





**CONESTAV-IPN**  
Biblioteca de Ingeniería Eléctrica



F0010006637

CENTRO DE INVESTIGACION Y DE  
ESTUDIOS AVANZADOS DEL  
I. P. N.  
BIBLIOTECA  
INGENIERIA ELECTRICA

CENTRO DE INVESTIGACION Y DE  
ESTUDIOS AVANZADOS DEL  
I. P. N.  
BIBLIOTECA  
INGENIERIA ELECTRICA

CLASSIF	87-21
NUMER	B1-10673
RECHA	9-14-87
PROCES	Don

XM

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS  
DEL  
INSTITUTO POLITECNICO NACIONAL

DEPARTAMENTO DE INGENIERIA ELECTRICA  
SECCION DE COMPUTACION

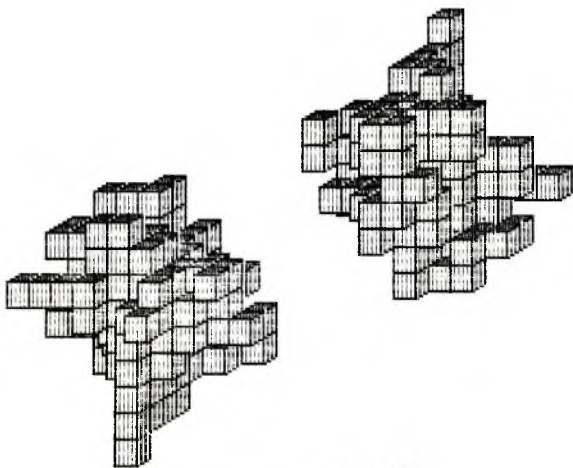
"UNA REPRESENTACION LINEAL DE SOLIDOS"

Tesis que presenta el Lic. Feliú D. Sagols Troncoso para  
obtener el grado de MAESTRO EN CIENCIAS en la especialidad  
de INGENIERIA ELECTRICA. Trabajo dirigido por el Dr.  
Guillermo Morales Luna.

Becario del COSNET.

México, D.F., Abril de 1987

CENTRO DE INVESTIGACION Y DE  
ESTUDIOS AVANZADOS DEL  
I.P.N.  
BIBLIOTECA  
INGENIERIA ELECTRICA



CENTRO DE INVESTIGACION Y DE  
ESTUDIOS AVANZADOS DEL  
I. P. N.  
BIBLIOTECA  
INGENIERIA ELECTRICA

## AGRADECIMIENTOS:

A la Sección de Computación del Centro de Investigación y Estudios Avanzados del IPN. En particular al Dr. Guillermo Morales Luna, por la dirección que hizo de este trabajo y sus valiosas sugerencias.

Al Departamento de Matemáticas de Centro de Investigación y Estudios Avanzados Del IPN., por las facilidades brindadas para utilizar su Laboratorio de Cómputo, especialmente su equipo donado por CONACYT.

A los Drs. Renato Barrera y Mike Porter K., quienes aceptaron hacer la revisión del trabajo y aportaron importantes ideas para mejorarlo.

Al Consejo del Sistema Nacional de Educación Tecnológica (COSNET). Por la ayuda brindada para la realización de este trabajo.

CENTRO DE INVESTIGACION Y DE  
ESTUDIOS AVANZADOS DEL  
I. P. N.  
BIBLIOTECA  
INGENIERIA ELECTRICA





Dedicado a mi mama.

Sra. Ana María Troncoso de SagoIs.



## PREFACIO

Un problema cuya solución tiene varias aplicaciones en la geometría computacional, es el de recorrer contornos de imágenes conexas discretizadas mediante pixels. Recorrer significa encontrar una sucesión de direcciones (norte, sur, este y oeste) de forma tal que partiendo de alguna arista en la frontera de la imagen, si nos movemos según las direcciones indicadas vayamos siempre sobre la frontera del sólido. (En regiones sin hoyos siempre es posible).

En esta tesis tratamos la generalización de recorridos a tres dimensiones. Explicado de manera simple, significa que si nos dan un sólido conexo discretizado en pequeños cubos, el lector encontrará un algoritmo para pintar con una línea continua todas las caras del sólido sin pasar por una dos veces. El problema no lo resolvemos en toda su generalidad pero damos en un buen número de casos una solución bastante útil que se puede programar con algoritmos que tienen complejidad lineal. El tema que tratamos puede parecer un juego, pero en realidad tiene muchas aplicaciones en geometría tridimensional. Veamos qué puede encontrar el lector en este trabajo:

En el capítulo I, explicamos con todo detalle el problema de recorrer la frontera de una imagen bidimensional y damos los lineamientos para generalizar el problema a tres dimensiones. El capítulo II contiene una serie de definiciones básicas así como el formalismo elemental para la determinación de recorridos. En el capítulo III, damos un algoritmo muy eficiente para determinar la gráfica de adyacencia de caras de un sólido, (esto equivale a encontrar el cascarón conexo de un sólido), veremos que para encontrar recorridos sobre sólidos no hay más que hacer una simple generalización del algoritmo presentado en este capítulo. El capítulo IV es la parte medular del trabajo, pues en él damos el algoritmo para recorrer las caras de un sólido. Finalmente en el capítulo V resumimos algunas aplicaciones de este tipo de recorridos, y mencionamos resultados sobre implantaciones de los algoritmos en la computadora (Se uso una IBM-PC con Turbo-Pascal).



CAPITULO I.....	1
CADENAS DE FREEMAN.	
1.1 Descripción de las cadenas de Freeman.....	1
1.2 Construcción de las cadena de Freeman de dos pixels adyacentes.....	4
1.3 Compresión de cadenas de Freeman.....	5
CAPITULO II.....	8
DEFINICIONES BASICAS	
2.1 Conceptos elementales.....	8
2.2 Matriz de representación de sólidos.....	10
2.3 El concepto de recorrido.....	10
2.4 Observaciones sobre el planteamiento del problema.....	13
CAPITULO III.....	16
GRAFICA DE ADYACENCIA DE CARAS DE UN SOLIDO	
3.1 Definiciones básicas relacionadas con la gráfica de conectividad de caras.....	16
3.2 Estructuras de datos (usando Pascal).....	19
3.3 Gráfica de adyacencia de un cubo unitario.....	21
3.4 Gráfica de adyacencia de dos cubos adyacentes... ..	22
3.5 Gráfica de adyacencia de un sólido general.....	25
3.6 Algoritmo para encontrar la gráfica de adyacencia de caras de un sólido.....	30
3.7 Ejemplo de aplicación del algoritmo para encontrar la gráfica de adyacencia de un sólido. .	32
3.8 Complejidad en el algoritmo.....	34
3.9 Observaciones y conclusiones.....	34
CAPITULO IV.....	36
ALGORITMO PARA ENCONTRAR UN RECORRIDO SOBRE LAS CARAS DE UN SOLIDO	
4.1 Recorridos para cubos unitarios.....	36
4.2 Biyección entre las sucesiones hamiltonianas ortogonales y los vértices de un cubo unitario..	40
4.3 Recorridos sobre sólidos formados por dos cubos unitarios adyacentes.....	45
4.4 Recorridos sobre sólidos formados por más de dos cubos unitarios.....	51
4.5 Algoritmo para recorrer sólidos simples.....	55
4.6 Recorridos sobre sólidos no simples.....	59
4.7 Generación de recorridos usando un nuevo tipo de asignación de recorridos a cubos unitarios.....	70
4.8 Conclusiones.....	74
CAPITULO V.....	76
RESULTADOS Y APLICACIONES	
5.1 Algoritmo para generar sólidos aleatoriamente... ..	76
5.2 Resultados de los algoritmos expuestos en el capítulo anterior.....	79
5.3 Aplicaciones.....	80



CADENAS DE FREEMAN.

El problema que se estudia en este trabajo, es el de representar un sólido formado por "pixels" (pequeños cubos del mismo tipo) en tres dimensiones, utilizando una estructura secuencial, donde aparecen las caras en la frontera de dicho sólido. Esta representación es completamente libre de apuntadores y además admite técnicas de compresión de código.

Antes de empezar a estudiar los algoritmos, es conveniente hablar del problema visto en sólo dos dimensiones.

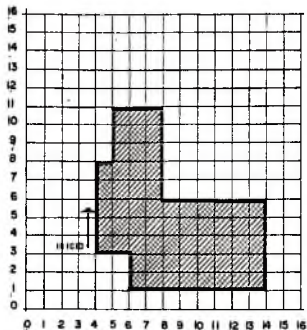
Para representar en la memoria de una computadora una imagen bidimensional, hay muchos métodos bien conocidos que van desde la representación matricial de la imagen, hasta las elegantes estructuras creadas por Sammet (Ref 1) conocidas como "Quad-Trees".

A continuación se describe cómo son y la forma en que se representan imágenes utilizando cadenas de Freeman.

1.1 DESCRIPCION DE LAS CADENAS DE FREEMAN

Partamos suponiendo una imagen bidimensional representada usando pixels; ese decir, sobre una cuadrícula se dibuja una imagen llenando de negro los cuadros que pertenecen a la imagen, y dejando en blanco los que no, (suponemos imágenes en blanco y negro). A cada uno de los cuadros que componen la cuadrícula se les llama "pixel". Por simplicidad diremos que una imagen es el conjunto de pixels negros sobre la cuadrícula que le corresponda. En el inciso a) de la fig. 1.1 se muestra una imagen bidimensional.





**NOTA:**

La referencia coordenada de un pixel, se hace a su vértice inferior izquierdo

**a) IMAGEN BIDIMENSIONAL CONEXA**

**Coordenada de referencia: (4,3)**

**Sucesión:**

**NNNNNENNNEEESSSSSEEEEEESSSSS  
OOOOOOOONNOO**

**DONDE:**

**N = Dirección Norte  
S = " Sur  
E = " Este  
O = " Oeste**



**b) CADENA DE FREEMAN ASOCIADA CON LA IMAGEN DE a)**

**Fig. 1.1 CADENAS DE FREEMAN**

Digamos formalmente el tipo de imágenes sobre las que se manejan cadenas de Freeman.

#### DEFINICION 1.1

Sea  $I$  una imagen bidimensional representada mediante pixels.

1. Una Curva sobre  $I$ , es una sucesión finita  $C_1 = c_1, \dots, c_n$ , donde cada  $c_i$  es un pixel en la imagen  $I$ .
2. Se dice que un pixel  $p_1$  es vecino ó adyacente de un pixel  $p_2$ , si  $p_1$  es distinto de  $p_2$  y tienen una arista en común.
3. Una curva  $C_1 = c_1, \dots, c_n$  es continua, si para todo  $i$  entre 1 y  $n-1$  el pixel  $c_i$  es vecino al pixel  $c_{i+1}$ .
4. Una imagen bidimensional es conexa, si para cualesquier dos pixels  $p_1$  y  $p_2$  en  $I$ , existe una curva continua  $C_1 = c_1, \dots, c_n$ , sobre  $I$  tal que  $p_1 = c_1$  y  $p_2 = c_n$ .

//

En lo que sigue, siempre supondremos imágenes conexas.

Las cadenas de Freeman guardan el contorno de la imagen. Lo más simple es cuando la imagen está formada por un pixel en este caso sus aristas conforman la cadena. Es conveniente dar una nomenclatura apropiada a cada arista sobre el pixel. Los nombres asociados se muestran en la figura 1.2

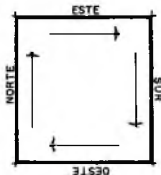


FIG. 1.2 NOMENCLATURA DE ARISTAS EN UN PIXEL.

Estos nombres corresponden a la dirección en que recorre cada arista un caminante que da vueltas alrededor del pixel en sentido de las manecillas del reloj.

Por convención, aceptemos la abreviaturas:

N = Norte.

E = Este.  
O = Oeste.

Cada pixel tiene coordenadas sobre la cuadrícula; con éstas y la secuencia de aristas N, E, S, O (en este orden), se puede ubicar un pixel en el espacio y describir su contorno. Esto es precisamente la cadena de Freeman de un pixel.

Generalizando, a cada pixel sobre la imagen se le identifica por una pareja  $(i, j)$  que corresponde a las coordenadas que ocupa sobre la cuadrícula. La representación en cadenas de Freeman de una imagen conexa  $I$ , consiste en tomar algún pixel que tenga una de sus aristas sobre la frontera de  $I$  (es decir, una arista que pertenece sólo a un pixel en  $I$ ), anotar sus coordenadas  $(x_0, y_0)$ , y construir una sucesión  $f_1, f_2, \dots, f_n$ , donde  $n$  es el número de aristas sobre la frontera de  $I$ . Cada  $f_i$  se asocia con estas aristas y toma valores dentro del conjunto  $\{N, E, S, O\}$ .

Regresando a la imagen de la figura 1.2, y suponiendo que el pixel mostrado ocupa la posición  $(1,1)$ , entonces la cadena de Freeman asociada es:

Coordenada de referencia =  $(1,1)$ .  
Sucesión (cadena):  $f_1 = N, f_2 = E, f_3 = S$  y  $f_4 = O$ .

## 1.2 CONSTRUCCION DE LA CADENA DE FREEMAN DE DOS PIXELS ADYACENTES

Cuando  $I$  tiene dos pixels de la forma en que se muestra en la fig. 1.3 y suponiendo que se han anotado las coordenadas del pixel de la izquierda y son  $(x_0, y_0)$  se tiene la cadena de Freeman siguiente:

Coordenada de referencia =  $(x_0, y_0)$ .  
Sucesión (Cadena):  $N, E, E, S, O, O$ .

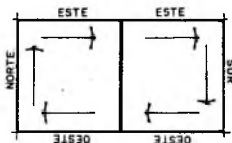


Fig. 1.3 FUSION DE LAS CADENAS DE FREEMAN DE DOS PIXELS ADYACENTES.

Obsérvese que las aristas de adyacencia se anulan y no intervienen en la cadena, sólo aparecen las que están sobre la frontera.

Del esquema anterior es claro cómo para una imagen complicada, se puede generar su cadena de Freeman.

En la figura 1.1, se muestra una imagen bidimensional conexa, y su cadena de Freeman.

### 1.3 COMPRESION DE CADENAS DE FREEMAN

Veamos ahora cómo hacer compresiones a las cadenas de Freeman para almacenarlas en menos espacio. Como ejemplo, en la figura 1.1 aparece 5 veces seguidas la arista N, en vez de poner 5 N's se pondría un factor de repetición de 5 y luego una sola N.

Un ejemplo más complicado aparece en la fig. 1.4 utilizando notación de listas para representar la sucesión de caras (que esperamos sea evidente). Se tendría la cadena:

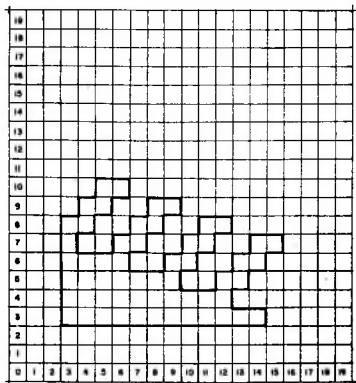
Coordenada de Referencia: (3,4)

Sucesión:

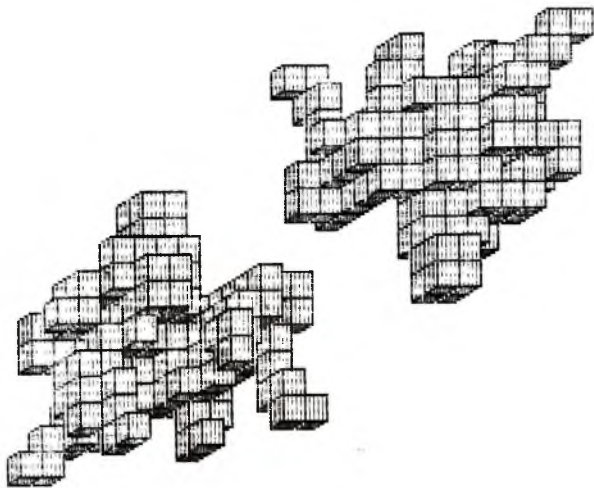
5(N) 4(3(N E) E 3(S O) S 2(E)) S 2(E)) S 12(O)

La cantidad de aplicaciones que han tenido las cadenas de Freeman, principalmente para almacenar de manera barata imágenes bidimensionales, es una prueba de su utilidad. Existen actualmente algoritmos para la transformación de cadenas de Freeman a otras representaciones.

El problema fundamental de este trabajo es el de hacer la generalización de las cadenas de Freeman a 3 dimensiones.



**Fig. 1.4 IMAGEN PARA EJEMPLIFICAR LA COMPRESION DE CODIGO.**



## CAPITULO II

### DEFINICIONES BASICAS

En este capítulo, nos concentraremos en la definición rigurosa del problema, así como de los elementos fundamentales que aparecen a lo largo del trabajo. Como se mencionó en el Capítulo I, se trata de generalizar el concepto de Cadena de Freeman a tres dimensiones. Las imágenes que se manejan, están formadas por pixels tridimensionales e inicialmente se manejan dentro de una representación matricial (ver definición 2.5). El mismo concepto de pixel se puede definir en tres dimensiones. Veamos esto detalladamente:

#### 2.1 CONCEPTOS ELEMENTALES

##### DEFINICION 2.1

Un rectángulo tridimensional es un conjunto de la forma  $[a_1, b_1] \times [a_2, b_2] \times [a_3, b_3]$  donde  $a_i, b_i$  son números reales para  $i = 1, 2, 3$ .

//

##### DEFINICION 2.2

Un conjunto pavimentable en  $R^3$  es un subconjunto de  $R^3$  que se puede expresar como unión finita de rectángulos tridimensionales.

//

##### DEFINICION 2.3

Un Sólido Partido Regularmente, es un conjunto pavimentable en  $R^3$  expresado como una unión de rectángulos que se intersectan por parejas en a lo sumo una cara; además, cada rectángulo es un cubo de arista 1.

A cualquier rectángulo que forme parte de un sólido partido regularmente le llamaremos cubo unitario.

//

##### NOTA 2.1

Los cubos unitarios se consideran cerrados en el sentido topológico.

Los sólidos que trataremos a lo largo de este trabajo son al igual que en el caso bidimensional de "una sola pieza", a continuación explicaremos bajo que topología se define esta idea de conexidad para sólidos partidos regularmente.

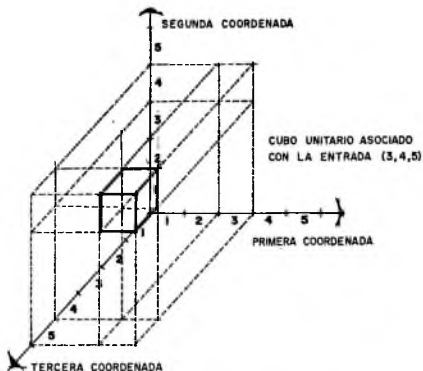
**DEFINICION 2.4 (Sólido conexo).**

Sea  $S$  un sólido partido regularmente. Se dice que  $S$  es conexo si para cualesquier dos puntos  $A$  y  $B$  en  $S$ , existe una curva continua  $C$  que conecta ambos puntos, de manera que  $C$  pasa por sólo puntos en el interior topológico de  $S$ . (Aquí tomamos la continuidad usual de curvas en el espacio).

//

Por ejemplo, el sólido mostrado en el inciso b) de la figura 2.2 no es conexo.

EN EL RESTO DEL TRABAJO, SIEMPRE QUE SE HABLE DE UN SOLIDO, IMPLICITAMENTE DEBE ENTENDERSE SOLIDO PARTIDO REGULARMENTE Y CONEXO.



