Two-Dimensional Selections for Feature-Based Data Exchange

Ari Rappoport[*] Steven Spitz[†] Michal Etzion[‡]

Abstract

Proper treatment of selections is essential in parametric feature-based design. Data exchange is one of the most important operators in any design paradigm. In this paper we address two-dimensional selections (faces and surfaces) in feature-based data exchange (FBDE). We define the problem formally and present algorithms to address it, in general and in various cases in which feature rewrites are necessary. The general algorithm operates at a geometric level and does not require solving the persistent naming problem, which is required for selection support inside a single CAD system. All algorithms are applicable to the Universal Product Representation (UPR) FBDE architecture, and the general algorithm is also applicable to the STEP parametrics specification.

1 Introduction

Parametric feature-based design is the dominant modeling paradigm in modern CAD systems [Hoffmann93, Shah95]. Data exchange is in general a problem of substantial theoretical and practical value. Consequently, feature-based data exchange (FBDE) is an attractive issue to address in Solid Modeling. Moreover, FBDE is a problem that is difficult and challenging technically, from both an architectural and an algorithmic point of view.

In this paper we present an algorithm that addresses one of the fundamental issues in FBDE: selections of two-dimensional entities. The problem of representing selections that serve as feature arguments inside a single CAD system is usually called the 'persistent naming' problem, and is known to be challenging [Kripac97]. The problem of representing selections in the context of data exchange is different, as will be discussed below.

In a previous paper [Rappoport05], we introduced the problem of handling selections in FBDE, and gave a detailed solution to the case where the selected entities are one-dimensional (edges and curves). In this paper we complete that

[*]The Hebrew University. `http://www.cs.huji.ac.il/~arir`
[†]Proficiency Inc.
[‡]Proficiency Ltd.

work by presenting a solution to the case where the selected entities are two-dimensional (faces and surfaces). The 2-D algorithm is different from the 1-D one, and utilizes the latter where possible.

Our algorithm operates in the context of the Universal Product Representation (UPR) FBDE architecture [Rappoport03, Spitz04]. Nonetheless, the algorithm is applicable to a wider class of architectures [Mun03], including the extension of STEP to support FBDE [Pratt04]. Although the selections problem was not explicitly recognized by the STEP effort, it should not be too difficult to integrate our general algorithm into a STEP implementation.

As far as we know, the present paper is the first one that identifies two-dimensional selections for FBDE as a non-trivial problem and offers a solution. The STEP specification only specifies the file format and does not recognize the problem explicitly, similarly to other related academic efforts such as the EREP project [Hoffmann93], while [Mun05] presents a method that helps users address some selections manually.

Sections 2 and 3 discuss 2-D selections, in feature-based design and in the context of feature-based DE respectively. Section 4 reviews the UPR architecture. Section 5 describes the problem formally, and Section 6 presents the first phase of the algorithm, the computation of a selection cover. The second phase, dealing with the more complex case of rewrites, is described in Section 7. Section 8 discusses our implementation, and we conclude with a discussion. We have made an effort to make the paper stand on its own; specifically, it can be read without reading the previous 1-D selections paper. Some of the presentation of the previous paper, mostly in Sections 4 and 5, is thus repeated here.

2 2-D Selections in Parametric Feature-Based Design

In parametric feature-based design, the model is represented by a directed a-cyclic graph of operations called features. Most features either create new geometry or modify a part's existing geometry (some features only insert or modify meta-data and other attributes). The graph can be 'evaluated' after each feature, generating an object that is represented using a boundary representation (Brep). The Brep contains two components: a graph (topology) of vertices, edges, faces and shells (*Brep entities*) and their interconnectivities, and concrete geometry corresponding to each of these entities.

During interactive design, the user defines new features or edits existing features, and sees a 3-D graphical view of the current Brep on the screen. However, the Brep is not used only for viewing; it is also used for defining the arguments of some of the subsequent features. This is one of the major differences between the parametric feature-based design paradigm and classic Constructive Solid Geometry. Enabling such argument selection constitutes a primary constraint on the nature of Brep representations, as discussed in [Rappoport96].

Every feature has arguments that define its semantics. Brep entities serve as feature arguments in most of the useful and commonly used features. This is done by letting the user select a set of Brep entities on the 3-D view and define them as arguments to the present feature. There may be several different such arguments for a single feature.

Brep entities can be 0-D (vertices, points), 1-D (edges, curves, loops), 2-D (faces, surfaces) or 3-D (shells). The interesting cases in which those entities serve as feature arguments are the 1-D and 2-D cases. In [Rappoport05] we have given many examples for the 1-D case, perhaps the most central of which is the edge round (or fillet) feature.

Here we focus on the 2-D case. In general, there are two types of 2-D selections: selections whose goal is to select a *surface*, and selections whose goal is to select a *face* (or a set of faces). The difference is that in the former case the feature only needs the carrier surface, while in the latter case a bounded subset of a carrier surface is needed, a subset which can be represented as the union of a set of bounded faces.

Following are some central examples for features whose arguments include user selected two-dimensional Brep entities:

- Extrude Until Face: the Extrude feature is the most common geometry defining feature. It takes a parametric 2-D sketch and extrudes it to create a 3-D shape. Extrude comes in many different variations. One useful variation is when it is defined to be *Until Face*, where the created 3-D shape is trimmed when it is blocked by a selected existing Brep face or by the carrier surface of that face. Other variants include *Until Next* and *Until Last*, which terminate at the next or last face (or carrier surface) encountered, respectively.

- Draft: this is a complex feature mostly used for plastic injection molding. A set of Brep faces (or subsets of Brep faces) is skewed at a specified angle, modifying prior geometry in a global manner. The user needs to select the set of faces to be skewed, and to optionally sketch curves on those faces to define face subsets.

- Shell: in this feature, the user selects a set of faces to be removed from the current Brep. The remaining faces are 'thickened' and then trimmed in order to create a valid 3-D solid.

