UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

A C A T L A N

MODELOS SINTACTICOS PARA EL RECONOCIMIENTO E INTERPRETACION DE ELECTROCARDIOGRAMAS

T E S I S

QUE PARA OBTENER EL TITULO DE:

LICENCIADO EN MATEMATICAS APLICADAS Y COMPUTACION

P R E S E N T A :

VICTOR MANUEL VELAZQUEZ ESTUDILLO

ASESOR: SERGIO V. CHAPA VERGARA

STA. CRUZ ACATLAN, MEXICO 1991

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

APENDICE.

INTRODUCCION.

A pesar del desarrollo y utilización que la computación está generando en nuestro país, son muchas las áreas en las que todavía no se implementa su uso, reduciendo así las ventajas que esta tecnología representaría en las diferentes áreas de la producción, tanto de bienes como de servicios. Son muchos los factores que determinan las limitaciones que México tiene para incorporarse al sistema computarizado, entre ellas principalmente el aspecto económico y la dependencia a nivel tecnológico.

En la medicina por ejemplo, existen diferentes especialidades en las que utilizan sistemas por computadora, tal es el caso de la medicina deportiva o en la de traumatología por citar algunas. Pero hay otros aspectos donde no se utilizan y su uso representaría beneficios, en un área tan importante como lo la salud, un caso específico es el de la cardiología.

La lectura e interpretación de los electrocardiogramas por su grado de dificultad, sólo puede realizarse por personas especializadas y de manera manual. Esto depende en la rapidez con que se pueda desde el tomar el electrocardiograma, su interpretación y diagnóstico; lo cual para un paciente con cardiopatía grave resulta muy peligroso.

Por lo que, se sugiere un sistema que conste de los siguientes procesos: señal electrocardiográfica, codificador, reconocedor e interpretador y analizador de cadena; de estos solamente se trabajará sobre el reconocedor e interpretador siendo el objetivo de esta tesis. Representando con ello un avance que servirá para posteriores investigaciones, como son el generar un mecanismo de comunicación entre el ECG y la computadora, etc.

Dentro del área de Inteligencia Artificial, el Reconocimiento de Patrones, "Pattern Recognition", se aplica en diferentes disciplinas como: geografía, biología, ingeniería, robótica, etc.; también puede aplicarse en la medicina para la elaboración de diagnósticos con base en electrocardiogramas.

El Reconocimiento de Patrones se constituye de dos procesos fundamentales: la clasificación, que permite el reconocimiento de planos y el

análisis sintáctico que examina una cadena de caracteres y determina si cumple con los requisitos de un problema dado.

El "Reconocimiento de Patrones" es un método de análisis sintáctico que contiene cuatro modelos de interpretación (autómata finito, Horowitz, Stockman y autómata estocástico), que brevemente se esbozarán, pues se analiza con más detalle en el desarrollo de esta tesis, ya que de ellos se retomarán los elementos para la elaboración del sistema reconocedor electrocardiográfico.

De estos modelos, los dos primeros utilizan expresiones regulares, el tercero lenguaje de contexto libre y el último el análisis probabilístico.

En el análisis probabilístico se realiza una simulación del funcionamiento del corazón en la producción del estímulo y genera el electrocardiograma correspondiente, además de una cadena de caracteres que sirve para determinar si el estado del corazón es normal o anormal. Mientras los modelos de autómata finito, Horowitz y Stockman, realizan un análisis a la cadena generada por el electrocardiograma, para indicar si ésta es reconocida y aceptada.

La metodología que seguiremos para el análisis de estos modelos está estructurada partiendo del más simple al más complejo, por ello se presentará en el desarrollo del trabajo de la siguiente manera: Autómata Finito, Autómata Estocástico, el modelo de Horowitz y el de Stockman.

1) Modelo de expresión regular con alfabeto {0, 1}. Esta representación hace una aproximación a la forma de onda del electrocardiograma, el reconocedor es un programa basado en Autómata Finito cuya expresión regular representa la señal del electrocardiograma. El reconocedor identifica si la cadena es o no la correcta.

2) Interpretación con Autómata Estocástico de Estados Finitos. Utiliza tres estados que simulan el funcionamiento del corazón. La forma de descriminar está basada en una búsqueda de árbol que contiene los estados y probabilidades de transición. La salida de un estado tiene asociado un caracter que será adicionado a la cadena. Además incluye un procedimiento de ruido a la señal de aquellos músculos que no pertenecen al corazón.

3) Explicación mediante el modelo de Horowitz, utiliza una gramática de expresión regular con alfabeto {n, p, 0}. Con este modelo se trata de hacer una aproximación a una función lineal por piezas, donde analiza la onda en

base a pendientes. El análisis que realiza permitirá adicionar un caracter del alfabeto a la cadena, si la pendiente es positiva, será p, si es negativa n y 0 si no tiene pendiente. El reconocedor es un Autómata Finito, pero la cadena electrocardiográficas también pueden ser analizados con una gramática de contexto libre.

4) Interpretación por el modelo de Stockman. En este modelo se lleva a cabo el reconocimiento por medio de una gramática de contexto libre. Basa sus elementos en la partición de la onda electrocardiográfica, donde cada uno de los elementos del complejo son un segmento con una longitud similar.

La contribución de este modelo es que considera un alfabeto de 12 producciones con lo cual logra tenerse mayor representación de las ondas para su interpretación.

De los modelos analizados el de Stockman es el más completo ya que la cadena que genera es pequeña y con mayor similitud a la onda electrocardiográfica en comparación con los otros modelos. Por lo que su lectura e interpretación suele ser más completa y sencilla.

Concluyendose que para realizar el diagnóstico de manera eficaz será necesario incluir en una base de datos las reglas que lo determinen y las cadenas cardiopáticas normales y anormales más comunes, con lo que se construirá un código que permita decodificar y clasificar la cadena electrocardiográfica que un paciente presente. Si la cadena cardiopatica de un paciente no se encuentra será analizada por el proceso analizador de cadenas para obtener su diagnóstico y después anexarla a la base de datos para utilizarla en casos posteriores.

CAPITULO 1.
PLANTEAMIENTO DEL PROBLEMA

1.1 INTERPRETACION DE ELECTROCARDIOGRAMAS

El corazón, que forma parte del aparato circulatorio junto con los vasos sanguineos, es un organo hueco de paredes musculares, se encuentra localizado entre los pulmones, en la parte anterior y media de la cavidad torácica, un poco a la izquierda. Interiormente tiene cuatro cavidades, dos superiores llamadas aurículas (derecho e izquierdo) y dos inferiores llamados ventrículos (derecho e izquierdo). Existe un tabique medio que separa las cavidades derecha e izquierda, la figura 1.1 muestra el esquema completo del corazón.

Cada aurícula se comunica con el ventrículo del mismo lado por un orificio llamado "Auriculoventricular", donde existen unas válvulas. La válvula del orificio auriculoventricular derecho tiene tres valvas y recibe el nombre de tricúspide. La del lado izquierdo sólo dos valvas y es llamado válvula mitral.

La pared de las aurículas es delgada y la de los ventrículos es gruesa, mucho más el ventrículo izquierdo que derecho.

El corazón está formado por tres capas: endocardio, miocardio y pericardio. El endiocardio es simplemente el epitelio plano que recibe interiormente todo el sistema circulatorio. El miocardio es la capa más gruesa del tejido musculoso. El pericardio es una membrana serosa, parecidas a las pleuras y al periteneo de la cavidad abdominal, que recubre exteriormente el músculo cardiaco.

En el ventrículo izquierdo tiene su origen la arteria Aorta. Durante su largo trayecto, la aorta emite muchas ramas colaterales, destinadas a la cabeza, extremidades superiores y a las vísceras y paredes de las cavidades torácica y abdominal. Del ventrículo derecho sale la arteria pulmonar.

A la aurícula izquierda avocan las cuatro venas pulmonares, dos precedentes de cada pulmón. A la aurícula derecha llegan las dos venas cavas, superior e inferior. La vena cava superior lleva al corazón la sangre procedente de la mitad superior del cuerpo y la cava inferior la procedente de la mitad inferior.

El corazón es el motor del aparato circulatorio, cuando éste se contrae se realiza el bombeo de sangre. Los vasos sanguíneos (arterias, capilares y venas), son los conductos por los cuales circula la sangre.

El corazón se contrae de una forma ritmica. La fase de contracción del músculo cardiaco es llamado "Sistole"; y la relajación es llamada "Diástole". La contracción auricular precede a la ventricular. La sangre siempre circula en el mismo sentido.

Durante la diástole ventricular la sangre pasa de las aurículas a los ventrículos a través de las válvulas mitral y tricúspide.

El aumento de presión que produce en los ventrículos, en el sístole ventricular, cierra las válvulas mitral y tricúspide y abre las válvulas sigmoidales, ocasionando que la sangre sea expelida al sistema arterial. Finalizada la sístole ventricular, sucede una nueva diástole. En la Fig. 1.1, se muestra el esquema del corazón indicandose las partes esenciales que la componen.

El ElectroCardioGrama, definido con las siglas *ECG*, es un registro muy útil de la función del corazón (actividad eléctrica), éste registra los impulsos eléctricos que estimulan al corazón y producen su contracción, también suministra información útil acerca del corazón durante las fases de reposo y recuperación.

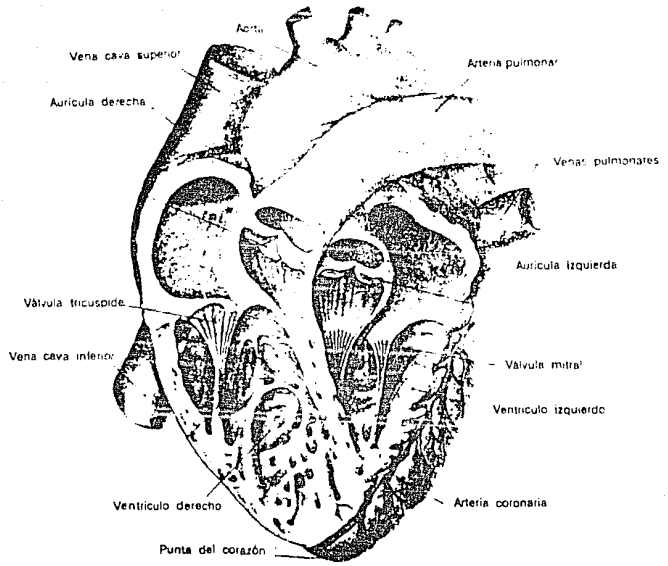


Fig. 1.1, Esquema del corazón.

La contracción de las aurículas y la activación de los ventrículos que produce el corazón y el relajamiento del mismo genera una onda producida en el ECG, fig. 1.2, con segmentos p, pq, qrs, st y t.

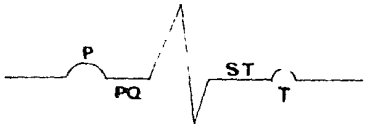


Fig. 1.2 Complejo generado por un ECG.

La onda P es producida con la contracción de las aurículas derecha e izquierda, fig. 1.3 . Esta contracción es estimulada por un impulso eléctrico que inicia en el nodo SA

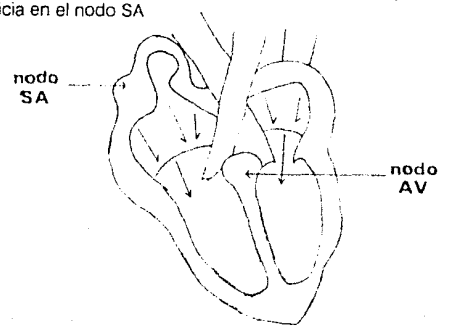


Fig. 1.3 Contracción auricular.

Después el impulso llega al nodo AV, donde ocurre una pausa de 1/10 segundo, que permite que la sangre llegue a los ventrículos. En seguida, el nodo AV es estimulado e inicia un impulso eléctrico que se dirige por el haz de his y las ramas del mismo, pasando por las fibras de Purkinje, a las células del Miocardio, produciendo la contracción simultánea de los ventrículos.

El complejo QRS representa la actividad eléctrica de la contracción ventricular, fig 1.4 muestra la activación. La onda Q siempre será la primera en el complejo. En un caso normal, la onda Q siempre va hacia abajo, aunque es muy común la falta de ésta. La deflexión hacia arriba siempre es la onda R y se encuentra después la onda S, ver fig. 1.2.

Una vez generado el segmento QRS se tiene una pausa ST, seguida de una onda T que es el reposo de los ventrículos y permitirá que pueda volver a estimularse.

Las señales tienen una línea media imaginaria, las cuales cuando tienen deflexión hacia arriba se consideran positivas, y si la deflexión es hacia abajo son negativas.

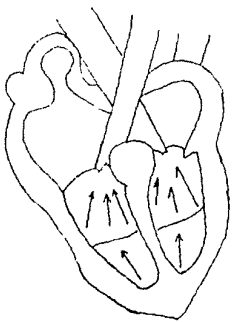


Fig. 1.4 Contracción ventricular.

1.1.1 EL PAPEL ELECTROCARDIOGRAFICO.

Se trata de un papel milimétrico cuadriculado, como se observa en la fig. 1.5. La calibración del aparato se hace de tal manera que 1 milivoltio equivale a 1 cm.

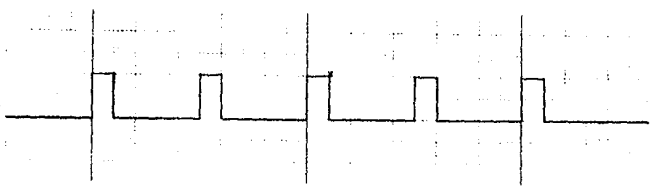


Fig. 1.5 Papel electrocardiográfico, se muestra la calibración.

La velocidad de transporte del papel es de 25 mm/seg., por lo tanto en cada segundo se recorren (25 mm.).

El objetivo es que el especialista pueda interpretar el funcionamiento del corazón. Por ejemplo, todo segmento del electrocardiograma en un corazón normal debe cumplir con las especificaciones siguientes: La deflexión hacia arriba, no debe ser mayor de 3 cuadritos y las que van hacia abajo no mayores de 2 cuadritos. El ancho del complejo QRS no debe ser mayor de 3 cuadritos.

1.1.2 DERIVACIONES CONTENIDAS EN UN ELECTROCARDIOGRAMA.

El ECG ordinario consta de 2 derivaciones distintas que permiten que el corazón sea observado en varias direcciones y su análisis sea más completo, las derivaciones están divididas en:

- Derivaciones de miembros. Estas son consideradas como muestras estandar y se identifican como: DI, DII, DII, AVR, AVF, AVL.

- Derivaciones de precordiales. Cubren la imagen anatómica del corazón sobre la pared torácica. Son conocidas como V, V2, V3, V4, V5 Y V6.

DERIVACIONES DE MIEMBRO.

Cada derivación unipolar es como una ventana que vé diferentes partes del corazón, fig. 1.6.

VR - Vé el interior de la Aurícula y Ventrículos derechos.
VL - Vé la pared libre del ventrículo izquierdo.
VF - Vé la cara diafragmática del corazón.

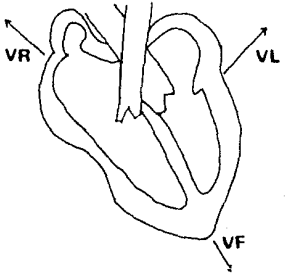


Fig. 1.6 Dirección de observación de las derivaciones de miembro unipolar.

Para obtener las derivaciones DI, DII, DIII, los electrodos envían impulsos eléctricos que generan la señal, como se muestra en la Fig. 1.8. Para la derivación DI, el brazo izquierdo envía un impulso eléctrico negativo, y el brazo derecho un impulso positivo. Para la derivación DII, el brazo izquierdo envía un impulso eléctrico negativo, y el eléctrodo localizado en alguno de los miembros inferiores un impulso positivo.

Para la derivación DIII, el brazo derecho envía un impulso eléctrico negativo, y en alguno de los miembros inferiores un impulso positivo.

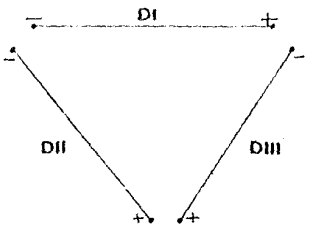


Fig. 1.8 Tipo de impulso eléctrico que manda el electródo para obtener la derivación.

Las restantes derivaciones de miembros AVR, AVL y AVF, se caracterizan por envíar un electródo impulsos positivos y el resto de los electródos forman una tierra común negativa.

Para tomar la derivación AVR, el electrodo localizado en el miembro superior derecho enviará un impulso eléctrico positivo. Para la derivación AVL, el electrodo localizado en el brazo izquierdo enviará el impulso positivo . Para tomar la derivación AVF, el electrodo localizado en alguno de los miembros inferiores enviará el impulso positivo.

Tabla 1.1, Observaciones.

Derivacion	Observaciones
V1	Observa el ventrículo derecho, la parte alta del Septum ventricular y las dos Aurículas.
V2	Observa el ventrículo derecho y el Septum Interventricular
V3	Se encuentra frecuentemente en la transición de ambos ventrículos y observa parte de la cara inferior del ventrículo izq.
V4	Observa la cara anterior del ventrículo izquierdo.
V5 y V6	Analizan la cara lateral del Ventrículo izquierdo

La forma de onda que produce un ECG normal para las derivaciones precordiales se muestran en la Tabla 1.2.

Estas ondas pueden tener un cambio mínimo en su forma, si ésto ocurre no existe anomalia en el corazón.

Las derivaciones se ordenan en la tira electrocardiográfica de la siguiente manera:

DI, DII, DIII, AVR, AVL, AVF, V1, V2, V3, V4, V5, V6.

Y se recomienda tomar cuatro complejos por derivación para tener una mayor visión del trazo, para que de ésta manera pueda identificarse alguna patología cardiovascular.

Por ello deben recordarse las reglas generales, mostradas en la Tabla 1.3, para la correcta localización y semiología electrocardiográfica:

Tabla 1.3 Reglas Generales.

Derivaciones	Partes que analizan del Corazón
DII y V1	Son las mejores derivaciones para estudiar las Aurículas
DI,VL,V4,V5 y V6	Son las derivaciones que estudian al Ventrículo izquierdo
DII, DIII y VF	Estudian la cara posterior e inferior (diafragmática), del Corazón.
V1 y V2	Son las derivaciones del Ventrículo derecho y del Septum Ventricular.
V3	Estudia la transición entre ambos ventrículos.
V3 y V4	Observan la cara interior del corazón.
V5 y V6	Sirven para observar la pared libre del corazón.

Se debe analizar cada una de las derivaciones de manera individual para llegar a un diagnóstico acertado del funcionamiento cardiaco del paciente.

Cabe mencionar que existen cinco tipos de observaciones básicas en todo electrocardiograma, como son: Ritmo, Frecuencia, Infarto, Eje e Hipertrofía que serán explicados posteriormente.

1.2 Representación de estados Normales y Anormales

Para determinar cuales son los estados normales y anormales del funcionamiento cardiaco, que son obtenidos a través de la onda del electrocardiograma, se definen las siguientes observaciones obligatorias que deben hacerse para leer un electrocardiograma y conocer el estado del corazón, cómo se presentan dichas ondas, cuando su estado es anormal.

Para leer un ECG debe hacerse cinco tipos de observaciones generales:

1.- Frecuencia.
2.- Ritmo.
3.- Eje.
4.- Hipertrofia.
5.- Infarto.

1.2.1 FRECUENCIA.

La Frecuencia se da en ciclos por minuto y depende del nodo SA. Una frecuencia normal se encuentra en 80 mm/seg. El nodo SA es el marcapaso normal del corazón, el cual a un intervalo de tiempo produce un impulso eléctrico que genera una contracción del músculo cardiaco. En el caso de que éste falle, pueden activarse otros marcapasos para iniciar el latido del corazón. Dichos marcapasos, conocidos como ectópicos, tienen frecuencias variadas que pueden ser 75/min, 30-40/min y algunas frecuencias de urgencia de 150-250/min.

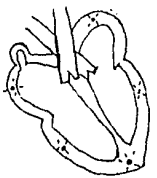


Fig. 1.10 Marcapasos ectópicos.

