Non-manifold and special purpose modelling

Ian Stroud

Present address: **École Polytechnique Fédérale de Lausanne,**
DGM-IMECO-LCAO,
DGM Écublens,
CH-1012 Lausanne,
Switzerland

1. Introduction

1.1 Background

Originally boundary representation modelling required that models be "two manifold", that is, be point sets with the property that at any point on the boundary a small enough sphere is cut into two pieces, one inside and one outside the object (figure 1.1)

Another definition that is used is that at any point on the boundary the region surrounding the point is homeomorphic to a disc (two-connected condition).

This condition facilitates operations and also helped early development by reducing the number of special cases that had to be considered. The manifold condition is essentially a geometric condition so it facilitates geometric algorithms. Later on it became clear that this condition can be relaxed to allow a wider variety of models to be used an it is this type of relaxed models and the effects that this tutorial is about.

Although it is a fairly basic topic, it is, perhaps, necessary to do a brief recap of the boundary representation data structure as a background to later descriptions. The (almost) original datastructure was the "winged-edge" datastructure developed by Baumgart and adapted by Braid for solid modelling in the BUILD system, [Brai79] (fig 1.2). This structure maintains links to the neighbouring edges, the right and left loops and the start and end vertices.

Another structure which has grown in popularity uses so-called "half-edges" (or "coedges, or "loop-edge links, or ...) fig 1.3. This representation has gained popularity because it is possible to have a ring of "half-edges" round the edge so as to allow more than two faces to meet at the edge (Weiler [Weil85]). Although true, a personal view is that this is not an optimum representation, as will be discussed later. However this representation greatly facilitates traversing loops of edges, an important operation. In fact, the winged-edge representation, with some extensions, can also be used for non-manifold modelling. The third type of datastructure which can be mentioned is the "edge-vertex link" structure which uses links between the edge and the vertex to record connectivity (figure 1.4). This is the least used of the three datastructures.

The special representations described in this tutorial all need the connectivity information recorded in the data structure to a greater or lesser degree, hence the reason for describing them. Please note, though, that all three are equivalent, there is no need to restrict discussions to one single type.

The original winged-edge structure assumed that there were exactly two faces at an edge. For a normal object this is correct and it was conceived as a representation for modelling normal objects for computer vision. However, when applied in CAD etc., there are other needs and so more flexibility is required. Even by the end of the 1970's compromises had to be reached. Sheet objects were modelled using a 'rubber' face on one side, for example. The rubber face was used to preserve the topological necessities but it had no geometry and was used solely to 'close' the model so that every edge had two faces [Brai79]. Wire models were also allowed, but these were still 'Eulerian' became sheet objects when closed, so were rather specialised examples of the general model structure. A much bigger break came with the need to integrate real wireframe, surface and solid models in one system so as to provide a wide range of design tools ([Kjel82], [GPM81]). Wireframe models were supported in the GPM system by allowing them to have no faces. Sheet

objects were represented using back-to-back faces so that they function like geometrically degenerate volumetric objects.

Weiler's dissertation [Weil86] presented a so-called "radial edge" structure of links round an edge to allow multiple faces to meet there. In a strict structure there should be an even number of faces round the edge, arranged alternately as "left" and "right" edge-face relations - meaning effectively that a set of non-overlapping wedges of material meet at the edge. (The information about "left" or "right" is recorded in the link connecting the edge and a loop of the face). The notion of 'wedges' is explicitly used in the non-manifold datastructure proposed by Luo and Lukacs [LuLu91].

So, to sum up, the modelling datastructure should be able to support structures with 0, 1 or an even number of faces meeting at any edge.

The different types of model are illustrated in figures 1.5, 1.6, 1.7, from [Stro94]. These figures illustrate the interchangeability of the representations as well as the representations themselves. This is important to note because it means that the same basic functionality can be supported in different ways by the underlying modelling methods. The figures illustrate the different interpretations of the different types of model. So, in figure 1.5, the non-manifold sheet model with radial edge structure at the top can be interpreted as a degenerate sheet model in two ways, either as a connected set of faces, or as four distinct pieces, depending on how the multiple half-edges are split into pairs. Similarly for a partial model, the internal edges have two half-edge links; the boundary edges just one. The cubes in the top picture with the single non-manifold edge can be interpreted as joined structures or equally well as two separate cubes. The other two figures show similar results for interpreting degenerate models as radial edge or partial models, and the method for building back the undefined side of a partial model to form a degenerate model.

1.2 General tools

There are a number of general tools which are useful generally for modelling. These are tools for finding connected sets of entities (e.g. “all edges in body”, “all edges round vertex”, etc.), an entity with respect to one or two other entities (e.g. “edge clockwise from edge round loop”, etc.). There are similar tools for use with radial-edge non-manifold objects, e.g.

all links round edge
all loops round edge

which can be added to the normal set. Other operations, such as “all edges round vertex”, need to be adapted to take account of cases where there are multiple edge sets at the vertex.

Another important operation is for sorting links around an edge in the radial edge type of model. The links should be arranged in order, clockwise or counter-clockwise round the edge, depending on the edge’s orientation. The links are ordered using the geometry of the faces corresponding to the link. A vector into each face from the edge is used for sorting, the vector for first link giving the zero angle vector, the other angles calculated relative to this. The order of the links round the edge is important so that there is no ‘material within material’. Normally the links are arranged in pairs round the edge indicating entering material or leaving material, depending on the orientation flag of the link compared with the edge. Looking at the links round the edge, there should never be two adjacent links implying entering material. This is especially important in sheet object models

where the geometrical information only is not enough, because there are two back-to-back faces which have the same incidence angle at the edge. Note, also, the potential difficulties that might occur when several sheets have the same incidence angle at the edge. This should be comparatively rare, but might occur.

1.3 Euler operators

It is worth noting, briefly, the work of Luo and Lukacs ([LuLu91], [Luo92]) on extending the B-rep data structure. The important addition in their structure was the addition of what they called "wedges" and "bundles" for non-manifold edges and vertices. These can be thought of as equivalents of "loops" in faces for representing multiply connected boundaries. Every edge has at least one wedge (pair of loop-edge links), non-manifold edges have more than one. Similarly, every vertex has at least one bundle of edges, non-manifold vertices have more than one bundle. Luo92 also describes an extensive set of Euler operators to create and manipulate models with these extra elements.

The important thing about this structure is that it removes the inherent ambiguity of the radial-edge structure by associating loop-edge links in pairs. This is important for being able to differentiate between the cases where two objects just touch and where they just touch each other. For the radial edge structure the interpretation depends on the way the loop-edge links are paired during an operation, and this is not well defined. With the Luo and Lukacs data structure this is explicit.

1.4 User philosophy

With complete volumetric, manifold models the model can be treated with general geometric and topological operations. This is not always true with these special representations, hence it is necessary to have a 'philosophy' of their use so as to be able to handle special cases. One example of this is when wire models are used to represent, for example, the centre-lines of pipes. The model is, in reality, made up of two parts, a geometric part represented using the wireframe model and an implicit, unevaluated part. Similarly there is an interpretation of sheet objects as being specialised representations of thin plates, the 'sharp' edges of these corresponding to narrow faces in the volumetric model. The operation to offset these models works well enough when the offset is small, but complications can arise when the offset becomes too large. In effect, the idealisation starts to break down.

Handling general special representations requires more attention and is more difficult than working with special representations for particular applications. With particular applications the user can be restricted or advised about special conditions instead of having to cope with them later.

2. Wireframe modelling

Wireframe models are the most difficult of the special representations to handle. They consist of edges and vertices only, so the specific adjacency information related to faces is missing. Wireframe models can be useful for several purposes, as design sketches or as idealisations, but their use has to be restricted to avoid interpretation problems or ambiguity.

2.1 Uses of wireframes

There are several ways in which wireframe modelling techniques can be used in an integrated system. These are outlined below and then techniques for handling them will be described in the next section.

2.1.1 Wireframes as object models

Wireframe models were used for some time in CAD systems for representing three-dimensional models. They contained all the essential lines needed, but the simplified structure which made their implementation easy also implied that erroneous or ambiguous structures could be built. Models of Klein bottles or Möbius strips can be created as wireframe models. Ambiguous models such as Requicha's example (figure 2.1) can also be created. In effect there is a loss of information between the designer and the system, the designer's intention is not entirely recorded and the system could not advise him/her of his/her misconceptions.

2.1.2 Wireframes as design sketches

Another way of using wireframe models is as design sketches. The designer is allowed to build up a rough model with a series of lines which are gradually refined. The reason for using a wireframe model is to avoid requiring that the model be disconnected and so disrupt the way the designer works. See, for example, figure 2.2. Another possibility is to let the designer create a partial model, say the rotational profile, and have the system complete the shape automatically. Here, though, there appears a direct conflict between flexibility and support. The rotational profile could be swept directly to produce a sheet object instead of being made into a closed shape and swept to produce a rotational solid. From a modelling point of view it is not possible to exclude either option so it is necessary either to restrict the user or offer alternatives, which increases system complexity.

2.1.3 Wireframes as idealisations

This is the clearest way of using wireframe models. The model itself can be thought of as a 'symbol' which stands for a more complex shape. An example of this is the use of a wireframe model as the centrelines of pipes. In this case the wireframe is used mainly as a repository for geometry and the connectivity information is sufficient to determine interactions.

2.2 Wireframe modelling

Wireframe models use the same basic edge-vertex structure as complete B-reps, but the loop/face structure is missing and hence the connectivity round vertices works in a rather different way.

As already stated, the big drawback with wireframe models is the lack of complete information. For this reason it is sometimes necessary to rely on geometric tests, which are not always guaranteeable. As the complexity of the wireframe model increases so does the potential for intractable problems. For this reason it is necessary to be clear about the use of wireframe models and to limit their application to simple tasks. The user should not be allowed to create very general models, investing time and effort into something which may contain errors and need a lot more effort to convert later.

2.2.1 The wireframe structure

It is possible to have Eulerian wire objects. These were built up using Euler operators and were used in the first systems to make general shapes which were then swept into solids, for example. These had left and right loop pointers, as with other edges, but they pointed to the same loop. When the loop was closed a sheet object was made (figure 2.3).

A normal wireframe structure has vertices and edges only. The first step is to decide whether to represent the wireframe model as a separate model or have it as a special case of a general model type. It is probably better to have it as a separate model type.

The next decision is whether the wireframe model is a single piece or whether it can consist of several separate pieces. If a single piece only is allowed then all edges and vertices are connected. If multiple pieces are possible then it is absolutely necessary to link all edges and/or vertices into a chain so that all pieces can be found.

To implement edge-vertex connectivity, each vertex points to an edge and the edges are linked in chains round the start and end vertices, using winged-edge pointers or linked half-edges (figure 2.4). However, the order the edges are linked round the vertex need not (and probably does not) have any special significance. Not, also, that there is not the same need for multiple edge sets at the vertex as with non-manifold objects, it is to be expected that all edges at the vertex form a single chain.

If a wireframe model has multiple pieces then some conversions may require that these be separated as a preprocessing step. The separation process is relatively simple and is analogous to the process of separating solid or sheet pieces. The first edge in the wireframe is selected and all connected edges put into a list. Then the chain of edges in the wireframe is scanned, if an edge is found which is not in the list then this belongs to a separate part and the edge and all connected edges are moved to a new wireframe. The process is repeated until all edges have been checked.

