33
29.

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
"ARAGÓN"

"INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL"

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

P R E S E N T A N:

HILDA PATRICIA LOPEZ CRUZ
MARINA RESENDIZ CERON

MÉXICO AGOSTO 1997

DEDICATORIA

AL QUE NOS ACOMPAÑA EN CADA INSTANTE DE NUESTRA EXISTENCIA:
NUESTRO SEÑOR JESUCRISTO

A LA MEMORIA DE AQUEL QUE SIEMPRE PERMANECERA VIVO EN MI CORAZON Y EN MI PENSAMIENTO: M. G. C.

A FRANCISCA CRUZ H.

Y

A LA MEMORIA DE JOSE LOPEZ V.

MIS QUERIDOS PADRES

A MIS HERMANOS Y HERMANAS:

ADRIAN †	LUISA
VIRGINIA	JOSE
REYNA	Y
LILIA	FERNANDO.

PATRICIA L. C.

AGRADECIMIENTOS

A DIOS, DOY LAS MAS HUMILDES GRACIAS POR HABERME REGALADO LA VIDA

AGRADEZCO A MIS PADRES SUS ORACIONES, SU APOYO Y SU CARIÑO

A LA UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO POR DARME LA OPORTUNIDAD DE SER UNIVERSITARIA

AGRADEZCO ESPECIALMENTE AL INGENIERO AMILCAR MONTERROSA ESCOBAR POR ACEPTAR LA DIRECCION DE ESTE TRABAJO

AL INSTITUTO MEXICANO DEL PETROLEO POR FOMENTA Y APOYAR LAS ACTIVIDADES ACADEMICAS.

PATRICIA L. C.

DEDICATORIA

A MIS PADRES: VALENTIN RESENDIZ Y LEONOR CERON† POR DARME VIDA Y LIBERTAD PARA VIVIRLA.

A MIS HERMANOS: ADRIAN, NOE, SUSANA, VALENTIN, ISAAC E ISRAEL, PORQUE HAN SIDO LA COMPAÑIA QUE SIEMPRE LLEVO EN MI CORAZON PARA NO SENTIRME TAN SOLA.

A MI ESPOSO FRANCISCO J. RAMIREZ Y MI PEQUEÑO OCTAVIO CESAR PORQUE SU INFINITA PASIENCIA Y AMOR ME DIERON FUERZAS PARA TERMINAR ESTE TRABAJO

Y EN GENERAL A TODA MI FAMILIA POR EL APOYO BRINDADO DURANTE TODA MI VIDA

UN AGRADECIMIENTO MUY ESPECIAL A MI MADRE POR ENSEÑARME EL CAMINO A SEGUIR

GRACIAS

MARINA R. C.

AGRADECIMIENTOS

AGRADESCO A LA INSTITUCION QUE ME BRINDO EL APOYO PARA FORMARME NO SOLO A NIVEL ACADEMICO, SINO TAMBIEN A NIVEL PERSONAL, LA UNAM.

AL INGENIERO AMILCAR MONTEROSA ESCOBAR POR EL TIEMPO Y EL APOYO BRINDADO PARA LA REALIZACION DE ESTE TRABAJO.

A TODAS AQUELLAS PERSONAS QUE AL CREER O NO CREER EN MI ME DIERON FUERZAS PARA SUPERARME Y LLEVAR ESTA TESIS A SU TERMINO.

GRACIAS

MARINA R. C.

INDICE

INTRODUCCION

Sin lugar a dudas, la computación ha tenido un papel trascendental en el desarrollo científico y tecnológico del hombre moderno. Desde tiempos remotos, la necesidad de procesar datos ha llevado al ser humano a agudizar su creatividad; gracias a la cual ha podido desarrollar los elementos y técnicas que le sean útiles para poder elaborar mejor y más rápidamente su trabajo, es así como se descubren desde el ábaco hasta las más modernas computadoras, desarrollando con ello un estudio más intenso en lo que se refiere a la lógica matemática, la teoría de probabilidades, la teoría de los algoritmos, el álgebra booleana, etc.

Gracias a esta evolución se han logrado obtener sistemas capaces de auxiliar al hombre en sus tareas manuales y en los últimos tiempos, hasta en las intelectuales. La Inteligencia Artificial es la encargada de desarrollar las técnicas y las tecnologías que puedan ser aplicadas a un sistema computacional con el fin de crear los llamados "sistemas inteligentes".

El objetivo primordial de el presente trabajo es servir de apoyo en el estudio de la materia inteligencia artificial, la cual se imparte dentro de la Universidad Nacional Autónoma de México en la carrera de ingeniería en computación; por tal motivo se ha tratado de abarcar en su mayor parte el temario establecido por la institución, estructurando el trabajo en siete capítulos: los cinco primeros pretenden ilustrar lo que se denomina la parte teórica, en el primero de ellos, se dará una definición formal de la I. A., también se presentan las áreas en las cuales se divide la I. A.; así como una definición concreta de cada una de ellas. En este mismo capítulo se encontrarán los acontecimientos históricos más sobresalientes de esta ciencia y para finalizar se dedica un espacio a los más grandes investigadores en la materia, al igual que a las instituciones que han y siguen colaborando en el fortalecimiento y desarrollo de la I. A.

El segundo capítulo es una recopilación de las técnicas más importantes para el manejo de los datos utilizadas en la I. A. debido a que la información es lo más importante en el manejo de sistema, ésta debe estructurarse de manera eficaz y al mismo tiempo el sistema debe contar con mecanismos de búsqueda con la habilidad suficiente para un optimo desarrollo de sus funciones. Por lo anterior, en este capítulo se proporciona por un lado la explicación de los métodos para representar el conocimiento, es decir; la forma en la que podemos organizar la información con la que contamos, para posteriormente utilizarla con mayor facilidad y por el otro se exponen los tipos de búsqueda utilizados más comúnmente.

El tercer capítulo se dedica a una rama particular de la I. A. , los sistemas expertos; se ha otorgado este espacio en especial porque de todas las ramas que engloba la I. A. estos sistemas tienen un desempeño singular, ya que son capaces de almacenar todos los conocimientos que pueda tener un ser humano experto en cualquier especialidad, como consecuencia de esto han tenido una difusión en gran escala en prácticamente cualquier

actividad. Con el objeto de presentar de manera sencilla y rápida lo referente a los sistemas expertos, sólo se abordan de una manera general sus antecedentes históricos, se describen los componentes que forman a estos sistemas y el modo que interaccionan entre sí; además se proporciona la definición de sistema experto y algunos ejemplos de estos tipos de sistemas.

El aprendizaje es una habilidad propia del hombre, aún en nuestros días existen varias incógnitas sobre cómo se lleva a cabo este proceso, sin embargo, dentro del marco de la I A. se han establecido varios métodos de aprendizaje, los sistemas de I. A. utilizan estos métodos para asimilar los conocimientos de alguna manera ya que no cuentan con mecanismos naturales. Entre los tipos de aprendizaje que se han desarrollado, los más utilizados son: por estudio de diferencia, por experiencias, utilizando modelos, utilizando árboles de identificación, por perceptrones y por redes neuronales; estos tipos de aprendizaje van desde el manejo de ejemplos ya resueltos, hasta la simulación del comportamiento de las neuronas humanas al resolver un problema, pasando por la representación de diferentes maneras de los mismos. Toda esta información se encuentra en el cuarto capítulo.

En los sistemas diseñados bajo las técnicas de la I. A., se tiene un planteamiento de problemas casi siempre declarativo, por que ha provocado la creación de lenguajes especializados para la I. A. con la intención de desarrollar los sistemas de manera más adecuada para tal fin, que con cualquier otro lenguaje de alto nivel, algunos de estos lenguajes son el IPL-II, primer lenguaje de la I. A., sin embargo después surge el LISP el cual fue más popular y por lo tanto de mayor uso, tanto así que logró tener máquinas especializadas para su funcionamiento; otro lenguaje muy popular para tales fines es el PROLOG. También se han desarrollado diferentes herramientas para apoyar a los programadores e investigadores éstas son conocidas como sistemas vacíos (shell) que han tenido gran aceptación al igual que los lenguajes orientados a objetos. Lo referente al software utilizado en la I. A. se menciona en el quinto capítulo.

Para reafirmar los conceptos introducidos en los capítulos anteriores, en el capítulo seis se presenta un ejemplo de aplicación de un sistema experto, el cual orienta al usuario sobre el restaurante que más le convenga, considerando sus gustos, su presupuesto, la zona en dónde vive y otros requerimientos que él mismo proporciona; dicho sistema es desarrollado por medio de un sistema vacío (Vp-expert) con el fin de mostrar lo útiles que son estas herramientas para el desarrollo de sistemas expertos.

Para concluir con el presente trabajo en el capítulo siete se presentan algunas de las aplicaciones que tiene la I. A., para tener una idea general de la utilidad de esta ciencia y además conocer el trabajo que con su ayuda se facilita en magnitud considerable, además se dan a conocer algunas de las metas que se pretenden alcanzar a corto, mediano y largo plazo por los sistemas basados en las técnicas de la I. A., ciencia que por mucho tiempo tendrá una evolución constante que impactará y revolucionará al mundo.

CAPITULO I

INTRODUCCION A LA INTELIGENCIA ARTIFICIAL

La Inteligencia Artificial (I. A.) es una ciencia que se ha desarrollado a una velocidad enorme en las últimas décadas (gracias al avance en forma conjunta de los equipos y técnicas de computación); en las cuales a colocado un gran número de sistemas operacionales muy importantes y de gran trascendencia.

Todas las disciplinas que la I. A. engloba, se retroalimentan para lograr que surjan equipos y programas que van desde robot's, hasta reconocedores del lenguaje humano tal y como lo hablamos diariamente. Este avance se debe en gran medida a que el número de investigadores y de instituciones dedicadas a la I. A. ha crecido al darse cuenta de la importancia y utilidad que implica dicha ciencia en la vida del hombre.

1.1 DEFINICION

Desde tiempos antiguos, el hombre ha venido inventando y descubriendo diferentes técnicas y herramientas, para lograr un mejor nivel de vida. Una nueva ciencia de entre todos estos adelantos surgió aproximadamente en la década de los años de 1950; la llamada Inteligencia Artificial.

Actualmente existen diferentes definiciones de Inteligencia Artificial (I. A.) entre las que destacan las siguientes:

1.1.1 Inteligencia Artificial

"Tiene por objeto analizar los comportamientos humanos en lo relativo a la percepción, la comprensión y la decisión, con el fin de producirlos eventualmente con la ayuda de una máquina: el ordenador."[1]

"Se define como la ciencia que trata de la comprensión de la inteligencia y del diseño de máquinas inteligentes es decir, el estudio y la simulación de las actividades intelectuales del hombre (manipulación, razonamiento, percepción, aprendizaje, creación ...)."[2]

"Es el estudio de como lograr que los computadores hagan cosas que, por el momento, las personas hacen mejor."[3]

"Es la ciencia de fabricar máquinas que hacen cosas que requerían anteriormente de la inteligencia humana. "[4]

---

[1] J. P. Aubert, R. Schomberg, "Inteligencia Artificial." p. 90
[2] J. P. Sánchez y Beltrán, "Sistemas expertos: una metodología de programación." p. 5
[3] E. Rich, "Inteligencia artificial." p. 9
[4] Marvin Minsky, "Practical aplications of expert systems." p. 1

"Es una rama de la informática, que tiene como métodos para procesar información la representación del conocimiento usando símbolos en lugar de números y la heurística o reglas basadas en la experiencia. "[5]

Se puede observar claramente que en estas definiciones se manejan implícitamente dos disciplinas que son parte esencial en el desarrollo de la I. A., las cuales son: la Psicología y la Ingeniería.

La psicología básicamente trata del estudio del pensamiento del hombre para comprender el comportamiento de las personas, y la ingeniería que construye herramientas para la simulación de las actividades del hombre; es decir para mejorar los mecanismos de "inteligencia" en las máquinas, en lo que se refiere a la adquisición de conocimientos, la toma de decisiones, la percepción, y el razonamiento.

Ahora bien, se ha hablado mucho de la inteligencia, y aunque todos tenemos una idea de lo que es, no se puede dar una definición concreta, precisa o clara de la misma, ésto dificulta la programación de sistemas que pretendan imitar a la inteligencia humana. Por ejemplo, cuando se quiere calcular la derivada de una función se siguen ciertos pasos ya establecidos, los cuales se pueden programar en la computadora; pero si queremos dar una explicación formal a lo que soñamos la noche anterior, no podríamos indicar los pasos que seguimos para recordar lo que soñamos, simplemente decimos lo que soñamos. "...La gente no podía explicar como pensaba, pero podía decir lo que pensaba. "[6] .

Otros elementos difíciles de explicar formalmente son las reacciones de las personas frente a diferentes situaciones, la capacidad de entender mensajes que no sean del todo claros, el dar prioridades o jerarquías a los acontecimientos que nos rodean y por último reconocer tanto similitudes entre dos hechos a pesar de las diferencias que éstos tengan, como diferenciar eventos aunque sean similares. A todo esto es lo que llamamos sentido común. Como ya se mencionó estas son actitudes comunes para el ser humano, sin embargo existen tareas que los ordenadores o computadoras efectúan de manera más eficiente que el hombre, de entre las que destacan:

- Manipulación de grandes cantidades de información. En una base de datos se puede extraer la información de un individuo con todos los datos; dirección, teléfono, etc., lo cual a una persona se le dificultaría recordar toda la información que se requiere, así como la rápida localización del dato.

- Realización de operaciones numéricas de una manera más precisa y rápida. Cuando la operación es simple, el cálculo es rápido y fácil; pero cuando el grado de complejidad aumenta, es más difícil hacer el cálculo manualmente, no así para la computadora.

[5] Bruce G. Buchanan, citado en "A fondo: Inteligencia Artificial," Louis E. Frenzel Jr., p. 21
[6] Herbert Childt, "Utilizacion de C en la inteligencia artificial." p. 5

- Hacer procedimientos mecánicos repetitivos (en serie). El hecho de que la máquina no se canse implica que el desgaste físico no se manifieste en la máquina (como ocurre en el ser humano), con ello se garantiza que al finalizar el día de labores ésta siga trabajando al mismo nivel.

Después de indicar algunas definiciones de la I. A., presentaremos una definición que expresa de manera personal el significado de esta ciencia.

La inteligencia artificial: Es la ciencia que se encarga del estudio y simulación de algunas facultades tanto físicas como intelectuales propias del hombre, así como del diseño y desarrollo de dispositivos encaminados a realizar dichas actividades, siendo su principal herramienta de desarrollo la computadora.

De acuerdo a la definición dada puede decirse que la I. A. tiene dos grandes categorías de estudio: técnicas básicas y tecnologías. Las técnicas básicas constituyen la base sobre la que operan los mecanismos basados en I. A., las tecnologías son conjuntos de técnicas que ayudan a resolver tipos de problemas específicos.

Es importante mencionar que esta ciencia pretende desarrollar "máquinas inteligentes" sin reproducir en ellas la totalidad del proceso intelectual del hombre debido a que éste, más que procesar y ejecutar información, la entiende y analiza.

Conforme ha pasado el tiempo el concepto de inteligencia se ha venido entendiendo de manera diferente, por ejemplo, hace 60 años se consideraba inteligente a una persona que supiera leer y escribir, pero en nuestros días saber solamente ésto ya no es suficiente, este tipo de evolución tiene una semejanza con los programas de I. A., porque cuando se logra que un sistema supere algunas capacidades del hombre, dicho sistema ya se puede considerar fuera de la misma. Un ejemplo claro son las calculadoras de bolsillo, que en su momento se llegaron a creer extremadamente sofisticadas, y en la actualidad ya se consideran comunes.

1.2 PRINCIPALES RAMAS DE ESTUDIO

La I. A. es una ciencia bastante extensa, es por ello que su estudio se divide en áreas más especializadas que varían según el autor. Aquí presentaremos las más importantes:

1.2.1 Aprendizaje automático

Es una técnica básica, ésta es trascendental para la I. A., ya que investiga y desarrolla la manera en que un programa pueda aprender por si sólo al adquirir nuevos conocimientos basándose tanto en los hechos que

tiene la base de conocimientos* (modelos), como en nuevos hechos (experiencias), para relacionarlos y obtener resultados que antes no formaban parte del sistema.

1.2.2 Búsquedas inteligentes

La gran cantidad de posibles soluciones que se tienen para un mismo problema crea la necesidad de desarrollar ciertos mecanismos que ayuden a minimizar el tiempo de localización de la solución deseada y a los que se les ha dado el nombre de búsquedas inteligentes. Estas búsquedas consisten en recorrer la base de conocimientos que contiene todas las posibles soluciones; también llamada "espacio de búsqueda" para encontrar un conocimiento que se adapte a los hechos conocidos y que se considere como una solución optima. Estas búsquedas son técnicas básicas.

1.2.3 Procesamiento del lenguaje natural (P.L.N.)

Es una técnica básica y es una de las áreas más complejas dentro de la I. A. ya que se encarga de la comprensión del lenguaje natural propio del hombre, su objetivo principal es lograr la comunicación de manera directa con las máquinas; algunos factores que aquí se involucran son: interpretación, comprensión, traducción y generación de frases. No es un problema sólo de gramática (hablar y escribir correctamente) ni de traducción palabra por palabra (vocabulario) sino que depende en buena medida del tema que se este tratando para establecer el significado conceptual de cada palabra.

1.2.4 Programación automática

Estudia la manera de lograr que un programa genere automáticamente otros programas que resuelvan problemas anteriormente especificados por el usuario. La programación automática no sólo genera programas, también apoya al programador en cualquier parte del proceso de programación ya sea para diseñar, verificar, depurar, evaluar u optimizar. Es considerada como una tecnología.

1.2.5 Representación del conocimiento

Como su nombre lo indica, la representación nos auxilia para organizar optima y eficazmente los conocimientos que el sistema va a utilizar para poder dar solución a la diversa gama de problemas que se le presenten. Esta representación va de acuerdo al tipo de problema que se quiera solucionar así como al criterio del programador, estos dos parámetros son los lineamientos para poder elegir el tipo de representación que vayamos a utilizar. Forma parte de las técnicas básicas de la I. A. Con ayuda de esta técnica podemos solucionar problemas bastante complejos, sin embargo, para lograrlo necesitamos de una gran cantidad de conocimientos y más aun de procesos o mecanismos para manipularlos.

---

* Contiene la información especifica de un campo o especialidad.

1.2.6 Robotica

Es una tecnología, la cual tiene por objeto el diseñar y desarrollar máquinas que sean capaces de realizar procesos mecánicos y manuales, dichas máquinas contienen un sistema de control y un sistema sensorial; la interacción de estos sistemas le permiten responder a los cambios que han surgido a su alrededor y de esta manera muestran un nivel de "inteligencia".

1.2.7 Sistemas expertos (S. E.)

Es la tecnología que se encarga del diseño y desarrollo de programas los cuales tienen como fin el imitar el comportamiento intelectual de un experto en una determinada área. También se les conoce como sistemas fundamentados en el conocimiento. (El capitulo III está dedicado a esta rama).

1.2.8 Visión por computadora

Rama que estudia la identificación de objetos a través de un dispositivo el cual sea capaz de percibir e interpretar imágenes, la visión por ordenador es muy utilizada en el área de la robotica. Su objetivo primordial consiste en proporcionar a los ordenadores una herramienta útil para la interpretación de los objetos captados y la comprensión de lo que visualiza. Esta rama es considerada como una tecnología.

1.3 ANTECEDENTES HISTORICOS DE LA I.A.

El principio de la I. A. no es muy específico, algunos autores consideran que se inició desde antes de la segunda guerra mundial con la aparición de la lógica formal y la psicología cognoscitiva, también se dice que fue cuando se desarrollo la primera computadora o cuando Alan Turing creó la máquina con capacidad de almacenamiento de programas, y otros más señalan que el primer paso formal en el desarrollo de la I. A. fue la conferencia de Dartmouth.

Existen otros acontecimientos igual de importantes, ya que desde 1930 Leibniz, Boole, Hilbert, Godel y Church dejaron ver la existencia de problemas cuya solución no es algorítmica (no se soluciona a través de cálculos numéricos, sino a través de representaciones simbólicas). Entre 1942 y 1943 Charles Babbage (también conocido como el padre de la computación) inventa su ya conocida máquina analítica. En 1942 nació la cibernética desarrollada por Winner, que estudia los fenómenos de retroalimentacion de información en el cerebro; basados en estos mismos conceptos surgen los estudios e investigaciones de las redes neuronales (1942) que pretenden modelar la estructura interna del cerebro con el fin de reproducir algunas de sus funciones.

Todos estos acontecimientos son antecedentes para establecer una cronología dentro de la historia de la I. A. que dividiremos en cinco etapas fundamentales: cada etapa consta de una década, comenzamos con los años 50's, hasta abarcar gran parte de los años 90's.

1.3.1 Década 1950

A principios de esta década en 1950 Alan Turing (1912-1954) escribe un documento llamado "Computer Machinery and Intelligence" por el cual se le conoce como el padre de la I. A. y en este mismo año da a conocer su famoso "juego de imitación", mejor conocido como la prueba de Turing o Test de Turing, éste consiste en ver al S. E. como una caja negra para determinar si su comportamiento semeja al de una persona, para ésto se utilizan tres cuartos cada uno con un ordenador, en el primer cuarto se encuentra un experto humano, en el segundo el S. E. que se está examinando y en el tercero una persona llamada examinador que desconoce la localización exacta de los dos anteriores; el examinador debe determinar basándose en preguntas y respuestas, quién es el S. E. y quién el experto humano. Una regla importante del juego es que sólo el entrevistador está obligado a contestar con la verdad, ésto es para evitar que se pregunte "¿eres un S. E.?", si contestarán ambos con la verdad, el juego habría terminado. Si el examinador no es capaz de determinar quién es el experto humano, se dice que el sistema es un verdadero S. E.

En esta década tiene lugar una reunión de científicos (1956) con diferentes especialidades: ingenieros, matemáticos, psicólogos y neurólogos, también llamada Conferencia de Dartmouth en la que el común prevaleciente entre los integrantes de la conferencia era la utilización de la computadora para llevar a cabo sus investigaciones, de entre las cuales figuraba el de simular algunos procesos de la inteligencia humana. En dicha conferencia surge una nueva ciencia gracias a la mezcla de distintos elementos de las disciplinas representadas en esta reunión; a esta nueva ciencia se le dio el nombre propuesto por Jonh McCarthy: Inteligencia Artificial. El objetivo principal de esta reunión fue que todos los rasgos de la inteligencia del hombre incluyendo el aprendizaje pudieran ser simulados por el ordenador. Si bien es cierto que en esta conferencia no se dieron grandes resultados científicos, es importante indicar que es en ella donde se le dió el nombre de lo que hoy conocemos como I. A.

La tecnología de las Redes Neuronales es parte fundamental en estos años; aunque ya se había empezado a utilizar desde 1943, fue hasta 1957 cuando se logró diseñar una máquina capaz de percibir, asociar y dar una respuesta ante un problema específico, ésta recibió el nombre de Perceptrón, ideado por Frank Roseblett de la Universidad de Cornell y aunque fue aceptado con optimismo, se dejó ver en el libro escrito por Minsky y Paper que ese tipo de máquinas sólo eran optimas para resolver problemas poco complejos. Otro ordenador elaborado bajo esta misma tecnología fue el NETTALK, concebido por Terry Sejnowsky diseñado para aprender a leer.

Aproximadamente en el año de 1957 apareció el primer lenguaje orientado a la I. A. el IPL II elaborado por Allen Newell, J. C. Shaw y Herbert Simon. Aunque fue desarrollado especialmente para esta ciencia no fue acogido por los científicos en la materia no así el LISP que se dio a conocer un año después, creado por John McCarthy (abordaremos más sobre el LISP en el capítulo V).

1.3.2 Década 1960

En estos años se analizan los problemas de explosión combinatoria y empiezan a surgir algoritmos para encontrar la mejor solución en el menor tiempo posible (algoritmos de poda), también es en este decenio cuando aparece el principio de la resolución automática (1955) formulada por J. Alan Robinson que es la base para el lenguaje PROLOG.

Estos avances ayudaron para que aparecieran programas tales como el juego del ajedrez (Newell y Simon), el "Logic theorist" (Newell y Simon), siendo este último utilizado en la comprobación de teoremas matemáticos; el Macsyma que es un programa para realizar cálculos matemáticos y un programa de apoyo a los psiquiatras escrito por Joseph Weizenbaum llamado ELIZA, por citar algunos. A pesar de su aparente éxito este programa no se considera "inteligente" ya que lo único que hace es preguntar utilizando parte de las mismas frases que se le dan, aunque llegó a desorientar a algunas personas que creían en realidad que estaban conversando con un psicólogo. Su propio diseñador lo desacredita en su texto "El Poder de la Computadora y la Razón Humana".

Es también en estos años cuando aparece el S.E. Dendral (1967) de la Universidad de Stanford que se basa ya en la representación del conocimiento y es considerado como el pionero de los Sistemas Expertos, su función es determinar la estructura química de los compuestos orgánicos.

Los años de la década de 1960 son importantes porque se genera un cambio significativo en la manera de ver cómo se podrían abordar mejor las actividades que pretende simular la I. A. En los 50's se analizó el funcionamiento del comportamiento humano desde su base, el cerebro, a través de modelos que simulaban estas actividades; pero al no obtener éxitos muy notables, se decide abordalas de otra manera, tratando ahora de imitar el comportamiento ya exteriorizado como lo señala J. P. Sánchez: "...No conociéndose los mecanismos generales de resolución de la mente humana se penso en simular los mecanismos para campos muy concretos del conocimiento. Es decir, se limita la forma externa o comportamiento aparente, que es precisamente el enfoque completamente opuesto a la línea de investigación de las redes neuronales."[6] . Surgen entonces las diferentes representaciones del conocimiento, tipos de búsqueda, etc. (ver capítulo 2), que forman parte esencial de los llamados S. E. considerados como lo representativo en la tercera etapa de evolución de la I. A.

[6] J.P.Sánchez y Beltrán, op. cit., p. 15

1.3.3 Década 1970

La existencia de máquinas poco adecuadas fue un obstáculo para el desarrollo de la I. A. e incluso se considera aventurado el querer implementar este tipo de sistemas con estas herramientas tan escuetas; pero ya en los años de 1970 en adelante, empezaron a aparecer computadoras con capacidades mucho mayores que podían manipular los datos a nivel simbólico y no a nivel numérico, este adelanto fue un gran paso porque originó grandes avances en el procesamiento del lenguaje natural, la representación del conocimiento, la resolución de problemas, etc., estos éxitos contribuyeron a su vez al enriquecimiento de algunas áreas como la robotica y ayudaron en el seguimiento de los S. E., primer producto comercial de la I. A.

Hacia 1972 aparece el lenguaje PROLOG, realizado por el francés Alain Colmerauer; al igual que el LISP está diseñado para trabajar bajo los lineamientos de la I.A., trae consigo mejoras como la base de datos incorporada, rutinas de retroseguimiento y una sintaxis muy simple que le valió el ser preferido por los japoneses en sus computadoras de la quinta generación. Antes de este anuncio (1981), el PROLOG no era muy aceptado en E. U., pero después de ésto ha tenido mayor cabida que el propio LISP.

Los programas sobresalientes en este periodo son el PROSPECTOR (1974) realizado para encontrar prospecciones mineras y el MYCIN (1977) cuya función es el apoyar en el diagnóstico y terapia de enfermedades infecciosas de origen bacteriano. Ambos sistemas son expertos con logros palpables de suma importancia.

Por último, ocurre un suceso trascendental para los amantes de la I. A. en 1970 se publica la primer revista especialmente dedicada a este tema, conteniendo los últimos avances logrados hasta entonces.

1.3.4 Década 1980

Al darse a conocer los éxitos ya obtenidos en la I. A., el interés por esta ciencia se desborda nuevamente; en consecuencia aparecen nuevos proyectos con un mayor apoyo por parte de los países más industrializados para desarrollar sistemas en áreas prácticas de carácter general.

A su vez se crean diferentes centros de investigación en la materia y se celebran eventos diversos como: el Primer Encuentro Nacional sobre la I. A. en 1984 y el Primer Encuentro Internacional donde se trataron temas de la ingeniería del conocimiento en 1985; entre otros.

En esta década, la I. A. toma fuerzas para lograr uno de sus más firmes objetivos: el dotar de "inteligencia" al computador.

1.3.5 Década 1990

En ésta década se crean y a su vez se mejoran algoritmos basados en redes neuronales, con el fin de elaborar mecanismos de aprendizaje que simulen más acertadamente a lo que se sabe ocurre en el cerebro humano -merced a los trabajos de Santiago Ramón y Cajal- en el sentido de que dota a las personas de abundantes capacidades como por ejemplo: el de reconocer un rostro con diferentes expresiones y con ligeros cambios físicos, o el poder expresarse por medio de un lenguaje, entre otras muchas facultades únicas en el ser humano, y que hasta hace algunos años no se habían podido reproducir en las computadoras, sin embargo, en la actualidad y con la tecnología de las redes neuronales ya se han tenido logros importantes en el reconocimiento y clasificación de patrones confusos o poco claros. En éstos últimos años, los eruditos en la materia han diseñado una sofisticada red neuronal capaz de leer en voz alta.

En esta década se viene presentando un uso más extenso en los algoritmos genéticos, gracias a éstos se puede tener un campo mucho más amplio de posibles soluciones de un determinado problema. Esta técnica se basa en la teoría de la evolución, más propiamente dicho esta basada en el mecanismo de selección usado por la naturaleza, en la cual señala que en una población los seres más aptos son los tienen mayor probabilidad de sobrevivir. Aun cuando los algoritmos genéticos imitan los efectos de la selección natural, operan a niveles muy diminutivos en comparación con la evolución biológica; sin embargo, actualmente, se sabe que "... la aplicación más común de los algoritmos genéticos ha sido la solución de problemas de optimización, en donde han mostrado ser muy eficientes y confiables."[9]

1.4 INVESTIGADORES

Como ya hemos mencionado, Alan Turing fue uno de los más grandes investigadores de la I. A., sin embargo es necesario reconocer los estudios de algunos otros científicos que en este mismo contexto han hecho aportaciones fundamentales a la I. A. y que gracias a sus estudios dicha ciencia se ha ido fortaleciendo.

Los siguientes renglones se encargan de darnos a conocer brevemente algunos de los investigadores así como su trabajo desarrollada dentro de la I. A. y en algunos casos de la informática en general.

1.4.1 Alex Bernstein

Al igual que Samuel, creo un "jugador", pero esta vez de ajedrez en el cual utilizó la heurística para buscar la mejor solución (o los mejores movimientos) y empezó a utilizar algunos métodos para eliminar posibilidades, esta técnica ayudó a mejorar su programa. Bernstein no alcanzó el nivel de desarrollo de Samuel

[9] Carlos A. Coello, "Soluciones avanzadas", año 3 num. 17 p.5

entre otras razones por las diferencias de los mismos juegos, sin embargo, los métodos que utilizó ahora forman parte esencial para la I. A.

1.4.2 Allen Newell y Herbert Simon

Aquí debemos mencionar a Oliver Selfridge, quien elaboró un programa de reconocimiento de patrones en el cual Newell puso un gran interés, y en la actualidad es muy utilizado en las diferentes áreas de la I. A. Lo que hicieron en forma conjunta Allen y Herbert fue el desarrollo de un programa que usaba razonamiento heurístico y procesaba la información, este es el Logic theorist. También estos dos científicos en colaboración con un tercero (J. C. Shaw) fueron los creadores del Lenguaje de Procesamiento de Información IPL II que apareció antes del LISP, por último cabe mencionar que Newell se interesó en las relaciones entre computador y cerebro, gracias a lo cual concibió la idea de que podía desarrollar un estudio científico que consistiría en elaborar un patrón del comportamiento humano en un computador.

