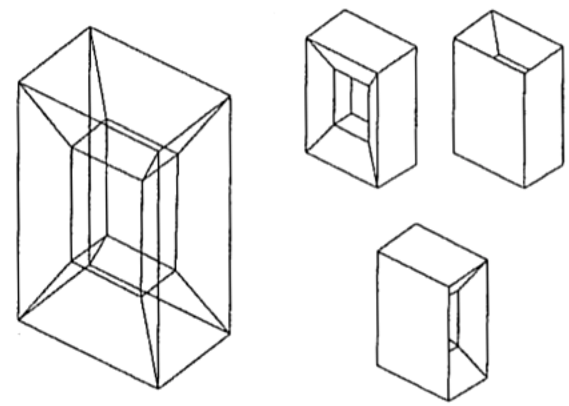
Geometric Modeling Systems

- Wireframe Modeling Systems
- Surface Modeling Systems
- Solid Modeling Systems

Wireframe Modeling Systems

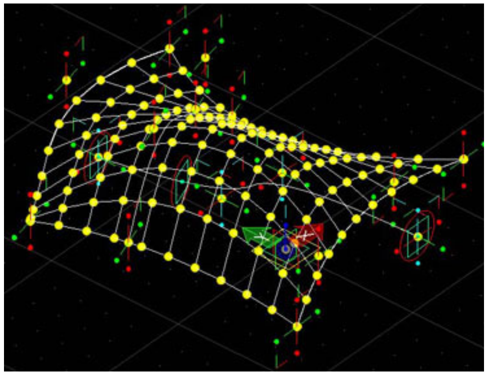
Shapes are represented by defining points in space and connecting them by lines



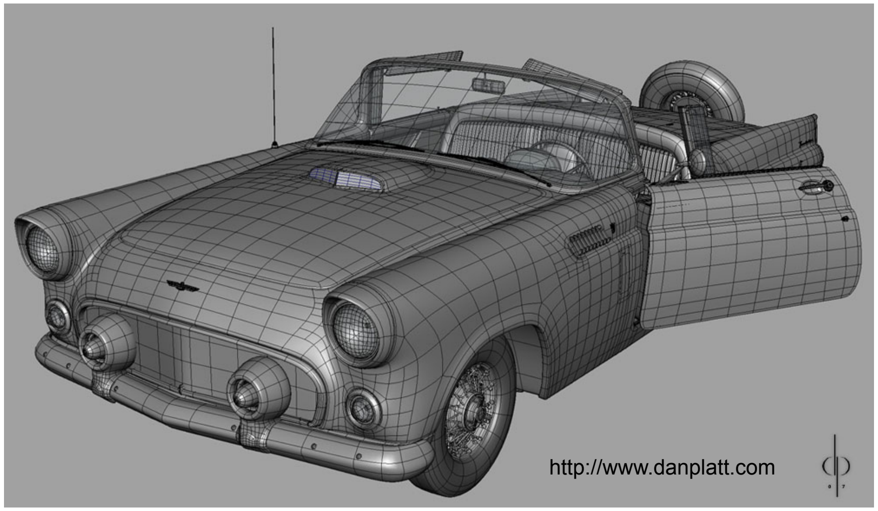
- Simple user input
- Easy software development
- Low computational power demanded
- Ambiguous represenation
- No information about solid object and surfaces
- No engineering tools applicable
- Limited usability

Surface Modeling Systems

1. Input points interpolation
2. Input curves nets interpolation
3. Translation and revolution of specified curves



- Used to create models with complex surfaces
- Can be used for NC tool path machining
- Surface connectivity information available

