UNIVERSIDAD LA SALLE

ESCUELA DE INGENIERIA

INCORPORADA A LA U.N.A.M.

TEORIA, TECNICAS, METODO DE CONSTRUCCION Y APLICACIONES DE SISTEMAS EXPERTOS

TESIS PROFESIONAL

QUE PARA OBTENER EL TITULO DE

INGENIERO MECANICO ELECTRICISTA

P R E S E N T A :

JOSE MANUEL ESTEVEZ Y GABIAN

D I R E C T O R D E T E S I S:

ING. PATRICIA VASQUEZ AGUILERA

MEXICO, D.F. 1989

INTRODUCCIÓN

CAPITULO I INTELIGENCIA ARTIFICIAL

CAPITULO II DESARROLLO DE SISTEMAS DE INTELIGENCIA ARTIFICIAL

INTRODUCCION

LA CIENCIA SE COMPONE DE
ERRORES, QUE A SU VEZ SON
LOS PASOS HACIA LA VERDAD

JULIO VERNE

- Me propongo desconectar algunos de tus circuitos, especialmente los que se relacionan con tus funciones superiores -Dice el doctor Chandra.

- ¿ Tendré sueños ? -Pregunta SAL

- Claro que soñaras. Todas las criaturas inteligentes sueñan, pero nadie sabe por qué..... Tal vez sueñes con HAL, como a menudo me pasa a mí.

(Dialogo entre la computadora SAL 9000 -Hermana de HAL 9000, computadora de la nave Discovery- y el creador de ambas, en la novela 2010: ODISEA DOS, de Arthur C. Clark)

Si el radio fué una extensión del oído y la televisión de la vista, la computadora constituye una extensión de la Inteligencia. Su vertiginoso desarrollo y su influencia en innúmerables áreas de la actividad humana, transforman al mundo. Las fábricas, las casas, los edificios son convertidos en lugares inteligentes. Cambia los métodos de producción y ha hecho evolucionar las tácticas de combate. Constituye una herramienta para la investigación científica y tecnológica. Gracias a los ordenadores y a la evolución de los sistemas se ha logrado la conquista del espacio; así como su influencia se deja sentir en las más variadas actividades humanas como en el mundo del arte, en la música, en la pintura e inclusive en los espectáculos.

Hoy en día son integradas impresionantes redes computacionales tanto nacionales como extrangeras, a través de las cuales la información viaja de un extremo a otro. Máquinas inteligentes asesoran tanto a empresas como a inversionistas y permiten presentar simuladores para el avance de la ciencia. En la actualidad es asombrosa la velocidad de operación de las computadoras, así como la capacidad de almacenamiento de éstas y de sus equipos periféricos; con esto se abre paso a la sociedad de la información.

La Inteligencia Artificial, una de las partes más avanzadas de las tecnologías de la información, nació con el objeto de estudiar actividades humanas para las que no se disponía de métodos que describiesen cómo se realizaban; pretende, por tanto, ocuparse de la producción de modelos y programas del llamado comportamiento inteligente. Su

nacimiento en el campo informático en 1956, proviene de la carencia de algoritmos para la descripción de algunas actividades cognoscitivas tan simples, aparentemente, como reconocer visualmente un objeto o traducir un breve texto entre dos idiomas conocidos.

El objetivo final que los investigadores tienen es el de crear máquinas que aprendan por ellas mismas, asocien razonamientos, deduzcan realidades, tomen decisiones, hablen, reconozcan varios idiomas, procesen imágenes; en resumen sueñan en crear ordenadores que piensen como humanos.

Aproximadamente en el año 1971 los investigadores mexicanos comienzan a verse interesados por esta nueva rama de la ciencia contemporanea; es en este año cuando un investigador mexicano el Doctor Adolfo Guzman crea en el MIT un algoritmo para el reconocimiento de patrones en los objetos, este se convirtio con el tiempo en un algoritmo clasico de visión por ordenador.

Apartir de el año de 1976 hasta la fecha se puede hablar de una evolución de los Sistemas Expertos en México; tanto en el área de investigación para la creación de nuevas arquitecturas de procesamiento paralelo, así como de nuevos sistemas que sean más potentes y eficaces.

Se puede señalar que desde 1976 los Sistemas Expertos como término a adquirido una gran popularidad y su evolución en México a sido sorprendente, así mismo año con año existen más, en muy diversificadas áreas. Los trabajos desarrollados

son expuestos en congresos llevados a cabo con cierta regularidad.

En el desarrollo de Sistemas Expertos una de las partes más importantes en la creación de estos, es plantear una metodología de construcción desde el inicio de el desarrollo; para que así este pueda ser lo más eficaz posible y al mismo tiempo sirva de base para implementaciones futuras.

Dicha tesis esta integrada por siete capitulos que dan un enfoque de una teoria para la construcción de sistemas expertos, tanto sus técnicas como algunas de sus aplicaciones en un entorno muy reducido. El contenido de cada uno de los capitulos se mencionara a continuación:

En el capitulo 1 se hablara sobre la historia de la Inteligencia Artificial, concepto y ramas que componen esta nueva ciencia surgida cómo tal hace menos de 30 años.

En el capitulo 2 se mencionan las características generales de como se deve resolver un problema referente a la Inteligencia Artificial, así como las formas de representación del conocimiento y los métodos y las estructuras de control para poder resolver el problema ha resolver.

En el capitulo 3 se plantean las diferentes formas de representación del conocimiento, de una manera más detallada, y cada uno de los elementos que las componen.

En el capitulo 4 se hace referencia a una pequeña evolución historica de los Sistemas Expertos, así mismo como a su clasificación segun la actividad que desempeñan, se da un concepto de los mismos y se plantea un ciclo de construcción de los Sistemas Expertos.

En el capitulo 5 se definen las características de los lenguajes de programación más viables para poder desarrollar Sistemas Expertos e inclusive para cualquier aplicación de Inteligencia Artificial, así como las características principales de cada uno de ellos para poder ser usados en la construcción de Sistemas Expertos; se hace una pequeña referencia a futuro de los lenguajes de programación orientados en el área de discusión de esta tesis.

En el capitulo 6 se tratan brevemente los objetivos y se definen los entornos de programación, así como las herramientas que se usan para los entornos de programación inteligente.

En el capitulo 7 se hace referencia general de dos campos de aplicación, como son el área médica y en el área industrial, ya que sobre ellos se han desarrollado muy variados sistemas a un nivel mundial; así mismo se le da un trato especial a las aplicaciones desarrolladas en México, en los diferentes centros de investigación y estudio.

Como punto final a tratar se plantearan una serie de conclusiones obtenidas por el desarrollo de este trabajo.

CAPITULO I

INTELIGENCIA ARTIFICIAL

NO EXISTE EN EL MUNDO NADA MAS
PODEROSO QUE UNA IDEA A LA QUE
LE HA LLEGADO SU TIEMPO.

VICTOR HUGO

1.1 EVOLUCION HISTORICA

1.1.1 Prehistoria de la Inteligencia Artificial

Así como de alguna forma los soportes mecánicos para la automatización de cálculos aritméticos se sitúan en la prehistoria de los ordenadores, la prehistoria de la Inteligencia Artificial abarca desde los primeros tiempos de nuestra civilización hasta mediados del siglo veinte. En este periodo se producen hechos que podemos agrupar en dos líneas: Una de ellas, directamente relacionada con la construcción de autómatas que simulaban desde el punto de vista el comportamiento humano o animal, y que solían funcionar en la ayuda de su amo. La otra línea, referente a la información y a la automatización del razonamiento lógico y matemático.

En relación con los autómatas, hace notar Pamela McCorduck que siempre se ha relacionado con los aparatos mecánicos complejos. Los hombres, intuitivamente, han comparado la complejidad del funcionamiento de una máquina con su propia vida.

Acaso la primera mención de los autómatas aparezca en la Iliada, donde se lee que Vulcano fabricaba << veinte trípodes que tenían ruedas de oro para que de propio impulso pudieran entrar donde los dioses se reunían y volver a casa>> y que era ayudado en su cojera por << dos estatuas de oro semejantes a vivientes jóvenes pues tenían inteligencia, voz y fuerza >>.

Al llegar el racionalista siglo XVIII, el siglo de los autómatas por antonomasia, Descartes defendió su tesis del << animal-máquina >>: los seres vivos, salvo el hombre, son meros mecánismos. La Mettrie, en 1747, va más allá con su

escandaloso << L'homme machine >>: también el hombre y su comportamiento inteligente son explicables en términos exclusivamente mecánicos.

Ciertamente existían admirables mecanismos en sus días, por ejemplo, los de Jacques de Vaucanson: el flautista (1737) que movía realmente los dedos para producir una melodía; o el pato (1738), que era capaz de nadar, batir las alas, comer y expulsar excrementos simulados.

El español Torres Quevedo, ya en 1912, construyó también un notable autómata para jugar el final de ajedrez de rey y torre contra rey. En 1929 se presentaba en Francia el << Phildog >>, que seguía un rayo luminoso de una linterna y ladraba si la intensidad luminosa era excesiva. Y como estas podrían citarse docenas de realizaciones sorprendentes.

El escritor Capek difunde en 1920 una palabra destinada a tener un gran éxito: << robot >>. En su obra << RUR >> aparecen unos seres creados para realizar las tareas que el hombre no quiere hacer.

Sin embargo, hasta la llegada de los ordenadores electrónicos no dispusieron los científicos y técnicos de una herramienta que permitiera la ejecución de tareas más complejas por parte de dispositivos mecánicos; que hiciera posible, por así decir, la construcción de un robot.

Pero ahora hablare de la segunda de las líneas que se distinguieron de la prehistoria de la Inteligencia Artificial: la automatización del razonamiento, Leibnitz buscó un álgebra universal que permitiera deducir todas las verdades acerca del mundo.

Al llegar el siglo XIX, los matemáticos sienten por su parte la necesidad de buscar para su razonamiento más solidas bases. La aparición de las paradojas lleva al desarrollo de los sistemas formales y de la lógica matemática.

Las teorias de la computabilidad y de los autómatas proporcionaron el vinculo entre la formalización del razonamiento y las máquinas que estaban a punto de surgir tras la Segunda Guerra Mundial.

Un último elemento importante que citaré en laprehistoria de esta disciplina es la *cibernética*. Los conceptos de retroalimentación, control, sistemas autoorganizados etc., debidos a Wiener y otros, pusieron énfasis en la conducta global de sistemas << localmente sencillos >>. La cibernetica influyó en muchos campos debido a su naturaleza fundamentalmente interdisciplinar, ligando entre si la fisiología neuronal (McCullosh y Pitts), la teoria de la información de Shannon, la lógica matemática y la naciente tecnología informática. De esta forma, las ideas de los creadores de la cibernética llegaron a ser parte del espíritu del tiempo, e influyeron fuertemente en los primeros investigadores de la Inteligencia Artificial.

1.1.2 El nacimiento de la Inteligencia Artificial

Ambiciosos y optimistas se mostraban los pioneros de la Inteligencia Artificial durante los primeros años de la era informática. Pero el fracaso de la mayoría de los proyectos mostró que los problemas que intentaban resolver eran demasiado complicados, tanto teórica como tecnológicamente.

En 1950 Alan Turing presentó una comunicación sobre el

tema de la Inteligencia Artificial, titulado *Inteligencia y Funcionamiento de Máquinas*. En este trabajo propone un test <<*Test de Turing* >> para determinar cuando una máquina posee Inteligencia Artificial.

Posteriormente, en 1955, fue creado un lenguaje de procesamiento por Allen Newell, J.C. Shaw y Herbert Simon que fue considerado como el primer lenguaje de Inteligencia Artificial; era el IPL-II (*Information Procesing Language-II*).

En el año de 1956, la Conferencia de verano sobre Inteligencia Artificial, organizada por J. McCarthy, Marvin Minsky, Nathaiel Rochester y Claude Shanon, con el patrocinio de la Fundación Rockefeller, reunió a todos los que trabajaban en el recién estrenado campo de la Inteligencia Artificial. La lógica teórica fué considerada como el primer programa de la Inteligencia Artificial y usada para resolver problemas de búsqueda heuristica, junto con los principios matemáticos de Whitehead y Rusell.

Destacando también que a mediados de los años cincuenta John McCarthy, y posteriormente el MIT (Instituto Tecnológico de Massachussets), diseñaron el lenguaje LISP (List Procesing).

Por suerte o por desgracia, las cosas no resultaron tan fáciles. Ejemplo de ello es la traducción automática. Los intentos de traducir un texto de un idioma a otro por medio de una máquina se remontan al menos al memorándum que W. Weaver distribuyó en 1949. Durante el decenio de 1950 este fué el tema de moda: se celebrarón congresos y proliferaron los grupos de investigación.

Hacia 1957 Newell, Shaw y Simon comienzan a desarrollar

el Resolvente de Problemas Generales (GPS). Este programa aplica técnicas de resolución codificada, para resolver diferentes problemas ambientales.

Otro caso parecido es el del reconocimiento de formas. En 1958, Rosenblat presentó el *Perceptron*, máquina en red que debia ser capaz de simular la visión humana; máquinas demasiado sencillas para la tarea que se les había encomendado.

Sin embargo, no todo fue fracaso. Newell, Shaw y Simon desarrollaron el *Logic Theorist*, en 1959 Gelernter escribió su programa para resolver problemas de geometría elemental. Slage comenzaba en el MIT la automatización de la integración simbólica con su programa SAINT, origen de lo que unos años más tarde sería el programa MACSYMA.

En 1960, los investigadores del MIT comienzan un proyecto sobre Inteligencia Artificial bajo la dirección de John McCarthy y Marvin Minsky.

Por supuesto que la fé en las posibilidades del << ordenador pensante >> no era compartida por todos. J.R. Lucas (en 1961), planteó una objeción bastante sensata: << como puede pretenderse que una máquina iguale el funcionamiento de la mente humana >>.

De todas formas, el resultado más espectacular de este período fue el programa de Samuel para jugar a las damas, que se presentó en 1961 y era capaz de aprender de su experiencia, es decir, tener en cuenta sus errores y éxitos pasados, para determinar sus jugadas en una partida posterior.

1.1.3 Los años difíciles de la Inteligencia Artificial

Tales habían sido las expectativas levantadas por la Inteligencia Artificial, y tantos sus fracasos, que el desánimo sucedió al optimismo inicial. El mundo exterior se desentendió de los trabajos de investigación, y la financiación de muchos proyectos se volvió problemática, tanto en América como en Europa. No obstante, la Inteligencia Artificial se fué consolidando y, aprendiendo de sus fracasos, buscó nuevos enfoques para los viejos problemas.

Bajo la dirección de E.A. Feigenbraum y J. Feldman se publicó en 1963 la colección *Ordenadores y razonamiento*; en ella aparece el artículo de M. Minsky *Pasos hacia la Inteligencia Artificial*.

En el año de 1964 se publicó la tesis doctoral de D.G. Bobrow sobre su sistema STUDENT, que es un programa de lenguaje natural que comprende y resuelve problemas elevados de algebra.

Posteriormente, en 1965, la Universidad de Stanford empezó a investigar sobre Sistemas Expertos con su proyecto de *Proyecto de Programación Heurística* (HHP). En este año se comienzan los trabajos de investigación sobre el primer Sistema Experto: el DENDRAL, desarrollado también en la Universidad de Stanford por un grupo en el que estaba J. Lederberg, E.A. Feigenbaum, B.G. Buchanan y otros. DENDRAL analiza información sobre componentes químicos para determinar su estructura.

Es en 1966 cuando se publica, en *Comunicaciones de la Asociación para Máquinas Calculadoras*, un programa de ordenador para el estudio de comunicación hombre-máquina

mediante el lenguaje natural interactivo, ELIZA, que fué creado por Weizenbaum como un programa de psicología que simula las respuestas de un terapista en diálogo interactivo con un paciente.

La polémica sobre la Inteligencia Artificial y sus posibilidades tuvo uno de sus episodios más curiosos, protagonizado por el profesor Weizenbaum, pues su programa ELIZA era un simulador efectivo y sencillo de un << psicoterapeuta no directivo >>. Las personas que dialogaban con ELIZA creian que hablaban con un psicologo auténtico. De ahí surgió la idea de usar programas de ordenador para el diagnóstico y la terapia de los trastornos paíquicos.

También en el año 1966 R.D. Greenblat empieza a desarrollar un ordenador para jugar al ajedrez, capaz de competir con éxitos en torneos. Fue bautizado como *programa de ajedrez de Greenblat*.

Entre 1968 y 1972 se construyó, por la SRI Internacional un robot móvil, SHAKEY, capaz de recibir instrucciones y de planear acciones inteligentes para realizar tareas. También en el año 1968 M. Minsky publicó el *Procesamiento de información semántica*.

Fué apartir de 1969 cuando se produjo la institucionalización de la comunidad científica que trabaja en Inteligencia Artificial, al tener lugar el Primer Congreso Internacional de Inteligencia Artificial. En 1970, apareció el primer número de la revista *Inteligencia Artificial*, que desde entonces publica trabajos acerca de las más destacadas investigaciones en curso. ¿ En qué consistió este nuevo enfoque ? Dicho brevemente, el énfasis se trasladó de las reglas y procedimientos generales de

deducción a la acumulación de conocimientos concretos acerca de un campo bien delimitado de la realidad. Una muestra de ello la tenemos en el célebre programa SHRDLU de Terry Winograd, presentado al comienzo de este período, y que era una parte de un proyecto de comprensión del lenguaje natural capaz de comprender y ejecutar correctamente órdenes dadas en inglés acerca de << *mundo de bloques* >>. Ello era posible poque el programa tenía todos los conocimientos necesarios acerca de su limitado y simplificado mundo.

En 1970, P.H. Winston publicó una tesis doctoral *Descripciones de ejemplos de aprendizaje estructural*, que describe un programa, ARCHES, que aprende de ejemplos.

En este año también el proyecto de la Inteligencia Artificial de MIT, se convierte en Laboratorio de Inteligencia Artificial del mismo bajo la dirección de M. Minsky y S. Papert.

El mismo año, y en la Universidad de Pittsburgh, J. D. Myers y H. E. Pople, empiezan a trabajar sobre un sistema, INTERNIST, para ayudar a los médicos en diagnóstico de las enfermedades humanas. De la misma forma, Alain Colmerauer y sus colegas empiezan el desarrollo del lenguaje de programación PROLOG.

En el año 1971, y en la Compañía SRI internacional, N. Nilsson y R. Fikes completan sus trabajos sobre el STRIPS, que planifica proyectos mediante secuencia de operadores. También en este año empezó a usarse el programa MACSYMA. El programa realiza cálculos integro-diferenciales y simplifica expresiones simbólicas. Las entradas y salidas son simbólicas y el programa es una base de conocimientos usada ampliamente por matemáticos, investigadores, físicos e ingenieros.

Entre 1971 y 1976, la Agencia de Proyectos Avanzados para la defensa de los E.E.U.U. (DARPA) financió las investigaciones relacionadas con la capacidad de comprensión del lenguaje dentro del Programa de Investigación para la Comprensión del Lenguaje (SUR), algunos de los programas resultantes fueron: SPEECHLIS, HWIN, HEARSAY I y II, DRAGON y HARPY.

Hacia 1972, W. Woods desarrolló un sistema de recuperación de la información para un sistema gramatical de lenguaje natural (LUNAR), que fue usado por los geólogos para evaluar los materiales traídos de la Luna por la misión Apolo XI.

En 1973, se constituyó una comunidad de investigación soportada por el Instituto Nacional de Salud, la SUMEXAIM (Proyecto Experimental de Inteligencia Artificial en Medicina con Ordenador de la Universidad Médica de Stanford), para el desarrollo de técnicas de Inteligencia Artificial.

En 1975, DARPA financia la investigación en visión artificial a través del *Programa de Comprensión de Imagen*, que incluye un desarrollo de la teoría de la visión y del hardware para el procesado de imágenes. También este año se publicó la colección *La Psicología de la visión con ordenador*, y dentro de ella *Una base para representación del conocimiento*, que discute la utilidad de las estructuras en rejillas para organizar el conocimiento en sistemas y que incluyen lenguaje natural y visión. También se cita el trabajo *Comprendiendo las líneas sacadas de escenas con sombras*, que discute una nueva forma para usar los límites sombreados para interpretar imágenes visuales.

También en ese año se publicó el libro *Representación e*

Inteligencia de D.G. Bobrow y A. Collins, que incluye trabajos importantes sobre representación del conocimiento.

H. L. Dreyfus lanzó ataques contra la Inteligencia Artificial desde otro punto de vista. Basándose en la introspección afirmó que ciertos aspectos del pensamiento humano son esencialmente inimitables por las razones que el propuso: los ordenadores son máquinas discretas, en consecuencia carecen de corporeidad y no pueden compartir nuestras experiencias.

En cualquier caso, las controversias no desanimaron a los investigadores, que empezaban a alcanzar resultados tangibles. En esto les ayudaba grandemente el acelerado progreso del soporte material, con la aparición de la integración de los circuitos y el aumento consiguiente de capacidad y potencia de cálculo. El soporte lógico empezaba a adaptarse a las necesidades de la Inteligencia Artificial. En 1975, Alain Colmerauer define el lenguaje PROLOG de programación lógica. Un instrumento así libera al programador de la necesidad de especificar los procedimientos de resolución de un problema: sólo hay que enunciar los hechos y reglas conocidas y señalar las metas a las cuales se quiere llegar.

1.1.4 El desarrollo actual

Las instituciones públicas y los gobiernos percibieron los grandes adelantos que, calladamente, habia realizado la Inteligencia Artificial en la decada de los 70's. Las empresas y organizaciones se vieron seducidas por las posibilidades que los Sistemas Expertos y la << ingenieria del conocimiento >> tenían para aumentar su eficiencia: el siempre citado caso del yacimiento de molibdeno descubierto por PROSPECTOR.

En 1979, el Ministerio de Industria y Comercio Internacional del Japón decidio desarrollar una nueva generación de ordenadores que cumplieran las necesidades previsibles de la decada de los 90's. En 1981 se publicaba el informe elaborado con las contribuciones de 150 personas, aprobando el gobierno japonés los créditos presupuestarios para lo que se llamó *el Proyecto de la Quinta Generación de Ordenadores*. En ese mismo año se publicó el primer volumen del *Handbook de Inteligencia Artificial*, por A. Barr.

Un año más tarde, en 1982, se lanza oficialmente en Tokio, el ICOT, Instituto Japonés para Generación de Nuevas Tecnologías de Cálculo. Paralelamente, se crea en los E.E.U.U. la MCC, Corporación Tecnológica de Microelectrónica y Microordenadores, para responder al programa japonés de la Quinta Generación. También el Reino Unido pone en marcha el Programa ALVEY de Tecnología Avanzada de Información, para realizar investigación sobre ordenadores de Quinta Generación.

En Noviembre de 1984 se celebró una Conferencia Internacional sobre la Quinta Generación. Sus actas muestran los resultados conseguidos hasta ahora por el proyecto, las dificultades y los próximos pasos que se darán.

1.2 Concepto de Inteligencia Artificial

Para hablar de una ciencia, aún a nivel introductorio, es necesario previamente fijar qué se entiende por el nombre que se le ha dado, mas en el caso cuando se trata de muy reciente surgimiento.

Una forma ingenua por la que se puede atravesar para definir lo que es la Inteligencia Artificial, podría pasar

por la definición de las palabras que componen la nueva ciencia << Inteligencia >> y << Artificial >>. Por desgracia, nos encontramos ante dos de esas voces que se resisten a todos los intentos de encerrarlas en unos límites claros. Por ello se utilizará una vía indirecta, proponiendo un experimento cuya realización nos permita determinar si un ente tiene o no inteligencia.

Se trata del conocido test de Turig (Turing 1950). Consideremos el siguiente juego al que llamaremos << juego de imitación >>. En él participan tres personas: un hombre A, una mujer B y un interrogador C, de uno u otro sexo, que se sitúa en una habitación aparte y ha de determinar cuál de los otros dos es el hombre y cual es la mujer. Para ello C puede plantear preguntas tanto a A como a B, que responderán de forma que la vía de comunicación no sirva de ayuda al interrogador. A y B no están obligados a decir la verdad. Si C no es capaz de descubrir el sexo en un tiempo razonable, se considera que ha perdido el juego.

Ahora surge la pregunta: ¿ Qué sucede cuando una máquina sustituye a A en el juego ? ¿ Se equivocará tanto el interrogador como lo hace cuando en el juego participan un hombre y una mujer ? Si así fuera, no cabe duda de que podríamos hablar de una << máquina pensante >>; podríamos decir que se consiguió una manifestación de Inteligencia Artificial.

Evidentemente, ninguna máquina del tiempo de Turing, ni aún del nuestro, tiene la menor posibilidad de superar este test. Sin embargo, Turing afirmaba:

<< Personalmente, creo que dentro de unos cincuenta años se podrán programar perfectamente ordenadores con una capacidad de almacenamiento de diez elevado a la novena

potencia, para hacerlos jugar tan bien al juego de imitación que un interrogador normal no tendra mas del 70% de posibilidades de efectuar la identificacion correcta a los cinco minutos de plantear preguntas >>.

Pero se sigue sin llegar a una definición ni siquiera aproximada de lo que es la Inteligencia Artificial. Son dos los objetivos que se pueden dar cuando se investiga en este campo; se hará hincapié en uno u otro, por lo mismo se darán definiciones diferentes.

El primer objetivo puede ser el estudio de los procesos cognoscitivos en general, es decir, el desarrollo de una teoria sistemática de los procesos inteligentes, se produzcan donde se produzcan. Bajo este aspecto, la definición mas concisa es la de Hayes: << El estudio de la inteligencia como computación >>.

