UNIVERSIDAD AUTONOMA DE GUADALAJARA

Escuela de Ingeniería en Computación

"INTRODUCCION A LA INTELIGENCIA ARTIFICIAL Y A LOS SISTEMAS EXPERTOS"

TESIS PROFESIONAL

QUE PRESENTA:

MARCO ANTONIO PACHECO VELAZQUEZ

PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

GUADALAJARA, JAL., 1990.

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

I N D I C E

INTRODUCCION

INTRODUCCION

Hace tan sólo veinte años, el computador era para la mayoria de la gente un aparato enigmático, una especie de "genio en botella" capaz de llevar al hombre a la Luna, o de almacenar informacion detallada sobre todos los ciudadanos de un pais. Se escuchaban cosas muy espectaculares y tambien espeluznantes acerca de las computadoras. Hoy, la situación ha cambiado radicalmente.

De ello es responsable la introducción y expansion, a lo largo de la decada de los ochentas, de las computadoras personales y domésticas, la cual ha permitido al hombre común familiarizarse con la computación. Aunque sea indirectamente (quién no tiene un familiar o un amigo trabajando directamente con una computadora), todos sabemos a estas alturas lo que es una computadora: Una maquina con circuitos logicos a la que se puede ordenar la realizacion de ciertas tareas mediante un programa, el cual es un conjunto de instrucciones que especifican cómo realizar una o varias operaciones. Definitivamente ha dejado de ser aquella misteriosa e incompresible máquina electrónica que hasta hace pocos años unos cuantos iniciados estaban en condiciones de utilizar.

Esta popularización de las computadoras, realmente

espectacular por la velocidad con que ha tenido lugar, ha generado un significativo fenómeno. Si antes se les tenía cierto respeto, a las puertas de la última década del siglo XX, la computación personal despierta enormes espectativas, deseos e ilusiones, en muchas ocasiones imposibles, todavía, de verse cumplidos. ¿ Cuáles son estos ideales ? ¿ Cuántos han sido ya satisfechos y cuántos lo serán en un futuro próximo? En definitiva, ¿ Cómo será y que le podremos pedir a la computadora de los ya inmediatos años noventas ?

Uno de los sueños que más se arraigaron cuando comenzaron a venderse las primeras computadoras de uso personal es el de disponer de una máquina diligente capaz de trabajar, en exclusiva para uno mismo o para una empresa como un docil sirviente. Pero no tardó mucho en desaparecer esa idea de la cabeza de quienes pensaban así. Ya que era necesario tener un poco de conocimiento previo en cuanto al manejo de dicha máquina. Así es como empezaron a comprender la necesidad de desterrar la idea de que la computadora trabajaría automáticamente. El interior de la máquina (El hardware), debería alojar una materia blanda intangible (El software), si se deseaba convertir un simple paquete de circuitos integrados en el talento universal que habían soñado.

En aquella época se hizo famoso el término "cerebro electrónico" para describir las sobresalientes

capacidades de las nuevas y misteriosas máquinas, un concepto que despertaba en muchos un temor: si la computadora podría pensar tan inteligentemente o más que el hombre, entonces también podría algún día arrebatarle su trabajo intelectual.

Ya en los años cincuentas, los investigadores de la Inteligencia Artificial aseguraban que no pasaría mucho tiempo hasta que una computadora pudiera establecer por sí sola teoremas matemáticos, batir al campeón del mundo de ajedrez o incluso traducir idiomas automáticamente. Sin embargo, tales predicciones aún no se han realizado. Lo más que se ha conseguido en el transcurso de éstos años es la simulación de ciertos procesos formalizables del pensamiento humano aplicados a tareas concretas. El pensamiento como tal queda, todavía fuera del alcance de los chips, por potentes que estos se fabriquen.

Una variante simplificada de los modelos nueronales, tambien perteneciente al campo de la inteligencia artificial, son los llamados "Sistemas Expertos", que emulan los conocimientos y experiencia de especialistas humanos en distintos campos de la ciencia y la técnica muy concretos. Esto con el fin de proponer soluciones a problemas complejos. La arquitectura de estos programas de ayuda al profesional, común a todas las aplicaciones, permite desarrollar "peritos

electrónicos" para prácticamente cualquier especialidad: diagnóstico médico y mecánico, control de procesos fabriles, prospecciones geologicas, previsiones metereológicas, investigaciones sociológicas, gestión financiera y de empresas, y un largo etcétera.

El creciente aluvión de datos procedentes de todos los confines del mundo hace difícil tomar decisiones acertadas en un tiempo prudencial; son muchos los ejecutivos que se ven irremediablemente sobrecargados a la hora de organizar y valorar la información. Por eso utilizan cada vez mas la computadora para obtener una prespectiva general de la evolucion medioambiental, de la situación coyuntural o de los efectos de un nuevo medicamento. Con ello se conjura el miedo de perder la visión de conjunto, pero a menudo tambien se delega, inconscientemente, parte de la responsabilidad a la máquina.

Si por ejemplo, alimentamos la computadora con grandes cantidades de datos sobre los efectos de la contaminación atmosférica, y le hacemos combinar esas informaciones siguiendo determinadas reglas, podremos, por lo general, remitirnos a los resultados con toda confianza. Lo que hemos hecho ha sido simular la toma de decisiones para observar si las consecuencias serán positivas o negativas en el mundo real. De modo que trasladamos a la computadora la responsabilidad de

elegir el camino correcto.

Pero por muy útil que sea una simulacion por computadora para resolver algunos problemas, tampoco está totalmente exenta de riesgo. Cada vez más decisiones, que deberiamos tomar los humanos, se forjan en el interior electronico de una computadora. Desde luego en confiabilidad no hay quien la supere. Retiene todos los datos que se le introduzcan, puede fusionar inteligentemente los conocimientos separados de humanos expertos, puede analizar todas las combinaciones y posibilidades de un problema, puede mantener actualizada su informacion mediante la interaccion de humanos expertos, etcétera.

Vamos a explorar los Sistemas Expertos en detalle . A fines de 1970 los sistemas expertos y la ingenieria del conocimiento necesaria para construirlos habian desarrollado algunas caracteristicas básicas comunes.

Los sistemas basados en el saber se inclinan fuertemente hacia lo aplicado, porque se caracterizan por grandes cantidades del saber especifico.

El proyecto que inició el desplazamiento de la inteligencia artificial hacia una perspectiva basada en el saber fue DENTRAL, un sistema experto que permitió deducir estructuras químicas a partir de los datos disponibles de los quimicofisicos. El proyecto inicio en 1965.

DENTRAL, a pesar de su potencia y utilidad evidente, no fue aclamado de modo intensivo, inmediato y universal.

¿Qué es en definitiva un Sistema Experto? Es un programa de computadora dotado del saber y de la capacidad que le permita actuar a nivel de un experto. Un nivel de un experto significa, por ejemplo, el nivel de un Doctor en medicina cuando diagnostica o prescribe tratamientos, o de un Doctor en ciencias o de personas con mucha experiencia que llevan a cabo tareas de Ingenieria, Cientificas o de direccion empresarial. El Sistema Experto es un apoyo intelectual de alto nivel para el especialista humano, lo cual explica el otro nombre que se le da: Ayudante Inteligente.

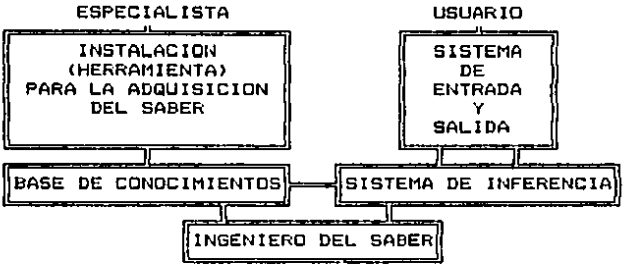
Los Sistemas Expertos se suelen construir de modo que pueden explicar las lineas de razonamiento que les hicieron tomar tal o cual decision. Algunos incluso explicar porque rechazaron algunas lineas de razonamiento y escogieron otras. Esta transparencia es un elemento importante en los Sistemas Expertos. Los diseñadores de los Sistemas Expertos se esfuerzan mucho en conseguirlo porque saben que la utilización de un Sistema Experto dependerá en último extremo de la confianza que inspire a sus usuarios, y la confianza se basará en que el conocimiento sea transparente, explicable.