1.4.3 Arthur Samuel

A pesar de su ingenuidad al creer incapaz a un ordenador para "aprender" por sí sólo, creó un programa que precisamente hacía ésto, al que se le llamó "Jugador de damas de Samuel" que fue el primero en aprender de sus propios errores, lo que le dió cabida en la I. A. Este sistema ideado por Samuel en 1947, jugaba al principio a un nivel básico, pero para su gran sorpresa, en 1961 ya era uno de los mejores jugadores de damas.

1.4.4 Claude Shannon

Apoyado en el álgebra booleana describió como se comportan los circuitos eléctricos en conmutación; sus aportaciones ayudaron al desarrollo de la ciencia informática y condujeron al sistema binario de almacenamiento de la información, lo que hoy en día es utilizado en el computador digital. El pensaba usar al computador en el juego del ajedrez y a pesar de que abandonó sus estudios, sus ideas escritas en varios artículos, fueron la base para varias investigaciones realizadas dentro del marco de la I. A.

1.4.5 John McCarthy y Marvin Misky

Ambos fueron los organizadores de la ya mencionada Conferencia de Dartmouth, al primero de ellos se le debe la creación del procesador de listas (LISP), lenguaje usado para la programación de sistemas de I. A.; otra de las aportaciones de este científico en el renglón de la informática es, sin lugar a dudas el proceso conocido como tiempo compartido que como se recordará es el que permite utilizar un solo computador por varios usuarios al mismo tiempo.

Misky compartió con McCarthy la fundación de uno de los centros de mayor prestigio en investigaciones acerca de I. A.: el Laboratorio de Inteligencia Artificial. A Minsky se le deben los estudios de organización y representación de estructuras del conocimiento que es una de las áreas de mayor utilización en diferentes técnicas de la I. A. Estos dos científicos han sido ganadores del premio Turing, que proporciona la Association for Computing Machinery (ACM) a aquellos científicos innovadores en la ciencia de la informática.

1.4.6 Norbert Wiener

Todos sabemos que desde la época de Newton, los científicos realizaron investigaciones acerca de los procesos de la transferencia de energía, apoyándose en esta teoría Norbert Wiener propuso un modelo en el cual sugería ya no la transferencia de energía, sino la transferencia de información, esto contribuyo en mucho a la mejor modelación de fenómenos describiendo este tipo de transferencia; a la cual llamo Cibernética. La Cibernética esta enfocada principalmente a las similitudes funcionales entre los hombres y las computadoras, aunque estas semejanzas no fueron muy claras para Wiener, fueron unas de las primeras investigaciones realizadas en favor de la I. A.

1.5 INSTITUCIONES

No podemos dejar a un lado los trabajos elaborados en torno a la I. A. tanto por parte de universidades como de diversas compañías, ya que sus estudios han sido trascendentales en la propagación y desarrollo de esta ciencia. A continuación mencionaremos algunas de las universidades que gracias a sus investigaciones han logrado tener prestigio a nivel internacional en lo referente a la I. A., también se dará a conocer las compañías más importantes involucradas en los estudios de I. A.

1.5.1 Instituto Tecnológico de Massachusetts (MIT)

Son muchos los centros educativos que han llevado a cabo indagaciones acerca de la I. A. de diferente índole, de entre las cuales podemos citar al Instituto Tecnológico de Massachusetts (MIT) ya que fue en él donde Jonh McCarthy, Warren McCulloch, Claude Shannon y Norbert Wiener, entre otros realizaron algunos de sus proyectos descritos anteriormente, además de estos científicos el MIT albergó a otros personajes importantes como son: Daniel Bobrow, creador de un programa de lenguaje natural que lee los problemas algebraicos, los transforma en ecuaciones y los resuelve; Bert Raphael autor del SIR (Recuperación de Información Semántica), sistema que puede reunir hechos y hacer deducciones gracias a su diseño el cual le permite elaborar preguntas y de esta manera obtener mayor información, también fue colaborador en la creación de un robot móvil llamado SHAKEY. Además de estos estudios, en el MIT se siguen investigando los campos de la representación del conocimiento, sistemas expertos, robótica, visión por ordenador, reconocimiento del habla y resolución de problemas.

1.5.2 Universidad Carnegie-Mellon (CMU)

Esta universidad tiene una importancia singular debido a los estudios realizados por Newell y Simon acerca del procesamiento de la información. Hubo algunos proyectos desarrollados bajo las instalaciones del CMU como los realizados por Feigenbaum y Raj Reddy, al primero se le atribuye la creación de DENDRAL con el que dio preámbulo a la aproximación basada en reglas que hoy en día es muy común en los S. E. Por su parte

Reddy contribuyó en la elaboración de HEARSAY que es uno de los primeros elementos para la investigación del entendimiento y comprensión del habla. No todo termina aquí, el CMU continúa indagando en áreas como son visión por ordenador, reconocimiento del habla, automatización de fábricas, y otros proyectos más.

La I. A. le debe mucho a diferentes instituciones de las cuales mencionaremos a la Universidad de Stanford, ya que ésta va a la vanguardia en estudios de los S. E. basados en reglas y su producto más afamado en este renglón es el MYCIN. No sólo en instituciones de E. U. se indaga en el terreno de la I. A. sino que también en diferentes partes del mundo como lo son las universidades de Tokio, Toronto y Marsella.

1.5.3 Digital Equipment Corporation

En lo referente a las compañías que han participado en la evolución de la I. A. figuran el DEC (Digital Equipment Corporation) por ser el pionero en la fabricación de los miniordenadores que como es bien sabido, tuvieron un mejor costo comparados con los grandes ordenadores de IBM. Una de las importantes producciones de la DEC es el ordenador llamado VAX debido a que fue muy utilizado en los laboratorios de I. A. antes de que aparecieran las llamadas máquinas LISP así como el VAX 11/780 ya que competía con los grandes ordenadores en cuanto a los cálculos. La DEC es fabricante de hardware de apoyo en la ciencia de la I. A. y ha contribuido en el desarrollo de los primeros S. E. comercializados.

1.5.4 Texas Instruments

TI Trabaja en distintos proyectos para el departamento de defensa de los E. U. encaminados a implementar actividades de las cuales podemos citar la asistencia al mantenimiento de equipos, y el desarrollo de un microprocesador LISP (en una sola pastilla), con un potencial diez veces superior a los procesadores que contienen decenas de chips.

1.5.5 Compañía Schlumberger

Investiga en diversas áreas gracias a sus dos centros especializados, en uno de ellos se realizan estudios básicos en el terreno de la visión por ordenador y representación del conocimiento y el otro se encarga del desarrollo de S. E. siendo el más conocido el programa que interpreta datos geológicos, este programa sirve de apoyo en las exploraciones que realiza la compañía en la prospección del petróleo.

CAPITULO II

TECNICAS DE REPRESENTACION PARA EL MANEJO DE DATOS EN UN SISTEMA DE INTELIGENCIA ARTIFICIAL

Como ya se ha mencionado, la Representación del Conocimiento es muy útil para la solución de problemas; siempre y cuando se tengan los conocimientos suficientes, y además que éstos se encuentren ordenados para que el acceso a los mismos se haga de manera rápida y sencilla. Con el objeto de crear mecanismos que cumplan con estos requerimientos, los ingenieros del conocimiento[*] se han dado a la tarea de establecer técnicas para representar el conocimiento en un computador; basándose en los estudios que ha realizado la psicología cognoscitiva en lo referente a la creación de modelos teóricos de representación del conocimiento que se asemejen al del cerebro humano.

2.1 REPRESENTACION DEL CONOCIMIENTO

Existen varias maneras de representar el conocimiento cada una con sus ventajas y desventajas propias; es decir, hay algunas formas que han dado buenos resultados para determinadas labores pero para otro tipo de tareas ya no surten los mismos efectos dados anteriormente. De aquí se deduce que debemos escoger la representación más apropiada según nuestros requerimientos, esta elección es de gran importancia, puesto que la eficiencia del sistema dependerá en gran medida del método escogido.

En esta sección del capítulo, prestaremos atención a algunos tipos de representación del conocimiento como son: reglas de producción, sistemas de reacción y redes semánticas; también abordaremos de manera general la lógica proposicional y la lógica de predicados.

La lógica es una forma bien definida para representar el conocimiento; no sólo se reduce a las tablas de verdad que muchas veces utilizamos, sino que es una disciplina que proporciona las herramientas para poder describir hechos del mundo real, para ello utilizamos proposiciones a las cuales les damos valores "verdadero" o "falso". Dentro de los sistemas de lógica más comunes encontramos la lógica de proposicional y la lógica predicados.

2.1.1 Lógica proposicional

La lógica proposicional es la lógica más simple, nos permite determinar si una proposición cualquiera es cierta o falsa; es decir, nos facilita los elementos para demostrar los valores de verdad de cada afirmación que se haga acerca de objetos específicos. Para entenderlo mejor consideremos las siguientes expresiones:

1.- La numeración es infinita. (V)
2.- La vida promedio del ser humano es 150 años. (F)

[*]El ingeniero del conocimiento es un especialista en I.A. el cual se encarga de organizar en la computadora los conocimientos proporcionados por el experto en el tema que se esté abordando.

Esta lógica trata también de las llamadas expresiones compuestas, aquellas que se enlazan por medio de los conectivos "..."y", "o", "implica" y "equivalencia".[1] Podemos notar que hay diferentes reglas para ampliar la veracidad o falsedad de las expresiones, todo va a depender del conectivo lógico que utilicemos. Por ejemplo: si tenemos una proposición 'p' cierta y otra 'q' falsa, al formar una expresión compuesta con el conectivo "o" esto es 'p o q' esta será cierta, sin embargo si el conectivo es "y", 'p y q' la expresión será falsa.

2.1.2 Lógica de predicados

La lógica de predicados es conocida como el cálculo de predicados y es una ampliación de la lógica proposicional, debido a que esta última tiene ciertas limitaciones, al utilizar proposiciones ya creadas ó bien definidas, que impiden hacer generalizaciones. Como medio para solucionar lo anterior, la lógica de predicados utiliza sentencias en las cuales aparecen objetos; o variables siendo éstos la unidad fundamental de la lógica de predicados, a la cualidad del objeto se le conoce como predicado. Por ejemplo en la siguiente expresión:

" TODO NUMERO NATURAL ES MAYOR QUE CERO "

Aquí la variable es número y debemos observar que no se refiere a un número específico sino a todos aquellos que pertenecen al género de números naturales, es aquí donde radica la potencialidad del cálculo de predicados, ya que debido a la utilización de variables, permite generalizar y especificar las proposiciones. Continuando con la frase anterior, ésta quedaría expresada en el cálculo de predicados como:

NUMERO NATURAL (X) Consecuentemente nos lleva a:
MAYOR QUE CERO (X)

Así la variable "X" abarca a todos los números naturales; resulta evidente que al sustituir por un número natural cualquiera obtendremos una proposición lógica verdadera.

Otro punto importante que debemos resaltar del cálculo de predicados es que éste utiliza dos cuantificadores, éstos son:

$\forall$ (todo)

$\exists$ (existe)

Estos dos tipos de cuantificadores se utilizan para expresar proposiciones como las siguientes:

1. $\forall x$ perro (x) es mamífero (x)
2. $\exists x$ perro (x) y nombre (x, duque)

[1] Enrique Castillo, "Sistemas expertos: Aprendizaje e incertidumbre." p.44

La primera proposición se traduce como: "Todos los perros son mamíferos" y la segunda como: "Hay un perro llamado duque".

Como podemos observar, la lógica nos dota de proposiciones falsas o verdaderas, no obstante en nuestro entorno real las cosas no siempre suceden así, ya que en algunas áreas de la vida la validez o en su defecto la negación de expresiones dependen de una cierta probabilidad en la cual se deben considerar muchas cosas que juegan un papel importante en la obtención del valor calificativo que se les va a dar, este punto es muy común para nosotros debido a que en la vida cotidiana nos enfrentamos con situaciones de estas características (eventos probabilísticos), por ejemplo: si a la hora del desayuno observamos la taza de una determinada persona, pensaríamos que está tomando café, pero también podría ser té, la mayoría de nosotros manejamos este tipo de eventos con cierta facilidad sin embargo para lograr que la computadora las asimile es muy complicado. Para estos casos se utilizan dos grandes áreas: la probabilística y la lógica difusa, la primera como su nombre lo indica, hace uso de la probabilidad de ocurrencia de diferentes eventos para obtener una respuesta y la segunda se refiere a la evaluación de proposiciones lógicas que no tienen bien definido sus valores.

2.1.3 Reglas de producción y sistemas de reacción

Las reglas de producción son una forma típica de representar el conocimiento en los S. E. La estructura básica de las reglas es "Si...entonces" (if...then...) en donde la parte posterior a la palabra SI, se le conoce como antecedente o premisa y a la parte posterior a ENTONCES se le llama consecuente o conclusión. De esta manera se consiguen expresiones como las siguientes:

SI el sensor se enciende ENTONCES activar la alarma
SI la pista esta vacía ENTONCES permitir aterrizaje

Teniendo en cuenta esta estructura, podemos formar términos completos; es decir, sentencias compuestas por uno o más conectivos, ya sea "*y*" u "*o*" esto nos lleva a obtener expresiones como las que se muestran a continuación:

SI tiene alas *y* grazna *o* tiene plumas *y* vuela ENTONCES es una ave
SI es mamífero *y* maúlla *o* tiene pelo *y* ladra ENTONCES es un animal doméstico

Observamos claramente que con éste tipo de enunciados se va haciendo más compleja la base de conocimientos; a pesar de ésto las reglas de producción tienen numerosas ventajas sobre otras formas de representar el conocimiento, dichas ventajas son la maneabilidad de las reglas, es decir, que pueden modificarse de manera relativamente sencilla, ésto implica hacer cambios, ampliarlas y en ocasiones hasta suprimirlas sin alterar considerablemente la base del conocimiento, la facilidad de agrupación que posee y su sencillez.

En casos particulares de los sistemas basados en reglas, los modelos "ENTONCES" indican acciones a realizar, sustituyendo así a las afirmaciones. A esta forma de sistema se le conoce como "sistemas de reacción"[2], debido a que obedecen al antecedente ejecutando la acción.

Un ejemplo muy simple de la utilización de estas sentencias es el siguiente:

Supongamos que se esta diseñando un robot para llenar cajas con botellas de aceite y que tenga la capacidad de determinar cuando éstas estén completas y tomar la siguiente caja, para lograr que el robot lleve a cabo esta actividad se hace uso de una sentencia similar a la siguiente:

SI el número de botellas es igual a 24 ENTONCES toma otra caja.

Como estas sentencias existen muchas otras utilizadas principalmente en los procesos repetitivos con fines industriales.

Las desventajas que se le atribuyen a las reglas de producción es la utilización de metareglas, "una meta-regla (una "regla acerca de otra regla") nos guía en la ejecución de un sistema experto determinando bajo qué condiciones tenemos que considerar unas reglas en lugar de otras."[3] debido a la dificultad que tiene para fijar relaciones, la velocidad en el proceso de inferencia se ve afectada por el número de reglas que aumentan considerablemente, la posibilidad que tiene de aceptar repeticiones y en algunos casos hasta oposiciones.

2.1.4 Redes semánticas

Las redes semánticas fueron presentadas por Quillian, se fundamentan en la forma en que relaciona los conocimientos la mente humana, fue la primera representación que se desarrolló. Están compuestas de un conjunto de nodos ú objetos y por arcos, ramas o enlaces que indican las relaciones existentes entre los nodos por medio de una leyenda colocada encima de dichos arcos. Los nodos representan hechos o conocimientos.

Para tener una idea más clara de este tipo de representación se presenta una gráfica sencilla de una red semántica en la figura 2.1

[2] Patrick Henry Winston , "Inteligencia artificial." p.144
[3] Henry C. Mishkoff, "A fondo: Inteligencia artificial." p.64

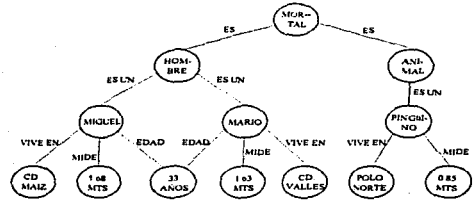


FIGURA 2.1

Existen reglas que facilitan deducir hechos por ejemplo para la red mostrada como pingüino es un animal y un animal es mortal, entonces se dice que el pingüino también es mortal.

Por medio de las redes semánticas, es bastante fácil responder a preguntas elaboradas en torno al tema que en ellas se está representando. Por ejemplo, en la red mostrada en la figura 2.2 es muy sencillo responder a preguntas como ¿Huatulco es centro turístico de México?, ¿el río Tamuín está en Brasil?, ¿el Amazonas está en Guadalajara?, etc.

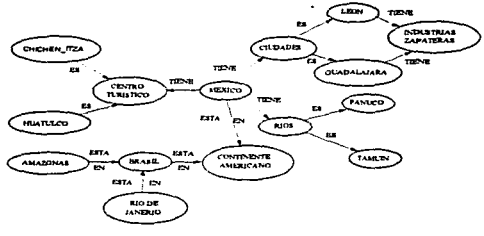


FIGURA 2.2

En los dos ejemplos presentados en está sección es fácil referirse a las redes semánticas de una manera gráfica y bien relacionada, obviamente, no podemos representar el conocimiento de está forma dentro del computador; para ello las redes semánticas tienen un caso peculiar llamado *ternas objeto-atributo-valor*, comúnmente en ellas se eliminan los enlaces y son sencillas de programar en lenguajes utilizados en I. A. Bajo este mismo concepto una porción de red de la figura 2.2. quedaría representada de la siguiente forma:

segmento de red semántica:
MEXICO tiene CENTRO TURISTICO es CHICHEN-ITZA
expresada en la terna:
MEXICO Centro turistico CHICHEN-ITZA

Algunas de las ventajas al utilizar las redes semánticas son su generalización que permite describir hechos y objetos. Es conveniente utilizarlas cuando se requiere representar un conocimiento que está estructurado jerárquicamente. También la herencia forma parte de sus ventajas puesto que algunos nodos tienen esta fundamental característica a la cual se le atribuye que la información contenida en un nodo no se tenga que repetir a otro ú otros nodos que de alguna manera dependen del nodo de mayor jerarquía.

Ahora bien considerando las desventajas de este tipo de representación del conocimiento tenemos que son muy poco manejables por lo que resulta complicado realizar modificaciones en ellas, otra de sus desventajas aparece cuando se tienen una gran cantidad de conocimientos ya que se ocupan enormes sistemas gráficos para representarlos y por consiguiente resulta complicado establecer las relaciones que en las redes se pretenden definir.

2.1.5 Plantillas (marcos)

Propuestos por Marvin Minski, los marcos o "frames" consisten en la división de objetos en sus componentes. Los marcos contienen un nombre y un conjunto de "slots" comúnmente llamadas ranuras estas son el lugar utilizado para almacenar las descripciones del objeto, en la mayoría de los casos estos valores o atributos tienen una característica especial: la herencia, la cual a la hora de hacer modificaciones permite cambiar el valor de mayor jerarquía y con ello todos los inferiores adquieren el cambio realizado.

Cuando en las plantillas se describen tipos o cosas individuales, éstas reciben el nombre de ejemplares, cuando en ellas se describe un conjunto de cosas completas, se les conoce como clases; es decir, son plantillas de clase o de ejemplares según sea el caso.

Imaginemos el siguiente conocimiento generalizado acerca de los negros de un planeta llamado "GOM".

1.- Los pilotos y los policías del planeta son negros
2.- Casi todos los negros del planeta son altos
3.- A la mayoría de los negros del planeta les gusta leer
4.- La mayoría de los pilotos del planeta son bajos
5.- A la mayoría de los policías del planeta no les gusta leer

Para asignar ejemplares a sus clases correspondientes o de las que son miembros, se utiliza el descriptor "ES UN", éste es la forma sintetizada de *es un miembro de la clase*. El descriptor "A.K.O." por ser las siglas de la frase *a kind of* es la forma sintetizada de *es un tipo de*, y es utilizado para unir clases entre sí. En el ejemplo siguiente se muestra el uso de éstos dos descriptores (figura 2.3).

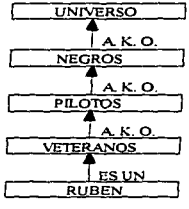


FIGURA 2.3

Es importante observar que en la figura existen jerarquías de las clases. En el marco, la jerarquía que hay es muy simple, en ella el individuo Rubén pertenece a la clase de veteranos; los veteranos a su vez son una subclase de los negros, considerando otra perspectiva se tiene que los pilotos son una superclase inmediata de los veteranos y los negros es una superclase de los veteranos. Para crear plantillas ya sea de clase o de ejemplares y para poder manejarlas más eficientemente, son de mucha utilidad los llamados procedimientos de acceso; los más sobresalientes son:

- Constructor de clases. Crea una plantilla, puede hacer plantillas de clase que contengan más de una superclase inmediata.

- Constructor de ejemplares. Crea una plantilla de ejemplares, la entrada es el nombre de la que forma parte del ejemplar y la salida es el ejemplar que pertenece a esa clase.

- Escritor de descriptores. Coloca valores de descriptor. sus entradas son la plantilla, el nombre del descriptor y el valor que se va a colocar.

- Lector de descriptores. Su fin es rescatar valores de los descriptores, las entradas son la plantilla y el nombre del descriptor, su salida es el valor que pertenece a dicho descriptor.

Resulta importante conocer un poco más a fondo el procedimiento Cuando-Se-Construye (C-S-C). Lo que hace relevante a este procedimiento es la herencia, por medio de la cual se pueden mover los valores de un descriptor de una clase a un ejemplar, sin ninguna dificultad; como lo mencionamos al principio, gracias a la herencia el descriptor que tenga la clase automáticamente lo adquiere el o los ejemplares que pertenecen a esa clase específica, también este procedimiento es provechoso como una opción para compartir el conocimiento. Debemos reconocer que al trabajar con un conocimiento compartido se obtienen ventajas como son: la facilidad de construcción, y la corrección de los errores, la disponibilidad que presta para mantenerlo actualizado, y distribuido, ya que se distribuye automaticamente, otra gran ventaja es la de tener la base para la programación orientada a objetos; debido a que ésta se centra en este tipo de conocimiento.

Consideremos ahora el siguiente ejemplo, teniendo en cuenta el conocimiento ficticio elaborado anteriormente acerca de los negros del planeta "GOM".

Se nos pide colocar el valor en la ranura correspondiente a estatura de un nuevo negro.

para dar un valor a la ranura correspondiente a la estatura cuando se crea un nuevo negro.
* Colocar la palabra **ALTO** en la ranura indicada.

para dar un valor a la ranura correspondiente a la estatura cuando se crea un nuevo piloto.
* Colocar la palabra **BAJO** en la ranura indicada.

Analizando el ejemplo, es claro que para construir un nuevo negro, se dice que de estatura es alto, contradictoriamente al construir un nuevo piloto se dice que es bajo. Si consideramos a Rubén como el nuevo elemento a construir, debemos elegir un calificativo de su estatura, puesto que en los dos procedimientos, encaja nuestro personaje como piloto (bajo) y como negro (alto); para hacer una selección idónea, recurriremos a un método que para este caso resulta muy sencillo. Dicho método consiste en elaborar una Lista Precedente de Clases (L.P.C.), en ésta se enlistan de manera ordenada las clases a las que pertenece el nuevo miembro que se pretende crear, para nuestro caso esta lista quedaría de la siguiente manera:

- Rubén
- Clase veteranos.
- Clase pilotos***
- Clase negros***
- Clase universo.

El símbolo *** indica que en esa clase se guarda el procedimiento, en el primer caso (pilotos) nos indica que Rubén es bajo pero en el otro (negros) indica que es alto, ahora el calificativo que debemos aplicar siempre será el primero en aparecer en la lista. Según esta formación la elección se hace por el procedimiento de la clase de los pilotos, entonces consecuencia tenemos que Rubén resulta ser bajo.

Cabe subrayar que en este caso particular, no hubo mucho problema al elegir el calificativo deseado, merced a que se trató de la jerarquización más simple, pero cuando aparecen más de un descriptor "ES UN" o "A. K. O." se tiene una jerarquización más completa y por ende, también son más difíciles de manipular, aunque de igual manera se hace uso de una L. P. C. ésta es más complicada de elaborar porque se tiene una jerarquización con más ramas y por tanto debemos decidir cómo hacer uso de la jerarquía de clases en la lista, las opciones con las que contamos son los tipos de búsqueda, de los que figuran la búsqueda en profundidad, la búsqueda por anchura, etc., pero también existen algunos tipos de jerarquías en las que es necesario insertar nuevos descriptores "ES UN" y "A. K. O." en las cuales los planteamientos de búsqueda ya no exponen L. P. C's acertadas; y como consecuencia nos lleva a necesitar un mecanismo capaz de producir una L. P. C. en la que cada superclase directa de una clase aparezca en la lista antes de la superclase directa que se encuentre a su lado derecho, para elaborar una lista con éstas características, contamos con un procedimiento bastante interesante de ordenamiento topológico que nos asegura un orden correcto de las superclases directas en la lista.

Demos un ejemplo para demostrar lo que se ha dicho hasta ahora. Sugerimos una nueva creación de negros, referentes al conocimiento imaginario realizado al principio de este apartado.

Para elaborar una L. P. C. que pertenezca a la plantilla mostrada en la figura 2.4 recurriremos a la búsqueda en profundidad de izquierda a derecha con una mejoría, dicha mejoría consiste en garantizar que cada clase se encuentre antes de sus superclases, considerando que esta mejoría resulta eficiente, lograremos unas L. P. C's como las siguientes:

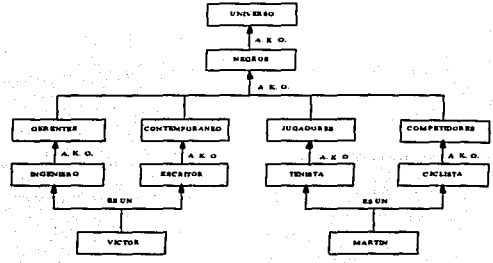


FIGURA 2.4

Víctor	Martín
Clase ingeniero	Clase tenista
Clase gerentes	Clase jugadores
Clase escritor	Clase ciclista
Clase contemporáneo	Clase competidores
Clase negros	Clase negros
Clase universo	Clase universo

Vemos que la lista generada para ambos negros es acertada, pero como ya lo mencionábamos antes, el problema surge cuando se colocan más descriptores nuevos. Supongamos que en este caso sean dos los nuevos descriptores; entonces la plantilla original quedaría como la mostrada en la figura 2.5. Estas nuevas adiciones, provocan alteraciones a la L.P.C.; por lo tanto es aquí donde se hace uso del mencionado ordenamiento topológico el cual para llevarlo a cabo, se sugieren seguir los siguientes pasos:

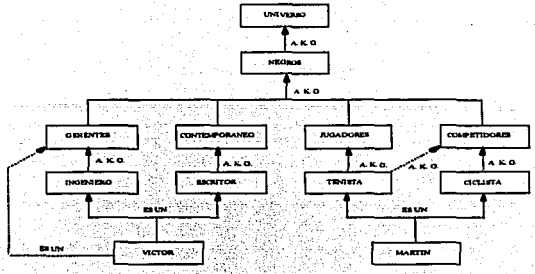


FIGURA 2.5

1) Elaborar una lista completa incluyendo al ejemplar con todas sus clases, observemos que esta lista es diferente a la L. P. C., sin embargo es esencial para formar ésta última.

2) Elaborar una lista de pares para cada elemento que forma la lista hecha en el paso uno, incluyendo al ejemplar.

3) Buscar el elemento "desprotegido" siendo éste aquel que aparece en la parte izquierda de los pares, pero sin encontrarse en la parte derecha.

Aplicando estos incisos al ejemplo mostrado anteriormente tenemos:

1) Para Victor

Victor, ingeniero, gerentes, escritor, contemporáneo, negros, universo.

Para Martín

Martín, tenista, jugadores, ciclista, competidores, negros, universo.

2) En este paso para formar la lista que se nos pide conviene auxiliarnos de gráficas por separado de cada elemento de la lista anterior, la gráfica consiste en el elemento y sus clases o superclase directa, conectados como se muestra en la figura 2.6

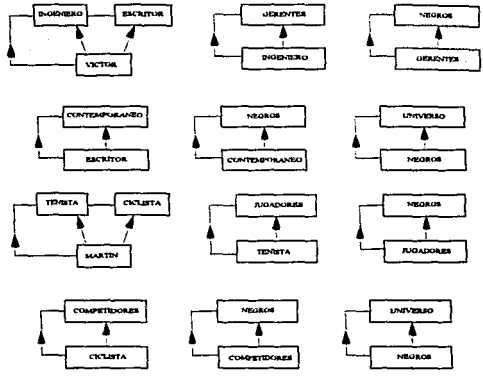


FIGURA 2.6

3) Finalizamos rastreando todos los pares; con el objeto de hallar aquel elemento que se encuentre en la parte izquierda de los pares pero sin ocupar el lado derecho de cualquier par. Siempre que encontremos un elemento con estas características lo sumaremos a la L. P. C., eliminando a su vez los pares en donde se encuentre el mismo elemento. efectuando todas estas acciones queda lo siguiente:

NODO	PARES	LISTA DE PRECEDENCIAS DE CLASE (L.P.C)
Víctor	Víctor-ingeniero, ingeniero-escritor	Víctor
Ingeniero	Ingeniero-gerentes	Ingeniero
Gerentes	Gerentes-negros	Gerentes
Escritor	Escritor-contemporáneo	Escritor
Contemporáneo	Contemporáneo-negros	Contemporáneo
Negros	Negros-universo	Negros
Universo	Universo	Universo

Martín	Martín-tenista, tenista-ciclista	Martín
Tenista	tenista-jugadores	Tenista
Jugadores	jugadores-negros	Jugadores
Ciclista	ciclista-competidores	Ciclista
Competidores	competidores-negros	Competidores
Negros	Negros-universo	Negros
Universo	Universo	Universo

El primer elemento que cumple con los requerimientos anteriores es Víctor, por ello eliminamos esta pareja, colocamos a Víctor en la L. P. C. y continuamos la búsqueda hasta terminar.

Pasando ahora a los procedimientos conocidos como *demonios;* mencionaremos que estos procedimientos son una especie de activadores que están al pendiente o vigilando cuando éstos sean requeridos y activar el proceso especificado previamente en su diseño. Los procedimientos más comunes son: Cuando-Se-Pide (C-S-P), Cuando-Se-Lee (C-S-L), Cuando-Se-Escribe (C-S-E), Con-Respecto-A (C-R-A) y Cuando-Se-Aplica (C-S-A).

A los primeros C-S-P como su nombre lo indica, se les puede pedir valores de cualquier ranura de la plantilla, con esto emplean los valores de los descriptores dados previamente al crearse, de igual forma que pueden reemplazar estos mismos valores; por lo que el nuevo valor sería virtual.