- Offset: this feature creates a face (or a surface) defined as the offset (at a specified distance) of a selected face (or of a set of faces, or of a carrier surface.)

- Face Round (or Fillet, or Blend): this feature is very similar to the common Edge Round. It creates smooth surfaces between selected faces (or sets of selected faces), and removes from the Brep everything 'covered' or 'cut off' by those new surfaces such that the end result is a 3-D solid. Face Round is in some sense more powerful than Edge Round, because it can blend faces that are far from each other in the Brep topology.

- Sketch On Face: the user selects a face whose carrier surface is a plane embedded in 3-D space, and draws a parametric 2-D sketch on this plane. Usually, the sketch serves as one of the arguments of an Extrude feature.

Some CAD systems allow the user to select only a subset of selection required as the feature argument, completing the rest automatically. The main automatic completion method is to recursively add to the entities explicitly selected by the user all entities that are adjacent with smooth connectivity (e.g., G1 continuity).

In the following sections we will give examples for all of the above features except Sketch On Face, which for the problem in this paper is conceptually similar to Until Face.

3 Potential Problems with 2-D Selections in FBDE

CAD systems must represent selected Brep entities in a way that generalizes over the current geometry, because at any point in time the user may modify the parameters of any feature. When this happens, the system plays the feature history again, a process which usually results in a different geometry. A purely static geometric representation of the selections would hence not be valid. This problem is known as the persistent naming problem.

To tackle this problem, CAD systems abstract away some of the properties of the selected entities. Usually, usage is made of properties that are independent of the numerical values in the model and are functions of more intrinsic properties, such as Brep topology, identities of the features (or carrier surfaces) creating Brep entities, qualitative geometric properties (such as convexity), etc. In other words, they represent selections using generic names that are persistent under parameter changes (hence the term). For a general solution to generic naming of Brep entities see [Rappoport97], and for solutions in the context of CAD systems see e.g. [Kripac97].

When performing feature-based data exchange there are in principle no immediate parametric changes. On the contrary, the main goal is to construct a model in the target system that is as similar as possible to the model in the source system. Hence a full persistent naming solution is perhaps not needed. It is possible, as we do in this paper, to use representational methods that are different in character and are closer to Brep geometry.

In order to motivate the algorithm of this paper, we first discuss a naive solution and why it is not adequate. Consider the following algorithm for supporting selections for data exchange from system U to system W: (i) identify the selected face f in system U; (ii) locate the face f in system W; (iii) select it in W and use the selection as the argument of the desired feature (the one that the data exchange system defines presently.)

As it is phrased, this algorithm is wrong. Consider Figure 1, which shows a Draft feature. The top row shows the situation before defining the draft, in two different CAD systems. Note that the face drafted on the system on the right (B) does not exist as a single face on the system on the left (A). When exchanging from B to A, Step (ii) above would thus return a 'failure' answer, although it is clearly possible to complete the draft on the left by selecting two faces, as shown in the middle row (left).

The problem stems from the fact that the Breps on the two sides, although geometrically equivalent (as point sets), are topologically different. This situation may arise due to various reasons. For example, suppose that the box has been created by extruding the bottom face upwards. If the 2-D sketch defining the face contains the 'extra' vertex, the extrusion creates a corresponding vertical edge. The vertex may be there for the use of other features, for manufacturing information, for assembly operations, etc.

There is an even graver potential problem, when exchanging a draft from system A to system B. Suppose only one of the two smaller faces in A was to be drafted (the desired result is shown in the bottom row.) This face cannot be expressed as a union of B faces, because it is a proper subset of a B face. The result is thus impossible to exchange directly to system B, because it is impossible to define the desired selection. In Section 7 we will discuss our handling of such cases.

An example containing a cylinder and the Offset feature is shown in Figure 2. CAD systems differ in the way they represent cylinders in Breps. A cylinder can be represented in several ways: (i) using two circular edges and three faces, one cylindrical face and two disks (on the right); (ii) using an additional edge to split the cylindrical face (usually, this is done in order to establish a well-defined parameterization of the face or to avoid a single face having two bounding unconnected loops); and (iii) using two edges to split the cylinder (on the left. Usually, this is done in order to avoid the same edge appearing twice in the same loop.) Figure 2, left, shows the latter situation, where the user selected the top half-cylinder face and the offset is done on that face alone. Figure 2, right, shows the first situation, where the offset is done on the whole cylindrical face. Again, exchanging from right to left requires selection of two faces rather than a single one, and exchanging from left to right necessitates a wider rewrite.

4 Review of the UPR FBDE Solution

The solution presented in this paper for 2-D selections is applicable to a wide variety of FBDE architectures. However, our specific formal problem definition is done in the context of our UPR architecture. Therefore, in this section we describe it briefly, emphasizing those aspects that are relevant to this paper.

The UPR is a star architecture, like most other data exchange architectures. Export and import modules are responsible for communication with the source and target CAD systems respectively. Starness is not a requirement for our selection algorithms, which are applicable also to other FBDE architectures, for example direct source to target translations.

The UPR differs from all other data exchange approaches in that it recognizes that CAD systems differ from each other, both in terms of functional semantics and in terms of implementation of theoretically equivalent operations.