Los Sistemas Expertos actúan especialmente bien cuando la actividad mental se basa en razonar, no en calcular, y la mayoría de trabajos en el mundo son de este tipo. Una gran cantidad de trabajos profesionales parecen a través de formulas matemáticas, pero de hecho si se exceptúan las ciencias de base matemáticas, las elecciones difíciles, las cuestiones que distinguen a un especialistas de un principiante son simbolicas, deductivas y basadas en el saber experimental. Los expertos humanos han conseguido sus conocimientos no solo en el saber explícito de libros de texto y clases, sino también mediante la experiencia: repitiendo las cosas una y otra vez, fallando, acertando, gastando tiempo y fuerzas, luego aprendiendo a ahorrarlo al captar intuitivamente un problema , sabiendo cuándo conviene seguir el libro y cuándo hay que dejar las reglas al lado. De este modo se construye un repertorio de reglas empiricas útiles, o heuristicas, que combinado con el conocimiento sacado de los libros, convierte a la persona en un especialista experto.

El saber es el elemento clave en el desempeño de un sistema experto. Este saber es de dos tipos. El primer tipo incluye los hechos del dominio: los conocimientos compartidos por todos, en los que todos los especialistas están de acuerdo y que figuran en escritos, libros de textos y en las revistas

especializadas o que forman la base de las clases de un profesor. Tiene una importancia igual para la práctica de un campo del segundo tipo de saber, llamado saber o conocimiento heurístico, que es el conocimiento que se adquiere por medio de la práctica y del enjuiciamiento correcto. Es el saber experimental, el arte de hacer suposiciones acertadas, que un humano experto adquiere después de años de dedicación.

Un sistema experto, además del saber necesita un procedimiento de inferencia, o sea, un método de razonamiento que permita comprender y actuar sobre la combinación de conocimientos y de datos propios del problema. Los procedimientos de inferencia o métodos para la solución de problemas que utilizan los ingenieros del conocimiento no tienen que ser arcanos o complejos. Sirven incluso métodos sencillos, utilizados en los razonamientos de sentido común o enseñados en los primeros cursos de lógica. De hecho, utilizar procedimientos de inferencia sencillos tiene sus ventajas, porque los usuarios finales podrán comprenderlos fácilmente cuando repasen las líneas de razonamiento del sistema. Los usuarios finales solo confiarán en el razonamiento de un sistema experto y por tanto, solo lo usarán si pueden comprender fácilmente lo que este sistema hace. He aquí la estructura base de un sistema experto:



En la elaboración de esta tesis : expondre qué es un programa inteligente, que son los sistemas expertos, los diferentes mecanismos de inferencia, elegir uno de los diferentes mecanismos de inferencia, explicar cada uno de ellos y las diferentes formas en que pueden trabajar, y diseñare un sistema experto de proposito general para la comprensión más fácil de su utilidad en los diferentes campos que se deseen utilizar. A traves del desarrollo de esta Tesis, mostraré mis habilidades como Ingeniero en Computación, tomando como herramientas los conocimientos adquiridos en el transcurso de mis estudios en la Universidad Autonoma de Guadalajara.

El tema de tesis será: "Introduccion a la inteligencia artificial y a los sistemas expertos".

El cuerpo de la tesis está compuesto por:

En el diseño del sistema experto de propósito general usaremos el lenguaje de programación lógica

PROLOG, por ser el lenguaje oficial en el desarrollo de la quinta generación de computadoras.

ANTECEDENTES

A N T E C E D E N T E S

Es difícil establecer una fecha de comienzo exacta para lo que es comúnmente llamado inteligencia artificial. Quizá el crédito por el nacimiento debería darse a A.M. Turing por su invención de la computadora de programa almacenado. Las primeras computadoras fueron concretamente máquinas dedicadas que literalmente tenían que ser recableadas para resolver diferentes problemas. El hallazgo de Turing de que un programa podría ser almacenado como dato en la memoria de la computadora y ejecutado más tarde como la base para todas las computadoras modernas. El almacenamiento de programas permitía cambiar la función de la computadora rapida y fácilmente mediante la simple ejecución de un nuevo programa. Esta capacidad implica que una computadora podría ser capaz de cambiar su propia función (i esto es, aprender a pensar !).

Sin embargo, lo que se conoce comunmente como inteligencia artificial comenzo, cuando en el MIT, John McCarthy creo el LISP (uno de los lenguajes de inteligencia artificial). El LISP (List Processing) tiene ciertas semejanzas con el PROLOG.

El término inteligencia artificial se imputa generalmente a Marvin Minsky, tambien del MIT, quien en 1961 escribió un artículo titulado " Pasos hacia la

Inteligencia Artificial", en enero de 1961. Los años sesenta fueron un período de intenso optimismo sobre la posibilidad de hacer pensar a las computadoras. Despues de todo, fue la decada que vio la primera computadora que jugó al ajedrez, las primeras demostraciones matematicas por computadora y el famoso (y altamente publicitado) programa ELIZA, escrito en 1964 por Joseph Weizenbaum del MIT. ELIZA era un programa que actuaba como un psicoanalista Rogeriano. En este estilo de análisis, el psiquiatra toma un papel pasivo, generalmente repitiendo simplemente las propias respuestas del paciente más que dirigiendo la conversación. En aquella epoca, el programa causó bastante conmoción. La gente se preguntaba : " ¿ Deberían usarse las computadoras de esta manera ? ¿ Es posible que un psiquiatra computarizado pueda ser mejor que el psiquiatra humano ?
¿ Deberia permitirse esto ? " incluso el creador de ELIZA, escribió el libro " Poder de Computadora y Razón Humana", en 1976, ¡escencialmente para desacreditar su propio programa ! Los años sesenta fueron también un periodo de intenso temor a la automatización. A veces es duro comprender las emociones que existían hace sólo veinte años.

Debido al aparente éxito de la inteligencia artificial, parecia que el objetivo de producir un

programa que tenga inteligencia como la humana estaba justo a la vuelta de la esquina. sin embargo esto no iba a ser así.

Lo que no fue claramente comprendido en los años sesenta fue la dificultad de generalizar estos exitos concretos dentro de un programa. Un problema común es que según los programadores intentaban incrementar la generalidad de ciertos programas, estos intentos requerián mayores recursos computacionales de los que existian en ese entonces. Por ejemplo, la memoria se acababa rapidamente o el tiempo de ejecución se volvia demasiado lento.

Por la mitad de los años sesenta, fueron comunes las computadoras con grandes memorias y las velocidades de cálculo se habián incrementado espectacularmente. Sin embargo, con estas mejoras, muchos de los antiguos enfoques de la inteligencia artificial continuaban fallando debido a que habián ineficiencias inherentes. Para comprender esto, considere una tarea tan simple como ordenar un arreglo de números. Si se usa el método de la burbuja, el tiempo de ordenacion será proporcional a N^2, donde N es el número de elementos. Esto significa que si ordenar un arreglo de 10 elementos lleva un segundo, entonces ordenar un arreglo de 100 elementos llevara cien segundos, y así sucesivamente. Puede ver que, en algún momento, hay un arreglo que llevara tanto tiempo ordenar

como la vida promedio de una persona. No importa lo rápida que llegue a ser una computadora, un algoritmo cuyo tiempo es proporcional a N llegará a ser lento. Un mejor paso es cambiar la ordenación del algoritmo para que sea más eficiente, por ejemplo, mediante la ordenación rápida (Quick Sort) que es proporcional a N .

Al final de los años sesenta, se habían alcanzado varios éxitos (tales como el procesamiento de lenguaje natural), y la resolución de problemas en áreas específicas de la inteligencia artificial. Así se estableció la base para la introducción de primer producto comercial, el sistema experto. Uno de los primeros sistemas expertos fue el MYCIN, el cual fue desarrollado en la Universidad de Stanford y fue creado para ayudar a los médicos a diagnosticar enfermedades.

