UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
ACATLAN

"DESARROLLO DE UN SISTEMA EXPERTO EN AMBIENTE GRAFICO COMO FRONT-END A UNA BASE DE CONOCIMIENTOS INDUCIDA"

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
CLAUDIA ESPINO BECERRIL

ASESOR: LIC. JUAN CARLOS RENDON AGUILAR

ACATLAN, EDO. DE MEXICO 1997

Agradecimientos

Quisiera a través de este medio, agradecer al Instituto Mexicano del Petróleo por haberme brindado todas las facilidades para la realización de este trabajo de tesis. En forma muy especial, al M. en C. Víctor García Portilla quien me guió y compartió conmigo mucho de su saber.

A la Vida
por haberme permitido
llegar a este momento.

A mis Padres
que, gracias a sus
sabios consejos y confianza
me han impulsado a
seguir adelante

A Oscar
por el apoyo y cariño
que siempre me acompañan.

A mis amigos y personas
que de alguna manera contribuyeron
a la culminación de este trabajo.

Al Lic. Juan Carlos Rendón Aguilar
por su apoyo en el desarrollo
de esta tesis.

INDICE

ÍNDICE

	Pág.
INTRODUCCIÓN	1
CAPÍTULO I. DESARROLLO DE LA INTELIGENCIA ARTIFICIAL	5
I.1 HISTORIA DE LA INTELIGENCIA ARTIFICIAL	5
I.2 CONCEPTOS EN INTELIGENCIA ARTIFICIAL	8
Técnicas Heurísticas	8
Estrategias de Búsqueda	9
I.3 APLICACIONES DE LA INTELIGENCIA ARTIFICIAL	22
Sistemas Expertos	22
Robótica	22
Comprensión del Habla (voz)	24
Procesamiento del Lenguaje Natural	25
Redes Neuronales	25
I.4 HERRAMIENTAS METODOLÓGICAS DE LA INTELIGENCIA ARTIFICIAL	26
El lenguaje LISP	26
El lenguaje PROLOG	27
El lenguaje C	28
Comparación entre LISP y PROLOG contra C	29
CAPÍTULO II. INDUCCIÓN	31
II.1 INDUCCIÓN	31
II.2 TEORÍA DE LA RESONANCIA ADAPTATIVA	31
Redes Neuronales	31
Teoría de la Resonancia Adaptativa (ART)	61
II.3 ALGORITMO INDUCTION DECISION TREE (ID3)	70
Antecedentes: el Sistema de Aprendizaje Conceptual (CLS)	70
Induction Decision Tree (ID3)	71
CAPÍTULO III. SISTEMAS EXPERTOS Y REPRESENTACIÓN DE CONOCIMIENTO	84
III.1 SISTEMAS EXPERTOS	84
Concepto y Características de un Sistema Experto	84
Antecedentes Históricos	86

Arquitectura de un Sistema Experto	88
Estrategias de Inferencia	93
Metodología en el desarrollo de un Sistema experto	100
Aplicaciones de los Sistemas Expertos	101
III.2 REPRESENTACIÓN DE CONOCIMIENTO	107
Redes Asociativas	107
Frames (Marcos)	117
Reglas de Producción	120
Scripts	121
III.3 SHELLS Y ENTORNOS	121
CAPÍTULO IV: EL LENGUAJE DE PROGRAMACIÓN DELPHI	126
IV.1 INTRODUCCIÓN	126
Características de Delphi	126
IV.2 LA NUEVA PROGRAMACIÓN EN PASCAL	127
Programación Dirigida por Eventos	127
Programación Basada en Objetos	128
Objetos	129
Propiedades	130
Eventos	130
IV.3 EL AMBIENTE GRÁFICO DE DELPHI	132
IV.4 ESTRUCTURA DE UNA APLICACIÓN EN DELPHI	135
Unidades	135
Formas	138
IV.5 ALGUNOS COMPONENTES	142
1) Componente Label	142
2) Componentes Edit, MaskEdit y Memo	144
3) Componentes Button y BitBtn	146
4) Componentes RadioButton y CheckBox	146
5) Componente ListBox	146
6) Componente ComboBox	150
7) Componente Menu	150
8) Componentes para Cajas de Diálogo	150
IV.6 LAS BASES DE DATOS EN DELPHI	153
Modelos de Base de Datos	153
Creando y Accesando Tablas	154
Algunos Componentes	158
1) Componente Table	158
2) Componente DataSource	158

3) Componente DBGrid 158
4) Componente DBNavigator 158
Conectividad Abierta de Base de Datos (ODBC) 163

CAPÍTULO V. APLICACIÓN: UN SISTEMA EXPERTO DE PRODUCCIÓN PETROLERA 166
V.1 CONTEXTO DEL PROBLEMA 166
V.2 DESARROLLO 168
Estructuración de la Base de Conocimientos 168
1) Base de Datos con Registros Geofísicos Petroleros 168
2) Inducción 172
3) Base de Conocimientos 175
Módulo Generador de Reglas 183
1) Edición de Reglas 183
2) Modificación de Reglas 187
3) Adición de Reglas 187
Módulo para Consulta 193
Motor de Inferencia 198
V.3 APLICACIÓN A UNA BASE DE DATOS DE REGISTROS GEOFÍSICOS 210
1) Umbral Sugerido 210
2) Umbral manual 215

CONCLUSIONES 224

REFERENCIAS 228

INTRODUCCION

INTRODUCCIÓN

Desde mediados del siglo XX se ha venido dando a escala mundial, un interés cada vez mayor por los Sistemas Expertos, pudiendo decirse que no sólo están de moda, sino que constituyen uno de los temas de actualidad que más recursos humanos y materiales está consumiendo.

Antes de que los Sistemas Expertos fueran una realidad, éstos ya existían en la mente de muchos seres humanos, en efecto, las películas de ciencia ficción se encargaban de hacer ver que las máquinas podían sustituir al hombre en muchas funciones inteligentes, incluso hacerlo mejor que él. Entre ellas, la figura más representativa de este fenómeno es el "robot", a quien tradicionalmente se le ha dado una forma humana para resaltar aún más esta situación.

Entre los países que se han involucrado con estos conceptos se encuentra Estados Unidos, Francia, Alemania, Inglaterra y Japón, quienes crean día con día nuevas tecnologías en laboratorios de investigación o en empresas productoras de programas.

En el caso de nuestro país, la Inteligencia Artificial aún se encuentra en una etapa muy joven, prometiendo ser un espacio para el desarrollo tecnológico comercial y con fines de investigación. Cabe señalar que, existen empresas tanto de la iniciativa privada como del sector público (Instituto Mexicano del Petróleo), que se están interesando en la construcción de Sistemas Expertos para la resolución de problemas específicos. Por otro lado, la participación de Universidades e Institutos en esta área se está viendo incrementada, por ejemplo, el Instituto Tecnológico de Estudios Superiores de Monterrey, la Universidad Autónoma de México y el Instituto Politécnico Nacional son una muestra de ello. Asimismo, la incidencia de estos temas en congresos y simposiums va también en aumento.

Una de las áreas de mayor interés y demanda en Sistemas Expertos, ha sido sin duda alguna la Ingeniería, entre sus campos de acción, el petrolero parece ser bastante fructífero por su importante papel en la economía nacional.

Por ello en el presente trabajo se propone la construcción de un Sistema Experto para la Producción Petrolera, capaz de realizar diagnósticos con un alto grado de eficiencia bajo un entorno gráfico en el que el usuario pueda navegar amigablemente como lo hace en Windows.

De entre los muchos ejes temáticos en Inteligencia Artificial que podrían haberse incluido en este trabajo se han seleccionado 4, considerados como partes fundamentales para la comprensión de la aplicación.

El capítulo I titulado como "Desarrollo de la Inteligencia Artificial", pretende dar un contexto general de lo que es esta disciplina, subrayando algunos de los aspectos que hacen que una máquina o programa propiamente, sea considerado como poseedor de Inteligencia. Asimismo se exhiben sus campos más importantes que constituyen dimensiones muy específicas de aplicación, aunado a esto, también se hace referencia a algunas de sus Herramientas Metodológicas.

A partir de este marco teórico global los restantes capítulos se centrarán en aspectos más específicos del trabajo.

En el capítulo II intitulado como "Inducción", se echa una mirada al concepto de Inducción en Inteligencia Artificial, tomando como punto central la importancia de la información contenida en las Bases de Datos. Así se presentan 2 métodos que generan conocimiento, un modelo neuronal para datos cuantitativos y una técnica para información cualitativa.

El capítulo III denominado "Sistemas Expertos y Representación de Conocimiento", intenta brindar las bases conceptuales para la comprensión integral de los Sistemas Expertos. Asimismo, también se habla de algunas de las estructuras que existen para representar conocimiento. De igual manera se hace referencia al concepto de Shell y su implicación en la construcción de un Sistema Experto.

El capítulo IV titulado como "El lenguaje de programación Delphi", nos introduce a un nuevo tipo de programación que combina las estructuras comunes a los lenguajes de alto nivel como Pascal, con las características propias de la programación visual y orientada a objetos (propiedades, eventos y métodos), todo ello a través de una mirada al lenguaje Delphi.

El último capítulo llamado "Aplicación: un Sistema Experto para la Producción Petrolera" ya no es un apartado teórico, es un espacio en el que se pretende mostrar cómo se llevó a cabo la construcción y funcionamiento del Sistema Experto, el cual retoma conceptos de los capítulos anteriores. El desarrollo incluye además la constitución de la Base de Conocimientos, de manera que este aspecto también es punto de interés para abordarlo, centrando parte de su atención en la influencia que puede tener un modelo neuronal en la conformación de conocimiento y en las respuestas proporcionadas por el sistema.

Por otro lado, a través de la construcción del Sistema Experto se exhiben las características que puede proporcionar un lenguaje de programación como Delphi al aplicarse a un campo en específico.

Con lo anterior, se pretende dejar de manifiesto que los Sistemas Expertos pueden ser un gran aliado para llevar a cabo análisis cuando los expertos se encuentran ausentes y las necesidades, así como las responsabilidades requieren de tomar una decisión en la que no se puede perder tiempo ni dinero.

CAPITULO I

DESARROLLO DE LA INTELIGENCIA ARTIFICIAL

CAPÍTULO I
DESARROLLO DE LA INTELIGENCIA ARTIFICIAL

I.1 HISTORIA DE LA INTELIGENCIA ARTIFICIAL

La Inteligencia Artificial (IA) constituye un área de investigación que surge a partir de los años cincuenta de este siglo, sin embargo, sus ideas fundamentales tienen una larga historia compartida con la informática y la filosofía.

Contra lo que uno imaginase, las ideas de como construir una "máquina inteligente" son tan antiguas como variadas.

Se cree que los árabes medievales habían desarrollado un aparato para realizar cálculos teleológicos, el cual despertó el interés por desarrollar un artefacto que pudiera resolver cuestiones de teología, metafísica, y hasta de ciencia natural. Se dice que probablemente se trataba de una combinación del funcionamiento de una regla de cálculo y de los árboles de decisiones lógicas, construido a través de discos concentricos que giraban sobre un eje en común. Cuando se hacían coincidir ciertas marcas en un punto en particular, en otro punto del instrumento se combinaban determinadas informaciones que darían lugar a algo muy similar a un predicado referido a un tema en especifico

A partir del Renacimiento comienza a desarrollarse el actual concepto de ciencia y en tal desarrollo pueden identificarse dos aportaciones muy ligadas a la IA. Una de ellas es la construcción de autómatas, los cuales estaban basados en la tecnología de la construcción de relojes y que llegaron a imitar comportamientos motores al grado de que Le Metrie llegó a pensar que en el futuro podrían construirse seres humanos mecánicos, es decir, robots. La otra fue el gran desarrollo que tuvieron las matemáticas basadas en los números arábigos, ya que propiciaron el diseño de máquinas de cálculo como la de Pascal y la de Leibniz

A principios del siglo XIX, Charles Babbage diseña su máquina analítica, la cual es considerada como la primera calculadora artificial, con ella se demostró la posibilidad de poder construir un dispositivo artificial capaz de realizar cálculos complicados que en un momento dado podrían considerarse como inteligentes

En 1950, A. M. Turing, inventó su computadora de programas almacenados estos eran guardados como datos en memoria y posteriormente ejecutados creando de esta manera, la base para todas las computadoras modernas. Así, se dejó implicito que una

computadora podría ser capaz de aprender, razonar, hablar, explicar su razonamiento, y modificar de alguna manera su conocimiento propio.

Sin embargo, no es hasta 1956 cuando se construye el término de "Inteligencia Artificial", atribuyéndosele a John McCarthy, esto sucedió en una conferencia en la que se reunieron investigadores, en su mayoría americanos, con el objeto de discutir la posibilidad de que una computadora fuera capaz de poseer inteligencia.

En el periodo de 1956 a 1958, McCarthy crea el LISP (List Processing), un lenguaje de programación (el primero), útil en el manejo de procesos simbólicos para aplicaciones de Inteligencia Artificial.

Los años sesenta fueron un período de gran optimismo para la IA. En este período se desarrolló la primera computadora que jugaba ajedrez, las primeras pruebas matemáticas informatizadas y el famoso programa ELIZA que actuaba como un terapeuta rogeriano donde no existía la conversación psiquiatra-paciente. Para ejemplificar un poco como funcionaba dicho sistema, se muestra un fragmento de un diálogo con ELIZA[1]

Paciente: Los hombres son todos iguales
ELIZA: EN CIERTO MODO
Paciente: Siempre nos están fastidiando con algo
ELIZA: PUEDE PENSAR EN UN EJEMPLO ESPECÍFICO
Paciente: Bueno, mi novio me hizo venir aquí
ELIZA: SU NOVIO LA HIZO VENIR AQUÍ
Paciente: Dice que estoy deprimida la mayor parte del tiempo
ELIZA: SIENTO OÍR QUE ESTA DEPRIMIDA
Paciente: Es verdad. No soy feliz
ELIZA: CREE USTED QUE EL VENIR AQUÍ LA AYUDARÍA A SER FELIZ
Paciente: Necesito ayuda, eso es cierto
ELIZA: QUE SIGNIFICARÍA PARA USTED LOGRAR AYUDA
Paciente: Quizá pudiese aprender a entenderme con mi madre
ELIZA: HÁBLEME MAS SOBRE SU FAMILIA

Una pieza importante en este tiempo fue el Resolvedor de Problemas Generales (sus siglas en inglés: GPS), que fue desarrollado por Herbert Simon y Allen Newell. La experiencia con el GPS y otros proyectos similares proporcionó dos conclusiones: 1) El conocimiento es esencial para la conducta humana inteligente, por lo que la falta de éste en una forma contextual, limita la capacidad para resolver problemas y 2) Cuando el ser humano resuelve problemas heurísticos, necesita una guía para prevenir de alguna manera que se

[1] Rich, E. (1988) Inteligencia Artificial, págs. 76-77

genere un cúmulo de soluciones. Un ejemplo de esto lo constituye el problema del vendedor ambulante[2], donde dicho comerciante tiene que visitar una serie de ciudades, pero solamente una vez, siguiendo el camino más corto y regresando al lugar donde inició, tomando en cuenta que en todas las localidades existen carreteras directas que permiten la comunicación entre cada par de ciudades. Aquí, lo que pareceria más fructífero sería el explorar cada uno de los diferentes caminos, seleccionando aquel que proporcione la distancia más corta, sin embargo, para un número pequeño de ciudades funcionaría, pero conforme la cantidad de éstas aumenta, las posibles soluciones se incrementan considerablemente, por lo cual parece evidente la necesidad de contar con una herramienta que permita acotarlas de alguna manera.

El desarrollo de la IA continuó en los años setenta, periodo en el que Alain Colmerauer (en Marsella, 1972) crea otro lenguaje de programación, el PROLOG (Programming in Logic), que a diferencia del LISP, poseía una base de datos incorporada y una sintaxis bastante simple.

Hacia 1980, el LISP era el lenguaje preferido de la IA en Estados Unidos, mientras que PROLOG lo era para Europa. Sin embargo, en 1981 esta situación cambió tras el anuncio de los japoneses de que usarían PROLOG como base de sus computadoras de la Quinta Generación.

Poco a poco se fueron incorporando más términos a la IA, uno de ellos es el de "Sistemas Expertos", refiriéndose a programas que intentaban actuar como expertos humanos en ciertas tareas que involucran conocimiento.

En 1972, se desarrolló el MYCIN, un Sistema Experto usado para diagnosticar infecciones bacterianas en la sangre, siendo además considerado como el abuelo de los Sistemas Expertos. Otros Sistemas Expertos contemporáneos al MYCIN son el DENDRAL, el PROSPECTOR y el INTERNIS/CADUCEUS.

Hacia la mitad de los setentas, las computadoras que poseían grandes memorias eran muy comunes, asimismo, las velocidades de computación se habían incrementado, sin embargo, ante estas mejoras, existía una ineficacia inherente, por lo que se necesitaron crear otros algoritmos que hicieron que la IA requiriera de mejores rutinas para poder resolver los problemas que se le planteaban.

En 1977, el profesor Edward Feigenbaum creó el término de "Ingeniería del Conocimiento", que se refiere al diseño, desarrollo, examen e implementación de Sistemas Expertos, también es considerado como el padre de la comercialización de estos.

Se han realizado muchos esfuerzos mundiales en torno a la Inteligencia Artificial, entre ellos se puede mencionar el Proyecto de Computación Estratégica, en los Estados Unidos; el

[2] Para mayor explicación ver Rich, E. (1988) Op. cit., págs. 42-46

Proyecto de Computadoras de la Quinta Generación, en Japón; y el ESPRIT y el consorcio EUREKA en Europa.

De los ochenta a noventas, el interés en la IA continuó, habiendo actualmente cerca de 3000 Sistemas Expertos usados en el mundo, sin que por ello las investigaciones se detengan, por el contrario, prosiguen para tratar de cubrir las expectativas que cada día nacen.

I.2 CONCEPTOS EN INTELIGENCIA ARTIFICIAL

Técnicas Heurísticas

Una técnica heurística (del griego *heuriskein*) es un método que permite mejorar la eficiencia de un proceso de búsqueda, sacrificando posiblemente algunos aspectos, pero mejorando notablemente la calidad de los caminos que se exploran. De tal manera que, si se emplean buenas técnicas heurísticas, es de esperarse que se obtengan soluciones eficientes, incluso hasta óptimas, aún en problemas catalogados como difíciles, complejos o complicados.

Entre las técnicas heurísticas existen las denominadas de propósito general y las de propósito específico, que son útiles en una gran diversidad de problemas. Un ejemplo de las técnicas para el primer caso lo constituye el *algoritmo del vecino más próximo*, el cual trabaja seleccionando la alternativa localmente superior en cada caso. Aplicándola al problema del vendedor ambulante (ya mencionado anteriormente) se deduce el siguiente proceso[3] :

1) Elegir de manera arbitraria una ciudad de partida.

2) Para la elección de la subsecuente ciudad, observar que localidades aún no han sido seleccionadas y de esas, escoger la más cercana a la ciudad actual y dirigirse a ella en el siguiente paso.

3) Repetir el paso 2 hasta que se hayan visitado todas las ciudades.

En cuanto a las técnicas heurísticas de propósito específico, considérese la tarea de descubrir algunas ideas interesantes en una área en particular, para lo cual se emplea muy a menudo la siguiente técnica heurística:

"Si existe una función interesante de dos argumentos *f(x,y)*, mirar qué sucede si los dos argumentos son idénticos".[4]

[3] Ver Rich, E.(1988) Op. cit., págs. 42-46 para una explicación más amplia.
[4] Rich, E. (1988) Op. cit., pág. 44

Estrategias de Búsqueda

En IA, existen muchos problemas que exhiben cierta complejidad para ser resueltos a través de técnicas directas, por lo cual, resulta de mejor provecho el solucionarlos a través de métodos de búsqueda apropiados. De ahí la existencia de diversos procedimientos de búsqueda de propósito general, que pueden describirse independientemente de cualquier tarea y que constituyen una variedad de las técnicas de búsqueda heurística. Sin embargo, cuando se llegan a aplicar a problemas de carácter particular, su eficacia ya no va a ser la misma, pues ésta va a depender de la forma en que se explote el conocimiento, puesto que, por sí mismos son incapaces de superar la explosión combinatoria a la que se exponen estos procesos.

Cada proceso de búsqueda puede verse como un corte transversal de un grafo dirigido[5] , donde cada nodo podría significar un estado del problema, mientras que cada arco representaría una relación entre los estados que conecta. Así, el proceso de búsqueda tiene por objetivo encontrar un camino a través del grafo, comenzando con un estado inicial y concluyendo en uno o más de los estados finales.

Teóricamente, este grafo que requiere ser inspeccionado, debe estar construido por medio de reglas que determinan los movimientos permitidos en el espacio del problema, sin embargo, en la práctica, la mayoría de ellos nunca lo están, debido a que son demasiado voluminosos, asimismo, la mayor parte de su estructura no necesita ser explorada.

A continuación se presentan algunos tipos de búsqueda:

1) Generar y Examinar

Este método genera sistemáticamente todas las posibles soluciones de un problema, determinando cuando una solución se ha alcanzado. Consiste de los siguientes pasos:

1) Generar una posible solución. En algunos problemas esto significa que se va a generar un punto específico en el espacio del problema, mientras que para otros va a representar un camino desde el estado inicial.
2) Comprobar si ésta es una solución, comparando el punto en concreto o el final del camino elegido (según sea el caso) contra el conjunto de estados meta aceptables.
3) Si se concluye que se ha llegado a una solución, se finaliza. De lo contrario, regresar al paso 1) (Ver Figuras I.2.1 y I.2.2).

[5] Un grafo dirigido o digrafo G consiste de un conjunto de vértices (denominados también nodos o puntos) V y un conjunto de arcos E, donde cada arco se define como un par ordenado de vértices (v,w), en el cual v es llamado la "cola" y w la "cabeza" del arco. Así, frecuentemente este arco es representado de la siguiente manera: v→w. Aho, A., et. al. (1983) Data Structures and Algorithms. págs. 198-199.

Figura I.2.1 Método Generar y Comprobar, donde se puede observar que éste involucra un Generador y un Examinador

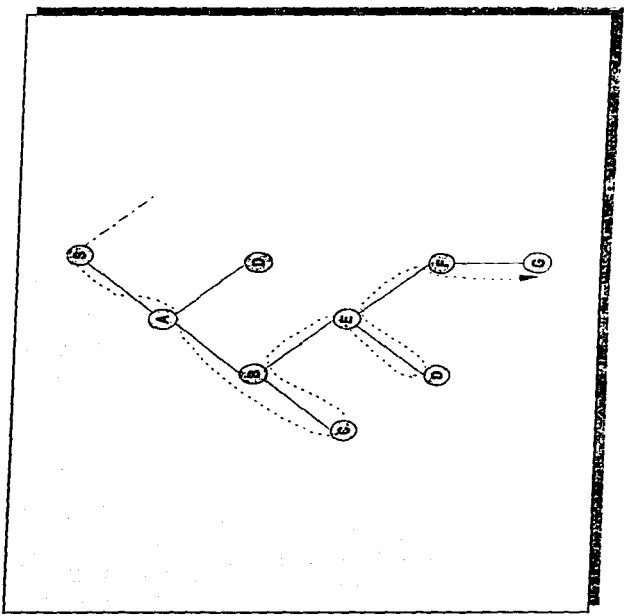

Figura 1.2.3 Figura que muestra el funcionamiento del Método Generar y Comprobar

Este algoritmo es un procedimiento de búsqueda que se considera de profundidad, debido a que deben generarse soluciones completas antes de su comprobación.

Para problemas sencillos éste método suele ser una técnica razonable, sin embargo, para problemas complejos, no es muy eficaz, pero cuando se combina para restringir el espacio de búsqueda, puede ser bastante efectivo, superando así, sus limitaciones. Un ejemplo de ello en IA lo conforma el programa DENDRAL, el cual tiene la capacidad para inferir la estructura de componentes orgánicos empleando datos de la espectografía de masas y de la resonancia nuclear magnética. Éste utiliza una estrategia denominada *planear-generar-comprobar*, en la cual un proceso de planificación que usa las técnicas de satisfacción de restricciones (se verá más adelante) crea listas de subestructuras recomendadas y contraindicadas, para posteriormente, emplear el método generar y comprobar que utiliza esas listas de tal manera que, sólo tiene que explorar un conjunto de estructuras ya más reducido.

2) Ascensión de Colinas

Este método constituye una variante de la técnica generar y comprobar, pero con la diferencia de que su función de comprobación no responde con un si o con un no, sino que se le agrega una función heurística que tiene como propósito proporcionar una estimación de que tan cerca está un estado de la meta para que, de esta manera, ayude al generador a decidir en que dirección debe moverse para llegar a la solución. Así, el procedimiento se explica a continuación:

1) Generar una primera solución (semejante al método generar y comprobar). Si ésta es una solución, entonces terminar, de lo contrario proseguir.

2) Ya que se tiene una solución, aplicar algunas reglas para generar un nuevo conjunto de soluciones propuestas.

3) Hacer lo siguiente para cada elemento del conjunto:

- **a)** Enviarlo a la función de comprobación. Si es la solución terminar.
- **b)** Si no lo es, ver si está más cerca de la solución que cualquiera de los elementos ya comprobados. En caso de serlo mantenerla, de lo contrario descartarla.

4) Tomar el mejor elemento encontrado anteriormente y usarlo como la siguiente solución propuesta.

5) Regresar al paso 2) (Ver Figura I.2.3).

Existe una situación importante que se puede presentar en el método de ascensión de colinas, y es que puede llegar el momento en que ya no sea posible avanzar, debido a

que no exista algún movimiento que pueda mejorar la acción. Esto sucede cuando se ha alcanzado un máximo local, una meseta o una cresta.

a) Un **máximo local** se conoce como un estado que tiene como característica ser mejor que todos sus vecinos, pero no con respecto a otros más lejanos. Para este caso se recomienda regresar a algún nodo previo e intentar en una dirección diferente, esto suena razonable si en ese nodo había otra dirección que parecía prometedora (Ver Figura I.2.4).

b) Una **meseta**, se conoce como una área plana en el espacio de búsqueda, en la cual todo el conjunto de estados vecinos tiene el mismo valor. Para esta situación se sugiere realizar un gran salto en alguna dirección para intentar llegar a una nueva sección del espacio de búsqueda[6] (Ver Figura I.2.5).

c) Una **cresta** se define como una área del espacio de búsqueda que es más alta que otras que se encuentran colindando con ella, pero que no puede atravesarse mediante movimientos elementales en cualquier dirección. En esta circunstancia se recomienda la aplicación de 2 o más reglas antes de realizar la comprobación, lo cual representa el moverse en varias direcciones a la vez (Ver Figura I.2.6).

Este método no siempre resulta efectivo, ni siquiera tomando las medidas que se proponen para las situaciones que se acaban de mencionar, pero normalmente se combina con otros métodos que lo hacen comenzar en una vecindad adecuada.

3) Búsqueda en Amplitud

Los métodos anteriores son procedimientos de búsqueda en profundidad, asimismo, son fáciles de realizar y pueden llegar rápidamente a una buena solución. Sin embargo, suelen desperdiciar mucho tiempo explorando caminos infructuosos. Por lo que, una estrategia alternativa la constituye una búsqueda en amplitud, la cual consiste en examinar todos los nodos de un nivel de un árbol antes de pasar al siguiente nivel.

Este procedimiento garantiza que se llegará una solución, siempre y cuando exista y suponiendo además que hay un número finito de ramas del árbol. Es decir, si hay una solución entonces existe un camino de longitud N, que va desde el estado inicial hasta el estado final. Por lo que, la búsqueda en amplitud inspeccionará todos los caminos de longitud 1, en seguida los de distancia 2 y así sucesivamente hasta que haya explorado todos los caminos de magnitud N (Ver Figura I.2.7).

---

[6] Schildt (1990) define espacio de búsqueda como el conjunto de todos los nodos en un árbol. Utilización de C en Inteligencia Artificial. Pág. 17

Figura I.2.3 Un ejemplo del Método de Ascención de Colinas. Esta técnica involucra una función heurística que ordena la elección para expandir los nodos. Los números que aparecen a un costado de los nodos representan las distancias entre las ciudades.

Figura I.2.4 Máximo local.

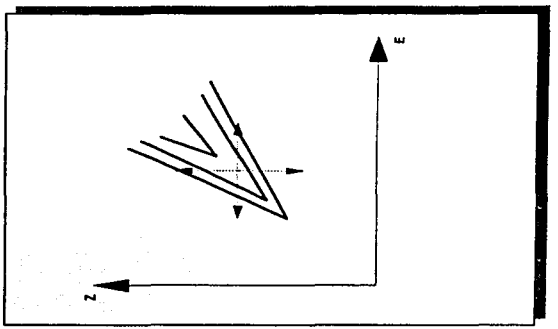

Figura I.2.6 Cresta.

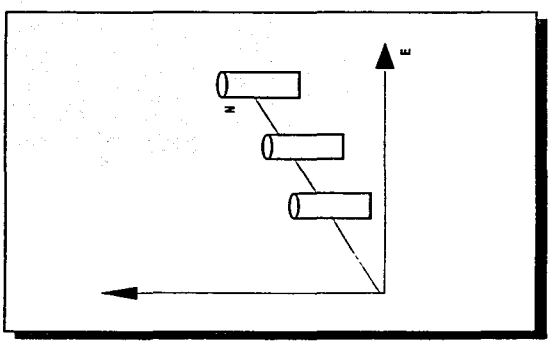

Figura I.2.5 Meseta.

Figura I.2.7 Un ejemplo de Búsqueda por Amplitud. Movimientos hacia abajo, procesados nivel por nivel hasta llegar al objetivo de la búsqueda.

Sin embargo, la búsqueda por amplitud presenta 3 problemas principales:

1) Necesita bastante memoria, pues el número de nodos en cada nivel del árbol se incrementa de manera exponencial con el número del nivel, además de que todos deben almacenarse a la vez.
2) Requiere demasiado trabajo, ya que el número de nodos que se necesita examinar se incrementa exponencialmente con la longitud del camino.
3) Los operadores irrelevantes incrementarán de manera gradual el número de nodos que se requiere inspeccionar.

4) Búsqueda del Mejor Nodo

Este tipo de búsqueda constituye una manera de combinar las ventajas que ofrecen los métodos de profundidad y los de anchura. A continuación se explica el procedimiento:

1) Seleccionar aquel nodo que parezca el más prometedor entre todos los que se hayan generado hasta el momento, lo cual se hace a través de la aplicación de una función heurística.
2) Expandir el nodo elegido. Si alguno de ellos resulta ser una solución, entonces se puede terminar con el proceso. De lo contrario continuar con el siguiente paso.
3) Agregar ese nodo al conjunto de nodos generados hasta esta etapa. Regresar al paso 1) (Ver Figura I.2.8).

5) Análisis Means-Ends

El estado de un sistema constituye una descripción de manera suficiente como para poder determinar el futuro. De tal forma que, en el espacio de estados, cada nodo representa un estado y cada línea de unión simboliza una transición de un estado a otro.

En el contexto de un problema, los estados corresponden a lo que sucede o a lo que sucederá. Por lo que, el estado actual corresponde a lo que ocurre en el momento, mientras que el estado meta, se refiere a lo que se espera en un futuro. Así, el problema es encontrar una secuencia de transiciones que conduzcan el estado inicial hacia el estado meta.

De esta manera, el propósito del Análisis Means-Ends es identificar un procedimiento que especifique una transición que vaya del estado actual al estado meta, pasando por un conjunto de estados intermedios. De tal forma que el procedimiento identificado pueda reducir las diferencias observadas existentes entre el estado actual y el estado meta (Ver Figuras I.2.9 y I.2.10).

Figura 1.2.8 Ejemplo de Búsqueda del Mejor Nodo. Este ejemplo muestra el principio de un procedimiento de búsqueda. Inicialmente existe un nodo único (Paso 1), por lo cual se expandirá, lo que hace que se generen 3 nodos (Paso 2). Posteriormente se les aplica la función heurística (en este caso es una estimación del costo de llegar a una solución desde un nodo determinado). Dado que el nodo D parece ser el más prometedor, éste se expande, lo cual produce los nodos sucesores E y F. A continuación se les aplica la función heurística (Paso 3). Sin embargo, ahora se observa que el nodo B es el mas prometedor, por lo que se expande, generando los nodos G y H, nodos que se evalúan (Paso 4), resultando ahora menos prometedores que el nodo E, de manera que se expande produciendo los nodos I y J (Paso 5). Más adelante se expandirá J, pues evidentemente se observa como el mejor y se continúa el proceso hasta encontrar una solución.

Figura I.2.9 Ejemplo del Análisis Means Ends. El Análisis Means-Ends involucra estados y procedimientos para reducir diferencias entre estados. El estado actual y el estado meta se muestran con líneas continuas. Los otros estados con línea punteada aún no se conocen y constituyen los estados intermedios

Figura I.2.10 Ejemplo del Análisis Means-Ends. El Análisis Means Ends produce una trayectoria que va del estado inicial al estado meta

6) Satisfacción de Restricciones

Existen muchos problemas en IA que pueden observarse como problemas de Satisfacción de Restricciones, donde el objetivo es encontrar un estado que pueda satisfacer un conjunto de restricciones. Entre algunos ejemplos de este tipo de problemas se encuentran los problemas de criptoaritmética, así como las tareas de diseño, las cuales deben crearse dentro de límites fijos de tiempo, costo y mantenimiento.

Así, estos problemas no requieren de un método de búsqueda propio, sino que se puede utilizar cualquiera de los ya mencionados. Lo que sucede es que la estructura de este tipo de cuestiones hace necesario aumentar una descripción del estado del problema junto con una lista de restricciones, las cuales se van a ir modificando conforme se vaya resolviendo el problema. De esta manera, el proceso se realiza con dos búsquedas concurrentes, una de ellas opera en el espacio del problema de las listas de requerimientos, mientras que la otra trabaja con el espacio del problema original.

La satisfacción de restricciones puede observarse desde otro punto de vista, esto es, como una forma de ascensión de colinas unimodal (es decir, sólo hay un máximo, el global), pues cada uno de los estados del espacio del problema en el que se realiza la ascensión de la colina, representa un conjunto de posibles soluciones que aún no han sido excluidas. Así, el progreso se define como el movimiento hacia un estado que representa un conjunto más reducido de soluciones posibles.

El procedimiento general consta de los siguientes pasos:

Realizar lo siguiente hasta que se haya encontrado una solución completa o hasta que todos los caminos hayan conducido a puntos muertos:

1) Seleccionar un nodo que no haya sido expandido aún en el grafo.
2) Aplicar las reglas de inferencia de restricciones al nodo seleccionado para generar nuevas restricciones.
3) Si el conjunto de restricciones contiene una contradicción, informar que ese camino es un punto muerto.
4) Si el conjunto de restricciones describe una solución completa, informar del éxito.
5) Si no se ha encontrado ninguna contradicción ni una solución completa, aplicar las reglas del espacio del problema para generar nuevas soluciones parciales que sean consistentes con el conjunto actual de restricciones. Insertar esas soluciones parciales en el grafo de búsqueda (Ver Figuras I.2.11 y I.2.12)[7]

[7] Para una explicación más detallada de este método ver Rich, E. (1988) Op. cit., págs. 103-108

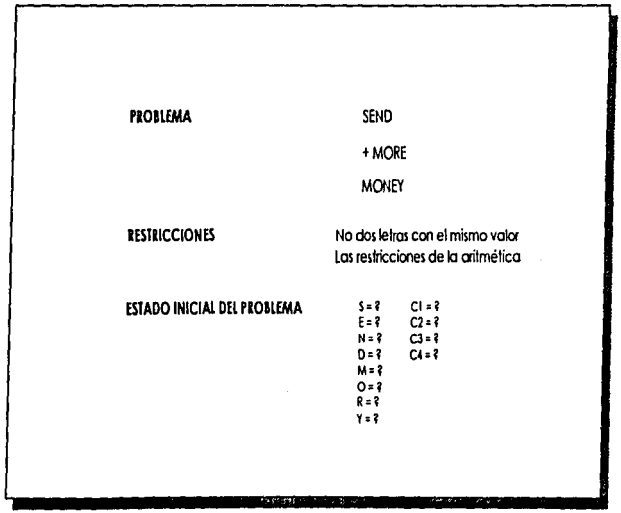

Figura I.2.11 Un ejemplo de Satisfacción de Restricciones. Este es un problema de Criptoaritmética, donde se muestra la lista inicial de requerimientos y el estado inicial del problema. La meta es un estado del problema en el cual se ha asignado un dígito a cada una de las letras, de tal manera que se satisfagan los requerimientos.

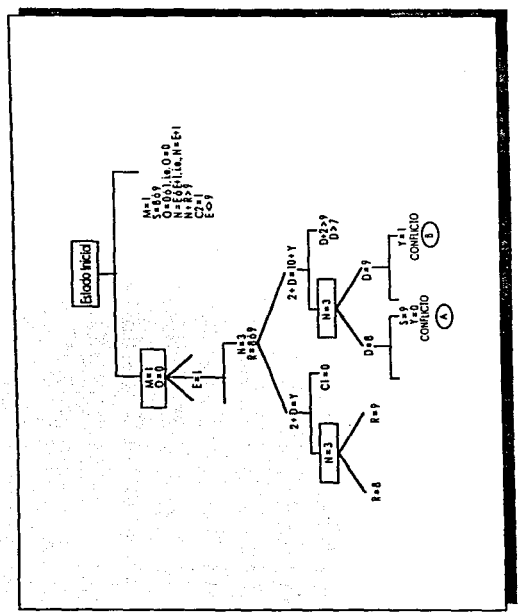

Figura I.2.12 Ejemplo de Satisfacción de Restricciones. Se muestran los resultados de los primeros ciclos procesados del ejemplo anterior.

I.3 APLICACIONES DE LA INTELIGENCIA ARTIFICIAL

El diseño y desarrollo de máquinas que exhiben lo considerado como características inteligentes, no sólo involucra una disciplina, por el contrario, comprende muchas y muy diversas ciencias y tecnologías, tales como la Lingüística, la Psicología, la Filosofía, el hardware y software computacional, Ingeniería Eléctrica, Matemáticas, Estadística, Biología, Ciencia de la Administración y Sistemas de Información Gerencial; contribuyendo cada una de ellas con una parte en específico e interactuando entre sí (Ver figura I.3).

Así la IA no es en sí misma un campo comercial, es un área que posee conceptos e ideas apropiadas para la investigación por lo que no son comercializadas. Sin embargo, proporciona los fundamentos científicos para desarrollar aplicaciones comerciales. Las principales las constituyen los Sistemas Expertos, la Robótica, Comprensión del Habla (voz), el Procesamiento del Lenguaje Natural y las Redes Neuronales.

Sistemas Expertos

Un Sistema Experto (SE) es un programa computacional que simula el comportamiento de un experto humano, en un área específica señalada como dominio del conocimiento (no se abundará aquí sobre sus características, pues más adelante hay un capítulo designado para ello).

Robótica

Una de las aplicaciones en la IA que ha más crecido es sin duda la Robótica.

Un Robot se ha definido como un manipulador multifuncional, reprogramable, que se encuentra diseñado para realizar diversas tareas, entre ellas materiales, partes, herramientas o, tareas especializadas. Entre las ventajas o beneficios que se pueden obtener de un Robot se enlistan las siguientes:

1) Incrementa la productividad
2) Reduce costos
3) Supera las habilidades de un ser humano
4) Proporciona flexibilidad en operaciones de manufactura por lotes
5) Mejora la calidad de los productos
6) Libera de aburrimiento al personal, así como de tareas repetitivas y ambientes hostiles.