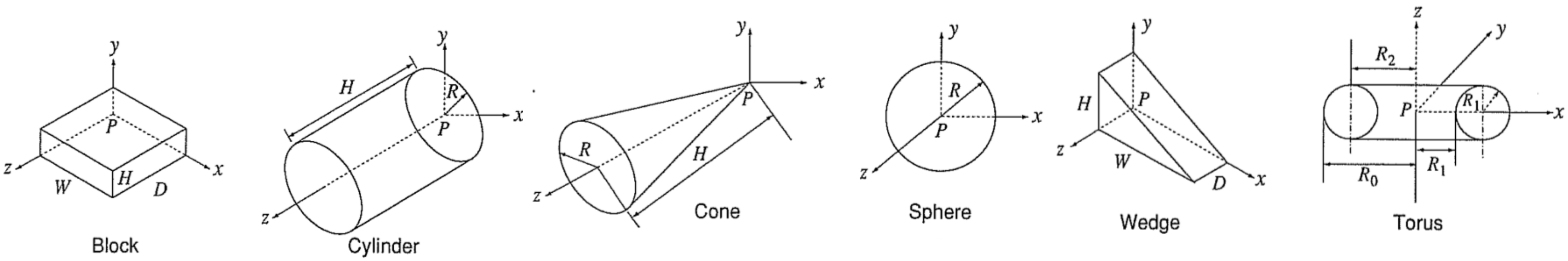


Solid Modeling Systems

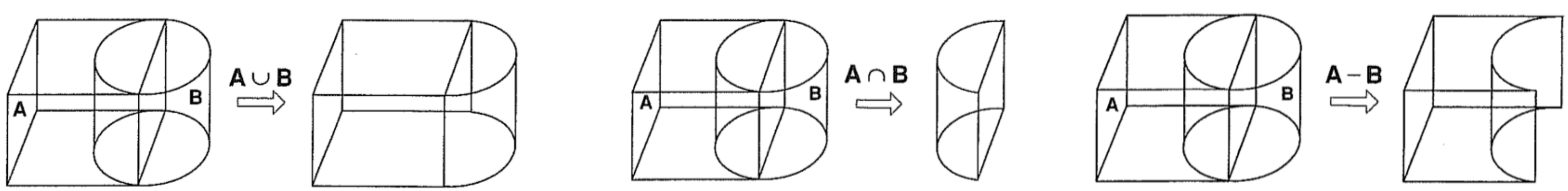
- Closed volumes are modeled
- Comprehensive solid information can be evaluated or embedded
- Surface information also included
- Best solution for engineering analysis tools