**Fig. 2.1 MARCO COORDENADO PARA LA REPRESENTACION DE SOLIDOS**



## 2.2 MATRIZ DE REPRESENTACION DE SOLIDOS

Como se mencionó en la primer parte de este capítulo, el algoritmo para determinar la gráfica de adyacencias, parte de la representación matricial del sólido. Aunque suponemos que le es familiar al lector, es conveniente definirla rigurosamente.

DEFINICION 2.5 (Matriz de Representación de Sólidos.).

La Matriz de Representación de Sólidos, sirve para representar Sólidos Partidos Regularmente. Se supone que el sólido se encuentra en el cuadrante positivo de un sistema coordenado X,Y,Z como el que se muestra en la fig. 2.1 Una matriz  $A_{m \times n \times k}$ , de este tipo consta de ceros y unos, con m, n y k enteros positivos, de manera que m es el número máximo de cubos en la dirección X, n en la dirección Y y k en la dirección Z.

//

La representación de un cubo unitario usando el método de la definición anterior se ilustra en la fig. 2.1

Si A es una matriz de representación de sólidos y  $A(i,j,k) = 0$  entenderemos que el cubo asociado con la coordenada (i,j,k) no pertenece al sólido; si  $A(i,j,k) = 1$  sí pertenece.

## 2.3 EL CONCEPTO DE RECORRIDO

Las cadenas de Freeman pueden considerarse como recorridos sobre la frontera de la imagen que describen. El algoritmo para encontrarlas es sencillo: Dada una imagen I, se "detiene un caminante" a la orilla de la imagen, escoge una dirección de movimiento y la sigue sin apartarse de la orilla. La sucesión que representa a la cadena de Freeman de I no es más que la anotación ordenada de las direcciones en que se movió el caminante, se termina cuando éste llega al punto de partida. La generalización de cadena de Freeman a tres dimensiones se puede pensar como si para un sólido S se pintara una línea continua que pase por todas las caras de su superficie, y no pase dos veces por una misma, a no ser que la curva regrese al punto de partida cerrando un ciclo. En este caso el algoritmo no es tan simple como en el caso bidimensional.

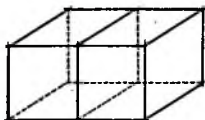
Formalicemos el concepto de recorrido sobre un sólido. Dicho en otras palabras definamos el concepto de "Cadena de Freeman en tres dimensiones". Para esto veamos algunas definiciones preliminares.

DEFINICION 2.6

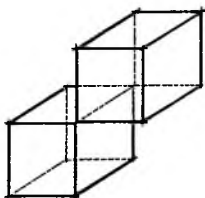
Sea S un sólido partido regularmente conexo.

- 1.- Sean  $C_1$  y  $C_2$  dos cubos unitarios de  $S$ . Se dice que  $C_1$  es adyacente a  $C_2$  si  $C_1$  y  $C_2$  tienen una cara en común. (Usaremos "vecino" como sinónimo de adyacente).
- 2.- Si una cara  $C$  en un cubo unitario de  $S$  está contenida en la frontera de  $S$ , se dice que  $C$  es una cara externa en  $S$ .
- 3.- Sean  $A$  y  $B$  dos caras externas en  $S$ . Se dice que  $A$  es adyacente a  $B$  si  $A$  y  $B$  tienen exactamente una arista en común.

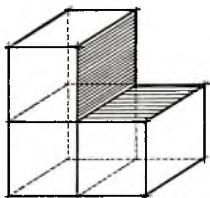
En la fig. 2.2 se muestran ejemplos de esta definición.



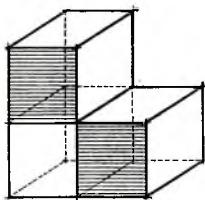
a) CUBOS ADYACENTES



b) CUBOS NO ADYACENTES



c) CARAS ADYACENTES



d) CARAS NO ADYACENTES

**Fig. 2.2 EJEMPLOS DE ADYACENCIA**

Y ahora la definición de recorrido.

**DEFINICION 2.7 (Recorrido sobre las caras de un sólido).**

Sea  $S$  un sólido partido regularmente. Un Recorrido sobre  $S$ , es una pareja ordenada de la forma  $(C,H)$ , donde  $C$  son las coordenadas de algún cubo unitario en  $S$  que tiene una de sus caras en la frontera de  $S$ , y  $H$  es una sucesión de la forma  $h_0, h_1, \dots, h_{n-1}$ , en la que  $n$  es igual al número de caras externas de  $S$ , y cada  $h_i$  se asocia a una y sólo una de tales caras;  $C$  son las coordenadas del cubo que contiene a  $h_0$ . Además para toda  $j$  entre  $1$  y  $n-1$  la cara  $h_{j-1}$  es adyacente a  $h_j$ . Si  $h_0$  es adyacente a  $h_{n-1}$ , entonces se dice que es un Recorrido Cíclico.

A la sucesión  $H$  de un recorrido, le llamaremos Sucesión Hamiltoniana del Recorrido.

//

En este trabajo siempre usaremos recorridos cíclicos. Obsérvese que con cambiar la cara inicial de un recorrido, se obtiene otro recorrido, aún cuando cíclicamente sean iguales. Para evitar ambigüedades se introduce la definición siguiente:

**DEFINICION 2.8**

1. Sean  $H$  y  $K$  dos sucesiones hamiltonianas, con  $H = h_0, h_1, \dots, h_{n-1}$ ;  $K = k_0, k_1, \dots, k_{n-1}$ . Se dice que  $H$  es cíclicamente igual a  $K$  si existe  $i$  entre  $0$  y  $n-1$  tal que  $h_j = k_{j+i \pmod n}$  para toda  $j$  entre  $0$  y  $n-1$ .
2. Sean  $R_1 = (C_1, H_1)$  y  $R_2 = (C_2, H_2)$  dos recorridos para un mismo sólido. Se dice que  $R_1$  es igual cíclicamente a  $R_2$  si  $H_1$  es igual cíclicamente a  $H_2$ .

//

Por el momento dejaremos pendiente la forma en que se representa cada  $h_i$ . Para ilustrar un recorrido, en la fig. 2.3 se ha dibujado una proyección paralela de cierto sólido, aunque en esta representación se usa una idea un tanto exótica, pues por un lado de la hoja se muestra una vista del sólido y en el otro lado, el sólido visto "de espaldas". Sin embargo, el lector encontrará muy cómoda esta gráfica cuando descubra que así es muy simple dibujar un recorrido sobre el sólido representado.

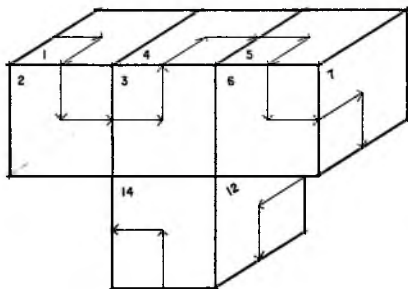
Sobre las caras del sólido de la fig. 2.3 se han escrito números de manera que si el lector los sigue consecutivamente partiendo de  $1$  (a veces tendrá que cambiar de lado de hoja), hallará una secuencia de caras que corresponde a una sucesión hamiltoniana y como empieza donde termina se trata de una secuencia hamiltoniana asociada con un recorrido cíclico. En el inciso b) de la fig. 2.3, se muestra este recorrido. Observe que

en este caso las caras externas se representan por un número.

Las flechas que aparecen sobre las caras externas del sólido de la fig. 2.3, sirven para marcar la secuencia de caras consecutivas.

#### 2.4 OBSERVACIONES SOBRE EL PLANTEAMIENTO DEL PROBLEMA

Como puede verse, el problema fundamental es cómo encontrar un recorrido sobre la frontera de un sólido  $S$ , este problema no se resuelve totalmente, pero se dan un conjunto de algoritmos muy eficientes para encontrar recorridos sobre cierto tipo de sólidos. Un detalle importante es que hasta ahora no hemos hallado un sólido para el cual no se pueda encontrar un recorrido haciendo uso de estos algoritmos.

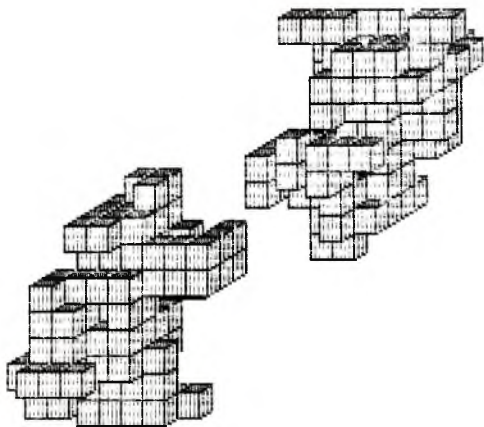


a) SOLIDO CON REPRESENTACION GRAFICA  
DE UN RECORRIDO

$$H = 1, 2, \dots, 18$$

b) SUCESION HAMILTONIANA DEL RECORRIDO

**Fig. 2.3 RECORRIDO CICLICO**

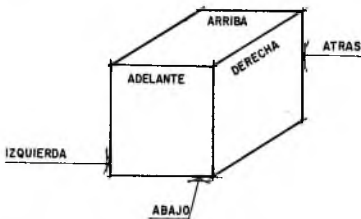


## GRAFICA DE ADYACENCIA DE CARAS DE UN SOLIDO

El problema enunciado en el Capítulo II, puede plantearse considerando los sólidos a recorrer como gráficas, donde los vértices corresponden a las caras externas del sólido y los arcos indican las que son adyacentes. En este capítulo definiremos formalmente el concepto de gráfica de adyacencia de caras de un sólido, y la forma en que es posible generarla partiendo de una representación matricial. Como se verá más adelante, esta gráfica juega un papel muy importante en la construcción del recorrido sobre el sólido; de hecho, el mayor esfuerzo de los algoritmos referentes al recorrido, se dedica a la construcción de la gráfica mencionada. El problema del recorrido en estos términos, equivale a encontrar un camino hamiltoniano sobre la gráfica del sólido.

### 3.1 DEFINICIONES BASICAS RELACIONADAS CON LA GRAFICA DE ADYACENCIA DE CARAS

Estudiemos sobre un cubo unitario como es su gráfica de adyacencias. Es conveniente introducir una notación especial, para identificar sus caras. Esta se muestra en la fig. 3.1.



**Fig. 3.1 CUBO UNITARIO**  
Nomenclatura de Caras.

Introduzcamos una manera simple de designar caras que quedan frente a frente en un cubo.

### DEFINICION 3.1

Se define la función CONTRARIA : CARAS  $\rightarrow$  CARAS, donde  
CARAS := {DERECHA, ARRIBA, ADELANTE, IZQUIERDA, ABAJO, ATRAS} como:

```
CONTRARIA(DERECHA) := IZQUIERDA
CONTRARIA(IZQUIERDA) := DERECHA
CONTRARIA(ARRIBA) := ABAJO
CONTRARIA(ABAJO) := ARRIBA
CONTRARIA(ADELANTE) := ATRAS
CONTRARIA(ATRAS) := ADELANTE
```

La nomenclatura de caras de la fig. 3.1 se puede heredar a las caras externas de sólidos en general de la manera siguiente: Si  $c$  es una cara externa en un sólido  $S$ , entonces existe un cubo  $CU$  único tal que  $c$  pertenece a  $CU$ , de esta manera a  $c$  le asociamos su nombre dentro de  $CU$ . A este nombre le llamaremos orientación de  $c$ .

En adelante supondremos que los valores para las orientaciones de caras externas están ordenados y que la función "Ord" (de Pascal) está definida para estas orientaciones, de manera que  $\text{ord}(\text{DERECHA})=0$ ,  $\text{ord}(\text{ARRIBA})=1$ ,  $\text{ord}(\text{ADELANTE})=2$ ,  $\text{ord}(\text{IZQUIERDA})=3$ ,  $\text{ord}(\text{ABAJO})=4$  y  $\text{ord}(\text{ATRAS})=5$ . El orden es el inducido por la función ord.

Hay dos razones para escoger este orden y aunque no se explican, se dice donde se usan:

1. El ordinal módulo 3 de caras paralelas coincide:

```
ord(DERECHA) mod 3 = ord(IZQUIERDA) mod 3
ord(ARRIBA) mod 3 = ord(ABAJO) mod 3
ord(ADELANTE) mod 3 = ord(ATRAS) mod 3
```

Ver paso 2 del algoritmo 3.1.

2. La secuencia DERECHA, ARRIBA, ADELANTE, IZQUIERDA, ABAJO, ATRAS, corresponde a una sucesión hamiltoniana para un cubo unitario. Ver definición 4.1.

Pasemos ahora a la gráfica de adyacencia de caras de un cubo unitario.

### DEFINICION 3.2

Sea  $C$  un cubo unitario, Se define la gráfica de adyacencia



de caras de  $C$ , como una gráfica 6 no ordenada de la forma  $(V,A)$  donde:

$V = \{Derecha, Adelante, Arriba, Izquierda, Atras, Abajo\}$  y  
 $A = \{(Derecha, Adelante), (Derecha, Arriba), (Derecha, Atras), (Derecha, Abajo), (Izquierda, Adelante), (Izquierda, Arriba), (Izquierda, Atras), (Izquierda, Abajo), (Adelante, Arriba), (Arriba, Atras), (Atras, Abajo), (Abajo, Adelante)\}$

//

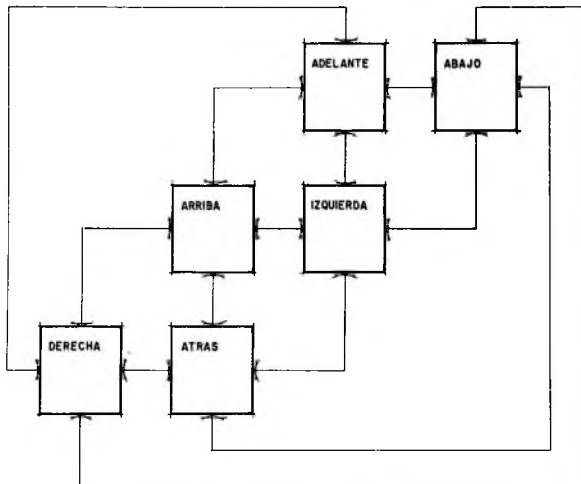


Fig. 3.2 **CUBO UNITARIO**. Gráfica de Conectividad.

La figura 3.2 ilustra la gráfica de adyacencia de caras de un cubo unitario. Obsérvese que los vértices corresponden a las caras del cubo y los arcos a las adyacencias de éstas.

Generalicemos el concepto de gráfica de adyacencias.

DEFINICION 3.3 (Gráfica de Adyacencia de un Sólido).

Sea  $S$  un sólido partido regularmente conexo. La Gráfica de Adyacencia de Caras de  $S$ , es una gráfica de la forma  $G_S = (V,A)$ , donde  $V = \{c \mid c \text{ es una cara externa en } S\}$  y  $A$  es el subconjunto del producto cartesiano  $V \times V$  tal que si la pareja  $(c_1, c_2)$  pertenece a  $A$ , entonces  $c_1$  es adyacente a  $c_2$ , y viceversa.

//

### 3.2 ESTRUCTURAS DE DATOS (USANDO PASCAL)

Para representar al sólido, se utiliza la matriz de representación de sólidos mencionada en la sección 2.2.

Los valores que toman las orientaciones de caras están dadas por el tipo:

```
type Orientacion = (Derecha, Arriba, Adelante, Izquierda,
                    Abajo, Atras);
```

El conjunto CARAS de la definición 3.1, tiene su dominio sobre este tipo.

Obsérvese que de esta manera se satisface el orden dado para las orientaciones.

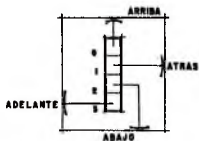
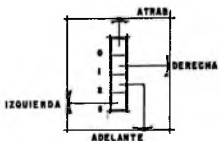
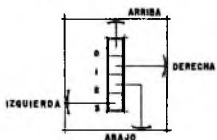
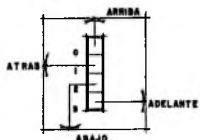
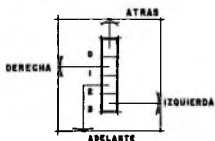
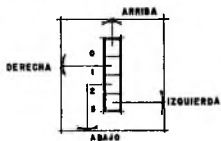
Pasemos a la representación de una gráfica de adyacencia de caras.

Sea  $G = (V,A)$  la gráfica de adyacencia de caras de algún sólido  $S$ . Cada nodo  $v$  en  $V$ , será representado por una estructura con tipo:

```
TCara = record
        Liga : Array[0..3] of ApCara;
        Tipo : Orientacion
    end;
ApCara = ^TCara;
```

Recordemos que  $v$  representa una cara externa en  $S$ , por lo tanto en esta estructura la variable Tipo debe indicar la orientación de la cara asociada a  $v$ ; el vector Liga debe contener cuatro apuntadores, de manera que para toda  $i$  entre 0 y 3,  $Liga[i]^{\wedge}$  sea la representación de un nodo  $w_i$  en  $V$  tal que la pareja  $(v, w_i)$  está en  $A$  y recíprocamente para toda pareja  $(v, w)$  en  $A$  exista un índice  $j$  tal que  $Liga[j]^{\wedge}$  corresponde a  $w$ .

El algoritmo para hallar la gráfica de adyacencia de caras de  $S$ , realiza las asignaciones necesarias para que lo dicho en el párrafo anterior se cumpla en  $S$ . De esta manera el algoritmo devuelve un apuntador a la representación de algún nodo en  $V$  y las coordenadas del cubo unitario al que pertenece ese nodo.

**DERECHA****ARRIBA****ADELANTE****IZQUIERDA****ABAJO****ATRAS**

En el vector correspondiente a cada cara, describimos a las caras que son vecinas a ésta. Cada componente  $ind_j$  da el tipo de la cara vecina según se muestra en las figuras

**Fig. 3.3 REPRESENTACION DE LA GRAFICA UNITARIA.**

Si pensamos en la gráfica de un cubo unitario, hay muchas maneras de hacer asignaciones en el vector Liga de las variables que representan sus caras. La figura 3.3 muestra una de estas formas. Por las razones que se dan en la sección 3.4 siempre representaremos las gráficas de cubos unitarios, como se muestra en la fig. 3.3.

En el resto del capítulo, haremos un abuso de notación y usaremos las estructuras aquí definidas como parte del metalenguaje, olvidando que pertenecen a Pascal pero conservando su mismo significado.

### 3.3 GRAFICA DE ADYACENCIAS DE UN CUBO UNITARIO

La gráfica de adyacencia de caras de un cubo unitario se puede manejar guardando un apuntador a una de sus caras y de ahí mediante un proceso de manipulación de apuntadores tener acceso a sus demás caras, sin embargo navegar siguiendo apuntadores es un proceso costoso. Por esta razón se incluye una estructura llamada TCubo que sirve para representar cubos unitarios.

```
Type TCubo = Array[Derecha..Atras] of ApCara;
```

Cada entrada en una variable de tipo TCubo representa una cara del cubo unitario. Se introduce esta estructura para que las asignaciones mostradas en la fig. 3.3 se hagan de una manera simple; por ejemplo, suponiendo la declaración:

```
var CU : TCubo
```

y que se ha ejecutado la instrucción:

```
for i := Derecha to Atras do new(CU[i])
```

se procede a establecer las conexiones propias de la gráfica de adyacencias del cubo unitario. Así, para la cara DERECHA de CU, se hacen las asignaciones:

```
CU[Derecha].Tipo := Derecha;  
CU[Derecha].Liga[0] := CU[Arriba];  
CU[Derecha].Liga[1] := CU[Atras];  
CU[Derecha].Liga[2] := CU[Abajo];  
CU[Derecha].Liga[3] := CU[Adelante];
```

Asignaciones semejantes se deben hacer para las demás caras.