Si la frecuencia está por debajo de los 60/min se considera una Bradicardia Sinusal y por arriba de los 120/min es una Taquicardia Sinusal.

¿ Cómo se puede calcular la frecuencia cardiaca ?

Considerando que la velocidad del papel es de 25 mm/seg, cinco cuadros grandes (25 mm), equivalen a un segundo, por lo tanto cada cuadro grande representa 0.20 de segundo, de esta manera: 300 cuadros grandes equivalen a un minuto.

De lo anterior para se tiene que calcular la frecuencia cardiaca basta con dividir 300 entre el número de cuadros grandes que separen dos ondas R.

1.2.2 RITMO.

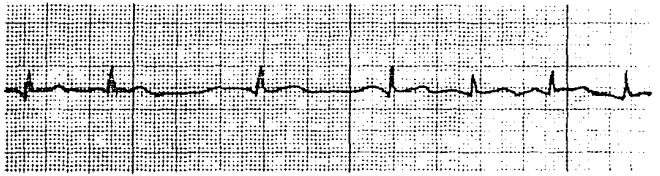
El Ritmo es la distancia entre ondas semejantes. Un ritmo normal (regular), tiene igual distancia entre ondas semejantes. El ritmo normal es llamado en general Ritmo Sinusal Normal, ya que nace en el nodo SA. Para afirmar que un ritmo es sinusal, se requiere que cada complejo QRS sea precedido de una onda P, positiva en DI, DII y DIII. Se tienen varios tipos de arritmias:

Ritmo Variable.

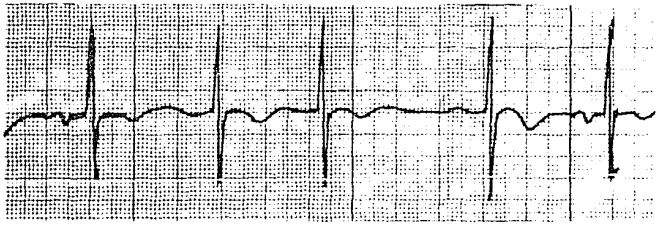
Por ritmo variable se entiende un grupo de ritmos irregulares en el cual se conserva el segmento normal (P-QRS-T) de las ondas, pero existe un cambio continuo en el ritmo.

Arritmia Sinusal.
- Ritmo variable.

- Ondas P idénticas.
- Frecuencia irregular.



Marcapaso Migratorio.
- Ritmo variable.
- Cambios de forma de la onda P, para una misma derivación.



Fibrilación Auricular.

- Ritmo Variable.
- No hay P verdaderas, sino espigas auriculares ectópicas.

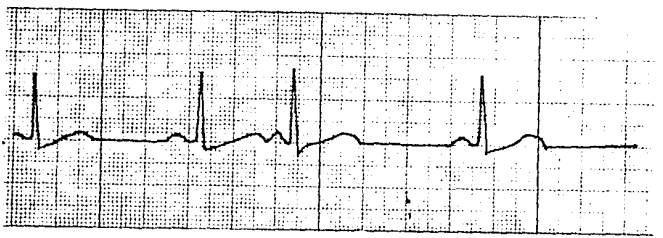
Extrasístoles y Fallas.

Las extrasístoles pueden identificarse como ondas que se presentan antes de lo esperado. Las fallas se refieren a zonas planas prolongadas de la línea basal.

Las ondas generadas debido a extrasístoles son ocasionadas por la activación prematura de varios focos ectópicos.

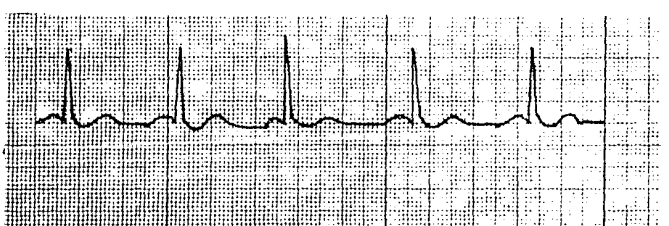
Extrasístole Auricular.

Ocurre cuando un foco ectópico de aurícula es activada, produciendo una onda P anormal antes de lo esperado.



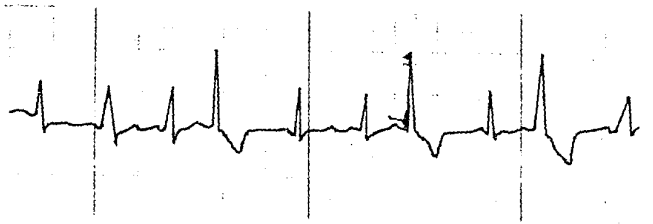
Extrasístole Nodal (AV).

Cuando se produce la activación de un foco ectópico.



Extrasístole Ventricular.

- QRS muy ancho.
- Pausa muy prolongada.

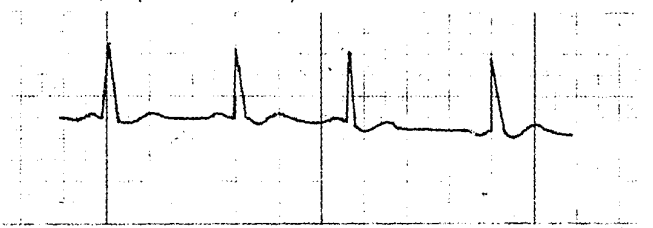


Sístole de Escape.

Cuando el marcapaso normal deja de producir estímulos durante uno o varios ciclos, se activa un foco ectópico "impaciente".

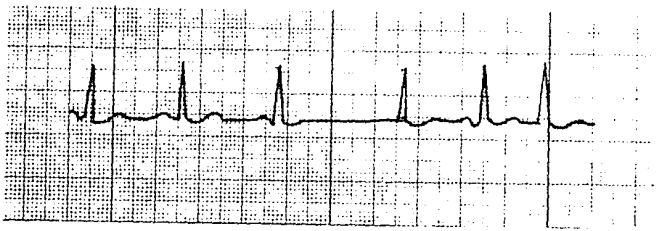
Escape Auricular.

Esto ocurre cuando un foco ectópico de la aurícula se activa y estimula las aurículas, después se normaliza por el nodo AV.



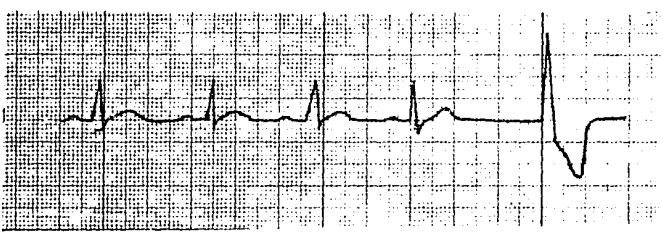
Escape Nodal (AV).

Nacen en el nodo AV y estimulan los ventrículos por el sistema de conducción normal, produciendo un QRS normal después de una pausa.



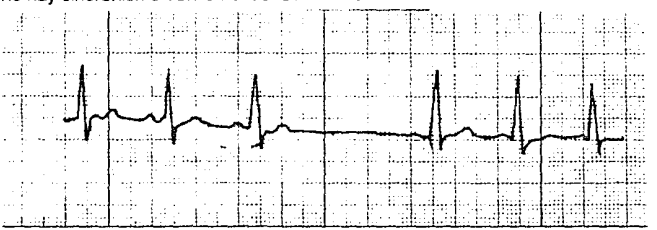
Escape Ventricular.

Nacen de un nodo ectópico de los ventrículos y producen una onda ventricular de la forma Extrasístole Ventricular después de una pausa en el ritmo.



Paro Sinusal.

Hay paro sinusal cuando de repente se detiene el nodo SA y no hay estímulos. Después empieza a funcionar otra nueva región automática, pero no hay sincronismo con la frecuencia anterior.

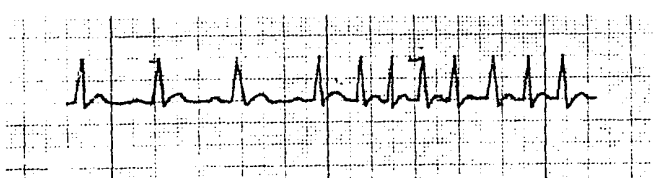


Ritmos Rápidos.

Taquicardia Paroxística.

Se entiende la aparición brusca de un ritmo cardiaco rápido, generado por un foco ectópico.

- Ritmo 150-250 por minuto.

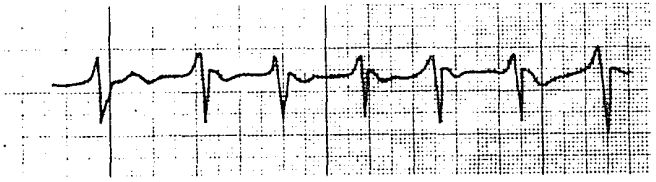


Taquicardia Paroxística Auricular.

Ocurre cuando se tiene una activación brusca de un foco ectópico auricular.

- Ritmo de 150-250 por minuto.
- Ondas P no idénticas.

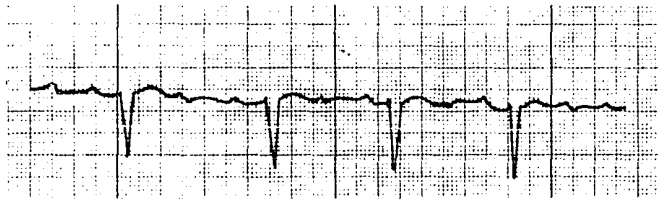
Pueden confundirse ondas P con ondas T.



Taquicardia Paroxística Auricular con bloques.

- Ondas P pequeñas, agudas y hacia arriba en DII y DIII
- Segmentos ST isoeléctricos.

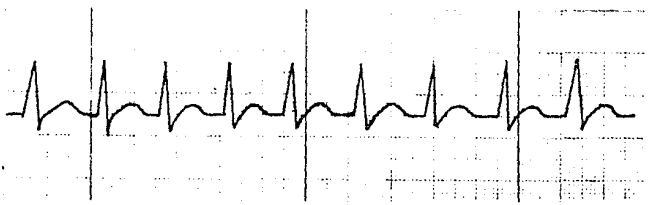
Normalmente indica intoxicación digital.



Taquicardia Paroxística Nodal.

Es una frecuencia alta (150-250) iniciada por un foco ectópico en el nodo AV.

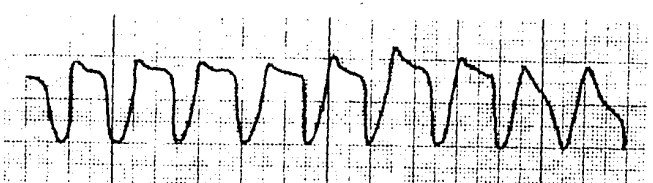
- Pueden existir ondas P invertidas.



Taquicardia Paroxística Ventricular.

Se origina en forma brusca por un foco ectópico ventricular. La frecuencia se encuentra entre 150 y 250 por minuto.

- Normalmente no aparecen las ondas P.



Aleteo Auricular.

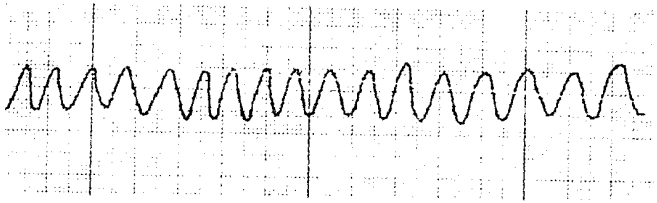
Nace de un foco ectópico auricular, las ondas P aparecen continuamente, cada una idéntica a la precedente.

- Frecuencia entre 250 y 350 por minuto.
- El QRS es de aspecto normal.

Aleteo Ventricular.

Ocurre con la activación de un foco ectópico ventricular único, con una frecuencia entre 200 y 300 por minuto.

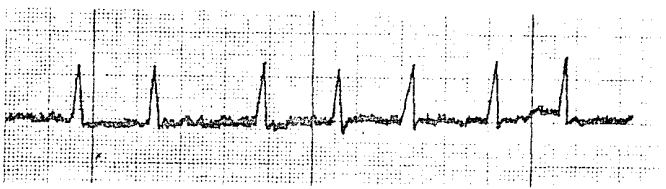
- No tiene forma de segmento normal.
- Tiene forma Sinusoide regular.



Fibrilación Auricular.

Se debe a la activación de muchos focos ectópicos de aurículas, disparando a frecuencias distintas, generando actividad caótica e irregular.

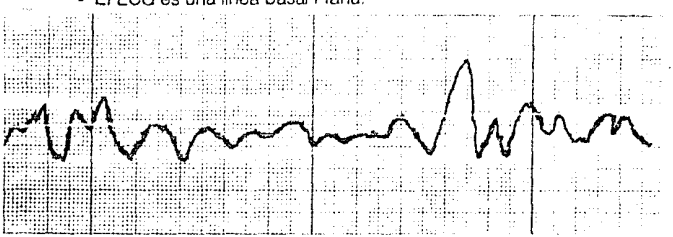
- Se manifiesta por una línea basal sin ondas P.
- QRS irregular, rápida o lenta.



Fibrilación Ventricular.

Ocurre cuando muchos focos ectópicos ventriculares se activan a frecuencias diferentes, apareciendo contracciones caóticas de los ventrículos.

- No hay bombeo eficaz del corazón.
- El ECG es una línea basal Plana.



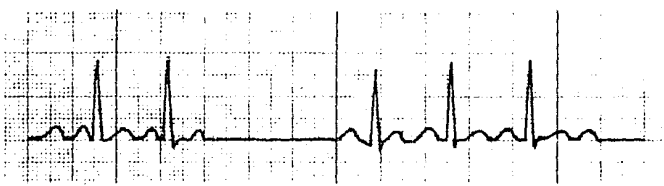
Bloqueos Cardiacos.

Bloqueo SA.

Existe bloqueo cuando el marcapaso se detiene momentaneamente. En el Bloqueo SA, el nodo SA se detiene, por lo menos un ciclo, y después entra a su actividad.

- Las ondas P son idénticas antes y después del bloqueo, por ser el mismo marcapaso SA.

- Falta algún ciclo.

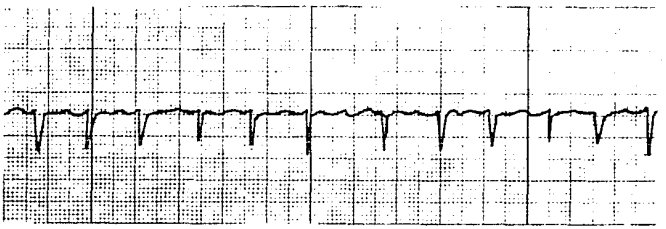


Bloqueo Auricular-Ventricular (AV).

Existe un transtorno en el nodo aurículo- ventricular que condiciona una dificultad para la conducción del estímulo aurícular hacia los ventrículos.

Bloqueo AV de primer grado.

- Prolonga el intervalo PR en más de 0.2 segundos.
- El segmento P-QRS-T es normal.



Bloqueo AV de segundo grado.

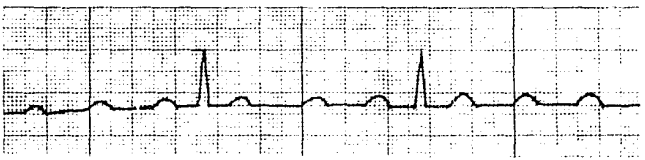
Existe cuando requiere de dos impulsos auriculares o más para iniciar una respuesta ventricular (QRS).

Bloqueo AV de segundo grado tipo Mobitz I.

Después de cada latido el PR va creciendo hasta que en el tercer o cuarto estímulo no aparece el segmento QRS, también es conocido como fenómeno de WENCKEBACH.

Bloqueo AV de segundo grado tipo Mobitz II.

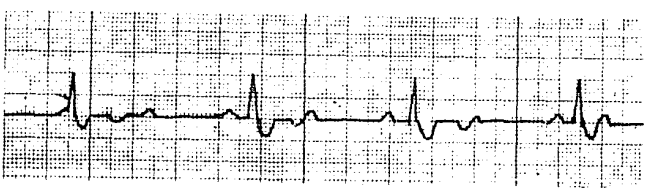
Algunas ondas P son conducidas al ventrículo pero otras no, ocasionando que falten QRS y haya más ondas P. Después de cada onda QRS aparece una onda P con PR menor a 0.40 y mayor a 0.08.



Bloqueo AV de tercer grado.

Condiciona frecuencias ventriculares muy bajas que provocan insuficiencia cerebral con pérdida de conocimiento y convulsiones (Sindrome de Stockes Adams).

El estímulo aurícular no se conduce al ventrículo y genera el estímulo de algún marcapaso ectópico. Existe mayor número de ondas P que complejos QRS, los PR son variables y son menores de 0.08 y mayores a 0.20, el QRS será más ancho y más lento, dependiendo del marcapaso.



1.2.3 EJE ELECTRICO.

Por Eje se entiende la dirección de la polarización que recorre el corazón y estimula las fibras, haciendo que se contraigan.

La dirección o eje es determinado por la forma de QRS (positivo o negativo), en las derivaciones de miembro y precordiales.

Para poder determinar el eje eléctrico a simple vista se realizan los siguientes casos: Consideraremos las tres derivaciones bipolares (DI, DII y DIII).

- Si las tres son positivas, el eje se encuentra situado en cuadrante normal (entre +90 y 0).

- Si existe un predominio de la negatividad en el DI, el eje esta desviado discretamente hacia la derecha.

- Si existe negatividad en DI y negatividad en DII, el eje se encuentra muy desviado hacia la derecha.

- Si hay negatividad en DIII y positividad en DI y DII, el eje está desviado discretamente hacia la izquierda.

- Si se encuentra que las derivaciones DII y DIII son predominantemente negativas, el eje está muy desviado a la izquierda.

El eje eléctrico indica el equilibrio de las fuerzas entre el ventrículo izquierdo y el derecho, como normalmente el ventrículo izquierdo es el predominante, el eje se encuentra en el cuadrante inferior izquierdo: (+90 a 0) (positividad en las tres derivaciones).

Cuando la masa del ventrículo izquierdo aumenta (hipertrofía ventricular izquierdo) o cuando la actividad de dicho ventrículo se retarda (bloque de rama

izquierda), el eje se mueve en forma moderada hacia la izquierda (+0 a -45) (Negatividad en DIII y positividad en DI y DII).

Cuando existe bloqueo de la conducción en el fascículo anterior de la rama izquierda (Hemibloqueo anterior izquierdo) el eje se desvía mucho hacia la izquierda (-45 a -90) (Negatividad en DII y DIII y positividad en DI).

1.2.4 HIPERTROFIA.

Por hipertrofia suele designarse un aumento de tamaño; en relación con un músculo, éste termino designa un aumento de masa muscular.

Las hipertrofias se encuentran en el estudio de las derivaciones precordiales.

La hipertrofia se encuentra en el análisis de la onda p. En éste caso, la onda P es bifásica, fig. 1.12, la onda es a la vez positiva y negativa.

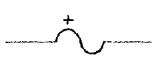


Fig. 1.12 Onda bifásica.

1.2.5 INFARTO.

El infarto del miocardio se debe a la oclusión de una arteria coronaria. En estas condiciones, una arteria queda sin circulación, esta oclusión puede ser relativa, en el sentido de que un individuo con arterias coronarias muy estrechas puede conservar una función normal en reposo; pero en caso de agitación o ejercicio, el corazón que late con más fuerza requiere una mayor cantidad de sangre (y de oxígeno), que sus coronarias no le pueden dar. Esta variedad de infarto puede ser muy grave, incluso mortal, como la oclusión coronaria clásica.

La agresión que sufren las fibras miocardicas durante el infarto del miocardio es variable. La mayoria, durante la fase aguda, se encuentra en un estado de severa afección metabólica conocido como "Lesión", algunas sufren de una afección menos importante "Isquemia", poco a poco algunas celúlas de lesión mueren dando origen a una zona de "necrosis", mientras otras mejoran convirtiendose en isquémicas, de tal manera que el área infartada queda constituida a los pocos días, sólo por zona de necrosis y zona de isquemia. Esta situación puede persistir el resto de la vida del paciente.

