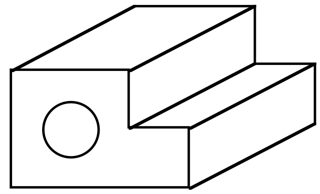
HIERARCHICAL REPRESENTATIONS OF THREE-DIMENSIONAL DATA

HANAN SAMET

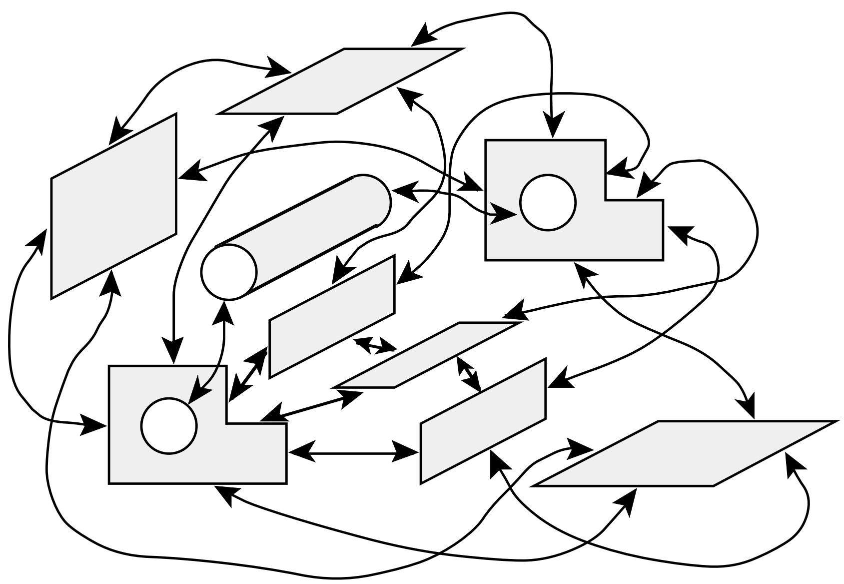
COMPUTER SCIENCE DEPARTMENT AND
CENTER FOR AUTOMATION RESEARCH AND
INSTITUTE FOR ADVANCED COMPUTER STUDIES
UNIVERSITY OF MARYLAND

COLLEGE PARK, MARYLAND 20742-3411 USA

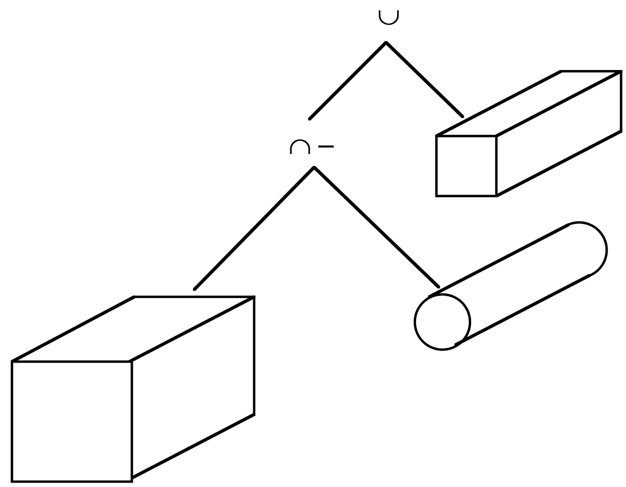
THREE-DIMENSIONAL DATA



1. Boundary model (BRep)
   - decompose boundary into set of faces, edges, and vertices
   - winged-edge representation captures topology



2. Constructive solid geometry (CSG)
   - combine primitive instances using geometric transformations and regularized Boolean set operations