El segundo objetivo es la consecución de sistemas automáticos capaces de llevar a cabo tareas hasta ahora reservadas en exclusiva a los seres humanos. Se podria definir asi a la Inteligencia Artificial como una disciplina tecnológica que tiene por objeto el diseño y la construcción de máquinas y programas capaces de realizar tareas complejas con pericia igual o mayor que la de un ser humano. Hay que tener en cuenta que aquellas que más simples parecen son las más complejas, por ejemplo, hablar, ver y escuchar.

Para el primero de los dos enfoques anteriores, el ordenador es una poderosa herramienta por su capacidad para representar cualquier sistema discreto de símbolos físicos. Pero la meta final es el estudio de la conducta inteligente y, en particular, de la conducta humana.

Este es el motivo por el cual la Inteligencia

Artificial enlaza con tantas otras ciencias. Así en el estudio de la conducta humana intervienen ciencias naturales como la neurofisiología y ciencias humanas como la psicología; en el estudio de los procesos de aprendizaje, la pedagogía; en otros aspectos importantes, como el reconocimiento del lenguaje natural, la lingüística; y en la realización práctica de los modelos, la informática y todo su entorno.

Aún cuando la Inteligencia Artificial es un campo relativamente joven (existe sólo desde hace tres decadas) y no existen definiciones estrictas sobre lo que es o representa, he aquí como la definen algunos expertos:

1.- Para M. Minsky profesor del MIT, << la Inteligencia Artificial es la ciencia de hacer máquinas que hacen cosas que realizadas por el hombre requieren el uso de inteligencia >>.

2.- No obstante, P.H. Winston director del Laboratorio de Inteligencia Artificial del MIT escribe: << El objetivo de la Inteligencia Artificial se puede definir como conseguir hacer ordenadores más útiles para comprender los principios que hacen posible la inteligencia >>.

3.- N. Nilson jefe del Departamento de Ciencia de Ordenadores de la Universidad de Stanford dice: << El campo de la Inteligencia Artificial tiene su principal contenido en aquellos procesos comunes que reúnen percepción y conocimiento, amén que el proceso pueda ser comprendido y estudiado científicamente >>.

4.- Tanto B. G. Buchanan (profesor adjunto e investigador de ciencia de ordenadores) como E. A. Feigenbaun (investigador principal del proyecto de Investigación sobre Heurística), ambos de la Universidad de Stanford, escriben << la investigación sobre Inteligencia Artificial es la parte de la ciencia de ordenadores que investiga procesos simbólicos, razonamientos no algoritmicos y representaciones simbólicas de conocimiento usados en máquinas inteligentes >>.

1.3 Areas de la Inteligencia Artificial

Las aplicaciones tecnológicas en las que los métodos de Inteligencia Artificial usados han demostrado con éxito que pueden resolver complicados problemas, se han desarrollado en sistemas que:

1.- Permiten al usuario preguntar a una base de datos en cualquier lenguaje que sea, mejor que un lenguaje de programación.

2.- Reconocen objetos de una escena por medio de aparatos de visión.

3.- Generan palabras reconocibles como humanas desde textos computarizados.

4.- Reconocen e interpretan un pequeño vocabulario de palabras humanas.

5.- Resuelven problemas en una variedad de campos usando conocimientos expertos codificados.

Los países que han apadrinado investigaciones en la Inteligencia Artificial han sido: E.E.U.U., Japón, Reino Unido y la CEE (Comunidad Economica Europea); lo han llevado a cabo a través de grandes compañias y cooperativas, así como universidades, para resolver problemas ahorrando dinero. Las aplicaciones más primarias de la Inteligencia Artificial se clasifican en cuatro campos, (figura 1.1): *Sistemas Expertos, lenguaje natural, robotica y visión, sistemas sensores y programación automática.*

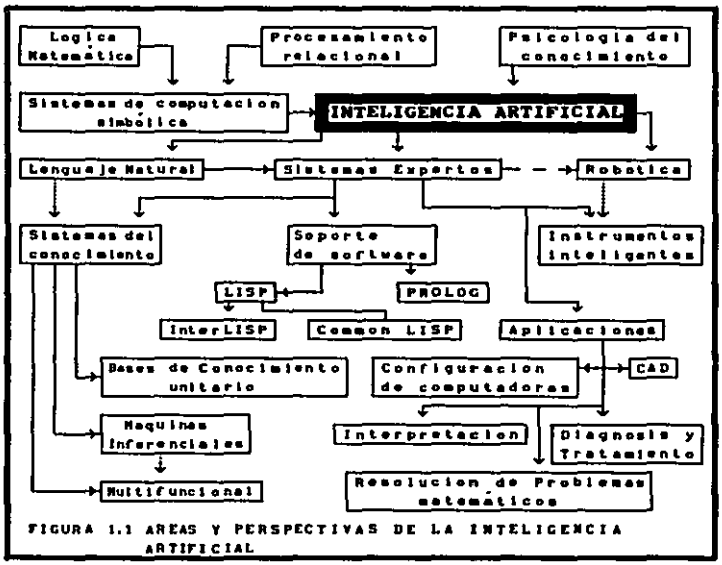


FIGURA 1.1 AREAS Y PERSPECTIVAS DE LA INTELIGENCIA ARTIFICIAL

1.3.1 Lenguaje Natural

Cuando un usuario decide comunicar algo a un sistema informático es porque desea que éste haga algo. Para

establecer esa comunicación, debe generar una declaración y transmitírsela al ordenador. El sistema debe entonces comprender la declaración, o sea, debe traducirla a un sistema de acciones apropiadas comprensibles por el ordenador antes de hacer o ejecutar alguna cosa. Por eso la comprensión es un proceso de traducción, en donde una declaración es entendida solamente respecto a un lenguaje particular y a un conjunto de acciones.

Los lenguajes naturales son atractivos porque son la herramienta de comunicación más poderosa entre las personas. Además, todo el mundo conoce lo suficiente sobre su propia lengua.

Existen una serie de normas o reglas que permiten determinar cuándo cuando es conveniente en un proyecto usar una interfaz en lenguaje natural. Entre otras son:

1.- El lenguaje natural es más costoso de interpretar que los lenguajes convencionales de programación, bien sea desarrollando todos los elementos básicos o adquiriéndolos por separado y adaptándolos a la aplicación.

2.- Aunque es fácil de aprender, pues todo el mundo conoce su propia lengua, siempre que se use un subconjunto de la entidad completa del lenguaje, el usuario debe conocer cuáles son los límites de operación.

3.- El lenguaje no reconoce declaraciones que estén fuera de su campo de actuación, de su forma de análisis gramatical o de su contexto de interpretación de la declaración.

4.- Debe ser conciso, para evitar un número excesivo de operaciones para interpretar la declaración.

5.- Debe ser preciso, pues la ambigüedades y declaraciones con más de un significado son uno de los grandes problemas del lenguaje natural.

Por otra parte, cuando un operador humano procesa una sentencia se crean varios grupos conceptuales para proveer un método para su entendimiento. El proceso de crear esas estructuras es tan importante como las mismas estructuras; el significado de una sentencia está relacionado no sólo con la forma de procesarla sino con el grado de fragmentación o el número y tipo de estructuras no adecuadas gramaticalmente, tanto en el lenguaje escrito como en el oral, son también consideraciones importantes para que el ser humano comprenda su significado o necesite elaborar varias alternativas fiables y posibles.

1.3.2 Robótica

La palabra robot no significa lo mismo para todo el mundo, por eso los investigadores de la Inteligencia Artifical trabajan por distintas vías para añadirle inteligencia a estas máquinas de distintas formas, tales como: visión, sensores táctiles, planificación o aprendizaje. Hasta ahora los robots disponibles comercialmente están dotados de visión sencilla y con algunos sensores táctiles.

Los robots se pueden clasificar en tres grupos principales: *robots industriales, robots androides y entornos automatizados* (factorias).

1.3.2.1 Robots industriales

Los investigadores intentan dotar a los robots de capacidad para seleccionar cosas sencillas entre varias posibilidades, manipulando las partes inteligentemente en entornos para los cuales no estaban programados. También están intentando aumentar la velocidad de respuesta a entradas de sus sensores. Igualmente es importante la facilidad para frenar rápida y exactamente ante obstáculos. Otro tipo es útil en la << fabricación poco ensamblada >>, en la que los robots sujetan y manipulan algunas partes con un alto grado de libertad, en contraste con el tipo de fabricación que requiere que determinadas partes se coloquen rápidamente en posiciones precisas.

1.3.2.2 Robots androides

Los robots tipo << androide >>, que hablan, pertenecen todavía a la ciencia-ficción. No obstante ha sido desarrollada una excitante máquina con una pierna elástica y ligera, así como otros tipos de robots móviles, útiles para transportar piezas, que incluso pueden ser usadas para transporte humano. Este tipo de robots entran en un grupo más amplio denominado << vehículos guiados automaticamente >>. En ellos los más complejos son programables y pueden cargar, descargar y ser controlados por radio.

Para poder bordear los obstáculos, los robots necesitan más movilidad de la que tienen actualmente. Para dotarlos, y que de una forma inteligente puedan realizar estas tareas a través de terrenos y obstáculos no familiares, deben tener unos buenos sistemas sensores (radar, sonar, táctiles y capacitivos).

1.3.2.3 Factorías

El esfuerzo en esta área es crear sistemas que se puedan modificar de forma rápida y barata para producir así variaciones en productos u otros nuevos. Puede ser útil en operaciones que obtienen de una vez un volumen de producción medio, con un alto nivel de variaciones en ellos.

1.3.3 Sistemas de visión

Estos sistemas reciben la forma, tamaño, localización, aristas o color de los objetos. Procesan e interpretan las imágenes para analizarlas e identificarlas. Han probado ya su utilidad para el análisis de recursos naturales desde sensores remotos, así como en la fabricación de partes escogidas (para ensamblaje y control de calidad) y en medicina para el examen interno del cuerpo.

Los investigadores quieren dotar a los robots de los sistema más avanzados de visión, extendiendo así la capacidad de estos para moverse en entornos complicados de manipulación de piezas.

Los sistemas de visión de alto nivel usan técnicas de Inteligencia Artificial (razonamiento simbólico) en su lucha contra las ambigüedades tales como: identificación de objetos según modelos, comparación de aspectos de los mismos con parámetros almacenados, así como los problemas de obstrucción. Estos sistemas de alto nivel de interpretación de imagenes están equipados con reglas heurísticas acerca del tamaño, forma y relaciones espaciales para su interpretación como imágenes.

Hay sistemas de visión de dos o tres dimensiones; los

primeros interpretan objetos o escenas de dos dimensiones, tales como fotografías aéreas o de secciones al microscopio.

1.3.4 Sensores táctiles

Estos sensores evalúan los objetos atendiendo a su peso, forma, textura, dirección, vibración, presión y temperatura. Los robots se beneficiarán de estos sensores, pues les suministrarán información información sobre áreas que no pueden ver cuando trabajan sus mordazas. También se usan cuando se necesita un control preciso al empuñar objetos.

Al igual que los sensores de presión, los táctiles utilizan tratamientos análogos al de imágenes: análisis, identificación e intensificación de imágenes con aproximaciones de bajo y alto nivel. Mediante la primera, el sensor recibe una impresión enviada como imagen, extrayéndose de ella los hechos para desarrollar e interpretar posteriormente el modelo de imagen objeto. En la segunda aproximación, el sistema comienza con una aproximación sobre el tipo de objeto de que se trata, y esta hipótesis es la que ordena el experimento; el sistema lo desarrollará para ver si se verifica la hipótesis. Combinando los sistemas de visión con los sistemas táctiles los robots serán mucho más versátiles.

1.3.5 Programación inteligente y programación automática

La primera es la aplicación de las técnicas de la Inteligencia Artificial a la construcción y transformación de programación automática. Para conseguir estos objetivos es necesario que los sistemas de programación automática

incorporen dos clases de conocimientos: acerca de los principios generales de programación y sobre el campo específico de la aplicación.

El objeto de la programación automática (la síntesis) es construir programas a partir de las especificaciones rigurosas, y no algorítmicas, que describen lo que hay que hacer, probando también el funcionamiento del programa mediante el uso de técnicas matemáticas para mostrar que se comporta de acuerdo con las especificaciones formales. Mediante compiladores construidos con las técnicas de Inteligencia Artificial (bases de conocimiento) se consigue que los códigos máquina a ejecutar sean más eficientes.

1.3.6 Sistemas Expertos

Una de las áreas de Inteligencia Artificial en la que más dinero se está invirtiendo es la de los Sistemas Expertos, también llamados sistemas basados en conocimiento. Estos aplican técnicas de razonamiento de Inteligencia Artificial a la resolución de problemas en áreas específicas para simular la aplicación de expertos humanos. La efectividad de uno de tales sistemas estriba en la cantidad de conocimiento que se le suministre. Se usan a menudo como asistentes o consultores inteligentes de los usuarios humanos para resolverles problemas rutinarios, dejándoles así libres para que se dediquen a aplicaciones más novedosas e interesantes.

Entre los logros que se pueden resaltar de Sistemas Expertos son:

1.- Actuar como localizador de averías en equipos de enseñanza.

2.- Asesorar médicamente sobre tratamientos de sospechosos de meninjitis y otras infeciones bacterianas en la sangre.

3.- Deducir la localización de grandes depósitos de molibdeno.

4.- Configurar sistemas complicados de cálculo en la fracción de tiempo requerida por el ingeniero experimentado.

Algunos Sistemas Expertos, entre ellos los sistemas tradicionales, suministran contestaciones en términos de medidas fiables, propagando a través del programa grados de certidumbre asociados a partes de información.

Se tardaron varios años en conseguir que un Sistema Experto fuese rápido y ese tiempo se redujo conforme los que trabajaban en su desarrollo ganaron familiaridad con los métodos para desarrollarlos, así como el hardware y el software que posibilita el proceso de desarrollos posteriores.

1.3.6.1 Sistemas expertos básicos

Hay un contraste entre las técnicas tradicionales de proceso de datos y las técnicas de resolución y razonamiento en la Inteligencia Artificial. Mientras las primeras usan cálculo numérico con algorítmos comprensibles, las segundas permiten a los Sistemas Expertos extraer conclusiones que no estaban programadas en ellos explícitamente. Algunas técnicas y elementos que hacen posible realizar inferencias novedosas en Sistemas Expertos: adquisición de conocimientos, heurística, métodos de representación de conocimiento y máquinas de inferencia.

1.- La *adquisición del conocimiento* es el proceso para extraer y formalizar el conocimiento de un experto y usarlo en un Sistema Experto.

2.- La *heurística* son reglas empíricas concernientes a una determinada área que un investigador aprende o descubre.

3.- Una *representación del conocimiento* es una estructura normal con un conjunto de operaciones que expresan descripciones, realizaciones y procedimientos que un experto proporciona a un Sistema Experto.

4.- Una *máquina de inferencia* es un protocolo de un programa para pilotear y resolver un problema, entre reglas y datos, en una representación del conocimiento.

CAPITULO II

DESARROLLO DE SISTEMAS DE INTELIGENCIA ARTIFICIAL

AL INTELIGENTE SE LE PUEDE
CONVENCER; AL TONTO PERSUADIR
CURT GOETZ

2.1 Introducción

En el procesamiento de datos tradicional, el sistema procesa el contenido de variables. Los sistemas de Inteligencia Artificial pueden también hacer esto, pero tienen la capacidad de manipular símbolos independientemente de sus valores. Esto hace posible resolver un problema cuando el valor de una variable no se conoce hasta un momento antes de que la respuesta sea necesaria.

En el proceso de datos, el programador, no la máquina, es quien determina todas las relaciones entre los símbolos. Pero en el procesamiento de símbolos, en un sistema basado en Inteligencia Artificial, el programa puede determinar las relaciones entre parte de los datos, es una característica de la programación en Inteligencia Artificial.

El aspecto distintivo de la Inteligencia Artificial es el énfasis en el almacenamiento y manipulación de las relaciones entre símbolos. Así, la tecnología de la Inteligencia Artificial tiene que ver con la representación, no exactamente de datos, sino más bien con el conocimiento encarnado en las relaciones entre los datos. Las sentencias que expresan áreas particulares de conocimiento son tratadas como datos en un programa de Inteligencia Artificial.

2.2 Sistemas Prototipo

En la producción de la mayor parte de los problemas de Inteligencia Artificial, el diseño de la solución no puede ser conocido de antemano. En lugar de eso, se recurre a una programación explicatoria, usando varias técnicas de

resolución e intentando producir una solución prototipo para una pequeña parte del problema de manera rápida.

Mejorando continuamente la capacidad del prototipo, el diseñador moldea el diseño para un sistema que por fin acaba haciendo lo que se pensaba. En el transcurso de todo este proceso, los sucesivos prototipos proporcionan un beneficio adicional: el sistema útil en el paso intermedio estando lejos del sistema convencional de partida. Un sistema experto cuidadosamente construido, por ejemplo, puede ser útil al 30-50% del nivel final. Los resultados del uso del sistema permiten a los usuarios proporcionar retroalimentación en la ejecución del trabajo. La retroalimentación se usa entonces para mejorar la ejecución, y el proceso continúa hasta que el sistema alcanza el nivel deseado. Un beneficio adicional del último prototipo es que el sistema es capaz de mostrar sus utilidades y limitaciones potenciales, antes de que gran cantidad de recursos se hayan puesto en juego. Un proyecto que no va a producir claramente los resultados pedidos, puede ser detenido antes de que consuma una mayor cantidad de recursos.

El mejor método de desarrollo de un prototipo al que se le pueden aplicar las técnicas de Inteligencia Artificial, es determinar la estructura del problema y las herramientas disponibles. Las herramientas consisten básicamente en varios métodos de representación del conocimiento y operadores, estructuras de control, lenguajes de Inteligencia Artificial, utilidades integradas (editores, depuradores, herramientas de gestión de codigos, gestión de pruebas, etc.) y hardware.

2.3 Representación del conocimiento

Los métodos de representación del conocimiento son combinaciones de estructuras de datos para almacenar junto con procedimientos de interpretación, para hacer inferencias sobre los datos almacenados y así mismo disminuir la búsqueda referida por el programa. Esta representación proporciona una descripción de comó estan interactuando los grupos de información.

Los investigadores de Inteligencia Artificial han creado una variedad de representaciones diferentes para los distintos tipos de conocimientos y no una simple, cerrada y definitiva representación.

Las formas de representación del conocimiento y las caracteristicas de cada una de ellas se trataran en un apartado especifico de esta tesis.

2.4 Reglas de producción

Las reglas de producción son un tipo de regla <<Si...Entonces...>>, basada en condiciones y acciones. Las descripciones de una aplicación dada o contexto de un problema son fundidas en una colección de condiciones, en unas reglas que hacen que las acciones de la regla a ejecutar den lugar a nuevas descripciones que producen más acciones, y así hasta que el sistema encuentre solución o se detenga. Las reglas de producción son los operadores en el sistema; las que se usan para manipular las bases de datos.

En un capitulo posterior se analizan las reglas de

producción en una forma más especifica asi como las estructuras de control de estas.

2.5 Arboles de búsqueda en las representaciones Espacio-Estado

Una de las herramientas más importantes en Inteligencia Artificial, son los árboles de búsqueda que permiten manejar problemas complicados cuya solución se dificulta por una proliferación de posibilidades, que generan a su vez nuevas posibilidades. La jerarquía de estas posibilidades en un caso concreto se describen mediante lo que se conoce como un árbol de posibilidades o árbol de búsqueda.

Un estado es un conjunto de condiciones o valores que describen un sistema en un punto especifico durante el proceso. Una representación espacio-estado es aquella en la que los operadores usan un nuevo estado, y sólo uno, en la base de datos, cada vez que se emplean. Las representaciones espacio-estado, que utilizan varios métodos de búsqueda a través de la base de datos para alcanzar soluciones, han sido usadas por juegos de ajedrez, sistemas para encontrar rutas, y problemas que incluyan muchos operadores y muchos posibles estados.

La búsqueda de un camino de solución a través del espacio-estado se puede ilustrar gráficamente mediante nodos y enlaces de conexión. Cada nodo representa un estado del sistema, y los enlaces representan la acción de un operador para cambiar el sistema de un estado a otro. Los nodos pueden tener punteros que señalen al nodo origen, de forma que cuando se alcance la solución podamos conocer su camino. En algunos métodos de búsqueda, los punteros también se usan

para marcar los caminos falsos.

Los gráficos son dibujos generalmente como árboles o redes. Un árbol es un gráfico que empieza por la cima con un nodo raíz, de manera que los nodos de un nivel tienen un solo origen. Esta estructura presenta solamente un camino desde el nodo raíz a otro nodo cualquiera. Los enlaces o conexiones entre nodos en un árbol se llaman ramas o ramificaciones. Los nodos en la parte inferior del árbol se denominan nodos terminales (figura 2.1).

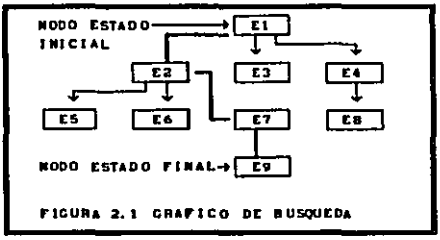


FIGURA 2.1 GRAFICO DE BUSQUEDA

Hay potencialmente un árbol de un espacio-estado que representa cada posible estado del sistema y otro árbol más limitando del camino que puede ser construido si el programa busca la respuesta eficientemente. Puede ser importante encontrar encontrar un modo eficiente de búsqueda porque el espacio potencial de búsqueda puede ser infinito o, al menos, tan grande que haga al proceso no viable. La clasica demostración de este problema, conocido como la explosión combinatoria, fué descrita por el doctor Claude E. Shannon en un ejemplo basado en el juego del ajedrez.

2.5.1 Métodos de búsqueda

Dentro de la Inteligencia Artificial, se conoce con el nombre de *búsqueda heurística* a un núcleo de ideas básicas utilizadas en muchos procedimientos inteligentes de resolución de problemas. Delante de muchos problemas, desconocemos en principio el algoritmo de resolución, ya sea porque el problema es demasiado complejo o ya sea porque el problema y su entorno cambian con el tiempo. La solución al sernos desconocida inicialmente, no hay más remedio que buscarla por el *método de prueba y error*. Hay que buscar la solución tanteando entre las acciones aplicables al problema, hasta encontrar una secuencia de ellas que conduzca a la solución.

Las acciones u *operaciones* posibles a realizar son transiciones desde una posición inicial a una posición final; dicho con otras palabras existe un *estado inicial* que debe alcanzar un *estado final*, pasando por unos estados intermedios. El gráfico representa lo que se ha dado en llamar *espacio de búsqueda*: el conjunto de estados posibles y de transiciones entre estados. Los caminos solución son secuencias de transiciones que conducen del estado inicial al estado final. En general nos interesa buscar el camino de mínimo costo o *camino óptimo*.

Una forma de reducir el costo de la búsqueda es utilizar información que nos ayude a elegir la transición más adecuada en cada caso. Este tipo de información que nos ayuda a descubrir la solución recibe el nombre de *información* y la *búsqueda heurística* es, pues, la búsqueda que utiliza dicha información.

La búsqueda de un camino solución entre los muchos caminos posibles, y la mayoría de los problemas interesantes también tienen espacios de búsqueda enormes; el número de elecciones o combinaciones posibles crece exponencialmente al avanzar la búsqueda. Esto es lo que se ha llamado *explosión combinatoria*.

En general, el comportamiento inteligente estriba en gran medida en encontrar y utilizar conocimientos que, limitando la búsqueda de soluciones, permitan resolver problemas que de otra forma serían intratables.

Búsqueda con encadenamiento hacia atrás: En esta búsqueda, el orden de expansión de los nodos es desde el nodo inicial descendiendo en un camino dado hasta que se encuentre la respuesta, o hasta que la búsqueda llegue a un determinado límite de profundidad. Si la búsqueda excede del límite o del fin de ese camino, el sistema retrocede y la búsqueda continúa en el nodo más cercano. Este tipo de búsqueda es particularmente eficaz cuando hay muchos caminos largos que conducen a una solución (figura 2.2a).

Búsqueda a lo ancho: En esta búsqueda, todos los nodos en un mismo nivel son expandidos antes de bajar a cualquiera de los nodos que están en un nivel inferior. Si en esta búsqueda debemos bajar hacia un nivel inferior, el proceso comienza desde un lado y moviéndose horizontalmente. La búsqueda a lo ancho se adapta perfectamente a sistemas que tienen unos pocos caminos de solución, evitando malgastar tiempo (figura 2.2b).

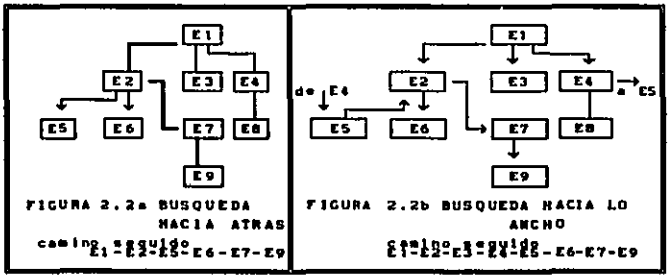


FIGURA 2.2a BUSQUEDA HACIA ATRAS
camino seguido E1-E2-E5-E6-E7-E9

FIGURA 2.2b BUSQUEDA HACIA LO ANCHO
camino seguido E1-E2-E3-E4-E5-E6-E7-E9

Búsqueda de encadenamiento hacia arriba: Este método empieza de la misma forma que el de búsqueda hacia atrás, pero la elección de qué nodos descendientes serán alcanzados se hace estimado cuál es el más cercano a la meta. Esto es suficiente cuando hay alguna manera de medir distancias al objetivo final; sin embargo, hay condiciones engañosas en las que la distancia real no está muy bién definida. Este método de búsqueda proporciona un sistema con medios para superar tales condiciones, tal como retornar y ensayar un camino que fué abandonado antes o aplicar más de una regla antes de evaluar el resultado en una orden a saltar a una parte diferente del espacio de búsqueda.