La traida clásica del infarto de miocardio agudo es Isquemia, Lesión e Infarto; pero pueden ocurrir aisladamente cualquiera de éstos elementos.

1.2.6 DESCRIPCION DEL ELECTROCARDIOGRAMA NORMAL.

En condiciones normales el electrocardiograma reune las siguientes reglas:

En cada derivación, los latidos que se registran suceden en forma regular y todos tienen una morfología similar.

Cada complejo QRS es precedido de una onda P, positiva en todas las derivaciones excepto en AVR, la onda P se encuentra separada de cada complejo QRS en forma clara por no menos de un cuadro pequeño y no de más de tres cuadros pequeños.

El complejo QRS es predominantemente positivo en las derivaciones "periféricos" (DI, DII, DIII, AVL y AVF), excepto en AVR en que es negativo, en AVL puede ser equifásico (una positividad y una negatividad de igual tamaño).

En las derivaciones precordiales tiene la morfología RS en V1 (positividad pequeña seguida de una negatividad mayor).

En V6 tiene la morfologia QR y en ocasiones existe una S pequeña (pequeña negatividad seguida de una gran onda positiva y en ocasiones una segunda positividad pequeña).

Las derivaciones V2 a V5 progresan paulatinamente entre estas dos morfologías, frecuentemente V3 muestra transición intermedia, RS.

No debe existir una onda Q, en ninguna derivación mayor de 1/3 del total del QRS o de más de un cuadrito de ancho. El ancho del QRS no debe ser mayor de 2 1/2 cuadritos.

La onda T es positiva en todas las derivaciones periféricas excepto AVR en donde es negativa.

La onda T es frecuentemente negativa en V1 y V2 especialmente en mujeres y niños, pero así fuera la porción descendente es más larga que la ascendente (asimétrica).

El segmento ST debe estar alineado con el segmento PR y con la línea isoeléctrica que une la onda T con el siguiente complejo, se toleran como normales desviaciones hacia arriba (desnivel positivo) o hacia abajo (desnivel negativo), no mayores de medio cuadrito.

El ritmo sinusal puede tener variante no necesariamente anormales que son: La taquicardia sinusal más de 100 latidos por minuto; la bradicardia sinusal, menos de 60 latidos por minuto.

La arritmia sinusal: Todos los complejos son normales, precedidos de onda P, pero la frecuencia cardíaca es irregular, aumenta con la inspiración y disminuye con la expiración, es un ritmo muy común en los niños y constituye una variante de lo normal.

El proceso para determinar el estado del paciente es muy difícil y debe analizarse con cuidado. Una vez realizado el análisis y diagnosticado, debe compararse con el patrón de normalidad del electrocardiograma para verificar

el diagnóstico y estar seguro de ello. Los métodos que se piensan aplicar para realizar el reconocimiento e interpretación del electrocardiograma hacen uso de reconocedores de patrones en sus diferentes áreas de estudio, como son compiladores, autómatas finitos estocásticos, etc..

1.3 MODELOS Y METODOLOGIAS PARA LA REPRESENTACION DE UN ELECTROCARDIOGRAMA

Como se ha mencionando el electrocardiograma es trazado en papel milimétrico donde los impulsos eléctricos que recibe el electrodo genera la onda ECG. Para obtener el diagnóstico debe aprenderse una serie de elementos , una de ellas es la forma de onda de las 12 derivaciones, que tiene un ECG de estado normal. Así, sólo las personas que tienen experiencia en el reconocimiento e interpretación dan un diagnóstico correcto.

En el análisis por computadora del ECG se podría tener un diagnóstico inmediato y la persona que toma el electrocardiograma, que puede o no tener experiencia, entregaría el resultado. De ésta forma la seguridad será mayor.

Para tratar de implementar un sistema reconocedor y proporcionador de resultados de un ECG se debe basar en un proceso que conste de las siguientes partes: señal electrocardiográfica, codificador, reconocedor e interpretador, analizador de cadena y diagnóstico; donde cada una de ellas proporciona elementos de análisis de la onda electrocardiográfica, como se muestra a continuación.

Para conocer más detalladamente los elementos que integran este proceso, a continuación se definen:

Señal electrocardiográfica: Es la señal generada por el ECG, ésta contiene las 12 derivaciones, percordiales y miembro. El ECG debe conectarse por medio del puerto serial de la computadora para que la señal pueda ser codificada.

Codificador: Una vez que la señal del ECG es enviada a la computadora, el codificador el cual es un programa de programación que permita detectar el

impulso eléctrico producido por el ECG y permita asignar el caracter correspondiente a una cadena de caracteres, dependiendo del modelo, para que pueda ser reconocida e interpretada. Dado que para saber cual es el caracter que debe adicionarse, se analiza la onda digitalmente a través de la computadora por medio de algún modelo matemático que identifique el tipo de pendiente encontrado. En los capítulos siguientes, se mencionan los modelos que pueden ser utilizados para realizar el análisis.

Reconocedor e Interpretador. En este proceso se realiza el reconocimiento de la cadena generada por el codificador. Los modelos que se analizarán posteriormente (autómata finito, Horowitz y Stockman), proporcionan los reconocedores de cadena, con el alfabeto que defina el modelo. Estos reconocedores son de expresiones regulares o de gramática de contexto libre.

Si la cadena cumple con las reglas que se definen en el reconocedor, se pasa al analizador de cadenas.

Analizador de cadenas: Cuando la cadena reconocida llega a este proceso, se analiza para obtener los elementos que permiten proporcionar un diagnóstico acertado. Así por ejemplo, para obtener un ritmo variable[1] debe verificarse si la distancia que existe entre la onda T y P son las mismas en todos los segmentos de una derivación. Así para el alfabeto {0, 1} y alfabeto {n, p, 0} es el número de "ceros" existentes entre estas ondas la que determinará esta cardiopatia y en el alfabeto {0, A, B, C, D, E, F, G, H, I, J, K, L} si en el análisis de la cadena en lugar de A aparece D o viceversa, en la comparación de varios segmentos, reflejaría este mismo estado.

La cadena que determina una fibrilación[2] presenta cambios continuos de caracteres, asi para el alfabeto {1, 0} se tendrían subcadenas 1011110 ó 1111010, para el alfabeto {n, p, 0} aparecerán npnp ó pnpn y en el alfabeto {0, A, B, C, D, E, F, G, H, I, J, K, L} aparecerán subcadenas KLKL ó LKLK.

Diagnóstico: Este proceso es determinado a partir del análisis realizado en el analizador de cadena, de esta forma se proporcionará la cardiopatia del

1 Descrito en la página 14

2 Descrito en la página 15

paciente. Con lo que se podría agilizar más el diagnóstico de este tipo de padecimientos.

Para el autómata estocástico de estados finitos se plantea el siguiente modelo:

La fuente de información es el sistema de ritmo, se define con mayor detalle en el capítulo tres, que toma como elemento principal un autómata estocástico de estados finitos, el cual opera en tiempo discreto y proporciona las fuentes que nos permita simular la estimulación del corazón.

El codificador interpreta los músculos del corazón, donde al ser éstos estimulados se genera una señal del ECG fija para todo el electrocardiograma, durante el intervalo en el que sigue estimulado.

Cuando la señal es enviada a la superficie del cuerpo para ser interpretados por el ECG, se corre el riesgo de que músculos que no pertenecen al corazón proporcionen señales que sean tomados por los electrodos y generen un electrocardiograma con señal ruidosa. Este ruido es proporcionado, en la simulación, por un canal ruidoso.

El decodificador permite obtener una secuencia discreta de símbolos por un detector que es usado para localizar sólo los puntos de activación, es decirel segmento QRS que son producidos al estimularse el corazón.

El proceso en el que se trabajará, es el de reconocedor e interpretador de electrocardiogramas, que tiene su aplicación en modelos sintácticos.

1.4 RECONOCIMIENTO DE PATRONES E INTERPRETACION DE ELECTROCARDIOGAMAS.

En el área de nuestro interés, el reconocimiento de patrones "Patern Recognition", (RP), es un elemento útil. Este tiene cabida en otras áreas diferentes a las de generar o reconocer imagenes, y ésto hace que el reconocimiento de patrones tenga aplicaciónes diversas.

Cuando se escucha el término Reconocimiento de Patrones, se piensa inmediatamente en imagenes visuales. Pero este, no es la única rama donde se incluye el efecto de RP, el patrón puede ocurrir en un evento visual, acústico, eléctrico, quimico, tangible o un elemento más abstracto.

Algunos conjuntos de objetos pueden estar representados por tipos de archivos llamado paradigmas, ejemplares, representatividades o miembros característicos, otros pueden ser definidos con alguna descripción adecuada.

El RP tiene dos facetas muy interesantes: El primero, el poder construir una máquina que reconozca buenos componentes para el diseño, y la segunda es la clasificación o descriminación.

PROBLEMAS QUE SE PRESENTAN EN EL USO DE RP.

Para reconocer una imagén cualquiera que sea, se debe tener los patrones bases que la forman, asi dependiendo la imagén que se desee reconocer es el número de patrones que utilizará.

Por ejemplo, si se desea reconocer diferentes clases de células, los patrones que se formen debe permitirlo, donde la clasificación se debe hacer en base a forma, color y textura.

Para reconocer una burbuja, los patrones deben estar regidos por la definición matemática de circulos. Pero al intentar reconocer alguna clase de caracteres, cuerpos celulares, huellas dactilares, partículas pequeñas, fracturas de huesos con fotógrafias radiológicas, entre otros, no se tiene aún una definición analítica y ésto lo hace más difícil.

Algo que debe mencionarse, es que el reconocer de cadenas también entra en el área de reconocedor de patrones, donde con el uso de un compilador o un autómata finito, puede realizarse dicha tarea.

El problema principal que enfrenta el reconocer de patrones, es el almacenar la gran cantidad de patrones que son generados para realizar reconocedor de imagenes

Primeramente el patrón debe ser transladado en una forma conveniente para su tratamiento. La nueva tecnologia, especialmente computadoras digitales contienen la flexibilidad del procesamiento y la rapidez que estas requieren. Esto permitirá que pueda tener un gran desarrollo y avance.

Todos los problemas de reconocimiento de patrones requieren construcción de información. Las técnicas de representación de patrones pueden solamente ser juzgados por sus ejecuciones con un sistema completo de reconocimiento de patrones [6].

Hay algunas formas de describir los patrones: Una forma particular es el vector, representando los resultados de un número de medidas tomadas en paralelo. Los vectores son una forma natural de descripción de patrones donde un número de unidades de medida separadas operan en paralelo. Los descriptores pueden ser precisos o vagos, numéricos o no numéricos.

Una carga de sistema, red eléctrica o diagrama PERT no puede facilmente ser representado en un vector o lista de caracteristicas. Podría hacerse usando una matriz adyacente o un arreglo de nodos y puntos. Sin embargo, nada de esto es particularmente conveniente y es mejor el uso natural de descripción de patrones.

La toma de decisiones (o clasificación), puede aceptar la descripción dada sin protestar, o requiriendo nueva información, tomando alguna declaración acerca del patrón.

RELACION ENTRE RECONOCIMIENTO DE PATRONES Y OTRAS AREAS.

Cibernética, Psicologia y RP : El RP es usada por psicologos en relación a la percepción animal y humana (principalmente acústico y visual). El RP se usa para emular organismos biologicos.

Diseño Lógico: El problema de diseño lógico puede ser visto como un espacio multidimensional particionado. El diseño lógico usa variables de entrada binaria y, muestra la aparente simplicidad del problema. Las técnicas heurísticas ofrecen la solución práctica solamente cuando hay un gran número de variables de entrada.

Taxonomía Numérica / Análisis de Cluster : Esta área es altamente desarrollada por biologos y psicologos y está muy estrechamente ligada al RP. Se tiende a usar algunas de éstas técnicas, pero éste se ha encontrado muy difícil de aplicar con grandes conjuntos de datos que el RP tiende a generar.

Máquinas de Estado Finito: Una máquina de estado finito puede llamarse un Reconocedor de Patrones. Sin embargo, la clase de secuencias de entrada que tal dispositivo puede detectar son muy limitados y de poco valor práctico, por ejemplo, alguna expresión regular que contenga los símbolos 0 y 1.

Compiladores : Un compilador realiza el reconocimiento de un segmento de datos, donde analiza sintácticamente los datos de entrada. El compilador genera un listado de errores de los datos que fueron introducidos.

Teoría de Gráficas: El isomorfismo gráfico es un problema altamente considerado como un aproximador, bajo la ayuda del RP. El hecho de decidir que una gráfica A es una subgráfica de otra, B(A B), es también un problema del RP. El RP incluye problemas más difíciles como es "subgráficas aproximadas". Los arcos y nodos de las gráficas usadas en RP están frecuentemente etiquetadas; la aproximidad de cada gráfica es mucho más difícil de preveer.

Detección y Procesamiento de Señales: A la ingeniería en comunicaciones le concierne el decidir si una señal es presentada. La Detección de señal está más directamente relacionado al análisis matemático exacto con RP, formandose un problema de una dimensión.

Procesamiento de señales y RP tienen un común interés en ciertas técnicas, especialmente transformadas de Fourier, Walsh y Hoar. En RP éstos son los tipos de patrones más usados:

- Patrones de una Dimensión.

- Patrones de dos dimensiones.

- Patrones de tres dimensiones.

Ingeniería de control / Sistemas de Alimentación: Estas tienen mucho en común con el RP. El concepto estado-espacio del control de ingeniería es equivalente en el desarrollo de espacios en RP.

Procesamiento de Datos: Esta es un área básica cliente del RP. Leer cheques, reconocer voces por una máquina, obtener una reducción de costos al incluir la computadora con el manejo de programas.

- Usando Sistemas de Información.
- Teoría de Información.
- Calculos Iterativos y Gráficos.
- Estudio de la Arquitectura de Computadoras.
- Teoría de la Probabilidad y Estadística, etc..

Estas son algunas de las áreas donde puede aplicarse el reconocimiento de patrones. De todas ellas, sólo haremos énfasis de la aplicación reconocimiento de patrones a través de reconocimiento gramáticales que nos permita reconocer una cadena de caracteres que produce el electrocardiograma.

En los siguientes capítulos haremos el análisis de la onda y generaremos el reconocedor del electrocardiograma para los métodos de Stockman, Horowitz, expresiones regulares y modelo estocástico de estados finitos.

CAPITULO 2.

INTERPRETACION COMO LENGUAJES REGULARES

La aplicación de expresiones regulares, es uno de los modelos más usados para especificar ciertos problemas que requieren algún proceso secuencial a alto nivel. Estos modelos están relacionados a los de reconocimiento de formas o aceptadores de lenguaje, donde ambos pertenecen al área de reconocimiento de patrones.

Las expresiones regulares consisten de un conjunto de elementos: Un alfabeto, operadores, operandos, etc. (se mencionarán más adelante), donde al adoptarlo a un conjunto de reglas, éstas quedan construidas para resolver el problema.

El aplicar autómatas finitos permitirá obtener un reconocedor de lenguaje, partiendo de la expresión regular, aplicando Autómatas Finitos No-Determinístico (AFN) y Autómatas Finitos Determinísticos (AFD), mayor referencia en [9] y [14].

El problema que se analizará, trata de generar una expresión regular que represente la forma de onda de un ECG.

2.1 RECONOCEDOR DE EXPRESIONES REGULARES Y AUTOMATAS.

Se darán los elementos necesarios para poder entender el significado de Autómatas y sus base:

DEFINICION. Un alfabeto es un conjunto finito de símbolos. Una cadena sobre un alfabeto es una secuencia de longitud finita de símbolos de . La cadena vacía se denota por y es la cadena que no contiene símbolos.

DEFINICION. Si x y y son cadenas, entonces se define concatenación como xy. En general, se tiene que si x, y y z son cadenas, entonces:

a).- $xy \neq yx$
b).- $(xy)z = x(yz)$ equivalente a xyz.

DEFINICION. Sea xyz una cadena, se dice entonces que x es el prefijo, y es una subcadena y z es el sufijo de la cadena.

DEFINICION. La longitud de una cadena x es denotada como $|x|$ y se define como el número total de símbolos de x.

DEFINICION. Un lenguaje L sobre un alfabeto Σ , es un conjunto de cadenas definidas sobre. Si L_1 y L_2 son dos lenguajes entonces L_1L_2 es la concatenación de los lenguajes y se define como:

$$\{x, y / x^{\epsilon} L_1 \text{ y } y^{\epsilon} L_2\}$$

DEFINICION. Sea L un lenguaje, definimos entonces lo siguiente:

a).- $L^{\circ} = \{^{\epsilon}\}$, Cadena vacía.

b).- $L^{i} = LL^{i-1}$; con $i = > 1$.

La cerradura de Kleen de L es denotada por L^* y es el lenguaje siguiente:

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

y la cerradura positiva de L está denotada por L^+ y es el lenguaje:

$$L^* = \bigcup_{i=1}^{\infty} L^i$$

EXPRESION REGULAR.

Lo que llamamos expresiones regulares sobre el alfabeto son exactamente aquellas expresiones que pueden construirse a partir de las siguientes reglas.

Cada expresión regular denota un lenguaje, y proporcionamos las reglas para la construcción del lenguaje denotandolo conjuntamente con las reglas de construcción de expresiones regulares.

Sea Σ un alfabeto, la expresión regular sobre Σ y los lenguajes que ellos generan se define recursivamente como:

1).- ϕ, es una expresión regular y denota el conjunto vacío.

2).- ε, es una expresión regular y denota el conjunto $\{\varepsilon\}$.

3).- Para cada $a \in \Sigma$, a es una expresión regular y denota el conjunto {a}.

4).- Si p y q son expresiones regulares denotando los conjuntos P y Q, entonces:

$(p+q)$, pq , p^* son expresiones regulares que se denotan como PUQ , PQ , P^*.

$\Sigma^* = \Sigma \cup \{\varepsilon\}$
con $L = \{ X / x \text{ y } y;$ donde x y y son cadenas$\}$

Siguiendo las líneas de producción puede formarse un lenguaje.

El lenguaje regular:

Expresión regular $= \Sigma \cup \{\varepsilon\}$ y que cumplan con:

a).- $p+q$

b).- pq

c).- p^*

2.1.1 AUTOMATAS FINITOS

Un reconocedor para un lenguaje L es un programa que toma como entrada una cadena x y responde "si" si x es una expresión de L y "no" si no lo es. Claramente, la parte de un analizador léxico que identifica la presencia de una señal en la entrada es un reconocedor para el lenguaje que define esa señal.

Suponemos que hemos especificado un lenguaje para una expresión regular R y damos una cadena x, deseamos conocer si x se encuentra en el lenguaje L denotada por R. Una forma de conseguir esto es checar si x puede descomponerse en una secuencia de subcadenas denotadas por las subexpresiones primitivas en R.

2.1.1.1 AUTOMATA FINITO NO DETERMINISTICO.

Una mejor forma de convertir una expresión regular en un reconocedor es el construir un diagrama de transición genaralizada desde una expresión. Este diagrama es llamado un autómata finito (no-determinístico). Este no puede ser, en general, fácilmente simulado por un programa simple, pero un autómata determinístico finito puede simularse fácilmente, y es construible partiendo de un autómata finito no determinístico (AFN).

Un AFN se define como una quintuple la cual consiste de :

S : Un conjunto de estados.

Σ^* : Un alfabeto.

δ : Función de transición.

δ \ S xΣ^* --> S

δ $(S_i,a) = S_j$

$S_i \in S$, $a \in \Sigma^*$, $S_j \in S$

S : Estado inicial.

F : Estados finales aceptantes.

TEOREMA. Sea una expresión regular, entonces siempre existe un AFN(M) de tal forma que acepte con las siguientes propiedades:

$M = (S,\Sigma,L,S_o,\{S_f\})$

1).- $||S|| <= L|^{\alpha}|$, donde $|^{\alpha}|$ denota la longitud de $^{\alpha}$

2).- Para cada a en $\Sigma \cup \{^{\epsilon}\}$, $L(S_f,a)$ es vacío.

3).- Para cada s en S , la suma de $||^{\delta}(S,a)||$ sobre todo

$a \in \Sigma \cup \{^{\epsilon}\}$ es a lo más dos.