Entre los países que emplean los Robots en sus industrias, Japón parece ser el líder en el ramo, siendo utilizados en los procesos de producción para realizar tareas como soldar con autógena, manejar materiales, en la carga y descarga de máquinas, en armado, pintado y terminados.

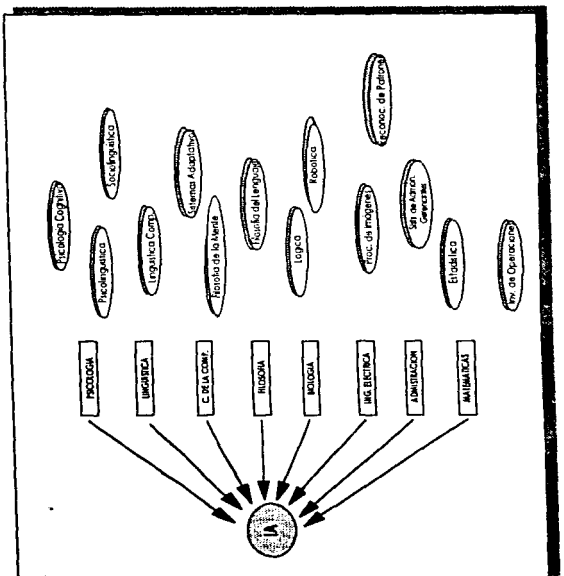

Figura I.3 Disciplinas de la Inteligencia Artificial.

Un Robot típico consiste de uno o más manipuladores (armas) y efectores (manos), un controlador y sensores que le proporcionan información del medio ambiente, permitiéndole retroalimentarse cada vez que lleva a cabo la tarea.

La Computación Visual juega un papel muy importante en la Robótica, pues la capacidad de un Robot para ver en un escenario de 3 dimensiones incrementa su habilidad para realizar su trabajo, como el poder inspeccionar defectos, ordenar partes, procesar documentos, etc.

Existen dos tipos de Robots:

1) Los Robots para exploración

Estos Robots se mueven en su medio ambiente, sintiéndolo y percibiéndolo.

2) Los Robots para manipulación

Éstos manipulan su medio ambiente o a los objetos que se encuentran en él, esto a través de sensores y manipuladores.

La mayoría de los robots industriales que hoy se usan, son los de manipulación o pick-and-place como también se les conoce.

La IA puede ayudar a convertir a los Robots en entidades inteligentes, no sólo para uso industrial, sino también en otras áreas como en el mantenimiento y reparación de barcos o equipos, en incendios y aún en la casa. En el espacio también son útiles, por ejemplo en la construcción y ensamble de grandes estructuras espaciales.

Comprensión del Habla (voz)

La Comprensión del Habla es otra área de la IA que se refiere a la comprensión del lenguaje hablado a través de una computadora. Es un proceso mucho más complejo que el procedimiento conocido como Síntesis del Habla[8] y del Reconocimiento del Habla[9] , pues el sistema puede poseer ya algunos conocimientos.

Existen dos factores principales que influyen en el proceso. Uno de ellos es la presencia de ruidos y otros tipos de inferencia, los cuales afectan las señales acústicas. El otro lo constituye la variación en cuanto la forma en que la gente habla.

Los proyectos más famosos en Comprensión del Habla son el HEARSAY y HARPY. El HEARSAY, un hablador dependiente[10] , el cual alcanzó un 90% de exactitud en el

[8] Esto se refiere a la Generación del Habla por medio de una computadora. Según Turban, E. (1992) en Expert Systems and Applied Artificial, este proceso es más simple que el Reconocimiento del Habla.
[9] Este proceso convierte el Habla a partir de un sonido en Patrones de Habla. Turban, E. (1992) Op. Cit.
[10] Esto es, un hablador que está diseñado para reconocer el Habla de una persona en particular. Turban, E. (1992) Op. Cit.

reconocimiento, mientras que el HARPY, un hablador independiente[11], lo superaba llegando a tener un 95% de precisión en el reconocimiento.

Procesamiento del Lenguaje Natural

La tecnología del lenguaje natural proporciona a los usuarios de computadoras la capacidad de poder comunicarse con ella, en una forma más directa, es decir, en su propio lenguaje. Sin embargo, prevalece un obstáculo que limita este propósito, pues no hay que olvidar la dimensión y diversidad de los lenguajes existentes.

Un trabajo en Procesamiento de Lenguaje Natural implica el uso de computadoras en el estudio de lenguajes, el cual se acentúa con la manipulación de símbolos, palabras y otras entidades lingüísticas.

Asimismo, involucra análisis de tipo sintáctico y procesos cognitivos que incluyen las estructuras de las oraciones, los significados de palabras, los modelos de los oyentes, así como también reglas de conversación.

En la actualidad, los investigadores en esta área, están incluyendo la visión como una nueva forma de representar y manipular el conocimiento, proporcionando una alternativa a problemas tales como, comprensión de metáforas y manejo de información lingüística inconsistente.

Redes Neuronales

Otra aplicación de la IA la constituye las Redes Neuronales. Una Red Neuronal puede verse como un sistema que simula la estructura y funcionamiento del sistema nervioso.

Esta tecnología inició alrededor de 1960, sin embargo, sólo en los últimos años es cuando ha tenido más auge.

Normalmente, las Redes Neuronales son empleadas en situaciones donde los datos son considerados como "ruidosos". Tales aplicaciones incluyen aquellas en que se necesita la interpretación de signos o donde se requiere de predicciones, por citar algunas.

En términos simples, las Redes Neuronales comprenden el entrenamiento de la red, en el cual el mecanismo o algoritmo empleado para el aprendizaje así como los casos examinados, son usados para alcanzar una respuesta deseada. De este modo, existen diferentes procedimientos para crear una Red Neuronal, sin embargo, el método de Retropropagación parece ser hasta el momento el más popular.

Para el futuro, se proyecta desarrollar más aplicaciones con Redes Neuronales. Entre ellas está la unión de éstas con los Sistemas Expertos y otras áreas de la IA.

[11] Se refiere a aquel hablador diseñado para reconocer el Habla de cualquier persona. Turban, E. (1992) Op. Cit.

I.4 HERRAMIENTAS METODOLÓGICAS DE LA INTELIGENCIA ARTIFICIAL

Como toda disciplina, la Inteligencia Artificial cuenta con herramientas metodológicas que le permiten abordar los problemas que se le presentan, es entonces que hace uso de los Lenguajes de Programación.

El lenguaje LISP

El LISP, (List Processing) es un lenguaje de programación funcional y simbólico. Es funcional en el sentido de que se basa en funciones orientadas al tratamiento recursivo de listas de *items*, esto es, no opera a partir de una secuencia ordenada de instrucciones con posibilidad de hacer bucles y direccionamientos *go to* . Más bien, se definen funciones con argumentos de entrada y valores de salida, permitiéndose de esta manera, una recursividad.

Por otro lado, es simbólico, ya que maneja estructuras de datos simbólicas como palabras y oraciones en vez de arreglos y números. Así , cualquier estructura de datos simbólica puede ser representada como un objeto, que puede ser simple (átomo) con la característica de ser indivisible; o compuesto (lista) que puede ser creado conteniendo cero o más objetos (*items*) cada uno, contando, además, con la posibilidad de componerse recursivamente.

Una tarea bajo LISP, no se interpreta como una secuencia de cálculos y decisiones consecutivas, sino como una estructura de funciones que transforman la información de entrada en la información de salida deseada.

La primera implementación de LISP fue en el periodo de 1959 a 1962, derivándose de ahí otras más y en muy diversas máquinas. Desde 1962 se han producido diversos dialectos del LISP, incluyendo principalmente al MACLISP e INTERLISP.

Aunque el LISP no ha sido estandarizado, existe mucho software auxiliar que facilita la conversión de programas de MACLISP a INTERLISP. Asimismo existe mucho interés en desarrollar una versión "estandar" del LISP, por lo que, en años recientes se ha trabajado intensamente en ella, creando una muy parecida al MACLISP llamada COMMON LISP.

En los últimos años se ha dado un significativo desarrollo del hardware para investigación en Inteligencia Artificial; lo cual ha dado lugar a la creación de las llamadas "máquinas LISP", ya que estas máquinas tienen la disposición de reemplazar la tradicional arquitectura tipo Von Neumann con otra que cuenta con la capacidad de poder evaluar directamente funciones y programas en LISP, ahorrándose con ello gran tiempo en ejecución.

Evaluación del LISP[12]

1	Expresividad	Bueno
2	Bien Definido	Excelente
3	Tipos y estructuras de datos	Bueno
4	Modularidad	Excelente
5	Facilidades de entrada-salida	Pobre
6	Transportabilidad	Regular
7	Eficiencia	Bueno
8	Pedagogía	Regular
9	Generalidad	Pobre

El lenguaje PROLOG

El lenguaje PROLOG (Programming in Logic) representa una nueva forma de programación, esto porque el estilo de PROLOG se basa en la idea de definir objetos y relaciones de inferencia entre clases de objetos. Asimismo, posee fuertes fundamentos teóricos en el cálculo de proposiciones.

Un programa en PROLOG se conforma de 5 partes:

1) Domain

Esta sección la constituyen las declaraciones de dominio, las cuales describen las diferentes clases de objetos que se emplearán en el programa.

2) Database

Aquí se encuentran los postulados de las Bases de Datos que son predicados que se utilizan en aquellas Bases que son de tipo dinámico.

3) Predicates

En esta parte se definen los predicados que se usarán en el programa.

4) Goal

Aquí se indica a PROLOG cual es el objetivo a alcanzar, esto a través del programa. Asimismo, se dice que es la forma de comunicarse con PROLOG, pues constituye la parte donde se le dice lo que tiene que realizar y lo que se va a buscar.

5) Clauses

En esta sección se declaran reglas y hechos del programa, que constituyen los datos del mismo.

Gran parte de la reciente atención a PROLOG se debe a su papel preponderante en el proyecto japonés de computadoras de la "Quinta Generación". Además de que su diseño representa una desviación de las ideas tradicionales sobre comportamiento de

[12] Evaluación hecha por Tucker, Allen B. (1987) Lenguajes de Programación. Págs. 368-369

programas, las cuales toman como base la arquitectura de máquina de Von Neumann. Sin embargo, y debido a que, el lenguaje no es ampliamente conocido, no ha variado mucho desde su primera implementación (1972), por lo que no se ha hecho ningún esfuerzo por estandarizarlo.

Evaluación del PROLOG[13]

1	Expresividad	Bueno
2	Bien Definido	Excelente
3	Tipos y estructuras de datos	Bueno
4	Modularidad	Bueno
5	Facilidades de entrada-salida	Bueno
6	Transportabilidad	Pobre
7	Eficiencia	Regular
8	Pedagogía	Pobre
9	Generalidad	Pobre

El lenguaje C

El desarrollo histórico del lenguaje C va muy de cerca al del propio UNIX. En el año de 1969, los Laboratorios Bell necesitaban de una alternativa para el sistema operativo Multics de la computadora PDP-7. Por lo que se escribió una versión original del sistema operativo UNIX en lenguaje ensamblador. Durante ese mismo periodo, Kenneth Thompson se encontraba desarrollando un lenguaje experimental, al cual llamó B y que fue diseñado después del BCPL, un lenguaje de programación de sistemas. Ya para 1972, Dennis Ritchie diseñó el C como una extensión del B; teniendo como diferencia principal una extensa condición de tipos estándares, que B no poseía.

Aunque C posee 5 tipos básicos de datos, no se considera un lenguaje potente en relación con Pascal o Ada. Sin embargo, C permite casi todos los tipos de conversiones.

C es singular en el sentido de que permite la manipulación de bits, bytes, palabras y el manejo de apuntadores, lo cual hace que se adapte muy bien a la programación a nivel de Sistema, donde este tipo de operaciones es muy común. Asimismo, C posee sólo 32 palabras clave, 27 provenientes del estándar de Kernighan y Ritchie y, 5 del comité de estandarización ANSI.

Pese a que en un principio fue desarrollado para ejecutarse en el Sistema Operativo UNIX, ahora ha ganado tanta popularidad que existen compiladores adecuados para todas las computadoras y Sistemas Operativos.

[13] Evaluación hecha por Tucker, Allen B. (1987) Op. Cit. Págs. 435-436

Evaluación del C[14]

1	Expresividad	Bueno
2	Bien Definido	Excelente
3	Tipos y estructuras de datos	Bueno
4	Modularidad	Excelente
5	Facilidades de entrada-salida	Bueno
6	Transportabilidad	Excelente
7	Eficiencia	Excelente
8	Pedagogía	Regular
9	Generalidad	Bueno

Comparación entre LISP y PROLOG contra C

Aunque se habla de lenguajes de IA por sus características y ventajas, también se referencia a otros lenguajes de programación como el C que están ganando terreno y que en un momento dado podrían sustituir a aquellos.

Por ejemplo, una de los aspectos que se está tomando en cuenta es la existencia de mayor cantidad de programadores que conocen y manejan el C, en comparación con otros lenguajes que comúnmente se utilizan en IA, por lo que parece evidente que si las técnicas de IA van a aplicarse a situaciones reales, necesariamente deberán ser codificadas en C.

Siguiendo con esa misma linea, sólo algunas técnicas se han abierto paso vertiginosamente en el mundo de la programación, esto debido a 2 causas: 1) La mayoría de los programadores cuentan con conocimientos básicos y formales sobre IA y 2) Mientras que la mayoría de la investigaciones en IA se hacen en lenguajes específicos como LISP y PROLOG, la mayoría de las aplicaciones reales están escritas en lenguaje común como C.

Así, todas las técnicas pueden ser implementadas en C, de hecho, el desarrollo de rutinas es más claro en este lenguaje que en otro de IA. Además, en un momento dado podría unirse un módulo escrito en C con otro hecho en PROLOG o LISP, sin embargo, no se recomienda, ya que existe la posibilidad de que se presenten problemas de direccionamiento, además, ello supondría que los programadores originales tendrían que aprender otro lenguaje más, o en su defecto, contratar otros. Por lo que, lo mejor sería implementar cualquiera de las técnicas de IA en un sólo lenguaje.

Es verdad que LISP posee una herramienta excelente para realizar máquinas de inferencia y PROLOG cuenta con instrumentos que permiten diseñar, gestionar y utilizar

[14] Evaluación hecha por Tucker, Allen B. (1987) Op. Cit. Págs. 480-481

Bases de Conocimiento, sin embargo, presentan limitantes en cuanto a las aplicaciones y además exigen muchos recursos para su implementación.

De ahí que existan motivos importantes para codificar en lenguaje C y que son:

1) Existen compiladores de C para casi cualquier computadora, así como para la mayoría de los sistemas operativos.
2) Un programa codificado y compilado en C puede ser tan rápido como si estuviera escrito en lenguaje Ensamblador.
3) El lenguaje C cuenta con la suficiente capacidad para poder implementar tanto Bases de Conocimiento como de datos, así como máquinas de inferencia, reconocedores de patrones y otras rutinas que se requieran para la aplicación.
4) Existen en el mundo más programadores en C que en LISP o PROLOG, lo cual facilita la codificación de las aplicaciones.

C++ es considerado como el lenguaje sucesor de C, pues incorpora facilidades para manejar objetos directamente desde su declaración. Éste se encuentra disponible en un gran número y variedad de máquinas, desde las micro hasta las supercomputadoras, ofreciendo todas la ventajas que proporciona el lenguaje C. Además, permite abstraer datos escondiendo detalles de implementación.

Asimismo, C++ permite trabajar en varios equipos, desarrollar módulos de un sistema en forma independiente uniéndolos al final, no obstante que cada uno tendrá sus propias características.

Debido a todo lo anterior, se vislumbra este lenguaje como el más apropiado para desarrollar gran variedad de aplicaciones, desde las simples hasta las complejas, tal como las requiera la IA.

CAPITULO II

INDUCCIÓN

CAPÍTULO II
INDUCCIÓN

II.1 INDUCCIÓN

Cuando se está diseñando un Sistema Experto, se necesita de la colaboración de un especialista en el área que pueda proporcionar sabiduría para el sistema; sin embargo existe un problema: la disponibilidad de esa persona o simplemente la imposibilidad de extraer el conocimiento a través de una entrevista. Es entonces donde se hace necesaria la intervención de un método que permita generar Reglas útiles al Sistema Experto sin la presencia de un erudito humano. Uno de tales procedimientos se basa en Técnicas Inductivas.

Así, se puede definir la Inducción como un proceso de razonamiento que se da a partir de un conjunto de hechos con el fin de proporcionar principios o reglas. En consecuencia, la Inducción es posible si existen ejemplos (muestras) que permitan crear patrones de decisión. De tal manera que, un ejemplo es sencillamente un registro de un evento pasado, el cual relaciona valores de un conjunto de factores de decisión (atributos) con algunos resultados obtenidos (las Tablas II.1.1 y II.1.2 lo muestran más claramente). Por lo que, un factor de decisión es sencillamente un número que es importante para predecir un resultado, esto es, una posible consecuencia de un ejemplo pasado. Sin embargo, para problemas con muchos factores de decisión o con un conjunto muy grande de ejemplos, el proceso de comparar nuevos valores con los de la muestra puede ya no funcionar, por lo que se requiere entonces emplear un algoritmo inductivo y, a partir de él, construir un árbol de decisión o un conjunto de reglas (Ver Figuras II.1.1 y II.1.2).

Por tanto, se presentarán algunos de estos métodos tales como el modelo neuronal denominado Teoría de la Resonancia Adaptativa y el algoritmo Induction Decision Tree.

II.2 TEORÍA DE LA RESONANCIA ADAPTATIVA (ART)

Redes Neuronales

El objetivo primordial de todos los sistemas neuronales es el llevar un control de manera centralizada de diversas funciones biológicas, donde algunas de ellas son las responsables de la provisión de energía, pues dicho sistema se encuentra conectado con la función de metabolismo, el control cardiovascular y la respiración.

FACTORES DE DECISIÓN			RESULTADO
LINEA DE VELOCIDAD	EDAD	TEMPERATURA	MOTOR
baja	viejo	alta	malo
normal	nuevo	normal	bueno
alta	medio uso	baja	bueno

Tabla II 1.1 Esta tabla ejemplifica los atributos tomados para un Problema de Producción que consiste en el Diagnóstico de un Motor

FACTORES DE DECISIÓN			RESULTADO
LINEA DE VELOCIDAD	EDAD	TEMPERATURA	MOTOR
alta	viejo	alta	malo
alta	viejo	normal	malo
normal	nuevo	normal	bueno
normal	viejo	alta	malo
alta	viejo	alta	malo
alta	nuevo	normal	bueno
normal	nuevo	normal	bueno
baja	nuevo	alta	malo
baja	nuevo	alta	malo

Tabla II.1 2 Esta tabla muestra un ejemplo de un evento pasado para el problema anterior.

Figura II.1.1 La Figura muestra un Árbol de Decisión para el Problema del Diagnóstico de un Motor.

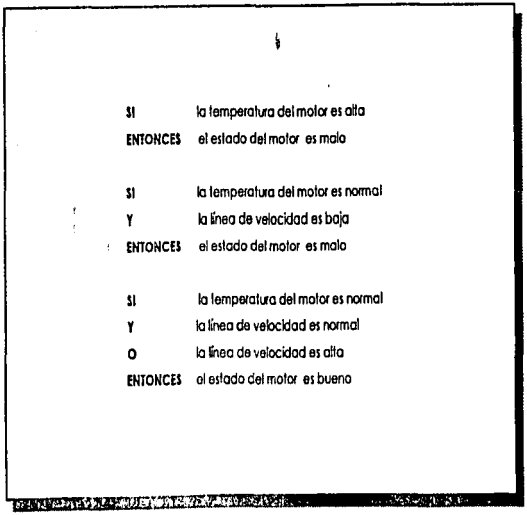

Figura II.1.2 Esta Figura muestra las Reglas Generadas a través de un Algoritmo de Inducción, referente al Problema de la figura anterior.

Todas estas funciones son ordinarias a la mayoría de los animales. No obstante, en los animales superiores, la mayor capacidad del sistema nervioso central está relacionada con el comportamiento. Por lo que, cuando se habla de Computación Neural, normalmente se hace referencia a las funciones sensoriales y motoras, así como al pensamiento, siendo interdependientes de una forma u otra, pero con la posibilidad de conceptualizarse alguna de ellas de una forma idealizada.

De esta forma, una de las propuestas de los sistemas inteligentes, es el llegar a crear computadoras que tengan la posibilidad de contar con arquitecturas y amplias capacidades de procesamiento, que tendrían la aptitud de poder imitar el funcionamiento del cerebro humano, para lo cual se requieren representaciones del conocimiento que se encuentren basadas en procesamientos en paralelo, así como la capacidad para reconocer patrones fundados en la experiencia. En este sentido, se han realizado infinidad de investigaciones, de las cuales, muchas de ellas se han enfocado a lo que se conoce como Redes Neuronales Artificiales (en inglés Artificial Neural Networks: ANNs), que son simples colecciones de elementos altamente conectados, que responden (o aprenden) de acuerdo a conjuntos de entradas.

1) La Neurona

a) Modelo Biológico

El Sistema Nervioso Central ha sido objeto de estudio por parte de la medicina, esto desde la época de la Edad Media, sin embargo, su estructura a detalle se conoció hasta hace solamente un siglo.

En la mitad del siglo XIX, ya existían dos escuelas que tenían sus propias concepciones acerca del Sistema Nervioso; por un lado se encontraban los **reticularistas**, quienes afirmaban que dicho Sistema, se encontraba constituido por redes continuas e interrumpidas de fibras nerviosas, mientras que por el contrario, los **neuronistas** aseveraban que estas redes estaban compuestas de un gran número de células interconectadas denominadas **neuronas**, por lo que, las disputas no se hacían esperar, sin embargo, alrededor de 1880, Camilo Golgi, inventó una técnica para el teñido de fibras nerviosas. Posteriormente, en 1881, dicha técnica fue empleada por el doctor español Santiago Ramón y Cajal, para que con ello refutara la doctrina de los reticularistas.

Posteriormente, la investigación detallada de la estructura interna de las células neurales, ha revelado que todas las neuronas se encuentran constituidas por los mismos elementos que forman parte de todas las células biológicas, pero conteniendo además ciertos componentes característicos que permiten diferenciarlas, existiendo así, cerca de 100 diferentes tipos de dichas neuronas.

En general, una neurona se compone de una parte central en forma de bulbo denominada **cuerpo celular** o **soma**, de donde se proyectan extensiones en forma de raíz llamadas **dendritas**, asimismo, también se encuentra el **axón**, el cual se ramifica en su parte final en un gran número de pequeños brazos.

Una de las particularidades que diferencian a las neuronas del resto de las otras células vivas es la capacidad que poseen para comunicarse. Así, en términos generales, las dendritas y el cuerpo celular funcionan como receptores al percibir señales de entrada (eléctricas y químicas), de esta manera, el cuerpo celular las combina e integra y emite señales de salida hacia otras neuronas dando origen a la **sinapsis**, es decir, la unión entre un axón neural y la dendrita de otra neurona. Por su parte, el axón cumple con la misión de transportar esas señas a los terminales axónicos, los cuales se encargan de distribuirlas a un nuevo conjunto de neuronas. De tal forma que éstas, se encuentran divididas en grupos interconectados denominados **redes**, donde cada grupo contiene diferentes neuronas que a su vez, están altamente entrelazadas una con otra. Así, el cerebro puede verse como una colección de redes neurales (Ver Figura II.2.1).

b) Modelo Artificial

Una Red Neural Artificial es un modelo que trata de emular una Red Neural Biológica, por consiguiente, lo que se pretende es reproducir, de alguna manera, el comportamiento del cerebro humano (Ver Figuras II.2.2 y II.2.3).

Así, la Computación Neural actual, emplea un conjunto de objetos de los Sistemas Neurales Biológicos, para con ello, implementar simulaciones en *software*, empleando dispositivos elementales de proceso denominados **neuronas artificiales**, las cuales se encuentran interconectadas en una arquitectura de red.

Comúnmente, en un modelo de red neural, se pueden encontrar 3 tipos de neuronas: las de entrada, las ocultas y las de salida.

En cualquier instante, cada i-ésima neurona se encuentra caracterizada por un valor numérico que se conoce como **estado de activación $a_i(t)$**; asimismo, asociado a cada unidad, existe una **función de salida f_i**, que tiene como objetivo transformar el estado actual de activación en una **señal de salida y_i**. Esta señal es enviada a través de canales de comunicación que apuntan a una sola dirección, pero hacia otras unidades de la red; en dichos canales, la señal se ve modificada de acuerdo a la sinapsis, es decir, el peso w_{ji}, el cual se encuentra asociado a cada uno de ellos (canales), esto de acuerdo a alguna regla determinada (Ver Figura II.2.4).

Así, las señales moduladas que han llegado a la j-ésima se combinan entre ellas, generando de esta forma, la entrada total Net_j

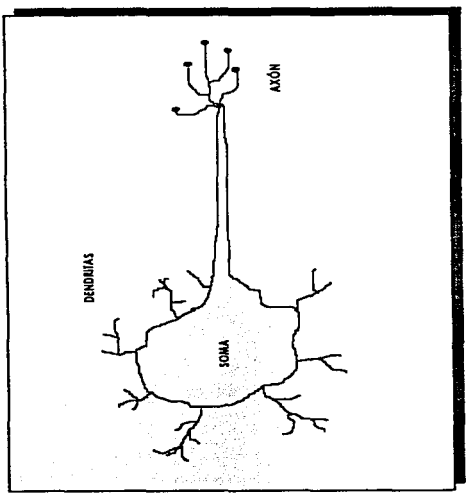

Figura II.2.1 Una Neurona Biológica

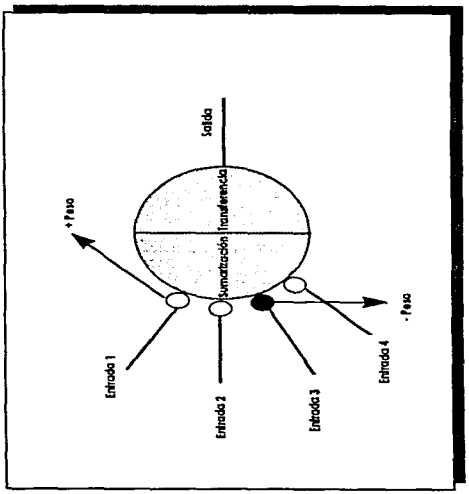

Figura II.2.2 Una Neurona Artificial

NEURONA BIOLÓGICA	NEURONA ARTIFICIAL
SOMA	NODO
DENDRITA	ENTRADA
AXÓN	SALIDA
SINAPSIS	PESO

Figura II.2.3 Comparación entre una Neurona Biológica y una Neurona Artificial.

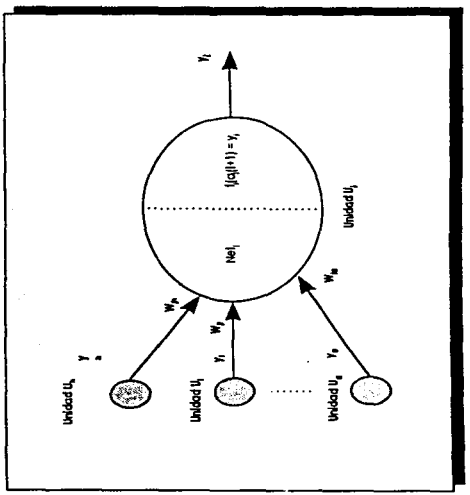

Figura II.2.4 Entradas y Salidas de una Neurona U_j

$$Net_i = \sum_i y_i w_{ji}$$

2) Definición de Red Neuronal

No existe en sí una definición que envuelva completamente lo que significa una Red Neuronal, pues dicha explicación varía de acuerdo a cada autor y a lo que pretenda referenciar, así, se pueden encontrar definiciones cortas, genéricas o incluso aquellas que pretenden explicar detalladamente lo que significa una Red Neuronal o Computación Neural. Por lo que aquí se presentarán algunas de ellas:

" Una nueva forma de computación, inspirada en modelos biológicos "[1] .

" Un modelo matemático compuesto por un gran número de elementos procesales organizados en niveles "[2] .

" ... un sistema de computación hecho por un gran número de elementos simples, elementos de proceso muy interconectados, los cuales procesan información por medio de su estado dinámico como respuesta a entradas externas "[3] .

" Redes Neuronales Artificiales son redes interconectadas masivamente en paralelo de elementos simples (usualmente adaptativos) y con organización jerárquica, los cuales intentan interactuar con los objetos del mundo real, del mismo modo que lo hace el sistema nervioso biológico "[4] .

" Una Red Neuronal es un sistema computacional que consiste de un número simple de elementos procesadores altamente interconectados, los cuales procesan información para determinar el valor de una señal de salida basada en valores de las señales de entrada "[5] .

---

[1] Hilera G., J. R. y Martínez H., V. J. (1995) Redes Neuronales Artificiales. Pág. 9.
[2] Hilera G., J. R. Y Martínez H., V. J. (1995) Op. Cit.
[3] Hecht-Nielsen (1988) "Neurocomputing: Picking the Human Brain", IEEE Spectrum, 25, págs. 36-41. Marzo, 1988. Reimpreso en el texto Artificial Neural Networks: Theoretical Concepts. (V. Vemuri ed.), págs. 13-18. IEEE Computer Society Press Technology; citado en Hilera G., J. R. y Martínez H. V. J. (1995) Op. Cit.
[4] Kohonen (1988): "An Introduction to Neural Computing" en Neural Networks. Vol. 1, págs. 3-16; citado en Hilera G., J. R. y Martínez H. V. J. (1995) Op. Cit.
[5] Engel, C.W. y Cran, M. (1990), "Pattern Clasification" en PC AI, pág. 20.

Así, de manera sencilla, el mecanismo con el que trabaja una Red Neuronal se diferencia notablemente de una computadora serial, pues mientras ésta es un procesador central que puede formar un arreglo de localidades de memoria, un modelo neural actúa en paralelo y no contiene datos almacenados.

De esta forma, se considera que la respuesta de los procesadores a los estímulos ocurre de una manera no lineal. Por otra parte, la influencia de uno sobre otro va a depender linealmente del peso existente entre sus conexiones, por lo que puede verse a una Red Neuronal como un sistema dinámico, no lineal, multivariable, con procesamiento distribuido en paralelo[6].

Por otra parte, a diferencia de los Sistemas Expertos, una Red Neuronal no requiere que el usuario especifique un número de reglas "si-entonces", sino que sólo necesita ejemplos específicos de valores de entrada con sus correspondientes valores de salida, pues la Red determina sus propias condiciones.

3) Historia de la Computación Neural o Neurocomputación

Desde muchos años antes a lo que se conoce como era cristiana, ya se daban explicaciones teóricas respecto al funcionamiento del cerebro y el pensamiento, como referencia de ello se encuentran los antiguos filósofos griegos como Platón (427-347 a.C.) y Aristóteles (384-422 a. C.), manteniéndose así, estas ideas en Descartes (1596-1650) y los filósofos empiristas del siglo XVIII.

De esta manera, las conocidas **máquinas cibernéticas**, a las cuales la computación neural pertenece, tiene más historia de lo que se piensa, pues se dice que Herón el Alejandrino, construyó alrededor del ano 100 a.C. un autómata hidráulico.

Ya en el siglo actual, Alan Turing, en 1936, fue el primero en estudiar el cerebro como una forma de ver el mundo de la computación, sin embargo, los primeros teóricos que concibieron los fundamentos de la computación neural fueron Warren McCulloch, un neurólogo y Walter Pitts, un matemático, quienes en 1943, proponen una Teoría General de Procesamiento de Información, que estaba basada en redes de selección binaria o elementos de decisión, a los cuales denominaban "neuronas", aunque no eran tan complejas como las biológicas. Cada uno de estos elementos $i = 1,...,n$, podían tomar sólo los valores $n_i = 0,1$; donde $n_i = 0$ representaba el estado de reposo, mientras que, $n_i = 1$ significaba el estado activo de cada unidad elemental.

Para simular un periodo finito de las neuronas, los cambios en el estado de la red suponen su ocurrencia en un tiempo discreto $t = 0,1,2,...$ Por lo que el nuevo estado de una

[6] Es decir, las neuronas actúan como simples dispositivos electrónicos que procesan información al mismo tiempo. Müller, Bernd y Reinhardt, Joachim (1991) Neural Networks, An Introduction, Pág. 4.

unidad neural estará determinado por la influencia de todas las neuronas, y estará expresado por la combinación lineal de sus valores de salida:

$$h_i(t) = \sum_j w_{ij}\, n_j(t) \qquad \text{(II.2.1)}$$

donde:

w_{ij} = representa la fuerza asociada a la sinapsis (peso) entre las neuronas j e i.

$h_i(t)$ = modela el potencial postsináptico total en la neurona i causada por acción de otras neuronas.

n_j = simboliza la salida.

Así, las propiedades de las Redes Neurales están determinadas por la relación funcional entre $h_i(t)$ y $n_j(t+1)$. En el caso más simple, la neurona se supone activa si su entrada excede un cierto umbral θ_i el cual puede variar entre una unidad y otra. Por lo que, la evolución de la red se encuentra determinada por la siguiente regla:

$$n_i(t+1) = \theta(h_i(t) - \theta_i) \qquad \text{(II.2.2)}$$

donde:

$\theta(x)$ es la función paso de la unidad, es decir, $\theta(x<1)=0$ y $\theta(x>1)=1$.

McCulloch y Pitts mostraron que tales redes, podían computarse, ya sea en una computadora digital, programable o en su abstracción matemática, la máquina de Turing, ya que en cierto sentido, la red contenía un "código de programa", el cual representaba el proceso computacional de la matriz w_{ij} . Así pues, la red difiere de la computación tradicional en que el programa no es ejecutado secuencialmente, sino en paralelo en cada unidad elemental, consistiendo de sentencias simples, esto es, la combinación de las ecuaciones (II.2.1) y (II.2.2).

Sin embargo, el diseño de una Red Neural del tipo McCulloch-Pitss, presentó un problema: Cómo cambiar las cuplas w_{ij} de manera que una tarea cognitiva específica sea ejecutada por la máquina, entendiendo por "tarea cognitiva" aquella tarea que requiere un procesamiento de información digital, tal como el reconocimiento de patrones acústicos u ópticos. Esta cuestión fue tratada en 1961 por Eduardo Caianello, quien da un algoritmo de "aprendizaje" , el cual sería capaz de permitir la determinación de las fuerzas sinápticas (pesos) de una Red Neuronal. Este algoritmo llamado mnemónico de la ecuación Caianello, incorpora una forma simple del principio básico de la regla de aprendizaje de Hebb.

Alrededor de 1957, Frank Rosenblatt y sus colaboradores estudiaron ampliamente un tipo específico de Redes Neuronales, al cual denominaron **PERCEPTRÓN**, puesto que representaba un modelo simplificado de los mecanismos biológicos del procesamiento de información sensorial, esto es, la **percepción**. De manera sencilla, un Perceptrón consiste de 2 estratos separados de neuronas, los cuales van a significar las respectivas capas, tanto de entrada como de salida (Ver Figura II.2.5). Las neuronas de la capa de salida reciben los signos sinápticos de la capa de entrada, pero no a la inversa, asimismo, las neuronas que se encuentran dentro de una capa no pueden comunicarse entre ellas. Así, el flujo de información es estrictamente direccional, esto es, una alimentación hacia adelante (feed-forward).

Este modelo tenía la capacidad de llegar a generalizar, es decir, después de haber aprendido una serie de patrones podía reconocer otros similares, aún cuando no se le hubieran presentado con anterioridad.

En 1959, Bernard Widrow y Marcial Hoff, llegaron a desarrollar su propio modelo neural, conocido como **ADALINE** (ADAptative LINer Elements), el cual constituyó la primera red neuronal aplicada a un problema real al utilizarse en filtros adaptativos para eliminar ecos en las líneas telefónicas.

Uno de los principales investigadores en Redes Neuronales desde los años 60, hasta la actualidad, es Stephen Grossberg, quien en 1967, construyó una red denominada **Avalancha**, la cual consistía de elementos de tipo discretos, cuya actividad iba variando con respecto al tiempo, a la vez que se satisfacía un conjunto de ecuaciones diferenciales continuas. Dicha red tenía la capacidad de resolver actividades tales como, reconocimiento continuo del habla y aprendizaje del movimiento de los brazos de un robot.

Para 1969 surgieron numerosas críticas que afectaron las investigaciones sobre Redes Neuronales, una de ellas es la de Marvin Minsky y Seymore Paper, quienes mostraron la existencia de cuestiones relativamente sencillas, las cuales no podían ser resueltas por 2 capas del Perceptrón. La más notoria de estas fue el problema de la función OR-exclusiva (XOR), pues requiere de 2 neuronas de entrada para que pueda ser conectada con una neurona de salida, de tal forma que, la unidad de salida es activada si y sólo si, una de las unidades de entrada es activada. Asimismo, dicho modelo era incapaz de clasificar clases que no eran separables linealmente.

A pesar de las críticas al Perceptrón, otros investigadores continuaron con su trabajo, tal es el caso de James Anderson, quien desarrolló un modelo lineal denominado **Asociador Lineal**, el cual se basa en el principio de que las conexiones existentes entre las neuronas son reforzadas cada vez que están activadas. Asimismo, diseñó una extensión del Asociador Lineal al cual llamó **Brain-State-In-a-Box(BSM)**.

Figura II.2.5 El Perceptrón Simple, constituendo de 2 capas de neuronas. Las neuronas de entrada alimentan a las de salida, pero no a la inversa.

Por otro lado, en Europa y Japón, también se realizaron investigaciones, por ejemplo, Kunihiko Fukushima desarrolló un modelo de Red Neuronal para el reconocimiento de patrones visuales al cual denominó **Neocognitrón**. Por su parte, Teuvo Kohonen, desarrolló un modelo parecido al de Anderson, pero de manera independiente.

En 1974, William Little, mostró cierta similitud entre una Red Neuronal del tipo propuesto por McCulloch y Pitts y los sistemas de momentos magnéticos elementales o spins.

En estos sistemas llamados **MODELOS ISING**, el spin s_i en cada enrejado i puede tomar sólo 2 diferentes orientaciones, ya sea arriba o abajo, denotada por $s_i = +1$ (si es arriba) y $s_i = -1$ (si es abajo). La analogía con la Red Neuronal se realiza identificando cada spin con una neurona, así como asociando la posición hacia arriba $s_i = +1$ con el estado activo $n_i = 1$ y la alineación hacia abajo $s_i = -1$ con el estado de reposo $n_i = 0$ (Ver Figura II.2.6).

Estas ideas fueron desarrolladas más tarde por el mismo Little y Gordon Sham en 1978, y por John Hopfield en 1982, quienes estudiaron cómo, tanto una Red Neuronal como un sistema Spin puede almacenar y transportar información.

Sin embargo, los modelos de Little y Hopfield poseen una notable diferencia, en lo referente a la manera en como el sistema se actualiza, pues en el primero, todas las neuronas (spins) se reconstruyen de una manera sincronizada, mientras que en el prototipo de Hopfield, éstas se actualizan secuencialmente, es decir, cada una en un cierto orden fijo o aleatorio.

La analogía de las Redes Neuronales con los sistemas Spin, resultó de gran beneficio, debido a los avances que se presentaron en el entendimiento de las propiedades de la termodinámica de los sistemas desordenados spins, los así llamados **SPIN GLASSES**, alcanzados en la década pasada.

