32
24.

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES ARAGON

INTELIGENCIA ARTIFICIAL: HERRAMIENTAS Y TECNOLOGIAS

TESIS QUE SUSTENTA:

JOSUE JURADO CORTES

PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

ASESOR:

ING. AMILCAR A. MONTERROSA ESCOBAR

MEXICO, FEBRERO, 1997

A MIS PADRES

POR SU APOYO Y ESFUERZO PARA BRINDARME
UNA FORMACION PROFESIONAL

A LOS PROFESORES

POR SU RESPONSABILIDAD EN LA TAREA
DE FORMAR PROFESIONISTAS

I N D I C E

INTRODUCCION		1
1	FUNDAMENTOS	3
1.1.	Historia	3
1.2.	Qué es Inteligencia Artificial?	7
1.3.	Ramas o divisiones	9
1.4.	Areas de aplicación	11
2	LENGUAJES DE PROGRAMACION	15
2.1.	Introducción	15
2.2.	PROLOG	18
	2.2.1. Fundamentos	18
	2.2.2. Conceptos, Estructuras básicas y ejemplos	19
2.3.	LISP	28
	2.3.1. Fundamentos	28
	2.3.2. Conceptos, Estructuras básicas y ejemplos	29
3	REPRESENTACION DE PROBLEMAS Y BUSQUEDA DE SOLUCIONES	41
3.1.	Conceptos básicos	42
3.2.	Representación del problema	46
	3.2.1. En un espacio de estados	46
	3.2.2. Representación reducida	51
	3.2.2.1. Gráficas AND/OR	55
3.3.	Métodos de búsqueda	60
	3.3.1. En el espacio de estados	60
	3.3.2. En la representación reducida del problema	68
	3.3.3. Búsqueda ordenada	71
4	REPRESENTACION DEL CONOCIMIENTO	76
4.1.	Introducción	76
4.2.	Esquemas de representación	78
	4.2.1. Lógico	78
	4.2.1.1 Procesos de Razonamiento	84
	4.2.2. Redes semánticas	87
	4.2.2.1. Manipulación y Razonamiento	93
	4.2.3. Marcos (Frames)	97
	4.2.3.1. Manipulación y Razonamiento	104
	4.2.4. Guiones (Scripts)	107
	4.2.4.1. Razonamiento basado en casos	110

5	SISTEMAS EXPERTOS	111
5.1.	Antecedentes y desarrollo	111
5.2.	Características, definición y conceptos relacionados	116
5.3.	Elementos estructurales	121
5.4.	Cuando se requiere desarrollar un SE ?	128
5.5.	Etapas de desarrollo de un SE	130
6	SISTEMAS DE LENGUAJE NATURAL	132
6.1.	Antecedentes y desarrollo	133
6.2.	Conceptos y aspectos importantes	135
6.3.	Estructura de los sistemas PLN	143
6.4.	Aplicaciones	151
7	ROBOTICA	157
7.1.	Qué es un robot ?	159
7.2.	Tipos de robots industriales	160
7.3.	Componentes y operación	161
7.4.	Por qué utilizar Robots ?	168
7.5.	Robots Inteligentes	170
7.6.	Aplicaciones	173
8	REDES NEURONALES	177
8.1.	Introducción	177
8.2.	Conceptos, elementos y características	179
8.3.	Proceso de desarrollo	187
8.4.	Redes Neuronales y Sistemas Expertos	194
8.5.	Redes Neuronales y otras tecnologías	201
8.6.	Areas de aplicación para Computación Neuronal	203

CONCLUSIONES
APORTACIONES
BIBLIOGRAFIA

INTRODUCCION

A principios de la década de los 90's en nuestro país empezó a escucharse fuertemente el término INTELIGENCIA ARTIFICIAL, que era asociado con conceptos altamente técnicos y limitado a personas muy preparadas, una área reservada para elegidos. De ese tiempo a la fecha el avance en este aspecto no ha sido significativo, pues aunque se cuenta ya con más fuentes de información, aún no se ha comprendido la importancia presente y futura que esta teniendo y tendrá esta rama de la informática dentro de la sociedad humana a nivel mundial.

Sabemos que el uso de la computadora se ha extendido enormemente y no tan sólo eso, si no que ahora es muy fácil tener acceso a archivos situados en otras partes del mundo. Nos damos cuenta de las ventajas que nos proporciona el software actual y de que entre más avanza el tiempo es mucho más fácil manejarlo y aprenderlo, muchos de estos logros se deben al uso de conceptos y técnicas de INTELIGENCIA ARTIFICIAL, y así como esta área muchas otras están recibiendo los beneficios de esta disciplina por lo que es importante que los profesionales de nuestro país tengan una idea clara de lo que es y lo que realmente puede abarcar la IA.

Por esta razón, el presente trabajo quiere ser una fuente de información *accesible* que de una visión clara, y fácil de comprender sobre el área de estudio, bases metodológicas y áreas de aplicación de la IA.

Esperando despertar el interés en los actuales y futuros profesionales de la informática (y aún de otras áreas) hacia esta disciplina que desde mi punto de vista es muy apasionante y prometedora para lograr beneficios que sean de gran utilidad para la humanidad.

El objetivo fundamental de este trabajo no es analizar detalladamente las técnicas y métodos que utiliza la IA. Pero si busca dejar un panorama claro y una visión general para la mayoría de profesionistas de la forma en que la IA esta tratando de llegar a su objetivo, además de dar una idea de como poder combinar sus tecnologías entre ellas mismas y con tecnologías convencionales, con lo cual , se podrán crear sistemas que serán de gran apoyo y utilidad en tareas de automatización tanto administrativas como técnicas y cotidianas, que encaminadas hacia un enfoque de apoyo al hombre, proporcionarán grandes beneficios a la humanidad.

1 FUNDAMENTOS

1.1. HISTORIA

Desde su inicio la historia de las investigaciones sobre Inteligencia Artificial (IA) ha sido muy diversa y controvertida, aunque la mayoría de textos y artículos concuerdan en señalar como pionero de esta rama al matemático británico Alan Mathison Turing y a su invención de la computadora de programas almacenados, así como también a su famoso experimento llamado "juego de imitación", que en su honor se conoce como EL TEST DE TURING, el cual fue propuesto en un famoso artículo periodístico (Computadoras e inteligencia) publicado en 1950 por la revista MIND.

En su trabajo proponía la creación de una máquina pensante y trataba al mismo tiempo de adelantarse a las eventuales objeciones que pudieran surgir contra ella. Al final concluía diciendo que tal máquina podría lograrse en un lapso aproximado de 50 años.

La publicación del artículo de A. M. Turing creó incertidumbre e inquietud en muchos científicos y fue el incentivo para realizar investigaciones más profundas cuyo objetivo era la creación de máquinas inteligentes.

Tomando en cuenta lo anterior se ha considerado conveniente dividir las investigaciones de IA en tres periodos:

ORIGENES	1950-1965
INTERMEDIO	1966-1976
CONTEMPORANEO	1976-a la fecha

FIG. 1.1 PERIODOS DE INVESTIGACION EN IA.

Durante la primera etapa las investigaciones se enfocaron hacia el desarrollo de programas estratégicos para la práctica de juegos en particular de

ajedrez. La teoría básica fue suministrada por A. M. Turing y Claude Shannon.

Dentro de esta etapa se bautizó esta rama de la computación como INTELIGENCIA ARTIFICIAL (IA). Esto sucedió en una conferencia de neurólogos, psiquiatras, matemáticos e ingenieros reunidos en Hanover (New Hampshire, E.U.) en el verano de 1956. Fue el profesor John McCarthy del colegio de Darmouth, quien propuso el nombre de inteligencia artificial. Tal concepto haría referencia al estudio del pensamiento humano y a como el proceso de pensar podría representarse digitalmente con máquinas y programas.

Entre ellos se encontraban Claude Shannon, autor de la teoría matemática de la comunicación, Marvin Minsky, matemático y neurólogo; Nathaniel Rochester, investigador de la información; Herbert Simon, economista (premio Nobel en 1978); Arthur Samuel, autor de un avanzado programa para el juego de damas... Auténticos pioneros de la materia en aquella época, y que hoy son reconocidos expertos en IA.

En el afán por representar el conocimiento humano, surgieron varios modelos de representación, entre los cuales podemos mencionar las redes semánticas desarrolladas por Quillian en 1968 como un modelo sicológico de memoria asociativa; los sistemas de producción desarrollados por Newell and Simon en 1972, la idea básica fue la representación del conocimiento como regla de la forma SI <*condición*> ENTONCES <*acción*>; los marcos de referencia propuestos originalmente por Minsky en 1975 , los cuales hicieron posible la representación de situaciones cotidianas.

Después de esta etapa de excesivo optimismo y esperanza la IA se hundió en una era de profundo desinterés. Fue etiquetada como una disciplina inútil y por tanto relegada, las empresas ignoraron este nuevo campo del saber. La investigación sobre IA quedo a cargo de unas cuantas universidades y grupos de

investigación respaldados por subvenciones gubernamentales.

De estos años de oscuridad y como resultado de la búsqueda de la máquina inteligente, surgieron diversos subproductos prácticos populares. Entre los cuales están sistemas informáticos en tiempo compartido, visualizadores gráficos, procesadores de texto, técnicas de diseño asistido por computadora, interfaces más accesibles para usuario, menús, iconos y ratones. Durante esta época se aprendió mucho sobre la naturaleza de IA y se reafirmaron sus bases teóricas.

En la década de los ochentas la IA comenzó a mostrar resultados prácticos, ya que los adelantos en hardware permitieron proporcionar la memoria suficiente para soportar los sistemas de IA. Los cuales mostraron su utilidad, por lo que en algunos casos se diseñaron para uso interno de varias instituciones. En la tabla siguiente se resumen algunos de los programas desarrollados durante los periodos intermedio y contemporáneo.

PERIODO INTERMEDIO	PERIODO CONTEMPORANEO
1966:ELIZA Weinzenbaum	1976:HAPPY MYCIN
	WHISPER
1967:SIN Moses	1977:GUS PUFF
MAC HACK VI Greenblatt	TEIRESIAS
DENDRAL Feigenbaum	NUDGE
1968:SIR Bertram Raphael	1978:B*TREE SEARCH
STUDENT Bobrow	
1971:MACSYMA Moses	1979:EXPERT
	PROSPECTOR
	EMYCIN
1972:SHROLU Winograd	1980:HEARSAY II
1973:GSP Ronald Kaplan	1981:ROSIE
1975:MARGIE Shank	
SOPHIE Brown and Burton	

FIG. 1.2 SISTEMAS DE IA DESARROLLADOS DURANTE LOS PERIODOS INTERMEDIO Y CONTEMPORANEO

Actualmente la investigación y el desarrollo continúan, se han creado herramientas para facilitar el desarrollo de aplicaciones de IA; ha surgido la computación paralela y especialmente las arquitecturas neuronales, con múltiples procesadores operando simultáneamente. Entre los métodos de búsqueda más novedosos, tenemos a los algoritmos genéticos, que se basan en los mecanismos de la selección natural y la genética, su principal propósito es la optimización de funciones.

Internamente algunas aplicaciones usan principios y/o técnicas de IA para realizar sus tareas; Las interfaces en lenguaje natural y las bases de datos inteligentes ya son aplicaciones comunes. Y existen sistemas expertos disponibles en una amplia variedad de aplicaciones.

La IA debe combinarse con el software tradicional, esto es, subrutinas de IA serán incorporadas en tal software comúnmente usado.

1.2. QUE ES LA INTELIGENCIA ARTIFICIAL

¿Qué es IA?, interesante pregunta que hace a la gran mayoría imaginar la figura de un robot inteligente, realizando funciones propias del ser humano; tareas tales como, labores domésticas, práctica de juegos, resolución de teoremas, conducción de un auto, etc. Y tal vez esta fantasía no este tan alejada de la realidad, en un futuro no tan distante, gracias a las investigaciones y estudios realizados en el campo de la Inteligencia Artificial

INTELIGENCIA ARTIFICIAL (IA) es la parte de la ciencia de la computación enfocada al diseño de sistemas inteligentes, esto es sistemas que exhiban las características que asociamos con inteligencia en el comportamiento humano como lo son, entendimiento del lenguaje, aprendizaje, raciocinio, solución de problemas etc.

Por tanto, la IA busca la simulación de procesos inteligentes (procesos de razonamiento humano), es decir, se busca que una computadora pueda pensar y analizar como lo haría un humano, ante cualquier situación que se le presentara.

Los investigadores en IA intentan lograr su objetivo basándose en la imitación del razonamiento humano simbólico y no del numérico; esto se debe a la observación del comportamiento humano y a la forma en que este asimila el conocimiento, por poner un ejemplo, es comúnmente sabido por todos que un niño, aprende a hablar antes de aprender a contar.

Por lo anteriormente mencionado comprenderá que en IA se habla en términos de símbolos y conceptos simbólicos en vez de cantidades numéricas. Entendiéndose por símbolos los nombres de los objetos y sus atributos mismos, tales como pelo, marrón, ojos, azul, etc. Mientras que los conceptos simbólicos nos expresan ideas sobre tales símbolos.

Obviamente los programas de IA tienen su punto fuerte en la capacidad para resolver problemas que implican símbolos, conceptos e ideas simbólicas; y su comportamiento inteligente esta determinado por su capacidad para manejar los símbolos como conceptos e ideas y no como una colección de objetos carentes de significado que solo los seres humanos pueden interpretar (Esto es lo que los hace diferentes a los programas de bases de datos y procesadores de texto).

La característica mostrada por los programas de AI para entender los símbolos en términos de conceptos, ideas y relaciones hace que tales programas parezcan tener, en cierto grado, sentido común.

Estos programas difieren de los programas convencionales en que contienen tanto conocimiento como información. La información (palabras, imágenes, etc.) esta interconectada e interrelacionada. Las interconexiones, que forman diversas estructuras, se usan para representar relaciones, las cuales a su vez, están relacionadas con lo que los seres humanos interpretan como conocimiento. Observe el cuadro de la figura 1.3

INFORMACION	Colección de palabras o caracteres
CONOCIMIENTO	interconexión de palabras Tienen estructura Se indican las interconexiones

FIG. 1.3 CARACTERISTICAS DE INFORMACION Y CONOCIMIENTO.

Como podrá observar la diferencia entre información y conocimiento radica en las relaciones representadas por las distintas partes de un programa.

1.3. RAMAS O DIVISIONES

La IA se puede dividir en tres categorías básicas:

1) SISTEMAS EXPERTOS Y HERRAMIENTAS EMPLEADAS PARA SU CONSTRUCCION.

2) SISTEMAS DE LENGUAJE NATURAL.

3) SISTEMAS DE PERCEPCION PARA VISION, HABLA Y TACTO.

Los sistemas expertos son programas que utilizan procesos de razonamiento similares a los que usan los humanos, en vez de técnicas tradicionales de programación, para resolver problemas en campos específicos del saber.

Estos procesos programados, que simulan el razonamiento humano, a su vez, están basados en conocimientos humanos experimentales o habilidades, que se codifican en el programa dentro de una estructura denominada *base de conocimientos.*

Gracias a este mecanismo codificado de cognición y razonamiento, los sistemas expertos pueden afrontar problemas muy complicados en cuanto a su solución. Los sistemas expertos han tenido un satisfactorio éxito comercial en campos tales como la electromedicina y el diagnóstico médico, la configuración de sistemas informáticos, la prospección petrolífera y el procesamiento de información.

La segunda división de la IA, denominada sistemas de lenguaje natural, incluye programas que manejan el idioma en que se expresa el usuario. Estos

proporcionan un medio accesible para comunicarse con la computadora, pues elimina la necesidad de aprender un lenguaje de programación.

Actualmente estos sistemas pueden comprender el inglés coloquial, precisar ambigüedades y resolverlas.

Dentro de la última división, tenemos que los sistemas de visión computarizada, por ejemplo, interpretan escenas plásticas e interfieren acerca de la calidad (ni roto ni doblado) u orientación física de los objetos que pasan frente a una cámara de televisión. Sin embargo las capacidades de estos sistemas están aún muy limitadas. Por tanto los sistemas de visión computarizada solo son utilizados bajo ciertas condiciones de iluminación y en escenarios simples. Tales condiciones se dan en ambientes fabriles. Allí, añadiendo capacidad visual a los sistemas robóticos se elimina la necesidad de constante reprogramación y reajuste de los brazos de los robots que suelen desajustarse incesantemente.

1.4. AREAS DE APLICACION

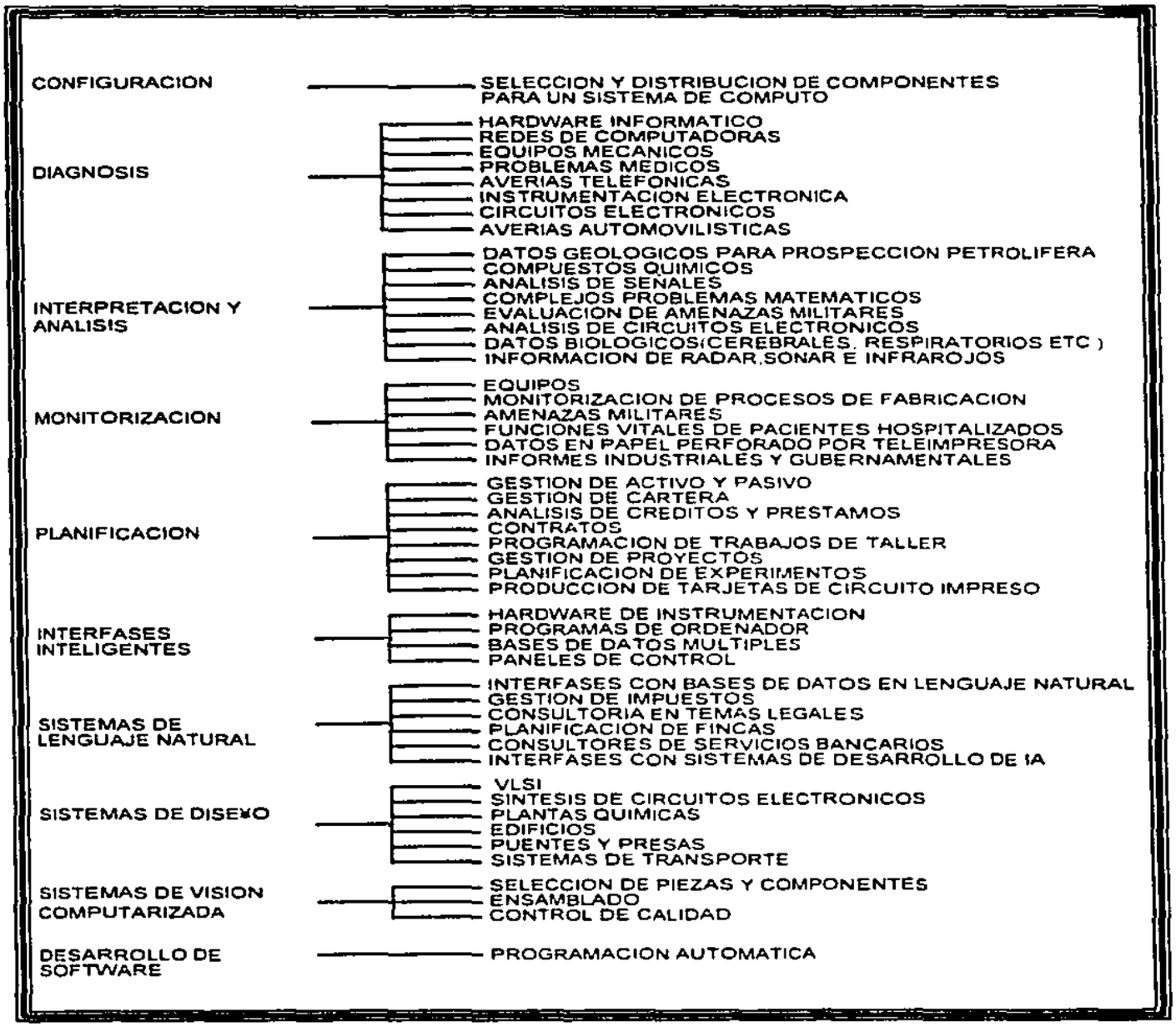


FIG. 1.4. TIPOS DE PROBLEMAS RESUELTOS POR IA

Como se observa en el cuadro de la figura 4.1., los programas de IA encuentran aplicación en una gran variedad de sistemas reales.

Los sistemas de configuración buscan la optimización de los elementos de un sistema de procesamiento de información tanto en hardware como en software. Estuvieron entre los primeros sistemas expertos comercializados y aún siguen teniendo mucha demanda.

Los sistemas de diseño llevan a cabo la planificación de un sistema en base a requisitos especificados; además de evaluar las consecuencias de las distintas opciones de diseño. Las empresas electrónicas desean incorporar técnicas de IA para el diseño de la nueva generación de microcircuitos electrónicos.

Los sistemas de interpretación y análisis escudriñan y descifran grandes volúmenes de datos. Estos pueden estar en forma gráfica, y pueden representar la respuesta, por ejemplo, de determinado medio a la transmisión de señales acústicas o eléctricas, bombardeo con partículas nucleares, exploración con energía infrarroja, barrido de radar o sonar, o interacción con campos magnéticos. Al recibir la interpretación dada por el sistema, el especialista humano, puede inferir ciertas características, tales como la estructura geológica del subsuelo, estructuras atómicas de compuestos químicos o la presencia de fuerzas enemigas en tierra, mar o aire.

Los sistemas de monitorización permiten prever la resistencia de una estructura frente a un terremoto, calcular las consecuencias de una amenaza bélica, determinar objetivos tácticos, detectar equipos o sistemas, supervisar procesos de fabricación y en algunos casos ayudar en la resolución simbólica de problemas algebraicos.

Para evitar desastres potenciales, además de las funciones de monitorización los sistemas de análisis efectúan juicios críticos en breves intervalos de tiempo. Estos sistemas son comunes en plantas de fabricación con control automático de proceso.

Los sistemas de planificación encuentran aplicación en los campos financiero, industrial y científico. Y a través de los conocimientos adquiridos de especialistas humanos efectúan las diversas tareas mostradas en el cuadro anterior.

Las interfaces inteligentes son programas de IA que facilitan a los usuarios el manejo de hardware o software excesivamente complejo. Esto elimina la necesidad de personas altamente especializadas.

Las interfaces de lenguaje natural actúan como primera línea de los sistemas de gestión de bases de datos. Esta acepta las peticiones de los usuarios en lenguaje natural (generalmente inglés), y las traduce en instrucciones, más complejas, utilizadas por los lenguajes formales de consulta de bases de datos. Actualmente la mayor parte de empresa de bases de datos de los EE.UU. están desarrollando interfaces con bases de datos en lenguaje natural debido a su gran demanda por parte de usuarios y ejecutivos empresariales.

Los sistemas de visión computarizada están siendo de gran utilidad para que los robots realicen tareas tales como selección de piezas, pintura usando pistolas pulverizadoras, etc. Un sencillo sistema de visión convierte al robot en una máquina inteligente capaz de identificar una pieza, colocarla, y adoptar la posición más idónea para efectuar las manipulaciones requeridas. Otras áreas importantes para aplicación de estos sistemas son, el control de calidad y la inspección visual.

Y por último tenemos a la programación automática, la cual, se refiere al desarrollo de sistemas expertos para diseño de programas de IA que automaticen el proceso de creación de otros programas. Los sistemas actuales de programación automática sólo resuelven problemas en campos muy específicos, tales como la recuperación y actualización de información en bases de datos. Específicamente, dentro de esta área se planea, que en un futuro se combinen los procedimientos tradicionales de programación con técnicas de AI y se podrán desarrollar programas relativamente complejos, pero muy fiables, pues en ellos se combinarán el lenguaje natural con los sistemas expertos.

2. LENGUAJES DE PROGRAMACION

2.1. INTRODUCCION

Dentro de los lenguajes de alto nivel, podemos distinguir lenguajes: procedurales y declarativos.

Los **lenguajes procedurales** (FORTRAN, PASCAL, COBOL, etc.), también conocidos como imperativos, se basan en la asignación de valores a variables y rutinas que indican como usar una solución, donde buscarla, cuando parar, etc. Estos lenguajes nos proporcionan resultados que de alguna manera implican términos o relaciones numéricas.

Los **lenguajes declarativos** (LISP, PROLOG), requieren la declaración de reglas y hechos sobre objetos o símbolos específicos, y permiten la manipulación de estructuras simbólicas, objetos y sus respectivas relaciones. Unas cuantas líneas de un lenguaje declarativo, pueden hacer tanto trabajo como muchas líneas de un lenguaje procedural.

Estos lenguajes brindan gran facilidad para que a través de ellos se desarrollen aplicaciones e investigaciones que, tomen en cuenta los conceptos y objetivos que persigue la IA.

Se dividen en tres categorías, las cuales se mencionan a continuación:

A) LENGUAJES DE APLICACIONES FUNCIONALES
B) LENGUAJES DE PROGRAMACION LOGICA
C) LENGUAJES ORIENTADOS A OBJETOS

Los lenguajes funcionales tienen como su más fuerte representante a LISP, el cual, se ha venido usando para la programación de IA desde hace 32

años, este lenguaje se basa en la utilización de funciones, las cuales, permiten la manipulación de estructuras de datos simbólicas para la obtención de un resultado. Sus fundamentos y características generales serán tratados en la siguiente sección del capítulo.

Entre los lenguajes de programación lógica destaca PROLOG, el cual, se basa en la especificación de hechos y reglas (correspondientes a un problema) a través de la lógica, tales hechos y reglas se almacenan en una base de datos, lo cual permite establecer relaciones y hacer inferencias para generar alguna respuesta o solución buscada. (En el siguiente apartado veremos algunas características de este lenguaje).

Dentro de la última familia un lenguaje muy característico es SMALLTAK, el cual fue desarrollado en el centro de investigación de Palo Alto de Xerox Corporation como parte de un proyecto destinado a investigar los diferentes modos en que un niño se puede comunicar con un computador, fue el primer lenguaje de programación orientada al objeto y permite a los programadores crear clases de objetos y objetos específicos, además de procedimientos para manipular dichos objetos.

Tradicionalmente, SMALLTAK se ha utilizado para trabajar con gráficos, ya que ofrece gran versatilidad para el manejo de estos. Con SMALLTAK es más fácil desarrollar una representación gráfica que una textual, porque trata los textos como si fuesen gráficos.

Su utilización primaria fue como soporte de interfaces gráficas en computadoras. También se aplicó mucho en aplicaciones de fotocomposición de periódicos, ya que esto requería una buena dosis de diseño gráfico.

Smalltak también es muy utilizado para el desarrollo de aplicaciones de IA en ingeniería, ya que el diseño de circuitos electrónicos requiere gran capacidad gráfica. Esto se debe, a que los ingenieros tienden a pensar visualmente en términos de circuitos lógicos, esquemas y planos, más que en términos semánticos. Por tal razón la orientación gráfica de SMALLTAK lo hace idóneo para aplicaciones de ingeniería o de composición tipográfica con fuertes requerimientos visuales.

En este capítulo nuestro objetivo será proporcionar las bases y conceptos fundamentales de los lenguajes LISP y PROLOG. La razón de esta elección es que son los más difundidos en este campo, aquel por los norteamericanos, y este por la comunidad europea, además de que los japoneses lo utilizarán en la implementación de sus computadoras de quinta generación y que incorporarán el procesamiento en paralelo.

2.2. PROLOG.

2.2.1. FUNDAMENTOS.

Prolog representa la abreviatura "PROgraming in LOGic". Es un lenguaje de programación declarativo y de orientación lógica, que fue desarrollado en 1973 por A. Colmerauer y P. Roussel en los laboratorios de IA de la Universidad de Marsella. Se han realizado trabajos posteriores en la Universidad de Edinburg en Gran Bretaña.

PROLOG se basa en manipulaciones lógicas de declaraciones o cláusulas establecidas; posee un motor de inferencia para búsqueda de hechos y construcción o prueba de conclusiones lógicas. Es idóneo para la gestión y consulta de bases de datos relacionales.

Ya que PROLOG es un lenguaje declarativo, para implementar una aplicación en este lenguaje se debe poner especial atención en la descripción adecuada del problema. Por tanto es primordial realizar la declaración de los siguientes elementos (ambiente de operacionalización):

1) Hechos conocidos
2) Reglas permitidas (que harán posible la combinación de hechos o inferencia de nuevos hechos)
3) El objetivo u objetivos deseados

Los cuales, se almacenarán en un conjunto de cláusulas, las cuales, conformarán la base de datos que contendrá la información del problema a resolver. Después de esto, Prolog buscará en la base de datos hasta satisfacer el objetivo u objetivos planteados.

2.2.2. ESTRUCTURAS BASICAS Y EJEMPLOS

Las explicaciones hechas en esta sección se basan en Turbo Prolog ya que es una versión de Prolog disponible para PC que conserva fundamentos, características generales y sintaxis de Prolog añadiendo algunas características entre las cuales la más sobresaliente es un ambiente de trabajo muy similar al de turbo pascal, turbo c, etc. y que nos permite una mayor facilidad en su manejo.

PARTES DE UN PROGRAMA.

Las partes principales de un programa proporcionan la información mencionada en la sección anterior, y estas se dividen en:

DOMAINS: Contiene declaraciones de dominio, es decir, descripción de las diferentes clases de objetos utilizados en el programa.

PREDICATES: Especificación de las relaciones a utilizar en el programa.

CLAUSES: Declaración de los datos del programa en forma de reglas y hechos, los cuales deben ser generales o universales.

GOAL: Indica a Prolog cuál es el objetivo a satisfacer, para esto Prolog utilizará la información suministrada en las otras tres secciones. Puede aparecer dentro del programa (interno), o bien en una ventana de edición en diálogo para pedir el "goal" en forma de predicado (externo).

Normalmente un programa tendrá al menos las secciones de predicates y cláusulas.

CLAUSULAS: HECHOS Y REGLAS.

Hechos

Como se mencionó anteriormente, la sección clauses debe contener declaraciones de hechos y reglas, pero que es para Prolog un hecho?, Para contestar a esta pregunta debemos comprender que Prolog trabaja con objetos y sus respectivas relaciones, entendiéndose como objeto cualquier cosa que pueda representarse simbólicamente en una computadora, y como relación, el papel que juega dicho objeto dentro del ámbito dado.

Por tanto un hecho es un suceso del mundo real, expresado a través de una declaración lógica, la cual, representará la combinación objeto-relación y será incondicionalmente verdadera.

La sintaxis para declarar un hecho es:

nombre_relación(objeto$_1$,objeto$_2$,...,objeto$_n$).

Ejemplos:

animal(vaca).
computadora(ibm).
tiene(casa,puerta).
hermano(Juan, María).
trabaja_en(Juan, Pemex).

Reglas

Recordemos que también podemos introducir reglas en la sección clauses, *una regla es una declaración lógica que permite relacionar hechos para inferir información sobre tales hechos. Su característica principal es el uso de la proposición SI-ENTONCES.* En otras palabras una regla nos afirma que algo es verdad si se cumple alguna condición. Gracias a esto Prolog deja de ser una base de datos común para convertirse en una máquina lógica pensante.

Observemos la siguiente declaración:

tiene(X,puerta) :- es_una_casa(X)

Esta declaración es conocida como una criterio de Horn en cálculo de predicados, y es la base de la inferencia basada en reglas que maneja Prolog. La parte izquierda de la declaración se conoce como consecuente, y será verdadera, sólo si la parte derecha (antecedente) es verdadera. Cabe mencionar que en este tipo de reglas Prolog utiliza el principio conocido como *unificación*, el cual, permite identificar las partes derecha e izquierda de la regla, buscando los valores de la variable, que permitirán una identificación correcta para el cumplimiento de la regla. Interpretada al español la regla anterior diría: *Si X es una casa, entonces X tiene una puerta.*

Observe que X esta en mayúscula, esto es, porque se trata de un objeto variable, y se asume que todas las variables empiecen con mayúscula. Cabe mencionar que también podemos utilizar "if" en lugar de ":-".

La sintaxis para declarar una regla es:

nombre_relación$_1$(obj$_1$,obj$_2$,..):-

nombre_relación$_2$(obj$_1$,obj$_2$,...),

...

nombre_relación$_R$(obj$_1$,obj$_2$...).

Las reglas permiten una manipulación de hechos que hacen a Prolog sumamente potente, como ejemplo considere los siguientes hechos:

dentro_de(recámara, Pedro)

dentro_de(casa, recámara)

Si quisiéramos inferir que Pedro se encuentra en casa, tendríamos que darle a Prolog la siguiente regla:

dentro_de(C,P):-dentro_de(R,P),dentro_de(C,R)

Al poner como objetivo: dentro_de(casa,pedro)
Prolog a través del uso de la regla y realizando la siguiente sustitución:

C -> casa
R -> recámara
P -> Pedro

Nos regresa el valor "TRUE"

Predicados

la sección predicates de Turbo Prolog especifica las características de los argumentos de las relaciones a usar en la sección clauses. La sintaxis que se sigue es la siguiente:

nom_predicado($argumento_1$,$argumento_2$,...$argumento_n$)

Dominio

indica a prolog que tipo de datos serán los argumentos de un predicado. Los más comunes para datos simbólicos son *symbol* y *string*, los cuales identifican cadenas de caracteres.
La sintaxis es:

nom_argumento = dominio

Conectores Lógicos

Prolog permite el uso de conectores lógicos, los cuales son similares a los empleados en declaraciones lógicas y son generalmente usados para la declaración de reglas. El siguiente cuadro muestra las notaciones posibles de los conectores en Prolog.

CONECTOR	NOTACIONES POSIBLES	
NEGACION	NOT	
CONJUNCION	AND	,
DISYUNCION	OR	;
IMPLICACION (SI-ENTONCES)	IF	:-

Ejemplo:

Para comprender la forma en que trabaja Prolog, observemos el siguiente programa:

```
domains
persona=symbol

predicates
padre(persona,persona)
madre(persona,persona)
abuelo(persona,persona)
rubia(persona)

clauses
padre(luis,jose).
padre(jose,miguel).
padre(luis,lupe).
madre(lupe,paulina).
abuelo(X,Y) IF padre(X,Z) AND padre(Z,Y) OR madre(Z,Y).
rubia(paulina).
```

Pidamos el siguiente objetivo:

goal: abuelo(luis,Nieto), rubia(Nieto)

El cual puede interpretarse como la pregunta: Quién es el nieto de Luis cuya característica es ser rubia?, es decir se quiere encontrar algún elemento que tenga las siguientes propiedades:

-Ser nieto de Luis y,

-Ser rubio.

Prolog buscará, entre los hechos que tiene almacenados, al objeto que cumpla con ambas propiedades respondiendo a las preguntas de izquierda a derecha, por tanto primero resolverá la cuestión *abuelo(luis,Nieto).* Para lo cual buscará en la sección clauses algún hecho o regla que contenga el objeto abuelo; al encontrarlo sustituye Luis por X y Nieto por Y, obteniendo:

abuelo(luis,Nieto) IF padre(luis,Z) AND padre(Z,Nieto) OR madre(Z,Nieto)

Como se trata de una regla, primero trata de verificar el antecedente encontrándose con el hecho:

padre (luis,Z)

Ahora busca entre los hechos declarados el primer valor de Z que cumpla con lo anterior, obteniendo Z=jose. Al seguir avanzando se encuentra un punto de decisión, tiene que elegir entre dos alternativas posibles. Prolog siempre elige la primera por lo que obtiene:

padre(jose,Nieto)

De lo cual, a través de otra búsqueda obtiene:

Nieto=miguel

Con esto se cumple la primer parte del objetivo "abuelo(luis,miguel)". Pero al sustituir el valor de Nieto encontrado en la segunda parte del objetivo obtenemos:

rubia(miguel)

Que no corresponde a los hechos declarados en la sección clauses. Por tanto Prolog realiza un backtrack (vuelta atrás), hasta el punto donde tomó su última elección, pero ahora elige la otra alternativa, obteniendo:

madre(Jose,Nieto)

Lo cual no corresponde con la información declarada en los hechos. Por lo cual, Prolog realiza otro backtrack, regresando al hecho:

padre(Luis,Z)

pero, ahora busca (si existe) otro valor de Z que cumpla con lo anterior y que no sea jose. Encontrando:

Z=lupe

Al resolver Prolog nuevamente:

madre(lupe,Nieto)

Obtiene: Nieto=paulina

Lo cual cumple con el también cumple con la segunda parte del objetivo:

rubia(Paulina)

Por tanto acaba el proceso y se entrega la respuesta correcta a la pregunta formulada en un principio.

A pesar de todo el proceso que se desarrollo para la satisfacción del objetivo, cabe señalar que Turbo prolog entrega el resultado mencionado en una fracción de segundos.

Otra punto importante de notar, es el papel que desempeña el "backtracking" que es un algoritmo propio de prolog, también llamado de retrobúsqueda. El cual permite, que al no encontrar una solución por un camino seleccionado, se pueda regresar al punto de decisión y elegir otra ruta; hasta que dicha solución (si existe) sea encontrada.

Backtracking.

Como se observó en el ejemplo PROLOG posee un mecanismo de control, el "backtracking" que es un medio de seleccionar la siguiente instrucción a ejecutar dentro de un programa. Mediante este mecanismo, si una regla seleccionada por el programa no conduce a una solución, el programa retrocede hacia el punto de decisión y selecciona otra regla. Finalmente después de intentar varias vías erróneas, el control del programa retrocede para seleccionar las reglas que conducen a una solución.

De lo anterior concluimos que PROLOG no dispone de medios para seleccionar las mejores rutas en la solución de un problema. Consecuentemente, nos podemos encontrar con un gran espacio de búsqueda que requerirá una gran cantidad de memoria.

Para evitar este problema David Warren, diseñador del compilador PROLOG del DEC-10, sugiere ordenar los objetivos de modo que se minimicen las búsquedas. Para lograr esto se debe dar prioridad a los objetivos con menor número de soluciones sobre aquellos con mayor número de soluciones. Por poner un ejemplo, en el programa realizado hubiera sido mejor poner la regla de la siguiente manera:

```
abuelo(X,Y) IF padre(X,Z) AND madre(Z,Y) OR padre(Z,Y).
```

Con lo cual hubiéramos evitado un "backtrack" a Prolog.

Existen dos predicados relacionados con el backtracking, corte (" !") y fallo ("fail").

Corte (" !")

Esta orden (o predicado especial) se escribe en los programas simplemente con un signo de exclamación. Su función es impedir la vuelta atrás (backtrack) actuando como una válvula en un sentido. Al encontrar un corte, se liberan las variables vinculadas en la búsqueda de un objetivo de sus valores actuales. Prolog no regresa al punto antes del corte y examina otros valores para esas variables.

Fallo ("fail")

Es otra orden para controlar la vuelta atrás; es un predicado sin argumentos que automáticamente falla y fuerza la vuelta atrás. En cierto sentido es opuesta al corte, el cual elimina la vuelta atrás.

2.3 LISP
FUNDAMENTOS.

LISP representa la abreviatura "LISt Processing", y fue desarrollado a final de la década de los cincuentas por John McCarthy. Cabe señalar que LISP es el segundo lenguaje más antiguo para programación de computadoras (FORTRAN fue el primero).

Como se mencionó anteriormente es un lenguaje de programación funcional, ya que, utiliza funciones para manipular estructuras de datos simbólicas (palabras u oraciones), en vez de manejar estructuras de datos numéricas. Y permite al programador la creación y definición de sus propias funciones que pueden ser anidadas y que frecuentemente son recursivas.

La principal característica de LISP es la facilidad que brinda para el manejo de listas (de ahí su nombre), las cuales, pueden estar representadas por palabras u oraciones. Ejemplos de estas pueden ser: *H, 4, color_de_ojos, escritorio,* etc.