En el caso de aplicar el operador cero, en alguna expresión regular los elementos que se obtienen son los mínimos, así si las expresiones regulares fueran algún a del alfabeto, el elemento vacío al construir el AFN(M) queda lo siguiente:

a).- $r = ^{\epsilon}$, es decir $\{^{\epsilon}\}$;

b).- ϕ , es decir el elemento vacío;

c).- r = a, cada elemento que pertenece al alfabeto.

Para construir un AFN con operadores i = 1, tenemos:

Caso 1: $\alpha = \beta + \Gamma$, donde ß y son$^{\Gamma}$ expresiones regulares. Si ß, son$^{\Gamma}$ expresione regulares tienen asociados dos AFN M_1 y M_2 respectivamente.

$M_1 = (S_1, \Sigma_1, \delta_1, q_1, \{f_1\})$ y $M_2 = (S_2, \Sigma_2, \delta_2, q_2, \{f_2\})$

$L(M_1) = L(\beta)$ y $L(M_2) = L(\Gamma)$

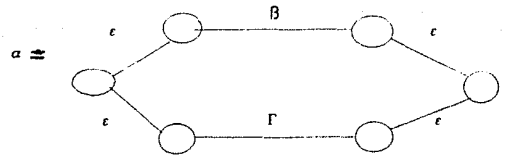


Fig. 2.1 Representación del AFN.

Caso 2:

$\alpha = \beta\Gamma$ .En donde ß y$^{\Gamma}$ son expresiones regulares entonces:

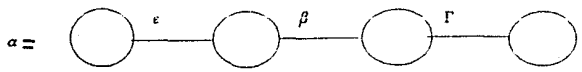


Fig. 2.2. Representación del AFN.

Caso 3:$\alpha = \beta^*$. En donde ß es una expresión regular definida como en el caso base siguiente.

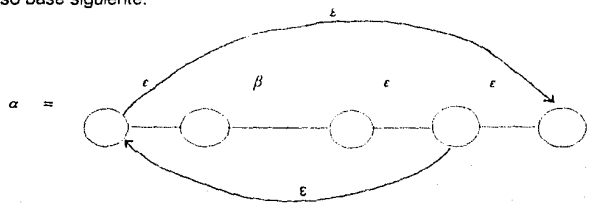


Fig. 2.3 Representación del AFN.

La transición de un AFN puede representarse convenientemente en forma tabular, es decir, por una tabla de transición. En la tabla de transición hay un renglón para cada estado y una cadena para cada símbolo admisible y para , si es necesario.

La entrada para un renglón **i** y símbolo **a** es el conjunto de posibles siguientes estados para el estado **i** con la entrada **a**.

Tabla 2.1

Estado	Entrada	Simbolo
0	{0,1}	{0}
1	-	{2}
2	-	{3}

El AFN acepta como entrada una cadena x si y sólo si existe un camino desde el estado inicial a algún estado aceptante, tal que las etiquetas sobre ese camino deletreen x.

El lenguaje definido por un AFN es el conjunto de cadenas entrada que acepta.

2.1.1.2 AUTOMATA FINITO DETERMINISTICO.

Las situaciones en que las funciones de transición son multievaluadas son la razón por la que es difícil simular un AFN con un programa por computadora.

Afortunadamente, existen versiones determinísticos de los automatas finitos que pueden simularse en forma directa, dado que un autómata finito determinístico (AFD) tiene como máximo un camino del estado inicial etiquetado por cualquier cadena.

Un autómata finito es determinístico si:

1).- No tiene transiciones con la entrada .

2).- Para cada estado s y símbolo de entrada a, hay a lo más una línea etiquetada a dejando s.

Para simular un AFD podemos crear una pieza de programa para cada estado, esa pieza de programa determina la transición apropiada por hacer en el símbolo de entrada actual.

Afortunadamente, para cada AFN podemos encontrar un AFD que acepte el mismo lenguaje. El número de estados de una AFD podrá ser exponencial al número de estados de un AFN, pero en la practica este (peor) caso raramente ocurre.

2.1.2 DE EXPRESIONES REGULARES A AUTOMATAS FINITOS.

Determinar cuando una cadena se encuentra en el lenguaje denotado por una expresión regular es una tarea fácil realizada por un autómata finito determinístico. Recordemos que una expresión regular es una notación natural para describir el lenguaje de una expresión.

Es relativamente fácil el ir de una expresión regular a un autómata finito no-determinístico. La construcción directa de un autómata finito determinístico es un poco más difícil.

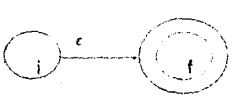
Se dará un algoritmo que produce un autómata finito no- determinístico a partir de una expresión regular.

ENTRADA: Una expresión regular R sobre un alfabeto .

SALIDA: Un autómata finito no-determinístico N que acepta el lenguaje denotado por R.

METODO: Primero descomponemos R en sus componentes primitivos. Para cada componente construimos un autómata finito inductivamente como sigue. Las partes (1) y (2) son la base y la (3) es la inducción.

1.- Para ε construimos el AFN.



2.- Para $a \in \Sigma$ construimos el AFN.

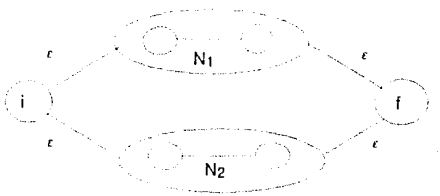
Cada vez que necesitamos un nuevo estado, damos a ese estado un nuevo nombre. Incluso si éste aparece varias veces en la expresión regular R, damos a cada instancia f un autómata finito separado con sus propios estados. De esta forma, no hay dos estados generados por los componentes básicos, o por la inducción construida con componentes que tengan el mismo nombre.

3.- Habiendo construido los componentes para las expresiones regulares básicas, procedemos a combinarlas de forma que correspondan a la manera en que expresiones regulares son construidas por expresiones regulares más pequeñas.

Para todas las expresiones regulares construimos un AFN con un estado inicial y uno final, y con las propiedades extra de que no más de dos líneas dejen un estado, y que ninguna línea ingrese al estado inicial o deje el estado final.

El límite de dos en el número de líneas que dejen cada estado es conveniente dado que permite una representación eficiente de la función de transición del autómata. Observamos que cada una de las anteriores propiedades se relaciona con el autómata básico construido en (1) y (2).

Suponemos ahora que tenemos AFNs N_1 y N_2 para las expresiones regulares R_1 y R_2. Para la expresión regular $R_1 \backslash R_2$ construimos el AFN compuesto



Donde i es un estado inicial nuevo y **f** un estado final nuevo. Hay una transición en desde el nuevo estado inicial a los estados iniciales N_1 y N_2. Hay una transición desde los los estados finales de N_1 y N_2 los cuales no son más finales en el AFN compuesto que tiene el estado final **f**.

CERRADURA VACIA.

Se define la ε-cerradura como el conjunto de estados que se construye a partir de las siguientes reglas:

a).- Se añade s a la ε - cerradura(s).

b).- Si t pertenece a la ε-cerradura(s) y existe una transición ε de t a u, entonces se añade u a la ε-cerradura(s).

2.1.3 GRAMATICAS Y RECONOCEDORES DE LENGUAJES.

TEOREMA. Si L es un conjunto regular, entonces L es aceptado por un AFD, ya que es generado por una expresión regular.

Para llegar a un núcleo del reconocedor de un lenguaje, tiene que hacerse un compilador, que contiene un analizador sintáctico y un analizador semántico.

Si se cuenta con el núcleo, tiene que incluirse el alfabeto y las reglas gramaticales, ésto con el generador automático de código nos dará como resultado un posible compilador.

Componentes de un compilador:

- "Scanner".
- "Parser".
- Generador de código.
- Optimizador de código.

COMPILADORES: Es un programa que traduce la instrucción de un lenguaje en un programa semánticamente equivalente, reconocible por la computadora. Donde esta forma reconocible es un código ensamblador o máquina.

La acción del "Parser" es aplicar sobre un programa fuente las propiedades sintácticas propias del lenguaje. El "Scanner" separa el texto de entrada en piezas o tokens para realizar un análisis lexicográfico y desplegar los errores.

Una vez que el texto ha sido analizado sintáctica y semánticamente se genera el código del programa fuente para dejarlo en lenguaje de máquina, antes de realizarse éste proceso pasa a la optimización de código para hacerlo más compacto.

Lenguaje : Un lenguaje es un conjunto de oraciones definidos sobre un alfabeto finito, denotado por V_t, tal que LV_t.

Una **gramática** consiste de un conjunto, no vacío, de reglas de reproducción las cuales son relacionadas a través de una sintáxis del lenguaje.

Una gramática se define como una cuadrupla:

$$G = (V_n, V_t, S, \Phi)$$

donde V_t y V_n son dos conjuntos de símbolos terminales y no terminales, S es un elemento de V_n que denota el símbolo de inicio. es un conjunto no vacío de la siguiente relación:

$$(V_t + V_n)^* V_n (V_t + V_n)^* \text{--} > (V_t + V_n)^*.$$

Un lenguaje L es un conjunto de oraciones especificadas a través de un conjunto de reglas de producción (α -> β); definidas por una gramática que se constituye como un sistema formado por un conjunto de símbolos terminales, un conjunto de símbolos no- terminales, un símbolo de inicio y un conjunto de oraciones que definen la sintáxis del lenguaje L, a través de operaciones de concatenación, unión y cerradura (producción).

Cabe, también señalar, que la regla de oraciones que forman un lenguaje L, están sujetos a restricciones gramaticales como:

- Gramáticas no restringidas.

- Gramáticas restringidas.
  * Gramáticas de contexto-sensitivo.
  * Gramáticas de contexto-libre.
  * Gramáticas regulares.

Gramática de contexto sensitivo.

Una gramática de contexto sensitivo contiene solamente producciones de la forma $\alpha \rightarrow \beta$, donde $|\alpha| <= |\beta|$ y $|\alpha|$ denota la longitud de .

Esta forma de restricción en una producción previene que ß sea no vacía. Esta forma involucra escribir α y ß como una producción $\alpha \rightarrow \beta$ donde $\alpha = \phi_1 A \phi_2$ y $\beta = \phi_1 t \phi_2$ (donde ϕ_1 y ϕ_2 pueden ser vacíos) y t debe ser no vacío.

DEFINICION. Gramática de contexto libre.

Las gramáticas de contexto libre contienen solamente producciones de la forma $\alpha \rightarrow \beta$, donde $|\alpha| <= |\beta|$ y $\alpha \in V_n$.

Las características que distinguen ésta gramática son:

1.- Consiste de producciones donde el lado izquierdo es representado por una clase de símbolo individual.

2.- Las gramáticas de contexto libre son capaces de especificar que ciertas variables fueron declaradas cuando éstas son usadas en alguna expresión del programa fuente. Sin embargo, éstas gramáticas pueden especificar más de las síntáxis del lenguaje de programación.

DEFINICION. Gramáticas regulares.

Contiene solamente producciones de la siguiente forma $\alpha \rightarrow \beta$, donde $|\alpha| <= |\beta|$, $\alpha \in V_n$ y ß tiene la forma α ß o d y donde $d \in V_t$, $\beta \in V_n$.

2.2 INTERPRETACION DE ELECTROCARDIOGRAMAS CON AUTOMATAS FINITOS.

Generación de señales de ECG Expresiones Regulares.

Para obtener la expresión regular que pueda representar el electracardiograma manejando un alfabeto {0, 1} vamos a considerar los siguientes puntos:

A).- En el electrocardiograma se tienen 4 tipos de ondas: curvas o sinoidales, rectas positivas, rectas negativas y rectas con pendiente cero. En el alfabeto sólo contamos con 2 caracteres diferentes, por lo tanto debemos hacer combinaciones de ellos para representar las formas de la onda.

B).- Debe tenerse la posibilidad de incluir las 12 derivaciones (de miembro y precordiales).

Para poder cumplir con los puntos anteriores debe tomarse en cuenta el conjunto de reglas siguientes:

a).- Si viene una onda curva (la onda P o T), puede indicarse mediante la presencia de 2 unos seguidos.

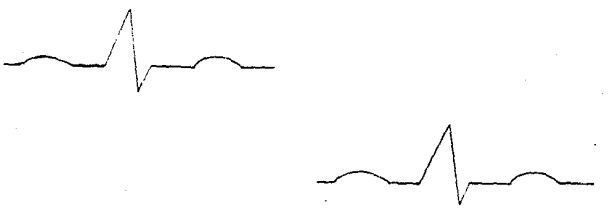
b).- Si viene una recta sin cambio o con pendiente cero (la onda P-R o S-T), se indicará mediante la presencia de 3 unos.

c).- Para las ondas rectas positivas y rectas negativas, si tiene 1 uno será una recta positiva y si se tiene la presencia de 4 unos, será recta negativa.

Como se observa, los unos indicarán que tipo de onda es la que se genera. ¿ Qué pasa con el símbolo 0 ?.

El alfabeto 0 representará el desplazamiento de 1 cuadrito (milímetro), a cualquiera de las direcciones del tipo de onda, indicando la longitud de cada uno.

La fig. 2.7 muestra la señal del ECG normal, la cadena obtenida al aplicar el alfabeto {1, 0} y la señal que produce la cadena si conforme a las reglas de producción trazaramos la onda.



11101100111001000111100001011100110011100

Fig. 2.7 Onda del ECG, onda que produce al interpretar la cadena y la cadena obtenida al aplicar la expresión regular.

Para representarse, la expresión regular que obtiene dicha aceptación de la onda es:

$$ER = ((111 . 0^*)^* + (11. 0^*)^* + (1.0^*)^* + (1111. 0^*)^*)^*$$

Dado que nuestro alfabeto contiene sólo dos elementos es difícil generar los diferentes tipos de onda que son producidos por el electrocardiograma.

La fig. 2.8 muestra la derivación DI de un paciente de sexo femenino de 18 años de edad a quien se le diagnóstica arritmia sinusal, la cadena que se genera es la siguiente:

111000000000001011110010000000111100000001110000100011110000

111000000000001011110010000000111100000001110000100001110000

111000000001011110010000000111100000001110000100011110000

111000000000001011110010000000111100000000111000010000111100000

La frecuencia que tiene la paciente es de 75 pulsaciones por minuto y sufre de arritmia sinusal no crítica.

Para poder determinar la existencia de una arritmia, necesitamos saber si la longitud de la recta que separa la onda T a la P es la misma entre cada complejo, de no ser así se diagnostica una arritmia.

Para saber el eje eléctrico del corazón se analiza los segmentos DI, DII y DIII la forma de onda de P (positiva o negativa). El eje eléctrico normal se

encuentra entre +90 y 0 grados, con la onda P positiva en las tres derivaciones.

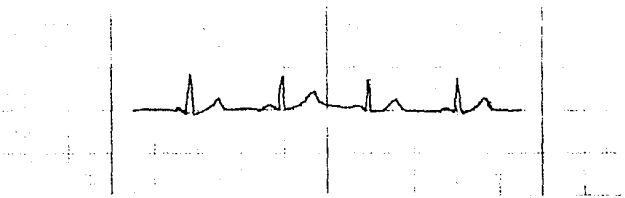


Fig. 2.8 Derivación DI de un ECG.

Como se puede observar la cadena que representa a la derivación DI, fig. 2.8, es muy grande y resulta más difícil el análisis de la cadena para dar a conocer el diagnóstico, si se toma en cuenta que sólo es una derivación de las 12 que se generan.

Para reconocer las diferentes cardiopatias que pueden encontrarse en el corazón debe analizarse la cadena en diferentes formas. Para determinar la arritmia no resulta fácil encontrar el inicio y fin de cada segmento para una nueva derivación. Para conocer el ritmo debe saberse la distancia que existen entre ondas P y T de cada segmento de la derivación, ésto resulta difícil por no saber el punto de inicio de estas ondas.

El programa que genera el reconocedor de lenguaje para la expresión regular encontrada para reconocer el ECG, se muestra en el apéndice A. Además se incluye el programa que hace el reconocimiento de la cadena.

En el capítulo siguiente se planteará un modelo de autómata estocástico de estados finitos, al cual llamaremos modelo del ritmo, será utilizado para simular la estimulación de los músculos del corazón. Se tienen tres estados que son representados por caracteres (A, B, C), y que para generar la cadena y saber cual es el caracter a insertarse se usa una búsqueda de árbol que contiene la probabilidad de transición de un estado a otro y donde la probabilidad mayor es la que determina la transición.

CAPITULO 3.

INTERPRETACION DE ELECTROCARDIOGRAMAS USANDO MODELOS ESTOCASTICOS DE ESTADOS FINITOS.

La interpretación del problema es definido usando un modelo de sistema de comunicación. Un autómata estocástico de estados finitos, llamado modelo del ritmo, es utilizado para simular los intervalos de tiempo en que los músculos del corazón son estimulados. La salida de éste modelo es codificado de unas señales ECG continuas por los músculos del corazón.

El ECG es decodificado en una secuencia discreta de símbolos llamado e identificado como la salida del detector. Las transformaciones de salida del modelo dentro del detector de salida es modelado o realizado por canal de ruido.

El problema de interpretación de las secuencias de salidas es resuelto con la búsqueda de ritmo del modelado de estados y la secuencia de salida, la cual tiene la mayor probabilidad de haber provocado una secuencia. Estas secuencias son llamadas interpretación de salida del detector. Dado un estado inicial del modelo ritmico y una secuencia del detector de salida, un procedimiento de búsqueda de árbol puede usarse para dar solución a éste problema.

La interpretación del sistema es definida utilizando un modelo de sistema de comunicación. Se usa un autómata estocástico de estados finitos "AEEF", para simular los estímulos del músculo cardiaco. Se muestra un ejemplo simple. El uso general de métodos sintácticos para el reconocimiento de patrones está descrito en [5] y [6].

DESCRIBIENDO LOS MUSCULOS DEL CORAZON.

Como se vio en el capítulo uno, el corazón tiene dos músculos:

* *EL ATRIAL*
* *LOS VENTRICULOS*

EL músculo Atrial es el menor de los dos, sus cámaras actuan como bombas de llenado, pero las cámaras ventriculares realizan el mayor trabajo. En el ciclo normal del corazón se estimula primero el músculo atrial , ésto ocasiona que ocurra un retardo antes de que los ventrículos sean estimulados lo cual permite al atrial su contracción mecánica. El sistema ritmico del corazón es responsable de la simulación de éstos.

Cuando se activa un músculo del corazón por medio de una estimulación, genera campos eléctricos los cuales pueden medirse en la superficie del cuerpo a través del electrocardiograma.

En algunos pacientes críticos existe un riesgo muy alto de malfuncionamiento de su sistema cardiaco. Si ciertas malfunciones no son detectadas rápidamente la muerte del paciente puede ocurrir en cualquier momento. Se recomiendan que los ECG's de estos pacientes se monitoreen constantemente.

El principal problema en el monitoreo del ECG es saber qué secuencia de eventos ocurren en el sistema que produce el ECG. Esta secuencia es llamada interpretación de las señales. Una vez que una secuencia ha sido encontrada, no es fácil decir cuando el sistema no está funcionando apropiadamente.

Una complicación aparece en la interpretación de este problema, el hecho de que potenciales generados por algunos músculos ajenos al corazón aparecen en el ECG, los examenes de los pacientes pueden producir señales como las mostradas en la fig. 3.1. Un sistema de monitoreo deberá ser tolerante a detección de errores causados por el ruido.

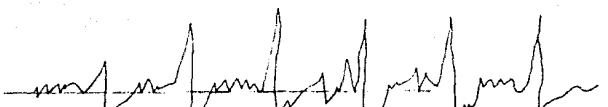


Fig. 3.1 Señal de ECG que incluye ruido.

El problema de interpretación, para éste modelo, fué descrito en el punto 1.3 del capítulo uno, donde se especifica el modelo de comunicación para la observación del sistema de ruido.

En este caso, la fuente de información es el sistema ritmo, que modelado por un AEEF (**Autómata Estocástico de Estados Finitos**), que opera en tiempo discreto y sus salidas representan a las fuentes de estimulación a los músculos del corazón.

Los músculos del corazón son representados como un dispositivo codificado. Cuando un músculo recibe estímulos el modelo ritmo, genera un pulso fijo en la señal del ECG durante el intervalo que sigue al estímulo.