Los procedimientos C-S-L y C-S-E, únicamente pueden ser activados en los procesos de lectura o bien de escritura, comúnmente éstos dos procedimientos son utilizados para garantizar que al cambiar un valor de un descriptor, se produzca otro cambio necesario automáticamente en otro valor que así lo requiera, por ejemplo: al escribir un valor en el descriptor pasatiempo de un escritor, escribiremos *lectura*, con este nuevo valor y con ayuda de estos dos procedimientos queda garantizado que el valor en el descritor vocabulario de un escritor quede *amplio*.

Los procedimientos C-R-A tratan casos específicos, es decir; condicionados a ciertas circunstancias particulares. Por ejemplo, supongamos que el rendimiento de un investigador cuando le piden hablar acerca de la política interna de un país es bajo, no así cuando escribe o comenta temas referentes con la tecnología.

Por último los procedimientos C-S-A son útiles como auxiliares para realizar la ejecución de una acción sobre un objeto determinado, es común usarlo en la programación a objetos y pueden ser compartidos automáticamente entre subclases con lo que evita manejar procedimientos por separado para cada clase.

2.2 BUSQUEDA

Algunas veces se considera la resolución de diferentes o variados problemas, como una medida aceptable, para determinar la inteligencia tanto de las personas como de las llamadas "máquinas inteligentes".

Dada la importancia de los problemas, éstos se han integrado en dos grandes grupos: en el primero de ellos se encuentran aquellos que se pueden resolver a través de métodos numéricos, llamados comúnmente de computación. En el segundo están los problemas que no se resuelven por medio de un algoritmo numérico. La I.A. se ocupa especialmente de los últimos.

El método utilizado por la I. A. para la resolución de problemas es la búsqueda. La búsqueda consiste en tratar de hallar una solución optima para un problema, al recorrer todo un universo de posibles soluciones conocido como "espacio de búsqueda".

Para una mejor visualización de este método, se han desarrollado esquemas gráficos con forma de árbol, representativos de los espacios de búsqueda, que se caracterizan por contar con un estado inicial, un espacio de búsqueda y un estado final ó estado objetivo, y los cuales se dividen principalmente en árboles de búsqueda y árboles de decisión.

2.2.1 Arboles de búsqueda

Cuentan con nodos que representan diferentes hechos de una base de conocimientos[*] la conexión que muestra las relaciones entre un nodo y otro se denomina arco o rama. Existe un nodo llamado raíz ya mencionado anteriormente como estado inicial. Para pasar al siguiente nodo descendiente ó hijo , se baja por las ramas, y así consecutivamente hasta llegar al nodo terminal (nodo objetivo); para diferenciarlo de los demás , se tiene en cuenta que los nodos terminales no tienen descendientes. Es importante identificar los nodos (ver fig. 2.7) para una mejor visualización de los mismos.

[*]Base de conocimientos, denominada también espacio de búsqueda, se abordara en el capitulo 3.

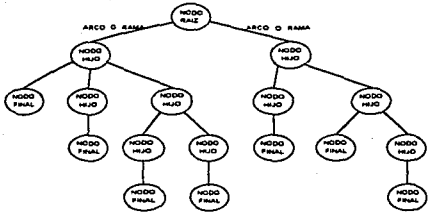


FIGURA 2.7

Generalmente no se esquematiza toda la base de conocimientos como un árbol de este tipo, porque sería demasiado extensa; esta representación se utiliza más bien de manera didáctica con el fin de comprender la forma en que una máquina de inferencia recorre la base de conocimientos utilizada.

2.2.2 Arboles de decisión

Este tipo de árbol al igual que el anterior, cuenta con nodos y ramas. La diferencia radica en los nodos descendientes que representan puntos de elección considerados como preguntas. Regularmente los nodos descendientes cuentan con dos nodos hijos y las ramas que los conectan toman los valores de verdadero ó falso (ver fig. 2.8).

Por su simplicidad se podrían considerar a este tipo de árboles como poco significativos, sin embargo es esta estructura lo que los hace ajustables a diferentes problemas y aplicables en gran medida a la búsqueda de soluciones.

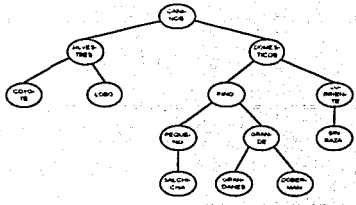


FIGURA 2.8

Para recorrer cualquiera de los tipos anteriores de árboles se han creado dos diferentes mecanismos que son: la búsqueda ordenada y la búsqueda no ordenada que a su vez se dividen en estrategias bien definidas, la fig. 2.9 muestra un esquema generalizado a este respecto.

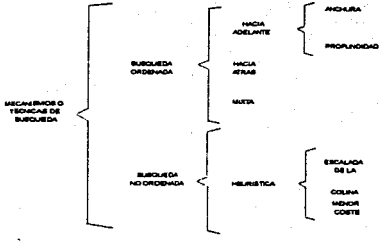


FIGURA 2.9

La estrategia utilizada depende del área de aplicación en la que se va a adoptar y, como es abierta al programador, no se puede afirmar que la elegida sea la mejor o la peor.

A continuación se describen de modo general los conceptos señalados en la figura 2.9.

2.2.3 Búsqueda ordenada

La búsqueda ordenada se refiere a la manera en que se va a recorrer el espacio de búsqueda que puede ser a través del encadenamiento hacia adelante, hacia atrás o bien el mixto.

En este mecanismo se enlazan (o más propiamente se encadenan) los nodos (conocimientos), que están representados a través de reglas de tal manera que la parte consecuente de una sea el antecedente de la otra y así sucesivamente, hasta encontrar la solución deseada o en su defecto se termine de recorrer el espacio de búsqueda.

2.2.4 Encadenamiento hacia adelante

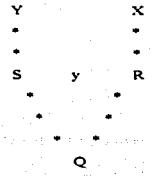
Se dice que se hace un encadenamiento hacia adelante cuando la búsqueda empieza en el estado inicial y termina en el estado objetivo. También se le ha dado el nombre de "encadenamiento guiado por los datos", ya que a partir de unos datos iniciales se van enlazando los conocimientos hasta llegar a la solución deseada.

Su funcionamiento se basa en la teoría del Modus Ponens, que señala lo siguiente:

SI X ENTONCES Y

si X es cierto, entonces Y también lo es.
Ejemplo:
R1 = Q si R y S
R2 = R si X ó T
R3 = S si Y ó U
Datos de entrada X, Y.
Considerando a X y Y como datos de entrada se tiene que:
Aplicando R2 se obtiene R porque si X ó T entonces R
Ahora se tiene X, Y y R
Aplicando R3 se obtiene S porque si Y ó U entonces S
Ahora se tiene X, Y, R y S
Aplicando R1 se obtiene Q porque si R y S entonces Q.
Al obtener R y S se da como cierta Q considerada también estado objetivo.

El árbol quedaría como a continuación se muestra:



La característica de ir generando nuevos hechos implica desarrollar diferentes métodos para su tratamiento, que son los llamados por anchura y por profundidad.

2.2.5 Búsqueda por anchura

Para definir esta búsqueda es necesario comprender lo que es un nivel; el nivel es la posición descendiente donde se encuentra cada nodo, es decir, el nodo raíz se encuentra en el nivel 0, sus descendientes directos en el nivel 1 y los descendientes directos de estos, en el siguiente nivel y así sucesivamente, un ejemplo de lo anterior se encuentra en la figura 2.10.

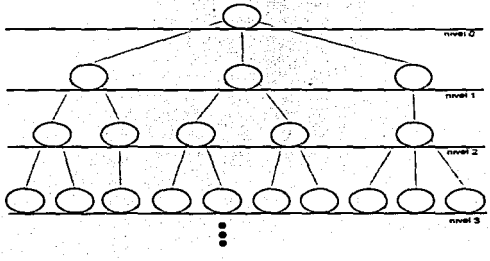


FIGURA 2.10

La búsqueda de anchura, comienza en el nodo raíz y sigue su análisis en el primer nodo descendiente de izquierda a derecha avanzando por todo el nivel, cuando ya se han recorrido todos los nodos de un nivel, se continua con el siguiente utilizando el mismo procedimiento (ver fig. 2.11)

Esta búsqueda termina cuando se encuentra un nodo considerado como objetivo, "...es decir, que se contemplan todas las posibilidades de un nudo, antes de pasar al siguiente modo o bifurcación del árbol"[4].

Cuando el árbol cuenta con pocos niveles, la búsqueda por anchura es la más indicada, sin embargo, si se tienen bastantes niveles, la búsqueda en profundidad es una mejor elección.

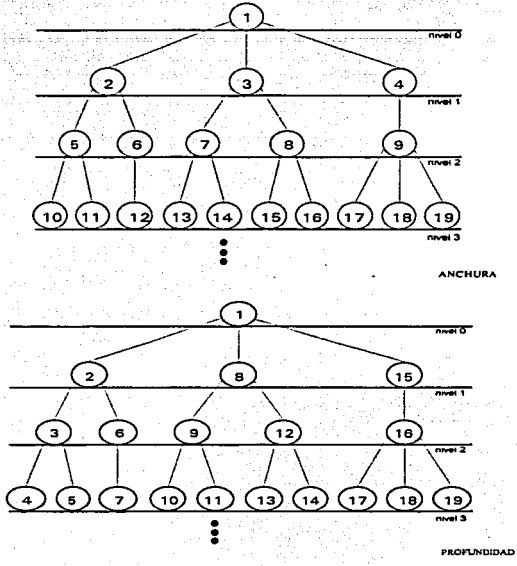


FIGURA 2.11:

[4] J. P. Sánchez y Beltrán, op. cit., p. 43

2.2.6 Búsqueda en profundidad

La búsqueda en profundidad empieza en el nodo raíz y baja a el nodo que se encuentra más a la izquierda, si este tiene descendientes, nuevamente vuelve a bajar a el nodo más a la izquierda pero del siguiente nivel, hasta que se encuentra el nodo objetivo, o en su defecto, se llega al final de la ramificación, es decir, a un nodo terminal, si esto sucede, se retrocede hasta un nivel en el que se pueda continuar la búsqueda por otro camino (ver fig. 2.11); " ...no se toma otra posibilidad en un nudo." (nodo) " hasta que no se ha desarrollado completamente una rama."[5]

Es recomendable utilizar el encadenamiento hacia adelante cuando existen más nodos terminales y menos nodos iniciales; en caso contrario, es mejor emplear el encadenamiento hacia atrás.

2.2.7 Encadenamiento hacia atrás

Este toma como referencia al nodo objetivo (solución) y va retrocediendo en el árbol hasta llegar a el nodo raíz. Este método es más que nada de comprobación ya que verifica la "no falsedad" de un hecho o solución, por ende, se debe conocer la solución e introducirla como dato de entrada y los demás datos pueden introducirse cuando el programa lo requiera, no así en el encadenamiento hacia adelante donde estos parámetros se introducen al principio solamente.

Su fundamento se basa en el Modus Tollens que señala lo siguiente:

SI X ENTONCES Y

y si X es falso, entonces Y también lo es.

Ejemplo:

R1 = Q si R y S

R2 = R si X ó T

R3 = S si Y ó U

Teniendo X, U demostrar que es cierto Q

Aplicando la R1,

Q es cierta si R y S

Aplicando la R2,

R es cierta si existe X ó T

y como existe X, R es cierta

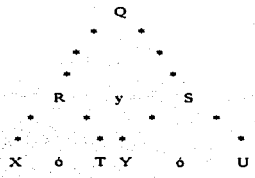
Aplicando la R3,

S es cierta si existe Y ó U

y como existe U, S es cierta

[5] Ibídem., p. 43

El árbol resultante es:



Existe otro tipo de encadenamiento que es una combinación de los dos anteriores, en donde se maneja tanto la información que se conoce, como el objetivo a probar. En realidad la mayoría de los programas de I.A. utilizan tal encadenamiento llamado mixto.

2.2.8 Encadenamiento mixto

Esta búsqueda es de tipo bidireccional ya que utiliza el encadenamiento hacia adelante partiendo del nodo inicial para encontrar nuevos datos (posibles solución), y después emplea el encadenamiento hacia atrás partiendo del estado final para verificar la solución que se tenga, o los datos obtenidos de la primera búsqueda.

Se da por terminada cuando ambas búsquedas al utilizarlas simultáneamente se encuentran en un nodo común, o si se utilizan separadamente, cuando estas lleguen a su fin.

Al utilizar este método, se puede caer fácilmente en lazos infinitos, que se deben evitar utilizando detectores y abridores de los mismos.

Ejemplo:

R1 = Gato si maúlla y es pequeño
R2 = Gato si es pequeño y cae en cuatro patas
R3 = Si Maúlla es felino
R4 = Es pequeño si es felino y mide menos de 40 cm. parado

Los datos de partida serían MAULLA y MIDE MENOS DE 40 CM. PARADO y se desea conocer si la solución GATO que se tiene es cierta o falsa. Lo más conveniente es utilizar primero el encadenamiento hacia adelante, porque si utilizamos el otro, podría resultar lo siguiente:

Al aplicar R1, MAULLA es dato de entrada y ES PEQUEÑO se va a la R4

Al aplicar R4, MIDE MENOS DE 40 CM. PARADO es dato pero FELINO no, por lo tanto no se cumple.

Al tratar de aplicar R2 se encuentra nuevamente ES PEQUEÑO y como ya se vio, faltan datos para su comprobación.

Lo mismo pasa al aplicar R3 ya que se desconoce si es FELINO.

La búsqueda termina sin ninguna solución en concreto, ahora veamos el mismo ejemplo pero utilizando el encadenamiento hacia adelante.

Como tenemos a MAULLA como dato aplicamos R3 y deducimos a ES FELINO

Al aplicar la R4, como ya sabemos que es FELINO y sabíamos que MIDE MENOS DE 40 CM. PARADO se da por cierto que es PEQUEÑO.

Si sabemos que ES PEQUEÑO y MAULLA podemos utilizar a R1 para determinar que es gato, por lo que:

GATO SE COMPRUEBA COMO CIERTA.

Es aquí donde se puede utilizar el encadenamiento hacia atrás ya que se dedujo un nuevo dato (FELINO) que se puede utilizar.

2.2.9 Búsqueda no ordenada

En las diferentes técnicas enlistadas en la búsqueda ordenada, se recorre el espacio de búsqueda en su totalidad, sin considerar el contenido de los nodos; yendo de un elemento a otro para encontrar la solución, como estos tipos de búsqueda no implican una "suposición inteligente", se dice que son técnicas de búsqueda a ciegas; las cuales bastante eficaces cuando el espacio de búsqueda no es muy amplio, ya que garantizan el encontrar la solución, la desventaja es que son muy lentas y existen casos en los que el espacio de búsqueda es inmensamente mayor y el utilizar cualquiera de las técnicas antes explicadas implicaría un tiempo excesivamente largo; inclusive de horas.

Pensemos en un árbol que se va extendiendo, a medida que aumentan los descendientes, aumentan los descendientes de estos mismos y así sucesivamente; y cada nuevo hijo incrementa un camino más al espacio de búsqueda, en otras palabras, va creciendo exponencialmente.

Para formarnos una idea más clara de la explosión combinatoria pensemos en una persona jugando el clásico juego de "¿En donde quedo la bolita?", en su mesa de juego tiene las tres cartas con las que cubre dicha bolita, esta persona podría tener tres posibles alternativas para mover las cartas lo que equivaldría a poder combinar estas alterativas a través de un teorema que señala la manera de saber el número total de posibles combinaciones que se podrían hacer y cuya formula es N! (N-factorial), para nuestro caso seria 3!=3x2x1=6 posibles caminos de solución, pero qué pasa si ahora esta persona decide jugar con cuatro cartas, tendría entonces cuatro posibles alternativas para mover la bolita, utilizando la fórmula tendriamos 4!=4x3x2x1=24 y si fueran 5 cartas serían 5!=5x4x3x2x1=129 diferentes alternativas, esto evidencia que un nuevo nodo incrementa exponencialmente el número de caminos a recorrer.

A esta expansión exponencial del campo de posibles soluciones se le ha nombrado "explosión combinatoria"[6] .

Considerando este hecho es fácil imaginar la importancia de crear mecanismos que permitan minimizar el espacio de búsqueda pero sin eliminar el resultado al que se pretende llegar. La búsqueda no ordenada de soluciones parece ser la respuesta a este problema.

2.2.10 Búsqueda heurística

Para comprender mejor lo que es una búsqueda Heuristica, primero definiremos a un heurístico como la aproximación que se hace para llegar a una solución, haciendo uso no solo de los datos con los que cuenta, sino también de la experiencia para indicar el camino a seguir en la solución del problema.

La búsqueda heurística sirve para reducir el espacio de búsqueda a través de reglas que nos permiten minimizar el número de caminos de un árbol al escoger sólo aquellos en donde sea más probable que se encuentre la solución; para realizar esta selección se basa en el metaconocimiento (es un conocimiento que sirve como auxiliar en la búsqueda de soluciones y permite la selección de una serie de conocimientos específicos), o en minimizar o exagerar algunos aspectos del problema para poder hacer una mejor selección del siguiente nodo a avanzar.

La información heurística que se tiene podría ser imprecisa y no garantizada pero gracias a ella se puede encontrar una buena solución en un tiempo mucho más corto que si se utilizara solo una búsqueda a ciegas. Las técnicas heurísticas mejoran el proceso de búsqueda al sacrificar la precisión a cambio de la rapidez, no obstante, aunque no garantice encontrar la mejor solución, si asegura dar una que cumpla con los lineamientos especificados para resolver el problema que se esté tratando. Como lo afirma Rich "...algunas técnicas heuristicas ayudan a guiar un proceso de búsqueda sin sacrificar ninguna aspiración de completitud que el

[6] Ibidem., p. 92

proceso pueda haber tenido previamente. Otras (de hecho, muchas de las mejores) pueden llevar a que pase inadvertido un camino excelente. Pero, en promedio, mejoran la calidad de los caminos que se exploran"[7] .

Existen dentro de la búsqueda heurística métodos diferentes; aquí se verá solo la búsqueda de escalada de la colina y menor coste que son los más representativos de la heurística.

2.2.11 Búsqueda de la escalada de la colina

Esta búsqueda tiene como base la búsqueda en profundidad. Lo que hace este algoritmo es buscar avanzar lo más posible (ya que cuanto más avance más cerca estará de su objetivo), para llegar al nodo que se encuentra cercano del nodo terminal, reduciendo así el número de nodos a analizar. Su nombre se deriva de "..la analogía de un excursionista perdido en la obscuridad, en mitad de una montaña; entonces, incluso en la oscuridad, el excursionista sabe que cualquier paso hacia arriba le sitúa en la dirección correcta."[8] Sin embargo, esto tiene sus inconvenientes ya que como existen las llamadas "mesetas", en las cuales todos los nodos visitados parecen tener la misma importancia , la búsqueda sigue como si fuera solo de profundidad y otro problema son las "crestas" que al no encontrar la solución a la primera se pasará por una cresta varias veces puesto que esta técnica utiliza también el retrosegimiento.

2.2.12 Búsqueda del menor coste

Esta a diferencia de la búsqueda de escalada de la colina, no pretende avanzar sino pretende llegar a la meta haciendo el menor esfuerzo aunque para ésto tenga que pasar por un mayor número de nodos. Es decir busca la manera más fácil para llegar a su objetivo; esta técnica también se fundamenta en la búsqueda en profundidad.

Las técnicas heurísticas aunque cuentan con un adelanto notable respecto a las búsquedas ciegas no siempre son aplicables, ya que la información almacenada en la base de conocimientos a veces no es suficiente como para que el algoritmo heurístico pueda tomar una decisión, ésta es la razón de la existencia de las búsquedas ciegas utilizables dependiendo del problema y de la información que se tenga del mismo; no olvidemos que la velocidad de resolución depende igualmente del problema, de la longitud del camino y del número de nodos que se tengan que recorrer.

De la elección del método o mecanismo a utilizar depende qué tan rápido se encuentre la solución y qué tan buena sea la misma. No es lo mismo una "solución optima" que una "buena solución". La solución optima es la mejor que existe considerando todos los conocimientos de los que se disponga, si se requiere de esto se debe utilizar la búsqueda a ciegas porque esta recorre toda la base de conocimientos; si por el contrario se

[7] E. Rich, op. cit., p. 43
[8] Herbert Schildt, op. cit., p. 35

pretende sólo alcanzar una buena solución esto significa hallar una que este dentro de los lineamientos requeridos sin importar si existe o no una mejor, para este tipo de casos las búsquedas heurísticas son una mejor elección.

CAPITULO III

SISTEMAS EXPERTOS

La creación de un capítulo en particular para esta rama de la I.A. no significa que las demás ramas carezcan de importancia e incluso no quiere decir que sean menos en ningún sentido, simplemente se ha creado este capítulo para los sistemas expertos porque éstos son el primer producto comercial que ha lanzado la I.A. y porque los logros alcanzados por este tipo de sistemas son sobresalientes por sí mismos.

3.1 ANTECEDENTES HISTORICOS DE LOS S. E.

A lo largo de toda su historia el hombre se ha enfrentado con necesidades de diferente índole, por lo que ha logrado desarrollar diferentes técnicas y tecnologías como medio para satisfacerlas o reducirlas al mínimo y la creación de los S. E. no son la excepción.

Los S.E., concepto introducido por Feigenbaum apenas en el año de 1977, han sido el resultado de un desarrollo surgido desde la década de los 50's cuando diversos investigadores comenzaron a ocuparse de los métodos generales de resolución de problemas; y poco después por trasladar estos métodos a los ordenadores, mismos que estaban desarrollando algunos investigadores ingleses y norteamericanos pero no en forma conjunta, estos científicos se ocupaban de desarrollar máquinas capaces de realizar operaciones numéricas complejas y fue solo un grupo reducido de ellos quienes se dedicaron al desarrollo de este tipo de máquinas, pero enfocadas al procesamiento de información simbólica, con lo que pretendían trabajar en procedimientos, conocimientos e ideas que les permitieran emular cierto nivel de razonamiento y conocimiento humano.

Aunque el software y el hardware no eran los adecuados en aquel entonces para trabajar bajo estos términos, se lograron avances significativos como la creación de los lenguajes de programación LISP y el PROLOG, que son los lenguajes creados con el fin de satisfacer las necesidades de la I. A., ya entre 1975 y 1980 gracias a la miniaturización surgen máquinas más potentes que permiten un avance aun mayor, aparecen las primeras computadoras de procesamiento simbólico en forma paralela, pero también surgen problemas como la representación tanto de los conocimientos en la memoria del ordenador, como de las relaciones entre estos, la explosión combinatoria y las grandes soluciones como los mecanismos de poda.

Cabe señalar que, la historia de los S.E. empieza a desarrollarse desde mucho tiempo atrás; sin embargo, es a partir de los 70's cuando toman un verdadero auge al darse cuenta los científicos de la dificultad de expresar los mecanismos de resolución de la mente humana partiendo desde el punto más elemental que serían sus células (a lo que se enfocaban más las redes neuronales) y al no tener avances significativos desde esta perspectiva se trata de retomar el problema pero pensando en simular nuevos mecanismos para campos muy concentrados del conocimiento e imitando la forma externa o comportamiento aparente del experto humano.

Para finalizar con esta breve historia de los S.E. mencionaremos las cuatro etapas que coinciden en señalar la mayoría de los autores.

1.- "*Etapa de inicio*" entre 1965 y 1970. Aunque algunos autores prefieran no indicar fechas de inicio otros señalan a 1965 como el año de partida de los S.E. Esta etapa se caracteriza porque en ella se fundan las bases teóricas para el desarrollo de los S.E., así mismo aparecen los primeros sistemas considerados precursores como lo son el DENDRAL (desarrollado por Edward Feigenbaum para el estudio de estructuras químicas) y el MACSYMA (desarrollado en el MIT, resuelve problemas de cálculo diferencial e integral).

2.- "*Etapa de prototipos*" entre 1970 y 1977. En ésta existe un mayor desarrollo de los lenguajes de programación dirigidos a los S.E. aparece el PROLOG (1972), también se retoma el LISP que ya hace casi una década que fué creado. Los primeros ordenadores que utilizan representación simbólica surgen apoyados en el proyecto DARPA de E.U. (1973) y se dan los S.E. más conocidos por sus logros como: el INTERNIST (para diagnóstico de medicina interna), el MYCIN (para diagnóstico y tratamiento de enfermedades infecciosas), el XCON (para configuración de computadoras) y el PROSPECTOR entre otros.

3.- "*Etapa de experimentación*" fluctúa entre 1977 y 1981; aquí se retoman sistemas como el MYCIN (considerado el primero de los S.E. que explica sus razonamientos y concluye de manera semejante a como lo haría un experto humano), que manejan valores de incertidumbre para su razonamiento entre 0 y 1. Aparecen en él dos bloques bien definidos: una base de conocimientos y un mecanismo de inferencia que permite una manipulación separada del motor de inferencia y la base de conocimientos. El uso real de esta maniobra es la capacidad de dejar la base de conocimientos libre por lo que son adaptables a diferentes áreas; lo único que hay que hacer es almacenar el conocimiento apropiado para nuestra aplicación. A este tipo de sistemas que cuentan con una estructura igual a la de un S.E. se le llama sistemas vacíos o concha (Shell) que más propiamente dicho es un elemento de apoyo para la creación de diferentes S.E. Dicho lo anterior podemos apuntar que esta etapa se caracteriza por la aparición de herramientas de apoyo para la creación de los S.E.

4.- "*La etapa de industrialización*" de 1981 en adelante. En esta última, diversas empresas se interesan no solo por adquirir estos nuevos sistemas sino por desarrollarlos, también muchos laboratorios vinculan sus investigaciones a este campo en particular. En octubre de 1981 se da a conocer el trabajo japonés denominado proyecto de la quinta generación, en el cual se pretende desarrollar los nuevos ordenadores simbólicos que adoptan como lenguaje fundamental el PROLOG.

3.2 DEFINICION

Para que un sistema sea considerado como experto, necesita contar con las siguientes características:

1.- Amplitud: Es decir, que tenga un extenso dominio del conocimiento, significa entonces que el S. E. debe tener la capacidad para responder no sólo a grandes preguntas sino también a las más elementales por sencillas que parezcan.

2.- Capacidad de aprendizaje: Es importante que un sistema responda a preguntas pero igualmente lo es que pueda elaborarlas y estructurarlas de tal manera que le permitan adquirir información adicional del usuario para incorporar nuevos conocimientos a su base.

3.- Claridad: Este punto se refiere a que la intercomunicación del sistema con el usuario debe ser clara y entendible al utilizar el lenguaje natural de las personas.

4.- Explicativo: Debe tener la capacidad suficiente para llegar a conclusiones y poder señalar a la vez qué tipo de razonamiento o pasos lo llevaron a dicha solución. Esta característica le brinda fiabilidad al sistema y como lo señala Sánchez y Beltrán: " La fiabilidad que le damos a un experto humano es función del grado de explicación que nos da tras haber resuelto un problema..."[1]

5.- Facilidad para su uso: Aún por personas ajenas a la rama computacional.

6.- Maleable: La flexibilidad del conocimiento debe permitir añadir, modificar o restringir alguna información dependiendo de los requerimientos. Así el sistema puede ser adaptable a las necesidades cambiantes, lo que da la pauta para no volverse obsoleto.

7.- Provechoso: Que su aplicación sirva para cubrir una necesidad concreta.

A todas estas características se les puede adicionar otra, la que le permite al usuario aprender del S.E., éste es un módulo adicional considerado por algunos especialistas en la materia como requisito imprescindible de un buen S.E.

Los S. E. son la primera aplicación operacional de la I. A., situándose entre la representación del conocimiento y la demostración, pero ¿qué tan difícil es representar el conocimiento?, démonos cuenta que, para el mismo ser humano la adquisición de conocimiento se basa en la práctica y la experiencia, siendo muy difícil

[1] J. P. Sánchez y Beltrán, op. cit., p. 20

de heredar; además se limita por el tiempo para adquirirla (cuando un experto muere se pierden la gran mayoría de sus conocimientos) esto no solamente es propio del humano ya que el S.E. se debe limitar también al tamaño de la memoria y al tiempo requerido de procesamiento. Por otro lado, existen varias características que se deben cuidar en los S.E.; tales como la programación, ya que si es defectuosa, aún teniendo los conocimientos necesarios, el sistema no sería capaz de resolver un problema que se le plantee, del mismo modo es importante que cuente con la capacidad de deducir datos nuevos tomando en cuenta los existentes en su base. Cuando un sistema ha superado estos puntos, podrá llegar a una solución optima, pero qué sucede cuando un conocimiento no es completo. En un programa convencional se necesitan todos los datos para resolver un problema, a diferencia de los S. E. que más de una vez trabajan con datos no solo incompletos sino hasta dudosos por ser de resolución no tanto numérica sino analítica, para solventar este problema hace uso de los heurísticos que serían lo equivalente a los algoritmos, otra diferencia entre estos dos tipos de programación es que en el tipo numérico sólo hay una solución mientras que en los analíticos se deben contemplar simultáneamente varias alternativas o hipótesis.

No olvidemos la gran contribución de los ingenieros del conocimiento que se dedican a transferir los conocimientos del experto a un sistema, en forma ordenada y clara, cuando un ingeniero del conocimiento interactúa de manera efectiva con el experto humano, hace que este último reflexione sobre la calidad y coherencia de sus conocimientos.

Conociendo las características deseables en un S. E. se puede decir que es un programa de computadora que simula los procesos intelectuales de un experto humano al contar con conocimientos en un campo de especialidad determinado y procedimientos que le permiten resolver o dar diferentes alternativas de solución a problemas que anteriormente solo se resolvían con la ayuda del experto humano.

A continuación se enlistan algunas razones para utilizar un S. E.

a) Para apoyar a personas no especializadas en el campo que se este utilizando con el fin de que existan más expertos logrando así su expansión gradualmente a todos los niveles.

b) Para que más personas tengan acceso al conocimiento (al utilizarlos como consultores).

c) Para mejorar la calidad del conocimiento del experto humano elevando la probabilidad de tener una optima solución.

d) Para que el conocimiento perdure aun cuando el experto humano muera, ó sufra de daños irreversibles que afecten sus conocimientos.

e) Para reducir los costos de acceso al conocimiento.

f) Para obtener soluciones rápidas y fiables.

Es importante que exista una conciencia de las limitaciones, tanto del experto humano como de un sistema; además hay que reconocer que éstos últimos, aun no pueden adquirir el conocimientos por sí mismos a través de práctica, por lo que en realidad no son expertos sino sistemas basados en el conocimiento.

3.3 COMPONENTES DE LOS S. E.

Todo S.E. se divide en cinco bloques bien definidos los cuales son: el motor de inferencia (M.I.), la base de conocimientos (B.C.), la base de hechos (B. H.), la interfaz con el experto (I.E.) y la interfaz con el usuario (I.U), todos ellos actúan independientemente el uno del otro, lo que facilita modificar la estructura de cualquier elemento sin afectar a los demás módulos. Podemos comprender mejor esta característica al conocer más de cerca las funciones de cada módulo en particular.

3.3.1 Motor de inferencia (M.I)

El M.I. es el controlador de todo el sistema de razonamiento por lo que sus funciones son amplias, primeramente pide al usuario la información de entrada respecto al problema que se va a tratar, después considerando estos datos y haciendo uso de los heurísticos recorre la base de conocimientos e implementa las reglas que coincidan con los datos de entrada utilizando algún método de búsqueda (como los planteados en el capitulo 2) hasta encontrar una solución; el determinar si ha encontrado o no una solución es también una de sus funciones.

3.3.2 Base de conocimientos (B.C.)

Un dato (utilizado en la B.C) es la información que puede variar de una a otra resolución.