Para aplicar estos resultados a las Redes Neuronales, es necesario reemplazar la ecuación (II.2.2) por una regla estocástica, donde el valor $n_i(t+1)$ va a estar determinado de acuerdo a una función probabilística, la cual va a depender de la intensidad de la entrada sináptica h_i. Asimismo, esta función de probabilidad contiene un parámetro T que juega el papel de "temperatura". Sin embargo, T no es entendido como la temperatura física de una Red Neuronal Biológica, sino como un concepto formal diseñado para introducir la estocasticidad en la red y así permitir la aplicación de las técnicas de la termodinámica estadística.

En años recientes, el interés por las Redes Feed-Forward (Perceptrones) ha vuelto a aparecer, luego del redescubrimiento de un algoritmo eficiente para la determinación de las fuerzas sinápticas, es decir, pesos, en redes multicapa, esto con la ayuda de las denominadas **CAPAS OCULTAS**. El poder de este método ya había sido conocido por Werbos hacia el año de 1974, para posteriormente ser retomado con el nombre de

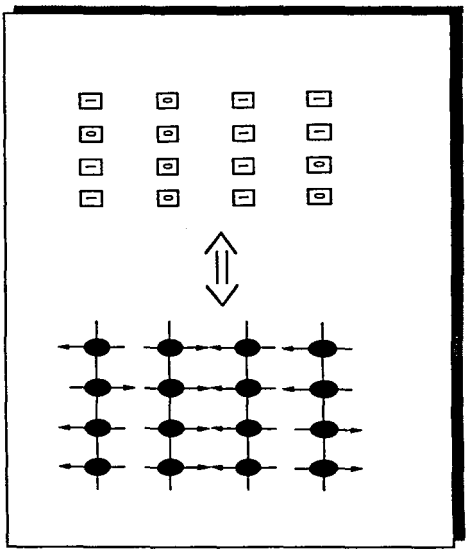

Figura II.2.4 Una correspondencia 1 a 1 existe entre los spins magnéticos con 2 diferentes orientaciones y las Redes Neuronales McCulloch-Pitts.

Retropropagación del error por diferentes grupos de científicos (Y. Le Con, 1986; D. B. Parker, 1985 y D. E. Rumelhart, 1986). Este algoritmo de aprendizaje se basa en un principio muy simple: las fuerzas sinápticas w_{ij} son modificadas iterativamente de manera que, la señal de salida difiera muy poco de la deseada. Dicha salida es alcanzada mediante la aplicación del método del Gradiente, el cual requiere de modificaciones δw_{ij}, así, la operación de la red corresponde a un mapeo no lineal entre la entrada y la salida. Por lo que, método puede ser aplicado muchas veces hasta que la convergencia sea alcanzada.

La Retropropagación del error es un particular ejemplo de una gran clase de algoritmos de aprendizaje, los cuales son clasificados como **APRENDIZAJE SUPERVISADO**, ya que en cada paso la red es ajustada comparando la salida actual con la salida deseada.

4) Elementos de una Red Neuronal Artificial

Como ya se mencionó anteriormente, una Red Neuronal Artificial se compone de distintos elementos, por lo que aquí se verán más a detalle.

a) La Neurona Artificial

Las neuronas constituyen las unidades, neurodos, células o Elementos de Procesamiento (Eps) de una Red, las cuales tienen como función recibir las entradas de las unidades vecinas o del mundo exterior, para posteriormente, calcular un valor de salida, el cual es enviado a todos los demás elementos. Cabe mencionar que cada EP tiene muchas entradas, pero sólo una salida y así, ésta servirá de entrada otro EP de la Red. De tal forma que, si se tienen N neuronas, éstas se pueden ordenar de forma arbitraria y designar la j-ésima unidad como U_j.

Por otra parte, se denomina **capa o nivel** al conjunto de EPs cuyas entradas provienen de la misma fuente (puede ser otra capa de neuronas) y además sus salidas se dirigen hacia el mismo destino.

b) Estado de Activación

Además del conjunto de unidades se requiere una representación de los estados del sistema para un cierto tiempo t. Entonces para ello se emplea un vector con N números reales, designado como A(t) y así, cada elemento va a representar el estado de activación de cada EP en el tiempo t.

Por lo que, la activación de una unidad U_i en el tiempo t se representa por $a_i(t)$ de tal suerte que A(t) se puede representar de la siguiente manera

$$A(t) = \{ a_1(t), a_2(t), \ldots, a_i(t), \ldots, a_N(t) \}$$

Todas las neuronas que forman parte de la Red se encuentran en un cierto estado, es decir, el estado de reposo o el de excitación, asignándoseles a cada uno un valor. Dichos valores (valores de activación) pueden ser de tipo discreto o continuo. Así, cuando se habla de valores discretos éstos pueden ser binarios o bien un pequeño conjunto de valores, por ejemplo, si se tratase de binarios, el número 1 representaría un estado de excitación o actividad, lo cual implicaría la emisión de un impulso de la neurona, por el contrario, si fuese un 0, ello indicaría un estado de reposo o pasividad.

Existen modelos en los cuales los estados de activación son continuos, en donde se les asigna un valor entre [0,1] o entre [-1,1], siguiendo normalmente una función sigmoidal (se explica más adelante).

Por otra parte, resulta importante conocer qué criterios siguen las neuronas para alcanzar su estado de activación, los cuales van a depender básicamente de 2 factores: uno de ellos es que, una neurona, al formar parte de una Red, no es una entidad simple, si no que forma parte de un sistema, es decir, de un todo, por lo que, su estado de activación dependerá en gran medida de las interacciones existentes entre todas las neuronas, pues el efecto que produce una unidad sobre otra será proporcional al peso de la conexión entre las dos. Asimismo, la señal que envía cada neurona va a estar sujeta a su propio estado de activación.

c) Función de Salida o de Transferencia

Entre cada neurona que conforma una Red Neuronal Artificial, existe un conjunto de conexiones que las unen, así cada unidad transmite señales a aquellas neuronas que se hallan conectadas con su salida. En términos matemáticos, cada unidad U_i tiene asociada una función de salida $f_i(a_i(t))$, la cual transforma su estado actual de activación $a_i(t)$ en una señal de salida $y_i(t)$, esto es:

$$y_i(t) = f_i(a_i(t))$$

Por consiguiente, el vector que contiene las salidas de todas las neuronas en el tiempo t se representa como:

$$Y(t) = (f_1(a_1(t)), f_2(a_2(t)), \ldots, f_i(a_i(t)))$$

Existen diferentes tipos de funciones de transferencia, entre las cuales se muestran las siguientes:

i) Función Escalón (Umbral)
ii) Función Lineal y Mixta
iii) Función Sigmoidal
iv) Función Gaussiana

De las cuales, la sigmoidal es la más empleada, pues utiliza un límite superior e inferior, lo que hace que exista un rango proporcional entre ambos.

I) Función Escalón (Umbral)

Esta función es la más simple de todos los modelos no lineales, asimismo es una función "todo-nada". Normalmente es utilizada cuando la activación de una neurona es de tipo binaria. Así, cuando la suma de las entradas que llega a una unidad es mayor o igual a que un cierto valor umbral, la activación es 1; si ocurriese lo contrario, dicha activación sería 0 o -1. Cabe mencionar que la función puede encontrarse desplazada sobre los ejes (Ver Figura II.2.7).

II) Función Lineal y Mixta

La Función Lineal o Identidad se empleó bastante en los modelos más antiguos. Equivale a no aplicar función de salida, expresándose como $f(x) = x$. Por lo cual se utiliza muy poco (Ver Figura II.2.8).

En el caso de neuronas con función mixta, se manejan dos límites, el inferior y el superior. Así, si la suma de las señales de entrada es menor o igual que el límite inferior, la activación corresponde a 0 o -1. Cuando dicha suma es mayor o igual al límite superior la activación de la neurona es 1. En la situación intermedia, es decir, cuando la suma de las entrada está entre el límite inferior y superior, la activación está definida como una función lineal de la suma de las señales (Ver Figura II.2.9).

III) Función Sigmoidal

En lo que respecta a la Función Sigmoidal, para casi todos los valores de entrada, la cantidad que se obtiene de dicha función es cercana a uno de los valores asintóticos, por lo que en la mayoría de los casos, el valor de salida se encuentra, en la zona alta o baja del sigmoide. De hecho, cuando la pendiente es elevada, la función tiende a la de escalón. Sin embargo, su importancia recae en el hecho de que su derivada siempre es positiva y cercana a cero para valores grandes, tomando su valor máximo cuando x=0, lo cual permite emplear Reglas de Aprendizaje que con otras funciones no se pueden utilizar (Ver Figuras II.2.10 y II.2.11).

Figura II.2.7 Función Escalón. Para ambas gráficas se ha tomado como un umbral cero. En el caso en que fuera otra cantidad, el escalón quedaría desplazado.

Figura II.2.8 Función Lineal.

Figura II.2.9 Función Mota. Para ambas gráficas se ha tomado como límite inferior -c y como límite superior c.

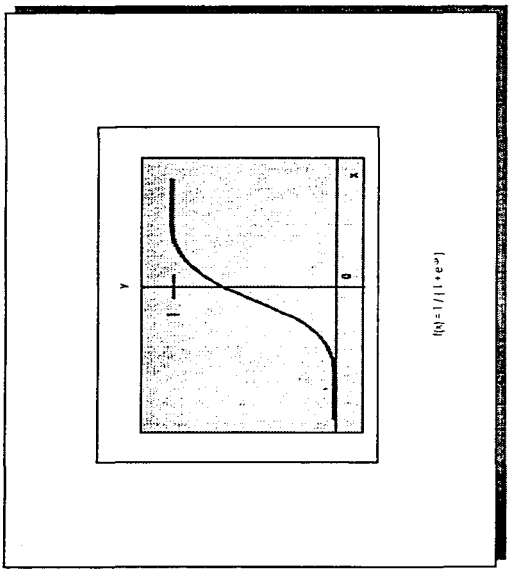

Figura II.2.10 Función Sigmoidal

Figura II.2.11 Derivada de la Función Sigmoidal de la gráfica anterior.

iv) Función Gaussiana

El centro y anchura de este tipo de función pueden ser ajustados, cosa que las hace más adaptativas que las sigmoidales. Así, para este tipo se requiere de 2 niveles ocultos, empleando neuronas con función de transferencia sigmoidal (Ver Figura II.2.12).

d) Conexiones entre Neuronas

Las conexiones que unen una neurona con otra, llevan asociado un peso, el cual es el que hace que una Red Neuronal adquiera conocimiento, pues cada EP recibe información del estado de activación de todas la unidades con las que se halla conectado.

Así cada conexión (sinapsis) entre la neurona i y la neurona j está ponderada por un peso w_{ji}. De esta manera, suele emplearse una matriz W que contiene todos los pesos w_{ji}, los cuales representan la influencia que tiene la neurona i sobre la neurona j. Los valores de la matriz W pueden ser tanto positivos como negativos e incluso nulos. Así, si un elemento w_{ji} es positivo indica que la interacción entre las unidades i y j es excitadora, en otras palabras significa que si la unidad i está activada, la neurona j recibirá la señal de i que tenderá a activarla. Por el contrario, si w_{ji} es negativo, ello representaría que la sinapsis entre ellas será inhibidora, esto es, si el nerodo i está activado enviará la señal a j que tenderá a desactivarlo. Asimismo, también puede darse el caso en el que existan conexiones inhibitorias desde una neurona hacia el resto de los EP de la misma capa, fenómeno que se le conoce como inhibición lateral. Por último, si w_{ji} es nulo esto significara la ausencia de conexión entre las unidades.

e) Regla de Aprendizaje

Existen muchas definiciones para el concepto de Aprendizaje, dependiendo del área de estudio y del contexto que se esté analizando. Sin embargo, de una manera general podría verse como una modificación del comportamiento debido a la interacción con el entorno y del resultado de experiencias que conducen al establecimiento de nuevos modelos de respuesta a estímulos externos.

En Biología, suele aceptarse que la información memorizada en el cerebro se encuentra relacionada con los valores sinápticos de las conexiones entre las neuronas, por lo que puede decirse que el conocimiento se halla en la sinapsis. De manera análoga, en las Redes Neuronales Artificiales, dicho conocimiento se alberga en los pesos de las conexiones existentes entre las neuronas, lo que implica que la Red va a aprender (o ganar experiencia) modificando y ajustando los valores de sus pesos, es por ello que se le considera a la Regla de Aprendizaje como el corazón de la Red. De tal manera que, cada modelo debido a sus características posee sus propias Reglas de Aprendizaje, entre las más

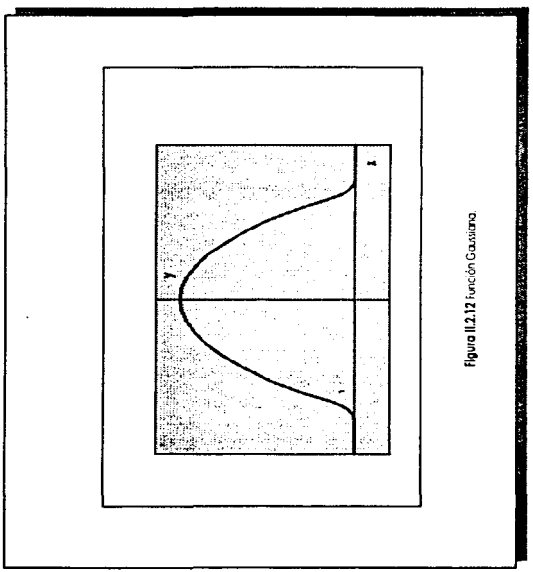

Figura II.2.12 Función Gaussiana.

conocidas se encuentran la Regla de Hebb y la Regla Delta, de las cuales se han realizado muchas variaciones.

5) Estructura de una Red Neuronal

Las neuronas poseen cierta distribución dentro de la Red Neuronal, esta se realiza a través de la formación de niveles o capas con un cierto número de unidades cada una. Dependiendo de su situación dentro de la Red, dichas capas pueden categorizarse en 3 tipos:

a) **Capa de Entrada:** Esta capa se encuentra en contacto directo con el mundo exterior y es la que se encarga de recibir la información que entra a la Red.

b) **Capas Ocultas:** Son aquellas que no poseen contacto directo con el exterior, es decir son internas. El número de éstas puede variar entre 0 y un número cualesquiera. En lo referente a las unidades que están en cada nivel oculto, pueden estar interconectadas de distintas maneras lo que determinará su topología.

c) **Capa de Salida:** Esta capa al igual que la de entrada, también se encuentra en contacto con el mundo externo, sólo que ésta transfiere la información de la Red.

6) Topología de las Redes Neuronales

La Topología o Arquitectura de las Redes Neuronales se refiere básicamente a la manera en como se encuentran organizadas y dispuestas las neuronas, así se ve que éstas se hallan formando capas o niveles como se mencionó anteriormente. En este sentido, se pueden identificar ciertos parámetros fundamentales como el número de capas, número de unidades por capa, grado de conectividad, así como el tipo de conexiones existente entre las neuronas.

De tal manera que, cuando se realiza una clasificación en términos topológicos, suelen identificarse Redes con una sola capa o nivel de neuronas o bien, Redes con múltiples capas.

a) Redes Monocapa

En este tipo Redes Monocapa se establecen conexiones laterales entre las neuronas que pertenecen a su única capa. Como ejemplo de este prototipo, se encuentra la Red BRAIN-STATE-IN-A-BOX y la Red de Hopfield (Ver Figura II.2.13).

Figura II.2.13 Topología de una Red Neuronal Hopfield.

b) Redes Multicapa

Estas Redes son aquellas que disponen de conjuntos de neuronas agrupadas en varios niveles o capas (2 o más). Así, normalmente todas las neuronas de una capa reciben señales de entrada de otro nivel anterior, enviando posteriormente señales de salida a una capa consecutiva (feedforward). Asimismo, existen Redes en las que se presenta la posibilidad de conectar las salidas de la neuronas de capas posteriores a las entradas de las capas anteriores (feedback).

Teoría de la Resonancia Adaptativa

Una de las maravillas del cerebro humano es su memoria, pues posee la capacidad de aprender cosas nuevas sin olvidar las que ya almacenó, por ejemplo, podemos reconocer a nuestros padres, aún si los dejamos de ver por mucho tiempo y a pesar de estar en contacto con personas y cosas nuevas. Así, esta característica tan peculiar del hombre sería muy deseable podérsela transmitir a una Red Neuronal Artificial, ya que muchos de los modelos Neuronales tienden a olvidar el conocimiento adquirido cuando se les agrega más información.

Generalmente cuando se está desarrollando una Red Neuronal para reconocimiento de patrones, lo normal consiste en obtener una muestra de datos y utilizarlos para entrenar dicha Red (es decir que aprenda), de tal manera que, mientras se encuentra en la fase de instrucción se están ajustando los valores de los pesos, para que posteriormente sean empleados en la etapa siguiente: el reconocimiento o diagnóstico, sin tener la posibilidad de modificarlos.

Este procedimiento puede ser suficiente y funcional para muchas situaciones donde el problema tenga límites definidos y que además sea estable. Sin embargo, el mundo real no siempre opera así, pues el entorno no está acotado y mucho menos es estable.

Esta situación que se acaba de describir es a lo que Stephen Grossberg llama **dilema de estabilidad y plasticidad**, el cual se puede esquematizar de la siguiente manera:

> "¿Cómo puede un sistema (capaz de aprender) seguir siendo adaptable (plástico) en respuesta a una entrada significativa y permanecer estable en respuesta a entradas irrelevantes?
>
> ¿Cómo sabrá el sistema alternar entre los modos plástico y estable ?
>
> ¿Cómo puede el sistema retener la información aprendido anteriormente y seguir aprendiendo cosas?"[7]

[7] Freeman, James A. y Skapura, David M. (1993) Redes Neuronales, Algoritmos, Aplicaciones y Técnicas de Programación. Pág. 308.

Como respuesta a tales preguntas Grossberg, Carpenter y otros colaboradores desarrollaron la **Teoría de la Resonancia Adaptativa (Adaptive Resonance Theory: ART)**, la cual constituye una extensión de los mecanismos de aprendizaje competitivo[8].

Así, lo que se pretende es categorizar (clusterizar) los datos que entran a la Red, de tal manera que, la información que sea semejante a la que ya se encuentre en el sistema, se clasifique formando parte de la misma categoría (grupo o clase), en caso contrario crear una nueva.

Entonces, para resolver el problema de estabilidad y plasticidad, proponen agregar un mecanismo de realimentación entre el estrato competitivo y la capa de entrada de la Red, facilitando el aprendizaje y evitando que se pierda la información ya almacenada.

De tal modelación resultaron dos arquitecturas neuronales muy adecuadas para problemas de clasificación de patrones, a las cuales denominaron ART1 (o simplemente ART) y ART2, que difieren una de la otra en lo que se refiere a la naturaleza de los patrones de entrada, pues las Redes ART1 requieren que sus valores de entrada sean de tipo binario, mientras que las ART2 son apropiadas para procesar informaciones con datos reales.

Por tanto, como la información que se manejará en este trabajo no es de tipo binario, sino real, entonces se aplicará el modelo ART2.

1) Arquitectura de una Red ART2

En el año de 1987, Carpenter y Grossberg propusieron una nueva versión del modelo ART, a la cual denominaron ART2. Dicho modelo a diferencia del ART1 tiene la capacidad de procesar información de entrada ya no binaria, sino de tipo real.

En general una Red ART consta de 2 capas de neuronas (entrada y salida) en las cuales existen conexiones hacia adelante (feedforward) y conexiones hacia atrás (feedback), de tal manera que, los nodos que pertenecen a la **capa de salida** poseen **conexiones autorecurrentes** y por ello, las estimaciones de sus pesos son +1, a su vez, estas mismas unidades se encuentran conectadas entre sí a través de enlaces laterales de inhibición con valores negativos ($-\varepsilon$)[9] con el fin de que se pueda asentar la competencia entre ellas. En lo referente a los pesos de las conexiones feedforward (w_{ji}) y feedback (v_{ij}), es donde se presenta la diferencia entre el modelo ART1 y ART2, pues en la primera Red

[8] En un aprendizaje competitivo los nodos de la Red compiten entre sí basándose en ciertos criterios dados, así el ganador clasifica en patrón de entrada. Freeman y Skapura (1993) Op. Cit.

[9] Este valor suele ser menor que 1/M, donde M es el número de neuronas de salida, con lo cual se garantiza la convergencia de la Red. Hilera y Martínez (1995) Redes Neuronales Artificiales. Pág. 222.

$$w_{ij} = \frac{1}{1+N} \qquad v_{ij} = 1$$

mientras que en un ART2 $w_{ij} = v_{ij}$ (Ver Figura II.2.14).

Sin embargo, la figura mostrada no constituye en su totalidad el funcionamiento de la Red, por lo que normalmente se emplea una representación sistémica como alternativa (Ver la Figura II.2.15). En dicha imagen pueden distinguirse 2 subsistemas, el **subsistema de atención** (attentional subsystem) y el **subsistema de orientación** (orienting subsystem).

En lo referente al Subsistema de Atención, éste tiene como función la de reconocer y clasificar la información que entra a la Red, para ello cuenta con estructuras que cumplen un papel específico en dicha tarea, así pues se encuentra la Memoria a Largo Plazo (LTM: Long Term Memory), la cual está compuesta por los pesos w_{ij} y v_{ij} (en este caso son iguales) que son los encargados de brindarle estabilidad a la Red al retener la información aprendida. Otra constitución que también se halla ahí es la Memoria a Corto Plazo (STM: Short Term Memory), que se localiza en la entrada y salida de la Red, de tal manera que, posee la capacidad de recordar los aspectos más inmediatos a través de la retención temporal de los valores de entrada en las neuronas que conforman las capas de entrada y salida. Finalmente, otra estructura que se puede encontrar es el Control de Ganancia de Atención, el cual actúa sobre los EPs de entrada (STM(ne)) con el fin de estabilizar el funcionamiento de la Red, aumentando con ello la sensibilidad o grado de atención de estas unidades.

En cuanto al Subsistema de Orientación, este cumple con la función de detectar si los datos que ingresan a la Red pertenecen o no a una categoría conocida por la misma, esto mediante el cálculo del porcentaje de semejanza existente entre la información de entrada y el representante de cada categoría que se encuentra almacenado en los pesos v_{ij} (LTM), si dicho porcentaje es inferior o igual a un parámetro ρ conocido como parámetro de vigilancia o umbral, entonces se puede decir que los datos que entraron a la Red, pertenecen a ese grupo, en situación contraria, el subsistema resetea (inhibe) la neurona vencedora de la capa de salida y prueba con otra categoría. En el caso que no haya semejanza alguna se forma una nueva clase.

Así, la arquitectura mostrada en la Figura II.2.14, no resulta suficiente, por lo que debe agregarse 2 neuronas adicionales que cumplirán con la función del Subsistema de Atención (nodo G) y el de Orientación (nodo R). La Figura II.2.16 lo presenta.

En dicho esquema, el nodo R recibe las salidas de las neuronas de entrada con peso -1 junto con el valor del umbral para que pueda realizar las comparaciones pertinentes, de tal manera que, si se presenta el caso en el que no exista semejanza alguna, dicho nodo activará la salida a través de una conexión de inhibición con peso negativo de gran valor

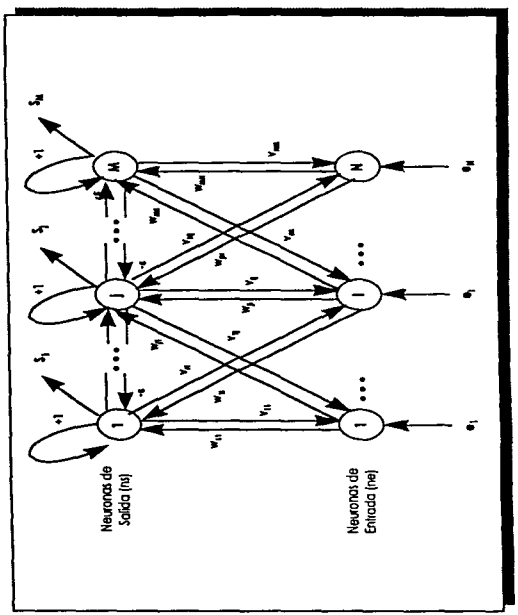

Figura II.2.14 Arquitectura de una Red Neuronal ART.

Figura II.2.15 Representación funcional de una Red ART.

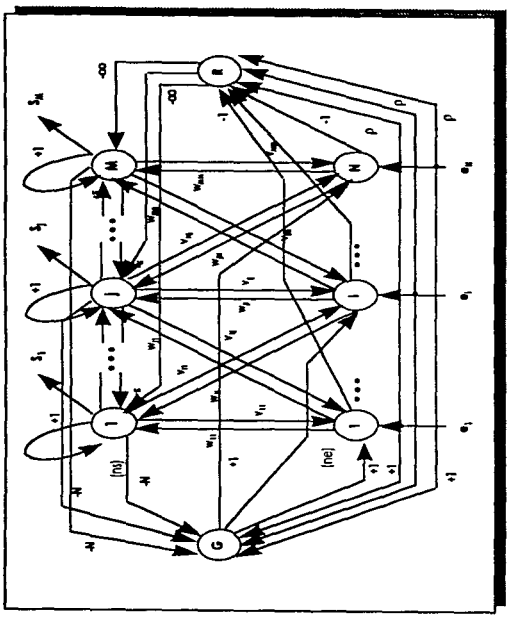

Figura II.2.16 Arquitectura completa de una Red Neuronal ART.

(-∞) y con ello anulará el efecto de la neurona que no representa la categoría para el patrón de entrada.

Por su parte, el nodo G permite realizar el Control de Ganancia de Atención, pues su papel es recibir los valores de las neuronas de la capa de salida con peso -N, así como los del patrón de entrada con valor 1, generando posteriormente, a través de una función de transferencia, (la función escalón) una salida que se aplicará a las unidades de la capa de entrada de la Red para que, de esta manera, ayuden a la activación de las neuronas.

2) Funcionamiento

El funcionamiento de un modelo ART2 puede describirse a través de un algoritmo como el que se muestra a continuación. Cabe mencionar que en algunos aspectos es parecido a la Red ART, pero en otros si es diferente[10] .

Paso 1. Se presenta el patrón de entrada denominado $E_k = (e_1^{(k)},....,e_N^{(k)})$ a la Red, en cuyo caso los valores $e_i^{(k)}$ son de tipo real.

Paso 2. Cada neurona (ne_i) que pertenece a la capa de entrada recibe el valor $e_i^{(k)}$ correspondiente y lo envía a todas las unidades del nivel de salida, esto a través de las conexiones w_{ij}.

Paso 3. Cada neurona (ns_j) de la capa de salida compite con las demás de su nivel hasta quedar solamente activa una de ellas. Así pues, se dice que la neurona vencedora será aquella j* que registre una mínima diferencia (distancia euclideana) entre el patrón de entrada y los pesos de las conexiones entre esta neurona (vencedora), respecto a las demás, lo cual se puede expresar de la siguiente forma:

$$S_{nj} = \begin{cases} 1 & MIN \|E_k - W_{ij}\| = MIN\left(\sum_{i=1}^{N} \|e_i^{(k)} - w_{ij}\|\right) \\ 0 & \text{Resto} \end{cases}$$

[10] Si se desea mayor información a cerca del funcionamiento del modelo ART (o ART1) puede consultarse el material que se presenta en las referencias al final del trabajo.

Paso 4. La neurona vencedora (n_{s^*}) envía su salida (es decir, 1) a través de las conexiones hacia atrás (v_{i^*}). De tal manera que cada unidad i-ésima de la capa de entrada recibe el valor:

$$x_i = \sum_{j=1}^{M} v_{ij} S_{nsj} = v_{ij} = w_{j^*i} \qquad \text{porque} \quad S_{nsj} = \begin{cases} 1 & j = j^* \\ 0 & j \neq j^* \end{cases}$$

Paso 5. Ahora se compara el patrón de entrada $E_k = (e_1^{(k)},....e_N^{(k)})$ con el representante de la j-ésima categoría (es decir, $X = (x_1,....,x_N) = W_{j^*} = V_{j^*}$), así, la relación de semejanza que se emplea es la siguiente:

$$\text{Relación de Semejanza} = \|E_k - X\| = \|E_k - V_j\| = \sum_{i=1}^{N} |e_i^{(k)} - w_{j^*i}|$$

Paso 6. Se compara la relación de semejanza con el parámetro de vigilancia (o umbral) establecido por el usuario, por lo que, si se cumple:

$$\| E_k - X \| > \rho$$

la neurona vencedora no representa la clase apropiada a la que pertenece el patrón E_k. así que se resetea o elimina del conjunto de posibles unidades vencedoras y se prueba una vez más (es decir se regresa al Paso 2). Si ya se ha probado con todas las salidas, entonces se agrega una nueva neurona a la capa de salida (ns_{M+1}) con pesos iguales a los componentes del patrón de entrada ($v_{iM+1} = w_{M+1i} = e_i^{(k)}$).

Paso 7. En el caso de que la semejanza sea igual o menor al umbral, se supondría que la neurona que se ha activado es la que representa la categoría más aproximada para el patrón de entrada E_k. así que ahora se procederá a ajustar los pesos v_{i^*} y w_{j^*i} para que el patrón de la nueva clase incorpore algunas características de la información que entró a la Red. esto de acuerdo a la expresión que a continuación se presenta:

$$w_{in}(t+1) = v_{ir}(t+1) = \frac{e_i^{(k)} + w_{in}(t)\, Num_r(t)}{Num_r(t) + 1}$$

donde $Num_r(t)$ es el número de patrones de entrada que se han considerado hasta el instante t como de la clase j".

3) Aprendizaje

En un modelo como el ART, el aprendizaje es de tipo ON LINE, debido a que no hay una distinción entre la fase de entrenamiento y la de funcionamiento, así, los pesos van variando durante la etapa de trabajo en el momento en que se aplica la información de entrada a la Red.

Por lo que, la Red utiliza un aprendizaje **no supervisado** de tipo **competitivo**, pues las neuronas de la capa de salida compiten para que sólo una sea la que permanezca activa ante una determinada unidad, de tal manera que, los pesos de las conexiones se ajustan en función del EP que haya salido ganador.

Asimismo, en este tipo de Redes puede darse además 2 tipos de aprendizaje, el **aprendizaje lento** y el aprendizaje **rápido**. El primer caso se da cuando el patrón de entrada se asocia a una de las categorías que ya existen, entonces se procede a ajustar los pesos para que el representante de esa clase se adapte a las características de la información que entró. El segundo caso ocurre cuando se ha incorporado una nueva categoría, por lo que ahora se busca representar la información de entrada, la cual será el prototipo de la nueva clase.

4) Aplicaciones del Modelo ART

Como ya se mencionó anteriormente, la Red ART está relacionada con tareas que involucran el Reconocimiento de Patrones, pues resulta útil en aquellas aplicaciones que necesitan del establecimiento automático de categorías para la clasificación de datos, en el caso de este trabajo de tipo real (ART2). Asimismo, también se ha empleado en áreas como la de Sistemas de Control y Diagnóstico Adaptativo.

Por otro lado, al presentar un funcionamiento autónomo y sin un supervisor, resulta de gran utilidad en la modelación de Procesos Biológicos y de esta manera, poder estudiar y predecir su comportamiento. Entre ellos se encuentra un sistema autónomo hecho por Carpenter, el cual fue diseñado para percibir y producir voz de manera muy análoga a como lo haría un humano. Así, el subsistema de percepción recibe la voz en forma de sonido, para establecer, a través de un modelo ART2, diferentes categorías fonéticas. Por su parte, el subsistema de producción, con la ayuda de un proceso de imitación, transforma

los códigos de voz oída en códigos de control de un motor para la generación de voz (para mayor explicación o ver más aplicaciones consultar el material que se presenta en las Referencias).

II.3 ALGORITMO INDUCTION DECISION TREE (ID3)

Por algunos años, mucha de la computación ha sido simplemente la colección, manipulación y transmisión de datos. Sin embargo, se prevé que para el futuro el interés principal de la computación se centrará también en la recolección, el manejo y transferencia pero ahora de conocimiento. De tal manera que, mucha de la investigación en Inteligencia Artificial, se ha enfocado a la Representación del Conocimiento de forma que pueda ser eficiente la colección, el almacenamiento y su utilización a través de una computadora.

Así, aquí se describirá un método que permite obtener conocimiento a través de un conjunto de datos, de tal manera que pueda ser representado por medio de series de sentencias "SI-ENTONCES", denominadas Reglas. Dicho algoritmo se conoce como ID3 (Induction Decision Tree) y constituye una descendencia del Sistema de Aprendizaje Conceptual (CLS, Concept Learning System) de Hunt en 1966.

Antecedentes: el Sistema de Aprendizaje Conceptual (CLS)

El Sistema de Aprendizaje Conceptual es un algoritmo que puede descubrir Reglas de Clasificación o un Árbol de Decisión, los cuales son útiles cuando se tiene una colección de ejemplos pertenecientes a 2 clases. Así, cuando ya se han obtenido las Reglas, éstas se emplean para clasificar una nueva muestra en una de las 2 clases.

Un ejemplo es clasificado iniciando en la raíz del árbol, para posteriormente realizar evaluaciones y ramificar hasta que un nodo se ha alcanzado, el cual determina si el ejemplo se encuentra o no.

El algoritmo consta de los siguientes pasos.

Paso 1. Si todos los ejemplos (denominados como un conjunto C) son positivos entonces crear un nodo y parar.
Si todos los ejemplos en C son negativos, entonces no crear nodo y parar.
En otro caso, seleccionar (usando algún criterio heurístico) un atributo A con valores $V_1, V_2, \ldots, V_n$ y crear el nodo de decisión.

Paso 2. Particionar los ejemplos entrenados en subconjuntos $C_1, C_2, \ldots, C_n$ de acuerdo a los valores de V.

Paso 3. Aplicar el algoritmo recursivamente a cada uno de los conjuntos C_i.

Induction Decision Tree (ID3)

El método ID3 fue desarrollado por J. Ross Quinlan en 1979, constituyendo un algoritmo Inductivo de Reglas de Propósito General comúnmente usado en la mayoría de los Sistemas Expertos Comerciales que emplean procedimientos de Inducción para generar Reglas.

A grandes rasgos, este algoritmo toma un conjunto de ejemplos del problema[11] y proporciona un árbol de decisión mínimo, utilizando herramientas heurísticas.

Asimismo, puede determinar si algún atributo es irrelevante para predecir el resultado final, sin embargo, no lo elimina, si no que simplemente no lo incorpora al árbol, guardándolo de alguna manera para un posible uso posterior, pues tal vez al agregar más información dicho factor pueda ser importante.

1) Árboles de Clasificación

Una estructura que ha sido ampliamente utilizada para representar el Conocimiento ha sido los Árboles de Clasificación o Árboles de Decisión como también se les llama.

Ahora bien, para ilustrar un Árbol de Clasificación vayámonos a un ejemplo, supóngase que usted desea invertir su dinero en una Compañía de la Industria de la Computación y para ello, acude con un amigo que es experto en ese campo para una consulta. Las siguientes líneas son un trozo de ese asesoramiento:

Experto: ¿La compañía es de hardware o software?

Usted: Software.

Experto: ¿Podrías decirme si el producto de la compañía es de tecnología nueva, de medio uso o viejo.

Usted: De medio uso.

Experto: Ese producto ¿tiene alguna competencia importante?

Usted: No.

Experto: Por lo que me has dicho, parece ser que el beneficio que proporcionaría a la empresa podría ser alto.

La Figura II.3.1 muestra esta misma conversación, pero a través de un Árbol de Clasificación, sólo que parcial, pues únicamente constituye una rama de todas las posibilidades que se pueden presentar, ya que dicho Árbol se va obteniendo por enumeración sucesiva de los valores asignados a cada atributo. Así, se fija como raíz el atributo A y se obtienen como vértices siguientes sus distintos valores: A_1, A_2, ..., A_n. Así, en cada vértice A_i, se obtienen los siguientes, a partir de los valores del siguiente atributo D: D_1, D_2, ... , D_i y así repetidamente (Ver Figura II.3.2).

[11] Algunos autores como Cuena, José (1986) llaman a estos ejemplos unidades de experiencia.

Figura II.3.1 Árbol de Clasificación que muestra los resultados de una consulta.

Figura II.3.2 Un posible Árbol de Clasificación, mostrándose en cada rama los valores de los atributos A, B, C, D y P, siendo el atributo A el nodo raíz.

Aunque los Árboles muestran las relaciones existentes entre sus distintos componentes, en un momento dado puede volverse muy difícil de manipular. Por lo que, existe una estructura que permite representar esa misma información sólo que de otra manera, dicha estructura se conoce como **Regla**. Así una Regla para la Figura II.3.1 sería la siguiente:

SI el tipo es software
Y la edad es medio uso
Y la competencia es no
ENTONCES el beneficio es alto

De tal manera que, dicha Regla está hecha para representar cada rama del Árbol, donde cada situación está representada por una palabra clave denominada **atributo**, a su vez existe una cuestión asociada con cada atributo, la cual puede ser almacenada con la regla en la forma de un **prompt**, como ejemplo se muestra el de a continuación:

prompt tipo
¿Es de hardware o software la compañía?

Así, la colección de Reglas es llamada **Base de Conocimientos**.

2) Adquisición del Conocimiento Bottleneck

Desafortunadamente, la complejidad que se presenta en los problemas del mundo real hace que sea difícil el poder diseñar Reglas con cierto detalle. Por ejemplo, en algunas áreas del problema la cantidad de información que se requiere para la solución es muy grande; en otras, el conocimiento no está bien definido para estructurarlo en forma de Reglas. Incluso hay casos en los que el problema es manejable, sin embargo, el número de expertos y el tiempo es reducido. Toda esta situación es a menudo conocida como Adquisición de Conocimiento Bottleneck (de cuello de botella). De ahí que se intente no simular como trabaja el cerebro humano, sino, proporcionar una manera de producir un Árbol de Clasificación a partir y directamente de un conjunto de ejemplos de un problema.

Para ilustrar como funciona el Algoritmo ID3 se trabajará con el problema de predecir si el beneficio proporcionado por una compañía dada se incrementará o disminuirá (mostrado al principio de esta sección). Supóngase que cuando le hace la consulta a su amigo, él le sugiere que recurra a revistas para familiarizarse con los incrementos y bajas de la industria, sin embargo parece ser muy difícil leer todo el material. Así que mejor se dispone a hacer una tabla que liste algunas de las compañías, algunas situaciones y si su beneficio se ha incrementado o no en los últimos años. Una muestra de esta tabla es la

Tabla II.3.1, donde las columnas representan los atributos Beneficio, Edad, Competencia y Tipo para cada valor almacenado en dicha tabla. Mientras que cada renglón es denominado un Ejemplo. Por otra parte, el atributo Beneficio es llamado **atributo de clase**, pues lo que se pretende es determinar la relación entre Beneficio (atributo de clase) y los valores de los otros atributos. Como se podrá observar, la tabla en sí no proporciona la información adecuada para predecir si el beneficio de la compañía se incrementará o disminuirá, por lo que es necesario usar esos ejemplos para construir un Árbol de Clasificación.