El ECG idealizado, generado por los músculos del corazón, es transmitido a la superficie del cuerpo, donde se encuentran colocados los electrodos del ECG. En éste punto la señal codificada es mezclada con un ruido de diversas músculos, misma que es representada por el canal que tiene el ruido.

Finalmente, la señal del ECG es decodificada en una secuencia discreta de símbolos por un detector que es usado para localizar los pulsos de activación en la salida del canal. Asumiendo que cada pulso es detectado con probabilidad de uno, pero detecciones falsas ocurren con probabilidad diferente de cero. La transformación de la salida del modelo de ritmo es modelada como un canal ruidoso sin memoria.

El sistema total, compuesto del modelo de ritmo y la transformación de sus salidas dentro de las salidas de los detectores, tienen la forma de un AEEF. La salida del sistema total es un detector de salida.

El problema de interpretación se define en terminos de este sistema total. Sea el modelo de ritmo en el estado w_0 en el tiempo 0. Entonces durante el intervalo de tiempo de 1 a N, sea $\underline{w}(\underline{a})$ la secuencia de estados del modelo ritmo, y $\underline{d}$ la secuencia del detector de salida. Entonces dados $\underline{d}$ y w_0 el problema de interpretación se reduce a encontrar $\underline{w}$ y $\underline{a}$, tales que $P(\underline{a},\underline{w}|w_0,\underline{d})$ sea el máximo, donde P indica la probabilidad de partir del alfabeto d, en el tiempo w_0, y generar la secuencia de estados w(a).

Un procedimiento de búsqueda de árbol es usado para resolver éste problema. Para un detector de salida dado se define una secuencia de árbol. La raíz del árbol representa el estado inicial del sistema total y las direcciones del árbol representan la secuencia de los estados del sistema total. Un algoritmo de búsqueda de árbol de costo uniforme es usado para encontrar la secuencia de estados con mayor probabilidad.

Se utiliza una versión simplificada del problema de interpretación del ECG para ilustrar el enfoque, en esta versión únicamente se toma en cuenta la onda R del ECG. El procedimiento puede ser usado en una situación de monitoreo, para probar la ocurrencia regular de onda R normal.

EL PROBLEMA DE INTERPRETACION.

En esta sección se define el problema de interpretación de la salida de un AEEF en el que este autómata se usa como un generador, de manera que el estado final se omite en la definición.

UN AEEF SE DEFINE DE LA SIGUIENTE MANERA:

Un AEEF es el cuadruple $M = (\Sigma, \Omega, p, w_0)$, donde se asume que el tiempo es discreto, tomando los valores enteros de 0,1,2,...,n donde:

Σ, es un conjunto finito de simbolos denominado el alfabeto de salida, donde en cada tiempo exactamente un simbolo de este conjunto es observado en la salida de M.

Ω, es un conjunto finito de estados denominado el espacio de estados.

w_0, es el estado unicial, el estado de M en el tiempo cero.

p, la función de probabilidad de transición, es una función de probabilidad condicional discreta tal que ...

$P(a,w'|w) >= 0$, para todo w,w' que pertenece$^{\Omega}$ y todo a que pertenece a $\sum$.

La función P debe satisfacer lo siguiente:

$$\sum_{a \in \Sigma} \sum_{w' \in \Omega} P(a,w'|w) = 1$$

EL AUTOMATA OPERA DE LA SIGUIENTE MANERA:

Si en el tiempo **t** el autómata está en el estado **w**, entonces el tiempo **t+1** producirá una salida **a**, y entrará al estado **w'** tal que ... $P(a,w'|w_0) > 0$.

En seguida la secuencia de estados y salidas son denotados por símbolos subrayados. Sea $\underline{a} = a_1 a_2 .. a_N$ una secuencia de símbolos de salida y los símbolos $\underline{w} = w_1 w_2 ... w_N$ una secuencia de estados. La probabilidad $P(\underline{a},\underline{w}|w_0)$ puede calcularse por ...

$$P(\underline{a}, \underline{w}|w_0) = \prod_{i=1}^{n} P(a_i,w|w_{i-1})$$

Suponiendo que se observa la salida de la máquina durante el intervalo de tiempo t = 1,2,...,N, y sea $\underline{w}$ la secuencia de estados internos de la máquina durante ese intervalo. El problema de interpretar la secuencia de salida $\underline{a}$ es hallar la secuencia de estados internos que produce dicha salida $\underline{a}$.

Puede no existir secuencia de estados, tales que $P(\underline{a},\underline{w}|w_0) > 0$ en cuyo caso ésta se rechaza, por otro lado podrán existir más de una secuencia de estados, en éste caso la regla de decisión de error mínimo pueden usarse para elegir la secuencia de estados internos que maximiza $P(\underline{w}|\underline{a},w_0)$. Dado que :

$$P(\underline{w}|\underline{a}, w_0) = \frac{P(\underline{a},\underline{w}| w_0)}{P(\underline{a} | w_0)}$$

y que $P(\underline{a}|w_0)$ es la misma para todas las elecciones de $\underline{w}$, esto es equivalente a maximizar $P(\underline{a},\underline{w}|w_0)$.

SOLUCION AL PROBLEMA DE INTERPRETACION.

Para una secuencia de salida dada $\underline{a} = a_1 a_2 \ldots a_N$ se define un árbol tal que la trayectoria de la raíz a cada nodo representa una secuencia de estados. Se asocia a cada nodo la probabilidad de la secuencia de estados que él representa. Este árbol puede entonces ser examinado para encontrar la secuencia de estados $\underline{w}$ para la cual $P(\underline{a},\underline{w}|w_0)$ sea máxima. Si no existe tal secuencia de estados entonces la secuencia $\underline{a}$ se rechaza.

Se denomina el conjunto de nodos en el árbol por V. La raíz del árbol es un elemento V denominado v_0. Cada nodo en V está etiquetada con un estado de . La raíz se etiqueta con el estado inicial w_0.

Los nodos se arreglan en niveles. Un nodo en el nivel i, para i = 0,1,...,N, etiquetado con w, corresponde a la posibilidad de que en el tiempo i el autómata estuviera en el estado w. Sea V_i el conjunto de nodos en el nivel i del árbol, V_0 consiste unicamente del nodo raíz.

Para una secuencia dada de simbolos de salida $\underline{a}$, se define el conjunto de salidas sucesores del estado **w** por $S(w,a) = \{w' \mid P(a,w'|w) > 0\}$. Los nodos sucesores de un nodo v V_i etiquetado **w**, están en V_{i+1} y representan las salidas a_{i+1} sucesores de **w**, para $i = 0,1,2,\ldots,N-1$.

De modo que para cada **w'** $S(w,a_{i+1})$ existe un nodo v' en V_{i+1} etiquetado como **w'**.

Existe una secuencia única de arcos y nodos, denominada una trayectoria, que conecta la raíz a cada nodo del árbol. La secuencia de etiquetas de estados en los nodos en una trayectoria se denomina el rendimiento de un nodo y puede definirse recursivamente. El rendimiento de la raíz es $Y(v_0) = \lambda$,

la secuencia vacía. Si el nodo **v'**, etiquetado **w'**, no es la raíz y el nodo **v** es el padre de **v'**, entonces $Y(v') = Y(v)w'$.

Por cada nodo **v** en el árbol, está asociado un valor de probabilidad $\lambda(v)$. Para la raíz, $\lambda(v_0) = 1$. Denotemos los primeros i símbolos de una secuencia $\underline{a}$ por $a = a_1a_2...a_i$. Si **v** V_i con ganancia $Y(v) = \underline{w}$, entonces $\lambda(v) = P(\underline{a}i,\underline{w}|w_0)$. $\delta(v)$ puede ser calculada recursivamente como sigue:

Si $v' \in V_i$, para $i = 1,2,...,N$, con etiqueta **w'**, y si el nodo **v**, etiquetado **w**, es el padre de **v'**, entonces $\delta(v') = \delta(v)P(a_iw'|w)$.

Para encontrar el nodo v^0 en v_N tal que $\lambda(v^0) = \max_{v \in V_N}(\lambda(v))$

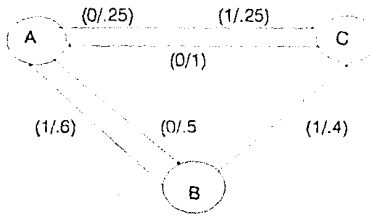
puede usarse un algoritmo de búsqueda ordenada, aunque excepto por razones prácticas, no es necesario usar algoritmos de búsqueda sino simplemente construir el árbol.

EJEMPLO SIMPLE.

Sea el siguiente autómata :

$\Omega = \{A,B,C\}$, $\Sigma = \{0,1\}$ y $w_0 = A$.

La función de probabilidad de transición se define por el siguiente diagrama:



donde el espacio de estados del autómata estáan indicados por las etiquetas del nodo. Un arco desde el estado w al estado w' se encuentra etiquetado con (a/q), donde a es la secuencia de salida y q la probabilidad de elección de salida. El AEEF contiene tres estados.

Suponga que se tiene la siguiente secuencia de estados: 010111. En la figura 3.4 se muestra el árbol de búsqueda para la secuencia anterior. El símbolo de estado en el nodo está representado por su etiqueta, el número dentro del nodo representa la probabilidad que tiene asociada. Los arcos están etiquetados con la salidad e probabilidad de transición del padre a hijo.

En éste árbol la raíz está etiquetado con el estado inicial A, y tiene un valor de probabilidad 1.0, después de analizar la secuencia de salida se ve que en el nivel 5 sólo existe un estado, entonces la única secuencia posible de estados es ABABAC.

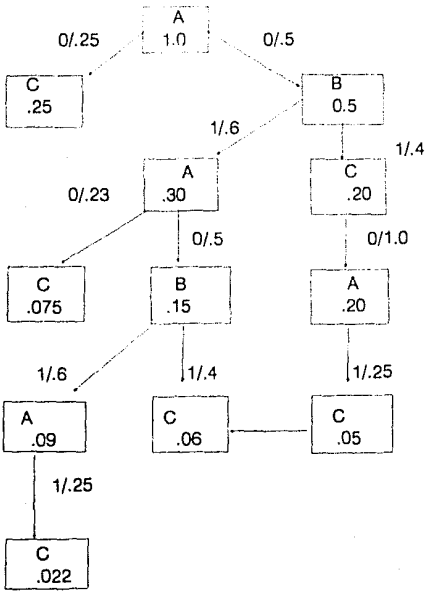


Fig. 3.2 Arbol de búsqueda para el AEEF.

Uno de los métodos que pueden emplearse para resolver el problema, es con el uso de una gramática regular, que define un caso tipíco, a partir del cual puede considerarse una clase de cadenas, mismas que pueden analizarse ayudandose de una métrica o función de error, éste enfoque es poco recomendable para los efectos de monitoreo debido a que la presencia de ruido puede facilmente conducir a resultados erroneos. Otro modelo que puede aplicarse es el usado por Stockman, el cual aplica un método en el que la segmentación y reconocimiento de primitivas son guiados por la sintáxis de las formas de onda.

En el apendice B se muestra la codificación que genera el electrocardiograma, basado en el autómata estocástico de estado finito.

CAPITULO 4.

INTERPRETACION DE ELECTROCARDIOGRAMAS POR EL MODELO DE HOROWITZ.

El reconocimiento de patrones, como se dijo en el capítulo uno, tiene aplicación en el análisis y reconocimiento de formas. Esta contiene un conjunto de propiedades.

Para hacer el análisis de una onda electrocardiografica, se divide en segmentos pequeños, cada una parecida a una onda primitiva. Estas ondas primitivas darán forma al electrocardiograma.

Cada una de las ondas primitivas, tendrán un caracter que las represente para formar la cadena, asi el reconocedor, hecho en un lenguaje de contexto libre, indicará si la cadena pertenece o no al modelo. El reconocedor se obtiene a partir de una expresión regular, que fundamenta las reglas en el comportamiento del corazón al encontrar en el ECG sólo rectas con pendientes positivas, rectas con pendientes negativas y rectas con pendiente cero.

4.1 MODELO MATEMATICO.

Horowitz [6], aplicó el aprovechamiento sintáctico para obtener un reconocimiento de ondas. Realizó una aproximación lineal, sobre una onda producida, el cual fué codificada como una cadena de símbolos terminales o primitivos. Para realizar el reconocimiento de las ondas se usa una gramática determinística de contexto libre.

El modelo matemático del cual se tomaron las bases fue el siguiente, considerando un conjunto de puntos discretos (X_1,Y_1), (X_2,Y_2), ...,(X_i,Y_i), ...,(X_n,Y_n) representada por la función $y = f(x)$, con $x_{i<xi+1}$ para todo $i = 1,2...,n$.

Para analizar la onda se toman tres pares de puntos $[(X^l,Y^l)$, (X^m,Y^m), $(X^r,Y^r)]$. El par (X^m,Y^m) es el punto medio de la onda, (X^l,Y^l) es el punto inicial de la onda del lado izquierdo, (X^r,Y^r) es el punto terminal de la onda lado derecho y además se sabe que $X^l < X^m < X^r$. Donde de los puntos anteriormente definidos, el punto (X^m,Y^m) es un máximo local si y sólo si, son satisfechas las siguientes expresiones simultaneamente:

Para todo i, $X^l <= X_i <= X^r$, $Y^m >= Y_i$ (4.1)

Para todo i, $X^l <= X_i <= X^m$, $Y_i <= Y_{i+1}$ (4.2)

Para todo i, $X^m <= X_i <= X^r$, $Y_i >= Y_{i+1}$ (4.3)

Existe un i tal que $X_i = X^l$, $Y_{i-1} >= Y^l$ (4.4)

Existe un i tal que $X_i = X^r$, $Y^r <= Y_{i+1}$ (4.5)

La expresión (4.1) describe la propiedad de l máximo local, (4.2) y (4.3) establecen los puntos no negativos a la izquierda y no positivos a la derecha requidos para los dos lados del máximo local.

Un punto no negativo donde (X^m,Y^m) es un mínimo local, puede ser definido en forma similar, solo por el cambio de dirección de las ecuaciones (4.1) a (4.5).

Debemos notar que X^m puede no ser el único, para los puntos X^l y X^r fijos.

Las siguientes tres expresiones se requieren para fijar el punto X^m como el punto central en un punto positivo, X^l como el punto final de la izquierda y X^r como el punto final de la derecha.

Existe i,j tal que $X_i \neq X_j$

$$Y_l = Y_j = Y^m \{(Y_{l-1} < Y_l) \text{ y } (Y_l > Y_{l+1}) \text{ y } X^m = (X_l + X_j)/2\} \quad (4.6)$$

Existe un i tal que $X_i = X^l$, $Y^l < Y_{i+1}$ (4.7)

Existe un i tal que $X_i = X^r$, $Y_{i-1} > Y^r$ (4.8)

Las dos expresiones siguientes son necesarias para eliminar multiples ocurrencias de X^l y X^r.

Existe i,j tal que $X_i = X^l$, $X_j <= X^l$

$$Y_j = Y^l \{(Y_{j-1} > Y_j) \text{ --> }$$

$$\forall k \mid X_j <= X_k <= X_j (Y_k <= Y^l) \} \quad (4.9)$$

Existe i,j tal que $X_i = X^r$

$$X_j = > X^r,\ Y_j = Y^r \{(Y_j <= Y_{j+1}) --> \forall k \mid X_i <= X_k <= X_j\ (Y_k = Yr)\ \} \qquad (4.10)$$

Las expresiones (4.6) a (4.10) son argumentos establecidos para puntos positivos. Para los puntos negativos sobre las y's son inversos en sus desigualdades de dichas expresiones.

Muy a menudo se usa el criterio de derivadas para detectar el punto extremo del modelo. Para ésto,

$$d_i = \frac{Y_{i+1} - Y_i}{X_{i+1} - X_i} \qquad (4.11)$$

corresponden a f', la primer derivada de la función análoga.

Por (4.1) a (4.3) y (4.6) la primer diferencia inversa (posiblemente hasta cero), si y sólo si, el extremo local hasido encontrado.

Existe i,j tal que

$$Signo(d_i) \neq Signo(d_j)\ \{[(i+1=j)\ \acute{o}\ \forall k \mid i < k < j (d_k = 0) --- >$$

que existe un k tal que

$$X_k = X^m\ [\ X_k = (X_{i+1} + X_j)/2\ ]\} \qquad (4.12)$$

Con estas expresiones se determina la sintáxis requerida para generar la expresión regular y se define en el punto 4.2.

4.2 INTERPRETACION DEL ELECTROCARDIOGRAMA.

OBTENCION DE LA EXPRESION REGULAR.

Como se ha observado, se trata de determinar cuál es la dirección del punto analizado (pendiente positiva, pendiente negativa y pendiente cero), que nos proporcionará el alfabeto a incluirse en la cadena. La cadena podrá ser reconocida y se indicará si es aceptada o no.

Para eso asignamos a el i-ésimo par de puntos de la onda $[(X_i,Y_i), (X_{i+1},Y_{i+1})]$, el símbolo w_i que contiene las características del segmento de línea, uniendo los dos puntos. Asignando $W = w_1w_2...w_i...w_{n-1}$, es la cadena resultante de la codificación de la onda, tal que:

$(w_i = p)$ implica $(d_i > 0)$ (4.13)

$(w_i = n)$ implica $(d_i < 0)$ (4.14)

$(w_i = 0)$ implica $(d_i = 0)$ (4.15)

donde p denota una pendiente positiva, n denota una pendiente negativa y 0 denota falta de pendiente.

Una expresión regular o un lenguaje de estado finito sobre el alfabeto {p, n, 0} puede construirse denotando un conjunto infinito de subcadenas de w $W = \{p, n, 0\}^*$ que representen puntos positivos y negativos.

Podemos adoptar las siguientes notaciones. Si P y Q son expresiones regulares, entonces:

1).- P + Q es la unión de los conjuntos denotados por P y Q.

2).- PQ es el producto o concatenación de los conjuntos denotados por P y Q.

3).- P^* es la transitividad reflexiva (cerradura de Kleen), del conjunto denotado por P, como se menciono en el capítulo dos.

Por (4.7), el lado izquierdo del punto positivo es iniciado con el símbolo P; por (4.12) podemos contener algún número (posiblemente cero), de los símbolo p y 0 solamente; y por (4.1) podemos finalizar con el símbolo p. La expresión regular para obtener el lado izquierdo de un punto positivo, está dado por:

$$P = p + p(p + 0)^*p \qquad (4.16)$$

De manera similar, el lado derecho del punto positivo puede iniciar y finalizar con el símbolo n y obtener algún número de n's y 0's solamente, por (4.1), (4.3) y (4.8).

Una expresión regular que represente la forma de onda en su lado negativo, está dado por:

$$Q = n + n(n + 0)^*n \qquad (4.17)$$

Combinando (4.16) y (4.17), se tiene un expresión completa, para el lado positivo:

$$R = P0^{*}Q \qquad (4.18)$$

y para el lado negativo:

$$S = Q0^{*}P \qquad (4.19)$$

De esta manera podemos tener una cadena ininterrumpida de puntos que formen la onda del electrocardiograma.

$$W = 0^{*}...R0^{*}S0^{*}R0^{*}S...0^{*} \qquad (4.20)$$

Tomando los elementos de las expresiones (4.16) a (4.20), puede construirse la gramática de contexto libre determinístico G que reconozca los puntos positivos y negativos de la onda que está representada por la cadena W.

$$G = (V_N , V_T , P , \{ W \})$$

donde:

V_N = {W, <onda, <punto+>, <punto->, <pos1>, <neg1>, <pos2>, <neg2>, <cero>}

V_T = {p, n, 0}

y P:

W-------> <cero> <onda> <cero>

W-------> <cero> <onda>

W-------> <onda> <cero>

W-------> <onda>

W-------> <cero>

<onda> -------> <punto + >

<onda> -------> <punto->

<onda> -------> <pos1>

<onda> ------> <neg1>

<punto + > ------> <punto-> <cero> <neg1>

<punto + > ------> <punto-> <neg1>

<punto + > ------> <pos1> <cero> <neg1>

<punto + > ------> <pos1> <neg1>

<punto-> ------> <punto + > <cero> <pos1>

<punto-> ------> <punto + > <pos2>

<punto-> ------> <neg1> <cero> <pos1>

<punto-> ------> <neg1> <pos1>

<pos1 > ------> <pos1> P

<pos1 > ------> <pos2> P

<pos1 > ------> P

<neg1 > ------> <neg1> n

<neg1 > ------> <neg2> n

<neg1 > -------> n

<pos2 > -------> <pos1> 0

<pos2 > -------> <pos2> 0

<neg2 > -------> <neg1> 0

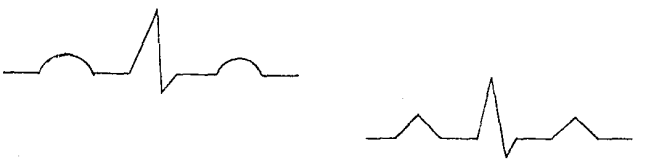
<neg2 > -------> <neg2> 0

<cero > -------> <cero> 0

<cero > -------> 0

Esta gramática permitirá reconocer una cadena de caracteres, que contengan el alfabeto {p, n, 0}, que represente la forma de onda generada por el electrocardiograma. La cadena será de menor tamaño que la obtenida en el capítulo anterior que sólo contenía el alfabeto {0, 1}.