Una vez hechas todas las asignaciones se guarda sólo un apuntador a alguna de las caras de CU y CU se desecha.

### 3.4 GRAFICA DE ADYACENCIA DE DOS CUBOS ADYACENTES

En el caso bidimensional vimos que la fusión de las cadenas de dos pixels adyacentes se lleva a cabo formando una sola cadena a partir de las de los pixels de la manera que se ilustra en la figura 1.4. Algo muy parecido se hace cuando queremos construir la gráfica de adyacencia de caras de un sólido formado por dos cubos unitarios adyacentes; basta eliminar las caras de contacto y reestablecer las ligas perdidas apropiadamente. Esto se muestra de manera gráfica en la figura 3.4, donde aparece en el inciso a) un sólido que consta de dos cubos unitarios adyacentes. Se muestra en el inciso b) la gráfica de adyacencia de caras de cada uno de ellos. En el inciso c) se han eliminado los nodos correspondientes a las caras de contacto y aparece la gráfica de adyacencia de caras del sólido del inciso a).

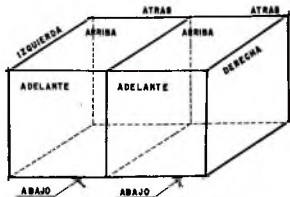
Veamos cómo se puede hacer este proceso en base a la gráfica de dos cubos unitarios.

Supongamos que las variables CUBO\_I y CUBO\_D han sido inicializadas apropiadamente y representan la gráfica de adyacencia de dos cubos adyacentes tales que la cara derecha de CUBO\_I hace contacto con la izquierda de CUBO\_D; y las variables AP\_D y AP\_I apuntadores respectivos a estas caras (ver fig. 3.5.a).

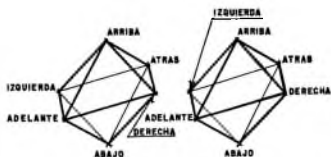
En la figura 3.5.b sobre el cubo CUBO\_I se ha sombreado la cara correspondiente a AP\_D.LIGA[0], mientras que en CUBO\_D, se ha sombreado la cara AP\_I.LIGA[0], llamemos a estas caras  $I_0$  y  $D_0$  respectivamente. Si  $G = (V,A)$  es la gráfica de adyacencia de todo el sólido de la fig. 3.5.a y los elementos  $v_0, w_0$  de  $V$  corresponden a las caras  $I_0$  y  $D_0$ , entonces la pareja  $(v_0, w_0)$  debe estar en  $A$ . Por esta razón el terminarse la eliminación de las caras de contacto, la cara  $I_0$  debe contener en su vector Liga una entrada para el apuntador  $D_0$  y viceversa. Como la entrada en Liga de  $I_0$  a la cara AP\_D no se necesita una vez eliminada AP\_D entonces en esta se puede guardar el apuntador a  $D_0$  con lo que queda establecida la primer liga, de manera análoga se procede con la otra.

Obsérvese que todo lo que se dijo en el párrafo anterior es cierto si se sustituye el índice y subíndice 0 por 1 y las referencias a la fig. 3.5.b se sustituyen por referencias a la 3.5.c. Lo mismo pasa si se sustituye 0 por 2 o 3 (imagine el lector las figuras). Esto es una consecuencia de la asignación de valores a los vectores Liga en un cubo unitario; es decir, la asignación mostrada en la figura 3.3.

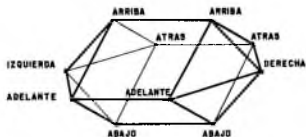
Si  $S$  es un sólido formado por sólo dos cubos unitarios adyacentes y  $G_s = (V,A)$  su gráfica de adyacencia, entonces si  $X$  e  $Y$  apuntan a las caras a fusionarse, las ligas han de establecerse coordenada a coordenada dentro de sus vectores Liga; ya que las parejas de caras en  $V$  asociadas con  $(X.Liga[i]^n, Y.Liga[i])$  son aristas en  $G_s$ .



a) Sólido formado por cubos unitarios ayacentes

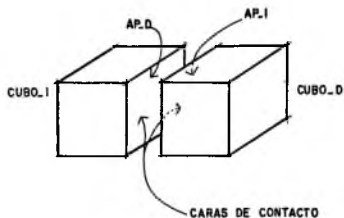


b) Gráficas de adyacencia de caras individuales.



c) Gráfica de adyacencia de caras del sólido del inciso a)

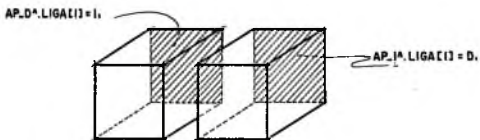
**Fig. 3.4 PROCESO DE FUSION DE LAS GRAFICAS DE ADYACENCIA DE CARAS DE DOS CUBOS UNITARIOS.**



a) DOS CUBOS ADYACENTES



b) LAS CARAS I\_0^A Y D\_0^A DEBEN LIGARSE ENTRE SI AL ELIMINAR LAS CARAS DE CONTACTO.



c) IDEM. b) PERO CON I\_1 Y D\_1

FIG. 3.5 RESTABLECIMIENTO DE LIGAS.

Lo anterior se usa en el paso 3 de algoritmo 3.1.

Pasemos al algoritmo de eliminación o fusión de caras:

#### ALGORITMO 3.1

1. [Fase Inicial]. Sean C1 y C2 sendos apuntadores a las caras a eliminarse.
2. [Verificación de que C1 y C2 se pueden eliminar].  
  
SI  $\text{Ord}(C1^{\wedge}.\text{TIPO}) = (\text{Ord}(C2^{\wedge}.\text{TIPO}) + 3) \bmod 6$   
ENTONCES la eliminación es posible y sigue el paso 3. De lo contrario termina.

Esta revisión es necesaria, pues sólo en pares de caras con tuplas respectivas (IZQUIERDA,DERECHA), (ARRIBA,ABAJO) ó (ADELANTE,ATRAS) es posible la eliminación (o fusión) de ambas caras.

3. [Establecimiento de las ligas que inciden en C1 a las caras del cubo al que pertenece C2].  
  
PARA  $i := 0$  HASTA 3 HACER  $\langle i$  representa entradas en LIGA de  $C1^{\wedge}$  $\rangle$ .  
  
CTemp :=  $C1^{\wedge}.\text{LIGA}[i]$ ;  
  
Encontrar en CTemp $^{\wedge}$  el índice j tal que  
CTemp $^{\wedge}.\text{LIGA}[j] = C1$  y hacer  
CTemp $^{\wedge}.\text{LIGA}[j] := C2^{\wedge}.\text{LIGA}[i]$ .
4. Repetir 3 intercambiando C1 y C2.
5. [FIN] Termina.

//

Como puede verse, la construcción de la gráfica de adyacencia de caras de un sólido, se puede hacer cubo por cubo eliminando las caras que se peguen. Esta es la idea central del algoritmo para construir tal gráfica, por el momento aún no hemos fijado la manera de recorrer los cubos en el sólido. En la siguiente sección se presenta cómo hacer esto.

### 3.5 GRAFICA DE ADYACENCIAS DE UN SOLIDO GENERAL

Empezaremos esta sección, indicando una manera ordenada de ir aplicando el algoritmo de eliminación de caras cubo por cubo a todos los cubos unitarios que forman un sólido, y de esta manera encontrar su gráfica de adyacencia.



Dado un sólido S, demos una enumeración de sus cubos unitarios de la forma  $C_1, \dots, C_n$ . Una forma de construir la gráfica de adyacencia de caras de S, es tomar el cubo  $C_1$  y construir su gráfica; después tomar el cubo  $C_2$ , construir también su gráfica, y si alguna de las caras de este último se pega con las de  $C_1$  entonces se les aplica el algoritmo de fusión de caras; una vez hecho esto, se procede a tomar el cubo  $C_3$  y repetimos el proceso. Obsérvese que pudiera ocurrir que haya que eliminar más de una cara para incorporar  $C_3$  a la gráfica construida. De esta forma se procesa cada cubo en S, para obtener finalmente la gráfica de adyacencias requerida. La idea es sencilla pero es difícil llevarla a la práctica, ya que es necesario "recordar" cuál es la posición actual de las caras de los cubos procesados que pueden eliminarse, lo cual la hace costosa.

En esencia, la idea final del algoritmo es la del párrafo anterior, aunque se minimiza la cantidad de memoria requerida escogiendo una enumeración ordenada de sus cubos. Esta se explica a continuación:

### DEFINICION 3.5 (Enumeración E).

La Enumeración E de Cubos Unitarios en un sólido S, denotado por  $E_s$ , toma la representación matricial A del sólido, y recorre los cubos en el orden lexicográfico de índices, donde varían más rápidamente los de la izquierda es decir:

$$\begin{aligned} &A(1,1,1) < A(2,1,1) < \dots < A(N,1,1) < \\ &A(1,2,1) < A(2,2,1) < \dots < A(N,2,1) < \\ &\dots \dots \dots < \end{aligned}$$

$$\begin{aligned} &A(1,N,1) < A(2,N,1) < \dots < A(N,N,1) < \\ &A(N,N,1) < A(N,N,2) < \dots < A(N,N,N) \end{aligned}$$

(N es el máximo índice por entrada en A).

Evidentemente, sólo habrán de considerarse los cubos correspondientes a entradas  $A(i,j,k)$  no nulas.

De esta manera,  $E_s$  es una función

$$E_s : \{m : m \text{ es un número natural entre } 1 \text{ y el número de cubos unitarios en el sólido}\} \rightarrow \{c : c \text{ es un cubo unitario de S}\}$$

$E_s(i) = c \iff c$  es el  $i$ -ésimo cubo en S según el orden definido arriba que corresponde a una entrada en A no nula. //

Veamos una definición auxiliar que nos será útil para estudiar algunas propiedades de la enumeración E.

### DEFINICION 3.6

Dado un sólido  $S$ , se definen las funciones siguientes:

DERECHAS  
ARRIBAS  
ADELANTES  
IZQUIERDAS  
ABAJOs  
ATRASS

(La "s" pequeña que parece en los nombres de las funciones en general se sustituye por el nombre que se le dé al sólido).

Cada una de estas funciones tiene como dominio los naturales no cero y como imagen al conjunto  $\{c \mid c \text{ es un cubo unitario de } S\}$ .

Para la función DERECHAS definimos:

DERECHAS(i) =  $c_0$  si y sólo si  $c_0$  es el  $i$ -ésimo cubo, según la enumeración  $E_s$ , que tiene vecino a la derecha (siempre que esto tenga sentido).

A las demás funciones se les define de manera análoga. //

Veamos un par de propiedades interesantes sobre la enumeración  $E_s$ .

### LEMA 3.1

Sea  $S$  un sólido partido regularmente conexo y  $F$  alguna de las funciones DERECHAS, ADELANTES, ARRIBAS, IZQUIERDAS, ATRASS ó ABAJOs. Entonces para cualesquier par de cubos  $c_1$  y  $c_2$ , a los que se les pueda aplicar la función inversa de  $F$ , se cumple  $F^{-1}(c_1) < F^{-1}(c_2)$  si y sólo si  $E_s^{-1}(c_1) < E_s^{-1}(c_2)$ .

### DEMOSTRACION

Puede verse fácilmente que una tal función  $F$  preserva el orden de la enumeración  $E_s$ .  
! !

### LEMA 3.2

Sean  $S$  un sólido partido regularmente conexo,  $A$  la representación matricial de  $S$  y  $c_1, c_2, v_1$  y  $v_2$ , cuatro cubos unitarios en  $S$  tales que  $v_1$  y  $v_2$  son vecinos a  $c_1$  y  $c_2$  respectivamente. Si  $E_s^{-1}(c_1) < E_s^{-1}(c_2)$ , entonces dada alguna de las condiciones:

1.  $v_1$  y  $v_2$  son vecinos izquierdos de  $c_1$  y  $c_2$  respectivamente.
2.  $v_1$  y  $v_2$  son vecinos traseros de  $c_1$  y  $c_2$  respectivamente.
3.  $v_1$  y  $v_2$  son vecinos inferiores de  $c_1$  y  $c_2$  respectivamente.

se tendría que  $E_{\bullet}^{-1}(v_1) < E_{\bullet}^{-1}(v_2)$ .

**DEMOSTRACION:**

Sean  $A(i_1, j_1, k_1)$  y  $A(i_2, j_2, k_2)$  las entradas en  $A$  correspondientes a los cubos unitarios  $c_1$  y  $c_2$  respectivamente. Entonces por hipótesis  $A(i_1, j_1, k_1) < A(i_2, j_2, k_2)$  según el orden lexicográfico definido.

Si se cumple 1 entonces las entradas de  $A$  correspondientes a  $v_1$  y  $v_2$  son respectivamente  $A(i_1-1, j_1, k_1)$  y  $A(i_2-1, j_2, k_2)$ , claramente  $A(i_1-1, j_1, k_1) < A(i_2-1, j_2, k_2)$  de donde se sigue la conclusión  $E_{\bullet}^{-1}(v_1) < E_{\bullet}^{-1}(v_2)$ .

Los incisos 2 y 3, se demuestran de manera análoga.

||

A continuación se da la propiedad por la cual tomamos la enumeración  $E_{\bullet}$ .

**PROPOSICION 3.1**

Sea  $S$  un sólido partido regularmente conexo y  $A$  su representación matricial, entonces  $DERECHAS(i) = c$  si y sólo si el vecino derecho  $c'$  de  $c$  cumple:  $IZQUIERDAS(i) = c'$ .

En otras palabras, el  $i$ -ésimo cubo  $c$  con vecino derecho, según  $E_{\bullet}$ , tiene como vecino derecho a un cubo  $c'$  tal que  $c'$  es el  $i$ -ésimo cubo, según  $E_{\bullet}$ , con vecino izquierdo.

**DEMOSTRACION**

La demostración es por inducción sobre  $i$ .

Veámoslo para  $i = 1$ .

Supongamos que  $S$  tiene un cubo  $D_1$ , tal que  $DERECHAS(1) = D_1$ , llamemos  $I_1$  a su vecino derecho; como  $I_1$  tiene vecino izquierdo, (a  $D_1$ ), hay un entero positivo  $k$  tal que  $IZQUIERDAS(k) = I_1$ . Si  $k > 1$  entonces existe  $I_1'$  tal que  $IZQUIERDAS(1) = I_1' \dot{e} I_1 < I_1'$ . Al vecino izquierdo de  $I_1'$  lo denotaremos por  $D_1'$ .

$IZQUIERDAS^{-1}(I_1') = 1 < k = IZQUIERDAS^{-1}(I_1)$ ; por el lema 3.1 se tiene que  $R^{-1}(I_1') < R^{-1}(I_1)$ , considerando que  $D_1$  y  $D_1'$  son vecinos izquierdos respectivamente de  $I_1$  e  $I_1'$ , se sigue del

lema 3.2 que  $E_{\square}^{-1}(D1') < E_{\square}^{-1}(D1)$ , es decir  $E_{\square}^{-1}(D1') < 1$ , lo cual contradice la manera en que se definió  $E_{\square}$ . Esta contradicción vino de suponer  $K > 1$ . Por lo tanto  $k = 1$  y con esto  $IZQUIERDAs(i) = I1$ .

La otra parte del "si y sólo si" se demuestra análogamente.

Suponiendo que la propiedad prevalece para  $i = n-1 > 0$ , mostremos que se cumple también para  $i = n$

Si  $S$  tiene un cubo  $Dn$  tal que  $DERECHAs(n) = Dn$ , llamemos  $In$  a su vecino derecho; como  $In$  tiene vecino izquierdo (a  $Dn$ ), entonces hay un entero positivo  $m$  tal que  $IZQUIERDAs(m) = In$ . Veamos que  $m = n$ , de donde quedará demostrada la proposición (nuevamente la otra parte del "si y sólo si" se demuestra análogamente.

Si  $m < n$ , entonces por hipótesis de inducción, como  $IZQUIERDAs(m) = In$  entonces  $DERECHAs(m) = Dn$ , lo que contradice la forma en que se eligió  $Dn$ . Imposible.

Si  $m > n$  entonces existe  $In'$  tal que  $E_{\square}(n) = In'$  y además  $In < In'$ . En este caso se llega nuevamente a una contradicción, de la misma manera que en el caso base.

Por lo tanto  $m = n$ .

||

### PROPOSICION 3.2

La proposición anterior se cumple, si cambiamos las funciones  $IZQUIERDAs$  y  $DERECHAs$  por  $ATRASs$  y  $ADELANTES$  respectivamente, también se cumple si las cambiamos por  $ABAJDs$  y  $ARRIBAs$  respectivamente.

### DEMOSTRACION

Análoga a la demostración de 3.1.

||

La utilidad de lo que dicen las proposiciones anteriores se explica a continuación.

Suponga el lector un sólido  $S$ . Tomemos los cubos de  $S$  siguiendo la enumeración  $E_{\square}$ , hasta que hallemos el cubo  $ADELANTES(1) ::= A1$ . Si continuamos la enumeración irán apareciendo  $ADELANTES(2) ::= A2$ ,  $ADELANTES(3) ::= A3$ , etc; pero lo que dicen las proposiciones es que  $ATRASs(1) ::= T1$  sólo puede aparecer después de  $A1$ , y que  $A1$  es vecino trasero de  $T1$ , también  $ATRASs(2) ::= T2$  aparece después de  $A2$  y  $A2$  es vecino trasero de  $T2$ , y así sucesivamente. Esto quiere decir, que si formamos las parejas de cubos  $(A1, T1)$ ,  $(A2, T2)$ , ... ,  $(An, Tn)$ ,

donde  $n = \max\{i \mid \text{ADELANTES}(i) \text{ est\aa definida}\}$ , entonces se pueden fusionar las caras de contacto en estas parejas de acuerdo al algoritmo 3.1, y as\i obtener todas las fusiones que involucren caras de tipo ADELANTE, ATRAS. Si lo que se ha dicho para (ADELANTES, ATRAS) se aplica a la tupla (DERECHAS, IZQUIERDAS), se pueden obtener las fusiones que involucren caras de tipo izquierda, derecha y lo mismo vale para (ARRIBAS, ABAJOS).

En realidad, no es necesario almacenar las parejas de cubos, ni siquiera los cubos completos sino s\o lo apuntadores a las caras de contacto. Esto se explica detalladamente en la siguiente secci\on.

### 3.6 ALGORITMO PARA ENCONTRAR LA GRAFICA DE ADYACENCIAS DE CARAS DE UN S\OLIDO

El algoritmo de adyacencia de caras de un s\olido  $S$ , hace uso de un vector llamado COLA con \i ndices de 0 a 2, donde COLA[0], COLA[1] y COLA[2], son colas donde se guardan apuntadores a caras. El algoritmo toma el cubo  $c_1 = E_{\square}(1)$ , que evidentemente no tiene vecinos a la izquierda, ni abajo ni atr\as. Se construye la gr\afica de adyacencia de caras de  $c_1$ ; si existe para  $c_1$  un vecino derecho, entonces se guarda el apuntador a la cara derecha de  $c_1$  en la cola COLA[0]; si hay un vecino arriba de  $c_1$ , se guarda el apuntador a la cara ARRIBA de  $c_1$  en la cola COLA[1] y si hay un vecino delante, se guarda el apuntador a la cara ADELANTE de  $c_1$  en la cola COLA[2]. Si EXISTE  $c_2 = E_{\square}(2)$ , entonces se construye la gr\afica de adyacencias correspondiente a  $c_2$ ; si  $c_2$  tiene vecino izquierdo, entonces por la proposici\on 3.1 necesariamente debe existir un apuntador a \e l en el tope de COLA[0] lo llamaremos "PDer" (observese que PDer es el apuntador a la cara derecha del vecino izquierdo de  $c_2$ ), considerando el apuntador "PIzq" a la cara izquierda de  $c_2$ , se hace la fusi\on de las caras apuntadas por "PDer" y "PIzq", mediante el algoritmo 3.1. Se hacen operaciones an\alogas en caso de que  $c_2$  tenga vecinos abajo o atr\as, en cuyos casos consideraremos las colas COLA[1] y COLA[2] respectivamente. Si  $c_2$  tiene vecinos a la derecha, arriba o delante, se guardan los apuntadores a las caras DERECHA, ARRIBA \o ADELANTE en COLA[0], COLA[1] \o COLA[2] respectivamente, seg\un sean las adyacencias que se presenten.