Solid Modeling: Modeling Functions

- Primitive creation functions. Predefined simple shapes



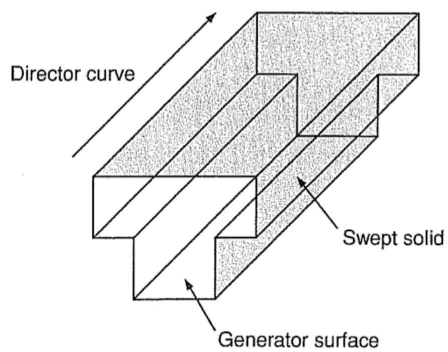
- Boolean operators

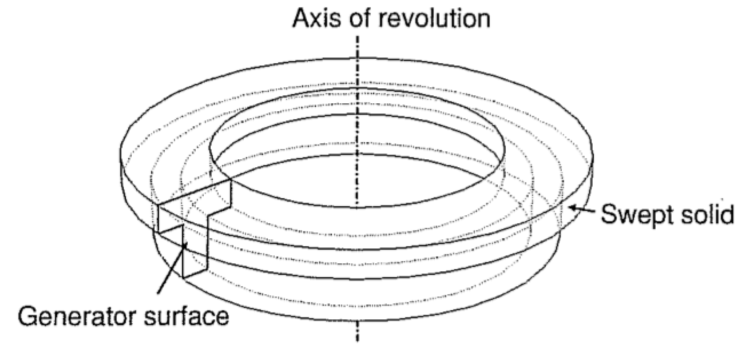


UNION INTERSECTION DIFFERENCE

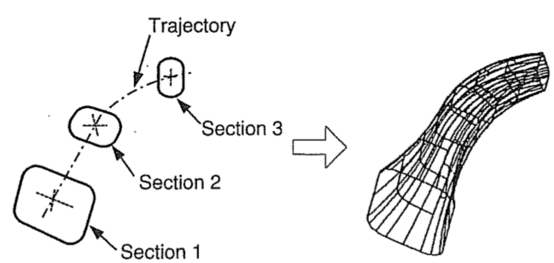
Solid Modeling: Modeling Functions

- Sweeping: translational or rotational

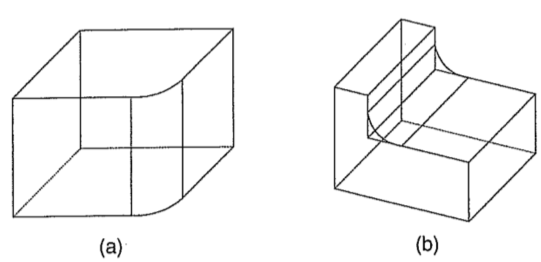




- Skinning: multiple cross section perimeter interpolation

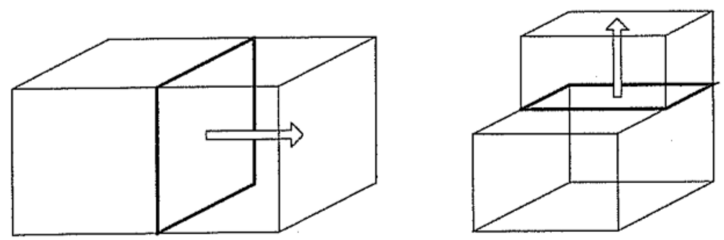


- Rounding

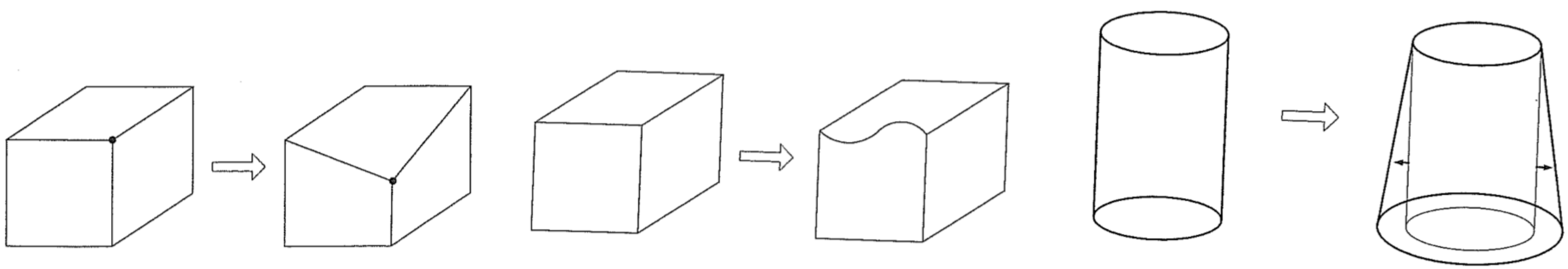


Solid Modeling: Modeling Functions

- Lifting: use a portion of a surface for lengthening part of the solid.