3) Construyendo el Árbol de Clasificación

Para construir el Árbol de Clasificación se selecciona uno de los atributos para que sea el nodo raíz. Una vez que se hizo esto, se particiona el conjunto de ejemplos en tablas más pequeñas, conteniendo cada una ejemplos con el valor del atributo seleccionado. Si por ejemplo se seleccionara el atributo "edad" como nodo raíz, la tabla se dividirá en 3 partes como lo muestra la Figura II.3.3. Como puede verse, cuando "edad" toma el valor de "viejo" el valor para el atributo de clase siempre es "bajo", mientras que cuando "edad" es "nuevo", el valor de clase es siempre "alto", por lo que, para estos dos casos no es necesario continuar la clasificación. Ahora bien, para el caso en que "edad" toma el valor de "medio uso", el valor para el atributo de clase toma 2 valores, por consiguiente, se requiere seleccionar un nuevo atributo y particionar de nuevo. En la Figura II.3.4 se muestran los resultados al seleccionar "competencia" como nuevo atributo, terminando así el proceso.

Una vez obtenido el Árbol de Clasificación puede continuarse con la producción de Reglas de la forma "si-entonces", siguiendo paso a paso cada rama que inicia del nodo raíz hasta el nodo terminal. Así, cada Regla va a ser una serie de condiciones conformándose de un atributo con su respectivo valor, seguido de una sencilla conclusión que contiene el atributo de clase y su correspondiente valor (Ver Figura II.3.5).

4) Entropía

El Árbol que se obtuvo en la Figura II.3.4 no es el único que podría construirse, pues si se eligiera algún otro atributo como nodo raíz, seguramente su estructura cambiaría y por consiguiente las Reglas que se produjeran también serían diferentes, de ahí el interés por definir un criterio que permita un orden de consulta y con ello producir un Árbol mínimo con la estructura más simple. Para ello, Quinlan propone como regla para la selección del atributo raíz aquel con mayor contenido de información respecto al conjunto de ejemplos, de tal manera que presenta la **entropía** como una medida de información. Aplicando este concepto al problema de la clasificación, se puede ver que un objeto puede ser

BENEFICIO	EDAD	COMPETENCIA	TIPO
Bajo	Viejo	No	Software
Bajo	Medio uso	Si	Software
Alto	Medio uso	No	Hardware
Bajo	Viejo	No	Hardware
Alto	Nuevo	No	Hardware
Alto	Nuevo	No	Software
Alto	Medio uso	No	Software
Alto	Nuevo	Si	Software
Bajo	Medio uso	Si	Hardware
Bajo	Viejo	Si	Software

Tabla II.3.1 Tabla que muestra el conjunto de Ejemplos para el problema en cuestion.

Figura II.3.3 Conjunto de Ejemplos para la tabla II.3.1 tomando como nodo raíz el atributo "edad".

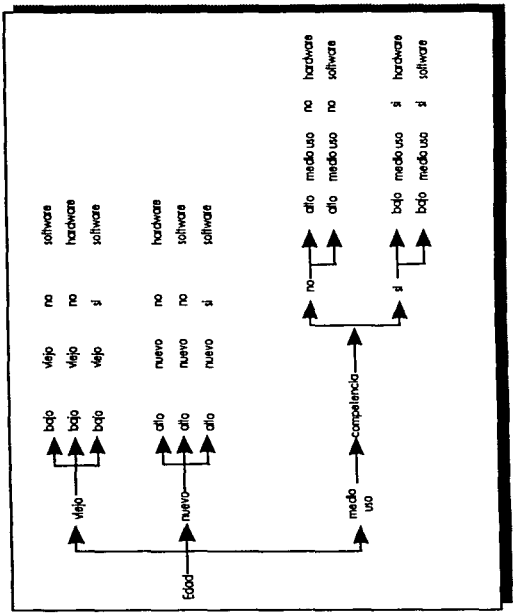

Figura II.3.4 Conjunto de Ejemplos para la Figura II.3.3 después de 2 particiones.

SI edad es viejo
ENTONCES beneficio es bajo

SI edad es nuevo
ENTONCES beneficio es alto

SI edad es medio uso
Y competencia es no
ENTONCES beneficio es alto

SI edad es medio uso
Y competencia es si
ENTONCES beneficio es bajo

Figura II.3.5 *Reglas producidas por el Árbol de Clasificación de la Figura II.3.4.*

clasificado en uno de varios grupos, formando clases, por lo que la entropía va a actuar como una medida de incertidumbre de la clasificación de un objeto, así, si la entropía es grande significará que dicha incertidumbre es alta y por lo tanto, no existe la seguridad de que ese objeto se halle en el grupo, por el contrario, si la medida es pequeña implicará más confianza en la agrupación.

De forma matemática, si un objeto puede ser clasificado en N diferentes clases denominadas c_1, c_2, ... c_N y además la probabilidad de que el objeto se encuentre en la clase i es $p(c_i)$, entonces la entropía de clasificación esta dada por:

$$H(C) = -\sum_{i=1}^{N} p(c_i) \log_2 p(c_i)$$

Aplicando la fórmula de la entropía para el conjunto de ejemplos mostrado en la Tabla II.3.1, se observa que existen 2 posibles valores para el atributo de clase, es decir "alto" y "bajo". Ahora bien, la probabilidad (la frecuencia de ocurrencia) de la clase con el valor de "alto" es 5 sobre el total de la muestra que es de 10, es decir 5/10, mientras que para el caso de "bajo", también es de 5/10, por lo que la entropía de la clasificación para el total del conjunto será:

$$\begin{aligned} H(C) &= -p(\text{alto}) \log_2 p(\text{alto}) - p(\text{bajo}) \log_2 p(\text{bajo}) \\ &= -5/10 \log_2 (5/10) - 5/10 \log_2 (5/10) \\ &= 1.00 \end{aligned}$$

Aunque este número representa la incertidumbre referente al beneficio basándose en los datos de la Tabla II.3.1, no dice nada respecto a la información contenida en los atributos individuales, por lo que se realizarán otros cálculos en el siguiente apartado.

5) Entropía de Clasificación de un Atributo

La entropía de Clasificación de un Atributo en específico representa la cantidad de incertidumbre de una respuesta, así, el proceso consiste en obtener de manera similar la entropía de clasificación para cada atributo y seleccionar aquél con mínimo valor.

Como primer paso para el cálculo de la entropía, se selecciona un atributo simbolizándose como H(C/A), entonces se particiona la tabla en otras más pequeñas como lo muestra la Tabla II.3.2, donde se ha seleccionado el atributo "competencia". Así, la entropía de cada subtabla $H(C/a_i)$ se determina a través de la siguiente expresión:

COMPETENCIA	BENEFICIO
No	Bajo
No	Alto
No	Bajo
No	Alto
No	Alto
No	Alto
Si	Bajo
Si	Alto
Si	Bajo
Si	Bajo

Tabla II.3.2 Parte del conjunto de ejemplos particionando en el atributo "competencia".

$$H(C/a_j) = -\sum_{i=1}^{N} p(c_i/a_j) \log_2 p(c_i/a_j)$$

donde la función $p(c_i/a_j)$ es la probabilidad de que el valor de la clase sea c_i cuando el atributo toma el j-ésimo valor.

Realizando los cálculos se obtiene:

H(C/competencia=no) = - { [p(alto/competencia=no)log$_2$(p(alto/competencia=no))]
+ [p(bajo/competencia=no)log$_2$(p(bajo/competencia=no))] }
= - [4/6 log$_2$(4/6) + 2/6 log$_2$(2/6)] = 0.918
H(C/competencia=si) = - { [p(alto/competencia=si)log$_2$(p(alto/competencia=si))]
+ [p(bajo/competencia=si)log$_2$(p(bajo/competencia=si))] }
= - [1/4 log$_2$(1/4) + 3/4 log$_2$(3/4)] = 0.811

La expresión "p(alto/competencia=no)" representa la probabilidad de que el valor de clase sea "alto" dado que el valor del atributo "competencia" es "no", esto es, el número de veces que "alto" aparece en el renglón de "competencia" cuando toma el valor de "no", dividido entre el número de casos en los que "competencia" es igual a "no".

Ahora bien, para encontrar la entropía de toda la tabla (después de la partición) H(C/competencia), se toma la suma de la entropía de cada uno de los valores de los atributos multiplicado por la probabilidad de que el valor aparezca en la tabla. Entonces la entropía de clasificación después de elegir un atributo particular, h(C/A) es el promedio de la entropía para cada valor a_j del atributo. Matemáticamente se expresa de la siguiente manera:

$$H(C/A) = \sum_{j=1}^{M} p(a_j) H(A/a_j)$$

donde M es el número total de valores para el atributo A.

Para el caso de este ejemplo se obtiene:

H(C/competencia) = (6/10)(0.918) + (4/10)(0.811) = 0.8752

Si se realizan los mismos cálculos para los demás atributos se obtiene que:

H(C/edad) = 0.4
H(C/tipo) = 1.0

Y como el valor H(C/edad) proporciona la entropía más pequeña, entonces, el atributo "edad" es el mejor para ser seleccionado como nodo raíz.

Una vez que se encontró el nodo que funcionará como raíz, se puede continuar con el proceso que se trató anteriormente, particionando y obteniendo entropías en el caso en el que se requiera de seleccionar algún nodo.

CAPITULO III

SISTEMAS EXPERTOS Y REPRESENTACIÓN DEL CONOCIMIENTO

CAPÍTULO III
SISTEMAS EXPERTOS Y REPRESENTACIÓN DEL CONOCIMIENTO

III.1 SISTEMAS EXPERTOS

Concepto y Características de un Sistema Experto

Los Sistemas Expertos constituyen una aplicación de la Inteligencia Artificial (IA) que se inició alrededor de 1950, pero que emergieron como un producto comercial en la década de los setenta, pues su aparición inaugural fue meramente a nivel de investigación universitaria, para convertirse posteriormente en una de las más importantes innovaciones de la IA desde que fueron mostrados como productos comerciales exitosos.

Existen muchas definiciones y conceptos que describen lo que es un Sistema Experto, sin embargo, todas ellas coinciden en señalarlo como un nuevo tipo de Software, que de alguna manera pretende manipular conocimiento[1] para resolver problemas en un campo específico y delimitado (Dominio del Conocimiento), situación que normalmente requiere de la experiencia humana, de ahí que su objetivo sea modelar el comportamiento humano para resolver dicha problemática cuando no se dispone de algún algoritmo. Además se les califica como Sistemas Basados en Conocimientos, precisamente porque su funcionamiento está determinado por la utilización de procedimientos y técnicas heurísticas similares a las que utiliza cualquier persona.

Así, el modo en que opera un Sistema Experto, puede verse como un medio de alcanzar niveles de competencia, similares a los poseídos por los seres humanos en campos específicos del saber. En este sentido, se dice que los Sistemas Expertos ofrecen cualidades muy semejantes a las de los individuos y, por ende, pueden llegar a reemplazar a una persona experta. En otros casos, simplemente son utilizados como herramienta para mejorar la comunicación entre doctos humanos.

De tal suerte que, el conocimiento de un Sistema Experto se obtiene precisamente de un experto, que no tiene que ser directamente un humano, sino que pueden ser otras fuentes como textos, periódicos, artículos y bases de datos, sin embargo, esta labor puede ser realmente compleja, pues una cosa es dominar cierto campo del saber y otra muy

---

[1] Para algunos investigadores, los Sistemas Expertos son, de cierta forma, sistemas optimizados de búsqueda de información, lo cual es cierto, pues la competencia del experto humano está basada, en alguna medida, en su gran capacidad para buscar y encontrar información. Simons, G. L. (1987) Introducción a la Inteligencia Artificial, Pág. 175.

diferente es llegar a estructurar el conocimiento en proposiciones coherentes y, que el Sistema lo utilice en el proceso de inferencia o razonamiento.

Una vez que el experto ya transmitió su conocimiento al Sistema, este saber puede estructurarse en forma de árbol, de tal manera que los nodos representarían información constituida en axiomas o reglas. En la parte superior de la disposición se encontraría la(s) hipótesis u objetivo principal a seguir, incluso varios planteamientos secundarios, donde cada uno contempla una circunstancia de la situación.

Los Sistemas Expertos presentan ciertas características que los hacen diferir notablemente de los Sistemas Convencionales, entre ellas se encuentran las siguientes:

a) Poseen una facilidad de **explicación**, es decir, tienen la capacidad de almacenar hechos en su memoria para con ello, generar dichas exposiciones acerca de cómo una conclusión particular fue buscada, y porque la información requerida es necesaria durante la consulta. Esto es importante, puesto que brinda al usuario la oportunidad de comprender el razonamiento del Sistema, así como permitirle al experto corregirlo y validarlo. Asimismo, posee la facultad para ofrecer un consejo inteligente o tomar una decisión.

b) La mayoría de los Sistemas Expertos manejan cierto rango de **incertidumbre**, de la misma manera que un hombre no tiene la completa certeza en sus respuestas. Como un docto humano, un Sistema Experto posee un mecanismo para manipular tal fluctuación y con ello, permitirle al usuario conocer el grado de imprecisión que se está generando cada vez que utilice el Sistema. Así, la Estadística Bayesiana y la Teoría de los Conjuntos Difusos son las técnicas más comunes para el manejo de incertidumbre en Sistemas Expertos. Una muestra de ello, es el caso del MYCIN, un Sistema Experto que incorpora un modelo de razonamiento, el cual emplea ecuaciones que realizan funciones similares a las expresiones utilizadas en la Teoría de los Conjuntos Difusos. Por ejemplo, la forma de un objeto se puede describir como rectangular (0.6), oval (0.2) y cuadrada (0.0). Es decir, los valores comprendidos entre 0 y 1 indican el factor certidumbre asignado a cada cualidad, situación que no ocurre en una base de datos normal, pues una característica puede ser imprecisa o tener asignados varios valores simultáneamente.

c) Otra característica de los Sistemas Expertos que los hacen ser diferentes a otros tipos de software, es que el conocimiento es codificado y mantenido como una entidad separada de la estructura de control del programa (Motor de Inferencia). Esto permite el refinamiento (agregar y modificar) de la Base de Conocimientos (hechos y heurísticas), y con ello, una mayor facilidad en la ejecución.

Todo esto implica que los elementos de la Base de Conocimientos también son independientes los unos de los otros (propiedad de granularidad o

modularidad del conocimiento), pero que se ponen en acción dinámica a través de un procedimiento de resolución, el cual es independiente de la naturaleza del conocimiento (Motor de Inferencia).

Así, a diferencia de la programación clásica, en este nuevo tipo de Software, las modificaciones hechas a cualesquiera de sus elementos no tiene algún efecto catastrófico en el desarrollo del programa.

Por otra parte, es posible tener varias Bases de Conocimientos con las mismas estructuras de control y así producir diferentes Sistemas Expertos.

d) Los Sistemas Expertos utilizan una representación simbólica para el conocimiento (reglas, redes, frames), por tanto, emplean un procesamiento de tipo simbólico más que numérico, esto se refiere básicamente al procedimiento a través de listas y símbolos, sin embargo, esto no implica que el Sistema no pueda ejecutar procesos numéricos, sino que tal vez para estos casos pueda resultar más fácil emplear lenguajes de programación convencionales. Por ello, se hace énfasis en la programación basada en reglas, esto es, la programación lógica en lenguajes relacionales como PROLOG que permiten una amplia gama de aplicaciones como consulta, tratamiento de datos, sistemas de control en línea y la posibilidad de razonamiento.

e) Otra peculiaridad de los Sistemas Expertos, es que utilizan métodos empíricos, los cuales se apoyan en conocimientos heurísticos que permiten encontrar la mejor solución y no la óptima.

f) Asimismo, cabe resaltar que un Sistema Experto es especialista en un ámbito y no en una tarea.

Todas las características mostradas son numerosas y por ello, no implica que un Sistema Experto deba cumplir con todas ellas, sin embargo, sirve para centrar la noción de lo que es un Sistema Experto.

Así, los Sistemas Expertos han mostrado ser eficientes en una gran cantidad de problemas de diferentes áreas, tales como Química, Biología, Ingeniería, Manufactura, Aeroespacio, Operaciones Militares, Finanzas, Meteorología, Geología, Geofísica y muchas más, los cuales requieren del tipo de inteligencia que posee un experto humano.

Antecedentes Históricos

Los primeros trabajos en Sistemas Expertos se inician alrededor de la década de 1950 con el grupo Rand-Carniege formado por Newell, Shaw y Simon, quienes desarrollan el GPS (General Problem Solver) que tiene por objeto resolver problemas de lógica elemental, ajedrez y álgebra superior.

Por otro lado, a finales de los cincuenta y principios de los sesenta, el grupo MIT que estaba constituido por Minsky y McCarthy crean los fundamentos de los Sistemas Expertos.

Cabe mencionar que los primeros Sistemas Expertos surgieron de las investigaciones realizadas en laboratorios de algunas Universidades. De tal manera que, fueron desarrollados únicamente como Resolvedores de Problemas para situaciones especiales, los cuales enfatizaban el uso de Conocimiento más que de Algoritmos y Métodos Generales de Búsqueda. Así, este enfoque marcó una significativa apertura de las Arquitecturas y Sistemas Convencionales que existían en Inteligencia Artificial[2], permitiendo de alguna forma, el crecimiento de una nueva clase de Sistemas Útiles, así como también el diseño de Sistemas Específicos.

De tal suerte que, el primer Sistema Experto que se realizó fue el DENDRAL, creado por Lederberg, Buchanan y Feigenbaum en la Universidad de Stanford en la época de los sesenta. Este Sistema era capaz de determinar la estructura de compuestos químicos (más adelante se detallará un poco). Utilizaba un conocimiento heurístico obtenido de químicos expertos. Así, durante el examen del DENDRAL se descubrió un número de estructuras que aún eran desconocidas para los expertos.

Ya con el empleo del DENDRAL, los investigadores obtuvieron más experiencia, sin embargo, se les presentó un problema, lo cual permitió el desarrollo del Meta-DENDRAL, un componente de aprendizaje para dicho Sistema Experto, que era capaz de aprender reglas a partir de ciertos ejemplos.

Después de terminar con el DENDRAL se inició el desarrollo del Sistema Experto MYCIN en la misma Universidad. El MYCIN es un Sistema que permite el diagnóstico de enfermedades en la sangre para posteriormente determinar una lista de recomendaciones para terapia. Como parte del Proyecto de Programación Heurística en Stanford, se crearon diferentes proyectos relacionados directamente con el MYCIN, incluyendo el componente de adquisición de conocimiento llamado THEIRESIUS, el elemento GUIDON así como también el factor Shell conocido como EMYCIN. Este último fue empleado para construir otros sistemas de Diagnóstico como el PUFF que diagnostica enfermedades en los pulmones. Por lo que, el EMYCIN se convirtió en el modelo de diseño para diferentes Sistemas Expertos comerciales.

Por otra parte, la ejecución del MYCIN mejoró notablemente cuando se le agregó conocimiento, ya que los exámenes indican que dicho Sistema llega a superar a algunos médicos experimentados, pues la Base de Conocimientos inicial contenía aproximadamente 200 reglas incrementándose gradualmente a más de 600 en los ochenta.

[2] Se utilizaban técnicas como Ascención de Colinas (Hill-Climbing), o Análisis Means-Ends. Op. Cit.

Otros Sistemas Expertos contemporáneos son el PROSPECTOR, un Sistema que asiste a geólogos en el descubrimiento de depósitos minerales; el R1, que es usado por la Corporación de Equipo Digital para seleccionar y configurar componentes de Sistemas Computacionales complejos. Asimismo, se encuentra el SAINT creado por Sagle para Integración Simbólica y el STUDENT hecho por Bobrow para resolver problemas de álgebra superior.

En los años setenta, el interés industrial por desarrollar Sistemas Expertos se incrementó, pues se vio su aplicación en áreas tales como diagnóstico, percepción, instrucción, aprendizaje, programación, y en reconocimiento de patrones e imágenes.

Asimismo, existen muchas empresas que han mostrado su inclinación por desarrollar Sistemas Expertos, entre ellas se encuentran, Digital Equipment, Texas Instruments, Xerox, Schlumberger, Hewlett-Packard, General Motors, IBM, Teknowledge, el Grupo Carnegie, Inference Corporation e Intellicorp. En lo que respecta a Universidades, se halla Stanford, MIT, Carnegie-Mellon y Rutgers.

Arquitectura de un Sistema Experto

La Arquitectura de un Sistema Experto consta de 5 bloques funcionales (ver Figura III.1.1), sin embargo, la estructura de estos componentes va a variar dependiendo del tipo de aplicación.

1) Base de Conocimientos

La Base de Conocimientos es la parte más importante de un Sistema Experto, pues es la que guarda todo su saber, de tal manera que, toda esta experiencia se halla expresada en hechos, reglas y procedimientos del dominio de la aplicación que son importantes para la solución del problema.

Los hechos son de la forma: 'El velero "Mary" tiene una longitud de 6 metros'. Así, puede emplearse una orientación conforme a objetos, para la representación de este conocimiento, por lo que, un objeto de una Base de Conocimientos podría ser "barco", "barco a motor", o "barco a vela", los cuales se encuentran relacionados de una manera tal que, un "barco a vela", por ejemplo, posea todas las cualidades de un "barco", además de todas las características particulares que posee un "barco a vela". Es decir, "hereda" las cualidades de "barco", haciendo falta sólo agregar sus rasgos específicos.

Este tipo de programación es a la que se le conoce como Programación Orientada al Objeto.

La clasificación en grupos respecto a las características, procedimientos y relaciones referente a algún objeto, a través de las técnicas de programación, puede

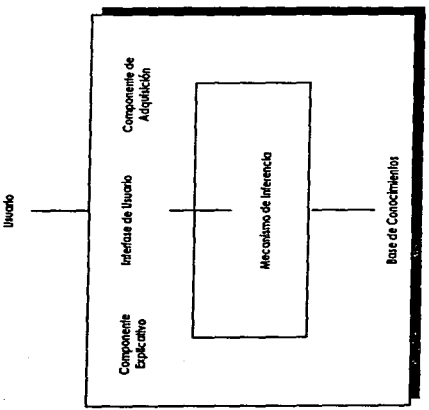

Figura III.1.1 Componentes de un Sistema Experto.

variar dependiendo de la aplicación. Asimismo, también el grado de orientación de esos entes estará supeditada por el planteamiento del problema.

También la Base de Conocimientos alberga Reglas, las cuales tienen la siguiente estructura:

SI premisas **ENTONCES** conclusión y/o acción.

En la parte que dice premisas se encuentran las vinculaciones lógicas que existen por las cualidades de los objetos. Mientras que en la sección de conclusión se agregan nuevos hechos y cualidades a la Base de Conocimientos y/o se ejecutan acciones. A todo esto se le denomina Programación Orientada a las Reglas.

Por otro lado, para la creación de una Base de Conocimientos existen aspectos que deben ser tomados en cuenta, los cuales permitirán una buena consistencia en ésta:

a) Definición de los objetos.

b) Relaciones entre los objetos.

c) Formulación y Procesamiento de las Reglas.

2) Mecanismo (Máquina) de Inferencia

El Mecanismo de Inferencia constituye la Unidad Lógica o Estructura de Control con la que se extraen Conclusiones en base a la manipulación de Conocimiento almacenado, esto a través de un método fijo de solución, el cual está configurado de una manera tal, que pretende imitar el procedimiento de un experto humano para solucionar problemas, si son solubles en este entorno.

Por lo que, la producción de una Conclusión se lleva a cabo cuando se aplican las Reglas sobre los hechos presentes, por ejemplo, considérese la siguiente Regla:

SI p y q **entonces** r

y los hechos p y q

Como se observa, p y q son justamente los hechos que se hacen referencia en la cláusula "si" de la regla, es decir, son las condiciones para que se pueda aplicar la regla. De este modo, aplicar la regla es deducir a partir de los actos p y q la acción r. Así cuando se da esta aplicación se dice que la Regla se dispara.

Por otra parte, un Mecanismo tiene que cumplir con varias funciones entre las cuales se mencionarán las siguientes:

a) Determinación de las acciones que se ejecutarán, así como el orden y la forma en que lo harán entre las diferentes partes del Sistema Experto.
b) Especificar las reglas que van a procesarse, asimismo cómo y cuándo se desarrollarán estas mismas.
c) Control de diálogo con el usuario.

La decisión en lo que respecta a los mecanismos para procesar las reglas, esto es, las estrategias de búsqueda son de mucha importancia para la efectividad del Sistema Experto, además de que varían de un contexto a otro, pues todos los problemas no son iguales, en tal situación es necesario que el Mecanismo de Inferencia se encuentre debidamente "adaptado" para la situación que se esté tratando.

Así, existen 2 principales métodos que se incorporan al Mecanismo de Inferencia para poder realizar la búsqueda de una manera más eficiente, estos son conocidos como **Encadenamiento hacia Adelante** (Forward Chaining) y **Encadenamiento hacia Atrás** (Backward Chaining), los cuales se verán más a detalle en una sección posterior, pero, por lo pronto se puede decir que el Encadenamiento hacia Adelante o Razonamiento Dirigido por los Datos (Data-Driven Reasoning), como también se le conoce, es utilizado para resolver problemas cuando los datos o ideas básicas, son usados como punto de partida. Con este método, el Sistema no inicia con un objetivo particular definido, sino que trabaja a través de los hechos para llegar a una conclusión (o varias) u objetivos. Por lo que, este tipo de Encadenamiento ha sido comúnmente empleado en Sistemas Expertos para el Análisis de Datos, Diseño y Formulación de Conceptos.

En lo que respecta al Encadenamiento hacia Atrás o Razonamiento Dirigido por el Objetivo (Goal-Directed Reasoning), también denominado así, éste constituye otra técnica de inferencia en la que, partiendo de un objetivo o hipótesis y trabajando hacia atrás, se pretende llegar a saber si esa conclusión es verdadera. De este modo, los Sistemas Expertos que utilizan este tipo de Encadenamiento son normalmente usados en Diagnóstico y Planeación.

3) Componente Explicativo (Interfaz Explicativa)

La existencia de un Componente Explicativo brinda la oportunidad, al usuario, de que el Sistema Experto pueda explicar como es que obtuvo sus conclusiones o, la razón por la cual es que hace determinadas preguntas, y con ello, exhibir su proceso de razonamiento para que el beneficiario del Sistema, lo examine.

Se insiste mucho en resaltar la importancia de que un Sistema Experto cuente con una Interfaz Explicativa, sin embargo, su creación es muy difícil y hasta el momento, no se ha llegado a poseer una que cumpla con todos los requisitos para que se considere como un buen Componente Explicativo, pues el objetivo es encontrar la forma de representar en un texto lo suficiente inteligible, las relaciones encontradas. Así, los Componentes Explicativos existentes, pueden ser suficientes para el desarrollador del Sistema, ya que está muy familiarizado con el entorno del procesamiento de datos, y a veces también el experto, pero para el usuario, que a menudo desconoce las sutilezas de dicho proceso, los Módulos Explicativos que prevalecen son todavía demasiado insatisfactorios.

4) Interfase de Usuario

La Interfase de Usuario constituye el medio por el cual el usuario va a poder comunicarse con el Sistema Experto. Por lo que, es importante tener presente ciertas situaciones durante su construcción (como los que se muestran a continuación), para que con ello, la ejecución del Sistema sea eficiente.

a) El aprendizaje respecto a su manejo no debe llevar mucho tiempo.
b) Debe prevenirse que en la introducción de los datos, estos no ingresen de manera errónea.
c) Los resultados deben aparecen en una forma clara y sencilla para el usuario.
d) Las preguntas, así como las explicaciones por parte del Sistema Experto deben ser comprensibles para la persona que lo utilice.

5) Componente de Adquisición

Este Componente de Adquisición tiene como finalidad el transferir el Conocimiento de uno o varios expertos a la Base de Conocimientos del Sistema. Para ello, debe contar con ciertas características como las que se enuncian a continuación:

a) El Conocimiento, esto es, las Reglas, Hechos o las Relaciones entre los objetos, deben poderse introducir de una manera sencilla y sin complicaciones.
b) Debe ser posible representar de una forma clara toda la información en la Base de Conocimientos.
c) Poderse comprobar automáticamente la Sintaxis empleada.
d) Contar con la factibilidad de poder accesar al código en el que fue hecho el Sistema Experto.

Estrategias de Inferencia

Como ya se ha mencionado anteriormente, debe formularse, de una manera explícita, un mecanismo que permita procesar el Conocimiento que se halla almacenado, de tal manera que, se evalúen las reglas y el conocimiento en hechos.

Así, existen 2 principales formas de apreciación para las reglas:

1) Encadenamiento hacia Adelante (Forward Chaining, Forward Reasoning).

2) Encadenamiento hacia Atrás (Backward Chaining, Backward Reasoning)

1) Encadenamiento hacia Adelante

El Encadenamiento hacia Adelante también es definido como Inferencia controlada por los Datos o método "if-added", debido a que el Motor de Inferencia emplea la información que le proporciona el usuario para con ello, desplazarse a través de la red de Y y O lógicos hasta que alcanza un punto terminal, es decir cuando ha llegado al objetivo. Así, este proceso se realizará tantas veces como sea necesario hasta obtener la meta o hasta que no quede ninguna regla más que pueda ser "disparada" (aplicada).

De este modo, un Mecanismo de Inferencia con este tipo de procesamiento, inicia con alguna información preliminar y trata de encontrar un objeto que esté en la Base de Conocimientos, el cual encaje con dicha referencia (ver Figura III.1.2).

Para comprender mejor el proceso de esta Estrategia de Inferencia, considérese que se cuenta con las siguientes premisas (conocimiento inicial):

El Producto es nuevo y
no hay Competencia

es decir,

SI el Producto es nuevo
Y la Competencia es no
ENTONCES .....

Como puede observarse, la fracción en puntos suspensivos representa la parte que deseamos conocer (objetivo), por lo que el Sistema Experto, específicamente la Máquina de Inferencia con Encadenamiento hacia Adelante, lo primero que haría sería comparar los hechos con las partes izquierdas de todas las Reglas de la Base de Conocimientos y así, donde hiciera un "match" (apareamiento, emparejamiento) significaría que las premisas han quedado satisfechas por esa regla.

Sin embargo, puede darse el caso de que no sea una sola regla la que satisfaga los hechos, sino un conjunto (conjunto conflicto), entonces el siguiente paso será decidir

Figura III.1.2 Forma Gráfica del Encadenamiento hacia Adelante.

que regla será aplicada (proceso de resolución del conflicto). Para ello existen varias formas de hacerlo (por ejemplo un sort), sólo que éstas van a depender de la naturaleza del problema, habilidad y experiencia de los diseñadores del Sistema Experto.

2) Encadenamiento hacia Atrás

El Encadenamiento hacia Atrás, también es conocido como inferencia controlada por el objetivo o método "if-needed" . Así, dicho procedimiento es el opuesto al Encadenamiento hacia Adelante, es decir, se parte de un objetivo o hipótesis y se va solicitando más información para confirmarla o negarla (ver Figura III.1.3).

Para entender un poco más como funciona este tipo de Encadenamiento, imagínese que su computadora deja de trabajar espontáneamente, por lo que, su primera hipótesis será que el motivo de tal problema se debe a una falta de corriente, así que, para comprobarlo escucha el ventilador, sin embargo éste funciona correctamente, de manera que, esta no es la razón, por ello, el supuesto en cuestión se rechazará. Entonces formulará una segunda conjetura, la cual consistirá ahora en que, la actividad del ordenador se ha deteriorado debido a una falla del software, de tal suerte que, para confirmarla o rechazarla resetea la máquina, no obstante, ésta se inicializa bien, por consiguiente, la segunda hipótesis fue la cierta.

Lo anterior constituye una muestra del funcionamiento de tal método, sin embargo, puede que aún no esté claro del todo, así que se procederá a presentar un nuevo ejemplo para ello. Supóngase que se cuenta con la Base de Conocimientos que se exhibe a continuación:

Manzana
- crece en los árboles
- redonda
- el color es rojo o amarillo
- no crece muy al sur

Uva
- crece en las vides
- de tamaño pequeño
- varios colores
- las vides no tienen espinas

Naranja
- crece en los árboles
- redonda
- el color es naranja
- no puede crecer en el norte

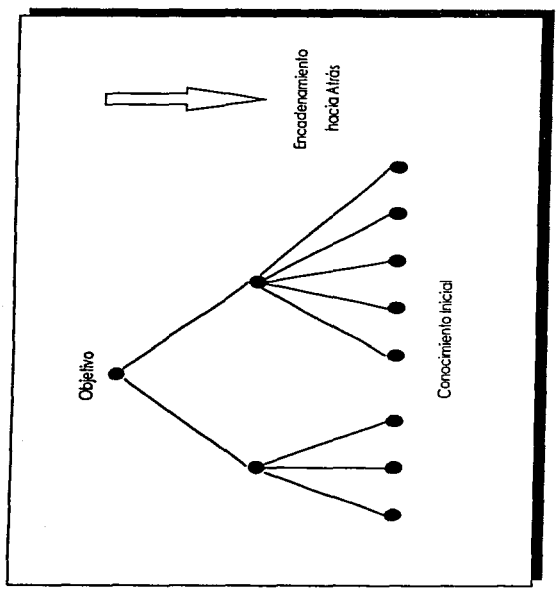

Figura III.1.3 Forma Gráfica del Encadenamiento hacia Atrás.

Ahora bien, si la fruta que se está hipotetizando es una manzana, y se aplica el método de Encadenamiento hacia Atrás, se produce el diagrama mostrado en la Figura III.1.4.

Ya se mostraron los 2 tipos principales de Estrategias de Inferencia, sin embargo, puede ser que en un Sistema Experto, el Mecanismo de Inferencia no se encuentre en una forma pura, es decir, Encadenamiento hacia Adelante o Encadenamiento hacia Atrás, sino como una combinación en la que el diseñador va adaptando los 2 mecanismos de acuerdo a la naturaleza del problema. Así, una forma de ello es la que se muestra a continuación a través del siguiente ejemplo:

Supóngase que se cuenta con una pequeña Base de Conocimientos como la que se muestra:

Objeto	Atributos
1	A, B, C
2	A, B, Y
3	B, X
4	A, B, D

En un nivel muy simple, el Motor de Inferencia iniciaría suponiendo que el objeto 1 es el objetivo, por lo que tratará de confirmarlo haciendo preguntas respecto a la posesión de los atributos. En el caso que dichas características sean afirmativas, el Mecanismo manifestará que el elemento en cuestión es la respuesta. De lo contrario, se procederá con el siguiente objeto (el 2) e investigará sobre sus atributos, continuando así hasta que encuentra el ente adecuado o bien, cuando ya no existan más objetos que analizar. Así, este proceso puede representarse a través del siguiente diálogo:

```
experto     ¿tiene A?
usuario     sí.
experto     ¿tiene B?
usuario     sí.
experto     ¿tiene C?
usuario     no.              /* rechazar 1 */
experto     ¿tiene A?        /* redundante */
usuario     sí.
experto     ¿tiene B?        /* redundante */
usuario     sí.
experto     ¿tiene Y?
```

Figura III.1.4 Encadenamiento hacia Atrás hasta el objeto manzana.

```
usuario     no.          /* rechazar 2 */
experto     ¿tiene B?    /* redundante */
usuario     si.
experto     ¿tiene X?    /* innecesario */
usuario     no.          /* rechazar 3 */
experto     ¿tiene A?    /* redundante */
usuario     si.
experto     ¿tiene B?    /* redundante */
usuario     si.
experto     ¿tiene D?
usuario     si.
experto     es el 4.     /* encontrado */
```

Como puede observarse, un Sistema Experto que trabaje de esta manera no sólo sería tedioso de usar, si no que no valdría la pena construirlo, pues de lo que se trata es de que actúe como un verdadero experto y éste no haría tantas preguntas redundantes e innecesarias, es decir, hace la misma cuestión varias veces cuando se supondría que el Sistema ya debería haber memorizado las respuestas (por ejemplo en el objeto 2), no teniendo caso volver a interrogar; por otra parte, se presentà el caso en el que no tendría que preguntar por el atributo X en el objeto 3, pues desde un principio se percibe la falta de la característica A. Así, sería deseable entonces que dicho Sistema produjera el siguiente diálogo:

```
experto     ¿tiene A?
usuario     si.
Experto     ¿tiene B?
usuario     si.
Experto     ¿tiene C?
usuario     no.          /* rechazar 1 */
experto     ¿tiene Y?
Usuario     no.          /* rechazar 2 */
                         /* rechazar 3 */
experto     ¿tiene D?
usuario     si.
experto     es el 4.     /* encontrado */
```

Metodología en el desarrollo de un Sistema Experto

El desarrollo de un Sistema Experto con posibilidades de éxito debe programarse muy cuidadosamente, siguiendo un esquema en el que pueden figurar las siguientes etapas:

1) Elección de la aplicación

El inicio en la construcción de un Sistema Experto consiste en definir correctamente el problema a resolver. Así, si un problema está bien resuelto por métodos numéricos clásicos, resulta inútil intentar una metodología de Sistemas Expertos. Por el contrario, si la situación no ha sido solucionado a la manera clásica más que en condiciones ideales, raramente alcanzadas o donde entran en juego numerosos parámetros de naturaleza simbólica que son difíciles de tener en cuenta en un modelo numérico, entonces se puede vislumbrar el empleo de Sistemas Expertos.

2) Búsqueda de un experto humano, datos o experiencia

Esta etapa se centra en buscar un experto humano que se encuentre en condiciones de participar en la construcción del Sistema Experto. En algunos casos bastará con una base de datos o experiencias que jugarán el papel del erudito humano.

3) Diseño del Sistema Experto

La tercera fase es el diseño del Sistema Experto en el que se incluyen las estructuras para almacenamiento, el motor de inferencia, los sistemas de explicación, la interfase con el usuario, etc.

4) Selección de la herramienta, Shell o lenguaje de desarrollo

Después de haber determinado el diseño del Sistema Experto, el desarrollador puede encontrarse con determinadas dificultades. Una de las más importantes es el problema de elección de la herramienta para el desarrollo del Sistema Experto, para ello debe elegir una que se adapte a las necesidades de la propia aplicación.

Si existe un Shell que cumpla con las condiciones requeridas por el diseño, se recomienda utilizarse, no sólo por motivos económicos, sino también por razones de fiabilidad. Por otro lado, si se elige un lenguaje de desarrollo debe tomarse en consideración ciertas cuestiones como capacidad de generalidad para resolver problemas y ser lo más sencillo posible, por citar algunas.

5) Desarrollo de un Prototipo

Con los medios decididos en la etapa anterior, puede procederse al desarrollo de un prototipo de Sistema Experto capaz de resolver los casos más simples. Este prototipo es muy importante puesto que permite probar la validez del formalismo y por consiguiente de la herramienta elegida.

6) Validación del Prototipo

En esta fase se hace una evaluación de sus realizaciones y de su Base de Conocimientos en un ciclo que se repite hasta que se consiguen los resultados apetecidos. En efecto, la faceta de validación puede provocar revisiones si la comparación de Sistema Experto y la de los expertos no es del agrado de estos.

7) Refinamiento y Generalización

En esta etapa es donde se van puliendo defectos e incluyendo nuevos casos no contemplados en el diseño inicial que puede ser una modificación a la Base de Conocimientos y posiblemente la estructura de control.

El resultado de estas modificaciones debe ser una mejora de las realizaciones del Sistema Experto sin que esto signifique que tales cambios sean drásticos. En fin, si las dificultades son serias, la razón puede proceder de errores de conceptualización o de identificación que necesitan una reformulación de determinados conceptos utilizados por el Sistema Experto.

8) Mantenimiento y Actualización

Estas 2 fases son de suma importancia si se quiere llegar a un producto de calidad y con éxito comercial. En esta etapa se habrá que atender a las demandas de los clientes, resolviendo sus problemas, contestando a sus preguntas corrigiendo errores y actualizando el sistema con los nuevos avances que se vayan produciendo. Por supuesto que esta última etapa tiene aplicación generalmente sólo a empresas que se dedican a la comercialización del software, ya que en otros casos se desarrollan productos para uso interno exclusivamente.