La fig. 4.21 muestra la onda del electrocardiograma, la cadena obtenida en la aplicación del modelo y la onda que generaría la cadena al ser interpretada con las reglas del modelo.



W = 0pn00pppnnnnp00pn00.

Fig. 4.21 Onda del electrocardiograma, cadena obtenida y onda generada al interpretar la cadena.

La fig. 4.22 muestra el segmento DI del ECG de un paciente al que se le diagnóstico Fibrilación Auricular y ritmo variable. La cadena que representa al segmento DI es el siguiente:

pnp00000ppn00pppppppppppnnnnnnnnnnnnpp0000np0p
pn0000000ppnnpppppppppppnnnnnnnnnnnnp0000000p
pn00000000ppn00pppppppppnnnnnnnnnnnnp00000n
pp000000000ppnnpppppppppnnnnnnnnnnnnp000000p
pn0000000ppnnpppppppppnnnnnnnnnnnnp000000p

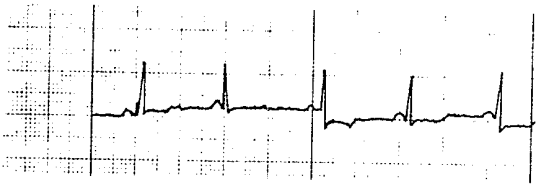


Fig.4.22 Derivación DI perteneciente a un ECG de un paciente con Fibrilación Auricular y ritmo variado.

Para encontrar el ritmo variado en la cadena, se puede hacer obteniendo la cadena del primer segmento y la longitud de ceros que separan a cada segmento, si la longitud es diferente en alguno de ellos se estaría hablando de un cambio ritmo.

La detección de una fibrilación auricular es más fácil determinarlo de manera directa del ECG, ya que tiene la forma de una sierra. Esta representación en la cadena no tiene la misma facilidad, ya que aparecerían caracteres p y n continuos.

La cadena obtenida en éste modelo es más pequeña en comparación a la generada con el alfabeto (0, 1), en el capítulo dos. De la misma forma que en el reconocedor de expresion regular de alfabeto (0, 1), sólo puede representarse los elementos del ECG en base a trazos rectos, pendiente positiva, negativa y pendiente cero el cual es una limitante de este modelo. Las ondas P y T son representadas como una aproximación todavía.

En el siguiente capítulo se analizará el modelo de Stockman que intenta hacer una aproximación en base a la separación de la onda en segmentos, que sea lo más idénticas a las primitivas del ECG. Las primitivas permitiran hacer una aproximación más exacta al ECG y generará una cadena más sencilla de ser analizada para obtener el diganóstico.

CAPITULO 5.

INTERPRETACION DE ELECTROCARDIOGRAMAS POR EL MODELO DE STOCKMAN

El modelo que analizaremos, realiza la partición de la onda en varios segmentos que son determinados por el conjunto de primitivas que forman la onda.

Así en el electrocardiograma pueden aparecer pendientes positivas largas o cortas, pendientes negativas largas o cortas, rectas, etc., como se vió en el capítulo uno.

Con la ayuda de una gramática de contexto libre, se hace el reconocimiento de la cadena que resulta en éste modelo. Las reglas que determinan la interpretación del electrocardiograma es más sencillo a las generadas por los modelos anteriores.

5.1 MODELO MATEMATICO.

En la implementación del método realizado por Stockman, se tomaron algunas consideraciones matemáticas que analizaremos. En esta implementación, la rutina de aproximación lineal es utilizada, sirviendo como un preprocesador, que constituye una variación del algoritmo "une-y-separa" [13], es como sigue:

1).- Dividir la onda en un número arbitrario de segmentos de igual longitud con puntos finales coincidentes, y calcular el error de aproximación lineal cuadrada para cada aproximación.

2).- Separar los segmentos que contengan la mayor disminución de la suma de los valores de error cuadratico del segmento (E), hasta que sea menor a una Tolerancia de Error Especificada (TEE).

3).- Ajustar los segmentos que contengan los incrementos menores en E, mientras E sea menor que TEE.

4).- Ajustar los segmentos en los puntos finales para el mínimo error E, entonces si algún ajuste se ha realizado regresar al punto tres.

La elección de una norma de error cuadratica media debe obtenerse. Para definir una norma de error lineal discreta en una dimensión a través de la coordenada y, se usa la siguiente expresión:

$$L_k = [\sum_i | y_i - (ax_i + b) |^k]^{1/k} \qquad (5.1)$$

donde a y b son los coeficientes (pendiente y constante) de la línea de aproximación del segmento, al conjunto de puntos $\{(x_i, y_i)\}$

Minimizando L_1, la norma de error media producida:

$$\min_{a,b} \left[\sum_i | y_i - (ax_i + b) | \right] \qquad (5.2)$$

Desafortunadamente los valores de a y b que satisfacen la expresión (5.2) no son únicos.

Si minimizamos L, la norma de máximo error produciría

$$\min_{a,b} \{ \max_i [| y_i - (ax_i + b) |] \} \qquad (5.3)$$

el cual es el más sensitivo a los puntos exteriores, aunque única (si dos x_i no son iguales), se requiere un programa lineal iterativo para calcular los valores óptimos de a y b que satisfagan (5.3).

Minimizando L_2, se producira una norma de error cuadratica

$$\min_{a,b} \{ \sum_i [y_i - (ax_i + b)]^2 \} \qquad (5.4)$$

De aquí se puede obtener los valores únicos de a y b que satisfagan (5.4), y queda como sigue:

$$a = \frac{n \sum_i x_i y_i - \sum_i x_i \sum_i y_i}{n \sum_i x_i^2 - \left(\sum_i x_i \right)^2} \qquad (5.5)$$

$$b = \frac{1}{n} \left(\sum_i y_i - a \sum_i x_i \right) \qquad (5.6)$$

Con **a** indefinido, si y sólo si, todos los x_i son iguales. La norma media de L_2, a lo largo de la coordenada y es invariante para algún $x' = cx$ (cambio en escala x) y linealmente proporcional a **c**, para $y' = cy$ (cambio en la escala de y). La suma de los valores del segmento de error es minimizado para ser compatible con la norma de L_2. El método no garantiza que la solución mínima global sea obtenida, aunque se tiene una probabilidad muy alta de que el punto extremo y los limites se encuentren en el conjunto óptimo local.

El resultado principalmente depende de TEE, el cual controla la resolución del ajuste.

Si la onda que se analiza es bipolar y tiene una forma asimétrica, tiene que tomarSE una amplitud madia, las constantes se pueden generar realizando y obteniendo sus valores de un análisis estadístico. Este puede ser calculado haciendo uso de los percentiles o aprovechando el histograma, donde se observará la pendiente de la línea del segmento base. Otra forma de obtener la línea del segmento base es calculandolo por una descripción heurística.

5.2 INTERPRETACION DEL ELECTROCARDIOGRAMA.

Stockman [13], aplicó el reconocimiento sintáctico a la onda producida por el pulso de la carótida. El uso de un parser que diseñará el análisis sintáctico de arriba-abajo y no-izquierda- derecha. La segmentación y el uso de reconocimiento primitivo son esencialmente usados para la sintaxis de la onda, las selecciones primitivas son segmentos de onda líneal y parabólicas.

La salida del preprocesador consiste de una onda compacta (x_1,y_1), $(x_2,y_2)...(x_i,y_i)....(x_{ns+1},y_{ns+1})$ y el conjunto de coeficientes del segmento de aproximación lineal $\{(a_j,b_j)\}$ para $j=1,...,ns$. Notamos que (x_j,y_j) es el punto final del segmento izquierdo (derecho). Se tiene una probabilidad muy baja de $a_j=0$ en algún segmento j. Se requiere de la tolerancia de la pendiente (STOL) es necesario para preveer un segmento relativo horizontal con una pendiente cero.

Para la aproximación de la onda, la cadena representativa será $W= w_1w_2...w_j...w_{ns}$, donde la w_j es asignada como sigue:

$(w_j=p)$ ímplica $(a_j > STOL)$ (5.7)

$(w_j=n)$ ímplica $(a_j < -STOL)$ (5.8)

$(w_j=0)$ ímplica $(|a_j| <= STOL)$ (5.9)

Que es similar a la obtenida por el método de Horowitz.

La onda producida por el pulso de la carótida tiene dos segmentos realizados, la primer división es la contenida para la Sistólica y la segunda para la Diastólica. La unión de los segmentos es hecho exactamente en la parte final de la Sisiólica y en la inicial de la Diastólica.

La gramática que se obtuvo es la siguiente:

< onda-carotida > --> < Sistólica > < Diastólica >

< Sistólica > --> < pen_larga > < máxima > pen_neg

< máxima > --> < M1 > < M2 > < M3 >

< máxima > --> < med_pos > < M3 >

< máxima > --> < M1 > med_neg

< Diastólica > --> < onda_diacrotica > pen_larga

< Diastólica > --> < pen_larga >

< onda_diacrotica > --> CAP

< onda_diacrotica > --> HOR

< onda_diacrotica > --> PEK

< onda_diacrotica > --> PEK CAP

< M1 > --> LSHOLD

< M2 > --> CCUP

< M3 > --> RSHOLD

< onda_pos > --> CAP

< onda_pos > --> CAP pen_neg

< onda_neg > --> VCUP

< onda_neg > --> VCUP pen_pos

HOR línea: corta o pendiente cero.

CAP,PEK,CCUP,VCUP -- parabola.

RSHOLD -- parabola, máxima parabola de la mitad derecha.

LSHOLD -- parabola, máxima parabola de la mitad izquierda.

El modelo generado por Stockman, como se ha mencionado, realiza la división de la onda en segmentos, que logra una mayor aproximación a ésta, comparada con la de Horowitz.

Un analizador sintáctico que permite tener una mayor certeza y aproximación a la onda electrocardiográfica se muestra enseguida:

$$G = \{ V_N, V_T, P, S\}$$

Donde :

$V_T = \{0,A,B,C,D,E,F,G,H,I,J,K,L\}$

V_N = {<ECK>, <PPR>, <QRS>, <STT>, <Línea_pos>, <P>, <PR>, <Línea_neg>, <Q>, <R>, <S>, <S1>, <S2>, <STT>, <ST>, <T>, <Línea_larga_pos>, <Línea_corta_pos>, <Línea_larga_neg>, <Línea_corta_neg>, <Línea_recta>, <onda_positiva>, <onda_negativa>, <pico_positivo>, <pico_negativo> <Línea>}

P :

<ECG> --> <PPR> <QRS> <STT>

<PPR> --> <Línea_recta> <P> <PR>

<PR> --> <Línea_pos>

<PR> --> <Línea_neg>

<PR> --> <Línea_rec>

<QRS> --> <Q> <R> <S>

<QRS> --> <R> <S>

<QRS> --> 0

<Q> --> <Línea_corta_neg>

<Q> --> 0

<R> --> <Línea_corta_pos>

<R> --> <Línea_larga_pos>

<R> --> <Línea_corta_neg>

<R> --> <Línea_larga_neg>

<S> --> <S1> <S2>

<S1> --> <Línea_larga_pos>

<S1> --> <Línea_corta_pos>

<S1> --> <Línea_larga_neg>

<S1> --> <Línea_corta_neg>

<S2> --> <Línea_larga_pos>

<S2> --> <Línea_corta_pos>

<S2> --> <Línea_larga_neg>

<S2> --> <Línea_corta_neg>

<S2> --> 0

<STT> --> <ST> <T> Línea

<ST> --> <Línea_larga_pos>

<ST> --> <Línea_corta_pos>

<ST> --> <Línea_larga_neg>

<ST> --> <Línea_corta_neg>

<ST> --> <Línea_rec>

<ST> --> 0

<P> --> <onda_positiva>

<P> --> <onda_negativa>

<P> --> <pico_positivo>

<P> --> <pico_negativo>

<T> --> <onda_positiva>

<T> --> <onda_negativa>

<T> --> <pico_positivo>

<T> --> <pico_negativo>

<Línea_pos> --> B

<Línea_neg> --> C

<Línea> --> A

<Línea_rec> --> D

<Línea_larga_pos> --> E

<Línea_corta_pos> --> F

<Línea_larga_neg> --> G

<Línea_corta_neg> --> H

<onda_positiva> --> I

<onda_negativa> --> J

<pico_positivo> --> K

<pico_negativo> --> L

S = {ECG}

Pueden cambiarse los puntos terminales que se tienen en el reconocedor sintáctico, por los elementos que se tienen en el alfabeto.

Los elementos primitivos que se utilizan en el reconocedor sintáctico son los siguientes:

Línea_rec : Es una línea recta con pendiente cero corta.

Línea_neg : Es una línea recta con pendiente negativa con longitud menor a 3 mm.

Línea_pos : Es una línea recta con pendiente positiva con longitud menor a 3mm.

Línea_larga_pos: Línea recta con pendientente positiva y longitud mayor de 3mm.

Línea_corta_pos : Línea recta con pendiente negativa y longitud menor de 3 mm.

Línea_larga_neg : Línea con pendiente negativa, longitud mayor de 3 mm.

Línea_corta_neg : Línea con pendiente negativa y una longitud menor de 3 mm.

Onda_positiva : Onda que tiene forma de parábola positiva.

Onda_negativa : Onda en forma de paráboła negativa.

pico_positivo : Es una onda en forma de **V** invertida.

pico_negativo : Es una onda con forma de **V**.

linea : Es una línea recta con pendiente cero larga.

0 : Falta del segmento.

Para demostrar la forma en la que se reduce el tamaño de la cadena usando el método de Stockman, comparada con los métodos anteriores, se realiza el ejemplo han hecho en los capítulos anteriores.

La onda que se analiza se representa en la Fig. 5.10, y la cadena que enera es la siguiente:

AJD0EGFDJ

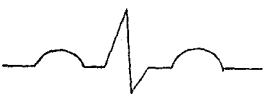


Fig. 5.10 Complejo de un ECG normal.

La fig. 5.11 muestra el segmento DI de un electrocardiograma que tiene un ritmo variado y se le diagnóstica además una fibrilación auricular.

La cadena de caracteres que representa el ECG mostrada en la fig. 5.11 son las siguientes:

KLKLKLJEGFKKKKKKFKKKKKKKKKJEGFKKKKKKKKFAJEGFAFAJEGFAFAF KDEGFA

La cadena que se genera es más pequeña que la producida por el modelo de Horowitz. Para saber si existe fibrilación auricular basta con observar si existen algunos caracteres KL en la cadena y que no represente a la onda P.

Para determinar el ritmo se debe conocer la distancia que separa la onda T de P, y saber si existe un caracter A o D entre cada segmento, como la longitud que representa no es la misma se determina la causa de ritmo variado.

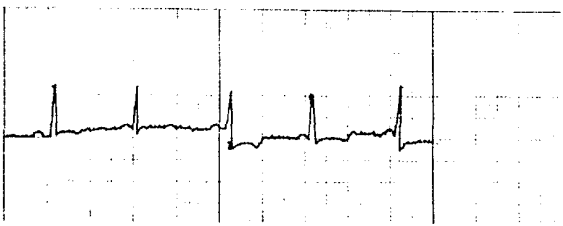


Fig. 5.11 Derivacion DI de un electrocardiograma con fibrilación auricular.

Las ventajas que tiene el modelo es que contiene todos los elementos representados por caracteres, del ECG es decir, el conjunto de primitivas que constituye el electrocardiograma tiene asociado un caracter para cada una de ellas. De esta forma se aproxima en mayor medida a la cadena de caracteres del ECG de un paciente dado.

Si consideramos el avance del área de la computación, ésta información podría ser analizada y manipulada por una base de Datos que contenga el conjunto de reglas que se requiere para poder determinar si es o no una cardiopatia el electrocardiograma que desea observarse.

Recordando que nuestro proceso es el reconocer cadenas que nos muestran el comportamiento del electrocardiograma y no el de generarlas. Los pasos que deben seguirse hasta antes de llegar a este punto, están

indicados en el capítulo uno, punto 1.3 y tienen la secuencia correspondiente hasta determinar el diagnóstico del paciente.

CONCLUSIONES.

Para representar los resultados obtenidos en los modelos para reconocer los electrocardiogramas: modelo de Horowitz, Stockman y de expresiones regulares, se hará en base a los segmentos DI y DII de un electrocardiograma original, Fig. 6.1.

El análisis se muestra en el orden siguiente:

a) Reconocedor de expresiones regular.

b) Reconocedor usando el modelo de Horowitz.

c) Reconocedor usando el modelo de Stockman.

Los segmentos que se muestran en la Fig. 6.1, pertenecen a una persona de 77 años de edad, que sufre de arritmia sinusal.

Fig. 6.1

Recordemos que el analizador, mencionado en el capítulo uno, punto 1.3, es el modelo que generaliza la forma de determinar y proporciona el diagnóstico del paciente, donde en base a un conjunto de reglas ésta es generada.

1) Reconocedor de expresiones regulares.

Retomando los elementos definidos en el capítulo dos, punto 2.2, donde se describe la forma de representación de los unos y ceros.

Las reglas definidas son las siguentes:

1. Línea recta con pendiente positiva.
11. Parábola.
111. Línea recta con pendiente cero.
1111. Línea recta con pendiente negativa.

La cadena que representa la onda electrocardiografica para el segmento DI, es el siguiente:

11101100111010011110000001000001110000111101110000
1100111010011110000001000001110000000000

y la cadena que nos representa a DII es:

11100110001110100011110000000010000001110011000 0
1110011000111010001111000000001000000111001100000 11100

Si recordamos que deben tomarse doce derivaciones y que para cada derivación se deben tomar 4 complejos, entonces la cadena que nos represente un electrocardiograma completo generará una cadena muy grande y que producirá un atraso de tiempo en el reconocimiento y en la interpretación de la cadena.

2) Modelo de Horowitz.

Horowitz, con la ayuda de una gramática de contexto libre y de una expresión regular, logró hacerse una aproximación lineal a una onda y que fue aprovechada para generar una cadena de símbolos primitivos que representará a éste.

Para la generación de la cadena se obtuvieron las siguientes reglas:

p. Si se interpreta una pendiente positiva.
n. Si se esta leyendo una pendiente negativa.
0. Si se interpreta una pendiente cero.

La cadena W que representa la gráfica de la Fig. 6.1, en el segmento DI es el siguiente:

00ppnn0ppnnnnnnnppppp0000p0000pn0ppnnnnnnnppppp0000p0000

La cadena W para el segmento DII se muestra enseguida:

00ppnn0pppnnnnnnnnnppppp00ppnn0pppnnnnnnnnnppppp00
ppnnn00

El electrocardiograma es representado por el modelo en forma cuadrada, es decir, las ondas no son consideradas como tal sino que son utilizadas como pendientes.

3) Reconocedor electrocardiografico usando el Modelo de Stockman.

El modelo de Stockman utiliza una gramática de contexto libre para generar el reconocedor, usa la partición de la onda en diferentes segmentos con la misma longitud, donde dichos segmentos son los primitivos de la gráfica que en la cadena son representadas por letras.