La finalidad de LISP no es la realización de cálculos matemáticos, ya que, en IA raramente se utilizan funciones matemáticas complicadas.

Para el principiante la característica que más llama la atención es el elevado número de paréntesis que rodean sus instrucciones. Pero a pesar de esto, es un lenguaje fácil de aprender y su sintaxis es muy sencilla.

CONCEPTOS ESTRUCTURAS BASICAS Y EJEMPLOS

Conceptos Basicos

A continuación se presentan algunos conceptos importantes del entorno LISP.

Los programas LISP son colecciones de procedimientos independientes denominados funciones.

Cualquier estructura simbólica de datos puede ser un objeto. Existen estructuras de datos simples y compuestas.

Los estructuras de datos simples son llamados átomos y consisten de 1 o más caracteres. Ejemplos:

```
A1
NOMBRE
ATOMO
NIL
CAJA-NEGRA
```

Los estructuras de datos compuestas se denominan listas y están formados por cero o más átomos o bien por cero o más listas. Las listas se representan con sus elementos encerrados entre paréntesis. Ejemplos:

```
(A B)
(CUADRO TRIANGULO CIRCULO)
(A B (C D) E)
(ESTA ES UNA LISTA)
()
```

Cabe mencionar que la lista que carece de átomos se conoce como lista vacía, la cual es representa por LISP como el átomo NIL, observe el siguiente ejemplo:

'0
= > NIL

En LISP se usa la notación de prefija para expresar funciones que manipulan operadores. Ejemplos:

(PLUS 4 3)
(DIFFERENCE 5 3)
(TIMES 9 7)

Permite funciones especiales llamadas predicados. Un predicado se evalúa de cierto o falso (En LISP falso se denomina NIL).

Las funciones LISP se evalúan independientemente, es decir, tan pronto como entra una expresión en el sistema, LISP la evalúa y nos entrega el resultado en el renglón inmediato. Esto se debe a que en LISP se interpreta, no se compila, esta característica hace que LISP sea un lenguaje altamente interactivo que da rápidas respuestas. Las reglas con que se evalúan las expresiones son las siguientes:

- El átomo numérico da él mismo como resultado.

- El átomo literal es considerado como una variable, y da el valor más recientemente afectado. Si no se ha afectado nada, el valor es indeterminado.

La lista que sea diferente de NIL es considerada como una llamada a una función. El primer elemento de la lista representa la función, y los restantes representan los argumentos.

Estructuras Basicas

En los ejemplos posteriores se usa un intérprete de LISP llamado MULISP, el cual posee la mayor parte de estándares del Lisp original, con la pequeña peculiaridad de que todo se maneja en mayúsculas.

Funciones Primitivas.

El poder de LISP lo da su flexibilidad, ya que nos permite moldearlo para definir funciones de acuerdo a las necesidades del programador, o modificar las ya existentes. Sin embargo existen funciones que no pueden ser modificadas, ellas son llamadas funciones primitivas.

En la programación convencional tenemos los operadores +,-,* y / los cuales operan con términos numéricos. Análogamente en LISP tenemos las funciones CAR, CDR y CONS para la manipulación de listas.

Para que una función pueda ser evaluada en LISP, se debe cumplir con la siguiente sintaxis:

(<nom_func> <arg1> <arg2> ...)

Donde:

nom_func = nombre de la función,
arg1 = primer argumento,
arg2 = segundo argumento, etc.

Cabe señalar que puede no haber argumentos.

CAR.

Esta función nos permite extraer el primer elemento de una lista, por lo que comúnmente para decir el primer elemento de una lista, los programadores LISP se refieren a este como el CAR de la lista. Como ejemplo analice las siguientes líneas.

```
(CAR '(JUAN JOSE ANA LUIS))
-> JUAN

(CAR '(A1 A2 A3 A4))
-> A1

(CAR '((A B)(C D)(E F))
-> (A B)
```

El carácter que antecede a las listas ('), es conocido como quote y le indica a LISP que tome la expresión tal como está escrita y que no la evalúe. El símbolo -> se usa para indicar la respuesta que LISP nos dará después de evaluar la expresión dada. Ejemplos:

```
(PLUS 2 2)
-> 4

'(PLUS 2 2)
-> (PLUS 2 2)
```

CDR

Esta función nos permite extraer todos los elementos de una lista exceptuando el primer elemento. Se pronuncia cuder. Analice los siguientes ejemplos:

```
(CDR '(JUAN JOSE ANA LUIS))
-> (JOSE ANA LUIS)

(CDR '(A B C))
-> (B C)

(CDR '((A B)(C D)(E F)))
-> ((C D) (E F))
```

CONS

Esta función construye una lista. CONS toma dos argumentos (átomos y/o listas). El primer argumento viene a ser el primer elemento de la lista, o

sea el CAR de la lista. El segundo argumento representará el resto de la lista, también conocido como el CDR de la lista. Observe los siguientes ejemplos:

```
(CONS 'JUAN '(ES INGENIERO))
-> (JUAN ES INGENIERO)

(CONS 'A '(B C))
-> (A B C)

(CONS 'A ())
-> (A)
```

ATOM

Es una función identificadora utilizada para saber si un objeto es un átomo. Esta función usa solo un argumento. Si el argumento es un átomo, ATOM regresa el átomo T(TRUE); de otra manera regresa NIL. Ejemplos:

```
(ATOM 'A)
-> T

(ATOM '(A))
-> NIL

(ATOM ())
-> T

(ATOM '(A B C D E))
NIL
```

EQ

Función comparadora utilizada para averiguar si dos argumentos son el mismo átomo. Esta función es útil sólo si los argumentos a comparar son átomos.

Ejemplos:

```
(EQ 'ABC 'ABC)
-> T
```

```
(EQ 'ABC 'ABD)
=> NIL

(EQ (CAR '(T1 T2 T3))(CAR(CDR'(C1 T1 A1))))
=> T
```

Hemos visto las funciones primitivas que posee LISP a continuación veremos como crear nuestras propias funciones a partir de las primitivas.

ALGUNAS FUNCIONES ADICIONALES PARA MANIPULACION DE LISTAS.

A continuación veremos algunas funciones que combinadas adecuadamente con las funciones primitivas, nos permitirán hacer una manipulación muy sencilla de nuestras listas.

SETQ.

Permite asignar a la variable X el valor de la variable Y. Observe los siguientes ejemplos:

```
(SETQ A 1)
-> 1

A
->1

(SETQ A 'B)
->B

(SETQ A 'C)
->C

A
->C
```

LENGTH.

Nos da el número de elementos en una lista determinada.

Ejemplos:

```
(LENGTH '(A B C))
-> 3

(LENGTH '(A (B C) D))
-> 3

(LENGTH '((A B) (C D)))
-> 2
```

RPLACA.

Reemplaza el car de una lista con la lista que se le especifique.

Ejemplos:

```
(SETQ X '((A B) C D))
-> ((A B) C D)

(RPLACA X 'M)
-> (M C D)

(RPLACA X '(N O (E F)))
-> ((N O (E F)) C D)

(RPLACA X 'P)
-> (P C D)
```

RPLACD.

Reemplaza el cdr de una lista.

Ejemplo:

```
(SETQ X '(A B C))
-> (A B C)

(RPLACD X '(F))
-> (A F)

(RPLACD X 'D)
-> (A . D)

(RPLACD X '((M N)))
-> (A (M N))
```

NTH

Regresa la posición de un elemento sobre una lista. Cabe hacer notar que considera la posición del primer elemento como cero.

Ejemplos:

```
(SETQ NUMEROS '(1 2 3 4 5 6 7 8 9))
-> (1 2 3 4 5 6 7 8 9)

(NTH 0 NUMEROS)
-> 1

(NTH 1 NUMEROS)
-> 2
```

```
(NTH 9 NUMEROS)
-> NIL
```

POP.

Usa la lista como si fuera una estructura de tipo pila y elimina el primer elemento de la misma. Ejemplos:

```
(SETQ PILA '(A (B C) D E F))
-> (A (B C) D E F)

(POP PILA)
-> A

PILA
-> ((B C) D E F)

(POP PILA)
-> (B C)

PILA
-> (D E F)
```

PUSH.

Agrega un elemento al frente de la lista.

Ejemplos:

```
(SETQ X '(A (B C)))
-> (A (B C))

(PUSH 'A X)
-> (A A (B C))

(PUSH '(E F) X)
-> ((E F) A A (B C))
```

MEMBER.

Determina si un elemento pertenece a la lista especificada. Member regresa una sublista que principia donde se hayo el elemento buscado, de otra manera regresa NIL.

Ejemplo:

```
(SETQ CLASE '(JUAN BETY ANA LALO GLORIA))
- > (JUAN BETY ANA LALO GLORIA)

(MEMBER 'JUAN CLASE)
- > (JUAN BETY ANA LALO GLORIA)

(MEMBER 'GLORIA CLASE)
- > (GLORIA)

(MEMBER 'MARY CLASE)
- > NIL
```

CREACION DE FUNCIONES

En la mayoría de lenguajes, los nombres de funciones son palabras reservadas. Las funciones son definidas por los creadores del lenguaje y no hay manera de agregar más. Claro que puedes escribir subrutinas subprogramas o procedimientos pero esto no es lo mismo que la creación de una nueva función.

LISP fue creado sobre el principio de que tu puedas crear funciones dependiendo de las necesidades que se te presenten.

DEFINIENDO UNA FUNCION

La macro DEFUN es utilizada para definir funciones. Mediante esta se asigna un nombre a la función, para posteriormente poder hacer uso de esta.

Observe el siguiente ejemplo:

```
(DEFUN CUBO (X)
(TIMES X X X))
-> CUBO
```

Como se observa es un ejemplo tradicional, esto se hace para que el principiante entienda un poco mejor con base en lo que esta acostumbrado a hacer. El nombre de la función es CUBO, y la tarea que realiza es la multiplicación de su argumento (X) por si mismo tres veces. Observe que al momento de la definición de la función el valor que retorna es el nombre mismo de la función.

Para usar esta función, hay que hacer lo siguiente:

```
(CUBO 3)
-> 27

(CUBO 2)
-> 8
```

Como habrá observado para la definición de una función se utiliza la siguiente sintaxis:

```
(DEFUN <NOMBRE DE LA FUNCION>
(<ARG1><ARG2>...)
(TAREA1)
(TAREA2)...)
```

Donde:

NOMBRE DE LA FUNCION: Es el nombre que se utilizará para hacer uso de la misma.

ARG1,ARG2,ETC.: Son los argumentos formales, que serán sustituidos por los reales al hacer la llamada de la función.

TAREA1,TAREA2,ETC.: Forman el cuerpo de la función e indican lo que esta realizará con los argumentos.

Ahora haciendo uso de algunas funciones primitivas definiremos una función para encontrar el segundo elemento de una lista:

```
(DEFUN SEG(LISTA)
       (CAR(CDR LISTA)))
```

Llamando a la función creada:

```
(SEG '(A B C D))
=> B
```

Como se observa, LISP permite la creación de nuevas funciones a partir de las funciones primitivas u otras funciones ya creadas (por el programador o por la versión de LISP usada), estas nuevas funciones se agregan al lenguaje y pueden ser usadas observando las mismas reglas que las ya existentes en el mismo.

3. REPRESENTACION DE PROBLEMAS Y BUSQUEDA DE SOLUCIONES

Dentro de la IA la resolución de problemas es una de las áreas típicas; inclusive se puede asegurar que la IA nace con la resolución de problemas a través de métodos de búsqueda.

Durante las primeras etapas de investigación en IA, el desarrollo de métodos para elegir las mejores alternativas al resolver un problema era primordial, se requería que los programadores diseñaran técnicas de búsqueda aceptables para la resolución de problemas. Esto dio origen a los denominados métodos de búsqueda que se convirtieron en una herramienta fundamental para la solución de problemas dentro de IA.

3.1. CONCEPTOS BASICOS

ELEMENTOS DE LOS SISTEMAS PARA SOLUCION DE PROBLEMAS

En términos generales, podemos decir que los sistemas para solución de problemas, constan de tres componentes fundamentales, los cuales son:

- UNA BASE DE DATOS
- UN CONJUNTO DE OPERADORES
- UNA ESTRATEGIA DE CONTROL

La *base de datos*, debe contener información sobre la delimitación y especificaciones propias del problema (en adelante *ámbito*), así como el objetivo buscado; puede estar compuesta por diferentes clases de estructuras de datos incluyendo arreglos, listas, conjuntos de expresiones de cálculo de predicados, redes semánticas, etc. Como ejemplos tenemos los siguientes:

- Al comprobar un teorema, el *ámbito* consiste de axiomas que representan afirmaciones, leyes y teoremas ya aprobados. El objetivo será la comprobación del teorema dado.

- En la solución de problemas sobre sistemas de robótica, *el ámbito* puede ser un *modelo del mundo real* consistente de sentencias que describen su alrededor o medio ambiente, y el objetivo será la descripción de lo que se realizará por medio de una serie de acciones del robot.

El *conjunto de operadores*, modificará la situación inicial de la base de datos para llegar a una situación deseada. Algunos ejemplos de operadores son:

1) En prueba de teoremas, reglas de inferencia, tales como modus ponendo ponens y resolución.

2) En ajedrez, reglas para movimiento de las piezas.

3) En integración simbólica, reglas para simplificar las expresiones a ser integradas, tal como integración por partes o sustitución trigonométrica.

La *estrategia de control* debe manipular de una manera adecuada la aplicación de los operadores (que operador aplicar y donde aplicarlo). Cabe señalar que, en ocasiones existe un control maestro (control centralizado) que administra los recursos para lograr la solución del problema. En otros casos el control recae sobre los operadores propios del problema (control descentralizado).

La elección de una buena *estrategia de control* para la aplicación de operadores permitirá un manejo óptimo de la información de la *base de datos*. Ya que, se busca llegar al objetivo planteado aplicando una secuencia de operadores a un *ámbito* dado, el cual como ya se mencionó, esta contenido en una base de datos. Como es de esperarse, la aplicación de un operador modificará las condiciones del problema; por esta razón, la base de datos debe conservar la estructura de datos principal, mostrando los efectos de cada secuencia aplicada. Esto permitirá a la *estrategia de control* investigar el efecto de varias secuencias de operadores en paralelo y centrar su atención sobre aquellas que se miren prometedoras.

FORMAS DE RAZONAMIENTO

Existen dos formas generales de razonamiento, es decir, dos maneras en las cuales buscamos la solución a un problema, estas son:

El *razonamiento progresivo* (o hacia adelante), consiste en la aplicación de operadores sobre una *situación* o condición inicial del problema para producir una situación diferente (avanzar en la solución del problema). La idea es que desde una situación inicial se llegue a una situación deseada. Por ejemplo, una situación inicial podría ser la colocación de una pieza de ajedrez sobre el tablero, al inicio del juego; la condición deseada, podría ser cualquier configuración del tablero que sea un jaque mate; y los operadores, serán las reglas de movimiento legales en el ajedrez.

El *razonamiento regresivo* (o hacia atrás) difiere del razonamiento progresivo, en que el operador se aplica a la situación deseada u objetivo (no a la situación inicial), la cual se convierte a uno o más subobjetivos que serán más fáciles de resolver, y cuyas soluciones bastarán para resolver el problema original. Estos subobjetivos pueden a su vez ser reducidos a más subobjetivos, y así sucesivamente, hasta que cada uno de ellos tenga una solución inmediata. Por ejemplo, dado un problema inicial de integración $1/(\cos^2 x)dx$ y un operador permitido $1/(\cos x)$ que puede ser representado como $(\sec x)$, podemos trabajar hacia atrás hasta una expresión cuya solución sea inmediata. La integral de $(\sec^2 x)$ es $(\tan x)$.

EJEMPLO.

Supongamos que deseamos volar de México a Sidney pero no existen vuelos directos entre estas ciudades. Por tanto, tendremos que buscar una combinación de vuelos, que nos permitan lograr la ruta deseada.

Hay dos formas de resolver este problema, a continuación se mencionan ambas.

A) Empezar con todos los vuelos que lleguen a Sidney y localizar la ciudad de donde partió cada vuelo. Después analizar los vuelos a dicha ciudad y localizar su origen. Continuar con el proceso hasta localizar un vuelo cuyo origen haya sido México. Resolviendo el problema de esta forma, iniciamos del objetivo (Sidney) hacia atrás, es decir utilizamos el razonamiento regresivo (manipulación del objetivo).

B) Localizar todos los vuelos cuyo origen sea México e investigar las ciudades hacia donde se dirigen (ciudades intermedias). Buscar el destino de los vuelos de tales ciudades; Continuar el proceso hasta localizar Sidney. Realizando este proceso, partimos de la situación inicial (México) hacia el objetivo, Por tanto se utilizó un razonamiento progresivo (manipulación de datos).

3.2. REPRESENTACION DEL PROBLEMA

3.2.1. EN EL ESPACIO DE ESTADOS

Esta técnica de representación consiste en mostrar la estructura de un problema en términos de todas sus posibles alternativas, por ejemplo, los movimientos permitidos en un juego específico.

Para definir un problema como un espacio de estados emplearemos los siguientes elementos:

- *Estados*, los cuales representan el espacio del problema, son estructuras de datos que nos informan la condición del problema en cada etapa del proceso de solución.
- *Estado inicial*.
- *Operadores o reglas de transformación*, los cuales hacen que el problema pase de un estado a otro.
- *Estado u estados objetivo*.

Al realizar la definición del problema, el conjunto de *estados* corresponde a todas las posibles situaciones o etapas del problema; y la solución a dicho problema estará dada por la secuencia de *operadores* que nos llevan de un *estado inicial* a un *estado terminal u objetivo*.

Para representar *el espacio de estados* se emplea comúnmente un diagrama de árbol, en el que cada nodo tiene a lo sumo un predecesor. Hay un nodo que no lo tiene y se llama *raíz o inicio*. Los nodos que no tienen sucesores son llamados *terminales* y los arcos orientados que unen los nodos entre sí, reciben el nombre de *ramas*.

En el diagrama de árbol los nodos nos indican los *estados* que se van generando y las ramas representan la aplicación de un *operador* el cual nos generará un nuevo estado. Por ejemplo, si al estado 1 le aplicamos tres operadores diferentes para obtener los estados 2, 3 o 4. Tales nodos reciben el nombre de nodos sucesores del nodo 1. Observe la figura 3.1

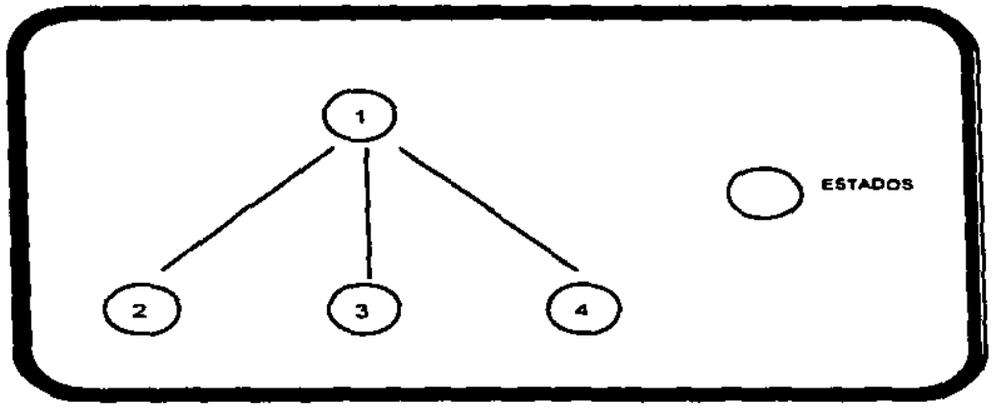


FIG. 3.1.

En la gráfica, una solución a un problema en el espacio de estados es el camino de nuestro nodo inicial hacia a un nodo objetivo (aquel nodo que contiene alguna solución deseada).

En la figura 3.2, una solución podría ser la aplicación del operador B dos veces seguido del operador D, para obtener el nodo objetivo o estado final. Puede haber otros estados finales y múltiples rutas para llegar a un estado final en particular.

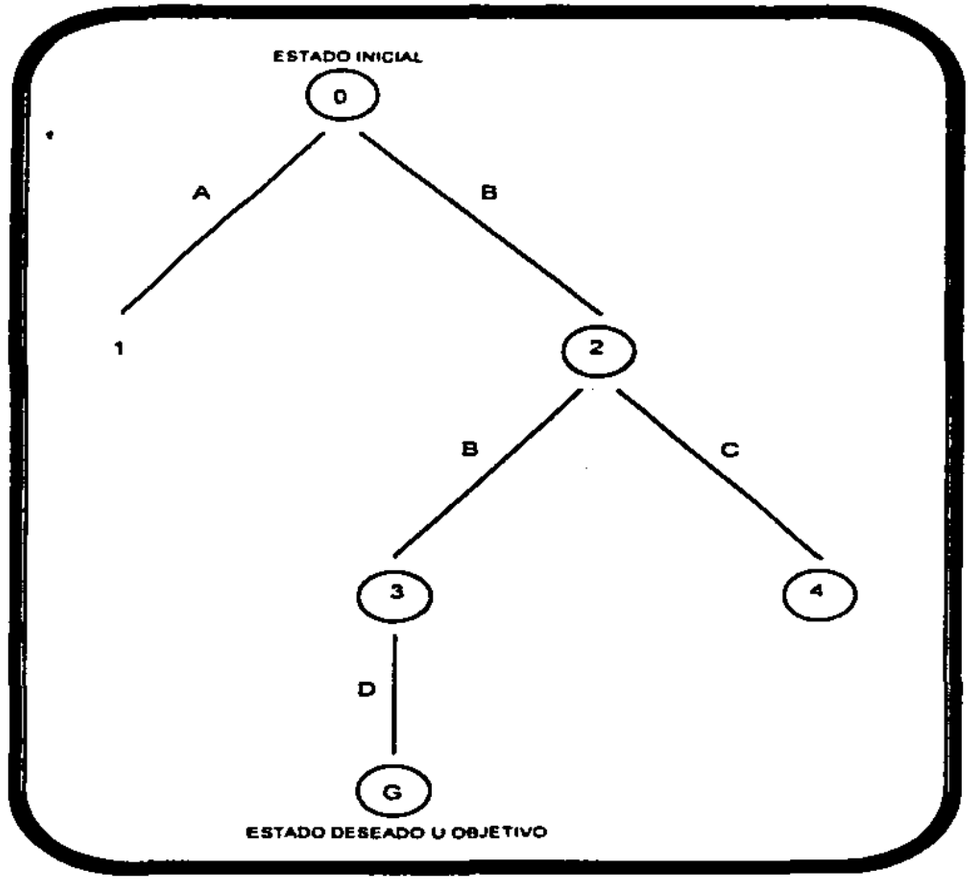


FIG. 3.2.

La características de este método de representación son:

1) Sus elementos se representan mediante una gráfica de árbol

2) Emplea el razonamiento progresivo,

3) Los operadores siempre toman una sola entrada y dan como resultado exactamente una salida.

Para entender mucho mejor los principios de este tipo de representación, abordaremos un ejemplo que aunque sencillo, se presta extraordinariamente para explicar esta técnica. Se trata del clásico juego llamado "8-puzzle", el cual consiste de un cuadro dividido en 9 subcuadros de igual tamaño, ocho de los cuales están numerados del 1 al 8 y el noveno subcuadro se encuentra vacío. Observe la figura 3.3 (A).

2	1	6
4		8
7	5	3

FIG. 3.3(A)

Los subcuadros se pueden deslizar horizontal o verticalmente a través del espacio que deja el noveno subcuadro. El problema consiste en mover los cuadros numerados a fin de conseguir alguna configuración deseada. Observe la figura 3.3 (B).

1	2	3
8		4
7	6	5

FIG. 3.3(B)

Para este problema, un estado es una configuración específica de subcuadros; por tanto, cada estado podría estar representado mediante una matriz de 3x3. Obviamente los operadores serían los movimientos permitidos a los subcuadros, estos podrían definirse separadamente para cada subcuadro. Sin embargo, es posible hacer una definición más inteligente si tomamos al subcuadro vacío como el objeto a ser movido y definimos los operadores en términos de los movimientos permitidos a tal subcuadro. En este caso tendremos sólo cuatro *operadores*:

⇒ARRIBA	El subcuadro vacío se moverá una posición hacia arriba.
⇒ABAJO	El subcuadro vacío se moverá una posición hacia abajo.
⇒IZQUIERDA	El subcuadro vacío se moverá una posición a la izquierda.
⇒DERECHA	El subcuadro vacío se moverá una posición a la derecha.

Algo importante de tomar en cuenta, es el hecho, de que en ocasiones un operador puede ser inaplicable, como ejemplo, observemos el caso de que el subcuadro vacío estuviese en una orilla del cuadro (fig. 3.3(C)) y aplicásemos un operador que causará que el subcuadro saliese de los límites del cuadro que lo contenía en un principio. Esto causaría que nos saliésemos de nuestro espacio de estados permitidos, que para este ejemplo siempre deberá corresponder con una matriz de 3X3.

7	2	
4	1	8
5	3	6

FIG. 3.3(C)

De lo anterior se concluye que otro característica del espacio de estados de un problema, es que debe tomar en cuenta los límites del problema, es decir, no debe salirse de su ámbito. El "8-puzzle", por ejemplo, tiene un espacio de estados de 9!/2 . Por tanto existen 9! posibles configuraciones de subcuadros, pero únicamente la mitad de este número esta permitida por las especificaciones del problema.

Por último es importante mencionar que en ocasiones la gráfica que representa el espacio de estados es demasiado grande; por lo que se conviene en generar sólo la parte de la gráfica, la cual, contenga la ruta de la solución deseada.

3.2.2. REPRESENTACION REDUCIDA

La representación reducida emplea los mismos elementos que la representación en espacio de estados. pero se distingue de esta en los siguientes puntos:

1) Sus elementos se representan mediante gráficas AND/OR
2) El Empleo del razonamiento regresivo,
3) Los operadores toman una entrada, pero pueden producir múltiples salidas.

En este método de representación las principales estructuras de datos están dadas por descripciones del problema u objetivos. Al principio se da la descripción de un problema; este es resuelto a través de una serie de transformaciones que al final convierten dicho problema en un conjunto de subproblemas cuyas soluciones son inmediatas. Las transformaciones o reglas permitidas se definen como "operadores". Un operador puede transformar un problema en varios subproblemas; los cuales deben ser resueltos para así determinar la solución del problema que los generó. En ocasiones, podemos aplicar varios y diferentes operadores a un mismo problema, o el mismo operador puede ser aplicado de diferentes formas. En este caso, es suficiente con resolver los subproblemas producidos por cualquiera de los operadores aplicados Un problema cuya solución es inmediata se define como "problema primitivo". Así, la representación de un problema mediante este método se define a través de los siguientes puntos.

- Una descripción inicial del problema.
- Un conjunto de operadores para transformar el problema en subproblemas.
- Un conjunto de descripción de problemas primitivos que proceden del objetivo inicial hacia atrás (regresivamente).

EJEMPLO.

Un ejemplo que se presta perfectamente para ejemplificar este tipo de representación, es el famoso problema de la torre de Hanoi. Imaginémonos que tenemos tres discos A, B, y C de diferentes tamaños, tenemos además tres postes o varillas 1, 2, y 3. Inicialmente los discos están apilados sobre el poste 1, con A que es el más pequeño, hasta arriba y C, que es el más grande, en la parte inferior. El problema consiste en pasar el apilamiento del poste 1 al poste 3, como se muestra en la figura 3.4, tomando en cuenta las siguientes restricciones.

A) Solo podemos desplazar un disco en cada oportunidad.

B) No podemos colocar un disco de mayor tamaño sobre otro más pequeño.

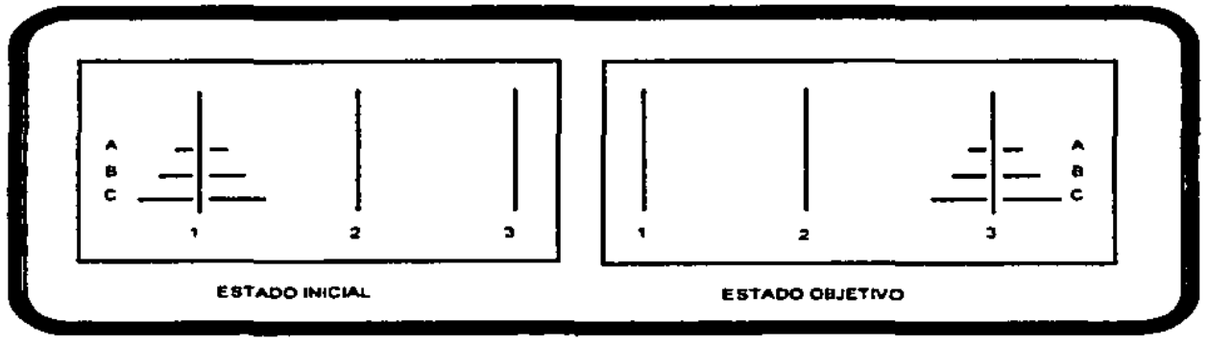


FIG. 3.4.

Solo necesitamos usar un operador en la solución: definiremos los postes como i, j, y k, así el problema de mover una pila de discos de tamaño $n > 1$ del poste i al poste k puede ser reemplazado por los siguientes tres subproblemas:

1. Mover una pila de tamaño n-1 de i a j.
2. Mover una pila de tamaño 1 de i a k.
3. Mover una pila de tamaño n-1 de j a k.

De aquí, observamos que el único problema primitivo es el de mover un disco de un poste a otro, procurando que el disco más pequeño sea colocado en la parte superior del poste. Si se encuentra un disco más pequeño que el ya colocado, este problema podría no tener solución (en vista de la definición del único operador posible).

Ahora podemos especificar cada descripción del problema mediante el tamaño n de la pila a ser movida, el número del poste origen y el número del poste destino. El problema original, de mover una pila de tres discos del poste 1 al poste 3, podría entonces ser representado como:

(n=3, 1 a 3),

Y la transformación del problema original a problemas primitivos (subproblemas con solución inmediata) puede ser representada por una gráfica de árbol, como se muestra en la siguiente figura.

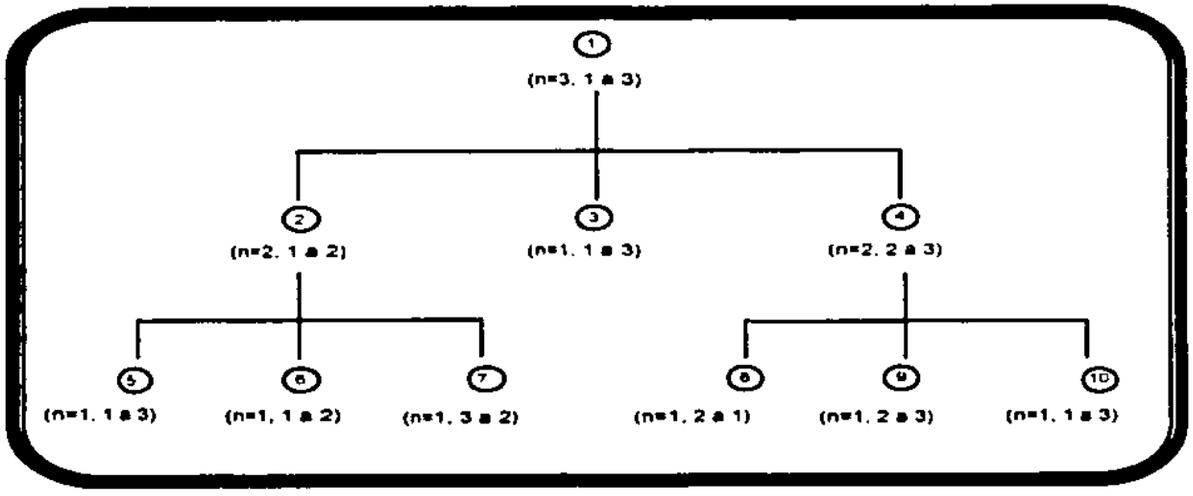


FIG. 3.5.

En la representación gráfica observamos que los nodos 2 y 4 nos conducen a dos subproblemas:

- El nodo 2 representa el subproblema de mover una pila de dos discos del poste 1 al poste 2. Este subproblema se resuelve expandiendo este nodo en los nodos 5, 6 y 7 que nos representan problemas primitivos, los cuales son solucionados mediante el movimiento del disco más pequeño al poste 3, moviendo el disco mediano al poste 2, y finalmente colocando el disco pequeño arriba del disco mediano.

- En el nodo 4 tenemos un subproblema similar el cual se resuelve a través de los nodos 8, 9 y 10.

El nodo 3 por tener solución inmediata (problema primitivo) carece de nodos descendientes. El problema se resuelve siguiendo la gráfica de abajo hacia arriba y de izquierda a derecha.
Recuerde que los nodos sin descendencia son llamados nodos terminales.

La secuencia de operadores a ser aplicada se debe distinguir de la secuencia de acciones a ser tomadas para lograr el objetivo. En el ejemplo de la torre de Hanoi, las acciones son los movimientos actuales de los discos. Esta secuencia está dada por los nodos terminales del árbol, leídos de izquierda a derecha. Y dependiendo del dominio del problema son considerados o no para su solución.

3.2.2.1. GRAFICAS AND/OR.

Para representar toda la variedad de situaciones posibles que pueden ocurrir en una reducción del problema, debemos hacer una generalización a la notación de gráfica de árbol. Tal notación generalizada es llamada gráfica AND/OR. Según Nilsson, una gráfica AND/OR se construye de acuerdo a las siguientes reglas:

1. Cada nodo representa, ya sea, un problema o un conjunto de subproblemas a ser resueltos.
2. Un nodo que represente un problema primitivo, será un nodo terminal y no tendrá descendientes.

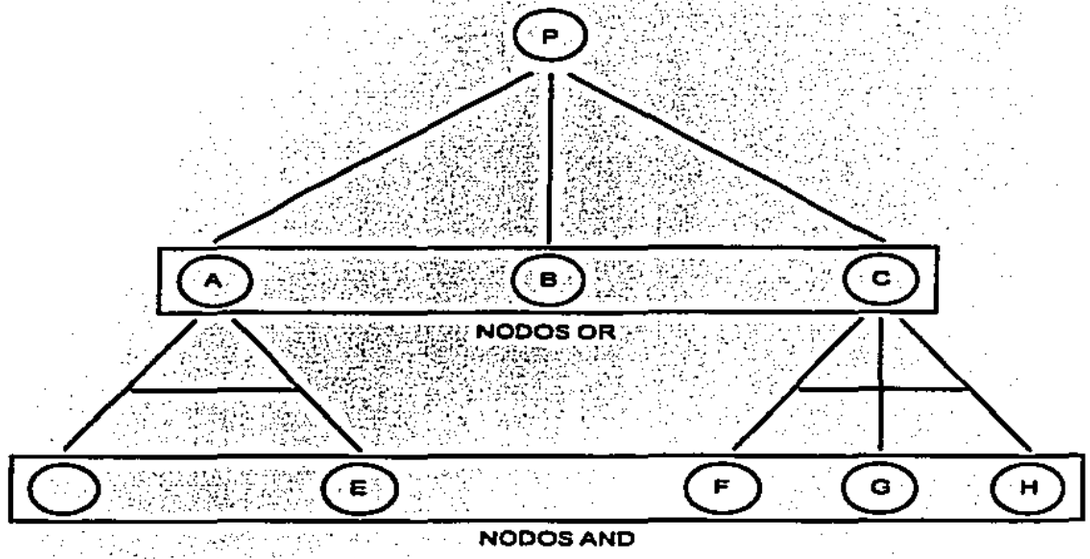


FIG. 3.6.

3. Para cada posible aplicación de un operador al problema P, que lo transforme en un conjunto de subproblemas, habrá una rama dirigida de P al nodo que represente el resultado de tal transformación. Por ejemplo la figura 3.6 ilustra la reducción de P en tres diferentes subproblemas: A,B y C. Primero debemos resolver cualquiera de los nodos A,B o C para que P pueda ser resuelto, A,B y C son llamados nodos OR.

4. La misma figura nos muestra la composición de los subproblemas A;B y C. Aquí, A={D,E}, B por si mismo es un solo problema, y C={F,G,H}. En general, para cada nodo que se expande uno o más subproblemas, existen ramas que van del nodo a cada uno de sus respectivos subproblemas (que también están representados por nodos). El conjunto de subproblemas se resolverá hasta que todos sus miembros sean resueltos. Los nodos de estos subproblemas son llamados nodos AND. Para distinguirlos de los nodos OR, las ramas que nos llevan hacia un nodo *sucesor* AND de un mismo nodo padre están unidas por una línea horizontal.

5. Una simplificación producida por las reglas 3 y 4 puede ser hecha en el caso especial en donde únicamente se puede aplicar un operador al problema P y este operador produce un conjunto de uno o más subproblemas. Como se muestra en la figura 3.7, el nodo OR intermedio que nos origina el conjunto de subproblemas puede ser omitido.

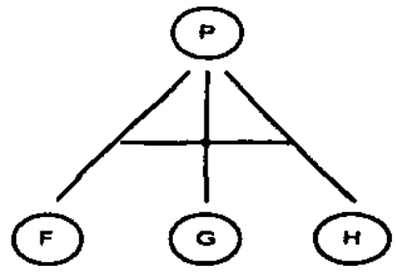


FIG. 3.7.

Resumiendo: cada posible aplicación de un operador a un nodo *n* estará representada por una rama que partirá del nodo *n* hacia un nodo sucesor (o varios nodos sucesores dependiendo de los operadores aplicados), este sucesor será definido como nodo OR y bastara con resolver uno de estos nodos sucesores (nodos OR), para dar solución al nodo que lo genero. Cada nodo OR (sucesor de n) representa un conjunto de subproblemas. Si el conjunto de subproblemas representados por un nodo OR *m* tienen sucesores, entonces habrá ramas dirigidas del nodo *m* a aquellos nodos que nos representen sus respectivos subproblemas. Estos nodos sucesores de *m* serán llamados nodos AND, esto a razón de que será necesario resolver todos estos nodos para dar solución al problema que representa el nodo *m*. Para distinguir los nodos AND de los OR, las ramas en la gráfica que van de *m* a sus sucesores AND estarán unidos por una línea horizontal.

En las figuras mostradas, cada nodo representa un problema distinto o un conjunto de problemas. Cada nodo excepto el nodo inicial tiene un nodo padre u origen, las gráficas son de hecho *árboles* AND/OR. Como una variación sobre la gráfica de la figura 3.6, asumamos que el problema A se puede reducir a D y E; y el problema C, a E, G y H. Entonces E puede ser representado ya sea, por dos nodos distintos o por un solo nodo como se muestra en la figura 3.8.

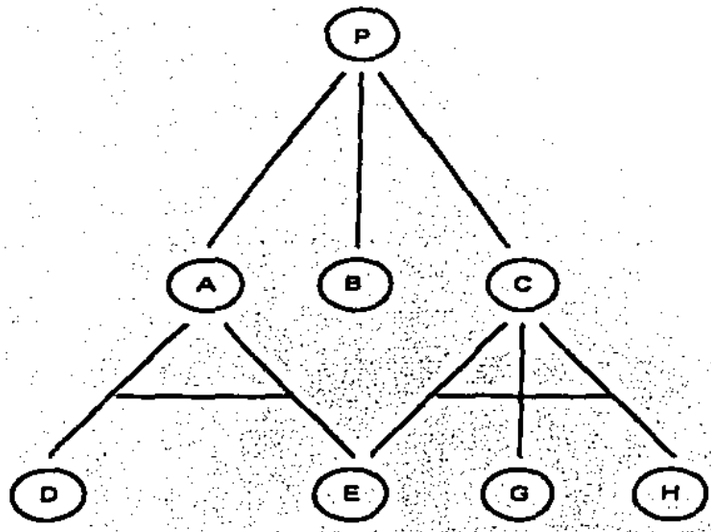


FIG. 3.8.