Aplicaciones de los Sistemas Expertos

La construcción de un Sistema Experto será adecuada en diversas situaciones, por ejemplo, en aquellas circunstancias en las que los expertos disponen de conocimientos complejos en un área muy delimitada, donde no se cuente con algoritmos elaborados (o donde los existentes no puedan solucionar algunos problemas) o bien cuando no existan teorías completas. Es más, incluso habiéndolas, en ocasiones resulta imposible analizar todos los casos teóricamente imaginables.

Así, en estas situaciones resulta necesario contar con el conocimiento que el experto posee y que ha adquirido a través de su experiencia, para con ello, llegar a una solución en un espacio de tiempo aceptable.

Los dos tipos de problemas descritos se caracterizan además por el hecho de que, aunque es factible la existencia de una o más soluciones, el camino hacia ellas no se encuentra previamente fijado. Sin embargo, el experto, gracias a la información que posee del problema y a su experiencia, encuentra una de esas. Por lo que, mientras dicha

solución pueda ser susceptible de repetición y el planteamiento del problema se encuentre claro, existe un razonamiento que puede ser reproducido por un Sistema Experto.

Por otro lado, la estructuración e implementación del conocimiento del experto requiere una gran cantidad de trabajo, así que sólo valdrá la pena realizar el esfuerzo de crear un Sistema Experto, cuando un Conocimiento sea válido durante un largo espacio de tiempo y, además vaya a ser utilizado por un gran número de personas. En este sentido, el Sistema, supondrá una descarga del experto en lo que respecta al trabajo rutinario y, por ende, una reducción de sus problemas, así como también de las decisiones erróneas, acelerando con ello, los procesos de toma de decisiones. La Tabla III.1.1 da una visión de los Campos de Aplicación en diferentes sectores.

Asimismo, a continuación se muestra, por áreas (sin abundar en todas), una descripción breve de diferentes Sistemas Expertos que se han venido construyendo, y así, comprender un poco más como funcionan estas aplicaciones de la Inteligencia Artificial.

1) Medicina

Computadoras con programas estadísticos han sido empleados, desde hace mucho tiempo como herramienta en la toma de decisiones en el área sanitaria. Uno de los principales campos de aplicación ha sido el Diagnóstico Médico, esto es, los síntomas son transmitidos a un ordenador y éste, mediante ciertos métodos, selecciona una enfermedad. Así, los programas más complejos utilizan técnicas de diagnóstico secuencial, lo cual implica pedirle al paciente la realización de nuevos análisis que proporcionen información para una determinación de mayor fiabilidad.

Al final de la década de los ochenta, ya se había investigado una amplia gama de Sistemas de Diagnóstico, llegando a la conclusión de que realmente son adecuados en la práctica médica, pues poseen la capacidad de almacenar grandes cantidades de datos sin distorsionarlos, ni deteriorarlos durante extensos periodos de tiempo, asimismo, permiten recuperar toda la información en la misma forma en que fueron almacenados, dando la oportunidad de efectuar con ella, complejas operaciones lógicas y matemáticas a alta velocidad. Por otra parte, muestran facilidad para presentar diversos diagnósticos posibles en una forma lógica.

La exactitud de un Diagnóstico Asistido por Computadora depende de múltiples factores, entre ellos, el alcance de los datos almacenados en la Base de Datos o Base de Conocimientos, la Complejidad del Diagnóstico y el Algoritmo de Selección. Así, algunos de estos Sistemas han llegado a mostrar una fiabilidad superior a la del experto profesional. Sin embargo, hay que reconocer sus importantes limitaciones, pues los aspectos psicológicos relacionados con un ordenador en la consulta médica, no pueden pasar desapercibidos.

De esta suerte que, los Sistemas Expertos Médicos que mayor interés han despertado son: MYCIN, CASNET, INTERNIST, PIP, el Consultor Digital de Terapia IRIS y EXPERT. Mientras que

entre los programas experimentales se encuentran PUFF (un Programa para el Estudio de la Función Pulmonar), HODKINS (un Sistema para Planificar el Tratamiento de la Enfermedad de Hodkins), HEADMED (un Consultor en Psicofármacos), VM (un Supervisor de Unidad para Cuidados Intensivos) y ONCOCIN (un Programa para Supervisar el Tratamiento de pacientes cancerosos desahuciados que siguen algún tratamiento experimental).

El MYCIN, es un Sistema Experto que se encuentra especializado en la Consulta y Diagnóstico de enfermedades como Bacteremia y Meningitis. Así, el Sistema tiene una gran versatilidad, pues no requiere que el médico consultor sea un experto en padecimientos infecciosos, debido a que los conocimientos almacenados en MYCIN se encuentran como un conjunto de reglas complementadas con factores de certidumbre.

Las Reglas de este Sistema, se hallan escritas en lenguaje LISP. A nivel individual, cada regla constituye una unidad de información especifica, la cual conduce a una ACCIÓN o conclusión, cuando se cumplen todas las condiciones de la PREMISA. A continuación se muestra una regla tipica de MYCIN.

```
PREMISA:    (AND(SAME(CNTXT INFECT PRIMARY)
                 (BACTEREMIA)
                 (MEMBF CNTXT SITE STERILESITES)
                 (SAME CNTXT PORTAL GI))
ACTION:     (CONCLUDE CNTXT IDENT BACTERIODES TALLY .7)
```

Otro Sistema Experto es el CASNET (Causal Asociation Network) creado por la Universidad Rutgers, el cual realiza Diagnósticos Médicos sobre el Glaucoma. Este programa considera la enfermedad como un proceso dinámico que se puede modelar en función de una serie de estados patológicos relacionados. Así, el Sistema puede identificar el cuadro patológico que desencadena el padecimiento y determinar el tratamiento más adecuado para cada caso.

Aunque la aplicación principal del CASNET es el Glaucoma, el Sistema posee un Esquema de Planteamiento y Procedimientos de Toma de Decisiones que pueden ser aplicables a otras áreas médicas como la Oftalmologia.

El Programa INTERNIST constituye otro Sistema Experto que fue creado por la Universidad de Pittsburg, el cual se encuentra especializado en medicina interna. Su funcionamiento consiste en introducir toda la información relacionada con el paciente, como síntomas, resultados de pruebas de laboratorio, historia, etc., para posteriormente diagnosticar las posibles enfermedades. Así, una vez que el programa ha discriminado entre las diferentes hipótesis, selecciona aquella que ofrezca mayor fiabilidad.

Los conocimientos del Sistema, respecto a la enfermedad, se organizan en una estructura de árbol, generando un modelo por cada uno de los nodos, por lo que el

diagnóstico corresponde al conjunto de nodos de dicha estructura, e incluye todos los síntomas. Actualmente existe una versión mejorada de INTERNIS, el INTERNIST II que diagnostica la enfermedad, dividiendo el árbol en subárboles sucesivos de menor tamaño. Cabe mencionar que este programa puede diagnosticar más de 500 enfermedades de medicina interna.

Otro Sistema Experto en medicina es el Present Illness Program (PIP), desarrollado en el Instituto Tecnológico de Massachusetts (MIT) y se encuentra especializado en enfermedades de tipo hepático. Sus conocimientos están configurados en estructuras (existen 36) que contemplan enfermedades, estados clínicos y estados fisiológicos del paciente.

El Iris, constituye otro Sistema de Aplicación Médica, diseñado en la Universidad de Rutgers, el cual puede realizar trabajos experimentales con otros Sistemas Expertos de Consulta, es decir, intenta facilitar el análisis a nivel experimental de representaciones alternativas de los conocimientos médicos, estrategias clínicas, etc.

El Sistema EXPERT también diseñado en la Universidad de Rutgers proporciona ayuda en el diseño y comprobación de modelos consultivos, incorporando la experiencia obtenida en áreas como reumatología, oftalmología y endocrinología.

2) Química

Los Sistemas Expertos encuentran aplicación actualmente en muchas áreas de investigación pura o aplicada, entre ellas se encuentra el análisis químico en problemas no numéricos, como la identificación de estructuras moleculares en compuestos orgánicos y la planificación de secuencias de reacciones, que conducen a la síntesis de complejos químicos.

La identificación de estructuras moleculares es de vital importancia en áreas como química, biología y medicina, pues existen muchas situaciones en las que no es posible emplear métodos analíticos, por ejemplo, la cristalografía por rayos X. Así, en estos casos los investigadores tienen que interpretar los datos obtenidos por otros medios como la espectrometría[3] de masas. No obstante, ya existen programas de IA que, a partir de información parcial pueden desarrollar todas las estructuras posibles. Los Sistemas Expertos como DENDRAL, CONGE, Meta-DENDRAL y CRYSALIS utilizan este tipo de planteamiento para identificar constituciones moleculares. Por otra parte, también existen Programas como LASHA, SECS y SYNCHEM que están especializados en la búsqueda de técnicas de laboratorio que permitan la síntesis de sustancias conocidas.

El programa heurístico DENDRAL que continuó al algoritmo que lleva el mismo nombre, identifica las posibles estructuras moleculares y los átomos constituyentes, a partir

[3] Este término se refiere al hecho de llevar a cabo un Análisis Espectral, es decir, separar los átomos según la masa del cuerpo.

del espectograma de la molécula que se esté investigando, siendo su objetivo primordial, sustituir el método algoritmico por una estrategia más económica. Sin embargo se presentaban problemas, pues los químicos tenían ciertas dificultades para expresar sus conocimientos convenientemente. Así, en 1970, comenzó el proyecto Meta-DENDRAL, cuyo propósito era el que pudiera inferir reglas de espectrometría de masas, esto a partir del estudio realizado por químicos especializados.

Ya a mediados de la década de los setenta, se descubrió las limitaciones del programa heurístico DENDRAL, debido a los problemas que presentaba el algoritmo, pues sólo se podían estudiar estructuras acíclicas como cetonas, alcoholes, éteres, aminas, etc. Por lo que, en 1976 se diseña el Sistema CONGEN.

Los programas DENDRAL se han utilizado con éxito para determinar las estructuras de varios tipos de moléculas como terpenos en productos naturales, esteroles marinos, impurezas químicas, antibióticos y feromonas en varias especies de insectos.

El Meta-DENDRAL, diseñado para inferir reglas de la espectrografía de masas en base a estructuras ya conocidas, aprende explorando cientos de pares de puntos homólogos de espectogramas de sustancias muy diversas.

Las reglas así establecidas, pueden ampliarse para incorporar nuevos datos. Asimismo, posee la capacidad derivar dichos paradigmas a partir de los ya establecidos, incluso redescubrirlos.

Otro Sistema Experto es el CRYSALIS, el cual centra su interés en el estudio cristalográfico[4] de las proteínas. Así, en dicho programa, las hipótesis se representan en base a una estructura jerárquica de los datos, donde es posible añadir, modificar y comprobar las suposiciones en una pizarra virtual.

3) Matemáticas

MACSYMA, cuya primera versión data de 1968, es un Sistema de gran potencia diseñado para prestar ayuda a matemáticos, científicos e ingenieros a solucionar problemas de índole matemática. Posee un elevado nivel de competencia en tratamiento de expresiones algebraicas. Asimismo, puede efectuar más de 600 tipos diferentes de operaciones matemáticas como diferenciación, integración, álgebra vectorial, matrices, ecuaciones, etc. El programa está escrito en un lenguaje de programación especial que una vez compilado en LISP comprende aproximadamente 230000 palabras.

4) Geología

Otras de las áreas en las que se están desarrollando Sistemas Expertos es la Geología. El Sistema más conocido es el PROSPECTOR, desarrollado por la compañía SRI International

[4] Este estudio permite cristalizar una substancia o materia para su investigación.

para prestar ayuda a los geólogos que prospeccionan[5] minerales en roca dura. En dicho Sistema, el usuario proporciona información referente a la región, como tipos de roca, minerales que se encuentran en la zona, productos que han sufrido alteraciones debido a fenómenos naturales, etc. De tal manera que, con esas referencias, el PROSPECTOR busca coincidencias entre los modelos que posee y la información de entrada y, si en un momento dado cree que es necesario, para llegar a una conclusión, solicitar más pormenores al usuario, entonces lo hace. Así, el beneficiario tiene la posibilidad de intervenir directamente en todas las etapas del proceso, es decir, proporcionar nuevos datos o modificar la información que ya se encuentra ahí. Por otra parte, los razonamientos de PROSPECTOR, se hallan controlados por una sofisticada Red de Inferencia, donde sus nodos corresponden a aserciones geológicas.

Actualmente, el Sistema cuenta con 5 modelos diferentes, los cuales fueron desarrollados en colaboración con 5 consultores especializados en temas geológicos, como sulfuros tipo Koroko, plomo/zinc tipo valle del Mississipi, cobre porfirítico tipo A, sulfuro de níquel y uranio en roca arenística. Conjuntamente, todos los modelos totalizan 541 aserciones y 327 reglas.

Por otro lado, en función de los datos de entrada, el Sistema ajusta la probabilidad de certidumbre de sus hipótesis.

5) Educación

La tecnología de la Computación se ha venido usando desde de la década de los sesenta en aplicaciones relacionadas con la planificación de recursos, evaluación de exámenes y administración de recursos educativos.

Por lo que, uno de los principales objetivos de la enseñanza asistida por computadoras (CAI: Computer Aided Instruction), ha sido precisamente, investigar técnicas que permitan la producción de programas educativos, los cuales den la oportunidad de estructurar las materias necesarias para los cursos, en base a las necesidades particulares de cada alumno. Así, en los primeros Sistemas de Instrucción Inteligente por ordenador (ICAI: Intelligent Computer Aided Instruction), el material de la asignatura, se impartía independientemente de los procedimientos de enseñanza, esto con el fin de llegar a generar problemas y comentarios diferentes para cada uno de los colegiales.

En la actualidad, las Técnicas de Inteligencia Artificial están contribuyendo al diseño de programas, que no sólo toman en cuenta los aspectos débiles de cada estudiante, sino que también se interesan por sus puntos fuertes.

[5] En este proceso de prospección se explora el terreno en busca de yacimientos de minerales beneficiables o de combustibles fósiles.

Entre los primeros Sistemas Inteligentes para este campo se encuentran el SCHOLAR, un tutor de Geografía; SOPHIE, una guía en localización de averías en equipos electrónicos y EXCHEK, un orientador en lógica y teoría de conjuntos. Así, todos estos programas tenían en común el centrar su atención en el contenido de las lecciones, además de poseer un elevado nivel de conocimientos y una gran capacidad para resolver una extensa diversidad de situaciones de modo interactivo.

Así pues, se ha mostrado brevemente algunas de las áreas en las que los Sistemas Expertos han incursionado, sin embargo, existen todavía más, por lo que en la Figura III.1.5 se presentan otras aplicaciones de dichos programas.

III.2 REPRESENTACIÓN DEL CONOCIMIENTO

Para llevar a cabo el procesamiento y manipulación de Conocimiento en Sistemas Expertos, necesariamente se requiere contar con un método que permita la formalización y estructuración de dicho saber, pues en la mayoría de los casos éste se obtiene mediante herramientas, como la entrevista a expertos, quienes en su mayoría, proporcionan la información en forma descriptiva, situación que muchas veces complica el proceso.
Así, existen métodos formales que permiten representar Conocimiento, como la lógica de predicados, lógica modal, lógica multivaluada y lógica difusa, sin embargo, en un momento dado pueden no llegar a mostrar lo que realmente se desea; por lo que se han desarrollado procedimientos que ofrecen una alternativa muy eficiente a la estructuración y procesamiento del saber. Entre ellos se encuentran las Redes Asociativas, los Frames, Reglas de Producción y los Scripts.

Redes Asociativas

Las representaciones en forma Red proporcionan un medio de organización y exhibición de conocimiento estructurado. De esta manera, en una Red, las piezas del conocimiento son categorizadas juntas en grupos semánticos coherentes, proporcionando una demostración gráfica de los objetos, sus atributos y las relaciones existentes entre ellos y otras entidades.

Así, las Redes Asociativas pueden verse como grafos dirigidos con nodos etiquetados y arcos o flechas. En la Figura III.2.1 se muestra un fragmento de una simple Red con estas características.

Las Redes Asociativas o Redes Semánticas (como también se les denomina) fueron introducidas por Quillan (1968) con el fin de modelar las semánticas de las palabras y sentencias en inglés. Asimismo, desarrolló un Sistema el cual encontraba los significados

Sector / Aplicación	Banca Seguros	Industria	Comercio Servicios	Encargos Estatales
Control de Procesos, Supervisión	- Observación de tendencias	- Control de procesos - Gobierno de procesos - Aviso de estados de excepción	- Observación de tendencias	- Control de centrales nucleares o de grandes redes (agua, gas)
Diseño		- Configuración - Instalaciones fabriles - Diseño de productos	- Requisitos de productos	- Redes distribución (correos, agua)
Diagnóstico	- Concesión de créditos - Comprobación de Hipotecas - Análisis de siniestros	- Motivo de fallo - Mantenimiento	- Concesión de créditos - Cálculo de riesgos	- Diagnóstico médico - Diagnóstico técnico
Planificación	- Análisis de Riesgos - Gestión de valores - Planificación de inversiones	- Funciones lógicas de proyectos - Proyectos	- Análisis de riesgos - Análisis de mercado	- Planificación de inversiones - Planificación de emergencias
Asesoramiento	- Asesoramiento de clientes	- Asesoramiento de clientes	- Asesoramiento de clientes -Servicios especiales	- Asesoramiento de clientes
Formación	- Formación de colaboradores - Formación del servicio exterior	- Formación de colaboradores	- Formación de colaboradores - Formación del servicio exterior	- Formación interna en cuestiones jurídicas

Tabla III.1.1 Campos de aplicación de los Sistemas Expertos.

Europa

- Sistema Experto para soportar el tratamiento de casos relacionados con la importación y exportación de azúcar (Instituto Belga de Administración, Everberg, Bélgica.
- Sistema Experto (ERASMUS) para decisión en el mantenimiento de caminos (ILOG; Gentilly , Francia).
- Sistema Experto (GESPI) para resolver problemas administrativos de localización de plataformas para llegada de trenes y planeación de una ruta para su salida sin que causen conflictos (GSI-ERLI, Charenton-Le Pont, Francia).
- Sistema Experto (XUMA) para asistencia en la protección ambiental de sitios contaminados (Kernforschungszentrum (Centro de Investigación Nuclear), Karlsruhe, Alemania).
- Sistema Experto (DAX/MED2) para la calidad en una unidad de control de transmisión automática (Universidad de Karlsruhe, Karlsruhe, Alemania).
- Sistema Experto (LAIDA) para el análisis de disturbios en redes eléctricas (LABEIN, Bilbao, España).
- Sistema Experto (AMETHYST) para el diagnóstico automático de fallas mecánicas (Aplicaciones Inteligentes, Ltd., Escocia).
- Sistema Experto (PFES) para la formulación de productos (Lógica, Cambridge, Inglaterra).
- Sistema Experto (RAP) para localizaciones navales (SD-Scicon, Surrey, Inglaterra).

Figura III.1.5 Ejemplos de algunos Sistemas Expertos en el mundo.

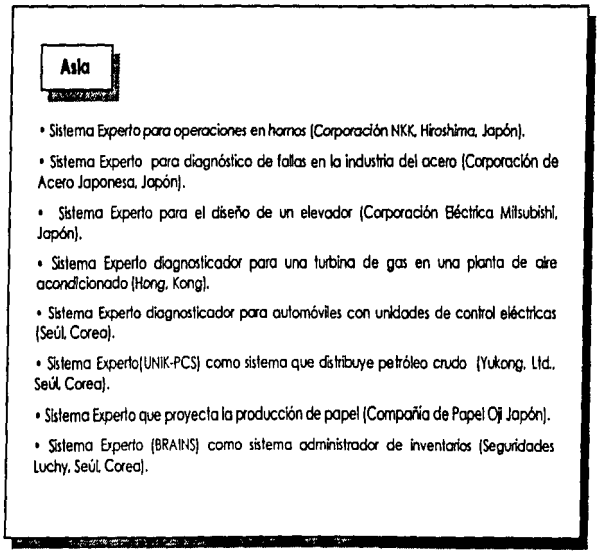

Figura III.1.5 Continuación.

México

- Sistema Experto (SECAL) para diagnosticar problemas en operaciones de calderas (ITESM/CRYSLER, Monterrey, México).
- Sistema Experto (SEMPREP) para diagnosticar fallas en una máquina textil (ITESM; Monterrey, México).
- Sistema Experto para ayudar en operaciones de barnizado (Instituto de investigaciones Eléctricas/Grupo CONDUMEX, Cuernavaca, México).
- Sistema Experto (SEND) para controlar las desviaciones de tono en colorantes (ITESM/PYOSA, Monterrey, México).
- Sistema Experto (SEMAT) para diagnosticar problemas en máquinas para tubos de cremas dentales (ITESM/CYDSA; Monterrey, México).
- Sistema Experto (AFFIN) para la evaluación de proyectos de inversión industrial (ITESM/FONEL Cuernavaca, México).
- Sistema Experto para la estimación de riesgo en la vida de un individuo (Seguros América, Cd. de México, México).

Figura III.1.5 Continuación.

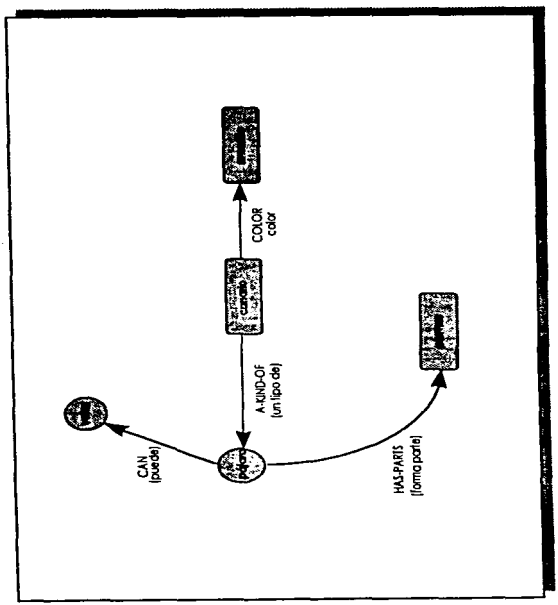

Figura III.2.1 Fragmento de una Red Asociativa.

entre éstas, a partir de los caminos que las conectaban, en las que sus conexiones eran determinadas a través de un tipo de "spreading activation" entre 2 frases.

El modelo de Quillian de Redes Semánticas, posee un grado intuitivo en la manera en que relaciona la información cuando ésta es agrupada a través de líneas de unión. Por lo que, este tipo de organización, de alguna manera, muestra la forma en la que el conocimiento es almacenado y retenido en los seres humanos.

Por otra parte, el conocimiento mostrado de una manera gráfica, puede algunas veces, ser más expresivo que otros esquemas de representación, de ahí su popularidad, sin embargo, los enunciados de las relaciones expresadas a través de los arcos, deben ser formuladas fuera de la Red. De esta manera, ha sido empleado en gran variedad de Sistemas, tales como Comprensión del Lenguaje Natural, Retención de Información, Base de Datos Deductivas, Sistema de Aprendizaje, Visión por Computadora y Sistemas de Generación de Habla.

1) Sintaxis y Semánticas de una Red Asociativa

No existe hasta la fecha una Sintaxis o Semánticas generales para una Red Asociativa, sino que, dichas Reglas tienen que ser diseñadas dependiendo de la implementación y contexto del programa, por consiguiente, estarán determinadas por el objeto y sus relaciones.

Sin embargo, se han llevado a cabo varios esfuerzos por establecer estándares aceptables, entre ellos se encuentran los realizados por Schubert, Goebel y Cercone (1979), Shapiro (1979), Hendrix (1979) y Brachman (1979).

Básicamente, el lenguaje de las Redes Asociativas está conformado por letras del alfabeto, símbolos relacionales, símbolos de conjuntos, dígitos decimales, nodos ovales o cuadrados y arcos dirigidos de longitud arbitraria. Los símbolos palabra utilizados son los que representan las constantes objeto y las *n*-relaciones. Los nodos son comúnmente usados para los objetos o nombres, y los arcos para las relaciones (Ver Figura III.2.2). Así, en la Figura puede distinguirse un número de conexiones arco en las que se incluyen predicados como ISA, MEMBER-OF, SUBSET-OF, AKO (a-kind-of), HAS PARTS, INSTANCE-OF, AGENT, ATTRIBUTES o SHAPED-LIKE, entre otros. También, aunque no es muy común, dichas relaciones arco se han empleado para expresar relaciones de modalidad (tiempo, humor), relaciones lingüísticas (fuente, objetivo), conexiones lógicas (o, no, y), cuantificadores (todos, algún), relaciones de conjunto (subconjunto, miembro) y atributos.

Un arco particular lo constituye el ISA (is a) debido a que toma un especial significado: volviendo al ejemplo de la Figura III.2.2, se dice que Bob es un (is a) profesor y que el Sistema de la Universidad Estatal es un (is a) instituto de alto aprendizaje. Así, dicha

Figura III.2.2 Nodos y tipos de Arcos en una Red Asociativa.

relación es a menudo empleada para representar el hecho de que un objeto es de un cierto tipo, o para expresar que éste es un subtipo de otro.

Por otra parte, la estructura de las Redes Asociativas permite la implementación de la propiedad de herencia, es decir, una forma de inferencia que es reconocida como una forma de razonamiento por default; de tal manera que, los nodos que son miembros o subconjuntos de otros nodos, pueden heredar propiedades desde sus niveles más altos de antecesores. Por ejemplo, en la Figura III.2.3 es posible inferir que un ratón tiene pelo y bebe leche.

2) Gráficas Conceptuales

Aunque no existen estándares que sean aceptados para una sintaxis y semánticas en una Red Asociativa, puede hacerse una aproximación empleando para ello, un tipo de representación a través del uso de Gráficas Conceptuales. El formalismo de estos Esquemas ha sido adoptado por muchos investigadores de IA, entre ellos, John Sowa (1984) y sus colaboradores.

Una Gráfica Conceptual es una representación gráfica de una percepción mental, la cual está compuesta de conceptos primitivos o básicos, así como de las relaciones existentes entre dichas concepciones. Por lo que, cuando se unen todas éstas de una manera coherente, forman una Estructura compleja de Conocimiento. Un ejemplo de tal Gráfica es la que se presenta en la Figura III.2.4, en donde los conceptos se hallan encerrados en cajas y las relaciones existentes entre éstos están enmarcados en óvalos. Mientras que, la dirección de la flecha corresponde al orden de los argumentos en las relaciones que ellos conectan.

Los símbolos de concepto se refieren a entidades, acciones, propiedades o eventos que ocurren en el mundo real. De manera que éste puede ser individual o de tipo genérico. En lo que a concepciones individuales se refiere, éstas poseen un campo, el cual va seguido por otro denominado referente. Por ejemplo, el concepto [PERSON:joe] tiene un tipo PERSON y un referente Joe. Así, los Referentes como Joe y soup en la Figura III.2.4, son llamados conceptos individuales porque ellos se refieren a entidades específicas. EAT y SPOON no tienen campo referentes pues son conceptos genéricos que se refieren a formas no particulares. Los conceptos como AGENT, OBJECT e INSTRUMENT se obtienen de una colección de concepciones estándares.

Por otro lado, una Gráfica Conceptual puede ser representada como un texto, así para el ejemplo mostrado en la Figura III.2.4, dicha simbolización quedaría expresada de la siguiente manera:

Figura III.2.3 Propiedad de Herencia en una Red Asociativa.

Figura III.2.4 Una Gráfica Conceptual.

[PERSON:Joe] ← (AGENT) ← [EAT] -
(OBJECT) → [FOOD:soup]
(INSTRUMENT) → [SPOON]

El lenguaje de las Gráficas Conceptuales está formado, tanto por letras mayúsculas como minúsculas, cajas y círculos, arcos dirigidos y un número de caracteres especiales incluyendo -, ?, !, *, #, @, ∀, –, ", :, [,], (,), →, ←, { y }. Algunos de ellos son empleados para exhibir la estructura de la Gráfica, mientras que otros son usados para determinar los referentes[6].

Frames (Marcos)

Los Frames fueron introducidos por Marvin Minsky (1975) como una estructura de datos capaz de representar un modelo mental de una situación estereotípica, como conducir un auto o, simplemente comer en un restaurante. De esta manera, el conocimiento acerca de un objeto es almacenado en la memoria como una unidad. Entonces, cuando una nueva situación se encuentra, se selecciona de esta fuente un Frame apropiado.

Los Frames en general, son estructuras parecidas a los registros que constan de una colección de elementos denominados slots y sus valores (slot values). De este modo, los slots pueden poseer un tamaño y tipo, así como nombres y valores o subcampos llamados facets (Ver Figura III.2.5).

Antes de su uso, los Frames pueden visualizarse como disposiciones preestructuradas de datos, así la distribución de éstos, como las definiciones de los slots, se hallan ya determinadas, de tal manera que, a lo largo de todo el proceso se van completando con información. Así que, en su desarrollo pueden existir Frames con la misma estructura pero diferente contenido (Ver Figura III.2.6).

Por otra parte, los valores de un slot (slot values) son de carácter heredable, de forma que, únicamente se requiere modificar el valor jerárquicamente superior en el slot y todas las instancias subordinadas del Frame reciben la misma cantidad (Ver Figura III.2.7).

Para llevar a cabo el procesamiento de los Frames, se debe contar con un conjunto de reglas y procedimientos, de la misma manera como ocurre con las Redes Asociativas, de tal manera que sean activados por determinados acontecimientos.

---

[6] Para una mayor explicación a cerca de esto, ver Patterson (1990) Introduction to Artificial Intelligence and Expert Systems. Pág. 126-136.

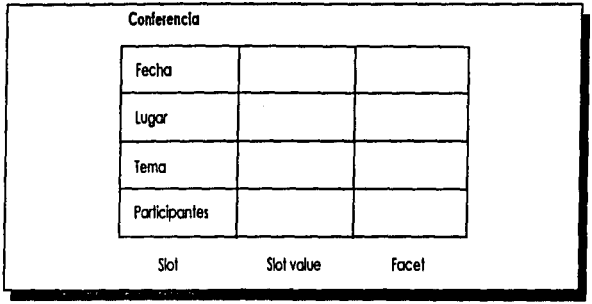

Figura III.2.5 Forma Gráfica de un Frame.

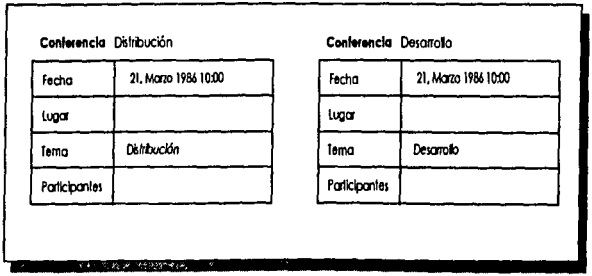

Figura III.2.6 Frames con misma Estructura pero con diferente Contenido.

Figura III.2.7 Figura que muestra la herencia jerárquica en un Frame, donde los valores heredados aparecen en cursiva.

Reglas de Producción

Las Reglas de Producción son una de las maneras más comprensibles que permiten representar el conocimiento. Dichas formas pueden percibirse como descripciones de acciones que van a depender de ciertas condiciones. Así, una sola Regla de Producción se considera como una unidad de conocimiento (chunk), siendo el componente más pequeño de que consta el Sistema.

La forma más común de una Regla de Producción consta de un SI seguida de un conjunto de condiciones, de manera que si son verdaderas se prosigue con un ENTONCES donde se ejecutará una acción, por ejemplo:

SI el color es rojo
ENTONCES es una manzana

Este tipo limitado de sintaxis llevó a los primeros Sistemas a poseer un alto número de Reglas, pues una de ellas no llegaba a representar todo lo que se quería. Así, los programas más modernos permiten incluir operadores booleanos (Y y O) en las condiciones, así como algunos comandos y permitir expresar más conocimiento, como se muestra a continuación:

SI hay dolor de cabeza
O (dolor en todo el cuerpo)
Y hay estornudos
ENTONCES tiene gripa
DE LO CONTRARIO es cansancio

Existen casos en los que se presenta la necesidad de manejar conocimiento vago, entonces para ello se requiere considerar factores de certeza (certainly factors), elementos que normalmente se encuentran entre los limites +1 y -1. Donde el +1 representaría "seguro que si"; el -1 "seguro que no" ; un +0.5 "probablemente si" y un -0.5 "probablemente no".

A continuación se presenta una Regla extraída del Sistema Experto MYCIN la cual considera factores de certidumbre.

SI (1) la infección es debida a bacterias primarias, y
(2) la localización del cultivo es una de las muestras estériles, y
(3) la entrada probable es el tacto gastrointestinal.
ENTONCES Es bastante probable (0.7) que la identidad de los organismos sea bacteria.

Una de las ventajas de trabajar con Reglas es su modularidad. En teoría, es posible agregar, borrar o modificar reglas sin afectar la Base donde se encuentren. Sin embargo, en la práctica puede encontrarse que no son tan independientes unas de las otras, produciéndose algunos efectos. Por otro lado, con una Base grande de unas 800 o 1000 Reglas casi es imposible la necesidad de agregar nuevo conocimiento.

Así, los Sistemas que manejan Reglas de Producción son denominados Sistemas de Producción (Ver Figura III.2.8).

Scripts

Los Scripts son otros esquemas de representación estructurada introducidos por Roger Schank (1977). Así, un Script es una estructura predefinida, similar al Frame, que fue utilizado para representar experiencias el cual contiene expectativas, inferencias y otro tipo de conocimiento que es relevante para una cierta situación, que puede ser tan simple como ver películas, comprar en un supermercado, comer en algún restaurante o visitar a un dentista.

La estructura de un Script se encuentra descrita en términos de actores, roles y escenas. Así en la Figura III.2.9 se muestra un ejemplo de un supermercado, donde el Script tiene 4 escenarios los cuales corresponden a los principales eventos que comúnmente ocurren en ese lugar.

III.3 SHELLS Y ENTORNOS

Los lenguajes de programación de la IA han sido diseñados para poder construir con ellos determinadas representaciones de la realidad. Una vez que se desarrollaron las primeras aplicaciones, se descubrió que era posible transferir algunos procedimientos generales válidos para la mayoría de los sistemas que tuviesen una representación del problema similar. Por ejemplo, el desarrollo del MYCIN (sistema experto para diagnosticar enfermedades infecciosas), dio lugar a la idea de eliminar los aspectos específicos del tema para dejar sólo la estructura o cáscara (shell en inglés) de lo que sería un sistema de diagnóstico en general. Así, nacen los *Shells* que son herramientas construidas con algún lenguaje de programación (por ejemplo, LISP o C), que añaden a las funciones de generales propias de éstos otras muchas funciones específicas para el desarrollo de Sistemas Expertos.

Así, un Shell puede ofrecer estructuras para Representar conocimiento, construir un Mecanismo de Inferencia o simplemente como apoyo en el diseño de un Componente

Figura III.2.8 Arquitectura de un Sistema de Producción.

SCRIPT NAME:	food market
TRACK :	supermarket
ROLES :	shopper
	deli attendant
	checkout clerk
	sacking clerk
	other shoppers
ENTRY :	
CONDITIONS :	shopper needs groceries
	food market open
PROPOS :	shopping cart
	display aisles
	market items
	checkout stands
	cashier
	money
SCENE1 :	Enter Market
	shopper PTRANS shopper into market
SCENE2 :	shopper PTRANS shopping-cart to shopper
	Shop For Items
	shopper MOVE shopper through aisles
	shopper ATRANS bags to shopper
	Exit Market
	shopper PTRANS shopper to exit market
RESULTS :	
	shopper has less money
	shopper has grocery items
	market has less grocery items
	market has more money

Figura III.2.9 Estructura de un Script de un Supermercado.

Explicativo. Sin embargo, en la actualidad aún no ha sido posible introducir todos estos componentes en un Shell. Por lo que sería deseable que éstos pudieran incluir los siguientes aspectos:

1. Formalismo para la Representación de conocimiento.
2. Algunos medios de estructuración para la Base de Conocimientos.
3. Mecanismo de Inferencia.
4. Una interfase de usuario, la cual pueda ser adaptada de acuerdo a las necesidades primarias del Sistema Experto, así como en la estructuración y ampliación de la Base de Conocimientos.
5. Apoyo para poder llevar a cabo la construcción de un Componente Explicativo.
6. Que cuente con ciertos mecanismos para realizar una comprobación en cuanto a la consistencia, búsqueda de errores y crecimiento de la Base de Conocimientos.

Por tanto, un Shell está conformado por algunas partes de un Sistema Experto, pero independientes del conocimiento. Así, con un Shell, el trabajo para el encargado del desarrollo disminuye, permitiéndosele concentrar más en la creación de la Base de Conocimientos, y sin la necesidad de poseer un profundo dominio del lenguaje de programación.

A pesar de las ventajas que puede poseer un Shell, también se han podido distinguir algunos inconvenientes como los que se enuncian a continuación:

1. Pese a la flexibilidad que ofrece un Shell, el encargado del desarrollo (del Sistema Experto) se encuentra ligado a los principios básicos del mismo, lo cual impide que éste no funcione como base para cualquier Sistema Experto, sino sólo para aquellos que tengan objetivos afines (diagnóstico, por ejemplo).
2. El responsable de la construcción del Sistema debe conocer el lenguaje de programación del Shell, así como la sistemática del mismo, situación que vuelve relativo el punto referente al dominio de dicho lenguaje.
3. Como continuación del punto anterior, para la modificación o adición de funciones, esta sólo es posible si existe una interfase para determinado lenguaje de programación, lo cual requiere un mayor conocimiento de éste, así como también un incremento en el tiempo.
4. Por último, la utilización de Shell todavía se encuentra muy limitada, debido a lo elevado de sus precios y escasa disponibilidad.

A pesar de los inconvenientes ya mencionados, para llevar a cabo una estructuración rápida de un Sistema Experto, se recomienda altamente, iniciar al menos con un Shell.

Por otra parte, un segundo grupo de herramientas propio de la IA viene constituido por los instrumentos que se han desarrollado como ayuda en alguna fase del proceso de desarrollo de sistemas basados en el conocimiento. Estas herramientas se orientan principalmente hacia el problema de la adquisición de conocimiento y el desarrollo de la interfaz de usuario.

Finalmente, y bajo esa misma idea de ofrecer herramientas que ya incluyen funciones específicas para desarrollar aplicaciones concretas, pueden incluirse otro tipo de productos como son los hipertextos y los sistemas multimedia. Estas herramientas no son específicas de la IA, aunque su desarrollo y utilización suele moverse muy cerca de ella.

CAPITULO IV

EL LENGUAJE DE PROGRAMACIÓN DELPHI

CAPÍTULO IV
EL LENGUAJE DE PROGRAMACIÓN DELPHI

IV.1 INTRODUCCIÓN

El lenguaje Delphi de Borland, es uno de los nuevos productos en la industria de la computación, así, éste surge como un nuevo tipo de programación orientada a objetos, con código en Pascal y además con ambiente visual, con el que se pueden crear entornos amigables, proporcionando también herramientas para diseñar componentes reusables.

Así, este capítulo intenta dar una breve explicación de lo que es Delphi, pues sería demasiado para este apartado, pretender dar un curso rápido, por lo que si se desea revisar detalladamente alguna cuestión, puede consultarse el material que se incluye en las referencias.