El alfabeto queda definido de la forma siguiente:

A. Línea con pendiente cero larga.

B. Línea con pendiente positiva media.

C. Línea con pendiente negativa media.

D. Línea con pendiente cero corta.

E. Línea con pendiente positiva larga.

F. Línea con pendiente positiva corta.

G. Línea con pendiente negativa larga.

H. Línea con pendiente negativa corta.

I. Parabola negativa.

J. Parabola positiva.

K. Pico positivo.

L. Pico negativo.

0. Indica falta del segmento.

La cadena que representa el segento DI, de la Fig. 6.1, es el siguiente:

AJABGEAHAJABGEAHA

y para el segmento DII:

AJABGEFKAJABGEFK

La cadena que resulta de la aplicación del modelo, tiene mayor aproximación al ECG original, ya que el alfabeto que usa representa a 12 producciones diferentes (primitivas), se brinda esta posibilidad.

La lectura y el análisis de la cadena para una interpretación y diagnóstico tendría una facilidad para un especialista, pero si el análisis se realiza por computadora éste sería más rápido y sencillo.

Si las reglas que se han generado para la interpretación y reconocimiento de las cadenas que produce el electrocardiograma son alimentadas en una base de datos que contenga todas las cadenas que indiquen estados normales y anormales, el diagnóstico que se obtenga corre menos riesgo a error.

En el caso de que la cadena obtenida de un paciente no se encuentre registrado en la base de datos, ésta debe analizarse para obtener su diagnóstico. Cuando esto se presente es recomendable almacenar la nueva cadena para utilizarse en casos posteriores; aunque si se cuenta con una base de datos inteligente ésta automaticamente la adicionaría.

Los trabajos que pueden producirse a partir de este proyecto, son el de elaborar el analizador de cadena que proporcione el diagnóstico de un electrocardiograma. En el desarrollo del proyecto se da de una forma u otra, una idea de los elementos que deben analizarse y del como se buscarían en la cadena, dependiendo del modelo reconocedor que se utilice. El uso de una base de datos agilizaría la obtención del diagnóstico.

Otro proyecto sería buscar un mecanismo que permita la conexión entre la computadora y el electrocardiograma. Cabe mencionar, como observación, que si cada segmento de la onda produce una señal eléctrica (voltaje), diferente puede tenerse un proceso de producción de la cadena a partir de éstos. Con esto podría generarse la cadena sin tener que analizar, ni rastrear la señal electrocardiográfica.

BIBLIOGRAFIA.

1.- Aho Hopcroft, Ulman, The Design and Analisys of Computer Algoritms, Addison Wesley, [1974]

2.- Ahula Narendra and Shachter B.J., Image Models, Computing Surveys, 4, December 1981.

3.- Blaise Liffiek, the Byte Book of Pascal, McGraw-Hill.

4.- Denes, P.B., A Scan-Type Graphics System for Iteractive Computing, [1975]

5.- Fu, K.S., Sequential Methods in Pattern recognition and Machine Learning, Academic Press, London and New York, [1972].

6.- Fu, K.S., Syntactic Pattern Recognition Aplications, Springer-Verlag, New York, [1977].

7.- Fukunaga,K., Introduction to StaticalPattern Recognition, Academic Press, New York and London, [1972].

8.- Henry, W.L., Crouse, L., Stinson R., Harrison,D.C., Computer Analysis of Cardiac Catheterization Data, Amer. J. Cardiology, [1968].

9.- Hopcroft, J.E, Ullman, J. D., Introduction to Automata Theory, Lenguages and Computation, Addison Wesley, [1979].

10.- Ingeman, P. 2., A Sintaxis Oriented Translator, New York, [1960].

11.- Minsky, M. L. , Computation: Finite and Infinite Machines, Prentice Hall, [1967].

12.- Sargur N. Srihari, Representation of Three-Dimensional Digital Images, Computing Surveys, 4, December 1981.

13.- Stockman, George, Kanal, L. N. and Kyle, M. C., StructuralPettern Recognition of Carotid PulseWaves Using a General Waverform System, Commun. ACM 19 688-645, December 1976

14.- Ullman, Introduction to Formal Languages & Automata Theory, Addison Wesley.

15.- Dr. Luis Alcocer Díaz Barreiro, Dr. Angel Gónzalez Caamaño, El Electrocardiograma, Ediciones Médicas Actualizadas S.A., 64 pp.

16.- Dr. Dubin, Dale, Electrocardiografía Practica (Lesión, trazado e interpretación), interamericana, 294 pp.

APENDICE A

PROGRAMA AUTOMATA.

```
{

        PROGRAMA QUE DADA UNA EXPRESION REGULAR, GENERA UN
AFD, LO MINIMINIZA, Y ESCRIBE UN PROGRAMA DE SALIDA, EL CUAL
SIMULA EL COMPORTAMIENTO DEL AUTOMATA


**************************************************************************}

program automata;

const
    epsilon = '/';
    uch = 'Z';
    apuno = 1;
    dosap = 2;
    totone = 255;
    cr = #13;
    esc = #27;

type

    tokens          =epsilon..uch;
    posiciones      = 0..totone;
    conjuntos       = set of posiciones;
    renglon         = array [tokens] of byte;

var

    alfabet, usados             : set of tokens;
    finales, eliminados         : set of byte;
    llave                       : array[0..50] of real;
    er                          : array[0..20] of char;
    caractua, carfinal          : posiciones;
```

```
    i,j,noedo,mayedo,edo,informest,
    priuno,pridos,ultimuno,
    nel,ultdos,indice                    : byte;
    cara,caracterf,carcont,ww,cri,crf,car: char;
    transpri                             : array[posiciones,tokens,1..2] of byte;
    transeg                              : array[byte] of renglon;
    conexpr                              : array[posiciones] of conjuntos;
    x1,x2                                : conjuntos;
    iguales, acexpre                     : boolean;
    x                                    : real;
    cont,xx                              : integer;


{**************************************************************************}


PROCEDURE junta(var priuno,ultimuno,pridos,ultdos : byte);

BEGIN
    transpri[edo + 1,epsilon,apuno]: = priuno;
    transpri[edo + 1,epsilon,dosap]: = pridos;
    transpri[ultimuno,epsilon,apuno]: = edo + 2;
    transpri[ultdos,epsilon,apuno]: = edo + 2;
    priuno: = edo + 1;
    ultimuno: = edo + 2;
    edo: = ultimuno
END;


{**************************************************************************}
```

```
function alfabeto(ac:char):boolean;

{  Este es el procedimiento que verifica que el elemento se encuentre dentro
   del alfabeto anteriormente definido                              *}

var
    c : integer ;
begin
   c:=ord(ac);
   if ((c = 48) and (c<=57) or (c >= 65) and ( c<= 90) or (c >= 97) and
     (c <=122)) then
     alfabeto:=true;
end;


{**************************************************************************}

procedure evaluacion;
{  Este es el procedimiento que valída la expresión regular              }
label inicio,fin,e1,e2,e3,e4,e5,e6,e7,e8,e9,e10,e11,e12,e13;
var
  i,a,pa : integer;
begin
    writeln('Introduzca su expresion regular');
    read(er);
    pa:= 0;
    i := 0;
    e2:
      if ((er[i] = '(') or (alfabeto(er[i]))) then goto e1
      else
      begin
        writeln('Error ...!, se esperaba "(" o elmento del alfabeto');
        goto inicio;
      end;
  e1:
```

```
  if (er[i] = '(') then goto e3
  else goto e4;
e3: pa:=pa+1;
i:=1+i;
if ((er[i] = '(') or (alfabeto(er[i]))) then goto e5
else
  begin
    writeln('Error ..!, se esperaba "(" o elmento del alfabeto');
    goto inicio;
  end;
e5:
  if (alfabeto(er[i])) then goto e4
  else goto e3;
e4: i:=1+i;
if ((er[i] = '+') or (er[i] = '.') or (er[i] = ')') or (er[i] = ESC)
          or (er[i] = '*')) then goto e6
else
  begin
    writeln('Error ..!, se esperaba operador ")" o fin de expresion');
    goto inicio;
  end;
e6:
  if ((er[i] = '+') or (er[i] = '.') or (er[i] = ESC) or (er[i] = '*'))
            then goto e7
  else goto e8;
e7:
  if ((er[i] = '+') or (er[i] = '.')) then goto e9
  else goto e10;
e9:
  i:=i+1;
  goto e2;
e8:
  pa := pa-1;
i:=i+1;
if (er[i] = ESC) then goto e11
```

```
    else goto e6;
    e10:
      if (er[i] = ESC) then goto e11;
    i:=i+1;
    if ((er[i] = '+') or (er[i] = '.') or (er[i] = ')') or (er[i] = ESC))
                then goto e12
    else
    BEGIN
     writeln('Error ..!, se esperaba operador binario, ")" o fin de expresion');
     goto inicio;
    END;
    e12:
       if ((er[i] = '+') or (er[i] = '.') or (er[i] = ESC)) then goto e13
       else goto  e8;
    e13:
       if ((er[i] = '+') or (er[i] = '.')) then goto e9
       else goto e11;
    e11:
       if (pa=0) THEN
       begin
         writeln('******** La expresión fue aceptada! ********');
         goto fin;
       end
       else
       begin
         writeln('Error ..!, los parentesis estn desbalanceados');
         goto inicio;
         end;
    fin:
       writeln('Para continuar teclear  <RETURN>');
       read(ww);
end;

{/*************************************************************************/}
```

```
PROCEDURE portada(var ww:char);
VAR
    x      : integer;
  BEGIN
    WRITELN('  Inicio de el programa  * AUTOMATA *');
    FOR x: = 1 TO 80 DO WRITE(' = ');
     WRITE(' Modulo :  LECTURA DE LA EXPRESION ');
   for i: = 1 to 10 do writeln;
    repeat
     evaluacion;
    until acexpre = false;
END;

{*************************************************************************}

PROCEDURE conkaten(var priuno,ultimuno,pridos,ultdos: byte);

BEGIN
    transpri[ultimuno,epsilon,apuno]: = pridos;
    ultimuno: = ultdos;
END;

{*************************************************************************}

PROCEDURE lacerradu(var priuno,ultimuno: byte);

BEGIN
    transpri[edo + 1,epsilon,apuno]: = priuno;
    transpri[edo + 1,epsilon,dosap]: = edo + 2;
    transpri[edo,epsilon,apuno]: = priuno;
    transpri[edo,epsilon,dosap]: = edo + 2;
    priuno: = edo + 1;
    ultimuno: = edo + 2;
    edo: = edo + 2;
END;
```

```
PROCEDURE falla;

BEGIN
    WRITELN;
    WRITELN('caracter no permitido ',chr(7));
    halt
END;


{*************************************************************************}

PROCEDURE lectdecara;

BEGIN
    read(er,cara);
    cara: = upcase(cara);
  END;

{*************************************************************************}

PROCEDURE lectdekaden(var priuno,ultimuno : byte);

BEGIN
    edo : = edo + 1;
    priuno : = edo;
    while cara IN alfabet do
    BEGIN
      usados: = usados + [cara];
      transpri[edo,cara,apuno]: = edo + 1;
      edo: = edo + 1;
      lectdecara;
    END;
    ultimuno: = edo;
END;
FUNCTION cardinal(a:conjuntos): integer;
```

```
var
    i,n:integer;
BEGIN
    n:=0;
    for i:=1 to totone do IF i IN a then n:=succ(n);
     cardinal:=n;
END;


{*************************************************************************}


PROCEDURE impriconj(a:conjuntos);
var
     sep:char;
     i:integer;
BEGIN
     IF cardinal(A)=0 THEN WRITELN('()')
     ELSE
        BEGIN
           sep:='(';
           for i:=1 to totone do
            IF i IN a THEN
              BEGIN
                 WRITE(sep,i);
                 sep:=',';
              END;
           WRITELN(')');
        END;
END;


{*************************************************************************}
```

```
FUNCTION nombconju(connum: byte):char;

BEGIN
    nombconju: = chr(connum + 64);
END;

{*************************************************************************}


PROCEDURE conbajosim (a:conjuntos; bajo:char; varx:conjuntos;
incluye:boolean);

var
   informest : posiciones;

PROCEDURE encuentra(nodo:byte;bajo:char;var x:conjuntos;incluye:boolean);
BEGIN
     IF x  x + [nodo] THEN
     BEGIN
        IF incluye THEN x: = x + [nodo];
        IF transpri[nodo,bajo,apuno]  0 THEN
        encuentra(transpri[nodo,bajo,apuno],bajo,x,true);
        IF transpri[nodo,bajo,dosap]  0 THEN
        encuentra(transpri[nodo,bajo,dosap],bajo,x,true);
     END;
END;

BEGIN
   x: = [];
   for informest: = 0 to totone do
      IF informest IN a THEN encuentra(informest,bajo,x,incluye);
END;

{*************************************************************************}
```

```
PROCEDURE encuencaden(var x:conjuntos);
var
    i:posiciones;
    nuevo:boolean;
BEGIN
    indice:=0;
    nuevo:=true;
    for i:=1 to carfinal do IF x = conexpr[i]
    THEN
      BEGIN
        nuevo:=false;
        indice:=i;
      END;
    IF nuevo and (x[]) THEN
     BEGIN
        carfinal:= succ(carfinal);
        conexpr[carfinal]:=x;
        indice:=carfinal;
     END;
END;


{*************************************************************************}


PROCEDURE leoexp(var priuno, ultimuno:byte);


PROCEDURE leotermi(var priuno,ultimuno: byte);

var
    pridos,ultdos : byte;
    mulop : char;
```

```
  PROCEDURE loefak(var priuno,ultimuno : byte);
BEGIN
     IF cara IN alfabet + ['('] THEN
     case cara of
     epsilon..uch:lectdekaden(priuno,ultimuno);

     '(':BEGIN
         lectdecara;
         leoexp(priuno,ultimuno);
         IF cara = ')' THEN lectdecara ELSE falla;
        END
     END
     ELSE falla
  END;
  BEGIN
    loefak(priuno,ultimuno);
    while cara IN ['*','.','('] + alfabet do
    BEGIN
       mulop: = cara;
       case mulop of
       '*':BEGIN lacerradu(priuno,ultimuno);lectdecara END;
       '.',epsilon..uch: BEGIN
                         IF cara = '.' THEN lectdecara;
                         leotermi(pridos,ultdos);
                         conkaten(priuno,ultimuno,pridos,ultdos);
                        END;
       '(':BEGIN
           loefak(pridos,ultdos);
           conkaten(priuno,ultimuno,pridos,ultdos);
          END
       END;
    END
  END;
  BEGIN
   leotermi(priuno,ultimuno);
```

```
    while cara = '+' do
    BEGIN
       lectdecara;
       leotermi(pridos,ultdos);
       junta(priuno,ultimuno,pridos,ultdos);
    END;
END;


{**************************************************************************}

{  Nota: Los bloques que aparecen entre "= = = = = = = = = = = = = =" fueron
agregados con el fin de minimizar el A.F.D.}


PROCEDURE Cambia(viejo,nuevo : byte);

var i : byte; c : char;

BEGIN
    for i: = 1 to mayedo do
      IF not (i IN eliminados) THEN
        for c: = '0' to uch do
          IF c IN usados THEN
           IF transeg[i,c] = viejo THEN transeg[i,c]: = nuevo
END;


{**************************************************************************}

Procedure Elimina_Estados;

var
    i,j    :byte;
    iguales  :boolean;
```

```
BEGIN

   {Eliminacion de estados}
   i:=1 ;j:=0;
   While(i <=mayedo) and (j <mayedo ) do
   BEGIN
    IF i IN eliminados THEN
      BEGIN
        j:=i+1;
        while j IN eliminados do j:=j+1;
        IF j IN finales THEN finales:=finales+[i]-[j];
        Cambia(j,i);
        eliminados:=eliminados+[j]-[i];
        transeg[i]:=transeg[j];
        llave[i] := llave[j];
      END;
     i:=i+1
   END;
   mayedo:=mayedo-nel;{Numero actualizado de estados}
END;



{*************************************************************************}


Procedure Genera_Codigo;
var
    i   :byte;
    cad  :string[9];
    ar   :text;

Procedure Escribe_Rutina(nest  :byte);
var i  :byte;
```

```
 BEGIN
     WRITELN(ar,' Procedure E',nest,';');
     WRITELN(ar,'  BEGIN');
     WRITELN(ar,'    salida: = ');
     IF nest IN finales
       THEN WRITELN(ar,'true;')
       ELSE WRITELN(ar,'false;');
     WRITELN(ar,'    Getchar (c);');
     WRITELN(ar,'    case c of');
     for car: = '0' to uch do
      IF car IN usados THEN
        IF transeg[nest,car]0 THEN
          WRITELN(ar,'    ',"'",car,"'",':E',transeg[nest,car],';');
     WRITELN(ar,'      #13  :{Fin de cadena};');
     WRITELN(ar,'      ELSE falla: = true');
     WRITELN(ar,'    END');
     WRITELN(ar,'  END;');
END;

BEGIN  { genera codigo}
     WRITE('de el nombre del archivo de salida (.pas por default ) :');
     READ(cad);
     assign(ar,cad + '.pas');
     REWRITE(ar);
     WRITELN(ar,'program auto; ');
     WRITELN(ar,' var salida,falla : boolean; c:char;');
     WRITELN(ar,' procedure getchar(var c: char );');
     WRITELN(ar,' BEGIN ');
     WRITELN(ar,'  READ(kbd,c);');
     WRITELN(ar,'  c: = upcase(c);');
     WRITELN(ar,'  IF not(c IN [#13,#27] ) THEN WRITE(c)');
     WRITELN(ar,' END;');
     for i: = 1 to mayedo-1 do
      WRITELN(ar,' procedure E',i,'; forward;');
     for i: = mayedo downto 1 do
```

```
      escribe_rutina(i);
    WRITELN(ar,'  BEGIN');
    WRITELN(ar,'    repeat');
    WRITELN(ar,'      falla:=false;');
    WRITELN(ar,'      e1;');
    WRITELN(ar,'      WRITELN;');
          WRITELN(ar,'          IF salida and (not falla) THEN WRITELN('
,'''','ok.','''',')');
    WRITELN(ar,'              ELSE IF c#27 THEN ',
         ' WRITELN(','''','cadena no reconocida.','''',');');
    WRITELN(ar,'      WRITELN');
    WRITELN(ar,'    until c=#27');
    WRITELN(ar,'  End.');
    close(ar);
END;


{*************Inicia programa principal*************}


BEGIN
    portada(ww);
    xx:=0;
    alfabet:=[epsilon..uch];
    usados:=[epsilon];
    for edo:=0 to totone do
    BEGIN
      conexpr[edo]:=[];
      for cara:=epsilon to uch do
      BEGIN
        transpri[edo,cara,apuno]:=0;
        transpri[edo,cara,dosap]:=0;
        transeg[edo,cara]:=0;
      END;
    END;
```

```
priuno:=1;
edo:=0;
lectdecara;
while cara  cr do leoexp(priuno,ultimuno);clrscr;
for cont:=1 to 5 do WRITELN;
WRITELN(
'              TABLA DE LA FUNCION DE TRANSICION DEL');WRITELN(
'               AUTOMATA FINITO NO DETERMINISTICO');
WRITELN;WRITELN;WRITELN;
WRITE(' ':7,'  estados    ',CHR(238),'     G');
for cara:='0' to uch do IF cara IN usados THEN WRITE('   ',cara,'   G');
WRITELN;
WRITE(' ':7,'  +-------------------+-');
for cara:='0' to uch do IF cara IN usados THEN WRITE('-------+-');WRITELN;
for informest:=1 to edo do
BEGIN
 WINDOW(1,13,80,22);
 xx:=xx+1;
 IF xx=8 THEN
  BEGIN
   xx:=0;WRITELN('       Pulse <CR>.....');
   REPEAT UNTIL KEYPRESSED;
   CLRSCR;
  END;
 WRITE('         ',informest:3,'     ');
 for carcont:=epsilon to uch do IF carcont IN usados THEN
  IF transpri[informest,carcont,apuno]=0 THEN
   WRITE(' ':10)
  ELSE
   IF transpri[informest,carcont,dosap]=0 THEN
    WRITE(' (',transpri[informest,carcont,apuno]:2,')')
   ELSE WRITE(' (',transpri[informest,carcont,apuno]:2,
        ',',transpri[informest,carcont,dosap]:2,') ');
 WRITELN;
END;WINDOW(1,1,80,24);
```

```
  gotoxy(19,24);
  WRITE('Para continuar pulse < RETURN >  ');READ(ww);clrscr;
  GOTOXY(1,6);
   WRITELN('                    TABLA DE LA FUNCION DE TRANSICION' );
WRITELN;
  WRITELN('                   DEL AUTOMATA FINITO ');WRITELN;
  WRITELN('                    DETERMINISTICO');WRITELN;
  WRITELN;WRITELN;
  caractua: = 1;
  carfinal: = 1;
  conbajosim([priuno],epsilon,conexpr[caractua],true);
  WRITE(' ':15,'      Est |');
  for cara: = '0' to uch do
   IF cara IN usados THEN WRITE(cara:2,' |');WRITELN;
   WRITE(' ':15,'      +-----+');
   for cara: = '0' to uch do
   IF cara IN usados THEN WRITE('---+');WRITELN;
  i: = 1;
  while conexpr[caractua]  [] do
  BEGIN
   WRITE('                ',nombconju(caractua):2,'  |');
    for cara: = '0' to uch do IF cara IN usados THEN
    BEGIN
      conbajosim(conexpr[caractua],cara,x1,false);
      conbajosim(       x1,epsilon,x2,true);
      encuencaden(x2);
      transeg[caractua,cara]: = indice;
      WRITE(nombconju(indice):2,' |,');
    END;
   WRITE('  ');impriconj(conexpr[caractua]);
   caractua: = succ(caractua);
  END;
  mayedo: = caractua-1;
```

```
{********************** generacion de llaves **************************}

  WRITELN;WRITELN;WRITELN;
  WRITE('Pulse < CR > para continuar');
  read(kbd,car);
  for i:= 1 to mayedo do
  BEGIN
    llave[i]:=0;x:=1;
    for car := '0' to uch do
      IF car IN usados THEN
      BEGIN
        llave[i]:=llave[i]+sqrt(transeg[i,car]*x);
        x:=x*10
      END
  END;


{********************* Deteccion de estados finales **********************}

  finales:=[];
  for i:=1 to mayedo do
    IF ultimuno IN conexpr[i] THEN finales:=finales+[i];


{*************** Deteccion de Estados no Finales Redundantes ***********}


  eliminados:=[];


  nel:=0;
```

```
{********** Numero de Estados Eliminados **********}

  for i:=1 to mayedo-1 do
  BEGIN
    noedo:=i+1;
    while noedo <=mayedo do
    BEGIN
      IF (llave[i]=llave[noedo]) and (not(noedo IN eliminados)) and
        (not(i IN eliminados)) and (not (i IN finales)) and
        (not(noedo IN finales))  THEN
       BEGIN
        eliminados:=eliminados+[noedo]; nel:=nel+1;
        Cambia(noedo,i)
       END;
      noedo:=noedo+1
    END
  END;

{********** Deteccion de Estados Finales redundantes **********}

  for i:=1 to mayedo-1 do
  BEGIN
   noedo:=i+1;
   while noedo <=mayedo do
     BEGIN
      IF (llave[i]=llave[noedo]) and (not(noedo IN eliminados)) and
        (not (i IN eliminados)) and (i IN finales)
        and (noedo IN finales)  THEN
       BEGIN
        eliminados:=eliminados+[noedo]; nel:=nel+1;
        finales:=finales-[noedo];
        Cambia(noedo,i)
       END;
```

```
      noedo: = noedo + 1
    END
  END;

  Elimina_Estados;


{********************** Escritura del AFD minimizado ******************}

  clrscr;
  WRITELN('              Automata Finito Deterministico Minimo');
  gotoxy(25,4);
  WRITE('3');
  for car: = '0' to uch do
   IF car IN usados THEN WRITE('    ',car,'   ');
     gotoxy(20,5);WRITE('_____|');
     for car: = '0' to uch do IF car IN usados THEN WRITE('__________');
     gotoxy(25,6);WRITE('|');
     WRITELN;
  for i: = 1 to mayedo do
  BEGIN
   gotoxy(20,wherey);WRITE(i:2,'   |');
   for car: = '0' to uch do
     IF car IN usados THEN WRITE('   ',transeg[i,car]:2,'   ');
   IF i IN finales THEN BEGIN
                  TEXTCOLOR(3 + BLINK);WRITE('   final');
                  NORMVIDEO;
                  END;
   WRITELN
  END;

  WRITELN;WRITELN;WRITELN;
  WRITE('Pulse  < CR > para continuar');
  read(kbd,car);
  clrscr;
```

```
  Genera_Codigo

{==================================}

END.{*********************** Fin de programa principal ****************}
```

```
program auto;
  var salida,falla : boolean; c:char;
  procedure getchar(var c: char );
  BEGIN
    READ(kbd,c);
    c:=upcase(c);
    IF not(c IN [#13,#27] ) THEN WRITE(c)
  END;
  procedure E1; forward;
 Procedure E2;
   BEGIN
     salida:=
true;
      Getchar (c);
      case c of
      '0':E2;
      '1':E2;
        #13  :{Fin de cadena};
        ELSE falla:=true
       END
     END;
 Procedure E1;
   BEGIN
     salida:=
true;
      Getchar (c);
      case c of
      '1':E2;
        #13  :{Fin de cadena};
        ELSE falla:=true
       END
     END;
  BEGIN
    repeat
     falla:=false;
```

```
    e1;
    WRITELN;
    IF salida and (not falla) THEN WRITELN('ok.')
          ELSE IF c#27 THEN   WRITELN('cadena no reconocida.');
    WRITELN
  until c = #27
End.
```