- Boundary modeling

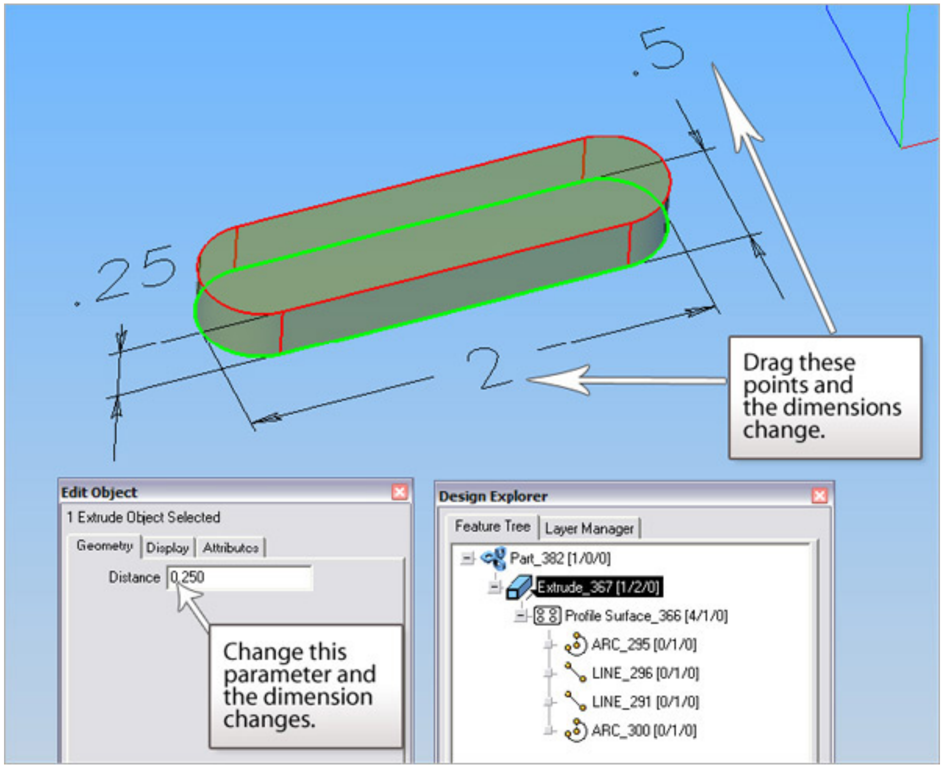


VERTEX REPLACEMENT EDGE REPLACEMENT SURFACE REPLACEMENT

Solid Modeling: Modeling Functions

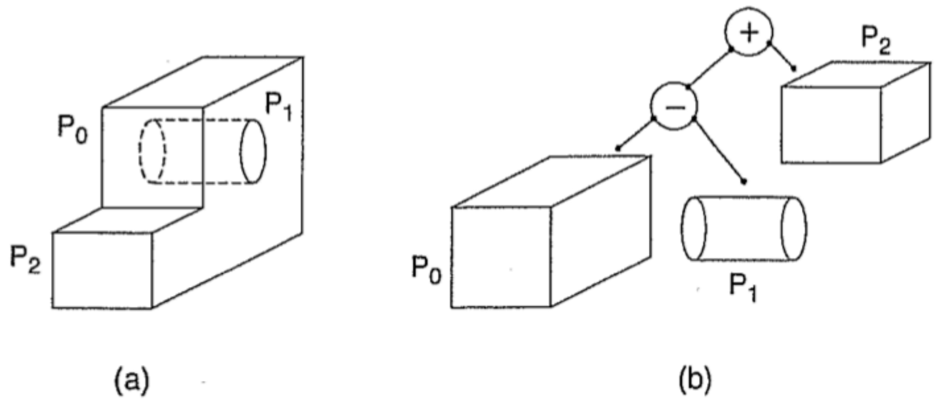
- Feature-based modeling: similar to primitive, but more complex and specialized for manufacturing, like “Hole’, or “Slot”,or “Pocket”…

- Parametric modeling: use relationships and and constraints



Data Structure

- **Constructive Solid Geometry (CSG) representation**
  - Application of boolean operations on primitives and stored in a tree type of data structure



  - Data structure is simple and compact
  - The solid stored is alwais valid
  - Easy parametric implementation
  - Only boolean operators allowed: shape definition severely limited
  - It requires lots of computation to derive information of boundary surface, edges, connectivity