La B.C. es una base de datos que cuenta con conocimiento declarativo (información) y conocimiento procedimental (reglas específicas o procedimientos de acción) que pone a disposición del M.I. para su utilización.

La capacidad de la B.C. es de suma importancia puesto que ésta le permite tener un campo de acción más amplio o completo del S.E., la información contenida en la B.C. debe ser sencilla, independiente, fácil de modificar, transparente (para facilitar su explicación) y relacional (para ser utilizada por las diferentes reglas).

3.3.3 Base de hechos (B.H.)

Un hecho es la información que no varía de una a otra resolución, la B.H. también llamada memoria de trabajo es de tipo transitorio y guarda los datos proporcionados por el usuario y los que ha arrojado el sistema después de la aplicación de las reglas a los mismos, sirve también como un tablero de consulta para el M.I. y señala el estado del sistema en cada instante; es decir, cuenta con un registro del estado.

3.3.4 Interfaz con el experto (I.E.)

La I.E. tiene por objetivo adicionar el conocimiento del experto humano a la B.C. para ello se apoya en el ingeniero del conocimiento que estructura dicha información de una manera tal que sea inteligible por la B.C. La función del ingeniero del conocimiento no es tanto la de programar, sino la de estructurar el conocimiento lo más compacto y comprensible posible al utilizar cualquier técnica de representación del conocimiento, como las introducidas en el capítulo 2.

Pero ahí no termina la función de la I.E. ya que debe poseer también la capacidad de poder implementar otros datos los cuales se deben validar y depurar esto se hace para evitar repeticiones (debe revisar que los datos sean nuevos), inconsistencias (lo más completos posibles), errores (que no sean incoherentes con datos ya almacenados anteriormente), etc.

3.3.5 Interfaz con el usuario (I.U.)

Esta es de tipo bidireccional de la forma S.E. ↔ usuario, este último puede buscar la comunicación con el experto para hacer uso de diferentes elementos como iconos, gráficas, ventanas y cualquier otro para evitar que esta comunicación sea tediosa o lenta.

La función de este módulo debe comenzar con una explicación rápida de como utilizar el sistema, después el usuario debe aclararle a la máquina el problema que desea se resuelva e introducir los datos con los que cuenta, es en ese momento cuando se nota la importancia de implementar un lenguaje natural y sencillo en el S.E. con el fin de establecer comunicación con cualquier usuario y para la óptima comprensión de los datos por el propio sistema.

Si al ir analizando el problema el sistema necesita más información, éste toma parte de la regla que se analiza y la convierte en pregunta para el usuario, además de tener capacidad de pedir información también debe hacer recomendaciones y tanto explicar como justificar su razonamiento al dar una solución. El S.E. no solo proporciona conclusiones, también da estimaciones y aproximaciones que igualmente se justifican y explican.

3.4 EJEMPLOS DE S.E.

A continuación enlistaremos algunos S. E. que ocupan ya un lugar importante en las diferentes áreas de aplicación como en: la instrucción o enseñanza, la interpretación, la planificación, la supervisión, la depuración, la reparación, el control, el diseño y el diagnóstico entre otras.

En aritmética:
MACSYMA: Experto en el cálculo diferencial e integral mediante la manipulación simbólica de expresiones algebraicas.

En aviación:
AIRPLAN: Ayuda al lanzamiento y recuperación de aviones desde un portaaviones.

En diseño:
R1=XCON: Para configuración de miniordenadores.

En electrónica:
SOPHIE: Simula un laboratorio de electrónica.

En la enseñanza:
BUGGY: Ayuda en el estudio de la aritmética.
GUIDON: Para la enseñanza de los estudiantes de medicina, relacionados con el diagnóstico de las diferentes patologías.
PROGRAMMERR'S APPRENTICE: Ayuda a la escritura de programas.
SCHOLAR: Enseña la geografía de América del sur.

En geología:
HYDRO: Para estimular los parámetros de comportamiento de cuencas hidrográficas a partir de sus características goelógicas y morfológicas.
PROSPECTOR: Estudio de yacimientos minerales en la explotación geológica.

En mecánica:
MECHO: Resuelve problemas de mecánica.

En medicina:
AI/RHEUM: Auxiliar en la rama reumatológica.

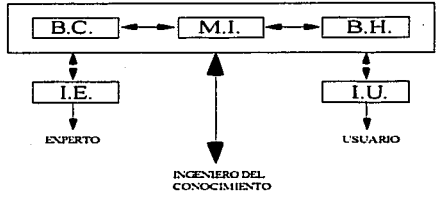
ANTICIPATOR: Prescripciones de antibióticos.
CADIAG: Ayuda en medicina interna.
CASNET: Empleado en oftalmología.
CENTAUR: Se utiliza en las enfermedades del pulmón (neumología).
CLOT: Problemas de coagulación.
GENESIS: Permite planificar y simular experimentos en el campo de la unión de genes.
HEAMED: Utilizado en psicofarmacología.
INTERNIST: Medicina interna.
IRIS: Auxiliar en oftalmología.
LOCALIZE: Creado para la neurología.
MEDIKS: se usa en medicina general.
MOLGEN: Trabaja en el campo de la genética molecular e ingeniería genética apoyándose en la síntesis y análisis del DNA.
MYCIN: Diagnostica enfermedades infecciosas en la sangre y meningitis.
NEPHROS: Utilizado en la rama renal.
NEUROLOGIST: Empleado en la neurología.
ONCOCIN: Protocolos de cáncer.
PUFF: Auxiliar en enfermedades pulmonares.
RADEX: Usado para la radiología.
RX: Creado para la reumatología.
SAM: Auxiliar en la neurología.
SEEK: Para ayudar en el estudio de la reumatología.
SPE: Diagnostica distintos tipos de inflamaciones en un paciente (electroforesis).
TOUBIB: Se emplea en medicina general.
VM: Monitorización respiratoria en cuidados intensivos.

En química:
DENDRAL: Identifica la estructura molecular de las sustancias desconocidas.

En telefonía:
ACE: Identifica problemas en redes telefónicas.

3.5 INTERACCION ENTRE MODULOS

Como ha indicado, los S.E. se dividen en cinco módulos que si bien interactuan entre si, son independientes los unos de los otros, esta modularidad es la verdadera fuerza de los S. E. La interacción entre los diferentes módulos está bien definida y se muestra en la siguiente ilustración:



ESQUEMA DE BLOQUES DE UN S. E.[2]

El modulo M.I. lo que hace es tomar la información suministrada por el usuario (I.U.) y la convierte en hechos almacenados en la B.H., o memoria de trabajo éstos a su vez son validados por las reglas contenidas en la B.H. (a través del M.I.) y así sucesivamente hasta encontrar la solución deseada. La secuencia en la que se examinan las reglas también la decide el M.I.

Si los hechos existentes no son suficientes, el M.I. hace uso nuevamente de la I.U. para solicitar más datos ó información, y si se encuentra la solución al problema con lo proporcionado anteriormente, notifica esta solución al usuario por medio de este mismo módulo.

La I.E. se utiliza para almacenar los primeros conocimientos en forma estructurada a la B.C., pero después de un tiempo se puede dar el caso (y así sucede la mayoría de las veces) de querer adicionar más conocimiento, esto se puede hacer utilizando el modulo de I.E., la cual es la característica de flexibilidad de los S. E. que les permite añadir, modificar o restringir información dependiendo de los requerimientos que se tengan.

[2] Ibidem., p. 32

CAPITULO IV

TIPOS DE APRENDIZAJE

Cuando se habla de aprendizaje no se toma en cuenta la manera en que realmente adquirimos el conocimiento porque las personas lo hacemos de una manera natural casi sin percibirlo, sin embargo al querer hacer que las computadoras "aprendan" nos damos cuenta de lo difícil que es en realidad. En los sistemas de I.A., lo optimo sería que aprendieran por si sólos y aunque se han dado grandes pasos para lograr este objetivo, aun no se ha llegado a alcanzar totalmente.

El aprendizaje se divide básicamente en dos tipos que son el repetitivo y el cognoscitivo. Esto no se debe confundir con los métodos por los que se puede adquirir el conocimiento ya que son mecanismos que permiten que el aprendizaje ocurra.

En el aprendizaje repetitivo los conocimientos se adquieren por memorización como aprendimos las tablas de multiplicar, las letras e inclusive las tareas que requieren de una secuencia de acciones como sería el caso de un carpintero que para hacer una mesa primero corta la madera, luego la pule, la ensambla y por último la pinta. El repetir una secuencia nos lleva poco a poco a la especialización y todo lo que se aprende son puntos específicos de información. Las secuencias que manejamos se les conoce más formalmente como procedimientos.

El otro tipo de aprendizaje es el cognoscitivo y es el más difícil de aplicar en una computadora ya que requiere de analizar, organizar y correlacionar elementos específicos del conocimiento; ésto nos asegura poder definir una clase a través de los atributos que la caractericen. Por ejemplo si hemos aprendido lo que es la letra "*a*" la reconocemos siempre aunque no esté escrita justo como la aprendimos, ya que contamos con el conocimiento específico que generalizamos para cualquier forma de escritura. Esta capacidad para generalizar los conocimientos básicos es lo que separa a la computadora del hombre y cuando se haya logrado traspasar esta barrera, sólo entonces los sistemas de I.A. podrán aprender realmente.

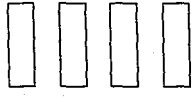
4.1 APRENDIZAJE POR ESTUDIOS DE DIFERENCIA

Para comenzar con este tipo de aprendizaje, es indispensable hacer un recordatorio de lo que es la inducción: es aquel razonamiento que parte de lo particular a lo general; nos será de gran utilidad tener presente esta definición para entender mejor lo que es un heurístico de inducción; ya que éste se utiliza mucho en el proceso de aprendizaje mediante el estudio de diferencias.

Fundamentalmente el estudio de diferencias, utiliza una diversidad de heurísticos de inducción gracias a los cuales los procedimientos conocen las especificaciones de la clase o especie que se pretende sean aprendidas; todo ello puede llevarse a cabo con ejemplos acertados y no acertados de dichas especificaciones. Primero

debemos entrenar al procedimiento con un elemento acertado de la clase por aprender, para elaborar una descripción preliminar, esta descripción tiende a crecer como resultado de toda la información que se proporciona de la especie, llegando así a obtener un *modelo de progresión*. Es importante señalar que los ejemplos no acertados son de suma importancia, ya que de ellos se derivan los detalles, especificaciones y características esenciales de la clase; en este sentido se dice que el procedimiento aprende o se entrena mejor con ejemplos no acertados que con los acertados, veámoslo más de cerca analizando el siguiente ejemplo:

Consideremos un conjunto de cuatro pequeños rectángulos como los mostrados a continuación, ahora supongamos que estamos entrenando a un procedimiento x sobre la forma de acomodar dichos rectángulos de tal modo que se forme un cuadrilátero sólido con ellos.



Como mencionábamos anteriormente, el procedimiento x debe comenzar su aprendizaje con un ejemplo acertado, para después continuar con la sucesión de ejemplos tanto acertados como no acertados, la sucesión se ejemplifica en la figura 4.1

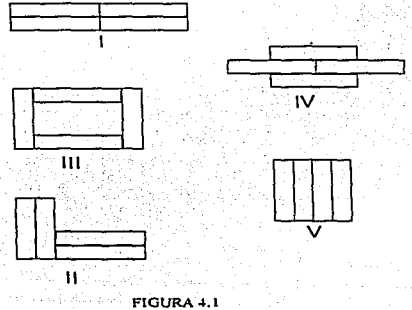


FIGURA 4.1

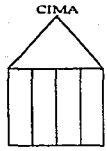
El procedimiento *x* utiliza los ejemplos no acertados para observar porqué no cumplen con lo que se requiere, aunque los rectángulos forman de algún modo un cuadrilátero ó una figura sólida, no es exactamente eso lo que se pide que aprenda; sin embargo estos ejemplos son de ayuda para el procedimiento, porque gracias a ellos obtiene información del acomodo correcto de los rectángulos lo que se conoce como la heurística de ***enlace necesario***. Este enlace como su nombre lo indica, es utilizado cuando el modelo progresivo posee un enlace, el cual no se encuentra en los ejemplos no acertados. Significa entonces que dicho enlace del modelo es necesario; es decir, es indispensable que exista.

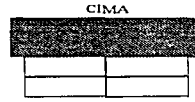
Por ejemplo en la fig. 4.1 los enlaces necesarios podrían ser que los rectángulos se encuentren unidos uno a otro, pero a su vez se encuentren alineados. Cuando se considera ésta heurística, ningún ejemplo que no tenga los enlaces necesarios será reconocido como cuadrilátero sólido.

Otra heurística importante, es la llamada de ***enlace prohibido***. Como es de imaginarse, esta heurística, es opuesta a la de enlace necesario, debido a que es utilizada cuando un ejemplo de los no acertados cuenta con un enlace que el modelo progresivo no tiene, es así como el enlace del ejemplo no acertado se transforma en un enlace prohibido, no debe ser encontrado. Como ejemplo de esta prohibición de enlace observemos en la fig. 4.1 el punto III, el acomodo de los rectángulos cumple con los enlaces necesarios (unidos uno a otro y alineados) no obstante existe una distancia entre ellos a la que podemos considerar como un enlace prohibido, puesto que esta distancia hace que no se cumpla con la solidez que se pide.

Estas dos heurísticas son consideradas las más relevantes en el aprendizaje por estudio de diferencias, puesto que en ellas existen las descripciones esenciales de una clase, además la información que de ellas se obtiene sirve para encaminar a la elaboración de deducciones ó conclusiones más acertadas. Existen otras heurísticas que también son utilizadas por este tipo de aprendizaje, algunas de ellas son:

Ascenso de árbol : Esta heurística se usa cuando el modelo en desarrollo contiene un elemento cualquiera en un lugar específico, pero en un ejemplo se encuentra un elemento diferente al del modelo en ese mismo sitio; es decir, supongamos que en las figuras que conformaron un cuadrilátero sólido (fig. 4.1) se quisiera identificar una cima en ellas, todo ello nos llevaría a algo como lo siguiente:





Para los dos casos es válido la cima mostrada en las figuras, a menos que se especifique una figura especial como cima, lo cual lleva al procedimiento de manejar la cima como si fuera una variable de tal forma que sea válida la figura que se nos ocurra colocar en lo alto del cuadrilátero y así poder dar una apertura general a la clase de cima.

Ampliación de conjunto: Esta heurística se utiliza en aquellos casos en los cuales un elemento del modelo en evolución es totalmente diferente a su correspondiente de un ejemplo dado, esta gran diferencia se refiere a que no existe la más mínima relación entre dichos elementos; más propiamente dicho, estos elementos son de clases totalmente diferentes, en estos casos el procedimiento se ve obligado a ampliar sus dominios hacia un conjunto nuevo, por lo tanto la clase nueva a utilizar la forman los dos conjuntos a los que pertenecen dichos elementos divergentes.

Intervalo cerrado: Es la heurística utilizada cuando el modelo progresivo hace uso de un número, o de una serie de números que se manejarían como intervalos. Supongamos que el modelo usa una medida y su correspondiente en el ejemplo es otra, el procedimiento elabora un intervalo que abarcaría las dos medidas proporcionadas, de igual manera se procedería con los intervalos manejados por el modelo y por el ejemplo, haciendo un sólo intervalo que cubra a los dos.

Enlace eliminado: El procedimiento hace uso de esta heurística en aquellos casos en los que aparece un enlace del modelo progresivo pero que no se encuentra en el ejemplo, simplemente el procedimiento suprime este enlace, ya que deduce que no es de importancia. (por ejemplo un enlace que se podría eliminar en el ejemplo de los rectángulos podría ser el tamaño de los mismos si los hubiera; el procedimiento no tomaría en cuenta las dimensiones proporcionadas por un ejemplo, si en otros ejemplos no se da el tamaño del rectángulo).

4.2 APRENDIZAJE POR EXPERIENCIAS

Es muy común que en un proceso de enseñanza-aprendizaje se instruya a los discípulos con antecedentes de un tema determinado y posteriormente con una gran variedad de ejemplos y ejercicios acerca de dicho tema; logrando así que los estudiantes identifiquen los rasgos más sobresalientes de los ejercicios, del mismo modo con la ayuda de estos ejemplos y los antecedentes proporcionados puedan reconocer y retener las generalidades del tema tratado, para poder hacer uso de ellas en el momento que se requiera.

Una forma típica de conseguir el aprendizaje por medio de explicaciones es confrontando una definición que se ha dado como antecedente con la descripción de un ejercicio, cuando se obtiene cierta similitud entre el precedente y el ejemplo se procede a construir lo que es conocido como regla de "antecedente-consecuente"[1] . Para una utilización eficaz de un antecedente se debe dar una definición o descripción correcta bajo el contexto que se esté manejando. Por ejemplo imaginemos que se tiene lo siguiente como antecedente.

Término: Es una expresión algebraica compuesta por varios símbolos, sin ser separados por los signos de suma (+) ó de resta (-). Siendo los principales elementos del término los siguientes:

Signo: pueden ser negativos ó positivos; los primeros van precedidos por el signo (+) aunque comúnmente suele suprimirse por ejemplo $+2x$ ó simplemente $2x$ es un término positivo, los segundos van precedidos por el signo (-) por ejemplo $-2x$ es un término negativo, a diferencia de los anteriores siempre debe aparecer el signo.

Coeficiente: Generalmente es el primer factor del término, por ejemplo en $2x$ el coeficiente es 2 y en $-6x$ el coeficiente es -6. Si el coeficiente es la unidad ésta es omitida por lo tanto en x su coeficiente es 1.

literal: Como su nombre lo indica son las letras que se encuentran dentro del término, por ejemplo en $3x$ la parte literal es x.

Exponente: Es el número pequeño colocado a la derecha y arriba de la parte literal del término así en $5x^3$ el exponente es 3, de igual forma que en los coeficientes, si el exponente es la unidad ésta no aparece por lo que en $-6x$ el exponente es 1.

Grado: Es el valor del exponente; por ejemplo un término es de cuarto grado si el valor del exponente es 4.

Polinomio: Es una expresión algebraica compuesta por más de un término. Ejemplo $9x^3 + 16x^2 - 2x$

[1] Patrick H. Winston, "Inteligencia artificial." p.391

Grado de un polinomio: Es el grado del término con el mayor exponente. Por ejemplo en el polinomio $2x^4-2x^2-12x$, el exponente mayor es 4; por lo que resulta ser de cuarto grado.

Ahora, si se pide que sea identificado el grado de un polinomio con los antecedentes que se han proporcionado, se llega fácilmente a la respuesta.

En la mayoría de los casos es recomendable dar una excelente representación del antecedente, con el objeto de hacer más fácil lo que es difícil, para el ejemplo anterior podemos utilizar una red semántica simple como la mostrada en la figura 4.2 en la cual se indica que en un polinomio "P" el término con mayor exponente da el grado de ese mismo polinomio.

FIGURA 4.2

En casos particulares en los que se logra describir un enlace como un nodo con enlaces más detallados es conocido como "nodo materializado"[2] de ésta manera, llegamos a la representación mostrada en la figura 4.3 en la cual el enlace término con el mayor grado se convierte en nodo materializado con sus enlaces y descripciones muy particulares, las cuales describen como es que se llega a obtener el grado del polinomio "P".

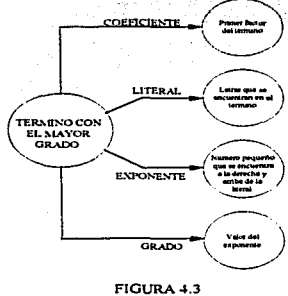


FIGURA 4.3

[2] Ibidem., p.393

En este tipo de conocimiento es común que se resuelvan problemas por medio de analogías obteniendo estas analogías por medio de una buena transferencia de los antecedentes a un ejercicio dado utilizando un registro conocido como "patrón de explicación"[3] siendo el registro los enlaces que provoca el antecedente; el patrón se utiliza para determinar las correspondencias de la transferencia. En base a todo esto, el procedimiento debe hacer uso correcto de los antecedentes, en el momento de comparar con un nuevo ejemplo; es decir, el procedimiento debe identificar que tanto los antecedentes como el ejemplo nuevo tengan un mismo contexto, por ello es necesario hacer énfasis en los enlaces que provocan situaciones similares, si éstos enlaces son pocos es muy probable que se trate de contextos distintos; por otro lado, si son varios es probable que se estén tratando dos situaciones bajo un mismo contexto.

Teniendo en cuenta la importancia de una buena ubicación en cada situación para poder distinguir si se trata de objetos similares; es necesario que los procedimientos cuenten con la capacidad suficiente para aprender en base a los ejemplos y las descripciones que se den tanto en el aspecto físico como en el de las funciones de cada uno de ellos en particular y además poder identificar que efectivamente la descripción física cubra las necesidades de la descripción funcional. Para relacionar la descripción de la estructura física con la descripción de las funciones del objeto, son necesarios precedentes para justificar como es qué, con base a las características físicas que posee, puede desarrollar adecuadamente sus funciones.

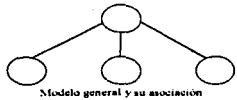
4.3 APRENDIZAJE UTILIZANDO MODELOS

Este tipo de aprendizaje ayuda a evitar errores de interpretación en un ejemplo cuando éste tenga una modificación, siempre y cuando esta misma sea lógica.

Aquí se maneja el concepto de modelos generales y modelos específicos; dentro de los cuales se utilizan ejemplos negativos y positivos bien definidos (libres de ruido) a estos se les determinan atributos que puedan ser valorados para crear un *espacio de versiones* que contribuyan a determinar si el ejemplo pertenece o no a una clase dependiendo de las características distintivas de dicha clase.

La representación del espacio de versiones se hace mediante un árbol de especialización y otro de generalización como los mostrados en la figura 4.4

[3] Ibidem., p.395



Modelo general y su asociación con muestras negativas.

a

Modelo específico y su asociación con muestras positivas.

b

FIGURA 4.4

Cada nodo dependiendo si es una muestra negativa o positiva (esto es conforme a la interpretación del ejemplo), se asociará con un modelo general o específico respectivamente y no podrán existir nodos sin asociación; así mismo, los enlaces son relaciones de generalización o bien especialización entre los modelos. La información que contienen los nodos le permiten al espacio de versiones tener un registro de los datos útiles de los ejemplos de aprendizaje sin tener que recordarlos con exactitud.

Al contrario de lo que se pensaría, el árbol no crece indefinidamente ya que se podan muchas ramas al ir avanzando en los ejemplos. Si es un ejemplo negativo, los modelos específicos que coincidan con él se eliminan y si es un ejemplo positivo se eliminan los modelos específicos que no coincidan con el mismo. Esto asegura que el modelo nuevo se modifique sólo un poco con respecto al anterior y "así los modelos generales se hacen más específicos para evitar coincidir y los específicos se hacen más generales para asegurar que coincidan"[4] , lo que al final produce un modelo que converge con todos los ejemplos positivos vistos y con ningún ejemplo negativo.

La figura 4.4a, nos muestra que cada uno de sus nodos se asocia con un modelo general el cual coincide con todos los demás nodos, por otro lado, la figura 4.4b señala a cada nodo como miembro de un modelo específico que coincide sólo con el primer ejemplo positivo visto. "La idea clave en el aprendizaje de espacio de versiones es que la especialización de los modelos generales y la generalización de los modelos específicos finalmente conduce a un sólo modelo garantizado como correcto"[5] .

[4] Ibidem., p.443
[5] Ibidem., p.442

Para entender mejor este tipo de aprendizaje veamos un ejemplo: supongamos que una estilista ha trabajado en estéticas de diferentes lugares y le interesa saber el porqué en el actual salón donde está trabajando existe un problema de caída de pelo más notorio, así crea la siguiente tabla basándose en atributos que ella supone son la causa más lógica del problema.

CLIENTE	TIPO DE SHAMPOO	SEXO	CORTE DE PELO	LAVA EL PELO	CAIDA DE PELO
1	Marca A	Femenino	Cada 2 meses	Diario	si
2	Marca B	Masculino	Cada 2 meses	Cada 3er. día	no
3	Marca A	Masculino	Cada mes	Diario	si
4	Marca C	Femenino	Cada 3 meses	Diario	no
5	Marca A	Femenino	Cada 3 meses	Cada 3er. día	no

El modelo específico se representará en el caso 1 como [Con shampoo A, Femenino, Cada 2 meses, diario] y el modelo general quedaría [?,?,?,?], nótese que aquí todos los atributos tienen valores. [?,?,?,?], en donde cada signo de interrogación podría tomar cualquier valor para el atributo y por último, si ponemos un modelo de la siguiente manera como [Con shampoo A,?,?, diario] en donde no se señalan los valores de ciertos atributos en este caso sexo ni cada cuando se corta el pelo y si se marcan los valores de los otros; se dice que es un modelo completamente específico y completamente general. Este tipo de modelo coincide con cualquier situación en la que se utiliza el shampoo A para lavarse el pelo diariamente si se quisiera generalizar el modelo como [Con shampoo A,?,?, diario] sólo se sustituiría el valor de una interrogación para quedar como [Con shampoo A,?,?,?] y si lo que deseamos es especializarlo se pondría el valor de algún atributo y quedaría como [Con shampoo A,?, Cada 2 meses, diario].

Aquí es donde se nota la importancia de este tipo de aprendizaje en el cual sólo se necesitan unas muestras y no todo un cuadro completo de atributos con valores: en el ejemplo visto el tipo de shampoo puede tener 3 posibles combinaciones A, B ó C, 2 posibilidades en el segundo atributo, 3 en el tercero y 2 en el cuarto, lo que nos daría una tabla con 3 X 2 X 3 X 2 = 36 posibles combinaciones sustituidos por los 5 ejemplos observados que tenemos. Pueden parecer pocas combinaciones pero en un ejemplo un poco más detallado el número se elevaría considerablemente.

Veamos el primer ejemplo que nos da la pauta para representar nuestro modelo más general [?,?,?,?] y ya que la primera muestra es positiva también construiremos el modelo más específico como [Con shampoo A, Femenino, Cada 2 meses, diario]

ESPACIO DE VERSIONES DEL EJEMPLO VISTO.

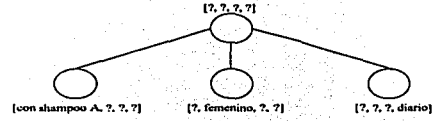
MODELO GENERAL

MODELO ESPECÍFICO

[con shampoo A, femenino, cada 2 meses, diario]

El ejemplo 2 por ser negativo hace que el modelo general se especialice; el nuevo modelo debe contener sólo una pequeña modificación del anterior, es por ésto que se toman como punto de partida los valores de los atributos del modelo más específico, tomando un atributo a la vez y sustituyendo los demás por signos de interrogación y así sucesivamente para asegurar al final la convergencia de ambos modelos.

MUESTRA NEGATIVA:
[con shampoo B, masculino, cada 2 meses, cada 3er. día]

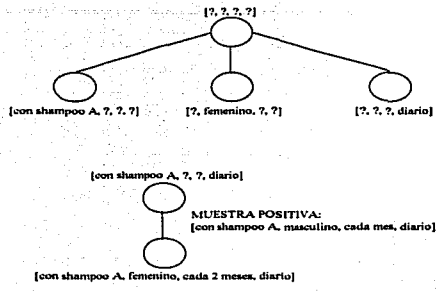


[con shampoo A, femenino, cada 2 meses, diario]

En este nuevo espacio de versiones no se marca la especialización [?,?, Cada 2 meses,?] porque converge con el ejemplo negativo. Veámoslo de la siguiente manera; el primer ejemplo es positivo y el segundo negativo, en el primero el shampoo A y en el segundo B (no coinciden), en el primero el sexo es femenino y en el segundo masculino (tampoco coinciden), sin embargo en el primer ejemplo se corta el pelo cada dos meses y en el segundo también, como coincide esta especialización se elimina porque nos señala que se presenta un ejemplo positivo y uno negativo, ésta no es causa directa de la caída de pelo por lo tanto la suprimimos, de esta manera vamos "podando" el árbol, para el cuarto atributo del primer y segundo ejemplo no coinciden por lo tanto no se elimina. El ir comparando los valores de los atributos tanto a nivel general como específico es un requisito; éste

señala que cada vez que un modelo general se especializa, las especializaciones deben ser generalizaciones del modelo específico y cada vez que se generaliza un modelo específico ésta debe ser una especialización de un modelo general, lo cual asegura la convergencia del modelo general y el modelo específico en algún punto, lo que nos conducirá al modelo final buscado.

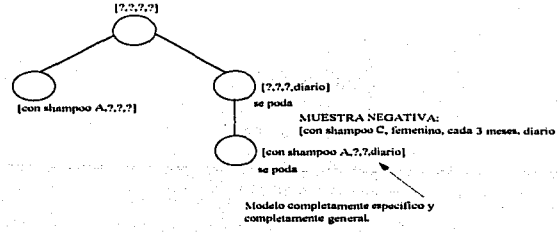
Como el siguiente ejemplo es positivo debemos generalizar el modelo específico.

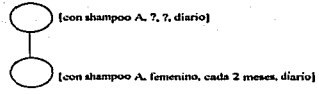


Surge entonces un nuevo modelo específico que se compara con el ya existente y los valores de los atributos que no coincidan se sustituyen por interrogaciones. Al comparar [Con shampoo A, Femenino, Cada 2 meses, diario] con [Con shampoo A, Masculino, Cada mes, diario] quedaría [Con shampoo A,?,?, diario].

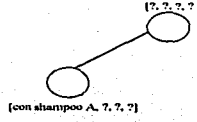
También debemos podar al modelo general [?, Femenino,?,?] ya que no converge con el ejemplo [Con shampoo A, Masculino, Cada mes, diario] por ser el valor de nuestro segundo atributo diferente, así que se toma el del ejemplo positivo y el otro se elimina.

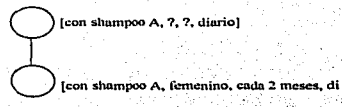
El siguiente ejemplo es [Con shampoo C, Femenino, Cada 3 meses, diario], por ser negativo se compara con los dos modelos generales que tenemos. El primero [Con shampoo A,?,?,?] como no coinciden permanece como está y el segundo [?,?,?, diario] como sí lo hace se especializa pero en dirección de una generalización del modelo específico, es decir, se debe tomar en cuenta también el modelo específico existente por lo que tenemos aquí un modelo "completamente específico y completamente general" siendo éste [Con shampoo A,?,?, diario]



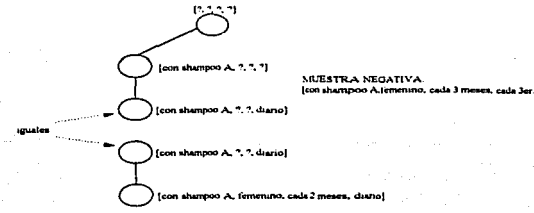


Cabe mencionar que una especialización de un modelo general nunca puede pertenecer a dos ó más modelos generales, si ésto ocurre se poda tanto la especialización como los modelos que la generen, este caso ocurre en nuestro árbol actual, si observamos [Con shampoo A,?,?, diario] puede ser una especificación de [Con shampoo A,?,?,?] por lo tanto se poda quedando de la siguiente manera:





Hemos llegado a nuestro último ejemplo [Con shampoo A, Femenino, Cada 3 meses, cada tercer día], como es un ejemplo negativo se especializa el modelo general; al checarlo con [Con shampoo A,?,?,?] si converge por lo que se debe especializar en dirección al modelo específico y la única manera de hacerlo es poniéndolo como [Con shampoo A,?,?,diario], construyendo así un modelo igual al específico.