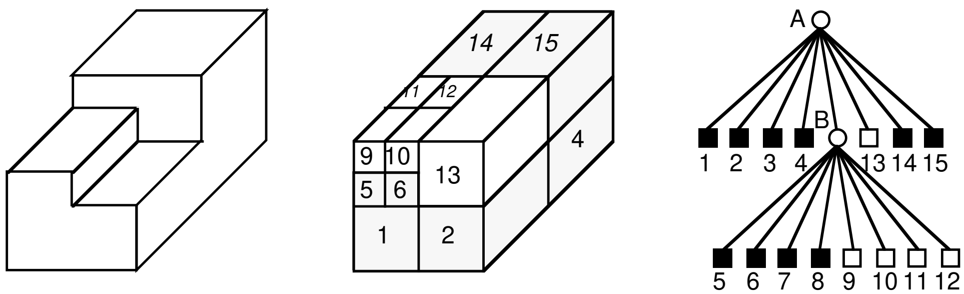
3. Interior-based
   - voxels or uniformly-sized cells (spatial enumeration)
   - cells of different size (cell decomposition-e.g., octree)

4. Sweep - volume swept by a planar or a two-dimensional shape along a curve

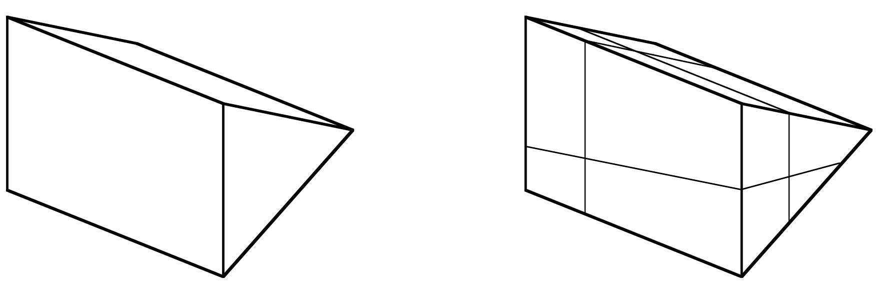
OCTREES

1. Interior (voxels)
   - analogous to region quadtree
   - approximate object by aggregating similar voxels
   - good for medical images but not for objects with planar faces

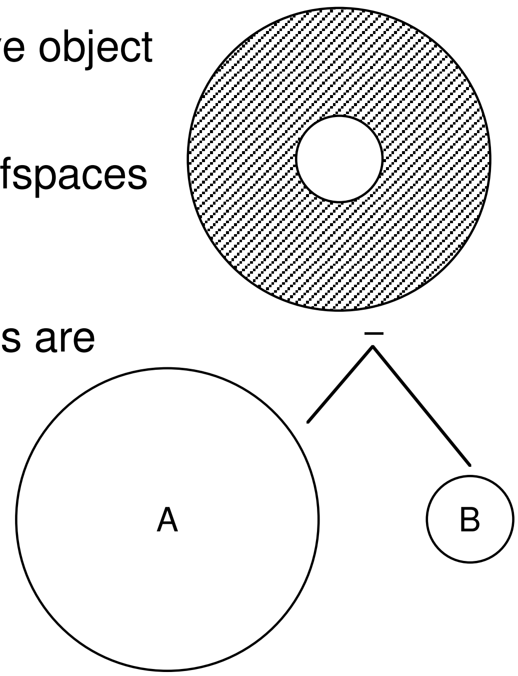
   Ex:



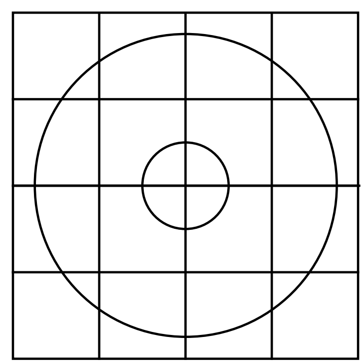
2. Boundary
   - adaptation of PM quadtree to three-dimensional data
   - decompose until each block contains
     a. one face
     b. more than one face but all meet at same edge
     c. more than one edge but all meet at same vertex
   - impose a spatial index on a boundary model (BRep)



