WINGED-EDGE DATA STRUCTURE

Hanan Samet

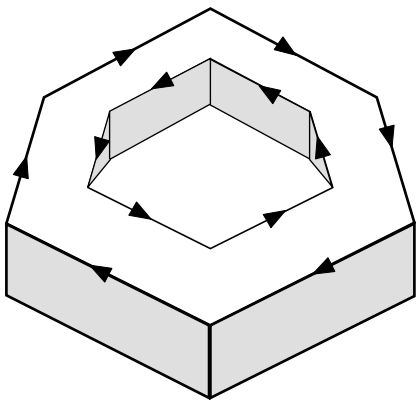
Computer Science Department and
Center for Automation Research and
Institute for Advanced Computer Studies
University of Maryland
College Park, Maryland 20742
e-mail: hjs@umiacs.umd.edu

EXAMPLE GEOMETRIC DATABASE

- Boundary model (BRep)
- Assume a solid object bounded by compact orientable two-manifold surfaces (i.e., only two faces meet at an edge)
- Primitive topological entities
  1. vertices
  2. edges
  3. faces
- Nonprimitive topological entities for multiply-connected objects (e.g., holes, internal cavities)
  1. shell—a maximally connected set of faces

     Ex: cube within a cube has two shells (internal and external)
  2. loop on a face—closed chain of edges bounding the face

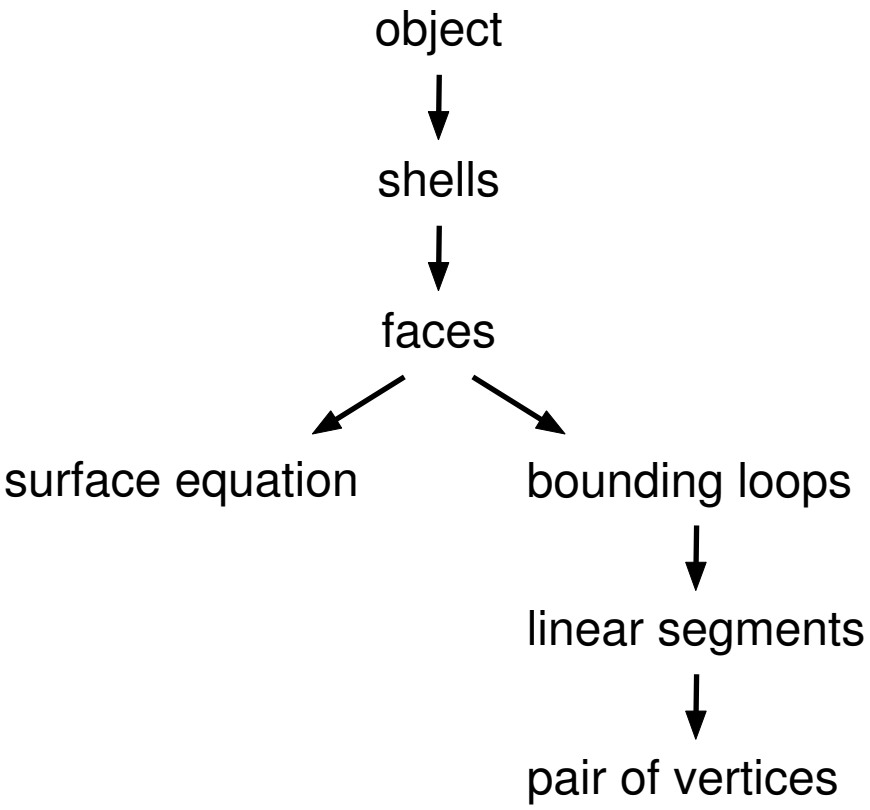
     Ex: nut with 2 loops for each of top and bottom faces



DESCRIPTION OF THE BOUNDARY OF AN OBJECT

1. Topological description
   - adjacency relationships between pairs of individual topological entities (e.g., edge-face, loop-face,…)
   - maximum of 25 adjacency relationships (ordered pairs)
2. Geometric description
   - shape and location in space of each of the primitive topological entities
     a. face: surface equation
     b. edge: endpoints, spline curve, etc.
     c. vertex: Cartesian coordinates
     d. shell and loop: none since a collection of primitive topological entities