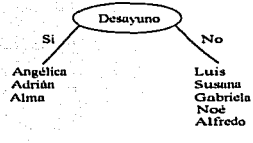


Al converger ambos modelos (general y específico) se ha encontrado la solución:
"El shampoo A al utilizarlo diariamente, produce la caída de pelo."

Para terminar señalaremos que el procedimiento de espacio de versiones, utilizado en el aprendizaje mediante el manejo de varios modelos, nos permite reconocer si los ejemplos son positivos o negativos a simple vista (los cuales son usados de manera simétrica) después, claro está, de haber manipulado sólo unos cuantos de ellos. Esto nos permite formarnos una idea de el modelo final aún antes de haber llegado al mismo.

4.4 UTILIZANDO ARBOLES DE IDENTIFICACION

El aprendizaje mediante arboles de identificación es el más utilizado actualmente. Un árbol de identificación también conocido como árbol de decisión es la representación de datos en forma de listas que señalan características de la clase que se este ramificando y en el cual considerando los atributos de los objetos, estos se pueden seguir por alguna rama.

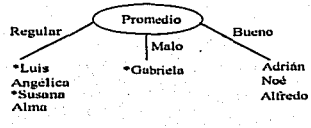


Este aprendizaje lo explicaremos a través del siguiente ejemplo: supongamos que se lleva a un grupo de alumnos de excursión, tres de los cuales sufre malestar estomacal, como el total de los niños son ocho, se considera la incidencia muy grande por lo que se pretende conocer la causa del síntoma considerando los valores de la tabla:

NOMBRE	PROMEDIO	EDAD	COMPLEXION	DESAYUNO	MALESTAR
Luis	regular	8	delgada	no	si
Angélica	regular	9	regular	si	no
Adrián	bueno	7	regular	si	no
Susana	regular	7	regular	no	si
Gabriela	malo	8	gordo	no	si
Noé	bueno	9	gordo	no	no
Alfredo	bueno	8	gordo	no	no
Alma	regular	7	delgada	si	no

Trabajaremos sobre esta tabla para demostrar que no es necesario considerar las 54 posibles combinaciones ($3_{promedio} \times 3_{edad} \times 3_{complexion} \times 2_{desayuno} = 54$), para llegar a saber lo que causa el malestar.

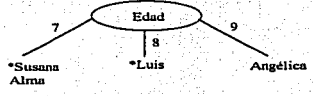
Para empezar debemos hacer un árbol de identificación que según el concepto de "Navaja de Occam"[6], debe ser lo más sencillo posible; sin perder de vista este concepto seleccionamos una prueba para el nodo raiz que divida las muestras que se tienen en subconjuntos, se pueden hacer varios árboles y después seleccionar el que tenga el subconjunto más homogéneo.

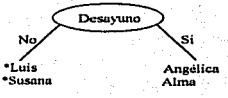


[6] Ibidem., P.459

NOTA: * = Tiene malestar

Si vemos el conjunto más adecuado es el de promedio ya que dos de sus tres ramas son homogéneos. Para la rama no homogénea se selecciona otra prueba para dividir las muestras nuevamente en subconjuntos lo más homogéneos posibles.





El conjunto que tiene sus ramas finalmente homogéneas es el de desayuno por lo tanto el árbol ha llegado a su fin.

El método que hemos utilizado para generar nuestro árbol es el de SPROUTER que resumiéndolo quedaría en los siguientes pasos:

1.- Seleccionar un nodo hoja con muestras no homogéneas.
2.- Seleccionar una prueba para dividir las muestras en subconjuntos lo más homogéneos posibles.
3.- Si los subconjuntos ya no pueden volverse más homogéneos el árbol deja de crecer.
4.- Si los subconjuntos pueden volverse más homogéneos ir al paso 1.

En un árbol grande no es posible visualizar si el dirigirse por una rama u otra nos garantice un conjunto homogéneo e inclusive tampoco que éste sea lo más homogéneo posible. Para auxiliarse se aplica la fórmula que mide la falta de homogeneidad de los subconjuntos producidos por una prueba a la cual se le da el nombre de desorden:

$$\text{desorden} = \Sigma_c -[(n_{bc}/n_b) \log_2 (n_{bc}/n_b)]$$

Sacada de la fórmula de desorden promedio

$$\text{desorden promedio} = \Sigma_b (n_b/n_t) X [\Sigma_c -[(n_{bc}/n_b) \log_2 (n_{bc}/n_b)]]$$

donde:

n_b	es el numero de muestras en la rama b.
n_t	es el numero total de muestras en todas las ramas.
n_{bc}	es el total de muestras en la rama b de la clase c.

"El desorden es 0 cuando el conjunto es perfectamente homogéneo y uno cuando el conjunto es perfectamente no homogéneo"[7] .

Ya que se tiene el árbol de identificación se convierte en reglas que surgen de cada trayectoria del árbol (desde el nodo raíz hasta el nodo hoja) y en los cuales se anotan los resultados de las pruebas como antecedentes y la clasificación del nodo hoja como consecuente. Quedando para nuestro caso de la siguiente manera:

Si	El promedio es regular
	El niño desayuno
Entonces	No tiene malestar
Si	El promedio es regular
	El niño no desayuno
Entonces	Si tiene malestar
Si	El promedio es bueno
Entonces	No tiene malestar
Si	El promedio es malo
Entonces	Si tiene malestar

[7] Ibidem., p.462

Para simplificarlas, se deben eliminar los antecedentes y las reglas innecesarias, y crear otras por omisión de las que comparten el mismo consecuente, éstas funcionarán cuando ninguna otra lo haga; así mismo si existen ataduras entre ellas se eliminarán con un rompedor de ataduras heurístico. Este método para generar reglas a partir de un árbol de identificación es conocido como PRUNER.

Para eliminar los antecedentes de las reglas se revisa si no son útiles en la función de la misma, a través de una tabla de contingencias en donde se coloca el antecedente y su negación en los renglones y el consecuente y su contraparte en las columnas.

Para determinar los valores correspondientes de las casillas de la tabla de contingencias retomaremos la tabla A.

En la regla 1, tenemos

Si	El promedio es regular
	El niño desayuno
entonces	No tiene malestar

Como la tabla de contingencias es sobre el promedio, buscamos a las personas que cumplan con el segundo antecedente, es decir, a las que desayunaron, en nuestra tabla A contamos con tres (sólo para una mejor visualización realizaremos una tabla llamada B, pero no es necesario hacerla cada vez que analicemos una regla).

NOMBRE	PROMEDIO	EDAD	COMPLEXION	DESAYUNO	TIENE MALESTAR
Angelica	regular	9	regular	si	no
Adrián	bueno	7	regular	si	no
Alma	regular	7	delgada	si	no

Después revisamos si el promedio es regular: así es, el resultado se debe poner en el primer renglón de nuestra tabla de contingencias, también revisamos si tuvo malestar, como no fue así, entonces el resultado se anota en la primera columna. Pasamos a la siguiente muestra y nuevamente revisamos el promedio: como no es regular el resultado debe estar en el segundo renglón, tampoco tuvo malestar así que se anota un punto en la columna uno. Como en la tercer muestra el promedio es regular y no tuvo malestar se incrementa un uno en la primer columna y el primer renglón de la tabla que estamos elaborando quedando de la siguiente manera:

	NO TIENE MALESTAR	SI TIENE MALESTAR
EL PROMEDIO ES REGULAR	2	0
EL PROMEDIO NO ES REGULAR	1	0

La tabla muestra que el promedio regular no influye por si sólo, en el malestar del estudiante o no el malestar, ésto sería razón suficiente para descartarlo, sin embargo recordemos que esta regla consta de dos antecedentes por lo que hay que revisar ambos antes de llegar a una conclusión de este tipo.

Para el otro antecedente de la misma regla tenemos:

	NO TIENE MALESTAR	SI TIENE MALESTAR
EL NIÑO DESAYUNO	2	0
EL NIÑO NO DESAYUNO	0	2

En este caso se ve que la condición si influye en el resultado y se debe conservar el antecedente. Como ya se ha verificado que la regla no se altera si el primer antecedente desaparece, éste se elimina quedando la regla:

Si El niño desayuno
entonces No tiene malestar

Al igual que para la primera regla, se deben de elaborar tablas de contingencia para cada uno de los antecedentes de las reglas restantes, como se muestra a continuación:

Regla 2
Si El promedio es regular
El niño no desayuno
entonces Si tiene malestar

	NO TIENE MALESTAR	SI TIENE MALESTAR
EL PROMEDIO ES REGULAR	0	2
EL PROMEDIO NO ES REGULAR	2	1

	NO TIENE MALESTAR	SI TIENE MALESTAR
EL NIÑO DESAYUNO	2	0
EL NIÑO NO DESAYUNO	0	2

Regla 3

Si El promedio es bueno

entonces No tiene malestar

Como en esta regla (y en la siguiente), sólo existe un antecedente, las ocho muestras con las que contamos se incluyen en la tabla.

	NO TIENE MALESTAR	SI TIENE MALESTAR
EL PROMEDIO ES BUENO	3	0
EL PROMEDIO NO ES BUENO	2	3

Regla 4

Si El promedio es malo

entonces Si tiene malestar

	NO TIENE MALESTAR	SI TIENE MALESTAR
EL PROMEDIO ES MALO	0	1
EL PROMEDIO NO ES MALO	5	2

Los antecedentes de las tres últimas reglas son importantes para que estas se disparen por lo tanto, ninguno de ellos se elimina.

Todos los antecedentes de una regla pueden eliminarse sólo cuando la totalidad de las muestras tengan el mismo resultado.

Para crear las reglas por omisión se revisan las reglas que surgieron del procedimiento anterior

Si El niño desayuno

entonces No tiene malestar

Si El promedio es regular

El niño no desayuno

entonces Si tiene malestar

Si El promedio es bueno

entonces No tiene malestar

Si	El promedio es malo
entonces	Si tiene malestar

De estas se pueden crear dos reglas por omisión considerando los consecuentes similares:

Si	Ninguna otra regla se aplica
entonces	No tiene malestar

Si	Ninguna otra regla se aplica
entonces	Si tiene malestar

Es obvio que no se pueden aplicar las dos reglas mostradas y eliminar las 4 anteriores, sólo se puede poner una de ellas que sería la que englobara el mayor número de reglas relacionadas con ella misma. Sin embargo, ambas reglas por omisión contienen dos reglas y se debe utilizar otro criterio para romper ataduras; podría ser también el conservar aquella regla que contenga las reglas más complejas, entendiéndose por esto las que contengan más antecedentes, tomando este criterio tenemos nuestro conjunto de reglas final de la siguiente manera:

Si	El niño desayuno
entonces	No tiene malestar

Si	El promedio es bueno
entonces	No tiene malestar

Si	Ninguna otra regla se aplica
entonces	Si tiene malestar

Otro rompedor de ataduras consiste en utilizar la regla por omisión que contenga el consecuente más común de nuestra tabla muestra (tabla A), el cual resulta ser el de "No tiene malestar". Utilizando este criterio el conjunto de reglas queda:

Si	El promedio es regular
	El niño no desayuno
entonces	Si tiene malestar

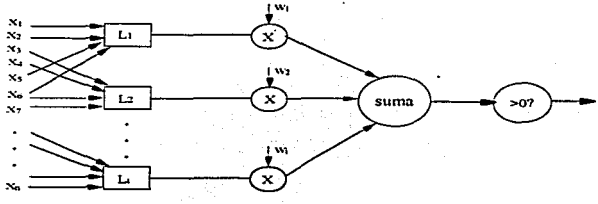
Si	El promedio es malo
entonces	Si tiene malestar
Si	Ninguna otra regla se aplica
entonces	No tiene malestar

La elección de la regla por omisión a utilizar depende en gran medida de la persona que requiera su utilización.

4.5 PERCEPTRONES

Los perceptrones son considerados como la red neuronal más simple y más especializada. Una red neuronal consta de varias neuronas y el perceptron es una de ellas.

El perceptron consta de tres elementos fundamentales: pesos entrenables multiplicativos, un sumador y una función de umbral, gráficamente se representa de la siguiente forma:



Donde:

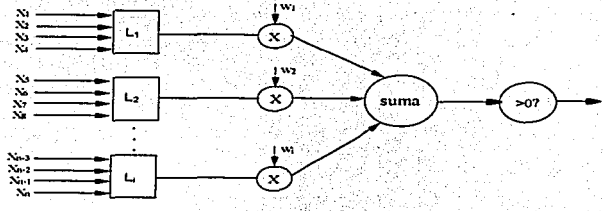
X_n	Son las "n" diferentes entradas al perceptron y su valor es binario (0-1).
L_n	Son las cajas lógicas, aparecen entre las entradas y los pesos del perceptron, su salida también es binaria (0-1) y su función es recibir una o más entradas y producir sólo una salida (internamente puede verse como una tabla lógica de decisión), aunque este tentado a utilizar sólo una caja para todas las

entradas, ésto haría que perdiera efectividad por lo que se sugiere que cada caja lógica de un perceptron atienda sólo un número reducido de entradas. Considerando "n" entradas la tabla interna de la caja manejaría 2^n posibles combinaciones y lo óptimo es que atienda "n" ó menos entradas.

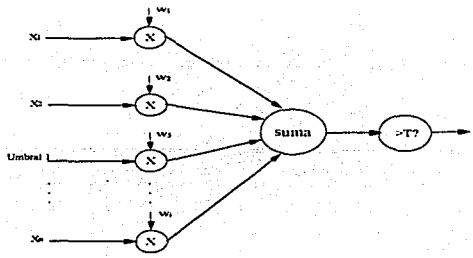
W_i	Son los pesos.
Suma	Es la suma ponderada. Se encarga de sumar los pesos ya relacionados con la salida de las cajas lógicas.
>0? = P	Da como resultado un 0 ó 1, ésta ya es la salida del perceptron y depende de si la suma ponderada es mayor que el umbral. Considerando al umbral como T tenemos que

$$P = \begin{cases} 1 & \text{si } \Sigma_i W_i \times L_i > T; \\ 0 & \text{en otro caso} \end{cases}$$

Otro tipo de perceptron es el limitado por el diámetro en el cual las entradas se distribuyen rectangularmente. Como se muestra a continuación:



Para entender mejor al perceptron consideremos el más simple que es el perceptron directo, donde sus cajas lógicas sólo tienen una entrada por lo que su salida es la misma, incluso pueden considerarse como perceptrones sin cajas lógicas.



Estos perceptrones son fácilmente entrenables para que su salida sea la deseada. Se comienza por colocar todos los pesos en cero, después se va probando con todas las muestras, cuando aparezca un valor no deseado, los pesos se aumentan; y si el valor es el esperado, no se modifica nada.

Supongamos que el valor no deseado es un 0 cuando se esperaba un 1 entonces el peso de las cajas lógicas que en ese momento producen el 1 se incrementa (lo más recomendable es en 1 unidad) y si por el contrario se logra un 1 cuando se requiere un 0, el peso de las cajas que producen el 1 se decrementa (en una unidad), esto nos ayuda a hacer menos probable un resultado indeseable. Los pesos de las cajas lógicas que producen un 0 en cualquiera de las dos situaciones no se alteran puesto que no influyen en el resultado final.

Por último se crea una entrada virtual extra cuyo valor es 1 el cual garantiza un umbral de 0; ésta se añade a la caja lógica. Es por esto que aunque no se utilice directamente, no se debe de eliminar del perceptron. Así garantizamos que el perceptron indique 1 (equivalente a si) siempre que la suma ponderada de las salidas de la caja lógica sea mayor que 0.

Podemos también representar las salidas y los pesos del perceptron en notación vectorial quedando $(L_1,L_2,...L_n)$ como el vector salida y $(W_1,W_2,...W_n)$ como el vector peso.

Ahora consideremos la siguiente tabla que nos servirá para ejemplificar como se puede entrenar al perceptron.

	PASO 1	PASO 2	PASO 3	
MUESTRAS	$X_1=(L_1)$	$X_2=(L_2)$	$X_3=(L_3)$	SALIDA DESEADA
1	0	0	1	0
2	0	0	1	1
3	1	1	1	1
4	1	1	1	1

Nótese que en un perceptron directo la entrada es igual que la salida.

En un principio el peso del perceptron es (0,0,0), revisando la tabla encontramos que en la segunda muestra del primer paso se produce un 0 cuando debería ser un 1 por lo que sumamos el vector de la muestra (0,0,1) al vector peso quedando ahora como (0,0,1), continuamos revisando la tabla y encontramos que en la muestra 2 del segundo paso hay un 0 y se esperaba un 1 por lo que se le suma al vector peso actual (0,0,1), el vector de la muestra (0,0,1) quedando un nuevo vector peso (0,1,0), el último error se encuentra en la primera muestra, tercer paso donde hay un 1 en lugar de un 0 lo que procede ahora es restar este vector (0,0,1) al de peso (0,1,0), quedando el vector peso final (0,0,1) el cual ya no se modifica. Así "el vector peso crece y se encoge conforme se desarrolla el aprendizaje"[8] .

4.6 REDES NEURONALES (R. N.)

Para poder entender mejor el aprendizaje por medio de las redes neuronales, es necesario considerar lo que actualmente se conoce sobre el funcionamiento de las neuronas, ya que es el mecanismo de estas complejas células el que se pretende imitar en las computadoras para lograr dotar de un cierto tipo de inteligencia a la máquina. Recordando un poco las investigaciones realizadas por el científico español Santiago Ramón y Cajal, nos encontramos con una descripción asombrosa de la composición de los tejidos cerebrales; esta estructura la forman miles de células nerviosas mejor conocidas como neuronas, las cuales generalmente están compuestas por un cuerpo celular con su respectivo núcleo, varias dendritas y un axón, éste último elemento es el que canaliza la salida de la neurona y con la ayuda de los árboles dendríticos se conecta hacia otras neuronas.

Actualmente se sabe que una neurona se encuentra "inhabilitada" hasta que recibe de otras neuronas algún tipo de excitación que rebase su nivel de umbral; ésta excitación provoca que la neurona produzca un tipo de reacción (salida) manifestada como impulsos que se propagan por el axon, según sea la intensidad de éste impulso para que pueda o no excitar a otras neuronas.

[8] Ibidem, p.513

Considerando lo anterior como plataforma para comprender a lo que se intenta llegar por medio de las R. N. artificiales, tenemos que estos sistemas están compuestos por elementos como el que se observa en la figura 4.5

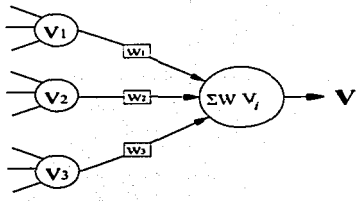


FIGURA 4.5

Notamos que en la figura a cada enlace le es asignado un peso determinado, los cuales son multiplicados por la entrada que le corresponde y seguidamente se suman todos los resultados obtenidos; posteriormente el resultado de la sumatoria es comparado con el nivel de umbral, si es mayor se dice que la neurona produce una salida "1" o simplemente que la neurona ha sido excitada en caso contrario la neurona no hace nada; es decir, queda "inhabilitada". Esto es a muy grandes rasgos la función de un sistema de redes neuronales artificiales, lógicamente un sistema de R. N. como ya se indico en la seccion anterior es una colección de perceptrones que se encuentran conectados entre sí, como lo podemos observar en la figura 4.6

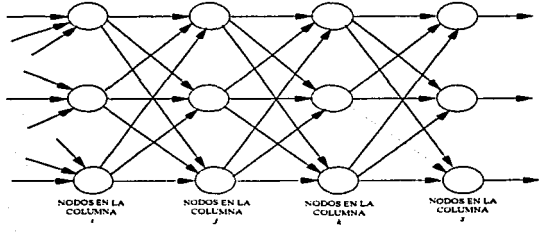


FIGURA 4.6

En estos tipos de sistemas las propiedades de la red en su conjunto dependen en gran medida de los elementos que las integran; así como de la conectividad y dinámica que exista entre ellos. El proceso de aprendizaje de los sistemas diseñados bajo estas características, consiste en entrenarlas. En dicho entrenamiento los pesos de las neuronas se modifican mediante una función de transferencia para que la red produzca el resultado que se desea, en este contexto se llegan a enfrentar problemas de diseño algorítmico para ir cambiando los pesos; aunque para ello se cuenta con el apoyo de algunos métodos matemáticos de optimización.

Un método para entrenar los pesos de las neuronas ya se vio en el tema de perceptrones, ahora veremos otro que ha resultado muy eficiente para la modificación de los pesos, el cual es el algoritmo de retropropagación, la idea general de la retropropagación es "hacer un cambio grande a un peso en particular, si el peso conduce a una reducción grande de los errores observados en los nodos de salida"[9] Si suponemos a Vd como un valor deseado para una muestra de entrada y a Vo como su valor obtenido, siendo W su peso asignado; si se observa una ligera diferencia entre Vd y Vo entonces resulta evidente un pequeño cambio en la salida del nodo; complementariamente, si esta diferencia es considerable, el cambio debe ser mayor, esto nos lleva a la conclusión de un cambio en el nodo, dicho cambio debe ser proporcional a la diferencia Vd - Vo para ese nodo en particular.

Es importante observar en la red de la figura 4.6 que al calcular un cambio de peso entre un nodo de la columna *i* y uno de la columna *j* (Wi-j), se necesita hacer uso de los cálculos requeridos para los pesos entre los nodos de las columnas *j* y *k* (Wj-k). Ahora esto expresado en un lenguaje matemático sería algo como lo siguiente:

Si y es una función de x, $y = f(x)$, para hacer incrementos a los valores de x
Como y esta en función de una sola variable x, entonces:

$$\Delta y = f(x)$$

Para hacer incrementos a los valores de x, el cambio debe ser proporciona a la derivada parcial de la función respecto de la variable x, lo que significa matemáticamente lo siguiente:

$$\Delta x \propto \frac{\partial y}{\partial x}$$

Ahora si y es función de varias variables x, y a su vez cada variable x es función de una variable z, para obtener la derivada parcial de y con respecto de z, hacemos uso de la "regla de la cadena" como se recordará, al hechar mano de esta regla; llegaríamos a la siguiente expresión:

[9] Ibidem p. 486

$$\frac{dv}{dz} = \Sigma \frac{\partial v}{\partial xi} \cdot \frac{\partial xi}{\partial z}$$

Teniendo como principal objetivo la mejoría de algunos pesos, necesitamos saber qué tan buenos son estos pesos y cómo se pueden mejorar; para ello, se suma el cuadrado del error de cada salida, al tenerlas todas se suman las entradas, anteponiendo el signo negativo para obtener un desempeño general con un máximo en cero, expresando ésto matemáticamente se obtiene lo siguiente:

$$P = -\Sigma (\Sigma(d_{sz} - o_{sz})^2)$$

En la que:

P, es el desempeño a obtener.

s, es un índice que fluctúa entre todas las muestras de entrada.

z, es un índice que fluctúa entre todas los nodos de salida.

d_{sz}, es la salida deseada para la muestra de entrada *s* en el nodo z-ésimo.

o_{sz}, es la salida real para la muestra de entrada s en el nodo z-ésimo."[10]

[10] Ibidem., p. 489

CAPITULO V

SOFTWARE DE APOYO PARA LA REALIZACION DE UN SISTEMA DE INTELIGENCIA ARTIFICIAL

Una de las razones por las cuales se decidió incluir el estudio del software de programación para S.E., fue el de conocer acerca de las herramientas para el desarrollo de sistemas de I.A. ya que éstas, facilitan en gran medida la creación de "sistemas inteligentes". Comparados con los lenguajes convencionales existentes, los lenguajes desarrollados para la I.A. se adaptan mucho mejor al ambiente requerido en esta materia. El LISP y el PROLOG han sido por mucho tiempo los lenguajes de programación más sobresalientes y por ende los más utilizados dentro del campo de la I.A.

5.1 EL LENGUAJE LISP

El lenguaje de programación LISP fue creado por John McCarthy al finalizar los años 50's en el MIT; es por ello considerado como uno de los primeros lenguajes de programación para aplicaciones en la I.A. McCarthy notó la necesidad de crear un lenguaje en el que se pudiera, más que almacenar datos, poder establecer relaciones entre ellos; es decir, un lenguaje en el cual se manejara más lo que nosotros conocemos como "sentido común".

Se le llamo LISP por sus siglas en inglés de "LISt Processor" (procesador de listas); es un lenguaje de procesamiento simbólico y como se indica en su nombre, maneja la información en forma de listas y también permite hacer uso de la recursión en sus procedimientos lo que es muy importante para los fines de la I.A. ya que con ésta, la estructura de la información es más flexible. Recordemos que la recursión es una forma de utilizar en la definición, el mismo término el cual se esta definiendo.

Continuando con algunos detalles del lenguaje, encontramos que una lista en LISP es una agrupación de elementos, normalmente conocidos como átomos, encerrados entre paréntesis; sus componentes pueden ser elementos individuales u otra lista. Existen listas vacías. Algunos ejemplos de listas son:

(0 1 2 3)
(a b c d e)
((Colombia Nicaragua) (Chile Paraguay))
( ) → Notación de lista vacía.

Un átomo en LISP es el elemento más pequeño dentro del lenguaje, éste puede ser un número, una palabra o bien un símbolo, debe tenerse en cuenta que en un átomo no es válido dejar espacios, ya que tiene la peculiar característica de ser indivisible, de ahí su nombre de átomo. A continuación se dan algunos ejemplos de átomos en LISP:

San_José	11.20	
Buenos_Aires	0.01	etc.

Existen varias funciones ya establecidas dentro del lenguaje de programación, estas funciones son llamadas "primitivas", las cuales realizan tanto operaciones de la matemática básica (+, -, *, /), como algunas otras manipulaciones simbólicas, entre las que destacan CAR, CDR, SET y otras más que se explicarán brevemente en los próximos párrafos.

Si bien es cierto que el LISP no es reconocido por su gran resolución matemática, esto no quiere decir que no se puedan desarrollar estas operaciones dentro del lenguaje. En LISP las operaciones suma, resta, multiplicación y división se efectúan de la siguiente manera:

(+ 7 13)	(- 20 3)	(* 8 29)	(/ 60 5)
20	17	232	12

En los ejemplos anteriores el primer componente que aparece en las listas indica la operación a efectuar y los siguientes dos elementos son llamados argumentos. Existe también un nombre específico para cada operación: PLUS, para la suma; DIFERENCE, para la resta; TIMES, para la multiplicación y QUOTIENT, para la división. Para no usar todo el nombre de la función LISP, se debe asignar la simbología correspondiente por medio de una expresión llamada SET, la cual se explicará más adelante. En LISP es posible efectuar operaciones un poco más complejas, como obtener funciones trigonométricas, exponenciales, etc., al final del tema se muestra una lista de algunas de estas funciones.

En lo referente a las operaciones simbólicas, iniciaremos con las primitivas de mayor relevancia.

CAR (Content of Address Register) En español sería el contenido de la dirección del registro, esta función cuando se aplica a una lista, devuelve el primer elemento de ésta; siempre y cuando no se trate de una lista vacía. Por ejemplo:

```
(CAR (0 1 2 3))
0
(CAR( (Colombia Nicaragua) (Chile Paraguay) )
(Colombia Nicaragua)
```

CDR (Content of Decrement Register) Contenido del decremento del registro esta primitiva recupera todos los elementos de una lista no vacía, con excepción del primero, por lo que a CDR se le conoce como el complemento de la primitiva anterior. Por ejemplo:

(CDR(0 1 2 3))
1 2 3
(CDR((Colombia Nicaragua) (Chile Paraguay))
(Chile Paraguay)

Con la combinación bien elaborada de éstas dos funciones es posible llegar a cualquier átomo de la lista a la que se le han aplicado las primitivas; es decir, si en la lista ((Colombia Nicaragua) (Chile Paraguay)) se desea únicamente obtener el átomo Nicaragua se tiene que desarrollar el siguiente proceso:

(CDR (CAR ((Colombia Nicaragua) (Chile Paraguay))))

LISP evalúa esta función como se hace normalmente con las funciones anidadas, comienza de la más interna a la más externa.

CONS Esta función permite insertar un nuevo elemento a una lista dada, este componente es colocado al principio de la lista (cabeza). Ejemplo:

(CONS 1 (2 3 4))
(1 2 3 4)
(CONS Cuba (Ecuador México Venezuela))
(Cuba Ecuador México Venezuela)
(CONS (a b c) (d e))
((a b c) (d e))

EQUAL Sirve para comparar dos expresiones ya sean átomos o listas, devuelve T (verdadero) cuando ambas son idénticas y NIL (falso) en caso contrario. Ejemplo:

(EQUAL (w x y z) (w x y z))
T
(EQUAL 12.07 13.07)
NIL

SET Con esta primitiva se pueden asignar valores a las variables. Aquí retomamos lo mencionado anteriormente, de la asignación de símbolos a las operaciones, con lo cual queda expresado de la siguiente forma:

(SET + PLUS)
(SET - DIFERENCE)

(SET * TIMES)

(SET / QUOTIENT)

Notamos en estos ejemplos que cada operación es asignada a su correspondiente símbolo, gracias a la expresión SET

(SET a (CAR (CDR (1 3 5 7)))) en este ejemplo, primero se evalúa CDR dando como resultado la lista (3 5 7) a la cual se le aplica la primitiva CAR, dando como resultado 3, que es el valor asignado a la variable "a", después de aplicar la función SET.

APPEND Por medio de esta función podemos unir dos listas, una enseguida de la otra formando una sola lista con las dos anteriores, como en el siguiente caso:

(APPEND (50 60 70) (80 90 100))

(50 60 70 80 90 100)

COND Esta primitiva permite elaborar funciones del tipo condicional. Se utiliza frecuentemente para tomar decisiones, su nombre se deriva de CONDitional, incluye la condición a cumplir con su respectiva acción a ejecutar; es estructurada de la siguiente forma:

(COND (condición acción)) Ejemplo:

(COND (EQUAL (x 50) acierto)

Lo anterior significa que cuando "x" sea igual a "50" devuelva la palabra "acierto".

DEFUN Gracias a esta primitiva el usuario puede definir procedimientos según sean sus necesidades, como se puede ver en su nombre DEfinir FUNción el usuario asigna el nombre a su nuevo procedimiento. Por ejemplo:

DEFUN penúltimo (1.0 2.1 3.2 4.3)

El nombre dado a la función es "penúltimo" y si el programador así lo indica, ya que se considera el nombre de la función, nos arrojará el penúltimo número de la lista que en este caso es 3.2

Estas expresiones son las más utilizadas dentro del lenguaje de programación LISP. A continuación se presenta un listado de algunas otras instrucciones utilizadas dentro del lenguaje.

ATOM Como se puede imaginar este predicado acepta solo un tipo de argumento, siendo su valor T, si el argumento es un átomo y NIL cuando no sea este caso

(ATOMP X)	(ATOMP V, Z, W)
T	NIL

EVENP Indica si un número es par

(EVENP 100)	(EVENP 19)
T	NIL

GREATERP Sirve para señalar en dos números si el primero de ellos es mayor que el segundo.

(GREATERP 718 691)	(GREATERP 15 74)
T	NIL

LESSP Esta primitiva indica en un par de números, si el primero es menor que el segundo.

(LESSP 4 9)	(LESSP 11 7)
T	NIL

LISTP Función que devuelve T si el elemento a analizar es una lista y NIL en otro caso.

(LISTP (h i j k))	(LISTP c)
T	NIL

LITERP Esta función nos auxilia cuando necesitamos conocer si el valor de un argumento es una letra o un átomo literal, indicándolo con T. En otro caso devuelve NIL.