PM-CSG TREES

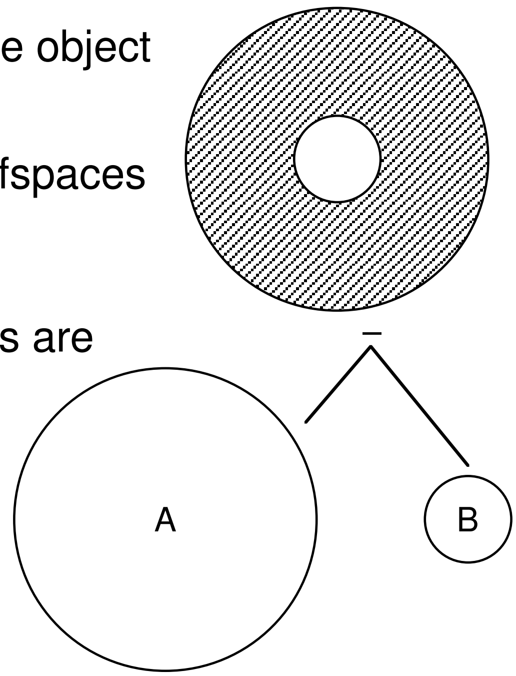


- Each leaf node refers to a primitive object instead of a vertex, edge, or face
- Primitives are not restricted to halfspaces
- Only one primitive object per cell
- Full complement of CSG operations are not present
  1. set union = gluing
  2. set difference = cutting (NO set intersection!)
- 5 types of nodes
  1. full — completely in 1 primitive object
  2. empty — not in any primitive object
  3. positive boundary — contains part of 1 primitive object while rest is empty
  4. negative boundary — contains a boundary between 2 primitive objects o_1 and o_2 such that o_1 is being subtracted from o_2
     - part corresponding to o_2 is really empty
  5. nasty — at lowest level of resolution such that no further decomposition is possible
     - e.g., the node may be occupied by more than one primitive object

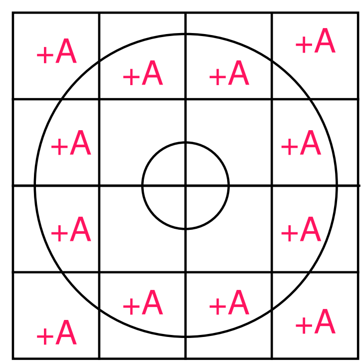


PM-CSG TREES

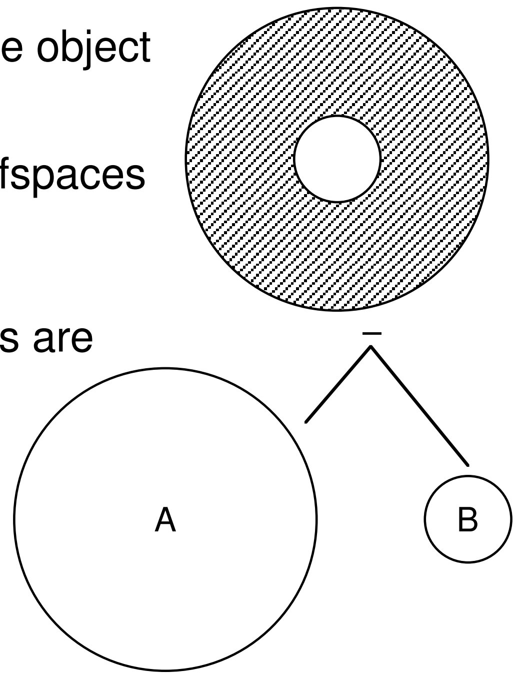
- Each leaf node refers to a primitive object instead of a vertex, edge, or face
- Primitives are not restricted to halfspaces
- Only one primitive object per cell
- Full complement of CSG operations are not present
  1. set union = gluing
  2. set difference = cutting (NO set intersection!)