Es importante mencionar que no es necesario que la gráfica nos de la descripción completa del espacio de búsqueda. Para encontrar la solución de un problema inicial, solo necesitamos construir la gráfica necesaria para demostrar que el nodo de inicio puede ser resuelto. Tal subgráfica es llamada *gráfica de solución* o en el más estricto de los casos de un árbol AND/OR, un *árbol de solución*. Para ambos casos debemos aplicar las siguientes reglas:

Un nodo tiene solución si,

1. Este es un nodo terminal (problema primitivo),
2. Este es un nodo no terminal que tiene como sucesores nodos AND y todos estos se pueden resolver, o
3. Este es un nodo no terminal que tiene como sucesores nodos OR y al menos uno de ellos puede ser resuelto.

Por ejemplo observemos la gráfica de la figura 3.9, los nodos 5, 6, 8, 9, 10, y 11 son nodos terminales, existen tres posibles subgráficas de solución {1,2,4,8,9}, {1,3,5,6,7,10} y {1,3,5,6,7,11}.

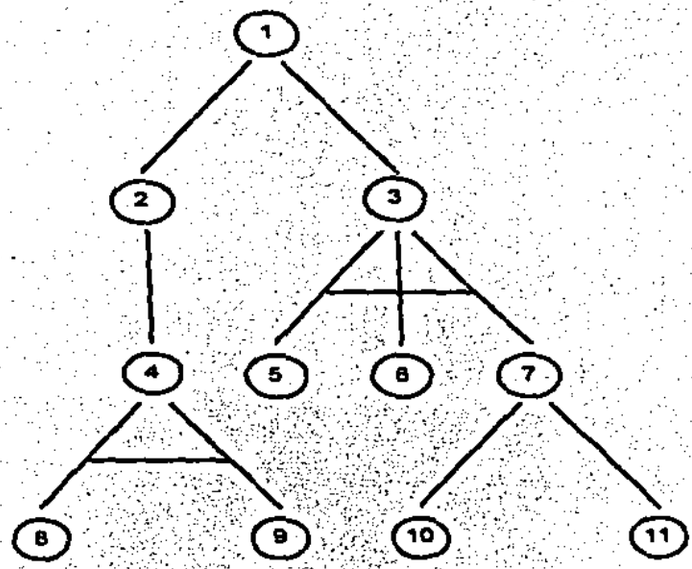


FIG. 3.9.

Contrariamente, un nodo carece de solución si,

1. Este es un nodo no terminal sin sucesores (Un problema no primitivo para el cual no se aplican operadores),
2. Este es un nodo no terminal con nodos sucesores AND y al menos uno de ellos carece de solución, o

3. Este es un nodo no terminal con nodos sucesores OR y todos ellos carecen de solución.

Observando la misma figura (3.9), el nodo 1 podría ser insoluble si todos los nodos en cualquiera de los siguientes conjuntos fueran insolubles {8,5}, {8,6}, {8,10,11}, {9,5}, {9,6}, {9,10,11}.

3.3. METODOS DE BUSQUEDA

3.3.1. EN EL ESPACIO DE ESTADOS

En esta sección, explicaremos algunos métodos de búsqueda para la representación en el espacio de estados, estos métodos asumen que, siempre habrá un procedimiento para encontrar todos los nodos sucesores a un nodo dado. Esto se logrará a través de la aplicación sucesiva de operadores a dicho nodo. Tal procedimiento es llamado expansión del nodo dado.

Además los algoritmos que describiremos toman en cuenta dos puntos:

1) El espacio de estados es una gráfica de árbol. Esto implica que solo hay un nodo inicial (el raíz) y el camino de este nodo a cualquier otro es único.

2) Cuando se expande un nodo creando un nodo para cada sucesor, el nodo sucesor siempre contiene apuntadores al nodo que le antecede. Esto hace posible el trazo de la ruta solución, cuando se genera un nodo objetivo.

3.3.1.1. BUSQUEDA DEL PRIMERO EN PROFUNDIDAD

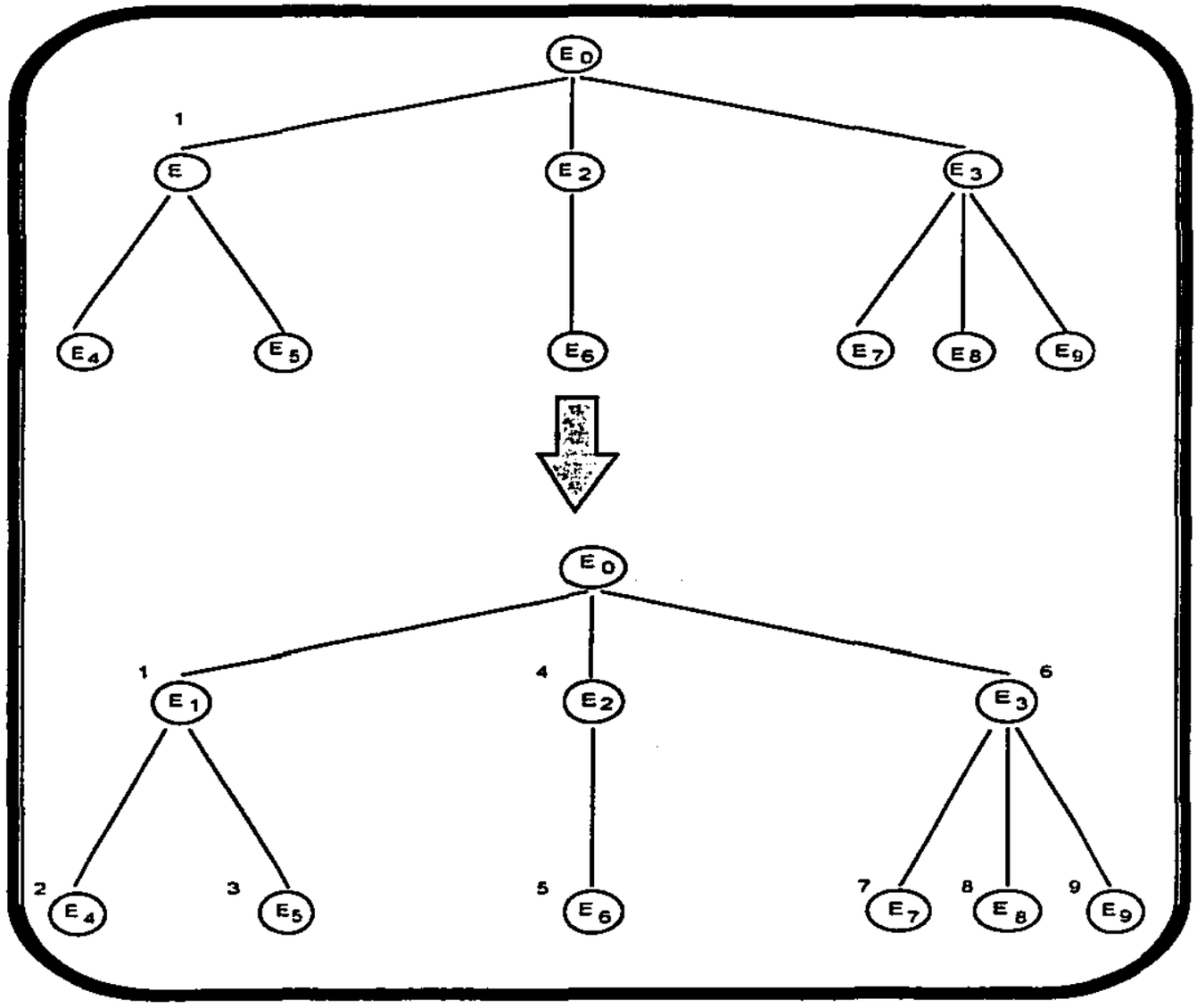


FIG. 3.10.

Esta búsqueda se caracteriza por la expansión del nodo más recientemente

generado o más profundo. Es decir, emplea una estructura tipo pila para realizar el análisis de sus nodos. La figura 3.10. muestra el orden en que se expanden los nodos de una gráfica dada.

La profundidad de un nodo en una gráfica se define mediante los siguientes puntos:

1) La profundidad del nodo inicial es cero.
2) La profundidad de cualquier nodo es un nivel más grande que la de el nodo que lo antecede.

Esta búsqueda sigue un camino a través del espacio de estados que va del nodo de inicio hacia abajo (orienta el análisis de padres a hijos). Si encuentra un estado que no tenga sucesores, abandona la búsqueda dentro de esa rama del árbol y continua con otra.

Para muchos problemas la gráfica de espacio de estados puede tener una profundidad infinita. Esto se previene asumiendo un límite de profundidad para los nodos a ser expandidos y considerar cualquier nodo que llegue a ese límite como un nodo sin sucesores. Es importante hacer notar que al utilizar un límite de profundidad, la ruta encontrada no será necesariamente la más corta.

A continuación describimos el algoritmo para este método utilizando un límite de profundidad.

1) Colocar el nodo inicial n en la pila PROSPECTOS, que contendrá los nodos a analizar, si este es un nodo objetivo, entonces la solución ha sido encontrada.
2) Si PROSPECTOS esta vacío, entonces no existe solución.
3) Sacar el nodo, n, de PROSPECTOS y colocarlo en la lista TRANSFORMADOS de nodos a expander.

4) Si la profundidad del nodo n es igual al limite de profundidad ir al paso 2.

5) Expander el nodo n, si este no tiene sucesores, ir al paso 2.

6) Colocar todos los nodos sucesores de n en PROSPECTOS.

7) Si cualquiera de los nodos sucesores de n es un nodo objetivo, la solución ha sido encontrada. De otra manera ir al paso 2.

EJEMPLO.

Considere el siguiente problema:

Se requiere mover un peón a través de la matriz de la figura 3.11A de arriba hacia abajo. El peón puede entrar a la matriz de cualquier manera en el renglón superior. Pero para sus movimientos debe tomar en cuenta las siguientes restricciones: De un cuadro que contenga 0, el peón debe moverse descendentemente si el cuadro que sigue también contiene cero; de otra forma, debe moverse horizontalmente. De un cuadro que contenga 1, no hay mas movimientos posibles. El objetivo es llegar a un cuadro que contenga cero en el ultimo renglón. Se asume un límite de profundidad de 5.

La gráfica generada por el algoritmo del primero en profundidad se muestra en la figura 3.11B. En el nodo E_0 el peón no ha entrado en la matriz. En los otros nodos, su posición se da con las coordenadas (renglón, columna), la numeración de los nodos nos da el orden en el cual estos salen de la pila PROSPECTOS. Cuando el algoritmo termina PROSPECTOS contiene a los nodos E_{17} (que es un nodo objetivo) y E_{18}; todos los otros nodos se encuentran en la lista TRANSFORMADOS. El camino de la solución encontrada se muestra en la gráfica. Ya que el algoritmo trata el espacio de estados como un árbol, y no como una gráfica general, no se percata de que los nodos E_2 y E_9 representan el mismo estado. Por consecuencia, la búsqueda descendente de E_9 duplica el trabajo hecho en E_2.

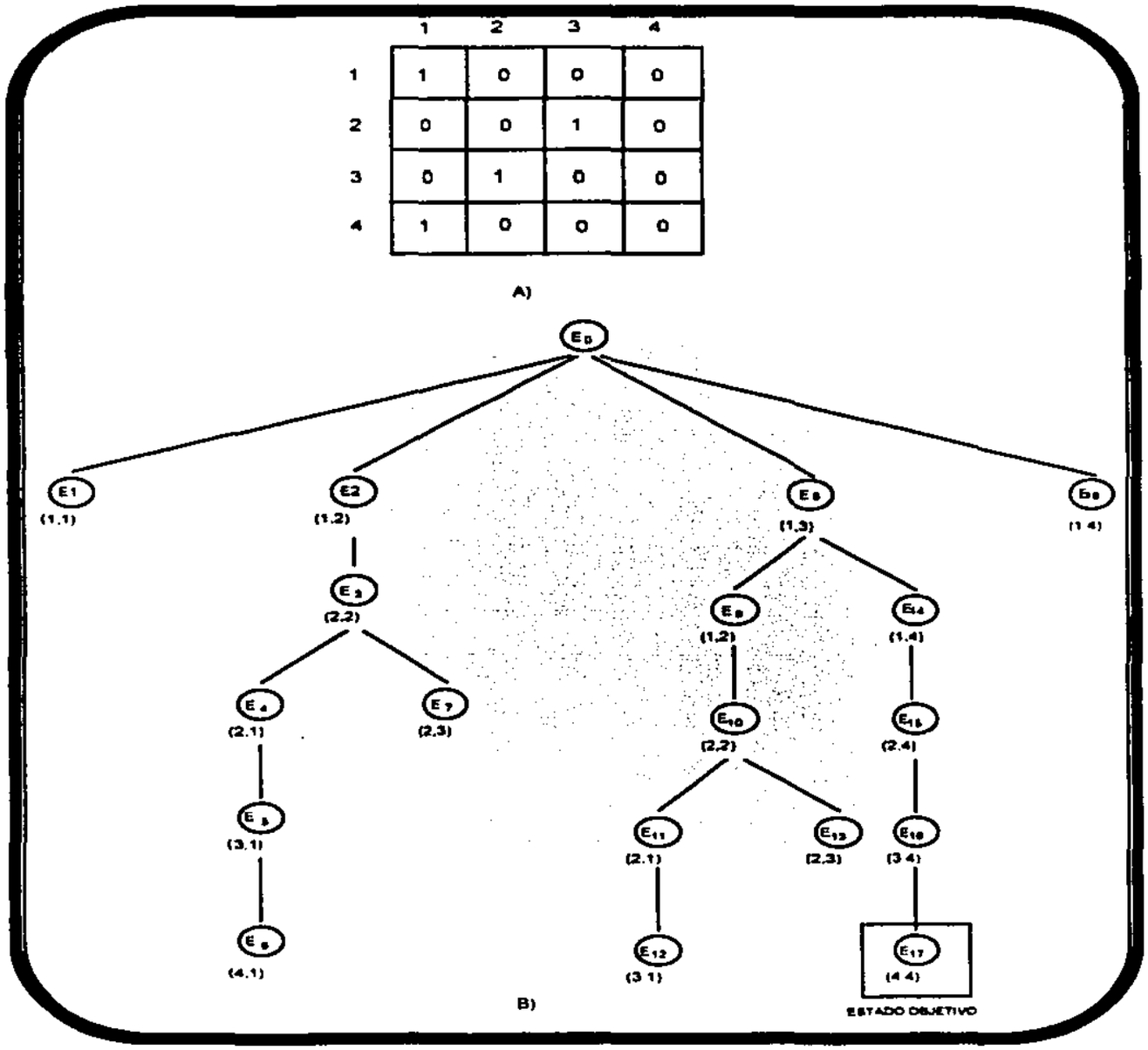

1 2 3 4
1 1 0 0 0
2 0 0 1 0
3 0 1 0 0
4 1 0 0 0
A)
E0
E1 (1,1)
E2 (1,2)
E3 (1,3)
E4 (1 4)
E5 (2,2)
E6 (1,2)
E14 (1,4)
E4 (2,1)
E7 (2,3)
E10 (2,2)
E15 (2,4)
E8 (3,1)
E11 (2,1)
E13 (2,3)
E16 (3 4)
E9 (4,1)
E12 (3 1)
E17 (4 4)
ESTADO OBJETIVO
B)

FIG. 3.11

3.3.1.2. BUSQUEDA DEL PRIMERO EN ANCHURA

Este método expande nodos tomando en cuenta su proximidad al nodo origen. Se considera cada posible secuencia de operadores de profundidad n antes de una de profundidad n+1, y cuando la búsqueda es muy larga, se garantiza que la solución encontrada representará la ruta más corta entre otras soluciones. La figura 3.12 muestra el orden de expansión para una gráfica dada.

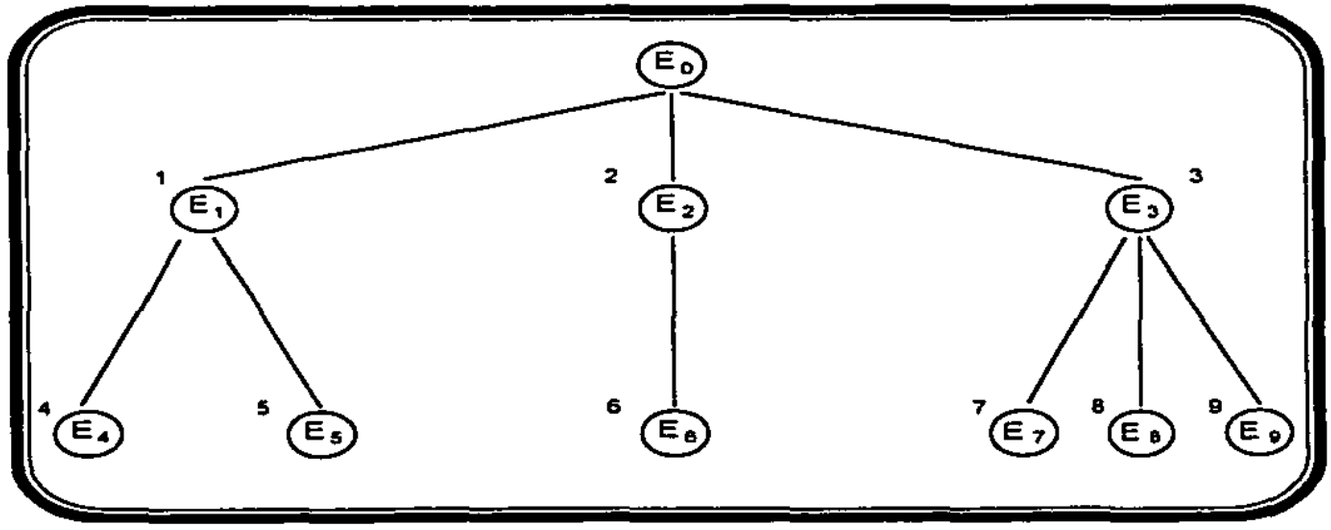


FIG. 3.12.

Esta técnica de búsqueda es opuesta a la del primero en profundidad, pues comprueba cada nodo sobre el mismo nivel antes de proceder a analizar un nivel más profundo. Es decir, emplea para su análisis una estructura tipo cola. En el final de la cola se van añadiendo los nuevos elementos a analizar y se van retirando los que hay al principio.

El siguiente algoritmo describe este tipo de búsqueda:

1) Poner el nodo origen en la cola PROSPECTOS de nodos sin expander si este representa un nodo objetivo, la solución ha sido encontrada.

2) Si PROSPECTOS esta vacío, no existe solución.

3) Sacar el primer nodo, n, de PROSPECTOS y colocarlo en la lista TRANSFORMADOS de nodos a expander.

4) Expander el nodo n, si este no tiene sucesores ir al paso 2.

5) Poner todos los nodos sucesores de n en PROSPECTOS.

6) Si cualquiera de los nodos sucesores es un nodo objetivo, la solución ha sido encontrada, de otra manera ir al paso 2.

EJEMPLO

Considere un arreglo de tres bloques colocados sobre una mesa, el estado inicial y el estado objetivo se muestran en la figura 3.13A. El único operador es MUEVE X a Y, el cual mueve un bloque X sobre otro bloque Y.

Como condiciones tenemos:

- X, el objeto a ser movido, no tenga objetos sobre el,
- Si es Y el bloque tampoco deberá tener nada sobre el.

En la figura 3.13B se muestra la gráfica que muestra la búsqueda generada por el algoritmo. Los estados están representados por los nodos E_0 a E_{10}. Como ejemplo tenemos al nodo E_1, el cual representa al estado sucesor logrado por el movimiento del bloque 1 sobre la mesa.

Los nodos son generados y expandidos en el orden dado por sus números, esto es, E_0, E_1, ... , E_{10}. Al terminar el algoritmo, encuentra E_{10} como su nodo objetivo, la lista TRANSFORMADOS contiene de E_0 a E_5 y la cola PROSPECTOS contiene de E_6 a E_{10}.

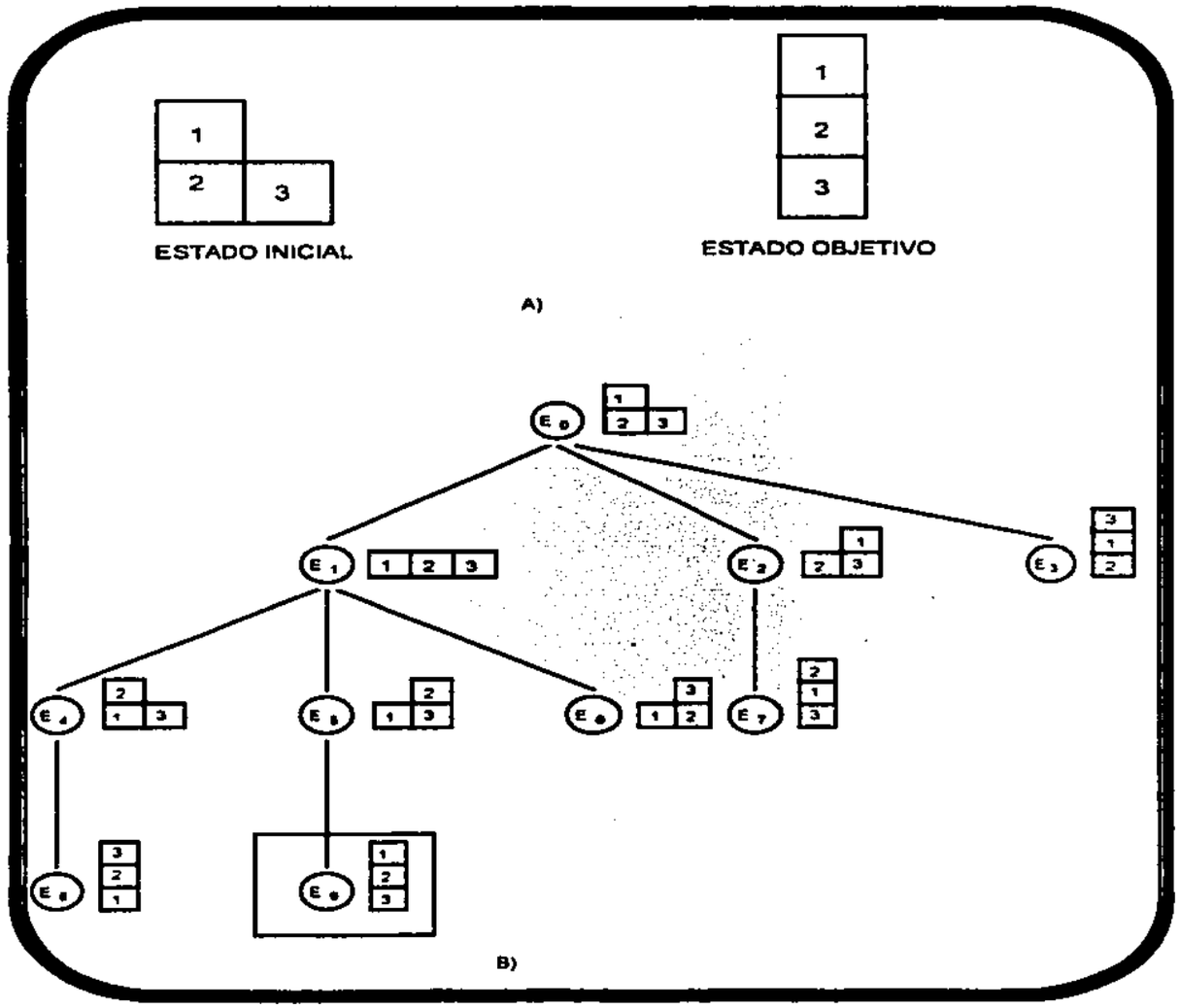


FIG. 3.13.

3.3.2. EN LA REPRESENTACION REDUCIDA

Para que un problema sea resuelto mediante este método debemos especificar un nodo de inicio (que nos representará el objetivo inicial o la descripción del problema), un conjunto de nodos terminales (descripción de problemas primitivos) y un conjunto de operadores para reducir los objetivos a subobjetivos.

Los algoritmos de búsqueda que se describirán toman en cuenta los siguientes dos puntos:

a) El espacio de búsqueda es una gráfica de árbol AND/OR, y

b) cuando un problema se transforma a un conjunto de subproblemas, estos se pueden resolver en cualquier orden.

3.3.2.1. BUSQUEDA DEL PRIMERO EN ANCHURA EN UNA GRAFICA AND/OR.

El siguiente algoritmo describe la búsqueda del primero en anchura en una gráfica de árbol AND/OR. Si existe una solución, este algoritmo encuentra aquella de menor profundidad, previendo que los nodos OR intermedios son ignorados para el cálculo de la profundidad en la gráfica. Se asume que el nodo de inicio no es un nodo terminal.

1. Poner el nodo de inicio en una cola de nodos sin expander, la cual llamaremos PROSPECTOS.
2. Remover el primer nodo, n, de PROSPECTOS.

3. Expander el nodo n, para generar todos sus sucesores a los cuales llamaremos m. Para cada sucesor m, si m representa un subconjunto de más de un subproblema, generar los sucesores de m correspondientes a cada subproblema. Enlazar mediante un apuntador a cada nodo que se va generando con su correspondiente antecesor. Colocar todos los nuevos nodos que no tengan descendentes en la cola PROSPECTOS.

4. Si no se generaron sucesores en el paso 3, entonces
   a) Etiquetar el nodo como insoluble.
   b) Si esta insolubilidad hace que sus antecesores sean insolubles, etiquetar aquellos también como insolubles.
   c) Si el nodo de inicio esta etiquetado como insoluble, el problema no tiene solución mediante este método
   d) Remover de PROSPECTOS los nodos que tengan antecesor insoluble

5. De otra manera, si se generaron nodos terminales en 3, entonces
   a) Etiquetar estos nodos como solucionables.
   b) Si la solución de estos nodos terminales hace que cualquiera de sus antecesores sea resuelto, etiquetar estos antecesores como solucionables.
   c) Si el nodo inicial esta etiquetado como solucionado, el problema ha sido resuelto.
   d) Remover de PROSPECTOS los nodos etiquetados como solucionables o aquellos que tengan antecesores solucionables.

6. Ir al paso 2.

3.3.2.2. BUSQUEDA DEL PRIMERO EN PROFUNDIDAD EN UNA GRAFICA AND/OR.

Podemos obtener el algoritmo para esta técnica de búsqueda, con un límite de profundidad, solo con modificar el paso 3 de el algoritmo anterior de la siguiente manera:

3. Si la profundidad de n es menor que el límite de profundidad, entonces:

4. Expander el nodo n, para generar todos sus sucesores a los cuales llamaremos m. Para cada sucesor m, si m representa un conjunto de más de un subproblema, generar los sucesores de m correspondientes a cada subproblema. Enlazar mediante un apuntador a cada nodo que se va generando con su antecesor inmediato. Colocar todos los nuevos nodos que no tengan descendentes en la pila PROSPECTOS.

5. (Recuerde que para esta técnica prospectos pasa a ser una estructura tipo pila).

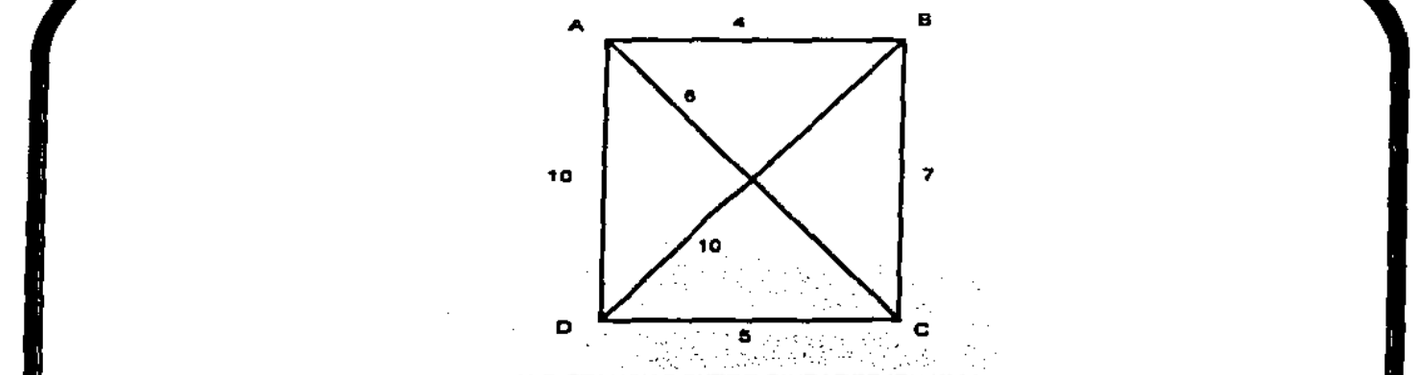
3.3.3. BUSQUEDA ORDENADA

Los métodos de búsqueda vistos hasta el momento, se conocen como métodos de búsqueda a ciegas ya que el orden en que se examinan los nodos esta determinado por la estrategia de búsqueda y no por alguna propiedad o característica de los estados.

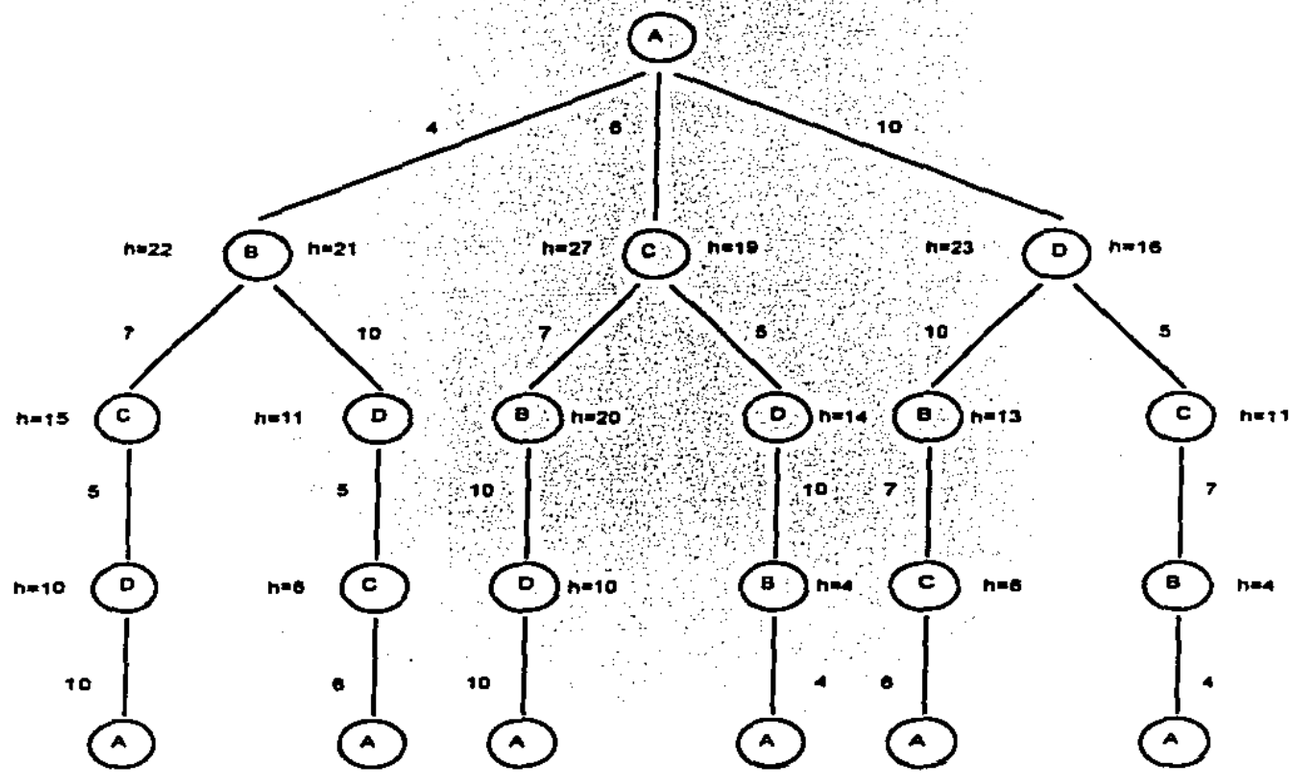
Al utilizar un método de búsqueda ordenada sucede lo contrario ya que aprovechamos propiedades o información heurística de los estados para determinar la dirección de la búsqueda (los nodos a expander).

Dentro de este método se define como costo de un camino en la gráfica de estados, el número de operaciones requeridas para llegar del nodo inicial al nodo objetivo o solución.

Para explicar esta técnica de búsqueda, abordaremos el siguiente ejemplo, imaginemos el caso de un agente viajero que se encuentra ante el siguiente dilema: Dado un número de ciudades a visitar y la distancia entre cada una de ellas, encontrar la ruta más corta para empezar y finalizar en la ciudad A y visitar cada una de las otras ciudades exactamente una vez. En la figuras 3.14 se muestran las distancias entre las ciudades y la gráfica de espacio de estados correspondiente.



A) DISTANCIAS ENTRE CIUDADES (EN KM.)



B) GRAFICA DE ESPACIO DE ESTADOS CON INFORMACION HEURISTICA

FIG. 3.14.

En la gráfica, los nodos representan las ciudades y las ramas representan las distancias entre ellas. Todos los caminos que van del nodo raíz al nodo objetivo tienen un costo, que en este ejemplo esta dado por la suma de las distancias que constituyen la ruta que va del nodo inicial al nodo objetivo. Observe que se puede llegar al nodo objetivo (A) a través de diferentes rutas. Pero en este método, la ruta que nos interesa es aquella que tenga una distancia mínima o mejor dicho que posea un menor costo, tal costo se representará con la letra f.

Al analizar la gráfica de la figura 3.14, debemos decidir cual de los nodos expander para resolver el problema dado. En el ejemplo propuesto se puede deducir el camino solución; sin embargo, por lo general, al realizar una búsqueda no sabemos si un nodo dado se encuentra en el camino solución y, menos aún, si es el de menor costo. Por tanto no sabemos el valor exacto de f, pero lo podemos estimar si consideramos el valor de f dividido en dos partes:

1. En la primera tendremos que considerar el costo de la ruta desde el nodo inicial hacia un nodo específico. A este costo le llamaremos g.

2. En segundo lugar, tenemos que tomar en cuenta el costo de la ruta desde ese nodo hasta el nodo meta, al que simbolizaremos con h.

Por tanto el valor de f estará dado por la siguiente expresión:

$$f = g+h$$

Para calcular el valor de g debemos tomar en cuenta que el nodo raíz tiene g=0 y los demás nodos tienen una g con un valor igual a la distancia que representan (desde el nodo de inicio hasta su posición).

El valor de h se puede determinar en la gráfica de estados o mediante alguna regla heurística. Para el ejemplo propuesto, este valor esta representado por la distancia que hay del nodo objetivo al nodo que se esta analizando.

Calculados los valores de f, podemos comenzar la construcción de la gráfica solución, para lo cual emplearemos una estructura tipo pila donde introduciremos los nodos a expandir ordenados de mayor a menor valor de f. La figura 3.16., explica la secuencia seguida para la solución del problema propuesto.

1) introducimos el nodo inicial A en la pila
2) Sacamos el nodo (A) de la pila lo expandimos, calculamos los valores de f para los sucesores de tal nodo. Note que como tenemos dos valores de h para cada sucesor de A, hubo necesidad de calcular un valor promedio.
3) Introducimos en la pila los sucesores de (A) ordenados de mayor a menor valor de f.
4) Sacamos B de la pila, lo expandimos y calculamos el valor f para sus sucesores.
5) Introducimos los sucesores de (B) en la pila ordenados de mayor a menor valor de f.
6) Como D entro al último lo sacamos de la pila, lo expandimos y calculamos el valor de f para su sucesor (C) que sigue siendo la menor.
7) Introducimos (C) en la pila.
8) Sacamos el último nodo de la pila lo expandimos y como su sucesor resulta ser el nodo solución la búsqueda termina.

La búsqueda realizada nos da una ruta (ABDCA) con una distancia de 25 kilómetros, la cual podemos asegurar que es la distancia más corta.

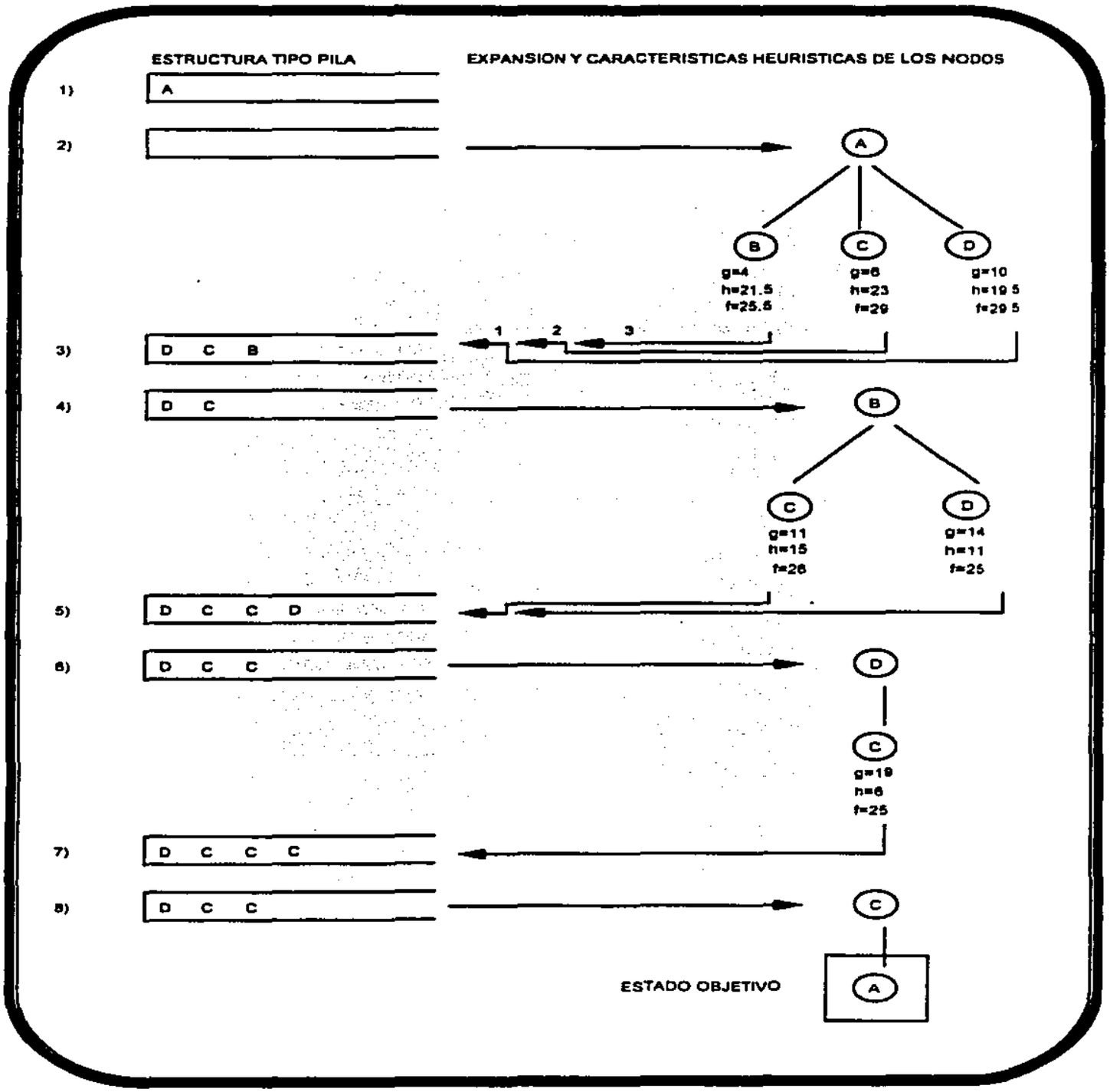

ESTRUCTURA TIPO PILA
EXPANSION Y CARACTERISTICAS HEURISTICAS DE LOS NODOS
1)
A
2)
A
B
C
D
g=4
h=21.5
f=25.5
g=6
h=23
f=29
g=10
h=19.5
f=29.5
1
2
3
3)
D C B
4)
D C
B
C
D
g=11
h=15
f=26
g=14
h=11
f=25
5)
D C C D
6)
D C C
D
C
g=19
h=6
f=25
7)
D C C C
8)
D C C
C
ESTADO OBJETIVO
A

FIGURA 3.16

4. REPRESENTACION DEL CONOCIMIENTO.

4.1. INTRODUCCION.

La investigación sobre inteligencia artificial comprende el desarrollo de sistemas computarizados que ejecuten tareas tales como hablar, analizar, jugar ajedrez, etc. Cuando hablamos sobre personas que pueden hacer estas cosas, asumimos que ellas tienen el *conocimiento* para hacerlas. En otras palabras, describimos el comportamiento inteligente en términos del conocimiento. Similarmente, consideramos que un programa de computadora que pueda jugar cartas, entender el lenguaje natural, o manipular un robot, posee cierto grado de conocimiento. Esto es, asumimos el conocimiento de un programa de la misma manera que lo asumimos para los humanos, basándonos en la observación de los conocimientos que se poseen. Por tanto, un programa capaz de realizar las tareas anteriormente mencionadas deberá, tener información sobre todos los objetos de la situación o problema a resolver (dominio), los eventos que pueden suceder, y como ejecutar tareas específicas de acuerdo a dichos eventos.