HIERARCHICAL REPRESENTATION



1. Decompose object into shells
2. Each shell is a collection of faces
3. Each face
   - surface equation, AND
   - collection of its bounding loops
4. Each loop is a chain of edges
5. Each edge is a collection of linear segments
6. Each linear segment is a pair of vertices

SPECIFYING A BOUNDARY MODEL

1. Set of topological entities defining the object's boundary
2. Subset of 25 different adjacency relationships between pairs of individual entities
    - relations are ordered
    - edge-face is different from face-edge
        a. edge-face associates with each edge the two adjacent faces
        b. face-edge associates with each face the edges that bound it

EXAMPLE BOUNDARY REPRESENTATIONS

1. Symmetric structure uses face-edge, vertex-edge, edge-face, and edge-vertex relationships

2. Face adjacency graph uses face-edge, vertex-face, edge-face, and face-vertex relationships

3. If multiple shells and multiply-connected faces

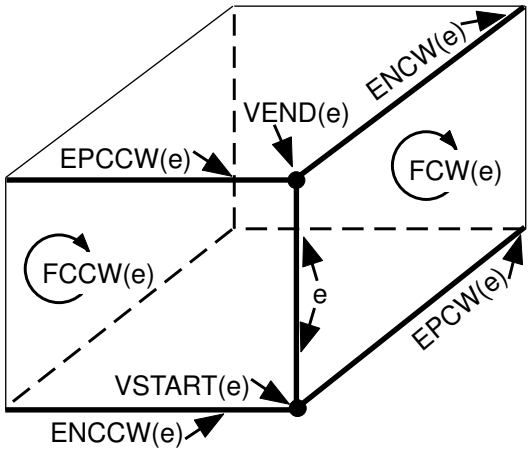
   - edge-face = edge-loop and loop-face relationships
   - face-edge = face-loop and loop-edge relationships

4. Edge-based

   - drawback of face-edge and vertex-face is that they violate the first normal form (1NF) since a variable amount of information is associated with the second attribute (i.e., edge and face, respectively)
   - no such problem with edge-face and edge-vertex relationships since assume 2-manifold surface (e.g., only two faces may meet at a vertex)
   - basis of winged-edge representation

WINGED-EDGE REPRESENTATION

- Physical interpretation of attributes of relations



- Vertex relation:

  1. vertex (primary key)

  2–4. *x*, *y*, *z* coordinate values (key)

  5. identity of an edge which starts at the vertex (ESTART)

     - enables extracting set of edges incident at a vertex in time proportional to the number of edges

- Face relation:

  1. face (primary key)

  2. identity of an edge which is part of the face (ESTART)

     - enables extracting set of edges comprising a face in time proportional to the number of edges

- Edge relation:

  1. edge (primary key)

  2–3. start (VSTART) and end (VEND) vertices (key)

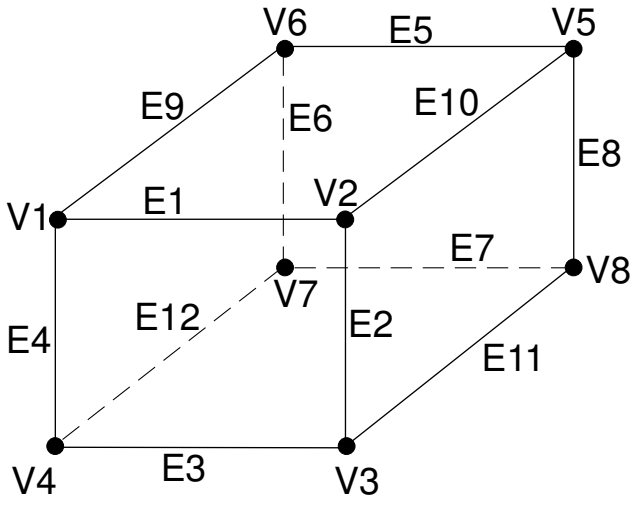
  4–5. two adjacent faces (FCW and FCCW) (key)