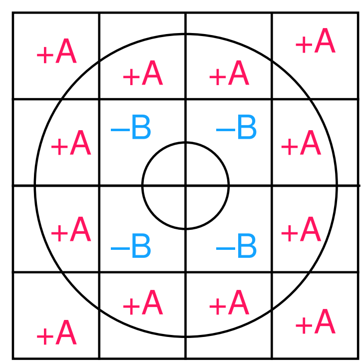
- 5 types of nodes
  1. full — completely in 1 primitive object
  2. empty — not in any primitive object
  3. positive boundary — contains part of 1 primitive object while rest is empty
  4. negative boundary — contains a boundary between 2 primitive objects O_1 and O_2 such that O_1 is being subtracted from O_2
     - part corresponding to O_2 is really empty
  5. nasty — at lowest level of resolution such that no further decomposition is possible
     - e.g., the node may be occupied by more than one primitive object



PM-CSG TREES

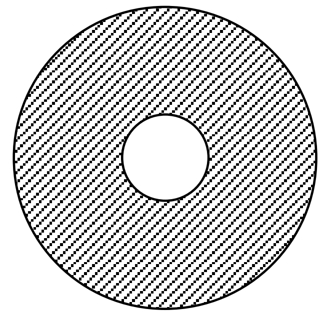


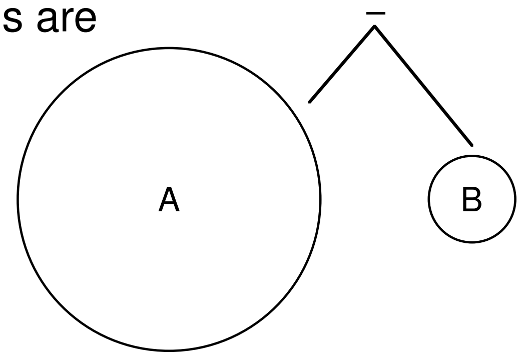
- Each leaf node refers to a primitive object instead of a vertex, edge, or face
- Primitives are not restricted to halfspaces
- Only one primitive object per cell
- Full complement of CSG operations are not present
  1. set union = gluing
  2. set difference = cutting (NO set intersection!)
- 5 types of nodes
  1. full — completely in 1 primitive object
  2. empty — not in any primitive object
  3. positive boundary — contains part of 1 primitive object while rest is empty
  4. negative boundary — contains a boundary between 2 primitive objects O_1 and O_2 such that O_1 is being subtracted from O_2
     - part corresponding to O_2 is really empty
  5. nasty — at lowest level of resolution such that no further decomposition is possible
     - e.g., the node may be occupied by more than one primitive object



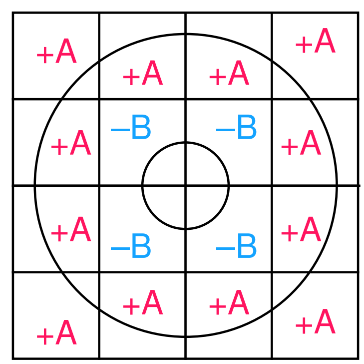
PM-CSG TREES

- Each leaf node refers to a primitive object instead of a vertex, edge, or face
- Primitives are not restricted to halfspaces
- Only one primitive object per cell
- Full complement of CSG operations are not present
  1. set union = gluing
  2. set difference = cutting (NO set intersection!)