La naturaleza del conocimiento y la inteligencia han sido discutidos por psicólogos, filósofos, lingüistas, educadores y sociólogos por cientos de años. En IA, *la representación del conocimiento* se realiza a través de *estructuras de datos*, las cuales son manipuladas mediante *procedimientos interpretativos*, que permiten un comportamiento inteligente. Por tanto dentro de esta área ha sido esencial el diseño de varias clases de estructuras de datos para almacenamiento de información en programas de computadora y, el desarrollo de procedimientos de manipulación inteligentes para que esas estructuras de datos hagan inferencias.

Lo anterior se debe a que una estructura de datos por sí sola no es conocimiento. Podemos decir, metafóricamente, que un libro es una fuente de conocimiento, pero sin un lector, el libro es solo un papel.

Se ha desarrollado una gran variedad de esquemas para representar el conocimiento, en este capítulo se presentan los conceptos y características básicos de los esquemas más populares.

4.2. ESQUEMAS DE REPRESENTACION

4.2.1. LOGICA

La lógica matemática, y en particular la lógica simbólica ofreció el primer medio para representación del conocimiento, ya que fue estudiada desde tiempos de los antiguos griegos (Aristóteles 384-322 A. C.). Esta lógica simbólica es un componente básico dentro de la IA, tiene la ventaja de ser conocida por la mayoría de ingenieros y científicos relacionados con la computación. Pero lo más importante, es que la lógica simbólica facilita la representación y manipulación del conocimiento a través de estrategias que poseen bases matemáticas sólidas.

La lógica proporciona un medio formal para expresar declaraciones (proposiciones) relativas a hechos y objetos del mundo real en forma tal que se puedan hacer inferencias y confirmar o negar las declaraciones hechas. Observe el diagrama de la figura 4.1.

FIG. 4.1.

Para que una computadora sea capaz de razonar usando la lógica, se deben utilizar algunos métodos para convertir las declaraciones y procesos de razonamiento a formas que puedan ser manipuladas por una computadora. El resultado de este proceso se conoce como lógica simbólica. La cual es un sistema de reglas y procedimientos, que a partir de premisas y usando una

variedad de técnicas lógicas, permiten hacer inferencias. Esta lógica *computacional* se divide en dos partes:

- Lógica o cálculo proposicional y,
- Lógica o cálculo de predicados

que se complementan y conjuntamente brindan una herramienta poderosa para representación del conocimiento.

LOGICA PROPOSICIONAL

La lógica o cálculo proposicional es utilizada para expresar proposiciones ciertas o falsas y deducir nuevas expresiones de las proposiciones previamente definidas.

Una *proposición* es una sentencia que puede ser cierta, o falsa. Una vez que su valor esta definido tal proposición pasa a ser una *premisa*, la cual puede ser usada para deducir nuevas proposiciones o realizar otras inferencias. La veracidad o falsedad de la nueva proposición se determina a través del uso de *reglas*.

Dentro de la lógica proposicional se utilizan símbolos, tales como las letras del alfabeto para representar proposiciones, premisas, o conclusiones. Considere el siguiente ejemplo:

A = El hombre es mortal
B = Socrates es un hombre
C = Socrates es mortal

Estas simples proposiciones parecen no ser muy interesantes o útiles. En el mundo real las proposiciones envuelven mucho más interrelaciones. Para formar premisas más complejas, podemos combinar dos o más

proposiciones usando conectores lógicos. Estos conectores u operadores son los famosos AND, OR, NOT, IMPLICACION y EQUIVALENCIA cuyos símbolos se muestran en la tabla de la figura 4.2.A).

CONECTOR	SIMBOLO
CONJUNCION (AND)	$\wedge$.&. U
DISYUNCION (OR)	$\vee$ .U. +
NEGACION (NOT)	$\neg$.~
IMPLICACION	$\supset$ $\longrightarrow$
EQUIVALENCIA	$\equiv$

FIG. 4.2.(A)

La tabla de valores arrojados por estos conectores se muestra en la fig. 4.2.B). En tal figura se usan la variables A y B (Que en un caso práctico pueden representar proposiciones).

A	B	A $\wedge$ B	A $\vee$ B	A $\longrightarrow$ B	$\neg$ A	A $\equiv$ B
F	F	F	F	T	T	T
F	T	F	T	T	T	F
T	F	F	T	F	F	F
T	T	T	T	T	F	T

FIG. 4.2.(B)

El resultado de la mayoría de conectores es generalmente conocido. El conector que presenta alguna dificultad es el de implicación (->) el cual puede razonarse de la siguiente manera, A->B será verdadera si A es falso o B es verdadero, esto es NOT A, OR B.

Al relacionar varias proposiciones a través de conectores podemos generar un gran conjunto de premisas. La expresión simbólica resultante que parece más una formula matemática. Puede ser manipulada por medio de reglas de la lógica proposicional o del álgebra boleana para inferir nuevas conclusiones. La veracidad o falsedad de la conclusión puede determinarse usando las premisas generadas.

CALCULO DE PREDICADOS

Para los propósitos de IA, la lógica proposicional por si sola no es muy útil, ya que no solo se necesita saber si las proposiciones son ciertas o falsas si no que es necesario establecer relaciones entre esos objetos, y generalizar esas relaciones sobre clases de objetos. Para cubrir esos objetivos existe la lógica o cálculo de predicados.

Por tanto podemos decir que la lógica de predicados usa los mismos conceptos y reglas de la lógica proposicional, agregando la capacidad de postular relaciones entre los objetos definidos y de generalizar tales relaciones en clases. Esto permite representar el conocimiento en detalles más finos. Ya que es posible representar proposiciones sobre objetos específicos, separar una proposición dentro de objetos sobre los cuales podemos hacer aserciones, especificar un objeto y sus características, o realizar afirmaciones sobre tal objeto. Además, de que hace posible el uso de variables dentro de proposiciones lógicas simbólicas. El resultado es el esquema más poderoso para representación de conocimiento que ha sido más aplicada hasta el momento para resolver problemas prácticos en una computadora. Recordemos que esto es la base del lenguaje PROLOG.

Dentro del cálculo de predicados, una proposición se divide en dos partes, argumentos (u objetos) y predicados (o aserciones). Un ejemplo de una proposición sería:

son-vocales(a,e,i,o,u)

Donde la proposición esta formada por el predicado *son vocales*, y sus argumentos son las vocales del alfabeto.

Los argumentos son los objetos sobre los cuales se hacen aserciones. En oraciones del lenguaje ordinario, los objetos son los sustantivos que sirven como sujetos y objetos de la expresión.

Los predicados son las aserciones que se hacen sobre tales objetos y representan relaciones específicas entre los mismos. Un predicado se aplica a un número específico de argumentos y puede tener valores de cierto o falso cuando objetos individuales son usados como argumentos. Un ejemplo de predicado de un argumento es el predicado *es-roja* que tendrá un valor de verdadero cuando se aplique al objeto individual *esta manzana* y falso cuando se aplique a la expresión *esta naranja*. Otros ejemplos de predicados son *menor-que-cero, mortal, hecho-de-papel, es-una*, etc.

Pero los predicados pueden tener más de un argumento. Un caso es el predicado matemático *es-mayor-que*, y un ejemplo es el siguiente:

es-mayor-que(7,4).

Los objetos físicos pueden ser comparados por el predicado de dos partes *mas-brillante-que*, el cual necesitará dos argumentos de comparación.

Los predicados de una parte (con un solo argumento), define una clase o

colección. Esto es, por cada predicado de una parte P, todos los objetos individuales X pueden ser sorteados dentro de dos grupos distintos, uno con aquellos objetos que satisfagan P (para los cuales P(X) sea cierta) y otros con aquellos que no satisfagan P.

Una vez que el conocimiento esta organizado tanto en lógica proposicional como de predicados, este esta listo para que a través de él se realicen procesos de inferencia.

4.2.1.1. PROCESOS DE RAZONAMIENTO

Ya que la lógica nos permite expresar conocimientos y hechos de manera simbólica. Debemos usar ese conocimiento para realizar inferencias, responder cuestiones, razonar o llegar a una conclusión. Para esto necesitamos utilizar reglas de inferencia, y la mejor regla de inferencia conocida es la regla del *modus ponendo ponens.*

MODUS PONENDO.

De acuerdo a este procedimiento, si hay una regla "IF A, THEN B," y si sabemos que A es cierto, entonces podemos inferir que B también es cierto. Usando simbología lógica, esto se puede expresar de la siguiente manera:

[A AND (A–>B)]–>B

A y (A–>B) son proposiciones en una base de conocimientos (cognoscitiva). Dada esta expresión, podemos reemplazar ambas proposiciones con la proposición B. En otras palabras podemos usar modus ponendo para llegar a la conclusión que B es cierta si las dos primeras proposiciones se cumplen (son ciertas).

EJEMPLO:
A: Es un día caluroso
B: Iremos a la playa
A–>B: SI es un día caluroso,
ENTONCES iremos a la playa

La primer premisa expresa el hecho de que el día es caluroso, la segunda dice que iremos a la playa.
Además, A implica B. Esto es si ambos A y A implica B son ciertos, B es cierto. Usando modus ponendo puedes deducir que iremos a la playa.

Una situación diferente es la inferencia en el caso de que B sea conocido como falso. Esto es llamado Modus tollendo.

MODUS TOLLENDO

De acuerdo a este procedimiento, cuando se sabe que B es falsa, y si hay una regla "IF A, THEN B," es válido concluir que A también es falso.

RESOLUCION.

La resolución es un método para producir nuevas cláusulas de un conjunto inicial. La resolución es un potente e *indirecto* método de inferencia, y frecuentemente es mal entendido.

Frecuentemente la resolución es usada para mostrar la inconsistencia de conjuntos de cláusulas, en este sentido el proceso de resolución produce una contradicción lógica. Y aunque esto pueda parecer una vía negativa para la inferencia, es completamente útil y proporciona un conjunto de cláusulas generadas de una forma especial, como a continuación se describe:

Se puede probar la veracidad de una cláusula, en el contexto de un conjunto de cláusulas que se sabe son ciertas, agregando la negación de tal cláusula al conjunto y buscando una contradicción.

Si el proceso de resolución no encuentra una contradicción. La negación de lo que buscamos probar es *lógicamente consistente* con la base de datos, En consecuencia la cláusula probada no puede ser cierta.

De esta manera, la resolución es un medio de verificar la validez de declaraciones o proposiciones nuevas a través de la refutación o contradicción.

4.2.2. REDES SEMANTICAS

CONCEPTO Y ELEMENTOS.

Uno de los primeros esquemas para representar el conocimiento fue la llamada *red semántica*. La cual, es una notación gráfica que representa relaciones jerárquicas entre objetos, acciones o eventos. Una red consiste de los siguientes elementos:

> *NODOS* (círculos o cuadros) y,
> *CONECTORES* (flechas)

Los nodos representan elementos del dominio, como pueden ser objetos (elementos físicos), conceptos, eventos, o acciones. Se pueden representar gráficamente por medio de círculos o cuadros, y deben estar etiquetados con los nombres de los elementos que representan. Los conectores enlazan los nodos, y representan predicados o atributos que indican las relaciones entre los nodos a los cuales enlazan. Gráficamente se muestran como vectores dirigidos, que van de un nodo a otro; se etiquetan con el nombre de la relación que representan

Los objetos que se representan en una red pertenecen a un clase. Cada objeto es una instancia (caso o ejemplo) de una clase y todos los objetos de una clase comparten atributos (Generalmente con valores diferentes para esos atributos).

Las clases pueden ser subclases de otras clases. Una subclase es una especialización (división) de una clase.

Recordemos que un predicado es una función que evalúa una declaración como cierta o falsa. Por tanto las relaciones que representaremos en una red semántica serán relaciones binarias.

CONSTRUCCION Y CARACTERISTICAS.

Supongamos que deseamos representar por medio de una red el hecho: *todas las gaviotas son aves.* Podríamos hacer esto mediante dos nodos, designados gaviotas y aves los cuales fuesen unidos por un conector que indicará la relación que queremos establecer, Esto se muestra en la figura 4.3.(A).

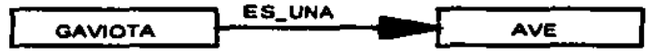


FIG. 4.3.(A)

El conector ES_UNA además podría indicarnos el hecho de que gaviotas es una subclase de la clase aves.

Si chava fuera una instancia de la subclase gaviotas y deseáramos insertarlo en la red, deberíamos agregar un nodo para chava como se muestra en la figura 4.3.B).

FIG. 4.3.(B)

Note que en este ejemplo representamos solo dos hechos, pero sería muy fácil deducir o mejor dicho inferir un tercero, esto es, que chava es una ave, simplemente siguiendo las conexiones ES UNA: *chava es una gaviota. Una gaviota es una ave. Por tanto, chava es una ave.* La facilidad con la cual es posible inferir información para realizar deducciones (herencia jerárquica), de este tipo, es la razón de la popularidad de las redes semánticas como una forma de representación del conocimiento. El principio de herencia en IA, y las inferencias que se pueden hacer en base al mismo, son similares a las conjeturas que hacemos sobre la clasificación de los organismos biológicos. Por ejemplo sabemos que los mamíferos tienen pelo y que los perros son mamíferos. Por tanto se puede inferir que los perros tienen pelo. Por tanto en problemas donde la mayor parte del razonamiento está basado sobre una compleja *taxonomía,* la red semántica será el esquema natural de representación.

Algunas veces la clasificación taxonómica, necesita representar conocimientos sobre propiedades de los objetos. Por ejemplo, podríamos desear representar el hecho de que los aves tienen alas, esto se logra a través de la red de la figura 4.4.

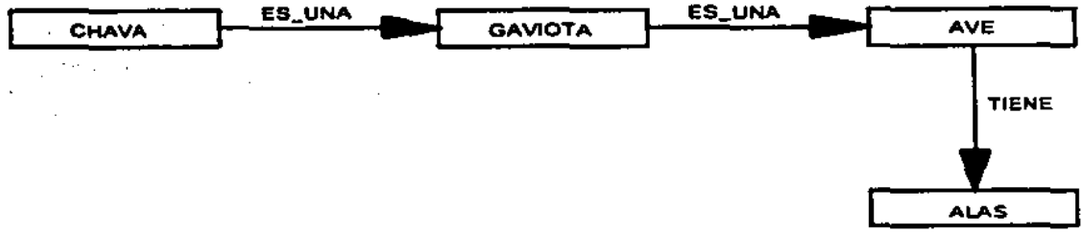


FIG. 4.4

Siguiendo esta representación es muy fácil llegar a la deducción de que las gaviotas tienen alas y que por tanto chava tiene alas. Para esto es necesario

seguir el conector jerárquico ES UNA, asumiendo que cualquier hecho afirmado sobre los nodos superiores en jerarquía puede ser considerado una afirmación sobre los nodos inferiores, esto es, sin tener que representar estas afirmaciones explícitamente en la red. En terminología de IA, esta clase de razonamiento es comúnmente llamada *herencia de propiedades.*

La herencia de propiedades se basa en el principio de que cualquier propiedad o atributo que sea declarada como verdadera para una clase de elementos, deberá ser cierta para cualquier ejemplo que pertenezca a dicha clase.

El conector ES UNA se conoce como *conector de herencia propia,* y gracias a él las propiedades descienden de los niveles altos hacia los niveles bajos.

Suponga que deseamos representar el hecho *chava posee un nido.* Nuestro primer impulso puede ser representar este hecho usando un conector de propiedad hacía un nodo que represente el nido de chava. Observe la figura 4.5.

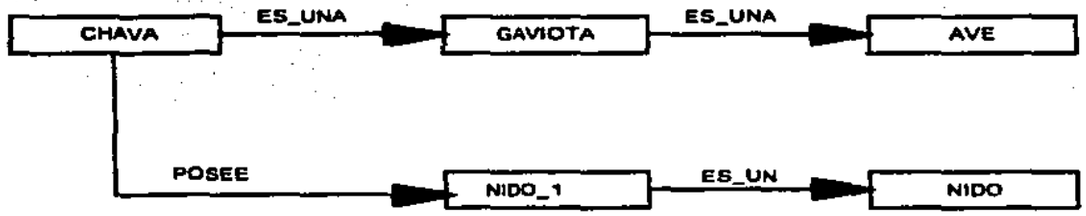


FIG. 4.5.

En la figura NIDO_1 representa una instancia de la clase general NIDO.

Como podrá haber observado, la cantidad de detalles que se pueden

incluir en una red semántica depende del tipo de problema a resolver. Por ejemplo si quisiéramos representar el hecho de que chava posee varios nidos podríamos hacerlo como se muestra en la figura 4.6.

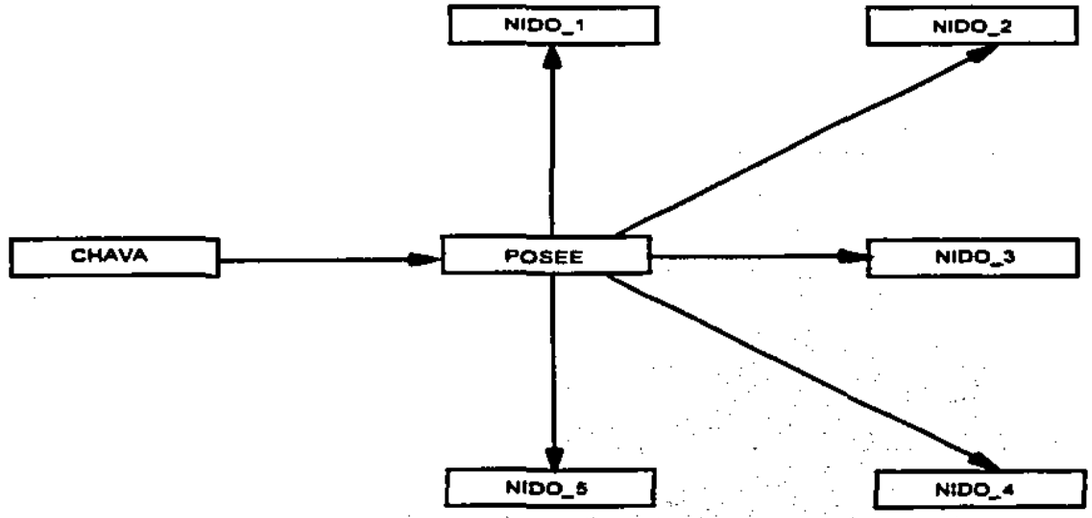


FIG. 4.6

Otro ejemplo podría ser introducir los atributos de varios objetos en la red. Por ejemplo supongamos que la pareja de Chava es Sammy y queremos mostrar las edades de ambos. Esto se podría expresar mediante la red de la figura 4.7.A).

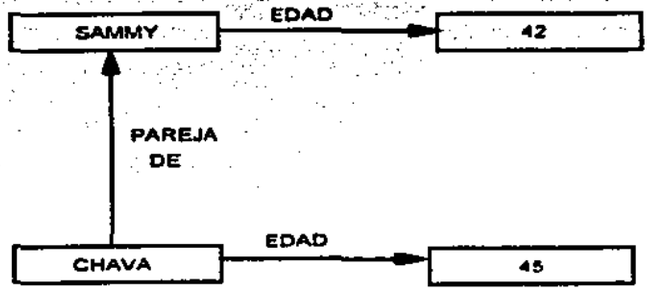


FIG. 4.7. (A)

Las edades están dadas como valores numéricos. Una forma más descriptiva de representar esta información se muestra en la figura 4.7.B). Donde se asigna un nodo de edad tanto para Chava como para Sammy, y se enlazan con el conector MAYOR_QUE nos brinda más claridad.

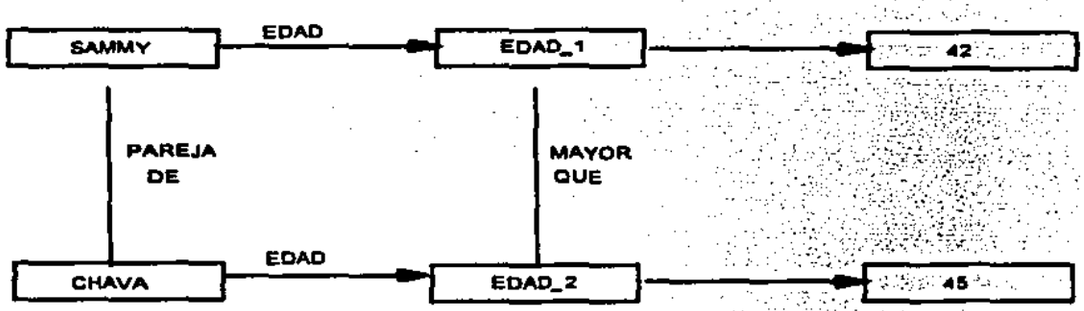


FIG. 4.7. (B)

Aunque la red semántica es una gráfica, esta no se representa así en la computadora. Los objetos y sus relaciones se establecen en forma verbal, y se programan en la computadora a través de los diferentes lenguajes de programación. La búsqueda a través de redes semánticas grandes puede resultar laboriosa. Por esta razón esta técnica es usada principalmente para procesos de análisis. Después, se ejecuta una transformación a reglas o marcos. Las redes semánticas son usadas básicamente como representaciones visuales de relaciones, y pueden ser combinadas con otros métodos de representación.

4.2.2.1. MANIPULACION Y RAZONAMIENTO

El significado de una red semántica se establece, a través de los procedimientos que interpretan o manipulan la red. Aunque las redes semánticas se usan más para organización del conocimiento que para inferencia. Es posible realizar procesos de razonamiento a través de ellas.

Una vez que se ha construida la red semántica, puede ser usada para solucionar problemas. Y la forma más usual es, realizar preguntas a la red sobre su dominio para que esta nos proporcione las respuestas.

Esto se lleva a cabo de la siguiente manera: se realiza una pregunta sobre el dominio de la red. La computadora busca progresiva o regresivamente a través de la red hasta hallar la respuesta.

La representación gráfica es trasladada a un programa de computadora. Para conducir a una búsqueda, cuyo punto de arranque estará dado por un nodo en partícular. Este punto de arranque lo determina usualmente la pregunta. La computadora después ejecuta varias técnicas de búsqueda y comparación través de la estructura de la red e identifica los objetos deseados así como sus relaciones para dar solución a la pregunta.

Los nodos variables en el fragmento (de red) son igualados durante el proceso de comparación a los valores que deben tener para hacer la comparación verdadera, Por ejemplo suponga que utilizamos la red de la figura 4.8.(A) como una base de datos.

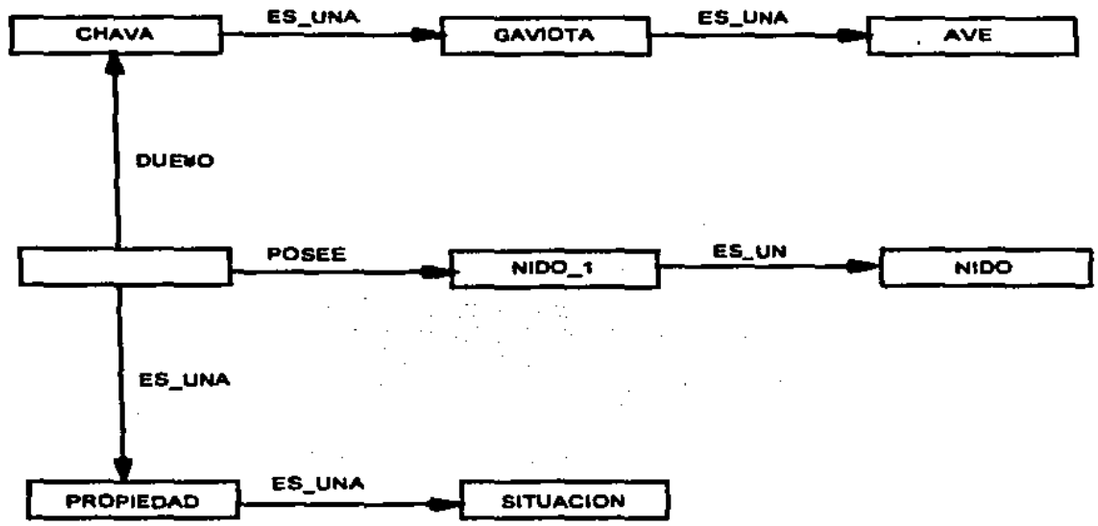


FIG. 4.8. (A)

Y suponga que deseamos responder la pregunta Qué cosa posee Chava?, nosotros podríamos construir el fragmento de red de la figura 4.8.(B).

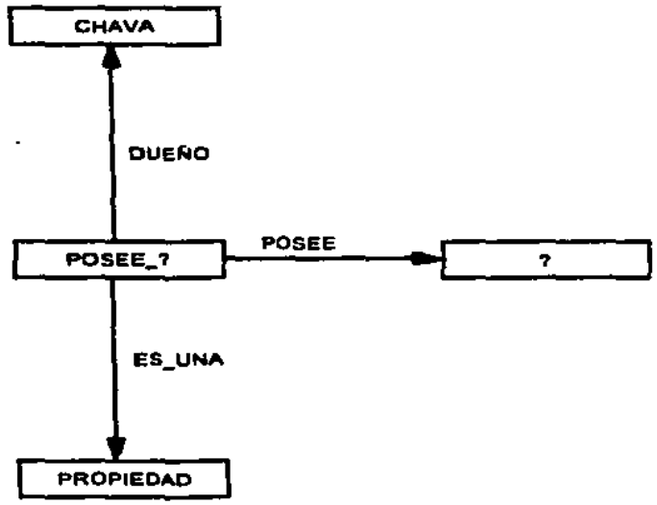


FIG. 4.8. (B)

El cual representa un caso de propiedad en el cual Chava es el dueño. Este fragmento es entonces comparado contra la base de datos de la red

buscando el nodo POSEE que tiene un conector DUEÑO hacia CHAVA. Cuando este es encontrado el nodo al que apunta el conector POSEE1 es limitado en la comparación parcial y este es la respuesta a la pregunta. En caso de no encontrarse comparaciones correctas, la respuesta podría haber sido, en efecto, Chava no posee alguna cosa.

El comparador puede hacer inferencias durante el proceso de comparación para crear estructuras de la red que no estén explícitamente presentes en ella. Por ejemplo, supongamos que deseamos responder la pregunta *Hay alguna ave que tenga un nido?* podríamos representar esta pregunta a través del fragmento de red mostrado en la figura 4.9.

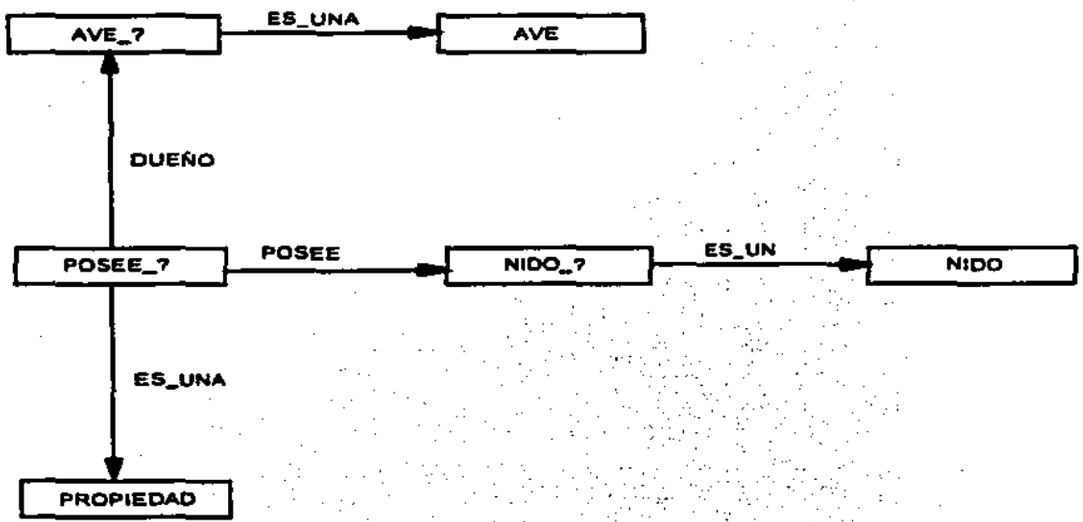


FIG. 4.9.

Aquí, los nodos AVE_?, NIDO_? y DUEÑO_? representan la relación a ser determinada ave-propiedad-nido. Note que la pregunta que representa el fragmento de red no compara la base de datos de conocimiento exactamente. el procedimiento de deducción podría tener que construir un conector ES_UN de CHAVA a AVE para hacer la comparación posible. El

comparador podría enlazar AVE_? a el nodo CHAVA, DUEÑO_? a DUEÑO_1, y NIDO_? a NIDO_1, y la respuesta a la pregunta podría ser Si, Chava, desde donde el nodo CHAVA fue limitado a AVE_? en orden para comparar el fragmento de la pregunta con la base de datos.

4.2.3. MARCOS (FRAMES)

Existe una abundante evidencia sicológica de que el ser humano usa un gran arreglo de conocimientos sobre experiencias anteriores para interpretar y analizar situaciones nuevas dentro de sus actividades (Bartlett, 1932). Por ejemplo, cuando visitamos un restaurante en donde nunca antes hemos estado, recordamos nuestras experiencias en otros restaurantes formando un gran arreglo de expectaciones, y esperamos encontrar los elementos u objetos típicos en todo restaurante: menús, mesas, meseros, etc. Sumado a esto, tenemos una gran expectación sobre la secuencia de eventos que se llevarán a cabo.

La búsqueda de una representación del conocimiento que hiciera posible representar situaciones cotidianas fue lo que dio origen a los llamados marcos de referencia. Los cuales fueron propuestos originalmente por Minsky (1975). Minsky comentaba "Cuando encontramos una nueva situación, o cambiamos substancialmente nuestro punto de vista sobre un problema, seleccionamos de nuestra estructura un marco de referencia. Es decir, recordamos un marco de referencia que adaptamos a la realidad cambiando algunos detalles si fuese necesario".

CONCEPTO, ELEMENTOS Y CONTENIDO DE UN MARCO

Un marco es una estructura de datos que representa todo el conocimiento sobre un objeto o situación en particular. Este conocimiento se organiza a través de una estructura jerárquica especial que permite el diagnóstico independiente del conocimiento. Los marcos son básicamente una aplicación de la programación orientada a objetos para IA y sistemas expertos. Cuando los programadores de IA desarrollaron ambientes POO (Programación Orientada a Objetos) se refirieron a los objetos como marcos, por eso cuando se habla sobre la representación del conocimiento basada en marcos, se sobreentiende el uso de POO en la construcción de las bases de conocimiento. Por tales razones los marcos son la base de la actual Programación orientada a objetos.

A través de un marco podemos describir un objeto, evento, ubicación o situación específicos. Ya que permite expresar y agrupar todos los valores o atributos que describen a un objeto, con lo cual es posible representar objetos complejos y situaciones enteras con lujo de detalle a través de una sola unidad.

Los marcos se usan normalmente como una herramienta para representar conocimiento estereotipado o, conocimiento construido sobre características y experiencias bien definidas y/o conocidas. En forma física, un marco es algo parecido a un esquema con categorías y subcategorías.

Existen marcos de clase y marcos de instancia, los marcos de clase, son marcos genéricos, los marcos de instancia (hoy conocidos como objetos), son marcos específicos que contienen los valores de los atributos de un marco de clase.

Un marco consta de los siguientes elementos:

Ranuras (Slots)
Facetas (facets)

A través de las ranuras se fragmenta o divide el conocimiento de una forma organizada. Ya que almacenan los distintos atributos o características del objeto o situación a representar. El conjunto de todas las ranuras contenidas en el marco describirán el objeto o situación representada por el mismo.

Una ranura puede describir conocimiento declarativo (tal como el color de un carro) o conocimiento procedural (tal como "activar cierta regla si un valor excede a nivel dado"). Esto marca la gran diferencia entre los marcos y las redes semánticas.

Un marco típico que describe a un auto se muestra en la figura 4.10. Observe que las ranuras describen atributos tales como fabricante, modelo, motor y otras características.

Cada ranura contiene una o más facetas. Las facetas (algunas veces llamadas subranuras) pueden contener atributos propios de la ranura, así como también hipótesis relativas al funcionamiento de un sistema experto, reglas para ejecutar acciones si ciertas condiciones se cumplen, apuntadores que enlazan las ranuras de un marco de referencia con otros marcos de referencia, una ranura misma puede ser un marco de referencia independiente, etc. A continuación se especifican las diferentes clases de información que puede almacenar una subranura:

Valores: Almacenan atributos tales como azul, rojo y verde para una ranura que represente color.

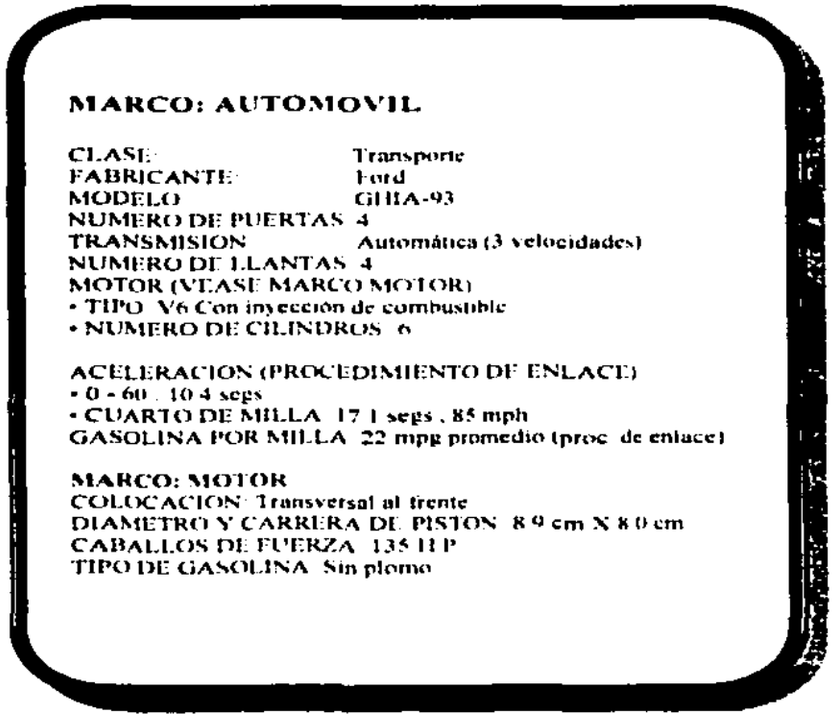


FIGURA 4.10

Default: Usada en caso de que la ranura este vacía (que se carezca de información). Por ejemplo, en el marco del automóvil un valor default es que el número de puertas de un auto es de cuatro. Esto significa que asumiremos que el auto tiene cuatro puertas a menos que se indique otra cosa.

Rango: Indica que clase de información puede aparecer en una ranura, por ejemplo, aceptar sólo números enteros de 0 a 100.

Si agregado(if added): Contiene información procedural o de enlace. Especifica que acción se debe realizar cuando se agrega (o modifica) un valor en la ranura. Estos procedimientos de enlace son llamados *demons*.

Si necesario(if needed): Usada en caso de que no sea proporcionado el valor de la ranura. Esta ejecuta un procedimiento (muy parecido a si agregado) que buscará o calculará tal valor.

Otros: Las ranuras pueden contener marcos, reglas, redes semánticas, o cualquier otro tipo de información.

Algunos procedimientos pueden enlazarse a través de ranuras y ser usados para producir valores de la misma. Por ejemplo, las ranuras con especificaciones heurísticas son procedimientos que proporcionan valores para las ranuras en un contexto en particular. Un aspecto importante de tales procedimientos es que pueden ser usados para direccionar o controlar el proceso de razonamiento. Además de que al ser llenadas las ranuras (con información), estas pueden ser usadas para disparar otro proceso.

En la figura 4.10., los procesos de aceleración y recorrido por kilometro son procedimientos de enlace. Ellos describen paso a paso el proceso mediante el cual se podría obtener esta información. Por ejemplo, para determinar la aceleración, tomamos el tiempo del recorrido entre 0 y 60 kph y el lapso de tiempo en el cuarto de kilómetro y la rapidez descrita. El procedimiento de enlace para determinar la distancia recorrida podría almacenar un procedimiento para llenar el tanque de gasolina, conducir un cierto número de km., determinar la cantidad de gasolina usada, y calcular la gasolina por km., en términos de km., por galón.

RELACIONES JERARQUICAS

La mayoría de sistemas de IA usan una colección de marcos que se enlazan o relacionan de una manera similar a la cual se enlazan los nodos de una red semántica. Los elementos que permiten las relaciones entre marcos sirven como apuntadores entre los mismos. La posibilidad de relacionar los marcos hace posible la construcción de una red de marcos. El conocimiento disponible se divide entre los marcos que formarán la red de marcos relacionados entre sí. La ranura con el valor is_a permite la herencia de propiedades en la estructura de marcos. Además los marcos pueden organizarse jerárquicamente, como se muestra en la figura 4.11 donde se muestra un ejemplo. Algunos puntos importantes que hay que notar son:

1. La estructura jerárquica facilita la herencia de propiedades.
2. El proceso de representación va de lo general a lo particular(específico).
3. En general, la red de marcos es más compleja, con apuntadores de varios marcos hacia un marco específico, el cual a su vez puede apuntar a varios otros marcos.