Veamos con todo rigor este algoritmo.

ALGORITMO 3.2 (GRAFICA DE ADYACENCIA DE CARAS DE UN S\OLIDO S, PARTIDO REGULARMENTE Y CONEXO)

ENTRADA:

La matriz  $A$  que representa al s\olido  $S$ , se supone que tiene dimensi\on  $n \times n$ .

## SALIDA:

Un apuntador C a una de las caras externas en S, así como la entrada (x,y,z) en A, del cubo al que pertenece esa cara. Las entradas del vector C<sup>^</sup>.LIGA contienen los apuntadores apropiados a las caras adyacentes a C<sup>^</sup> y así sucesivamente. Obsérvese que a partir de C se puede recuperar el conjunto de Arcos en la gráfica. El proceso es simplemente recorrer la estructura de enlaces direccionada por C, pero para nuestros propósitos (encontrar recorridos) esto no es necesario.

## ALGORITMO

PARA k := 1 hasta N HACER:

    PARA j := 1 HASTA n HACER:

        PARA i := 1 HASTA n HACER:

            SI A[i,j,k] = 1 ENTONCES:

                Se genera la gráfica de adyacencia de caras para el cubo unitario asociado con A[i,j,k] y se guarda en la variable CUBO. (Tiene tipo TCubo).

                PARA d := IZQUIERDA HASTA ATRAS HACER:

                    Si el cubo correspondiente a A[i,j,k] tiene un vecino en la dirección d, se saca de la cola COLA[ord(d) mod 3] un apuntador que se guarda en la variable CARA.

                    Se fusionan las caras apuntadas por CUBO[d] y CARA.

                    Se hacen x := i; y := j; z := k.

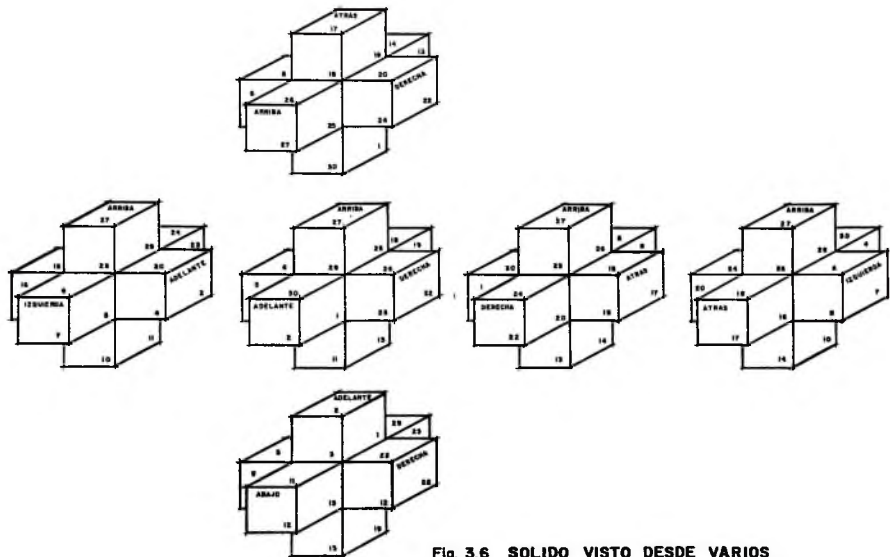
                PARA d := DERECHA HASTA ADELANTE HACER:

                    Si el cubo correspondiente a A[i,j,k] tiene un vecino en la dirección d, entonces se guarda el apuntador CUBO[d] en la cola COLA[ord(d)].

                Se hace C := CUBO[DERECHA]

Termina el algoritmo.

//



**Fig. 3.6 SOLIDO VISTO DESDE VARIOS ANGULOS**

### 3.7 EJEMPLO DE APLICACION DEL ALGORITMO PARA ENCONTRAR LA GRAFICA DE ADYACENCIAS DE UN SOLIDO

En la fig. 3.6, se muestra un sólido partido regularmente conexo, visto desde varios ángulos; para identificar sus caras, se les asignó un número entre 1 y 30. Después de aplicar el algoritmo para encontrar la gráfica de adyacencia de caras de este sólido, se hizo un listado de las variables que representan sus caras externas y se muestra en la tabla 3.1.

Los renglones en la tabla 3.1 muestran las entradas del vector LIGA de cada cara. Se dan los números de las caras a las que apuntan sus cuatro entradas respectivamente.

Para encontrar la Gráfica de Adyacencia de caras del sólido de la fig. 3.6, lo único que hay que hacer es recorrer cada una de las caras que aparecen en la tabla 3.1 e ir anotando los arcos que salen de ellas.

El lector puede verificar que los datos en la tabla 3.1 corresponden al sólido de la fig. 3.6 siguiendo la numeración sobre las caras en el sólido.

### 3.8 COMPLEJIDAD EN EL ALGORITMO PARA ENCONTRAR LA GRAFICA DE ADYACENCIA DE CARAS DE UN SOLIDO

La complejidad del algoritmo 3.2, en función del número máximo  $N$  de particiones en la representación matricial del sólido, (es decir, esta matriz tiene dimensión  $N \times N \times N$ ), es  $O(N^3)$  ya que el algoritmo 3.2 a lo mas tendrá  $N^3$  iteraciones; Esta complejidad puede hacerse lineal respecto del número de cubos en la matriz utilizando la técnica de Hash Extendido explicada en la referencia [2]. Con Octrees también se puede lograr una reducción importante.

### 3.9 OBSERVACIONES Y CONCLUSIONES

Hemos visto la forma de hallar la gráfica de adyacencias de un sólido. Problema que se resolvió con un algoritmo que presenta una complejidad lineal; sin embargo, se puede mejorar si en vez de trabajar con todo el sólido trabajamos con el cascarón del mismo. Así podríamos olvidarnos de procesar aquellos cubos que no tienen ninguna cara como cara externa sobre el sólido. En ese caso la complejidad una vez ya construido el cascarón sería proporcional el número de de caras sobre la superficie del mismo. Obsérvese sin embargo, que a partir de la gráfica de adyacencia de caras de un sólido  $S$ , fácilmente se puede determinar su cascarón.



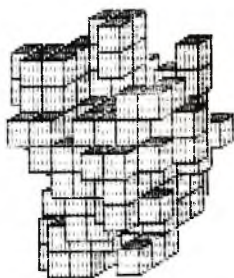
El algoritmo que permitirá determinar recorridos sobre sólidos no es más que una simple ampliación del algoritmo 3.2, y de ahí su importancia.

-----  
 TABLA CORRESPONDIENTE A LA GRAFICA DE ADYACENCIA DE CARAS  
 DEL SOLIDO MOSTRADO EN LA FIGURA 3.6

CARA	TIPO	CARA 0.	CARA 1.	CARA 2.	CARA 3.
1	DERECHA	30	23	3	2
2	ADELANTE	30	1	3	4
3	ABAJO	11	1	2	4
4	IZQUIERDA	30	5	3	2
5	ADELANTE	6	4	9	7
6	ARRIBA	8	28	5	7
7	IZQUIERDA	6	8	9	5
8	ATRAS	6	16	9	7
9	ABAJO	8	10	5	7
10	IZQUIERDA	9	14	12	11
11	ADELANTE	3	13	12	10
12	ABAJO	14	13	11	10
13	DERECHA	21	14	12	11
14	ATRAS	15	13	12	10
15	ABAJO	17	19	14	16
16	IZQUIERDA	18	17	15	8
17	ATRAS	18	19	15	16
18	ARRIBA	17	19	26	16
19	DERECHA	18	17	15	20
20	ATRAS	24	22	21	19
21	ABAJO	20	22	23	13
22	DERECHA	24	20	21	23
23	ADELANTE	24	22	21	1
24	ARRIBA	20	22	23	25
25	DERECHA	27	26	24	29
26	ATRAS	27	25	18	28
27	ARRIBA	26	25	29	28
28	IZQUIERDA	27	26	6	29
29	ADELANTE	27	25	30	28
30	ARRIBA	29	1	2	4

-----





### ALGORITMO PARA ENCONTRAR UN RECORRIDO SOBRE LAS CARAS DE UN SOLIDO

El problema de hallar un recorrido sobre las caras de un sólido  $S$ , o equivalentemente encontrar un camino hamiltoniano sobre su gráfica de adyacencia de caras, será tratado en este capítulo. Nuestro objetivo fundamental, es dar un algoritmo que en un buen número de casos sirva para determinar recorridos.

#### 4.1 RECORRIDOS PARA CUBOS UNITARIOS

Veamos de qué manera se puede encontrar un recorrido para un cubo unitario.

La figura 4.1.a muestra un cubo unitario. Las flechas que aparecen sobre sus caras, indican la sucesión hamiltoniana de un recorrido (C,H) en su gráfica de adyacencia. Esta representación gráfica, será usada a menudo (para una explicación detallada ver la sección 4.2). En este caso, partiendo de la cara DERECHA se tiene la sucesión  $H_1$ :

$h_0 =$  DERECHA,  $h_1 =$  ATRAS,  $h_2 =$  ABAJO,  
 $h_3 =$  IZQUIERDA,  $h_4 =$  ADELANTE,  $h_5 =$  ARRIBA.

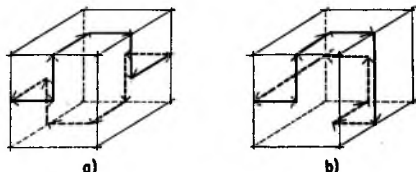


Fig. 4.1 RECORRIDOS SOBRE CUBOS UNITARIOS.

Si consideramos rotaciones, la sucesión hamiltoniana representada en la fig. 4.1.a genera otras, aunque hay algunas sucesiones que no se pueden generar así, una de éstas se muestra en

la fig. 4.1.b. Por ahora, sólo trabajaremos con sucesiones hamiltonianas sobre cubos unitarios como la mostrada en la fig. 4.1.a y las generadas al aplicar rotaciones de esta. (ver la definición de recorridos ortogonales dada más abajo).

Para formalizar los recorridos sobre sólidos, es importante conocer la manera en que se da la adyacencia de sus caras. En un cubo unitario esto es bastante simple y en general se tiene la siguiente observación.

En un cubo unitario para cada cara C, la única cara no adyacente a C es CONTRARIA(C).

Veamos cuáles recorridos usaremos sobre cubos unitarios.

DEFINICION 4.1 (Recorrido Ortogonal).

1. Una Sucesión Hamiltoniana Ortogonal H es una sucesión de la forma  $h_0, \dots, h_n$ , que contiene a todas las caras de un cubo unitario, tal que para toda i entre 0 y  $n-1$   $h_{i+1} = \text{CONTRARIA}(h_i)$ , y además los números  $\text{ord}(h_0)$ ,  $\text{ord}(h_1)$  y  $\text{ord}(h_2)$  son distintos a pares.
2. Un Recorrido Ortogonal O es una pareja de la forma  $(C, H)$ , donde C son las coordenadas de algún cubo unitario y H una sucesión hamiltoniana ortogonal.

//

El nombre de tal recorrido se debe a que cualesquiera tres caras consecutivas en el recorrido son ortogonales a pares.

NOTA 4.1

1. Las caras en un cubo unitario serán representadas por su orientación en tanto no se indique otra cosa.
2. A la sucesión H en un recorrido Ortogonal, le llamaremos Sucesión Hamiltoniana Ortogonal del Recorrido.

//

Se deja al lector verificar que el recorrido mostrado en la figura 4.1.a tiene una sucesión hamiltoniana ortogonal.

No se ha dicho que necesariamente un recorrido ortogonal deba ser un recorrido. Veamos que para un cubo unitario esto es así.

PROPOSICION 4.1

Si  $O = (C,H)$  es un recorrido Ortogonal sobre un cubo unitario, entonces  $O$  es un recorrido.

DEMOSTRACION:

Todas las caras de un cubo unitario por definición aparecen en la sucesión ortogonal  $H$ , por tanto sólo resta ver que para toda  $i$  entre 0 y 5,  $h_i$  es vecina a  $h_{(i+1) \bmod 6}$ . En un cubo unitario, la única cara que no puede ser vecina de  $h_i$  es CONTRARIA( $h_i$ ). Pero por definición de  $H$ ,  $\text{ord}(h_{(i+3) \bmod 6}) \neq \text{ord}(h_{(i+1) \bmod 6})$ , luego  $h_{(i+3) \bmod 6} \neq h_{(i+1) \bmod 6}$ , de donde  $\text{CONTRARIA}(h_i) = h_{(i+3) \bmod 6} \neq h_{(i+1) \bmod 6}$  por lo tanto,  $h_i$  es vecina de  $h_{(i+1) \bmod 6}$ . Se sigue la demostración.

! !

Las demás sucesiones hamiltonianas ortogonales se pueden generar rotando de manera apropiada el recorrido mostrado en la fig. 4.1.a, en total hay 8 recorridos diferentes, según indica la siguiente:

PROPOSICION 4.2

Sobre un cubo unitario, hay exactamente 8 sucesiones hamiltonianas cíclicamente diferentes.

DEMOSTRACION:

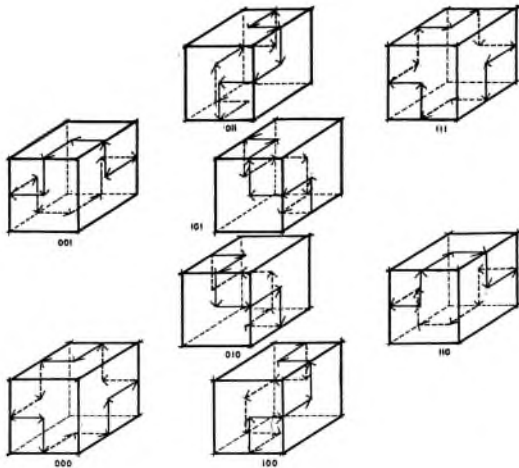
La demostración es por medio de un análisis combinatorio:

Una sucesión hamiltoniana ortogonal  $H$ , queda fijada por los valores de  $h_0, h_1$  y  $h_2$ ; pues  $h_3 = \text{CONTRARIA}(h_0)$  etc. Ahora bien,  $(\text{ord}(h_0) \bmod 3, \text{ord}(h_1) \bmod 3, \text{ord}(h_2) \bmod 3) = (0,1,2)$ , por lo tanto, los valores que pueden tomar los ordinales de  $h_0, h_1$  y  $h_2$  módulo 3 son permutaciones de los elementos 0, 1 y 2; y de éstas hay  $3! = 6$ . Dada una permutación en la que  $\text{ord}(h_0) \bmod 3 = 0$ ,  $h_0$  puede ser DERECHA, o bien IZQUIERDA; de la misma manera  $h_1$  puede tomar alguno de dos valores y lo mismo ocurre con  $h_2$ . En total, para cada permutación de 0, 1 y 2, podemos asignar valores a  $h_0, h_1$  y  $h_2$  de  $2 \times 2 \times 2 = 8$  maneras distintas. Hay (6 permutaciones)  $\times$  (8 asignamientos) = 48 recorridos. Hay 6 representaciones que son cíclicamente iguales. Es decir, a lo más hay  $48/6 = 8$  sucesiones hamiltonianas ortogonales distintas. Estas 8 sucesiones se muestran en la fig. 4.2, por lo que son exactamente 8 sucesiones hamiltonianas ortogonales cíclicamente diferentes las que hay.

! !

En la figura 4.2, se muestran las ocho sucesiones hamiltonianas ortogonales. Obsérvese que se ha establecido una biyección entre los recorridos y los vértices de un cubo de arista 1 ubicados en las coordenadas (0,0,0), (0,0,1), (0,1,0),

$(1,0,0)$ ,  $(1,1,0)$ ,  $(1,0,1)$ ,  $(0,1,1)$  y  $(1,1,1)$ . (La coordenada  $(i,j,k)$  se representa en el dibujo como  $ijk$ ).



**Fig. 4.2 RECORRIDOS ORTOGONALES.**

La biyección mostrada en la figura 4.2 es más que una simple nomenclatura, y de ésta hablaremos a continuación.

#### 4.2 BIYECCION ENTRE LAS SUCESIONES HAMILTONIANAS ORTOGONALES Y LOS VERTICES DE UN CUBO UNITARIO.

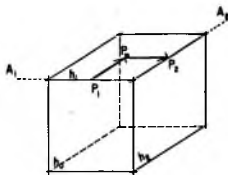
En esta sección, veremos una colección de propiedades sobre los recorridos ortogonales, que usaremos a lo largo del capítulo, empezemos explicando con todo detalle la representación gráfica de recorridos.

La representación gráfica que hemos usado en las figuras 4.1 y 4.2 consiste en dibujar sobre cada cara externa en el sólido dos segmentos de recta dirigidos. Para explicar cómo se dibujan, suponga que se tiene un recorrido  $O = (C,H)$ . Sean  $A_1$  y  $A_2$  las



aristas en el cubo que recorre  $O$ , tales que  $A_0$  es la arista de adyacencia entre  $h_0$  y  $h_1$  y  $A_1$  lo es entre  $h_1$  y  $h_2$ ;  $P_1$ ,  $P_2$  los puntos medios en  $A_1$  y  $A_2$  y  $P_M$  el centro de la cara  $h_1$ . En este

caso sobre  $h_1$  se trazan los segmentos dirigidos  $\overrightarrow{P_1C_M}$  Y  $\overrightarrow{C_MP_2}$ , de la manera que se indica en la figura 4.3. Obsérvese que para los recorridos de la figura 4.2, los segmentos en cada cara se cortan formando un ángulo de  $\pm 90$  grados. Esta se sigue de la condición de ortogonalidad de cualesquier tres caras consecutivas en el recorrido.



**Fig. 4.3 TRAZOS DE UN RECORRIDO ORTOGONAL SOBRE UNA CARA EXTERA.**

Si tomamos la representación gráfica de una sucesión hamiltoniana ortogonal, podemos pensar en la dirección de giro que forman los trazos que aparecen sobre las caras, considerando que unos van en sentido de las manecillas del reloj y otros en sentido contrario, respecto a un observador frente a la cara en cuestión. La figura 4.4, muestra para todas las posibilidades de los trazos sobre las caras los sentidos de giro. Para abreviar el manejo de este sentido de giro, se introduce la siguiente:

**DEFINICION 4.2**

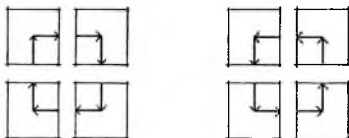
Sea  $H$  una sucesión hamiltoniana ortogonal y  $h_i$  un elemento en  $H$ , entonces se define el sentido de la sucesión sobre  $h_i$  como:

- 0 Si el sentido de giro de los trazos de la representación gráfica de  $O$  sobre la cara  $h_i$  es el de las manecillas del reloj. (Angulo de  $-90$  grados).