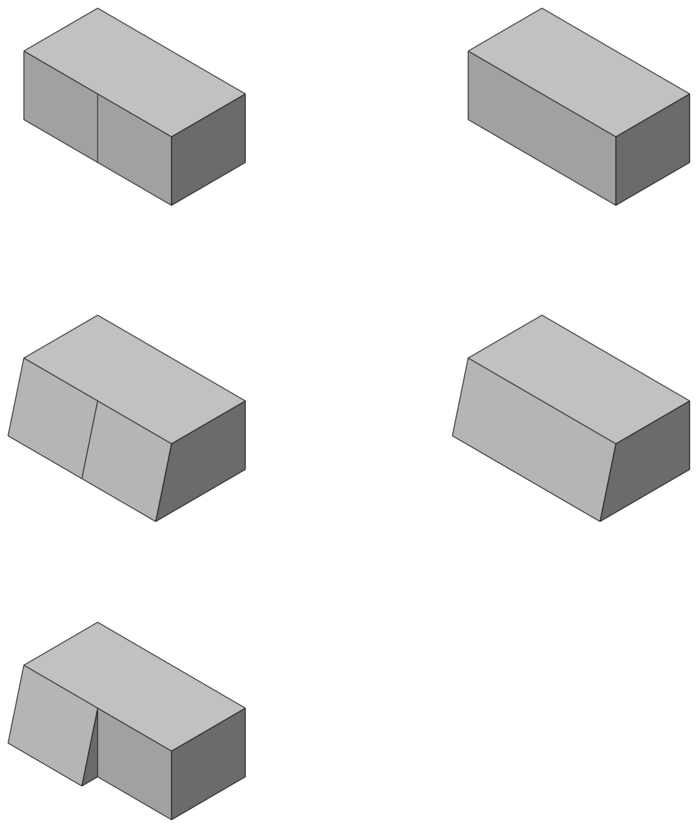

Figure 1: Draft. Top: situation in CAD systems A (left) and B (right) before the definition of a Draft. Note that in system A there is a Brep edge that does not exist in system B. Middle: one possibility for the situation after the draft. Bottom: another possibility for the situation after the draft. The small drafted face in A does not correspond to any face in B, so the draft cannot be directly defined in B. We address this using rewrites.

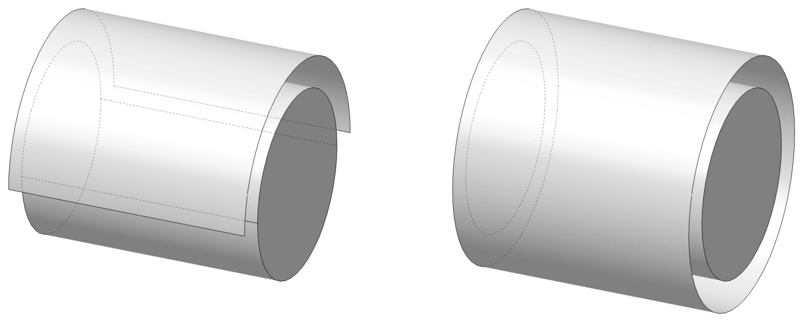

Figure 2: Offset. A cylinder can be represented using one or two cylindrical faces (right and left respectively.) If the user selects a single face as the argument of the offset feature, the results are different in the two cases (assuming no automatic propagation of selections is done by the system on the left.)

Due to market forces and the richness of the feature-based design paradigm, it is not realistic to dictate an ultimate set of features that are supported by a CAD system. Each CAD system provides features and sub-features that are not directly provided by other systems. In addition, due to the complexity of the semantics of certain features, a feature's implementation in one CAD system might result in geometry that is somewhat different from the geometry that a different system generates from the 'same' feature. That is, the feature is the same at a certain level of abstraction (usually, overall function as perceived by the user), but different at a detailed geometric level of abstraction. Finally, we should expect the geometry generated by features to be different due to the fact that different systems utilize different tolerances for different operations, a phenomenon that plagues ordinary geometric data exchange [Qi04].

The UPR is thus explicitly designed to handle the following two cases: (i) a data item explicitly supported by one system and not by another, and (ii) incompatibilities between systems that can be identified only during run-time due to lack of formal specification of implementational differences.

The UPR representation of a feature makes use of two central concepts to address the above: rewrites and verifications. Each feature has an associated set of rewrites, which intuitively are different ways to import that feature into a system that had not succeeded importing it using other means. Instead of dictating a certain fixed representation, the UPR allows an unlimited number of representations that attempt to simulate the semantics of a feature. Rewrites are applied at decreasing levels of abstraction, starting from fully parametric and ending at fully geometric. In [Spitz04] we have described an algorithm for implementing a 'Geometry Per Feature' rewrite, which replaces any feature by a piece of geometry identical to the feature's geometric effect. The Geometry Per Feature rewrite is the first geometric rewrite attempted when all parametric rewrites fail. There are additional geometric rewrites, e.g., replacing a *set* of features by their geometric effect.

In addition to rewrites, each feature stores a set of verification data, used to dynamically identify whether feature import has succeeded or not. Verification data is of three main types: volume and surface area, various higher order moments of inertia of the solid, and a cloud of points lying on the boundary of the solid or on the faces generated (or removed) by the feature alone. Verification data is computed at the source CAD system after invocation of the feature, then stored at the UPR. Verification data is computed at the target system during import and compared to the data stored at the UPR, to verify success of feature import.

When verification fails the system attempts a graceful recovery, e.g. by applying a different rewrite. Note that a source and target system feature may generate geometries that are slightly different but would still be regarded as equivalent for the sake of feature-based data exchange. For example, fillets are not necessarily required to produce the exact same geometry. Hence care must be taken when interpreting verification results.

All of those mechanisms are taken care of at the global architectural level and do not form a part of the selection algorithms, which operate at the feature internal level.

5 Assumptions

In this section we state and explain our assumptions on the context in which the selection problem occurs.

Our goal is to provide support for two-dimensional user selections that serve as feature arguments in feature-based data exchange systems. We assume the following assumptions, which do not pose any restriction on our algorithms because all parametric CAD systems obey them: (i) The FBDE system defines features one after the other in the target CAD system; (ii) When selections are specified the target CAD system holds a boundary representation (Brep) of the current model; (iii) The Brep conforms to the theoretical definition of a Brep, as embodied in the concept of a selective geometric complex (SGC) [Rossignac88]; (iv) The Brep is available for inspection and usage by the selections support algorithm. Selections can thus be specified to the target system in terms of identifiers of current Brep entities.