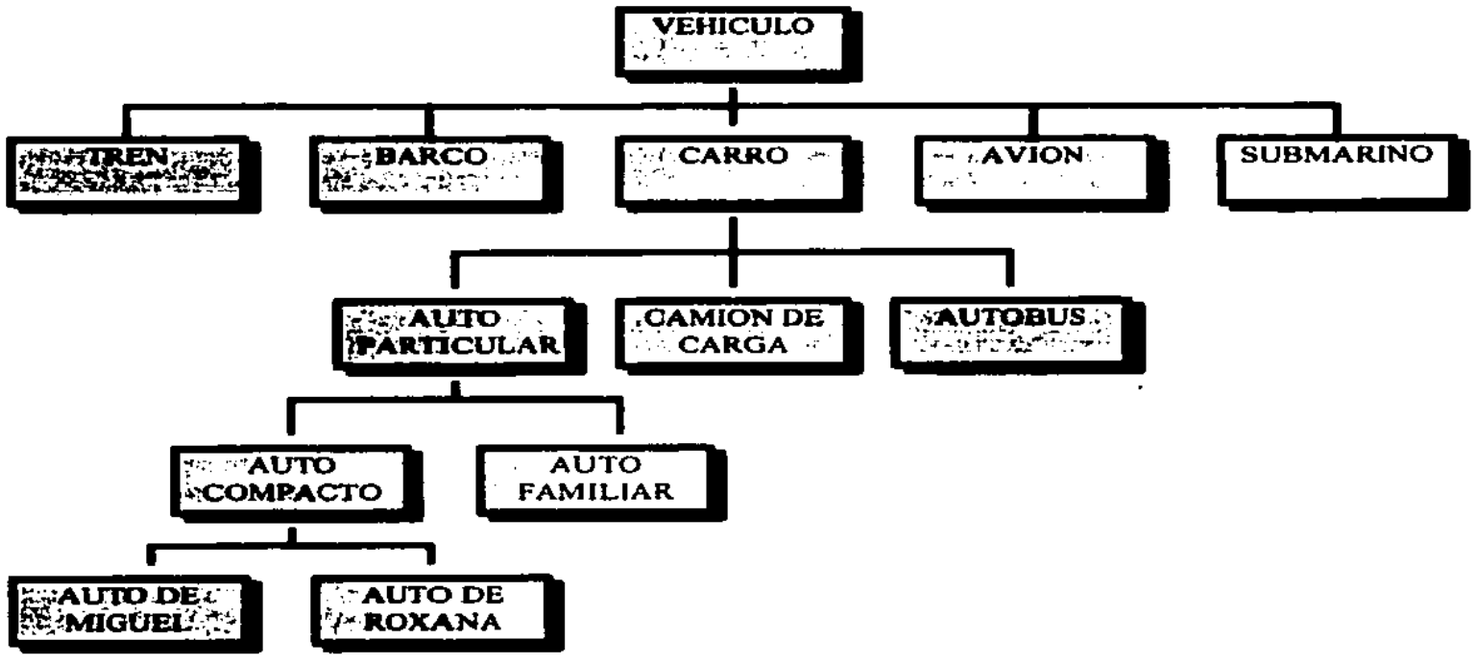

VEHICULO
TREN
BARCO
CARRO
AVION
SUBMARINO
AUTO PARTICULAR
CAMION DE CARGA
AUTOBUS
AUTO COMPACTO
AUTO FAMILIAR
AUTO DE MIGUEL
AUTO DE ROXANA

FIG. 4.11.

4.2.3.1. MANIPULACION Y RAZONAMIENTO

El razonamiento con marcos se realiza a través de las ranuras que proporcionan un mecanismo para una clase de razonamiento conocido como proceso de manipulación de expectaciones. Las ranuras vacías (expectaciones no confirmadas) pueden ser llenadas, de acuerdo a ciertas condiciones, con datos que confirmen las expectaciones o hechos esperados. De esta manera, el razonamiento basado en marcos busca confirmar las expectaciones para así realizar el llenado de las ranuras.

El proceso de razonamiento se basa en la búsqueda para la confirmación de varias expectaciones. A través de los marcos, es mucho más fácil realizar inferencias sobre nuevos objetos, eventos, o situaciones porque el conocimiento proporcionado por los marcos esta basado en experiencias previas.

El razonamiento en los marcos comúnmente se realiza a través de dos formas: usando reglas y empleando razonamiento jerárquico.

USANDO REGLAS

A través de una regla se puede razonar sobre las características de un marco haciendo referencia a los valores contenidos en las ranuras.

EJEMPLO

Se debe planear una ruta para dirigirse a cierto lugar donde el puente se debe eludir si el vehículo es demasiado pesado. Una regla como la siguiente podría ponerse en la base de conocimientos:

REGLA 1: SI PESO de MI-AUTO > 3000 libras
ENTONCES tomar DESVIO alrededor DEBIL PUENTE

Si el valor de la ranura PESO perteneciente al marco MI-AUTO es más grande que 3000 libras, entonces se debe desviar alrededor del puente.

Las reglas pueden ser usadas para procesar el conocimiento estructural contenido en una taxonomía. La siguiente regla, la cual podría formar parte de un conjunto para determinar a que tipo de estación de servicio se debería parar un conductor para cargar gasolina, se ilustra a continuación.

REGLA 2: IF ?VEHICULO is MIEMBRO-DE AUTOMOBILES
AND ?VEHICULO tiene_un ?MOTOR
AND ?MOTOR is MIEMBRO-DE DIESEL
THEN pon REPOSTAR de ?VEHICULO a TRUCKSTOP

En la primera línea el motor de inferencia intenta instanciar la variable ?VEHICULO CON CADA MARCO, pero solo guarda aquellas instanciaciones que son miembros de la clase AUTOMOBILES. Hasta este punto, varios vehículos podrían calificar, además se podrían establecer varías instancias de la regla. La segunda línea realiza la instanciación de la variable ?MOTOR con cualquier marco al que apunte la variable ?VEHICULO gracias a la relación has-a. En consecuencia, si vehiculo-2 tiene un motor y un tanque de gasolina, la regla podría ser instanciada dos veces - una con ?VEHICULO=vehiculo-2 y ?MOTOR=MOTOR y otra con ?VEHICULO=vehiculo-2 y ?MOTOR=TANQUE DE GASOLINA.

La tercer línea de la regla elimina cualquier instanciación de la regla para la cual ?MOTOR no es miembro de la clase DIESEL. La cuarta línea pone en la ranura REFUEL del vehículo asociado el valor TRUCKSTOP.

El razonamiento con un conocimiento taxonómico no es exclusivo del área de las reglas, Aunque muchos sistemas basados en marcos usan el conocimiento basado en reglas extensivamente. Sin embargo, el

conocimiento procedural del tipo encontrado en los sistemas de blackboard pueden procesar conocimiento basado en marcos rápida y eficazmente.

USO DEL RAZONAMIENTO JERÁRQUICO.

De acuerdo al razonamiento jerárquico, ciertas alternativas, objetos o eventos pueden ser eliminados en varios niveles de la búsqueda jerárquica. Por ejemplo en un nivel dado podríamos tener la regla:

REGLA: IF FABRICANTE de ?VEHICULO es FORD
THEN ?VEHICULO NO es MIEMBRO-DE DIESEL

Asumiendo que Ford no manufactura motores diesel, una regla en este nivel puede eliminar un número de posibilidades de clases y subclases.

4.2.4 GUIONES (SCRIPTS)

Un guión es un esquema de representación del conocimiento similar a un marco, pero además de describir un objeto, el guión describe prototipos de secuencias de eventos o sucesos. Para describir tal secuencia, usa una serie de ranuras que contienen información sobre personas, objetos y acciones que forman parte de los eventos.

Los componentes más comunes que se pueden encontrar en un guión, y que están representados por ranuras son los siguientes:

Condiciones de entrada. Las cuales describen situaciones que deben cumplirse para que la serie o libreto del guión se lleve a cabo.

Situación, libreto o serie. describe el evento que se desarrollara en el guión.

Accesorios. describe los objetos usados en la secuencia de eventos descrita en el guión.

Papeles. describe la gente que participará en el guión.

Escenas. Describe la secuencia de eventos que deberá ocurrir.

Resultados. Proporciona información sobre los posibles resultados al término del guión.

GUION: RESTAURANTE

SITUACION: Establecimiento de comida rápida

PAPELES: Cientes y Servidores

ESCENARIO: Cajera
Comida
Dinero
Servilletas
Palillos
Salsas

CONDICIONES DE ENTRADA:
Cliente ambriento
Cliente con dinero

ESCENA 1: ENTRADA
- Cliente estaciona el carro
- Cliente entra al Restaurante
- Cliente espera para pagar
- Cliente observa el menu y decide que pedir

ESCENA 2: ORDEN
- Cliente pide su orden
- Mozo entrega la orden
- Cliente paga el servicio

ESCENA 3: COMIDA
- Cliente toma salsa,palillos y servilletas
- Cliente toma su orden y ocupa una mesa
- Cliente come rapidamente sus alimentos

ESCENA [illegible] (OPCIONAL): TOMAR Y SALIR
- Cliente toma sus alimentos para llevar

ESCENA [illegible]: SALIR
- Cliente limpia la mesa
- Cliente tira la basura
- Cliente se marcha del establecimiento
- Cliente sube a su auto nuevamente

RESULTADOS
- Cliente calmo su hambre
- Cliente gasto dinero

FIG. 4.12

Un ejemplo típico se muestra en la figura 4.12. Este es una variación del conocido ejemplo del restaurante que es muy usado en IA para mostrar como se representa el conocimiento a través de un guión. El ir a un restaurante presenta una situación estereotipada (tipificada y reiterativa) con condiciones de entrada, accesorios, papeles y escenas predecibles. Como se observa en la figura el guión describe cuidadosamente lo que ocurre en restaurantes de comida rápida. Esto se logra a través de ranuras que presentan miniguiones dentro de un guión principal. Y en conjunto describen en forma organizada y ordenada las diferentes etapas del proceso.

La utilidad de los guiones esta en su capacidad para representar situaciones estereotípicas y acciones que la gente realiza cotidianamente. Gracias a esto permite la predicción de eventos dentro de situaciones específicas. Este conocimiento parece generalmente seguro, pero en situaciones para solución de problemas (mediante computadora), tal conocimiento debe frecuentemente simularse para resolver un problema en particular usando IA.

Para hacer uso del guión se debe almacenar el conocimiento en la computadora de una manera simbólica. El mejor medio para hacerlo es a través del uso de LISP u otro lenguaje simbólico. Después de esto se puede cuestionar a la computadora sobre personas o situaciones. La cual a través de un proceso de búsqueda y comparación alrededor del guión busca la respuesta correcta. Por ejemplo, Cuál es la primera acción del cliente? bien, estaciona su carro y después entra al restaurante. Como vemos esto es totalmente predecible gracias al guión.

4.2.4.1. RAZONAMIENTO BASADO EN CASOS

La idea básica del razonamiento basado en casos es adaptar soluciones que fueron usadas para resolver problemas anteriores y usarlas para resolver nuevos problemas. Esto se hace por medio de los siguientes pasos:

- Buscar en memoria aquellos casos que resolvieron problemas similares al problema actual
- Adaptar las soluciones encontradas para solucionar el problema actual, tomando en cuenta las diferencias entre los problemas anteriores y el problema actual.

La búsqueda de los casos debe tomar en cuenta lo siguiente:

- Caracterizar la entrada del problema, asignando las características propias de este.
- Recuperar de la memoria los casos que cumplan con tales características, y
- Seleccionar el caso o casos que igualen la entrada.

Como se dará cuenta el problema central es la identificación correcta de la información. Para esto se utiliza la ayuda de los guiones.

Al disponer de guiones, se pueden realizar inferencias sobre lo que sucederá en situaciones descritas en dichos guiones de una manera muy fácil. Por esta razón muchas veces es posible decir que el razonamiento no es más que la aplicación del guión. En consecuencia al tener la mayor cantidad de guiones posibles el proceso de pensar disminuye.

Por último es importante mencionar que a través de un estudio Riesbeck y Shank postulan, que la mayoría de la gente al ponerse en el dilema de realizar un proceso de pensar o elegir un guión ya hecho, siempre elegirá el guión.

5. SISTEMAS EXPERTOS

Los primeros trabajos sobre SE (Sistemas Expertos) se remontan a la década de los 60's y desde entonces su uso se ha incrementado enormemente pasando de la investigación a la práctica, dando resultados excelentes en diferentes áreas como: agricultura, negocios, química, ingeniería, medicina, finanzas, geología, comunicaciones, sistemas de computación, electrónica, etc.

Por lo anterior se considera a los SE como la primera aplicación verdaderamente operacional de IA. Además de ser una de las aplicaciones más interesantes y apasionantes.

5.1. ANTECEDENTES Y DESARROLLO

Durante la década de los 60's dentro de la comunidad investigadora de IA dominaba la creencia de que unas pocas leyes de razonamiento conjuntadas con poderosas computadoras podrían producir un desempeño superior al humano. Un intento en esta dirección fue el *General Problem Solver* (GPS).

El GPS fue desarrollado por Newell y Simon para su máquina de teoría lógica, que fue un intento por crear una computadora inteligente. Por tanto, puede ser considerado como un predecesor de los SE. El GPS trata de generar los pasos necesarios para llevarnos de una situación inicial hacia una situación u estado deseado. Por cada problema a ser resuelto, se proporcionaba la siguiente información:

1) Un conjunto de operadores que pueden transformar una situación de varias formas,

2) Una sentencia sobre las precondiciones que cada operador necesita tener para poder ser aplicado,

3) Una lista de postcondiciones que deberán cumplirse después de que los operadores sean usados.

Además podía incluirse un conjunto opcional de reglas heurísticas para indicar que operador aplicar primero (En términos de SE esto se conoce como una *base de reglas*).

El GPS intentaba encontrar los operadores que minimizarán la diferencia entre un objetivo y el estado analizado. Algunas ocasiones los operadores no eran aplicables sobre los estados analizados (ya que sus precondiciones no eran apropiadas). En tales casos, el GPS proponía el mismo un subobjetivo: para cambiar tal estado a otro que fuera apropiado para los operadores. En ocasiones se generaban varios subobjetivos antes de que el GPS pudiera resolver un problema.

Desarrollo

El cambio de programas de propósito general a programas de propósito específico se presento a mediados de los años 60's con el surgimiento de DENDRAL desarrollado por E. Feigenbaum en la Universidad de Stanford, seguido por el desarrollo de MYCIN. En este tiempo los investigadores habían reconocido que el mecanismo para solución de problemas solo era una pequeña parte de un sistema completo para lograr la computadora inteligente.

La construcción de DENDRAL llevó a las siguientes conclusiones:

- Los programas como el GPS son débiles para ser usados como base en la construcción de SE más sofisticados (con alto rendimiento).
- Los resolvedores humanos de problemas son útiles solo si operan sobre un dominio muy limitado.
- Los sistemas expertos necesitan ser constantemente actualizados. Tal actualización puede ser hecha eficazmente mediante el uso de la representación basada en reglas.

- La complejidad de los problemas requiere una gran cantidad de conocimientos acerca del área a tratar.

A mediados de la década de los 70's, empezaron a emerger varios SE. En los cuales ya se reconocía el papel central del conocimiento, los científicos de IA trabajaban en el desarrollo de representaciones de conocimiento que pudieran ser manipuladas por la computadora para tomar decisiones, realizar inferencias etc. Gracias a estas investigaciones se concluyo que: *El poder de un SE radica en el conocimiento específico que este posea.*

El siguiente cuadro muestra una relación de algunos SE tradicionales, y su área de aplicación.

SISTEMA	AREA DE APLICACION
DENDRAL	QUIMICA
METADENDRAL	
HEARSAY	INTERPRETACION DEL LENGUAJE NATURAL
MACSYMA	MATEMATICAS
EXPERT	DIAGNOSTICO MEDICO
CASNET	
CADUCEUS	
MYCIN	
TEIRESIAS	HERRAMIENTA PARA CONSTRUCCION DE BASE DE CONOCIMIENTOS
PROSPECTOR	EXPLORACION MINERAL
ROSIE	HERRAMIENTA DE DESARROLLO PARA SE
OP5	
AGE	
CATCH	INVESTIGACION POLICIACA (RASTREO DE FOTOGRAFIAS)
R1	CONFIGURACION DE EQUIPOS DE COMPUTO
XCON	PARA LA DEC

FIG. 5.1. SE Y ÁREA DE APLICACIÓN.

A principios de 1980, la tecnología de los SE, que había estado limitada al ámbito académico, empezó a surgir como una aplicación comercial. Donde destacaron XCON y XSEL (desarrollados en Digital Equipment Corp.) y CATS-1 (desarrollado en General Electric).

En adición a los SE construidos, se hizo un esfuerzo para desarrollar herramientas que aceleraran la construcción de SE. Estas herramientas auxiliaban en las siguientes tareas:

- programación (EMYCIN y AGE),
- adquisición del conocimiento (EXPERT y KAS),
- aprender de la experiencia (METADENDRAL y EURISKO.)

Tales herramientas estuvieron disponibles comercialmente en 1983. La mayor parte de los primeros desarrollos requerían hardware especial (máquinas LISP), pero las aplicaciones desarrolladas después de 1980, podían correr sobre computadoras comunes incluyendo microcomputadoras. Tales aplicaciones permitieron lograr los siguientes alcances en el área de SE:

- Disponibilidad de herramientas para facilitar el desarrollo de SE, con lo cual se minimizó su costo.
- Diseminación de SE en miles de organizaciones, algunas de las cuales requieren cientos o en ocasiones miles de sistemas específicos
- Integración extensiva de SE con otros sistemas de información basados en computadora, especialmente, bases de datos y sistemas para soporte de decisiones
- Incremento del uso de SE en muchas tareas, desde pequeños sistemas hasta complejos sistemas militares.

- Uso de la tecnología de SE como metodología para facilitar la construcción de sistemas comunes de información.

- incremento del uso de la programación orientada a objetos como una vía para la representación del conocimiento

- Desarrollo de sistemas complejos con múltiples fuentes de conocimiento, múltiples líneas de razonamiento, e información incompleta

- Uso de múltiples bases de conocimiento

5.2. CARACTERISTICAS, DEFINICION Y CONCEPTOS RELACIONADOS

Características y definición.

Algunas de las características descriptivas de los SE son las siguientes:

- Son programas que contienen los conocimientos de uno o varios especialistas humanos (expertos) en un determinado campo de aplicación.
- Capacidad para actualizar fácilmente sus conocimientos.
- Emulan el razonamiento humano para la resolución de problemas.
- Capacidad para explicar, advertir o recomendar y algunas veces justificar el uso de ciertas acciones y/o la obtención de una solución o respuesta.
- A través del sistema un usuario no especializado puede resolver problemas relativos a su área de aplicación de una forma sencilla rápida y confiable.

Aunado a esto Feigenbaum define un SE como un "programa inteligente que utiliza conocimientos y procedimientos de inferencia para resolver problemas que son lo suficientemente difíciles como para requerir experiencia humana para su correcta solución"

Con lo anterior podemos dar la siguiente definición de Sistema Experto:

Un Sistema Experto es un programa inteligente de computadora que contiene una gran cantidad de conocimiento en un área específica, el cual es procesado a través de técnicas especiales para realizar inferencias, tomar decisiones y consecuentemente resolver problemas que requerirían de un experto humano para su solución.

Conceptos relacionados.

Relacionados con el ámbito de un SE tenemos los siguientes términos:

Experiencia.

Es el amplio conocimiento sobre áreas específicas, adquirido a través de viajes, lecturas, y experiencias propias. Esta incluye los siguientes tipos de conocimiento:

- Hechos sobre un problema dado.
- Teorías sobre tal problema.
- Reglas y procedimientos proporcionados por el problema.
- Reglas de acción para situaciones específicas.
- Estrategias globales para resolver ese tipo de problemas.
- Metaconocimiento (conocimiento sobre el conocimiento).

Este tipo de conocimiento permite al experto tomar mejores y más rápidas decisiones en comparación con una persona inexperta al resolver problemas complejos. Sin embargo el adquirir la experiencia suficiente se lleva un largo tiempo (varios años).

Expertos.

Es difícil definir lo que es un experto, ya que generalmente hablamos acerca de grados o niveles de experiencia. (La cuestión es, que tanta experiencia debe poseer una persona para ser considerado experto.)

Los siguientes comportamientos son típicos de una persona con gran experiencia:

- Reconocimiento y formulación de problemas
- Solución rápida y apropiada del problema
- Explicación de la solución
- Aprendizaje a través de la experiencia
- Reestructuración del conocimiento
- Ruptura de reglas
- Determinación de relevancias

Los expertos pueden tomar un problema y transformarlo de manera que los guíe hacia una rápida y efectiva solución. Es necesaria la habilidad para resolver problemas, pero no es suficiente. Los expertos deberán ser capaces de explicar los resultados, aprender nuevas cosas sobre el dominio, reestructurar el conocimiento siempre que sea necesario, romper con ciertas reglas (aplicando las excepciones a estas) y determinar si su experiencia ha sido relevante. Todas estas actividades deben realizarse eficientemente (rápido y a un bajo costo) y efectivamente (con resultados de alta calidad).

Para emular a un experto humano, es necesario construir un programa de computadora que exhiba todas estas características.

Transferencia del conocimiento

Uno de los objetivos más importantes en la realización de un sistema experto es la transferencia del conocimiento de un experto hacia una computadora y después a usuarios inexpertos. Este proceso envuelve cuatro actividades:

1. Adquisición del conocimiento (del experto y/o otras fuentes),
2. Representación del conocimiento (dentro de la computadora),
3. Manipulación del conocimiento, y
4. Transferencia del conocimiento al usuario.

El conocimiento es almacenado en la computadora dentro de una base de datos. Se distinguen dos tipos de conocimiento: hechos y procedimientos (comúnmente reglas) proporcionadas por el dominio del problema.

Inferencia.

Para razonar los SE se basan además del conocimiento mencionado anteriormente, en observaciones y conocimientos no confirmados basados en experiencia e intuición (heurísticos). La manipulación de estos conocimientos permite que el sistema pueda realizar inferencias; el mecanismo encargado de esta función es conocido como motor de inferencia, el cual incluye procedimientos que proporcionan la solución del problema.

Reglas.

La mayoría de SE comerciales se basan en reglas; esto es, el conocimiento se almacena principalmente en forma de reglas (SI...ENTONCES).

Las propiedades y características que hacen a un SE diferente de los sistemas convencionales se presentan en la tabla de la figura 5.2.

SISTEMAS CONVENCIONALES	SISTEMAS EXPERTOS
Proporcionan gran facilidad para capturar, manejar y accesar información numérica (datos cuantitativos)	Proporcionan gran facilidad para Manejar información simbólica (datos cualitativos). Captura y acceso a juicios y conocimientos
Solucionan problemas mediante el uso de algoritmos o procesos repetitivos fijos previamente programados, que siempre esperan el mismo tipo de datos de entrada	Sus soluciones se basan en métodos que realizan inferencias a partir de información previamente almacenada
La información y su procesamiento se combinan en un programa secuencial	La base de conocimiento esta claramente separada del mecanismo de procesamiento (reglas de conocim iento separadas del control)
Los errores no son del programa (Son del programador)	El programa puede cometer errores
No explican porque se requieren los datos de entrada o como se obtienen los resultados o salidas	Pueden proporcionar una explicacion sobre el procedimiento que han seguido para llegar a una solución o respuesta justificando la forma de su obtención
El sistema opera solo cuando esta completamente terminado	El sistema puede operar con pocas reglas (como el primer prototipo)
Requiere toda la informacion pedida para operar	Puede operar con información incompleta o confusa
Manipula extensas B.D.	Manipula extensas bases de conocimiento
Representacion y uso de datos	Representacion y uso de conocimiento
La eficiencia es su mayor objetivo	La efectividad es su mayor objetivo

FIG. 5.2. CUADRO COMPARATIVO SE - SISTEMAS CONVENCIONALES.

5.3. ELEMENTOS ESTRUCTURALES

Un SE se compone de los siguientes elementos:

- Subsistema para adquisición del conocimiento
- Base de conocimientos
- Motor de inferencia
- Blackboard (área de trabajo)
- Interface de usuario
- Subsistema de explicación (justificación)
- Sistema de refinación del conocimiento

Observe el diagrama de la figura 5.3

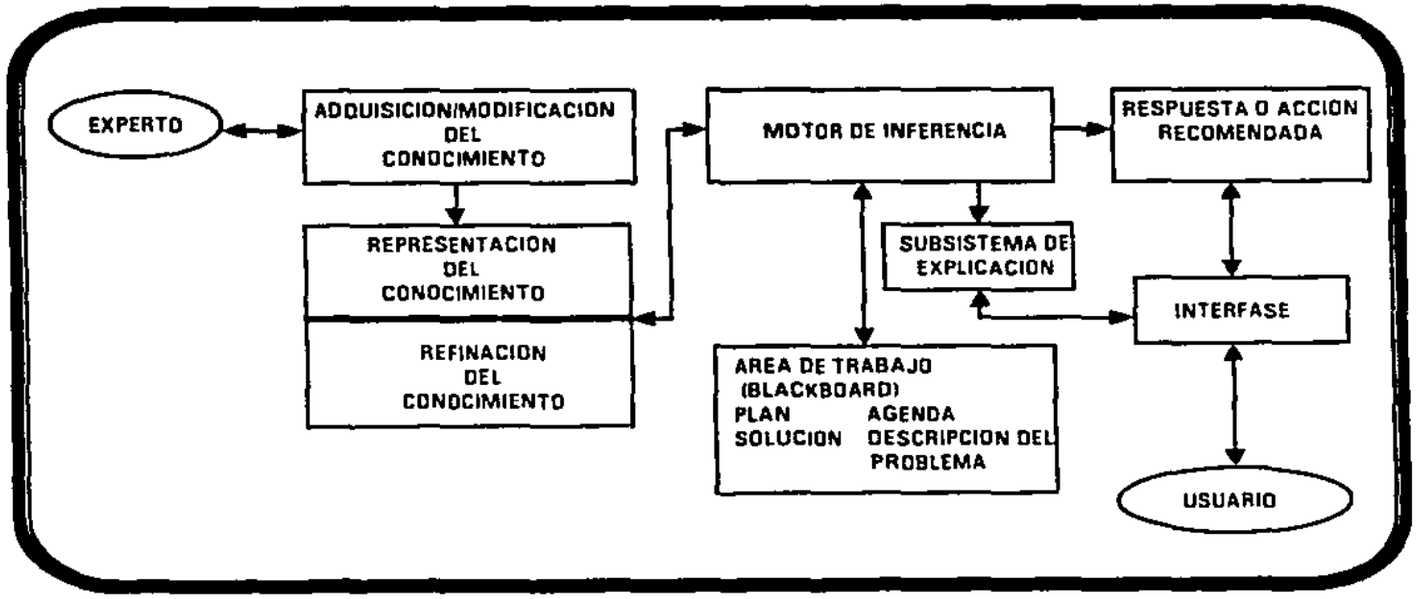


FIG. 5.3. RELACIÓN ENTRE ELEMENTOS ESTRUCTURALES DE UN SE.

Subsistema de adquisición del conocimiento

Se llama adquisición del conocimiento a la acumulación, transferencia, y transformación de conocimientos de un área determinada desde una fuente confiable a un programa de computadora para construir o aumentar la base de conocimientos. Las fuentes potenciales de conocimiento incluyen expertos humanos, libros, bases de datos, boletines de investigación, revistas científicas, etc.

La adquisición de conocimiento de un experto es una tarea compleja que frecuentemente crea cuellos de botella en la construcción de SE. Por tal motivo se requiere de un ingeniero de conocimiento para interactuar con uno o más expertos en la construcción de la base de conocimientos.

El ingeniero de conocimiento, ayuda al experto a estructurar el problema a través de la interpretación e integración de preguntas y respuestas, trazado de analogías, proposición de ejemplos, etc.

Base de conocimientos

La base de conocimientos contiene la información necesarios para entender, formular y resolver problemas. Esto incluye dos elementos básicos:

1) Hechos, tales como la situación del problema y teoría sobre el área del problema.
2) Heurísticas especiales, o reglas que direcccionan el uso del conocimiento para resolver problemas específicos en un dominio particular.

Recordemos que la heurística expresa juicios informales de conocimiento en una área de aplicación. Estrategias globales, que pueden ser heurísticas y teóricas, generalmente son incluidas en la base de conocimientos. El conocimiento es el componente principal de un SE. Este se almacena a través del proceso llamado representación del conocimiento.

Motor de inferencia

El cerebro de SE es el motor de inferencia, conocido además como la estructura de control o interprete de reglas. Este componente es esencialmente un programa de computadora que proporciona una metodología para razonar sobre la información contenida en la base de conocimiento y por consiguiente formular conclusiones. Este componente proporciona lineamientos para usar el sistema de conocimiento en el desarrollo de la agenda que organizará y controlará los pasos a seguir para resolver problemas después de que se ejecuta la consulta. El motor de inferencia consta de tres elementos:

1) Un intérprete, el cual ejecuta la acción elegida de la agenda aplicando la correspondiente regla de la base de conocimiento.

2) Un programador (scheduler) que mantiene control sobre la agenda. Estima los efectos de la aplicación de reglas de inferencia.

3) Un ejecutor de consistencia(consistency enforcer), que intenta mantener una representación consistente de la posible solución.

Blackboard (Area de trabajo)

Es una área de memoria usada para la descripción del problema, así como para especificación de datos de entrada, además es usada para grabar resultados intermedios. El blackboard graba hipótesis y decisiones intermedias. Tres tipos de decisiones pueden grabarse sobre el blackboard:

1) plan -como atacar el problema,
2) agenda -acciones potenciales esperando ejecución,
3) solución -hipótesis propuestas y vías alternativas de acción que el sistema ha generado hasta el momento.

El Blackboard existe solo en algunos sistemas expertos. Su uso es especialmente popular cuando varios expertos trabajan juntos para resolver un problema.

Considere el siguiente ejemplo: Cuando tu carro falla, metes los síntomas de la falla dentro de la computadora para almacenarlos en el blackboard. Como resultado de una hipótesis intermedia desarrollada en el blackboard, la computadora podrá sugerir que hagas algún chequeo adicional (por ejemplo checar si la batería esta correctamente conectada) y te pedirá el resultado de tu reporte. De nuevo esta información será grabada en el blackboard.

Interface de usuario

SE contiene un procesador de lenguaje para una comunicación amigable entre el usuario y la computadora. Esta comunicación podría llevarse mejor en lenguaje natural, y en algunos casos comprende menús y gráficas.

Susbsistema de explicación (justificador)

El subsistema de explicación puede ayudarnos a comprender el comportamiento del ES a través de respuestas a preguntas como las siguientes:

- Por qué fue hecha tal pregunta por el SE?
- Cómo se logró tal conclusión?
- Porqué se rechazo una alternativa?
- Cuál es el plan para lograr la solución?

Sistema de refinación.

Los expertos humanos poseen un sistema de refinación del conocimiento; esto es pueden analizar su propio rendimiento, aprender de este, y mejorarlo continuamente. Similarmente, tal evaluación es necesaria en el aprendizaje de la

computadora además de que el programa será capaz de analizar las razones de su éxito o falla.

Esto podría guiar a una mejoría que resultará en una mejor base de conocimientos y un razonamiento más efectivo. Tal componente no está disponible aun en SE comerciales, pero esta siendo desarrollado en SE experimentales en varias universidades e instituciones de investigación.

El elemento humano en los sistemas expertos

Al menos dos personas, y posiblemente más, participan en el desarrollo y uso de un SE. Como mínimo hay un experto y un usuario. Aunque frecuentemente también hay un ingeniero de conocimiento y un desarrollador del sistema. Cada uno juega un papel específico.

El experto.

El experto, es una persona que posee conocimiento especial, juicio experiencia y métodos junto con la habilidad para aplicar estos talentos en consultas y resolver problemas. Su papel consiste en proporcionar el conocimiento sobre como realiza las tareas que el sistema realizará. El experto sabe que hechos son importantes y entiende el significado de las relaciones entre ellos.

Ingeniero en conocimiento

Ayuda al experto a estructurar el conocimiento en cierta área mediante preguntas, establecimiento de analogías y ejemplos, resolviendo problemas conceptuales. El o ella es quien se encargará de desarrollar el sistema. La corta experiencia de un ingeniero en conocimiento es el mayor cuello de botella en la construcción de un SE. Para atacar este problema se están usando herramientas especiales y la investigación se esta enfocando a sistemas que eviten la necesidad de un Ingeniero en conocimiento.

El usuario

La mayoría de sistemas basados en computadora están orientados a un único modo de operación. En contraste, un SE puede tener varios tipos de usuarios:

- Usuario no experto buscando información. En tal caso el SE actúa como consultor.
- Estudiante que quiere aprender. En tal caso el SE actúa como instructor.
- El desarrollador del sistema que quiere incrementar la base de conocimientos. En tal caso el SE se comporta como un compañero.
- Un experto. En tal caso el sistema se comporta como un colega.

Dependiendo de la función que realizará el SE, puede haber otros participantes. Por ejemplo, un desarrollador puede auxiliar en la integración del SE con otro sistema computarizado. Un desarrollador de herramientas que puede proporcionar o desarrollar herramientas especiales. Proveedores que pueden proveer de herramientas especiales, y soporte STAFF que pueden proporcionar información técnica.

Los participantes y sus papeles se muestran en la figura 5.4. Note que una persona puede ejecutar varios papeles.

Por ejemplo algunos sistemas solo incluyen un experto y un usuario mientras que otros incluyen solo el desarrollador y los usuarios.

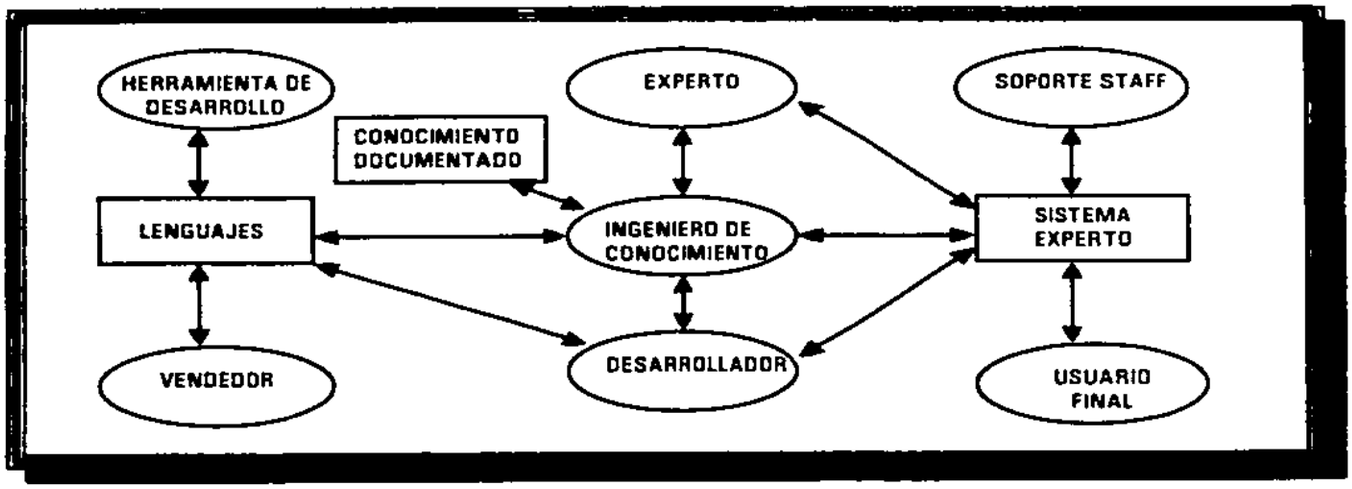


FIG.5.4. RELACIÓN ENTRE PARTICIPANTES DEL DESARROLLO DE UN SE.

5.4. CUANDO SE REQUIERE DESARROLLAR UN SE ?

La necesidad de un SE no solo se justifica por el hecho de que otras alternativas no sean apropiadas, según Waterman (1985) tienen que hacerse las siguientes consideraciones para determinar el desarrollo de un SE.

Requerimientos

Los siguientes doce requerimientos son totalmente necesarios para el éxito en el desarrollo de un SE.

1. La tarea no requiere sentido común.
2. La tarea no requiere habilidades físicas, sólo conocimiento.
3. Al menos existe un experto dispuesto a colaborar.
4. Tal experto puede explicar sus métodos para solución de problemas.
5. El experto también puede apoyar y convenir en la solución apropiada para el problema.
6. La tarea no es demasiado difícil.
7. La tarea esta bien comprendida y su definición es clara.
8. La definición de la tarea es bastante estable.
9. Las técnicas de solución convencionales por computadora, no son satisfactorias.
10. Se pueden tolerar resultados incorrectos o no óptimos.
11. Están disponibles tanto, información como casos de prueba.
12. El vocabulario solo implica unir cientos de conceptos.

Justificación

Tal como un sistema de información, un SE necesita ser justificado. De los siguientes ocho factores, al menos uno debe cumplirse uno para justificar un SE.

1. La solución del problema dará resultados exitosos
2. El SE puede preservar la experiencia humana además de que esta no se perderá
3. La experiencia es necesaria en muchas partes

4. La experiencia es necesaria en ambientes hostiles o difíciles
5. La experiencia mejora el rendimiento y/o la calidad
6. El sistema puede ser usado para instrucción
7. La solución que proporcionará el SE será más rápida que la que algún humano pueda proporcionar
8. El SE es más consistente y/o exacto que un humano

Además se sugiere tomar en cuenta los siguientes factores para determinar cuando es apropiado el desarrollo de un SE

1. Naturaleza del problema : El problema deberá tener una estructura simbólica y deberá haber heurística disponible para su solución. Además es deseable que la tarea pueda ser dividida.
2. Complejidad de la tarea : La tarea deberá ser ni muy difícil ni muy fácil para el experto humano
3. Alcance o extensión del problema : El problema deberá ser de tamaño manejable, además de tener un valor práctico.

5.5. ETAPAS DE DESARROLLO DE UN SE

Al igual que todo proyecto importante, la construcción de un SE debe seguir ciertas etapas. Primero se debe realizar un riguroso estudio del problema, durante esta etapa se deben comprender los siguientes pasos :

- Planteamiento de los objetivos a cubrir
- Definición de recursos disponibles (materiales, humanos y financieros)
- Establecimiento de los conceptos principales
- Estudio de viabilidad y costo

La *segunda etapa* comprende el *desarrollo formal del prototipo* del SE, el cual, comprende la construcción de la Base de conocimientos, adquiriendo el conocimiento de expertos, resultados empíricos de casos anteriores, libros de texto, artículos y boletines de información reciente, etc. El conocimiento se separa en sus aspectos declarativo y procedural. El desarrollo incluye la construcción del motor de inferencia, el área de trabajo (Blackboard), la facilidad de explicación y algunos otros requerimientos de software, tal como interfaces. Los principales participantes en esta etapa son los expertos, el ingeniero de conocimiento, y posiblemente los programadores (especialmente si existe la necesidad de una interface con otros programas de computadora). El conocimiento se presenta en la Base de conocimientos de forma tal que el sistema pueda realizar conclusiones emulando el proceso de razonamiento humano.

El proceso de esta etapa puede ser lento. Una herramienta frecuentemente usada para acelerar este proceso es SE shell, que incluye todos los componentes de un SE, pero carece del conocimiento.

Otra alternativa para agilizar este proceso consiste en evaluar si es posible utilizar para el proyecto en desarrollo algún SE existente en el mercado, lo que

supondría poder utilizar la estructura de reglas y la forma de realizar las deducciones del SE conocido. En este caso se utilizaría el SE elegido como molde y simplemente habría que adecuar las reglas del dominio del proyecto en construcción e implementarlas para, posteriormente, comprobar su validez.

Es normal que se produzcan numerosas consultas entre el jefe de proyecto y los expertos sobre el tema a fín de proponer todas las posibles variantes y mejoras mientras se construye el nuevo SE.

Después de la etapa de desarrollo y de que el sistema ha sido validado viene la *etapa de consulta,* durante la cual, el sistema es transferido a los usuarios. Cuando el sistema es consultado inicia un dialogo bidireccional con el usuario preguntándole sobre hechos específicos, después de que el usuario da su respuesta, el SE intenta hacer una conclusión. Esto es tarea del motor de inferencia, el cual decide que técnica de búsqueda usar para determinar como se aplicarán las reglas de la Base de Conocimientos para atacar el problema. El usuario puede preguntar por la explicación. La calidad de la capacidad de inferencia se determina por el método de representación de conocimiento usado y por el poder del motor de inferencia.