Uno de los sucesos más importantes de la inteligencia artificial que ocurrieron en los años sesenta pasó prácticamente desapercibido en los Estados Unidos hasta los años ochenta. Este suceso fue la creación del PROLOG (Programación Lógica), en 1972 por Alain Colmerauer en la Universidad de Marsella, en Francia. Como LISP, PROLOG es un lenguaje que fue diseñado para ayudar a resolver problemas relacionados con la inteligencia artificial.

QUE ES UN PROGRAMA INTELIGENTE?

C A P I T U L O I

¿ Que es un programa inteligente ?

Explicar o comprender lo que es un programa inteligente y como se diferencia un programa inteligente de los programas no inteligentes, es bastante complicado; ya que existe mucho debate sobre este tema.

Determinar que es un programa inteligente implica que se conoce lo que significa inteligencia. Inteligencia es " La capacidad para percibir hechos , proposiciones y sus relaciones, razonando sobre ellos." Escencialmente, en este contexto significa pensar. Se sugirio hace mucho tiempo que nadie puede explicar cómo piensa una persona, pero todo el mundo puede decir lo que piensa. El hecho es que nadie comprende realmente cómo piensa una persona.

Si se sigue con una interpretación estricta de la definición de inteligencia, entonces podria defenderse que todos los programas son inteligentes. Por ejemplo: La primera parte de la definición de inteligencia es la habilidad de percibir hechos, proposiciones y relaciones. Actualmente, las computadoras están asombrosamente bien adaptadas para realizar este tipo de tareas. Ciertamente, algunos tipos de información, tales como imagenes visuales, son mucho, mas dificiles de percibir para una computadora que para otros, pero la definición de inteligencia no exige que la percepción

tome una forma especifica. Por tanto, lo que las computadoras hacen normalmente satisface el primer requisito de inteligencia.

Sin embargo, ¿ Puede una base de datos razonar sobre estos hechos, lo cual es el segundo criterio de inteligencia ? La respuesta depende de lo que acepte como definición de razonar. Si acepta la manipulación de información de la base de datos (el acto de buscar, clasificar, procesar preguntas, archivar y similares) como razonar, entonces debe decir que la base de datos es un programa inteligente. Esto implica que la mayoria de los programas de computadoras manipulan informacion en forma razonable y logica. Por lo tanto, esta forma de razonar debe clasificarse como inteligencia.

De algún modo, esta conclusion es dificil de aceptar. Implica que virtualmente todos los programas pertenecen al campo de la inteligencia artificial. Implicación que no es cierta. Si intentamos justificar la capacidad para aceptar que una base de datos relacional es un programa pensante, podriamos decir que no puede ser un programa pensante porque lo que hace un programa de bases de datos no es solo análogo a pensar en la forma en que la gente piensa. Pero entonces se enfrenta con el hecho de que si la misma tarea la realiza un archivador, obviamente demanda inteligencia por parte del archivador. Por lo tanto, si una base de

datos realiza la misma tarea, entonces no está pensando; no obstante, si una persona realiza la tarea, entonces esta persona está pensando. Como ser humano, nos gusta pensar que es el cerebro lo que nos separa de los animales, que tenemos el monopolio sobre el pensamiento cognocitivo, podemos permitir que los más altos mamiferos puedan pensar pensamientos simples e incluso razonar en un nivel primitivo, mientras que nosotros los humanos vamos más alla.

Veamos este problema de otra forma. Uno podria decir que un perro bien adiestrado es inteligente si lleva a su dueño el periódico desde el auto. Podriamos incluso decir que un niño o una niña de un año es bastante inteligente si realiza la misma tarea. Sin embargo, la mayoria de la gente no estaria inclinada a decir que el robot era inteligente simplemente porque podria traer el periódico. La razón de esto es que la mayoria de la gente diria que el robot que trae el periódico es simplemente una màquina que ejecuta un programa creado por alguien en alguna parte, y que el robot no esta pensando sobre la tarea, sino simplemente esta realizandola, u otros dirian concretamente que el programador del robot esta pensando (a distancia en tiempo y espacio).

El ejemplo del robot y el periódico vuelve sobre el hecho de que la gente no sabe cómo piensa, la tendencia

es decir que el programa no puede ser inteligente porque es comprensible. Esto es llamado principio magico : en un nivel emocional, la mayoria de la gente opina que pensar es análogo a la magia. Debido a que ellos no comprenden los procesos del pensamiento, tienden a hacer la suposicion errónea de que cualquier dispositivo que es hecho y comprendido por los humanos no puede ser inteligente, porque los humanos lo comprenden. La mayoria de la gente piensa que la creación es menos que el creador.

Tambien está la problemática de libre albedrio. De hecho, Descartes, el filosofoso del siglo XVII, proclamó: "Pienso, luego existo". ¿ Puede una computadora, que aparentemente es un dispositivo deterministico, elegir alguna vez hacer algo ? No hay duda de que esta cuestión se convertirá en una de las mayores conclusiones filosoficas y legales del siglo XXI. Esta cuestión puede polarizar rapidamente a gran cantidad de programadores. Hay programadores que afirman que "una máquina es una máquina": No puede tener libre albredio porque no tiene mente, solo circuitos. El argumento de los otros programadores puede ser más persuasivo. Para comprender esto suponga que una computadora esta controlando la carga de un camión en un patio de ladrillos. Cuando alcanza cierto punto, la computadora detiene la cinta transportadora de

ladrillos. ¿ Decidió la computadora parar la carga ? ¡Si! claramente estaba en el control de la situacion y eligió parar la carga cuando el camión alcanzo cierto punto. Si la computadora no tomó la desicion, entonces ¿ Quién lo hizo ?

Mucha gente está convencida de que es imposible hacer un programa inteligente. Primero, la computadora es un dispositivo determinístico. Segundo, hay cuestiones de autonomía. Tercero, como poca gente conoce el proceso del pensamiento, ¿ Como puede alguien hacer a una computadora pensar ? Al igual que la conclusión de que todos los programas son inteligentes no podía establecerse, esta conclusion tampoco es satisfactoria.

Las dificultades que se ilustran en la discusión recién dada hacen alusión concreta a un fallo en la definición de inteligencia. Lo que la definición del diccionario omitia era el hecho de que el termino inteligencia implica inteligencia humana. Esta asociación implica y hace difícil admitir la posibilidad de que una máquina pueda pensar o un programa ejecutándose en una computadora pueda ser inteligente porque la mayoria de los programas no hacen la misma tarea de la misma manera que una persona. Pero, si se elimina esta implicación, es fácil decir que todos los programas son inteligentes.

Por lo tanto la definición de programa inteligente se hace directa:

"Un programa inteligente exhibe un comportamiento similar al de un humano cuando se enfrenta a un problema similar. No es necesario que el programa resuelva concretamente, o intente resolver, el problema de la misma forma que un humano."

Observese , que el programa no necesita pensar como un humano, pero debe actuar como tal.

Por lo tanto, los programas inteligentes, de alguna manera, exhiben un comportamiento inteligente similar al de un humano, mientras que los programas no inteligentes, no lo hacen.

SISTEMAS

EXPERTOS

CAPITULO II

SISTEMAS EXPERTOS

Los sistemas expertos son interesantes principalmente por dos razones: primera, los programas que se realizan son prácticos y útiles que satisfacen las necesidades de los humanos; y segunda, dichos programas se pueden realizar.

II.1.-¿ QUE ES UN SISTEMA EXPERTO ?

Los sistemas expertos son programas que imitan el comportamiento humano : Un humano experto, es una persona con conocimientos especiales y experiencia en una área determinada. Los humanos expertos responden a las necesidades de conocimientos específicos de ciencia, tecnología, organización , etc.. Los sistemas expertos usan la informacion que se encuentra en las bases de datos que son suministradas por el usuario para devolver una opinion sobre un tema especifico. De este modo, el sistema experto le pide que responda preguntas hasta que pueda identificar un objeto que se adecúa a sus respuestas. Para compreder lo que realiza un sistema experto de una forma sencilla considere el siguiente dialogo entre un sistema experto y un humano que esta buscando un consejo:

EXPERTO: ¿ Es verde?

USUARIO: No

EXPERTO: ¿ Es rojo?