$SENTIDO_H(h_i) := ($

- 1 Si el sentido de giro de los trazos de la representación gráfica de  $O$  sobre la cara  $h_i$  es contrario al de las manecillas del reloj. (Angulo de  $+90$  grados).

//



d) TRAZOS EN EL SENTIDO DE LAS MANECILLAS DEL RELOJ

b) TRAZOS EN SENTIDO CONTRARIO A LAS MANECILLAS DEL RELOJ

#### Fig. 4.4 SENTIDO DE GIRO DE RECORRIDOS SOBRE CARAS.

Veamos ahora una propiedad muy importante:

##### PROPOSICION 4.3

Sea  $H$  una sucesión hamiltoniana ortogonal, entonces para toda  $i$  entre  $0$  y  $S$ ,  $SENTIDO_H(h_i) \neq SENTIDO_H(h_{(i+1) \bmod S})$ .

Es decir, en un recorrido ortogonal, dos caras consecutivas tienen giros opuestos.

##### DEMOSTRACION

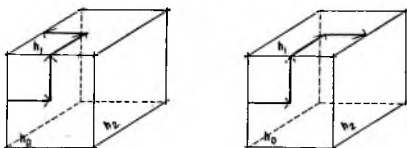
Si la proposición no se cumple, entonces existe  $i$  entre  $0$  y  $S$  tal que  $SENTIDO_H(h_i) = SENTIDO_H(h_{(i+1) \bmod S})$ . Ahora, debido al

caracter cíclico de  $H$ , podemos suponer sin perder generalidad que  $\text{SENTIDO}_H(h_0) = \text{SENTIDO}_H(h_1)$ .

Si suponemos que  $\text{SENTIDO}_H(h_0) = 1$  entonces giremos la representación gráfica de  $Q$ , hasta que la cara  $h_0$  quede de frente como se muestra en la figura 4.5, sobre la cara  $h_1$  se representan las dos formas posibles para el trazo del recorrido. La de la fig. 4.5.a corresponde al caso cuando  $\text{SENTIDO}_H(h_1) = 1$ , pero obsérvese en el dibujo que debería cumplirse que la cara sucesora de  $h_1$  sea igual a la cara predecesora de  $h_0$ , es decir,  $h_2 = h_2$ , lo cual es una contradicción con la definición de  $H$ , pues sabemos que  $h_5 = \text{CONTRARIA}(h_2)$ . Por lo tanto  $\text{SENTIDO}_H(h_1) = 1$ , como se muestra en la fig. 4.5.b.

El caso  $\text{SENTIDO}_H(h_0) = 0$ , se trata de manera análoga.

//



a) TRAZOS QUE NO FORMAN UN RECORRIDO ORTOGONAL

b) RECORRIDO ORTOGONAL

**Fig. 4.5 TRAZOS QUE FORMAN O NO, LA REP. GRAFICA DE RECORRIDOS ORTOGONALES.**

De estas construcciones resultan inmediatos los corolarios siguientes:

**COROLARIO 4.1**

Sea  $H$  una sucesión hamiltoniana ortogonal, entonces la sucesión  $\text{SENTIDO}_H(h_0), \dots, \text{SENTIDO}_H(h_m)$ , es  $0,1,0,1,0,1$  o bien es  $1,0,1,0,1,0$ .

//

Otras propiedades importantes de los recorridos ortogonales se pueden enunciar en términos de la cara que sigue o la que

precede a una cara determinada dentro de una sucesión hamiltoniana, para abreviar las referencias a estos elementos se introduce la definición siguiente:

DEFINICION 4.3

Dada una sucesión hamiltoniana ortogonal  $H = h_0, \dots, h_m$  se definen las funciones  $PRED_H$  y  $SUCC_H$  que van del conjunto  $\{h_0, \dots, h_m\}$  en sí mismo, de tal forma que para toda  $i$  entre 0 y  $S$ :

$$\begin{aligned} SUCC_H(h_i) &= h_{(i+1) \bmod S} \\ PRED_H(h_i) &= h_{(i+n-1) \bmod S} \end{aligned}$$

Es decir, para cada cara  $h$ ,  $SUCC_H(h)$  y  $PRED_H(h)$  son respectivamente las caras posterior y anterior en el recorrido  $H$ .

//

Invariantemente se cumple:

$$SUCC(PRED(h_i)) = PRED(SUCC(h_i)) = h_i$$

COROLARIO 4.2

Si  $H$  es una sucesión hamiltoniana ortogonal en un cubo unitario, entonces para toda  $i$  entre 0 y  $S$ ,

- a)  $SENTIDO_H(h_i) \neq SENTIDO_H(CONTRARIA(h_i))$ ;
- b)  $SENTIDO_H(SUCC_H(h_i)) = SENTIDO_H(PRED_H(h_i))$ ;
- c) Si  $h_j$  es adyacente a  $h_i$  y si  $h_j \neq SUCC_H(h_i)$  y si  $h_j \neq PRED_H(h_i)$ , entonces  $SENTIDO_H(h_i) = SENTIDO_H(h_j)$ ;
- d)  $SUCC_H(h_i)$  y  $PRED_H(h_i)$  son adyacentes.

DEMOSTRACION

- a) y b) son consecuencias inmediatas del corolario 4.1.
- c) Escribamos  $H$  a partir de  $h_0$  como:

$$\begin{array}{cccccc} PRED_H(h_i), & h_i, & SUCC_H(h_i), & h_{i+1}, & CONTRARIA(h_i), & h_m \\ 0, & 1, & 0, & 1, & 0, & 1 \end{array}$$

Sin perder generalidad mostraremos el resultado para  $i = 1$ .

(Abajo de  $h_i$ , se escribe  $SENTIDO_H(h_i)$ ), suponiendo sin pérdida de generalidad que  $SENTIDO_H(h_0) = 0$ .

Pero por el lema 4.1, sabemos que la única cara no

adyacente a  $h_1$  es  $\text{CONTRARIA}(h_1)$ ; entonces,  $h_3$  y  $h_6$  son las únicas caras adyacentes a  $h_1$  que no son  $\text{PRED}_H(h_1)$  ni  $\text{SUCC}_H(h_1)$ . Se tiene lo afirmado puesto que  $\text{SENTIDO}_H(h_3) \neq \text{SENTIDO}_H(h_6)$ .

- d) Si suponemos que  $\text{PRED}_H(h_1)$  no es adyacente a  $\text{SUCC}_H(h_1)$  entonces como ambas caras son adyacentes a  $h_1$ , se debe cumplir  $\text{CONTRARIA}(\text{PRED}_H(h_1)) = \text{SUCC}_H(h_1)$ . Luego por a),  $\text{SENTIDO}_H(\text{PRED}_H(h_1)) \neq \text{SENTIDO}_H(\text{CONTRARIA}(\text{PRED}_H(h_1))) = \text{SENTIDO}_H(\text{SUCC}_H(h_1))$ , lo que es una contradicción con b), de donde se sigue lo afirmado.

||

En la fig. 4.2 el lector puede verificar visualmente lo que dice este corolario.

Veamos ahora cómo las sucesiones hamiltonianas ortogonales se pueden identificar con los vértices de un cubo unitario.

#### DEFINICION 4.4

Sea  $RC$  la función que va del conjunto de sucesiones hamiltonianas ortogonales al conjunto  $\{(0,0,0), (1,0,0), (0,1,0), (1,1,0), (0,0,1), (1,0,1), (0,1,1), (1,1,1)\}$  tal que para cada sucesión  $H$ ,

$$RC(H) = (\text{SENTIDO}_H(\text{DERECHA}), \text{SENTIDO}_H(\text{ADELANTE}), \text{SENTIDO}_H(\text{ARRIBA}))$$

Así pues  $RC(H)$  es un vértice.

//

La elección de los sentidos de  $H$  únicamente en las tres caras DERECHA, ADELANTE y ARRIBA queda plenamente justificada por la siguiente propiedad de  $RC$ .

#### PROPOSICION 4.4

$RC$  es una biyección.

#### DEMOSTRACION

La demostración puede hacerse en base a las propiedades de las sucesiones hamiltonianas ortogonales pero es evidente de la fig. 4.2, donde se muestra explícitamente esta transformación.

||

#### 4.3 RECORRIDOS SOBRE SOLIDOS FORMADOS POR DOS CUBOS UNITARIOS ADYACENTES.

En el caso bidimensional notamos que si se podía recorrer la frontera de dos pixels adyacentes, entonces se pueden juntar los recorridos para formar uno solo (ver capítulo I). En este caso se eliminaron las aristas de adyacencia de ambos pixels. En general cuando se quiere agregar un pixel a una imagen ya recorrida, hay que "romper" el recorrido sobre la imagen, y restituirlo una vez introducido el pixel.

En tres dimensiones usando recorridos ortogonales, se puede hacer un tipo de "rompimiento" similar en algunos casos. Veamos esto detalladamente:

#### DEFINICION 4.5

Sean  $S_A$  y  $S_B$ , sólidos partidos regularmente conexos que sólo tienen una cara en común. Supongamos que existen recorridos  $R_A = (C_A, H_A)$  y  $R_B = (C_B, H_B)$  para  $S_A$  y  $S_B$  respectivamente. Denotemos por "a" y "b" a los elementos en  $H_A$  y  $H_B$  asociados con las caras de contacto de  $S_A$  y  $S_B$ .

Se dice que  $R_A$  y  $R_B$  son fusionables, si en el sólido formado por la unión de  $S_A$  y  $S_B$  se tiene que:

$PRED_{H_A}(a)$  es vecino de  $SUCC_{H_B}(b)$  y  
 $SUCC_{H_A}(a)$  es vecino de  $PRED_{H_B}(b)$

//

En el capítulo III se usó por primera vez el término "fusionar" (VER ALGORITMO 3.1); su uso en la definición 4.5 tiene que ver con la gráfica de adyacencias, aunque esto no se explica ahora. Por el momento mostremos que como el nombre lo sugiere, dos recorridos fusionables se pueden convertir en uno solo.

#### PROPOSICION 4.5

Sean  $S_A, S_B, R_A = (C_A, H_A), R_B = (C_B, H_B), "a" y "b",$  como en la definición 4.5, con  $R_A$  y  $R_B$  recorridos fusionables. Entonces el sólido  $S$  formado por la unión de  $S_A$  y  $S_B$  es recorrible.

#### DEMOSTRACION

Escribamos  $H_A$  como  $hA_1, \dots, hA_m$  y a  $H_B$  como  $b, hB_1, \dots, hB_n$ , donde  $m+1$  y  $n+1$  son el número de caras externas en  $S_A$  y  $S_B$  respectivamente.

Veamos que la sucesión  $HAB$  definida como  $hA_1, \dots, hA_m, hB_1, \dots, hB_n$  es una sucesión hamiltoniana para  $S$ .

Es claro que todas las caras externas de  $S$  aparecen en  $HAB$ .

Falta demostrar que dos caras sucesivas (cíclicamente) en

HAB son adyacentes. Esto lo sabemos de entrada para las caras:

$hA_i$  y  $hA_{(i+1) \bmod m}$  para  $i$  entre 1 y  $m$ . Pues  $H_A$  es una sucesión hamiltoniana.

También lo sabemos para las caras:

$hB_j$  y  $hB_{(j+1) \bmod n}$  para  $j$  entre 1 y  $n$ .

Por lo tanto, sólo falta ver que  $hB_n$  y  $hA_1$  son adyacentes y  $hA_m$  y  $hB_1$  también lo son; pero esto es precisamente lo que dice la definición 4.5 (véase quiénes son  $SUCC_{H_A}(a)$  etc), de donde se sigue que  $S$  es recorrible por  $R = (C, HAB)$ , donde  $C$  es la coordenada del cubo al que pertenece la cara  $hA_1$ .

||

Si tenemos un sólido formado por dos cubos unitarios, existen recorridos ortogonales fusionables para éstos. La siguiente definición dice como deben asignarse los recorridos ortogonales.

#### DEFINICION 4.6

Para cada triada  $(i, j, k)$  donde  $i, j, k$  pertenecen al conjunto  $\{0,1\}$ , se define la función de asignación de sucesiones hamiltonianas ortogonales  $i, j, k$ , denotada por  $A_{i,j,k}$ , que va del conjunto de cubos unitarios al conjunto de sucesiones hamiltonianas ortogonales tal que si un cubo tiene coordenadas  $(x, y, z)$  entonces  $A_{i,j,k}(x, y, z) = RC^{-1}((x+i) \bmod 2, (y+j) \bmod 2, (z+k) \bmod 2)$ .

//

Como ejemplo de la definición 4.6,  $A_{000}(0,0,0)$  es la sucesión hamiltoniana ortogonal 000 (ver fig. 4.2), de la misma manera  $A_{000}(5,2,3)$  es la sucesión 101.

Para motivar el porque de la definición 4.6, le recomendamos al lector que consiga algunos cubos de madera y construya con ellos el sólido mostrado en la fig. 3.6, anotando sobre cada uno de ellos sus coordenadas (fije las coordenadas de alguno y con estas derive las demás). Un vez hecho esto dibuje sobre cada cubo el recorrido que le toca según la función  $A_{000}$ . Sobre las caras externas del sólido quedarán marcadas líneas. Pues bien, si se siguen se observará un recorrido sobre todo el sólido. Lo mismo se observa si se asignan los recorridos a los cubos usando alguna de las otras funciones  $A_{i,j,k}$ . Es decir, en algunos casos como el sólido de la fig. 3.6, las funciones de asignación de sucesiones hamiltonianas generan recorridos de una manera muy simple.

Veamos en el caso más sencillo funcionan las funciones de asignamiento de sucesiones hamiltonianas.

#### PROPOSICION 4.6

Sean  $CA = (xA, yA, zA)$  y  $CB = (xB, yB, zB)$  dos cubos unitarios adyacentes con  $CA < CB$ . Entonces, para cada triada  $(i, j, k)$  con  $i, j, k$  en el conjunto  $\{0, 1\}$ , los recorridos  $OA := (CA, A_{i,j,k}(xA, xB, xC))$  y  $OB := (CB, A_{i,j,k}(xA, xB, xC))$  son fusionables.

#### DEMOSTRACION

La demostración se puede hacer usando las propiedades vistas para recorridos ortogonales; no obstante, haremos una demostración de tipo exhaustiva aunque sólo se verá un caso específico, los demás se dejan al lector.

Sean "a" y "b" las caras en  $HA$  y  $HB$  asociadas con las caras de contacto de  $CA$  y  $CB$  respectivamente.

Como los cubos  $CA$  y  $CB$  son adyacentes, entonces  $CA$  y  $CB$  coinciden en dos entradas y difieren en la otra, de manera que se tiene que cumplir exactamente una de las condiciones siguientes:

1. Si las entradas de  $CA$  y  $CB$  correspondientes a la primera coordenada difieren, entonces las otras entradas coinciden, es decir  $yA=yB$  y  $zA=zB$ . Como  $CA$  es menor lexicográficamente respecto a  $CB$  entonces  $xA+1 = xB$ .
2. Si las entradas de  $CA$  y  $CB$  correspondientes a la segunda coordenada difieren, entonces  $xA=xB$ ,  $yA+1=yB$  y  $zA=zB$ .
3. Si las entradas de  $CA$  y  $CB$  correspondientes a la tercera coordenada difieren, entonces  $xA=xB$ ,  $yA=yB$  y  $zA+1=zB$ .

Supongamos que se cumple la condición 1.

"a" es la cara DERECHA del cubo  $CA$ , "b" la cara IZQUIERDA de  $CB$ .

$HA$  puede ser alguna de 8 sucesiones hamiltonianas ortogonales.

0. Si  $C1$  tiene la sucesión hamiltoniana  $(0,0,0)$  entonces  $C2$  debe tener la  $(1,0,0)$ , por tanto  $SUCC_{HA}(a)$  es la cara ATRAS de  $CA$  y  $PRED_{HB}(b)$  es la cara ATRAS de  $CB$  y es evidente que estas caras son adyacentes. Del mismo modo las caras  $PRED_{HA}(a)$ ,  $SUCC_{HB}(b)$  son adyacentes. Por lo tanto la proposición se cumple en este caso.

Se deja al lector la tarea verificar de manera análoga que se cumple la proposición para los otros siete casos. (A primera vista esto parece una crueldad, basta que el lector observe que si acerca los cubos de la fig. 4.2 hasta que hagan contacto sus caras, entonces los trazos de la representación gráfica de los recorridos se enciman para cada par de cubos y tienen sentido contrario, de ahí

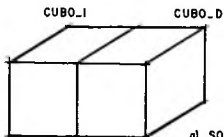


es inmediato que los recorridos sean fusionables).

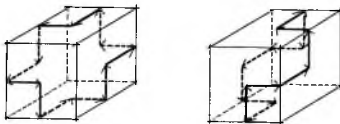
De manera análoga, se cumple la proposición para las condiciones 2 y 3.

||

Como ejemplo de la posición 4.6, tomemos los recorridos ortogonales  $D_I = (CA, HA) = (CI, RC^{-1}(0,0,0))$  y  $D_D = (CD, HB) = (CD, RC^{-1}(0,0))$  para los cubos CUBO\_I y CUBO\_D de la fig. 4.6.a respectivamente. En la fig. 4.6.b se muestran estos recorridos aunque ahora se han separado los cubos para que se puedan ver los trazos sobre las caras de contacto. Obsérvese que estos trazos se enciman al pegarse las caras de contacto pero con sentido contrario, de la misma manera que las cadenas de Freeman sobre las aristas de contacto de dos pixels. Esto permite construir el recorrido sobre los cubos unitarios mostrado en la fig. 4.6.c. Este "encimarse" equivale a que los recorridos sean fusionables.



a) SÓLIDO FORMADO POR DOS CUBOS UNITARIOS ADYACENTES.

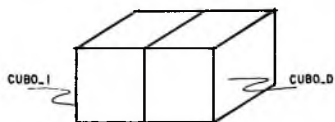


b) RECORRIDOS 000 y 100

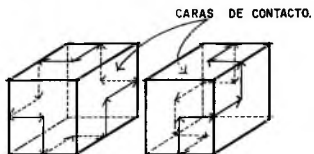


c) RECORRIDO SOBRE EL SÓLIDO DEL INCISO a)

Fig. 4.6 FUSION DE RECORRIDOS ORTOGONALES.

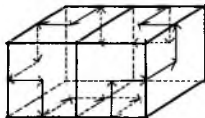


a) SOLIDO FORMADO POR DOS CUBOS ADYACENTES



b) RECORRIDOS ORTOGONALES.

OBSERVESE QUE SOBRE LAS CARAS DE CONTACTO LOS RECORRIDOS QUEDAN ENCIMADOS Y CON SENTIDOS OPUESTOS.



c) RECORRIDO PARA EL SOLIDO DEL INCISO a)

FIG. 4.7 EJEMPLO DE LA PROR 4.6

#### 4.4 RECORRIDOS SOBRE SÓLIDOS FORMADOS POR MÁS DE DOS CUBOS UNITARIOS

La proposición 4.6 afirma que todo sólido formado por dos cubos unitarios adyacentes es recorrible. Podría uno imaginar que así como se pueden pegar los recorridos ortogonales de dos cubos unitarios adyacentes, se pueden pegar para los cubos unitarios en un sólido arbitrario. En la fig. 4.8 se muestran sólidos que se recorren generalizando el método de la sección anterior, es decir usando las funciones de asignación de sucesiones hamiltonianas ortogonales; sin embargo, no es cierto que se pueda recorrer un sólido arbitrario. Ahora caracterizaremos los sólidos que se pueden recorrer siguiendo esta idea. Empecemos definiendo este tipo de sólidos y a continuación veremos porqué son recorribles de manera tan simple.