2.2.2 Setting wireframes into surfaces

A normal requirement for this is that there are only two edges at every vertex. This requirement makes it easy to determine connectivity information between edges. If it is known that a closed

form is to be created then it can be better to create the shape using Euler operators to begin with, as in figure 2.3.

The first step is to separate the wire frame model into separate pieces, as indicated in the previous section. Then, at every vertex, the two edges meeting at the vertex are set to point to each other. When all vertices have been processed there will be two loops of edges, one with implicit orientation in the same direction as the surface, one in the opposite direction. The appropriate faces are then assigned pointers to the given surface and a negated copy.

The reason for the restriction is to avoid having to make difficult geometric decisions about adjacency. The difficulty here is to determine what is 'outside' and what is 'inside'. Another consideration is that it is not possible to guarantee that general graphs are planar. A simple example, from L. Wingård, is shown in figure 2.5. The user should not be allowed to develop too complicated a wireframe model before converting.

2.2.3 Sweeping wires

Another method for converting wireframe models into sheet objects is to sweep them. Note, though, that visually the result may be difficult to distinguish from a solid produced by sweeping. The final result will depend on the choice of datastructure. The algorithm here produces a degenerate model as result, but it is relatively simple to convert this to a radial structure if necessary.

The process can be thought of as sweeping a very thin lamina made by splitting every edge of the wire and the non-terminal vertices appropriately (figure 2.6). It isn't actually done quite this way, but that is an alternative way of visualising the process.

The algorithm sweeps each edge of the wire separately. If no neighbour of the edge has been swept then the case is as in figure 2.7a. Two new edges are extended from the start and end vertices of the edge, new edges 2 and 3, and the extruded vertices closed with a new edge, edge 4. Half-edges are added to the wire or winged-edge pointers set appropriately to create two back-to-back faces, one with edge sequence 1 3 4 2, the other with edge sequence 1 2 4 3.

If one neighbour has been swept, as in figure 2.7b, then there is an extruded edge already, edge 2 in figure 2.7b. Effectively, edge 2 has to be split so that edge 2 lies between one previously created face and one new face, edge 4 lies between the other previously created face and the second new face. A new edge, edge 3, is added at the unextruded vertex as before. The two extruded vertices are joined with a new edge, edge 5, giving two new faces with edge sequences 1 2 5 3, and 1 4 5 3.

If neighbouring edges at both start and end vertices have already been swept, figure 2.7c, then both extruded edges have to be split in the same way as just described. The two extruded vertices are joined to create two new faces with edge sequences 1 3 6 4, and 1 2 6 5 respectively.

A special case occurs when there are more than two edges at a vertex. Here it is necessary to perform a geometric test to determine the adjacencies of the extruded faces.

2.2.4 Wireframe idealisations to solids

Wireframe models as idealisations are easier to handle than other conversions because the wireframe model is, essentially, only a repository of simple geometric information. The start and end positions of the wireframe edge, and the curve of the edge are used to generate solid pieces which are then put together by Boolean or 'local' Boolean operations.

With the normal Boolean operation the 'interaction boundaries', that is the boundaries where the objects cut each other, are determined and then the objects are joined, appropriately, at these boundaries. With a special Boolean a single boundary, or a reduced number of boundaries, are created by finding one portion of the boundary and extending that until the whole boundary is complete. The objects are then joined along this boundary only. This is illustrated in figure 2.8. It is also possible to create the final object by converting all wireframe edges (and possibly vertices) to solids and then using Boolean operations to join the separate pieces.

2.2.5 Wireframe to solid conversion

This is a bad idea. As already indicated, it is not advisable to allow very complicated models to be created before converting them. It is easy for errors to creep into the model because the modelling operations cannot detect faults when they are made. Another problem is that the models can be ambiguous or impossible to realise as a solid.

The problem with converting wireframe models to solids is to determine the adjacencies, round vertices or round face loops. If, at a vertex, the direction to the outside of the object then the projections of the edges at the vertex can be used to order the edges clockwise or counterclockwise round the vertex. The problem is that this direction is not always known. Finding cycles of edges bounding faces is even more difficult. Several possible cycles of edges can be found not all of which bound exterior parts of the object.

In general it is difficult to guarantee success, hence it is not wise to allow wireframe models to get too complicated.

2.3 Conclusions

Because wireframe models lack complete information it is wisest to use them with care and in limited applications only. Often an operation to augment a wireframe model means a loss of information because the designer's intention is not recorded. Wireframes can be used for creating design sketches, up to a point, and also as idealisations of solids. For closed shapes and for general building other techniques should be used.

3. Sheet objects

Sheet objects can be thought of as special types of volume objects, but, obviously, because of the geometric degeneracy there are special conditions.

As already stated in section 1.1 there are three ways of representing sheet objects: 1. using radial edges; 2) using degenerate models; and 3) as partial objects. Since these are approximately equivalent there is no need to make great distinctions, though some differences will be mentioned. In general, though, the radial edge and degenerate models have the same basic properties, being double-sided. Partial models have other special uses.

3.1 Uses of sheet models

Sheet objects can be used as intermediate objects, created from wireframe models by, say, sweeping or setting into surfaces, which are, in turn, swept into volumes. This transient use of sheet models is not worth distinguishing specially. Sheet objects are more interesting as idealisations or when used for special purposes.

Radial-edge and degenerate models are double-sided, hence are useful as intermediate models and as idealisations of thin plate models. As idealisations of thin plate objects the outer edges of the sheet objects correspond to degenerate faces. Note that the idealisation is valid when the plates are thin, if the plate thickness is too great then the topology of the expanded volume model will be very different from the sheet model topology, see figure 3.1 for example. Although this could be dealt with by making a sufficiently complicated algorithm it is certainly necessary to question the validity of doing so. If the offset is comparatively great then it seems dubious to make the idealisation in the first place. It is, again, a question of directing the user rather than being too dogmatic about the modelling technology.

One interesting use of partial models is as features which can be stitched into solids as part of design-by-features, as done by Ranta et al. [RIKM93].

3.2 Modelling with sheet models

Several algorithms for sheet models are dealt with in this section. This is not a complete list, it is meant as a sample to illustrate the general techniques.

3.2.1 Sweep extruding sheet models

The external edges of the sheet object can be extended by sweeping in a similar to sweeping wireframe models (figure 3.2). The process is slightly different because the sheet object already has an orientation so it is possible to use Euler-type operations to perform the operation.

As with wireframe sweeping, each edge is swept separately. To sweep an isolated edge (figure 3.3a) the edge is 'sliced', i.e. split lengthwise to produce a degenerate face with two edges. Two

extension edges are added with Make Edge and Vertex operators and the operation finished with a Make Face and Edge operation.

If one (figure 3.3b) or both (figure 3.3c) vertices of the edge already have extension edges then the process is more complicated because the vertex has to be split, too. As before the edge is first sliced to produce a degenerate face. The adjacent extension edge or edges are also split, and then the vertex or vertices with the extension edges are also split. If only one vertex was adjacent to an extension edge then an extension edge is added at the other vertex. Finally the extended vertices are joined with a new edge. This is easier to illustrate with the figures than to describe in words.

The final consideration here is what to do if the sheet model is represented using a radial edge structure. One way of doing this is to perform the algorithm as above and then collapse back the edges as a postprocessing step. Another way is similar to the method used to sweep wireframe models. Each edge has to be swept separately, as before. Each swept edge generates two new faces. If the start or end vertex or both has been swept then the corresponding extended edge is used, otherwise a new edge is added at the vertex. The two extended vertices are joined with a new edge. At each edge, the original edge, the two extended edges and the closure edge, two new loop-edge links are added. The order of the links round the edge also has to be determined, they should be ordered clockwise or counter-clockwise according to the basic principles of the modeller.

3.2.2 Expanding sheet models to solids

This operation is most appropriate when the offset is small. It is intended to produce an evaluated volume model from a sheet idealisation. Another important consideration is the degree of reality which is required. If the expanded sheet model has sheet pieces at different angles then it may be more realistic to have rounded pieces between the plates to simulate bending operations instead of having sharp transitions. This can be done as a post-processing step.

The easiest model type to change in this way is the degenerate form. This is because the internal structure already exists, it is only the external edges and vertices that need to be changed to created degenerate faces. If the model is represented using radial-edge links then the first step is to process all internal edges, converting the structure to a degenerate form by creating new edges and linking them into the data structure so that each edge has only two loop-edge links.

Once the structure is in the degenerate form the process of converting it consists of slicing the external edges and their start and end vertices to produce the narrow side faces. Each edge is sliced separately (figure 3.4a). If an edge at either the start or end vertex has already been sliced then the vertex, too, is sliced (figure 3.4b). Normally there should be only two external edges at a vertex, but there are cases when there can be three or more. In these cases the vertex splitting process is slightly more complicated.

The two-edge vertex split is straightforward, it is a normal vertex split operation. Starting with one of the sliced edge halves, all edges between that and the next slice edge are moved to a new vertex and the original and new vertex joined with an edge (figure 3.5).

The three or more edge split vertex is more difficult because the vertices have to be joined correctly. The split has to create or extend a central face, as indicated in figure 3.6. This is another minor detail which is fairly easy to overcome. It means that it is necessary to be able to identify the

internal edges created when splitting vertices. If there is only one such edge then this is sliced before starting. Then, as before, starting with the appropriate half of the sliced edge, the process works, say, counter clockwise round the vertex moving edges to the new vertex until one of the internal edges has been moved.

The geometry of the expansion process also needs to be handled. In many cases this can be done analytically but in some cases, when the sheet object surface is a free-form surface and the boundary curve is non-planar it may be necessary to use numerical geometry.

For simple geometry the curves of the external edges, sliced to create the narrow side faces, are offsets of the original curve in opposite directions, determined from the surface normals. With complicated geometry it is necessary to compute a set of offset points and fit a curve through these. For the remaining edges the curves are offset, if the adjacent faces lie in the same surface, or calculated as intersection curves from the adjacent offset surfaces.

The surfaces of the side faces can, in many cases, be determined as the surface generated by sweeping a straight line along the curve of the edge. For some cases, though, with very complicated geometry it may be necessary to calculate a set of points on the surface and fit the final surface through these.

The surfaces of the other faces are calculated as offsets of the original surfaces. Once these have been created it is possible to determine the curves of the internal edges by intersecting these.

Another consideration concerns the 'realism' of the process. In a realistic volume model of a thin plate object the internal edges would often be rounded because the plate would be made by bending. This can be done as a post-processing step, blending the internal edges with suitable radii.

It is also necessary to consider what happens if faces disappear during the offset process. This is certainly a risk, especially with large offsets, but it is necessary to consider carefully what should be done. It would be possible to take the self-intersecting model and evaluate it using Boolean-like operations. Another possibility is to flag an error and let the user adjust the model or offset radius before trying again. The latter possibility may be preferable, since the idealisation is most valid for thin plate models involving small offsets. The evaluation method remains a possibility if the user really does want an automatic solution, but this may produce strange results, with plate models cutting into or through each other.