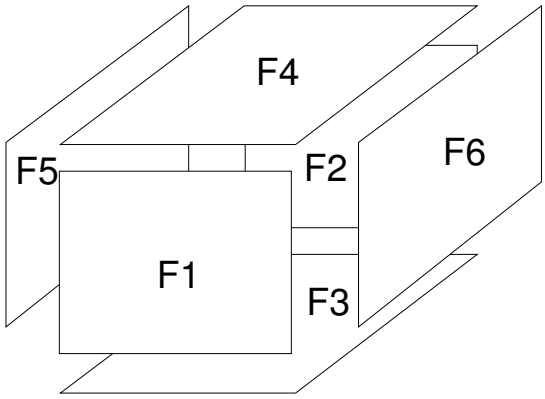
  6–7. preceding and next edges in one face (EPCW and ENCW) (key)

  8–9. preceding and next edges in other face (EPCCW and ENCCW) (key)

- Question: are the relations in 2NF, 3NF, or BCNF?

EXAMPLE OBJECT AND ITS REPRESENTATION





Vertex Table

VERTEX	X	Y	Z	ESTART
V_1	X_1	Y_1	Z_1	E_1
V_2	X_2	Y_2	Z_2	E_2
V_3	X_3	Y_3	Z_3	E_3
V_4	X_4	Y_4	Z_4	E_4
V_5	X_5	Y_5	Z_5	E_5
V_6	X_6	Y_6	Z_6	E_6
V_7	X_7	Y_7	Z_7	E_7
V_8	X_8	Y_8	Z_8	E_8

Face Table

FACE	ESTART
F_1	E_1
F_2	E_5
F_3	E_{11}
F_4	E_9
F_5	E_4
F_6	E_8

Edge Table

EDGE	VSTART	VEND	EPCW	ENCW	EPCCW	ENCCW	FCW	FCCW
E_1	V_1	V_2	E_4	E_2	E_{10}	E_9	F_1	F_4
E_2	V_2	V_3	E_1	E_3	E_{11}	E_{10}	F_1	F_6
E_3	V_3	V_4	E_2	E_4	E_{12}	E_{11}	F_1	F_3
E_4	V_4	V_1	E_3	E_1	E_9	E_{12}	F_1	F_5
E_5	V_5	V_6	E_8	E_6	E_9	E_{10}	F_2	F_4
E_6	V_6	V_7	E_5	E_7	E_{12}	E_9	F_2	F_5
E_7	V_7	V_8	E_6	E_8	E_{11}	E_{12}	F_2	F_3
E_8	V_8	V_5	E_7	E_5	E_{10}	E_{11}	F_2	F_6
E_9	V_1	V_6	E_1	E_5	E_6	E_4	F_4	F_5
E_{10}	V_5	V_2	E_5	E_1	E_2	E_8	F_4	F_6
E_{11}	V_3	V_8	E_3	E_7	E_8	E_2	F_3	F_6
E_{12}	V_7	V_4	E_7	E_3	E_4	E_6	F_3	F_5

REMOVING REDUNDANT INFORMATION IN THE RELATIONS

- Minimum information in the relations:

  1. vertex relation: vertex, *x*, *y*, *z*

  2. face relation: face, ESTART

  3. edge relation: edge, VSTART, VEND, FCW, FCCW

- Without FCW and FCCW we cannot get to the faces from the edges and hence we cannot determine the edges that are adjacent to a face

- EPCW and ENCW or EPCCW and ENCCW are inadequate by themselves to identify the edges that are adjacent to a face

- Face relation is not absolutely necessary

- Face relation (ESTART) is only useful to avoid having to perform an *O*(number of edges) search on the FCW or FCCW fields to determine an edge given a face

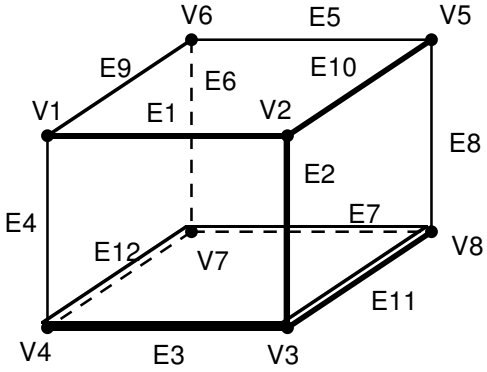
- The ESTART field should also be included in the vertex relation