USUARIO: Si

EXPERTO: ¿ Crece en los árboles?

USUARIO: No

EXPERTO: ¿ Crece en caña?

USUARIO: Si

EXPERTO: ¿ Tienen espinas las cañas ?

USUARIO: Si

EXPERTO: Es una frambuesa.

El objetivo de un sistema experto de fruta computarizada, es ser capaz de reproducir este dialogo. Más generalmente, un sistema experto intenta dar un consejo o una respuesta al usuario en el tema en el cual es experto.

II.2.-¿ CUALES SON LAS VENTAJAS DE UN SISTEMA EXPERTO ?

La utilidad de un sistema experto se basa principalmente en la eficiencia y conveniencia. A diferencia de un humano experto que esta expuesto a condiciones subjetivas: Tiene que dormir, comer, descansar, tomarse unas vacaciones, y otras cosas, el sistema experto está accesible las 24 horas del día durante todo el año. Además, se pueden crear muchos sistemas expertos , mientras que el numero de humanos expertos puede ser limitado, lo cual hace virtualmente imposible en muchas situaciones tener un humano experto disponible cuando se necesita. Ademas a diferencia de

los humanos, los sistemas expertos nunca mueren. EL programa de un sistema experto puede copiarse y almacenarse fácilmente.

Una ventaja menos importante de un sistema experto sobre los humanos expertos es su falta de personalidad. No todas las personalidades son compatibles. Si no se lleva bien con un humano experto, puede tener reservas para utilizar el conocimiento de un experto. La situación contraria también es cierta: Un humano experto a quien no le gusta usted puede no ser capaz de proporcionarle información fiable. Pero un sistema experto no tiene personalidad, así se eliminarían los problemas o prejuicios.

Una última ventaja de un sistema experto es que después de tener un experto computarizado, puede adquirir un nuevo sistema experto simplemente copiando el programa de una computadora a otra. Un humano necesita largo tiempo para convertirse en un experto en cierto campo , lo cual hace difícil adquirir nuevos humanos expertos.

II.3.-ALGUNOS EJEMPLOS DE SISTEMAS EXPERTOS COMERCIALES.

Si no fuera por un sistema experto llamado MYCIN, los sistemas expertos podrían haber permanecido en el laboratorio de investigación de la inteligencia artificial. Y no haber progresado dentro del mundo

exterior. Uno de los mayores problemas de la inteligencia artificial ha sido que mucha gente, creia que las tecnicas de la inteligencia artificial sólo funcionaban con problemas simples que requerian un conjunto estricto de reglas e hipótesis. Esa gente creia que nunca podria usarse la inteligencia artificial para resolver problemas dificiles. MYCIN cambió todo esto.

MYCIN fue el primer sistema experto exitoso en el mundo. Desarrollado en la Universidad de Stanford a la mitad de los años sesenta, MYCIN fue diseñado para ayudar a los médicos en el diagnostico de cierta enfermedades bacterianas. Diagnosticar las enfermedades significaba escencialmente comparar los síntomas exhibidos por el paciente con los síntomas de una enfermedad hasta que se encuentre una similitud. El problema es que para un médico es dificil diagnosticar todas las numerosas enfermedades que existen rápidamente y con seguridad. Por consiguiente, MYCIN satisfaria la necesidad de confirmar los diagnosticos de los medicos.

Otro ejemplo de sistema experto viable es PROSPECTOR, el cual fue desarrollado en 1978 por Richard Duda, Peter Hard y Rene Reboh. PROSPECTOR es un sistema experto en geologia ; predice la verosimilitud de que pueden encontrarse depósitos de ciertos materiales minerales en una región en particular. Ha habido variaciones de este sistema, incluyendo sistemas que son

expertos en predecir descubrimientos de petroleo, gas natural y helio.

El comienzo de los años ochentas vio la introducción de los sistemas expertos que podian proporcionar consultoría de impuestos, consejos sobre seguros y asistencia legal. Se cree que a final de los años noventa habrá un gran mercado de sistemas expertos personales para usar en casa y en la oficina. Estos sistemas cubrirán muchas materias.

II.4.- COMO TRABAJAN LOS SISTEMAS EXPERTOS.

Todo sistema experto tiene dos partes: La base de conocimientos y el mecanismo de inferencia.

La base de conocimiento es una base de datos que toma información y reglas especificas sobre cierta materia. He aquí dos términos que deberiamos conocer para la siguiente explicacion:

-Objeto: Conclusion que se define por sus reglas asociadas.

-Atributo: Una cualidad especifica que, junto con su regla, ayuda a definir un objeto.

En consecuencia, puede pensar en la base de conocimientos como en una lista de objetos con sus reglas y atributos asociados.

Para la mayoría de las aplicaciones de los sistemas expertos, puede definirse un objeto como una lista de

atributos que el objeto posee o no. De este modo, para la mayoría de los propósitos la regla que se aplica a un atributo establece que el objeto "tiene" el atributo. Por ejemplo un sistema experto en frutas podría tener una base de datos como esta:

MANZANA:

TIENE	Crece en los árboles.
TIENE	Forma redonda.
NO TIENE	Crece muy al sur.
TIENE	Color rojo o amarillo.

UVA:

TIENE	Crece en los árboles.
TIENE	Tamaño pequeño.
TIENE	Color púrpura.
NO TIENE	Espinas.
TIENE	Sirve para hacer vino.

NARANJA:

TIENE	Crece en los árboles.
TIENE	Forma redonda.
NO TIENE	Crece en el norte.
TIENE	Color naranja.

Si piensa en esta base de conocimientos, puede ver que puede simplificarla: Puede usar sólo una regla "TIENE" y usar la forma negativa del atributo si debe establecer una relación de "NO TIENE". De este modo, la regla se convierte simplemente en "POSEE", y la base de

datos simplificada aparece así:

MANZANA:

Crece en los árboles.

Redonda.

No crece muy al sur.

El color es rojo o amarillo.

UVA:

Crece en los árboles.

De tamaño pequeño.

Varios colores.

Las vides no tienen espinas.

NARANJA:

Crece en los árboles.

Redondo.

No puede crecer en el norte.

El color es naranja.

Aunque ciertos sistemas expertos sofisticados pueden necesitar una regla más compleja que simplemente "POSEE", está es suficiente en muchas ocasiones y simplifica en gran medida la base de conocimiento.

II.5.- EL MECANISMO DE INFERENCIA.

Es la parte del sistema experto que intenta usar la información que se le da para encontrar un objeto. Hay dos categorías claras de mecanismos de inferencia: Determinísticos y probabilísticos. Para tratar de

comprender esto imaginemos que hay dos expertos, uno en química y el otro en sociología. El experto químico puede decir con certeza que si el átomo en cuestión tiene dos electrones, entonces es un átomo de helio. No hay duda acerca de la identidad del átomo porque el número de electrones determina el tipo del elemento. Sin embargo, si preguntamos al experto sociólogo cual es la mejor forma de prevenir las salidas del bachillerato por parte de los estudiantes, el sociólogo dará una respuesta que está calificada como ser sólo probable o como que tiene una cierta probabilidad de exito. En consecuencia, la respuesta es verdadera, pero incierta.

La mayoria de las disciplinas no son deterministicas, sino probabilisticas en cierta medida. Sin embargo,para muchas de estas, la incertidumbre no es estadisticamente importante por lo que se pueden tratar como situaciones deterministicas.

Más alla de estas dos categorias claras de incertidumbre y certidumbre, hay tres formas basicas de construir el mecanismo de inferencia: encadenamiento hacia adelante, encadenamiento hacia atrás y valor de la regla. Las diferencias de los métodos se refieren a cómo intenta el mecanismo alcanzar su objetivo.

II.5.1.-ENCADENAMIENTO HACIA ADELANTE.