3.2.3 Creating sheet models from solids

The operation is useful for creating a thin plate model from a solid representing a housing, for example. This is only appropriate for creating degenerate or radial edge sheet models from a solid because the partial form is identical to the original solid.

This is actually a very simple operation, especially with degenerate models. The first step is to separate the shells into separate objects. Then, for each object, a negated copy of the shell is inserted into the object.

For the degenerate form of the sheet object this is sufficient. For the radial edge model the edges need to be merged. The merging process involves traversing all the original edges, identifying the

corresponding edge in the negated copy and moving the loop-edge links form the edge copy to the original edge. Since the order of the edges in the copy should be the same as the order of edges in the original model then there should be no necessity to perform extensive checking to identify matching edges.

3.2.4 Splitting sheet models along edges

Splitting edges is important, for example, for manipulating sheet models created from solids. One use is to be able to split off a face from a model made from a solid. Another use is to split edges so as to be able to flatten a model by unbending it. The steps for the three cases are summarised in figure 3.7.

The first step with a degenerate model is to find the edge matching the edge to be split, if there is no direct pointer. If none is found then possibly the best course is to flag an error and exit, but it would be possible to create an isolated matching edge and use this for splitting the sheet object. Assuming that there is a matching edge there are three cases: no common vertices; one common vertex and two common vertices. With no common vertices the split is equivalent to creating a slit in the object. With one common vertex the split extends an existing hole or boundary. With two common vertices the object is split into two pieces, or an inner hole is connected to the outer boundary.

With all three cases the loop pointers are handled by simple reconnection. So, if the edges are oriented in the same direction, the right loop pointers are swapped. If the edges are oriented in opposite directions then the right loop pointer of one edge is swapped with the left loop pointer of the other. The vertices are then merged so that, all edges attached to the start vertex of one edge are moved to the corresponding vertex of the other edge. Similarly all edges attached to the end vertex of the edge are moved to the corresponding vertex of the other edge.

In the case where there is one common vertex the loop pointers are handled in the same way but the process of handling the vertices differs, the common vertex is split and the separate vertices are merged. First the loop pointers and the separate vertices are merged. The edges at the common vertex then form two separate edge sets which can be traversed using the "edge clockwise round vertex from edge" relation. One of the edge sets should be moved to a new vertex.

With two common vertices both vertices have to be split. Again, once the loop pointers have been rearranged the original single edge set at each vertex is broken into separated parts which can be traversed and moved to new vertices.

With a radial edge model the normal process is to separate the loop-edge links into pairs attached to matching edges. Following this process the vertices can be split in the same way as outlined above.

3.2.5 Joining sheet objects at edges

This operation is most often used for joining boundary edges with just two faces at each edge, although it could be used for joining general edges as well. A precondition for the operation is that the edges lie in the same curve and that the vertices match, although the edges may be oriented in opposite directions.

The simplest version of joining sheet models is for the radial-edge type where the loop-edge links of one edge are transferred to the other edge, although the links have to be ordered correctly around the edge. If the edges are oriented in the opposite direction then the orientation in the links being moved have to be changed.

For the degenerate model the vertices of edges have to be handled correctly but the edges themselves remain. The loop points of the edges are swapped so that, if the edges are oriented in the same direction, the right loop of one edge becomes the right loop of the other edge. If one or both vertices of the edge are common then the vertices have to be split, moving one connected set of edges to a new vertex. If the vertices are separate then all edges at one vertex have to be moved so that they reference the appropriate corresponding vertex.

With a partial model the same considerations apply as for the degenerate model, but the surface orientations at the edges being joined must be the same.

3.2.6 Creating sheet models from surface models

A surface model is, already, approximately equivalent to a partial model. It has a topology and so it is relatively easy to perform the conversion. The partial model is the most appropriate form, but, of course, it is simple to add a back face and create a more complete sheet model. The other forms, degenerate or radial edge models are appropriate if the surface represents, say, a thin object like a car panel or aesthetic design, designed as a surface and then converted by offsetting to produce a thin, plate-like form.

As is described in standard texts on geometry, the natural form of a free-form surface is a rectangular patch. The natural form of the model, then, is a four -sided face, each edge having one loop pointer or one loop-edge link. There is a special case where the surface is three-sided, i.e. when one of the edges has zero length.

More complications come when dealing with trimmed patches. Here, too, there is a natural affinity between partial models and geometric form. The trim curves being, in modelling terms, edges. It is not really necessary to go into great detail, since this should be clear. The problem is being able to relate the geometrical entities appropriately, the correspondence is clear.

3.2.7 Bending and unbending sheet models

For thin plate idealisations it may be desirable to simulate bending or unbending operations. The bending operation is to produce three-dimensional models from a flattened shape. The unbending operation can be useful to produce the flattened shape from the three dimensional shape. A possible way of creating a housing in this way is to design the shape as a solid; make a plate model from the solid; cut some edges; and finally 'unbend' the sheet model to produce a flattened shape which can be cut from a sheet, see figure 3.8.

Both these operations basically rotate a selected portion of the object about a straight edge. The complicated part is to delimit the object part to be rotated. This is done in the same basic way as for bending volumetric models mentioned by Braid [Brai79] and described in [Stro92], i.e. defining

the bend seam, the part to be rotated and the angle, see figure 3.9. The important thing is that the bend seam separates the object into two pieces.

It is, perhaps, easiest to separate the two cases and deal with bending and unbending separately. For bending the bend seam can either be specified as a set of existing edges or created automatically by, say, specifying a line (and a possibly also a plane if the line does not lie on the surfaces where the bend seam is to be inserted) and a starting face and letting the operation create of find a connected series of intersection edges line in the plane. Whichever method is chosen, the edges in the seam have to be marked or recorded in some way so that they can be identified. Then one face in the part to be moved is chosen and put into a list. Faces are taken from this list successively and all adjacent faces checked and, if not already processed, added to the list providing that the face is not on the opposite side of the bend seam. When the list has been processed it should contain all adjacent faces which have to be modified. It is also possible to check whether the list contains all faces in the object. If it does, then the bend seam does not close off part of the object and the process should be aborted.

With the bending operation the bend angle is specified as an input parameter. The surfaces of all the faces in the list of faces to be moved, the curves of all adjacent edges except those in the bend seam and the vertex positions of all adjacent vertices have to be modified. The modification is, of course, a rotation about the bend axis by the given angle. An extra consideration is the degree of accuracy of the bend, in that the operation is a simulation of a physical process. In reality the bend cannot be done exactly, there will always be a small rounding. It may or not may not be important to represent this. If it is important then the bend edges need to be rounded/blended correctly. Obviously care must be taken if the bend radius is large to avoid faces disappearing.

For unbending, in the simple case where the seam is not blended, the bend seam edges should be existing edges, usual only single edges which create the bend seam with a matching edge. The unbend angle is calculated from the two face surfaces on opposite sides of the bend seam edge so that the face normals after the operation are parallel. In other respects the operation is similar. It is necessary to identify the connected face set to be moved.

Although basically similar, the operation is more complicated if unbending is intended as the inverse of a bending operation which blended edges. In this case the bend seam consists of a set of blend faces which need to be collapsed and the rest of the body dragged together. To be able to do this there are certain requirements about the geometry of the bend seam faces. The faces themselves must lie in rotational surfaces which have an identifiable axis in order to be able to calculate the appropriate transformation. The faces should be four-sided, exceptionally two or three are possible, but if the operation is inverting a blended bend then four would be normal. One pair of opposite edges will disappear and the other opposite pair should be merged into a single edge, so must lie in curves which will be coincident after the transformation.

Note, again, the special nature of this operation. It is intended for a special purpose and not as a general operation. Of course, in general, a user might identify a wide variety of possible faces as an unbend seam, but the general case is not appropriate for the operation. The logic is as an inverse of a specialised operation. If the preconditions are not met then the user should be informed that the face is not appropriate and the operation stopped.

3.2.8 Modelling operations on sheet models

One use of partial models is as feature primitives which can be edited into a shape. For this purpose it is possible to use Boolean operations, so long as the Boolean boundary doesn't cross the boundary of the partial model. The process of editing the partial model into a solid works the same as for an intersection operation, the partial feature model being the equivalent of an infinitely large solid.

This is illustrated in figure 3.10.

Boolean operations for degenerate and radial edge models are more awkward. Figure 3.11 shows the results of intersecting a cylinder and three kinds of sheet model. For the radial edge and degenerate models the intersection produces a double circular edge, one on each side of the sheet (figure 3.11a,b), while for the partial model a single ring is produced (figure 3.11c). However, for the radial edge model the two matching circular edges should be matched and coalesced. In all three cases the intersection process should produce a single intersection edge on the cylinder, but for the degenerate model case there should, in principle, be a pair of matching edges. The result of adding the cylinder is shown in figure 3.12. For the radial edge model there is a single non-manifold intersection seam. For the degenerate model there is a pair of matching seams matching the double seam in the cylinder. For the partial model one half of the cylinder is removed.

From one point of view the partial model is the best adapted for Boolean operations because it behaves analogously to a volume model, unless the Boolean operation extends over the boundaries of the partial model. In the other cases some extra work needs to be done to make sure that the operation behaves in the proper manner. The important question concerns why the Boolean operations should be used for sheet modelling. Examine figure 3.12 again, and examine the meaning of the result. Is it appropriate to have a model of mixed dimensions, i.e. with special sheet parts and volumetric parts? There is, of course, no standard answer, it depends on the application of the model type. Using addition for volume models and sheet models is questionable, but it may be useful to allow subtraction or intersection. Boolean operations between specialised sheet model representations is a different matter, but the considerations are different. Addition of sheet objects is possibly a useful function. Subtraction and intersection of sheets may or may not be useful. These are difficult questions to answer authoritatively because these specialised representations are idealisations, and the meaning of the idealisation needs to be considered when deciding which operations to allow. In any case, as illustrated above, the basic Boolean operation method can be used, but the number of special cases increases.

Sweeping, as an operation, has been dealt with for sweep extending sheet models. Sweeping of sheet models to produce volumes is documented elsewhere (e.g. [Brai79], [Stro92]).

More important, here, are the special purpose modelling operations, local operations, which produce special effects. A complete set of these cannot be produced because they are specialised tools for performing specific directed changes appropriate for an application or application area. Two examples, which will be dealt with here, are:

Reflection
Chamfer/Blend.

Reflection
The reflection operation for a sheet model can be specified by giving a reflection plane or by specifying a straight, external edge. If the operation is specified by giving an edge then the reflection plane is calculated from the surface normal of the face at the edge. The same sort of process is involved as for volumetric reflection, the object is checked to see that it doesn't cross the reflection plane, a copy of the complete model is made and transformed with a reflection matrix calculation from the reflection plane and then the original model and reflected copy are joined (if desired) at edges lying in the reflection plane.

Chamfer/Blend
This operation is only appropriate for internal edges of the sheet model, not for exterior edges.

For partial models this operation works exactly the same way as for a volumetric model.

For degenerate and radial-edge models the problem is that there are two sides to the object and the same change has to be made on both sides, otherwise a volumetric component will be introduced into the model. This process is illustrated in figure 3.13. With a radial-edge model the edge exists on both sides of the model, but with a degenerate model it is necessary to search for the matching edge, unless these are associated with pointers. The operation is otherwise similar to the volumetric case, the degenerate type of sheet model chamfering each edge being chamfered or blended separately. For the radial edge model the chamfer takes place between appropriate pairs of loop-edge links, the new edges created having to be merged back together after the operation. In this respect it is easier to consider converting the edge to be chamfered to a degenerate form, then performing the chamfer, and finally merging back the edges.