APENDICE B

PROGRAMA DE AUTOMATA ESTOCASTICO DE ESTADOS FINITOS.

```
program reconoce_electro;
Uses
    crt, graph;

const
    maxi_esta = 3;
    maxlonsal = 11;
    nomaxtab  =6;

  type
    matriz = array [1..maxi_esta,1..maxi_esta] of real;
    descriaut = record
      estado : 'A'..'N';
      emit  : '0'..'1';
      esta_sig : 'A'..'C';
      prob  : real;
    end;
    transicio  =  string [maxlonsal + 1];

  { DECLARACION DE VARIABLES GLOBALES }
var
   matrizesto,b,c : matriz;   { matriz que contiene los estados }
   automata : array [1..nomaxtab] of descriaut; { almacen del automata }
```

```
    secesta,s : transicio;
    probabi  : real;
    cuantas,i,j,k1,k2,noc : integer; {variables indices}


  {RUTINA QUE SE ENCARGA DE DIBUJAR UN MARCO EN PANTALLAS }
procedure ventana;
begin
    k1:=detect;
    Initgraph(k1,k2,'');
    SetBkColor(14);
    line(0,0,0,199);
    line(0,199,639,199);
    line(639,199,639,0);
    line(639,0,0,0);
    Outtextxy(200,20,'RECONOCEDOR DE ELECTROCARDIOGRAMAS');
end;


  {RUTINA QUE DIBUJA MARCO EN LA PANTALLA}
procedure venta;
begin
    k1:=detect;
    initgraph(k1,k2,'');
```

```
    SetBkColor(7);
    Outtextxy(100,15,'CUANTAS GRAFICAS DESEAS ....');
    gotoxy(5,5); read(noc);
end;

  {INICIACION DE VARIABLES, TALBAS DE TRANSICIONES Y LA MATRIZ DE
PROBABILIDAD}
procedure inicial;
var
   i,j : integer;
begin
    for i: = 1 to maxi_esta do
    begin
      for j: = 1 to maxi_esta do
        b[i,j]: = 0;
      b[i,i]: = 1;
    end;
    with automata[1] do
    begin
      estado: = 'A'; emit: = '0'; esta_sig: = 'B'; prob: = 0.5;
    end;
    with automata[2] do
    begin
      estado: = 'A'; emit: = '0'; esta_sig: = 'C'; prob: = 0.25;
```

```
end;
with automata[3] do
begin
   estado: = 'A'; emit: = '1'; esta_sig: = 'C'; prob: = 0.25;
end;
with automata[4] do
begin
   estado: = 'B'; emit: = '1'; esta_sig: = 'A'; prob: = 0.6;
end;
with automata[5] do
begin
   estado: = 'B'; emit: = '1'; esta_sig: = 'C'; prob: = 0.4;
end;
with automata[6] do
begin
   estado: = 'C'; emit: = '0'; esta_sig: = 'A'; prob: = 1.0;
end;
matrizesto[1,1]: = 0.0; matrizesto[1,2]: = 0.5;
matrizesto[1,3]: = 0.5; matrizesto[2,1]: = 0.6;
matrizesto[2,2]: = 0.0; matrizesto[2,3]: = 0.4;
matrizesto[3,1]: = 1.0; matrizesto[3,2]: = 0.0;
matrizesto[3,3]: = 0.0;
SetViewPort(5,30,635,195,ClipOn);   End;
```

```
  {Rutina que intercambia las matrices de los estados }
procedure intermatriz(a: matriz; var b: matriz);
var
    i,j : integer;
begin
     for i: = 1 to maxi_esta do
       for j: = 1 to maxi_esta do
         b[i,j]: = a[i,j];
end;


  {Multiplicacion de las matrices de estados }
procedure multiplica_matriz(a,b:matriz; var c:matriz);
var
    i,j,k : integer;
    s    : real;
begin
     for i: = 1 to maxi_esta do
     begin
       for j: = 1 to maxi_esta do
       begin
         s: = 0.0;
         for k: = 1 to maxi_esta do
           s: = s + a[i,k] * b[k,j];
```

```
            c[i,j]: = s;
          end;
        end;
end;

  {Rutina que busca el nodo de mayor probabilidad }
procedure busnodo(actual_esta,actual_sali :char; var l,r : descriaut);
var
    j: integer;

begin
    j: = 0; l.estado: = 'N'; l.prob: = -1;r.estado: = 'N'; r.prob: = -1;
    Repeat inc(j)
    until ((automata[j].estado = actual_esta) and
        (automata[j].emit = actual_sali));
    if j < = nomaxtab then
    begin
      l.estado: = automata[j].esta_sig;
      l.prob: = automata[j].prob;
    end;

    Repeat inc(j)
    until ((automata[j].estado = actual_esta) and
```

```
            (automata[j].emit = actual_sali));
        if j < = nomaxtab then
        begin
          r.estado: = automata[j].esta_sig;
          r.prob: = automata[j].prob;
        end;
end;

   {Funcion de busqueda de estados}
Function buscar(actual_esta:char; level : integer; curp:real; var p:real) : char;

var
    l,r : descriaut;
    resp : char;
    pl,pr : real;
begin
     if (level < = maxlonsal) and (actual_esta < > 'N') then
     begin
       busnodo(actual_esta,s[level],l,r);
       resp : = buscar(l.estado,level + 1,l.prob,p);
       if resp < > 'N' then
       begin
          secesta[level] : = actual_esta;
```

```
            p:= p*l.prob;
        end
        else    begin
            resp := buscar(r.estado,level + 1,r.prob,p);
            if resp < > 'N' then
            begin
                secesta[level] := actual_esta;
                p:=p*r.prob;
            end;
        end;
        buscar:= resp;
        end
        else
          if level  > maxlonsal then
          begin
              secesta[level] := actual_esta;
              buscar := actual_esta;
          end
          else
              buscar := 'N';
end;
   {Genera la cadena correspondiente al ECG }
procedure genera_cadena(var s: transicio);
```

```
var
    act1,i,j,k,l,sigl : integer;
    actual,sig : char;
Begin
     act1: = 1;
     for i: = 1 to maxlonsal do
     begin
       multiplica_matriz(matrizesto,b,c);
       intermatriz(c,b);
       k: = 1;
       while k = act1 do inc(k);
       sigl: = k;
       for j: = k + 1 to maxi_esta do
         if (b[act1,sigl]  b[act1,j]) and (act1j) then sigl: = j;
       case act1 of
          1 : actual : = 'A';
          2 : actual : = 'B';
          3 : actual : = 'C';
       end;
       case sigl of
          1 : sig : = 'A';
          2 : sig : = 'B';
          3 : sig : = 'C';
```

```
        end;
        if (actual = 'A') and (sig = 'C') then
        begin
          j: = random(2);
          if j = 1 then s[i] : = '1'
              else s[i] : = '0';
        end
        else
        begin
          j: = 0;
          repeat
            inc(j);
          until ((automata[j].estado = actual) and
              (automata[j].esta_sig = sig));
          s[i]: = automata[j].emit;
        end;
        act1: = sig1;
      end;
      s[0]: = char(maxlonsal);
end;

    {Agrega ruido a la cadena generada por AEF que pasara a la etapa
de reconocerlo}

procedure agrega_ruido(var s:transicio);
```

```
var
    i,j : integer;
    band : boolean;
begin
      band: = false;
      randomize;
      for i: = 1 to 4 do
      begin
        j: = random(maxlonsal) + 1;
        if s[j] = '0' then
        begin
          s[j]: = '1';
          band: = true;
        end;
        if not band then s[maxlonsal]: = '1';
      end;
end;

   {proceso que analiza y da diagn"stico de la cadena}
Procedure analiza;
var
    f0,f1:array [1..8] of integer;
    i,j,k,l  :integer;
```

```
    no,non,cierto,cierto1,cierto2 : boolean;
begin
   j:=0;
   k:=1;
   l:=1;
   no:=true;
   non:=true;
   for i:=1 to 8 do
   begin
     f0[i]:=0;
     f1[i]:=0;
   end;
   while k do
   begin
    while s[k]='0' do
    begin
     inc(j);
     inc(k);
    end;
    f0[l]:=j;
    j:=0;
    while s[k]='1' do
    begin
```

```
    inc(j);
    inc(k);
  end;
 k:=k+j;
 f1[l]:=j;
 l:=l+1;
end;
cierto:=false;
cierto1:=false;
cierto2:=false;

if f0[3]=4 then
  outtextxy(410,110,'FOCO ECTOPICO ACTIVADO');

if (f0[1]<f0[2]) and (f0[2]<f0[3]) and (f0[3]<f0[4]) then
begin
  outtextxy(410,110,'ARRITMIA SINUSAL');

  cierto:=true;
end;

if ((f1[1]=2) or (f1[2]=2) or (f1[3]=2) or (f1[4]=2)) then
begin
```

```
    cierto1: = true;
    outtextxy(410,110,'Ritmo Rapido');
  end;

  if ((f0[1] > 3) or (f0[2] > 3) or (f0[3] > 3) or (f0[4] > 3))
                then
  begin
    cierto2: = true;
    outtextxy(70,110,'Falta Segmento QRS, Bloqueo del nodo SA');
  end;

  if (not(cierto1)) and (not(cierto)) and (not(cierto2)) then
    outtextxy(410,110,'ECG NORMAL');
end;

 {Reconocimiento del electrocardiograma }
procedure reco_ecg;
var
  i : integer;
  r : char;
  p : string[11];
  probabilidad : real;
```

```
begin
    for i: = 1 to maxlonsal do
      secesta[i]: = ' ';
    probabi : = 1;
    probabilidad : = 1;
    secesta[1]: = 'A';
    r: = buscar('A',i,1,probabilidad);
    if r = 'N' then probabi: = -1
        else probabi: = probabi * probabilidad;
    secesta[0]: = char(maxlonsal);
    Outtextxy(150,120,'Secuencia de estados del AEF = ');
    Outtextxy(400,120,secesta);
    str(probabi:2:7,p);
    Outtextxy(276,130,'Probabilidad = ' + p);
end;

  {Rutina graficadora del electrocardiograma}
procedure dibuja_ecg(n:integer; s:transicio);
const
    dx = 10;
    dy = 5;
var
    x,y,x1,y1,i : integer;
```

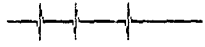
```
begin
    x:=80;
    case n of
      1 : begin
            y:=30; y1:=30;
            Outtextxy(x,40,'ECG con ritmo normal');
          end;
      2 : begin
            y:=80; y1:=80;
            Outtextxy(x,90,'ECG ruidoso');
          end;
    end;
    for i:= 1 to maxlonsal do
    begin
      line(x,y,x+dx-2,y);
      if s[i]='1' then
      begin
        line(x+dx-2,y,x+dx,y+dy);
        sound(800); delay(200); Nosound;
        line(x+dx,y-dy,x+dx,y+dy);
        line(x+dx,y-dy,x+dx+2,y);
      end
      else
```

```
            line(x + dx-2,y,x + dx + 2,y);
        x: = x + dx + 2;
    end;
    x1: = x + dx;
    Outtextxy(x1,y1,s);
end;

  {Programa principal}
begin
    cuantas: = 0; noc: = 0;
    venta;
    Repeat
        randomize;
        ClearViewPort;
        ventana;
        inicial;
        genera_cadena(s);
        dibuja_ecg(1,s);
        analiza;
        agrega_ruido(s);
        dibuja_ecg(2,s);
        reco_ecg;
        cuantas: = cuantas + 1;
```

```
        readln;
    until (cuantas = noc);
end.
```

RECONOCEDOR DE ELECTROCARDIOGRAMAS



01010010000

ECG con ritmo normal

01110111000

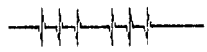
ECG ruidoso

Falta Segmento QRS, Bloqueo del nodo SA ARRITMIA SINUSAL

Probabilidad = 0.5000000

RECONOCEDOR DE ELECTROCARDIOGRAMAS

ECG con ritmo normal

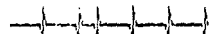


ECG ruidoso

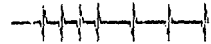
Falta Segmento QRS, Bloqueo del nodo SA ARRITMIA SINUSAL

Probabilidad = 0.5000000

RECONOCEDOR DE ELECTROCARDIOGRAMAS



ECG con ritmo normal



ECG ruidoso

Ritmo Rapido

Probabilidad = 0.2500000

APENDICE C

PROGRAMA, GENERADO POR EL PROGRAMA AUTOMATA, QUE RECONOCE LA CADENA A PARTIR DE LA EXPRESION REGULAR DEL CAPITULO CUATRO.

```
program auto;
    var salida,falla : boolean; c:char;
   procedure getchar(var c: char );
    BEGIN
     READ(kbd,c);
     c: = upcase(c);
     IF not(c IN [#13,#27] ) THEN WRITE(c)
    END;
   procedure E1; forward;
   procedure E2; forward;
   procedure E3; forward;
   procedure E4; forward;
   procedure E5; forward;
   Procedure E6;
    BEGIN
      salida: =
      false;
      Getchar (c);
      case c of
       '0':E6;
       'P':E4;
          #13  :{Fin de cadena};
     ELSE falla: = true
     END
   END;
   Procedure E5;
   BEGIN
     salida: =
     false;
     Getchar (c);
     case c of
      '0':E5;
      'N':E3;
         #13  :{Fin de cadena};
         ELSE falla: = true
```

```
  END
 END;
 Procedure E4;
 BEGIN
   salida: =
   false;
   Getchar (c);
   case c of
    '0':E6;
    'P':E1;
       #13  :{Fin de cadena};
   ELSE falla: = true
   END
 END;
 Procedure E3;
 BEGIN
   salida: =
   false;
   Getchar (c);
   case c of
     '0':E5;
     'N':E1;
        #13  :{Fin de cadena};
   ELSE falla: = true
 END;   END;
Procedure E2;
BEGIN
  salida: =
  false;
  Getchar (c);
  case c of
   '0':E2;
   'N':E3;
   'P':E4;
      #13  :{Fin de cadena};
```

```
      ELSE falla: = true
    END
  END;
  Procedure E1;
  BEGIN
    salida: =
    true;
    Getchar (c);
    case c of
     '0':E2;
     'N':E1;
     'P':E1;
       #13  :{Fin de cadena};
     ELSE falla: = true
    END
  END;
BEGIN
 repeat
  falla: = false;
  e1;
  WRITELN;
  IF salida and (not falla) THEN WRITELN('ok.')
        ELSE IF c#27 THEN   WRITELN('cadena no reconocida.');
  WRITELN
 until c = #27
End.
```