Características de Delphi

Entre las características que posee Delphi se pueden enunciar las siguientes:

1) Componentes extendibles y reusables.

   Delphi elimina la necesidad para los desarrolladores, de programar los mismos componentes como etiquetas (labels), botones (buttons) y aún cajas de diálogo (dialog boxes). Por ejemplo, en Windows, a menudo vemos los mismos "objetos" una y otra vez en diferentes aplicaciones. Así, las cajas de diálogo "Choose File" y "Save File" son muestras de componentes reusables. Asimismo, también se pueden personalizar elementos de acuerdo como lo requiera la aplicación. Por otra parte, Delphi posee objetos visuales y no visuales (más adelante se hablará de ellos) predefinidos como botones, objetos de datos, menús y cajas de diálogo. Así, los objetos de datos, por ejemplo, brindan la capacidad para desplegar información tan sólo al hacer "click" con el "mouse" (ratón) y sin necesidad de programar.

2) Soporta los VBX.

   Delphi tiene la propiedad de apoyar objetos VBX directamente y, a través de la paleta de componentes (Component palet) para de esta manera, facilitar el acceso a dichos entes y herramientas.

3) Aplicación y templado de formas.

   Delphi proporciona formas preconstruidas y empleo de plantillas que se pueden utilizar para crear una aplicación.

4) Ambiente de Desarrollo Personalizado.
   La paleta de componentes, el editor de código y las plantillas, son áreas donde Delphi puede ser completamente personalizado.
5) Compilación de Programas.
   Mientras que otros ambientes visuales de Windows pretenden compilar programas, usualmente lo hacen para sólo una parte de ellos, utilizando posteriormente un intérprete para crear un ejecutable. Delphi, por el contrario, produce ya ejecutables compilados, sin necesidad de contar con un intérprete, así la ejecución de los programas son más rápidos.
6) Capacidad para accesar datos.
   Delphi cuenta con un Motor de Base de Datos (Borland Database Engine: BDE), que admite accesar información de los formatos más populares, además de permitir la manipulación cliente/servidor por ejemplo con Sybase, servidores SQL, SQL Windows, Oracle e InterBase de Borland.
7) Facilidad para crear librerías DLL.
   Una librería DLL (Dynamic-Link Library) es un módulo ejecutable (con extensión .DLL) que contiene código que puede ser usado por otras DLL o aplicaciones. Entre estas se encuentra C++, Paradox, Visual Basic y Power Builder.
8) Manejo de Imágenes.
   Delphi posee una herramienta de diseño de imágenes que se puede usar para crear y editar bitmaps, iconos y cursores para ser desplegados en la aplicación que se esté realizando.

IV.2 LA NUEVA PROGRAMACIÓN EN PASCAL

Delphi es un ambiente de desarrollo que utiliza muchas características y conceptos de la Interfase de Usuario Gráfica (GUI) de Microsoft Windows, proporcionando un completo control de la aplicación bajo ese mismo entorno.

Los ambientes de programación visual, utilizan términos que hacen referencia a muchas "cosas" que son las que van a constituir la aplicación. Así, se emplean expresiones como objeto, propiedad y evento. Asimismo, surgen también conceptos como Programación Dirigida por el Evento y Programación Basada en Objetos como se explicará a continuación.

Programación Dirigida por Eventos

La Programación Dirigida por Eventos (PDE), ha sido ya desarrollada aún antes de las GUIs, por lo que puede ser implementada en un gran número de formas. Así, con la

introducción del mouse y las posibilidades que éste brinda al usuario, la PDE se ha convertido en un requerimiento para ellos mismos y para los desarrolladores.

Antes de los ambientes de la PDE, los estilos de programación con procedimientos "top-down" eran considerados como un arte, pues eran muy útiles cuando se construían piezas de código que manejaban grandes cantidades de procesamiento. Sin embargo, en este tipo de programas se empleaban complejos menús, así como secuencias de claves que se enlazaban con el procesamiento de la aplicación.

Así, la PDE no surge como un arma que tiene como objeto reemplazar este tipo de programación, sino como una herramienta capaz de complementarlo, esto al proporcionar una separación entre la interfase de usuario y el procesamiento. Por lo que, Delphi y otros ambientes dirigidos por el evento, proporcionan áreas de trabajo que permiten al programador concentrarse más en la lógica de la aplicación.

Por otro lado, la PDE no es algo completamente desconocido por nosotros, pues en la vida real podemos encontrar también situaciones dirigidas por el evento, por ejemplo, considere la sala de su casa, en donde las siguientes ocurrencias son elementos provocados (o conducidos) por otro evento externo:

a) Su televisor cambia después de que presiona el botón del control remoto.
b) Un amigo llama. En respuesta a la señal su teléfono suena.
c) Usted no contesta el teléfono. Después de cierto tiempo la contestadora responde para tomar el mensaje.
d) El amigo no deja mensaje, decide visitarle. En la puerta, su amigo toca el timbre. En respuesta a este evento, usted sale para abrir la puerta.

La ocurrencia de eventos en la vida real que resultan de una o más acciones específicas originan eventos. Por lo que, la clave para la PDE es determinar qué eventos requerirán un manejo especial. Así, cuando se está trabajando en Windows, un evento puede ser cuando el usuario da un doble click en una región particular de la pantalla. Por otra parte, también DOS proporciona medios para manejar eventos similar a Windows.

Programación Basada en Objetos.

La Industria de Software se encuentra promoviendo una nueva tecnología y metodologías para el desarrollo de programas.

Así, en la mitad de los 80's, dicha Industria introdujo el concepto de Programación Orientada a Objetos (POO), poseyendo un conjunto de objetivos claramente definidos y con ello, ser capaz de proporcionar a los desarrolladores herramientas tangibles para facilitarles la creación de sus aplicaciones.

La POO ha sido considerada como un tipo de programación "top-down" superior, esto debido a su capacidad para crear código reusable y por ser un buen candidato para modelar situaciones del mundo real.

Algunos desarrolladores de software empiezan a poner ya más atención a los sucesores de actuales compiladores como C++ de C, pues los últimos años de los 80's fueron testigo del arribo de extensiones de lenguajes orientados a objetos para DOS, como los compiladores Pascal y Modula-2, abriendo las puertas a otros como C++, SmallTalk y Actor armados con características de POO.

Aunque estos lenguajes poseían esas características, carecían de la capacidad para dibujar fácilmente objetos visuales y administrar la interacción con eventos externos. Asimismo, era una tarea difícil el aprendizaje y manejo de éstos para los programadores.

Por lo que, Microsoft introdujo Visual Basic como un ambiente de programación visual, basado en el lenguaje BASIC (Beginners All-Purpose Symbolic Instruction Code). Aunque Visual Basic carece de punteros, implementa un pequeño número de elementos de POO.

Con Delphi de Borland, no hay necesidad de decidir entre programación visual sin las características de la POO, o viceversa. Sino por el contrario, pues permite usar el lenguaje de Objetos de Pascal en un ambiente de programación visual con extensiones de POO en una forma pura.

Objetos

En Delphi como en otros ambientes visuales, los objetos son ítems como botones (buttons), etiquetas (labels), cajas de lista (list boxes), etc. Así, un objeto es un término definido ambiguamente que no ha sido utilizado para conceptualizar alguna cosa específica. Sin embargo, para propósitos de este trabajo se considerará al objeto como un componente que existe en una rutina.

Los ambientes de programación visual agregan una nueva dimensión para crear aplicaciones para permitir que los objetos sean dibujados en la pantalla. Sin la programación visual, este dibujado requeriría necesariamente de código, situación que en un momento dado puede volverse un tanto tediosa.

Así, gracias a las herramientas visuales, se puede trabajar con los objetos observando directamente los resultados en la pantalla. Ahora bien, si por ejemplo el objeto es cambiado de lugar, sus atributos (tamaño, posición, propiedades) son registrados en el código que lo soporta, sin importar por ende su ubicación.

Propiedades

Como ya se adelantó un poco en el apartado anterior, los objetos poseen ciertas propiedades, las cuales incluyen, entre otras cosas, ítems como color, ancho y posición. Así, éstas pueden afectar la apariencia de un elemento, pues tienen la función de describir sus detalles.

Por otro lado, para un objeto, las propiedades son algo como variables locales en un procedimiento, esto es, sólo pueden ser usadas por el mismo. Así, cuando se modifica una de ellas, sólo se afecta a ese proceso. De la misma manera, cuando se altera la propiedad de un objeto sólo se está afectando a éste.

Como ejemplo del funcionamiento de las propiedades en Delphi, considérese que se tiene el objeto de tipo Humano, el cual posee atributos como edad, nombre, sexo y dirección. Utilizando entonces el código objetos de Pascal[1], se puede manipular dicho objeto de la siguiente manera:

```
{ Actualizar la propiedad dirección de Humano }
Humano.Dirección:= 'Zandunga No. 43';
{ Actualizar la propiedad 'PuedeBeber' inspeccionando la edad }
If Humano.Age >= 18 then
   Humano.PuedeBeber:=true
Else
   Humano.PuedeBeber:=false;
```

Eventos

Las aplicaciones Windows emplean métodos dirigidos por el evento para administrar las interacciones entre el programa y el usuario. De tal manera que el ambiente incluye, tanto acciones del beneficiario, como del mismo entorno (Windows). Por lo que, la mayoría del código escrito en Delphi será provocado desde eventos generados por el usuario y por el sistema.

Para comprender las dinámicas de la PDE en Delphi, considérese una típica compañía y sus empleados. Cada trabajador tiene un título y una descripción de su labor, definiendo sus habilidades, responsabilidades y roles en las actividades diarias de la empresa, quien interactúa con el mundo exterior a través de una serie de eventos como llamadas telefónicas, correos, faxes, etc. Cada uno de ellos es manejado por el empleado adecuado. Por ejemplo, en el evento de que suena el teléfono de Bob, él responde, sin

[1] No se abundará sobre el lenguaje Pascal, en todo caso podría consultarse cualquier material bibliográfico referente a Pascal o el que se muestra en las Referencias en caso de que exista alguna duda en lo que respecta a estructuras de control o sintaxis.

embargo no necesita contestar a otra situación, sino la suya, así como otros no tienen que responder el teléfono de Bob.

Como en el ejemplo de la compañía, Delphi responde al mouse, a una tecla y a eventos del sistema, invocando las piezas apropiadas de la lógica o procedimientos de la aplicación. Ahora bien, si uno de ellos no se encuentra asociado a un evento dado éste es ignorado.

Para dar una idea general a cerca del ciclo de vida de un evento en Delphi, considérese el evento generado por el click del objeto button:

1. El evento es generado por el usuario cuando da un click al button llamado botonOK.
2. El objeto botonOK reconoce la acción como un evento que puede ser manipulado.
3. Delphi busca un procedimiento que coincida con el nombre del objeto que está siendo representado (botonOK) más el nombre del evento (Click). En este ejemplo Delphi ejecuta el procedimiento botonClick().

Esta lógica se encuentra representada usando la sentencia case en el siguiente código:

```
Case NuevoEvento of:
   Case Click:
      botonOKClick();
   Case Enter:
      botonOKEnter();
   Case Exit:
      botonExit()
Else:
      IgnorarEvent();
End;
```

En Delphi se puede fácilmente ligar un evento al código usando un *event handler*, el cual es simplemente una pieza de código de la aplicación, es decir, son las puertas que permiten ligar a éste con los eventos generados en el programa.

Por otra parte, todos los objetos generalmente responden a eventos.

IV.3 EL AMBIENTE GRÁFICO DE DELPHI

El entorno gráfico de Delphi ha sido diseñado pensando en que sea capaz de producir verdaderos resultados, proporcionando un conjunto flexible y moderno de herramientas que se pueden usar para crear aplicaciones rápidamente.

Así, el ambiente en Delphi se muestra en la Figura IV.3.1 en donde se puede observar que éste consiste básicamente de 4 ventanas que se abren cuando se carga Delphi por primera vez. Por lo que, a continuación se describirá brevemente cada uno de dichos entornos.

1) Ventana Principal (Main Window)

La Ventana Principal es el control central del ambiente gráfico de Delphi. A su vez, ésta contiene 3 distintos elementos, que ofrecen una funcionalidad única (Ver Figura IV.3.2), los cuales son:

a) Barra de Menús (Menu Bar)

Esta barra proporciona menús para todas las ventanas, asimismo permite manejar el diseño de la aplicación, manipular la interfase Windows, configurar el ambiente Delphi y obtener ayuda, entre otras cosas. Sólo es necesario darle un click y con ello se accesará a cada uno de los submenús.

b) Barra Rápida (Speed Bar)

La Barra Rápida aparece a la izquierda de la ventana principal. Proporciona un rápido acceso a las operaciones comunes que incluyen abrir y salvar un archivo, cortar y pegar un texto, ejecutar un programa, abrir y seleccionar formas, así como unidades.

c) Paleta de Componentes (Component Palette)

Esta paleta se localiza a la derecha de la ventana principal, permitiendo seleccionar las herramientas necesarias para el desarrollo de la aplicación, esto a través de su forma multipágina, pues en cada hoja se encuentran diversos componentes con diferente funcionalidad, dependiendo de lo que se desee incluir en el programa, así por ejemplo se hallan elementos sencillos como "labels"[2], para escribir algún texto; "buttons" para crear botones u otros más complejos como "DBgrid" para manipular bases de datos.

2) Ventana del Inspector de Objetos (Object Inspector Window)

La localización por default de esta ventana es en la parte izquierda inferior. Es una cara multipágina donde se puede visualizar fácilmente las propiedades y eventos que se encuentran disponibles para el objeto que se halla seleccionado dentro de la ventana de la forma (Ver Figura IV.3.1).

[2] Para evitar confusiones no se hará traducción para los objetos, sino que se respetará el manejo de Delphi.

Figura IV.3.1 Figura que muestra el entorno de Delphi cuando se carga por primera vez.

Figura IV.3.2 Figura que muestra los componentes de la ventana principal.

3) Ventana de la Forma (Form Window)

La Ventana de la Forma se encuentra ubicada en la parte derecha del área de trabajo, arriba del Editor de Código (Ver Figura IV.3.1). Dicha superficie contiene todo la parte gráfica que será visible para el usuario.

De tal manera que, cuando una aplicación es ejecutada, la ventana de la forma y los objetos visuales se convierten en el punto central de la aplicación. Así, ésta puede considerar una sola forma o varias de ellas, presentadas cada una de manera apropiada de acuerdo a como se haya diseñado.

4) Ventana del Editor de Código.

En esta ventana es donde se puede editar el código de la aplicación, localizándose en la parte derecha de la superficie de trabajo, por debajo de la cara de la forma (Ver Figura IV.3.1). Es aquí donde el programador hace uso de sus conocimientos de programación para diseñar y controlar la aplicación.

IV.4 ESTRUCTURA DE UNA APLICACIÓN EN DELPHI

Una aplicación en Delphi está compuesta por lo que es un Proyecto, donde a su vez recaen ciertos componentes denominados Formas y Unidades como se observa en la Figura IV.4.1.

De tal manera que, un Proyecto es un archivo (con extensión .DPR) que puede verse como el programa principal, pues es aquí donde se listan todas las Unidades y Formas usadas por el mismo. Así que cada vez que se agregue una Unidad o Forma al Proyecto, Delphi la adicionará debajo de la cláusula **uses**, tal como se muestra en la Figura IV.4.2.

Unidades

Las Unidades son los archivos del código fuente de la Aplicación (sobra mencionar que está en Pascal), manteniendo la extensión .PAS (como se observa en la Figura IV.4.1). Inicialmente, una Unidad consta de los siguientes elementos:

1) Encabezado (Unit Heading).

El encabezado de una Unidad especifica el nombre de la misma, siendo éste el que se emplea cuando es referenciada por otro módulo o por el mismo Proyecto en la cláusula **uses**. Cabe señalar que dicho nombre debe ser único, pues dos unidades no pueden ser usadas al mismo tiempo (Ver Figura IV.4.3).

Figura IV.4.1 Estructura de una Aplicación en Delphi.

Figura IV.4.2 Un Proyecto en Delphi.

2) Interfaz (Interface)
En esta parte es donde se declaran las constantes, los tipos, variables, así como procedimientos y funciones (únicamente los encabezados) que son públicas (ver Figura IV.4.4).

3) Implementación (Implementation)
En la parte que corresponde a la implementación se definen los cuerpos de todos los procedimientos y funciones que son públicos. Además se pueden declarar también constantes, tipos, variables, procedimientos y funciones que son privadas y no se encuentran disponibles para los usuarios de otra Unidad (ver Figura IV.4.5).

4) Inicialización (Inicialization)
La inicialización es la última parte de una Unidad, que consta de la palabra reservada **Inicialization**, seguida de una o más sentencias a ejecutarse. Así, independientemente de que haya parte de inicialización, deberá terminarse con un **end**. Por compatibilidad con las primeras versiones de Turbo y Borland Pascal, la palabra reservada **begin** puede sustituir a **Inicialization** (Ver Figura IV.4.6).

Formas

Una aplicación usualmente contiene múltiples Formas, de tal manera que, este componente constituye el centro de las aplicaciones Delphi.

Por lo que, uno puede diseñar una aplicación agregando otros elementos en una Forma para emplearla como ventana, caja de diálogo, o simplemente para introducir información.

Así, una Forma incluye funcionalidades estándar tales como:

1. Menú de Control
2. Botones para minimizar y maximizar
3. Barra de Títulos
4. Bordes Ajustables (Ver Figura IV.4.7)

Como ya se mencionó, una Forma es un componente que posee propiedades y eventos que permiten personalizar la aplicación. Así, entre los atributos más comunes se encuentra **Caption**, con la que se le puede dar un título a la misma, por ejemplo:

```
Forma1.Caption:='Menú Principal';
```

y entre los eventos se puede hallar **ActiveControl**, que significa que mientras la Forma se encuentre activa se realizará cierta acción, como:

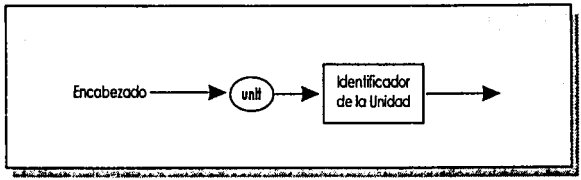

Figura IV.4.3 Figura que muestra el encabezado de una Unidad.

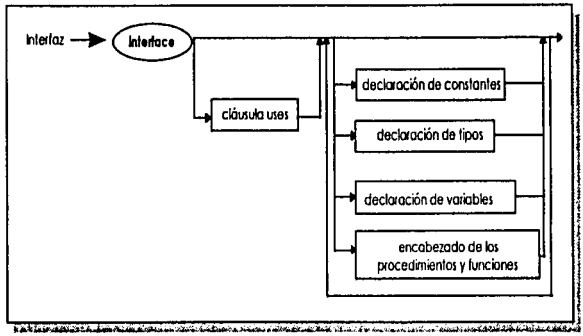

Figura IV.4.4 Figura que muestra la Interfaz de una Unidad.

Figura IV.4.5 Figura que muestra la implementación de una Unidad.

Figura IV.4.6 Figura que muestra la inicialización de una Unidad.

Figura IV.4.7 Figura que muestra una Forma.

```
Procedure TForm1.Forma1ActiveControl(Sender: Tobject)
                    Begin
                       Forma1.Caption:='Menú Principal';
                    End;
```

así se puede observar la estructura de un procedimiento, donde TForm1 significa la clase[3] de la Forma1, luego el nombre de esta, después el evento y, lo que está entre paréntesis es un parámetro que Delphi da por default. A continuación viene el bloque cerrado entre "Begin" y "End" con su clásico ";" del lenguaje Pascal.

IV.5 ALGUNOS COMPONENTES

Como ya se anticipó, en Delphi un objeto es tratado como un componente, el cual puede ser visual o no visual. La diferencia entre estos radica en que el primero puede visualizarse, tanto en la etapa de diseño de la aplicación, como en la fase de ejecución. En el otro caso, únicamente es visible en el primer paso, sin por ello dejar de intervenir en ambas facetas. Por lo que, en esta sección se mostrarán algunos de ellos, quizá los más comunes y sencillos, pues como se dijo al principio de este capítulo, no se pretende dar un curso rápido de Delphi, sino mostrar algunas de sus características, asimismo se resaltarán algunas propiedades o eventos de algunos componentes, no de todos, recordando que se puede consultar el material que se encuentra en las Referencias.

Primeramente se comenzará con los componentes de tipo visual, así la mayoría de éstos son utilizados para introducir datos, que a su vez va a depender del tipo de aplicación que se esté tratando. Entre los más comunes se encuentran los siguientes:

1) Componente **Label**.

El componente **Label** es bastante útil pues permite desplegar algún texto en la aplicación, lo cual es muy importante porque proporciona información al usuario y con ello, le facilita la comunicación, dándole también un ambiente agradable (Ver Figura IV.5.1).

Entre algunas de sus propiedades se encuentra el atributo **visible**, que hace posible que el usuario vea el texto. Así, esta cualidad posee 2 valores true (verdadero) y false (falso), por ejemplo:

```
Label1.visible:=true;
```

[3] Para más información sobre las clases en Delphi, ver Matcho, J.; et. al. Special Edition Using Delphi, págs 73-74.

Figura IV.5.1 Figura que muestra el componente Label utilizando la propiedad caption.

De tal manera que, cuando toma el valor true, el usuario ve el texto, en caso contrario simplemente no aparece nada.

Otra de sus propiedades la constituye la característica **caption**, la cual permite al desarrollador personalizar el texto que desea incluir en la aplicación, así lo puede modificar desde el código tantas veces como se requiera, por ejemplo:

```
Label1.caption:='Hola, esta es mi Aplicación';
```

Asimismo, otro atributo muy común es la propiedad **font**, que permite modificar el tipo, tamaño, estilo y color de la letra del texto.

2) Componentes **Edit**, **MaskEdit** y **Memo**.

Estos 3 componentes son utilizados para la introducción y despliegado de texto. Las principales diferencias entre estos es que por ejemplo el componente **Memo** permite líneas múltiples, mientras que los elementos **Edit** y **MaskEdit** sólo una.

Así, entre estos 2 últimos existen también desemejanzas, una de ellas es que el componente **MaskEdit** permite validar los datos que se introducen, sin necesidad de programar (Ver Figura IV.5.2), esto a través de la propiedad **mask**, directamente en el Inspector de Objetos.

Entre las propiedades más comunes, para estos dos objetos, se encuentra el atributo **text**, que es similar a caption del componente Label, pues permite desplegar texto, por ejemplo:

```
Edit1.text:= 'Hay 3 componentes';
```

Otra de sus cualidades que poseen (Edit y MaskEdit) es que los podemos alinear desde el código como se desee, utilizando para ello la propiedad **top**, como se muestra en la línea que sigue:

```
MaskEdit.top:=250;
```

Asimismo, es posible habilitar y deshabilitar cualquiera de los 3 componentes por medio del atributo **enabled**. Así, éste posee 2 valores, true y false, de tal forma que si toma true, se puede introducir información, en caso contrario, no se tendrá acceso a modificar la entrada de los datos, por ejemplo:

```
Memo1.enabled:=false;
```

Figura IV.5.2 Figura que muestra las propiedades y funciones de los componentes **Edit**, **MaskEdit** y **Memo**.

3) Componentes **Button** y **BitBtn**

Los componentes **Button** y **BitBtn** son muy empleados en cajas de diálogo (dialog boxes). Son los botones que comúnmente se ven en el ambiente Windows como "Aceptar", "OK" o "Cancelar". Sólo que existe una diferencia entre ambos elementos, pues al componente **BitBtn** se le puede agregar algún gráfico a través de la propiedad **glyph** (Ver Figura IV.5.3).

Una de las propiedades común a los 2 objetos (y en general a todos) es el atributo **name**, pues permite darle un nombre al componente, el cual se utilizará a la hora de programar, así por ejemplo en la línea de abajo se nombra "Boton" al elemento Button1:

```
Button1.Name:='Boton';
```

Entre los eventos más frecuentes para este componente se encuentra **OnClick**, el cual sucede cuando el usuario da un click al **Button** o al **BitBtn**, por ejemplo:

```
Procedure TForm1.BotonClick(Sender: Tobject)
```

entonces se ejecuta el bloque que se encuentre en el procedimiento.

4) Componentes **RadioButton** y **CheckBox**

Existen 2 componentes que son muy útiles para desplegar e introducir datos en un contexto visual como son los elementos **RadioButton** y **CheckBox**.

Normalmente se encuentran agrupados y encerrados en un marco. Se utilizan cuando existen varias opciones y se tiene que elegir, sólo que hay una diferencia entre ambos objetos, pues en el caso de los RadioButton, únicamente se puede seleccionar una entre las distintas alternativas, mientras que en los CheckBox es factible escoger una o más opciones (Ver Figura IV.5.4).

5) Componente **ListBox**

El componente **ListBox** es muy útil cuando se desea presentar una lista de ítems para selección. Así, cuando la relación es muy larga, cuenta con un "scrollbar" (Ver Figura IV.5.5) que permite deslizarse de principio a fin, según como el usuario lo necesite.

Por otra parte, entre su funcionamiento puede destacarse las tareas de agregar, borrar u ordenar los ítems de una manera dinámica como la aplicación lo vaya requiriendo.

Figura IV.5.3 Figura que muestra los componentes **Button** y **BitBtn**.

Figura IV.5.4 La Figura muestra ejemplos para los Componentes **RadioButton** y **CheckBox**.

6) Componente **ComboBox**

El componente **ComboBox** es similar al objeto list box, pues también despliega una lista de items. Sin embargo, posee una diferencia, ya que posee un pequeño espacio en la parte superior, mostrándose sólo el elemento que ha sido seleccionado o el que está colocado por default (Ver Figura IV.5.5).

El siguiente código muestra un ejemplo de como se inicializaría un texto (la primera línea) en un ComboBox si éste es blanco.

```
Procedure TForm1.ComboBox1Enter (Sender: Tobject)
Begin
   If ComboBox1.text:= ' ' then
      ComboBox1.text:=ComboBox.items.string[1];
End;
```

Hasta ahora se ha hablado únicamente de componentes visuales, por lo que se dará paso a explicar algunos de los que no son visibles a la hora de la ejecución de la aplicación.

7) Componente **Menu**

Entre los objetos de este tipo que son más utilizados se encuentra el componente **Menu**, pues permite crear menús, los cuales aparecen exactamente como fueron diseñados en esta etapa.

Así, después de dibujar el componente sobre la forma, simplemente con un doble click se accesa al "Menu Designer", donde se puede esbozar el menú de la aplicación (Ver Figuras IV.5.6 y IV.5.7).

8) Componentes para Cajas de Diálogo

El ambiente Windows soporta un conjunto de diálogos comunes que son usadas para diversas tareas como abrir y guardar archivos, seleccionar fuentes, seleccionar y configurar colores e impresoras. De tal manera que, Delphi proporciona componentes no visuales que soportan cajas de diálogos como:

Open, Save, Font, Color, Print, Printer Setup, Find, Replace

La Figura IV.5.8 muestra los componentes para estas cajas de diálogo[4].

[4] Solamente se mostrarán los componentes, pues las cajas de diálogo ya son conocidas en el ambiente Windows.

Figura IV.5.5 Figura que muestra los componentes **ListBox** y **ComboBox**.

Figura IV.5.6 Figura que muestra el Componente **Menu**.

Figura IV.5.7 La Figura muestra la forma de crear y diseñar un menú utilizando el Inspector de Objetos.

Figura IV.5.8 La Figura muestra los diversos componentes para diferentes cajas de diálogo estándar que proporciona Delphi.

Por otra parte, Delphi también proporciona herramientas para construir diálogos personalizados similares a las estándar, para lo cual puede consultarse el material de las Referencias.

IV.6 LAS BASES DE DATOS EN DELPHI

Las bases de datos proporcionan un medio genérico para almacenar datos. Así, un dispositivo de base de datos ofrece el mecanismo para manipular y visualizar la información existente en ellas, de otro modo, los programadores se verían forzados a escribir rutinas complejas para trabajar con archivos siempre que hubiera necesidad de accesar datos de una manera eficiente. En esencia el programador sería responsable de crear tanto la base de datos como el manejador de ésta.

Delphi tiene integrado un poderoso manejador de bases de datos. Hay componentes visuales que permiten accesar tablas y métodos para operar con registros que hacen que la manipulación de la base de datos sea tan sencilla como crear otro tipo de aplicación.

Modelos de Base de Datos

En muchas aplicaciones que utilizan bases de datos, la complejidad y funcionalidad se relaciona directamente con el diseño físico e infraestructura de la base de datos, por lo que a continuación se explorarán las 3 clases principales de estas.

1) Bases de datos independientes

Las bases de datos independientes son las más sencillas de manejar, pues tanto estas como su manejador residen en la misma máquina, haciendo que el diseñador no tenga que preocuparse de que dos personas intenten modificar el mismo registro a la vez. Así, este tipo de bases de datos es útil en el desarrollo de aplicaciones que se distribuyen a muchos usuarios, en donde cada uno mantiene una base de datos por separado.

2) Bases de datos de archivos compartidos

Una base de datos de archivos compartidos es casi igual a una de tipo independiente, con excepción de que puede ser accesada por diversos usuarios a través de una red. Esto permite una mayor accesibilidad, ya que la base de datos puede ser manipulada por diferentes máquinas. Sin embargo, no es conveniente cuando se necesita realizar un elevado volumen de cómputo y accesos simultáneos a la base de datos. Para esto, la solución es un diseño cliente/servidor.

3) Bases de datos cliente/servidor

La solución de término elevado para accesar bases de datos se resuelve a través del modelo cliente /servidor. En este caso una máquina dedicada, o servidor, se encarga de realizar la entrada a la base de datos para un grupo de clientes. En este diseño, el cliente

solicita al servidor que realice una tarea específica. Aunque existen ventajas de desempeño y flexibilidad en una arquitectura de este tipo, también tiene sus desventajas, pues por lo general son mucho más caras que las de archivos compartidos. Además el software cliente/servidor necesita de un protocolo como TCP/IP, para llevar a cabo una conversación.

Es muy frecuente que la elección del modelo correcto de base de datos para una aplicación sea una tarea difícil. Por fortuna el manejador de base de datos de Borland es lo suficientemente flexible para cambiar de base de datos con poco esfuerzo. Si esto no le da suficiente flexibilidad, el manejador de base de datos puede comunicarse con ODBC (más adelante se verá), que a su vez puede vincularse con casi todas las bases de datos en el mercado actual.

Creando y Accesando Tablas

Cuando se crea una aplicación con bases de datos, se accesa al "Desktop Database Engine" (BDE) que se encuentra instalado automáticamente cuando se instala Delphi.

Así, el Database Desktop (DBD) es como una miniversión de Paradox o dBASE para Windows, pues proporciona un método para crear, visualizar, editar, reestructurar, indexar, ordenar, cuestionar y manipular tablas[3], y que se encuentra en el menú "Tools" en la ventana principal. Asimismo, constituye un programa MDI (Multi-Document Interface: Interfaz de Multidocumentos), pues cada ítem que se abre es colocado en una ventana hija que siempre se encuentra dentro de la ventana madre del DBD, de tal manera que se pueden tener varias tablas abiertas y activas al mismo tiempo (Ver Figura IV.6.1).

Como ya se dijo, Delphi (al igual que otros manejadores) trabaja con tablas, de forma que, para crear una de ellas simplemente hay que seguir los menús y mensajes. Así, por ejemplo para esta operación, se abre el menú "File" (del DBD) en "New" y entonces se elige "Table" (Ver Figura IV.6.2), enseguida de esto se despliega la ventana "Table Type" y ahí se elige el tipo que se quiere generar, como Paradox, o dBASE (Ver Figura IV.6.3). Posteriormente, se abre una nueva ventana en la que se define su estructura, esto es, los nombres de los campos y su tipo (numéricos, binarios, long integer, float, character, etc.), el cual va a depender de la clase de tabla que se haya decido construir. Ahora bien, si se desea crear índices, también en esta herramienta se pueden definir (Ver Figura IV.6.4).
Por otra parte, en el caso que lo que se decidiera es abrir y no crear una tabla, en el menú "File" (del DBD) existe la opción "Open" para ello, o simplemente haciendo click en el icono correspondiente. Posteriormente, independientemente de cualquiera de las 2 formas de

[3] Una tabla es simplemente un archivo de datos que es miembro de una base de datos.

Figura IV.6.1 Figura que muestra el DBD con varias tablas abiertas de distintas plataformas.

Figura IV.6.2 Primer paso para crear una tabla.

Figura IV.6.3 Ventana "Table Type" donde se elige el tipo de Tabla a crear.

Figura IV. 6.4 La Figura muestra el lugar donde se define la estructura de una base de datos del tipo dBASE IV.

hacerlo, aparece una caja de diálogo donde se puede elegir aquella tabla que se desea accesar (Ver Figuras IV.6.5 y IV.6.6).

Algunos Componentes

Los componentes para bases de datos se encuentran en 2 páginas de la paleta de componentes, esto es "Data Access" y "Data Controls" (Ver Figura IV.6.7).

Así, entre los elementos que más se utilizan se encuentran los siguientes.

1) Componente Table

El componente Table es el primer elemento usado cuando se conecta una aplicación con los datos. Por lo que, una vez teniendo, una Tabla (creada o no en Delphi), puede accesarse directamente a ella utilizando sus propiedades. En la Figura IV.6.8 se muestran algunas de estas.

Como puede observarse en la Figura, algunas de sus propiedades más utilizadas son DatabaseName, la cual se emplea para ubicar a la base de datos, es decir puede ser un subdirectorio o un alias. Otros atributos que también se emplean son IndexName y TableName, siendo el primero para el nombre de un índice (si existe) y el segundo, el nombre de la Tabla, respectivamente.

2) Componente DataSource

El componente DataSource constituye el enlace entre la base de datos y los controles en las formas.

Así en la Figura IV.6.9 se muestra este elemento con sus propiedades. Entre las más comunes se encuentra DataSet, en el que se inscribe el nombre que se le asignó al componente Table, con lo cual se ligan ambos objetos.

3) Componente DBGrid

El componente DBGrid es una especie de cuadrícula en la que se puede introducir, insertar, borrar y desplegar información que se encuentra en una en una Tabla de una base de datos (Ver Figura IV.6.10). De tal manera que, utilizando sus propiedades se facilita la tarea de programación, pues solamente hay que emplear la propiedad DataSource para ligar el DBGrid con la Tabla.

4) Componente DBNavigator

El componente DBNavigator es un elemento que permite moverse a través de los datos de una tabla en una base de datos. De tal forma que permite además, ejecutar labores como insertar, editar, borrar, entre otras. Usualmente se emplea junto con el objeto DBGrid (Ver

Figura IV.6.5 Forma de accesar una tabla a través del menú.

Figura IV.6.6 Forma de accesar una Tabla por medio de un icono.

Figura IV.6.7 Figura que muestra las 2 páginas de componentes para bases de datos.

Figura IV.6.8 Figura el componente Table con sus propiedades en el Inspector de Objetos.

Figura IV.6.9 Figura que muestra el componente **DataSource** con sus propiedades.

Figura IV.6.10 Figura que muestra el componente **DBGrid** en ejecución.

Figura IV.6.11). Así, para enlazar el componente DBNavigator, se emplea la propiedad DataSource.

Existen otros componentes como DBEdit, DBListBox, DBComboBox, DBMemo, DBCheckBox, DBRadioButton que también son usuales, sólo que funcionan exactamente igual como los ya mostrados en la sección IV.4.5, pero manipulando bases de datos, por lo que una explicación sobre éstos sobraría (Ver Figura IV.6.12).

Conectividad Abierta de Base de Datos (ODBC)

El manejador de base de datos de Borland puede comunicarse con muy diferentes bases de datos. Esto resulta útil si en un momento dado el desarrollador desea cambiar las estructura de una base de datos. Así, para flexibilidad y compatibilidad con casi cualquier base de datos, está disponible ODBC.

ODBC son las siglas de "open database connectivity" (Conectividad abierta de base de datos). ODBC permite comunicarse con cualquier base de datos a través de una interfaz común, utilizando lo que se conoce como un manejador ODBC. Este manejador contiene un código que comprende los aspectos específicos de una base de datos en particular y proporciona acceso a ella mediante un conjunto estándar de llamadas API.

El uso de ODBC le da a una aplicación mayor flexibilidad, ya que cambiar su tipo de base de datos sería tan sencillo como modificar el manejador ODBC.

A menudo, un desarrollador querrá crear una aplicación pero no podrá escalarla inicialmente. ODBC le permite usar una base de datos independiente basada en archivo para el desarrollo y después cambiar al modelo cliente/servidor sin hacer grandes modificaciones al código fuente. Es útil también al crear una aplicación Delphi que requiera ser compatible con los datos existentes. Por ejemplo, si un sistema está en un servidor Sybase SQL, y se necesita construir un proceso frontal para el acceso de datos, Delphi puede emplear ODBC para conectarse y comunicarse con la base de datos. Otro ejemplo de compatibilidad con antecedentes es cuando se trata de portar a Delphi algunas aplicaciones escritas en un lenguaje diferente.

En la terminología de ODBC, el manejador ODBC es la biblioteca que comprende cómo comunicarse con la base de datos fundamental. Los manejadores ODBC a menudo están disponibles con el proveedor de la base de datos o con compañías distribuidoras de software.

Para usar un manejador ODBC y comunicarse a través de ODBC con Delphi, se necesita crear un nuevo manejador de base de datos de Borland. Nótese que esto pudiera parecer confuso, ya que ODBC tiene manejadores ODBC que son manejadores de datos y el manejador de base de datos de Borland tiene manejadores internos, de manera que un manejador de base de datos de Borland debe configurarse para accesar un manejador ODBC.

Figura IV.6.11 Figura que muestra el componente **DBNavigator** en ejecución.

Figura IV.6.12 Figura que muestra algunos componentes como **DBEdit**, **DBMemo** y **DBRadioButton**.

CAPITULO V

APLICACIÓN:
UN SISTEMA EXPERTO DE PRODUCCIÓN PETROLERA

CAPÍTULO V
APLICACIÓN: UN SISTEMA EXPERTO DE PRODUCCIÓN PETROLERA

V.1 CONTEXTO DEL PROBLEMA

En estos tiempos parece realmente necesario emplear herramientas innovadoras y de vanguardia para la solución de problemas, ofreciendo además, nuevas alternativas en la toma de decisiones y permitir que éstas se efectúen con un mayor grado de conocimiento o certeza. Así, la Inteligencia Artificial constituye una de tales técnicas que, a pesar de ser muy joven, ya ha proporcionado muchos frutos en diversas áreas del saber.

De ahí el interés por desarrollar un Sistema Experto aplicado a un campo tan productivo como la Ingeniería Petrolera, empleando específicamente para ello, una base de datos con Registros Geofísicos Petroleros. Sin embargo, dicho sistema requiere no sólo de información, sino conocimiento que pueda llegar a ser almacenado (Base de Conocimientos), pues éste no se encuentra conformado de hechos aislados, por el contrario precisa de acontecimientos con alguna relación entre sí, es decir, experiencia que un especialista en el área puede poseer. No obstante, en muchas ocasiones, no es posible contar con la presencia de un experto humano que pueda orientar en ese sentido el desarrollo de un Sistema Experto.

Dada esa situación, se tiene que recurrir a otros procesos inteligentes, que permitan obtener automáticamente el conocimiento, a partir de una serie de datos o ejemplos de situaciones históricas, como los Métodos Inductivos. Así, en este caso, se utilizó un modelo de Red Neuronal, denominado Teoría de la Resonancia Adaptativa (ART-2, por sus siglas en inglés) el cual ya fue descrito en el capítulo II (ver Figura V.1.1).