Búsqueda bidireccional: Se puede partir de dos tipos de sistemas de producción, los que parten desde una base de datos inicial hacia un objetivo final o base de datos final y los que parten desde una base de datos final y aplicando reglas en sentido o dirección inverso se acercan a la base de datos inicial.

Al primer tipo de sistema de producción se le llama *sistema de producción de avance por síntesis*, ya que intentan

sintetizar la solución a partir de datos iniciales; y el segundo tipo recibe el nombre do *sistemas de producción analítico*, ya que va analizando el objetivo o solución reduciéndolo a subjetivos sucesivos hasta llegar a los datos iniciales.

La complejidad de un problema no es invariante respecto a la dirección de búsqueda. Para algunos problemas, el mejor método de búsqueda es una combinación de los dos anteriores. Estos sistemas de producción, llamados *bidireccionales*, pueden utilizar reglas en las dos direcciones, partiendo de la base de datos inicial y de la base de datos final.

El proceso terminará cuando las dos fronteras de búsqueda, la sintética y la analítica se encuentren en forma apropiada. La figura 2.3 (a) muestra un caso favorable de búsqueda bidireccional, donde se han expandido menos nodos que en la búsqueda unidireccional.

La figura 2.3 (b) muestra un caso desfavorable, donde las dos fronteras de búsqueda no intersectan, resultando una expansión de nodos doble que la búsqueda unidireccional.

Búsqueda en gráficos Y/O: Una técnica conocida de resolución de problemas consiste en descomponer el problema en subproblemas independientes y más pequeños y luego resolver todos los subprolemas por separado. Los sistemas de producción cuyas bases de datos son descomponibles, reciben el nombre de *Sistemas de Producción Descomponibles*. Un ejemplo ilustrara esta técnica:

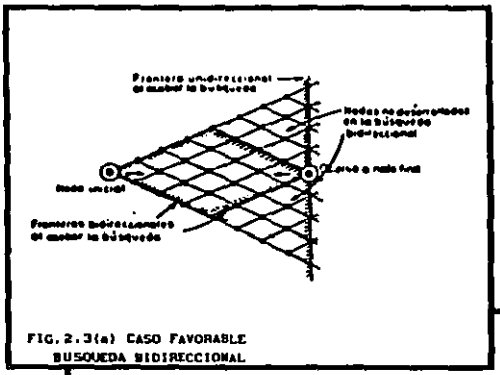


FIG. 2.3(a) CASO FAVORABLE
BUSQUEDA BIDIRECCIONAL

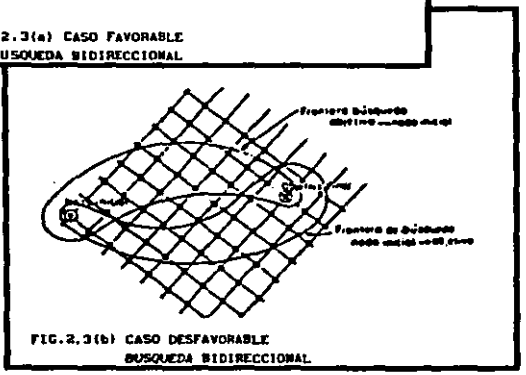


FIG. 2.3(b) CASO DESFAVORABLE
BUSQUEDA BIDIRECCIONAL

Supongamos el problema de resolver la integral:

$$(1 / \sqrt{(1-x^2)} - x^2 / \sqrt{(1-x^2)}) \, dx$$

Podemos resolver el problema o bien descomponerlo en dos integrales indefinidas

$$dx / \sqrt{(1-x^2)} \qquad y \qquad x^2 \, dx / \sqrt{(1-x^2)}$$

o bien aplicandole a la integral una regla de cálculo

que la transforme en la integral:

$$\int (1-x^2)\ dx$$

La figura 2.4 representa este proceso mediante lo que se ha dado en llamar un árbol Y/O. Las dos primeras ramas de la izquierda representan el proceso de descomposición y van unidas por un arco para indicar que ambos nodos sucesores han de ser resueltos y no sólo uno de ellos como sucede en los árboles ordinarios. La rama de la derecha representa el proceso de resolución alternativa, la aplicación de la regla de transformación de la integral.

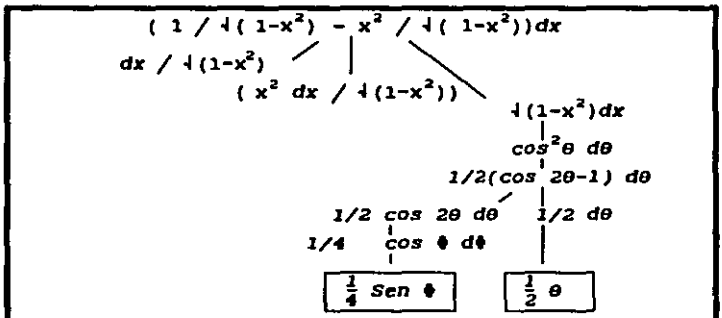


FIGURA 2.4 EJEMPLO DE UN ARBOL Y/O PARA LA RESOLUCION DE UNA INTEGRAL.

Al igual que en los gráficos ordinarios, los gráficos Y/O contienen nodos que representan la base de datos global del problema. Los nodos que representan una base de datos descomponible tienen por sucesores las bases de datos componentes. Estas bases de datos componentes hay que resolverlas todas, no son bases de datos alternativas.

En la figura 2.5 se muestra un ejemplo de gráfico Y/O

implícito. El nodo A es el nodo inicial, y los nodos H y G son los nodos terminales. La tarea del sistema de producción es hallar un subgrafico solución desde el nodo inicial a los nodos terminales. En la figura 2.6 se muestran dos subgráficos, soluciónes diferentes.

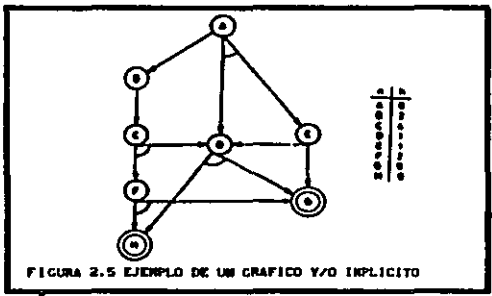


FIGURA 2.5 EJEMPLO DE UN GRAFICO Y/O IMPLICITO

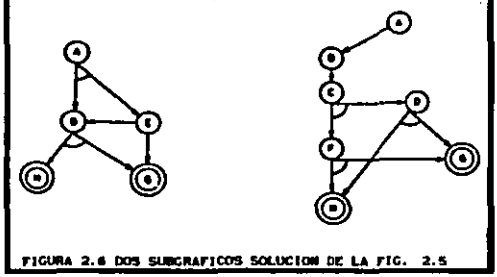


FIGURA 2.6 DOS SUBGRAFICOS SOLUCION DE LA FIG. 2.5

2.5.2 Estructuras de control de los procedimientos de la búsqueda

El ciclo de control de un procedimiento de búsqueda es la secuencia de pasos a realizar para generar un nuevo estado dentro de los estados o espacio de búsqueda. En

general, podemos distinguir cinco posibles componentes del ciclo de control:

1.- Determinar el conjunto de estados sobre los que es posible continuar la búsqueda o *estados activables*.
2.- Escoger un estado activable.
3.- Determinar el conjunto de reglas aplicables.
4.- Escoger una regla.
5.- Aplicar una regla escogida y memorizar el nuevo estado obtenido o estado actual.

Segun sea la forma de incorporar estos componentes en un ciclo particular de control, tendremos los diferentes tipos de estrategias de control que muestra la figura 2.7.

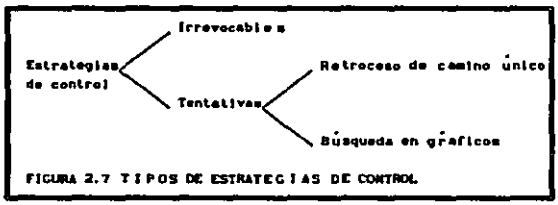


FIGURA 2.7 TIPOS DE ESTRATEGIAS DE CONTROL.

Distinguimos dos tipos principales de estrategias de control: las *irrevocables* y las *tentativas*. En las *estrategias irrevocables* sólo hay un estado activable que es el estado actual del problema; en esta estrategia no hay necesidad de los dos primeros componentes del ciclo de control; esto es sólo se conserva en memoria el estado actual.

En las *estrategias tentativas* intervienen los cinco puntos del ciclo de control, esto es, en cada ciclo consideramos la posibilidad de abandonar el estado actual -el último producido- para continuar la búsqueda por otro estado generado anteriormente, aplicándole una regla distinta. Se trata de dejar abierta la posibilidad de volver atrás y seguir por otro camino. Pero para poder volver atrás hemos de guardar en memoria los estados generados anteriormente. Si guardamos en memoria sólo una secuencia de estados, es decir, un camino, tendremos la estrategia de búsqueda por *retroceso de camino único*. Si mantenemos en memoria todo el grafico de estados generados, tendremos una estrategia de *búsqueda en gráficos*.

2.5.2.1 Estrategias de control irrevocables

La estrategia de control irrevocable se basa en el criterio de que tenemos la suficiente información o conocimiento local del problema para poder elegir la mejor regla aplicable al estado actual. Esta estrategia también es aplicable cuando sabemos que el hecho de aplicar una regla equivocada no impedirá la subsiguiente aplicación de una regla correcta, el único efecto negativo de las equivocaciones es el alargar la búsqueda (figuras 2.8 (a) y 2.8 (b)).

2.5.2.2 Estrategias retroceso en camino único

Dentro de las estrategias tentativas, las de retroceso en camino único son las más sencillas de implementar y las que requieren menos memoria, porque sólo guardan en memoria el camino de búsqueda en proceso de expansión.

Sin embargo, en cada ciclo se considera la posibilidad de volver atrás y escoger un estado de dicho camino distinto del actual. Dos son los motivos principales para provocar un retroceso:

1.- El estado actual es un estado de <<impasse>> -no tiene sucesores- y no podemos continuar la búsqueda por él.

2.- Cuando después de aplicar un cierto número de reglas sin haber encontrado aún la solución, nos hace dudar del interés para continuar con el estado actual.

En los casos en los que poseemos abundantes conocimientos sobre el problema -por ejemplo, en los sistemas expertos-, las estrategias tentativas por retroceso en camino único son adecuadas.

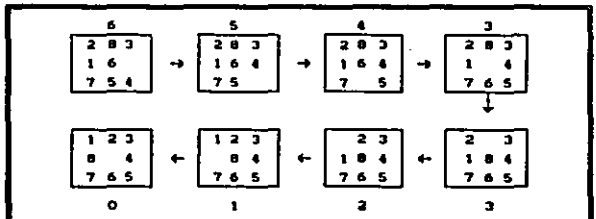


FIGURA 2.8 (a) EJEMPLO RESUELTO MEDIANTE ESTRATEGIA IRREVOCABLE

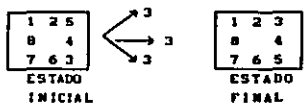


FIGURA 2.8 (b) EJEMPLO IRRESOLUBLE MEDIANTE ESTRATEGIA IRREVOCABLE

2.5.2.3 Estrategias de control en gráficos

En la estrategia por retroceso sólo mantenemos en memoria el camino en proceso de extensión; los estados que han provocado un fallo son olvidados, aun cuando es posible que más adelante de la búsqueda tengamos que volver sobre alguno de ellos. Una estrategia más flexible es aquella que conserva en memoria el conjunto de todos los estados generados y las transiciones entre ellos, cuyo conjunto constituye un grafico. Todos los nodos de este grafico son candidatos para continuar la búsqueda en cada ciclo. Las figuras 2.9 y 2.10 muestran partes de graficos generados por estrategias de búsqueda de gráficos.

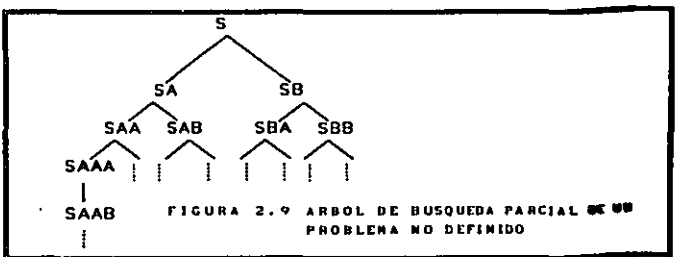


FIGURA 2.9 ARBOL DE BUSQUEDA PARCIAL DE UN PROBLEMA NO DEFINIDO

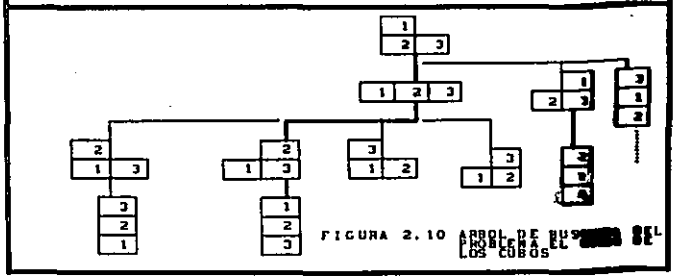


FIGURA 2.10 ARBOL DE BUS[illegible] DEL PROBLEMA EL[illegible] LOS COBOS

El proceso de generación de un gráfico de búsqueda puede ser definido esquemáticamente como sigue:

Procedimiento general de búsqueda de gráficos	
INICIALIZACION	1.- Crear un gráfico G con sólo el nodo inicial S. Colocar S en la lista llamada ABIERTA
	2.- Inicializar lista vacía, llamarla CERRADA.
TERMINAR FRACASO	3.- LOOP: SI ABIERTA= ø terminar con fracaso.
SELECCION	4.- Sacar primer nodo $\in$ ABIERTA y colocarlo CERRADA. Llamarlo *n*.
TERMINACION CON EXITO	5.- Si *n* es un nodo objetivo (terminal) terminar con éxito y dar solución siguiendo el camino desde *n* hasta S indicado por los punteros (se introducen en el paso 7).
EXPANSION	6.- Expansionar *n* generando el conjunto M de sus sucesores que no sean ya ramificaciones de *n*.
CONSTRUCCION Y ACTUALIZACION DEL GRAFICO	7.- Colocar un puntero hacia *n* para cada nodo de M que $\notin$ ABIERTA ni $\notin$ CERRADA y añadirlas a la lista ABIERTA. Para cada miembro de M $\in$ ABIERTA o $\in$ CERRADA decidir si hay que cambiar el puntero de sus sucesores.
	8.- Reordenar la lista ABIERTA (heurística).
	9.- Ir al LOOP (3).

Este procedimiento es suficientemente general para englobar una gran variedad de algoritmos. Su comprensión necesita de algunos comentarios. El procedimiento genera un gráfico G y un árbol T con los mismos nodos que G pero con la diferencia de que en T cada nodo tiene un único antecesor

marcado por un único puntero. G contiene todos los posibles caminos desde el nodo inicial S hasta cualquier nodo *n*. T en cambio sólo contiene, para cada *n*, el camino preferente desde S hasta *n*.

El árbol T está definido por los punteros generados en el punto 7. Los nodos de la lista ABIERTA son aquellos que aún no han sido expandidos (generación de todos sus sucesores por aplicación de reglas). Los nodos de la lista CERRADA son o bien nodos estériles o bien nodos ya expandidos. El procedimiento termina con fracaso cuando la lista abierta esta vacía, lo cual significa que no disponemos de más nodos para expandir y aún no se ha llegado al estado o nodo final.

Si en lugar de un gráfico tenemos un árbol, entonces los pasos 6 y 7 son más sencillos ya que ninguno de los sucesores de un nodo dado puede haber sido generado anteriormente, por tanto, no es necesario el cambio de punteros en el punto 7. En cambio, cuando tenemos efectivamente un grafico puede ser que generemos un mismo nodo por segunda vez. Comprobar esto en cada ciclo puede ser costoso, y por ello existen algoritmos que evitan comprobarlo. En consecuencia, el árbol de búsqueda puede contener nodos redundantes, lo cual puede concluir a calculos superfluos. En el procedimiento presentado no es asi. Si se genera un nodo repetido, es posible que el nuevo camino desde S a dicho nodo sea menos costoso que el ya establecido en el gráfico actual de búsqueda. Sin embargo, interesa tener todo momento el camino menos costoso desde S a un nodo dado. Por tanto, tendremos que modificar el árbol cambiando el padre o antecesor del nodo generado por segunda

vez, y si se cambia el antecesor de un nodo ya expandido habrá que reconsiderar también los antecesores de ese antecesor.

2.6 Cálculo de predicados

La lógica formal o el cálculo de proposiciones, como todos los esquemas de representación, es simplemente una forma de escribir expresiones sobre el mundo. El lenguage de la lógica se ha desarrollado desde hace miles de años y se sigue estudiando activamente. Usando la lógica, la gente puede hacer proposiciones sobre el mundo. Una proposición es un enunciado que tiene un valor de verdad, cierto o falso.

Los conectores se usan para combinar proposiciones y formar enunciados más largos y complejos. Los conectores más comunes son «*y*», «*o*», «*no*», y la frase «*Si...entonces...*». Los objetos individuales son identificados a través del uso de constantes, variables, o funciones. Ejemplos de constantes son «*gato*», «*Javier*», «*la bicicleta de Maria*». Variables tales como X, Y, A, se usan para designar objetos no nombrados todavía. Las funciones constan de los nombres de las funciones y los de los objetos asociados. Una función se aplica a un objeto o a varios, y da como resultado un objeto. Por ejemplo, «*actividad-preferida-de*» (BENITO, MONTAÑISMO) se puede utilizar para representar la relación entre BENITO y el MONTAÑISMO.

Podemos también hacer enunciados sobre las relaciones entre objetos usando predicados. Los predicados operan sobre objetos y distintas funciones (que simplemente retornan un

objeto cuando se les llama), el valor de un predicado es evaluado como cierto o falso. Una cadena de proposiciones o predicados se puede unir por medio de conectores lógicos para formar proposiciones más largas y complejas.

Las reglas de inferencia y las reglas para deducir nuevas proposiciones están ya dadas. Hay reglas para introducir y eliminar los conectores. Una regla simple de inferencia es el «*modus ponens*». Esta regla nos deduce una proposición «*Q*» si ya tenemos dos proposiciones: "Si P, entonces «*Q*» y «*P*»".

Cuando los predicados se usan como enunciados, suponemos que estamos declarando afirmacines ciertas. Podemos hacer estas afimacionescombinando constantes o variables con predicados. Cuando hacemos un enunciado en lógica formal usando una variable, decimos que la proposición es cierta para todos los objetos. Cuando hacemos una pregunta en lógica usando una variable, estamos preguntando por la existencia de un individuo o individuos que hacen el enunciado cierto. Al enunciar una regla de que todos los humanos son mortales, usamos una variable y decimos: «Si X es humano, entonces X es mortal». Para formular la pregunta «alguien es mortal», podríamos usar la forma: « X es mortal ».

Para resolver un problema con lógica proposicional, enlazaremos elementos del dominio del problema a los nombres de funciones, nombres de predicados, y símbolos de constantes usados en la proposición.

Podemos usar el cálculo de predicados de distintas maneras para resolver problemas de Inteligencia Artificial.

Para usar este método, se describira un dominio en lógica y un razonamiento sobre él.

Despues de escribir las proposiciones sobre el dominio, necesitamos mostrar las fórmulas que describen el resultado o la pregunta se siguen de estas proposiciones. El mejor método autómatico todavía hoy es la resolución. Esta técnica es básicamente una prueba por deducción al absurdo. Supone que el resultado es falso y muestra que esa condición conduce a una contradicción.

La resolución es comoda porque solo hay una regla de inferncia. El programa no tiene que ocuparse de la regla a escoger. Sin embargo, la resolución es lenta y las pruebas son difíciles de seguir por una persona. Los dominios heurísticos se incluyen a vecos para dirigir la prueba. Esta técnica es todavía un área de investigación de la Inteligencia Artificial.

2.7 Redes semánticas

Las redes semánticas se pueden ilustrar por diagramas que consisten en nodos y arcos. Los nodos presentan objetos, acciones, o sucesos. Los arcos de conexión, llamados enlaces, representan las relaciones entre los nodos. Un enlace podría significar que el objeto en un extremo es un atributo de un objeto en el otro extremo; o podría significar que uno implica al otro, o cualquier otra cosa que se defina en su lugar. Esta información es almacenada en la memoria del computador como estructura de valores-atributos.

En una representación a un único objeto se le puede aplicar más de una red. Estas redes diferentes vienen a significar los diferentes contextos en los que el objeto puede ser descrito. Más adelante se detallan en una forma más especifica las reglas y los objetivos de las redes semánticas.

2.8 Dependencias conceptuales

Las dependencias conceptuales son representaciones utilizadas para almacenar información en sistemas con lenguaje natural. Un mapa de dependencia conceptual de el significado de sentencias para dirigir gráficos compuestos de un pequeño número de conceptos básicos, o primitivos, significados de acciones, estados, o cambios de estados. Un sistema de lenguaje natural parafrasea los conceptos de los textos de entrada, poniéndolos de forma que el sistema los pueda usar en el diseño de inferencias, ejecute traslaciones, o responda preguntas con referencia a una base de datos. La paráfrasis es un lenguaje intermedio escrito en diagramas de nodos y flechas y acomodado a la manipulación del ordenador.

Las dependencias conceptuales equipadas con once *primitivas* han sido utilizadas por el profesor R. C. Schank. Schank usa las primitivas junto con otros pocos conceptos para generar las paráfrasis de sentencias. Las dependencias conceptuales se pueden usar para traducir textos de un idioma a otro, mediante paráfrasis de dependencia conceptual entre los dos lenguajes naturales.

El uso de primitivas permite a un sistema representar las similitudes y diferencias en sentencias cuyo significado profundo puede ser menos obvio cuando están en inglés. El análisis de primitivas también proporciona una base para hacer inferencias.

CAPITULO III

REPRESENTACION Y USO DEL CONOCIMIENTO

LOS ORDENADORES SON INUTILES.
SOLO PUEDEN DAR RESPUESTAS.

PABLO RUIZ PICASSO

3.1 ¿ Que es el conocimiento ?

Resolver un problema, comunicarnos con los demás, aprender un método, orientarnos en una ciudad, determinar el empleo del tiempo, son algunas de las actividades que normalmente realizamos a diario. Cada una de ellas requiere un proceso de razonamiento en el que entran en juego diferentes clases de conocimiento, unas veces de carácter técnico especifico y otras acerca del mundo en el que vivimos.

La Inteligencia Artificial tiene como objetivo construir modelos computacionales (programas) que al ejecutarse resuelvan tareas con resultados similares a los que obtendría una persona. Por ello, el tema central de esta disciplina es el estudio del conocimiento y su manejo, no sólo para realizar aquellas actividades reputadas como difíciles sino también las más básicas, que permiten a una persona cualquiera interaccionar con su entorno.

En un principio el interés de los investigadores se centró en el diseño de técnicas de representación y manipulación generales, aplicables a problemas en cualquier dominio. Problemas típicos que se abordaron fueron los puzzles, juegos (ajedrez, damas,...), etc. El enfoque seguido consistia en definir un posible espacio de soluciones y explotarlo mediante procesos de búsqueda, para encontrar un camino de coste mínimo.

La gran cantidad de conocimiento subyace en estos procesos puso de manifiesto la importancia de su estructuración y tratamiento especifico para delimitar en cada caso la información relevante, reduciendo drásticamente la búsqueda.

Las líneas actuales de investigación tratan de dar respuesta a las siguientes preguntas: qué formalismo es el más adecuado para describir diferentes tipos de conocimiento (sobre objetos, propiedades, acciones, relaciones casuales, suposiciones, creencias, metaconocimiento, etc.), cómo debe almacenarse para que pueda ser modificado y utilizada con eficacia de forma automática.

3.2 Representación del conocimiento

La representación del conocimiento constituye, el problema central en el proceso de construcción de un sistema experto (también se presenta en otras investigaciones, como la comprensión del lenguaje natural, la robótica o las bases de datos inteligentes). Al ser la representación del conocimiento un formalismo que sirve de soporte a los fenómenos estudiados, el poder de la inferencia que se puede poner en práctica se encuentra, estrechamente ligado a la elección de la representación. Inicialmente esta elección se presenta como una alternativa entre la presentación *procedural* y representación *declarativa*. Una representación procedural describe las relaciones entre los elementos de conocimiento utilizados, mientras que la representación declarativa permite una expresión del conocimiento en forma de *elementos* independientes y deja que un mecanismo de razonamiento se encargue de combinar estos elementos para hacer deducciones. No se detallará el uso de procedimientos para representar los conocimientos. Esquemáticamente si se centra la atención en la representación declarativa, sea pura como en las reglas de producción y las redes semánticas, o bien mixta como en el caso de los *Frames*, podemos decir que hay dos formas de abordar el problema:

+ La primera, a partir de la lógica de predicados, permite utilizar un conjunto de resultados matemáticos bien definidos. Actualmente está reconocido que la lógica de predicados es un medio eficaz para representar el conocimiento de naturaleza declarativa.

+ La segunda no hace referencia a la lógica (al menos en apariencia). Los problemas se plantean de formas diferentes. Los *Frames de MINSKY* constituyen un ejemplo.

Buchanan subraya tres criterios a seguir para la representación del conocimiento en un sistema experto: *la extensibilidad, la sencillez y el cáracter explícito del conocimiento.*

La extensibilidad: las estructuras de datos y los programas deben ser lo bastante flexibles para permitir extensiones de la base de conocimiento sin necesitar pesadas revisiones del programa.

La sencillez: la representación como la extensión del conocimiento debe ser simple para aquellos que no posean los suficientes conocimientos en el área.

Hay dos formas de mantener la sencillez conceptual: dar al conocimiento una forma tan *homogénea* como sea posible, o bien escribir funciones de acceso especiales para representaciones no homogéneas.

El conocimiento debe ser explícito: esto constituye un elemento importante para la búsqueda de errores y la expresión de las explicaciones ofrecidas por el sistema.