The model defined by the features up to F (the feature containing the selection that needs to be defined) is thus assumed to be correct in the sense that the pointset at the target system is identical up to a geometric tolerance to that at the source system. The geometric tolerance issue, resulting from floating point computation and inexact algorithms, is one of the major problems in solid modeling and here we do not attempt to deal with it completely.

We also assume here that identity of selection geometry is accessible through the API of the source CAD system. Virtually all modern CAD systems provide such access to Breps. They usually do not provide an interface to the persistent names, but they do provide Brep interfaces to selections.

We do not assume that feature histories or Brep topologies are identical at the source and target systems. Feature histories may be different owing to rewrites or different feature repertoires of the source and target systems. Brep topology (the boundary graph that represents the vertices, edges and faces and their interconnectivity relationships) varies from system to system, as discussed in Section 3.

For simplicity of exposition we assume in this paper that the Brep scheme in both source and target systems is 2-manifold. Extension to non-manifold geometry is beyond our scope here. Regarding terminology, 'face' is used for a single bounded connectivity component of a two-dimensional part of the boundary of the solid, lying on a single carrier surface.

6 The Selection Cover Algorithm

In this section we present the first phase of our 2-D selection algorithm. In this phase we export selection information from the source system, find a target model cover of the source selection, and classify its status with respect to further actions needed to complete a proper selection. We describe the general scheme (6.1), the export process and what data is stored in the UPR (6.2), and the import selection cover algorithm (6.3).

6.1 The General Scheme

We had already noted above that a generic selection representation in the style of persistent naming solutions is not essential in our case, because the geometries on both sides can be assumed to be identical up to a tolerance. It is thus of conceptual elegance to try to utilize only that static geometry. The general idea is then:

- Export the geometry of the selection into the UPR.
- When selections need to be defined during import, select a subset of the current Brep (in the target system) that covers as tightly as possible the selections stored in the UPR.
- If there are faces in the UPR selection that cannot be exactly covered by faces in the current Brep, create new faces or split existing faces in the current model so that an exact cover is obtained, or use other feature rewrites to preserve feature semantics as much as possible.

Section 3 showed that the second step is not trivial. It is not the case that every selected face in the source system corresponds to exactly one face in the target system. We cannot make assumptions regarding the topology of the two Breps, only about their geometry.

6.2 Export to UPR

The goal of the export process is to make sure that all data needed during the import phase is available in the UPR. Since the import algorithm only needs the geometric pointset defining the selection, it is straightforward to represent it in the UPR. Due to the philosophy behind the UPR, which is designed to support the *union* of object types generated by CAD systems, we use the same surface types used in the source CAD system. In principle there should not be any degradation in the quality of the representation, and specifically no loss of tolerance. We refer to the selection as stored in the UPR as the 'source selection'.

In some situations we may export relevant symbolic model data along with the selection geometry. For example, when the model contains several parametric history graphs, the ID of the graph containing each part of the selection can be stored with the selection geometry, in order to make it easier to locate the target image of that body during import or in order to identify the correct body if several bodies overlap geometrically. The IDs of the owning features of each selection part can be used in the same manner (in the terminology of some CAD systems, an owning feature of a Brep entity is the first feature that had caused the entity to be added to the model's Brep). These kinds of techniques are obvious and we will not elaborate on them further in this paper.

6.3 Import: Selection Cover

Recall that the Brep generated by the features preceding the feature whose arguments are our selections is assumed to be present in the target system. What we seek in phase 1 of the import algorithm is a set of entities belonging to that Brep that are an exact cover of the selection geometric pointset. If there is such a set, we are done. We may need to compute the connectivity structure of the entities to be selected as well as the entity set, in case the target system does not allow an arbitrary selection order. If there is no exact cover, we need to know this and provide as much information as possible to phase 2 of the import process, the rewrite phase. For simplicity the algorithm is described for a single connectivity component of the selection set. We assume a preprocessing stage in which all such components have been identified. The algorithm should be invoked on those iteratively. The algorithm targets the case where the selection is comprised of a face or a set of faces; selection of a carrier surface is easily implemented using a point and the normal at the point.

There are many potential methods to compute an exact cover. Below we describe a method that relies on the power of the 1-D selection algorithm from [Rappoport05]. That algorithm computes a cover for each source selection edge separately, using point projection for an initial mapping plus a recursive search based on edge overlap tests. It is possible that the cover computed for an edge is not an exact cover, but the algorithm ensures that the union of the covers of all selection edges is an exact cover of the whole 1-D selection set.

The 2-D selection cover algorithm (Figure 3) is different from the 1-D algo-

```
Import: Selection Cover Algorithm
  Denote by c the 1-D boundary of the whole source selection set.
  Use the 1-D selection algorithm in order to identify c in the target
    system (call it c').
  If c was covered exactly by c'
    Find in the source system a single point p strictly inside the selection.
    In the target system, locate a face f' containing p.
    Recursively tag adjacent faces starting from f' and ending when
      reaching an edge of c'.
    Return successfully.
    // note: the above works even when c is empty.
  Else
    // c contains points that are not in c'
    // (happens when there are missing edges in the target system model)
    // or c' contains points that are not in c
    // (happens when the 1-D selection algorithm finds a cover that's too large)
    // or both
    compute c-c' and give it to a rewrite algorithm.
    // these are edges we want to insert explicitly into the model, if we can.
```

Figure 3: The selection cover phase in the import part of the 2-D selections algorithm.

rithm in that it deals with all selection faces simultaneously, not separately. We start by invoking the 1-D selection algorithm for the 1-D boundary (denoted by c) of the 2-D selection set. The 1-D boundary c is always well defined; it could be empty when the selection set contains all of the boundary of the solid. The reader may be surprised to learn that this case does happen in practice, when users use the Copy Faces feature in order to copy the whole connectivity component of a solid (this is not good design practice when the CAD system provides a Copy Body feature, of course.)