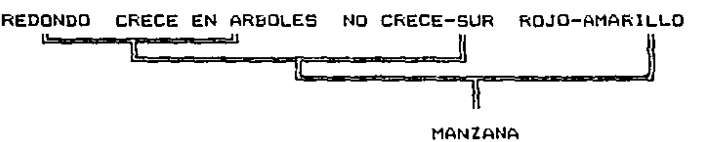
El encadenamiento hacia adelante es llamado algunas

veces, guiado por los datos (DATA-DRIVEN) porque el mecanismo de inferencia usa información proporcionada por el usuario para moverse a traves de la red hasta alcanzar un punto terminal, el cual es un objetivo.Si el mecanismo de inferencia no puede encontrar un objeto usando la información existente, entonces pide más. Las reglas que definen el objeto crean el camino que conduce a él, por consiguiente, la única forma de alcanzar el objeto es satisfacer todas las reglas. De esta forma, un mecanismo de inferencia de encadenamiento hacia adelante comienza con alguna información y trata de encontrar el objeto que satisface dicha informacion.

Para comprender como trabaja el encadenamiento hacia adelante, imaginemos que mi coche esta funcionando mal y telefoneo al mecánico, que es el experto en este caso, para que me de una opinión sobre el problema. Entonces el mecanico me pide que descubra lo que esta mal. Yo explico que mi coche tiene una perdida de potencia, pega tirones, el motor sigue funcionando después que se abrió el switch de encendido y que no lo he afinado en varios meses. Usando esta información, el mecánico me dice que lo más cierto es que mi coche necesita una correción de ciertas partes para que quede bien.

Recordemos la base de conocimiento de la fruta descrita anteriormente para crear un diagrama que

muestre cómo alcanzaría el objeto manzana un mecanismo de inferencia por encadenamiento hacia adelante, cuando se le da los atributos propios como se muestra a continuación:



Como podemos ver, un sistema de encadenamiento hacia adelante esencialmente construye un árbol desde las hojas hasta la raíz.

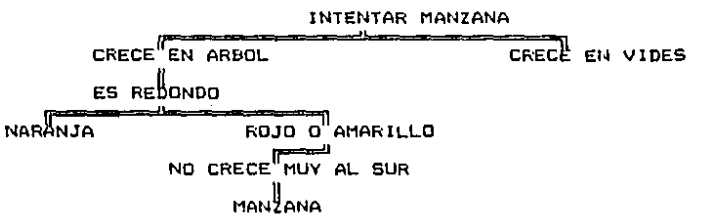
II.5.2.-ENCADENAMIENTO HACIA ATRAS.

El encadenamiento hacia atrás es el opuesto al encadenamiento hacia adelante: Un mecanismo de inferencia por encadenamiento hacia atrás comienza con una hipótesis (Un objeto) y pide la información para confirmarlo o negarlo. El encadenamiento hacia atras es llamado a veces guiado por los objetos (Object-Driven) porque el sistema empieza con un objeto e intenta verificarlo.

Para comprender como trabaja el encadenamiento hacia atrás, imagine que la computadora para de trabajar repentinamente. La primera hipotesis que formulariamos es que se ha cortado el suministro de energía electrica.

Para comprobar esto, escuchamos que el ventilador de la computadora sigue trabajando. Oyendolo funcionar, rechazamos está hipótesis y procedemos a otra.La hipótesis número dos es que la computadora se ha estropeado debido a un defecto en hardware. Para confirmar o rechazar esta hipótesis, damos un reset a la computadora para ver si se inicializa con exito. Desafortunadamente, la computadora no inicializa; la segunda hiposis era cierta.

Si la fruta en cuestion es manzana, aplicar inferencias de encadenamiento hacia atrás en la base de conocimientos de fruta produce el diagrama que se muestra a continuación:



Como se muestra en el diagrama, el encadenamiento hacia atrás poda un árbol. Esto es lo opuesto al encadenamiento hacia adelante, el cual construye el árbol.

II.5.3.-METODO DEL VALOR DE LA REGLA.

El mecanismo de inferencia del valor de la regla es superior teoricamente tanto a un sistema de

encadenamiento hacia adelante como a uno de encadenamiento hacia atrás porque requiere de información que tiene mayor importancia, dado el estado actual del sistema. Es comúnmente un mecanismo de encadenamiento hacia atrás mejorado. La teoria general de operacion es que el sistema requiere como su próxima pieza de información la que suprimirá la mayor incertidumbre del sistema.

Para comprender el enfoque del valor de la regla, imagine que ha llamado al médico porque su hijo está enfermo. El doctor primero pregunta si el niño tiene fiebre porque la respuesta a esta pregunta estrecha el mayor número de probabilidades. Si se responde si a la primera pregunta, entonces el doctor hará preguntas que tengan mayor impacto dado el estado actual. Este procedimiento se continuará hasta que el doctor haya hecho su diagnóstico. En este ejemplo, el punto clave es que el doctor selecciona cada pregunta para procesar más rápidamente la información hasta llegar a la conclusión.

El problema de los sistemas del valor de la regla es que son dificiles de implementar. Hay dos razones para esto: primero, en la vida real las bases de conocimientos es a menudo tan grande que el numero de combinaciones posibles excede lo que puede ser abarcado fácilmente. Por lo tanto, el sistema no puede determinar simplemente que información elimina la mayor

incertidumbre en cualquier estado dado. Segundo, los sistemas con valor de la regla requieren que la base de datos contenga no solo la información estandar objeto-atributo, sino tambien un cuantificador de valor que hace mas difícil la construccion de la base de conocimientos. Sin embargo, hay ciertas situaciones que conducen por si mismas a inferencias basadas en el valor de la regla más que otras. Tambien, cuando son implementados, los sistemas con valor de la regla generalmente hacen un trabajo mejor que los otros dos métodos.

II.6.-ELECCION DE UN METODO.

¿Cuál es el mejor método? La solucion a esta pregunta se basa en parte a la preferencia personal, porque los tres metodos haran la tarea que se quiera hacer. Como se estableció anteriormente los sistemas del valor de la regla son difíciles de implementar por la complejidad que ello encierra, pero lo explicaremos en el momento de hacer referencia al metodo de encadenamiento hacia atras.

El metodo de encadenamiento hacia adelante hace más fácil el proceso de derivar la mayor cantidad de información de la base de conocimientos porque construye un árbol, ya que encuentran comunmente todos los objetos posibles que encajan con los atributos. Pero existen

varios problemas: conocer como esta escrito cada uno de los atributos del objeto, que es nuestro dato de entrada en el mecanismo de inferencia, haber tenido un error de ortografía al momento de escribir el dato, o no conocer todos los atributos que contiene dicho objeto. A la larga esto puede ser tedioso, ya que el uso no correcto de la base de conocimiento nos estaria dando una redundancia no controlada. Supongamos la siguiente información de una base de datos de un sistema experto de frutas:

OBJETO	ATRIBUTO
Manzana	crece_en_árboles, roja_o_amarilla, redonda
Naranja	naranja, crece_en_árboles, redonda
Uva	púrpura, crece_en_vides, no_espinas
Frambuesa	crece_en_cañas, roja, espinas
Pera	crece_en_arboles, roja_o_verde, no_redonda
Sandia	crece_en_vides, no_espinas, verde, grande
Cereza	roja_o_amarilla, crece_en_árboles, pequeña
Mandarina	piel_suave,redonda,crece_en_arboles,naranja.

Nuestro objetivo es encontrar un objeto que encaje con los siguientes atributos:

- Crece en los arboles.
- Verde.
- No redomda.

Problema uno: El atributo numero un no encaja con ninguno de los establecidos en la base de datos del

sistema experto, ya que contiene la palabra "los", y el guión de subrallado no se encuentra el dato de entrada.

Problema dos: El atributo verde se encuentra en la base de datos del sistema experto una sola vez en el objeto: sandia, pero no crece en arboles.

Problema tres: Todos los atributos se encuentran con minusculas.

Problema cuatro: El atributo numero tres tiene un error de ortografia.

Problema cinco: El objeto buscado es: Pera. pero no se cumple el atributo roja_o_verde.

Esto seria facil de solucionar teniendo en cuenta que las personas que vayan a utilizar el sistema experto deben ser personas bien capacitadas en el uso de una computadora, y que sean ellas mismas las que esten atendiendo dicho sistema todo el tiempo, ademas deben ser personas que conozcan la base de datos de pies a cabeza, y en este ejemplo la base de datos es demasiado pequeña, pero supongamos que estemos trabajando con bases de datos demasiado grandes.