4. Non-manifold modelling in solids

Non-manifold modelling in solids may be useful for particular applications, but the complications introduced by having general structures cause many complications for general algorithms. It is necessary, yet again, to be clear about the philosophy of use, the reason for having the special representations so as to have a consistent understanding of how to handle special cases.

Many of the same considerations already mentioned for sheet models are also relevant for solids as well. There is the same potential ambiguity for radial edges as already mentioned, unless they are arranged in pairs. There is the same lack of associativity between degenerate edges, unless they are specifically associated.

The ambiguity of radial edges can be illustrated by considering the case of two cubes which have a common non-manifold edge. The question of ambiguity arises when it is necessary to know whether the cubes just touch each other, or whether there is an infinitely thin neck of material link the two cubes. The question is important when operating on the edge. If, say, the edge is chamfered then there are two different results, depending on the interpretation. Either the chamfer operation separates the modified cubes or fattens the neck (figure 4.1).

The lack of associativity between edges in the degenerate model is important, for example, when splitting an object through the degenerate edges. The resulting section face will have two coincident vertices which need to be merged to create the correct result (figure 4.2). If there is no indication that an edge is part of a 'degenerate' edge set, and no link between the coincident edges, then it is difficult to know when a vertex has to be merged without extensive checking.

Two applications to illustrate the use of non-manifold modelling in solids might be finite-element analysis and process planning.

4.1 Finite element analysis

Here it is necessary to create a cellular structure, dividing the interior of the object into volumetric components while maintaining the cells as a whole.

The cells are created by inserting internal faces into the object. The internal faces are linked to the exterior of the object at normal edges, which then become non-manifold edges. Each volumetric element created this way can then be handled separately while all the cells are still integrated as a whole to represent the original model.

One important process is to be able to identify the connected faces which make up a cell. Here the use of the “link clockwise/counterclockwise round edge from link” function is important so that the correct face sets can be found. The definition of clockwise and anticlockwise is done with respect to the direction of the edge. To find the external faces of the object, the clockwise function is used, starting at an external face and proceeding round the object. To find all faces in a cell, the counter-clockwise relation is used.

One facility, present in the ACIS modeller, is the use of double-sided faces. These add an extra dimension to the preceding discussion. The double-sided faces are logical as a means of saving

space, but they add an extra set of special cases which must be handled. They function, in effect, in the same way as a pair of back-to-back faces, but must be treated differently. The single link for a double sided face works, in effect, as a pair of links, when checking material-within-material or pairing links. The forms described before are closer to treating non-manifold models as special cases, the double-sided face is a step away from that, and should hence be treated with more caution.

4.2 Process planning

The same sort of cellular structure, mentioned above, is also useful in process planning. In process planning there are usually several options which can be chosen to manufacture a part. The different options give rise to different sets of features and different intermediate models. The use of non-manifold models in this case is to be able to record the different stages of manufacture as a compound, cellular model. The same considerations outlined above apply here also.

References

Brai79 Braid, I.C. "Notes on a geometric modeller", CAD Group document 101, Cambridge University Computer Laboratory, 1979.

GPM81 "Volume Module system specifications", GPM Report 18a,b,c, Dept. of Manufacturing Systems KTH Sweden and VTT Finland, 1981.

Kjel82 Kjellberg, T.A. "Integrerad datorstöd for mänsklig problemlösning och mänsklig kommunikation inom verkstadsteknisk produktion begränsat till produktutveckling, produktionsberedning, konstruktion och tillverkningsberedning. En systemansats baserad på produktmodeller", Ph.D. dissertation, Dept. of Manufacturing Systems, KTH, 1982.

Luo92 Luo, Y, "Solid modelling for regular objects: Renewed Theory, Data Structure and Euler Operators", Ph.D. dissertation, MTA/Sztaki, 1992.

LuLu91 Luo, Y., Lukacs, G. "A Boundary Representation for Form Features and Non-Manifold Solid Objects", Proc. First ACM/SIGGRAPH Symposium on Solid Modeling and Applications, pp 45-60, 1991.

Stro90 Stroud, I.A. "Modelling with degenerate objects", Computer Aided Design, vol. 22 number 6, 1990.

Stro92 Stroud, I.A. "Definition of solid modelling operations using a uniform set of elementary procedures", Ph.D. dissertation, Hungarian Academy of Sciences, 1992

Stro94 Stroud, I.A. "Boundary modelling with special representations", Computer-AIDED Design journal vol 26 number 7, July 1994

Weil86 Weiler, K. "Topological structures for geometric modeling", Ph.D. dissertation, Rensselaer Polytechnic Institute, Troy, New York, August 1986

