

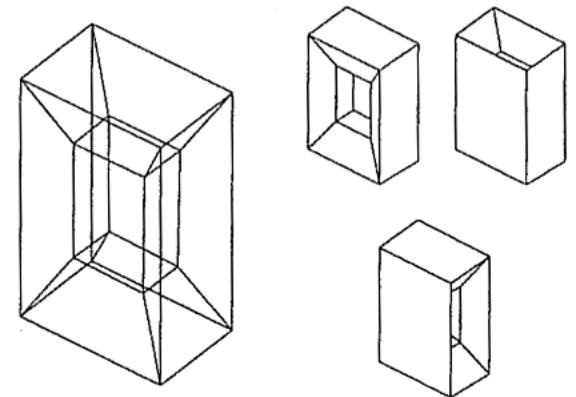
Geometric Modeling Systems

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

Wireframe Modeling Systems

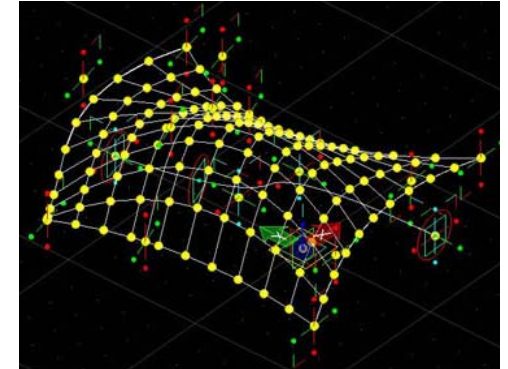
Shapes are represented by defining points in space and connecting them by lines

- Simple user input
- Easy software development
- Low computational power demanded
- Ambiguous representation
- No information about solid object
- 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

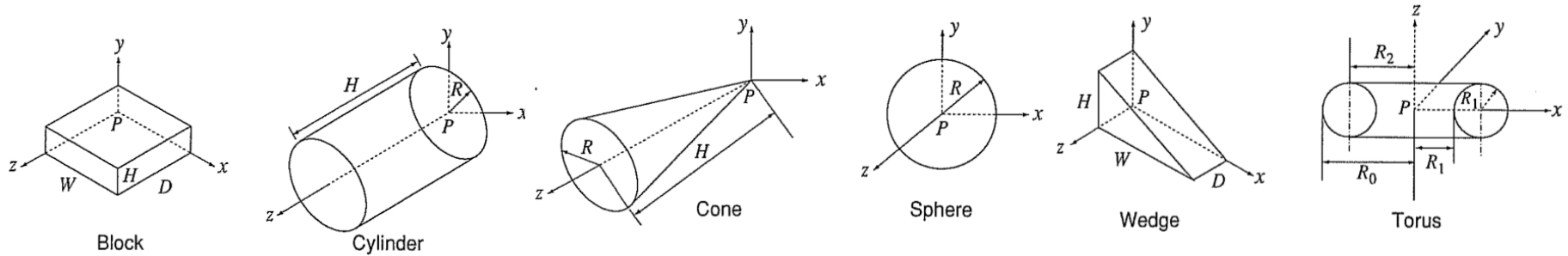


Solid Modeling Systems

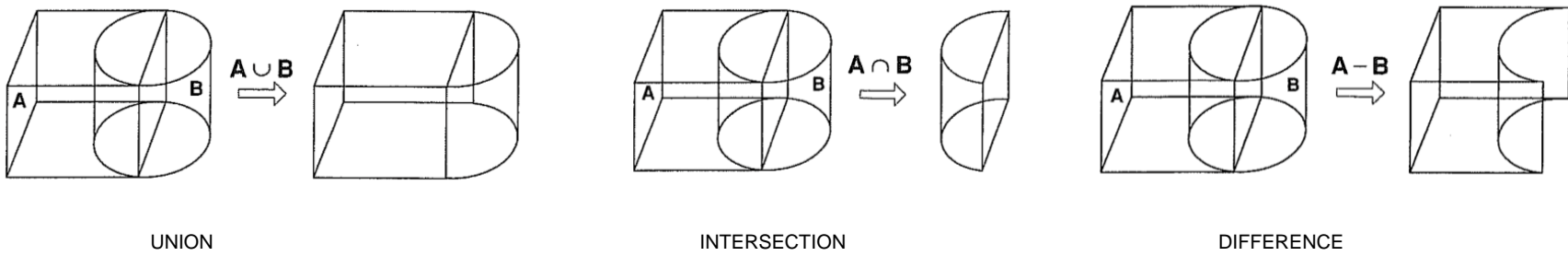
- Closed volumes are modeled
- Comprehensive solid information can be evaluated or embedded
- Surface information also included
- Best solution for engineering analysis tools