(LITTERP y)	(LITTERP 493)
T	NIL

MINUSP Para los números, esta primitiva se utiliza para cambiar el signo.

(MINUSP -2)	(MINUSP -1)
2	1

MEMBERP Esta función acepta dos argumentos, siendo el primero un átomo y el segundo una lista, la expresión devuelve T si el primer argumento es igual a algún miembro del segundo y proporciona NIL cuando no ocurra ésto.

(MEMBERP 1 ((1,3), (2,4), (5,7)))	(MEMBERP 0 ((1,3), (2,4), (5,7)))
T	NIL

NULP Señala con T cuando su argumento es una lista vacía. Proporciona en cualquier otro caso NIL.

(NULP ( ))	(NULP a, b, c)
T	NIL

NUMBERP Devuelve T si un átomo es un número. de lo contrario proporciona NIL.

(NUMBERP 13)	(NUMBERP (m n o))
T	NIL

ODDP Se utiliza para señalar si un número es impar.

(ODDP 17)	(ODDP 10)
T	NIL

ONEP Indica si un número dado es 1

(ONEP 1)	(ONEP 6)
T	NIL

ZEROP Devuelve T si un número es cero. NIL en cualquier otro caso.

(ZEROP 0)	(ZEROP -28)
T	NIL

Como podemos observar en todas las expresiones anteriores. su terminación es con la letra "P", dicha letra nos indica que se trata de un "predicado". El lenguaje de programación LISP utiliza los predicados para determinar las características de los argumentos que se someten a evaluaciones específicas.

Hablando en términos del LISP. todas las funciones anteriores son estándares del lenguaje. sin considerar los dialectos del mismo (INTERLISP. COMMONLISP. FRANZLISP. etc.)

5.1.1 Algunas funciones aritméticas

Debemos tener en mente que este tipo de funciones. en algunos casos dependen en gran medida del entorno. o dialecto que se este manejando del lenguaje.

EXP Representa la función del exponencial en base e del argumento. si este último no es un número se marca un error.

(EXP 0)	(EXP 3)
1	20.09

INT Esta función es utilizada para redondear el valor del argumento a un entero. cuando se trata de números enteros con parte decimal. Si el argumento es un número real INT no hace nada; quedando en mismo valor.

(INT 0.79)	(INT 26.15)
1	26

SQR Es la función típica utilizada para la obtención de la raíz cuadrada de un argumento dado. Cuando se introduzca un argumento no numérico se señala error.

(SQR 403)	(SQR cd)
20.075	ERROR

DIFERENCE Acepta dos argumentos, realizando posteriormente la resta del primero de ellos menos el segundo.

(DIFERENCE 16 20)	(DIFERENCE 13 W)
- 4	ERROR

DIVIDE Función aritmética la cual efectúa la operación de división, por lo que acepta dos argumentos, de los cuales el segundo debe ser diferente de cero.

(DIVIDE 17 2)	(DIVIDE 25 (MINUS 8))
8.50	- 3.125

LN Como es sabido, esta función devuelve el logaritmo del argumento, por lo cual dicho argumento debe ser un número positivo de lo contrario marca un error.

(LN 19)	(LN -21)
2.944	ERROR

RAMAINDER Función que devuelve el residuo de una división al proporcionar dos argumentos, el segundo debe ser diferente de cero.

(REMAINDER 28573 9)	(REMAINDER 14 3)
7	2

TIMES Esta función acepta un número ilimitado de argumentos del tipo numérico realizando posteriormente el producto de ellos, cuando esta función no recibe ningún argumento proporciona un 1 y en el caso de recibir un argumento no numérico marca error.

(TIMES 21 9 (MINUS 6))	(TIMES 4 36 j)
- 1134	ERROR

MAX Al igual que la anterior esta función recibe *n* argumentos, de los cuales MAX devuelve el argumento de mayor valor.

(MAX 0.9 j 1.0)	(MAX -9 -2 -5)
ERROR	-2

MIN Esta función es muy similar a la anterior, pero como su nombre nos indica, devuelve de una lista de argumentos el de menor valor.

(MIN 0.01 0.1 0.3)	(MIN 149 783 264)
0.01	149

PLUS Esta función también puede aceptar varios argumentos, dando como resultado la suma de todos ellos. Al igual que algunas otras funciones marca error cuando no se le proporciona un valor numérico.

(PLUS 168 46)	(PLUS 6.3 4.5 9.2)
214	20.0

En lo que se refiere a las tres funciones trigonométricas más utilizadas, es conveniente señalar que "este grupo de funciones no tiene un comportamiento estandar en los entornos LISP"[1],los ejemplos mostrados a continuación son validos para el dialecto UPCLISP.

SIN Representa a la función trigonométrica conocida como el seno, el valor del argumento debe estar en radianes, proporcionando después el valor de la función seno, también en radianes.

Sen 0 = 0.00000E+00	Sen 2 = 9.092974E-01

COS Función conocida como el coseno, recibe un argumento numérico en radianes, devolviendo posteriormente el valor del coseno de su argumento.

Cos 1 = 5.403023E-01	Cos 0 = 1.00000E+00

TAN Es la llamada función tangente, la cual devuelve como resultado la tangente del argumento que se le ha proporcionado.

Tan -2.5 = 7.470223E-01	Tan 3.158 = 8.4075445E-03

[1] U. Cortés, C. Sierra, "LISP" p.76

5.2 EL LENGUAJE PROLOG

Pasando ahora al PROLOG, el otro lenguaje de programación utilizado en I.A., diremos que la concepción de éste lenguaje se le debe a Alian Colmerauer en Marsella, Francia; en los primeros años de la década de los 70's, Colmerauer intentaba desarrollar un lenguaje en el que se lograra una programación enfocada hacia problemas del tipo analítico, por lo cual se vio en la necesidad de utilizar lo que se conoce como lógica de primer orden, tratando a esta lógica como cláusulas con reglas de inferencia apropiadas, le dió como resultado el lenguaje de programación PROLOG, que es la abreviatura de PROgramming LOGic (programación Lógica).

Para el desarrollo de un programa en PROLOG, se deben definir los predicados, los cuales consisten en afirmaciones acerca de un objeto. Como ejemplo de un predicado tenemos el siguiente:

vende (libros, felipe)

La interpretación a esta expresión sería: Felipe vende libros. Como se observa, para interpretar las sentencias, solo hay que relacionar de manera lógica todos los componentes que intervienen en el enunciado. Bajo esta estructura, los enunciados en PROLOG siempre tendrán un valor ya sea, "verdadero" o "falso". Es posible que se observe un grave error de ortografía en el nombre del sujeto ya que se trata de un nombre propio, pero tengamos presente que en este lenguaje de programación los nombres escritos con letras mayúsculas son tomados como variables, a ésto se hará referencia más adelante.

Al igual que el LISP, el PROLOG está más enfocado al procesamiento simbólico que al cálculo aritmético de funciones matemáticas. Para la programación en PROLOG, primero se tienen que establecer claramente los predicados o hechos acerca de los objetos del tema a tratar, los cuales van a formar parte de la base de conocimiento, esto es posible gracias a que PROLOG es un lenguaje que permite representar el conocimiento en forma de reglas de producción; luego se deben definir ciertas reglas para la correcta utilización y relación entre los datos previamente indicados. Lo anterior forma parte de la base del programa. PROLOG echará mano de todos estos conocimientos para dar respuesta a las preguntas que posteriormente se le harán alrededor del tema. Todo esto es en pocas palabras lo que se tiene que llevar a cabo para la elaboración de un sistema en PROLOG.

Con la finalidad de conocer cómo se declaran los hechos y las reglas dentro del ambiente de este lenguaje, partamos de la definición de hechos; en PROLOG un hecho no es otra cosa más que indicar las cláusulas acerca de un objeto o sujeto y las relaciones que existen en estos elementos, recordando el ejemplo dado al principio "vende (libros, felipe)", la primera parte de esta expresión es llamada predicado y es el que

marca la relación entre los componentes que se encuentran entre paréntesis; estos últimos elementos son nombrados argumentos. También se pueden definir expresiones con un solo argumento, por ejemplo:

frío (día) grande (casa) etc.

Casi siempre este tipo de sentencias se utilizan para asignar calificativos al objeto referido. Es transcendente destacar algunos puntos esenciales para elaborar sentencias en PROLOG:

- Todo nombre, tanto de los atributos, como el de la relación que hay entre éstos, debe comenzar con letra minúscula.

- Al escribir expresiones, primero se debe comenzar con la relación (predicado) que hay entre los objetos; después, se escriben los objetos (argumentos) éstos últimos deben ir encerrados entre paréntesis y separados por comas.

- Cada sentencia puede tener más de un argumento, como el mostrado a continuación:

compiten (karate, david, hugo)

Como ya se ha mencionado. Al conjunto de cláusulas se le da el nombre de conocimiento. Para lograr una base confiable debemos ser cuidadosos en el sentido de proporcionar los hechos suficientes y de manera adecuada para que éstos puedan ser bien manipulados por los mecanismos de inferencia, cuando el sistema requiera hacer uso de tal base.

A manera de muestra, consideremos los siguientes hechos que dan información acerca de ciertas comidas nutritivas:

comida (nutritiva, pescado)
comida (nutritiva, pollo)
comida (nutritiva, verduras)
comida (no_nutritiva, pastas)
comida (nutritiva, bistek)
comida (no_nutritiva, harinas)

Una vez que se tienen los predicados bien estructurados en la base de conocimientos, podemos realizar preguntas; éstas son muy parecidas a los hechos, por ejemplo: con la información que se tiene arriba podemos hacer preguntas, como las que a continuación se plantean:

comida (nutritiva, harinas) ó
comida (nutritiva, pollo)

Estas interrogaciones se interpretan más o menos como: "¿las harinas son comida nutritiva?" y "¿el pollo es comida nutritiva?" a lo que el sistema responderá en el primer caso "falso" y en el segundo "verdadero" (false, true).

En el lenguaje de programación PROLOG se pueden emplear variables, éstas solo son identificadas como tales por el lenguaje, cuando su nombre comienza con letra mayúscula, también podemos hacer preguntas que contengan variables, por ejemplo si al sistema se le hace la pregunta de qué comida es nutritiva, pero que dicha comida no contenga carne; considerando de antemano que el sistema cuenta con la información de qué comidas son elaboradas a base de carne y cuáles no lo son; el sistema podrá responder sin ninguna dificultad a preguntas tales como :

comida (nutritiva, X), comida (no_carne, X)

A estas interrogaciones el sistema tendría primero que satisfacer uno por uno los objetivos (goal) que se le presentan; lógicamente intenta satisfacer el primero, buscando para ello una comida la cual sea nutritiva, según lo que se le ha dado de información, las comidas resultan ser el pescado, el pollo, las verduras y el bistek; cualesquiera de ellas satisfacen el primer objetivo, pero al sustituir alguna de estas comidas en el lugar de la variable "X", ésta pasa ahora a ser constante; lo que en términos propios del lenguaje se le conoce como *instanciación*, al instanciar una variable para cumplir con este objetivo, también se realiza la sustitución en los objetivos que contengan la misma variable, con lo que el pescado, el pollo y el bistek cumplen el primer objetivo pero no satisfacen el segundo; por lo tanto, el sistema sigue buscando en su base de conocimientos, hasta encontrar otra comida que al instanciarse con la variable cumpla satisfactoriamente con los dos objetivos. En los casos en los que el sistema no puede satisfacer un objetivo, se regresa a la última instancia y comienza de nuevo el proceso; es decir, toma otro posible elemento y verifica si éste cumple con lo pedido y así sucesivamente, para llevar a cabo ésto, el sistema utiliza lo que conocemos como encadenamiento hacia atrás al igual que una búsqueda en profundidad.

Aparte de tener un basto conocimiento almacenado se deben establecer algunas reglas para una mejor utilización y manipulación de los hechos; éstas reglas se estructuran de la forma SI - ENTONCES, por lo que las reglas suelen utilizarse como definiciones más específicas en torno del objeto, por ejemplo siguiendo con la tónica culinaria expresada en los hechos anteriores, algunas reglas podrían ser las que a continuación se plantean:

Comencemos por definir en términos del castellano la regla que deseamos vaciar en PROLOG. La llamaremos regla R1

SI la mojarra es una comida de pescado
y el pescado es comida nutritiva,
ENTONCES la mojarra es una comida nutritiva

Luego expresada bajo la sintaxis del PROLOG tomaría más o menos la siguiente forma:

es_comida (nutritiva, mojarra) if
es_comida (pescado, mojarra) and
es_comida (nutritiva, pescado)

Ahora R1 traducida al lenguaje común al que estamos acostumbrados se lee de la siguiente manera: "la mojarra es comida nutritiva SI la mojarra es comida de pescado y el pescado es comida nutritiva", aunque acomodado de otra manera se expresa lo mismo que en R1, así como se formuló R1 se puede elaborar una abundante información en torno a las comidas tratadas.

Refiriéndonos a las constantes de este lenguaje de programación, diremos que aquellas son los nombres establecidos de los objetos y de las relaciones que entre ellos existen, dentro de esta clasificación figuran los también los números, dichos números son enteros y normalmente se utilizan para el desarrollo de operaciones aritméticas esenciales como son: +, -, *, /, aunque también suelen utilizarse dentro de los hechos, por ejemplo para indicar el valor de cada platillo en la base a la que nos hemos estado refiriendo, sería algo muy similar a lo siguiente:

precio (48, bistek)
precio (35, pollo)

Este lenguaje de programación permite definir operaciones aritméticas para calcular varias cosas, por ejemplo para el área de un rectángulo se puede utilizar la siguiente regla:

area_rectangulo (R, A) if
base (R, L1) and
altura (R, L2)
A = L1 * L2

a esta regla la podemos interpretar como:

A es el área del rectángulo R SI:

L1 es la base de R

y

L2 es la altura de R

El área se obtiene multiplicando L1 por L2.

En PROLOG, como en los otros lenguajes de programación es necesario especificar los valores de las variables independientes con el fin de conocer el valor de la variable dependiente. En este caso para asignarle a "A" el valor del resultado de la multiplicación, se deben conocer los valores de todas las variables que se involucren en la operación, en nuestro ejemplo se deben conocer L1 (base) y L2 (altura) para realizar la operación y otorgar un valor concreto al área "A".

5.3 ¿QUE LENGUAJE ES MEJOR, LISP O PROLOG?

Al concluir con la breve semblanza de estos dos lenguajes se presenta la pregunta obligada: ¿qué lenguaje utilizar para desarrollar un sistema experto LISP ó PROLOG?. Ocurre algo similar cuando debemos elegir una de dos cosas que se nos presentan y ambas nos ofrecen aspectos muy semejantes, ¿cuál utilizar?, tengamos en cuenta que el LISP es casi 10 años más antiguo que PROLOG, de ahí que LISP lo aventaje en tener máquinas especialmente desarrolladas para su uso. PROLOG utiliza relativamente menos espacio de memoria con respecto al LISP; además, su capacidad de representar y manejar la información lo hacen atractivo para crear S. E., aunque el LISP posee una gran potencialidad y flexibilidad; prueba de ello es la siguiente nota:

"LISP es un lenguaje de construcción de sistemas, un lenguaje de implementación, mientras que PROLOG es esencialmente un lenguaje de nivel usuario, un lenguaje de aplicación.

Es por lo que un sistema PROLOG se podría escribir en LISP mientras que la inversa es difícilmente abordable."[2]

Podemos seguir comparando a estos lenguajes de programación, para hacer una elección; sin embargo, en ocasiones imagina una combinación de ambos lo que "permite a los usuarios de LISP disfrutar de las posibilidades de PROLOG sin salirse del entorno LISP y a la recíproca."[3] Como resultado de la combinación del LISP con el PROLOG se tiene a LISLOG, siendo en esencia una mezcla de las particularidades de cada lenguaje como son: átomos, listas, cláusulas, reglas etc., gracias a esta combinación es posible usar a LISP o a

[2] Chatain Jean-Noël, "Sistemas expertos métodos y herramientas", p.148
[3] Ibidem p.183

PROLOG dentro de la programación hecha en LISLOG, cabe mencionar que la sintaxis de LISLOG es como la del LISP.

5.4 OTROS LENGUAJES

Con lo comentado hasta aquí podemos conocer un poco la estructura de los dos lenguajes utilizados en la aplicación más comercial de la I.A.; aunque dichos lenguajes sean los preferidos por los especialistas en la materia no significa que sean los únicos, ya que en la actualidad existen otros lenguajes que están teniendo un avance importante en esta área, también se cuenta con herramientas que nos ayudan a crear un S. E. sin necesidad de ser talentosos programadores; estas herramientas conocidas como *shells*, están compuestas por todos los bloques ó módulos de un S. E., pero sin la información, es por ello, que como ya lo habíamos mencionado, son sistemas vacíos, solo hay que estructurar la base de conocimientos.

Los lenguajes de programación orientados a objetos son algunos de los que se están utilizando para la programación simbólica, en estos lenguajes, un objeto es una especialización de la clase general a la cual pertenece, merced a ello el objeto en particular hereda las características de la clase de la que fue extraído, aunque la característica que hace más atractivos a los lenguajes orientados a objetos para el desarrollo de S. E. es que su " sistema de control es fácilmente convertible en un motor de inferencia; por otra parte el conocimiento se puede representar en forma híbrida marco-regla de producción."[4]

Actualmente existen muchos lenguajes basados en la programación orientada a objetos, de los cuales uno de los más importantes es SMALLTALK, posiblemente se le deba a su parecido con el LISP en el aspecto de las listas así como en el cumplimiento de la característica básica de los lenguajes especializados para la I.A.: el manejo de procesos simbólicos. Desde luego este tipo de lenguajes tiene sus desventajas; de entre ellas la que podría afectar más para los fines de los S. E. es su poca legibilidad, agregándose a ello "una cierta falta de eficiencia en la implementación debido al uso en forma masiva de gestión dinámica de memoria."[5]

Otros lenguajes desarrollados bajo la programación orientada a objetos son los mostrados enseguida de los cuales se debe hacer mención porque destacan de alguna manera de entre todos los existentes.

KOOL. (Knowledbe Objet Oriented Language), fue creado en la década de los 80's, esta orientado a objetos, permite representar el conocimiento en forma de reglas de producción y marcos; además cuenta con un

[4] J. P. Sánchez y Beltrán, "Sistemas expertos: una metodología de programación", p. 110
[5] J. Mompín Poblet, "Inteligencia artificial concepto, técnicas y aplicaciones", p. 214

motor de inferencia con encadenamiento mixto, matarreglas y una interfase especialmente para LISP entre otras cosas. KOOL esta escrito en el lenguaje LISP.

LOOPS. También orientado a objetos, su forma de representar el conocimiento es a base de reglas y marcos, funciona en INTERLISP-D de XEROX. Al igual que el anterior está escrito en LISP.

LRO2. Está escrito en LISP, pero orientado a objetos por lo que es considerado como una extensión de LISP, lo mismo que el XLISP aunque éste último esta escrito en lenguaje C.

OOGMS. Este lenguaje es útil en la creación de sistemas gráficos de ahí su nombre: Object Oriented Graphical Modeling System; fue desarrollado en los años 80's en lenguaje C. Existen muchos más lenguajes orientados a objetos, pero la mayoría de ellos tienen características muy similares a los presentados aquí.

5.5 HERRAMIENTAS

Pasando nuevamente a los *shells* diremos que éstos cuentan también con ventajas y desventajas en su utilización; de las primeras, podemos decir que ofrecen una relativa rapidez y facilidad en la creación de S. E., son simples de manejar ya que no requieren grandes conocimientos informáticos, en cuanto a las desventajas que se les atribuyen es su hermetismo y poca flexibilidad, esto se debe a que son sistemas casi en su totalidad elaborados. Existe una abundante gama de *shells* comerciales de gran diversidad, a continuación se exhibe un listado de algunos de estos sistemas.

ADVISOR. Es un sistema que esta basado en reglas de producción. Tiene un sistema de búsqueda orientado por sus objetivos y esta escrito en el lenguaje de programación PASCAL.

ARITY EXPERT SYSTEM. Este sistema permite representar el conocimiento como reglas de producción y marcos, tiene entre sus características, búsqueda en profundidad y encadenamiento hacia atrás, escrito en PROLOG.

ART. Fue creado en los 80's, admite la representación del conocimiento en forma de marcos y reglas, utiliza la búsqueda exhaustiva y encadenamiento mixto, es útil en la medicina y en las finanzas, esta escrito en el dialecto COMMONLISP de LISP.

CRIQUET. Permite la representación del conocimiento por medio de reglas de producción, utiliza encadenamiento hacia atrás y hacia adelante, escrito bajo LE_LISP.

ESP ADVISOR. Este sistema representa el conocimiento mediante reglas de producción y marcos, usa encadenamiento hacia atrás y hacia adelante, búsqueda exhaustiva en profundidad, esta desarrollado con el lenguaje de programación PROLOG.

EXPERT EASE. Fue desarrollado a principios de los 80's, acepta representación del conocimiento por medio de reglas de producción, utiliza encadenamiento hacia atrás, tiene la capacidad de generar reglas por medio de ejemplos, además de que es útil para desarrollos de sistemas de diagnostico médico, esta escrito en PASCAL.

EXSYS. Utiliza marcos y reglas de producción para representar el conocimiento, admite el encadenamiento hacia atrás y hacia adelante, EXSYS justifica sus deducciones ó resultados por medio de explicaciones, posee una interface de comunicación con dBIII y otras bases de datos, utiliza la búsqueda en profundidad y encadenamiento hacia atrás y hacia adelante, esta desarrollado en lenguaje C.

GOLEM. Permite la representación del conocimiento por medio de reglas de producción, usa el encadenamiento hacia adelante y hacia atrás, también posee la capacidad de explicar sus conclusiones y esta escrito en el lenguaje de programación PASCAL.

INSIGHT2. Usa las reglas de producción como representación del conocimiento, utiliza la búsqueda en amplitud y modos de encadenamiento hacia atrás y hacia adelante, contiene interfases de comunicación con algunas bases de datos como dBIII, esta escrito en PASCAL (TURBO).

KEE. Este sistema representa el conocimiento por medio de marcos, reglas y redes semánticas, usa encadenamiento hacia adelante y hacia atrás, búsqueda en profundidad y anchura, contiene un editor de gráficos, ha sido utilizado para el desarrollo de sistemas satelitales, KEE está escrito en el dialecto INTERLISP de LISP.

KES. Representa el conocimiento mediante reglas y marcos, su modo de encadenamiento es hacia atrás, hacia adelante y mixto, utiliza búsquedas heurística y probabilística, posee un editor de textos, esta desarrollado en lenguaje C.

M1. Utiliza las reglas como representación del conocimiento, permite la búsqueda en profundidad, encadenamiento hacia atrás y hacia adelante, puede dar explicaciones de sus resultados además de crear preguntas en forma automática, M1 está escrito en lenguaje C.

PERSONAL CONSULTAN. Fue desarrollado en los años 80's, acepta las reglas de producción y los marcos para representar el conocimiento, usa encadenamiento hacia atrás y hacia adelante, también contiene una interfase de comunicación con algunas bases de datos y está escrito en lenguaje LISP.

RULE MASTER. Permite la representación del conocimiento por medio de reglas, encadenamiento hacia adelante y hacia atrás, una característica importante de RULE MASTER es que posee un "creador de reglas" por medio del cual desarrolla reglas de producción en base a los ejemplos que le son proporcionados, además puede interactuar con algunas bases de datos, esta desarrollado en lenguaje C.

S1. En este sistema la forma de representación del conocimiento es por medio de reglas de producción y marcos, usa encadenamiento hacia adelante y hacia atrás y búsqueda exhaustiva en profundidad, proporciona explicación de sus conclusiones, S1 esta desarrollado para usarse en máquinas LISP, está escrito en GLISP.

TIGRE-1. Admite la representación del conocimiento como reglas de producción, usa el encadenamiento hacia atrás y una búsqueda exhaustiva, permite agregar funciones del LISP, esta escrito en LE_LISP.

VP-EXPERT Esta herramienta posee un motor de inferencia, el cual utiliza una búsqueda hacia adelante y hacia atrás; cuenta también con la habilidad de convertir información de diferentes fuentes como lo son LOTUS 1-2-3 o dBASE, archivos de base de datos e incluso archivos de texto en código ASCII., para poderla manipular, además Puede tomar información de una base de datos o un archivo de trabajo para crear una base de conocimientos. Una de sus principales cualidades es que genera automáticamente preguntas relacionadas con el tema. Vp-expert permite representar el conocimiento de la forma:

"IF	Conocimiento = Traducible
THEN	Sujeto = Compatible"

Se podría pensar en que las aplicaciones de este sistema vacío son limitadas al área científica o profesional pero no es así también puede ser usado en labores domesticas como lo son la jardinería, el cuidado de los niños, etc.

Después de conocer un breve esquema de los lenguajes de programación idóneos para la I. A. y de las herramientas que nos son útiles para crear nuestros propios sistemas, se nos presenta otra pregunta importante ¿qué utilizar, un lenguaje ó una herramienta y cuál?, para tomar una decisión hay que tener presentes algunos aspectos que son relevantes como pueden ser: qué lenguaje se adapta más al tipo de sistema que se desea elaborar y por otro lado, si se pretende utilizar una herramienta para su desarrollo, hay que verificar que ésta realmente tenga las características necesarias para que nos permita elaborar nuestro sistema sin mayores problemas. Asimismo, debemos considerar otros puntos que también pueden influir, en el tiempo que nos llevaría el desarrollo del sistema así como en el costo.

5.6 UN EJEMPLO DE APLICACION DEL PROLOG

Para finalizar con este capítulo se presenta un ejercicio a resolver por medio de un programa muy simple, elaborado en PROLOG. Se trata de identificar la relación familiar que existe entre algunos individuos. Primeramente presentamos en la figura 5.1 un diagrama a bloques en el cual se muestra un pequeño árbol genealógico, con el fin de identificar más rápidamente la relación que se guarda en dicha familia.

Al correr el programa éste pide el objetivo a cumplir (Goal:) aquí se le pide al programa que identifique el parentesco que existe entre los individuos, por ejemplo: tio(raúl,marcos) o madre(julio,barbara), a lo que el sistema responderá si es cierto (TRUE) o en su defecto falso (FALSE), según sea el caso. También es posible manejarlos como variables; es decir, de la forma tio(raúl,X) dando como resultado X = marcos, para el caso particular de los abuelos el programa está diseñado para mostrar en la salida, tanto el materno como el paterno, ésto es que la variable X de abuelo(raúl,X), toma dos valores: jorge y rodrigo, éste mismo caso es observado para las abuelas, abuemat y abuepat; en los dos últimos casos se refiere a los abuelos maternos y paternos respectivamente.

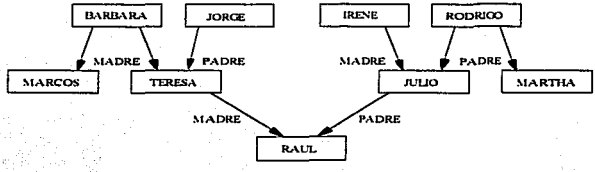


FIGURA 5.1

```
/*
Programa que ilustra la relación familiar de la figura 5.1
elaborado en turbo prolog 1.1
*/

domains
    persona = symbol

predicates
    hombre (persona)
    mujer (persona)
```

```
    padre (persona)
    madre (persona)
    hermana (persona)
    padres (persona)
    hermano (persona)
    tio (persona)
    abuelo (persona)
    tia (persona)
    abuela (persona)
    abuemat (persona)
    abuepat (persona)

clauses
    hombre (raul).
    hombre (julio).
    hombre (rodrigo).
    hombre (jorge).
    hombre (marcos).
    mujer (martha).
    mujer (irene).
    mujer (teresa).
    mujer (barbara).
    madre (raul, teresa).
    madre (marcos, barbara).
    madre (julio, irene).
    madre (teresa, barbara).
    padre (julio, rodrigo).
    padre (teresa, jorge).
    padre (raul, julio).
    padre (martha, rodrigo).

    padres (A, B) if madre (A, B).
    padres (A, B) if padre (A, B).
```

```
hermano (A, B) if
    hombre (B) and
    padres (A, P) and
    padres (B, P) and
    A < > B.
hermana (A, B) if
    mujer (B) and
    padres (A, P) and
    padres (B, P) and
    A < > B.

tia (A, T) if
    padre (A,P) and
    hermana (P, T).
tio (A, T) if
    madre (A,P) and
    hermano (P, T).

abuela (A, L) if
    madre (A, P) and
    madre (P, L).
abuela (A, L) if
    madre (P, L) and
    padre (A, P).

abuelo (A, L) if
    padre (P, L) and
    madre (A, P).
abuelo (A, L) if
    padre (A, P) and
    padre (P, L).

abuemat (A, L) if
    madre (A, P) and
    madre (P, L).
```

```
abuemat (A, L) if
    madre (A, P) and
    padre (P, L).

abuepat (A, L) if
    padre (A, P) and
    madre (P, L).
abuepat (A, L) if
    padre (A, P) and
    padre (P, L).
```

Al ejecutar el programa fuente, se obtienen respuesta a eventos como los siguientes:

```
Goal:  madre(raul,teresa)
TRUE

Goal: hermano(julio,marcos)
FALSE

Goal: padre(raul,X)
X=julio

Goal: abuelo(raul,X)
X=jorge
X=rodrigo

Goal: hermana(teresa,X)
No solution

Goal: abuepat(raul,X)
X=irene
X=rodrigo
```

CAPITULO VI

UTILIZACION DE LA INTELIGENCIA ARTIFICIAL PARA LA SOLUCION DE PROBLEMAS

Con lo que se ha visto hasta estos momentos, se tiene una visión general de los rasgos más importantes de la I. A.; no así de una aplicación concreta de esta materia, por lo que en este capítulo implementaremos un sistema basándonos en la tecnología de S.E. y demostraremos cuanto se facilita el trabajo en la elaboración del mismo al utilizar el Vp-expert, esta herramienta presenta ventajas interesantes (como las vistas en el capítulo 5) que lo hacen atractivo para tal fin.

El problema a solucionar consiste en sugerir un restaurante, según lo que el usuario desea; como es el tipo de comida que prefiera, el presupuesto con el que cuenta y el lugar por donde vive; entre otros. Teniendo bien definidas las entradas y salidas del sistema éste se estructuro de la siguiente forma: comienza por pedir la zona en la que vive la persona y si cuenta con automóvil, si la respuesta a esta última pregunta es negativa; solamente se considerara la zona proporcionada, en caso contrario contempla tanto la zona introducida como las zonas aledañas, después de esto el programa despliega algunos nombres de platillos de los cuales el usuario tendrá que elegir los de su preferencia; así se detecta el tipo de comida requerida; posteriormente pide el estilo del restaurante deseado, el cual puede ser formal o informal esto hace referencia al tipo de ropa que el establecimiento acepta, también es importante el presupuesto con el que se cuenta, el número de personas que irán y por último si tienen contemplado tomar vino; estas tres últimas preguntas sirven para obtener un gasto real por persona. Con toda esta información, el sistema elige de una base de datos todos aquellos restaurantes que cumplan con los requisitos de entrada y los muestra en la pantalla.