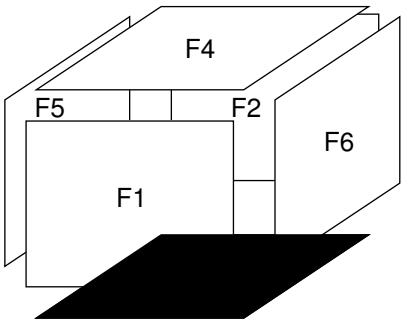
CW EDGES IN A FACE

1. look up face F in face table and find an edge E
2. find E in edge table; repeat until read E again
   - if F=FCW(E) then next edge is ENCW(E)
   - else F=FCCW(E) and next edge is ENCCW(E)

- Ex: get edges in face F_3 in clockwise order

FACE	ESTART
F_1	E_1
F_2	E_5
F_3	E_7
F_4	E_9
F_5	E_4
F_6	E_8





EDGE	VSTART	VEND	EPCW	ENCW	EPCCW	ENCCW	FCW	FCCW
E_1	V_1	V_2	E_4	E_2	E_{10}	E_9	F_1	F_4
E_2	V_2	V_3	E_1	E_3	E_{11}	E_{10}	F_1	F_6
E_3	V_3	V_4	E_2	E_4	E_{12}	E_{11}	F_1	F_3
E_4	V_4	V_1	E_3	E_1	E_9	E_{12}	F_1	F_5
E_5	V_5	V_6	E_8	E_6	E_9	E_{10}	F_2	F_4
E_6	V_6	V_7	E_5	E_7	E_{12}	E_9	F_2	F_5
E_7	V_7	V_8	E_6	E_8	E_{11}	E_{12}	F_2	F_3
E_8	V_8	V_5	E_7	E_5	E_{10}	E_{11}	F_2	F_6
E_9	V_1	V_6	E_1	E_5	E_6	E_4	F_4	F_5
E_{10}	V_5	V_2	E_5	E_1	E_2	E_8	F_4	F_6
E_{11}	V_3	V_8	E_3	E_7	E_8	E_2	F_3	F_6
E_{12}	V_7	V_4	E_7	E_3	E_4	E_6	F_3	F_5

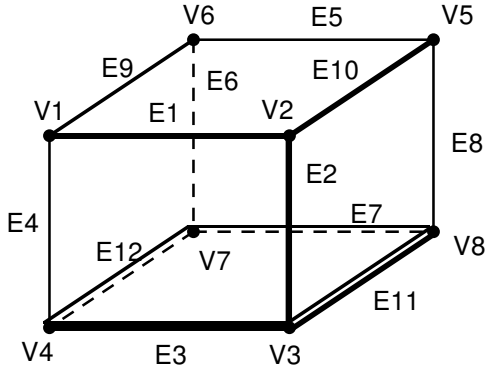
1. look up face F_3 in face table and find edge E_7
2. look up edge E_7 in edge table and since F_3=FCCW(E_7), next edge is ENCCW(E_7)=E_{12}
3. look up edge E_{12} in edge table and since F_3=FCW(E_{12}), next edge is ENCW(E_{12})=E_3
4. look up edge E_3 in edge table and since F_3=FCCW(E_3), next edge is ENCCW(E_3)=E_{11}
5. look up edge E_{11} in edge table and since F_3=FCW(E_{11}), next edge is ENCW(E_{11})=E_7; now, we are done!

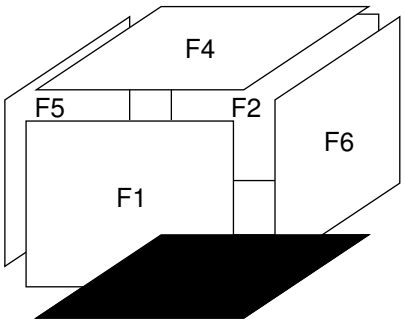
CCW EDGES IN A FACE

1. look up face F in face table and find an edge E
2. find E in edge table; repeat until read E again
   - if F=FCW(E) then next edge is EPCW(E)
   - else F=FCCW(E) and next edge is EPCCW(E)