If the 1-D selection algorithm has succeeded in finding an exact cover c' at the target system, all we have to do is collect the faces that its bounds. We can do that by finding an arbitrary point strictly inside the selection, locating a face f' on which it lies in the target system (there may be more than one such face, but it doesn't matter), and performing a recursive search from f' that ends when reaching the 1-D selection boundary c'. Alternatively, we can start the search from any edge in c' in a direction determined by orientation considerations to be in the selection (we would need to synchronize orientations to do that reliably.) Note that this method works even if the 1-D boundary c is empty – it would simply select all faces in the connectivity component of the solid.

Note that the halting criterion is different from that used in the 1-D algorithm, where we halted when reaching an edge that does not overlap the selection pointset. The 2-D criterion is more efficient, because it does not require expensive face overlap tests. It utilizes the power of the 1-D selection algorithm. We could not have used a similar condition in the 1-D algorithm ('halt the edge propagation when reaching vertices that constitute the selection boundary'), because edge propagation can reach a specified vertex in many different and complex paths across the 2-D boundary of a 3-D 2-manifold solid.

If the 1-D selection algorithm has not succeeded in finding an exact cover, it says so and marks source edges that were exactly covered, source edges that were partially covered, source edges that were not covered at all, and target edges that are not fully covered by the source selection (note that here the cover is in the opposite direction.)

Source edges that are partially covered or not covered correspond to edges that are 'missing' (completely or partially) from the target system. Examples are the vertical edge of the drafted face in Figure 1, left, and the straight edge on the cylindrical face in Figure 2, left.

Target edges that are not fully covered by the source selection exist when a target face covers 'too much' of a source face. Examples are the horizontal edges of the drafted face in Figure 1, right, and the circular edges (at the top and bottom) of the cylinder in Figure 2, right.

Any connected two-dimensional subset of the boundary of a 3-D solid is completely determined by its one-dimensional boundary and a single point in it. As a result, if the 1-D selection algorithm is correct (proven in [Rappoport05]) then the algorithm above is correct as well.

7 Rewrites

The second and more complex phase of the import 2-D selection algorithm is when rewrites are necessary because an exact selection cover could not be found. In this section we classify the possible types of rewrites and give examples and algorithms for most of them.

7.1 Face/Carrier Rewrite

The simplest type of rewrite is when the feature semantics really needs only the carrier surface of a face, but for some reason it is not possible to define the carrier in exactly the same manner in both source and target systems.

As an example, take the 'Mirror By Plane' feature, which requires the selection of a datum plane (introduced to the model by some earlier operation), and unites the current model with its mirror across the plane. Suppose that in the source system there is no such feature but there is a 'Mirror By Face' feature, having exactly the same semantics but which requires the user to select a face, not a plane, and uses the face's plane. Suppose that the target system has only Mirror By Plane and does not have Mirror By Face. The selection algorithm from the previous section will succeed finding the face, but the feature import will fail as a whole because the target system doesn't have the corresponding feature.

A simple rewrite of the source Mirror By Face to a target Mirror By Plane will solve the problem. The rewrite should modify the name of the feature and also convert the selection of a plane to the selection of a face. The face is specified by 'any planar face lying on a given plane'. Implementing this selection is easy.

Note that in this case the selection algorithm from Section 6 is not entered at all. This rewrite is done at a higher level in the system, and is a feature rewrite, not a selection-only rewrite.

7.2 Face/Edge Round Rewrite

A situation that is somewhat similar to the previous one is with the Round (Fillet) features. In Figure 4, we see two CAD systems (left, right), before (top) and after (bottom) a Face Round feature (in this case it creates a fillet.) On the right, the fillet is around the full perimeter of the vertical cylinder, because the cylinder is represented as a single cylindrical face. On the left, the cylinder is broken into two faces, and the fillet does not go beyond the thin box.

When exchanging from left to right, we get a wrong result. In this case rewriting the Face Round feature to an ordinary (edge based) Round feature solves the problem. The rewrite should identify which edges between the two selected faces were covered (replaced) by the fillet, and provide those edges to the Round feature on the right.

Note that this solution is not always possible, because it is in principle possible that there are no intersection edges between the two faces of a Face Round feature; the power of this feature is in defining long distance fillets and rounds (imagine the intersection edge on the left as a small face completely obliterated by the fillet.) If this is the case there is no simple symbolic solution, and the situation can be addressed by the rewrite described next (Edge or Face Insertion), or by the general Geometry Per Feature rewrite [Spitz04].

7.3 Edge or Face Insertion Rewrite

The most common case in which the algorithm of Section 6 fails is when there are 'missing' edges in the target system. A direct rewrite for tackling this problem is to explicitly insert those edges into the model. Some CAD systems support a 'split face by adding a new edge' feature. In this case this feature is used for inserting the required edges.

This is a nice example of the power of the rewrite concept in the UPR architecture – as we see here, a rewrite can insert a wholly new feature if it is needed in order to enable the parametric import of another feature into the target system.

Unfortunately, not all CAD systems provide to the user a Split Face feature. It is still possible that this operations is available to CAD extension programmers through the CAD system's API, and can be added internally to the feature graph. In this case we proceed as before.