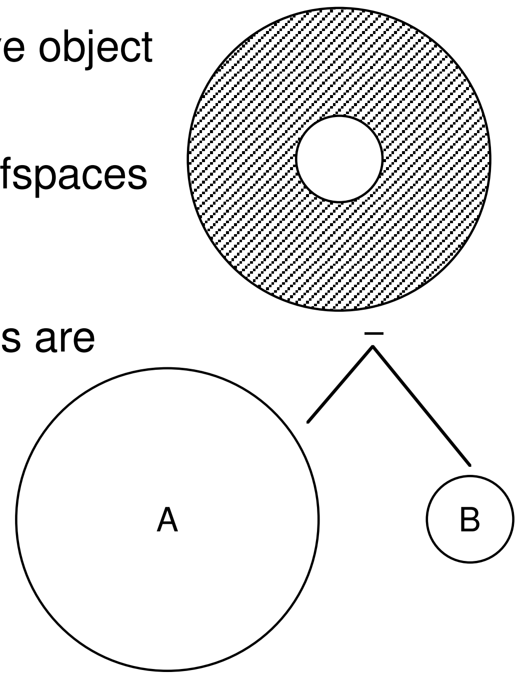
- 5 types of nodes
  1. full — completely in 1 primitive object
  2. empty — not in any primitive object
  3. positive boundary — contains part of 1 primitive object while rest is empty
  4. negative boundary — contains a boundary between 2 primitive objects O_1 and O_2 such that O_1 is being subtracted from O_2
     - part corresponding to O_2 is really empty
  5. nasty — at lowest level of resolution such that no further decomposition is possible
     - e.g., the node may be occupied by more than one primitive object



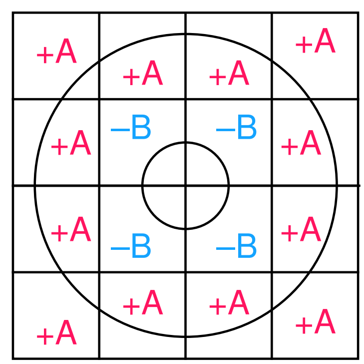
- Problem: why no set intersection as in conventional CSG?

PM-CSG TREES

- Each leaf node refers to a primitive object instead of a vertex, edge, or face
- Primitives are not restricted to halfspaces
- Only one primitive object per cell
- Full complement of CSG operations are not present
  1. set union = gluing
  2. set difference = cutting (NO set intersection!)



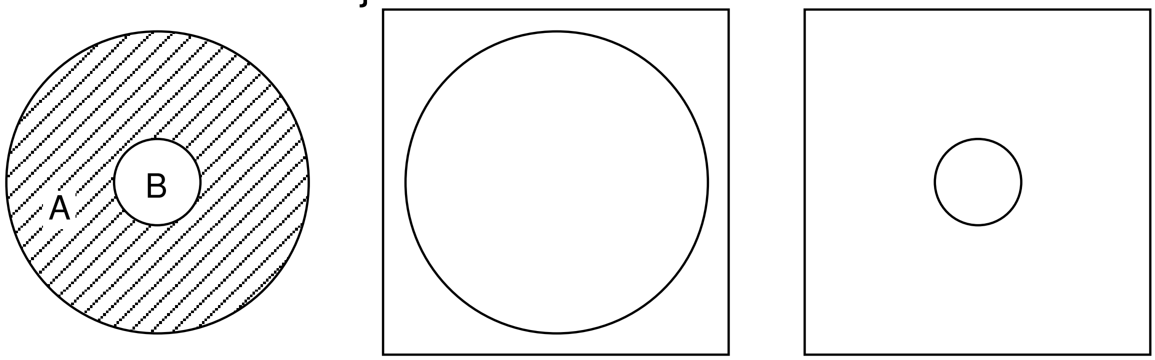
- 5 types of nodes
  1. full — completely in 1 primitive object
  2. empty — not in any primitive object
  3. positive boundary — contains part of 1 primitive object while rest is empty
  4. negative boundary — contains a boundary between 2 primitive objects o_1 and o_2 such that o_1 is being subtracted from o_2
     - part corresponding to o_2 is really empty
  5. nasty — at lowest level of resolution such that no further decomposition is possible
     - e.g., the node may be occupied by more than one primitive object