Ya que usualmente el usuario no tiene experiencia, el sistema debe ser fácil de usar, y aunque actualmente la mayoría de entradas son a través del teclado se considera que muy pronto esto será mediante la voz. Esta etapa también es usada por el desarrollador para probar el sistema (pueden probarse la interface y la facilidad de explicación).

La última fase es conocida como *etapa de implementación* que también comprende documentación y planes de mantenimiento para el futuro.

6. SISTEMAS DE LENGUAJE NATURAL

Después de los SE, el procesamiento del lenguaje natural (PLN) es la aplicación más extensa de la IA. Se refiere a métodos de IA para establecer comunicación con una computadora en un lenguaje humano generalmente el inglés. Hoy en día nos comunicamos con la computadora a través de comandos o programas enteros en un lenguaje de programación (En un futuro esta comunicación será establecida mediante la voz). En respuesta recibimos resultados de procesos o mensajes diversos como podrían ser los mensajes de error, peticiones de entrada, etc.

Para utilizar debidamente una computadora, se requiere aprender comandos, lenguajes, paquetes y terminología específica. Esto toma tiempo y práctica por ello a muchas personas no les parece tan atractiva y evaden su uso. Los ambientes gráficos con dispositivos apuntadores como ratones, lápices ópticos, así como las pantallas sensibles al tacto (Touch Screen), han hecho mas atractiva y amigable la comunicación con una computadora, pero aún estos dispositivos no forman parte de un lenguaje humano propiamente dicho como el que desearía un usuario sin ninguna experiencia en este campo.

Muchos problemas podrían minimizarse y tal vez eliminarse si pudiéramos comunicarnos con la computadora en nuestro propio lenguaje (como si se tratara de otro ser humano). Sin la necesidad de usar comandos o lenguajes especiales. Simplemente pulsando las ordenes, instrucciones, o información en forma de oraciones o sentencias ordinarias y que la computadora fuera capaz de interpretarlas para darnos una respuesta.

La razón para incluir el procesamiento del LN dentro de la IA es que el LN se considera una forma clara de comportamiento inteligente.

6.1. ANTECEDENTES Y DESARROLLO

Los primeros trabajos sobre el Procesamiento del Lenguaje Natural (PLN) coinciden con los orígenes de las aplicaciones informáticas a nivel comercial. En aquel tiempo se pensaba que la traducción era un proceso mecánico que las computadoras harían fácil y rápidamente. Así en 1946, Weaver y Booth presentaron el primer sistema de Traducción automática, seguido poco después por el GAT ("Georgetown Automatic Traslator") y en 1961 por CETA ("Centre d'études pour la traduction Automatique") en Grenoble.

Los sistemas desarrollados constaban de un diccionario bilingüe que relacionaba las palabras de los dos idiomas entre sí y, una gramática para cada idioma que añadía las terminaciones a las palabras y ordenaba los elementos de la oración.

Era época en que el PLN se apoyaba en métodos bastante rudimentarios como los analizadores de pattern matching, utilizados en sistemas como ELIZA, SIR, STUDENT o BASEBALL.

Los escasos avances logrados por los especialistas (Bar Hillel, Weaver, Booth, etc.) los convencen de que la comprensión del lenguaje natural precisaba algo más que un simple diccionario y una gramática.

En los años 70 comienzan a construirse las primeras interfaces en LN, primero con bases de datos y después con otras aplicaciones algunos ejemplos son: LUNAR (Woods 1970) y LADDER (HENDRIX 1974). Y aparecen sistemas más serios para el PLN como las ATN (Woods, 1970), las gramáticas procedurales (Winograd, SHRDLU 1971) y varios analizadores más de muy

diversa índole (GUS, MARGIE, RUS, SAM, PLANNER, JETS, Etc...).

Ya en los 80 aparecen las gramáticas lógicas y los formalismos de unificación que contribuyen a la construcción de sistemas más evolucionados como ATLAS e interfaces para base de datos como TEAM, CHAT-80, DATALOG.

En los 90 ya se ha detectado la necesidad de adquisición masiva de información léxica y gramatical y se han construido sistemas para el aprovechamiento de fuentes de información léxica ya existentes, como los diccionarios para uso humano en soporte informático.

Actualmente, la mayoría de sistemas de lenguaje natural usan el logro del análisis sintáctico y semántico. Los procesos son variados y complejos, pero en este capítulo solo trataremos los procedimientos básicos.

6.2. CONCEPTOS Y ASPECTOS IMPORTANTES.

Lenguaje

Sistema de comunicación que incorpora expresiones escritas y habladas para intercambiar pensamientos y sentimientos. Para esto se auxilia de una amplia variedad de sonidos, signos y símbolos para crear palabras, oraciones, párrafos, etc.

Lenguajes Formales

Son lenguajes desarrollados para un propósito especial. Un ejemplo son los lenguajes de programación que permiten una forma limitada de comunicación entre las computadoras y los seres humanos.

Lingüística

Estudia la manera en que se estructura y usa un lenguaje. Un lingüista es una persona que se especializa en el estudio de un lenguaje e investiga la organización de símbolos, palabras frases reglas y procedimientos para ver la forma en que deben ser usados.

Léxico

Contiene en forma organizada las palabras y frases comúnmente usadas en un lenguaje. Es un diccionario que lista alfabéticamente todas las palabras de un lenguaje especificando su correcta escritura y puntuación y proporcionando además sus definiciones y pronunciación. Un glosario es un subconjunto de un diccionario que define palabras términos o frases en un campo especial de interés.

Gramática

Es un conjunto de reglas para organizar las palabras y formar oraciones y pensamientos completos. Se compone de dos partes: sintaxis y semántica.

Sintaxis

Se refiere a la forma en la cual se ensamblan las palabras para formar oraciones o sentencias. Es un método para poner las palabras (sustantivos, verbos, adjetivos, etc.) en un orden específico para que tengan una organización correcta de acuerdo al lenguaje.

Semántica

Se refiere al significado dentro de un lenguaje. Estudia las relaciones entre las palabras y la forma en que son ensambladas para representar un pensamiento específico. Proporcionando las formas de analizar e interpretar lo que se quiere decir. *El objetivo de la semántica es establecer el sentido adecuado de una oración.*

Contexto

Es un marco de refererencia con respecto al cual las palabras y oraciones adquieren un significado determinado. Juntas todas las oraciones se enlazan y toman un sentido, pero en forma individual, cada oración contiene sólo una pieza de un todo y frecuentemente esta sujeta a una interpretación que puede tener múltiples acepciones. Esta interpretación es más fácil cuando las oraciones son vistas en forma conjunta. Por tanto el contexto ayuda en la clarificación del significado o mensaje.

Pragmática

Se refiere a lo que la gente realmente dice o escribe. Frecuentemente decimos o escribimos una cosa pero el significado real es otro. Por ejemplo puedes

preguntar ¿Qué hora es?, pero realmente para tí significa que se te hace tarde para una cita. El contexto y la pragmática juegan un papel principal en el entendimiento y comprensión del lenguaje. Ya que una cosa es comunicar algún mensaje, y otra saber el significado real de tal mensaje. El contexto y la pragmática llenan el hueco frecuentemente dejado por la sintaxis y la semántica.

EL PROBLEMA DE LA AMBIGÜEDAD.

Muchos problemas han surgido al tratar de que la computadora comprenda el lenguaje natural (LN) ya que, este es inherentemente ambigüo y si no se le proporcionan al programa todas las ayudas necesarias a través de conocimientos generales y específicos, así como también el contexto completo de la frase u oración, podemos tener muchas interpretaciones sobre una misma cláusula. Sumado a esto tenemos que hay convenciones de conversación entre los humanos. (ciertas respuestas no aplican con determinadas preguntas). Por ejemplo la pregunta: "¿Cuál fue el mejor vendedor del mes de abril?" no podemos contestarla con un "Si". Tales convenciones deben ser consideradas al momento de construir un sistema para PLN.

A continuación se ejemplifican algunos tipos de ambigüedad existentes en los lenguajes naturales:

1) Léxica:

a) Se sentó en el banco
b) Entró en el banco y fue a la ventanilla
c) El avión localizó el banco y comunicó su posición

¿Qué tipo de conocimiento se debe utilizar y como utilizarlo para conjeturar que (probablemente) la aparición de *banco* en (a) se refiere a un mueble, mientras que en (b) se refiere a una entidad financiera y en (c) una vez examinado el contexto se refiere a un banco de pesca.

2) Estructural o sintáctica:

El vendedor de frutas de Veracruz

¿Queremos decir que el vendedor es de Veracruz, o bien que son las frutas las que son de Veracruz?

3) Indeterminación Semántica

Dame el salario de analistas y programadores

¿Queremos el total o lo queremos empleado por empleado?

4) Referencia

Pon este paquete encima de esa mesa

En caso de que hubiera varios paquetes y varias mesas ¿Cuál paquete? y ¿encima de cuál mesa?

Para intentar solucionar esto es necesario poseer una gran cantidad de conocimiento. Por tanto algo primordial al considerar un sistema PLN es evaluar la complejidad de las oraciones que el sistema recibirá para su interpretación. Esta evaluación será de gran influencia para la elección del software y hardware necesarios para la aplicación

TECNICAS DE PROCESAMIENTO.

Existen dos técnicas que son ampliamente usadas en sistemas para PLN. Estas son :

1. *Búsqueda y Comparación de palabras* y,
2. *El análisis sintáctico y semántico.*

Búsqueda y Comparación de palabras

Como ya se dijo antes los primeros sistemas para PLN usaban este tipo de análisis. El programa recibía una sentencia de entrada y la almacenaba en un buffer. Era capaz de reconocer cuando iniciaba y terminaba una palabra por medio de la búsqueda de espacios en blanco y signos de puntuación. Al tiempo de identificar cada palabra, esta era usada en un proceso "pattern matching" el cual comparaba esta con una lista de palabras y frases prestablecidas. Por tanto cada palabra o frase que se quisiera que el programa reconociera debería ser almacenada previamente como parte del programa

Cada palabra de la sentencia o texto de entrada es comparada con aquellas almacenadas en el programa. Si la palabra no es encontrada el programa responde con un mensaje por ejemplo "No entiendo", "Por favor reescribe tu mensaje", o algún otro mensaje que provoque que el usuario pulse un nuevo mensaje o lo escriba de manera diferente. Este proceso continua hasta que la palabra es reconocida.

Por otra parte si se localiza la palabra, se selecciona para ser usada posteriormente en la conformación de una respuesta. Si hay más palabras se continua con el proceso de comparación, en caso contrario se selecciona una respuesta apropiada para ser enviada al usuario. Este proceso se presenta en la figura 6.1.

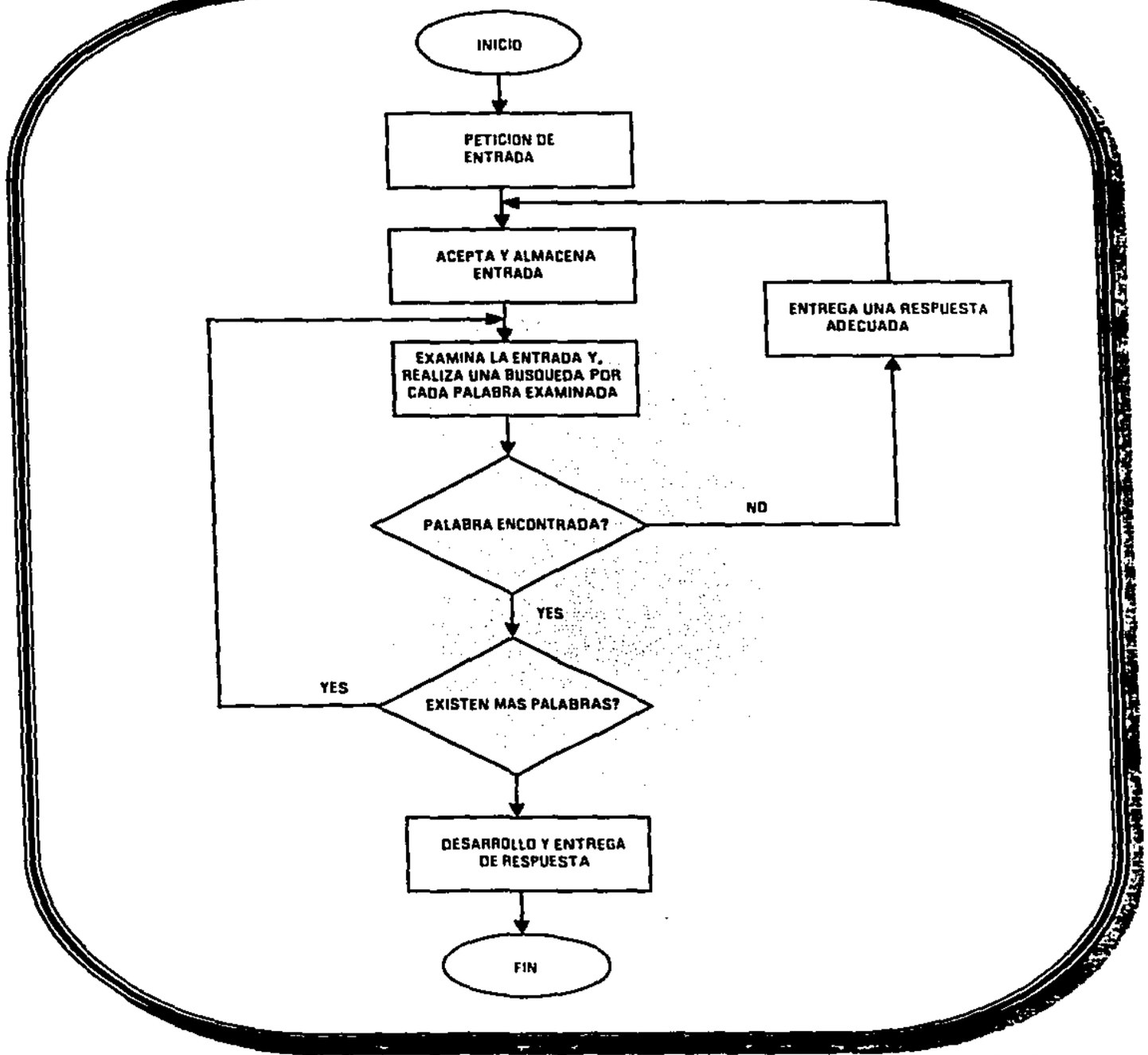


FIG. 6.1. PROCESO DE BUSQUEDA Y COMPARACION DE PALABRAS.

El método más simple para generar la respuesta es la relación de palabras seleccionadas con un conjunto de respuestas específicas almacenadas en el programa. Las palabras encontradas (palabras especiales) disparan el uso de esa respuesta. El programa se dirige a una subrutina donde la respuesta es almacenada, y posteriormente enviada al usuario.

En otros casos, se puede tener que desarrollar un proceso adicional. Por ejemplo, el programa podría recordar una porción de texto de la sentencia o frase de entrada y entonces combinarla con una porción preprogramada para formar una respuesta completa.

En otras ocasiones las palabras seleccionadas son reconocidas como especiales para que se realice una acción específica. Por ejemplo la palabra "Quiero" podría ser reconocida en una sentencia de entrada y ser una señal para que el programa respondiera con las palabras "Qué quieres ?". El usuario podría pulsar algo como: "Quiero saber más acerca de las computadoras". El programa PLN podría detectar las palabras "Quiero" y "saber" para incluir "Que quieres" en la respuesta, que podría ser "Que quieres saber de mas sobre las computadoras ?". Como observamos hay una gran cantidad de posibilidades dependiendo de las sentencias de entrada y los mensajes de salida apropiados.

Análisis sintáctico y semántico

El proceso de comparación de palabras es ampliamente usado en el lenguaje natural, pero su utilidad es restringida porque simplemente no puede hacer frente a todas las variaciones que tiene el lenguaje. Por tal razón, se han buscado y desarrollado métodos más sofisticados para analizar una sentencia de entrada y extraer el significado real de esta.

La solución más obvia y directa para el problema es la ejecución de un análisis sintáctico y semántico detallados sobre la sentencia de entrada. De esta manera se determina más exactamente su estructura y su significado.

El análisis sintáctico determina la estructura gramatical de la frase, se efectúa mediante un proceso llamado parsing o análisis de la oración, que consiste en descomponerla en sus elementos constituyentes. Una vez descompuesta la oración se procede a su interpretación semántica.

En el siguiente apartado se describe la estructura de los sistemas de LN que usan esta técnica.

6.3. ESTRUCTURA DE LOS SISTEMAS DE PLN.

El diagrama de la figura 6.2. muestra los elementos de un programa para PLN. Los principales elementos son: el Parser (analizador), el léxico, el intérprete, la base de conocimientos y el generador.

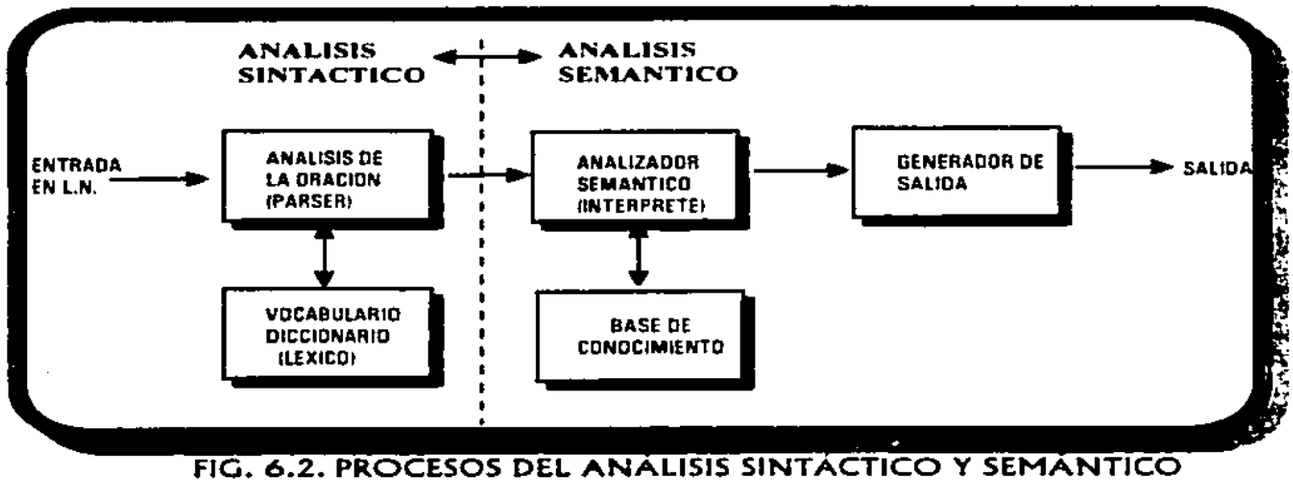


FIG. 6.2. PROCESOS DEL ANÁLISIS SINTÁCTICO Y SEMÁNTICO

Parser

El parser el primer elemento dentro de un sistema de lenguaje natural. Es una pieza o módulo de software cuya función es analizar sintácticamente la información (sentencias u oraciones) de entrada. Para esto cada palabra es aislada al tiempo que se identificada su papel o función dentro del lenguaje. Después el parser genera una estructura llamada *árbol sintáctico* en el cual se organizan las palabras para su análisis conjunto. Tal estructura mostrará la clasificación (sujeto, verbo, adjetivo, etc.) de todas las palabras y la manera correcta en que deben estar ordenadas de acuerdo al lenguaje.

El proceso completo de análisis es análogo al proceso de estructura de

oraciones que se enseña en las clases de español. El parser identifica el sujeto y el predicado para después identificar dentro de ellos mismos otros elementos. Este análisis sintáctico es el primer paso para tratar de extraer el significado de la oración u enunciado.

Recordemos que la oración (O) esta compuesta de un sujeto (S) y un predicado (P).

O –> S + P

Dentro del sujeto los nombres o sustantivos (ST) o sus sustitutos, los pronombres o sustantivos pronominales desempeñan la función de núcleo del sujeto (NS). Por tal razón a la estructura que se presenta en el sujeto se le llama frase nominal (FN).

El sujeto puede ser un sólo nombre o sustantivo pero usualmente contiene partes adicionales como son los modificadoras (M) que añaden determinaciones o calificaciones al núcleo del sujeto.

S –> M + NS + M

Ejemplo: La muchacha rubia

La frase nominal o sujeto puede incluir además de los componentes mencionados un complemento del sustantivo, el cual estará constituido por una preposición (PS) más una frase nominal en adelante llamaremos a esta combinación frase preposicional (FP).

FN=S–> M + NS + FP

Ejemplo: La hermana de ese niño

El Predicado es una frase verbal, ya que su núcleo es un verbo y puede presentar varios casos de modificadores como son: el objeto directo, objeto indirecto, y objeto circunstancial. Esta formado por un verbo (V), y dependiendo de la clase de modificador se agregan algunos de los siguientes componentes: frases nominales, frases preposicionales o adverbios (ADV).

P –> V + FN + FP + ADV

Ejemplo: Julio compró una gran casa para su familia en Puebla

El trabajo del Parser consiste en examinar cada palabra en la oración y generar la estructura que identificará todas las palabras y las ordenará correctamente. Un ejemplo se muestra en la figura 6.3.

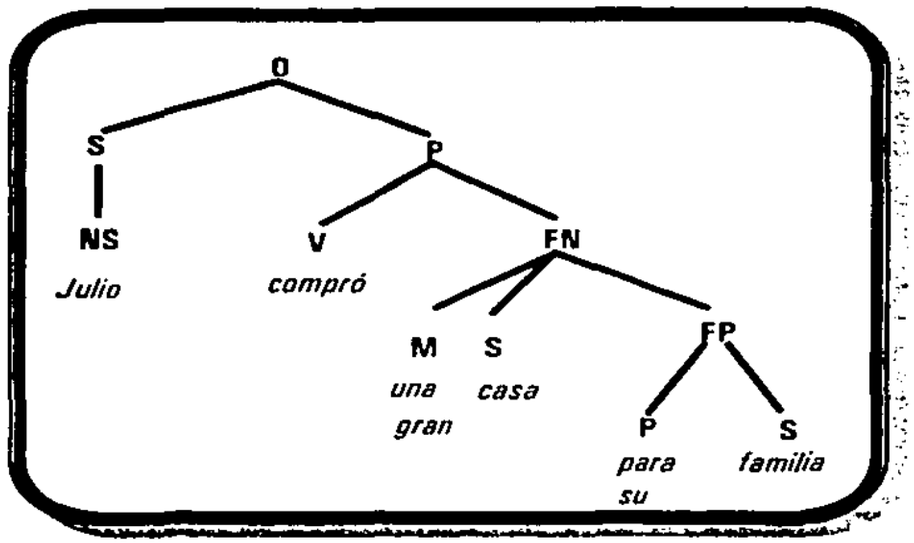


FIG. 6.3. ESTRUCTURA GENERADA POR EL PARSER PARA LA ORACIÓN DE EJEMPLO

Un popular método de análisis es la red de transición aumentada (ATN). Una ATN esta compuesta de estados o nodos que son enlazados por flechas. Las flecha representan las condiciones necesarias para pasar de un estado a otro.

Un ejemplo de ATN se muestra en la figura 6.4. Esta ATN es usada para analizar una oración compuesta de sujeto y predicado. Consideremos el siguiente enunciado :

Julio Compró una gran casa para su familia

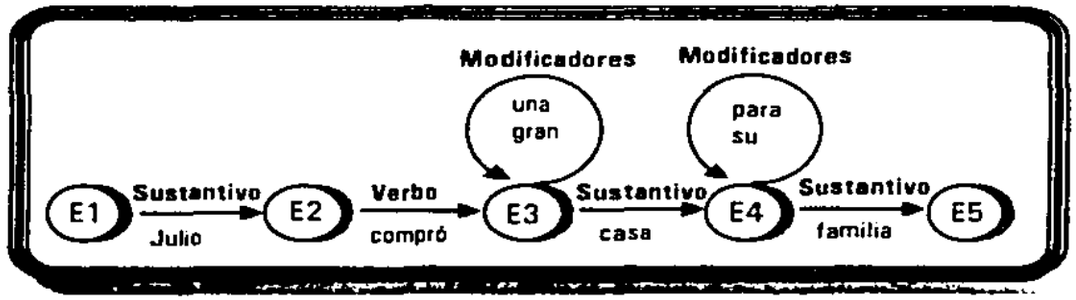


FIG. 6.4. ATN PARA LA ORACIÓN DE EJEMPLO

El estado de inicio de la ATN es E1, el cual requiere que al probar la primer palabra esta sea un sustantivo, para pasar al siguiente estado. La ATN busca en el léxico y encuentra que "Julio" es un sustantivo, con lo cual pasa automáticamente al estado E2. En la figura podrá observar las diferentes condiciones para cambiar de estado, observe que para los estados E3 y E4 existen dos condiciones, se buscan modificadores mientras existan y después se busca el sustantivo para pasar al siguiente estado. Al llegar al estado E5, el análisis de la oración se da por concluido. Esta ATN puede enlazarse o servir de referencia a otras ATN's para analizar grandes oraciones o diferentes tipos de enunciados. Gracias a esta se construyen estructuras de datos en memoria que permiten identificar todas las palabras, así como las relaciones gramaticales entre estas.

Léxico.

El parser trabaja conjuntamente con el analizador léxico para realizar el análisis sintáctico. El léxico contiene todas las palabras que el sistema es capaz de reconocer.

Tiene almacenada la escritura correcta de las palabras y además de su función (sustantivo, verbo, adjetivo, etc.) dentro del lenguaje. Para palabras que pueden tener más de un significado, abarca todos los significados permitidos por el sistema. Una de las formas de evitar el uso de ciertas palabras es limitar las entradas al analizador léxico sólo a aquellos términos que son permitidos por el sistema. Pero si se quieren usar todas las posibles combinaciones, se debe incorporar un componente adicional que ayude a determinar como será usada esa palabra o palabras. Esto significa que tal palabra debe ser relacionada con otras palabras dentro de la oración para determinar su función y significado. Los sistemas más sofisticados de PLN tienen esta capacidad.

Algunos parsers ejecutan un análisis de morfemas. Estos toman cada palabra y la dividen en sus morfemas individuales. El analizador léxico contiene varios morfemas. De esta manera, se puede determinar un significado más preciso. Otros sistemas de PLN no realizan el análisis de morfemas, pero en cambio almacenan todas las posibles variaciones de la palabra raiz. El resultado final es básicamente el mismo.

El parser y el analizador léxico trabajan juntos para seleccionar palabras de las oraciones y crear el árbol de análisis respectivo pero aunque se ha realizado completamente el análisis sintáctico, la computadora no entiende esto. Por tanto se requiere de un análisis semántico.

Durante su operación, el parser juega el papel de un patrón de comparación. Una vez que una palabra ha sido identificada, El parser busca a través del analizador léxico comparando esta con todas las palabras almacenadas allí. Si la encuentra, es apartada con el resto de información ya identificada. El parser continua el análisis con la información restante y por último construye el árbol sintáctico.

Con esto el sistema esta listo para realizar el análisis semántico.

Intérprete y Base de Conocimientos

El intérprete trabaja junto con la base de conocimientos para determinar el significado de las sentencias u oraciones. La base de conocimientos conceptualmente puede ser dividida en dos partes Base de Conocimiento General (contiene conocimientos del mundo en general y conceptos básicos de lingüística) y Base de conocimiento específico.

Para determinar el significado de una oración, el sistema debe saber cosas sobre las palabras y como se enlazan para formar oraciones comprensibles. La base de conocimientos es el medio principal para comprender lo que ha sido dicho. Recordemos que hay diferentes formas de implementar una base de conocimientos (reglas de producción, redes semánticas, marcos y guiones)

La gran cantidad de conocimiento contenido en las bases general y específica impone restricciones sobre posibles interpretaciones del árbol sintáctico, y múltiples desambigüedades, dando significados apropiados.

El propósito del intérprete es usar el árbol sintáctico para consultar la base de conocimiento. De este modo el interprete puede responder preguntas

usando la base de conocimientos. Si la entrada es una oración, el intérprete determina el significado mediante la búsqueda de palabras y frases en la base de conocimientos. Se podrían disparar reglas, marcos o scripts referenciados, o redes semánticas buscadas.

El intérprete puede además realizar inferencias desde la oración de entrada. Muchas oraciones no proporcionan un pensamiento completo, pero los humanos son capaces de inferir su significado gracias a su conocimiento general. Este es el caso de algunos sistemas de PLN más sofisticados.

Generador

El generador usa una entrada comprendida para crear una salida útil. El intérprete crea otra estructura de datos que representa el significado e interpretación de la oración y almacena esta en memoria. Esa estructura de datos puede entonces ser usada para iniciar una acción adicional. Si el PLN es parte de una interface, la estructura de datos será usada para crear códigos especiales y controlar otra pieza de software. Esta puede dar los comandos necesarios para iniciar alguna acción. En un SMBD, el generador podría escribir un programa en un lenguaje de consulta (query) para empezar la búsqueda de información específica. En otros sistemas, la acción puede ser la generación de una respuesta tal como una oración o una pregunta.

En su forma más simple el generador estándar, produce salidas basadas en el significado de la sentencia de entrada. Sin embargo, los generadores más sofisticados entregan respuestas verdaderamente originales.

En el modelo presentado en la figura 6.2., los bloques individuales muestran los elementos del sistema y como trabajan juntos. Se describe como primer tarea el análisis sintáctico y después el análisis semántico. En la

práctica hay sistemas que trabajan de esta manera, sin embargo las investigaciones y experimentos han mostrado que los sistemas que combinan los dos análisis tienen rendimiento superior. En otras palabras cuando se inicia el análisis sintáctico, al mismo tiempo se realiza el análisis semántico. Esto ayuda ayuda a minimizar las ambigüedades. Tal integración puede contener todos los bloques mostrados en la figura pero ellos están enlazados para comunicarse uno con otro palabra por palabra. De esta manera, la interpretación es mejor y el proceso es más rápido.

6.4. APLICACIONES

Los sistemas de PLN han encontrado aplicación en varias áreas. Las más importantes son: Interfaces PLN, traslación de un LN a otro LN, traslación de un lenguaje de computación a otro lenguaje de computación, análisis de gramática y manejo de documentos

El mayor uso del PLN son las interfaces de software empleadas para simplificar y mejorar la comunicación entre un programa de aplicación y el usuario. Evita el invertir gran cantidad de tiempo en el aprendizaje de lenguajes y comandos especiales.

Las interfaces de LN permite al usuario operar el programa de aplicación o desarrollar aplicaciones propias con lenguaje natural sin tener que aprender comandos especiales. Por tanto, hace que el software sea transparente para el usuario que se enfocará hacia su trabajo ya que evitará la tediosa tarea de aprender procedimientos o lenguajes especiales. De esta manera toda clase de usuarios inexpertos serán capaces de usar la computadora.

El uso más común de las interfaces de LN se encuentra en los DBMS que almacenan y restauran enormes cantidades de información. Como es sabido la consulta y manipulación (búsqueda, ordenamiento, etc.) a tal información se realiza a través de instrucciones especiales que usualmente forman parte de un lenguaje conocido como query el cual nos permite accesar los datos deseados y darles el formato requerido. Esto hace que la mayoría de DBMS no sean tan accesibles para usuarios inexpertos en el manejo de BD y que el manejo de DBMS lo tengan que hacer programadores o usuarios con

alguna experiencia, lo cual limita la utilidad de la BD.

Actualmente, gran cantidad de software DBMS incluye u ofrece una opción de interfaz en LN. Como resultado, virtualmente cualquier persona puede sentarse frente a una computadora y disponer de la BD simplemente tecleando en detalle lo que desea como salida.

INTELLECT.

Es una de las interfaces en LN más viejas y más ampliamente usados. Es una interface para mainframes y fue diseñada primariamente para ser usada con DBMS en ambientes de sistema operativo IBM.

Permite al usuario crear BD usando un LN. El léxico incorporado puede ser modificado para una aplicación en particular. Un editor permite al usuario construir y mantener diccionarios especiales para aplicaciones específicas. Solo se pueden agregar palabras significados o sinónimos.

En ocasiones la interface accede directamente a la BD. Pero además puede generar código en uno de los numerosos lenguajes query usados para accesar información en un DBMS. Por ejemplo INTELLECT está disponible en una configuración que usa un lenguaje query llamado FOCUS. INTELLECT permite al usuario pulsar una sentencia en LN y luego la transforma a comandos de FOCUS que accesarán los datos deseados.

INTELLECT es probablemente la interface de LN más vendida para mainframes; sin embargo otras compañías ofrecen sistemas similares. Otro sistema para mainframes es EXPLORER que es una interface de lenguaje natural para BD.

Algunos de los competidores fuertes de intellect son los siguientes:

RAMIS II ENGLISH (Mathematica, Inc.).
SPOCK (Frey Associates, Inc.)
NaturalLink (Texas Instruments)
BBN Parlance (BBN Labs, Inc.)

Interfaces Incrustadas

La tendencia en las interfaces de LN es incorporarlas dentro de las aplicaciones para evitar un costo extra. Un ejemplo de esta inclinación es Paradox. Lotus 1-2-3, versión 3.2, también incluye algunas capacidades de PLN.

PARADOX

DBMS relacional con tecnología de IA. No tiene capacidad de PLN, pero tiene una capacidad de inferencia que permite al usuario recuperar información de la base de datos con solo un rápido conocimiento de detalles específicos. A los usuarios se les presenta una representación gráfica de un registro vacío llamado tabla de query, la cual tienen que llenar con la información similar a la que desean buscar o consultar dentro de la base de datos. Paradox analiza esta información de entrada e infiere cual información esta buscando el usuario y ejecuta una acción apropiada desplegando sobre la pantalla una tabla de información sobre los datos encontrados. Los usuarios pueden estipular que campos deberían estar incluidos en el reporte y/o realizar cambios en la tabla de query. Paradox proporciona dos ventajas: fácil de usar y correcta interpretación de lo que el usuario requiere. El programa usa una vía de optimización de consulta heurística, que mejora la eficiencia de la búsqueda en la BD.

Otro tipo de paquetes de software están esperando incorporar interfaces en LN. Las hojas de cálculo, por ejemplo, podrían ser más fáciles de utilizar. Un ejemplo es HAL (Human Access Languaje), el cual viene con Lotus 1-2-3 (versiones 3.1 y posteriores). Permite al usuario realizar varias funciones de programa solamente con teclear comandos en inglés.

Otra opción son las interface en LN para sistemas operativos. Ya se ha creado un programa usando técnicas de IA para el sistema operativo UNIX. Que como ya sabemos es un sistema para equipos grandes con capacidades multiusuario y multitarea. Aunque hay versiones disponibles para PC como XENIX. UNIX es un sistema operativo poderoso y de gran utilidad, sin embargo es un poco difícil de usar por su compleja estructura de comandos. La interface de LN llamada UNIX Consultant ha sido desarrollada para permitir al usuario preguntar en inglés como realizar acciones específicas en UNIX. Por ejemplo, para borrar un archivo, debemos usar el comando "rm" seguido por el nombre del archivo. Si se ignora hacer esto, simplemente se puede preguntar en inglés "How do I eliminate a file ?" UNIX consultant responderá tu pregunta. el programa por si solo no ejecutará la acción, pero te dirá como hacerlo.

INTERFACES PARA SISTEMAS EXPERTOS

El PLN es especialmente importante para los SE. La mayoría de las siguientes actividades se benefician grandemente de una interface en LN: Manejo de diálogos con usuarios inexpertos, consultas, explicación y recomendación en términos de usuario.

Como se menciono anteriormente, el PLN empezó a usarse en BD. Tal combinación requiere extensivas definiciones de términos de negocios en el diccionario PLN. los SE que ya tienen conocimiento almacenado sobre esos

términos en su base de conocimientos pueden ser agregados para incrementar las capacidades del sistema.

La figura 6.5., muestra una estructura antes y después de tal combinación. El inciso a) muestra el uso típico de una BD con un DBMS; el usuario debe saber el lenguaje del DBMS, lo cual le puede tomar semanas. La parte b) agrega el PLN, lo cual, hace la comunicación más fácil y menos limitada. El diccionario de PLN debe prepararse para entender la diversidad de lenguaje natural (por ejem: terminología de negocios) y trasladarlo a comandos del DBMS.

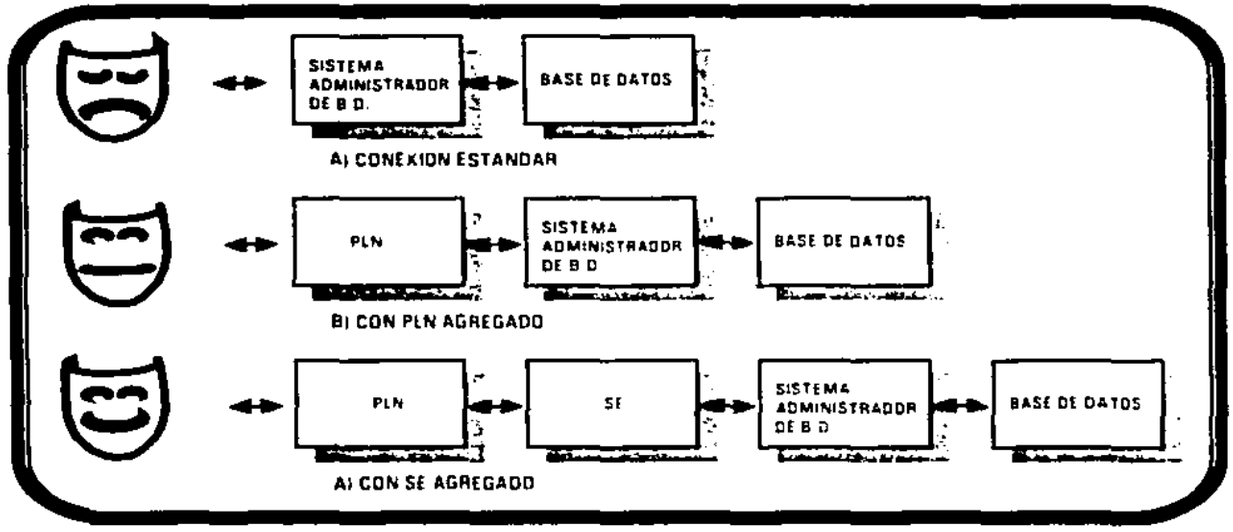


FIG. 6.5. FACILIDAD DE COMUNICACIÓN DE ACUERDO A COMPONENTES AGREGADOS.

La conexión del PLN y el DBMS puede ser muy difícil especialmente cuando se usan grandes computadoras. Aquí es donde el SE puede ser de gran ayuda. Tal como se muestra en la parte c), un SE puede mejorar la comunicación permitiendo que el PLN use el conocimiento del SE. UN producto comercial que combina SE y PLN es "Conversational Advisory System".

INTERFACES PARA PROGRAMAS DE APLICACION

El PLN puede ser usado como una interface para programas de aplicación. Por ejemplo, El Banco de Seguridad del Pacífico desarrollo una interface para su sistema de manejo de dinero (SPACIFICS). EL sistema brinda a los clientes del banco la capacidad para monitorear, manejar, e iniciar actividades sobre canales de comunicación, en inglés. Los usuarios no necesitan aprender la compleja interface. El programa también genera reportes.

En adición a las herramientas específicas para BD y hojas de cálculo, existen más herramientas PLN de propósito general. Un ejemplo es NL-Builder (Synchronetics, inc.), este producto, el cual esta disponible para computadoras personales y minicomputadoras, es un shell que permite la construcción de una variedad de aplicaciones de LN que entienden la traslación entre lenguajes.