Data Structure

- **Boundary Representation (B-Rep)**
  - Boundary information collected in a tree table:

Three tables for storing B-Rep

Face Table

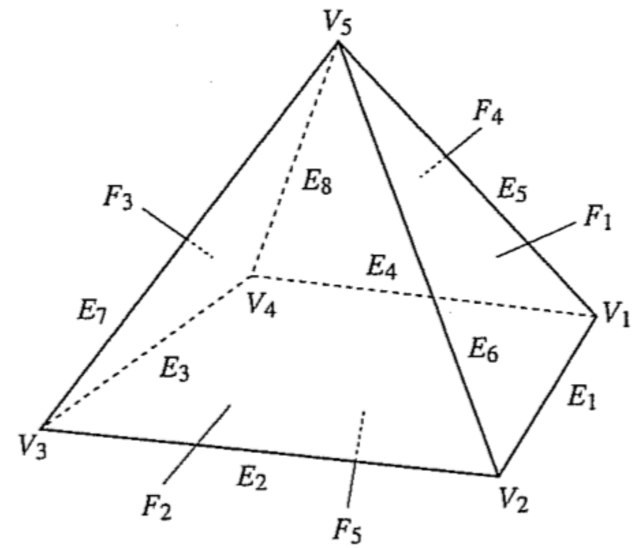
Face	*Edges*
F_1	E_1, E_5, E_6
F_2	E_2, E_6, E_7
F_3	E_3, E_7, E_8
F_4	E_4, E_8, E_5
F_5	E_1, E_2, E_3, E_4

Edge Table

Edge	*Vertices*
E_1	V_1, V_2
E_2	V_2, V_3
E_3	V_3, V_4
E_4	V_4, V_1
E_5	V_1, V_5
E_6	V_2, V_5
E_7	V_3, V_5
E_8	V_4, V_5

Vertex Table

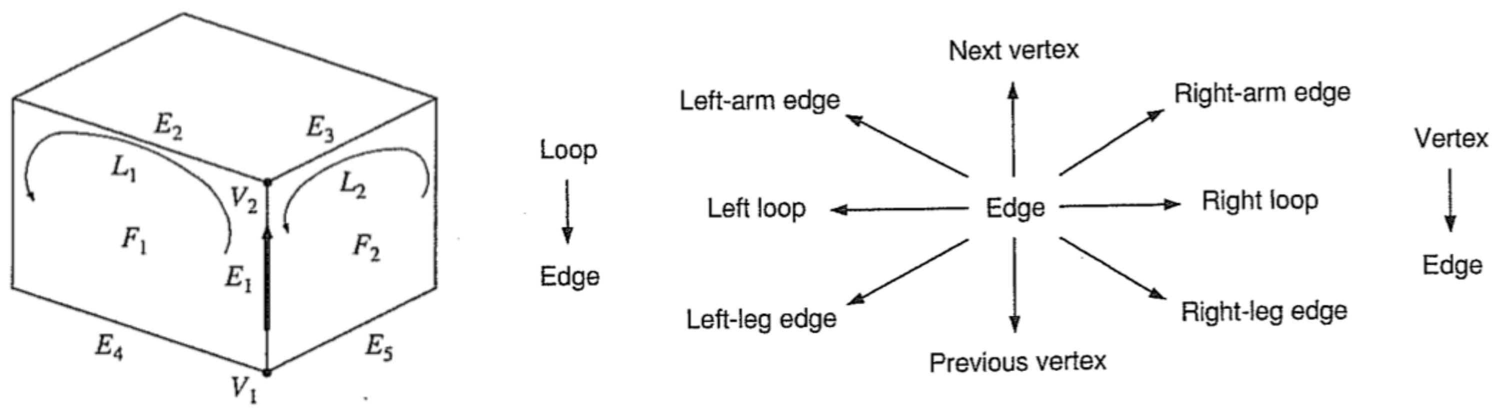
Vertex	*Coordinates*
V_1	x_1, y_1, z_1
V_2	x_2, y_2, z_2
V_3	x_3, y_3, z_3
V_4	x_4, y_4, z_4
V_5	x_5, y_5, z_5
V_6	x_6, y_6, z_6