Solid Modeling: Modeling Functions

- Primitive creation functions. Predefined simple shapes

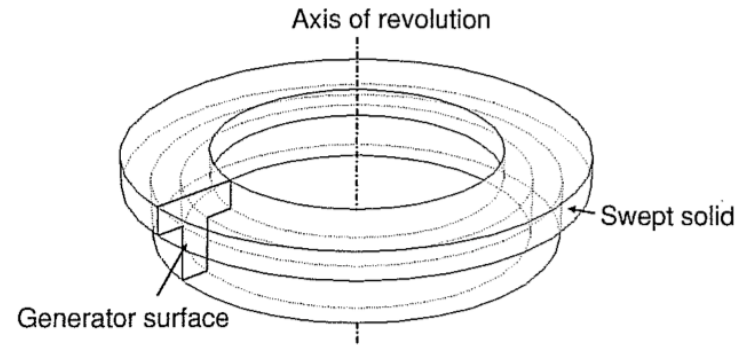
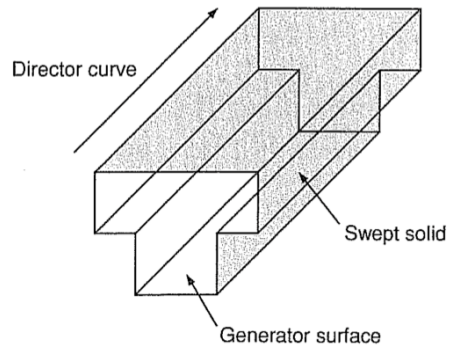


- Boolean operators

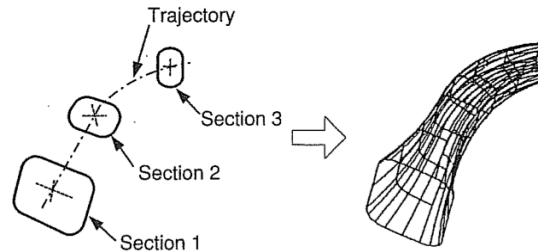


Solid Modeling: Modeling Functions

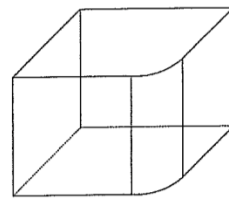
- Sweeping: translational or rotational



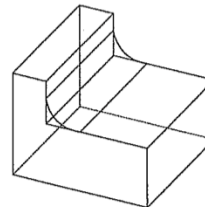
- Skinning: multiple cross section perimeter interpolation



- Rounding



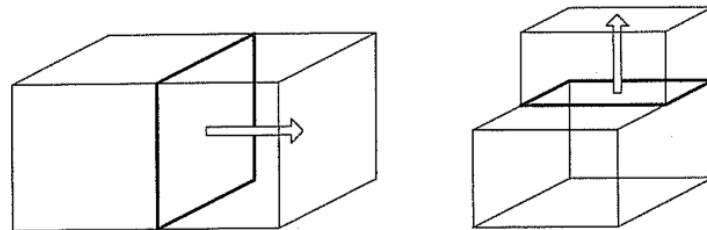
(a)



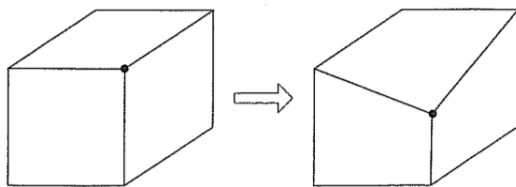
(b)