3.2.1 Sistemas de producción

Los sistemas de producción fueron propuestos por *POST* como un mecanismo general de cálculo. Numerosos sistemas expertos se basan en la utilización de reglas de deducción. El conocimiento del experto es representado mediante un gran número de reglas simples, utilizadas para dirigir el diálogo entre el sistema y el usuario a fin de deducir conclusiones. En la practica, este esquema de cálculo mediante reglas de producción se opone a los sistemas procedurales, si bien, desde un punto de vista formal, es estrictamente equivalente a ellos.

3.2.1.1 Descripción de un sistema de producción

Los sistemas de producción pueden esquematizarse mediante tres elementos básicos (figura 3.1):

- Un conjunto de reglas (base de conocimiento).
- Una base de datos (o de hechos).
- Un intérprete de reglas llamado motor de inferencias.

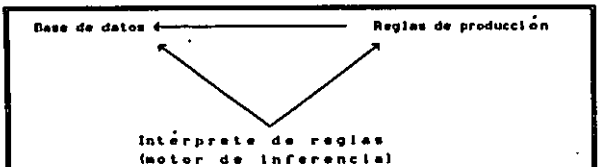


FIGURA 3.1 MODULOS DE UN SISTEMA DE PRODUCCION

Las reglas son relaciones dadas en forma de implicaciones. También pueden interpretarse como condiciones a cumplir para desencadenar una acción dada. Son la expresión de conocimiento general de forma:

Si < CONDICIONES > Entonces < ACCIONES >

La activación de las reglas constituye una cadena de acciones dirigida por *Modus Ponens*. La organización y el acceso a las reglas son importantes. El acceso puede variar desde un esquema muy simple, en el cual las reglas son utilizadas en un orden predeterminado, hasta esquemas más complejos que integran dispositivos que permiten la *resolución de conflictos*, es decir, procesos dinámicos de selección de la regla a activar. De forma general, una regla es evaluada por referencia al contenido de la base de datos.

La base de datos constituye la memoria a corto plazo del programa. Contiene los *hechos*, es decir, relaciones que no están expresadas en forma de implicaciones. Representan un conocimiento relevante del caso particular del individuo a tratar que ha podido ser suministrado al sistema, o bien deducido por éste.

El intérprete permite demostrar un objetivo a partir de los hechos de la base de datos y de la base de reglas de conocimiento. Esta demostración se hace mediante unificación limitada, también llamada << *pattern matching* >>. Este método de demostración automática permite seguir las diferentes operaciones y, en consecuencia, es de más facil comprensión para el usuario, como se muestra en el la figura 3.2.a. Esta figura pone en evidencia las tres fases que caracterizan el uso de las reglas de deducciones sucesivas.

Estas fases corresponden a la figura 3.2.b.

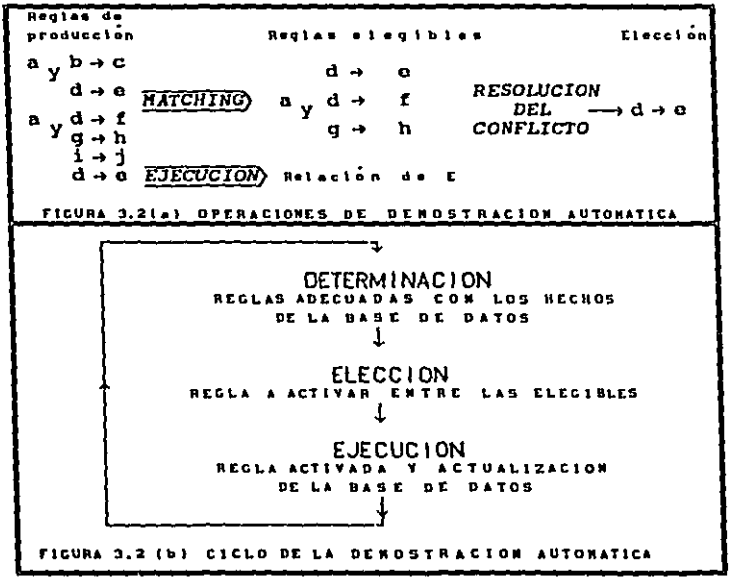


FIGURA 3.2(a) OPERACIONES DE DEMOSTRACION AUTOMATICA

FIGURA 3.2 (b) CICLO DE LA DEMOSTRACION AUTOMATICA

El enfoque de reglas de producción permite expresar conocimiento de manera puramente declarativa de forma modular. En Hay que añadir que los sistemas de producción son sensibles a los cambios de estado del sistema. Las acciones pueden modificar la base de datos, lo que afecta a la elección de la siguiente regla a activar.

Es interesante observar que los dos puntos de vista, que son modelización en psicología cognoscitiva y la elaboración de sistemas expertos con GPS, sistemas tales como MYCIN o DENDRAL, llegan ametodologías similares y

utilizan reglas de producción. Newell y Simon consideran estos sistemas como un buen medio para modernizar los procesos cognoscitivos, ya que tienen la generalidad de la máquina de *Turing*, permiten la introducción en desorden de nuevas reglas, y las reglas de producción pueden ser un modelo posible de la memoria humana a largo plazo.

3.2.1.2 Estructura de control en un sistema de reglas de producción

Es evidente que la misma expresión del conocimiento mediante reglas de producción implica ya un elemento de control. El sentido de aplicación de la regla es de importancia fundamental. Es decir que el razonamiento debe disminuir en lo posible el número de alternativas.

Hay otras informaciones de control que pueden integrarse en la expresión de la regla. Pero si se quiere disponer de un sistema inteligente que tome iniciativas variadas y plantee cuestiones pertinentes, debe de ser capaz de tener en cuenta todo cambio que se produzca en su entorno, es decir, en particular en la base de hechos. Esto obliga a que la estrategia de elección de la regla a activar, es decir la resolución de conflictos, permita la toma en consideración de estos cambios. El sistema dispone en este caso de una estructura de control elaborada. De forma general, las estructuras de control propuestas en los sistemas de producción pueden clasificarse según tres tipos:

* El uso exhaustivo de las reglas permite alcanzar un objetivo. En este caso, el procedimiento de control propiamente dicho es particularmente elemental, ya que no tiene en cuenta la

evolución de la base de hechos. Puede utilizarse en el caso en que el universo a estudiar sea pequeño, si las reglas de producción están ponderadas y se el resultado obtenido con cada una de estas reglas, separadamente, puede ser reforzado por la utilización conjunta de éstas.

* La activación de una regla según un criterio de elección. Los criterios de elección son múltiples y variados.

    + La elección de una regla recae en la regla utilizada con mayor frecuencia o en la utilizada más recientemente.

    + La elección de una regla puede basarse en los hechos contenidos en la base de datos. La prioridad, en este caso, será dada a la regla que se asocia con el hecho considerado más importante.

* El control se efectua por metarreglas: la expresión del conocimiento comporta su propio modo de utilización, la lógica del razonamiento se expresa mediante reglas parecidas a las otras. Por ello el, el mismo motor de inferencia se utiliza para razonar sobre el conocimiento (nivel-objeto) o sobre el control (meta-nivel). No siempre es posible razonar sobre el control; a menudo las informaciones están integradas en el código del intérprete, lo que las hace implícitas o inaccesibles; sin embargo, esta posibilidad debe investigarse por las ventajas sustanciales que presenta una representación

uniforme a todos los niveles. Permiten reordenar, mediante depuración, una lista de reglas que conducen a un objetivo. Más que por cualquier detalle sintáctico, son metareglas por el hecho de que expresan *<<conocimiento sobre el conocimiento>>*.

En conclusión, se presenta un esquema general de un sistema basado en reglas de producción en la figura 3.3 inspirado en esquemas de Frederik Hayes-Roth y Cols.

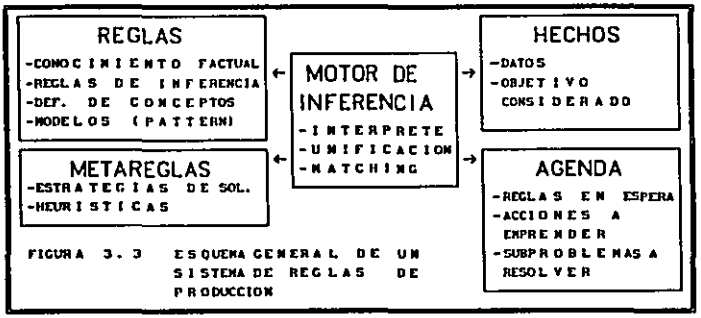


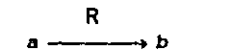
FIGURA 3.3 ESQUEMA GENERAL DE UN SISTEMA DE REGLAS DE PRODUCCION

3.3 Redes Semánticas

Davis y King han subrayado que las reglas de producción son un formalismo interesante para representar conocimiento mediante sentencias, pero que no es natural utilizarlo para representar conocimientos declarativos tales como las relaciones entre diferentes objetos de un dominio, por ejemplo, relaciones de pertenencia que permiten expresar una clasificación. Algunos investigadores han intentado

encontrar alternativas para representar el conocimiento. En los orígenes, encontramos los trabajos de Quillian sobre la búsqueda de un formalismo que permita representar la *memoria semántica*. El significado de las palabras se representa mediante un gráfico formado por nodos que identifican conceptos relacionados entre sí mediante enlaces de diferentes tipos.

Una red semántica es un gráfico cuyos nodos representan entidades, individuos, situaciones, y cuyos arcos orientados son instancias de las relaciones binarias (los nodos representan términos y los arcos representan predicados en lógica). La red representa el conjunto de relaciones binarias.



representa la relación R(a,b)

El interés de esta representación del conocimiento declarativo se basa en varios puntos:

- Las redes semánticas permiten codificar hechos o conceptos presentables en cualquier otro sistema formal. Además, introducen una disciplina de representación que obliga a expresar el conocimiento en forma de relaciones binarias antes que en forma de relación n-aria.

- Dan un esquema muy útil para almacenar información.

Cada individuo se representa mediante un simple nodo, y todas las relaciones referidas a este individuo son directamente accesibles a través de los arcos conectados con este nodo. Las estructuras de datos que codifican la información propiamente dicha pueden servir para guiar la búsqueda de informaciones.

3.4 Frames

Minsky partió de la idea de que tenemos estructuras estereotipadas de información en la memoria y de que, cada vez que se presenta una situación nueva seleccionamos uno de estos estereotipos intentando hacerlo coincidir con los datos de la situación. Minsky ha introducido la terminología de *Frames* para unificar cierto número de ideas sobre la representación del conocimiento. Los frames se han considerado como una alternativa a las redes semánticas y al cálculo de predicados que permite una representación mixta del conocimiento, es decir, que permiten combinar representaciones declarativas y procedurales, lo que ofrece una flexibilidad interesante.

Un frame es una estructura de datos, una expresión, dada para representar una situación tipo. Los frames se pueden considerar como un generalización de las redes en el caso en que los elementos, los objetos, son más complejos. Un frame contiene slots (huecos) que a su vez pueden ser frames o bien simples identificadores.

Los frames son esencialmente conjuntos de propiedades. El cuerpo de un frame representa entidades compuestas por atributos, los cuales tienen asociados descriptores que indican, por un lado, la forma de dar valor a cada atributo

y, por otro, las tareas a ejecutar cuando se ha evaluado un atributo.

El uso de los frames para representar el conocimiento exige establecer dos procesos:

* el primero, dirigido por el problema, permite asignar un valor a un atributo.

* el segundo, ascendente, está dirigido por los datos para ejecutar la tarea que ellos necesitan.

La asociación a los atributos de procedimientos activados después de la evaluación de su valor es una característica importante de los frames. Se pueden distinguir dos clases de procedimientos:

* Los procedimientos domésticos o siervos: activados sólo mediante solicitud.

* Los demonios: activados automáticamente cuando un dato se coloca en una instancia del frame.

Los frames que permiten describir y representar todos los aspectos de una situación dada están relacionados entre sí y frecuentemente forman un árbol de frames, a veces una red.

Además es posible dar a ciertos atributos valores por defecto en ausencia de información o, incluso, atribuir a un slot un valor heredado de un frame diferente.

En una representación del conocimiento mediante frames

pueden utilizarse varias reglas de inferencia:

- La primera regla de inferencia utilizada es la instanciación. Dado un frame que representa un concepto, se puede generar una instancia del concepto rellenando los slots del frame.

- La segunda regla fué sugerida por Minsky. Se utiliza explícitamente en ciertas aplicaciones de los frames: "*criteriality inference*". Si se han rellenado todos los slots de un esquema, esta regla nos permite inferir que existe una instancia del concepto expresado por el frame.

Las representaciones tipo frames ponen el énfasis en la estructuración del modelo conceptual sobre el dominio, por ello ofrecen mecanismos potentes de descripción de objetos; además, los algoritmos de razonamiento son de propósito específico, fundamentados en la organización de la base de conocimiento, lo que les permite seleccionar en dicha base las partes relevantes a los objetivos que se plantean en cada momento.

Dentro de los formalismos basados en frames, encontramos diversas alternativas (KRL, KL-ONE, KEE, REDES, UNITS, etc.), que difieren en cuanto a su sintaxis, claridad semántica y capacidad expresiva, pero casi todas ellas suscriben los siguientes principios:

1.- La representación ofrece un marco descriptivo estructural y se compone de un lenguaje para definir bases de conocimiento y un intérprete capaz de manejarlas para realizar procesos específicos de búsqueda e inferencia.

2.- La base de conocimiento está formada por redes de elementos (son los llamados frames, unidades, etc). Un elemento puede definirse como un objeto primitivo o por composición de otros elementos.

Cada elemento, a su vez, puede tener propiedades y relaciones. Se definen como un conjunto de descriptores y el valor de cada uno de ellos puede expresarse:

a) *De forma explícita:* mediante una declaración que especifique el valor o las clases de valores correspondiente, o bien asociando procedimientos que pueden activarse por diferntes mecanismos de invocación (nombre, valores activos, mensajes, etc) y que se ejecutarán para generarlos o modificarlos.

b) *De forma implícita:* mediante valores por defecto con mecanismos de herencia asociados a la taxonomía.

En resumen: un frame describe en forma estructurada una clase de objetos o una instancia perteneciente a una clase. Una base de conocimiento se constituye por un conjunto de frames organizados en jerarquías de orden parcial.

CAPITULO IV

CONCEPTO Y METODO DE CONSTRUCCION DE SISTEMAS EXPERTOS

LOS DEBILES TIEMBLAN ANTE LA OPINION, LOS TONTOS LA DESAFIAN, LOS SABIOS LA JUZGAN, LOS EXPERTOS LA DIRIGEN

MARIE J. ROLAND

4.1 Introducción

No creo que exista algún desacuerdo frente a la afirmación de que un experto es un complejo sistema cognoscitivo. Esté es el punto de partida para el presente planteamiento: un sistema experto de alguna manera iguala alguna función cognoscitiva de un experto humano.

Los Sistemas Expertos se pueden clasificar de acuerdo a la actividad cognoscitiva especifica para lo que se han creado. La siguiente lista pretende dar una clasificación tentativa de los Sistemas Expertos segun las actividades cognoscitivas para las que fueron creadas:

+ *Actividad clasificadora:* el objetivo de estos sistemas es clasificar el universo que nos rodea (objetos, eventos o situaciones) generalmente bajo condiciones de incertidumbre de información respecto a sus caracteristicas. Es sobresaliente notar que en estos sistemas el conocimiento proviene de muestreos; ejemplos de estos son los sitemas INTERNIST y MYCIN.

+ *Búsqueda en un espacio cerrado de soluciones:* Este tipo de sistemas buscan una solución y en ocasiones tienen que generar una solución satisfactoria. A este tipo de expertos pertenecen el sistema DENDRAL.

Búsqueda de procedimientos y planeación: La intención de este tipo de sistemas es de tratar de dividir los problemas en subproblemas resolubles inmediatamente. El sistema experto EL (solución de circuitos de transistores) y, MOGEN (planeación de experimentos de la genética molecular) pertenecen a esta clase.

Búsqueda en un espacio de soluciones abierto: Son sistemas cognoscitivos que descubren leyes o regularidades en un espacio cerrado de soluciones, esto es, el diseñador no prevee que puedan salir soluciones no contempladas y por lo tanto también ejercen actividades de descubrimiento inductivo.

4.2 Definiciones.

Los *Sistemas Expertos* constituyen la expresión informatica del conocimiento de las personas experimentadas en un tema. Esto es posible debido a una nueva arquitectura de sistemas de información cuyos elementos son:

1.- Una *base de conocimiento* en la que aparece formulada, en forma propcesable, la manera de entender el tema objeto del sistema.

2.- Un *programa motor inferencial*, capaz de producir las respuestas del sistema a partir de los datos de planteamiento de las mismas y de la base de conocimiento.

3.- Un *programa explicativo* que, a demanda del usuario, es capaz de explicar las razones que han conducido a cada respuesta posible.

Este tipo de estructura, al separar el conocimiento del proceso que lo utiliza para obtener resultados, ofrece las ventajas fundamentales de:

Generalidad, en dos aspectos:

a) Un mismo conocimiento puede utilizarse para obtener respuestas a distintos problemas.

b) La estructura del conocimiento separada permite su formación e interpretación con lógicas no clasicas.

Posibilidad de aprendizaje y, por tanto, programación automática: la estructura de conocimiento separada puede permitir la definición de procesos de obtención de unidades de conocimiento, en función de una valorización de los resultados producidos por el sistema y los observados realmente.

4.3 Génesis historica del concepto.

Los Sistemas Expertos nacen de la convergencia de dos líneas clásicas en el desarrollo de la Inteligencia Artificial:

1.- La línea heurística, basadas en la acumulación de una serie de restriccines limitadas en la búsqueda en estados o espacios de problemas.

2.- La *línea de deducción automática*, basadda en la mecanización de las teorias de interpretación de fórmulas lógicas.

La investigación en 1960 identificó y exploró varios propositos generales; tecnicas de resolución de problemas. Este trabajo introdujo y refinó el concepto de búsqueda heuristica como un importante modelo de resolución de problemas. A mediados de la decada de los 60's algunos

investigadores en la Universidad de Stanford desarroyaron el primer sistema experto llamado DENDRAL, y el proyecto de MACSYMA en el M.I.T.; el análisis de química organica en el caso de DENDRAL, y la integración simbólica y simplificación de formulas para MACSYMA.

Los sistemas fueron diseñados para manipular y explorar simbolicamente problemas expresados que fueron conocidos como dificiles para la resolución por humanos. Los problemas fueron caracterizados por su alto número de soluciones, posibilidades que tuvieron que ser examinadas conforme a los requerimientos del problema, estas especificaciones crecian en complejidad -cuanto más grandes eran las especificaciones del problema y más dificil era saber si todas sus soluciones eran validas. Está explosión combinatoria en el espacio de búsqueda para la solución muy seguido destruia los animos y las habilidades de los investigadores.

En la época de 1968-1972 fué cuando los Sistemas Expertos eran el tema preferido de los investigadores y por lo mismo se dedicaron de lleno al diseño de estos. La llamada *ingenieria del conocimiento* fue acumulando técnicas y herramientas para la representación, recolección y empleo de conocimiento experto en un campo dado.

Hacia 1976, D. B. Lenat escribe **AM**, un programa típico de aprendizaje que define y evalua conceptos matemáticos con teoría de conjuntos y números. En este mismo año se publica una tesis doctoral sobre el sistema TERESIAS que usa metaniveles de conocimiento para entrar y actualizar bases de conocimiento usadas en Sistemas Expertos.

Un año despues (1977), se completa la primera de una serie de aplicaciones de un sistema experto usando el

lenguaje PROLOG; en el año de 1978 se publica un trabajo sobre un sistema experto, PROSPECTOR, que ayuda al análisis de la información relacionada con exploraciones geológicas.

Posteriormente, a la vista del éxito de estos primeros sistemas, se ha producido una difusión generalizada de su uso, de manera en 1986 se conocían aproximadamente 186 Sistemas Expertos en funcionamiento, aunque debido a la lentitud en la incorporación del conocimiento todavía no rebasen de veinte el número de ellos en utilización efectiva por los usuarios finales.

4.4 El problema de la ingeniería del conocimiento.

La aparición de una nueva arquitectura de sistema informatico organizada en base a una estructura representativa del conocimiento y procedimiento de interpretación como semántica de dicha estructura, ha producido un cambio cualitativo en el diseño informático de datos y procedimientos para cálculo de funciones a partir de ellos.

Para construir este tipo de sistemas es preciso constar con:

1.- *Paradigmas de representación del conocimiento* que sean capaces de incorporar en su estructura un grado de complejidad suficiente para el tipo de problemas a resolver por el sistema.

2.- *Procedimientos de interpretación*, estos deven de poder obtener las respuestas según modelos de razonamiento.

3.- *Métodos efectivos de adquisición del conocimiento.*

Para hacer posible la ingeniería del conocimiento, es preciso seleccionar los paradigmas no sólo basados en sus valores teóricos de representación, sino también en criterios de:

Constructividad: Se deven incorporar conceptos inteligibles y correlativos de la forma de entrada de los expertos en el tema y al mismo tiempo por incorporación progresiva de conocimiento sobre distintas versiones sucesivas.

Operatividad: Deben ser evaluables por un procedimiento efectivo y con el máximo de contenido teórico para resolver la clase de problemas que es objeto de representación en el paradigma.

El paradigma sobre el que se basan la mayoría de las aplicaciones actuales, por haberse desarrollado para él tanto técnicas de formulación como metodologías de construcción, es el basado en reglas que relacionan ternas concepto (contexto u objeto)- atributo valor cuyas caracteristticas generales se tratan en el siguiente apartado.

4.4.1 ¿ Porque se va a usar un sistema experto ?

Un problema se presta para ser considerado como un sistema experto cuando:

1.- Una solución del problema tiene una rentabilidad tan alta que justifica el desarrollo de un sistema, pues las soluciones son necesidades del área y no se ha trabajado en otros métodos para obtenerla.

2.- El problema puede resolverse sólo por conocimiento experto que puede dar forma a los conocimientos necesarios para resolver el problema.

3.- El problema no puede resolverse por algoritmos particulares sino que necesita conocimiento experto para poder llegar a la solución.

4.- Se tiene acceso a un experto que puede dar forma a los conocimientos necesarios para resolver el problema. La intervención de este experto dará al sistema la experiencia que se necesita.

5.- El problema puede no tener solución única. Los Sistemas Expertos funcionan mejor con problemas que tienen un cierto número de soluciones aceptables.

6.- El problema cambia rápidamente, o bien el conocimiento es el que cambia rápidamente, o sus soluciones son las que cambian constantemente.

El desarrollo de un sistema experto no se considera que está acabado una vez que este funciona, sino que se continúan desarrollando y actualizando tanto el conocimiento del sistema como los métodos de procesamiento, quedando

reflejados los procesos o modificaciones en el campo, área o sistema.

4.5 El paradigma Concepto - Atributo - Valor.

Este paradigma, al que pueden referirse la mayoria de los Sistemas Expertos implementados actualmente, tiene su origen en el planteamiento del sistema MYCIN. Un paradigma deve valorarse en función de su capacidad para describir el universo de discusión del sistema, la posibilidad de construcción de procesos de razonamiento para resolver problemas, y las posibilidades construidas del conjunto.

El marco conceptual se define por un conjunto de objetos cada uno de los cuales tiene una serie de atributos a los que pueden asignarse valores de un conjunto prefijado. Por ejemplo:

+ Objeto → MESA
+ Atributos → FORMA, TAMAÑO, ESTILO, NUMERO DE PATAS
+ Valores → - FORMA → RECTANGULAR, CIRCULAR, ETC.
  - TAMAÑO → GRANDE, MEDIANA, PEQUEÑA
  - NUMERO DE PATAS → 3, 4, 6
  - ESTILO → CLASICO, MODERNISTA, GOTICO, ETC.

El esquema de razonamiento se plantea basándose en un conjunto de conceptos y realidades como objetos básicos. De ellos, algunos son objetos solución y otros son objetos dato. El objetivo del proceso de razonamiento a modelar es la definición de los valores de los atributos de los objeto dato. Cabe la posibilidad también de definir en un objeto atributos dato y atributo solución.

Las bases de conocimiento en este tipo de paradigma se

formulan como reglas de relación entre ternas objeto-atributo-valor del tipo:

$$O_i.A_j.V_k,\ O_m.A_h.V_r,\ldots \xrightarrow{q} O_p.A_k.V_s$$

que permiten definir unos valores de atributos a partir de otros con grado de certeza q.

El tipo de problemas que se resuelve con este paradigma es el de clasificación generalizada, es decir, son sistemas que no generan soluciones nuevas; para ello, el esquema general de razonamiento tiene una estructura del tipo siguiente:

a) Proceso inicial de abstracción de los conceptos descriptivos de los datos: es decir, son aquellas reglas que infieren valores de atributos más generales de clasificación del objeto o relación a partir de los valores de los atributos de detalle.

b) Aplicación de reglas heurísticas de solución: aquellas que relacionan valores de atributos de clasificación de objetos datos con valores de atributos de clasificación de objetos solución.

c) Refinamiento: aquellas reglas que relacionan valores de atributo de detalle en objetos solución con valores de atributo-clase más generales en los mismos objetos.

Como consecuencia el razonamiento sobre objetos dato es ascendente hacia conceptos más generales que permitan la inferencia de la solución y en los objetos solución el

proceso es descendente hacia las clasificaciones generadas previamente.

Las reglas de abstracción de datos así como las de refinamiento son reglas que se basan en criterios definidos de clasificación y normalmente tienen grados de implicación total excepto cuando la clasificación se haga por criterios difusos. En las reglas heurísticas que presentan formas de resolver el problema por vía de síntesis de teorías más complejas, las reglas de este tipo se formulan con nivel de implicación parcial.