7. ROBOTICA

INTRODUCCION

Desde tiempos antiguos se han inventado máquinas que han ayudado a realizar el trabajo de una manera mas fácil y rápida. La complejidad de tales máquinas ha avanzado de acuerdo al desarrollo de la ciencia y la tecnología, lo cual las ha hecho más productivas y precisas.

Las máquinas son dispositivos que nos ayudan a multiplicar nuestras capacidades. Aunque en algunas tareas, entregan productos totalmente terminados sin requerir de la intervención humana. Este proceso se conoce como *automatización.*

La idea que concibió a un robot, fue la creación de una máquina de propósito general que pudiese realizar muchas de las funciones físicas de un humano. Por tanto un robot es una máquina que intenta duplicar algunas de las capacidades físicas de una persona. Así como la IA intenta emular el cerebro del hombre, un robot intenta emular sus capacidades de manipulación física. Las dos ideas van de la mano.

Los sistemas basados en sensores, tales como sistemas para visión, tacto y procesamiento de señales, cuando se combinan con IA definen una amplia categoría de sistemas conocida como robótica. Un robot es un dispositivo electromecánico que puede ser programado para ejecutar tareas manuales.

No todos los robots son considerados como parte de IA. Un robot que realiza solo las acciones para las que ha sido programado se considera un robot tonto, pues no posee más inteligencia que la de un lavaplatos. Un robot inteligente emplea alguna clase de aparatos sensoriales, tal como una cámara, que percibe la información sobre la operación del robot y su medio

ambiente. La parte inteligente del robot le permite responder y adaptarse a los cambios en su medio ambiente, aunque solo siga instrucciones despreocupadamente.

La diferencia entre una máquina automática y un robot inteligente es que el robot siente su medio ambiente y modifica su comportamiento como resultado de la información percibida. El robot inteligente esta pensado para tener capacidades y atributos que asemejen a las de los humanos. Por ejemplo, algunos robots son distinguidos de la automatización común por su habilidad para tratar con la incertidumbre.

7.1. QUE ES UN ROBOT ?

Al hablar de un robot generalmente se piensa en una máquina electromecánica diseñada para emular tanto como sea posible los atributos físicos y mentales de un humano. La máquina que se imagina probablemente sea parecida a un humano. Esta idea se basa en películas y/o historias de ficción. Robots de este tipo son conocidos como androides.

Androides y Cyborgs.

Los androides son sintéticamente humanos con partes mecánicas y cerebro electrónico. Un buen ejemplo de androides es R2D2 de la película la guerra de las Galaxias.

Otro término muy escuchado y asociado con la robótica es cyborg. Un cyborg es un humano al cual se le ha implementado algún dispositivo electromecánico. Un cyborg usa partes parecidas a las de un robot para corregir una deficiencia física o para incrementar sus capacidades físicas o mentales. Tal humano se robotiza parcialmente. Se piensa que nunca se producirá un duplicado electromecánico idéntico al hombre. Pero se podrá crear un verdadero androide de gran utilidad.

Se han hecho muchos intentos para definir el término robot. La mejor definición es probablemente la que ha emergido de las manufactureras y usuarios de robots industriales. El Instituto de Robótica de América define un robot industrial de la siguiente manera: Un robot es un " *manipulador multifuncional reprogramable, diseñado para mover materiales, partes, herramientas, u otros dispositivos especiales a través de varios movimientos previamente programados para ejecutar una variedad de tareas*".

7.2. TIPOS DE ROBOTS INDUSTRIALES

En la industria se distinguen dos tipos básicos de robots: el brazo móvil, y el robot totalmente movible.

Robot de Brazo Móvil

Es el robot industrial predominante, este tipo de robot se encuentra en el lugar donde va a ejecutar su trabajo y generalmente se asegura al piso o se monta sobre una base de tal forma que no pueda moverse. Ya que queda bien fijo el robot, que asemeja a un brazo, puede tener los siguientes movimientos: hacia arriba o hacia abajo, hacia atrás o hacia adelante según haya sido programado.

En algunas aplicaciones especiales, el brazo puede tener un movimiento total. Para esto puede montarse sobre vías y deslizarse a lo largo de una línea de producción.

Robots Móviles

Un robot móvil es capaz de mover todas sus partes (articulaciones). Estos robots son generalmente brazos manipuladores montados sobre una pequeña estructura movible con ruedas. En circunstancias especiales se pueden usar otros métodos de propulsión. Por ejemplo, se han inventado robots con piernas, las cuales, les proporcionan gran movilidad en terrenos difíciles. Existen robots acuáticos que usan un motor de propulsión y un control para movimiento de aletas que les permitan desplazarse en tal medio. Independientemente del tipo de movimiento implementado, los robots móviles deben tener un sistema de propulsión. Comúnmente, este sistema de propulsión consiste de uno o más motores eléctricos que operan con una batería.

Un robot móvil se desplaza independientemente hacia donde hay que realizar el trabajo, colocándose en el lugar adecuado para ejecutarlo. Por esta razón, los robots móviles, son excelentes para reemplazar al hombre en ambientes difíciles de trabajo. Pueden reemplazar al hombre dentro de plantas nucleares, campos minados, fondo del mar, el espacio exterior, etc. Además de poder ser usados en casos de incendios y amenazas de bomba.

La mayoría de robots móviles son controlados por un operador mediante un extenso cable. En otros casos, son controlados por radio. El operador usualmente puede observar cuando el robot ejecuta la tarea, colocándose a distancia por cuestiones de seguridad. Los sensores del robot proporcionan señales de retroalimentación para que se realicen los ajustes correspondientes.

En ambientes donde el operador no pueda observar el trabajo. El robot debe enviar una señal que comunique su localización, posición y estado. Generalmente, para este propósito, se usan cámaras de vídeo. Y con la información visual proporcionada, el operador usa el radio control para operar al robot como sea requerido.

7.3. COMPONENTES Y OPERACION

Para comprender la importancia de la IA en los robots del futuro, necesitamos saber como operan los robots en la industria. A continuación se explicará cuales son y como operan los principales componentes de un robot industrial.

El típico robot industrial consiste de cuatro componentes : el brazo manipulador, el dispositivo final, el impulsor, y el controlador. Los robots pueden configurarse de diferentes formas mediante variaciones y combinaciones de estos elementos básicos.

Brazo Manipulador

El brazo manipulador intenta ejecutar un trabajo similar al que realiza el brazo de un humano. Este frecuentemente consta de: hombro, codo y muñeca. El brazo puede rotar o deslizarse con una flexibilidad total. Existen una variedad de formas de diseñar brazos manipuladores. En la industria se emplean cinco tipos básicos. Sus nombres reflejan el método mediante el cual alcanzan cualquier punto dado dentro del espacio definido por su alcance.

El brazo más simple se muestra en la figura 7.1a., se basa en coordenadas cartesianas o rectangulares. La idea es mover el componente final del brazo a una localización dada en el espacio para ejecutar alguna función. Este punto en el espacio esta definido por los ejes x, y, z de un sistema de coordenadas rectangulares. El brazo del robot puede moverse de izquierda a derecha, de arriba hacia abajo y de adelante hacia atrás. Como el brazo se mueve en tres posibles direcciones (dentro de sus límites), traza la figura de un cubo en el espacio de trabajo. El dispositivo final, por supuesto, puede colocarse en cualquier lugar del cubo.

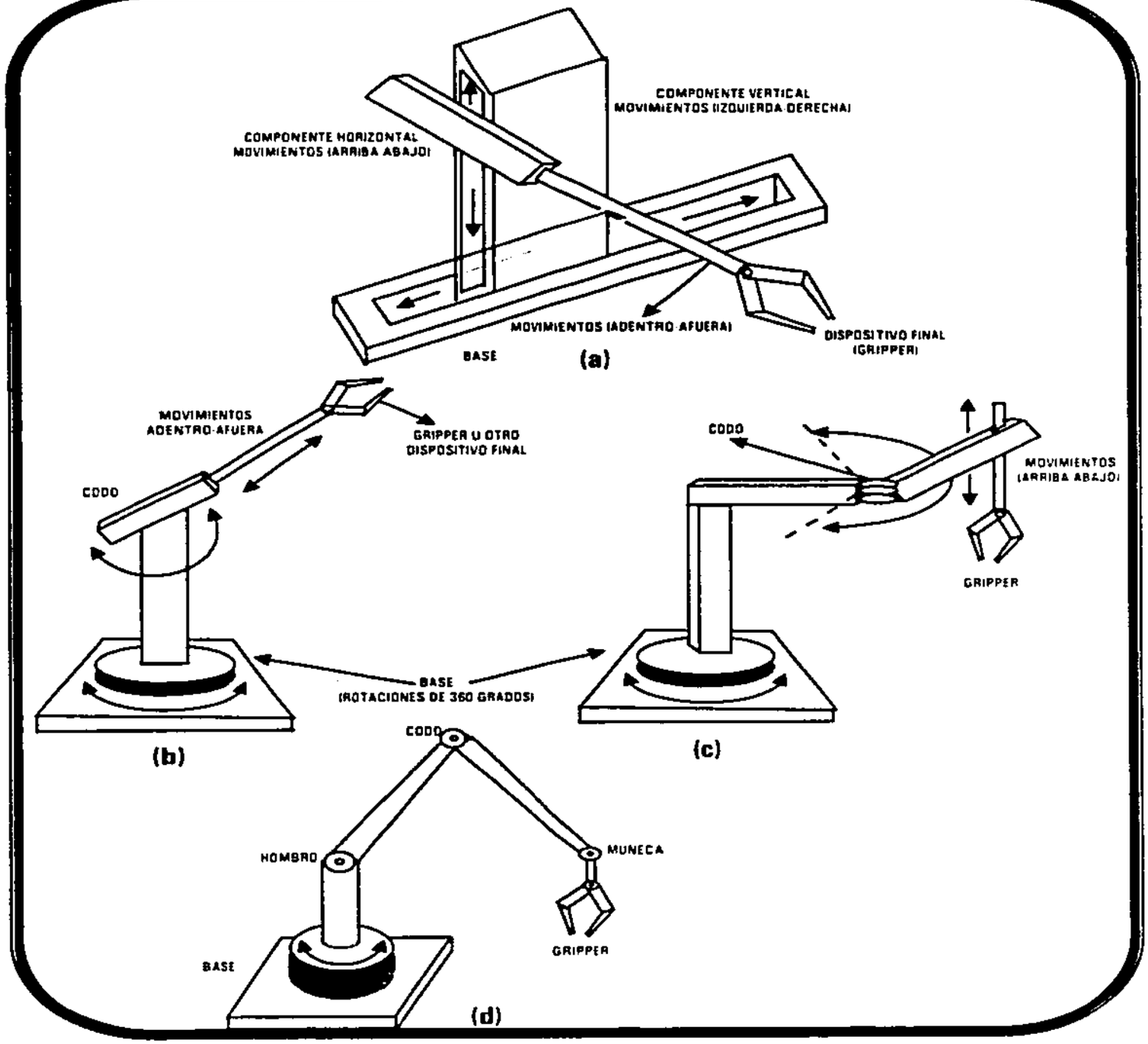


FIGURA 7.1. VARIEDAD DE BRAZOS MANIPULADORES

Otra variación del brazo manipulador emplea coordenadas cilíndricas, con lo cual, el brazo entero puede rotar 360 grados.

El sistema de coordenadas esférico o polar, representa la tercera variación. Obsérvelo en la fig. 7.1b. El componente vertical, es reemplazado por un enlace que emula al hombro. El movimiento vertical se logra por medio de la rotación sobre este componente. Por tanto, el brazo entero puede girar 360 grados, además de poder hacer movimientos hacia adentro y hacia afuera (eje z). Este movimiento trazado fuera del espacio es lo que representa la porción de una esfera.

Un cuarto tipo de brazo es el conocido como articulado (figura 7.1d.), al igual que el sistema de coordenadas esféricas o polares, este es capaz de rotar 360 grados, pero, además posee articulaciones que asemejan al hombro, al codo y a la muñeca humanos, gracias a esto el brazo adquiere mayor flexibilidad de movimiento, por lo cual, puede trazar porciones mas grandes de una esfera. Este tipo de brazo es probablemente el más usado en la industria debido a su gran flexibilidad.

Un quinto tipo de brazo se ilustra en la figura 7.1c. Este es conocido como SCARA (Selective, Compliance Assembly Robot Arm). Su área de trabajo traza un cilindro en el espacio. Al igual que la mayoría de los otros brazos, este puede rotar 360 grados. Pero gracias a la articulación que emula al codo puede realizar movimientos horizontales y verticales. Esto le permite doblarse sobre sí mismo en ambas direcciones. Los brazos de este tipo en ocasiones son usados para realizar trabajos de ensamblado que requieren de grandes velocidades (por ejemplo, piezas de equipo electrónico).

Dispositivo final

Este componente aparece al final del brazo, para ejecutar tareas específicas (soldado, atornillado, etc.). La mayoría de estos dispositivos se diseñan especialmente para el trabajo que realizarán, se acoplan al brazo y frecuentemente son intercambiables.

Uno de los tipos mas populares de dispositivos finales es el gripper. El cual, es una estructura tipo pinza, que simula un pulgar y un dedo. Esto hace al robot capaz de tomar objetos de un tamaño determinado. Aunque existen estructuras más sofisticada que cuentan con tres o más dedos para agregar flexibilidad.

Algunos otros agregan sustancias especiales que faciliten la adhesión de objetos; o mangueras de succión para atraer objetos ligeros.

El dispositivo final se coloca normalmente en la articulación que hace las veces de muñeca y proporciona varias capacidades adicionales, la muñeca puede rotar hacia arriba o hacia abajo y hacia al frente o hacia atrás. Una vez que el brazo ha sido colocado, estos movimientos adicionales de la muñeca proporcionan mayor flexibilidad para realizar el trabajo.

Todos los métodos de movimiento del brazo y la muñeca se clasifican por el grado de libertad que poseen sobre sus ejes. El grado de libertad se refiere al numero de ejes sobre los cuales los componentes del robot pueden moverse. Como observamos en los tipos de brazos ellos tienen tres grados de libertad. Con la incorporación de la muñeca se adquieren tres ejes más. Por tanto, se obtienen seis grados de libertad, que permiten una excelente flexibilidad. Los robots modernos pueden tener veinte o mas grados de

libertad.

Impulsor

El impulsor, es la fuente básica de energía para el movimiento del brazo, controla las uniones, y operaciones de el dispositivo final. Se emplean tres tipos básicos de fuentes para impulsión: **eléctrica, hidráulica y neumática.**

Los impulsores eléctricos son motores de AC o DC que proporcionan la energía para la rotación o movimiento. Estos se usan en combinación con partes mecánicas que pueden ser engranajes o poleas para permitir los movimientos básicos. Normalmente existe un motor por unión. Un alto porcentaje de brazos para robot son activados por motores eléctricos, los cuales pueden ser adecuadamente controlados por circuitos eléctricos.

Otra fuente de energía comúnmente usada es un sistema hidráulico consistente de una bomba con algún líquido, tal como aceite bajo presión. A través de válvulas y otros mecanismos, este líquido puede controla todos los movimientos del brazo. Los sistemas hidráulicos son extremadamente poderosos. Pues son capaces de levantar mayores pesos que los sistemas eléctricos, por tal razón, son usados en aplicaciones que requieren mayor resistencia y durabilidad. Aunque algunas veces es posible la combinación entre el los sistemas eléctricos e hidráulicos dentro de un brazo.

Pocos brazos emplean sistemas neumáticos como fuente de energía. En estos, el aire comprimido es conducido a través de líneas para producir movimiento. Los sistemas neumáticos son débiles y se usan para trabajos que requieren tareas muy ligeras. Pero existen otros tipos de energía, tal como, la solar que podrían prevalecer en el futuro.

Controlador

El cuarto componente en un sistema robotizado. El controlador tiene la función de enviar señales a la fuente de propulsión con objeto de mover el brazo hacia una posición específica y activar el dispositivo final.

Los controladores generalmente son circuitos electrónicos. Que pueden ser controladores programables especiales, útiles para proporcionar al robot una secuencia de instrucciones para movimiento y acciones específicas. Pero, el controlador más flexible es una computadora, la cual permite que el brazo la reprogramación versátil para una amplia variedad de aplicaciones.

Los controladores más sofisticados son servomecanismos, que proporcionan mayor control sobre el brazo y su posición. El servo-controlador usa sensores que envían señales de retroalimentación hacia el controlador acerca de la posición de componentes determinados. El servomecanismo emplea esas señales para determinar la posición actual del brazo, y el momento en que alcanza la posición deseada.

Programación

Para lograr que el brazo realice tareas específicas se requiere de programación. La cual, permite proporcionar al controlador el conjunto de coordenadas de inicio y de terminación, así como una ruta en partícular, en caso de haberla.

Ya que los robots más modernos usan computadoras como controladores. Se han desarrollado lenguajes especiales de programación para control de robots, los cuales, agilizan y simplifican el proceso de decirle al robot lo que tiene que hacer, como y cuando.

7.4. PORQUE UTILIZAR DE ROBOTS ?

Existen tres razones importantes para el empleo de robots: incrementar la productividad, reemplazar humanos en ambientes difíciles de trabajo, reducir costos e incrementar la calidad de vida.

Incremento de la productividad. Negocios e industrias están muy consternados por la productividad de sus trabajadores, pues requieren de una mayor cantidad de trabajo en el menor tiempo. Pero, la alta productividad es difícil de mantener, ya que los organismos humanos se agotan, se aburren o se enferman; necesitan periodos vacacionales y tiempos de descanso; tienen la necesidad de comer y platicar. Todo esto aunado a que en otras ocasiones trabajan muy lentamente.

En cambio los robots no exhiben ninguna de estas deficiencias, ya que, trabajan continuamente, no requieren de descanso, no son afectados por situaciones emocionales o acciones desmotivadoras. Ellos simplemente realizan su trabajo día tras día. Por estas razones los robots son ideales para labores de ensamblado donde la misma tarea se ejecuta una y otra vez.

Trabajo en ambientes difíciles. Muchos trabajos en la industria son peligrosos para los humanos, ya que, los trabajadores están expuestos al manejo de materiales peligrosos incluyendo objetos cortantes, piezas de metal pesadas, temperaturas elevadas, químicos tóxicos, y altos niveles de ruido. A pesar de se tomen todas las precauciones posibles en tales ambientes, los accidentes ocurren y los trabajadores resultan heridos o muertos. Tales ambientes son apropiados para robots - ellos pueden ayudar a eliminar accidentes, y evitar todas las consecuencias, desperdicios y gastos.

Reducción de costos y calidad de vida. Los robots además permiten ahorrar dinero. Y aunque el costo de construcción e integración del robot a su ambiente de trabajo es considerable, ese costo es eventualmente recuperado. A la largo plazo, resulta más económico que pagar a un humano para que realice el trabajo.

7.5. ROBOTS INTELIGENTES

La mayoría de robots son tontos, pues no pueden pensar o adaptarse a su medio ambiente; solo ejecutan acciones preprogramadas. Tales robots son útiles, pero su empleo esta limitado a situaciones donde la inteligencia y la adaptabilidad no son necesarias.

Al hablar de "*inteligencia*" en relación a los robots se piensa en el uso de la información para modificar acciones. Los robots inteligentes intentan emular las capacidades de sensibilidad y toma de decisiones, además de que pueden adaptarse por si solos a condiciones de incertidumbre y modificar sus acciones. Un sistema inteligente de robótica debería de ser capaz de interactuar con su medio ambiente y adaptar su comportamiento de acuerdo al cambio de condiciones.

Los sistemas inteligentes de robótica ofrecen beneficios potenciales a operaciones de manufacturación, superando el rendimiento en determinadas tareas, por ejemplo pueden:

- Reducir los requerimientos de partes exactas o accesorios especiales mediante la adaptación a variaciones y desalineamientos ;
- Permitir la automatización para el diseño de productos ; y
- Ejecutar una compleja combinación de tareas que de otra forma requeriría de múltiples herramientas y operaciones.

Los robots inteligentes son el mejor ejemplo de la aplicación de varías tecnologías de IA con objeto de lograr un sistema automatizado de manufacturación. Todas las ramas de IA, especialmente los SE y la visión computarizada, se pueden aplicar a la robótica. Estas técnicas se unen para proporcionar al robot percepción y pensamiento.

Como se proporciona la inteligencia

La figura 7.2. es un diagrama a bloques que representa los principales componentes de un robot inteligente. El ambiente de trabajo representa los objetos a ser manipulados. El bloque que representa el brazo del robot incluye el dispositivo final y el sistema de impulsión. El brazo es manipulado por la computadora mediante una interface.

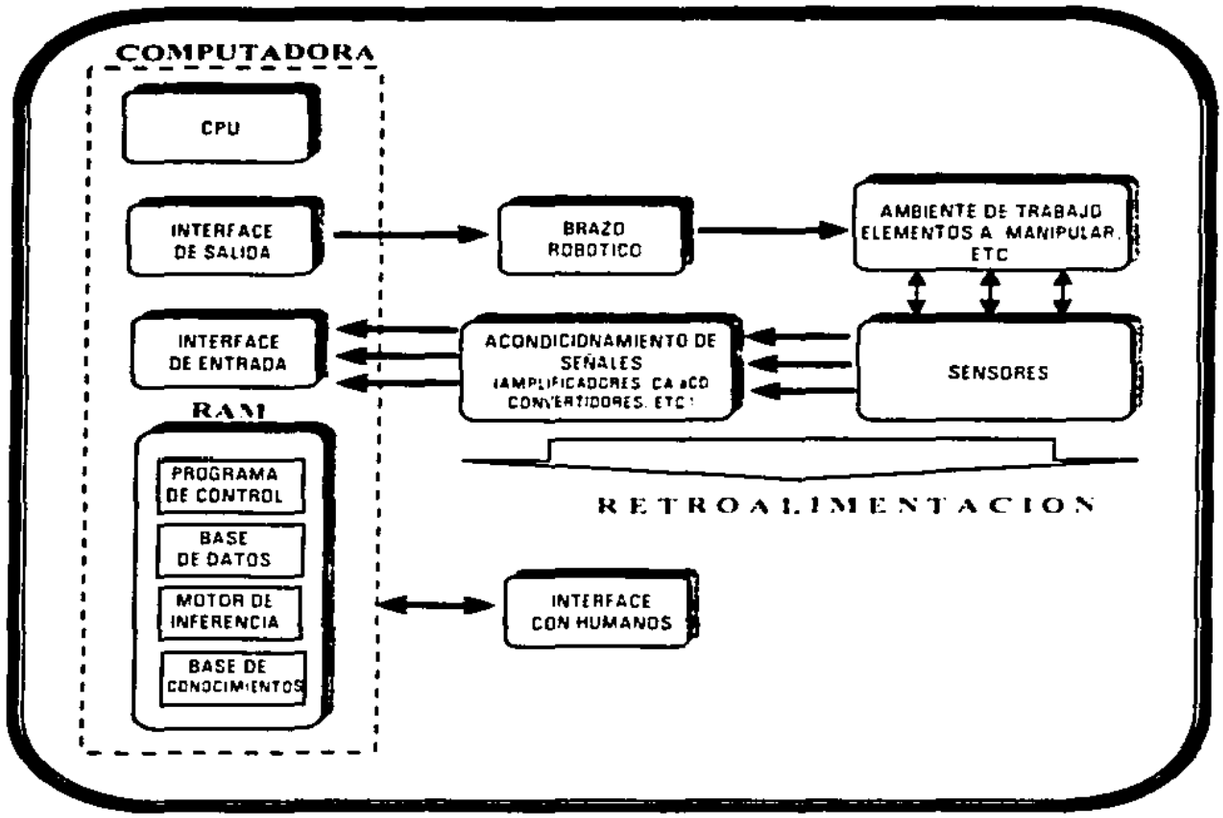


FIGURA 7.2 COMPONENTES BASICOS DE UN ROBOT INTELIGENTE

Los sensores monitorean varias condiciones del ambiente de trabajo. Las señales eléctricas generadas son convertidas de AC a DC, para hacerlas compatibles con la interface de entrada a la computadora. Esta parte del sistema es el mecanismo de retroalimentación.

La operación inicia con sensores que monitorean el ambiente y generan señales eléctricas y datos binarios compatibles con la computadora. La información binaria se almacenada en la memoria de la computadora (RAM). Estos datos representan señales de retroalimentación que le dicen a la computadora lo que esta ocurriendo.

En el momento en que los datos de retroalimentación son analizados, se hace referencia a la base de conocimientos para tomar decisiones. Las salidas del mecanismo de inferencia modifican el programa de control, y este envía señales al robot mediante la interface de salida. Los sensores pueden entonces percibir cambios que modifiquen la base de datos. Con nuevos hechos a considerar, el motor de inferencia usa la base de conocimientos para tomar decisiones sobre que cambios hacer en el programa de control a fin de corregir cualquier desviación. La inteligencia de los robots no se debe a los sensores, si no, a la interpretación de la información y la capacidad de toma de decisión de un SE integrado. Para que los robots exhiban un alto nivel de inteligencia, deben ser capaces de aprender, y esta capacidad la proporcionará un SE

7.6. APLICACIONES

Los robots ejecutan una variedad de tareas. A continuación se mencionan algunas de funciones comunes.

Aplicaciones Generales

Generalmente los robots se emplean para tareas de manufacturación tales como: soldadura, manipulación de materiales, ensamblado y pintura.

Soldadura. Es la aplicación más frecuente de los robots. Los brazos se colocan a lo largo de la línea de producción, se mueven y colocan marcas para soldar en localizaciones predeterminadas. Hoy en día la mayoría de cajas y marcos son hechas usando robots soldadores. El brazo robótico puede colocarse el mismo en el lugar exacto y realizar el soldado en la marca designada. Esto lo hace de una forma mas rápida y mas consistente que la gente.

La tarea de soldado es realizada mucho mejor por los robots que por los humanos. Esta actividad es peligrosa y requiere concentración y larga experiencia. Además en algunas ocasiones debido a el lugar donde se requiere soldar, el trabajador tiene que adoptar posiciones muy difíciles. Además de que la inhalación de los humos y gases propios de esta actividad pone en riesgo la salud humana.

Manipulación de materiales. Otra aplicación es la transferencia de materiales, componentes y piezas de un lugar a otro empleando un robot. Estos brazos son generalmente referidos como robots de " toma y coloca ". El brazo robótico es usado para cargar o descargar partes a nuevas localizaciones, para mover componentes desde una área de trabajo a una correa transportadora o

viceversa. Algunos brazos son capaces de seleccionar partes específicas, tomarlas y moverlas hacia otra parte.

Ensamblado. En muchas aplicaciones de manufacturación, los robots ejecutan el ensamblado de algunas máquinas o aparatos. Los componentes individuales son tomados por el brazo, y unidos en una secuencia apropiada para armar una máquina. Una aplicación común es la colocación de circuitos sobre sus respectivas tarjetas impresas. Robots especiales toman los circuitos integrados, transistores, resistores, capacitores, y otros componentes, para después colocarlos correctamente sobre tarjetas impresas. Las tarjetas son entonces transferidas automáticamente a una máquina soldadora donde la operación de ensamblado final tiene lugar. Aplicaciones mas complicadas son el ensamblado de computadoras portátiles y lavaplatos.

Rociado y Pintura. Otra aplicación ampliamente usada es el rociado. Para esto se coloca al final del brazo del robot una boquilla especial conectada a una manguera y a una fuente de líquido empleada para rociar componentes durante el proceso de manufacturación. La forma mas común de rociar, por supuesto, es pintando. El brazo puede estar programado para realizar movimientos precisos y proporcionar una rociada de pintura al objeto más inusual en un periodo corto de tiempo. La industria automotriz emplea robots para pintar autos. Los robots además pueden rociar sustancias químicas, por ejemplo, para prevenir la formación de hongos o la corrosión, para efectos de lubricación mediante el rociado de aceite, etc.

En la mayoría de aplicaciones de rociado, las condiciones del ambiente son extremadamente difíciles. Los humanos deben estar físicamente protegidos para que sus cuerpos no sean rociados con la pintura u otro líquido. Por tanto, es preferible el uso de robots, pues ejecutarán el trabajo mas rápido, mejor y sin riesgos de accidentes humanos.

Otras operaciones. Existen otras aplicaciones. Por ejemplo, con la adición de un componente adecuado al final del brazo, se pueden ejecutar una increíble variedad de operaciones; por ejemplo, funciones de desatornillado, apretado y remachado. Y otros tipos de brazos robóticos son usados en operaciones de fundición. En la mayoría de los casos se colocan herramientas especiales al final del brazo que permitan ejecutar operaciones específicas.

Aplicaciones Específicas.

Los sistemas expertos proporcionar la tecnología para que los robots inteligentes sean capaces de hacer frente a ambientes no estructurados. En instituciones de investigación se realizan extensos experimentos.

Ensamblado. La computadora personal portátil (IBM), es totalmente ensamblada por robots inteligentes. Para esto se utiliza un proceso de automatización flexible, el cual hace uso de estaciones de trabajo modulares que son programadas para ejecutar una variedad de tareas. La computadora es ensamblada, probada, empacada y embarcada por robots. Los robots usan potentes sistemas de sensores para ejecutar tareas o detectar errores. Una vez detectados los errores o malfuncionamientos, la información se transmite a un SE, la cual diagnostica la situación y reporta el resultado a un supervisor humano (solo hay una persona en toda la planta). La planta ensambla alrededor de 100,000 computadoras al año y aun se busca incrementar la producción y reducir los costos.

Soldador Experto. El sistema Adaptiweld fue uno de los primeros robots soldadores inteligentes disponibles comercialmente. El robot contiene el conocimiento de expertos soldadores, lo cual, le permite calcular la cantidad de líquido a emplear y el punto adecuado para realizar el soldado, esto gracias a un sistema de visión de 3-D. Gracias a estos componentes, planea la tarea de

soldado a ser ejecutada sin necesidad de supervisión humana. Las reglas de la base de conocimientos permiten al sistema tomar la mejor opción para las condiciones específicas de la soldadura.

Sistema de manipulación automática de materiales. DEC tiene implementado un sistema de manipulación de materiales en dos de sus plantas manufactureras. Estos sistemas controlan el inventario y la generación oportuna de reportes precisos sobre el progreso y calidad del trabajo. Los elementos principales de cada sistema son un par de robots que transportan elementos ensamblados y dos SE que determinan cuando y donde se despacharán los elementos.

Los SE, llamados *dispatcher* y *Mover*, son los controladores para el sistema de manipulación de materiales. *Dispatcher* determina el orden en el cual los elementos son despachados y que estaciones de trabajo los enviaran. *Mover* coordina y maneja los elementos mediante una cinta transportadora.

Dispatcher usa la información de su base de conocimientos para seleccionar el mejor elemento a despachar, dependiendo de la demanda y del estado del trabajo.

8. REDES NEURONALES

8.1. INTRODUCCION

Por más de cuatro décadas, las investigaciones en IA se han enfocado a la computarización del razonamiento humano. Orientándose la mayoría de herramientas al procesamiento secuencial. Un enfoque diferente comprende la construcción de computadoras con arquitecturas y capacidades de procesamiento que igualen algunas de las capacidades del cerebro humano (Realizar varías funciones a la vez).

Por ejemplo, consideremos el proceso que envuelve el hecho de entender una oración hablada. Para comprender el mensaje enviado, el cerebro humano, debe tomar en cuenta :

El significado de las palabras y frases individuales,
El contexto dentro del cual la sentencia fue expresada y,
El matiz del tono.

Lo cual implica resolver numerosos factores simultáneamente antes de que el cerebro pueda procesar la expresión. La forma en que se ejecutan estas tareas hace difícil el pensar que se pudiera implementar mediante sistemas secuenciales. Las Redes Neuronales Artificiales (RNA, también conocidas como modelos de Procesamiento Distribuido en Paralelo PDP), intentan dar una solución adecuada a estos problemas. Ya que son una tecnología de procesamiento de información basada en estudios sobre el cerebro y el sistema nervioso.

El interés por las RNA surge a fines de los 80´s. La motivación fue la necesidad de procesar información de la manera que lo hacía el cerebro

humano, avanzar en tecnología computacional y progresar dentro de la neurociencia hacia un mejor entendimiento de los mecanismos del cerebro.

Declarada la década del conocimiento por el gobierno de los E. U., los 90's parecen extremadamente prometedores para comprender el funcionamiento del cerebro y el pensamiento humano. La computación neuronal debe desempeñar un importante papel en esta área de investigación.

8.2. CONCEPTOS, ELEMENTOS Y CARACTERISTICAS

CONCEPTOS

Una red neuronal artificial (en adelante RNA) es un modelo que intenta emular a una red neuronal biológica. Esta formada por nodos, los cuales, actúan de forma similar a las neuronas del cerebro humano. La neurona artificial es análoga a la neurona biológica: recibe señales (estímulos), las cuales, emulan a los impulsos electroquímicos que las dendritas de las neuronas biológicas reciben de otras neuronas. La salida de la neurona artificial corresponde a las salida de la neurona biológica sobre su axón. Estas señales artificiales pueden producir cambios tal como ocurre en el momento de la sinápsis.

Una definición mas formal sería :

Una RNA consta de elementos procesadores (EP), conectados entre si, cada EP reacciona a estímulos que proceden del exterior de la red o de otros nodos, los procesa -dando distinto peso a cada uno- y genera una única señal de salida que será transmitida hacia otros nodos.

Un EP debe tener la capacidad de almacenar señales y procesarlas de acuerdo a pesos o funciones. Algunos ejemplos son : circuitos electrónicos, computadoras o partes de un programa.

ELEMENTOS

Estructuralmente una RNA consta de :

Elementos procesadores (nodos),

Interconexión.

Los elementos procesadores son dispositivos que reciben múltiples entradas, las procesan y generan una salida, la cual, es enviada a través de la interconexión, como entrada para otros elementos procesadores. Observe la figura 8.1.

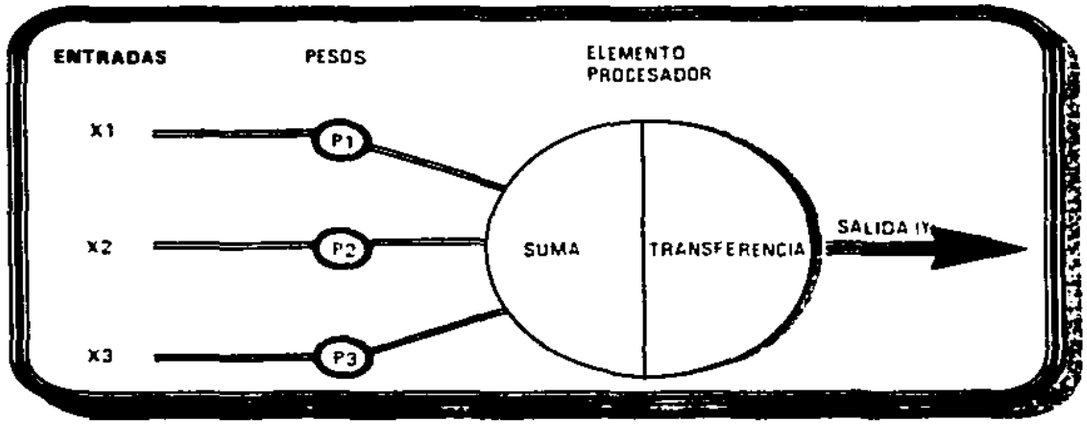


FIGURA 8.1. COMPONENTES DE UN NODO
(ELEMENTO PROCESADOR)

La interconexión, es la forma en que se enlazan o conectan los elementos procesadores, esta determinará tanto la estructura de la RNA como las reglas para disparar un proceso (señal positiva o negativa).

Los elementos procesadores (neuronas artificiales) pueden agruparse en varias capas: de entrada, intermedia (llamada capa oculta), y de salida. Observe la figura 8.2.

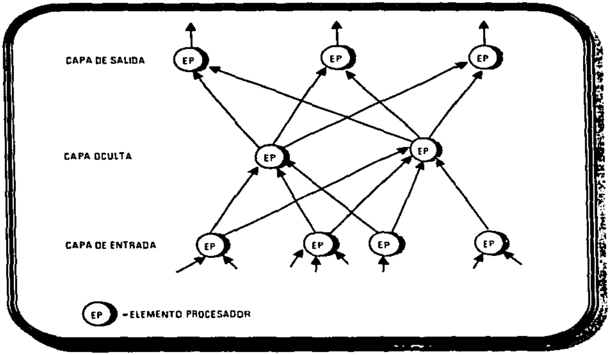


FIGURA 8.2. RNA CON TRES CAPAS

Componentes de un Elemento Procesador

Entradas

Cada entrada corresponde a un atributo. Por ejemplo, si el problema consiste en aprobar o desaprobar un préstamo, los atributos pueden ser : el nivel de ingresos, la edad, o la propiedad de una casa. El valor de un atributo es la entrada a la red. Esta entrada puede se interna (provenir de un EP anterior) o externa (en el caso de que el EP pertenezca a la capa de entrada).

Pesos

El peso es un número real que representa la importancia de una conexión de entrada a un EP. Los pesos son elementos clave, ya que es a través de ajustes repetitivos de estos que la red "aprende". Cada entrada de un EP tendrá un peso asociado.

Función de Estado

La más común es la función sumatoria, la cual, permite calcular el nivel de activación de la neurona. Basada en este nivel, la neurona puede o no producir una salida (Alto ó bajo). La función sumatoría asocia cada peso con su respectiva entrada (multiplica cada valor de entrada (X_i) por su peso (P_i)), y suma todos los elementos asociados para proporcionar el nivel de activación, Y. La fórmula para *n* entradas a un elemento procesador es:

$$Y = \sum_{i}^{n} P_i X_i$$

Para varias neuronas procesadoras (j) la fórmula es:

$$Y_j = \sum_j P_{ij} X_i$$

La salida de la función de estado es la entrada para la función de transferencia.

Función de Transferencia

Es una función matemática no lineal, usada para convertir datos a una escala específica. Existen dos tipos básicos de funciones: continuas y discretas. Las funciones continuas mas empleadas son: Rampa, Sigmoid, Arco Tangente y tangente hiperbólica. Entre las discretas tenemos a la de escalón y la de umbral.

La selección de una función específica determinará la operación de la red. Como ejemplo tenemos la función Sigmoid (o función de activación lógica)

$$Y_T = \frac{1}{1 + e^{-Y}}$$

Donde Yt es el valor transformado (o normalizado) de Y.

El objetivo de esta transformación es modificar los niveles de salida a un nivel razonable (por ejemplo entre 0 y 1). Esta transformación se realiza antes de que la salida logre el siguiente nivel. Sin tal transformación, el valor de salida podría ser muy grande, especialmente cuando se tiene una estructura de varias capas. Algunas veces se usa un valor de inicio en lugar de la función de transferencia. Por ejemplo, cualquier valor de 0.5 o menor será cargado a cero; cualquier valor arriba de 0.5 es cargado a uno. Observe la figura 8.3.

La transformación puede realizarse a la salida de cada EP, o puede ser ejecutada sobre la salida total de la red.

Salidas

La salida de la red es la solución al problema. Por ejemplo en el caso del préstamo la respuesta puede ser *si* o *no*. La RNA asigna valores numéricos, por ejemplo, +1 para si y 0 para no. El propósito de la red es calcular el valor de la salida.