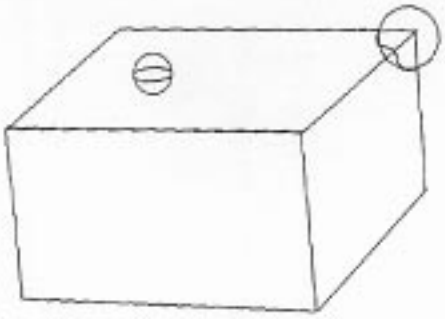


Figure 1.1

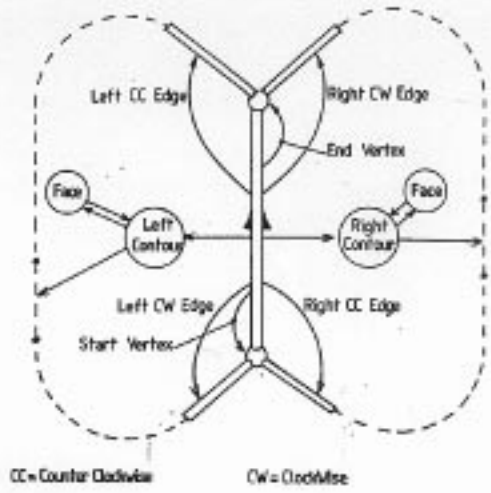


Figure 1.2

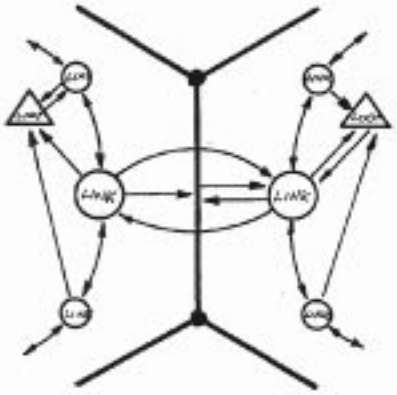


Figure 1.3

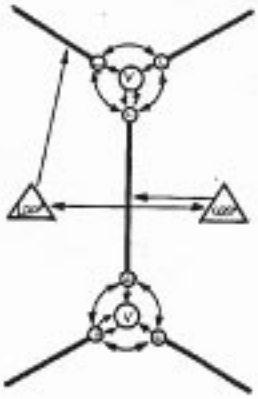


Figure 1.4

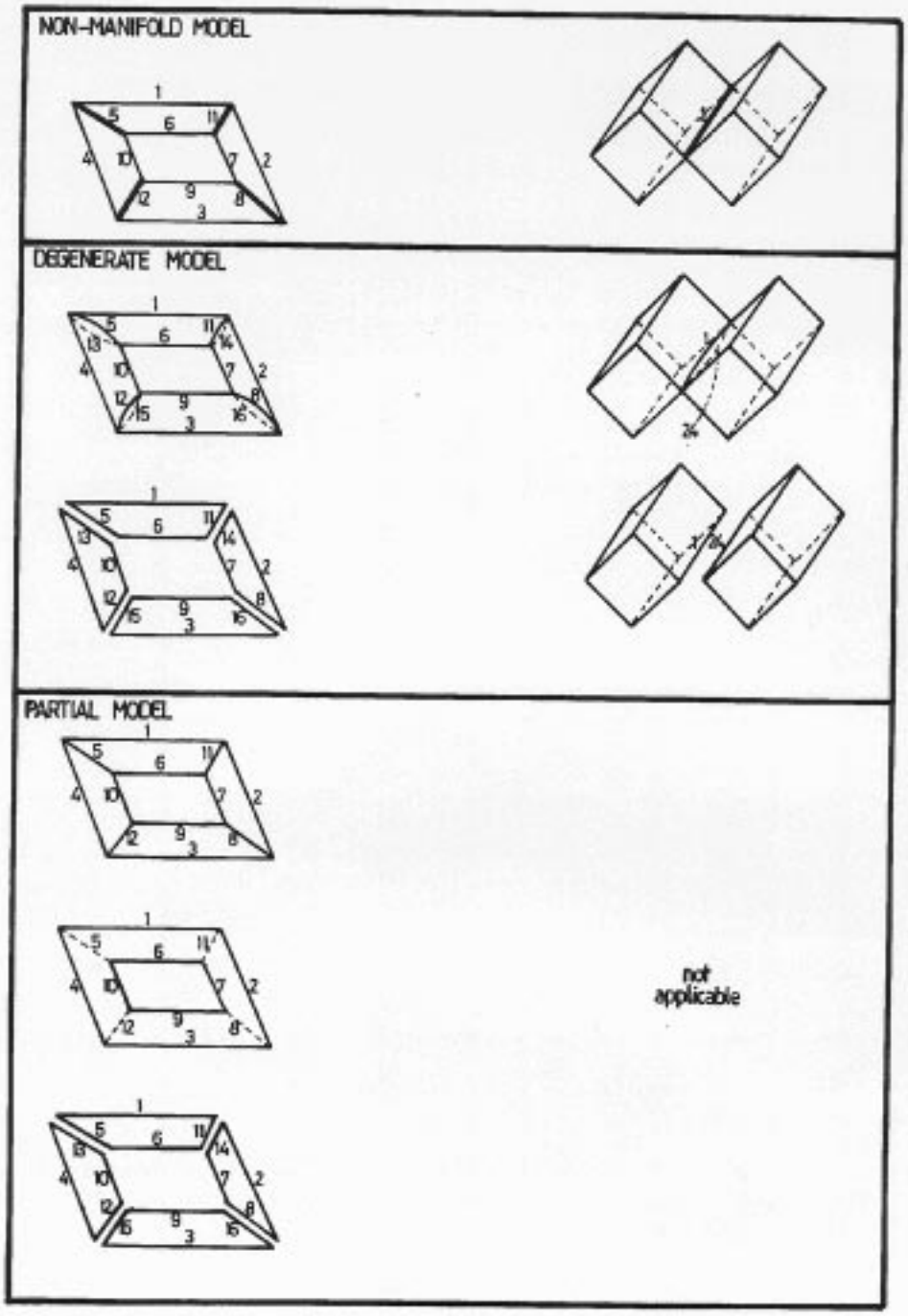

NON-MANIFOLD MODEL
DEGENERATE MODEL
PARTIAL MODEL
not applicable

Figure 1.5

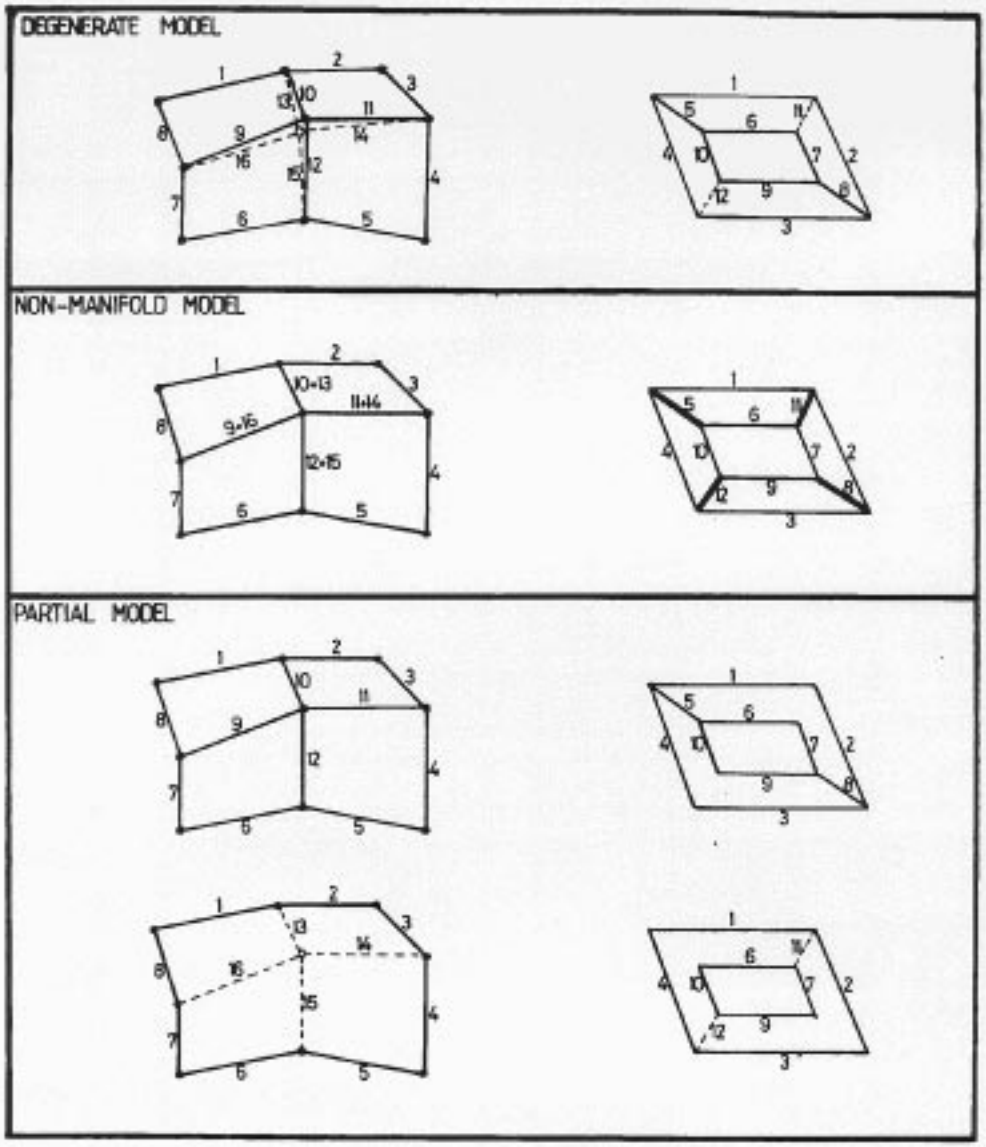


Figure 1.6

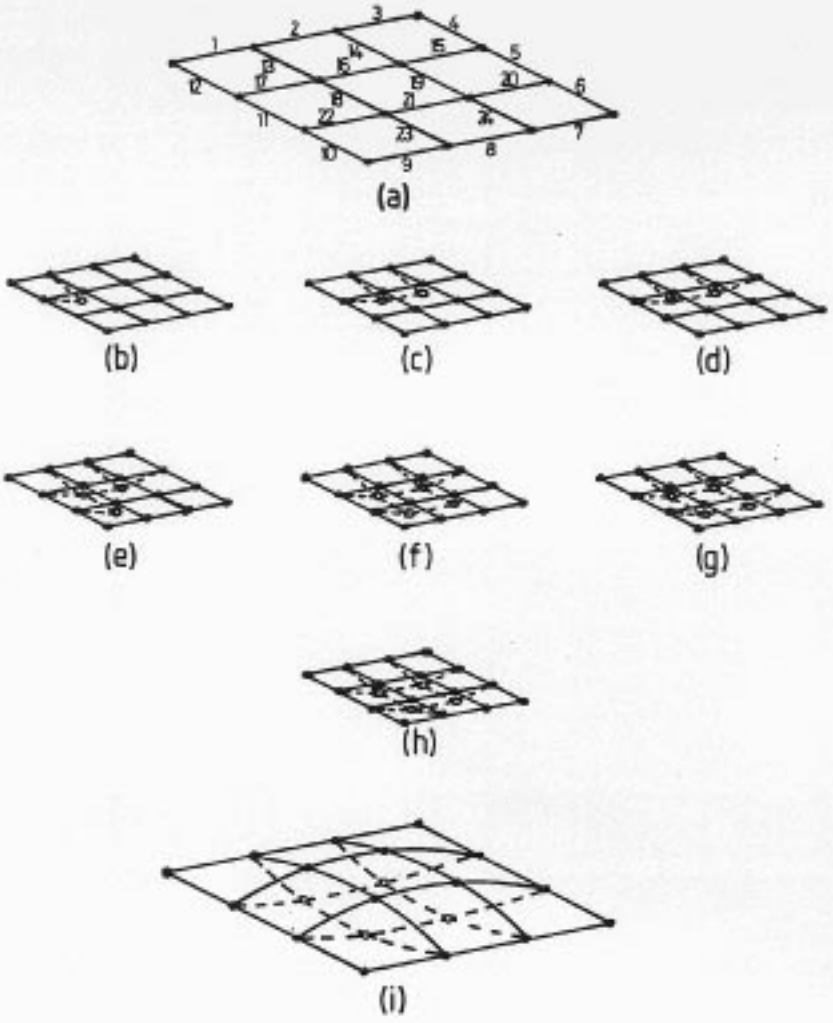


Figure 1.7