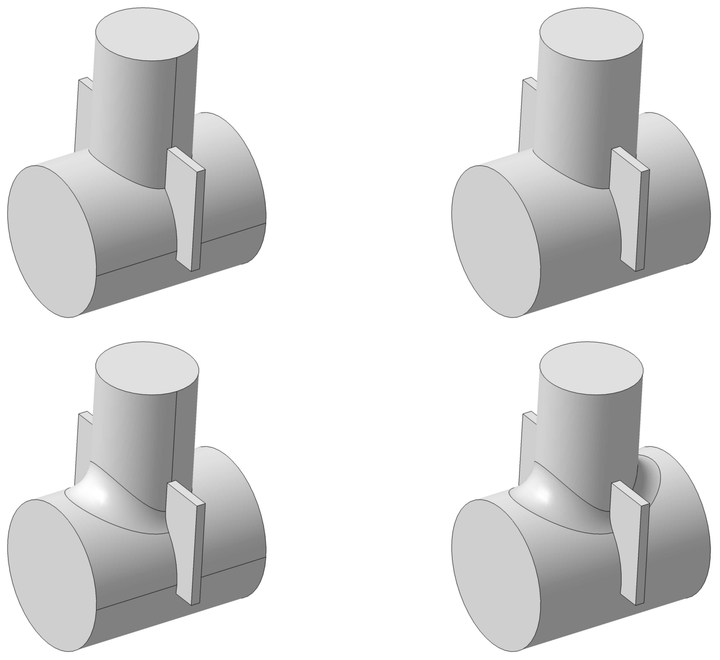
Figure 4: Face Round. Two CAD systems (left, right), before (top) and after (bottom) of the Face Round feature.

When the above is not possible, in many systems it can still be emulated. Many systems contain a Patch feature whose arguments are a solid Brep B and an open surface sheet S having a material side and boundary edges that are assumed to lie on the boundary of the solid B. The result of the feature is to glue S to the boundary of B and discard that part of B's boundary that lies on the non-material side of S. We had used the Patch feature in our algorithm for solving the Geometry Per Feature (GPF) problem [Spitz04].

To emulate Split Face using Patch, we first prepare surface sheets that coincide with part of the target face and whose boundary edges include the desired selection edges. We compute these sheets by traversing the edges returned by the algorithm in Figure 3, and piecing them together to form connectivity components (there may be several connectivity components, and each is patched into the model using a different Patch feature.) Once we have those edges, we intersect them with the carrier surface of the target system face, creating trimming curves that specify the sheet to be patched in a well defined manner. Equivalently, we can simply invoke a face-face intersection algorithm [Patrikalakis02] between the source and target faces. However, care should be taken in the implementation of this algorithm, because the faces are known to overlap substantially, possibly resulting in numerical problems for the intersection algorithm.

Having computed the desired sheets to be patched, the material side for each is specified to be identical to that of the face. We then use each of these sheets as the argument of a Patch feature. This process usually results in the desired selection edges being added to the current Brep.

Figure 5 shows an example (the full solid is shown at the top, and cross sections in the middle and bottom.) A circle is sketched on the top planar face, and an Extrude Cut Until Face feature is performed. The face that serves as the 'until face' is a cylindrical face. However, when the cylinder is represented using only a single face, it is not clear whether the cut should stop as in the middle row or as in the last row. Initially it seems that the latter is wrong, but the reader should note that the intersection of the initial box with the cylinder creates a loop in the top part of the cylinder, so the Extrude Cut passes through that loop. The problem occurs whenever the cylinder (or any other non-planar surface) is represented using a single cylindrical face (on the left, the face contains a single edge, and on the right it does not contain any such edges.)

The problem can be solved by preparing the desired target face and patching it to the model. The desired target face is represented as the geometry of the place where the Extrude Cut is supposed to stop, which is the intersection of the Extrude Cut section and the desired part of the cylinder. That face, after the Patch, is used as the Until Face of the Extrude Cut in the target system.

Unfortunately this solution is not guaranteed to work in every CAD system, because it depends on the specific implementation of the patch feature and on the target system's general Brep policy. Many systems actively unify faces by removing edges separating them when those faces are considered to be the 'same' face, e.g., when they are co-planar. The system may refuse to do the Patch simply because it detects that no portion of the previous Brep is removed and hence the feature is considered redundant. The problems here are the classic problems of what is considered by CAD systems to be a valid face [Mäntylä88].

Figure 6 shows another example in which Patch solves the problem. However, in this case the problem stems from a totally different reason. Suppose that before the Shell, the object is defined as a box from which a slot is subtracted. In some systems, the associativity between the two top faces is retained to an extreme extent – the system remembers that they were originally the same face, and does not let users select each one individually. In such systems, exchanging the middle row is not possible, and the bottom row is the only possible outcome.

In such cases what we need to do is 'disconnect' the associativity between the problematic faces. In all of the systems that we handled, this can be done by patching into the model an orphan face identical to one of the top faces. Because the patched face can have an arbitrary geometry, the CAD system no longer assumes that the patched face lies on the same plane as the other top face, and breaks the associativity requirements between them. Now selecting only the patched face is possible, so Shell produces the desired results.

This example is of course related to the 'real' persistent naming problem, but implementing selections in terms of persistent naming (instead of in geometric terms like we do in this paper) would not solve the problem – in any case the two faces have the same persistent name and this must be broken by applying