#### DEFINICION 4.7

1. Se dice que un sólido  $S$  es simple, si  $S$  es un cubo unitario o bien  $S = S' \cup C$ , donde  $S'$  es un sólido simple y  $C$  es un cubo unitario tales que  $S'$  y  $C$  sólo tienen una cara en común.
2. Un sólido es no simple, si no es simple.

//

Todos los sólidos de la fig. 4.8 son simples.

Obsérvese que todo sólido simple es un sólido partido regularmente y conexo.

En la proposición 4.7, se muestra por qué los sólidos simples son recorribles.

#### PROPOSICION 4.7

Todo sólido simple es recorrible.

#### DEMOSTRACION

Sea  $S$  es un sólido simple.

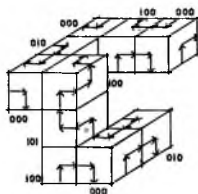
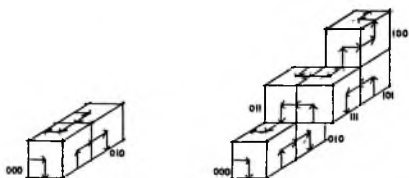
La demostración se hará por inducción sobre el número  $n$  de cubos en  $S$ .

#### CASO BASE:

Si  $n = 1$ , demos el recorrido ortogonal  $(C, A_{000})$  donde  $C$  es la coordenada del cubo unitario  $S$ .

Si  $n = 2$ , la demostración se sigue de la proposición 4.6.

#### CASO INDUCTIVO:



**FIG. 4.8 SOLIDOS SIMPLES**

JUNTO A CADA CUBO SE INDICA LA SUCESSION  
HAMILTONIANA ORTOGONAL ASOCIADA EN EL  
RECORRIDO.

Supongamos que la proposición se cumple para  $n = k$ , veamos que también se cumple para  $n = k+1$ .

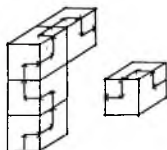
Como  $S = S' \cup C$ , donde  $S'$  es un sólido simple formado por  $k$  cubos unitarios, entonces  $S'$  debe ser recorrible; más aun, no hay contradicción con el caso base si suponemos que en la representación gráfica del recorrido de  $S'$ , aparecen sobre las caras externas de  $S$  los trazos de los recorridos ortogonales asignados por la función  $A_{000}$ .

En este punto la demostración puede seguirse como la de la proposición 4.6 considerando las caras de contacto entre  $S'$  y  $C$ . Sin embargo, es más ilustrativo utilizar un argumento de tipo geométrico.

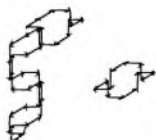
Recorramos el cubo  $C$  usando la sucesión hamiltoniana  $A_{000}(C)$ . Entonces sobre las caras de contacto de  $S$  y  $C$  aparecen los trazos de los recorridos encimados y en sentido contrario, de manera que si "desprendemos" los recorridos del sólido  $S$  y el cubo  $C$  como se muestra en la figura 4.9.b, podemos pensar que este proceso visto sobre el trazo es equivalente a "anudar" los trazos en los puntos  $P_1$  y  $P_2$  mostrados en la fig. 4.9.c y tirar los pedazos de en medio. En la fig. 4.9.d se vuelven a pegar estos trazos sobre el sólido, formando un recorrido único. Se deja al lector formalizar este proceso.

||

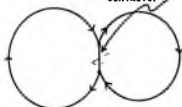
Esta proposición nos da un buen número de sólidos que pueden recorrerse. Si reflexiona el lector un poco, se dará cuenta que para programar el método de recorrido no hay más que hacer algunas ampliaciones al algoritmo para encontrar la gráfica de adyacencia del sólido asociado. Este es el objetivo de la próxima sección.



a) EN LA FIG. APARECE UN SOLIDO SIMPLE S' Y UN CUBO UNITARIO C. S' Y C FORMAN UN SOLIDO S. (VER INCISO e)).

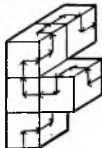
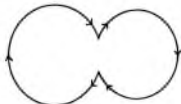


TRAZOS SOBRE LAS CARAS DE CONTACTO.



b) LOS TRAZOS SOBRE LAS CARAS DE UN SOLIDO SE PUEDEN VER COMO UN "LAZO" CIRCULAR QUE PASAN POR TODAS SUS CARAS EXTERNAS. SI AISLAMOS LOS DE S' Y C, ESTOS SE VEN COMO EN LA FIG.

c) DESDOBLANDO LOS LAZOS DEL INCISO d) Y JUNTANDOLOS COMO EN e). SE OBSERVA QUE SOBRE LAS CARAS DE CONTACTO SE ENCIMA CON SENTIDOS OPUESTOS



d) JUNTAR S' CON C ES COMO CORTAR EN P1 Y P2 LOS TRAZOS ENCINADOS.

e) RECORRIDO FINAL PARA S.

FIG. 4.9 PROCESO INDUCTIVO PARA RECORRER SOLIDOS SIMPLES.

#### 4.5 ALGORITMO PARA RECORRER SOLIDOS SIMPLES

En la demostración de la proposición 4.7, vimos que los recorridos ortogonales dados por las funciones de asignación de sucesiones hamiltonianas ortogonales, inducen recorridos sobre todo el sólido. La idea del algoritmo que se da a continuación es la de llevar las funciones de asignación a las estructuras de datos del capítulo anterior.

Dado un sólido simple  $S$ , para encontrarle un recorrido  $R = (C,H)$ , se determina primero su gráfica de adyacencia caras, en cada cara externa debe decirse por donde "llega" el recorrido y por donde "sale"; por esto al tipo  $TCara$  se le agregan dos campos llamados "ent" y "sal", el objetivo del algoritmo es que al terminarse, para cada cara externa  $a$  en  $S$ ,  $PRED_H(a^{\wedge})$ , que es donde llega el recorrido, debe ser lo mismo que  $a^{\wedge}.LIGA[a^{\wedge}.ent]$  y  $SUCC_H(a^{\wedge})$ , que es por donde sale, debe ser lo mismo que  $a^{\wedge}.LIGA[a^{\wedge}.sal]$ .

La demostración de la proposición 4.7 indica que al menos hay 8 maneras diferentes de lograr el objetivo del párrafo anterior usando las funciones de asignación  $A_{i,j,k}$ . En el resto de la sección usaremos para los recorridos sólo la función  $A_{000}$ , pero de igual manera podría haberse elegido cualesquiera de las otras.

La base del algoritmo es construir las 8 sucesiones hamiltonianas ortogonales posibles para cubos unitarios y guardarlas en un número igual de variables de tipo  $TCubo$  (ver sección 3.7). En la proposición 4.7 vimos la identificación que hay entre las sucesiones hamiltonianas ortogonales y los vértices de un cubo con vértices de la forma  $(i,j,k)$ , donde  $i,j,k$  son elementos de  $\{0,1\}$ , por lo que una manera natural de representar los ocho recorridos ortogonales es mediante variables de tipo:

```
Type T_Recorridos = array[0..1,0..1,0..1] of TCubo;
```

De esta manera si declaramos:

```
var RECORRIDOS : T_Recorridos
```

en la entrada  $(i,j,k)$  de  $RECORRIDOS$  se guarda  $RC^{-1}(i,j,k)$ ,  $i,j,k$  en  $\{0,1\}$ . Para hacer ésto, debemos asignar valores adecuados a los campos "ent" y "sal" en  $RECORRIDOS[i,j,k]$ . Las tablas que se presentan a continuación muestran explícitamente los asignamientos necesarios para cada sucesión hamiltoniana ortogonal. (El lector debe consultar las figuras 3.3 y 4.2 para comprobar los asignamientos).

ASIGNACIONES PARA LA SUCESION HAMILTONIANA ORTOGONAL  $RC^{-1}(0,0,0)$

ENTRADA EN RECORRIDOS[0,0,0]	ENT	SAL
DERECHA	2	1
ARRIBA	0	3
ADELANTE	3	2
IZQUIERDA	0	3
ABAJO	2	1
ATRAS	1	0

ASIGNACIONES PARA LA SUCESION HAMILTONIANA ORTOGONAL  $RC^{-1}(0,0,1)$

ENTRADA EN RECORRIDOS[0,0,1]	ENT	SAL
DERECHA	1	0
ARRIBA	1	2
ADELANTE	0	3
IZQUIERDA	3	2
ABAJO	3	0
ATRAS	2	1

ASIGNACIONES PARA LA SUCESION HAMILTONIANA ORTOGONAL  $RC^{-1}(0,1,0)$

ENTRADA EN RECORRIDOS[0,1,0]	ENT	SAL
DERECHA	3	2
ARRIBA	3	2
ADELANTE	0	1
IZQUIERDA	1	0
ABAJO	1	0
ATRAS	2	3

ASIGNACIONES PARA LA SUCESION HAMILTONIANA ORTOGONAL  $RC^{-1}(0,1,1)$

ENTRADA EN RECORRIDOS[0,1,1]	ENT	SAL
DERECHA	0	3
ARRIBA	0	1
ADELANTE	1	2
IZQUIERDA	2	1
ABAJO	2	3
ATRAS	3	0



ASIGNACIONES PARA LA SUCESION HAMILTONIANA ORTOGONAL  $RC^{-1}(1,0,0)$

ENTRADA EN RECORRIDOS[1,0,0]	ENT	SAL
DERECHA	3	0
ARRIBA	1	0
ADELANTE	2	1
IZQUIERDA	1	2
ABAJO	3	2
ATRAS	0	3

ASIGNACIONES PARA LA SUCESION HAMILTONIANA ORTOGONAL  $RC^{-1}(1,0,1)$

ENTRADA EN RECORRIDOS[1,0,1]	ENT	SAL
DERECHA	2	3
ARRIBA	2	3
ADELANTE	1	0
IZQUIERDA	0	1
ABAJO	0	1
ATRAS	3	2

ASIGNACIONES PARA LA SUCESION HAMILTONIANA ORTOGONAL  $RC^{-1}(1,1,0)$

ENTRADA EN RECORRIDOS[1,1,0]	ENT	SAL
DERECHA	0	1
ARRIBA	2	1
ADELANTE	3	0
IZQUIERDA	2	3
ABAJO	0	3
ATRAS	1	2

ASIGNACIONES PARA LA SUCESION HAMILTONIANA ORTOGONAL  $RC^{-1}(1,1,1)$

ENTRADA EN RECORRIDOS[1,1,1]	ENT	SAL
DERECHA	1	2
ARRIBA	3	0
ADELANTE	2	3
IZQUIERDA	3	0
ABAJO	1	2
ATRAS	0	1

Estos valores se asignan dentro de un proceso inicial y se usan de la manera siguiente:

Supongamos que en la construcción de la gráfica de adyacencia de caras de un sólido simple  $S$  estamos fusionando la gráfica de un cubo  $C$  con la parte de la gráfica de  $S$  ya construida. Si  $C$  tiene coordenadas  $(x,y,z)$  entonces tomamos la sucesión hamiltoniana ortogonal representada en  $\text{RECORRIDOS}(x \bmod 2, y \bmod 2, z \bmod 2)$  y copiamos los valores de "ent" y "sal" de la manera siguiente:

```
i := x mod 2; j := y mod 2; k := z mod 2;
for ori := DERECHA TO ATRAS DO
begin
  C[ori]^ent := RECORRIDOS[i,j,k,ori]^Ent;
  C[ori]^sal := RECORRIDOS[i,j,k,ori]^Sal
end;
```

Así es como logramos el recorrido generado por la sucesión de asignamiento  $A_{000}$ .

Las instrucciones pascal mostradas arriba se agregan al algoritmo 3.2, así como el procedimiento que inicializa la variable  $\text{RECORRIDOS}$ . El lector debe observar que el algoritmo 4.1 es sólo una extensión del algoritmo 3.2.

ALGORITMO 4.1 (RECORRIDOS SOBRE LAS CARAS EXTERNA DE UN SOLIDO  $S$ , PARTIDO REGULARMENTE Y CONEXO)

ENTRADA:

La matriz  $A$  que representa al sólido  $S$  y se supone que tienen dimensión  $n \times n$ .

SALIDA:

Un apuntador  $C$  a una de las caras externas en  $S$ , así como la entrada  $(x,y,z)$  en  $A$ , del cubo al que pertenece esa cara. Las entradas del vector  $C^{\wedge} \text{LIGA}$  contienen los apuntadores apropiados a las caras adyacentes a  $C^{\wedge}$  y así sucesivamente. A las variables  $\text{ent}$  y  $\text{sal}$  de esta estructura se les asignan valores adecuados para que representen un recorrido según lo dicho en esta sección. Las mismas observaciones del algoritmo 3.2 son válidas.

ALGORITMO

Se asignan valores iniciales  $\text{RECORRIDOS}$

PARA k := 1 hasta N HACER:

PARA j := 1 HASTA n HACER:

PARA i := 1 HASTA n HACER:

SI  $A[i,j,k] = 1$  ENTONCES:

Se genera la gráfica de adyacencia de caras para el cubo unitario asociado con  $A[i,j,k]$  y se guarda en la variable CUBO. (Tiene tipo TCubo).

Se copian los asignamientos de los campos ent y sal de la variable RECORRIDOS en CUBO, según lo dicho en esta sección.

PARA d := IZQUIERDA HASTA ATRAS HACER:

Si el cubo correspondiente a  $A[i,j,k]$  tiene un vecino en la dirección d, se saca de la cola  $COLA[ord(d) \bmod 3]$  un apuntador que se guarda en la variable CARA.

Se fusionan las caras apuntadas por  $CUBO[d]$  y CARA.

Se hacen  $x := i$ ;  $y := j$ ;  $z := k$ .

PARA d := DERECHA HASTA ADELANTE HACER:

Si el cubo correspondiente a  $A[i,j,k]$  tiene un vecino en la dirección d, entonces se guarda el apuntador  $CUBO[d]$  en la cola  $COLA[ord(d)]$

Se hace  $C := CUBO[DERECHA]$

Termina el algoritmo.

//

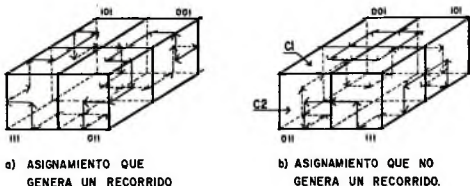
Las extensiones al algoritmo 4.1 se muestran en negritas.

#### 4.6 RECORRIDOS SOBRE SÓLIDOS NO SIMPLES

La proposición 4.7 garantiza que el algoritmo 4.1 recorre cualquier sólido simple. Sin embargo, ¿Qué pasa si aplicamos el algoritmo 4.1 a un sólido no simple? La respuesta es que en algunos casos obtendremos recorridos y en otros no; de esto hablaremos a continuación.

Los incisos a) y b) de la fig. 4.10 muestran el mismo sólido S, pero obsérvese que para cada uno se ha usado una función de

asignamiento de recorridos ortogonales diferente. De esto resulta que los trazos en a) forman un recorrido, pero los trazos en b) no. Pongamos nuestra atención en el inciso b); si seguimos los trazos partiendo de la cara c1 obtenemos una trayectoria que se cierra antes de haber pasado por todas las caras externas de S; si empezamos en la cara c2 se observa algo semejante aunque los ciclos formados son distintos. Estudiemos porqué ocurre esto. La razón se explica mediante un argumento de tipo geométrico:



**FIG. 4.10 ASIGNAMIENTOS DIFERENTES DE RECORRIDOS ORTOGONALES A UN SÓLIDO.**

En la fig. 4.11.a, se muestra el mismo sólido de la fig. 4.10 pero se ha aislado uno de sus cubos unitarios, de manera que el sólido S' resulta simple y por tanto sabemos que los trazos mostrados en el dibujo corresponden al de un recorrido. De manera similar a la explicada en la fig. 4.9, en 4.11.b, se aíslan los "lazos" extendidos del recorrido tanto de S' como de C. Poner C en su posición dentro de S' para formar S, equivale a pegar los lazos asociados en los puntos P<sub>1</sub>, P<sub>2</sub>, P<sub>3</sub> y P<sub>4</sub> desechando los segmentos mostrados en 4.11.c. Ahora quedan dos "lazos" disjuntos sobre la superficie de S como se indica en 4.11.c y de ahí sigue que esto no corresponda a un recorrido.

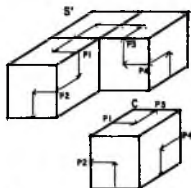
La fig. 4.10 sugiere que si tenemos suerte el algoritmo 4.1 sirve para determinar recorridos aun sobre sólidos no simples. Sin embargo, hay sólidos como el de la fig. 4.12 que bajo ningún asignamiento es recorrible; invitamos al lector a que construya los 8 asignamientos y observe que nunca queda un recorrido. En algunos casos es posible modificar los valores finales de los campos ent y sal de las caras de manera que aun cuando no se obtenga un recorrido después de aplicar el algoritmo 4.1, se logre un recorrido a partir de las asignaciones hechas; por ejemplo, en la fig. 4.13.a se muestra lo mismo que en 4.10.b pero cambiando algunos trazos sobre cuatro de sus caras y el resultado obtenido es un recorrido sobre el sólido.

Observamos que sobre las caras de sólidos no simples en

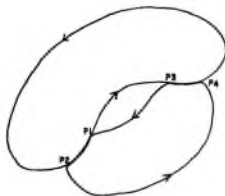
general se forman ciclos disjuntos. Definamos esto rigurosamente.

**DEFINICION 4.8**

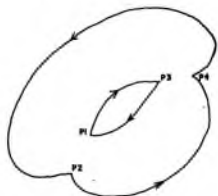
Sea  $S$  un sólido partido regularmente y conexo.



a)

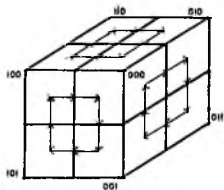


b)



c)

**FIG. 4.11 FUSION DE RECORRIDOS QUE NO FORMA RECORRIDO.**



**FIG. 4.12 SOLIDO NO RECORRIBLE BAJO CUALQUIER ASIGNAMIENTO.**

1. Un ciclo sobre S es una pareja de la forma  $CI = (C, H)$ , donde H es una sucesión de caras externas distintas en S de la forma  $h_0, \dots, h_{n-1}$ ; tal que  $h_i$  es vecina de  $h_{(i+1) \bmod n}$  y C son las coordenadas del cubo al que pertenece  $h_0$ .

Observe que un recorrido sobre S es un ciclo que contiene todas las caras externas en S.

2. Se dice que S es recorrible por ciclos si existen  $CI_1 = (C_1, H_1), CI_2 = (C_2, H_2), \dots, CI_k = (C_k, H_k)$  ciclos sobre S tales que:
  - a)  $\{c; c \text{ es cara externa en } S\} = \{c; \text{ existe } i \text{ tal que } c \text{ es elemento de la sucesión } H_i\}$  y
  - b) Para cada cara externa c de S, si c es elemento de  $H_i$  y también de  $H_j$ , entonces  $i = j$ .

Ahora una propiedad general sobre sólidos.

#### PROPOSICION 4.8

Todo sólido S partido regularmente y conexo es recorrible por ciclos.

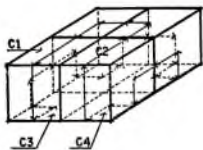
#### DEMOSTRACION

Sigue de aplicar cualquier función de asignamiento de sucesiones hamiltonianas ortogonales a los cubos unitarios de S de igual manera que en el algoritmo 4.1. La demostración se hace por inducción sobre el número de cubos en el sólido y sólo es una generalización simple de la idea presentada en la fig. 4.11, que se deja al lector.