  - Works only with planar polyhedra
  - It needs a workaround for faces with inner and outer boundaries
  - Deriving connectivity information from the tables can be cumbersome
  - Variation from B-Rep uses double linked lists: **half-edge data structure**

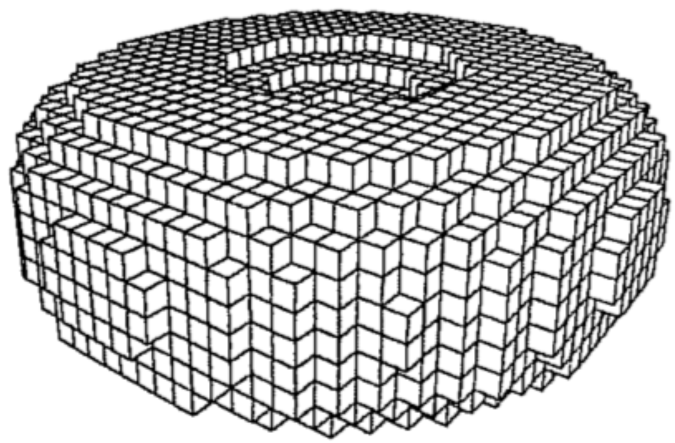
Data Structure

- **Winged-edge data structure**
  - Similar concept to half edge data structure (B-rep) but instead of being centered on faces definition it is based on edges definition:



Data Structure

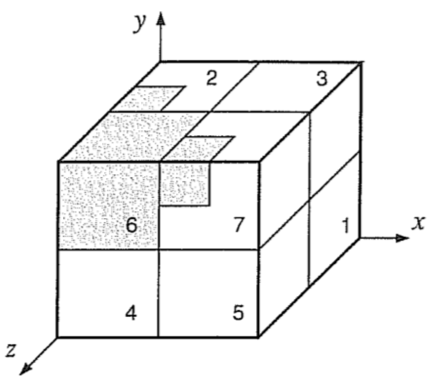
- **Decomposition Model Structure**
  - Voxel Representation: it is a 3D extension of a raster representation: Voxel stands for Volume-Pixel

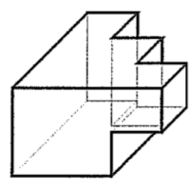


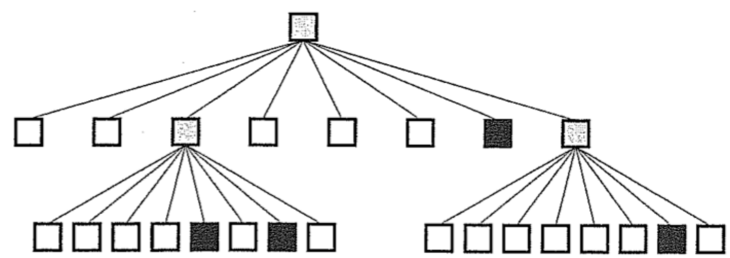
  - Easy to represent arbitrary shapes
  - Modification of this type of data structure is very difficult
  - Easy to evaluate solid properties
  - Memory usage dramatically incremented with size (or resolution)
  - Voxel representation is inherently an approximation of the original solid

Data Structure

- **Decomposition Model Structure**
  - Octree: similar to Voxel decomposition but volume in which the solid is enclosed is iteratively subdivided based on if the current octant contains some of the defined geometry (grid-based mesh generation approach)