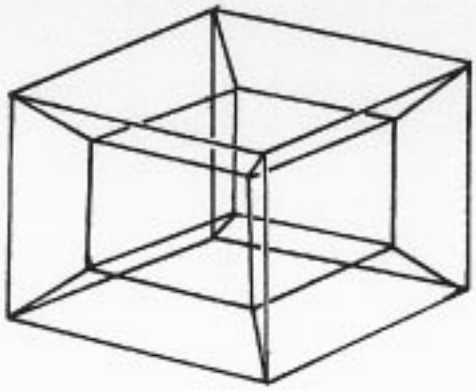


Figure 2.1

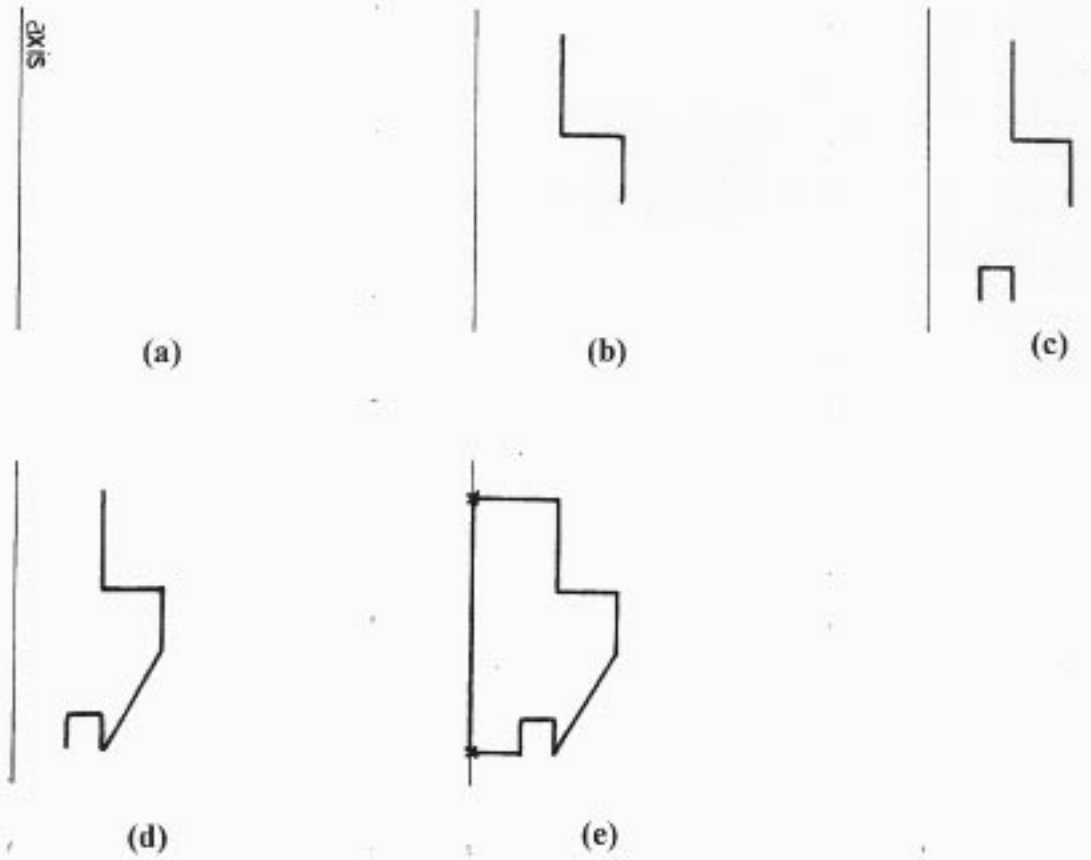


Figure 2.2

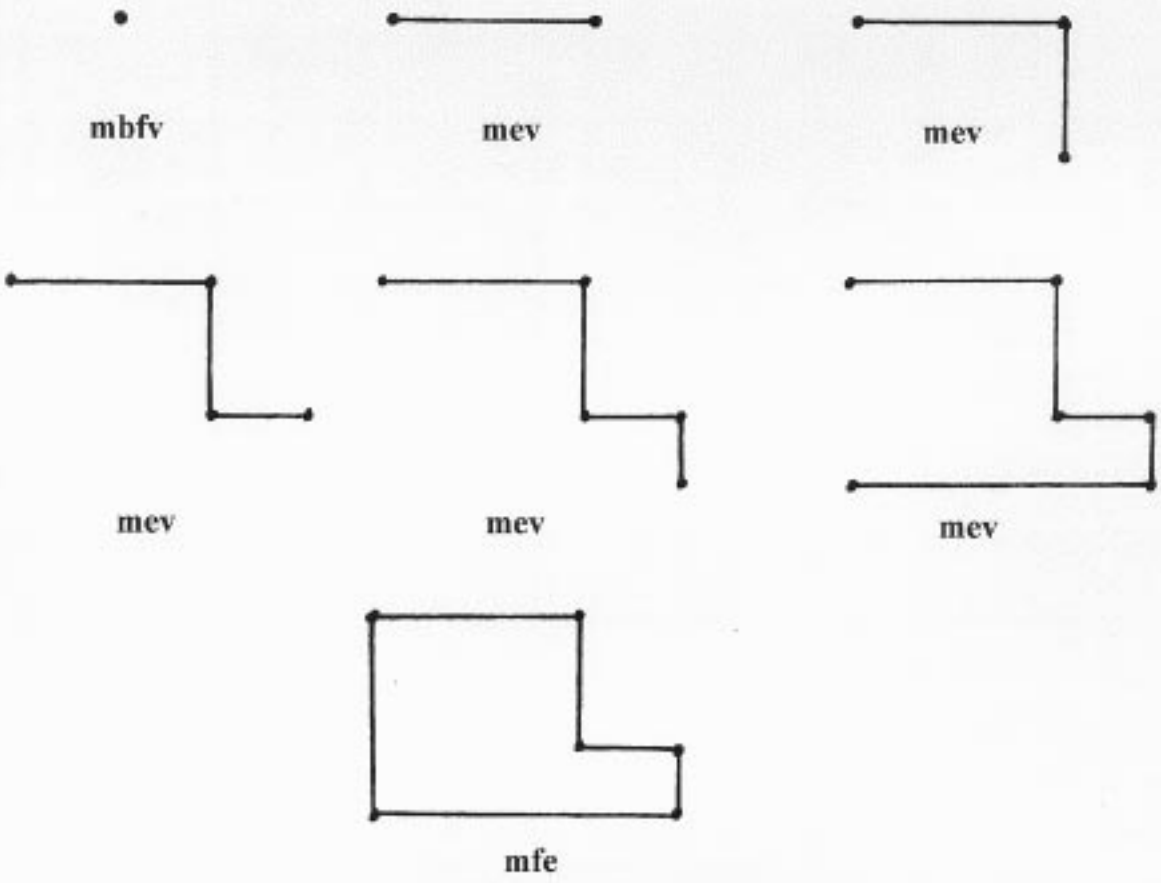


Figure 2.3

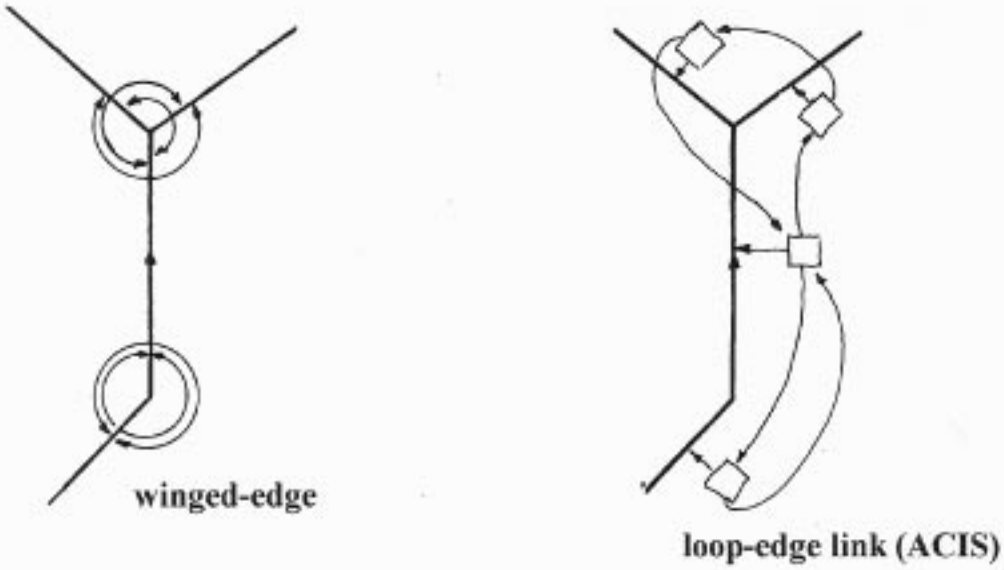


Figure 2.4

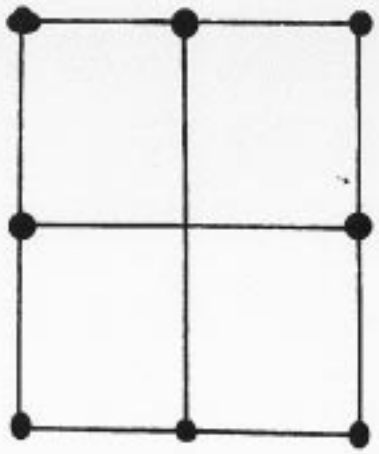


Figure 2.5

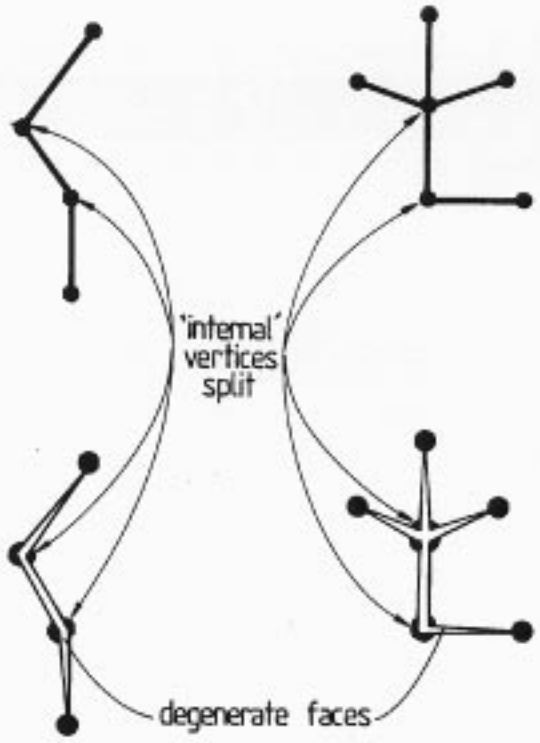


Figure 2.6

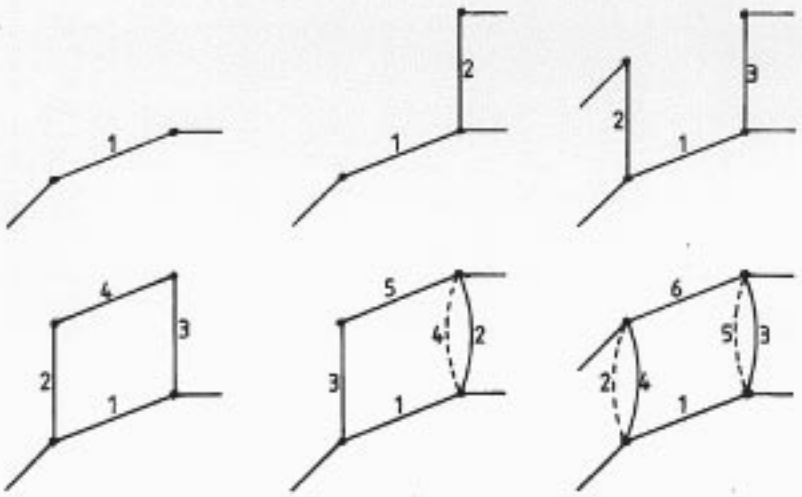


Figure 2.7

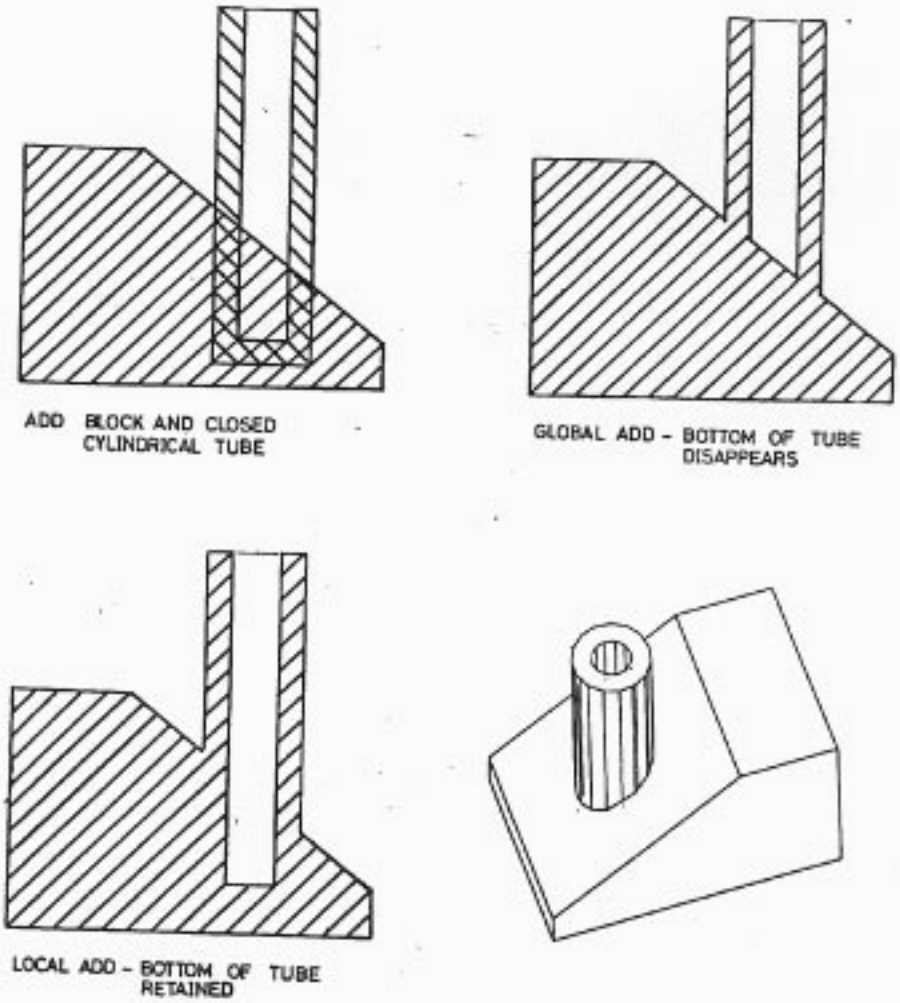


Figure 2.8

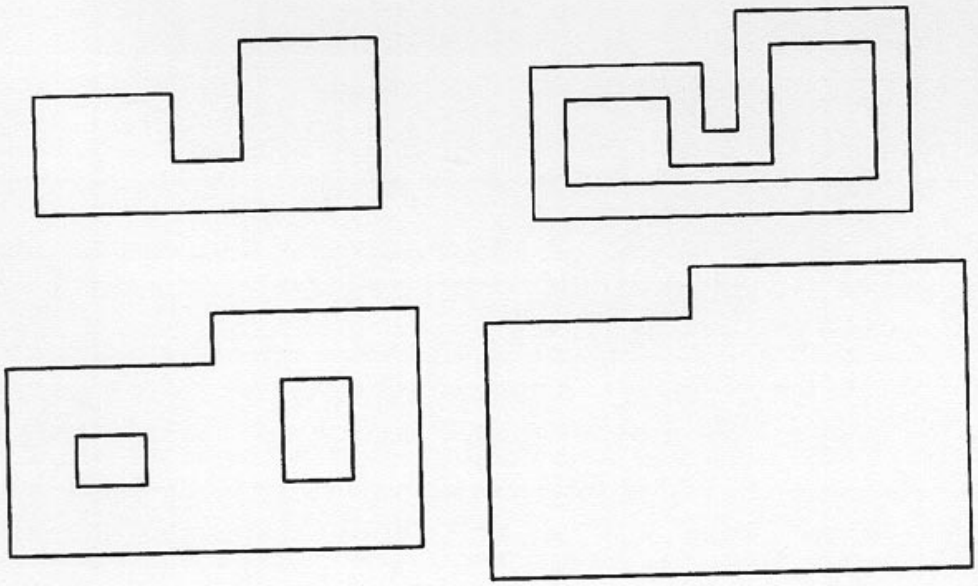