La estructura general de los procesos de razonamiento de sistemas clasificativos se representa en la figura 4.1.

En el momento actual, se es consciente de que este paradigma es criticable por una serie de problemas teóricos (exceso de simplicidad, dificultad de evaluar la interacción o independencia de antecedentes de distintas reglas, dificultades de formulación del razonamiento aproximado, otras).

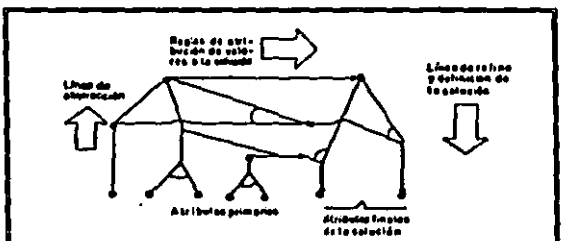


FIGURA 4.1 ESQUEMA DE RAZONAMIENTO EN SISTEMAS CLASIFICATIVOS

La estructura general de estas metodologías puede resumirse en los pasos siguientes:

1.- *Definición,* basandose en sonsultas iniciales con personas expertas en el tema, para realizar las distintas afirmaciones de la base de conocimientos.

2.- *Obtención de datos* en donde aparezcan casos resueltos manejando los conocimientos con los que se quieren representar el paradigma.

3.- *Modelización de los procesos de análisis de los distintos documentos* o del proceso de evolución de estados en el tiempo de estos distintos sistemas.

4.- *Formulación de casos* de una primera basse de reglas por abstracción de los modelos de razonamiento.

5.- *Proceso de crítica y modificación* basándose en la simulación de los distintos casos con el sistema de reglas formuladas y valoración de sus respuestas, basándose en la documentación existente y el criterio de las personas expertas.

4.5.1 Motores de inferencia.

A un Sistema Experto se le exige capacidad para extraer conclusiones apartir de unos hechos y/o reglas contenidas en su base de conocimientos y de la información que el ususario

le proporciona; es decir deve ser capaz de efectuar inferencias lógicas.

El control de propagación de las premisas hasta la conclusión es lo que se conoce como motor de inferencia. La máquina inferencial se encarga de efectuar el razonamiento, controla las estrategias del sistema y encadena la búsqueda a la solución del problema. Indicando con las preguntas del usuario, manipula la base de conocimientos y procede sin ambigüedades a la solución, mediante alguno de los métodos y estrategias de Inteligencia Artificial; siendo las más comunes encadenamiento hacia adelante y encadenamiento hacia atras (figura 4.2).

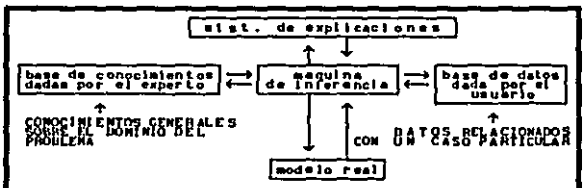


FIGURA 4.2 FUNCIONAMIENTO A BLOQUES DE UN SISTEMA EXPERTO

Quiza uno de los postulados principales es el de la separabilidad de el conocimiento como tal, del mecanismo de inferencia que lo usa. Este postulado ha tenido muy buenos frutos porque ha permitido la búsqueda de representaciones del conocimiento, olvidandose momentáneamente del problema de su manejo y dando con ello origen a extensas discusiones sobre las formas del como podría representarse el conocimiento para una disciplina determinada.

El motor de inferencia produce respuestas (con grado de certeza asociado) a los problemas planteados al sistema. Para ello, aplica un proceso de búsqueda y acumulación de certeza apoyado en la base de conocimiento.

En el paradigma que se estudia, la memoria de trabajo se describe mediante conjunción de valores de atributos de los distintos conceptos (objetos y relaciones). El proceso de inferencia trata de definir los grados de certeza de los valores de los atributos premisa. Para ello, cabe mencionar tres de las estrategias de control:

a) *Estrategia con encadenamiento hacia adelante* (control guiado por los datos).
b) *Estrategia con encadenamiento hacia atrás* (control guiado por los objetivos).
c) *Estrategias mixtas de ambos tipos.*

En la primera se parte de un conjunto inicial de valores en la memoria de trabajo y se obtienen todos los valores deducibles de ellos por aplicación de las reglas de manera similar a la indicada al tratar de los sistemas de producción.

La estrategia de razonamiento hacia adelante resulta adecuada para aquellos tipos de problemas en que no puede delimitarse con mucha precisión la respuesta a obtener.

También resulta adecuada esta estrategia en los sistemas de control en los que deben analizarse los grados de certeza de un conjunto importante de incidencias y actuar en consecuencia dentro de la gama importante de acciones de alarma y recomendaciones de operación.

Sin embargo cuando se trata de sistemas de consulta en que los atributos para definir la respuesta son en número bastante limitado, puede resultar excesivamente costoso el proceso de hacia adelante; en este caso resulta más versatil utilizar la técnica de encadenamiento hacia atrás.

Este proceso puede ordenar las reglas en orden de mayor a menor nivel de implicación; ademas limita la selección de reglas pertinentes a un objetivo de un nivel de implicación umbral.

El proceso adelante-atrás ordena las reglas a efectos de análisis, no sólo en razon de su nivel de implicación sino en orden a la pertinencia de su consecuente respecto de los objetivos finales (para ello devería prescindirse de la formulación recursiva y operar con una lista de objetivos intermedios).

La estructura del procedimiento permite que el sistema formule preguntas al usuario sobre aquellos objetivos intermedios de los que dispone reglas. Esta es una forma muy parecida de plantear cuestiones a la forma humana.

Las características del motor de inferencia dependen fuertemente del método elegido para representar la imprecisión, lo que se hace, basicamente, siguiendo dos caminos diferentes. Por una parte se considera el *modelo probabilistico* y por la otra los basados en el *modelo posibilistico de Zadeh.*

4.5.1.1 El modelo Probabilístico: El motor de inferencia de PROSPECTOR.

La base del sistema de razonamiento aproximado utilizado en el Sistema Experto PROSPECTOR es un modelo probabilístico bayesiano[1], con algunas modificaciones. En esencia, la incertidumbre de los hechos y reglas que conforman la base de conocimientos del sistema tratada por medio de unas ciertas probabilidades «*subjetivas*» asociadas a los hechos (probabilidades apriori), y de unos grados de necesidad y suficiencia con que el antecedente implica el consecuente, asociados a las reglas.

Los antecedentes de las reglas pueden ser conjunciones, disyunciones o negaciones de varias premisas; las más usadas son:

$$V\ (A\ y\ B) = Mín\ (V(A),\ V(B))$$
$$V\ (A\ o\ B) = Máx\ (V(A),\ V(B))$$
$$V\ (no\ A) = K - V(A)$$

donde V representa el grado de certeza y K es igual al máximo del intervalo más el mínimo. Indicando estas en terminos de probabilidad tenemos:

$$P\ (A \cap B) = Mín\ (P(A),\ P(B))$$
$$P\ (A \cup B) = Máx\ (P(A),\ P(B))$$
$$P\ (\neg A) = 1 - P(A)$$

A cada regla del tipo « Si.. E entonces.. H » se le asignan dos parametros (LS, LN) definidos respectivamente por:

---

1 Modelo probabilistico bayesiano: Es una forma de reconocimiento de patrones. Busca aquella información que vulnera la hipotesis vigente, para excluir el uso de reglas frente a ciertos hechos que las hacen inoperativas.

$$LS = \frac{P(E/H)}{P(E/\overline{H})} \qquad y \qquad LN = \frac{P(\overline{E}/H)}{P(E/\overline{H})}$$

Donde:

LS : Grado de suficiencia de la regla; un valor elevado de E aumenta la probabilidad de H.

LN : Grado de necesidad de la regla.
Cuando LN $\longrightarrow$ 0; E $\longrightarrow$ 0 y P(H) baja.

Una de las características más destacables de este motor reside en la utilización que hace de la regla de Bayes para actualizar la probabilidad apriori de H a partir de la observación de E. Para ello si O(H) y O(H/E) son las cantidades definidas respectivamente por:

$$O(H) = \frac{P(H)}{P(\overline{H})} \qquad y \qquad O(H/E) = \frac{P(H/E)}{P(\overline{H}/E)} \quad ,$$

entonces por bayes

$$O(H/E) = LS * O(H)$$

lo que permite actualizar la probabilidad de H. Análogamente, si se constata de que E es falso, entonces resulta que

$$O(H/E) = LN * O(H).$$

Lo que a su vez permite actualizar también la probabilidad de H.

Suponiendo ahora que el usuario, en lugar de observar E, observa un cierto E'(E'≠E y $E' \neq \overline{E}$) al cual pude asociar un cierto grado de relevancia P(E/E'). Se trata de actualizar P(H) a partir de esta observación, es decir, se trata de calcular P(H/E). Para ello se asume que se E es cierto (o falso) entonces la observación E' relevantes a E no proporcionan información adicional acerca de H, es decir,

$P(H/E,E') = P(H/E)$ y $P(H/E,E') = P(H/E)$, entonces se obtiene la expresión:

$$P(H/E') = P(H/E)\ P(E/E') + P(H/\bar{E})\ P(E/\bar{E}')$$

que no es más que una interpolación lineal entre $P(H/E)$ y $P(H/\bar{E})$.

En el marco teórico, la ecuación anterior proporciona la resolución al problema de la actualización de H dado un conocimiento incierto acerca de la verdad o falsedad de E.

Por cuanto se refiere a la actualización de las probabilidades en el caso de que existan diversas reglas del tipo $E_1 \rightarrow H$; $E_2 \rightarrow H$;....; $E_n \rightarrow H$, cada una de ellas con sus parámetros asociados (LS_1, LN_1), el procedimiento de cálculo es el siguiente:

a) En el caso de que no haya problemas de inconsistencia y que todos los E_i sean condicionalmente independientes es decir,

$P(E_1,....,E_n/H) = \prod_{i=1}^{n} P(E_i/H)$, de la constatación de la recurrencia de todos los E_i, se obtiene facilmente que

$$O(H/E_1,....,E_n) = \left(\prod_{i=1}^{n} LS_i \right) \cdot O(H).$$

b) De manera similar, si todos los E_i son falsos, entonces

$$O(H/E_1,....,E_n) = \left(\prod_{i=1}^{n} LN_i \right) \cdot O(H)$$

c) En el caso general de evidencias inciertas y de inconsistencia, habiendo observado E'_i y calculando $O(H/E'_i)$ -haciendo, si es necesario, las modificaciones oportunas para soslayar los problemas de inconsistencia se define la relación de verosimilitud efectiva como

$$LS'_i = \frac{O(H/E'_i)}{O(H)}$$

y, finalmente, aceptando que los E'_i sean independientes, se obtiene una expresión similar a las dos anteriores:

$$O(H/E'_1,.....,E'_n) = \left(\prod_{i=1}^{n} LS'_i \right) o\ O(H).$$

4.5.1.2 El modelo evidencial. El motor de inferencia de MYCIN.

La teoría de la evidencia proporciona al sustrato teórico al llamado Razonamiento Evidencial *(Evidential Reasoning)* que ya ha sido parcialmente utilizado en algunos sistemas como MYCIN. La teoría de la Evidencia difiere de la teoría de la Probabilidad al no asumir que todos los resultados posibles de una experiencia puedan ser observados en una forma precisa.

Shortliffe y Buchanan en su sistema MYCIN, experto en enfermedades de origen infeccioso propusieron el siguiente esquema.

Para valorar el grado de incertidumbre (o de certeza) asocia unas probabilidades subjetivas sobre los hechos y unas probabilidades condicionadas sobre las reglas, es

decir, a « Si E entonces H » se le asocia P(E/H). La asociación de dichas probabilidades es uno de los problemas de este tipo de sistemas ya que cuando no tiene información sobre la certeza o falsedad de los hechos o reglas lleva a estados paradógicos, además de introducir problemas de inconsistencia.

A partir de estas probabilidades define en cada regla el llamado grado de credibilidad de la hipótesis H dada la evidencia E.

4.5.1.3 El motor de inferencia del M1.

El entorno de creación de Sistemas Expertos llamados M1, utilizable sobre todo en computadoras personales, usa un motor de inferencia algo especial en el que se observan características de los dos modelos anteriormente señalados.

En este sistema, el grado de certeza viene dado por un factor de certeza, FC, tomado en el intervalo [0,100]. Cabe resaltar que en el M1 se trabaja a partir de un nivel mínimo de certeza, correspondiente a un FC = 20, de forma que si se llega a la conclusión que con un FC < 20 no se toma en cuenta para posteriores inferencias.

Los factores de certeza en el M1 pueden ser introducidos de tres formas:

1.- El usuario contesta una pregunta calificandola con un FC (si no se especifica se supone FC = 100).

2.- Un hecho de la base de conocimientos lleve asociado un FC.

3.- La conclusión de una regla contenga un FC.

El sistema básico de inferencia que utiliza el M1 es el siguiente:

Dado « X es A entonces Y es B » con FC = α, y « X es A » con FC = β se concluye « Y es B » con FC = $\alpha\beta/100$.

4.5.1.4 El modelo Posibilistico.

Las variables lingüísticas, las distribuciones de posibilidad y la lógica difusa son las herramientas básicas de este modelo. Con él se trata de representar el hecho que la mayor parte del razonamiento humano es más aproximado que exacto y que los valores que toman muchas de las variables usadas en el discurso humano son, usualmente, palabras y no números. Visto desde esta perspectiva, el objetivo que con él se persigue es el proporcionar una base sistemática para modelizar el razonamiento aproximado, es decir, extraer conclusiones a partir de datos representados mediante proposiciones expresadas en lenguaje natural o sintético.

4.5.2 Técnicas de aprendizaje.

El problema del aprendizaje puede formularse de la manera siguiente:
Dada una muestra de ejemplos, definido cada uno por:

a) Una conjunción de ternas Objeto-Atributo-Valor (tanto de objetos causa como de objetos efecto intermedio o final).

b) Un peso indicativo de medida del número de veces que se ha presentado en la muestra,

obtener un conjunto de reglas que describan la muestra con un grado aceptable de precisión (es decir, cada regla se comprueba por un número suficiente de casos).

Los tipos de reglas a obtener pueden ser dos:

1.- Reglas que relacionan una terna apartir de otras como premisas.

2.- Reglas que relacionan una función lógica de distintas ternas a partir de otras ternas como premisas.

El primer tipo de reglas es directamente utilizable para inferencia por un sistema dentro de este paradigma; son realmente reglas que definen un concepto de clasificación preestablecido.

El segundo tipo de reglas es un vehículo para descubrir nuevos posibles conceptos de clasificación no definidos previamente. Una vez definido e introducido este concepto, se obtendran las reglas utilizables por el motor de inferencia.

4.6 Entornos específicos para la construcción de Sistemas Expertos.

La demanda industrial de productos inteligentes ha puesto de manifiesto el importante papel, a veces crítico, que juegan las herramientas en la construcción de un Sistema Experto. Como consecuencia fueron apareciendo herramientas y prestaciones disparejas en costos y con dudosas garantías de

funcionamiento. Sin embargo, los problemas fundamentales comienzan apenas a ser atacados. Esto es la culminación de metodologías que permitan construir un Sistema Experto con rigor, robustez, rapidez y eficiencia; y el desarrollo de nuevas herramientas que faciliten la implementación de los principios metodológicos de forma sencilla, segura y flexible.

4.6.1 Herramientas de ayuda a la construcción de Sistemas Expertos.

La construcción de un Sistema Experto requiere de una serie de fases (figura 4.3). Cada una de ellas tiene que cumplir unos objetivos, una metodología y unas herramientas especificas, que se analizaran a lo largo del presente apartado.

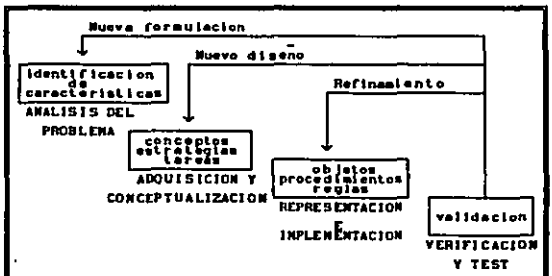


FIGURA 4.3 CICLO DE CONSTRUCCION DE UN SISTEMA EXPERTO

4.6.1.1 Análisis del problema.

De acuerdo con los requerimientos del usuario, los recursos disponibles y las características del problema, se fijan los objetivos generales y las tareas específicas que deben ser realizadas por el Sistema Experto. El ingeniero debe de evaluar cada tarea y proponer los recursos necesarios.

Las herramientas de ayuda a esta fase de construcción se pueden clasificar en dos niveles:

1.- *Ayuda a la escritura del proyecto.* Son sistemas formados por editores de textos y gráficos orientados a facilitar la expresión de los distintos elementos que intervienen en la organización de un proyecto.

2.- *Ayuda a la organización del proyecto.* Herramientas que ayudaran al ingeniero a utilizar a utilizar el conocimiento acerca de los recursos, tareas a realizar, restricciones específicas de cada tarea (tiempo de ejecución, prioridades, etc.) esto es asignación óptima de los recursos.

El primer tipo de herramientas existe en el mercado pero aún no se integraron a un entorno de desarrollo de Sistemas Expertos; las segundas son las que más ayuda pueden prestar al ingeniero que desarrolla el sistema.

4.6.1.2 Adquisición del conocimiento y conceptualización.

El objetivo consiste en identificar todos los elementos que intervienen en la solución del problema (conceptos, procedimientos, reglas de inferencia, heurísticos, casos especiales, métodos de razonamiento, restricciones, etc.). La información puede obtenerse de distintas maneras o diversas fuentes:

+ Diálogo directo con expertos en el tema del problema.

+ Fuentes bibliograficas .

+ Datos empíricos (por observaciones o aparatos de medición)

+ Datos gráficos (imagenes, dibujos, diagramas), etc.

A medida que se va obtenido el conocimiento es necesario depurarlo, seleccionando aquellos conceptos básicos que haran posible el funcionamiento del sistema.

La realización de esta fase implica llevar a cabo diferentes procesos:

1.- Análisis del conjunto de informaciones que proporcionan las fuentes de conocimiento.

2.- Abstracción para obtención de conceptos.

3.- Selección y clasificación de la información.

4.- Inferencia de nuevos conceptos, relaciones o propiedades a partir de las ya existentes.

5.- Generación a partir de casos concretos.

6.- Inducción, etc.

La adquisición del conocimiento requiere una buena metodología; ya que esta fase es la más compleja y donde se necesita de una intensa ayuda. La filosofía de este tipo de herramientas es el de ayudar a la labor del ingeniero; se trata de Sistemas Expertos en la construcción de Sistemas Expertos; la implementación de estas herramientas requiere de cooperación de distintas áreas de inteligencia artificial: lenguaje natural, tratamiento de imagenes, aprendizaje, planificación, etc.

4.6.1.3 Representación del conocimiento e implementación.

La actividad del consultor del Sistema Experto en esta fase se concreta en las siguientes tareas:

1.- Elección de un formalismo de representación del conocimiento.

2.- Elección de una arquitectura que permita coordinar y manejar los distintos tipos de elementos que intervienen en la solución del problema.

3.- Creación de una base de conocimiento utilizando el formalismo y la arquitectura elegida.

4.- Diseño de la interfaz del Sistema Experto con el usuario y con el resto del entorno lógico.

4.6.1.3.1 Lenguajes para la representación del conocimiento.

Existen tres grupos de lenguajes distinguidos de las siguientes maneras:

+ Lenguajes orientados a objeto.
+ Lenguajes basados en lógica.
+ Lenguajes de tipo funcional.

Desde el punto de vista del ingeniero es importante determinar la capacidad expresiva del lenguaje en dos aspectos:

1.- La adaptación del formalismo a la definición de los objetos que forman la base de conocimiento.

2.- La manera en que pueden expresarse los procesos de razonamiento y control sobre la solución del problema.

a) Definición de los objetos

Los objetos o entidades conceptuales pueden definirse de diferentes formas:

1.- Definición estática: Expresando las caracteristicas individuales de los objetos (estructura, elementos, partes, etc.).

2.- Definición relacional en función de su dependencia, conexión jerarquica, etc. con el resto de los objetos.

3.- Definición de las propiedades que debe verificar el objeto.

Un formalismo ideal debe permitir al ingeniero definir los objetos de la forma más natural y flexible; esto es que se deja la libertad para utilizar la ddescripción idónea a cada objeto.

b) Definición y control de los procesos de resolución de problemas.

Los mecanismos de razonamiento que el ingeniero debe manejar implican una amplia gama de procesos:

+ Deducción
+ Inducción
+ Búsquedas selectivas de información
+ Reducción de problemas a subproblemas
+ Planificación
+ Analogías, otras.

En la mayoria de los entornos existen mecanismos predefinidos en relación al interprete del lenguaje de representación del conocimiento.

Las herramientas necesarias para eliminar este problema deben proporcionar al ingeniero facilidades para definir y manejar como objetos de la base de conocimiento, los elementos que intervienen en cada proceso (tareas,

objetivos, planes); cada uno de ellos puede tener una semántica predefinida que permita:

1.- Expresar de forma declarativa el control de los procesos de inferencia (limitación del espacio de búsqueda, metarreglas, definición de prioridades, etc.).

2.- Organizar o utilizar una arquitectura especifica basada en:
+ Agenda de tareas
+ Procesos cooperativos

3.- Definir razonamiento incierto o aproximado

Como complemento se necesitan funciones especificas que realicen:

+ Selección y generación de tareas.
+ Tratamiento en paralelo de distintas tareas.
+ Análisis y tratamiento de casos de error en la solución de las tareas.
+ Integración de distintos tipos de conocimiento (simbólico, numerico, gráfico).

4.6.1.3.2 Editores para la actualización y manejo de la información.

Estos pueden ser de tres tipos:

1.- Editores estructurales (sintácticos): Definición de los objetos. Verifica que en el momento de su creación la corrección sintáctica, ayuda en la estructura del lenguaje.

2.- Editores gráficos: Facilitan la visualización de la estructura y relaciones de los objetos de la base de conocimientos. Se pueden usar para crear, añadir o modificar la base de conocimientos.

3.- Editores basados en la semántica de los objetos. A medidad que se crea la base de conocimientos, el editor verifica la consistencia semántica de los objetos de acuerdo con la estructura, propiedades o relaciones previamente definidas.

4.6.1.4 Herramientas de ayuda a la implementación y prueba del Sistema Experto.

Los errores en un Sistema Experto se detectan de distintas formas:

- Conclusiones falsas
- Respuestas inconexas a preguntas
- Deducción de hechos irrelevantes
- No encontrar la solución a los problemas previstos.

La fuente común de los errores se encuentra casi siempre en la definición de la base de conocimientos: objetos, reglas, relaciones, procedimientos o tareas incorrectas, que son la causa del mal funcionamiento del sistema. Las herramientas orientadas a facilitar el proceso de validación permiten al ingeniero:

1.- Detectar las inconsistencias derivadas de la definición de ela base de conocimientos.

2.- Localizar objetos, relaciones o propiedades que faltan en la base de conocimientos.

3.- Obtener caminos de los procesos de inferencia.

4.- Interrogar al sistema de forma selectiva acerca de la obtención de determinados resultados, con cuestiones del tipo como: cómo has obtenido..., porqué has obtenido..., que pasaría si..., etc.

5.- Funciones para evaluar la velocidad de cálculo y para mejorar la implementación de los diferentes objetos sin variar su definición.

CAPITULO V

LENGUAJES DE PROGRAMACIÓN

EL EXITO NO DA NI QUITA SENTIDO A LAS COSAS

A.C. DEL CASTILLO

5.1 Introducción

Para desarrollar un capitulo como éste, es necesario resolver una cuestión: ¿ tiene sentido hablar de lenguajes específicos de Inteligencia Artificial y al mismo tiempo para los Sistemas Expertos ?

Muchos procesos de hoy se consideran absolutamente mecánicos (por ejemplo la resolución de operaciones aritmeticas) en otros tiempos se consideraba que para resolver ese tipo de problemas se necesitaba inteligencia. Como consecuencia sería practicamente imposible hablar de lenguajes de programación para la Inteligencia Artificial y en consecuencia para los Sistemas Expertos, ya que la variedad de enfoque técnicos impediria la existencia de un lenguaje que contuviera todas las variantes posibles para cada uno de ellos.

Con respecto a la pregunta, es un hecho que la gente que trabaja en Inteligencia Artificial considera algunos lenguajes como inherentes a esta área. Por consecuencia, sí parece que se puede hablar de lenguajes específicos para Inteligencia Artificial. Sin embargo, se podria atribuir que el éxito de LISP es histórico. Esto es, LISP fue creado para trabajar con problemas de Inteligencia Artificial en una época que las alternativas más viables eran FORTRAN o COBOL, claramente insuficientes para esta área.

5.2 Criterios para medir la calidad de un lenguaje de programación.

Para entender los aspectos que hacen que un lenguaje de programación sea considerado <<bueno>> o <<malo>> es

preciso, en primer lugar, comprender que un lenguaje de programación es, ante todo, la herramienta básica del programador. Por otro lado, la calidad de un lenguaje vendrá también condicionada por criterios de costo que vienen asociados (costo de utilización, costo de traducción, eficiencia del código objeto generado, etc.).

Para estudiar la eficacia de un lenguaje como herramienta, podemos dividir la tarea del programador en las tres fases siguientes:

1.- En la primera fase el programa *se diseña y se escribe*. El programador deseará que el lenguaje sea simple (su utilización no debe de añadir complejidad al problema a resolver), que se adecúe al tipo de problema a tratar que esté diseñado para servir de soporte para el método de diseño que utilice el programador. También, y en especial si el problema ha de ser resuelto por un equipo de programadores y no por uno solo, el lenguaje ha de tener mecanismos de modularización y facilidades para compilación separada que simplifiquen la división del trabajo y la posterior integración de cada una de las partes del programa. El lenguaje debe de suministrar mecanismos que faciliten la reutilización de software ya realizado. Finalmente, ligado a la fase de escritura o codificación del algoritmo en el lenguaje dado, se puede pedir que éste no sea ni excesivamente prolijo (que haya que escribir demasiado, incluso para programas cortos), ni demasiado rígido en cuanto al tipo de construcciones que ofrezca, ya que esto haría pesada la tarea de codificación.