- Problem: why no set intersection as in conventional CSG?
- Solution: if operand primitives are not disjoint, then can't always separate them so each cell has just one primitive

EXAMPLE OF PM-CSG TREE CONSTRUCTION

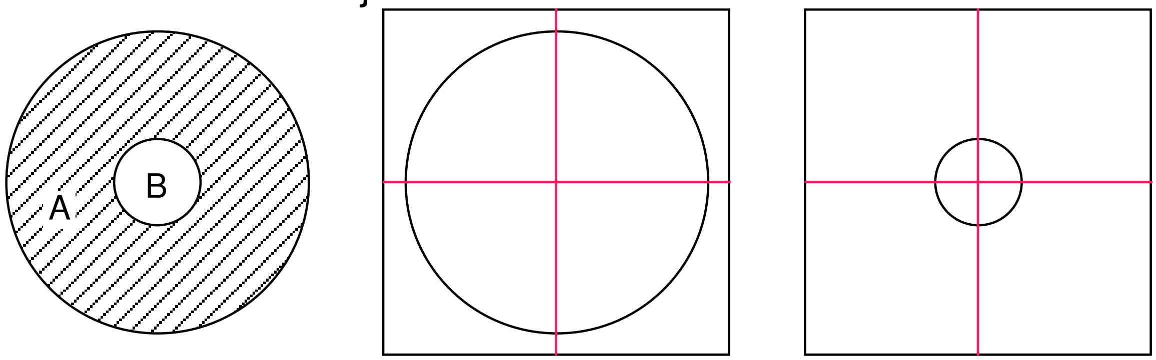
- Ex: two circular objects



1. Each PM-CSG tree consists of one boundary node

EXAMPLE OF PM-CSG TREE CONSTRUCTION

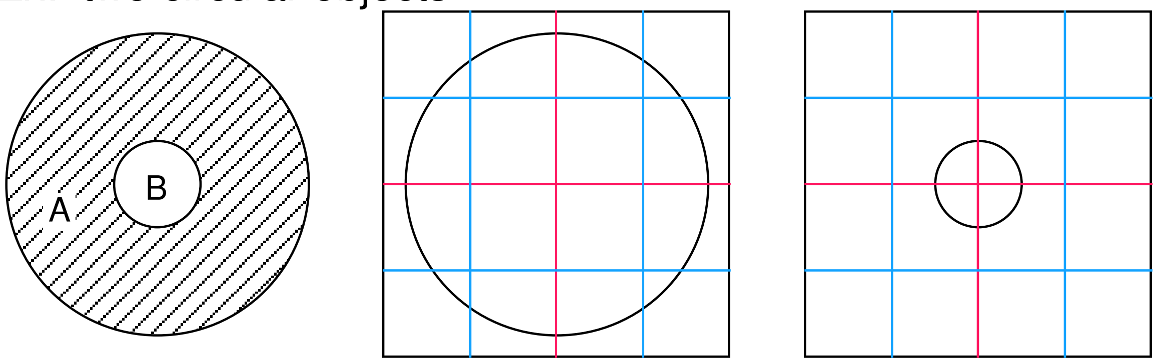
- Ex: two circular objects



1. Each PM-CSG tree consists of one boundary node
   - taking their difference does not yield a PM-CSG tree leaf node
   - decompose both trees as neither node is full or empty
2. Each node in the trees is a boundary node