Es conveniente mencionar que la base de datos se elaboró en dBase III, en ella se recopilo toda la información suficiente para cada restaurante, como la zona en la que se ubica, el nombre del restaurante, el domicilio, la especialidad (tipo de comida), el estilo, (formal, informal) el costo promedio por persona y algunas observaciones generales; (ver tabla A) en este punto resalta la importancia de la información contenida en la base, ya que mientras más información contenga ésta (número de restaurantes), será más amplia la gama de establecimientos que cumplan con las características de entrada, lo que significa que la probabilidad de encontrar un restaurante acorde a los gustos y presupuesto del usuario tiende a aumentar. En este contexto, consideramos conveniente aclarar que la información del costo promedio por persona de cada restaurante es ficticia debido a la dificultad de establecer un costo real, es por ello que para ejemplificar una variedad en la base de datos se recurrió a la falacia en la mayor parte de la información.

ZONA	RESTAURAN	DOMICILIO	COSTO_PROM	ESPECIAL	ESTILO	OBSERV
CENTRO	BOCANA	AV INSURGENTES No 774, COL. DEL VALLE	25 00	ARGENTINA	INFORMAL	ABRIMOS DE LUNES A SABADO DE 8 00 A 1 00 HRS, MUSICA AL ESTILO ARGENTINO, ACEPTAMOS TARJETAS DE CREDITO
CENTRO	LANCERS	AV INSURGENTES No 2018, COL FLORIDA	46 00	ESPANOLA	INFORMAL	ABIERTO DIARIAMENTE DE 12 30 A 20 00 HRS, ACEPTAMOS TARJETAS DE CREDITO, CONTAMOS CON ESTACIONAMIENTO, MUSICA VIVA
CENTRO	DARUMA	PORFIRIO DIAZ No 534, COL NOCHE BUENA	60 00	JAPONESA	FORMAL	ABIERTO DIARIAMENTE DE 13 00 A 21 30, CONTAMOS CON ESTACIONAMIENTO, ACEPTAMOS TARJETAS DE CREDITO
NORTE	LA CUMBRE	PUEBLA No 213, COL ROMA	45 00	ARGENTINA	INFORMAL	ABIERTO DIARIAMENTE DE 13 00 A 19 00 HRS
NORTE	CANTAMAR	LONDRES No 279 COL JUAREZ	35 00	DEL_MAR	INFORMAL	ABIERTO DIARIAMENTE DE 11 00 A 22 00 HRS, ACEPTAMOS TARJETAS DE CREDITO
NORTE	HONFLEUR	AMBERES No 14-A, COL JUAREZ	51 00	FRANCESA	FORMAL	ABIERTO DE LUNES A SABADO DE 13 00 A 1 00 HRS, ALTA COCINA FRANCESA, ACEPTAMOS TARJETAS DE CREDITO, CONTAMOS CON ESTACIONAMIENTO
ORIENTE	HEVIA	LUIS MOYA No 130, COL CENTRO	75 00	ARGENTINA	FORMAL	ABIERTO DIARIAMENTE DE 7 00 A 20 00 HRS, CONTAMOS CON ESTACIONAMIENTO, ACEPTAMOS TARJETAS DE CREDITO, RESERVACIONES AL 521-90-50
ORIENTE	SHANGHAI	DOLORES No 30, COL CENTRO	75 00	JAPONESA	FORMAL	ABRIMOS DIARIAMENTE DE 12 00 A 23 00 HRS, ACEPTAMOS TARJETAS DE CREDITO
ORIENTE	EL MORO	SAN JUAN DELETRAN No 42, COL CENTRO	41 00	MEXICANA	FORMAL	ABRIMOS DIARIAMENTE LAS 24 00 HRS, ACEPTAMOS TARJETAS DE CREDITO, CONTAMOS CON ESTACIONAMIENTO, MUSICA VIVA
PONIENTE	ARRECIFE	CAMPOS ELISEOS No 218, COL POLANCO	30 00	DEL_MAR	INFORMAL	ABIERTO DIARIAMENTE DE 13 00 A 1 00 HRS, AMBIENTE FAMILIAR
PONIENTE	LA CASA	GALILEO No 50, COL POLANCO	60 00	ESPANOLA	FORMAL	ABIERTO DE 13 00 A 20 00 HRS,ACEPTAMOS TARJETAS DE CREDITO, AMBIENTE FAMILIAR, CONTAMOS CON ESTACIONAMIENTO VIGILADO
PONIENTE	FINESSE	ANATOLE FRANCE No 98, COL POLANCO	51 00	FRANCESA	FORMAL	ABRIMOS DE LUNES A SABADO DE 13 30 A 22 00 HRS, ACEPTAMOS TARJETAS DE CREDITO, ESTACIONAMIENTO PROPIO,ESPECIALIDAD, LENGUADO VERONIQUE
SUR	RIOJA	AV INSURGENTES No 2390, COL SAN ANGEL	32 00	ESPANOLA	INFORMAL	ABRIMOS DIARIAMENTE DE 10 30 A 22 00 HRS, ACEPTAMOS TARJETAS DE CREDITO, CONTAMOS CON ESTACIONAMIENTO
SUR	IBARAKI	ALTAVISTA No 142, COL SAN ANGEL	70 00	JAPONESA	FORMAL	ABRIMOS DIARIAMENTE DE 13 00 A 12 00 HRS, ACEPTAMOS TARJETAS DE CREDITO, CONTAMOS CON AMPLIO ESTACIONAMIENTO
SUR	LOS IRABIEN	AV. DE LA PAZ No. 45, COL. SAN ANGEL	45 00	MEXICANA	FORMAL	ABRIMOS DIARIAMENTE DE 7 00 A 20 00 HRS, ACEPTAMOS TARJETAS DE CREDITO, CONTAMOS CON ESTACIONAMIENTO, ESPECIALIDAD TACOS DE TUETANO

TABLA A

Teniendo presente las características más importantes que se consideraron para la elaboración del sistema se presenta a continuación un listado del programa.

```
BKCOLOR = 0;
RUNTIME;
ENDOFF;
ACTIONS
COLOR = 15
DISPLAY      "      SUGERIMOS EL MEJOR LUGAR PARA COMER
                    DE ACUERDO A SUS GUSTOS Y PRESUPUESTO
                    <     PRESIONE ENTER PARA COMENZAR     >
_    "
CLS
   FIND RUMBO_ASIG
   CLS
   FIND COMIDA
   FIND TIPO
   CLS
   FIND PREUNIT
   FIND PRESU_ASIG
      WHILEKNOWN RUMBO
         RESET RUMBO
         POP RUMBO_ASIG, RUMBO
         RESET PLATILLO
         FIND PLATILLO
      END
      CLS
   DISPLAY  "   SESION    TERMANADA  ~ ";

RULE 1
IF
   RUMBO < > UNKNOWN
THEN
   PLATILLO = SI
   RESET COMIDA
   FIND COMIDA
      WHILEKNOWN TIPOCOMIDAS
         RESET TIPOCOMIDAS
         POP COMIDA, TIPOCOMIDAS
         RESET ESTABLECIMIENTO
```

```
        FIND ESTABLECIMIENTO
      END
      CLOSE BASEREST;

RULE 2
IF
   TIPOCOMIDAS < > UNKNOWN
THEN
   ESTABLECIMIENTO = SI
      WHILEKNOWN RESTAURAN
         GET RUMBO = ZONA AND TIPOCOMIDA = ESPECIAL AND TIPO = ESTILO AND
              PRESU_ASIG >= COSTO_PROM, BASEREST, ALL
         RESET MUESTRA
         FIND MUESTRA
      END
      CLOSE BASEREST;

RULE 3
IF
   RESTAURANT < > UNKNOWN
THEN
   MUESTRA = SI
   CLS
   DISPLAY  "
      EN LA ZONA {ZONA}
      ES RECOMENDABLE EL RESTAURANTE {RESTAURANT}
      SU ESPECIALIDA ES LA COMIDA {ESPECIAL}
      EL COSTO PROMEDIO APROXIMADO ES
      $ {COSTO_PROM} POR PERSONA
      {OBSERV}  -  ";


                        !    REGLAS PARA DETERMINAR LA ZONA    !

RULE 4
IF
   RUMBO_DADO = CENTRO OR
   RUMBO_DADO = NORTE OR
   RUMBO_DADO = SUR OR
   RUMBO_DADO = ORIENTE OR
```

```
    RUMBO_DADO = PONIENTE AND
                AUTO = NO
THEN
    RUMBO_ASIG = (RUMBO_DADO);

RULE 5
IF
    RUMBO_DADO = CENTRO AND
                AUTO = SI
THEN
    RUMBO_ASIG = PONIENTE
    RUMBO_ASIG = ORIENTE
    RUMBO_ASIG = SUR
    RUMBO_ASIG = NORTE
    RUMBO_ASIG = CENTRO;

RULE 6
IF
    RUMBO_DADO = NORTE AND
                AUTO = SI
THEN
    RUMBO_ASIG = PONIENTE
    RUMBO_ASIG = CENTRO
    RUMBO_ASIG = ORIENTE
    RUMBO_ASIG = NORTE;

RULE 7
IF

    RUMBO_DADO = ORIENTE AND
                AUTO = SI
THEN
    RUMBO_ASIG = NORTE
    RUMBO_ASIG = CENTRO
    RUMBO_ASIG = SUR
    RUMBO_ASIG = ORIENTE;

RULE 8
IF
    RUMBO_DADO = PONIENTE AND
```

```
                AUTO = SI
THEN
    RUMBO_ASIG = NORTE
    RUMBO_ASIG = CENTRO
    RUMBO_ASIG = SUR
    RUMBO_ASIG = PONIENTE;

RULE 9
IF
    RUMBO_DADO = SUR AND
                AUTO = SI
THEN
    RUMBO_ASIG = PONIENTE
    RUMBO_ASIG = CENTRO
    RUMBO_ASIG = ORIENTE
    RUMBO_ASIG = SUR;

                !   REGLAS PARA DETERMINAR EL TIPO DE COMIDA    !

RULE MEX
IF
    GUSTO = MOLE OR
    GUSTO = CONSOME OR
    GUSTO = TACOS
THEN
    COMIDA = MEXICANA;
RULE MAR
IF
    GUSTO = PESCADO OR
    GUSTO = CAMARON OR
    GUSTO = PULPO
THEN
    COMIDA = COMIDA_DEL_MAR;

RULE ITA
IF
    GUSTO = PIZZAS OR
    GUSTO = SPAGHETTI OR
    GUSTO = CARNES_SICILIANAS
THEN
```

```
    COMIDA = ITALIANA;

RULE JAP
IF
    GUSTO = VARIEDAD_ARROZ OR
    GUSTO = TEPANYAKY OR
    GUSTO = DURAMA
THEN
    COMIDA = JAPONESA;

RULE FRA
IF
    GUSTO = CAMAMBERT OR
    GUSTO = ENSALADAS_DULCES OR
    GUSTO = PASTAS_CON_QUESOS
THEN
    COMIDA = FRANCESA;

RULE ARG
IF
    GUSTO = CARNES_ARGENTINAS OR
    GUSTO = BIFE_CHORIZO OR
    GUSTO = BIFE_LOMO
THEN
    COMIDA = ARGENTINA;

RULE ESP
IF
GUSTO = PAELLAS OR
GUSTO = CORTES_CARNE OR
GUSTO = KOKOTXAS
THEN
    COMIDA = ESPAÑOLA;

                    !   REGLAS PARA DETERMINAR EL PRESUPUESTO   !

RULE 10
IF
PRESUPUESTO > 0.00 AND
NUMPER > 0
```

```
THEN
    PREUNIT = (PRESUPUESTO / NUMPER);

RULE 11
IF
VINO = NO
THEN
    PRESU_ASIG = (PREUNIT* 0.80)
ELSE
    PRESU_ASIG = (PREUNIT* 0.4 0.80);

                        !    SECCION DE PREGUNTAS AL USUARIO    !

ASK RUMBO_DADO:  " POR CUAL DE LAS SIGUIENTES ZONAS VIVE ? " ;
CHOICES RUMBO_DADO: CENTRO, NORTE, SUR, ORIENTE, PONIENTE;

ASK AUTO:  "TIENE AUTOMOVIL  ? " ;
CHOICES AUTO SI, NO;

ASK GUSTO:  "QUE TIPO DE COMIDA PREFIERE ? " ;
CHOICES GUSTO: MOLE, CONSOME, TACOS, PULPO, CAMARON, PESCADO, SPAGHETTI,
PIZZAS, CARNES_SICILIANAS, DURAMA, VARIEDAD_ARROZ, TEPANYAKY, CAMAMBERT,
ENSALADAS_DULCES, PASTAS_CON_QUESOS, CARNES_ARGENTINAS, BIFE_CHORIZO,
BIFE_LOMO, CORTES_CARNE, PAELLAS, KOKOTXAS;

ASK TIPO:  " QUE ESTILO DE RESTAURANTE PREFIERE ? ";
CHOICES: TIPO: FORMAL, INFORMAL;

ASK PRESUPUESTO:  " CUAL ES SU PRESUPUESTO ? " ;

ASK NUMPER:  " CUANTAS PERSONAS ACUDIRAN ? ";

ASK VINO:  " DESEA TOMAR VINO ? ";
CHOICES VINO: SI, NO;

PLURAL: RUMBO_ASIG, COMIDA, GUSTO;
```

Como se observa claramente en Vp-expert la estructura del programa se lleva a cabo por medio de reglas de producción de la forma SI - ENTONCES (IF - THEN).

Cuando se ejecuta Vp-expert, esto se logra con vpx, se muestra un menu con las siguientes opciones:

1.HELP 2.INDUCE 3.EDIT 4.CONSULT 5.TREE 6.FILENAME 7.PHAT 8.QUIT

1.HELP. Esta primera opción se utiliza para consultar la ayuda de Vp-expert.

2.INDUCE. Con la cual es posible crear bases de conocimiento a partiendo de bases de datos elaboradas en dBase o en hojas de calculo, mostrando un menú para elegir la opción adecuada

3.EDIT. Opción que permite editar los archivos con extensión KBS.

4.CONSULT. Es usada para hacer consultas al programa. En esta opción se tiene un menú como el siguiente:

1.HELP 2.GO 3.WHATIF 4.VARIABLE 5.RULE 6.SET 7.QUIT

1.HELP. Contiene la información acerca de la ayuda.

2.GO. Es usado para consultar el programa.

3.WHATIF. (Qué pasa si) con ella es posible cambiar el valor de una variable.

4.VARIABLE. Esta muestra el valor final de las variables.

5.RULE. Muestra las reglas de la base de conocimiento.

6.SET. Esta instrucción fija algunos parámetros por default. En este punto se puede observar un menú como el siguiente:

1.HELP 2.TRACE 3.SLOW 4.FAST 5.QUIT

1.HELP. Es para obtener ayuda.

2.TRACE. Con esta opción la consulta puede ser mostrada en forma gráfica o de texto.

3.SLOW. Se encarga de hacer más lenta la acción de las reglas.

4.FAST. Regresa a la velocidad normal.

5.QUIT. Regresa al menú de CONSULT.

7.QUIT. Regresa al menú principal.

5.TREE. Muestra un árbol de búsqueda, en él se observa cómo se llego a la solución, permite elegir que esta demostración pueda hacerse en forma gráfica o en forma de texto en su menú:

1.HELP 2.TEXT 3.GRAPHICS 4.QUIT

1. HELP Se utiliza para obtener ayuda.
2. TEXT Despliega el texto en forma de árbol mostrando el camino que el motor de inferencia sigue.
3. GRAPHICS Crea una representación grafica de la consulta.
4. Es la salida del submenú.

6.FILENAME. Es usada para cargar un nuevo archivo con extensión KBS.
7.PATH. Esta opción permite elegir o cambiar PATH.
8.QUIT. Utilizada para abandonar el sistema.

Conociendo algunas generalidades para utilizar la herramienta, a continuación se explica brevemente los tres bloques que son necesarios para estructurar un sistema en Vp-expert estos son:

1.- El bloque de acciones. En este primer bloque se determina claramente objetivo y además se dan las instrucciones ordenadas; las cuales conducirán a la solución. En este bloque debe incluirse la palabra reservada ACTIONS ya que es la que contiene las instrucciones que indican al motor de inferencia las variables que debe encontrar; la palabra reservada FIND sirve para identificar estas variables cuyos valores son requeridos, en el programa mostrado se utilizan también en este bloque cláusulas como DISPLAY, la cual se recurre a ella para mostrar un mensaje. BKCOLOR utilizada para la presentación al darle un color diferente al fondo de la pantalla, dicho color depende del número dado. RUNTIME elimina los dos botones de las ventanas en la base de conocimiento durante una consulta al Vp-expert. COLOR es usado para cambiar el color del texto. GET esta cláusula sirve para obtener información de un archivo de base de batos, en este caso previamente creada en dBase III, también se ocupa la cláusula POP usada para manipular los valores por separado de las variables que pueden tomar más de un valor (conocidas como variables purales), el funcionamiento de esta cláusula es como el de una pila; y por último RESET sirve para remover cualquier valor asignado a la variable nombrada y lo regresa a UNKNOWN; UNKNOWN es utilizado en el bloque de reglas e identifica variables cuyos valores no son conocidos.

2.- El bloque de reglas. Como su nombre lo indica, en esta sección se indican las reglas estas se componen de 4 elementos principales: nombre, premisa, conclusión y un punto y coma al final de cada regla. El primer elemento contiene la palabra reservada RULE posteriormente se debe indicar el nombre de la regla, la premisa debe comenzar con la palabra IF seguida de sus condiciones. (Vp-expert acepta hasta 10 condiciones), en la conclusión de la regla se emplea la palabra THEN aunque en algunos casos es necesario emplear la palabra reservada ELSE. En las reglas también se pueden utilizar los operadores relacionales: =, <, >, <=, >=, <> y los operadores lógicos AND y OR; En resumen las reglas se estructuran dentro de Vp-expert de la siguiente manera:

```
RULE <NOMBRE>
IF <CONDICION 1>   AND / OR
   <CONDICION 2>   AND / OR
   <CONDICION 3>   AND / OR
        .
        .
        .
```

THEN <CONCLUSION 1>
<CONCLUSION 2>
.
.
.

ELSE <CONCLUSION 3>
.
.
.

En éstas últimas también se utiliza la palabra CLOSE la cual permite resetear lo primero que fue grabado por el apuntador en el archivo.

3.- El bloque de enunciados. Este último bloque contiene la información necesaria para la consulta de la base de conocimientos. En la mayoría de los casos Vp-expert otorga características especiales a las variables contenidas en la base de conocimiento. Por ejemplo: ASK, CHOICES y PLURAL son los enunciados que se ocuparon en la elaboración del programa anteriormente mostrado; la primera de ellas reconoce a todas aquellas variables que su valor será proporcionado por el usuario al llevar a cabo la consulta, el segundo enunciado se utiliza para mostrar los posibles valores que puede tomar una variable y PLURAL se usa cuando se requiere que le sean asignados más de un valor a una variable durante la consulta.

Para finalizar este capítulo a continuación se muestra la salida que ofrece el sistema al dar las siguientes respuestas a la pregunta que elabora:

zona = norte
automóvil = si
gustos = consome y pastas con quesos
estilo = informal
presupuesto = 120
personas = 2
vino = no

Como se puede observar, el sistema considera las zonas: norte, oriente, centro y poniente; esto es por lo establecido en la regla 6 la cual indica que si el rumbo es la zona norte y se cuenta con automóvil, entonces se tienen que contemplar las zonas mencionadas. Después de que el usuario marca sus gustos, para este ejemplo el sistema defina que se trata de comida mexicana y francesa respectivamente, al obedecer a las reglas MEX y FRA, también establece el estilo del restaurante que se prefiere y para finalizar obtiene un presupuesto por persona ésto lo logra haciendo uso de las reglas 10 y 11; de acuerdo a los datos de entrada que proporciona el

usuario se determina un gasto disponible de $48 por persona, es por ello que el programa ofrece como salida únicamente aquellos establecimientos que no rebasen dicho costo, los cuales son:

En la zona norte:

Café de Paris, y Galería para la comida francesa, El Mirador, Guadiana 19 y Jardín de Prendes son los restaurantes seleccionados para la comida mexicana.

En la zona oriente

Para la comida francesa El Danibio, Les Moustaches y Normandie. Y para la mexicana El Cardenal, La Nueva Opera y Lincoln.

En la zona centro

Los elegidos son Bavaria, Choice y La Casserole para la comida francesa y Antonio's, Café México y hoyo para la mexicana.

En la zona poniente

El Buen Comer, Fouquet's de Paris y Tandoor para la comida francesa y Azulejos, Café del Bosque y Casa de Campo para la comida mexicana.

CAPITULO VII

APLICACIONES Y PERSPECTIVAS DE LA INTELIGENCIA ARTIFICIAL

En el último capítulo del trabajo, trataremos de sintetizar algunas de las aplicaciones y perspectivas de esta rama de la computación, la I. A. como lo indicamos al principio (capítulo I) se divide en áreas tales como: la robotica, el procesamiento natural del lenguaje, etc., las cuales han tenido una relevancia importante cada una de ellas en las áreas que han sido utilizadas. Como resultado de los trabajos de investigación realizados en cada área, consideramos útil conocer cómo se aplican las tecnologías de la I. A., en dónde se ocupan y cuáles son las posibles tendencias de esta materia.

La I. A. tiene grandes retos en los que actualmente viene trabajando. Uno de ellos y quizás el más importante es lograr desarrollar un Hardware que tolere los requerimientos de trabajo de la I.A. Para ello se ha trabajado en técnicas avanzadas de integración a escala muy grande; ésta consiste en fusionar en un único chip el mayor número de componentes. Hasta el momento se ha logrado almacenar más de 55 000 transistores y circuiteria de apoyo, en un espacio de aproximadamente 2 cm^2 y como se mencionó anteriormente se sigue estudiando la manera de optimizar aun más el espacio disponible.

A nivel de Software se investiga la manera de lograr un procesamiento paralelo (que atienda varias tareas a la vez) y con esto dejar atrás el procesamiento secuencial (realiza una tarea a la vez).

7.1 APLICACIONES Y PERSPECTIVAS DE LA I. A.

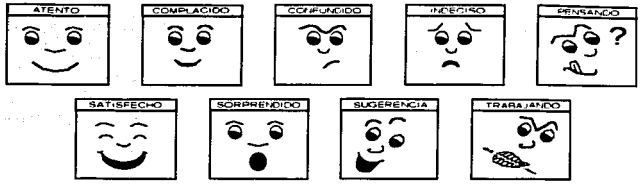
7.1.1 Agentes Inteligentes

También conocidos como agentes informáticos, están siendo desarrollados para asistir a los usuarios en diferentes tareas, desde ayudarlos a planear viajes hasta recomendar casas; considerando los gustos y necesidades personales de quien lo consulte.

Una característica importante de éstos es que no son un simple programa, así mismos se ven como un ente independiente que interactua con los humanos para establecer comunicación y saber sus necesidades, haciéndolas propias y tratando de resolverlas.

Estos agentes guiarán al usuario a través de complejos espacios cibernéticos en busca de una solución del problema que ahora es de ambos y podrán a la vez tomar decisiones que antes eran personales del usuario, así mismo lo podrán representar cuando éste lo desee; llegando a ocupar el puesto de secretario personal.

Se pretende que estos agentes inteligentes sean manipulados a través del habla o de expresiones faciales a las que responderán de igual manera mostrando en la pantalla sus estados de ánimo.



Los atributos que requiere un sistema para ser llamado agente inteligente son:

INTEGRACION:	Que el entendimiento con el usuario sea optimo.
EXPRESIVO:	Debe aceptar requerimientos de diferentes especialidades
META ORIENTADA:	Debe determinar cómo y cuándo llevar a cabo una meta
COOPERATIVO:	La colaboración con el usuario debe de ser al 100 %.
DAR GUSTO AL CLIENTE:	El agente debe adaptarse a los diferentes usuarios

Sin embargo los problemas a los que se enfrentan los diseñadores de estos sistemas son:

- Falta de vocabulario colaborativo
- Sobrecarga de información
- Precisión en la tarea de síntesis
- Falta de confianza

en los cuales ya se esta trabajando.

Los agentes inteligentes se comportan de la siguiente manera:

1.- Se crean agentes informáticos que saldrán a la búsqueda de la información solicitada.
2.- El agente que llegue con datos satisfactorios se dividirá en dos y saldrá de nuevo a la búsqueda.
3.- El agente que no regrese con documentos útiles se elimina.

Así se crea una población de agentes útiles al usuario.

En la actualidad solo el proyecto CYC desarrolla este tipo de sistemas de una manera relevante. La corporación CyCorp de Agustin ha desarrollado un proyecto llamado CYC, utilizando los agentes informáticos. Este sistema consta del conocimiento previo que se necesita para entender todo aquello que las personas al hablar ya dan por hecho de que todos conocen; también tiene la capacidad de compartir su conocimiento directamente con otros sistemas (cualidad que no tienen hasta ahora los S. E.). En él, se han incorporado hechos tan comunes como la manera en que se toma la sopa, también creencias generalizadoras y observaciones

comunes, inclusive datos contradictorios se tienen almacenados en su base de conocimientos, por ejemplo "CYC sabe que Drácula era un vampiro pero al mismo tiempo sabe también que los vampiros no existen"[11] . Puede manipular también metáforas y analogías para la solución de un problema al valerse de los contextos de ficción con los que cuenta. Este sistema aunque no ha alcanzado su desarrollo total, ha logrado avances significativos que le permiten ser considerado como una de las perspectivas más ambiciosas de la I.A. en el desarrollo de una base de conocimientos compartida y capaz de actualizarse por sí sola.

7.1.2 Alcance del Tiempo Real por la I. A.

Esta investigación se está desarrollando por dos tipos de investigadores, los primeros son los de la I. A. y los segundos los del tiempo real, los cuales conjuntan sus esfuerzos para lograr que un sistema responda lo suficientemente rápido, así como lo haría cualquier humano ante la situación que se le presente.

Los sistemas basados bajo este concepto tendrán cabida en el control de las plantas de poder nuclear, las naves aéreas, los vehículos, etc., en donde no sólo es necesario escoger las acciones apropiadas en diferentes situaciones, sino también que esas acciones sean dadas en tiempos óptimos que permitan al sistema estar dentro de los dominios del tiempo real.

La NASA esta desarrollando un sistema de este tipo para operar una nave espacial a la velocidad de la luz tomando en cuenta el ambiente incierto al que se enfrenta (como hoyos negros, trampas difíciles de detectar por los sensores, etc.), obstrucciones y terrenos riesgosos, a los que debe reaccionar en un tiempo real para prevenir peligros impredecibles en su ruta de operación.

7.1.3 Algoritmos Genéticos

En el curso del desarrollo de las tecnologías de la I. A. los científicos para facilitar su trabajo y aprovechar más adecuadamente los recursos con los que cuentan (principalmente software y hardware), hacen uso de las técnicas básicas de la materia, como la representación del conocimiento, utilizada muy comúnmente en el diseño de S. E.; al igual que el aprendizaje automático y las búsquedas inteligentes, entre otras que están teniendo un lugar aceptable, un ejemplo claro son los algoritmos genéticos.

Uno de los objetivos principales al utilizar los algoritmos genéticos es pretenden hacer que la máquina tenga la capacidad de solucionar un problema sin tener que decirle cómo se debe de proceder para encontrar la respuesta.

Los algoritmos genéticos se basan en la teoría de la evolución de Darwin (la selección natural). Primero se crean diversos programas generados aleatoriamente, que se encuentran dentro del dominio de un problema

[11] Douglas B. Lenat, "Investigación y Ciencia.", Nov., 1995, p. 26.

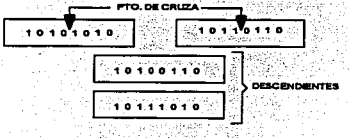
(creándose un espacio de programas). A continuación se les plantea un problema, cuando estos programas dan la solución se clasifican según el grado de exactitud de la solución que presentan. Al contrario de lo que se podría pensar, algunos programas de la población evolucionan hasta generar soluciones satisfactorias. Los programas con mayor grado de exactitud se seleccionan para hacer una cruza con ellos y así lograr la reproducción.

Es aquí donde los sobrevivientes intercambian material cromozómico que está compuesto por genes: cada gen consta de un conjunto de valores (0-1). La información de los genes pasara a un programa nuevo que constará al final de dos partes. Esta operación se usa para crear 2 programas-hijo, a partir de 2 programas-padre, asegurando así lo mejor de éstos.

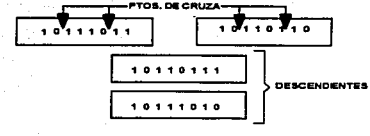
Los descendientes pasarán a formar parte de la población de la siguiente generación, que estará constituida por los individuos más aptos y más adaptables a los cambios que se producen en su entorno.

Las dos formas más comunes de reproducción son:

a) Uso de un punto único de cruza.



b) Uso de dos puntos de cruza.



Una vez realizada la selección y reproducción, los programas que no fueron seleccionados se eliminan, así como los programas-padre quedando sólo la población de hijos.

Después de todo este procedimiento, el sistema cuenta con un universo de programas, que pueden contener hasta cientos de miles de programas de computadora que están listos para satisfacer las diferentes necesidades del usuario, además de ser estructuras activas capaces de ejecutarse en su forma actual.

Algunas aplicaciones

- Un algoritmo que aprende a controlar el flujo del gas de una red ficticia (modelada de una real), tomando en cuenta la presión, la cantidad de gas que se quiere mandar a cierta parte y el costo de la misma.

- La creación a través de Algoritmos Genéticos de una turbina politápica de alta velocidad (utilizada por los motores de aviones comerciales). El tiempo de ahorro en este tipo de proyectos es considerable; un ingeniero tarda hasta 8 semanas en construir un modelo satisfactorio, mientras que el sistema de Algoritmos Genéticos produce uno con el doble de mejoras en menos de un día.

- Producir diferentes arquitecturas para arneses electrónicos de automóviles. Los Arneses constan de varios elementos como cables, conectores, fusibles, etc. y su función es transmitir corriente a los diferentes dispositivos eléctricos del automóvil. El arnés se debe configurar dependiendo de las características del carro para proporcionar la corriente necesaria y el voltaje suficiente a cada uno de los dispositivos eléctricos.

7.1.4 Procesamiento del Lenguaje Natural

El procesamiento del lenguaje natural quizá sea la rama de la I. A. más atractiva e interesante, ya que como lo habíamos mencionado anteriormente, su finalidad es la de lograr que la computadora comprenda el lenguaje natural de las personas, para conseguir con ello una comunicación más abierta entre las computadoras y los seres humanos; al hacer posible esta comunicación, las instrucciones que se dan actualmente a las computadoras pasarían a segundo plano o hasta desaparecerían, debido a que la máquina debe entender instrucciones habladas o escritas, por ejemplo: "formatea el disco del drive A", como lo señala al respecto Herbert Childt ". . . evitaría la programación normal y el protocolo del sistema operativo. . ."[2] Para alcanzar este ambicioso objetivo es indispensable un riguroso trabajo por parte de los investigadores, ya que no es una tarea fácil de desarrollar porque las computadoras necesitan mayor precisión en la comunicación que las personas. El arte de reconocer las palabras va más haya de solo escucharlas, implica también comprenderlas.

Se está desarrollando un sistema que tolere el error en el lenguaje (puntuaciones incorrectas, frases incompletas, etc.), la imprecisión en este (errores de deletreo, palabras saltadas, modismos, etc.), que determine si existe un ruido ambiental captado por el micrófono, corrija la redacción de la persona que está dictando, su gramática y elimine las repeticiones, aprendiendo de los ejemplos y reaccionando en un tiempo real; que la máquina entienda lo que se está diciendo, aun está por encima de lo que estos sistemas pueden hacer.