Solid Modeling: Modeling Functions

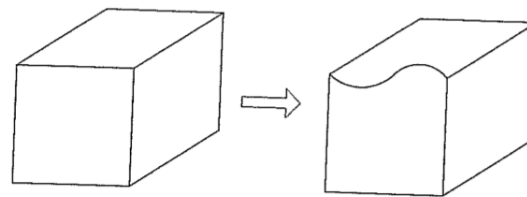
- Lifting: use a portion of a surface for lengthening part of the solid.



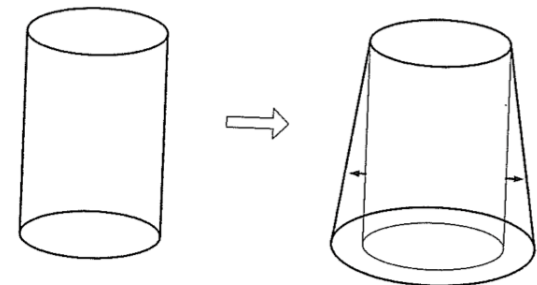
- Boundary modeling



VERTEX REPLACEMENT



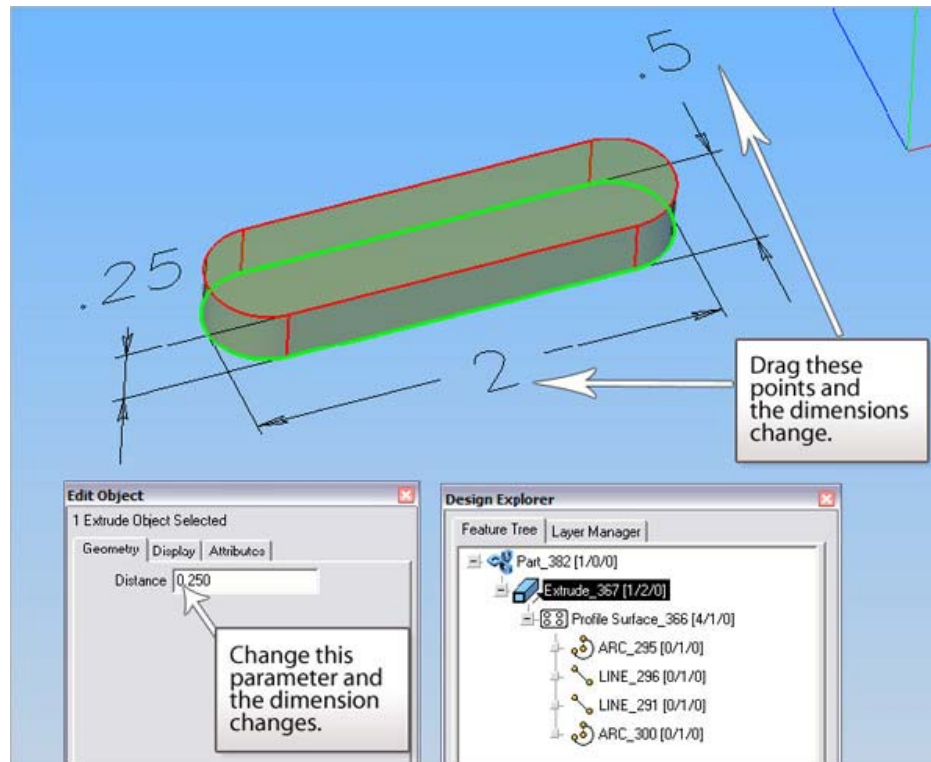
EDGE REPLACEMENT



SURFACE REPLACEMENT

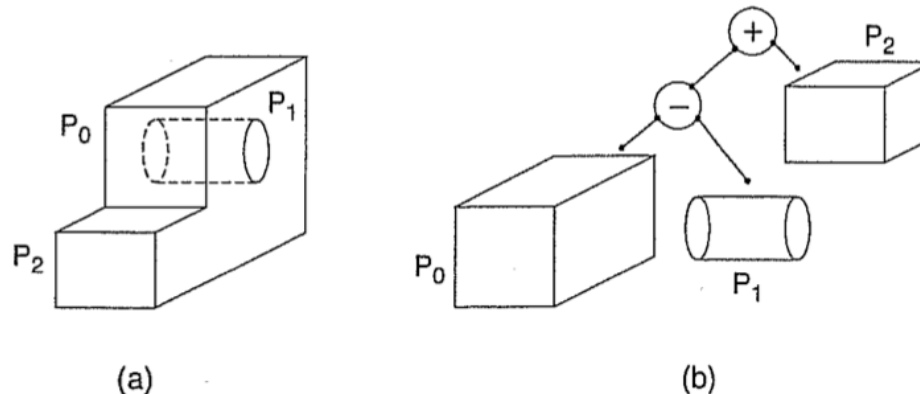
Solid Modeling: Modeling Functions

- Feature-based modeling: similar to primitive, but more complex and specialized for manufacturing, like “Hole”, or “Slot”, or “Pocket”...
- Parametric modeling: use relationships and 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 always 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

