# UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

Facultad de Contaduría y Administración

# SISTEMAS EXPERTOS, SU FUNCIONAMIENTO

**SEMINARIO DE INVESTIGACION INFORMATICA**

Que para obtener el título de

LICENCIADO EN INFORMATICA

p r e s e n t a

**REBECA RODRIGUEZ RAMIREZ**

Asesor del Seminario: L.I. Concepción Camargo Fajardo

México, D. F. 1995

A mis padres:

Por su apoyo incondicional, comprensión.
Y sobre todo, por su gran amor.

A Ray:

Por ser mi hermano menor.

A Miguel Angel:

Por su apoyo, cariño y gran paciencia.

# INDICE

## Capítulo 3

**Representación del conocimiento**



## Capítulo 4



## Capítulo 5

**Herramientas para el diseño de sistemas expertos**

## Anexo

# INTRODUCCION

El hombre siempre ha tenido la necesidad de inventar instrumentos que faciliten su trabajo, así a través del tiempo ha ido creando herramientas u objetos que le han permitido mejorar las propias condiciones de vida. Desde que el hombre existe, se ha destacado de los demás seres vivos por una curiosidad sin límites, curiosidad que le ha llevado a buscar las respuestas a toda la serie de fenómenos naturales que le han rodeado por siempre.

La serie de acontecimientos históricos que marcan el proceso evolutivo de la humanidad están ligados, invariablemente, con los inventos y descubrimientos que han cambiado la manera de ser y de pensar de generaciones enteras. Quien puede negar el enorme valor que tuvo para el hombre prehistórico el descubrimiento del fuego, o quien puede concebir el inconmensurable impacto que tuvo en la vida del ser humano el invento de la rueda.

Y así, podríamos hacer un recuento de los inventos y descubrimientos de la humanidad, maravillándonos con cada uno de ellos y con los beneficios obtenidos a partir de la portentosa creatividad de hombres y mujeres dotados con inteligencias superiores. Pero no es el propósito del presente trabajo.

Centremos nuestra atención en una pequeña porción de ese vasto universo del conocimiento humano: la computadora.

El ser humano siempre se ha preocupado por calcular. En un principio se limitaba a calcular el número de bienes que poseía el y su propia comunidad, pero el crecimiento de las sociedades hacia formas de organización más complejas hizo a su vez más complejos los cálculos y creo nuevas necesidades en el hombre. La aparición de la computadora como una medio rápido y eficaz para realizar complejas operaciones de cálculo matemático y para procesar grandes volúmenes de información fue producto de esta necesidad.

Si bien la computación en sus inicios tan sólo simplificó procesos largos y tediosos y redujo el tiempo requerido en la obtención de soluciones, también se subdividió en diversas disciplinas que se abocarón a la tarea de investigar las amplias y diferentes posibilidades de aplicación que tenía la nueva ciencia en varios de los campos del conocimiento humano.

Una de esas disciplinas dio la pauta para que se iniciaran diferentes investigaciones tanto en el campo del procesamiento de información como en el área de la investigación de los procesos de pensamiento y razonamiento humano.

Ante la evidente importancia de la computación como herramienta para procesar todo tipo de información, algunos científicos empezaron tratar de adaptar en ella sus investigaciones con el ambicioso objetivo de lograr que una maquina simule el tipo de razonamiento humano, esta disciplina de la computación es conocida con el nombre de Inteligencia Artificial.

Los primeros desarrollos generados en esta área del conocimiento fueron los relacionados a redes neuronales con las cuales se intentó generar inteligencia imitando la transmisión de pulsos eléctricos de neurona en neurona como sucede en el cerebro para generar información, por medio de los componentes electrónicos de la computadora.

Algunos investigadores que no encontraron gran desarrollo en esta área, dirigieron sus estudios hacia la búsqueda de nuevos métodos que aun cuando no utilizaran como base el funcionamiento del cerebro humano si pudieran llegar a los mismos resultados, ya no de una manera global sino en objetivos particulares, creando de esta manera otras áreas de desarrollo para la inteligencia artificial como lo son la robótica, lenguaje natural, visión por computadora, sistemas expertos, etc.

En los sistemas expertos también llamados sistemas basados en el conocimiento, se ha encontrado un amplio campo de desarrollo. En estos se acumula el conocimiento de uno o varios expertos en una materia específica y se intenta con base en mecanismos internos del sistema lograr que en un problema dado en su área de interés obtener una solución satisfactoria,

tal como un experto lo haría. En la actualidad se han desarrollados sistemas de este tipo que han apoyado diferentes áreas del conocimiento como lo son la industria, finanzas, juegos, etc. alcanzando su mayor desarrollo en la medicina.