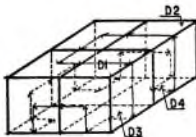
||

Nuestro objetivo es dar un algoritmo que a partir de un recorrido por ciclos encuentre en el mejor de los casos un recorrido; si no, que sirva para reducir el número de ciclos si es que se puede. La idea es juntarlos como se indica a continuación.

La fig. 4.13.a muestra nuevamente al sólido de la fig. 4.10, sin embargo se han cambiado los trazos de las caras  $c_1, c_2, c_3$  y  $c_4$  de tal forma que aparece un recorrido sobre S. Esta forma de "pegar" ciclos no es única; véase que en 4.13.b también aparece un recorrido aunque ahora las caras alteradas son  $d_1, d_2, d_3$  y  $d_4$ . Como se ve, hay que encontrar una estrategia para romper ciclos y restituirlos formando uno solo; el criterio para saber en que parte hacer el rompimiento es algo que aun no hemos determinado en general de manera práctica (existe pues el problema es finito y puede usarse algún algoritmo de vuelta atrás), lo más que se ha podido hacer es dar un criterio que en un buen número de caso sirve. Expliquémoslo:

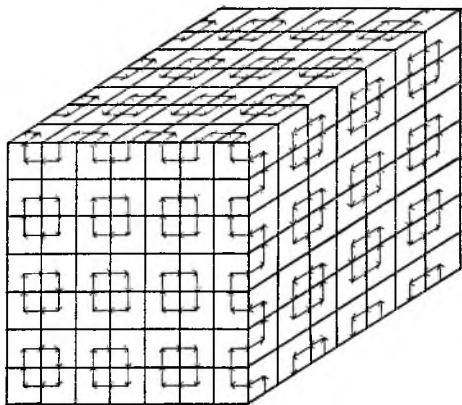


a) LOS TRAZOS DE LA FIG 3.9.b SE MODIFICAN SOBRE LAS CARAS C1, C2, C3 Y C4 PARA FORMAR UN RECORRIDO.



b) IDEM. a) PERO CON LAS CARAS D1, D2, D3 Y D4.

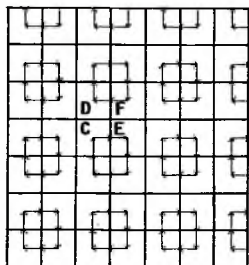
**FIG. 4.13 PEGADO DE CICLOS PARA FORMAR UN RECORRIDO.**



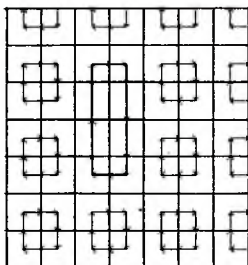
**FIG. 4.14 CICLOS PRODUCIDOS AL USAR RECORRIDOS ORTOGONALES.**



La fig. 4.14 muestra un cubo S formado por varios cubos unitarios. Después de asignarle recorridos ortogonales a estos cubos, observamos que sobre la superficie de S aparecen ciclos y la mayoría de estos tienen forma cuadrada e involucran sólo cuatro caras. En general, sobre sólidos que tienen áreas planas extensas en su superficie se observa lo mismo al hacer cualquier asignamiento. Esto puede aprovecharse para pegar ciclos de la manera siguiente: (la explicación se hace para la fig. 4.15, pero es fácilmente generalizable).



a)



b)

FIG. 4.15 PEGADO DE DOS CICLOS.

Primero tomamos una cara C arbitraria sobre S (ver fig. 4.15.a) y marcamos con un trazo más fuerte el ciclo al que pertenece C; luego nos fijamos en un vecino de C no marcado, por ejemplo la cara D; ahora, vemos que  $E := C^{\wedge}.LIGA[C^{\wedge}.sal]$  es vecino de  $F := D^{\wedge}.LIGA[D^{\wedge}.ent]$  (las caras se indican en 4.15.a). Esto es suficiente para saber que los ciclos involucrados se pueden pegar, pues si hacemos los asignamientos

$C^{\wedge}.sal := Entrada\ en\ C^{\wedge}.LIGA\ que\ tiene\ un\ apuntador\ a\ D^{\wedge}.$

$D^{\wedge}.ent$  := Entrada en  $D^{\wedge}.LIGA$  que tiene un apuntador a  $C^{\wedge}$ .  
 $E^{\wedge}.ent$  := Entrada de  $E^{\wedge}.LIGA$  que tiene un apuntador a  $F^{\wedge}$ .  
 $F^{\wedge}.sal$  := Entrada en  $F^{\wedge}.LIGA$  que tiene un apuntador a  $E^{\wedge}$ .

entonces los trazos se modifican en la forma que aparece en la fig. 4.15.b; finalmente marcamos todo el ciclo producido al pegar.

Para marcar ciclos, se introduce una nueva variable booleana llamada "rec" en la variables de tipo TCara, y se va recorriendo el ciclo, poniendo "rec := true".

A continuación seguimos recorriendo el ciclo a partir de la cara que sigue a  $C^{\wedge}.sal$  y continuamos buscando posibles candidatos a pegarse al ciclo construido. Pongamos con todo detalle este algoritmo.

#### ALGORITMO 4.2

Algoritmo para pegar los ciclos que aparecen sobre la superficie de un sólido S partido regularmente, conexo y no simple. (Siempre que se pueda).

##### ENTRADA:

Un apuntador llamado CARA\_REFERENCIA a una variable de tipo TCara que nos conecta a la gráfica de adyacencia de caras de S. Se supone que previamente se ha aplicado el algoritmo 4.1.

##### SALIDA:

El mismo apuntador de la entrada, sólo que ahora se han modificado algunos campos ent y sal de las variables que representan las caras externas, de modo que se reduce el número de ciclos, y en el mejor de los casos se llega a un recorrido para S.

##### ALGORITMO:

1. Se hace  $CP := CARA\_REFERENCIA$ ;
2. Se marca el ciclo al que pertenece la cara CP;
3. MIENTRAS exista CS, cara no marcada vecina a una de las caras del ciclo al que pertenece CP HACER: (CS no debe haberse visitado antes).
  - 3.1  $CP :=$  cara sobre el mismo ciclo de CP vecina a CS;

3.2 CPDes := CP^.LIGA[CP^.sal]; {Cara Posterior a CP}

3.3 CSAnt := CS^.LIGA[CP^.ent]; {Cara Anterior a CS}

A continuación, si es posible se rompen los ciclos y asociados con CP y CS, restituyendo las ligas de manera apropiada. Esto es lo que hace la siguiente sección de código.

3.4.1 SI CPDes es vecina de CSAnt ENTONCES:

```
CP^.sal := entrada en CS^.LIGA que apunta a CP.  
CS^.ent := entrada en CP^.LIGA que apunta a CS.  
CSAnt^.sal := entrada en CPDes^.LIGA que apunta  
a CSAnt.  
CPDes^.ent := entrada en CSAnt^.LIGA que apunta  
a CPDes.
```

CP = CS;

Se marca el ciclo al que pertenece CP.

3.4.2 DE LO CONTRARIO:

```
CP := CP^.LIGA[CP^.sal] {Observe que estamos  
cambiando de ciclo}
```

TERMINA EL ALGORITMO.

//

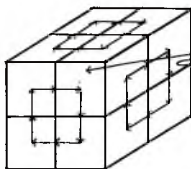
Veamos paso a paso como funciona el algoritmo 4.2 con el sólido mostrado en la fig. 4.12. En adelante le llamaremos "sólido C".

Inicialmente partimos de la gráfica de adyacencia de caras del sólido C y un apuntador a una de sus caras como se muestra en la fig. 4.16.a.

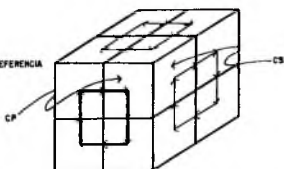
En el paso 1 hacemos CP = CARA\_REFERENCIA y se marca el ciclo al que pertenece CP. La primera cara no marcada que hallamos al seguir el ciclo de CP es CS que se encuentra buscando en las entradas de CP^.LIGA. (Ver fig. 4.16.b). De esta manera entramos a la iteración correspondiente al paso 3.

De acuerdo a los pasos 3.2 y 3.3 encontramos CPDes y CSAnt, que como puede verse en la fig. 4.16.c son vecinas, luego podemos pegar los ciclos correspondientes haciendo los asignamientos de 3.4.1. Al terminar hacemos el asignamiento CP = CS, para obtener la configuración que se ve en la fig. 4.16.d.

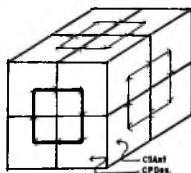
En la siguiente iteración del paso 3, se observan los cambios mostrados en las figuras 4.16.e y 4.16.f. la tercera iteración da como resultado la estructura representada en 4.16.g.



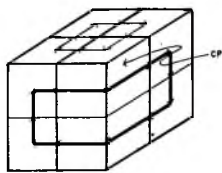
a) TRAZOS PRODUCIDOS POR UNA ASIGNACION DE RECORRIDOS ORTOGONALES AL SOLIDO MOSTRADO.



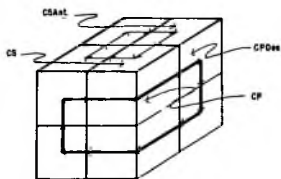
b) CONFIGURACION AL TERMINAR LOS PASOS 1 Y 2.



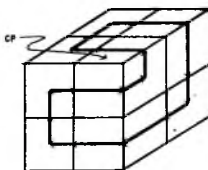
c) EJECUCION DE LOS ASIGNAMIENTOS 3.1, 3.2 Y 3.3.



d) CONFIGURACION AL TERMINAR LA PRIMER ITERACION DEL PASO 3

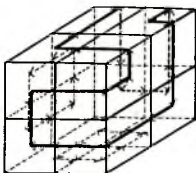


e) EJECUCION DE LOS ASIGNAMIENTOS 3.1, 3.2 Y 3.3 DE LA SEGUNDA ITERACION DEL PASO 3.

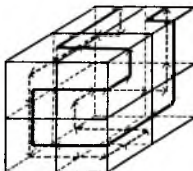


f) CONFIGURACION AL TERMINAR LA SEGUNDA ITERACION DEL PASO 3

FIG. 4.16 APLICACION DEL ALGORITMO 4.2



g) CONFIGURACION AL TERMINAR TRES  
ITERACIONES DEL PASO 3.



h) RECORRIDO FINAL.

FIG. 4.16 APLICACION DEL ALGORITMO 4.2 CONTINUACION.

Si continuamos de esta manera, se obtiene finalmente el recorrido mostrado en 4.16.h.

Sin embargo, la estrategia de pegado del algoritmo 4.2 no es completa, pues existen sólidos como el mostrado en la fig. 4.17 donde hay ciclos que no se pueden pegar a ningún otro; en el caso de la fig. 4.17 el ciclo marcado con línea fuerte no se puede pegar con los otros. Esto se debe a que si recorremos este ciclo, la condición 3.4 es siempre falsa.

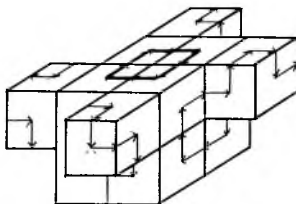


FIG. 4.17 CICLO NO 'PEGABLE' USANDO EL ALGORITMO 4.2

Aun cuando parece muy complicado el problema mostrado en la fig. 4.17, se puede resolver haciendo uso de los recorridos para cubos unitarios del tipo mostrado en la fig. 4.18. Sobre esto se hablará en la próxima sección.

#### 4.7 GENERACION DE RECORRIDOS USANDO UN NUEVO TIPO DE ASIGNACION DE RECORRIDOS A CUBOS UNITARIOS.

El objetivo fundamental de esta sección es encontrar una forma alternativa de asignación de recorridos sobre cubos unitarios para resolver (no totalmente) los problemas vistos en la sección precedente. Veamos este nuevo tipo de recorridos sobre cubos unitarios.

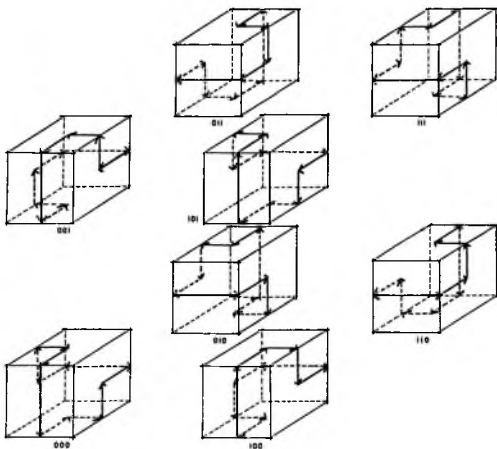


FIG. 4.18 RECORRIDOS SEMIORTOGONALES.

La fig. 4.18 muestra un dibujo que nos recuerda los recorridos ortogonales de la fig. 4.2, la diferencia está en la forma de los trazos sobre las caras de los cubos unitarios, ya que sobre algunas de ellas aparecen trazos rectos. Podríamos ver una caracterización rigurosa de los recorridos mostrados en la fig. 4.18, pero creemos que el lector puede encontrarla fácilmente. Para identificar el tipo de recorridos mostrados en la figura, les asignaremos el nombre de Recorridos

## Semiortogonales.

Lo importante de estos recorridos es que cumplen la misma propiedad de los ortogonales, en cuanto a que los trazos sobre las caras de contacto en los cubos representados en la fig. 4.18 coinciden y tienen sentidos opuestos. Como recordará el lector ésto fué suficiente para recorrer sólidos simples. Así, del mismo modo como se usaron funciones de asignamiento de recorridos ortogonales también pueden usarse recorridos semiortogonales. La proposición 4.7 puede demostrarse usando recorridos semiortogonales.

Dejemos por el momento pendientes los recorridos semiortogonales para definir algo que se conjuga con éstos para determinar recorridos.

### DEFINICION 4.9

Sea S un sólido partido regularmente conexo.

- 1.- Se dice que C es un cubo libre en S, si 5 o más caras de C son caras externas en S.
- 2.- La parte no simple de S, es el sólido S' que se obtiene al aplicar el algoritmo siguiente:

S' := S.

MIENTRAS QUE S' tenga cubos libres HACER:

S' := S' - {c | c es un cubo libre en S'}

TERMINA.

//

El lector puede verificar fácilmente que la parte no simple de un sólido simple es el vacío. También puede ver que la parte no simple del sólido de la fig. 4.17 es la mostrada en 4.12. A continuación se muestra el porqué es importante considerar la parte no simple de un sólido.

### PROPOSICION 4.9

Sea S un sólido partido regularmente conexo. Si la parte no simple de S es recorrible, entonces S es recorrible.

### DEMOSTRACION

La demostración es por inducción sobre el número de iteraciones del algoritmo de la definición 4.9.2 para hallar la parte no simple S' de S.

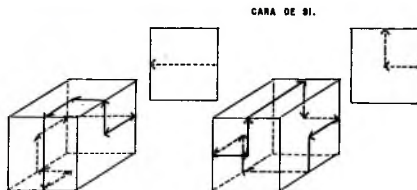
CASO BASE: 0 iteraciones.

En este caso nada hay que probar pues  $S' = S$ .

**CASO INDUCTIVO:** Supongamos que se vale para  $k$  iteraciones, pero para encontrar  $S'$  necesitamos  $k+1$  iteraciones.

Si para hallar  $S'$  necesitamos  $k+1$  iteraciones, sea  $S_1$  el sólido obtenido después de una iteración, como el algoritmo para hallar la parte no simple de  $S_1$  tiene  $k$  iteraciones,  $S_1$  es recorrible.

Sean  $C_1, \dots, C_n$  los cubos libres en  $S$  que se eliminaron en la primera iteración. Los trazos en la cara externa de  $S_1$ , de contacto entre  $C_1$  y  $S_1$  vista de frente, puede tener alguna de las formas que se muestra en la fig. 4.19, para cada caso se da un recorrido semiortogonal que se puede usar para obtener un recorrido sobre el sólido  $S_1 \cup C_1$ . Lo mismo vale para  $C_2, \dots, C_n$ . Es decir se pueden incorporar a  $S_1$  los cubos eliminados en la primer iteración, logrando finalmente un recorrido para  $S$ .



CUBOS ELIMINADOS EN  
LA PRIMER ITERACION.

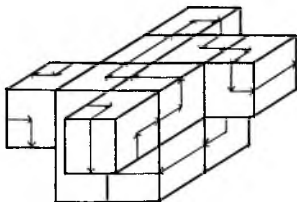
SOLO CONSIDERAMOS ESTE TIPO DE TRAZOS PARA LAS CARAS  
DE  $S_1$  PUES SALVO ROTACIONES SON LAS UNICAS QUE SE  
PRESENTAN.

**FIG. 4.19 POSIBLES TRAZOS SOBRE LAS CARAS DE  $S_1$**

En la fig. 4.20 se muestra una aplicación de esta proposición para recorrer el sólido de la fig. 4.17. Los trazos en 4.20 no muestran de manera muy clara el recorrido, para entender mejor como están, si se quitan los "chipotes" se verá el



recorrido en 4.16.hj; los demás cubos se recorren usando recorridos semiortogonales apropiados.



**FIG. 4.20 SOLUCION AL PROBLEMA DE LA FIG. 4.17**

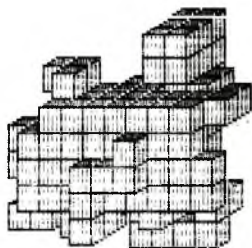
#### 4. CONCLUSIONES

En este capítulo hemos estudiado una estrategia que pretende encontrar recorridos sobre sólidos conexos partidos regularmente. Los algoritmos se basan principalmente en la gráfica de adyacencias de cara del sólido y los recorridos ortogonales.

Desgraciadamente en muchos casos el algoritmo 4.1 genera un recorrido por ciclos para sólidos. En algunos casos es posible solucionar este problema aplicando el algoritmo 4.2 al resultado de 4.1, sin embargo siguen existiendo sólidos que no se pueden recorrer así.

La razón por la que un sólido no se puede recorrer mediante la combinación de los algoritmos 4.1 y 4.2, es que puede tener muchos "chipotes" o dicho de mejor manera muchos cubos libres. Este se puede solucionar en alguna medida si se elimina la parte simple de un sólido y se recorre lo que quede. Sin embargo siguen existiendo sólidos no recorribles (trate el lector de imaginar un ejemplo).

Actualmente contamos con estrategias de pegado de ciclos que permiten recorrer sólidos no considerados en los algoritmos presentados, desgraciadamente tienen una complejidad computacional alta y su formalización es difícil de ponerse. Por otro lado los algoritmos dados se comportan muy bien en la práctica pues para los sólidos que hemos probado siempre se han encontrado recorridos. Hemos construido sólidos que no se pueden recorrer, pero no creemos que sea tan malo tener un recorrido por ciclos, siempre y cuando el número de éstos no sea muy grande.



## RESULTADOS Y APLICACIONES

Los algoritmos discutidos en los capítulos III y IV, se programaron en Pascal. En este capítulo mostramos los resultados obtenidos y damos aplicaciones de los recorridos sobre sólidos.

Para ver si los algoritmos del capítulo IV son aceptables para hallar recorridos, se probaron aplicándolos a sólidos generados aleatoriamente. Consideramos que la generación es de interés, por lo que se presenta en la primer sección del capítulo.

### 5.1 ALGORITMO PARA LA GENERACION ALEATORIA DE SOLIDOS PARTIDOS REGULARMENTE Y CONEXOS.

El objetivo de este algoritmo es encontrar aleatoriamente la representación matricial A de un sólido S conexo partido regularmente. La idea se explica a continuación:

Se supone que A tiene dimensión  $N \times N \times N$ , donde N es algún número natural positivo.