Estos son solo unos de los problemas que se podria encontrar, pero pueden existir mucho más, que irian apareciendo durante el uso del sistema experto.

A mi parecer el mejor método es el de encadenamiento hacia atrás porque requiere solo información suficiente para encontrar un objeto. debido

a que los sistemas de encadenamiento hacia atras son guiados por el objetivo (goal-driven), ya que comienza con una hipoteis (un objeto) y pide la informacion para confirmarlo o negarlo.

En el nivel màs primitivo el mecanismo de inferencia comienza suponiendo que el objeto 1 es el objetivo y trata de confirmar esto preguntando si el objetivo tiene los atributos del objeto y. Si tiene todos los atributos, entonces el mecanismo de inferencia manifiesta que el objeto 1 es la respuesta. De lo contrario, pasa a tomar el objeto 2 como el objetivo, y realiza los pasos anteriores de forma similar que el objeto 1; y asi, consecutivamente hasta encontrar el objetivo. Esto permite que entre en el sistema información relevante. Un sistema de encadenamiento hacia atràs es útil cuando se requiere encontrar un objeto. Este método es màs fácil de implementar y produce un sistema experto que parece lo que usted espera que parezca. Supongamos la información que se encuentra en la base de datos del ejemplo anterior.

```
Es/tiene/hace        redonda    s
Es/tiene/hace        rojo_o_amarillo     n
Es/tiene/hace        crece_en_arboles    s
Es/tiene/hace        naranja    s
naranja encaja con los atributos
¿ Continuar ?  s
```

Es/tiene/hace piel_suave s

mandarina encaja con los atributos

¿ Continuar ? n.

Esta es una manera más sencilla para llegar a la solución, además de que Turbo Prolog nos ofrece la característica de encadenamiento hacia atras.

DISENO DE UN SISTEMA EXPERTO DE PROPOSITO GENERAL

CAPITULO III

DISEÑO DE UN SISTEMA EXPERTO DE PROPOSITO GENERAL

En este capitulo diseñaremos un sistema experto de proposito general que usa el metodo de encadenamiento hacia atras. Es llamado de proposito general porque permite usar el mismo mecanismo de inferencia con distintas bases de datos. Incluiremos algunas rutinas para crear las bases de información.

Antes de ver como escribir el codigo del programa para un sistema experto, veremos lo necesario para crear el mecanismo de inferencia. Supongamos que la base de conocimientos consistira solo de objeto y sus atributos, usando una pequeña base de conocimientos como se muestra a continuación:

OBJETO	ATRIBUTOS
1	A,B,C
2	A,B,Y
3	B,X
4	A,B,D

En el nivel más primitivo el mecanismo de inferencia comienza suponiendo que el objeto 1 es el objetivo y trata de confirmar esto preguntando si el objetivo tiene los atributos del objeto y. Si tiene todos los atributos, entonces el mecanismo de inferencia manifiesta que el objeto 1 es la respuesta. De lo contrario, pasa a tomar el objeto 2 como el objetivo, y

realiza los pasos anteriores de forma similar que el objeto 1; y así, consecutivamente hasta encontrar el objetivo. Este proceso se repite hasta que se encuentre el objeto apropiado, o bien no hay mas objetos. Supongamos que el objetivo es el objeto 4, tendrá lugar el siguiente diálogo:

EXPERTO: ¿Tiene A?

USUARIO: Si

EXPERTO: ¿Tiene B?

USUARIO: Si

EXPERTO: ¿Tiene C?

USUARIO: No

EXPERTO: ¿Tiene A?

USUARIO: Si

EXPERTO: ¿Tiene B?

USUARIO: Si

EXPERTO: ¿Tiene Y?

USUARIO: No

EXPERTO: ¿Tiene B?

USUARIO: Si

EXPERTO: ¿Tiene X?

USUARIO: NO

EXPERTO: ¿Tiene A?

USUARIO: Si

EXPERTO: ¿Tiene B?

USUARIO: Si

EXPERTO: ¿Tiene D?

USUARIO: Si

EXPERTO: Es el cuatro.

Trabajar con un sistema experto como éste no solo sería tedioso de usar, sino que no mereciera la pena en la mayoría de los casos cuando hubiera más de un puñado de objetos en el sistema. Este tipo de expertos tiene dos defectos: primero, preguntas sobre el mismo atributo varias veces, segundo hace una pregunta innecesaria, porque en el objeto 3 el atributo principal no tiene A sino B, el mecanismo de inferencia será más eficiente si pasa por alto todos los objetos que no se ajusten al estado actual. Lo que se necesita es un mecanismo de inferencia que, cuando se da la misma base de conocimientos con el objeto 4 como objetivo, produzca el siguiente diálogo:

EXPERTO: ¿Tiene A?

USUARIO: Si

EXPERTO: ¿Tiene B?

USUARIO: Si

EXPERTO: ¿Tiene C?

USUARIO: No

EXPERTO: ¿Tiene Y?

USUARIO: No

EXPERTO: ¿Tiene D?

USUARIO: Si

EXPERTO: Es el cuatro.

Con este tipo de mecanismo de inferencia como objetivo, las especificaciones que deben conformar el mecanismo de inferencia debe tener los siguientes pasos:

1.- El sistema experto no debe preguntar dos veces por el mismo atributo.

2.- El sistema experto rechazaría inmediatamente y pasaría por alto cualquier objeto que no tenga los atributos necesarios de conocimiento conocidos.

3.- Si se le ordena, el sistema experto debería ser capaz de manifestar porqué está siguiendo una línea de razonamiento.

La tercera especificación se añade no solo como una forma de verificar que el sistema experto está operando correctamente, sino como un metodo de educar al usuario.

III.1.- ESTRUCTURACION DE LA BASE DE CONOCIMIENTOS.

El primer paso para crear un sistema experto es definir la estructura de la base de conocimiento. En turbo prolog definiremos la base de conocimientos de la siguiente manera:

```
database
    info(symbol,lista)
    si(symbol)
    no(symbol)
```

III.2.- CARGA DE LA BASE DE CONOCIMIENTO.

El siguiente programa podria escribirse por separado del sistema experto, pero la mayoría de los sistemas expertos comerciales lo harian de la siguiente manera, por simplicidad de las rutinas. Introducir el nombre de un objeto y llama a atributos, que se encarga de introducir los atributos que se asocian con este objeto:

```
introducir:-
   write(" ¿Qué es ? : "),
   readln(OBJETC),
   OBJETC <> "Fin"
        atributos(Objetc,[]),
   write(" ¿ Más ? "), !,
   readln(Q),Q=s,
   intoducir.

atributos(O,List):-
   write(O,"Es/tiene/hace : "),
   readln(attribute),
        attribute <> "Fin"
   añadir (attribute,List,List2),
   atributos(O,List2).
atributos(O,List):-
   assert(info(O,List)),
   escribirlista(List,L),!,
   nl.
```

Debido a que el usuario sólo debería tener que introducir una vez la informacion en la base de conocimiento, al final de cada seccion de entrada, otra rutina salvará la informacion usando la orden SAVE del turbo prolog. Por simplicidad, este sistema experto supondrá siempre que la base de conocimiento esta contenida en el archivo X.dat. Bajo este supuesto, el primer predicado ejecutado será Start, el cual se muestra a continuación:

```
Start:-
   consult("X.dat")
   fail.
Start:-
   assert(si(end)),
   assert(no(end)),
   write("¿Introducir información ? (S/N): "),
   readln(A),
   A=s,
   not(introducir),
   save("X.dat"),!,
   preguntar([]).
Start:-
   preguntar([]), purgar.
Start:-
   purgar.
```

Esta es la forma en que trabaja START. La primera

cláusula comunica al sistema que lea en la base de datos X.DAT y luego falla. Esto fuerza a turbo prolog a intentar la segunda clausula. Puede pensar que no hay motivo para incluir la primera cláusula porque podria simplemente situar consult("X.DAT") como primera línea de la segunda cláusula. Sin embargo, lo malo de esto es que la primera vez que el sistema se ejecute, no existirá X.DAT. Esto significa que consult fallará. Así, si consult fuera parte de la segunda parte de la segunda cláusula, las lineas de código después de consult nunca se ejecutarian.