Figure 3.1

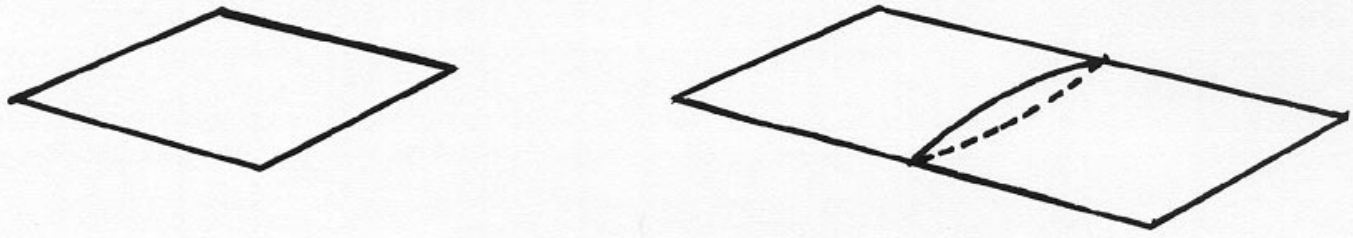


Figure 3.2

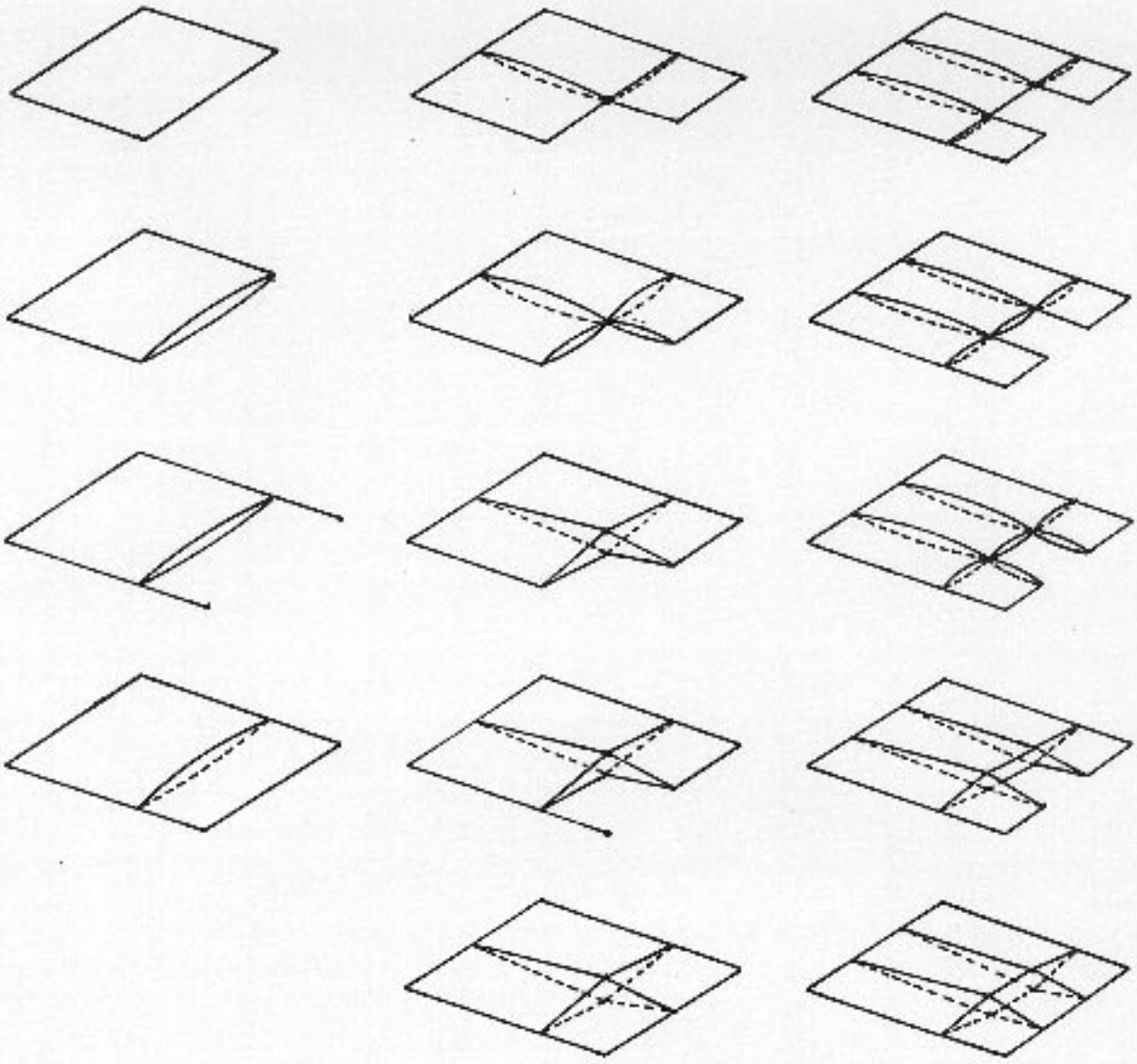
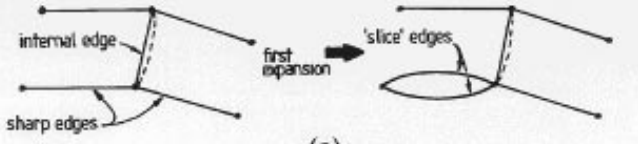
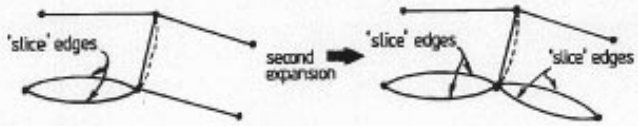


Figure 3.3



(a)



(b)

Figure 3.4

Figure 3.5

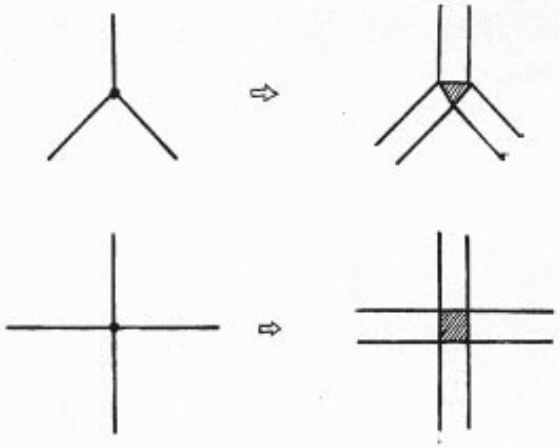


Figure 3.6

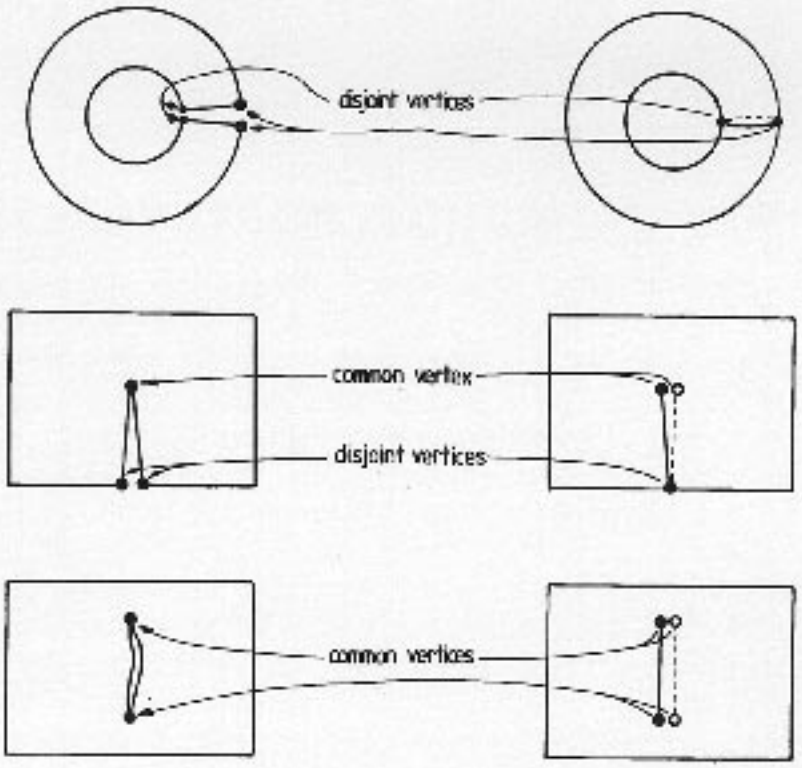


Figure 3.7

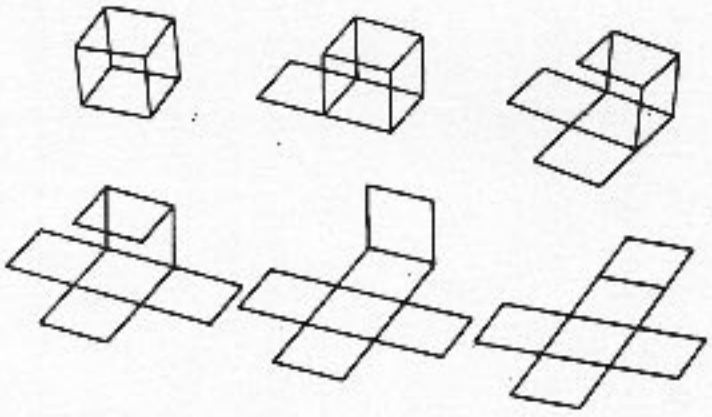
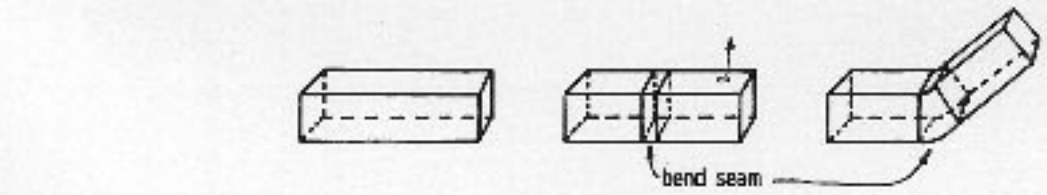