- Ex: get edges in face F_3 in counterclockwise order

FACE	ESTART
F_1	E_1
F_2	E_5
F_3	E_7
F_4	E_9
F_5	E_4
F_6	E_8





EDGE	VSTART	VEND	EPCW	ENCW	EPCCW	ENCCW	FCW	FCCW
E_1	V_1	V_2	E_4	E_2	E_{10}	E_9	F_1	F_4
E_2	V_2	V_3	E_1	E_3	E_{11}	E_{10}	F_1	F_6
E_3	V_3	V_4	E_2	E_4	E_{12}	E_{11}	F_1	F_3
E_4	V_4	V_1	E_3	E_1	E_9	E_{12}	F_1	F_5
E_5	V_5	V_6	E_8	E_6	E_9	E_{10}	F_2	F_4
E_6	V_6	V_7	E_5	E_7	E_{12}	E_9	F_2	F_5
E_7	V_7	V_8	E_6	E_8	E_{11}	E_{12}	F_2	F_3
E_8	V_8	V_5	E_7	E_5	E_{10}	E_{11}	F_2	F_6
E_9	V_1	V_6	E_1	E_5	E_6	E_4	F_4	F_5
E_{10}	V_5	V_2	E_5	E_1	E_2	E_8	F_4	F_6
E_{11}	V_3	V_8	E_3	E_7	E_8	E_2	F_3	F_6
E_{12}	V_7	V_4	E_7	E_3	E_4	E_6	F_3	F_5

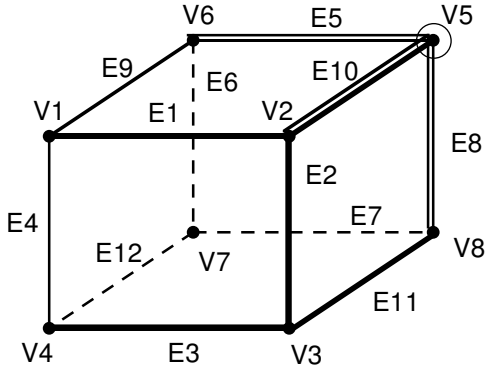
1. look up face F_3 in face table and find edge E_7
2. look up edge E_7 in edge table and since F_3=FCCW(E_7), next edge is EPCCW(E_7)=E_{11}
3. look up edge E_{11} in edge table and since F_3=FCW(E_{11}), next edge is EPCW(E_{11})=E_3
4. look up edge E_3 in edge table and since F_3=FCCW(E_3), next edge is EPCCW(E_3)=E_{12}
5. look up edge E_{12} in edge table and since F_3=FCW(E_{12}), next edge is EPCW(E_{12})=E_7; now, we are done!

CW EDGES MEETING AT A VERTEX

1. look up vertex V in face table and find an edge E
2. find E in edge table; repeat until read E again
   - if V=VSTART(E) then next edge is EPCW(E)
   - else V=VEND(E) and next edge is EPCCW(E)

- Ex: get edges meeting at vertex V_5 in clockwise order

VERTEX	X	Y	Z	ESTART
V_1	X_1	Y_1	Z_1	E_1
V_2	X_2	Y_2	Z_2	E_2
V_3	X_3	Y_3	Z_3	E_3
V_4	X_4	Y_4	Z_4	E_4
V_5	X_5	Y_5	Z_5	E_5
V_6	X_6	Y_6	Z_6	E_6
V_7	X_7	Y_7	Z_7	E_7
V_8	X_8	Y_8	Z_8	E_8



EDGE	VSTART	VEND	EPCW	ENCW	EPCCW	ENCCW	FCW	FCCW
E_1	V_1	V_2	E_4	E_2	E_{10}	E_9	F_1	F_4
E_2	V_2	V_3	E_1	E_3	E_{11}	E_{10}	F_1	F_6
E_3	V_3	V_4	E_2	E_4	E_{12}	E_{11}	F_1	F_3
E_4	V_4	V_1	E_3	E_1	E_9	E_{12}	F_1	F_5
E_5	V_5	V_6	E_8	E_6	E_9	E_{10}	F_2	F_4
E_6	V_6	V_7	E_5	E_7	E_{12}	E_9	F_2	F_5
E_7	V_7	V_8	E_6	E_8	E_{11}	E_{12}	F_2	F_3
E_8	V_8	V_5	E_7	E_5	E_{10}	E_{11}	F_2	F_6
E_9	V_1	V_6	E_1	E_5	E_6	E_4	F_4	F_5
E_{10}	V_5	V_2	E_5	E_1	E_2	E_8	F_4	F_6
E_{11}	V_3	V_8	E_3	E_7	E_8	E_2	F_3	F_6
E_{12}	V_7	V_4	E_7	E_3	E_4	E_6	F_3	F_5