Los dos assert con que comienza la segunda cláusula se usan para inicializar las dos bases de datos de soporte. Después, el programa preguntará si el usuario desea introducir más, o nueva información. Si el usuario responde si, entonces se ejecuta el predicado introducir. Después del regreso de introducir, la rutina salva la información en el archivo del disco y llama al predicado de alto nivel del mecanismo de inferencia preguntar. Si el usuario elige no introducir más información, entonces la segunda cláusula de start falla y la cláusula final ejecuta una llamada a preguntar.

III.3.- CONSTRUCCION DEL MECANISMO DE INFERENCIA.

El mecanismo de inferencia es la fuerza conductora del sistema experto. Aunque en sentido abstracto es

bastante simple, implementar un mecanismo de inferencia puede ser bastante difícil.

El predicado de alto nivel del mecanismo de inferencia es preguntar y se muestra a continuación:

```
preguntar:-
   info(O,A),
   not(miembro(O,L),
añadir(O,L,L2),
   anterioressi(A),
   anterioresno(A),
   intentar(O,A),
   !, write(o,"Satisface los "),nl,
   write(" atributos actuales "),nl,
   write( " ¿ Continuar (S/N) ? "),
   readln(Q), Q=s,
   preguntar (L2).
```

Preguntar está basada en las capacidades de la vuelta atrás propia del turbo prolog. Preguntar trabaja como sigue. Primero, pregunta a la base de conocimientos para ligar un objeto y sus atributos a O y A. Suponiendo que haya algo en la base de datos, este paso tendrá exito. Luego prueba no encontrar el mismo dato dos veces, y luego, las rutinas anterioressi y anterioresno prueban los objetos que no se encuentran en el estado actual del sistema: anterioressi comprueba la lista de atributos de O para asegurarse de que contiene todos los

atributos que el usuario ha comunicado al sistema que debe tener el objeto; y anterioresno la comprueba para asegurarse de que entre los atributos de O no se encuentra ninguno que haya sido rechazado. Estos chequeos satisfacen las especificaciones primera y segunda, que se dieron anteriormente.

El proceso entero parece bastante facil, pero falta ver las rutinas de soporte. Primero, he aqui anterioressi y anterioresno:

```
anterioressi(A):-
   si(T),!,
   xanterioressi(T,A,[]),!.

xanterioressi(end,_,_):- !.
xanterioressi(T,A,L):-
   miembro(T,A),!,
   aNadir(T,L,L2),
   si(X), not(miembro(X,L2),!,
   xanterioressi(X,A,L2).
anterioresno(A):-
   no(T),
   xanterioresno(T,A,[]),!.
xanterioresno(end,_,_):-!.
xanterioresno(T,A,L):-
   not(miembro(T,A)),!,
   aNadir(T,L,L2),
```

```
   no(X), not(miembro(X,L2)),!,
   xanterioresno(X,A,L2).
```

Los predicados anterioressi y anterioresno se usan simplemente para enviar llamadas a sus respectivas rutinas de soporte. Estos predicados ayudan a restringuir la vuelta a atrás del turbo prolog. Además, el corte (!) se usa ligeramente para impedir la vuelta atras a varios puntos. El predicado soporte miembro determina cuándo un elemento es parte de la lista.

La porción del mecanismo de inferencia que interactua con el usuario es preguntar. Cada vez que se encuentra un objeto que encaja con el conjunto actual de hechos, preguntar imprime el nombre del objeto y pregunta al usuario si deberia continuar. Si se teclea s, entonces el sistema llama a preguntar recursivamente. La lista que pasa a preguntar forza a turbo prolog a encontrar objetos que no han sido examinados todavia en la base de conocimientos.

Intentar no devuelve una solución, sino solamente encuentra algo se falso o verdadero.

```
intentar(_,[]).
intentar(O,[X|T]):-
   si(X),!,
   intentar(O,T).
intentar(O,[X,T]):-
   write(" Es/tiene?hace ",X, " " ),
```

```
    readln(Q),
    procesar(O,X,Q),!,
    intentar(O,T).
```

Intentar extrae un atributo cada vez de la lista de atributos del objeto y pregunta al usuario si este atributo particular es parte del objeto objetivo. La segunda clausula se usa para impedir preguntas redundantes. Debido a que el sistema experto es de proposito general, el mensaje de preguntar es tan general como es posible.

La porción intentar usa el predicado de soporte procesar, que toma diferentes acciones basadas en la respuesta del usuario. Procesar responde imprimiendo el objeto que se esta buscando actualmente y la lista actual de los atributos, con la why.

III.4.- UN PROGRAMA DE UN SISTEMA EXPERTO.

```
domains
    lista=symbol*
database
    info(symbol,Lista)
    si(symbol)
    no(symbol)
predicates
    juntar(lista,lista,lista)
    escribirlista(lista,integer)
    introducir
```

```
    purgar
    añadir(symbol.lista,lista)
    preguntar(lista)
atributos(symbol,lista)
    procesar(symbol,symbol,symbol)
    start
    anterioressi(lista)
    anterioresno(lista)
    xanterioressi(symbol,lista,lista)
    xanterioresno(symbol,lista,lista)
    intentar(symbol,lista)
    miembro(symbol,lista)
    xwrite(symbol)
goal
    start
clauses
Start:-
    consult("X.dat")
    fail.
Start:-
  . assert(si(end)),
    assert(no(end)),
     write("¿Introducir información ? (S/N): "),
  readln(A),
 A=s,
 not(introducir),
```

```
 save("X.dat"),!,
 preguntar([]).
Start:-
      preguntar([]), purgar.
   Start:-
      purgar.
introducir:-
       write(" ¿Que es ? : "),
       readln(OBJETC),
       OBJETC <> "Fin"
         atributos(Objetc,[]),
       write(" ¿ Más ? "), !,
       readln(Q),Q=s,
       introducir.
    atributos(O,List):-
       write(O,"Es/tiene/hace : "),
       readln(attribute),
         attributo <> "Fin"
       anadir (attribute,List,List2),
       atributos(O,List2).
     atributos(O,List):-
        assert(info(O,List)),
        escribirlista(List,L),!,
        nl.
     anadir(X,L,[X,L]).
     preguntar:-
```

```
        info(O,A),
        not(miembro(O,L),
        añadir(O,L,L2),
        anterioressi(A),
        anterioresno(A),
        intentar(O,A),
        !, write(o,"Satisface los "),nl,
        write(" atributos actuales "),nl,
        write( " ¿ Continuar (S/N) ? "),
        readln(Q), Q=s,
    preguntar (L2).
    anterioressi(A):-
        si(T),!,
    xanterioressi(T,A,[]),!.
    xanterioressi(end,_,_):- !.
    xanterioressi(T,A,L):-
        miembro(T,A),!,
        añadir(T,L,L2),
        si(X), not(miembro(X,L2),!,
        xanterioressi(X,A,L2).

    anterioresno(A):-
        no(T),
        xanterioresno(T,A,[]),!.
    xanterioresno(end,_,_):-!.
    xanterioresno(T,A,L):-
```

```
        not(miembro(T,A)),!,
        añadir(T,L,L2),
        no(X), not(miembro(X,L2)),!,
        xanterioresno(X,A,L2).
  intentar(_,[]).
  intentar(O,[X|T]):-
        si(X),!,
        intentar(O,T).
  intentar(O,[X,T]):-
        write(" Es/tiene?hace ",X, " " ),
        readln(Q),
        procesar(O,X,Q),!,
        intentar(O,T).
procesar(_,X,s):-
  assert(si(X)),!.
procesar(_,X,n):-
  assert(no(X)),!,fail.
procesar(O,X,why):-
  write(" Creo que puede ser : "),nl,
  write(O,"porque tiene : "),nl,
  si(z),xwrite(Z),nl,
  Z="end",!,
  write("Es/tiene/hace ",X,"?"),
  readln(Q),
  procesar(O,X,Q),!.
xwrite(end).
```

```
xwrite(Z):-
    write(Z).
purgar:-
    retract(si(X)),
    X=end,fail.
purgar:-
    retract(no(X)),
    X=end.
juntar([],List,List).
juntar([X|L1],List2,[X|L3]) if
    juntar(L1,List2,L3).
escribirlista([],_).
escribirlista([Head|Tail],3):-
    write(Head),nl,escribirlista(Tail,1).
escribirlista([Head|Tail],I):-
    N=I+1,
    write(Head," "), escribirlista(Tail,N).
miembro(N,[N|_]).
miembro(N,[_,T]):-
    miembro(N,T).
```

III.5.- EJECUCION DE UN EJEMPLO.