EXAMPLE OF PM-CSG TREE CONSTRUCTION

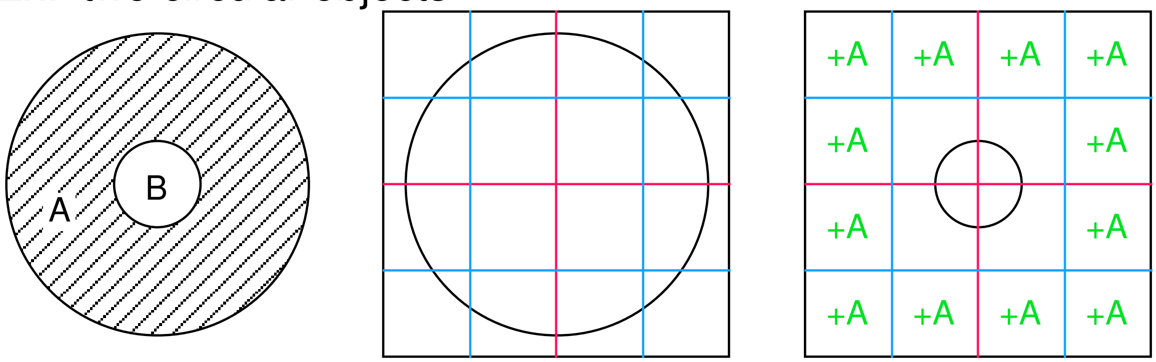
- Ex: two circular objects



1. Each PM-CSG tree consists of one boundary node
   - taking their difference does not yield a PM-CSG tree leaf node
   - decompose both trees as neither node is full or empty
2. Each node in the trees is a boundary node
   - taking their difference does not yield any PM-CSG tree leaf nodes
   - decompose corresponding nodes in both trees as none of the nodes resulting from the subtraction is full or empty
3. Trees contain empty, full, and boundary nodes

EXAMPLE OF PM-CSG TREE CONSTRUCTION

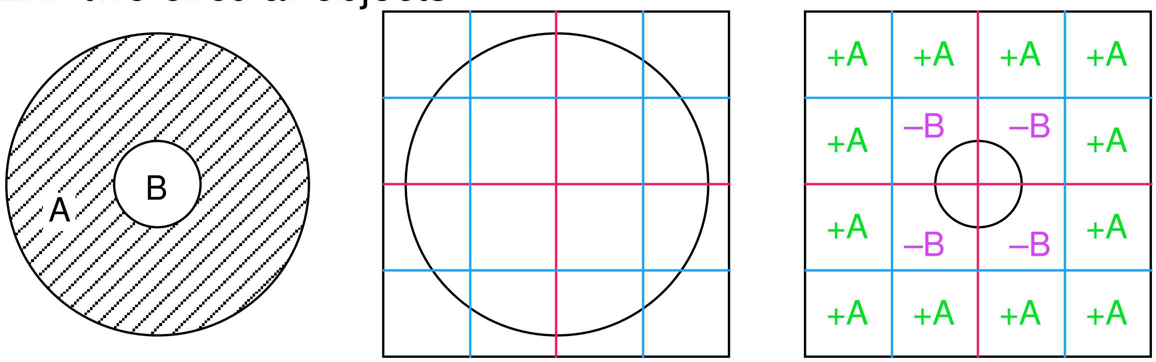
- Ex: two circular objects



1. Each PM-CSG tree consists of one boundary node
   - taking their difference does not yield a PM-CSG tree leaf node
   - decompose both trees as neither node is full or empty
2. Each node in the trees is a boundary node
   - taking their difference does not yield any PM-CSG tree leaf nodes
   - decompose corresponding nodes in both trees as none of the nodes resulting from the subtraction is full or empty
3. Trees contain empty, full, and boundary nodes
   - boundary minus empty yields positive boundary nodes

EXAMPLE OF PM-CSG TREE CONSTRUCTION

- Ex: two circular objects



1. Each PM-CSG tree consists of one boundary node
   - taking their difference does not yield a PM-CSG tree leaf node
   - decompose both trees as neither node is full or empty
2. Each node in the trees is a boundary node
   - taking their difference does not yield any PM-CSG tree leaf nodes
   - decompose corresponding nodes in both trees as none of the nodes resulting from the subtraction is full or empty
3. Trees contain empty, full, and boundary nodes
   - boundary minus empty yields positive boundary nodes
   - full minus boundary yields negative boundary nodes