Figure 3.8



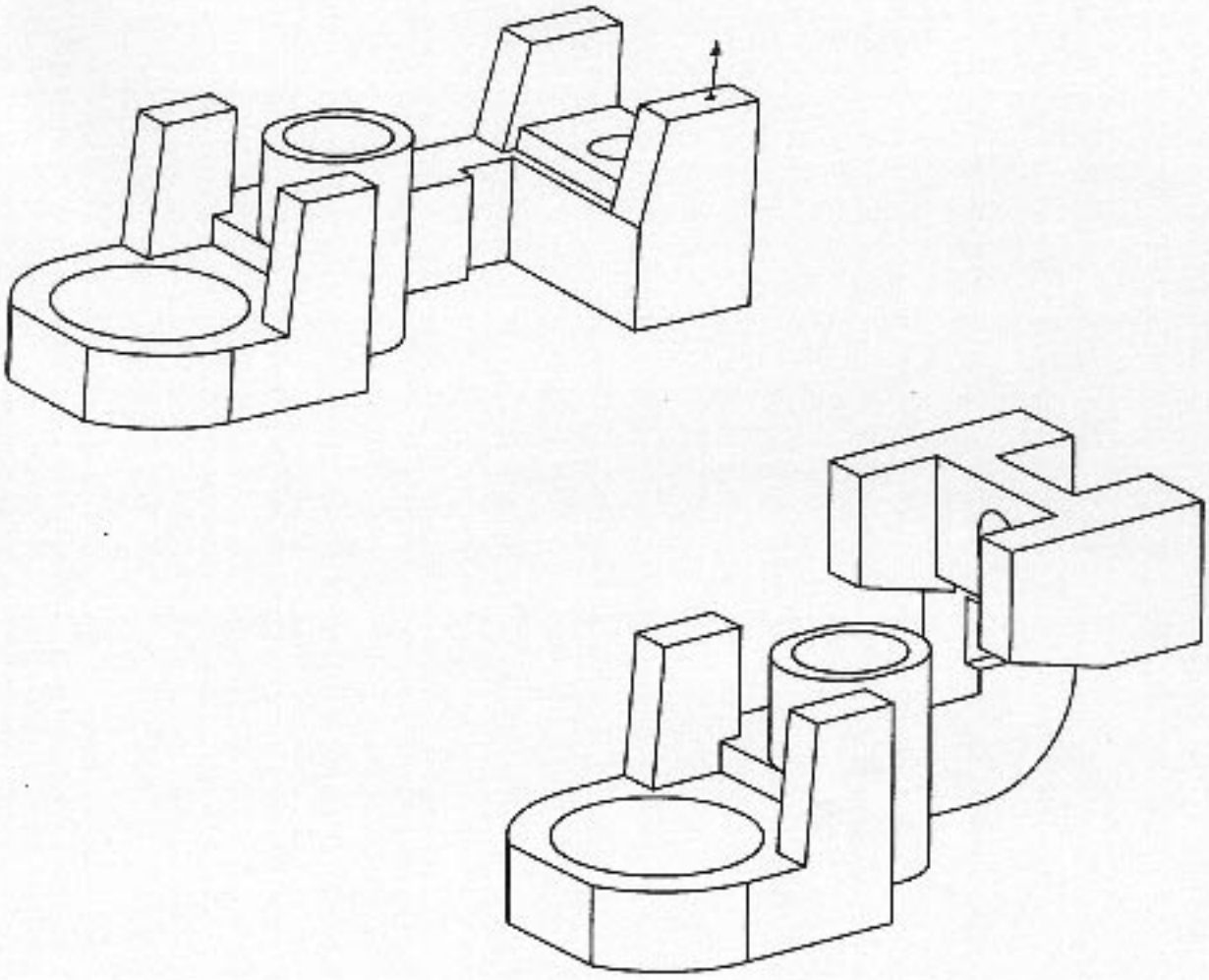


Figure 3.9

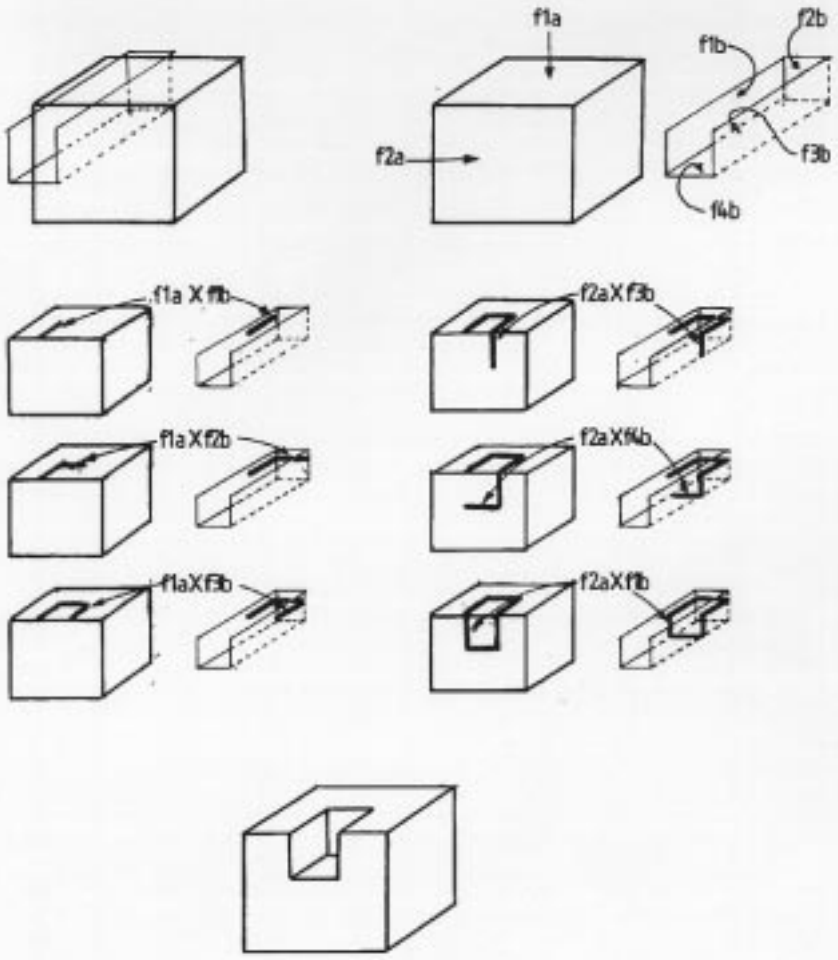

f1a
f2b
f1b
f2a
f3b
f4b
f1a X f1b
f2aX f3b
f1a X f2b
f2aX f4b
f1aXf3b
f2aX f1b

Figure 3.10

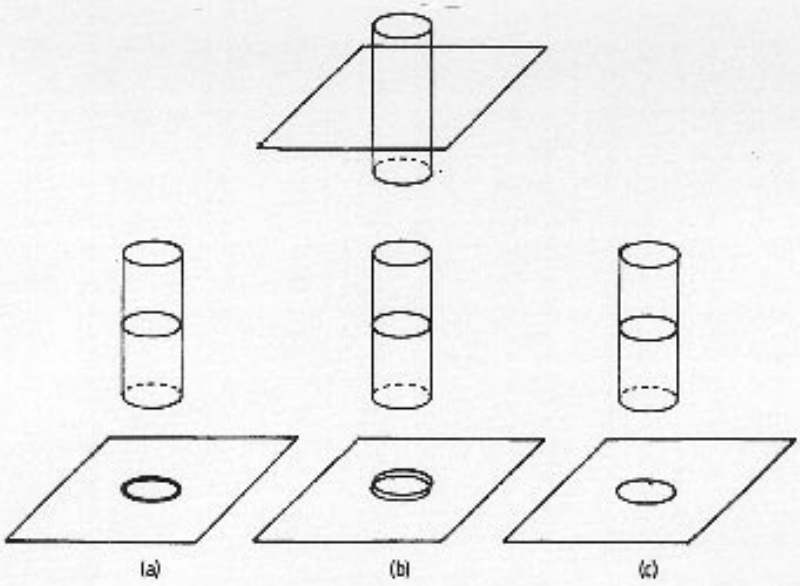


Figure 3.11

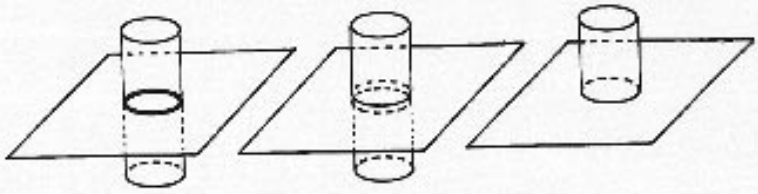


Figure 3.12

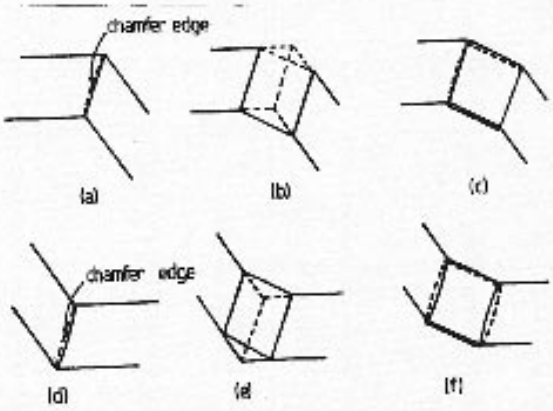


Figure 3.13

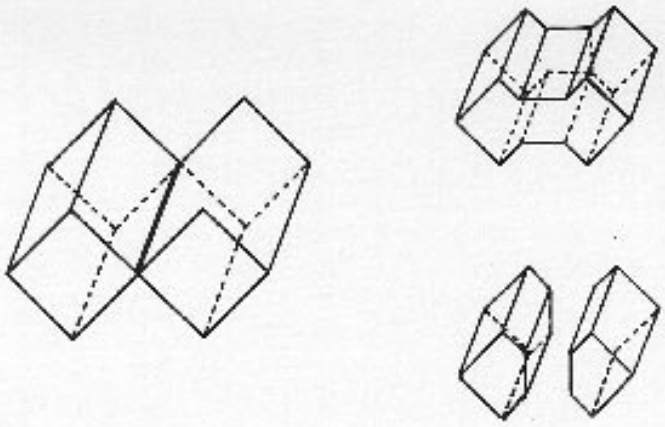


Figure 4.1

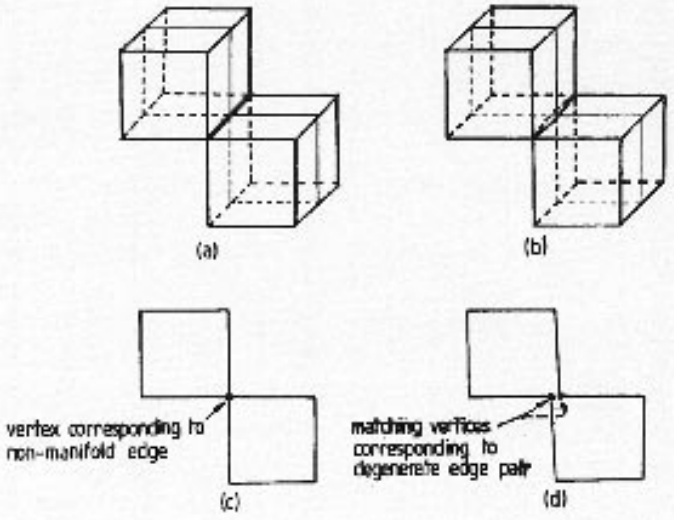


Figure 4.2