La tecnología de procesamiento del lenguaje natural es utilizada en varios programas para computadoras con un aceptable manejo tanto del léxico como de la gramática del lenguaje (principalmente del lenguaje inglés), la comunicación que se ha tenido se ha logrado gracias a la construcción de interfaces conocidas como Interfaces de Lenguaje Natural. Aunque no se ha desarrollado un programa el cual entienda realmente todo lo que se dice, debido a que como lo señala Harry Tennant "es mucho más fácil construir un sistema que pueda hacer cálculo matemático que uno que entienda el lenguaje. La razón estriba en que el entendimiento del

[2] Herbert Childt, op. cit., p. 99

lenguaje natural requiere una enorme cantidad de conocimiento: conocimiento acerca del lenguaje y de la comunicación en general, pero fundamentalmente conocimiento del mundo." Sin embargo se tiene como dato una aplicación del procesamiento del lenguaje natural; ésta es utilizada en los programas traductores que existen en el mercado, algunos de estos traductores son usados en las compañías telefónicas (como el SPHINX III) que ofrecen el servicio de llamadas de larga distancia, ya que los traductores permiten que el usuario no solo hable, sino que también escuche en su lengua nativa, ésto en cualquier parte del mundo.

Estos sistemas de traducción, deberán ser capaces de reconocer el lenguaje y traducirlo en un 100%, algunas compañías telefónicas ya lo usan pero su nivel de traducción es considerada de un 90%. Cuando se alcance este 100%, tendrán cabida no solo en la telefonía, sino también se usarán para traducir películas y tener acceso a multi-idiomas automatizados, entre otros.

El desarrollo de el procesamiento del lenguaje natural es el que hasta ahora ha marcado la pauta en esta aplicación, aun así, se han estado creando elementos propios como el JONUS, primer sistema de traducción del lenguaje a principios de los años '90, un homólogo de éste es el ATR desarrollado en Japón y el SIEMENS de Alemania. Después de estos llegaron el VERMOVIL de Alemania, el JONUS I (con 500 palabras en su vocabulario) y el JONUS II que permite la manipulación de diálogos humanos conversacionales espontaneos (cuenta con más de 3,000 palabras) en los siguientes idiomas:

- Aleman
- Coreano
- Español
- Ingles
- Japones

El sistema de asistencia "Bell Atlantics" (campana del Atlántico) se encarga de los sistemas de grabación y remueve de ellos las llamadas pausas, reduciendo así la duración de las llamadas en un 8.3%, logrando un ahorro en el material ocupado para grabarlas.

7.1.5 Programa de Cálculo Estratégico DARPA

El objetivo a alcanzar por este sistema es desarrollar máquinas inteligentes aplicables a aspectos bélicos (seguridad nacional) y usar los avances logrados para acelerar el desarrollo económico.

Las áreas que se pretenden desarrollar son:

- Las técnicas avanzadas para la integración a escala muy grande (VLSI) y otras tecnologías microelectrónicas.

- Retroalimentar sus logros con los de algunas áreas de I. A., como lo son la visión por ordenador, el reconocimiento del habla, el lenguaje natural y los S. E.
- Desarrollo de multiprocesadores y procesadores de señal para mejorar la arquitectura de los ordenadores.

En concreto las aplicaciones militares que DARPA pretende mejorar son tres: el ayudante de piloto, los sistemas autónomos y el sistema de gestión de batallas.

I) AYUDANTE DE PILOTO

Un sistema que ayuda al piloto al proporcionarle la información que éste requiere, rápida y eficazmente así como tomar algunas decisiones. Este sistema es de suma importancia para el apoyo del manejo de aviones militares que contienen un gran número de interruptores que confunden fácilmente al piloto.

El último objetivo de este tipo de sistema es el lograr que el ayudante de piloto se convierta en realidad en piloto al controlar el avión por sí solo atendiendo todas las tareas y características de ejecución, así como lograr el aterrizaje en plataformas marinas.

Otra forma en la que se ha tratado este problema es por medio de las señales neuronales que pretenden enseñar al piloto la manera de manipular las ondas cerebrales llamadas "potencial evocado del cerebro" (EP), producidas por estímulos tales como ruidos fuertes o destellos de luz, y a través de éstas controlar ciertas funciones del avión aun teniendo pies y manos ocupadas en el pilotaje.

Este método actualmente se esta tratando en la Base Aérea Wright-Patterson en Dayton por el grupo de Grant MacMillan.

II) SISTEMAS AUTONOMOS

Los Sistemas Autónomos son aquellos que no necesitan de intervención humana para trabajar; tal es el caso de automóviles, e incluso algunos misiles entre muchos otros.

La realización de este tipo de sistemas a nivel militar es obvio, por ejemplo en una misión se pueden enviar aviones, misiles y tanques autónomos, para evitar poner en peligro las vidas del personal militar.

III) GESTION DE BATALLAS

La toma de decisiones en condiciones de incertidumbre será el campo en el que se desenvuelva el Sistema de Gestión de Batallas, el cual asesorará a los militares (u otros usuarios), tomando en cuenta tanto los activos (personal) como los recursos de artillería (vehículos, aviones, municiones, etc.) y los datos que se tenga (preciso e imprecisos). Este sistema se valdrá del reconocimiento del lenguaje natural y de los S. E. para mantener una comunicación más precisa y así apoyar en todas las fases de la toma de decisiones.

7.1.6 Programación Automática

La otra tecnología de la I. A. la cual posee un interesante campo de investigación es la programación automática, ya que con el perfeccionamiento de esta tecnología, se reduciría en gran escala el tiempo que se invierte para el desarrollo y funcionamiento de los programas; debido a que en un sistema basado en la programación automática debe ser capaz de crear otro programa el cual produzca las variaciones o cambios necesarios según sean los requerimientos que vayan surgiendo. Hasta ahora no se cuenta con un sistema de programación automática de propósito general, no obstante se están realizando importantes investigaciones en torno a esta rama de la I. A. en los principales centros de investigación de Estados Unidos; como indicio de esto se tiene el sistema SAFE, dicho sistema "construye una especificación formal del programa a partir de una especificación informal..."[4] como éste, existen algunos otros sistemas en fase de prueba que son desarrollados en instituciones, sin tener todavía aplicaciones potenciales.

Otras aplicaciones de esta rama se encuentran en:

I) SISTEMA DE APRENDIZAJE

Un sistema que cuenta con aprendizaje automático debe de poder mejorar su comportamiento en base a su propia experiencia, es decir, dependiendo de todo lo que sus sensores obtengan del medio ambiente que lo rodee. Se pretende que los sistemas que cuenten con la capacidad de aprendizaje, a través de la observación puedan: a) determinar el problema, b) obtener técnicas de resolución y c) aprender la teoría. Ejemplo de lo anterior sería:

- Que el sistema "lea" por si solo un libro, lo entienda y conteste a las preguntas que se le realicen en torno al contenido de dicho libro.

- Al observar a una persona en su labor, aprenda la manera de realizar la misma actividad, esto implica no sólo sus movimientos, sino también su vocabulario.

II) SISTEMA DE REPLICACION POR SI SOLO

No es sólo el hecho de realizar una tarea predeterminada, sino que después de algún tiempo de hacerla, desarrollar diferentes mecanismos para optimizarla (ahorrando tiempo, energía, materias primas, etc.), y así lograr un avance significativo en la producción de la empresa.

Los sistemas de replicación por si solos cuentan con un controlador que les permite el monitoreo, diagnostico y manipulación de la maquinaria existente, para distribuirla de la mejor manera y obtener los resultados que se requieran (aumento en la producción hasta un 30%).

[4] Henry C. Mishkoff, op.cit., p.144

Un sistema de este tipo es el que controla una planta depuradora de agua obteniendo como resultado la optimización del tiempo de proceso.

7.1.7 Proyecto de la Quinta Generación

Los Japoneses pretenden a través de este proyecto:

a) Acelerar la velocidad de recuperación de datos, esto es muy importante considerando el número de datos que manejan los sistemas de I. A. El proyecto de la Quinta Generación ya ha dado como fruto la "Máquina de Base de Datos Relacional" propia para el almacenamiento y recuperación de datos.

b) Desarrollar un ordenador con un motor de inferencia más eficaz que estando terminado permita realizar programas para apoyar otras áreas del proyecto. Ya se ha logrado crear un prototipo llamado "Máquina Personal de Inferencia Secuencial", escrito en PROLOG.

c) Desarrollar en los ordenadores las facultades de "ver", "escuchar" y "hablar", para facilitar aún más su manejo, por medio de la visión por ordenador, el reconocimiento del habla y el procesamiento del lenguaje natural, y por último.

d) Lograr que la computadora cree programas por sí sola valiéndose de la llamada Programación Inteligente.

El conjunto de todas estas características dará como resultado la computadora de la Quinta Generación.

7.1.8 Realidad Virtual

La Realidad Virtual (R. V.) permite valiéndose de las diferentes áreas de la I. A. crear "mundos" semejantes al real, por lo que la R. V. no implica solo la representación de objetos, también debe de permitir una inmersión, interacción y navegación del usuario a través del sistema.

La inmersión se refiere al contacto pleno con el mundo virtual por medio de visiocascos, guantes de datos y sobre todo de la inmersión mental propia de cada uno de nosotros (es el pensar seriamente que se está en un mundo real).

La interacción permite manipular los objetos del mundo virtual a nuestro antojo obteniendo los cambios que propiciamos en un tiempo real.

La navegación proporciona el medio por el cual el usuario puede viajar en el universo virtual a diferentes lugares, en diferentes épocas; es decir, "Los mundos virtuales no se reducen a mundos de imágenes. En ellos se experimentan también sensaciones táctiles, auditivas, etc."[5]

Los sistemas de R. V. manejan los Agentes Inteligentes como personajes virtuales (clones) que guían a las personas a través del mismo, según sus requerimientos.

Actualmente ya existen diferentes dispositivos que varían en precio y calidad para las personas interesadas en la R. V., como:

a) Visiocascos. Se están fabricando actualmente por varias compañías. Permiten aislar al usuario del mundo real al ocupar la totalidad de su campo visual, dar a la imagen el acercamiento o alejamiento requerido dependiendo de la prioridad del mismo y por último una interacción del mundo virtual con los movimientos de la cabeza.

b) Guante de datos. Permite la comprensión de los movimientos de la mano por el sistema. Existen guantes que contienen fibras ópticas pero estos son extremadamente delicados y después de flexionarse continuamente se quiebran. Otro material usado es el Nilón que también ha dado buenos resultados.

c) Toolkits. Son paquetes que permiten entre otras cosas, importar imágenes de CAO (como Autocad), modelar los objetos para darles una apariencia más real, darles brillo y sombra, manipular los principales periféricos utilizados por la R. V. (visiocascos, y guantes de datos), etc.

Todos estos usan imágenes, sonidos, movimientos, gestos y el sentido del tacto, propios de la comunicación entre personas, para una fácil manipulación.

También existen ya (aun no comerciales) los sensores "biosensoriales", son pequeñas pastillas que se colocan en la piel para mandar información de los movimientos de todas las partes del cuerpo.

La principal limitación de la R. V. ha sido la velocidad de los ordenadores disponibles en el mercado, sin embargo a partir de 1996 y con la aparición de los microprocesadores 486 y Pentium, se ha logrado superar esta barrera haciendo posible la R. V. en un tiempo real, aunque ciertamente un poco restringida (permiten mostrar escenas un tanto sencillas y una resolución de objetos sino optima, si aceptable).

[5] Mundo Científico, Philippe Quéau, No. 148, Vol. 14, p.610.

Las aplicaciones de este campo son infinitas, al parecer no existe algo que no se pueda representar o apoyar en los sistemas de R. V., y solo para ejemplificar se mencionaran algunos:

- La robótica podría ser utilizada para manipular robots en ambientes peligrosos o zonas alejadas e incluso inaccesibles como lo son: en incendios (utilizados como rescatistas), en las misiones previstas a Marte (o a cualquier otro lugar del universo), el ingresar a un cráter volcánico o a fondos marinos, hacer tareas en centrales nucleares, etc. y cuando se requiera desconectar al robot utilizar un simulador para ver las implicaciones de las acciones que realizamos.

- En la medicina apoyo a cirujanos en sus diferentes especialidades y en la formación de nuevos médicos. Pensemos en una operación realizada por un doctor el cual esta siendo asesorado por otro especialista a miles de kilómetros de distancia.

- En las comunicaciones permite que las personas que se comuniquen puedan ingresar a la R. V. y encontrarse cara a cara en el espacio virtual de su preferencia para continuar con su charla.

- El videojuego puede convertirse en un verdadero reto al combatir frente a frente con nuestros héroes o villanos favoritos.

- En lo cultural podríamos viajar a los museos que contarían con diversas salas en donde estarán las ciudades, templos, etc., de las diferentes culturas de todos los tiempos, todo esto al alcance de la mano.

En el festival Imagina '92 se logró el encuentro de dos personas separadas por miles de kilómetros en un ambiente virtual que simulaba el monasterio de CLUNY (actualmente en ruinas).

7.1.9 Robotica

Otra tecnología que está teniendo una sobresaliente utilización, principalmente en la industria es la robotica; esta tecnología se ha venido utilizando como auxiliar en los procesos un tanto repetitivos o bien en aquellos que se puedan desarrollar en serie, las secuelas dejadas por la robotica en las industrias donde es aplicada son muy benéficas para ésta última tanto en lo económico como en la productividad; al obtener garantías como las de contar con un aumento considerable de producción, elevación de la calidad del producto, costos bajos en la operación y/o mantenimiento, disminución de errores, etc. Cuando son implementadas algunas otras tecnologías con la robotica, es posible crear robots mucho más sofisticados; es decir, máquinas, las cuales sean capaces de "entender su entorno y tomar las acciones inteligentes apropiadas en respuesta a distintas

situaciones externas."[6] Los científicos en la materia están ocupándose arduamente en trabajos industriales en los que todo sea manipulado y controlado por robots prescindiendo de la ayuda humana.

Actualmente ya existen diferentes robots encaminados principalmente a la producción en serie haciendo tareas que no necesitan de un razonamiento más halla de una simple programación. Un robot se puede desarrollar de diferentes maneras, pero para ser considerado como inteligente debe de contar con la siguientes características:

ACCION.- Es la habilidad del robot para manipular los objetos que lo rodean y así cambiar su entorno.

PERCEPCION.- Es la capacidad de percatarse de los cambios en el entorno ya sea por el movimiento de un objeto o inclusive del mismo robot.

PROCESOS DE RAZONAMIENTO.- Cuando se le encomienda una tarea al robot, este debe de poder planificar sus acciones para llevarla a cabo.

Los robots inteligentes no son simplemente máquinas, son también entes capaces de manipular una gran información proporcionada por sus sensores y discriminar de ésta la que no les es útil para sus fines.

Los vehículos móviles autónomos también se consideran como una modalidad de robots. En 1986 apareció el primero de ellos llamado Navlab I, que contaba con sensores y diferentes computadoras, luego llego el Navlab II, ambulancia del ejército con alta movilidad y vehículo de multirrodada. El más reciente de esta generación es el Navlab V una camioneta comercial Van de General Motors que fue capaz de conducirse correctamente desde Washington hasta San Diego en el verano de 1995.

7.1.10 Salas Inteligentes

Las salas inteligentes han sido desarrolladas gracias a empresas o instituciones tales como el Instituto Tecnológico de Massachusetts (MIT), que permiten a través de sensores y cámaras dotar al ordenador de los sentidos de vista y oído no propios de los mismos.

En estas salas inteligentes se construyen ambientes cotidianos y comunes como el de una oficina, una casa o hasta el interior de un automóvil y son capaces de reconocer las expresiones de las personas, determinando quien es el individuo que se encuentra en la sala (si ya ha estado antes en ella), su estado de animo, la actividad que esta realizando y hasta lo que quiere que realice el sistema sin pedirlo de una manera directa.

[6] Henry C. Mishkoff, "A fondo: inteligencia artificial," p. 126

La primera sala inteligente del MIT fue construida en 1991 y para 1996 ya existen cinco más (3 en Boston, 1 en Japón y 1 en el Reino Unido) y en proyecto otras tres (en Paris, Nueva York y Dallas).

El último objetivo de las salas inteligentes es asesorar y ayudar a las personas a hacer sus labores diarias con el mínimo esfuerzo, sin embargo el estudio dentro de un área restringida siempre es un poco incómodo, por lo que también se han desarrollado "ropas inteligentes", que constan de cámaras, micrófonos y ordenadores integrados en el propio vestuario. Con unos simples anteojos en forma de visor y algunos micrófonos en forma de antenas receptoras se pueden obtener datos valiosísimos como los nombres de las personas con las que tratamos y datos de las actividades realizadas o por realizar.

Actualmente las salas inteligentes enfocan sus actividades en el desarrollo de proyectos tales como:

- Determinar las acciones que realiza un conductor y así lograr que el auto se maneje por sí solo.
- En el entendimiento del lenguaje de señas (sordomudos) para realizar las instrucciones de las personas que no pueden hablar aunque éstas no se encuentren frente a un computador.
- Elaborar "tarjetas de crédito capaces de reconocer a sus propietarios y saber de este modo si han sido robadas"[7].
- El proyecto de Ambiente de Video Interactivo de Vida Artificial (AVIVA), este pretende reconocer a la persona que esta dentro de la sala inteligente y utilizando su descripción corporal, realizar un modelo dentro del ambiente virtual que interactue con seres residentes de la misma y determine la manera en que estos entes estarán al servicio del usuario.

7.1.11 Señales Neuronales

Este proyecto implica un logro para las personas que sufren de alguna discapacidad corporal. Se pretende a través de las señales que emite nuestro propio cuerpo lograr el control de la computadora y un poco después el poder manipular aparatos de ayuda para aminorar el impacto personal de estas incapacidades.

Los sistemas que manejan señales neuronales no solo son un conjunto de sensores en la piel, también se requiere en estos una programación especial que permita decodificar estas señales y codificarles en lenguaje entendible para la máquina; para este efecto, actualmente ya se cuenta con el equipo llamado BIOMUSA.

[7] Investigación y Ciencia, Alex P. Pentland, Junio 1996, p 17.

Por su extensión se ha dividido la investigación de las señales neuronales en tres ramas:

a) Señales EMG (electromiograficas).- Que son impulsos eléctricos transferidos a través de los nervios y músculos del cuerpo. Aunque no se cuente con un miembro (como una pierna), el cerebro manda impulsos eléctricos que llevan implícita la información para que se mueva ese miembro; este sistema pretende decodificar esa información para manipularla en un sistema y lograr que este responda a las peticiones del cerebro.

b) Señales EOG (electrooculograficas).- Transmitidas por el ojo. Estas señales muestran la tensión producida por los ojos cuando cambian su orientación y se pretende que a través de ordenes ópticas el usuario que lo necesite o lo requiera pueda manipular el ordenador.

c) Señales EEG (electroencefalograma).- Transmitidas por el cerebro, estas se recolectan con unos electrodos fijados en el cuero cabelludo. Se han distinguido ya 5 diferentes ondas que son:

i.- ALFA.- Se producen cuando hay distracción.
ii.- BETA.- Se producen cuando hay actividad intensa.
iii.- THETA.- Se producen cuando hay tensiones emocionales.
iv.- DELTA.- Se producen cuando hay periodos de sueño profundo.
v.- MU.- Se producen cuando hay movimientos físicos o intención de moverse.

Aunque las señales EEG no han tenido un desarrollo tan grande como las señales EMG y EOG, ya se han logrado avances como los siguientes:

- La manipulación de un cursor por medio de ondas ALFA de un paciente inmovilizado por una enfermedad.

- La manipulación de un cursor por medio de ondas MU a través de actividades motoras como una sonrisa, masticación, deglución, etc.

7.1.12 Sistemas Expertos

Particularmente en el ámbito computacional las técnicas y tecnologías de la I. A. tienen aplicaciones en varias áreas del quehacer humano, que van desde la ayuda en la educación de los infantes y adolescentes, hasta el apoyo en los más ambiciosos proyectos de investigación en varias áreas, como prueba de ésto se tiene a los S. E. rama que probablemente fue la primera en utilizarse para fines tanto educativos, como de diagnostico medico, obteniendo un grado considerable de aceptación. El extenso campo de aplicación de los S.E. radica en que éstos pueden ser creados para cualquier finalidad y prácticamente en cualquier lenguaje de programación o incluso con la ayuda de alguna de las herramientas, como las mencionadas en el cap. III.

I) EL AJEDREZ

Si bien es cierto que ya existen varios sistemas capaces de jugar al ajedrez, contra los mejores expositores de esta actividad, también es cierto que no todo esta dicho por los sistemas diseñados para este objetivo.

El sistema FRITZ 4.0 (aun en etapa de prueba), esta considerado dentro de los S. E. que ya han alcanzado niveles superiores en el dominio del ajedrez y no se duda en pensar que su inteligencia es "natural". A simple vista podría parecer que el ajedrez es solo un juego que no debiera ser considerado como problema, sin embargo, éste no solo consta de movimientos, sino también de tácticas y estrategias que llevan implícito un razonamiento notable en un tiempo real.

En la actualidad (1997) el sistema DEEPBLUE de IBM logro anotar un buen punto para los sistemas de ajedrez, al ganarle al actual campeón mundial Gary Kasparov en una serie de 6 juegos, proeza hasta hoy no alcanzada por ningún otro sistema de este tipo.

II) RAZONAMIENTO BASADO EN CASOS

El razonamiento de los actuales S. E. está basado en reglas que permiten alcanzar la solución, utilizando diferentes tipos de búsqueda. Actualmente se esta desarrollando un método para utilizar ya no reglas, sino conocimientos de casos ya resueltos, que se utilizan para hacer analogías o asociaciones y así llegar a la solución del problema planteado.

Las ventajas más importantes de este método con respecto al de reglas es que si se tiene que incorporar un nuevo conocimiento a la "base de casos" (su similar de la base de conocimientos), esta incorporación se hace de manera automática, mientras que si fallara el de reglas, se tendrían que revisar las reglas hasta determinar cual es la que falla e incorporar otras nuevas cuidando de que no se repitan o no se contradigan con las ya existentes.

Los sistemas que están diseñados bajo este concepto también ofrecen la alternativa de explicar el porqué de la solución alcanzada de una manera natural y entendible.

Este razonamiento requiere de tres tipos de procesos los cuales son:

Matching.- Se encarga de establecer la similitud entre los diferentes casos que se tienen y el que se está procesando.

Función de Semejanza.- Como su nombre lo indica, determina el grado de similitud entre los casos.

Indexación.- Permite almacenar nuevos ejemplos al sistema de una manera adecuada que permita el acceso a cierta información del ejemplo cuando se requiera en otro ejemplo similar.

El camino por recorrer de estos sistemas aun es largo, no obstante, ya existen aplicaciones en el dominio legal HYPO.- genera argumentos tanto para la defensa como para el fiscal. En la resolución de disputas MEDIATOR.- señala compromisos y alternativas para las partes en conflicto. En la medicina CASEY.- que diagnostica los posibles estados internos y fallas del corazón tomando en cuenta los síntomas del paciente, etc.

7.1.13 Visión

En este campo se trabaja arduamente para dotar a la máquina con la capacidad de interpretar y reconocer imágenes de datos que posteriormente se conviertan en modelos de tercera dimensión que muestren escenas del mundo real. Esto no es tan fácil ya que las computadoras se ven desbordadas por muchos datos resultado en buena parte por la movilidad del mundo real.

Lo que se requiere para poder manipular la visión activa es tener una gran capacidad de memoria y un tiempo de procesamiento real.

A finales de los 70's, principios de los 80's se logró desarrollar un sistema que manejaba la visión por computadora, éste no alcanzo los objetivos finales que se tenían, pero a cambio se lograron grandes avances en:

- Los problemas ópticos inversos
- El color
- Las propiedades geométricas y
- El contraste.

La sombra de los objetos es otro punto importante a considerar. Se han desarrollado diferentes técnicas para solucionar este problema como la que consiste en la comparación de los elementos geométricos del modelo con otros similares en el mismo paisaje; está no solo permite la identificación de sombras sino también el reconocimiento de objetos parcialmente ocultos.

Los programas desarrollados dentro de la rama de la visión por computadora han tenido implicaciones muy eficaces, en muchas áreas, principalmente en la militar, ya que son utilizados para la identificación de objetos o blancos, también han tenido aplicaciones en la geología, como auxiliares en la creación o reconstrucción de mapas contando para ello solamente con imágenes algunas veces con lineamientos imperceptibles para el ojo humano (fotografías satelitales o muy antiguas). Esta tecnología acupada también como auxiliar en la reconstrucción tanto de retratos hablados, ya sea partiendo de bosquejos elaborados por los

dibujantes o para dar posibles cambios físicos en la filiación de las personas, como en la reconstrucción de rostros de individuos teniendo la osamenta del cráneo, de la misma forma es utilizada para la identificación de huellas digitales.

Todos estos sistemas en estado de desarrollo son una prueba más de que ninguna de las ramas en las que se divide la I. A. actúan por sí solas, la constante interacción entre ellas es la fuerza de esta gran ciencia.

La contribución a nivel económico es notable ya que se considera que gracias a los sistemas desarrollados por la I. A. en diferentes industrias han salvado en su conjunto cada año hasta 1 billón de dólares; y a nivel personal ha logrado que la gente tome conciencia de la importancia del conocimiento y de la dificultad de adquirirlo y representarlo.

CONCLUSIONES

La tecnología avanza a pasos agigantados y la I. A. como miembro activo de la ciencia de la computación, tiene aportaciones muy significativas que contribuyen notablemente a dicho avance en general y en particular ofrece abundantes campos de investigación.

En sus inicios la I. A. fue un tema que creó controversias y polémicas, aún ahora hay quienes no comparten el nombre que se le ha dado a esta ciencia; sin embargo, queda claro que los esfuerzos realizados por los científicos en el desarrollo y perfeccionamiento de las técnicas de la I. A. han tenido en su mayoría buenos resultados. Las repercusiones de esta ciencia a nivel técnico y económico ya se han puesto de manifiesto en diversas partes del mundo ahorrando millones de dólares y miles de horas hombre en el desarrollo de proyectos.

Así como la I. A. ha tenido grandes éxitos, también tiene enormes retos que afrontar, tal vez uno de los más interesantes sea el de hacer que las computadoras realmente puedan ser entes pensantes; con que se podrá lograr un importante avance en la implementación de las funciones y facultades intelectuales que sobreponen al hombre ante cualquier otro ser vivo. Es importante señalar que todo aquello que se obtenga como resultado de investigaciones, para el cumplimiento de nuevos objetivos debe utilizarse razonablemente y en beneficio del propio hombre.

Este escrito desea mostrar el principio mediante el cual trabaja la I. A. que siendo una ciencia tan extensa y relativamente nueva, aun no alcanza la madurez total, de igual forma se espera que el presente trabajo sea útil a los inscritos en las materias de inteligencia artificial, sistemas expertos o alguna otra vinculada con esta ciencia, o bien sirva a aquellas personas interesadas en el mundo de la inteligencia artificial.

BIBLIOGRAFIA

1.- Adarraga Pablo, Zaccagnini José.
Psicologia e Inteligencia Artificial.
Ed. Trotta.

2.- Angulo Usategui José Ma., Del Moral B. Anselmo.
Guia fácil de la Inteligencia Artificial.
Ed. Paraninfo.

3.- Aubert J. P., Schombery R.
Inteligencia Artificial.
Ed. Paraninfo.

4.- Berk A. A.
LISP el Lenguaje de la Intelegencia Artificial.
Ed. Anaya Multimedia.

5.- Castillo Enrique, Alvarez Elena.
Sistemas Expertos: Aprendizaje e incertidumbre.
Ed. Paraninfo.

6.- Chatain Jean-Noel, Dussauchoy Alain.
Sistemas Expertos métodos y herramientas.
Ed. Paraninfo.

7.- Clocksin W. F., Mellish.
Programación en Prolog.
Ed. Gustavo Gili, S. A. Colección Ciencia Informática.

8.- Cortes Ulises, Sierra Carlos.
LISP.
Ed. Marcombo Boixareu.

9.- Cuena José, Fernandez Gregorio.
Inteligencia Artificial: Sistemas Expertos.
Ed. Alianza Informatica.

10.- Fu K.S., González R.C., Lee C.S.G.
Robotica: control, detección, visión e inteligencia.
Ed. McGraw-Hill.

11.- Frenzel Louis E Jr.
A fondo: Sistemas Expertos.
Ed. Anaya Multimedia.

12.- Frost Richard.
Bases de Datos y sistemas expertos ingeniería del conocimiento.
Ed. Díaz de Santos S. A

13.- Gaverter B. William.
Intelligent Machines: an introductory perspective of artificial intelligence and robotics.
Ed. Prentice-Hall.

14.- Giannesini Francis.
Prolog.
Ed. Addison-Wesley Iberoamericana.

15.- Graubard R. Stephen.
El nuevo debate sobre la inteligencia artificial, sistemas simbólicos y redes neuronales.
Ed. Gedisa.

16.- Hartnell Tim.
Inteligencia artificial conceptos y programas.
Ed. Anaya Multimedia.

17.- Jackson C Philip, Jr.
Introduction to A. I.
Ed. Dover publications, Inc New York.

18.- Meisel William S.
Computer-Oriented appraches to pattern recognition, Vol. 83 in mathematics in science, and engineering.
Edited by Richerd B. University of Southern California.

19.- Mishkoff C. Henry
A Fondo: Inteligencia Artificial.
Ed. Anaya Multimedia.

20.- Mompin Poblet José.
Inteligencia Artificial, conceptos, técnicas y aplicaciones.
Ed. Marcombo Boixaren.

21.- Nebendahl Dieter.
Sistemas Expertos introducción a la técnica y aplicación.
Ed. Marcombo S. A.

22.- Nillsson Nils J.
Principios de Inteligencia Artificial.
Ed. Diaz de Santos S. A.

23.- Queinnec C.
Programación en LISP.
Ed. Paraninfo.

24.- Rauch-Hindin Wendy.
Artificial Intelligence in business, science and industry.
Ed. Prentice-Hall.

25.- Rich Elaine, Knight Kevin.
Inteligencia Artificial.
Ed. McGraw-Hill.

26.- Sánchez y Beltrán José Pablo.
Sistemas Expertos una metodología de programación.
Ed. Macrobit.

27.- Schildt Herbert.
Utilización de C en I. A.
Ed. McGraw-Hill

28.- Tou Julius T., González Rafael C.
Pattern Recognition Principles.
Ed Addison-Wesley

29.- Winston Patrick Henry.
Inteligencia Artificial.
Ed. Addison-Wesley Iberoamericana

OTRAS OBRAS CONSULTADAS

1.- Computer: Innovative Technology For Computer Professionals
Editor in chef: Edward A. Parrish

June 1994	January 1995	July 1995	July 1996	August 1996	October 1996
Vol. 27	Vol. 28	Vol. 28	Vol. 29	Vol. 29	Vol. 29
No. 6	No. 1	No. 7	No. 7	No. 8	No. 10

2.- Investigación y Ciencia
Dir. Gral. Francisco García Guillen

Febrero 1992	Septiembre 1992	Noviembre 1995	Junio 1996	Diciembre 1996
No 185	No. 192	No. 230	No. 237	No. 243

3.- Mundo Cienfífico
Dir. Gral. Dominique Chouchan
Núm. 148.
Vol. 14

4.- Soluciones Avanzadas
Dir. Gral. Carlos Vizcaino Sahagún
Año 3.
Num. 17.
Enero 1995