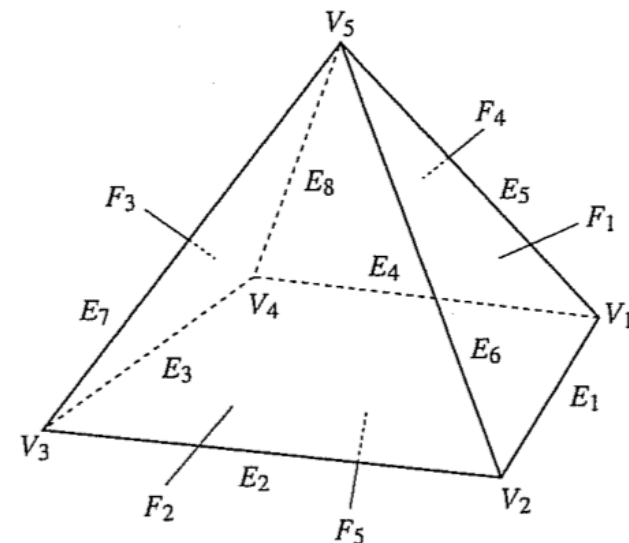
Data Structure

- **Boundary Representation (B-Rep)**

- Boundary information collected in a tree table:

Three tables for storing B-Rep

Face Table		Edge Table		Vertex Table	
Face	Edges	Edge	Vertices	Vertex	Coordinates
F ₁	E ₁ , E ₅ , E ₆	E ₁	V ₁ , V ₂	V ₁	x ₁ , y ₁ , z ₁
F ₂	E ₂ , E ₆ , E ₇	E ₂	V ₂ , V ₃	V ₂	x ₂ , y ₂ , z ₂
F ₃	E ₃ , E ₇ , E ₈	E ₃	V ₃ , V ₄	V ₃	x ₃ , y ₃ , z ₃
F ₄	E ₄ , E ₈ , E ₅	E ₄	V ₄ , V ₁	V ₄	x ₄ , y ₄ , z ₄
F ₅	E ₁ , E ₂ , E ₃ , E ₄	E ₅	V ₁ , V ₅	V ₅	x ₅ , y ₅ , z ₅
		E ₆	V ₂ , V ₅	V ₆	x ₆ , y ₆ , z ₆
		E ₇	V ₃ , V ₅		
		E ₈	V ₄ , V ₅		