2.- La segunda fase que podemos contemplar es la de *validación y puesta a punto*. Aquí se debe de exigir a los lenguajes mecanismos que incrementen la fiabilidad de los

programas a través de la detección rápida de posibles errores (por ejemplo la declaración de variables). También es necesario que el lenguaje sea legible para que, en caso de que los errores que aparezcan en ejecución, no sean muy complicados de detectar ni de hallar el origen del error.

3.- Finalmente, la tercera fase, que podemos llamar *mantenimiento*, se ocuparía de todo el trabajo que se puede realizar sobre un programa, desde que éste entra en funcionamiento hasta que se desecha. El lenguaje debería tener mecanismos de fiabilidad, ser (en lo posible) independiente de máquinas concretas, así como tener construcciones modulares que simplifiquen las posibles modificaciones.

Una cualidad que también está ligada a la visión de un lenguaje como herramienta de construcción de programas, es la integración del lenguaje con otras herramientas de diseño de software, especialmente en el marco de los llamados *entornos de programación*, ya que estas herramientas pueden suplir deficiencias aparentes del lenguaje.

Como ya se indicó anteriormente, los criterios de calidad previamente reseñados se les unen criterios de costo normalmente en términos de dos parámetros: *costo de traducción y costo de ejecución*. El primero se refiere a la eficiencia de los traductores del lenguaje y el segundo a la eficiencia de la ejecución de los programas (escritos en dicho lenguaje) una vez traducidos. Un lenguaje malo puede incrementar considerablemente el trabajo de un programador.

Por otra parte, hay que tener en cuenta que la importancia de cada uno de los criterios que han sido mencionados, o su aplicación a un lenguaje concreto, varía

con el tiempo. En efecto, por un lado el tipo de aplicaciones a resolver cambia, por otro la tecnología de harware o software avanza; ya vimos que los entornos de programación pueden alterar las cualidades de un lenguaje, y que el abaratamiento de los recursos de harware influyen en restar importancia al costo de ejecución.

Por último, hay que hacer la consideración de que el éxito de un lenguaje de programación no depende exclusivamente de sus méritos, sino que existen toda una serie de factores externos que lo condicionan, factores técnicos, sociales y economicos.

5.3 ¿ Como ha de ser un lenguaje de programación para los Sistemas Expertos ?

Todos los criterios reseñados en el apartado anterior son aplicables a los lenguajes de programación que pretendan ser de utilidad en Sistemas Expertos. En principio, el ciclo de vida del software de Sistemas Expertos no es esencialmente diferente al de otro tipo de aplicación. En concreto, criterios tan importantes como *la fiabilidad, la modularidad, la legibilidad, etc.*, tambien son aquí de plena aplicación.

Sorprendentemente, los lenguajes más generalizados en Inteligencia Artificial, LISP y PROLOG (almenos en sus versiones más simples y comunes), no cumplen los requisitos mínimos que ha de tener un lenguaje para la producción de software en media/gran escala: su ausencia de tipificación atenta contra modularidad, son poco legibles, etc.

Desde un punto de vista profesional, se nota que LISP es un lenguaje que en su concepción básica tiene 25 años y que PROLOG está todavia en fase experimental. La base de su exito es muy simple es un lenguaje que se adecua al problema a resolver.

LISP en el momento en que fu diseñado ofrecia una serie de aspectos insólitos con respecto a los lenguajes de la época, *tratamiento simbólico, recursividad y un mecanismo relativamente simple para la definición de estructuras de datos complejas,* que lo hacián especialmente adecuado para el trabajo en un área en la cual la mayor parte de los problemas son, efectivamente, de tratamiento simbólico, abundan los problemas de búsqueda o problemas en los que hay que expresar, por ejemplo, el conocimiento de un dominio por medio de estructuras de datos complejas.

Otro aspecto que hace agradable al LISP es su carácter de lenguaje funcional, que confiere una simplicidad muy atractiva.

PROLOG, evidentemente, tiene estos dos aspectos, pero además presenta otros que lo hacen aún más atractivo. Por un lado, su implementación << *automática* >> de la búsqueda exhaustiva y de la deducción tan útiles en muchos de los problemas de la Inteligencia Artificial, al igual que la herramienta para hacer *pattern matching* que nos proporciona el mecanismo de unificación. Por otro, su potencialidad para presentar información que ha de ser utilizada en procesos deductivos le hacen ser de gran utilidad en el área de Sistemas Expertos.

5.4 LISP y los lenguajes funcionales

Los *lenguajes funcionales o aplicativos* son aquellos en que todas sus construcciones son funciones en el sentido matemático del término. Esto es, en un lenguaje funcional no hay instrucciones y por lo tanto no existe la instrucción de asignación. Un programa funcional o aplicativo es una funcion que se define por composición de funciones más simples; para ejecutarlo se aplica a los datos de entrada (los datos de la función) y se obtiene un resultado (el valor calculado por la función).

El origen de este tipo de lenguajes funcionales se encuentra en los primeros años de la informática. Matemáticos que trabajaban en la caracterización del concepto de función computable (Gödel, Church, Kleene) definieron lo que podrían haber sido los primeros lenguajes de programación funcional.

Más tarde, a finales de los cincuenta, entre los primeros lenguajes de alto nivel que aparecen se encontraría un lenguaje funcinal, el LISP, definido por J. McCarthy, que desde entonces y hasta ahora prodría considerarse como el lenguaje por excelencia para la Inteligencia Artificial. Sin embargo, es necesario resaltar que las variantes de LISP más utilizadas en la práctica incorporan muchos elementos no funcionales en ayuda de la eficiencia.

A continuación se hara una pequeña introducción al LISP. Como en realidad LISP no hay uno, sino miles, se describira la parte más simple del COMMON LISP (intento americano de estandarizar este lenguaje.

El lenguaje LISP, como todo lenguaje funcional, está

definido a partir de un número muy reducido de conceptos. Esencialmente, en LISP hay átomos, listas y s-expresiones.

Los átomos pueden denotar valores numéricos, simbólicos o funciones (predefinidas o no). Por ejemplo, son átomos 12 (que denota el valor numérico 12), LUNES (que puede denotar simplemente el símbolo LUNES) o + (que denota la operación de suma). El átomo NIL denota, simultáneamente, la lista vacía y el valor booleano <<falso>>. El valor <<cierto>> puede venir denotado por el átomo T o por cualquier otra s-expresión distinta de NIL.

Las listas se forman agrupando átomos u otras listas entre paréntesis. Son, por ejemplo, listas:

```
(1 2 3 4)
(+ (+ 5 2)(+ 5 3)
(LUNES MARTES MIERCOLES)
(UNO 2 NIL (UNO)((DOS)))
```

Una s-expresión es una lista o un átomo. Una s-expresión denota una expresión a calcular aplicando la operación definida por el primer átomo a los resultados de las demás s-expresiones que constituyen la lista. Por ejemplo, la evaluación de la s-expresión (+ (* 5 2) 3) daría como resultado 13.

Si no queremos que una s-expresión sea evaluada, porque denote una estructura de datos, se le ha de preceder de una comilla (QUOTE), por ejemplo:

```
'(LUNES MARTES MIERCOLES)
```

LISP suministra las siguientes funciones básicas:

Operaciones aritméticas, como la suma (+), la resta (-),la multiplicación (*) o la división.

Operadores de manejo de listas, como CAR (extrae el primer elemento de una lista), CDR (nos devuelve como resultado toda la lista salvo el primer elemento), CONS (nos añade un elemento a una lista) o APPEND (concatenación de listas). Por ejemplo:

```
(CAR '(1 2 3) = 1
(CDR '(1 2 3) = (2 3)
(CONS 1' (2 3) = (1 2 3)
(CONS' (1 2) '(3 4) = ( (1 2) 3 4)
(APPEND' (1 2) '(3 4) = (1 2 3 4)
```

Operaciones lógicas y de comparación, como los comparadores igual (EQUAL), menor (<) o mayor (>); predicados que nos establecen las posibles características de un objeto, como ATOM (que nos dice si algo es un átomo) o LISTP (que nos dice si es una lista); y operaciones lógicas como AND, OR y NOT.

LISP también ofrece operaciones para hacer que un átomo denote un valor concreto o una función. Por ejemplo, si hacemos:

```
SETQ(DIAS' (LUNES MARTES MIERCOLES))
```

y, a continuación, pedimos la evaluación de (CAR DIAS) obtendremos como resultado LUNES. Además, podemos definir funciones por medio de la operación DEFUN que nos permite asociar a un átomo una función; por ejemplo:

```
(DEFUN SEGUNDO (L)
  (CAR (CDR L)))
```

nos define una función cuyo nombre será SEGUNDO y que nos obtiene el segundo elemento de la lista que le sirve como parámetro; por ejemplo, si la aplicamos a DIAS, (SEGUNDO DIAS), obtendremos como resultado MARTES.

Para obtener cierta potencia de la definición de funciones, LISP ofrece el condicional COND, que tiene la forma:

```
(COND ( <test 1> <expr 1> )
      ( <test 2> <expr 2> )
                ....
      ( <test n> <expr n> ) )
```

Su significado es el siguiente: se evalúan test 1, test 2, .. hasta encontrar un i tal que test i es cierto; en ese momento se ejecuta expr i.

Por ejemplo, con ayuda de COND y utilizando una definición recursiva podemos definir la función factorial:

```
(DEFUN FACT (N)
       (COND ((= N 0) 1)
             (T (*N (FACT (-N 1) ) ) ) ) )
```

En general, se asume que en LISP, como en todos los lenguajes funcionales, la única forma de definir repeticiones es por medio de la recursión, sin embargo, LISP (y otros lenguajes) pueden ofrecer formas iterativas como, por ejemplo, el operador MAPCAR. Este operador nos permite aplicar una operación repetidamente a todos los miembros de

una lista. Por ejemplo, (MAPCAR FACT'(345)) nos daría como resultado (6 24 120).

A menudo, distintas versiones de LISP ofrecen construcciones que pueden ser consideradas más o menos atípicas, normalmente para paliar deficiencias del lenguaje por ejemplo, pueden aparecer formas iterativas impuras que incrementen la eficiencia en tiempo de ejecución; o construcciones para definir módulos o tipos abstractos de datos que suplan esta falta. Sin embargo, hay dos aspectos que al ser inherentes al lenguaje difícilmente podrán ser alterados: la ausencia de tipificación, en LISP no hay tipos, con lo que esto conlleva a la falta de fiabilidad, y el *scoping* dinámico, esto es, el ámbito de una declaración no es estático (no depende de su situación en el texto fuente) sino dinámico (depende de la secuencia de ejecución). Este aspecto afecta o puede afectar gravemente a la legibilidad y por ello, de nuevo, a la fiabilidad.

5.5 PROLOG y la programación lógica

La idea básica de la programación lógica es relativamente simple. Se expresa en una frase: <<Algoritmos = Lógica + Control>>. En principio, esta frase es aplicable a cualquier tipo de programación: cualquier lenguaje de programación no es más que un lenguaje formal (con unas reglas sintácticas para la definición de construcciones perfectamente definidas) con una semántica precisa sin ambigüedades (el programador sabe o debería saber, cómo se ejecutarán sus programas). En cambio, el *control* asociado (su forma de ejecución) suele ser trivial.

Por otra parte, si pensamos en los lenguajes ligados a la lógica matemática clásica (lógica de proposiciones, lógica de predicados, etc.) la situación es la contraria: lo difícil es definir su forma de ejecución.

En general, cuando se habla de programación lógica se entiende que el lenguaje de programación será un lenguaje <<lógico>> en el sentido tradicional, normalmente una restricción de cálculo de predicados de primer orden.

El lenguaje más conocido de programación lógica (de hecho el término *programación lógica* viene del nombre de este lenguaje) es PROLOG creado por A. Colmeraurer en los setenta. La base de PROLOG es la lógica clausal y su forma de ejecución el principio de resolución.

La lógica clausal es una restricción de la lógica de predicados de primer orden en que las únicas definiciones admitidas son de la forma:

$$p(X1,\ldots,Xn):-p1\ (\ldots),\ldots,pm\ (\ldots).$$

(La propiedad p aplicada a $X1,\ldots,Xn$ se cumple si se cumplen $p1,\ldots,pm$ aplicadas a sus respectivos parámetros).

Un caso particular de reglas clausales son las afirmaciones:

$$p(X1,\ldots,Xn).$$

(La propiedad p aplicada a $X1,\ldots,Xn$ se cumple siempre).

Un programa en PROLOG consta de una serie de cláusulas o afirmaciones que definen unos predicados o propiedades.

Por ejemplo:

```
padre ('Juan','Pedro'). /* Juan es el padre de pedro */
padre ('Juan','Ramón').
padre ('Juan','Luis').
padre ('Luis','Francisco').
padre ('José','Nicolas').
hermano (x,y) :- padre (z,x), padre (z,y), x\=y./* x es
hermano de y si hay un z que es padre de ambos y además x e
y no son la misma persona */
```

Al introducir en el ordenador el programa, el sistema crea una *base de datos* con las definiciones y, a continuación, es capaz de responder preguntas que se basen en dichas definiciones. Por ejemplo:

```
? - padre (x, 'Luis'). /* ¿Quién es el padre de Luis? */
```

El sistema responderá:

```
x = 'Juan'
? - padre ('Juan',x). /* ¿Quién es el x cuyo padre es Juan?
*/
x = 'Pedro'; x = 'Ramón'; x = 'Luis'
? - hermano ('Luis',x). /* ¿Quién es hermano de Luis? */
x = 'Pedro'; x = 'Ramón'
? - hermano ('Pedro','Luis'). /* ¿Es Luis hermano de Pedro?
*/
yes
? - hermano ('Pedro',x), padre(x,y). /* ¿Quienes son los x e
y tales que x es hermano de pedro y padre de y? */
x = 'Luis'; y = 'Francisco'
```

Como se puede ver a través de este ejemplo, en PROLOG a diferencia de otros lenguajes más convencionales, los programas no tienen que predefinir qué es dato y qué es

resultado (para el mismo programa se admiten preguntas `hermano ('Luis',x)` y `hermano ('Pedro','Luis')` y además son capaces de producir resultados múltiples (por ejemplo, `padre ('Juan',x)` produce tres resultados).

La forma de trabajar del intérprete de PROLOG se puede resumir en dos palabras: *unificación y vuelta atrás*. En concreto, el funcionamiento es el siguiente: en un momento dado el intérprete tiene que demostrar que un cierto número de *objetivos* se cumplen (si estos objetivos tuvieran variables se tendría que descubrir para qué valores se hacen ciertos). Por ejemplo, ante la pregunta:

```
? - hermano ('Pedro',x), padre (x,'Francisco').
```

el intérprete tiene dos objetivos a cumplir: `hermano ('Pedro',x)` y `padre (x,'Francisco')`. Inicialmente intentará cumplir el primer objetivo, para ello buscará en la base de datos una cláusula cuya parte izquierda se unifique con el objetivo, esto es, una cláusula que al substituir sus variables y las del objetivo (si las hay) por valores o por otras variables, la cabeza de la cláusula y el objetivo se conviertan en el mismo término. En este caso, `hermano ('Pedro',x)` y la cabeza de la sexta cláusula se unifican.

Si la cláusula tuviera parte derecha, los objetivos de la condición (una vez aplicada la substitución definida por la unificación) se incorporan a la lista de objetivos a cumplir, substituyendo al anterior objetivo. Esto, en nuestro caso sería:

```
padre (z,'Pedro'), padre (z,y), 'Pedro' \ = y, padre (y,'Francisco').
```

y el proceso continúa con la nueva lista.

Si la cláusula unificada no tuviera condición, el objetivo se elimina de la lista. Además, si una variable de un objetivo se substituye por un valor como consecuencia de una unificación, esta substitución se produce también en los otros objetivos de la lista que contengan la variable. Por ejemplo, al satisfacer el objetivo `padre (z,'Pedro')` la unificación se produce en la primera cláusula, substituyendo z por `'Juan'`; como consecuencia la nueva lista quedaría:

```
padre ('Juan',y), 'Pedro' \ = y, padre (y,'Francisco').
```

El proceso continuaría intentando satisfacer el siguiente objetivo, `padre ('Juan',y)`, que se resolvería unificando con la primera cláusula y quedaría la lista:

```
'Pedro'\ = 'Pedro', padre ('Pedro','Francisco').
```

Si un objetivo no se puede cumplir (es falso), el sistema da marcha atrás hasta la última decisión tomada (última unificación) y la reconsidera; es decir, intenta otra unificación. En nuestro caso, reconsideraría la unificación `padre ('Juan',y)` con `padre ('Juan','Pedro')`, cambiándola por la unificación con la segunda cláusula, con lo que la lista de objetivos quedaría:

```
'Pedro'\ = 'Ramón', padre ('Ramón','Francisco').
```

Ahora, el objetivo `'Pedro'` es distinto de `'Ramón'` se cumple, con lo que el sistema intentaría demostrar `padre ('Ramón','Francisco')`; como eso no es cierto (ya que en la base de datos no hay ninguna cláusula que lo afirme) habría una nueva marcha atrás y se volvería a reconsiderar la

unificación de padre ('Juan',y), que ahora se produciría con la tercera cláusula, con lo que se tendría:

'Pedro'\ = 'Luis', padre ('Luis','Francisco').

Ambos objetivos se cumplen. En este momento el sistema daría como resultado los valores por los que se han substituido las variables de los objetivos iniciales en el proceso de unificación. En nuestro caso:

x = 'Luis'

Para encontrar soluciones alternativas (si existen) el sistema daría de nuevo marcha atrás e intentaría reconsiderar las unificaciones hechas hasta que no se encontrara ninguna nueva que satisfaciera los objettivos; en este momento terminaría. Si no encontrara ninguna solución (los objetivos son falsos) el sistema terminara diciendo que no hay solución.

El PROLOG puro, tal como se ha descrito (es evidente que en un nivel primario), presenta un grave problema en el mecanismo de vuelta atrás. Según se ha visto, cuando después de producirse una anificación para satisfacer un objetivo algo falla, el sistema retrocede hasta encontrar otra unificación posible entonces continúa. Sin embargo, a veces podemos estar seguros de que ésa era la única unificación posible para dicho objetivo y, por tanto, no queremos que sea reconsiderada, ya sea por motivos de eficiencia (la reconsideración de la unificación es una pérdida de tiempo real de proceso), ya sea por motivos de corrección (la reconsideración de la unificación puede llevar a resultados erróneos). Para resolver este tipo de problemas se ha introducido un mecanismo de limitación de la vuelta atrás

llamado operador de corte, que se denota por una exclamación. Si en un programa aparece la cláusula:

```
p (...) : -p1 (...) ,...,pi (...) !,...., pn (...)
```

si durante la satisfacción de un objetivo se produce una unificación con esa cláusula y además el corte se ejecuta, esto es, los objetivos p1 (...) ,..., pi (...) se verifican, las unificaciones que permitieron resolver p1 ,..., pi y la propia p se congelan; es decir, en caso de vuelta atrás no serían reconsideradas.

Una de las utilizaciones más usuales del corte es la implementación de la negación. Supongamos que queremos que un objetivo p se cumpla si otro objetivo q no se cumple, y viceversa. El siguiente programa definiría p:

```
p  :  -q,!,fail.      /* fail es un objetivo que nunca se
     cumple */ p.
```

En efecto, supongamos que hay que ver si el objetivo p se satisface. El sistema inicialmente unificaría con la primera cláusula y consideraría que tiene que cumplir la siguiente lista de objetivos:

```
q ,! , fail
```

Si q fuera falso el sistema daría directamente marcha atrás e intentaría encontrar otra cláusula para unificar p y, en tal caso, la segunda le serviría. Como además la segunda cláusula no tiene condición, p sería cierto directamente. Si *q* fuera cierto, a continuación el sistema ejecutaría el corte, lo cual supondría la congelación de *p* y de *q* de tal forma que, al no poder cumplir fail,p no sería reconsiderado y, por tanto, se consideraría falso.

PROLOG es un lenguaje especialmente asociado para resolver problemas de *búsqueda*, esto es, problemas cuya solución más natural se encuentra siguiendo la técnica de *prueba y error*.

La solución en PROLOG de problemas es especialmente simple: soló hace falta especificar el problema y el sistema hace la búsqueda, mientras que en un lenguaje convencional el programador tendría que programar dicha búsqueda.

Los mayores problemas de este lenguaje vienen de la falta de legibilidad (y, por tanto, de fiabilidad) que supone el uso del corte, de la ausencia de tipificación, lo que supone una falta de fiabilidad del lenguaje (al igual que en el paso del LISP), y de la ausencia de construcciones modulares. Recientemente, ha habido propuestas para combinar PROLOG con lenguajes de especificación algebraica de tipos abstractos de datos, combinando la ejecucion por si tuvieran éxito, resolverían la mayor parte de los problemas antes citados.

5.6 Expectativas a futuro

En principio, parece que tenemos LISP y PROLOG (o variantes suyas) para muchos años. Sin embargo, en los últimos tiempos una nueva línea parece haberse introducido con éxito en Sistemas Expertos: la programación orientada a objetos y el lenguaje que la encarna, SMALLTALK.

SMALLTALK tiene características muy interesantes que le hacen ser un candidato con futuro en esta área. Por una parte, cumple con (lo que parece ser) los requisitos mínimos

de un lenguaje de programación para Sistemas Expertos: incorpora tratamiento simbólico y su concepto de *array* es similar a las listas de LISP. Pero, por otra parte, incorpora aspectos que suplen las deficiencias más notables de los lenguajes que se han descrito.

La filosofía básica de SMALLTALK, la programación orientada a objetos, está basada en dos ideas: la programación con módulos y el concepto de herencia.

En SMALLTALK la construcción básica son los objetos *(módulos)* que agrupan un conjunto de variables, que definen su estado, y que sólo pueden ser modificadas a través de las operaciones que ofrece el objeto en su interfaz. Cuando un objeto desea que otro efectúe una operación le envía un *mensaje*. Esto es, los objetos se comunican por intercambio de mensajes (que son ordenes de ejecución de operaciones).

Para definir un objeto, es preciso primero definir su *clase*, ya que todo objeto es una *instancia* de una clase. La definición de una clase incluye la definición del conjunto de variables que constituirán el estado interno de los objetos de esta clase y la definición de los metódos (descripción de la implementación de las operaciones) asociados a las operaciones que servirán para manipular dichas variables. La definición de una clase también incluye la declaración de la superclase de la que deriva. En SMALLTALK las clases se organizan jerárquicamente. Toda clase *C* puede tener varias subclases *C1,....,Cn*, en ese caso se dice que la clase *C1,....,Cn* tienen como superclase a *C*. Cuando se declara que la clase C1 es una subclase de *C2*, *C1* <<hereda>> las variables y los métodos de *C2* (y, evidentemente, puede incluir nuevas variables o métodos). Este es uno de los mecanismos de reutilización más potentes

que han aparecido hasta ahora en un lenguaje de programación.

En conjunto SMALLTALK parece ser un lenguaje muy bien diseñado. Es pequeño y compacto, ya que está basado en un número reducido de ideas. Evidentemente, cumple a plena satisfacción los requisitos de modularidad, reutilización, fiabilidad, etc. Parece razonablemente fiable, a pesar de ser poco rígido, devido a la clase de polimorfismo que soporta. Su utilización se realiza en el marco de un entorno interactivo con el que está totalmente integrado (hasta el punto que no se concibe el lenguaje sin el entorno), lo que le da un atractivo adicional. Quizá el mayor defecto que se le pueda encontrar es la poca legibilidad que le da una sintaxis algo enrevesada. También puede citarse como defecto, aunque muy poco importante debido a su entorno habitual de utilización (estaciones unipersonales), una cierta falta de eficiencia en la implementación debida al uso en forma masiva de gestión dinámica de memoria.

SMALLTALK ha tenido ya influencia en LISP, habiendose definido variantes de este lenguaje con construcciones para la programación orientada a objetos.

CAPITULO VI

ENTORNOS DE PROGRAMACIÓN

NO SE DEBE DEJAR CON
LIGEREZA LO QUE ES
DELICADO DEJAR SIN
RESOLVER

JUAN L. VIVES

6.1 Introducción.

Para la construcción de sistemas basados en las técnicas y métodos de la Inteligencia Artificial, del mismo modo para cualquier sistema informático, se debe de construir un programa. En base a este planteamiento se puede generar una pregunta ¿ que relación existe entre los entornos y la Inteligencia Artificial ? Los *entornos de programación* son los que se utilizan al construir un sistema inteligente, o cualquier sistema.

6.2 Definición y objetivos.

El concepto de *Entorno de Programación* es relativamente reciente ya que aparece, al menos con esta denominación a mediados de la decada pasada, anteriormente se conocia como *sistemas de ayuda al desarrollo, programación asistida por computadora CAP* ("computer aided programming") u otros conceptos que denotan conceptos similares.

Una definición concreta de entornos de programación puede ser: « son un conjunto de herramientas de software que asisten al programador en un contexto de programación ». El objetivo de la construcción de los entornos de programación es el integrar en un único marco conceptual y en un único sistema de programación varias actividades del programador. Se pude considerar como una capa superpuesta al sistema operativo para ayudar al programador en su tarea. La versión inglesa del término es «programing environment» y en algunos paises de habla castellana se les conoce por *ambiente de programación*.

Un aspecto importante de los entornos de programación

es para que tipo de programación estan orientados pequeña o en gran escala «*programming-in-the-small* y *programming-in-the-large*». Cada uno de estos tipos de entornos de programación ponen énfasis en aspectos muy diferentes uno esta orientado al trabajo individual y el otro a problemas de integración. Es frecuente distinguir los orientados a gran escala como *Entornos de Ingenieria del Software* («Software Enginiering Environments»).