Retomando los fundamentos de los capítulos teóricos (I a IV), se tomó la decisión de emplear como herramienta de desarrollo para el Sistema Experto, el lenguaje de programación orientado a objetos Delphi de Borland, debido a sus características como poseer una gran variedad de elementos para el desarrollo visual de aplicaciones, crear con facilidad entornos amigables propiciando la interacción usuario-programa y trabajar con código en Pascal, entre otras (ver capítulo IV para una descripción más detallada), lo cual hace que dicho lenguaje se adecue perfectamente a las necesidades requeridas y objetivos del Sistema Experto.

Así, el propósito del Sistema Experto, denominado de Producción Petrolera, lleva como objeto diagnosticar el volumen de hidrocarburos presente en un yacimiento, manejando además un grado de certeza expresado a través de un valor de ocurrencia todo ello a partir de un conjunto de características del mismo pozo.

Figura V.1.1 Figura que muestra de manera gráfica el proceso desde que se tiene una Base de Datos hasta llegar a la construcción del Sistema Experto.

V.2 DESARROLLO

En esta sección se explicará de manera concisa la forma en la que se desarrolló el Sistema Experto, sin dejar pasar por ello, su funcionamiento. Así, dicho sistema se encuentra conformado básicamente por 4 partes (más adelante se describirán, ver Figuras V.2.1 y V.2.2), donde cada una de ellas tiene definida su propia tarea:

- ESTRUCTURACIÓN DE LA BASE DE CONOCIMIENTOS
  1) Base de Datos con Registros Geofísicos
  2) Inducción
  3) Base de Conocimientos
- MÓDULO GENERADOR DE REGLAS
  1) Edición de Reglas
  2) Modificación de Reglas
  3) Adición de Reglas
- MÓDULO PARA CONSULTA
- MOTOR DE INFERENCIA

ESTRUCTURACIÓN DE LA BASE DE CONOCIMIENTOS

La Base de Conocimientos juega un papel muy importante en el desempeño de un Sistema Experto, sin embargo, en ocasiones su obtención puede volverse difícil y complicada. En este caso en particular, ésta fue producto de toda una etapa de procesamiento (ver Figura V.1.1). Así, este apartado se encuentra destinado a explicar los factores que permitieron su conformación (ver Figura V.2.3).

1) Base de Datos[1] con Registros Geofísicos Petroleros

El petróleo o gas que se produce en la actualidad, proviene de depósitos que se encuentran en el espacio poral de las rocas de una reserva. De ahí que, para explotar un yacimiento es necesario llevar a cabo estudios de muy diversa índole, entre ellos las prospecciones físicas, es decir, exploraciones y mediciones (Registros) hechas al subsuelo.

[1] Recuérdese que, una base de datos es una colección de datos que se hallan guardados en archivos, organizados de tal manera que, faciliten el almacenamiento, modificación y obtención de la información. Piatetsky-Shapiro, Gregory (1991). Knowledge Discovery in Databases. Pág. 5.

Figura V.2.1 Figura que muestra la estructura completa del Sistema Experto de Producción Petrolera

Figura V.2.2 Figura que muestra la pantalla de inicio del Sistema Experto de Producción Petrolera, donde se muestran deshabilitadas sus 4 partes básicas.

Figura V.2.3 Figura en la que se presenta la ubicación estructural de la Base de Conocimientos en el Sistema Experto.

basadas en el análisis de las características (físicas) que presenta el terreno, así como las variaciones superficiales de los campos naturales de la Tierra.

De esta manera, para poder evaluar la posibilidad de que un reservorio produzca hidrocarburos es fundamental conocer las diferentes características y dimensiones del yacimiento. Por lo que, en este caso en particular, se empleó una base de datos con Registros Geofísicos Petroleros, conteniendo información obtenida por ingenieros especialistas en este campo, de la cual se eligieron 10 atributos de la reserva como premisas, para que a partir de ellas se determinara la cantidad de hidrocarburos presente en dicho reservorio.

Así, la estructura que presenta esta base de datos con Registros Geofísicos es la que se presenta a continuación:

Descripción del pozo	Nombre del campo	Tipo	Tamaño	Decimales
Rayos Gamma	RAYOS_GAMA	N	12	8
Porosidad	POROSIDAD	N	12	8
Saturación de Agua	SATURA_H2O	N	12	8
Volumen de Arcilla	VOLUM_ARCI	N	12	8
Resistividad de la Formación	RESIS_FORM	N	12	8
Exponente de Cementación	EXPO_CEME	N	12	8
Resistividad de la Roca	RESIS_ROCA	N	12	8
Diámetro de Invasión	DIAM_INVA	N	12	8
Resistividad del Agua	RESIS_H2O	N	12	8
Profundidad de la Formación	PROFUNDIDA	N	12	8
Saturación de Hidrocarburos	**SATURA_HC**	**N**	**12**	**8**

En la Figura V.2.4 se muestra un fragmento de una base de datos con estas características.

2) Inducción

Como ya se ha hecho referencia, el Sistema Experto empleó un método inductivo basado en el modelo neuronal ART-2, que le permitió obtener su conocimiento. Dicho programa fue desarrollado en FoxPro y es ejecutado desde el mismo Sistema Experto. Cabe señalar que, este proceso se encuentra localizado en la parte designada como "Inducción" del menú del Sistema (ver Figura V.2.5), por lo que sólo hay que hacer un "click", para que éste se realice. Sin embargo, para tal efecto, se requiere que los usuarios posean la clave de acceso que se solicita, esto para mayor seguridad de la Base de

Microsoft FoxPro - Demopmx1

File Edit Database Record Program Run Browse Window Help

Volum_arci	Resis_lodm	Espe_come	Resis_roca	Diam_inva	Resis_h2o	Profundida	Saturac_hc
0.08700000	142.53600000	-1.33200000	0.27400000	30.41400000	1.05500000	3119.0950000	0.33200000
0.01900000	137.11500000	-0.97700000	0.02700000	54.07700000	11.07000000	3061.4870000	0.43200000
0.16000000	1209.6280000	2.18300000	7.09000000	0.00000000	0.09300000	2612.2120000	0.26500000
0.06100000	166.40000000	0.79300000	0.19800000	90.81300000	4.75500000	3012.4150000	0.38400000
0.12800000	46.91200000	39.99700000	0.20000000	38.12800000	0.82500000	2851.7850000	0.32200000
0.11400000	561.87300000	4.26100000	1.86100000	0.00000000	0.55200000	2529.9160000	0.29700000
0.08300000	17.37600000	-3.07500000	0.03200000	37.18700000	0.37500000	2716.7580000	0.31900000
0.11800000	146.99500000	-0.30.000000	0.47900000	129.98700000	0.14700000	2867.3300000	0.29700000
0.08100000	41.29800000	-2.19100000	0.07300000	61.26300000	0.46600000	2683.2300000	0.32100000
0.04200000	338.08700000	0.40800000	0.19800000	37.57700000	0.42200000	2939.5670000	0.37500000
0.10200000	29.59200000	-4.11100000	0.07800000	43.64100000	0.48500000	2733.2180000	0.30600000
0.00200000	64.79400000	-2.15500000	0.00500000	80.56300000	21.82600000	3053.9670000	0.43500000
0.12500000	17.52500000	58.00000000	0.06300000	24.94800000	0.98600000	2585.3300000	0.29800000
0.11500000	90.35300000	-0.51800000	0.28300000	47.73800000	0.05100000	2642.9970000	0.29100000
0.07900000	190.29400000	-0.03600000	0.31300000	16.44300000	0.75300000	3102.6350000	0.33700000
0.10400000	186.05700000	-0.06300000	0.48100000	40.16200000	0.32300000	3098.6730000	0.30900000
0.01400000	55.94300000	12.62500000	0.00900000	35.48500000	38.67100000	2918.8410000	0.45800000
0.06700000	88.69000000	-1.11000000	0.11200000	55.65700000	0.67900000	2898.4190000	0.34800000
0.16700000	39.09100000	-1.70200000	0.23000000	31.74400000	0.07000000	2921.6100000	0.25100000
0.06000000	60.84100000	-5.91900000	0.06400000	58.67700000	2.27600000	2728.9510000	0.36200000
0.03800000	66.01400000	-0.47200000	0.03400000	62.43000000	0.09800000	2722.5500000	0.37700000
0.21100000	169.78200000	1.84900000	1.77400000	0.00000000	0.33200000	2506.4470000	0.27000000
0.02600000	914.17300000	2.03600000	0.26500000	25.43400000	3.26800000	2979.1910000	0.41800000
0.09300000	42.74600000	4.72700000	0.01500000	36.90800000	2.74300000	2718.8920000	0.34100000

Demopmx1 Record 1/300 Exclusive Ins Num

Figura V.2.4 Figura que muestra un fragmento de una base de datos con Registros Geofísicos Petroleros.

Figura V.2.5 Figura en la que se presenta el módulo de "Inducción".

Conocimientos y así, evitar pérdida o modificación de información no deseada (ver Figura V.2.6).

El método ART-2, es una Red Neuronal que requiere de ciertos valores de entrada como lo muestran las Figuras V.2.7, V.2.8 y V.2.9. En la primera representación se le proporciona al usuario la oportunidad de elegir el archivo que contiene la base de datos con los Registros Geofísicos, esto por medio de una caja de diálogo; posteriormente se requiere la introducción de un valor para el umbral, lo cual puede hacerse de manera manual o, cederle al programa (algoritmo neuronal) la opción de que éste lo calcule, pues en él se incluye una heurística que permite sugerirle uno. Para terminar con la inserción de información, se le solicitan otros datos como el número de instancias, es decir, el número de características que se desea diagnosticar[2], así como el valor para depurar centroide[3].

De esta forma, el programa se encuentra listo para ejecutarse y generar el archivo que contiene las Reglas, el cual es una base de datos denominada REGFF.DBF, como la que se muestra en la Figura V.2.10.

3) Base de Conocimientos

Una vez que los datos primarios (Registros Geofísicos Petroleros) han sido procesados, se genera una nueva base de datos, conteniendo el conjunto de Reglas de Producción que conformarán la Base de Conocimientos para el Sistema Experto.

En cuanto a la estructura de la misma, ésta se ve modificada, pues el atributo que se pretende estudiar (volumen o saturación de hidrocarburos) desaparece, dando origen a 2 nuevos campos, y de esta manera, presentarlo por medio de un intervalo; agregándosele además otra columna, la cual constituye el factor de certidumbre que acompaña a la Regla (véase Capítulo III). Un aspecto que hay que resaltar es que, este factor es un valor muy importante, pues se utiliza para jerarquizar las Reglas de Producción, es decir, se lleva a cabo una ordenación en base a esta cantidad, colocando en el primer sitio aquella que posee el valor más grande, siguiéndole sucesivamente las demás en orden descendente.

Referente a la estructura que presenta la Base de Conocimientos, ésta se muestra en seguida:

---

[2] En este caso se trabajó con valor de "1".

[3] Este valor se refiere al número de elementos por grupo que la Red Neuronal considerará como válido, así si el valor es 2 significará que el mínimo de componentes presente en cada agrupación será de 3, ignorando los que contengan 2 o menos.

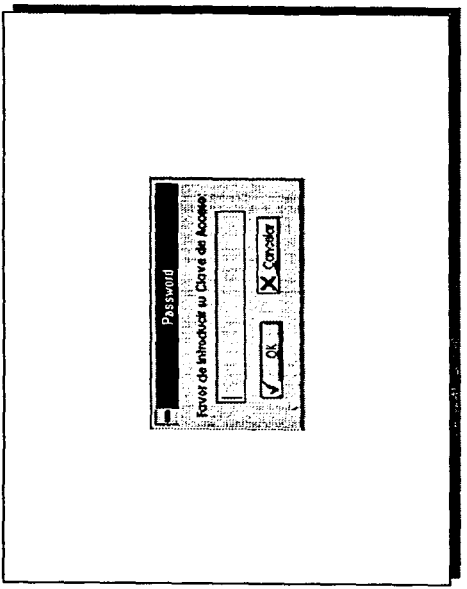

Figura V.2.6 Figura que exhibe la caja de diálogo para introducir clave de acceso.

Figura V.2.7 Figura que muestra el método ARI-2 en ejecución, solicitando el nombre del archivo que contiene los Registros Geofísicos.

Figura V.2.8 Figura que presenta el método ARI-2 en ejecución, solicitando el tipo y valor de umbral.

Figura V.2.9 Figura que muestra el método ART-2 en ejecución, solicitando algunos valores como número de instancias (deshabilitado, pues se trabajó con valor fijo de "1") y elementos para depuración.

Microsoft FoxPro - Regfl

File Edit Database Record Program Run Browse Window Help

Expo_como	Resis_roca	Diam_inva	Resis_h2o	Profundida	Inst_menor	Inst_mayor	Ocurrencia
48.52000000	0.26700000	77.91600000	4.02200000	2736.2680000	0.230000	0.360000	
39.99700000	0.68500000	64.26100000	10.89400000	2856.3570000	0.230000	0.430000	16
5.77300000	1.57000000	67.35100000	0.43500000	2495.1690000	0.000000	0.240000	13
13.53700000	0.45200000	49.36900000	1.69800000	2602.7630000	0.270000	0.350000	13
38.05100000	0.43600000	73.81600000	5.42600000	2549.4230000	0.260000	0.360000	13
19.59500000	0.28300000	84.54400000	1.95400000	2669.2100000	0.260000	0.350000	11
7.30700000	0.21100000	83.84800000	44.39200000	3084.0420000	0.340000	0.460000	10
23.98300000	0.20300000	59.03300000	1.93400000	2654.5790000	0.240000	0.340000	10
16.22900000	3.02100000	55.65400000	7.84100000	2941.3960000	0.100000	0.420000	9
0.00000000	0.30000000	42.30400000	4.03900000	2790.8250000	0.140000	0.400000	9
0.79300000	0.19800000	104.21100000	6.36600000	3026.7400000	0.380000	0.420000	7
0.00000000	1.17300000	23.00300000	0.13300000	3184.6260000	0.100000	0.250000	7
1.99100000	0.09300000	81.13400000	35.67400000	3061.4870000	0.380000	0.450000	6
10.58900000	0.26100000	96.09100000	3.76700000	2720.7210000	0.320000	0.410000	6
0.00000000	0.43400000	30.41400000	1.05500000	3148.6600000	0.290000	0.330000	5
8.47800000	1.92400000	0.00000000	0.19500000	2489.6830000	0.050000	0.210000	5
0.02800000	1.53500000	40.16200000	1.08100000	3109.9510000	0.280000	0.340000	4
19.58600000	0.05800000	35.49500000	56.09100000	2926.1560000	0.370000	0.460000	4
5.88100000	0.17000000	74.78700000	21.45200000	2957.8550000	0.330000	0.440000	4
11.20300000	0.52000000	129.98700000	1.29500000	2934.0810000	0.300000	0.330000	4
0.00000000	0.12100000	42.00700000	5.17500000	2987.7260000	0.390000	0.410000	4
1.29700000	0.54300000	129.98700000	1.31500000	2869.1590000	0.290000	0.330000	3
1.28800000	0.19800000	50.22800000	3.46200000	2939.5670000	0.380000	0.410000	3
6.87400000	0.26500000	35.52700000	17.59300000	2979.1910000	0.420000	0.450000	3

Regfl Record 1/99 Exclusive Ins Num

Figura V.2.10 Figura en la que se presenta un fragmento de una base de datos que contiene Reglas de Producción.

Descripción del pozo	Nombre del campo	Tipo	Tamaño	Decimales
Rayos Gamma	RAYOS_GAMA	N	12	8
Porosidad	POROSIDAD	N	12	8
Saturación de Agua	SATURA_H2O	N	12	8
Volumen de Arcilla	VOLUM_ARCI	N	12	8
Resistividad de la Formación	RESIS_FORM	N	12	8
Exponente de Cementación	EXPO_CEME	N	12	8
Resistividad de la Roca	RESIS_ROCA	N	12	8
Diámetro de Invasión	DIAM_INVA	N	12	8
Resistividad del Agua	RESIS_H2O	N	12	8
Profundidad de la Formación	PROFUNDIDA	N	12	8
Instancia Menor	**INST_MENOR**	**N**	**10**	**6**
Instancia Mayor	**INST_MAYOR**	**N**	**10**	**6**
Ocurrencia	**OCURRENCIA**	**N**	**6**	**0**

Así, la Figura V.2.10 muestra una fracción de una Base de Conocimientos con dichas propiedades. Por otro lado, también las Reglas de Producción tienen su propia estructura, tal y como se ilustra a continuación:

SI la **EMISIÓN DE RAYOS GAMMA** es menor o igual a *valor1* **Y**
la **POROSIDAD DE LA ROCA** es menor o igual a *valor2* **Y**
la **SATURACIÓN DE AGUA** es menor o igual a *valor3* **Y**
el **VOLUMEN DE ARCILLA** es menor o igual a *valor4* **Y**
la **RESISTIVIDAD DE LA FORMACIÓN** es menor o igual a *valor5* **Y**
el **EXPONENTE DE CEMENTACIÓN** es menor o igual a *valor6* **Y**
la **RESISTIVIDAD DE LA ROCA** es menor o igual a *valor7* **Y**
el **DIÁMETRO DE INVASIÓN** es menor o igual a *valor8* **Y**
la **RESISTIVIDAD DEL AGUA** es menor o igual a *valor9* **Y**
la **PROFUNDIDAD DE LA FORMACIÓN** es menor o igual a *valor10* **Y**

ENTONCES el **VOLUMEN DE HIDROCARBUROS** está entre *valor_inferior* y *valor_superior* con una ocurrencia de *frecuencia_regla/no_total_de_Reglas_en_Base_Conoc*

donde las palabras en mayúscula (excepto SI, ENTONCES e Y) representan cada uno de los atributos del pozo (Emisión de Rayos Gamma, Porosidad de la Roca, etc.), mientras que las expresiones en cursiva "valor1" hasta "valor 10" corresponden a cada uno de los valores de las características del yacimiento almacenados por la Base de Conocimientos. En lo referente a "valor_inferior" y "valor_superior", estos son utilizados para indicar los límites, tanto inferior como superior entre los que se encuentra el volumen de hidrocarburos. Respecto al numerador "frecuencia_regla", éste hace referencia al valor de ocurrencia que se ha estado mencionando y por último, el denominador "no_total_de_Reglas_en_Base_Conoc", representa el total de Reglas presente en la Base de Conocimientos.

Para ilustrar la fusión de ambas estructuras, supóngase que se tiene la siguiente Base de Conocimientos:

REGLA	RAYOS GAMA	POROSIDAD	SATURA H2O	VOLUM ARCI	RESIS FORM	EXPO CEME
1	93.570	0.055	0.166	0.087	142.636	48.520
2	88.960	0.016	0.178	0.019	137.115	39.997
3	91.072	0.033	0.200	0.160	1209.628	13.537

RESIS ROCA	DIAM INVA	RESIS H2O	PROFUNDIDA	INST MENOR	INST MAYOR	OCURRENCIA
0.267	77.916	4.022	2736.266	0.23	0.38	18
0.685	64.261	10.894	2856.357	0.23	0.43	16
0.452	49.369	1.698	2602.763	0.27	0.35	13

entonces, si se desea leer la Regla 1 se hará de la siguiente manera:

SI la **EMISIÓN DE RAYOS GAMMA** es menor o igual a *93.570* **Y**
la **POROSIDAD DE LA ROCA** es menor o igual a *0.055* **Y**
la **SATURACIÓN DE AGUA** es menor o igual a *0.166* **Y**
el **VOLUMEN DE ARCILLA** es menor o igual a *0.087* **Y**
la **RESISTIVIDAD DE LA FORMACIÓN** es menor o igual a *142.636* **Y**
el **EXPONENTE DE CEMENTACIÓN** es menor o igual a *48.520* **Y**
la **RESISTIVIDAD DE LA ROCA** es menor o igual a *0.267* **Y**
el **DIÁMETRO DE INVASIÓN** es menor o igual a *77.916* **Y**
la **RESISTIVIDAD DEL AGUA** es menor o igual a *4.022* **Y**
la **PROFUNDIDAD DE LA FORMACIÓN** es menor o igual a *2736.266* **Y**

ENTONCES el **VOLUMEN DE HIDROCARBUROS** está entre *0.23* y *0.38*
con una ocurrencia de *18/47*

y así para cada una de las demás Reglas almacenadas en la Base de Conocimientos.

MÓDULO GENERADOR DE REGLAS

Este apartado se encuentra dedicado a presentar la forma en la cual se llevó a cabo el manejo de las Reglas de Producción (obtenidas a través del método inductivo), así, el Sistema Experto contiene un módulo en el que se permite editar, modificar y agregar otras nuevas Reglas a la Base de Conocimientos, según como la aplicación lo vaya requiriendo (ver Figura V.2.11), dicha sección se denominó "Reglas" y se encuentra localizada en la parte izquierda del Menú (ver Figura V.2.12).

Sobra decir que dicho conocimiento se halla en formato .DBF el cual es perfectamente compatible con Delphi. Por lo que, para poder accesar a la información de las bases de datos, se emplearon fundamentalmente 5 componentes de este entorno: Table, DataSource, DBGrid, DBEdit y DBNavigator[4]. Asimismo, se crearon tablas internas que permitieran un mejor manejo de la información.

1) Edición de Reglas

El módulo de "Reglas" contiene una sección especial en la que el usuario puede editar las Reglas de Producción que se encuentran en la Base de Conocimientos, y así, poder visualizarlas de una manera más clara que si las estuviera viendo directamente como una base de datos (ver Figura V.2.13). Asimismo, tiene la oportunidad del empleo del "mouse" para un manejo mucho más fácil del entorno.

En la parte superior de la pantalla, se encuentra desplegada la Base de Conocimientos, esto a través de una tabla, donde el usuario puede trasladarse al primer registro o al último de ellos, utilizando las flechas que se localizan en la parte inferior de la cuadrícula o, si lo desea, hacer uso de la barra de desplazamiento que se halla en el extremo derecho de la misma.

En la parte inferior, como podrá observarse, se encuentra una disposición de texto en la que el usuario puede ver de manera explícita, jerarquizada y apropiada algunas de las Reglas que le hayan sido de interés esto a través de la estructura SI-ENTONCES.

Cabe señalar que, el botón de "Cerrar" que se presenta en la parte inferior derecha, termina con la ejecución del módulo, por lo que al oprimirlo no se finalizará con la aplicación, pudiendo realizar otros procesos.

[4] Una descripción de estos componentes ya fue presentada en el capítulo IV.

Figura V.2.11 Figura en la que se muestra la ubicación estructural del Módulo Generador de Reglas.

Figura V.2.12 Figura que muestra la localización del Módulo de Reglas.

Figura V.2.13 figura que exhibe la ejecución del submódulo "Editar Reglas".

2) Modificación de Reglas

El Sistema Experto cuenta con una sección especial en la que el usuario puede realizar modificaciones a una o varias Reglas de Producción de la Base de Conocimientos (ver Figura V.2.14). Sin embargo, para llevar a cabo esta operación, se requiere que el interesado conozca la clave de acceso, pues sin ella no es factible la ejecución de dicho movimiento.

Una vez que se ha alcanzado lo anterior, se le despliega al usuario una pantalla en la que además de que se editan las Reglas en una tabla (similar al caso anterior), también se le presenta en la parte inferior izquierda una opción en la que, con hacer "click" y en seguida seleccionar el valor del campo en la cuadrícula, se pueden realizar la cambios. Cabe señalar que el Sistema calcula un intervalo por columna (es decir, por atributo), por lo que la cantidad a introducir no debe ser ni inferior ni superior a ese margen, de lo contrario se emite un mensaje y la modificación no se lleva a cabo (ver Figura V.2.15). Asimismo, si el nuevo valor es correcto, entonces el sistema pide confirmar, esto con el objeto de asegurar que verdaderamente se desea realizar la alteración. Por otra parte, cada modificación es individual, es decir, una confirmación para cada cambio (ver Figura V.2.16).

En cuanto al botón de "Cerrar" , sucede como en la sección anterior, es decir, únicamente afecta a este proceso, por lo que podrán realizarse otras operaciones después de oprimirlo.

3) Adición de Reglas

Una opción más con la que cuenta el Módulo Generador de Reglas, es la flexibilidad para adicionar nuevas Reglas de Producción a la Base de Conocimientos, esto para el caso en el que el usuario (experto o no) desee incluir alguna de ellas que no haya sido obtenida por inducción, pero que exista la necesidad de contemplarla por el Sistema Experto.

Así, el agregar una Regla implica cierto grado de conocimiento, por lo que es necesario que la persona que vaya a realizar la operación conozca la clave de acceso a la que se ha estado haciendo referencia.

Entonces, cada vez que se elija esta operación aparecerá un mensaje en el que se le indicará al usuario que de aceptar, la Regla introducida se agregará a la Base de Conocimientos, de lo contrario se cancelará el proceso (ver Figura V.2.17). De esta manera, si el interesado continúa, se iniciará con la introducción de información. Primeramente, se le solicitará el valor de 2 instancias, esto es un límite inferior y otro superior para representar el intervalo en el cual se encontrará el atributo a diagnosticar (volumen o saturación de hidrocarburos). Asimismo, se le indicará al usuario la inclusión de la cifra de ocurrencia (ver Figura V.2.18), es decir, un número entero que indique la frecuencia con la que se presenta

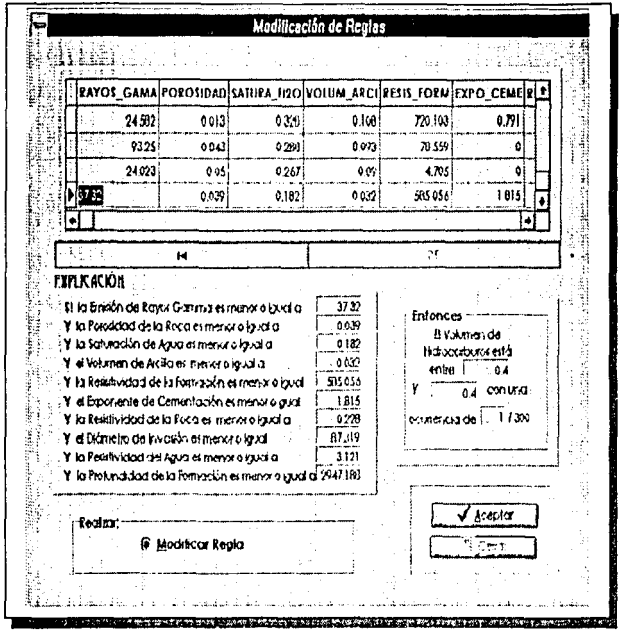

Figura V.2.14 Figura que presenta en ejecución el submódulo "Modificar Reglas"

Figura V.2.15 Figura que muestra el error que ocurre cuando se desea intreemplazar un valor de la Base de Conocimientos con una cantidad que está fuera de rango

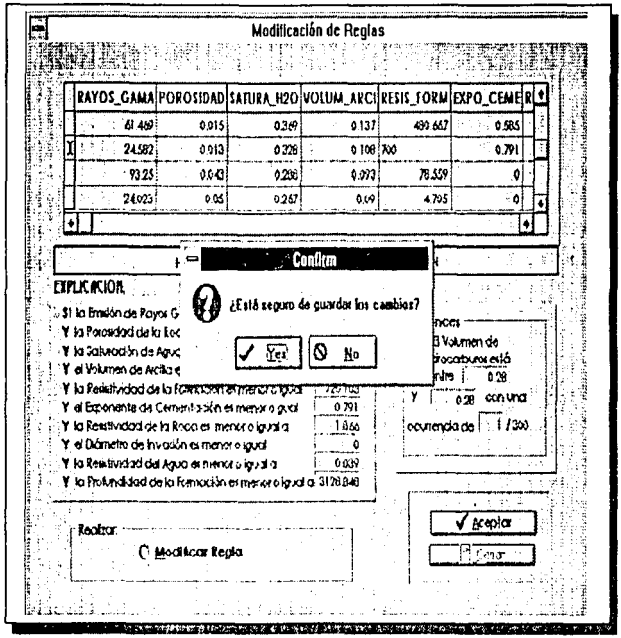

Figura V.2.16 Figura en la que se presenta en ejecución el submódulo "Modificar Reglas" en la fase de confirmar cambios.

Figura V.2.17 Figura en la que se muestra un mensaje para el inicio del proceso de agregar Reglas de Producción.

Figura V.2.18 Figura en la que se le solicita al usuario la introducción de algunas características para la nueva Regla que se vaya a introducir.

la Regla[5]. Posteriormente, al haber introducido esos datos, se le solicita al usuario los valores para cada una de las características del yacimiento (ver Figura V.2.19). Así, al dar "click" al último botón de "Aceptar", el sistema se encuentra listo para almacenar el nuevo saber. Sin embargo, el Sistema Experto cuenta con un mecanismo que permite detectar la existencia de 2 Reglas idénticas, por lo que, ante esta situación sólo emite un mensaje sin permitir la inclusión de dicho conocimiento repetido.

Cabe señalar que, también existe la opción (en cada una de las pantallas) de cancelar la operación, sin que con ello se altere la ejecución del Sistema Experto, pudiendo realizar algún otro procedimiento.

MÓDULO PARA CONSULTA

El Sistema Experto cuenta con una sección donde el usuario, experto o con al menos cierto conocimiento en el área, puede realizar una consulta (ver Figura V.2.20), para ello solamente es necesario introducir la información solicitada por el sistema y en seguida, éste comienza a trabajar para proporcionar una respuesta. Cabe señalar que, dicho Módulo para Consulta, se encuentra en el menú que se mostró en la Figura V.2.2.

En el caso de este trabajo, el propósito de la Consulta residió en llegar a determinar el volumen o saturación de hidrocarburos presente en un yacimiento, esto en base a otras características del mismo, como emisión de Rayos Gamma, Porosidad de la Roca, Saturación de Agua, Exponente de Cementación, etc.

Para tal efecto, se construyó una rutina que tiene por objetivo introducir cada uno de los datos que se requieren, sin embargo, para llevar a cabo esta inserción, se diseñó además, un procedimiento que permite calcular un intervalo por columna (es decir, por campo o atributo) en base a la información que se encuentra guardada en la Base de Conocimientos (similar al caso de Modificación de Reglas). Así, por ejemplo, si el intervalo para el campo "EXPO_CEME" es [5.7730, 48.5200], el usuario tendrá que introducir un valor que se halle comprendido entre esos números para que el proceso continúe, de manera que, si la cantidad que introdujo sale del rango, se produce un mensaje de error (ver Figura V.2.21), esperando a que se corrija: en situación contraria, es decir, que se halla introducido un valor correctamente, se prosigue con el siguiente atributo, restringido también por un intervalo y así sucesivamente con las demás características del pozo

Todo esto fue diseñado con el objeto de que el usuario (aunque se encuentre familiarizado con el tema) tenga una idea más clara del conocimiento que se encuentra almacenado y pueda realizar su consulta de manera más eficiente. Así si descubre que lo

[5] Esto se refiere al factor de certidumbre que se explicó en la sección Base de Conocimientos.

Figura V.2.1 Figura que muestra la pantalla de captura para los atributos del yacimiento para la nueva Regla de Producción.

Figura V.2.20 Figura en que se exibe la localización estructural del Módulo para Consulta

Figura V.2.21 Figura en la que se presenta una sesión de Consulta, donde se ha introducido un valor fuera de rango.

que desea consultar no se encuentra explícitamente en la Base de Conocimientos, puede recurrir a otros procesos, ya sea ejecutar el Algoritmo Inductivo, consultar a otro (o un) especialista para agregar nuevo conocimiento o sencillamente, realizar alguna prueba con el método de Encadenamiento.

Una vez que los datos solicitados por el Sistema Experto han sido capturados satisfactoriamente, el programa los memoriza (almacena momentáneamente durante esta Consulta) y con ello, indaga en la Base de Conocimientos hasta encontrar alguna Regla que satisfaga los valores introducidos, esto es, la estructura SI-ENTONCES que ya se mostró en el apartado Base de Conocimientos. Así, supóngase que se insertaron los siguientes valores:

Exponente de Cementación = 20
Resistencia de la Roca = 0.8
Profundidad = 2700[6]

y que se encuentran las siguientes Reglas en la Base de Conocimientos[7]:

REGLA	EXPO_CEME	RESIS_ROCA	PROFUNDIDA	INST_MENOR	INST_MAYOR	OCURRENCIA
1	21	0.5	2800	0.0120	0.0150	20
2	23	0.85	2750	0.0000	0.0200	15
3	25	1	3000	0.0300	0.03500	10

entonces, la Regla con la que el Sistema Experto se quedará y utilizará para dar su respuesta será la segunda, pues es la que verifica la siguiente estructura:

SI el **EXPONENTE DE CEMENTACIÓN** es menor o igual a *23* **Y**
la **RESISTIVIDAD DE LA ROCA** es menor o igual a *0.85* **Y**
la **PROFUNDIDAD DE LA FORMACIÓN** es menor o igual a *2750*

ENTONCES el **VOLUMEN DE HIDROCARBUROS** está entre *0.0000* y *0.0200*
con una ocurrencia de *15/45*

Y aún cuando la tercera también cumple, el sistema sigue tomando la segunda, esto por la jerarquía que existe en la Base de Conocimientos (es decir, tiene mayor prioridad):

[6] Solamente se tomaron 3 atributos como ejemplo para ilustrar el proceso.
[7] Recuérdese que las Reglas de Producción se hallan jerarquizadas en la Base de Conocimientos.

proporcionándole al usuario como respuesta a su Consulta un volumen para Hidrocarburos de 0.0000 y 0.0200 con una ocurrencia de 15/45. La Figura V.2.22 muestra el mensaje que se emite ante esta situación, mientras que la Figura V.2.23 ilustra el caso contrario.

De igual forma, como en el caso de secciones anteriores, existe la posibilidad de abortar la operación, lo cual puede hacerse a través de los botones que indiquen "Cerrar" y sin que se altere la ejecución de otros procedimientos.

MOTOR DE INFERENCIA

Uno de los componentes de todo Sistema Experto que requiere de mucha atención es el Motor de Inferencia, pues en él se puede llevar a cabo un tipo muy especial de consulta, no como las que tradicionalmente se efectúan en un Sistema de Información convencional, sino una que involucre algún factor inteligente (ver Figura V.2.24).

Retomando un poco lo que ya se explicó en el capítulo III, existen varias estrategias inferenciales de las cuales hace uso un Mecanismo de Inferencia, es decir, encadenamiento hacia adelante, encadenamiento hacia atrás, así como una combinación entre ambas, y cuya utilización va a depender en gran medida de la aplicación.

En el caso de este trabajo, existían varias situaciones que debían tomarse en cuenta para la elección de la técnica entre ellas el valor de certidumbre que acompaña a la Regla, el cual es proporcionado por el algoritmo inductivo y que fue el que permitió jerarquizar todas las Reglas de la Base de Conocimientos, colocando la de mayor valor al principio de la misma, lo que indicaría una mayor frecuencia, siguiendo en orden descendente todas las demás.

De manera que, se decidió utilizar una combinación de ambos tipos de estrategias, por lo que, siguiendo con la idea planteada en el capítulo III, se procedió a diseñar un algoritmo en el que el encadenamiento siempre iniciara con la primera Regla de la Base de Conocimientos. Cabe señalar que, dicho módulo se encuentra en la sección Sistema Experto que se puede visualizar en el menú (ver Figura V.2.25).

Antes de explicar el funcionamiento de la estrategia empleada, debe recordarse que se emplearon 10 atributos del pozo como premisas para llegar a una conclusión, de manera que, las siguientes líneas muestran la estructura que siguió cada una de las Reglas (Ver Figura V.2.26):

SI la **EMISIÓN DE RAYOS GAMMA** es menor o igual a *valor1* **Y**
la **POROSIDAD DE LA ROCA** es menor o igual a *valor2* **Y**
la **SATURACIÓN DE AGUA** es menor o igual a *valor3* **Y**

Figura V.2.22 Figura que muestra el mensaje proporcionado por el Sistema Experto cuando ha encontrado alguna Regla que satisface los datos introducidos por el usuario.

Figura V.2.23 Figura en la que se presenta el mensaje proporcionado por el Sistema Experto cuando no ha sido posible localizar alguna Regla que verifique todos los datos introducidos por el usuario.

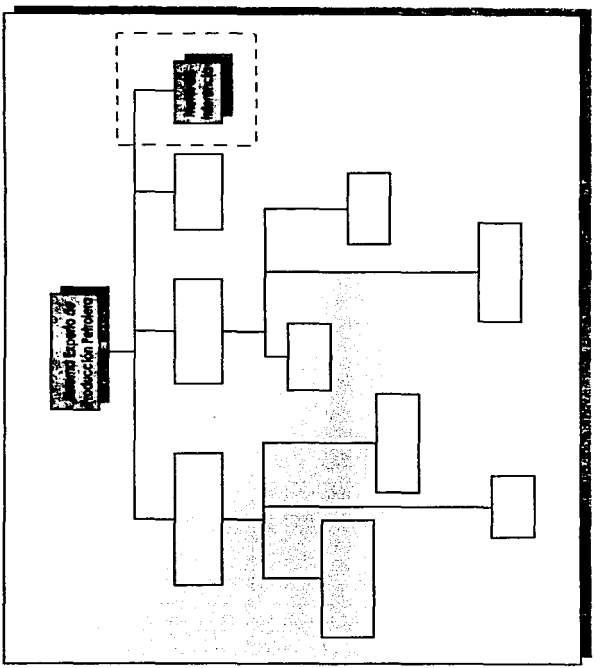

Figura V.2.24 figura que presenta la ubicación estructural del Motor de Inferencia del Sistema Experto

Figura V.2.25 Figura que muestra la localización del módulo donde se encuentra el Motor de Inferencia.

Figura V.2.26 Figura que muestra de manera gráfica la estructura de las Reglas de Producción en la Base de Conocimientos.

el **VOLUMEN DE ARCILLA** es menor o igual a *valor4* **Y**
la **RESISTIVIDAD DE LA FORMACIÓN** es menor o igual a *valor5* **Y**
el **EXPONENTE DE CEMENTACIÓN** es menor o igual a *valor6* **Y**
la **RESISTIVIDAD DE LA ROCA** es menor o igual a *valor7* **Y**
el **DIÁMETRO DE INVASIÓN** es menor o igual a *valor8* **Y**
la **RESISTIVIDAD DEL AGUA** es menor o igual a *valor9* **Y**
la **PROFUNDIDAD DE LA FORMACIÓN** es menor o igual a *valor10* **Y**

ENTONCES el **VOLUMEN DE HIDROCARBUROS** está entre *valor_inferior* y *valor_superior* con una ocurrencia de *frecuencia_regla/no_total_de_Reglas_en_Base_Conoc*

Así, regresando con la forma de trabajo de la Máquina de Inferencia, éste siempre inicia el encadenamiento con el primer atributo de la Regla que se encuentra en el sitio primario, es decir, comienza haciendo una cuestión muy similar a la siguiente:

"¿El porcentaje de Emisión de Rayos Gamma es <= 88.906000[8]?"

a lo cual el usuario puede contestar con un "si"[9] o un "no" (ver Figura V.2.27). Si la respuesta es afirmativa, el Sistema Experto continúa preguntando por el siguiente atributo, por ejemplo la Porosidad de la Roca y así sucesivamente. En caso de que el interesado proporcione una negativa a la cuestión, se pueden identificar diversas situaciones a las que se dará explicación.