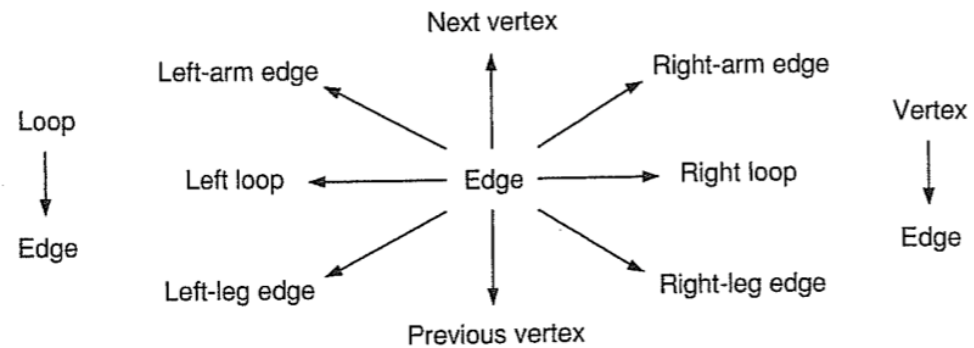
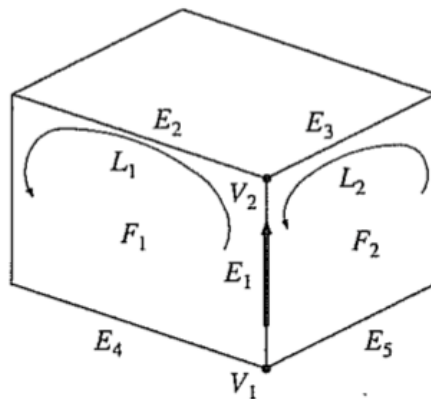


- Works only with planar polyhedra
- It needs a workaround for faces with inner and outer boundaries
- Deriving connectivity information from the tables can be cumbersome
- Variation from B-Rep uses doubly linked lists: **half-edge data structure**

Data Structure

- **Winged-edge data structure**

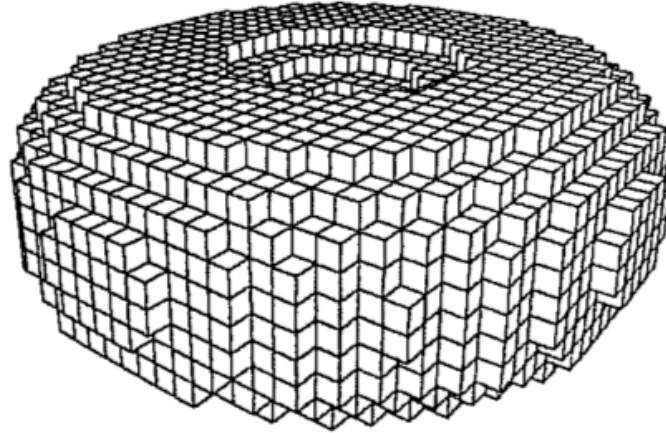
- Similar concept to half edge data structure (B-rep) but instead of being centered on faces definition it is based on edges definition:



Data Structure

- **Decomposition Model Structure**

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

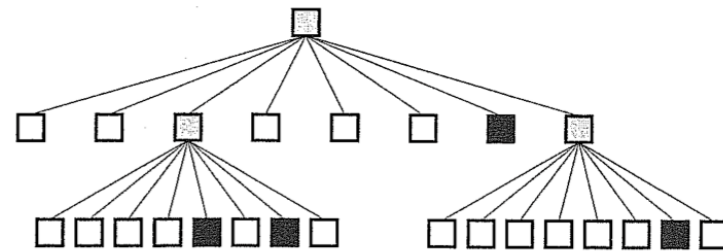
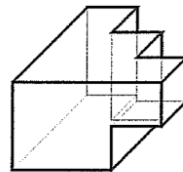
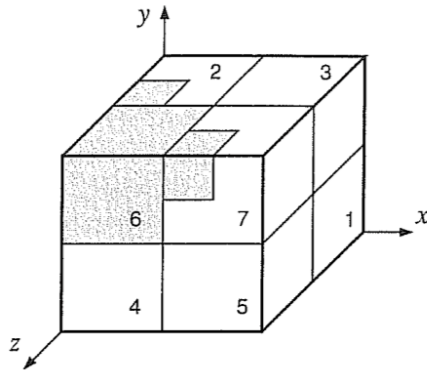