Una vez introducido el programa del sistema experto en una computadora, introduzca la siguiente información a la base de conocimientos:

OBJETO	ATRIBUTO
Manzana	crece_en_árboles, roja_o_amarilla, redonda
Naranja	naranja, crece_en_árboles, redonda
Uva	púrpura, crece_en_vides, no_espinas
Frambuesa	crece_en_cañas, roja, espinas
Pera	crece_en_árboles, roja_o_verde, no_redonda
Sandia	crece_en_vides, no_espinas, verde, grande
Cereza	roja_o_amarilla, crece_en_árboles, pequeña
Mandarina	piel_suave,redonda,crece_en_arboles,naranja

Con el sistema dotado de conocimiento, podremos usarlo para identificar diferentes tipos de frutas. He aqui la ejecucion del programa:

```
Es/tiene/hace        redonda    s
Es/tiene/hace        rojo_o_amarillo      n
Es/tiene/hace        crece_en_árboles     s
Es/tiene/hace        naranja    s
naranja encaja con los atributos
¿ Continuar ?  s
Es/tiene/hace        piel_suave      s
mandarina encaja con los atributos
¿ Continuar ? n.
```

CONCLUSIONES

CONCLUSIONES

Hoy en día, la computadora es uno de los instrumentos más importantes de los ingenieros y demas especialistas, y se esta volviendo igualmente importante en otros campos. Ha dejado de ser ese aparato enigmático, para convertirse en algo más familiar, pero no ha dejado de ser un misterio para la gran mayoria de personas con las que compartimos diariamente.

Es cierto que la mayoria de las personas saben que es una computadora, por medio de ideas fijas que se han creado ha cerca de ella, ya que solo han conocido ciertas areas en la que podemos hacer uso de ella, por ejemplo: Las terminales de los bancos, las terminales de las agencias de viajes, los cajeros automaticos, etcetera. Se creia en los años pasados que una computadora podra realizar el trabajo de una persona casi automáticamente al momento de instalarla, sin poner en sus circuitos toda la información necesaria para realizar esa tarea.

Existe todavia el temor : si la máquina podria pensar tan inteligentemente o más que un hombre, entonces tambien podria algun dia arrebatarle su trabajo intelectual. Dicho temor seguira existiendo hasta el momento en el cual la humanidad tenga la posibilidad de saber con certeza que es una computadora, cuales son sus ventaja y desventajas.

La construcción de un sistema experto, ofrece la oportunidad de acercarse al ser humano y descubrir la simplicidad y la naturalidad con que se enfrenta a ciertas situaciones, y a la complejidad para obtener ese comportamiento en un computador. Una reflexión puede conducir a reconocer, lo sublime de la inteligencia y lo grandioso de aquel, que dió tal facultad al ser humano.

La Inteligencia Artificial constituye actualmente una disciplina, que aun en su pequeño pero creciente desarrollo, ofrece a todos métodos por los cuales permiten hacer frente, en forma eficiente y productiva, a problemas no solo en investigación básica y desarrollo, sino tambien a nivel de campo real en diversos ambitos profesionales e industriales.

Los métodos de la Inteligencia Artificial extiende la capacidad del computador hacia el procesamiento simbolico dando lugar, en última instancia, por una parte a que la persona recurra al computador con una nueva actitud mental y por otra a una nueva forma de emplear el mismo en la solucion de problemas.

La Ingenieria de Conocimiento es la disciplina cuyo objetivo fundamental es la construccion, desarrollo, aplicacion y mantenimiento de sistemas basados en el saber.

El Ingeniero del Conocimiento debe consultar a humanos expertos, con la finalidad de investigar que

conocimiento especifico y que criterios se utilizarán para resolver los problemas propios de su campo. Así mismo es capaz de transferir el conocimiento obtenido hacia un computador.

Un Sistema Experto es un conjunto de programas que posee amplios y profundos conocimientos en un determinado campo, y que es capaz de asegurar en la solucion de problemas en ese campo.

Un Sistema Experto debe ser capaz de :

- Conversar.
- Preguntar.
- Responder.
- Sugerir.
- Concluir.
- y Justificar tanto las sugerencias como las conclusiones.

Pueden haber Sistemas Expertos para diferentes tipo o categorias:

- Diagnosis.
- Busqueda.
- Resolución de problemas y planeacion.
- Instruccion.
- Etcetera.

Cuando un humano experto hace frente aun problema, recurre tanto a principios y teorias generales, como a experiencias y criterios particulares.

Uno de los objetivos del Ingeniero del Conocimiento es transferir el saber profundo y superficial hacia un computador constituyendo de esta forma, la base de datos del Sistema Experto.

La expresion de Inteligencia Artificial la empleo por primera vez J. McCarthy, en un conferencia de Dartmouth College, New Jersey en 1956. Y suele considerarse que nace como disciplina academica la Inteligencia Artificial, en la misma conferencia de Dartmouth.

El sistema experto desarrollado en el capitulo 3 comienza su busqueda con la primera entrada en la base de conocimientos y procede simplemente en orden secuencial a lo largo de la lista de objetos. Este proceso está bien para pequeñas cantidades de objetos, ya que cuando se usan grandes bases de conocimientos podrian causar problemas. Por ejemplo, si hubieran mil objetos, y el que se describe es el 968 y tiene muchos atributos comunes, entonces a este tipo de experto le podria llevar mucho tiempo para encontrar la solución. Intentar resolver este problema, junto con otros conduce a un campo llamado : Ingenieria del conocimiento, descrita anteriormente.

Aunque esta mas alla del proposito de esta tesis, presentar una discusion completa sobre este tema, es importante estar atento a las dificultades que puede

encontrar cuando se intente construir una base de conocimiento. El area de la ingenieria del conocimiento es completamente nueva y todavia hay mucho por aprender.

Una buena forma de organizar la informacion es situar cerca de la base de concimientos aquellos atributos que causan la poda de la mayor parte del árbol. Para las bases de datos grandes, de nuevo puede ser dificil determinar que atributos causan el mayor efecto.

Rara vez se puede encontrar un humano experto y al cual se pueda extraer los conocimientos; primero, es dificil determinar quien es o no un verdadero experto. Ese problema es especial cuando se conoce poco sobre la materia. Segundo, dos expertos diferentes en la misma materia tienen a menudo opiniones contradictorias, lo que hace dificil saber cual usar.

Además, la mayoria de los humanos expertos no saben que saben; los humanos no pueden hacer un mapeo de memoria, de la misma forma que una computadora. Por consiguiente puede ser dificil extraer la informacion necesaria.

Las bases de conocimientos no son lo mismo que las bases de datos, aunque se afirme a menudo lo contrario, ya que la base de conocimientos son hechos, hipotesis, creencias y heuristicas; métodos para tratar con la base de datos a fin de conseguir los resultados deseados.

como un diagnóstico, o una interpretacion o una solucion al problema; y la base de datos es un conjunto de datos sobre objetos y acontecimientos sobre el cual trabajará la base de conocimientos para conseguir los resultados deseados.

BIBLIOGRAFIA

BIBLIOGRAFIA

Leigh, William E.,

Prolog to expert systems.

Santa Cruz, California, U.S.A.:

Mitchel Publishing, inc., 1987.

Feigenbaum, Edward A. y McCorduck, Pamela,

La quinta generacion.

Barcelona España.:

Editorial Planeta, S.A., 1984.

Schildt, Herbert,

Advanced Turbo Prolog.

Santa Cruz, California, U.S.A.:

Mc Graw Hill, Inc., 1987.

Garnham, Alan,

Artificial Intelligence.

New York, New York, U.S.A.:

Routledge & Kegan Paul, 1988.