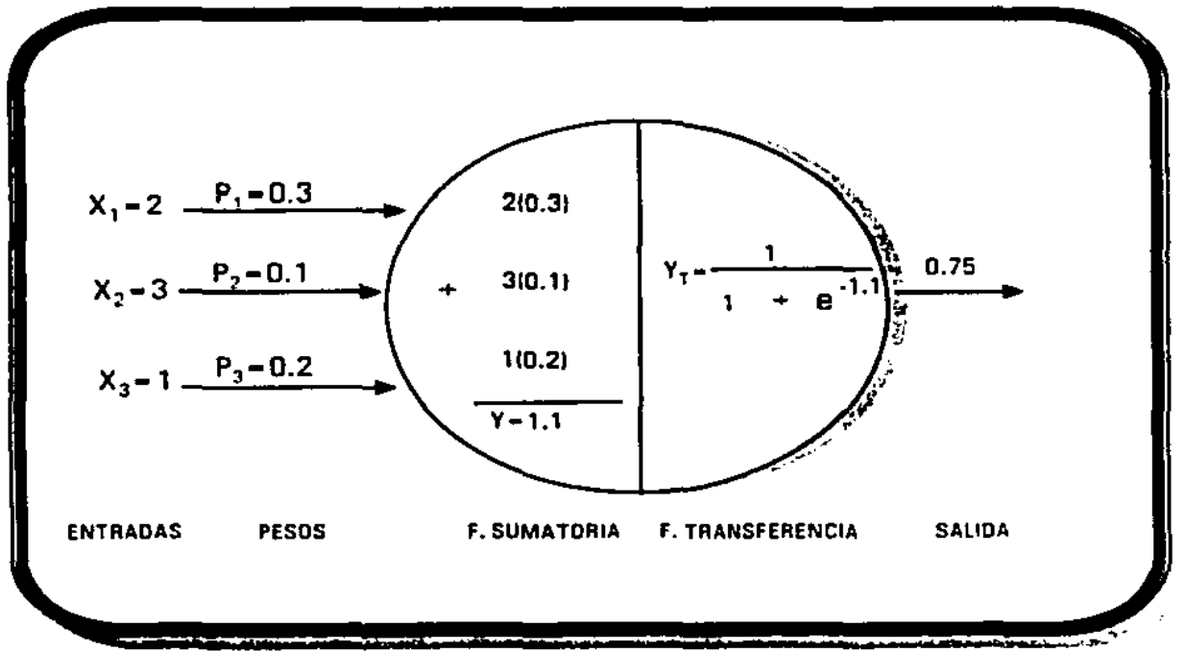


FIG. 8.3. ELEMENTOS PARA PROCESAMIENTO DE INFORMACION

CARACTERÍSTICAS.

Capacidad de Aprendizaje.

Una RNA tiene la capacidad de aprender, ya que, modifica su comportamiento en respuesta a su medio ambiente. Al alimentarla con un conjunto de entrada, realiza sus propios ajustes (en los pesos) a fin de lograr o salidas deseadas. La red es alimentada repetidamente con un conjunto de ejemplos (conjunto de instrucción) y un conjunto de salidas conocidas (objetivos) con el fin de ser instruida.

El proceso usual de aprendizaje o instrucción comprende tres pasos:

1) Cálculo de salidas.
2) Comparación de salidas obtenidas con las salidas deseadas.
3) Ajuste de los pesos y repetición del proceso.

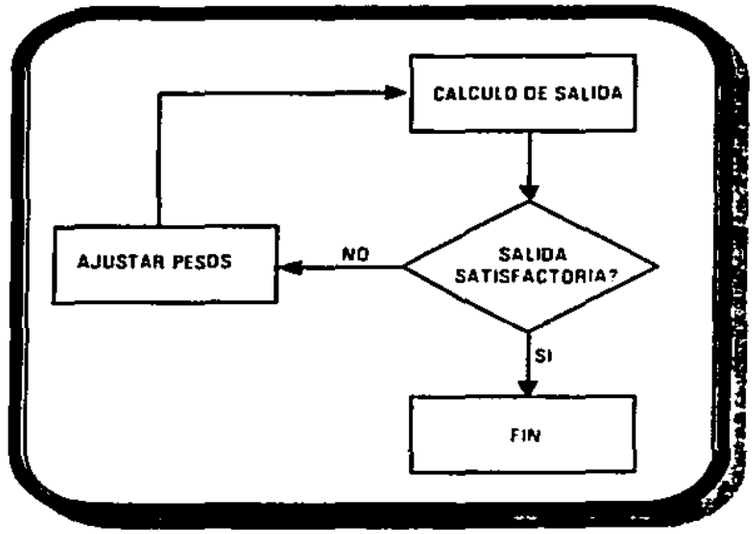


FIG. 8.4. PROCESO DE APRENDIZAJE DE UNA RNA

Generalización.

Una red debe actuar de manera insensible a las variaciones en sus entradas. Debe ser capaz de ver a través de las distorsiones del modelo, ya que esto es vital para el proceso de reconocimiento. Recuerde que el procesamiento de información con una RNA se basa en actividades de reconocimiento de patrones (asociación de un patrón de entrada con una salida correcta)

Abstracción.

Una característica común de los sistemas es la habilidad para clasificar flujos de datos de entrada sin el conocimiento explícito de reglas y usando patrones arbitrarios de peso para representar las categorías de la memoria (fig. 8.2).

La red debe ser capaz de abstraer la esencia de un grupo de entradas; por ejemplo, una red puede ser alimentada con secuencias de versiones distorsionadas de letras "A", después del proceso de instrucción y obteniendo un conjunto estable de pesos, la salida será una forma perfecta de letra "A".

El objetivo de la RNA es, establecer patrones que asemejen a prototipos idealizados que se le indican, de una manera tal que imiten en cierta forma al modelo de aprendizaje de un ser humano.

Aplicabilidad.

Una red neuronal puede tener gran aplicación en tareas de reconocimiento de patrones, es decir, las computadoras que funcionan con redes neuronales no se programan, se entrenan, y este entrenamiento será más eficiente a medida que tenga más experiencia y dará respuestas adecuadas ante un número mayor de escenarios diferentes, lo cual una computadora convencional no puede realizar.

8.3. PROCESO DE DESARROLLO

El proceso de desarrollo de aplicaciones de RNA es similar a las estructuras metodológicas de diseño tradicionales para sistemas de información por computadora. Exceptuando algunos pasos o consideraciones adicionales. En el proceso que describiremos, se asume que los pasos preliminares del desarrollo del sistema, tales como la determinación de requerimientos de información y la conducción del análisis de factibilidad del proyecto, han sido completamente terminados. Estos pasos son genéricos para cualquier sistema de información.

Como se muestra en la figura 8.5, el proceso de desarrollo para una aplicación de RNA comprende nueve pasos.

1.2 Colección y Preparación De Datos

Los primeros dos pasos en el proceso de desarrollo de una RNA comprenden la recolección de datos y su clasificación en dos conjuntos, uno de instrucción (para aprendizaje de la red) y otro de prueba. Los casos de aprendizaje se usan para ajustar los pesos, los casos de prueba son usados para la validación de la red.

Con la ayuda de un experto en el área se debe hacer un cuidadoso análisis de la aplicación, a fin de establecer delimitaciones y alcances del mismo. El desarrollador debe identificar y clasificar los datos del problema, para formularlo de manera que sea posible su solución mediante una RNA. Por ejemplo, las descripciones textuales deberán ser reformuladas de tal forma que el conocimiento pueda ser descrito numéricamente. También es importante considerar la estabilidad del conjunto de entrada y sus posibles variaciones, ya que podrían presentarse condiciones que requirieran cambios en el número de entradas a la red.

En este punto, la causa para cancelar el proyecto sería la imposibilidad o extrema dificultad de expresar la información de la forma en que la red la requiera.

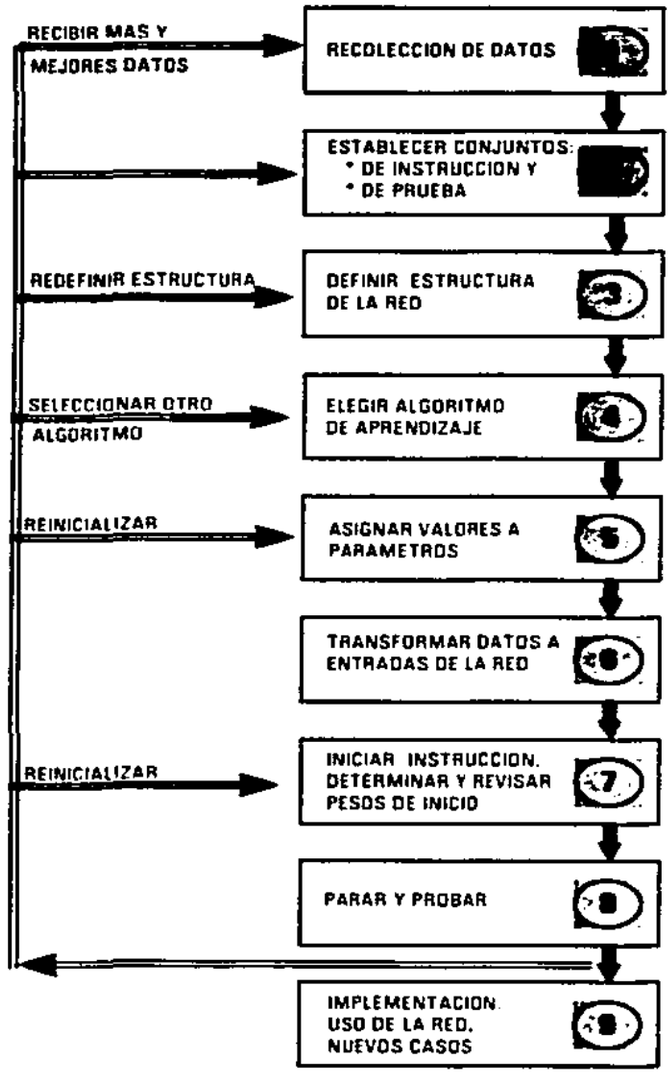


FIGURA 8.5 PROCESO DE DESARROLLO DE UNA RNA

La colección de datos de alta calidad requerirá de una cuidadosa disminución de: ambigüedad, errores y aleatoriedad en la información. Los datos deben abarcar todo el dominio del problema (operación, excepciones, posibles limitaciones, etc.).

Las ambigüedades se solucionarán a través del uso de múltiples fuentes de entrada, lo cual nos proporcionará una mayor confiabilidad. Porque aunque los conjuntos mas grandes de datos incrementan los tiempos de proceso durante el aprendizaje, a cambio de esto obtendremos un aprendizaje mas exacto que nos guiará hacia una convergencia más rápida para obtener un buen conjunto de pesos.

3 Definir Estructura de la Red

El paso 3 consiste en definir la estructura de la RNA, es decir, establecer el número de neuronas(EP 's) y capas a usar. Recuerde las diferentes capas en que se pueden agrupar los EP's (fig. 8.2.), son:

Capa de entrada. Patrón de clasificación o comparación
Capa Oculta. Representa las conexiones internas de la RNA
Capa de Salida. Resultado de la clasificación o solución

En la mayoría de modelos para RNA, todos los EP's en la capa de entrada se conectan a todos los EP's de la capa oculta y estos a su vez se enlazan a los EP's de la capa de salida (fig. 8.2).

Una RNA puede tener mas de una capa oculta, pero el esquema de conexión es similar, con los EP's de la segunda capa oculta conectados a los EP's de la primer capa oculta y a los EP's de la capa de salida.

Se han desarrollado e implementado muchos arquitecturas de redes, algunos modelos son los de : propagación hacia atrás, Hopfield, Perceptron, Boltzman, Memoria asociativa bidireccional, etc. Existen tres arquitecturas representativas :

1. Sistemas De Memoria Asociativa

La memoria asociativa se refiere a la habilidad de recordar situaciones completas a partir de una situación parcial. Estos sistemas correlacionan datos de entrada con información almacenada en la memoria. La información puede ser recordada aunque se tengan entradas incompletas o ruidosas (no muy bien definidas), y la degradación del rendimiento solo se refleja en la lentitud del proceso. Los sistemas de memoria asociativa pueden detectar similitudes entre entradas nuevas y patrones almacenados. La mayoría de arquitecturas de RNA pueden ser usadas como memorias asociativas.

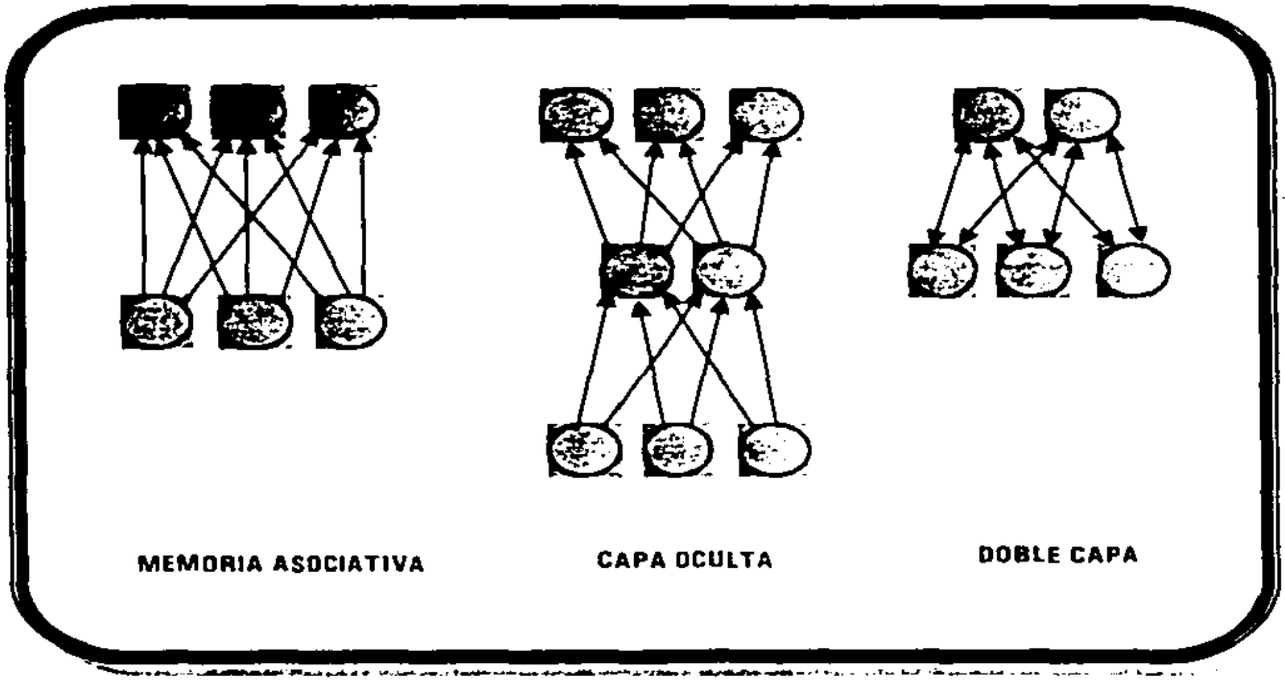


FIGURA 8.6 ESTRUCTURAS REPRESENTATIVAS

2. Capa Oculta

Los sistemas de memoria asociativa pueden tener una o mas capas intermedias (ocultas), las cuales son conexiones internas que tienen la capacidad para aprender a través del cambio de sus pesos e identificar las categorías de los datos de entrada.

3. Doble Capa

La estructura de doble capa, emplea la alimentación hacía adelante y hacía atrás para el ajuste parámetros y análisis de datos que alimentaran al sistema. Los parámetros pueden ser ajustados para estabilizar la sensitividad del sistema y producir categorías significativas (provechosas).

En la mayoría de sistemas comerciales de RNA para PC, los EP's de una capa se conectan únicamente a los EP's de una capa adjunta. Los EP's pertenecientes a una capa no se conectan entre sí.

Aunque pueden existir combinaciones de arquitecturas según las necesidades de procesamiento de la aplicación.

4 Elegir Algoritmo de aprendizaje

Mediante el algoritmo de aprendizaje la RNA calcula el grado de error. Este debe seleccionarse antes de iniciar la instrucción, para lo cual, deben tomarse en cuenta las herramientas de software que van a ser usadas. La decisión debe hacerse en esta etapa porque la estructura elegida y los datos pueden requerir de un ajuste para poder emplear el algoritmo seleccionado.

Lippman distingue entre dos categorías de algoritmos basándose en el formato de la entrada: entrada de valor binario (0 y 1) o entrada de valor continuo. Cada uno de estos puede además ser dividido en dos categorías : aprendizaje supervisado y aprendizaje no supervisado.

Aprendizaje supervisado. Consiste en proporcionar un objetivo a la red, a fin de que este controle o supervise el aprendizaje. Usa un conjunto de entradas para las cuales se conoce la salida deseada. La diferencia entre la salida deseada y la actual permite calcular las correcciones a los pesos de la RNA. Este tipo es empleado por la propagación hacia atrás (backtracking) y la red de Hopfield.

Aprendizaje no supervisado, Dado un conjunto de datos de entrada, a la RNA se le permite jugar con ellos para intentar descubrir regularidades y relaciones entre las diferentes partes de las entradas. No se proporciona un objetivo, la red se autoregula. El usuario debe examinar las salidas para asignar significado y determinar la utilidad de los resultados. Un ejemplo de este tipo de aprendizaje es La Teoría de Resonancia Adaptativa

El algoritmo más usado es el de propagación hacia atrás (backtracking), el cual se realiza en dos fases:

1. Primero las entradas son enviadas hacia adelante a través de la red para producir una salida,
2. Después la diferencia entre la salida deseada y la actual produce una señal de error que es enviada hacia atrás a través de la red para modificar los pesos de sus entradas.

5 Asignación de valores a parámetros

En este paso se realiza la inicialización de parámetros y pesos de la red, cuyos valores se modificaran como respuesta a la retroalimentación. Las RNA deben disponer de varios parámetros para llevar a la red hacia el nivel de rendimiento deseado. La asignación inicial será importante para determinar la eficiencia y grado del aprendizaje. Se pueden designar parámetros para: determinar el grado de aprendizaje (alto o bajo), validación y prueba durante la preparación, etc.

6 Transformación de datos

Este paso consiste en el proceso de transformación de los datos (de prueba e instrucción) al tipo y formato requeridos por la red y su algoritmo. Tal proceso puede implicar desarrollo adicional de software. Los diferentes procesos (almacenamiento, manipulación, etc.) deben ser diseñados para que la red pueda reaprender cuando así lo requiera. La forma en que se representen y ordenen los datos determinará la eficacia de los resultados entregados por la red.

7.8 Instrucción (entrenamiento de la Red) y prueba

Esta fase consiste en la presentación del conjunto de datos de instrucción a la RNA para que esta modifique sus pesos en respuesta al flujo de datos presentado. Esta etapa se basa en un proceso interactivo de presentación de entradas y salidas deseadas, durante este proceso, La RNA calcula la salida y realiza un ajuste de pesos hasta obtener la salida deseada. El artificio consiste en cambiar los pesos en la dirección correcta, esto es, hacer cambios que fortalezcan la minimización de errores. Después de varías iteraciones se debe lograr un conjunto consistente de pesos, que trabaje con todos los datos de instrucción que se deriven.

El tiempo de instrucción estará determinado por la selecciones tanto de la estructura como de las condiciones iniciales.

9 Implementación

En este paso, se presenta el conjunto estable de pesos obtenido. Ahora la red puede reproducir las salidas deseadas proporcionando entradas similares a las que se encuentran en el conjunto de aprendizaje. La red esta lista para usarse como un sistema único o como parte de otro sistema.

8.4. REDES NEURONALES Y SISTEMAS EXPERTOS

En algunos casos las RNA pueden ejecutar tareas mejor que un SE, Pero en otros casos las dos tecnologías no entran en competencia. Las características de las tecnologías son diferentes y en algunos casos pueden complementarse.

Un SE representa un logro simbólico lógico, mientras que una RNA emplea procesamientos numéricos asociativos para emular modelos de sistemas biológicos, Observe en la figura 8.7., el contraste en algunas características.

CARACTERISTICAS	SISTEMAS EXPERTOS	REDES NEURONALES
MANEJO DE INFORMACION	MAYORMENTE SIMBOLICA	NUMERICA
RAZONAMIENTO	LOGICO	ASOCIATIVO
EXPLICACION	DISPONIBLE	NO DISPONIBLE
PROCESAMIENTO	SECUENCIAL	PARALELO
VALIDACION Y VERIFICACION	DIFICIL, LENTA	RAPIDA
DIRIGIDO POR	CONOCIMIENTO	DATOS
MANTENIMIENTO	DIFICIL	FACIL

FIGURA 8.7. CARACTERISTICAS RNA-SE

Los SE ejecutan razonamientos basándose en reglas y hechos preestablecidos para una área de conocimiento bien delimitada; combinando bases de conocimientos (reglas y hechos) con información sobre casos específicos de la aplicación correspondiente. El razonamiento puede ser explicado y las bases de conocimiento pueden modificarse fácilmente con solo agregar reglas, independientemente del motor de inferencia.

Los SE son especialmente adecuados para aquellas aplicaciones que cuentan con entradas precisas y conducen a salidas lógicas. Además de tener la capacidad de interactuar con el usuario.

Computación Neuronal

Una aplicación se desarrolla estableciendo un conjunto apropiado de datos para instrucción que permiten que el sistema aprenda y generalice para operar con datos futuros. Las entradas que asemejan exactamente a los datos de instrucción son reconocidas e identificadas, mientras que los datos nuevos (incompletos o versiones distorsionadas) pueden ser comparados y asemejados con patrones reconocidos por el sistema.

Comparación De Características

Para aplicaciones estables con reglas bien definidas, los SE proporcionan un buen rendimiento. Además de permitir la explicación de acciones, lo que ayuda al usuario a entender las conclusiones y el proceso de razonamiento. En contraste en una RNA, el conocimiento se representa como pesos numéricos; las reglas y el proceso de razonamiento no se explican al usuario.

Las RNA son preferibles al SE cuando no se dispone de reglas ya sea porque el problema es muy complejo o no se dispone de experiencia humana. En caso de poder generar los datos de instrucción, el sistema puede ser capaz de aprender para funcionar tan bien o mejor que un SE. Este logro además tiene el beneficio de facilidad de mantenimiento. Las modificaciones se realizan mediante la reinstrucción con datos actualizados. Otra ventaja de las RNA es la rapidez de operación después de que la red ha sido instruida.

Comparación Del Proceso De Desarrollo

Durante el proceso de desarrollo de un SE y una RNA existen importantes paralelismos, Observe la figura 8.8. Los ingenieros de conocimiento y neurocomputación tienen la función de representar el conocimiento, ya sea, como hechos y reglas (SE) o como conjuntos de datos (RNA).

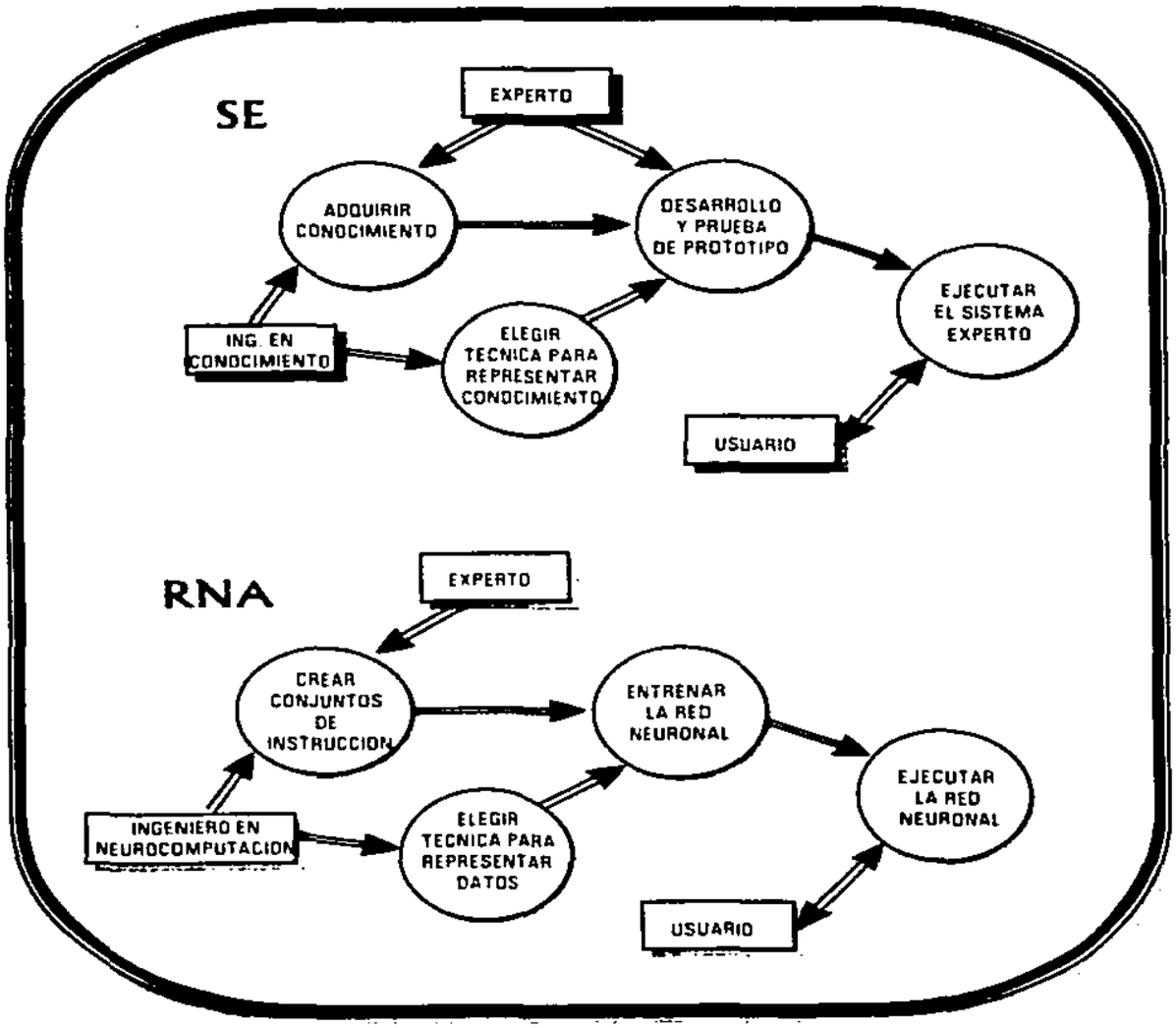


FIGURA 8.8. PROCESOS DE DESARROLLO (SE, RNA)

En ambos casos, se requiere un experto en el área para asegurar la exactitud de las reglas o conjuntos de datos. Sin embargo, los requerimientos de las RNAs son significativamente menores.

Los desarrolladores emplean tiempo y esfuerzos considerables para formular y probar reglas, o crear conjuntos de datos precisos con el formato apropiado para presentarlos a la RNA para su entrenamiento. En cada caso, la validación y verificación del sistema es una tarea importante y difícil.

El paralelismo en el desarrollo de los SE y las RNA y su naturaleza complementaria hacen que los **sistemas híbridos** sean una área prometedora para la investigación y desarrollo. La integración de estos y posiblemente otros componentes inteligentes con sistemas convencionales promete ser una importante área para la investigación, desarrollo e implementación de futuros sistemas.

Las RNA En La Adquisición Del Conocimiento.

La principal limitación de un SE es la adquisición del conocimiento. Pero las RNA pueden emplearse para adquirir conocimiento que más tarde pueda ser usado en los SE. Una RNA puede realizar una adquisición del conocimiento para aquellas situaciones en las que se tengan disponibles datos históricos.

En situaciones donde las reglas no pueden ser determinadas directamente, o donde puede tomar largo tiempo el adquirirlas, una RNA puede ser de utilidad para una rápida identificación de conocimiento implícito mediante el análisis automático de información histórica. La RNA analiza los conjuntos de datos para identificar patrones y relaciones que puedan subsecuentemente proporcionar reglas útiles para los SE. La RNA puede ser la técnica exclusiva para adquisición del conocimiento, o puede

usarse como técnica complementaria.

Otra contribución de las RNA a la adquisición del conocimiento puede presentarse cuando la interface con un experto sea mas eficiente con un módulo de SE que realice preguntas y direccione la recolección de datos de manera eficiente y comprensiva. La instrucción de una RNA puede entonces procesar información rápidamente para producir hechos y consecuencias asociados. Después, el módulo del SE puede ejecutar un análisis más completo y reportar resultados. De esta manera, no son necesarias tantas reglas explícitas ya que la RNA contiene el conocimiento general incrustado en los pesos de sus conexiones y produce conocimiento específico, que será relevante para el usuario de la aplicación.

Integración De Las RNA Y Los SE

La naturaleza complementaria de las RNA y los SE permite novedosas aplicaciones y soluciones para problemas complejos cuando se combinan las dos tecnologías. Existen dos formas de usar SE y RNA : sistemas incrustados y sistemas distribuidos.

Sistemas incrustados

En la configuración del sistema incrustado, los componentes del SE y la RNA están integrados o estrechamente acoplados sobre el mismo sistema. Por ejemplo el componente RNA podría representar la base de conocimientos como conexiones (pesos). El sistema podría estar diseñado para que los pesos representen ramas en la base de reglas lógicas para que las líneas de razonamiento puedan ser explicadas. Alternativamente, la RNA de memoria asociativa puede almacenar relaciones entre los patrones de entrada y sus correspondientes conclusiones.

Sistemas Inteligentes Distribuidos

Otra opción es el permitir que los SE y las RNA, aunque pertenezcan al mismo sistema, funcionen independientemente, y se enlacen a través de líneas de comunicación para transferencia de datos. Los componentes de las RNA pueden trabajar estrechamente con otro software que proporcione funciones especiales o subrutinas para todo el sistema.

Un tipo de sistema distribuido ofrece pre y post-procesamiento con software standar o SE para conectarse con componentes de RNA. Esta configuración puede emplear un componente de SE para recolectar datos del usuario y además para requerimientos de razonamiento al presentar las conclusiones finales. El componente de la RNA analizará datos y proporcionar la información requerida por el SE. Una ventaja de esta configuración es la posibilidad de usar archivos o datos de entrenamiento para cambiar el comportamiento del sistema sin necesidad de tener conocimiento, o cambiar las reglas en la base de conocimientos. Otra variación interesante sería la inclusión de una base de casos que serviría para proporcionar la instrucción de entrada a la RNA, y que podría ser actualizada fácilmente cuando existiera nuevo conocimiento, además de usarse periódicamente para entrenar a la red.

Ejemplos :

Un prototipo de sistema integrado emplea el shell de SE AUBREY y la herramienta para RNA Neuroshell para alertar a los usuarios sobre los requerimientos de recursos para desarrollo de sistemas de bases de datos. La RNA analiza datos experimentales sobre la cantidad de tiempo y esfuerzo requeridos por proyectos anteriores de bases de datos. El sistema proporciona la flexibilidad de emplear nuevos archivos para suministrar información sin tener que agregar nuevas reglas o información extraída de análisis de datos externos.

Un sistema desarrollado por Hanson y Brekke realiza requerimientos de recursos para mantenimiento de redes o estaciones de trabajo para la NASA. El sistema basado en reglas determina los recursos finales, pero una RNA proporciona tiempos completos para requerimientos de servicios. Los resultados se basan en información sobre casos históricos y listas de requerimientos actuales de servicios. La RNA es fácilmente reentrenada a través de nuevos conjuntos de datos.

8.5. REDES NEURONALES Y OTRAS TECNOLOGIAS

Procesamiento De Lenguaje Natural

Las redes RNA tienen la capacidad para soportar el reconocimiento de la voz basándose en el almacenamiento de información de entrenamiento sobre estructuras del lenguaje y comparación de estas con patrones de entrada. Un *front-end* de RNA reconoce fragmentos cortos del lenguaje, y otras RNA construyen palabras combinando tales fragmentos. Finalmente, otro componente clarifica las posibles ambigüedades entre las palabras. Por ejemplo un sistema desarrollado por Kohonen tiene la mas alta precisión y el vocabulario más extenso disponible actualmente.

Un sistema llamado NETtalk, es una red de tres capas que sintetiza lenguaje de un texto. El sistema no tiene programadas reglas y puede pronunciar un alto rango de texto en inglés. Los pasos de la fase de entrenamiento imitan las fases del desarrollo de niños que están aprendiendo a hablar. Un requerimiento importante para los sistemas de procesamiento de lenguaje natural es la discriminación de significados de palabras dependiendo del ambiente textual. Las RNA pueden ser usadas para establecer asociaciones entre las palabras que comúnmente se usan juntas. Durante la operación, la RNA checa estas conexiones de las palabras en la oración que será analizada y elige la mejor interpretación.

Procesamiento de Señales, Visión Computarizada y Robótica.

Las máquinas autónomas han sido el objetivo final de la investigación y desarrollo de la IA. Esto comprende: procesamiento e interpretación de sensores, coordinación entre la percepción visual y acciones mecánicas, sentividad para percepción de ambientes, y la habilidad de aprender y adaptarse a los cambios de un entorno determinado. En todas estas tareas la computación neuronal puede jugar un papel importante.

En algunos trabajos ya existen sistemas capaces de interpretar datos aprendidos para efectuar movimientos mas uniformes y poder variar sus velocidades según las necesidades de situaciones específicas.

Un ejemplo de la integración de RNAs en sistemas robóticos se da en las máquinas de aprendizaje autónomo (Handelman). Estas aprovechan las características complementarias de los SE y las RNAs para construir un subsistema que responda a sus necesidades. El sistema combina estrategias de conocimiento reflexivo y RNAs en un módulo declarativo cuya función es modificar al sistema cuando el brazo del robot empieza a desviarse del objetivo (movimiento) deseado. Un módulo basado en reglas monitorea otras tareas para invocar reglas o componentes de computación neuronal, dependiendo de que las operaciones sean apropiadas o se deba realizar un ajuste. Inicialmente, el módulo basado en reglas supervisa el aprendizaje de la RNA, que proporciona los datos para el movimiento del brazo. Subsecuentemente, en modo de operación, la RNA asocia nuevas entradas con salidas relevantes lo cual permite una mayor detección de posibles cambios en el movimiento.

En el componente para visión u otros sistemas del robot, las RNAs usan la característica de memoria asociativa para interpretar información visual digitalizada, tal como grados de iluminación u objetos y elegir una comparación exacta con una imagen en memoria. El sistema se alimenta con información visual de entrada y extrae sus características, para esto se emplean subsistemas, con características especiales para extracción de información. La disponibilidad de circuitos integrados para procesamiento en paralelo permite la fácil integración de RNAs con el hardware del robot.

8.6. AREAS DE APLICACION PARA COMPUTACION NEURONAL

1. **Reconocimiento de patrones**, generación de lenguaje hablado - sistemas que aprenden fonemas en orden para pronunciar texto en inglés.

   Transmisión de datos - rápida comparación de patrones de datos usados en técnicas de comprensión para transmisión de voz, imagen y texto.

   Detección de movimiento para aplicaciones militares - identificación de aviones, análisis de terreno, etc.

   Aprendizaje de robots - coordinación mano-ojo mediante instrucción para asir objetos; posible uso con ensamblado de sistemas en estaciones espaciales.

   Automatización de operaciones en ambientes difíciles - observatorios terrestres, plantas de potencia, vehículos submarinos.

   Reconocimiento de caracteres - mecanografía y taquigrafía, aún si hay distorsión, verificación de firmas en cheques, etc.

   Sistemas para reconocimiento de voz – control de dispositivos activados por voz, dictado.

   Diagnóstico de equipo defectuoso – análisis de casos basados en instrucción, monitorización de datos para identificar fallas en circuitos eléctricos.

2. **Interpretación de datos** donde se requieren herramientas analíticas para hacer generalizaciones o realizar conclusiones provenientes de grandes cantidades de datos de diferentes fuentes o de sensores :

   Servicios financieros – identificación de patrones en el mercado de valores y asistencia en estrategias para acciones mercantiles.

   Evaluación de prestamos – análisis de méritos para otorgar prestamos, basado en informaciones anteriores.

   Lanzamiento y diagnóstico de motor de cuetes –instruir redes neuronales con sensores.

   Diagnóstico médico – instruir redes neuronales con casos de pacientes anteriores.

   Información de tarjetas de crédito – rápida detección de fraudes por patrones de compra.

   Pronósticos en líneas aéreas – predicción de demanda de asientos después de instrucción con información histórica ; modificación rápida mediante una reinstrucción con datos actualizados.

   Evaluación de personal y candidatos para empleo- comparación de datos personales con requerimientos de trabajo y criterios de rendimiento, Permite flexibilidad y tolera de información incompleta.

3. **Optimización** – Técnicas tales como la máquina de Botzman y simulaciones que encuentran soluciones aceptables a problemas que envuelven muchos parámetros.

CONCLUSIONES

- La Inteligencia Artificial, es un área de investigación, que proporciona bastantes herramientas y tecnologías tendientes a la simulación de las forma de pensar y razonar del ser humano.

- Durante la década de los 90's la Inteligencia Artificial ha tenido un gran resurgimiento y varias de sus aplicaciones se han incorporado plenamente a la vida cotidiana, en áreas tales como: medicina, administración, finanzas, educación, aplicaciones militares, etc.

- Tecnologías proporcionadas por la investigación en IA, tales como: Sistemas Expertos y Redes Neuronales pueden combinarse entre sí y con sistemas tradicionales, para lograr sistemas totalmente automatizados con capacidades de aprendizaje y toma de decisiones.

- La IA es un gran opción para aquellas empresas mexicanas, preocupadas por la exactitud, rapidez y oportunidad de la información, de tal forma que el recurso humano, no se vea envuelto en tareas rutinarias y cotidianas, y a través de apoyos y asesoramientos automáticos realice una toma de decisiones óptima y oportuna.

APORTACIONES

- Fuente documental clara y accesible para el estudiante o futuro profesionista, tanto de disciplinas humanísticas como técnicas.

- Proporcionar aspectos tradicionales y básicos de la Inteligencia Artificial, así como una visión general del alcance y la aplicación de sus áreas de investigación.

BIBLIOGRAFIA

- Robert J. Schalkoff. **Artificial Intelligence: An Engineering Aproach.** McGraw-Hill, Singapore, 1990.

- Jean -Pasacal Aubert, Richard Schomberg. **Inteligencia Artificial.** Paraninfo, Madrid, 1986.

- José Ma. Angulo Usateguí, Anselmo del Moral Bueno. **Guía fácil de Inteligencia Artificial.** Paraninfo, Madrid, 1986.

- Efraim Turban. **Expert Systems and Applied Artificial Intelligence.** Macmillan, USA, 1992.

- Wendy B. Rauch-Hinding. **Aplicaciones de la Inteligencia Artificial en la Actividad Empresarial, la Ciencia y la Industria.** Diaz de Santos, Madrid, 1989.

- Avron Barr, Edward A. Feigenbaum. **The Handbook of Artificial Intelligence (vol. I).** Addison - Wesley, USA, 1989.

- José Negrete Martínez. **Inteligencia, Aunque sea Artificial.** Limusa, México, 1990.

- José Negrete Martínez. **Inteligencia Experimental en computadoras.** Limusa, México, 1991.

- David W. Rolston. **Principios de Inteligencia Artificial y Sistemas Expertos.** McGraw-Hill, México, 1992.

- Elaine Rich, Kevin Knight. **Inteligencia Artificial.** Mcgraw-Hill, Madrid, 1994.

- Phillip R. Robinson. **Aplique Turbo Prolog.** McGraw-Hill, Madrid, 1990.

- Wendy L. Milner. **Common Lisp a Tutorial.** Prentice-Hall, USA, 1988.

- Tarun Khanna. **Foundations of Neural Networks.** Addison Wesley, USA, 1990.

- Instituto Politécnico Nacional. **Simposium Internacional de Computación (Memorías).** México, 1993.

- Sociedad Mexicana de Inteligencia Artificial. **Reunión Nacional de I.A. (Memorías).** Limusa, México, 1991.