La forma en que pueden ser diseñados los sistemas expertos depende en gran parte del objetivo del mismo, del tipo y tamaño de información que manejará.

Así como la inteligencia artificial es una ciencia relativamente nueva con un amplio campo de investigación, el área de los sistemas expertos al ser una rama que se desprende de ella, no contiene aun métodos ya determinados y únicos para su diseño.

La investigación de técnicas que permitan un mayor desarrollo para los sistemas expertos y un mejor funcionamiento aun está en sus inicios, mejorando las técnicas ya existentes o creando otros métodos con la aparición de cada nuevo sistema experto.

En este trabajo se expondrán algunas de las técnicas más utilizadas en el diseño de sistemas expertos, para ello se han subdividido en sus componentes básicos, como lo son la forma en que se representará el conocimiento, las reglas que deben seguirse y la búsqueda de soluciones, lo que permitirá una mejor comprensión de su funcionamiento. Es importante señalar que las técnicas aquí mostradas no son las únicas que existen pero si algunas de las más utilizadas.

El trabajo inicia con una explicación de lo que es la inteligencia artificial, su historia y algunas ramas que la componen, para pasar después a el área en que se desarrollará este tema, los sistemas expertos y su funcionamiento, los diferentes elementos que los componen y la relación que existe entre los mismos para lograr el objetivo que persiguen. Como han evolucionado hasta llegar a ser una importante herramienta en el desarrollo y apoyo de diferentes áreas en la actualidad.

Y finalmente algunas de las herramientas que pueden ser utilizadas para diseñar y conjuntar todos los elementos antes mencionados, como lo son los lenguajes de programación especialmen-

te diseñados para generar aplicaciones basadas en inteligencia artificial y los generadores de sistemas expertos, que son herramientas altamente sofisticadas. De ambos se comentarán las características más importantes.

# CAPITULO 1

## Inteligencia artificial, conceptos básicos

### 1.1 La historia de la inteligencia artificial

Charles Babbage, nacido en 1792 es considerado el padre de la informática moderna por el desarrollo de las primeras máquinas capaces de procesar información, la máquina diferencial (Difference Engine) y la máquina analítica (Analytical Engine).

Años después el Dr. Herman Hollerith diseñó un tabulador como un medio para llevar a cabo el censo norteamericano de 1890, fue la primer máquina de computación que usó medios no mecánicos de proceso. Su enfoque era hacer que la presencia de una perforación en una tarjeta generara corriente eléctrica que hacía avanzar un paso a un contador, fue comercializado posteriormente por IBM.

En 1935, Konrad Zuse introdujo la aritmética binaria en su computadora Z1, una máquina totalmente mecánica. Su modelo posterior el Z2, utilizaba resistencias electromecánicas en sustitución de los conmutadores mecánicos y cintas de papel perforado como entrada. El IBM Mark 1, presentado en 1943 también utilizaba resistencias electromagnéticas.

En los años cuarenta aparecen las computadoras consideradas de la primera generación. La ENIAC (Electronic Numerical Integrator And Calculator) en esta computadora se eliminaron todas las partes en movimiento y se sustituyeron con bulbos (18 000) los cuales eran activados con impulsos eléctricos. Esta máquina estaba dirigida a la computación de tablas balísticas de cañones y misiles. En ella se instalaba el proceso a seguir en la máquina a definiendo las

posiciones de los interruptores y las conexiones eléctricas. Para poder modificar el proceso que la máquina iba a realizar era necesaria la participaban ingenieros especializados y llevaba acabo en varios días.

En 1945 John Von Neumann comenzó el diseño de la EDVAC (Electronic Discrete Variable Automatic Computer). Por primera vez se incluía en el proyecto de diseño de una computadora digital electrónica el concepto de control por programa almacenado. Esta máquina empezó a funcionar hasta 1952, pero antes basadas en sus publicaciones se construyeron varias computadoras de programa almacenado, entre ellas EDSAC en 1949 y que recibió el mérito de ser la primer computadora con programa almacenado que funcionó. Más adelante surgió la UNIVAC en ella se utilizó la cinta magnética para almacenar información fuera de la computadora y transmitirla a altas velocidades a ella misma.