Una de ellas es que, el sistema salta a la siguiente Regla, hasta encontrar aquella que cumpla con las premisas ya satisfechas, es decir, supóngase que ya se confirmo positivamente la pregunta anterior, es decir que el porcentaje de Emisión de Rayos Gamma haya sido menor o igual a 88.906000, pero que en la siguiente cuestión (para la característica Porosidad de la Roca) no sea así, entonces, el sistema no volverá a hacer la pregunta ya planteada, sino que internamente buscará alguna Regla subsecuente en la que el valor para dicho atributo sea menor o igual que la cantidad 88.906000 (porque el usuario así lo confirmó), de manera que, si se verifica esta cuestión, lo que el sistema procederá a realizar es preguntar respecto a la siguiente característica de esa misma Regla (ver Figura V.2.28), repitiendo para ésta el mismo procedimiento de verificación; mientras el

[8] Esta cantidad va a depender del valor que se encuentre almacenado en la Base de Conocimientos.
[9] "Yes" en el Sistema Experto debido a que se utilizó una caja de diálogo predeterminada por Delphi.

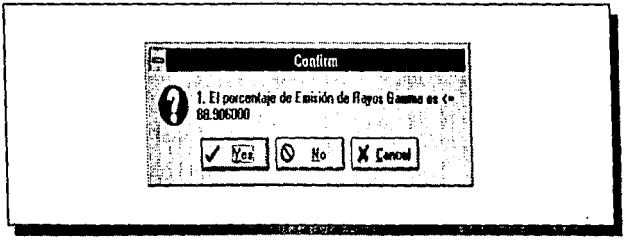

Figura V.2.27 Figura que presenta la forma en que se le pregunta al usuario respecto a algún atributo en una sesión de la Máquina de Inferencia.

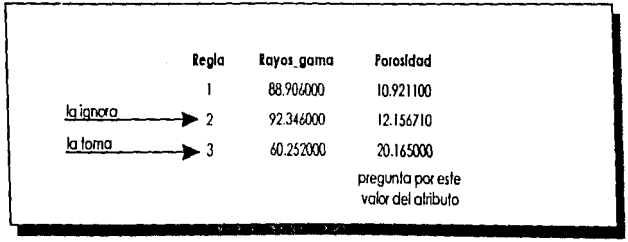

Figura V.2.28 Figura que muestra una de las situaciones que pueden presentarse en el proceso de Encadenamiento.

Sistema Experto irá memorizando cada una de las respuestas que el usuario proporciona[10], esto con el fin de que vaya corroborando cada una de las respuestas a medida que el proceso va avanzando.

Otra de las cosas que se puede presentar es que el Sistema Experto, en su afán de encontrar alguna Regla que satisfaga todas las respuestas del usuario, recorra completamente la Base de Conocimientos sin hallarla, por lo que, si el número de atributos verificados es menor que 9, el sistema simplemente comunicará la falta de información; por el contrario, si se confirmaron 9 de las 10 características del yacimiento (las primeras), entonces se le presenta al usuario la opción de que el Sistema Experto le proporcione alguna "sugerencia", permitiéndole elegir una solución[11] entre varias (ver Figuras V.2.29 y V.2.30).

Por otro lado, cuando el número de atributos verificados es nulo, significa que el Sistema Experto se encuentra en la primera fase del proceso, es decir, está tratando de satisfacer la pregunta para el primer atributo, entonces la respuesta es sencillamente un mensaje en el que se indica la no existencia de una Regla en la Base de Conocimientos, capaz de satisfacer la cuestión planteada, por lo que el Sistema Experto ya no continúa la búsqueda (ver Figura V.2.31).

Cuando la situación es diferente a lo ya descrito, esto es, se ha llegado a satisfacer todas las respuestas del usuario, significa que se encontró una solución para esta situación en particular, por lo que el Sistema Experto notifica el éxito de la operación, proporcionando, tanto el intervalo en el que se halla el volumen de hidrocarburos, como su valor de ocurrencia (ver Figura V.2.32). Un aspecto que cabe señalar en este punto es que, el Sistema Experto toma la primer Regla que satisfaga las respuestas otorgadas por el usuario, pues es la que mayor jerarquía tiene en la Base de Conocimientos.

Por otra parte, como en los demás procedimientos del sistema, también aquí existe la opción de cancelar el proceso de Encadenamiento y poder reanudarlo justamente donde se quedó o, simplemente terminar con dicha actividad, pudiendo realizar alguna otra operación.

Hasta aquí simplemente se ha descrito la forma en que funciona la Máquina de Inferencia, por lo que, en la sección posterior, se muestra propiamente ya una aplicación a una Base de Datos con Registros Geofísicos, en donde se puede percibir a más detalle, el proceso inferencial aunado a algunas situaciones que pueden presentarse.

---

[10] Así, el sistema primeramente memoriza 88.906000.

[11] Esto va a depender del conocimiento que se encuentre almacenado.

Figura V.2.29 Figura en la que se muestra la iniciativa de continuar con la sección de Sugerencias.

Figura V.2.30 figura que presenta en ejecución la sección de Sugerencias.

Figura V.2.31 Figura que muestra el mensaje que presenta el Sistema Experto cuando no ha encontrado una Regla que satisfaga las respuestas proporcionadas por el usuario.

Figura V.2.32 La Figura exhibe la el mensaje que envía el Sistema Experto cuando se ha encontrado alguna Regla que verifique las respuestas del usuario.

V.3 APLICACIÓN A UNA BASE DE DATOS CON REGISTROS GEOFÍSICOS.

El apartado anterior se centró básicamente, entre otras cosas, a presentar la forma en la que funciona cada uno de los componentes del Sistema Experto. Por lo que, en esta sección se presentan algunas pruebas llevadas a cabo, ello con el fin de mostrar la influencia de la elección del umbral en la conformación de la Base de Conocimientos y en consecuencia, el funcionamiento de la Máquina de Inferencia.

Cabe señalar que, en lo referente a la depuración en centroides (ver Inducción en V.2) se manejará un valor de "0", esto para poder apreciar todas las Reglas de Producción formadas, aún cuando posean como valor de ocurrencia "1".

Así, para efectos prácticos se tomó como base de datos inicial el archivo demopmx1.dbf, conteniendo 300 registros con la información petrolera. Mientras que, el programa con el algoritmo neuronal inductivo fue el denominado inducci.exe.

A continuación se presentan cada una de las situaciones planteadas, identificadas por valor de umbral.

1) Umbral sugerido

Como ya se indicó en una sección anterior, el programa con el método inductivo cuenta con una opción para que el valor del umbral sea proporcionado por éste, así que al realizar dicho proceso, se generaron (con el archivo demopmx1 que se mencionó) un total de 98 Reglas de Producción con valores de ocurrencia como se muestra en seguida:

No. de Reglas	Valor de Ocurrencia
1	25
1	16
3	13
1	11
2	10
2	9
2	7
2	6
2	5
5	4
11	3
16	2
50	1

Así, para comenzar con el trabajo de la Máquina de Inferencia, supóngase que se posee tentativamente[12] algunos valores como los que siguen:

RG	POR	S_H2O	VO_ARC	RE_FOR	EX_CE	RE_RO	DI_INV	RE_H2O	PROF	
90	0.05	0	0	80	50	0.1	80	2	3000	..... (I)

donde:

RG = Emisión de Rayos Gamma
POR = Porosidad de la Roca
S_H2O = Saturación de Agua
VO_ARC = Volumen de Arcilla
RE_FOR = Resistividad de la Formación
EX_CE = Exponente de Cementación
RE_RO = Resistividad de la Roca
DI_INV = Diámetro de Invasión
RE_H2O = Resistividad de Agua
PROF = Profundidad de la Formación

y se lee "SI el porcentaje de Emisión de Rayos Gamma es <= 90 Y el valor de la Porosidad es <= 0.05 Y ... Y la Profundidad de la Formación es <= 3000"[13].

Entonces, el proceso de Encadenamiento del Motor de Inferencia, inicia en la primera Regla de su Base de Conocimientos y, dependiendo de la respuesta del usuario, seguirían las demás. A continuación se muestra de que manera sería el diálogo entre dicho usuario y el Sistema Experto de Producción Petrolera:

Pregunta 1. ¿El porcentaje de Emisión de Rayos Gamma es <= 88.906000?	No	Busca otra Regla
Pregunta 2. ¿El porcentaje de Emisión de Rayos Gamma es <= 93.516000?	SI	Memoriza este valor

[12] Tentativamente, pues no se manejan valores exactos, sino aquellos que vayan cumpliendo con la desigualdad planteada en cada pregunta que el Sistema Experto realiza, más adelante como se vaya desarrollando el ejemplo se verá.

[13] Nótese que solamente se presenta las premisas, pues la conclusión la tiene que proporcionar el Sistema Experto, es por ello que no se ha incluido.

Pregunta 3. ¿El valor (fracción) de la Porosidad de la Roca es <= 0.108000?	Si	Memoriza este valor
Pregunta 4. ¿El valor (fracción) de la Saturación de Agua es <= 0.213000?	Si	Memoriza este valor
Pregunta 5. ¿El porcentaje de Volumen de Arcilla es <= 0.070000?	Si	Memoriza este valor
Pregunta 6. ¿La Resistividad de la Formación es <= 55.943000?	No	Busca otra Regla en la que se satisfagan todas las premisas verificadas.
Pregunta 7. ¿La Resistividad de la Formación es <= 303.223000?	Si	Memoriza este valor
Pregunta 8. ¿El Exponente de Cementación es <= 5.881000?	No	Busca otra Regla en la que se satisfagan todas las premisas verificadas.
Pregunta 9. ¿El Exponente de Cementación es <= 19.418000?	No	Busca otra Regla en la que se satisfagan todas las premisas verificadas.
Pregunta 10. ¿El Exponente de Cementación es <= 65.233000?	Si	Memoriza este valor
Pregunta 11. ¿La Resistividad de la Roca es <= 0.022000?	No	Busca otra Regla en la que se satisfagan todas las premisas verificadas.
Pregunta 12. ¿La Resistividad de la Roca es <= 0.100000?	Si	Memoriza este valor
Pregunta 13. ¿El Diámetro de Invasión (pulg.) es <= 129.987000?	Si	Memoriza este valor
Pregunta 14. ¿La Resistividad del Agua es <= 6.558000?	Si	Memoriza este valor
Pregunta 15. ¿La Profundidad (mts.) de la Formación es <= 2804.846000?	No	Busca otra Regla en la que se satisfagan todas las premisas verificadas.
Pregunta 16. ¿La Profundidad (mts.) de la Formación es <= 2810.332000?	No	Busca otra Regla en la que se satisfagan todas las premisas verificadas.

Como habrá de observarse, el sistema indaga para encontrar una Regla que satisfaga aquellas condiciones que ya fueron resueltas por el usuario, sólo que en este caso las Reglas de la Base de Conocimientos ya se han agotado, es decir, no hay más que revisar, por lo cual el sistema tiene como opción mostrar algunas de ellas que, verificando las 9 características anteriores del pozo, puedan serle de utilidad (ver Figura V.3.1).

Resumiendo, en este ejemplo, los **valores iniciales tentativos** fueron:

RG	POR	S_H2O	VO_ARC	RE_FOR	EX_CE	RE_RO	DI_INV	RE_H2O	PROF
90	0.05	0	0	80	50	0.1	80	2	3000

mientras que, los memorizados por el Sistema Experto son:

RG	POR	S_H2O	VO_ARC	RE_FOR	EX_CE	RE_RO	DI_INV	RE_H2O	PROF
93.5[14]	0.1	0.2	0.1	303.2	65.2	0.1	129.987	6.558	*[15]

Así, existen 2 Reglas como alternativas que sugirió el sistema, y que son las que se muestran a continuación:

RG	POR	S_H2O	VO_ARC	RE_FOR	EX_CE	RE_RO	DI_INV	RE_H2O	PROF
20.516	0.055	0.166	0.031	245.345	0.91	0.1	129.987	6.558	2804.846
12.391	0.016	0.178	0.025	136.521	0	0.038	85.252	0.752	2810.332

con la siguiente conclusión:

I_MEN	I_MAY	OCURRENCIA
0.42	0.42	1
0.42	0.42	1

donde:

I_MEN = Extremo inferior del intervalo para el Volumen de Hidrocarburos
I_MAY = Extremo Superior del intervalo para el Volumen de Hidrocarburos
OCURRENCIA = Valor de ocurrencia de la Regla

[14] Estos valores se han redondeado para efectos prácticos en la presentación
[15] Este lugar tiene un asterisco debido a que el sistema no encontró alguna Regla que verificará esta característica del pozo que respondió el usuario.

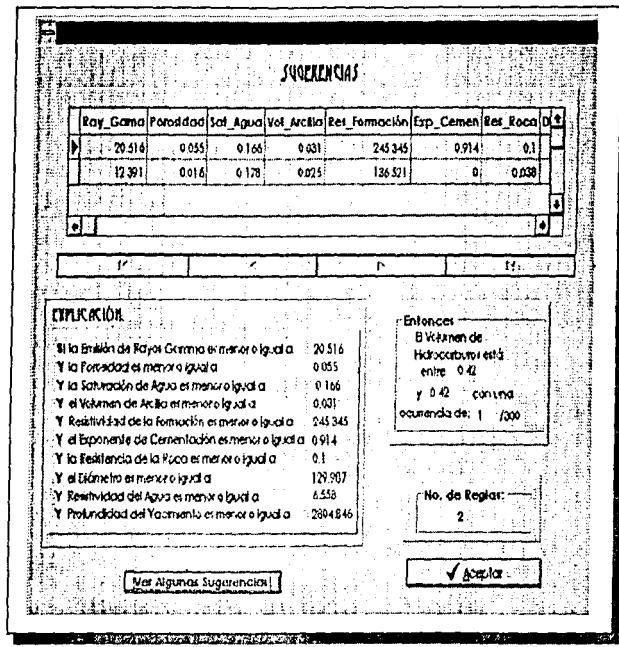

Figura V.3.1 La Figura presenta 2 alternativas como respuesta para que el usuario decida cual de ellas le puede ser de utilidad.

y además se verifican los valores almacenados por el Sistema Experto con las Reglas encontradas, es decir 20.516 <= 93.5, 12.391<=93.5, y así sucesivamente para las demás premisas.

Concluyendo este ejemplo, se puede decir que, de acuerdo a los datos manejados por el usuario, no sobresale una Regla específica que satisfaga las exigencias de éste, existiendo 2 alternativas de las cuales puede elegir la que más le convenga, pues en ambos casos, el volumen de hidrocarburos es el mismo así como su ocurrencia. Por lo que, una de ellas sería la siguiente:

SI la **EMISIÓN DE RAYOS GAMMA** es menor o igual a *20.516* **Y**
la **POROSIDAD DE LA ROCA** es menor o igual a *0.055* **Y**
la **SATURACIÓN DE AGUA** es menor o igual a *0.166* **Y**
el **VOLUMEN DE ARCILLA** es menor o igual a *0.031* **Y**
la **RESISTIVIDAD DE LA FORMACIÓN** es menor o igual a *245.345* **Y**
el **EXPONENTE DE CEMENTACIÓN** es menor o igual a *0.91* **Y**
la **RESISTIVIDAD DE LA ROCA** es menor o igual a *0.1* **Y**
el **DIÁMETRO DE INVASIÓN** es menor o igual a *129.987* **Y**
la **RESISTIVIDAD DEL AGUA** es menor o igual a *6.558* **Y**
la **PROFUNDIDAD DE LA FORMACIÓN** es menor o igual a *2804.846* **Y**

ENTONCES el VOLUMEN DE HIDROCARBUROS está entre *0.42* y *0.42*
con una ocurrencia de *1*/300

2) Umbral manual

Otra de las opciones con las que cuenta el método inductivo en el Sistema Experto, es el poder elegir manualmente el valor para el umbral, así en esta sección se presentan varias situaciones con dicha alternativa, en las cuales se emplean los mismos archivos que en el caso anterior (demopmx1.dbf e inducci.exe), así como los mismos valores tentativos para efectuar el Encadenamiento, esto con el fin de llevar a cabo una comparación entre los diferentes ejemplos.

a) Con valor de umbral 0.00001

Se llevó a cabo una prueba considerando un valor para umbral de 0.00001, una cantidad muy pequeña; de manera que el algoritmo inductivo llegó a generar un total de 286 Reglas de Producción con los siguientes valores de ocurrencia:

No. de Reglas	Valor de Ocurrencia
14	2
272	1

Así, para mostrar el funcionamiento del Motor de Inferencia en esta situación, tómese los valores (I) ya considerados anteriormente con sus respectivas notas.

Entonces, el método de Encadenamiento sigue el proceso que se ilustra a continuación:

Pregunta 1. ¿El porcentaje de Emisión de Rayos Gamma es <= 74.125000?	No	Busca otra Regla
Pregunta 2. ¿El porcentaje de Emisión de Rayos Gamma es <= 90.203000?	Si	Memoriza este valor
Pregunta 3. ¿El valor (fracción) de la Porosidad de la Roca es <= 0.078000?	Si	Memoriza este valor
Pregunta 4. ¿El valor (fracción) de la Saturación de Agua es <= 0.130000?	Si	Memoriza este valor
Pregunta 5. ¿El porcentaje de Volumen de Arcilla es <= 0.014000?	Si	Memoriza este valor
Pregunta 6. ¿La Resistividad de la Formación es <= 55.943000?	No	Busca otra Regla en la que se satisfagan todas las premisas verificadas.

Como puede percibirse, el Sistema Experto sólo alcanza a efectuar 6 preguntas, pues al llegar a la característica de Resistividad de la Formación, el usuario contesta negativamente a la cuestión, por lo que el sistema indaga con las demás Reglas, tratando de encontrar alguna de ellas que satisfaga las premisas anteriores, sin embargo, el procedimiento ha llegado al final de la Base de Conocimientos, por lo que, como sólo se verificaron 4 atributos, no existe Regla que satisfaga las afirmaciones del usuario (ver Figura V.3.2). A continuación se muestran los valores iniciales tentativos que se emplearon, así como los memorizados por el sistema:

RG	POR	S_H2O	VO_ARC	RE_FOR	EX_CE	RE_RO	DI_INV	RE_H2O	PROF
90	0.05	0	0	80	50	0.1	80	2	3000

RG	POR	S_H2O	VO_ARC	RE_FOR	EX_CE	RE_RO	DI_INV	RE_H2O	PROF
90.20	0.078	0.130	0.014	*	*	*	*	*	*

Figura V.3.2 Figura que muestra la respuesta dada por el Sistema Experto para el caso a), cuando el valor de umbral es pequeño (0.00001).

b) Con valor de umbral 0.009

Otras de las pruebas realizadas se llevó a cabo con un valor para umbral de 0.009, una cantidad un poco más grande que la anterior, por lo que el algoritmo inductivo generó un total de 56 Reglas de Producción con los siguientes valores de ocurrencia:

No. de Reglas	Valor de Ocurrencia
1	37
1	27
1	25
1	22
1	18
2	14
2	13
1	12
1	7
2	6
3	5
3	4
7	3
8	2
22	1

Considerando los valores iniciales tentativos, empleados en los casos anteriores, se llevó a cabo el proceso de la Máquina de Inferencia, el cual se presenta a continuación:

Pregunta 1. ¿El porcentaje de Emisión de Rayos Gamma es <= 93.250007	SI	Memoriza este valor
Pregunta 2. ¿El valor (fracción) de la Porosidad de la Roca es <= 0.0980007	SI	Memoriza este valor
Pregunta 3. ¿El valor (fracción) de la Saturación de Agua es <= 0.6120007	SI	Memoriza este valor
Pregunta 4. ¿El porcentaje de Volumen de Arcilla es <= 0.1900007	SI	Memoriza este valor
Pregunta 5. ¿La Resistividad de la Formación es <= 103.7390007	SI	Memoriza este valor

Pregunta 6. ¿El Exponente de Cementación es <= 48.5200007	No	Busca otra Regla en la que se satisfagan todas las premisas verificadas.
Pregunta 7. ¿El Exponente de Cementación es <= 158.9050007	Si	Memoriza este valor
Pregunta 8. ¿La Resistividad de la Roca es <= 0.0510007	No	Busca otra Regla en la que se satisfagan todas las premisas verificadas.

Como se observa, se presenta la misma situación que en el ejemplo anterior, pues solamente se llegaron a verificar 6 atributos, por lo que no existe alguna Regla de Producción en la Base de Conocimientos que satisfaga las respuestas proporcionadas por el usuario. Así, se muestran en seguida, tanto los valores iniciales tentativos, como los datos que memorizó el Sistema Experto:

RG	POR	S_H2O	VO_ARC	RE_FOR	EX_CE	RE_RO	DI_INV	RE_H2O	PROF
90	0.05	0	0	80	50	0.1	80	2	3000

RG	POR	S_H2O	VO_ARC	RE_FOR	EX_CE	RE_RO	DI_INV	RE_H2O	PROF
93.25	0.098	0.612	0.190	103.739	158.90	*	*	*	*

c) Con valor de umbral 0.07

Por último, se realizó una prueba con un valor para umbral de 0.07, una cantidad mayor que en los casos anteriores, con lo cual se llegó a conformar un total de 4 Reglas de Producción en la Base de Conocimientos, cuyos valores de ocurrencia se muestran en la siguiente tabla:

No. de Reglas	Valor de Ocurrencia
1	278
1	20
2	1

De manera que, el proceso de Encadenamiento siguió el siguiente diálogo entre usuario y Sistema Experto:

Pregunta 1. ¿El porcentaje de Emisión de Rayos Gamma es <= 93.516000?	SI	Memoriza este valor
Pregunta 2. ¿El valor (fracción) de la Porosidad de la Roca es <= 0.242000?	SI	Memoriza este valor
Pregunta 3. ¿El valor (fracción) de la Saturación de Agua es <= 1.000000?	SI	Memoriza este valor
Pregunta 4. ¿El porcentaje de Volumen de Arcilla es <= 0.647000?	SI	Memoriza este valor
Pregunta 5. ¿La Resistividad de la Formación es <= 491.640000?	SI	Memoriza este valor
Pregunta 6. ¿El Exponente de Cementación es <= 628.886000?	SI	Memoriza este valor
Pregunta 7. ¿La Resistividad de la Roca es <= 3.593000?	SI	Memoriza este valor
Pregunta 8. ¿El Diámetro de Invasión (pulg.) es <= 129.987000?	SI	Memoriza este valor
Pregunta 9. ¿ La Resistividad del Agua es <= 239.073000?	SI	Memoriza este valor
Pregunta 10. ¿La Profundidad (mts.) de la Formación es <= 3184.626?	SI	Memoriza este valor

Como se habrá de percatar, el sistema realizó un total de 10 preguntas al usuario, verificando afirmativamente todas ellas, lo cual significa que existe una Regla de Producción que satisface las respuestas del interesado, y por ende, encontrará respuesta (ver Figura V.3.3). Así, a continuación se presenta los valores iniciales tentativos empleados en el proceso de Encadenamiento:

RG	POR	S_H2O	VO_ARC	RE_FOR	EX_CE	RE_RO	DI_INV	RE_H2O	PROF
90	0.05	0	0	80	50	0.1	80	2	3000

Las siguientes cantidades expresan los valores almacenados por el Sistema Experto, los cuales coinciden con los de la Regla, pues todas la preguntas fueron contestadas afirmativamente:

Figura V.3.3 Figura que muestra la respuesta proporcionada por el Sistema Experto para el caso c), cuando el valor de umbral es grande (0.07).

RG	POR	S_H2O	VO_ARC	RE_FOR	EX_CE	RE_RO	DI_INV	RE_H2O	PROF
93.516	0.242	1.000	0.647	491.64	628.886	3.593	129.987	239.073	3184.6

Mientras que, los valores siguientes representan la respuesta del Sistema Experto al proceso de inferencia:

I_MEN	I_MAY	OCURRENCIA
0.000	0.520	278

En otras palabras, la respuesta junto con la Regla de Producción encontrada se expresa de la siguiente manera:

SI la **EMISIÓN DE RAYOS GAMMA** es menor o igual a *93.52* **Y**
la **POROSIDAD DE LA ROCA** es menor o igual a *0.242* **Y**
la **SATURACIÓN DE AGUA** es menor o igual a *1.000* **Y**
el **VOLUMEN DE ARCILLA** es menor o igual a *0.647* **Y**
la **RESISTIVIDAD DE LA FORMACIÓN** es menor o igual a *491.64* **Y**
el **EXPONENTE DE CEMENTACIÓN** es menor o igual a *628.886* **Y**
la **RESISTIVIDAD DE LA ROCA** es menor o igual a *3.593* **Y**
el **DIÁMETRO DE INVASIÓN** es menor o igual a *129.987* **Y**
la **RESISTIVIDAD DEL AGUA** es menor o igual a *239.073* **Y**
la **PROFUNDIDAD DE LA FORMACIÓN** es menor o igual a *3184.6* **Y**

ENTONCES el VOLUMEN DE HIDROCARBUROS está entre *0.000* y *0.520*
con una ocurrencia de 278/300

donde se verifican los valores almacenados por el Sistema Experto con los valores de la Regla encontrada, es decir 93.516 <= 93.516, 0.242 <= 0.242 y así sucesivamente para los demás atributos.

Como se pudo apreciar, la elección del umbral en el método inductivo es realmente importante, pues sus efectos en la Base de Conocimientos son bastante perceptibles. Esto se explica por el hecho de que, cuando el umbral es pequeño, el modelo neuronal tiende a reunir un gran número de Reglas, debido a que se forma un radio[16] muy diminuto que llega a abarcar pocos patrones, lo cual no da oportunidad para la agrupación de más

[16] Se forma una especie de cota de error.

elementos, pero si facilita la constitución de más centroides como en el caso a) que se presentó, donde con un valor para umbral de 0.00001 se generaron 286 Reglas.

Por otro lado, cuando este parámetro es grande, la conformación de Reglas disminuye, ya que el rango es más amplio, permitiendo en mayor grado la agrupación, pero reuniendo pocos grupos, tal es la situación en el ejemplo c) que se mostró, donde con un valor par umbral de 0.07, se crearon solamente 4 Reglas.

Sin embargo, el que se elija un umbral pequeño para obtener numerosas Reglas puede resultar engañoso, pues tendería a pensarse en la existencia de una gran cantidad de conocimiento almacenado, pero por otra parte, se correría el riesgo de que su valor de ocurrencia fuera muy pequeño[17], por lo que, a la hora de que se ejecutara el Motor de Inferencia y éste encontrara alguna Regla que verificara todos los cuestionamientos hechos, existiría la posibilidad de que tuviera muy poca representatividad (quizá "1"). Ahora bien, si la situación fuera diferente, umbral grande para generar pocas Reglas, quizá algunas tendrían alta ocurrencia y posiblemente en cualquier consulta a la Máquina de Inferencia se proporcionaría la misma respuesta. Ambas situaciones resultan un tanto problemáticas y en un momento dado, podrían dificultarle el trabajo al usuario. Por lo que, una alternativa sería el intentar con la opción que presenta el Sistema Experto, es decir, iniciar con un umbral sugerido por el programa y a partir de ahí, realizar pruebas con valores más pequeños o más grandes según el usuario lo crea conveniente, pues su elección resulta ser una cuestión que involucra cierto conocimiento.

[17] Esto obviamente va a depender también del valor para depurar centroides.

CONCLUSIONES

CONCLUSIONES

La labor llevada a cabo en este trabajo, se centró en el desarrollo de un prototipo de Sistema Experto para la Ingeniería Petrolera, cuyo propósito fue diagnosticar el volumen de hidrocarburos presente en un yacimiento. Para tal efecto, se utilizó el Lenguaje de Programación Orientado a Objetos Delphi de Borland.

Hasta ahora, dicho sistema se encuentra apto para realizar las operaciones más comunes que se pueden presentar en una sesión de trabajo, es decir, poder visualizar la Base de Conocimientos, agregar y modificar Reglas de Producción, efectuar alguna consulta, emplear el Método de Encadenamiento o, generar una Base de Conocimientos cuando se considere necesario (esto al ejecutar el algoritmo inductivo).

Por lo que, en base a todo el trabajo llevado a cabo, se pueden asentar varias reflexiones a modo de conclusión, así como también, establecer algunas consideraciones que se observaron resultarían muy útiles para refinar dicho prototipo que, debido a falta de tiempo y alcances propuestos no pudieron realizarse.

Por principio, se reafirma el hecho de que, las Bases de Datos, efectivamente constituyen una verdadera fuente de riqueza informativa, pues en este caso particular, los datos contemplados en los Registros Geofísicos Petroleros, desempeñaron un papel sumamente importante al detectarse en ellos, ciertos patrones de comportamiento que fueron muy útiles para la conformación de las Reglas de Producción en la Base de Conocimientos. De manera que, explotar las Bases de Datos, tratando de aprovechar al máximo su contenido, puede ser realmente de alto beneficio para cualquier empresa, ya que dichos datos pueden llegar a producir información estratégica útil en la toma de decisiones, planeación o supervisión.

De tal suerte que, existen diversos enfoques matemáticos que, permiten que dicha explotación de datos se lleve a cabo de una manera más sistemática y eficaz, entre ellos se encuentra el amplio campo de la Inteligencia Artificial, destacando la Inducción de información. En la situación aquí presentada, dicho método se aplicó a una Base de Datos con 300 registros, en la que fue posible obtener una Base de Conocimientos de dimensión

variable, esto debido a la sensibilización del parámetro de vigilancia del modelo neuronal (desarrollado en FoxPro), lo cual proporcionó flexibilidad para manipular la Base de Datos y observar sus repercusiones directamente en la Base de Conocimientos, donde efectivamente pudo comprobarse que las respuestas otorgadas por el Sistema Experto se vieron afectadas ante tal susceptibilidad.

Como puede percibirse claramente, por lo menos en esta etapa de desarrollo (construcción de la Base de Conocimientos) del Sistema Experto, no fue necesario requerir de la presencia de un experto en el área, situación que en un momento dado puede ser ventajosa, ya que permite ahorrar tiempo y dinero.

Por otra parte, el empleo del lenguaje de programación Delphi, realmente proporcionó valiosas herramientas para desarrollar el sistema, entre ellas el uso de componentes (objetos) predefinidos por el entorno, los cuales permitieron crear fácilmente pantallas, tablas, botones, mensajes, etc., con lo que se le brindó al usuario un ambiente amigable, facilitando además, la comunicación entre éste y el Sistema Experto, situación que en la actualidad se ha vuelto muy importante. De igual manera, se empleó el paradigma orientado a objetos con que opera implícitamente Delphi, combinando asimismo la programación estructurada en Pascal con la programación por eventos, sin olvidar el concepto de modularidad.

Una de las cosas que cabe hacer mención, referente a este lenguaje, es que, a pesar de la facilidad con la que se pueden crear aplicaciones, en particular, la iniciación fue un tanto difícil, pues, por ser un producto nuevo en el mercado, el material bibliográfico se encontraba muy limitado.

Hasta el momento, el funcionamiento del Sistema Experto ha sido satisfactorio, pues ha aportado respuestas coherentes e interesantes cuando se utiliza el módulo Consulta o el Método de Encadenamiento, proporcionando además, explicaciones cuando éstas se necesitan, lo cual permite darle al usuario un mayor conocimiento de lo que está sucediendo y con ello, éste pueda tomar, en una etapa posterior, las decisiones más adecuadas. Cabe señalar que, cada respuesta otorgada por el sistema, va acompañada por un factor de certidumbre, de acuerdo al contenido de la Base de Conocimientos, lo cual resulta una ventaja con respecto a los sistemas convencionales. En cuanto a los

procesos de Edición, Agregado y Modificación de Reglas, el Sistema Experto también los realiza sin ningún problema.

Como se señaló al principio, aún le faltan varias cosas a este prototipo, entre ellas sería conveniente diseñar elementos con características particulares de acuerdo a las necesidades propias de la aplicación, así como funciones que se puedan reusar y con ello disminuir la cantidad de código y trabajo, reduciendo a la vez, tiempo que puede ser útil en otras fases del desarrollo del sistema.

De igual manera, sería favorable, llegar a generalizar el número de atributos contemplados para el diagnóstico, pues ampliaría el contexto del análisis y mejoraría la aplicabilidad del Sistema Experto.

Por otro lado, ya en esta etapa, resulta importante la presencia de un experto en el campo, esto debido a la necesidad de validar las Reglas de Producción en la Base de Conocimientos, pues aún cuando éstas surgieron de un método inductivo que considera una serie de datos históricos (300 registros en este caso, aunque podrían incrementarse), no se garantiza por completo que contengan la experiencia que puede tener un especialista en el área.

Asimismo, sería muy recomendable agregarle al Sistema Experto, un módulo que proporcione ayuda cada vez que el usuario lo requiera, incluyendo además algunas explicaciones teóricas en cuanto al mismo.

Otra de las cosas que también resultaría de mucho beneficio, sobre todo para el usuario, es incorporarle otros Métodos Inductivos, así como algún módulo que permita hacer comparaciones entre ellos y poder detectar cual podría ser más conveniente (fiable) en un momento dado.

Así, para cualquier modificación o adición de módulos, el Sistema Experto cuenta con la ventaja de que es flexible, ya que como la programación es modular no es necesario revisar todo a detalle para poder hacerle los cambios necesarios para su mejor funcionamiento.

Aunque todavía esta en una fase de prototipo, puede decirse que un Sistema Experto en Ingeniería, en particular en la Petrolera, realmente sería útil y de alto beneficio, ya que, por un lado podría aprovecharse toda la información que los ingenieros obtienen y por otro, ahorrarse complicados procesos de cálculo y análisis que pueden verse reflejados en recursos económicos y humanos, al contarse con otros medios más prácticos de realizar dichos procedimientos.

Finalmente, cabe señalar que la Inteligencia Artificial constituye una disciplina, por lo menos en nuestro país, poco desarrollada, pero con una historia llena de aciertos, pues ha venido a renovar muchos de los procesos calificados como complicados y laboriosos. De manera que, resultaría conveniente hacer una consideración de los métodos y aplicaciones inteligentes que en la actualidad se están sugiriendo, con el fin de incorporar en la tecnología de nuestro país, los elementos que se juzguen necesarios, de tal suerte que el trabajo aquí presentado constituye una de tales propuestas.

REFERENCIAS

REFERENCIAS

Adárraga M., Pablo y Zaccagnini S., José Luis. (1994) Psicología e Inteligencia Artificial. Ed. Trotta. Madrid, España. Págs. 16-25, 30-32.

Aho, Alfred V.; Hopcroft, John E. y Ullman, Jeffrey D. (1983) Data Structures and Algorithms. Ed. Addison-Wesley Publishing Company. Nueva York. Estados Unidos. Págs. 198 y 199.

Aviña G., Rosalinda G. y Valles M., Ana G. (1991) Sensibilización de Parámetros en una Red Neural que reconoce Patrones en Bases de Datos. Tesis para Licenciatura. Universidad Tecnológica de México. Méx., D.F. Págs. 4-13, 18-21.

Coleman, Tom (1989) Expert Systems for the Data Processing Professional. Ed. NCC Publications. Inglaterra. Págs. 62-68.

Cuena, José; et. al (1986) Inteligencia Artificial: Sistemas Expertos. Ed. Alianza. Madrid, España. Págs. 123-131.

Chatain, Jean-Noël y Dussauchoy (1988) Sistemas Expertos, Métodos y Herramientas. Ed. Paraninfo. Madrid, España. Págs. 31-33.

Del Valle Toledo, Enrique (1987) Apuntes de Introducción a los Métodos Geofísicos de Exploración. U.N.A.M. Facultad de Ingeniería. Págs. 27-38.

Durkin, John (1991) "Designing an Induction Expert System" en AI Expert. Diciembre. Págs. 29-35.

Engel, Charles W. y Cran, Margaret. (1990) "Pattern Classification. A Neural Network Competes with Humans" en PC AI, Mayo/Junio. Págs. 20-22.

Freeman, James A. y Skapura, David M. (1993) Redes Neuronales, Algoritmos, Aplicaciones y Técnicas de Programación. Ed. Addison-Wesley/Díaz de Santos . Wilmington, Delaware, Estados Unidos. Págs. 18-23, 307-309.

Hilera González, José Ramón y Martínez Hernando, Víctor José (1995) Redes Neuronales Artificiales, Fundamentos, Modelos y Aplicaciones. Ed. Addison-Wesley Iberoamericana. Madrid, España. Págs. 2-11, 45-72, 219-252.

Kurzweil, Raymond (1994) La Era de las Máquinas Inteligentes. Consejo Nacional de Ciencia y Tecnología (CONACyT). Méx., D.F.

Laurence, Jeannette (1990) "Untangling Neural Nets" en Dr. Dobb's Journal. Abril. Págs. 38-44.

Matcho, Jonathan; et. al. (1995) Special Edition Using Delphi. Ed. Que Corporation. Indiana, Estados Unidos.

Medsker, Larry y Liebowitz, Jay. (1994) Design and Development of Expert Systems and Neural Networks. Ed. Macmillan College Publishing Company. Nueva York, Estados Unidos. Págs. 1-29, 33-37, 69-71.

Müller, Bernd y Reinhardt, Joachim. (1991) Neural Networks, An Introduction. Ed. Springer-Verlag. Berlín, Alemania. Págs. 2-4.

Nebendahl, Dieter (1988) Sistemas Expertos, Introducción a la técnica y aplicación. Ed. MARCOMBO Boixareu. Barcelona, España. Págs. 2, 27-39, 55-82.

Osier, Dan; et. al. (1996) Aprendiendo Delphi 2 en 21 días. Ed. Prentice Hall Hispanoamericana S. A. Edo. de México, México.

Patterson, Dan W. (1990) Introduction to Artificial Intelligence and Expert Systems. Ed. Prentice-Hall. Nueva Jersey, Estados Unidos. Págs. 126-145, 325-336.

Piatetsky-Shapiro, Gregory (1991) Knowledge Discovery in Databases. Prentice-Hall. Nueva Jersey, Estados Unidos. Pág. 5.

Rich, Elanie (1988) Inteligencia Artificial. Ed. G. Gili S. A. de C. V. Barcelona, España. Págs. 9-62.

Schildt, Herbert (1989) Turbo Prolog, Programación Avanzada. Ed. McGraw-Hill. México, D.F. Págs. 62-70.

Schildt, Herbert (1990) Utilización de C en Inteligencia Artificial. Ed. McGraw-Hill. Madrid, España. Págs. 15-40.

Schulemberger (1972) Interpretación de Perfiles. Volumen I-Fundamentos. Nueva York, Estados Unidos. Págs. 1-10, 77, 78.

Simons, G. L. (1987) Introducción a la Inteligencia Artificial. Ed. Díaz de Santos. Madrid, España. Págs. 175-208.

Thompson, Beverly y Thompson, William (1986) "Finding Rules in Data " en BYTE. Noviembre. Págs. 149-156.

Tucker, Allen B. (1987) Lenguajes de Programación. Ed. McGraw-Hill. Méx., D.F. Págs. 336-370, 406-437, 438-482.

Turban, Efraim. (1992) Expert Systems and Applied Artificial Intelligence. Ed. Macmillan College Publishing Company. Nueva York, Estados, Unidos. Págs. 21, 22, 24.

Universidad Tecnológica de la Mixteca (1992) Inteligencia Artificial en México. Méx., D. F. Pág. 52.

Winston, Patrick H. (1992) Artificial Intelligence (3a. Edición). Ed. Addison-Wesley Publishing Company. Nueva York, Estados Unidos. Págs. 47-60, 63-79