is based on an integrated feature-based kernel, which serves as the common representation in order to link design by features and feature recognition processes. Within this integrated kernel the FERG is closely linked to a shape feature object graph, which is a geometric representation structure covering the geometric and topological interrelations, as intersections and closures, between shape features. The shape feature object graph is theoretically based on the SGC (Selective Geometric Complex) defined in [Ros90].

The ongoing development of the integrated feature-based modeling system is based on ACIS (STI), a non-manifold solid modeling toolkit [ACI92a, ACI92b]. The system is realized in C++ on AT&T compiler.

Acknowledgments

This work is partly supported by the cooperation between the Istituto per la Matematica Applicata, C.N.R., in Genova, Italy, and the Fraunhofer-Institut für Graphische Datenverarbeitung in Darmstadt, Germany, and the Human Capital & Mobility project No. ERB CHBGCT 930380 on "Integration of Design By Features and Automatic Feature Recognition in an Open CAD Framework" supported by the

EU. The authors wish to thank Dr. H. Elter and Dr. J. Rix for their contribution to clarify certain aspects of this work.

REFERENCES

[Abh87] S. Abhyankar and C. Bajaj, "Automatic Rational Parameterization of Curves and Surfaces IV: Algebraic Space Curves", *ACM Trans. on Graphics*, 1987

[ACI92a] Spatial Technology Inc., "ACIS - Interface Guide, Release 1.4", Colorado (USA), 1992

[ACI92b] Spatial Technology Inc., "ACIS - API Guide, Release 1.4", Colorado (USA), 1992

[Bru94] G. Brunetti, "Feature Entity Relation Graph - An Algebraic Representation Structure to support Interactive Feature-Based Design", Diplomarbeit (Diploma Thesis), TH-Darmstadt, Germany, 1994

[Buc85] B. Buchberger, "Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory", *Multidimensional Systems Theory*, N.K. Bose, ed., D. Reidel Publishing Co., Dortrecht, Germany 1985, pp. 184-232

[Chu90] J. C. H. Chung and M. D. Schussel, "Technical Evaluation of Variational and Parametric Design", *Computers in Engineering*, Volume 1, 1990, pp. 289-298

[DeM94a] T. De Martino, B. Falcidieno, F. Giannini, S. Haßinger, and J. Ovtcharova, "Feature based modeling by integrating design and recognition approaches", *Computer-Aided Design, Volume 26, Number 8*, Aug. 1994

[DeM94b] T. De Martino, B. Falcidieno, F. Gamba, F. Giannini, "Feature Kernel Model: Reference Model for Integrated Feature-Based Modeling Systems", *AICA Annual Conference*, Volume 2, Palermo, Italy, 1994, pp. 1467-1478

[Hof89] C. M. Hoffmann, "Geometric and Solid Modeling: An Introduction", Morgan Kaufmann, San Mateo, Calif., 1989

[Hof93] C. M. Hoffmann, "Implicit Curves and Surfaces in CAGD", *IEEE Computer Graphics & Applications*, Jan. 1993, pp. 79-88

[Kal90] M. Kalkbrenner, "Implicitization of Rational Parametric Curves and Surfaces", tech. report, RISC Linz, Linz, Austria 1989

[Kap92] D. Kapur and Y.N. Lakshman, "Elimination Methods: An Introduction", *Symbolic and Numerical Computation: An Integration*, D. Donald, D. Kapur and J. Mundy, eds., Academic Press, New York, 1992

[KoP93] H. Ko, M. Park, H. Kang, Y. Sohn, and H. Kim, "Integration methodology for feature-based modeling and recognition", *Computer Engineering*, 1993, pp. 23-34

[Laa93] T. Laakko and M. Mäntylä, "Feature modelling by incremental feature recognition", "Computer Aided Design, Volume 25, Number 8", 1993, pp. 479-492

[Man90] D. Manocha and J. Canny, "Implicitization of Rational Parametric Surfaces", *The Mathematics of Surfaces (Proc. 4th IMA Conf.)*, Oxford University Press, England 1990

[Ovt92] J. Ovtcharova, G. Pahl, and J. Rix, "A Proposal for Feature Classification in Feature-based Design", *Comput. & Graphics Vol. 16, No.2*, 1992, pp. 187-195

[Ros90] J. R. Rossignac and M. A. O'Connor, "SGC: A dimension-independent model for pointsets with internal structures and incomplete boundaries", Geometric Modeling for Product Engineering, M. J. Turner, K. Preiss (Editors), Elsevier Science Publishers B. V. (North-Holland), IFIP, 1990

[Sha93] J. J. Shah and M. T. Rogers, "Assembly Modeling as an Extension of Feature-Based Design", *Research in Engineering Design (1993) 5*, Springer-Verlag London Limited, 1993, pp. 218-237

[Sha94] J. J. Shah, A. Ali, and M. T. Rogers, "Investigation of Declarative Feature Modeling", *Proceedings of the Computers in Engineering Conference (ASME)*, 1994