La segunda generación de computadoras, se basaba en el transistor como sustituto del bulbo, apareció en los años cincuenta, el transistor (más pequeños y veloces) fue inventado en los laboratorios Bell. Era posible colocarlos muy próximos uno al otro, con lo cual los impulsos eléctricos viajaban distancias menores y debido a que estaban hechos de una sustancia sólida eran más resistentes y más confiables. Su funcionamiento generaba mucho menos calor que los bulbos. Al final de la década de los cincuenta hicieron su aparición otras computadoras más sofisticadas, en ellas se empleaban transistores para las operaciones aritméticas, núcleos de ferrita para la memoria y discos o cintas magnéticos para el almacenamiento. Aquí se evoluciona del lenguaje de máquina al lenguaje ensamblador y algunos de alto nivel.

A principios de los setenta con la llegada de las técnicas de miniaturización electrónica (la implantación de miles de transistores en una minúscula plaqueta de silicio, Chip) surgieron las computadoras de la tercera generación. Estos circuitos son capaces de operar y transmitir información en pocas millonésimas de segundo, además de que el costo se reduce. Se desarrollan los primeros sistemas operativos y la multiprogramación en tiempo real y modo interactivo.

Al aumentar los niveles de integración y surgir nuevos lenguajes de programación aparecen las máquinas de cuarta generación, esto es en la década de los ochentas. Se desarrollan la integración de microcircuitos a gran escala (Large Scale Integration) que permite concentrar en una superficie de un centímetro cuadrado lo necesario para almacenar 64,000 unidades de información, con esta generación surgen las microcomputadoras. Aparecen lenguajes de programación de todo tipo y las redes de transmisión de datos.

En 1981 la industria japonesa lanzó las características de lo que sería la quinta generación de computadoras. En estas computadoras el diseño cambia profundamente y abandona la arquitectura de Von Newmann para utilizar un diseño más distribuido. Utilización de componentes de muy alta integración, y lenguaje natural, computadoras con inteligencia artificial y muy alta velocidad de procesamiento.

A lo largo de la historia de las generaciones de computadoras hubo hechos que influyeron notablemente para el desarrollo de la inteligencia artificial.

Alan Turing contribuyó notablemente a la aparición de la inteligencia artificial, en 1937 publicó una ponencia sobre "números computables", donde expuso por primera vez el concepto de la "máquina universal de Turing".

En su artículo manifiesta que la máquina puede desarrollar cualquier procedimiento matemático siempre que se le proporcione una tabla adecuada de instrucciones (programa), este modelo era tan moderno que permitió representar todas las computadoras que surgieron en décadas posteriores.

En 1974 Turing escribe un informe llamado "maquinaria de computación e inteligencia", en el cual comento acerca de la posibilidad de que las máquinas pudieran ser capaces de pensar y hablar, a lo que él llamaba "el juego de la imitación", actualmente denominado Test de Turing. En esta prueba el examinador separado de los examinados (persona y máquina) les realiza diferentes tipos de preguntas en forma independiente. Si el examinador no puede distinguir con

base a las respuestas de los examinados cual es la máquina y cual es el ser humano, la máquina puede ser calificada de inteligente.

La historia de la inteligencia artificial es la historia de la ciencia informática ya que la inteligencia artificial es una rama de la informática, el término inteligencia artificial se cree que fue utilizado por primera vez en 1956 por John McCarthy profesor auxiliar de matemáticas del Dartmouth College en Hanover, New Hampshire, Estados Unidos. Convocó a una conferencia considerada como el inicio del la inteligencia artificial como disciplina independiente de la informática. Con ella pretendía reunir a los investigadores de dicho campo y establecer comunicación entre ellos. Varios de los asistentes -Alcen Newell. Hebert Simon, Marvin Minsky y John McCarty- son reconocidos universalmente como destacados pioneros en IA.

Los primeros investigadores en AI se concentraron exclusivamente en la solución de problemas de tipo general. Los humanos resuelven los problemas aplicando conocimientos relativos al problema a resolver. Pronto se comprendió que tal vez las computadoras podrían programarse para resolver problemas de modo similar. Esto condujo a nuevos estudios de como representar conocimientos en los sistemas informáticos y obtener conclusiones lógicas (inferencia) a partir de ellos. Durante la década de los setenta el tema fundamental de investigación fue el estudio de los sistemas basados en el conocimiento conocidos como sistemas expertos. Otros temas de interés en inteligencia artificial fueron la traducción automática, los juegos y el comportamiento de los autómatas.