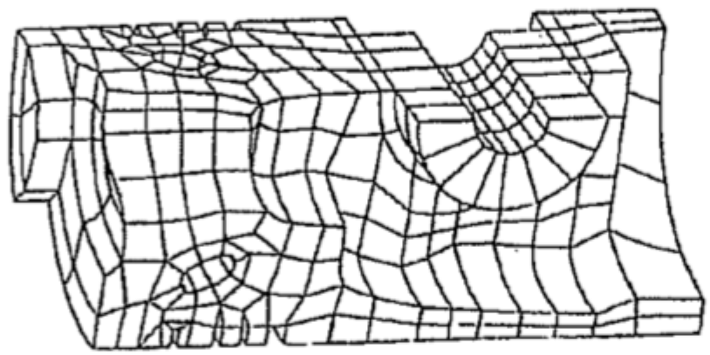






Data Structure

- **Cell Representation: similar to voxel but each cell can have an arbitrary shape.**



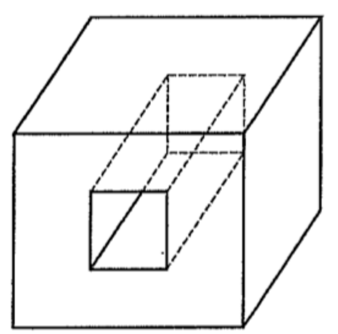
B-Rep Data Structure Modification: Euler Operators

- B-Rep data structure modification is not as straight forward as for CSG.
- Euler-Poincare formula for topology entries:

$$v-e+f-h=2(s-p)$$

v= number of vertices
e= number of edges
f= number of faces
h= number of hole loops
s= number of shells
p= number of passages

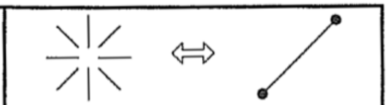
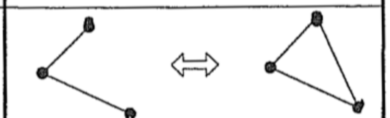
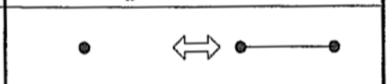
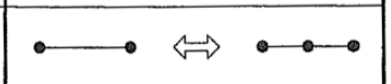
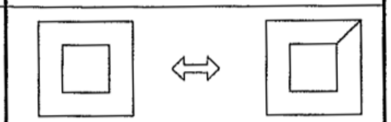
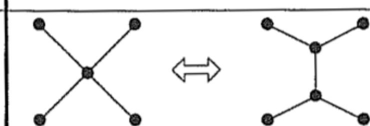
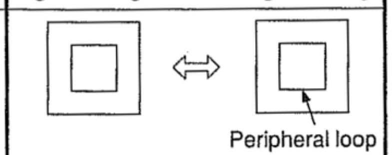
Example:



$$16-24+10-2=2(1-1)$$

B-Rep Data Structure Modification: Euler Operators

Since the topology entries are not linear independ, Euler operators are introduced to handle linked modifications among topologies

MEVVLS (KEVVLS)	make (kill) edge, two vertices, loop, shell	
MEL (KEL)	make (kill) edge, loop	
MEV (KEV)	make (kill) edge, vertex	
MVE (KVE)	make (kill) vertex, edge	
MEKH (KEMH)	make (kill) edge, kill (make) hole	
MZEV (KZEV)	make (kill) zero length edge, vertex	
MPKH (KPMH)	make(kill) peripheral loop, kill (make) hole loop	

B-Rep Data Structure Modification: Boolean Operators

- Also in B-Rep data structure Boolean operators are not as straight forward as for CSG.
- Method for 2D (3D similar but more complex):
  - define face A and face B on which to apply the Boolean operator
  - Find face C
  - Find all edges (including edges of C)
  - Classify all edges according to each face:
    - inside, on outside A; inside, on outside B
  - Collect edges based on Boolean:
    - Union: reject “in A” & “in B”
    - Intersection: reject “out of A” & out of B”
    - Subtraction: reject “on A” & “in B” and “out of A” & “in B”