- Easy to represent arbitrary shapes
- Modification of this type of data structure is very difficult
- Easy to evaluate solid properties
- Memory usage dramatically incremented with size (or resolution)
- Voxel representation is inherently an approximation of the original solid

Data Structure

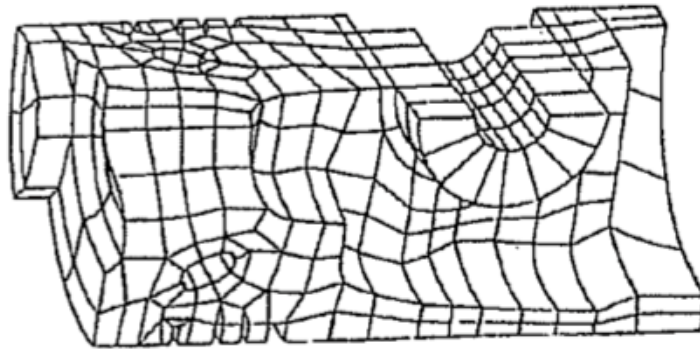
- **Decomposition Model Structure**

- 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**



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

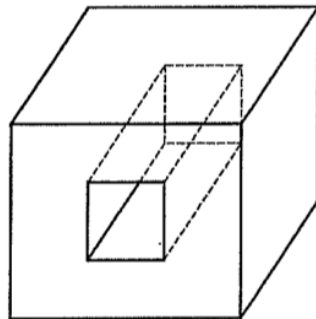
f = number of faces

h = number of hole loops

s = number of shells

p = number of passages



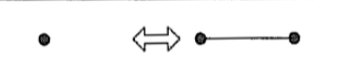
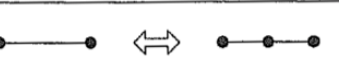
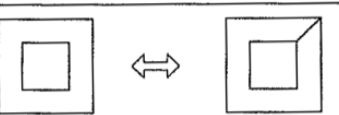

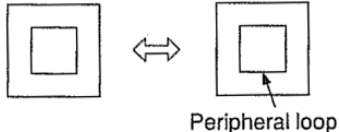
Example:



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

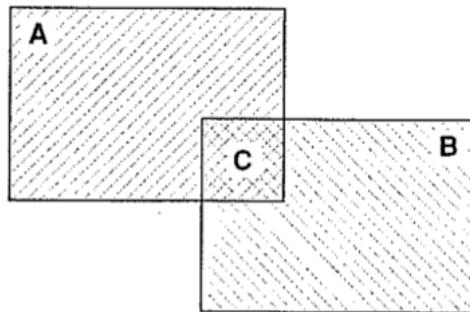
B-Rep Data Structure Modification: Euler Operators

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

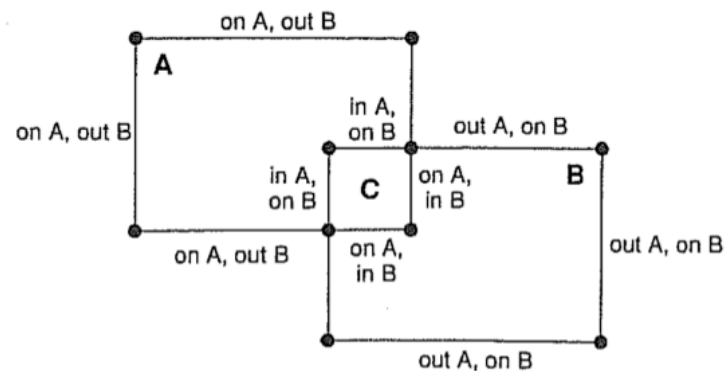
MEVLS (KEVLS)	make (kill) edge, two vertices, loop, shell	
MEL (KEL)	make (kill) edge, loop	
MEV (KEV)	make (kill) edge, vertex	
MVE (KVE)	make (kill) vertex, edge	
MEKH (KEMH)	make (kill) edge, kill (make) hole	
MZEV (KZEV)	make (kill) zero length edge, vertex	
MPKH (KPMH)	make(kill) peripheral loop,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



(a)



(b)