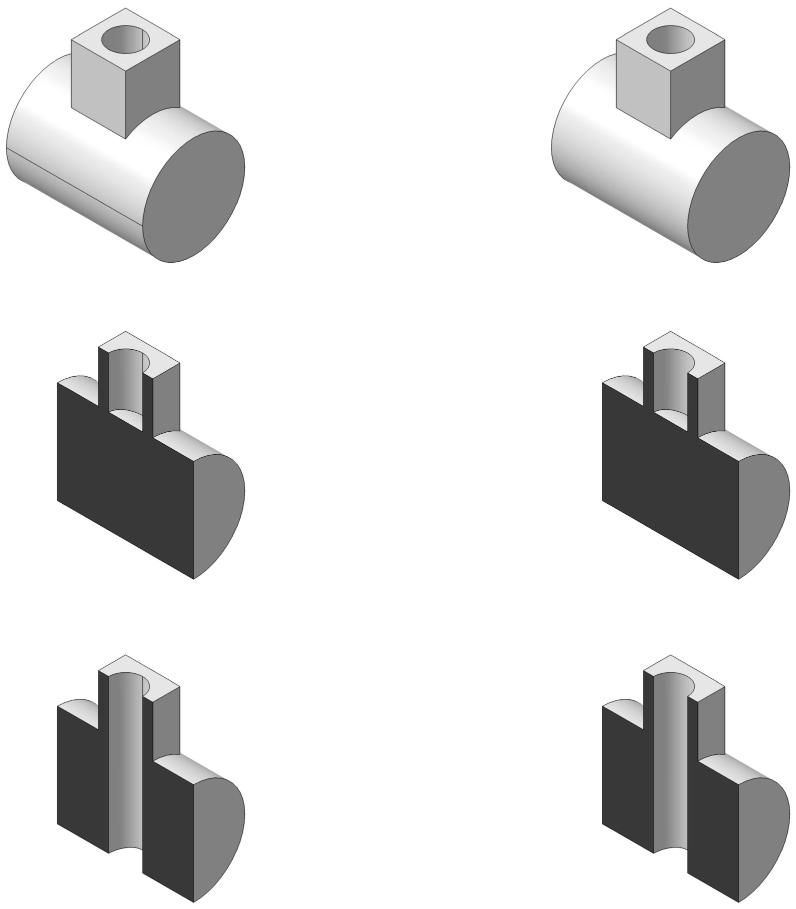

Figure 5: Until Face. Top: solid view of an Extrude Cut of a circle sketched on the top face, defined to be Until Face of the cylinder. On the CAD system on the left, a cylinder is represented using three faces (two circles and one cylindrical face) and three edges (two circular edges and one straight edge). On the right, a cylinder is represented using a single cylindrical face, two circular faces, and two circular edges. Middle: cross section of one option for when the Extrude Cut stops. Bottom: cross section of another option for when the Extrude Cut stops. (On the left, cylinders are represented using an edge that cuts across the cylindrical face, and on the right there are no such edges.) An example such as this can occur with any non-planar surface, not necessarily a cylinder.

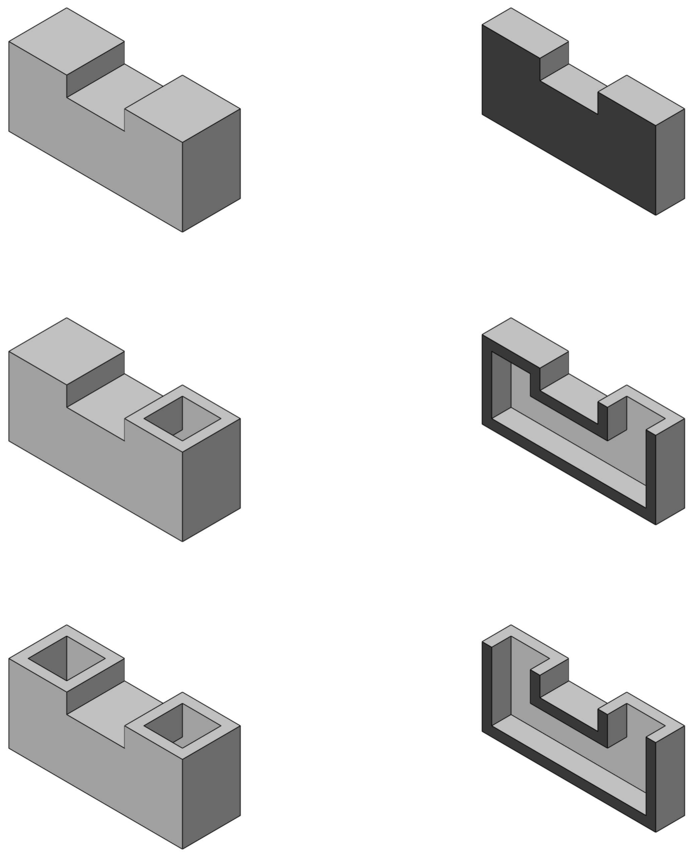

Figure 6: Shell. Left: solid view. Right: cross section view. Top: before a Shell feature. Middle: the selection argument of the Shell feature is the top right face. Bottom: the selection argument of the Shell feature are both top faces. In some CAD systems, the middle option is not possible when the two top faces have originated from the same operation.

an explicit operation that creates another name to one of the faces.

7.4 Orphan Face Insertion Rewrite

In some cases, it is sufficient to add the desired until face as an ‘orphan’ (or datum) face of the part, without actually patching it so that it becomes a part of the solid’s Brep. For example, in Figure 5 this is possible. In Figure 6 it is not possible, because the argument of the Shell feature must be a Brep face of the solid.

Note that both options exhibit the same degree of loss of associativity, so the orphan option may be preferable due to performance and tolerancing considerations. Another consideration is which option is better from the point of view of user interaction in case people would need to look at the resulting model. In both options the feature graph at the target system does not look the same as that in the source system, and it is unclear which option is less confusing. We tend to feel that the orphan option is better because it uses fewer operations: it only uses one additional feature (inserting the orphan), while the Patch option uses two additional features (inserting the orphan and then patching it).

7.5 Reparameterization Rewrite

Consider again Figure 6. What if the target system does not allow patching a new face and calling it a different face? in other words, what if the associativity between the two top faces simply cannot be broken?

In this case we would need to rebuild the solid using different operations. For example, instead of defining a box and removing a slot from it, we could have sketched the vertical face as a 2-D sketch and use the Extrude feature in order to create the solid. In both cases the same pointset is created using two features, but the parameterization (feature history) is totally different.

General reparameterization of feature-based solids is an extremely difficult open problem in solid modeling. In our opinion it is one of the fundamental problems of the field, whose solution would throw light on many aspects of general shape modeling. It is certainly beyond the scope of the present paper. We have given the above example in order to complete the description of cases in which 2-D selections might need rewrites.