La denominación de entornos de programación coincide con la aparición de dos de sus antecesores directos UNIX e Interlisp. La idea de sistema operativo virtual, como capa de software superpuesta al sistema operativo y que ejerce una función de colchón.

UNIX. El modelo de capas, en el que el entorno de programación es una capa más, la utilizada para desarrollar programas. Otra aportación es la integración de todas las herramientas en un sólo sistema.

Lo que distingue en especial a los entornos de programación es el marco de integración de todas las herramientas dispersas en un operativo único.

INTERLISP. La aportación es la de un sistema orientado a escribir programas en un lenguaje determinado (como generalidad LISP).

En realidad existe un objetivo general único, el disponer de medios para superar el problema de la creciente complejidad del software que se diseña; es decir, de ayuda a si misma. Dentro de este objetivo general se encuentran objetivos especificos como los siguientes:

1.- Hacer más sencilla la edición de un problema, y más fiable el resultado.

2.- Soportar los requerimientos de un método.

3.- Dibujar diagramas de diversos tipos.

4.- Verificar el trabajo de diseño.

5.- Depurar y ejecutar pruebas.

6.- Efectuar transformaciones sobre los programas.

7.- Programar en forma fiable sobre la terminal.

6.2.1 Cajas de herramientas.

Denominados *tool box* o *tool environment*, se trata de entornos de programación cuyo uso no obliga al programador a trabajar con un lenguaje o método determinado; al contrario, pone a disposición del programador un *buffet* de posibilidades que el programador escoge las que mejor se acomoden a su estilo de programar. Devido a esta flexibidad son las más difundidas comercialmente en la actualidad.

Además de programas habituales de ayuda (compiladores, editores, etc.) suelen contener también ayudas descendentes de programas de una notación a diversos lenguajes, ayudas a la documentación, ejecución simbólica, menús, ayudas de construcción de prototipos y otras; UNIX, sigue este enfoque.

6.2.2 Basados en un lenguaje.

Son entornos de programación orientados a la producción de programas en un lenguaje especifico. Suelen construirse en base a un editor dirigido por la sintaxis, de forma que este se constituya en núcleo y las herramientas trabajen directamente sobre la representación interna.

El ya citado INTERLISP pertenece a esta clase. El *Cornell Program Synthesizer* trabaja sobre PC/CS. Además se encuentran para PROLOG o SMALLTALK.

6.2.3 Basados en un método.

Son entornos que permiten al programador trabajar con un método determinado, según su forma habitual de trabajar.

Estos son productos cuyo objetivo es el de servir de soporte al desarrollo de programas según un método particular. Por ejemplo el entorno OBJ se basa en un método de diseño modular mediante tipos abstractos de datos.

6.2.4 Programación gráfica.

El estilo impuesto por distintas microcomputadoras (como McIntosh) basado en editores con ventanas, mouses han impactado y actualmente se trabaja con ellos en INTERLISP o en el entorno de programación de SMALLTALK.

6.3 Ingenieria del software.

La Ingenieria del software continúa acrecentando su desarrollo e importancia económica, a pesar de encabezar los sectores de más baja productividad. Esta situación se deve a la escasa automatización del proceso de producción de un programa (figura 6.1). Las diferentes fases del ciclo de construcción de un programa continúan realizándose manualmente. Las herramientas existentes se aplican a tareas marginales (detección de errores) o asisten al programador en aspectos como edición y optimización.

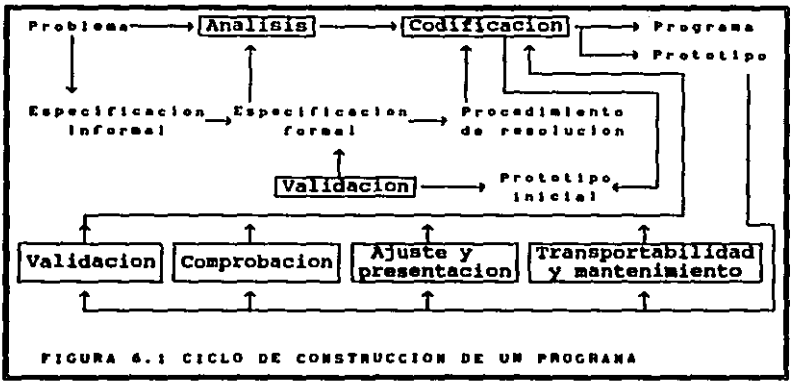


FIGURA 6.1 CICLO DE CONSTRUCCION DE UN PROGRAMA

Por este hecho las líneas de trabajo se han encaminado en diferentes direcciones:

1.- *Verificación automática de programas*: Basándosse en las ideas sobre verificación, se construyen sistemas para demostrar formalmente la corrección de un programa apartir del texto fuente y de un conjunto de aserciones. Las

aserciones expresan propiedades que deven ser verificadas por el programa en determinados puntos del mismo: entrada, salida o bucles.

2.- *Síntesis automática de programas:* La idea básica consiste en automatizar las últimas fases del ciclo de construcción de un programa (figura 6.2). El programa se obtiene a partir de la especificación del problema. Los sistemas desarrollados por distintas métodologias (inducción, deducción) usan diferentes formalismos de especificación (ejemplos, lenguajes formales, etc.).

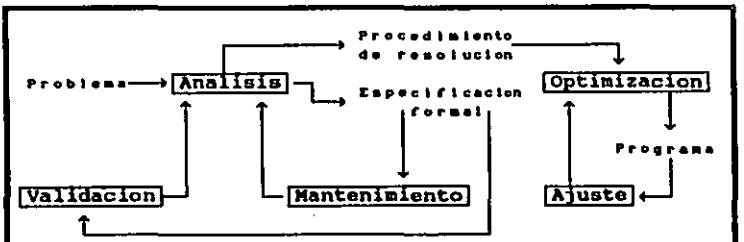


FIGURA 6.2 CONSTRUCCION DE UN PROGRAMA CON ENTORNO INTELIGENTE

3.- *Sistemas inteligentes de ayuda a la programación:* La síntesis automática de programas eficientes aplicables a problemas reales es todavía un objetivo inalcanzable con la tecnología y los medios actuales. Consiste en desarrollar herramientas que proporcionen asistencia inteligente en las partes más

criticas del ciclo de construcción. Un sistema inteligente puede:

a) Guiar el proceso de refinamiento de un algoritmo.

b) Detectar inconsistencias devidas a modificaciones

c) Sujerir estructuras más idóneas.

d) Asistir a la tarea de documentación.

6.4 Herramientas concretas

Todo lo expuesto hasta ahora se refiere a entornos de programacion de aplicación general (aunque alguno de ellos pueda tener dominios específicos).

Existe una serie de productos para la producción de determinados problemas; por ejemplo:

a) *Sistemas de producción de compiladores*, como los programas de UNIX (YACC).

b) *Sistemas para la producción de sistemas de gestión:* como RPG o alrededor de una base de datos, los llamados «lenguajes de cuarta generación».

c) *Sistemas para la construcción de sistemas expertos.*

6.4.1 Entornos existentes para la producción de Sistemas Expertos

La primera generación de entornos para la construcción de sistemas expertos tuvo su origen en laboratorios de investigación. La comercialización ha obligado a revisar en profundidad numerosos aspectos: estructura, objetivos, técnicas de implementación, tipo de máquina necesaria para el funcionamiento, y otras. El número, alcance y características de herramientas que integran los entornos dependen de distintos parámetros.

6.4.1.1 El tipo de forma para expresar el conocimiento.

Se distingen dos grandes grupos: formalismos basados en reglas y formalismos basados en objetos o frames. Los entornos pertenecientes al primer grupo se inspiran según el mecanismo de inferencia.

Son los sistemas más difundidos, sobre todo para PC compatibles, reglas formadas por atributo-valor con factores de certeza y mecanismo de inferencia dirigidos por objetivos.

Otra familia más reducida esta formada por sistemas basados en reglas de producción inspirados en el OPS (reglas con funciones booleanas en las premisas y acciones en lás conclusiones, mecanismo de inferencia dirigido por los datos), (figura 6.3).

Herramientas	Entornos					
	1	2	3	4	5	6
Representación del conocimiento						
-Reglas de producción				*		
-Reglas de inferencia	*	*	*		*	*
-Estructuración de objetos					*	
Mecanismos de razonamiento						
-Hacia adelante		*	*	*	*	*
-Hacia atrás	*	*	*	*	*	*
-Factores de certeza		*	*			
Resolución de conflictos						
-Primera regla encontrada	*		*	*	*	*
-Antecedente más sencillo		*				
Creación de la base de datos						
-Editores textuales	*	*	*	*	*	*
-Editores gráficos					*	*
Implementación y prueba						
-Trazadores de inferencia	*	*	*		*	*
-Porqué	*	*	*		*	*
-Cómo	*	*	*		*	*
-Librería de casos	*	*	*		*	*
Realización de la interfaz con el usuario						
-Menú	*	*	*	*	*	*
-Frases cortas		*	*		*	*
-Ayudas	*		*		*	*
-Distintos niveles de preguntas			*		*	*
Interfaz con el sistema operativo						
-Lenguajes	P	p		L	L	L
-Bases de datos		*				
-Otros						
Hardware necesario						
-Tipo de máquina	□	□	□	□	□	μ
-Memoria principal	θ	θ	c	θ	ω	ω
-Memoria secundaria					ϑ	
Software						
-Sistema operativo	M	M	M	M	M	
-Lenguajes de implementación	P	p	P	L	L	L
Precio @ USA	1	2	3	4	5	6

SIMBOLOGIA

1 → 895 U.S.D
2 → 495 U.S.D
3 → 10000 U.S.D
4 → 1000 U.S.D
5 → 3000 U.S.D
6 → 5000 U.S.D

P → PROLOG
p → PASCAL
L → LISP
□ → PC
μ → McINTOSH
M → MS-DOS

1 → ES/P ADVISOR
2 → INSIGHT 2
3 → M 1
4 → OPS 5
5 → PC PLUS
6 → NEXPERT

θ → 128 K
c → 192 K
ω → 512 K
ϑ → 10 M

FIGURA 6.3 TABLA COMPARATIVA DE LAS CARACTERISTICAS DE LOS ENTORNOS BASADOS EN REGLAS.

El segundo gran grupo mencionado está basado en la idea de frame y la programación orientada a objetos. El objetivo básico consiste en permitir la descripción declarativa de distintos elementos que intervienen en la solución del problema: elementos materiales, conceptos, tareas, procedimientos, reglas de inferencia, etc.

La estructuración de los objetos en taxonomías relacionadas mediante su pertenencia a clases presenta múltiples ventajas sobre formalismos basados unicamente en reglas:

1.- Se realizan procesos de inferencia de manera sencilla y eficiente.

2.- La estructuración de los objetos permite aplicar las reglas de inferencia de forma selectiva y más eficiente limitando el espacio de búsqueda en la base de datos.

3.- La definición explícita de la estructura de los objetos, sus relaciones y propiedades permite la utilización de herramientas de visualización gráfica, que facilita la construcción y el desarrollo del sistema experto.

Este tipo de representación requiere en general máquinas grandes y su coste es elevado (figura 6.4).

Herramientas	Entornos			
	1	2	3	4
Representación del conocimiento				
-Relaciones entre objetos y mecanismos de herencia	*	*	*	*
-Procedimientos	*	*	*	*
-Tipos de reglas	*	*	*	*
-Inferencia	*	*	*	*
-Producción	*	*	*	*
-Restricciones	*			
-Hipótesis	*	*		*
Mecanismos de razonamiento				
-Hacia adelante	*	*	*	*
-Hacia atrás		*	*	*
-Factores de certeza				
Búsqueda de objetos				
-Contextos	*	*	*	*
-Definible por el usuario	*	*	*	*
Resolución de conflictos				
-Primera regla encontrada	*		*	*
-Antecedente más sencillos		*		
-Definible por el usuario	*	*	*	*
Arquitecturas				
-Meta-reglas	*	*	*	*
-Agenda		*		*
-Pizarra	*			
Creación de la base de datos				
-Editores textuales	*	*	*	*
-Editores gráficos	*	*	*	*
-Multiventanas	*	*	*	*
Relación de la interfaz con el usuario				
-Ventanas usuario	*	*	*	*
-Interfaz lenguaje natural				*
Interfaz con el sistema operativo				
-Lenguajes	*L*	*L*	*L*	*L y P*
-Base de datos				
Precio $ USA	1	2	3	4

SIMBOLOGIA

1 → 80,000 U.S.D 4 → 50,000 U.S.D 1 → ART 2 → KEE
2 → 60,000 U.S.D *L* → LISP 3 → LOOPS 4 → KC
3 → 300 U.S.D *P* → PROLOG

FIGURA 6.4 TABLA COMPARATIVA DE LAS CARACTERISTICAS DE LOS ENTORNOS BASADOS EN OBJETOS O FRAMES.

6.4.1.2 La máquina en que funciona el entorno

Las características de la máquina imponen restricciones importantes tanto en el número de herramientas como en el tipo de formalismos de representación del conocimiento y la eficiencia del sistema experto que se puede construir con el entorno. Se distinguen tres categorías:

1.- *Sistemas orientados a ordenadores personales* (PC compatibles, Macintosh, IBM, HP, etc.).

2.- *Sistemas orientados a máquinas convencionales de mediana o gran potencia* (UNIX, VAX, CRAY).

3.- *Sistemas orientados a máquinas LISP.*

Se tiene que tomar un énfasis especial al último grupo de máquinas pues son las que han sido concebidas para la realización de procesos simbólicos; además, poseen interfaz gráfica, tratamiento numérico de la información y una conexión con máquinas convencionales. No pueden ser consisderadas como máquinas de quinta generación (figura 6.5) pero ofrecen la única aproximación valida por el momento.

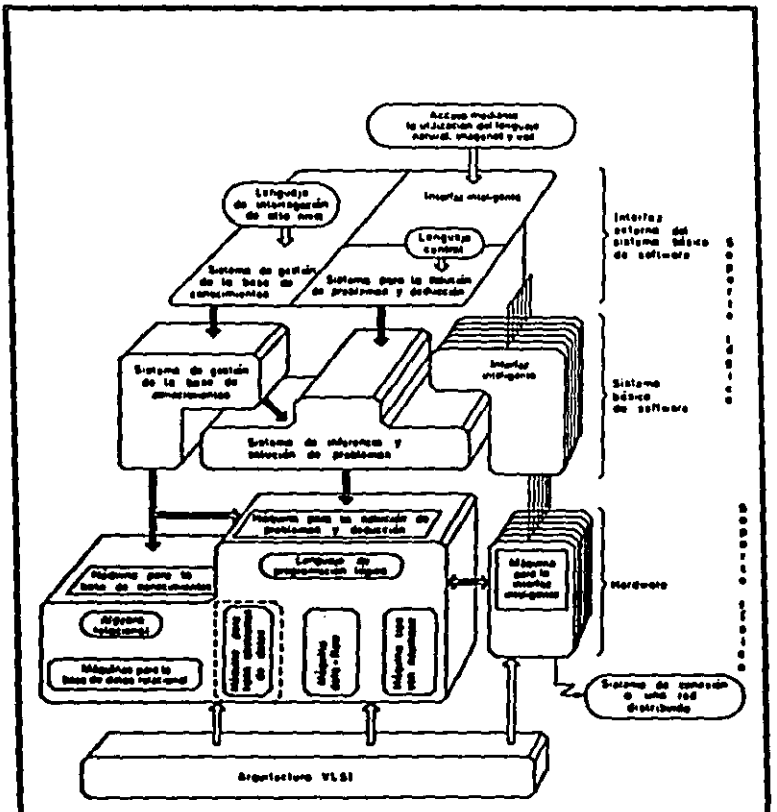


FIGURA 6.5 CONFIGURACION BASICA DE UN SISTEMA DE QUINTA GENERACION

6.4.1.3 Arquitectura para definir y manejar los procesos de razonamiento.

Los modelos y herramientas para definir y controlar el proceso de solución de un problema están relacionados con las características de la máquina y con el formalismo de expresión del conocimiento. Para tener una idea más exacta del alcance del entorno vale la pena tomar en cuenta los siguientes puntos:

a) *Gestión y control de hipótesis.* El ingeniero puede definir y manejar la generación y verificación de hipótesis que el sistema se encarga de validar.

b) *Gestión y control de tareas.* El entorno permite definir unas tareas específicas, y al mismo tiempo un mecanismo de control que permite ejecutarla en un estado del sistema.

c) *Utilización de modelos de razonamiento con datos inciertos o inexactos.* Los más utilizados están basados en aproximaciones probabilísticas: modelos bayesianos en combinaciones *ad-hoc* de valores de certeza (*Shortlife* en MYCIN) y, finalmente, en la lógica borrosa.

d) *Mecanismo de aplicación de reglas.* En su mayoría el proceso de inferencia es un mecanismo predefinido. En la aplicación de reglas pueden darse dos casos diferentes:

1.- Varias reglas son aplicables a una situación.

2.- Una regla es aplicable a varias situaciones. Las estrategias de elección más usuales son:

+ Aplicar la primera regla seleccionada.
+ Aplicar la más frecuente.
+ Aplicar la última.
+ Aplicar la menos frecuente.
+ Aplicar la regla cuyo antecedente sea más simple o sea más complejo.

6.4.1.4 Tipos de reglas.

El mecanismo de razonamiento de el entorno viene definido, en gran parte, por la semántica asociada a las reglas. Los tipos de reglas más significativos son los siguientes:

Reglas de producción. Las premisas expresan condiciones que, desencadenan las *acciones* expresadas en las conclusiones de las reglas.

Reglas de inferencia. Son premisas y conclusiones que se refieren exclusivamente a lo existente en la base de datos.

Reglas para el manejo y propagación de restricciones. Es una fase especial de las reglas de producción.

Reglas para el manejo y validación de hipótesis. Permiten controlar el proceso de generación de hipótesis y su tratamiento en paralelo.

CAPITULO VII

APLICACIONES

EL FUTURO TIENE MUCHOS NOMBRES:
PARA LOS DEBILES ES LO INALCANZABLE
PARA LOS TEMEROSOS LO DESCONOCIDO
PARA LOS VALIENTES ES LA OPORTUNIDAD

VICTOR HUGO

7.1 Introducción

Como se ha explicado en otros apartados de esta tesis un sistema experto es una entidad capaz de resolver problemas de un determinado dominio siguiendo una secuencia similar a la que utilizaria un experto humano. Aunque la tecnología de sistemas expertos es relativamente nueva, ha sido exitosamente utilizada en disciplinas tales como diagnóstico médico, prospección geológica, configuración de equipos de cómputo, diagnostico de fallas de máquinas rotatorias e inclusive en diversas aplicaciones financieras. Actualmente existen cientos de sistemas expertos prototipos en una gran cantidad de ramas del conocimiento.

7.2 Sistemas Expertos para Medicina

Las aplicaciones médicas han sido siempre uno de los campos de atención preferente. A principios de los años cuarenta, una de las primeras aplicaciones, los sistemas de procesamiento de datos lograron la automatización de historias clinicas. Como un origen de sistemas exprtos en medicina se puede pensar en un trabajo de diagnostico atomático y la representación del conocimiento médico. En 1978 el número de aplicaciones médicas llegaba a más de 800 y según Kulikowski el número total de articulos publicados sobrepasaria las 2000.

En la tabla 7.1 se da una relación de los sistemas expertos más importantes desarrollados en el campo médico. En la columna de modelo se indica la técnica usada para la representación del conocimiento. A continuación se comenta brevemente algunos de los más significativos.

MYCIN, desarrollado en la Universidad de Stanford, el sistema más conocido y el que más a influido en el desarrollo de otros. Su arquitectura es basada en reglas de producción con factores de certidumbre, fué la base del sistema esencial EMYCIN con el que se desarrollaron posteriormente PUFF, SACON, CLOT, DART, etc. El dominio de conocimiento de MYCIN es el diagnóstico y tratamiento de enfermedades infecciosas. Las evaluaciones de MYCIN demostraron que su nivel de experiencia es comprobable al de los expertos humanos, y superior al de los médicos residentes.

PUFF, sistema experto desarrollado para enfermedades pulmonares, desarrollado con EMYCIN, este sistema se codifico en BASIC y se utiliza en el « Centro Médico del Pacífico », sobre un sistema de computo Digital PDP - 11. El sistema recoge la historia clinica del paciente y las salidas de un espirómetro, e interpreta los resultados de las pruebas.

CASNET (Causal ASociational NETwork), procede de la Universidad de Rutgers: su dominio de conocimiento es en oftalmología y se usa una red semántica para representar el conocimiento en cuatro niveles como puede verse en la figura 7.1, CASNET dialoga con el usuario planteando una serie de preguntas sobre el paciente y sus manifestaciones.

EXPERT a dado como resultado al llamado SPE (Serum Protein Electrophoresis) para representación de datos de electroforesis de proteinas del suero. Este sistema no sólo da los gráficos resultantes deel análisis sino también se interpretación.

Sistema	*Dominio de aplicación*	*Modelo*
ABEL	Problemas Electrolíticos	RS
ANTICIPATOR	Prescripciones de antibióticos	RP
APES	Dermatología	RP
CAA	Electrocardiología	RS
CADUCEUS	Medicina interna	RS y E
CASNET	Oftalmología	RS
GUIDON	Enseñanza	RP
INTERNIST	Medicina interna	RS y E
IRIS	Oftalmología	RP, RS y E
MYCIN	Enfermedades infecciosas	RP
NEPHROS	Insuficiencia renal	RS
NEUROLOGIST	Neurología	RP
PIP	Nefrología	E
PUFF	Neumología	RP
RX	Reumatología	RP
SPE	Electroforesis	RP
VM	Monotorización respiratoria	RP

TABLA 7.1 RELACION DE SISTEMAS EXPERTOS ORIENTADOS A MEDICINA
RP = Reglas de Producción
RS = Redes semánticas
E = Estructuras (frames)

INTERNIST desarrollado en la Universidad de Pittsburg, tiene como objetivo el diagnostico en todo el campo de la medicina interna. El sistema a evolucionado en una versión llamada CADACEUS, que este último contempla un diagnostico de 550 enfermedades.

PIP (Present Illness Program) en el campo de nefrología, este solo fue herramienta para el estudio del conocimiento, fue desarrollado en el MIT.

RX desarrollado por Robert Blum en Stanford, concretamente se ha utilizado junto con la base de datos ARAMIS (American Rheumatism Association Medical Information System) para inducir nuevos conocimientos médicos del

análisis de los datos. Para ello incluye conocimientos sobre patofisiología y sobre estadística. Su idea básica es la de automatizar el ciclo Conocimientos medicos-Observación de datos- Planteamiento de una hipótesis-Diseño estadístico de un experimento-Resultados-Nuevos conocimientos, (este puede ser el proceso de la actividad médica (figura 7.2).

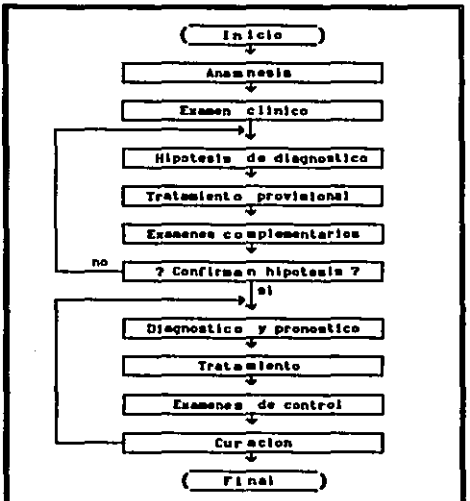


FIGURA 7.2 PROCESOS DE LA ACTIVIDAD MEDICA

7.3 Sistemas Expertos para la producción industrial.

Tareas tan comunes en procesos industriales como el diseño de nuevas piezas, el control de la producción o la supervisión han sido hasta ahora difícilmente implementados en las computadoras, devido a su complejidad. Sin embargo se

han desarrollado sistemas enormemente complejos que permiten realizar parte de estas tareas. El problema principal se tiene cuando se le exige al sistema mayor flexibilidad, integración y facilidad para incorporar nueva información.

Existen enfoques diferentes para la construcción de sistemas expertos aplicables a procesos industriales, según el grado de interacción que exista entre el proceso y el usuario:

a) Enfoque interactivo: El propio sistema entabla comunicación con el usuario, formulando preguntas y deduciendo hechos hasta proponer una o varias soluciones. Este enfoque es muy usado en sistemas de diagnóstico de averías (sistema experto DART), de reparación y en general de mantenimiento (sistema experto ACE).

b) Por lotes «Batch»: El sistema experto parte de unos datos suministrados y el sistema experto explora y construye las soluciones sin diagnosticar con el usuario. Se han aplicado en procesos que requieren funcionamiento automático (control de calidad tareas de montaje). Planificadores de procesos (ISIS II) y sistemas sw sucesión de operaciones a realizar sobre una pieza (GARI).

c) Enfoque híbrido: Es el más usado en esta aplicación especifica de sistemas expertos. Se usan para el desarrollo de nuevos productos, tal es el caso del sistema CRITTER.

Este útimo tipo de sistemas expertos han sido muy enfocados al área de diseño en el campo de la electrónica, devido a la necesidad de crear nuevos circuitos de elevada complejidad. Se han construido gran variedad de sistemas para el diseño de circuitos VLSI (Very Long Scale Integration) y también para circuitos de baja escala de integración. En esta área se desarrolla el sistema DAA de la Universidad de Carnegie-Mellon, el sistema DFT de la Universidad de Syracuse, El sistema PEACE de la Universidad de Manchester, el sistema REDESIGN de la Universidad de Rutgers y el sistema CRITTER de la misma Universidad.