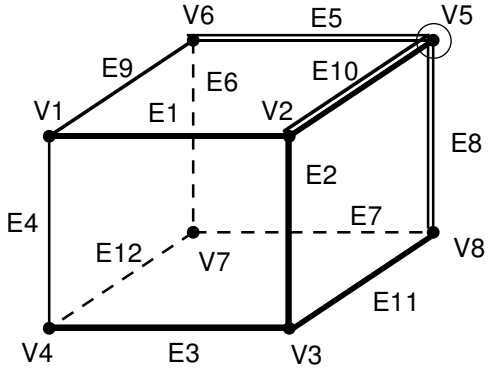
1. look up vertex V_5 in vertex table and find edge E_5
2. look up edge E_5 in edge table and since V_5=VSTART(E_5), next edge is EPCW(E_5)=E_8
3. look up edge E_8 in edge table and since V_5=VEND(E_8), next edge is EPCCW(E_8)=E_{10}
4. look up edge E_{10} in edge table and since V_5=VSTART(E_{10}), next edge is EPCW(E_{10})=E_5; now, we are done!

CCW EDGES MEETING AT A VERTEX

1. look up vertex V in face table and find an edge E
2. find E in edge table; repeat until read E again
   - if V=VSTART(E) then next edge is ENCCW(E)
   - else V=VEND(E) and next edge is ENCW(E)

- Ex: get edges meeting at V_5 in counterclockwise order

VERTEX	X	Y	Z	ESTART
V_1	X_1	Y_1	Z_1	E_1
V_2	X_2	Y_2	Z_2	E_2
V_3	X_3	Y_3	Z_3	E_3
V_4	X_4	Y_4	Z_4	E_4
V_5	X_5	Y_5	Z_5	E_5
V_6	X_6	Y_6	Z_6	E_6
V_7	X_7	Y_7	Z_7	E_7
V_8	X_8	Y_8	Z_8	E_8



EDGE	VSTART	VEND	EPCW	ENCW	EPCCW	ENCCW	FCW	FCCW
E_1	V_1	V_2	E_4	E_2	E_{10}	E_9	F_1	F_4
E_2	V_2	V_3	E_1	E_3	E_{11}	E_{10}	F_1	F_6
E_3	V_3	V_4	E_2	E_4	E_{12}	E_{11}	F_1	F_3
E_4	V_4	V_1	E_3	E_1	E_9	E_{12}	F_1	F_5
E_5	V_5	V_6	E_8	E_6	E_9	E_{10}	F_2	F_4
E_6	V_6	V_7	E_5	E_7	E_{12}	E_9	F_2	F_5
E_7	V_7	V_8	E_6	E_8	E_{11}	E_{12}	F_2	F_3
E_8	V_8	V_5	E_7	E_5	E_{10}	E_{11}	F_2	F_6
E_9	V_1	V_6	E_1	E_5	E_6	E_4	F_4	F_5
E_{10}	V_5	V_2	E_5	E_1	E_2	E_8	F_4	F_6
E_{11}	V_3	V_8	E_3	E_7	E_8	E_2	F_3	F_6
E_{12}	V_7	V_4	E_7	E_3	E_4	E_6	F_3	F_5

1. look up vertex V_5 in vertex table and find edge E_5
2. look up edge E_5 in edge table and since V_5=VSTART(E_5), next edge is ENCCW(E_5)=E_{10}
3. look up edge E_{10} in edge table and since V_5=VSTART(E_{10}), next edge is ENCCW(E_{10})=E_8
4. look up edge E_8 in edge table and since V_5=VEND(E_8), next edge is ENCW(E_8)=E_5; now, we are done!