Se parte de una representación matricial correspondiente a un cubo unitario C colocado en el centro de A. Luego se genera un número aleatorio entre 0 y 5, que indica donde se puede poner un cubo vecino a C de acuerdo a lo siguiente:

VALOR DE NUMERO GENERADO	EL VECINO SE PONE
0	A la DERECHA
1	ARRIBA
2	ADELANTE
3	A LA IZQUIERDA
4	ABAJO
5	ATRAS

se repite este paso de generación de vecinos de C un número de veces al que llamaremos  $N\_INTENTOS(0)$ .

Para distinguir el momento en que apareció cada cubo durante la generación, asociamos a cada uno un número al que llamaremos nivel, que corresponde al orden en que fué generado. Este número se define para C como cero y para los vecinos de C hasta ahora generados como 1.

Para seguir formando el sólido, tomamos los vecinos de C y para cada uno de ellos les generamos vecinos de manera similar

a la usada con C. Estos nuevos vecinos tienen por definición el nivel 2. El número de intentos utilizados para encontrar los vecinos de los cubos en el nivel 1 es en todos los casos un mismo número no negativo al que llamaremos  $N\_INTENTOS(1)$ .

De manera análoga se usan números no negativos  $N\_INTENTOS(2)$ ,  $N\_INTENTOS(3)$  etc que indican el número de intentos permisibles para tratar de generar vecinos de los cubos en los niveles 2, 3, etc.

Podemos pensar que  $N\_INTENTOS$  es una función que va de los naturales positivos en los naturales. Para garantizar que la generación del sólido en algún momento va a terminar, pedimos que  $N\_INTENTOS$  sea no creciente y que exista un nivel  $k$  tal que  $N\_INTENTOS(k) = 0$ .

La fig. 5.1 muestra una función decreciente que tiende a cero, de manera que si tomamos la restricción de ésta en los naturales positivos obtendremos una forma apropiada para  $N\_INTENTOS$ . Obsérvese que a medida que sea más lento el decrecimiento de esta función, lograremos sólidos más grandes. En nuestro algoritmo optamos por la función:

$$f(x) = (1 - (x - 0.9) / (x + 1)) * 10$$

que se comporta de la manera mostrada en la fig. 5.1. La restricción de esta función a los naturales puede determinarse como:

$$F\_INTENTOS(x) = \text{trunc}(F(x))$$

que será usada en el algoritmo 5.2.

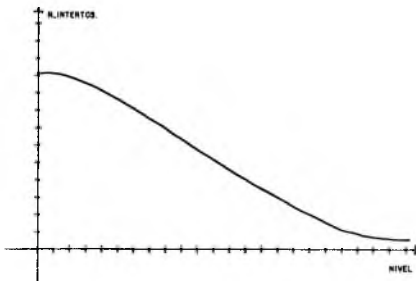


FIG. 5.1 FORMA TÍPICA DE LA FUNCIÓN  $N\_INTENTOS$ .

A continuación se da con todo detalle este algoritmo.

## ALGORITMO 5.2

### GENERACION ALEATORIA DE SOLIDOS

#### ENTRADA:

Una A matriz de tipo TMatriz. Se supone que al entrar contiene al sólido vacío.

#### SALIDA:

La misma matriz de la entrada, sólo que ahora contiene la representación de un sólido partido regularmente conexo.

#### OBSERVACIONES SOBRE EL ALGORITMO.

Cuando se genera un vecino V para algún cubo, se revisa que V no esté ya en el sólido. Si está ignoramos a V, pero si no entonces ponemos la entrada correspondiente de A en TRUE y guardamos en una pila las coordenadas de V, así como el nivel en que fué generado. A esta pila le llamaremos PILA y solo aparece en el algoritmo a través de dos procedimientos denominados METE y SACA que tienen el sentido usual en el manejo de pilas.

Los procedimientos METE y SACA tienen la forma:

```
METE(PILA,x,y,z,nivel); SACA(PILA,x,y,z,nivel)
```

donde: (x,y,z) son las coordenadas del cubo  
nivel es el nivel que le corresponde al cubo.

La dimensión de A es MaxNCubos x MaxNCubos x MaxNCubos.

Usamos una función llamada RANDOM(n), que genera números aleatorios entre 1 y n.

#### ALGORITMO.

Empezamos generando las coordenadas del cubo inicial y lo colocamos en PILA para iniciar el algoritmo. (x,y,z) son las coordenadas del cubo.

```
x := MaxNCubos div 2; y := x, z := x;  
METE(PILA,x,y,z,1);
```

REPITE

```

SACA(PILA,x,y,z,nivel);

N_Intentos := F_INTENTOS(nivel);

PARA I := 1 HASTA N_Intentos HACER:

    NCara := Random(6) - 1;

    SI NCARA > 2 HACER suma := -1 DE LO CONTRARIO suma
                                     := 1;

    SEGUN EL VALOR DE NCARA HACER:

        0: METE(PILA,x+suma,y,z,nivel+1);
        1: METE(PILA,x,y+suma,z,nivel+1);
        2: METE(PILA,x,y,z+suma,nivel+1)

HASTA QUE PILA estè vacía.

TERMINA EL ALGORITMO.

```

## 5.2 RESULTADOS DE LOS ALGORITMOS EXPUESTOS EN EL CAPITULO ANTERIOR

Como ya se dijo, se programaron en Pascal los algoritmos para determinar la gráfica de adyacencia de caras de un sólido y el algoritmo para encontrar recorridos ortogonales. También se programó el algoritmo para generar sólidos partidos regularmente conexos y se probaron de acuerdo a lo siguiente.

1. Al generar aleatoriamente un sólido se encuentra su parte no simple y trabajamos con ella. Esto se hace porque buscamos primero un recorrido sobre su parte no simple y si se encuentra, reincorporamos los cubos que falten de la forma explicada en la proposición 4.9.
2. Los algoritmos se programaron con la generalidad suficiente para que la asignación de recorridos ortogonales pueda elegirse arbitrariamente.
3. Se programaron también asignaciones de recorridos semiortogonales.
4. Se hizo una búsqueda exhaustiva de recorridos, agotando los asignamientos de recorridos ortogonales y semiortogonales hasta encontrar un recorrido.
5. Una vez determinado un recorrido para la parte no simple de un sólido, se "pegaron" los cubos eliminados.

El resultado obtenido fuè por demàs satisfactorio, en ningùn caso se obtuvo un sólido que no fuera recorrible. Por esta razón, aun cuando contamos con estrategias mäs complicadas de pegado de

recorridos, creemos que no es necesario complicar los algoritmos.

Los listados de estos programas no se dan por ser largos.

### 5.3 APLICACIONES

#### 1. GENERACION DE CADENAS PARA REPRESENTAR EN MEMORIA SECUNDARIA LA SUPERFICIE DE UN SOLIDO.

Esto es el equivalente a las cadenas de Freeman, sólo que en tres dimensiones. Estas cadenas tienen la forma:

$$k_0, \dots, k_n,$$

donde  $n$  es el número de caras sobre el sólido. Cada elemento de esta cadena se asocia con una cara externa; por supuesto se debe guardar la referencia coordenada de la cara a la que se asocia  $k_0$ . A diferencia de los elementos en una sucesión hamiltoniana,  $k_i$  es una referencia relativa respecto a  $k_{i-1}$ , para toda  $i$  entre 1 y  $n$ . Veamos con todo detalle esto:

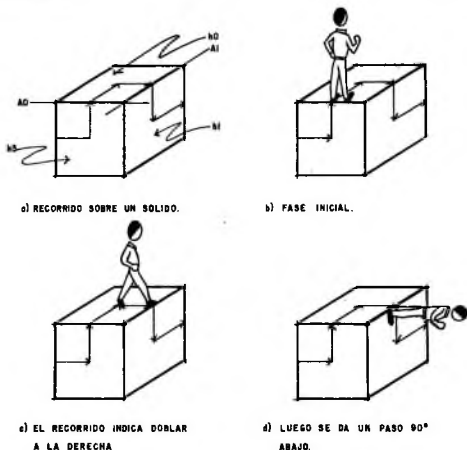


FIG. 5.2 GENERACION DE CADENAS



En la fig. 5.2.a, se muestra un sólido S y un recorrido (C,H) para S con  $H = h_0, h_1, \dots, h_{n-1}$ .

La generación de la cadena empieza parando un observador sobre la cara  $h_0$  junto a la arista de adyacencia  $A_0$  entre  $h_0$  y  $h_{n-1}$ , dando la espalda a la cara  $h_{n-1}$  como se muestra en 5.2.b. La idea es guardar en  $k_0$  la dirección en que tendría que moverse el observador para que caminado sobre la cara  $h_0$  llegue a la cara  $h_1$  y se pare nuevamente junto a la arista  $A_1$  de contacto entre  $h_0$  y  $h_1$ , dando la espalda a la cara  $h_0$  (suponemos una trayectoria mínima).

Al inicio de su movimiento el observador tiene tres opciones para caminar sobre la cara  $h_0$  para llegar a la arista  $A_1$ : A la derecha, adelante y atrás. En el recorrido mostrado en 5.2.a debe moverse a la derecha (véase la fig. 5.2.c). Lo único que le falta es pararse en  $h_1$ , para esto tiene que dar un paso ya sea 90 grados abajo, de frente o 90 grados arriba, (ver en la fig. 5.2.d que el recorrido indica ir hacia abajo).

La dirección de movimiento del observador tiene 3\*3 posibilidades y depende de la forma del recorrido usado, 4 bits son suficientes para almacenar esta información. En los dos primeros se guarda si el movimiento fue a la derecha, adelante o a la izquierda, y en los otros dos se guarda si el movimiento para llegar a  $h_1$  fue abajo, de frente o arriba, de acuerdo al código:

Primeros dos bits:

DERECHA	-----	00
ADELANTE	-----	01
IZQUIERDA	-----	10

A estos bits les llamaremos  
CODIGO DE TRAYECTO.

Siguientes dos bits:

ABAJO	-----	00
DE FRENTE	-----	01
ARRIBA	-----	10

A estos bits les llamaremos  
CODIGO DE DOBLADO.

(Los bits los contamos de izquierda a derecha).

De esta manera se guarda la dirección de movimiento del caminante en  $k_0$ , de manera análoga se construyen  $k_1, k_2, \dots, k_{n-1}$ .

Definamos rigurosamente las cadenas.

## DEFINICION 5.1

Sean  $S$  un sólido partido regularmente y conexo, y  $R = (C, H)$  un recorrido sobre  $S$  con  $H = h_0, h_1, \dots, h_{n-1}$ . La cadena de recorrido para  $S$  según  $R$  denotada por  $C_R$ , es una triada de la forma  $C_R = (C, T, K)$ , donde  $C$  es la misma  $C$  de  $R$ ,  $K$  es una sucesión de la forma  $K = k_0, k_1, \dots, k_{n-1}$  construida de la forma indicada arriba y  $T$  es el tipo de cara asociado con  $h_0$ .

//

Para ejemplificar estas cadenas, la fig. 5.3 muestra un sólido  $S$  y un recorrido sobre su superficie. La sucesión  $K$  de la cadena asociada al recorrido es:

$k_0 = 0000$	$k_{15} = 1000$
$k_1 = 1010$	$k_{16} = 0010$
$k_2 = 0000$	$k_{17} = 1000$
$k_3 = 1000$	$k_{18} = 0000$
$k_4 = 0000$	$k_{19} = 1000$
$k_5 = 1000$	$k_{20} = 0000$
$k_6 = 0010$	$k_{21} = 1010$
$k_7 = 1000$	$k_{22} = 0000$
$k_8 = 0000$	$k_{23} = 1000$
$k_9 = 1000$	$k_{24} = 0000$
$k_{10} = 0000$	$k_{25} = 1000$
$k_{11} = 1010$	$k_{26} = 0010$
$k_{12} = 0000$	$k_{27} = 1000$
$k_{13} = 1000$	$k_{28} = 0010$
$k_{14} = 0000$	$k_{29} = 1000$

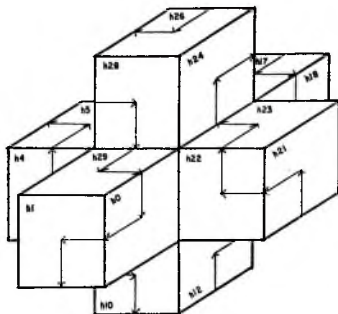
El lector puede observar que  $k_0 = k_{10} = k_{20}$ ;  $k_1 = k_{11} = k_{21}$ ; ...  $k_9 = k_{19} = k_{29}$ . Esto se puede usar para escribir de manera compacta la cadena. (Usaremos notación de listas):

$3(0000, 1000, 0000, 1010, 0000, 1000, 0000, 1000, 0010, 1000)$

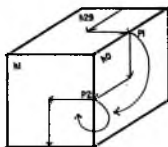
Recordemos que estas cadenas tienen un caracter cíclico, de manera que si hubieramos tomado como cara inicial del recorrido a  $h_{29}$ , entonces se podría comprimir más el código como:

$3(2(1000, 0000), 1010, 2(0000, 1000), 0010)$

Podría hablarse de algoritmos sobre cómo compactar cadenas pero esto no lo tocamos aquí.



e) RECORRIDO SOBRE UN SOLIDO.

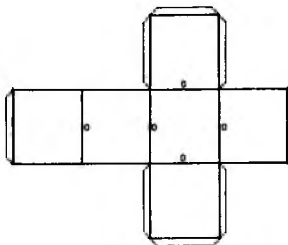


b) PARA LLEGAR DE P1 A P2 SE DOBLA A LA DERECHA Y DESPUES ABAJO.

FIG. 5.3 GENERACION DE CADENAS.

## 2. DESDOBLADO DE SÓLIDOS

Una forma de construir un cubo usando cartulina y pegamento es hacer los trazos mostrados en la fig. 5.4, se corta sobre su contorno, se dobla sobre cada una de las aristas en dirección apropiada y finalmente se pone pegamento sobre las pestañas y se arma el cubo. Nuestro objetivo en esta sección es motivar cómo un recorrido sobre un sólido  $S$ , genera de manera natural un esquema equivalente para construir con cartulina ó otro material laminado el sólido  $S$ .

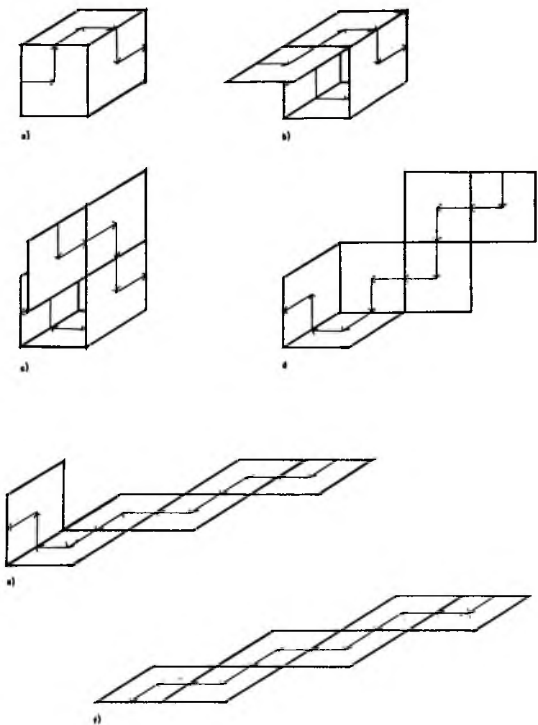


**FIG. 5.4 MODELO PARA CONSTRUIR UN CUBO.**

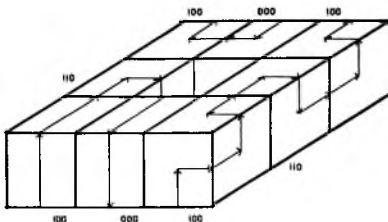
Se puede explicar formalmente en que consiste el problema de desdoblar un sólido siguiendo un recorrido dado, pero creemos que es más ilustrativo que el lector siga los dibujos de la fig. 5.5. Obsérvese que esto no da un esquema para la construcción de un sólido con cartulina diferente al mostrado al de la fig. 5.3.

Sin embargo hay un pequeño problema si queremos usar esto empleando cartulina. La fig. 5.6 muestra un toro y un recorrido asociado. Invitamos al lector a que desdoble este sólido utilizando el método de la fig. 5.5; encontrará que hay dos cuadros correspondientes a caras distintas sobre el toro que estrictamente deben ocupar la misma posición en la cartulina. Pensamos que esto se encuentra muy ligado al orden topológico de un sólido pero no lo hemos estudiado.

Con todo y el problema manifestado en la fig. 5.6 creemos que el desdoblado inducido por un recorrido puede ser de utilidad para la modelación de sólidos colocando cara por cara de manera automática (por un robot).



**FIG. 5.5 DESDOBLAMIENTO DE UN CUBO UNITARIO.**



**FIG. 5.6 SÓLIDO CON IRREGULARIDADES AL DESDOBLARSE**

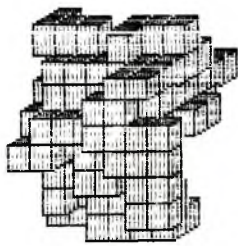
El lector no encontrará difícil ver que una vez hallada la cadena de un sólido según algún recorrido es muy simple determinar el desdoblamiento asociado.

#### 5.4 CONCLUSIONES

La importancia de los algoritmos presentados se hace patente no sólo desde el punto de vista teórico sino por la utilidad que puedan tener en todas aquellas aplicaciones que requieran representar linealmente la frontera de un sólido. Pueden servir para almacenar sólidos tridimensionales, para transmitir sólidos por una línea de comunicación, para modelar sólidos por medios mecánicos, etc. Seguramente el lector encontrará algún tipo de aplicación a la forma de recorridos presentados. Queda como problema abierto la demostración de la hipótesis de que todo sólido partido regularmente conexo es recorrible; si bien no logramos demostrarla, tampoco podemos rebatirla.

Lo que el lector debe considerar como más importante en este trabajo es:

- a) El algoritmo para encontrar la gráfica de adyacencias de un sólido.
- b) Los recorridos ortogonales.
- c) El algoritmo para fusionar recorridos.
- d) El algoritmo para pegar ciclos.






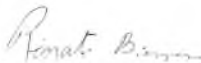
## REFERENCIAS.

1. **Computer Processing On Line Drawing Images.**  
Herbert Freeman.  
ACM. 1974.
2. **Sammet Computing Surveys.**  
Sammet H.  
"Quad Tree from Binary Arrays"  
Com. Graphics & Image Processing Vol. 13 #1. June-1984
3. **On the Performance Evaluation of Extendible Hashing and  
Trie Searching.**  
Philippe Flajolet.  
Acta Informatica 20.3 345 469(1983).

El jurado designado por la Sección de Computación del Departamento de Ingeniería Eléctrica del Centro de Investigación y de Estudios Avanzados del I.P.N., aprobó esta tesis el 14 de abril de 1987.



Dr. Mike Porter K.



Dr. Renato Barrera



Dr. Guillermo Morales Luna

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL  
INSTITUTO POLITÉCNICO NACIONAL

**BIBLIOTECA DE INGENIERIA ELECTRICA**  
FECHA DE DEVOLUCION

*El lector está obligado a devolver este libro  
antes del vencimiento de préstamo señalado  
por el último sello.*

29 OCT. 1987

30 SET. 1988

14 DIC. 1988

05 ENE. 1989

10 AGO. 1994

9 DIC. 1994

13 MAR. 1985

14 AGO. 1996

~~5 SET. 1996~~

22 AGO. 1996

DEVOLUCION