7.4 Sistemas Expertos desarrollados en México.

Como se ha visto hasta ahora las aplicaciones de los Sistemas Expertos en el mundo abarcan muchos campos de investigación y a su vez esto conlleva a la creación de tecnología de vanguardia; México en este campo no se ha quedado muy resagado, por lo mismo existen instituciones que se dedican a la investigación de nuevas tecnologías.

7.4.1 Desarrollos en el Instituto de Investigaciones Electricas (IIE Cuernavaca, Morelos).

En el Instituto de Investigaciones Electricas los diversos grupos involucrados en el desarrollo de Sistemas Expertos han trabajado en dos direcciones principales:

1.- El desarrollo de nuevas herramientas para construcción de Sistemas Expertos.

2.- El desarrollo de Sistemas Expertos utilizando las herramientas y lenguajes disponibles comercialmente.

A continuación se describirán brevemente los más importantes:

7.4.1.1 Creación de Nuevas Herramientas

En esta área se han desarrollado dos herramientas importantes para la construcción de Sistemas Expertos y se han desarrollado estudios sobre el procesamiento de *lenguaje natural*. Los departamentos involucrados en estos desarrollos son el Depatamento de Análisis de Redes y el Departamento de Energía Nuclear.

La herramienta desarrollada en el Departamento de Redes[1], esta ayuda para la construcción de Sistemas Expertos basada en reglas de producción. Sin embargo, estas reglas son posteriormente transportadas a su equivalente en red semántica para lograr un mejor tiempo de respuesta.

La herramienta desarrollada en el departamento de Energía Nuclear[2], esta ayuda esta orientada a los Sistemas Expertos en diagnóstico y esta basada en una novedosa extensión de las tecnicas de Análisis Probabilistico de Seguridad (APS). Este modelo hace uso de la evidencia empírica y en modelos lógicos de los sistemas bajo

---

[1] Una herramienta integrada de reglas-red para desarrollo de sistemas expertos en tiempo real. Trabajo presentado en la III Reunion de trabajo sobre Inteligencia Artificial. Oaxaca, Oax, Mexico 1990.

[2] Juan Arellano y Morrys Schwarzblat.A new approach for the development of diagnostic expert systems. ANS/CNS 6th.Pacific Basin Nuclear Conference, Beijing, China, Septiembre 1987.

consideración para construir la base de conocimientos. Esto se usa, junto con las probabilidades de falla para cada componente, para generar estrategias de diagnóstico muy eficientes. Esta herramienta permite que la adquisición del conocimiento se haga en forma rápida y sistemática, reduciendose el tiempo de desarrollo del Sistema Experto.

7.4.1.2 Aplicaciones desarrolladas

Sistema Experto en control de voltaje[3]. Es un Sistema Experto que se construyó en el departamento de Análisis de Redes sobre la herramienta desarrollada en el mismo. El Sistema Experto ayuda al operador de un sistema eléctrico de potencia a resolver problemas de control de voltaje en las etapas de planeación y operación fuera de la línea. El Sistema Experto ayuda al operador a decidir, en un tiempo relativamente corto, qué hacer ante un problema de voltaje en particular. El sistema puede hacer uso de algoritmos especializados para decidir, pero también puede usar reglas empíricas generadas con base en la experiencia del operador y/o a estudios realizados sobre una red eléctrica.

Sistema Experto auxiliar en la operación de una planta cementera. Este sistema tiene la finalidad de axiliar al personal responsable de operación y mantenimiento de una planta cementera. Este sistema fue construido en base a la herramienta desarrollada en el departamento de redes.

Sistema Experto para análisis de pruebas de presión de

---

[3] Sistema Experto en control de voltaje, D. Burciaga, L. Castillo y E. Linas G. Sc Mex 93, Mexicon 86, Guadalajara, Jal.

pozos geotérmicos[4]. Este sistema uso la herramienta desarrollada por el departamento de Energía Nuclear. Dicho sistema tiene capacidad para analizar una importante clase de pruebas de presión, en las que intervienen un sinnúmero de pozos de producción y pozos de observación en yacimientos homogeneos saturados con liquido. Del análisis se obtienen estimaciones de la transmisión y del coeficiente de almacenamiento del área del yacimiento involucrada en la prueba, y de la detección y localización de fronteras hidrológicas en dicha área. En el sistema se acoplaron modelos matemáticos, técnicas de optimización, conocimientos heurísticos y programas de graficación, por lo que el sistema es una herramienta tecnológica poderosa que proporciona a los ingenieros de yacimientos una ventajosa alternativa para el análisis de pruebas de presión.

Sistema Experto para diagnóstico de vibraciones en equipo rotatorio. Fue desarrollado en el departamento de equipos mécanicos. Este sistema tiene la finalidad de análizar vibraciones generadas por las máquinas rotatorias y con base en ello diagnostica posibles fallas.

Sistema Experto para diagnóstico de fallas de un equipo de torre de esmaltado. Este sistema se construyo con la herramienta comercial EXYS (basado para PC (Personal Computer)), fue desarrollado en el departamento de Sistemas de Información.

Sistema Experto para la asignación de recursos a las

---

[4] ANAPRES: An Expert System for interference well/test analysis. Victor Arellano Eduardo Iglesias, J.Arellano y Morris Schwarzbalat. Universidad de Stanford. Enero 1988. Este sistema fue presentado en la IV Reunion Nacional de Inteligencia Artificial en la Universidad de las Americas-Puebla. Cholula, Puebla, Mexico 1987.

nuevas subestaciones y líneas de transmisión[5]. El Departamento de servicios Técnicos de la Gerencia de Generación y Transmisión de CFE solicito al Instituto de Investigaciones Electricas la elaboración de un sistema de información computarizado para efectuar la determinación de los recursos humanos requeridos (ingenieros y técnicos) para las especialidades de: Subestaciones, Líneas, Protecciones, Control y Comunicaciones en el área de transmisión de la propia Gerencia, esto es con la finalidad de poder poner en operación y dar mantenimiento a las nuevas subestaciones que estaran en todo el pais. La deteerminación de recursos se realiza sobre el total de las instalaciones contempladas en el Programa de Obras e Inversiones del Sector Eléctrico (POISE). El sistema es capaz de generar reportes de los recursos humanos requeridos desde el nivel instalación hasta el nivel nacional, pasando por los niveles zona y región, esto para cada una de las cinco especialidades de transmisión.

7.5 Desarrollos en otras Instituciones y Universidades

Sistema Experto para diagnóstico de fallas en sistemas eléctricos de automoviles[6]. Para la programación de este sistema se usa PROLOG en micro PC. El sistema opera en forma interactiva. El usuario plantea su problema dando condiciones del automovil y através de un encadenamiento hacia adelante el sistema busca en su base de datos si tales

---

[5] I-RVP-88-AT-09 Ponencia presentada y aprovada por el cap de potencia del IEEE Seccion Mexico. Acapulco, Gro. Mexico. Agosto 1988.

[6] Proyecto presentado en la RVP de Acapulco, Guerrero, Mex. 1988. Asi mismo fue presentado en la IV Reunion Nacional de IA en la Universidad de las Americas Cuola, Cholula, Puebla, Mexico 1987. Este proyecto fue desarrollado por M. en C. Omar A. Jaran Varela (Universidad la Salle), Dr. Enrique Arce Medina (E.S.I.M.E.-Instituto Politecnico Nacional) y por el M. en C. Salvador Sanchez (E.S.I.M.E.-Instituto Politecnico Nacional).

condiciones estan tipificadas. Despues por un encadenamiento hacia atrás parte de una hipótesis y revisa aquellos hechos que la ausentan. Finalmente selecciona entre las posibles reglas aquellas que corroboran los hechos, dando como conclusión el diagnóstico de la causa de la avería y recomendando posibles acciones que la remedian. El sistema fue probado con datos del sistema eléctrico (especificaciones y operación) de automoviles Chrysler de seis cilindros, dando resultados satisfactorios.

SIEXMAN (SIstema EXperto de MANtenimiento)[7]. Este sistema se llevo acabo por requerimientos de mantenimiento de sistema de computo y para poder llevar a cabo una detección más rapida de las fallas en este equipo. Este sistema consta de dos partes: Modulo de carga de la base de conocimientos y el Modulo de ejecución del sistema (máquina de inferencias). La primera parte del sistema pide al usuario reglas o bien visualizar alguna parte de la base de conocimientos y la segunda parte evalua la base de conocimientos segun los datos proporcionados por el usuario. El lenguaje de desarrollo fué Turbo PROLOG.

Sistema Experto para la determinación de la dieta ideal, utilizado por personal subespecializado[8]. Es un sistema escrito en PROLOG. La carecteristica principal del sistema es que utiliza una base de conocimientos que se encuentra representado bajo la forma de dase de datos relacional. Se distinguen dos tipos de bases de datos una de tipo estático y otra de tipo dinámico. La última se va formando según transcurre la consulta del sistema, y la

---

[7] Sistema desarrollado en la Universidad la Salle por el Ing.Patricia Vasquez y el Ing. Juan Carlos Navarro M. Este proyecto fué presentado en la IV Reunion Nacional de Inteligencia Artificial en la Universidad de las Americas-Puebla. Cholula, Puebla, Mexico 1987.

[8] Sistema desarrollado por el Dr. Alberto Gutierrez Lopez y la Dra. P. Castro M del departamento de biomatematicas del Instituto de Investigaciones Biomedica U.N.A.M., Mexico D.F. (Marzo 1987).

primera contiene las caracteristicas de los alimentos.

Sistema Experto para elevar el rendimiento del atleta[9]. Dicho sistema aconsejará al entrenador y a los especialistas en medicina deportiva sobre las rutinas de acondicionamiento físico y los hábitos dieteticos e higiénicos a que se deberá someter el atleta. Esta compuesto por cinco modulos interrelacionados entre sí (figura 7.3).

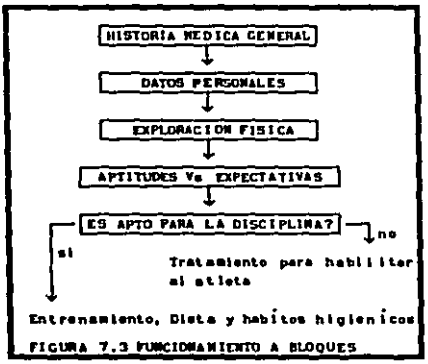


FIGURA 7.3 FUNCIONAMIENTO A BLOQUES

7.6 Un caso especial Sistemas Expertos en la industria: El caso BYTEC-AUTREY.

La creciente necesidad de nuevas soluciones a problemas cada vez más complejos en la industria han dado nuevas alternativas para los administradores y tomadores de decisiones. La Inteligencia Artificial y en particular los Sistemas Expertos, estan comenzando a entrar de lleno en la

[9] El proyecto se lleva a cabo en la Univ. la Salle y trabajan en el proyecto el M. en C. Omar E. Ierán Varela, el Dr. Asbrubal Almazan Melendez y la investigadora Adriana Martinez Delgado.

industria y los medios de producción. En paises de alto grado tecnologico son ya una realidad y año con año se invierten grandes sumas de capital por concepto de investigación y desarrollo. En México esta área de desarrollo ha llamado fuertemente la atención de Universidades y de centros de investigación, mientras que en la industria es apenas muy poco el interes mostrado en esta nueva tecnología, de hecho solo se ve como una opción alternativa de producción.

El uso de un Sistema Experto para contemplar técnicas estadisticas y de Investigación de Operaciones en el manejo de un inventario de gran volumen ha sido el problema que el grupo industrial mexicano: la Organización AUTREY. Antes de desarrollar el Sistema Experto en cuestión en 1985 BYTEC instalo una solución parcial para el manejo de inventarios al que le dio el nombre de ALMA.

Una vez determinado que las nuevas soluciones al manejo de invetarios de Organización AUTREY deberian incorporar tecnicas de Inteligencia Artificial, se ha comenzado a trabajar en ellas. El nombre interno que recibio el proyecto fue: LIMBO, un sistema que deve de tomar en cuenta una serie de factores para poder llegar a una decisión adecuada de compra de inventario en un periodo relativamente corto de tiempo (una semana). Estos factores se consideraron susceptibles de modelarse como modulos independientes dentro del sistema dadas sus caracterústicas que a continuación se describen:

Estadistica: Es necesario contar con una metodología de pronostico y modelación adecuada de la demanda de cada producto en cada filial (138,000 pronosticos semanalmente). Surge, entonces, la necesidad de crear un sistema o modulo

experto que se encargue de automatizar la tarea de pronostico. Este modulo debe contar con un fuerte apoyo algoritmico y sera capaz de emular a un estadistico vivo en el tratamiento de una serie de problemas.

Politicas Corporativas: Organizacion AUTREY cuenta con una serie de politicas corporativas que deven ser consideradas al realizar una sugerencia de compra.

Noticias: Dentro de la base de datos de productos debe guardarse información sobre hechos y factores esternos que han influido sustancialmente en las demandas: epidemias, desastres naturales, ofertas y promociones, huelgas, decisiones politicas.

Espectativas: Dadas ciertas condiciones, ¿ que es lo que un experto vivo en manejo de inventarios farmaceuticos sugeriria ? Este modulo es el que más se aproxima a la idea de experto clasificatorio y debe responder adecuadamente a la pregunta anterior, emulando el conocimiento del experto. Debe ser capaz, entre otras cosas de identificar productos sustitutos.

Retroalimentación y Control: Todos los factores mencionados anteriormente deben influir en la decisión final e interactuar entre si. Es necesario un modulo que sirva como moderador de este dialogo y que se encargue de administrar y ponderar las diferentes opiniones generadas a la vez de dirigir el proceso de consulta a las bases de datos y la interacción con los modulos no expertos del sistema.

La decisión de compra de inventario depende, como se ha visto de varios tipos de experiencia. Además, estos

conocimientos deben ser transportables de manera sencilla a las futuras aplicaciones de Inteligencia Artificial que se lleven a cabo en la Organización.

Estas consideraciones llevaron a pensar en LIMBO como un conjunto de Sistemas Expertos en *cooperación*, cada uno de ellos aportando su propia experiencia al problema e interactuando con los demás (figura 7.4). El siguiente paso en el diseño de LIMBO es determinar el tipo de herramienta a utilizar. Las caracteristicas de LIMBO hacen necesario un entorno capaz de llevar a cabo tanto una presentación adecuada de cada tipo de conocimiento como la ejecución de fuertes rutinas algoritmicas y un uso adecuado de la memoria. La estrategia seguida fue comenzar la construcción de un prototipo en OPS 5, luego se hara una evaluación para ver si es conveniente seguir sobre esta línea o migrar a algun ambiente similar.

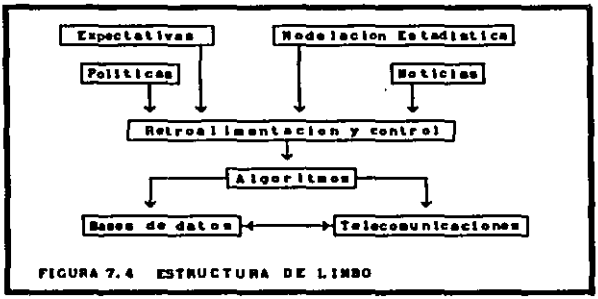


FIGURA 7.4 ESTRUCTURA DE LIMBO

CONCLUSIONES

UNA CONCLUSION ES EL LUGAR
DONDE LLEGASTE CANSADO DE
PENSAR

ANONIMO

Un sistema experto debe radiar las capacidades de un experto humano, de esta misma manera se conocen diferentes casos exitosos de sistemas expertos con esta capacidad.

Para poder construir un sistema experto con grandes posibilidades de exito se debe de contar con la capacidad de identificar y análizar el conocimiento extraido de los diferentes expertos humanos, sobre los cuales se va a definir un entorno cerrado de acción de ese experto computacional; esta capacidad es completamente independiente del método seguido en la construcción del sistema experto, porqué como ya se ha visto a lo largo de este trabajo de tesis la representación del conocimiento es una parte importantisima a considerar en la construcción de este tipo de sistemas.

Hablando un poco de los entornos de programación uno de los principales problemas a resolver para la implementación de un sistema experto es el de la integración del sistema a las herramientas diseñadas para la construcción de sistemas expertos; esto es porqué cada uno de los problemas tienen caracteristicas diferentes, diferentes campos de actuación y el dominio de aplicación puede ser muy variado o inclusive desviado de las caracteristicas del sistema. La herramienta puede estar hecha para resolver un problema muy especial y este tipo de herramientas vienen como paquetes cerrados, solo para la aplicación y no para modificaciones de éste; en un momento dado puede ser más complejo modificar el paquete que construir el sistema experto.

Las principales caracteristicas que un sistema experto debe de poseer para poder ser considerado como un verdadero sistema experto con todos los conceptos que este termino

lleva involucrados y al mismo tiempo uno de los mayores problemas en su creación son:

1.- El mecanismo de inferencia debe de estar separado de la base de conocimientos.

2.- La inferencia debe de ser heuristica.

3.- El sistema debe de ser explicativo.

4.- La experiencia de el sistema debe de ser de gran tamaño y su rendimiento alto.

5.- El sistema experto debe ser capaz de aprender.

6.- El sistema experto debe de manejar conocimiento incompleto o impreciso (sistema experto en sentido estricto).

GLOSARIO

PARA ESCRIBIR SOLO HAY QUE TENER ALGO QUE DECIR

CAMILO J. CELA

ACTIVACION DE UNA REGLA: Forma en la cual se utiliza una regla determinada. Esta regla esta enunciada en la base de *Conocimientos.*

ADQUISISCION: Forma en la cual se puede reuinir todo el *Conocimiento* necesario sobre un cierto universo.

ARBOL DE BUSQUEDA: Es la representación grafica de todas las soluciones posibles de un problema determinado.

ARQUITECTURA: Es la construcción interna de un Ordenador.

BASES DE DATOS: Es la parte de un programa experto en donde se encuentra todo el *Conocimiento* sobre el problema a resolver; esta base de datos está en espera de la habilitación por medio de un motor de inferencias.

BUSQUEDA HEURISTICA: Ideas basicas utilizadas en procedimientos inteligentes de resolución de problemas.

CALCULO DE PREDICADOS: Vease capitulo número II (pag.45).

CONCEPTUALIZACION: Abstracción del *Conocimiento* para poder llevar a cabo su representación.

CONOCIMIENTO: Es la adquisición de las nociones de las cosas por medio de la inteligencia.

DEPENDENCIAS CONCEPTUALES: Vease capitulo número II (pag.47).

EDITORES: Es un contexto de programación que permite a un usuario escribir y modificar un texto. Este tipo de contextos puede ser tan complicado como la situación lo requiera.

ENTORNOS DE PROGRAMACION: Herramientas de *Software* que asisten al programador en un contexto de programación.

ESTADO: Condición de una variable o variables ante cierta situación especifica.

ESTRATEGIA: Forma en la cual va ha ser atacado un problema por un sistema en particular.

ESTRATEGIAS DE CONTROL: Son las utilizadas para determinar en un programa la búsqueda de las soluciones.

EXPLOSION COMBINATORIA: Es un problema cuyas soluciones son infinitas en una cierta rama del árbol solución.

EXPRESION: Conjunto de terminos que representan ciertas relaciones entre si.

FACTORIAS: Se puede considerar como la fabrica del futuro, esta completamente automatizada, este tipo de fabricas tendran la capacidad de tener variaciones en los productos; asi como la implementacion de nuevos productos.

FRAMES: Es una forma de representacion del *Conocimiento*, orientada a objetos (vease capitulo numero III).

GRADO DE CERTEZA O FACTOR DE INCERTIDUMBRE: Constante que se le atribuye a una regla determinada. Factor de ocurrencia de una regla, este factor esta enunciado junto con las reglas en la base de *Conocimientos*.

GRAFICOS DE BUSQUEDA: Es una representacion grafica de las soluciones de un problema, generalmente tienen la forma de un arbol o una red.

HARDWARE: Es la parte fisica de un Ordenador (p.e. circuitos, targetas, transistores, IC, etc.).

IMPLEMENTACION: Forma en la cual se puede llevar a cavo el funcionamiento de un sistema.

INTERFAZ: Intermediario de comunicacion entre dos sistemas diferentes.

INTELIGENCIA ARTIFICIAL: Vease capitulo numero I (pag.12).

INTERPRETE DE REGLAS: Vease motor de inferencia, capitulo numero IV (pag.74).

LENGUAJE DE PROGRAMACION: Entorno de programacion que ayuda a comprender al ordenador lo que el programador quiere que haga. De una manera similar es un instrumento para poder desarrollar sistemas de una forma sencilla y eficiente.

LISP (LISt Procesing): Lenguaje de programacion del tipo funcional.

MAQUINA DE INFERENCIA: También llamada motor de inferencias. Es el encargado de controlar la resolución de un problema; esto es, propagar los datos a partir de un estado inicial, para así poder llegar a un estado final o conclusión.

METODOS DE BUSQUEDA: Vease capitulo número II (pag. 31).

MICROORDENADORES: También llamados microcomputadores. Cuentan con las partes fundamentales de un Ordenador, la diferencia principal es la escala de integración, es mucho mayor. Vease Ordenador.

MODELO: Representación de un objeto, o procedimiento para poder realizar otro con sus mismas características y elementos.

OPTIMIZACION: Vease Refinamiento.

ORDENADOR: También llamado computador. Es un sistema capaz de realizar el procesamiento de información, almacenamiento y así mismo como su utilización.

PARADIGMA: Capacidad de descripción de un universo de aplicación de un sistema determinado.

PATRONES: Modelo para poder obtener una cosa similar.

PERCEPTRON: Máquina creada en 1958, para simular la visión humana.

PROCESAMIENTO DE DATOS: Forma en la cual se ejecuta un cierto número de instrucciones dentro de un programa.

PROLOG (PROgraming LOGic): Lenguaje de programación basado en la lógica.

RAZONAMIENTO SIMBOLICO: Es un tipo de razonamiento que mediante técnicas de Inteligencia Artificial, la máquina determina la relación entre los datos.

RECONOCIMIENTO DE PATRONES: Método de identificación utilizado principalmente en sistemas de visión, para el reconocimiento de formas, aristas, siluetas y otros.

REDES SEMANTICAS: Forma de representación del *Conocimiento*. Vease capitulo número III (pag.57).

REGLAS DE INFERENCIA: Vease capitulo número IV (pags.77-83).

REGLAS DE PRODUCCION: Forma de representación del *Conocimento*. Vease capitulo número III (pag. 52).

REFINAMIENTO: Depuración de un sistema, para así obtener la efectividad plena para la cual fué diseñado.

REPRESENTACION: Forma gráfica en la cual se interpreta el *Conocimiento* de un cierto número de efectivos.

ROBOT: Máquina capaz de realizar una actividad humana mediante la programación.

SISTEMAS EXPERTOS: Vease capitulo número IV (pag. 65).

SISTEMA EXPLICATIVO: Parte fundamental de un Sistema Experto. Ayuda al usuario logre un entendimiento más profundo del sistema y viceversa.

SISTEMAS AUTOMATICOS INTELIGENTES: Son un conjunto de bloques que usando técnicas de Inteligencia Artificial pueden resolver un problema en forma autónoma.

SOFTWARE: Parte operativa de un Ordenador.

VALIDACION: Vease Implementación.

BIBLIOGRAFIA

EL UNICO LIMITE A NUESTRA REALIZACION
DEL MAÑANA SERAN NUESTRAS DUDAS DE HOY

F. D. ROOSEVELT

+ The Handbook of Artificial Intelligence.
Volumenes I y II
Avron Barr y Edward A. Feigenbaum
Addison-Wesley Publishing Company, Inc. Septiembre 1986

+ Artificial Intelligence. Tools, Techniques and Aplications.
Tim O'shea y Marc Eisenstadt
Harper y Row Publishers, New York 1984

+ Introduction to Artificial Intelligence.
Eugene Charniak y Drew McDermott
Addison-Wesley Publishing Company, Inc. 1987

+ ? Puede una máquina pensar ?
A.M. Turing (Traducción de Manuel Garrido y Amador Anton)
Depto. de Lógica y Filosofía de la Ciencia
Universidad de Valencia, 1974

+ Método y Programas (MP). Inteligencia Artificial en Medicina (Sistemas Expertos).
M. Fieschi
Masson S.A., Barcelona

+ Introducción a LISP
H. Farreny (Versión castellana de Eric Farreny, Carbona)
Masson S.A., Barcelona-México 1986

+ Mentes y Máquinas
Alan Ross Anderson, A.M. Turing, entre otros
(Traducción de Karl Wendl, revisada por Agusti Bartrm)
UNAM, México 1970

- **Memorias: IV Reunión Nacional de Inteligencia Artificial**
  Universidad de las Americas-Puebla
  Cholula, Puebla, México 1987

- **Inteligencia Artificial: Conceptos, Técnicas y Aplicaciones**
  Serie: Mundo Electronico
  Por varios Autores bajo la coordinación de José Monpin Poblet, Director de la revista Mundo Electronico.
  Marcombo: Boixareu Editores, Barcelona-México 1987

- **Memorias: Primera Reunión de Verano de Potencia.**
  Volumen de Sistemas de Potencia
  Volumen de Alta Tensión
  IEEE Sección México, Acapulco, México 1988

- **Seminario de Sistemas Expertos**
  Alvaro A. Calvo Meneses, G. Margarita Solis Alfaro,Edwin Garro Molina, Fabio Eduardo Muñoz Jiménez y José Ronald Argüello V.
  Escuela de Ciencias de la Computación e Informatica
  Facultad de Ingenieria,
  Universidad de Costa Rica, Abril 1989

- **Tesis: Sistema Experto de Mantenimiento.**
  Juan Carlos Navarro
  Universidad La Salle, Diciembre 1988

- **Memorias: VI Reunión de Inteligencia Artificial**
  Universidad Autonoma de Queretaro e Instituto Tecnológico de Queretaro (Campus Queretaro), Qro, Qro, México 1989.