7.6 Summary

To summarize this section, in some cases rewrites can be completely symbolic. In most cases, our capability of creating selection faces explicitly where they did not previously exist is a function of the feature repertoire of the target system. When the target system does not allow a direct or emulated Split Face feature, it may still be possible to import the feature parametrically, by replacing it by a totally different feature combination that achieves the same geometric effect. This should be examined on an individual feature basis. The UPR architecture enables doing that through its support of feature rewrites, using a single feature or a number of features.

8 Implementation

The 2-D selection framework described in this paper has been implemented in the UPR architecture at Proficiency. The current UPR implementation supports the five high-end CAD systems in the market: Catia V4, Unigraphics, I-DEAS, ProEngineer, and Catia V5. Several versions and most of the design features of each system are supported. The data exchange process is controlled by a web server through a web-based user interface. The server locates export and import 'agents' over a network and distributes export and import jobs according to load parameters. The software is being used routinely in production. The number of real parts that have been successfully exchanged is in the hundreds of thousands.

In our implementation, the UPR file stores all relevant data, including the selection data, represented geometrically as described in this paper. Intermediate computations are done on the UPR data structure or using the software library of the target CAD system, according to implementational convenience.

We implemented the rewrites concepts of the first three types (face-carrier, adding edges, adding faces). Patching faces is done as part of the general Geometry Per Feature rewrite. Faces are added as separate bodies (orphans) when needed. Missing edges were added for import into I-DEAS. The fourth rewrite type was implemented for the very small number of cases that were encountered in practice.

9 Discussion

The issue of supporting two-dimensional selections is a crucial one in feature-based data exchange systems and algorithms. 2-D selections are used as feature arguments in important features such as Extrude, Draft, Offset, Shell and Face Round. Selections are what endows models with true associativity, and FBDE systems must support selections in a way that is as close as possible to the design intent as expressed in the source CAD system.

In this paper we have presented the first solution to this important problem, following our solution to the twin problem of 1-D selections [Rappoport05]. Our solution is applicable to a wide variety of FBDE architectures, among them the UPR and the STEP architectures, which are the only documented ones at present (note that the UPR has been fully implemented in practice, unlike STEP.) Our algorithms have been implemented in the UPR architecture, and are being used on a daily basis in real projects.

An interesting future work on our problem is to base the algorithm on the persistent names used by CAD systems rather than on geometric data alone. At present CAD systems do not expose those names, but the reliability and perhaps performance of selection exchange could be increased when utilizing persistent names.

A highly challenging topic arising from our problem (as well as from other problems) is that of reparameterization of parametric feature-based models. This topic is both very deep theoretically and has useful practical applications.

Acknowledgements. The Proficiency UPR implementation is a collective effort of the Proficiency development team, headed by Alex Tsechansky.

References

Hoffmann93 Hoffmann, C.M., Juan, R., Erep, an editable, high-level representation for geometric design and analysis. In: P. Wilson, M. Wozny, and M. Pratt, (Eds), Geometric and Product Modeling, pp. 129-164, North Holland, 1993.

Kripac97 Kripac, J., A mechanism for persistently naming topological entities in history-based parametric solid models. Computer-Aided Design, 29(2):113–122, 1997. Also: proceedings, Solid Modeling '95, pp. 21–30, ACM Press, 1995.

Mäntylä88 Mäntylä, M., An Introduction to Solid Modeling, Computer Science Press, Maryland, 1988.

Mun03 Mun, D., Han, S., Kim, J., Oh, Y., A set of standard modeling commands for the history-based parametric approach. Computer-Aided Design, 35:1171-1179, 2003.

Mun05 Mun, D., Han, S., Identification of topological entities and naming mapping for parametric CAD model exchange. Intl. J. of CAD/CAM, 5:69–82, Dec. 2005.

Patrikalakis02 Patrikalakis, N.M., Maekawa, T., Shape Interrogation for Computer-Aided Design and Manufacturing. Springer Verlag, 2002.

Pratt04 Pratt, M.J., Extension of ISO 10303, the STEP standard, for the exchange of procedural shape models. Proceedings, Shape Modeling International 2004 (SMI '04).

Qi04 Qi, J., Shapiro, V., Epsilon-solidity in geometric data translation, TR SAL 2002-4, Spatial Automation Laboratory, University of Wisconsin-Madison, June 2004.

Rappoport96 Rappoport, A., Breps as displayable-selectable models in interactive design of families of geometric objects. Geometric Modeling: Theory and Practice, Strasser, Klein, Rau, (Eds), Springer-Verlag, pp. 206-225, 1996.

Rappoport97 Rappoport, A., The Generic Geometric Complex (GGC): a modeling scheme for families of decomposed pointsets. Proceedings, Solid Modeling '97, May 1997, Atlanta, ACM Press.

Rappoport03 Rappoport, A., An architecture for universal CAD data exchange. Proceedings, Solid Modeling '03, June 2003, Seattle, Washington, ACM Press.

Rappoport05 Rappoport, A., Spitz, S., Etzion, M., One-dimensional selections for feature-based data exchange. Proceedings, Solid Modeling '05, June 2005, MIT, ACM Press.

Rossignac88 Rossignac, J.R., O'Connor, M.A., SGC: a dimension-independent model for pointsets with internal structures and incomplete boundaries. In: Wozny, M., Turner, J., Preiss, K. (eds), *Geometric Modeling for Product Engineering,* North-Holland, 1988. Proceedings of the 1988 IFIP/NSF Workshop on Geometric Modeling, Rensselaerville, NY, September 1988.

Shah95 Shah, J.J., Mantyla, M., Parametric and Feature-Based CAD/CAM, Wiley, 1995.

Spitz04 Spitz, S., Rappoport, A., Integrated feature-based and geometric CAD data exchange. Proceedings, Solid Modeling '04, June 2004, Genova, Italy, ACM Press.