## 1.2 Inteligencia artificial

Hasta el momento no existe una definición de lo que es la Inteligencia artificial, pero para poder comprender su objetivo podríamos resumirla en las siguientes características: es la rama de la ciencia de la informática que pretende lograr que las computadoras realicen actividades de manera semejante a el hombre, no de una manera mecánico sino mediante el procesamiento de información, es decir de una manera inteligente. La investigación en inteligencia artificial

contempla varios objetivos diferentes, encaminados a hacer a la computadora más útil para los seres humanos. Un objetivo es aprender más acerca de la forma en que funciona el cerebro humano al crear un modelo por computadora de él.

Para otros el objetivo de la IA es enseñar a la computadora como manejar de manera inteligente información completa, como aprender a partir de la experiencia, como plantear modelos del mundo y como planear con inteligencia como resolver problemas. En este tipo de programación en lugar de permitir que el analista determine y codifique la solución completa del problema en concreto, se ofrece una representación más general del ambiente del problema y se deja que la computadora busque una solución dentro de los límites de dicho problema.

## 1.3 Algunas técnicas en inteligencia artificial

Las técnicas en IA intentan en forma explícita, trasladar el proceso del razonamiento al programa para resolver problemas.

Algunas de las técnicas más utilizadas se muestran a continuación.

**Sistemas de producción**

Las arquitecturas de sistemas de producción son un destacado método de representación de conocimientos en sistemas de AI. Originalmente fueron propuestos por A. Newell, 1973, como modelos de razonamiento humano. El planteamiento de los sistemas de producción está ligado al concepto de "la regla", cada uno de los cuales es esencialmente un par operativo patrón ---> acción.

Un sistema en IA puede acumular conocimientos en forma de reglas de producción. Las reglas se toman del experto, se comprueba su coherencia, se programa en el sistema de IA. Simultáneamente se analiza que reglas son apropiadas y cuales inadecuadas.

La búsqueda de una solución a problemas utilizando IA implica la aplicación de muchos procesos inteligentes, es útil estructurar este tipo de programas de forma que faciliten la descripción del proceso de búsqueda. Los sistemas de producción proporcionan tales estructuras.

Los elementos principales de un sistema de producción en inteligencia artificial son una base de datos global, un conjunto de reglas de producción y un sistema de control.

La base de datos global es la estructura central de datos usada en un sistema de producción de IA. La complejidad de esta base de datos depende de la aplicación para la que será creado el sistema.

Las reglas de producción operan sobre la base de datos global. Cada regla tiene una precondición que puede ser satisfecha o no por la base de datos global. Si la precondición es satisfecha, la regla puede ser aplicada. La aplicación de una regla produce un cambio en la base de datos. El sistema de control determina qué reglas deben ser aplicadas y suspende el proceso cuando la base de datos global satisface una condición de terminación.

Una estrategia de control especifica el orden en el que se compararán las reglas con la base de datos y una manera de resolver los conflictos que surgen cuando diversas reglas coinciden a la vez.

Hay varias diferencias entre la estructura de estos sistemas de producción y los sistemas convencionales de cálculo que usan los programas organizados en forma estructurada.

"Todas las reglas pueden acceder a la base de datos global; no hay porciones de la base de datos a las que deben restringirse determinadas reglas. Las reglas no llaman a otras reglas; la comunicación entre reglas tiene lugar tan sólo a través de la base de datos global"[1]

---

[1] Principios de Inteligencia Artificial
Nilsson, Nil J. Ediciones Díaz Santos, 1987

Estas características de los sistemas de producción son compatibles con el desarrollo progresivo de los grandes sistemas en IA que requieren conocimiento creciente. Una dificultad para el uso de sistemas convencionales de programas organizados estructuradamente en las aplicaciones de IA es que las adiciones o cambios en los diversos programas existentes, en las estructuras de datos y en la organización de las subrutinas.

El diseño de los sistemas de producción es modular, y pueden efectuarse cambios en la base de datos, en el sistema de control o reglas, con relativa independencia.

Existen varias clases de sistemas de producción, según los tipos de sistemas de control que utilizan, las propiedades de sus reglas y de las bases de datos, así como la forma en que se aplican problemas específicos.

**Heurística.**

Para solucionar eficientemente la mayoría de problemas difíciles a menudo es necesario construir una estructura de control que aunque no nos garantice que encontremos la mejor respuesta, proporcione una respuesta adecuada.

"La técnica heurística es aquella que mejora la eficiencia del proceso de búsqueda, en promedio mejoran la calidad de los caminos que se exploran. Usando buenas técnicas heurísticas, se puede esperar lograr buenas e incluso óptimas soluciones a problemas difíciles"[2]

La principal utilidad del conocimiento heurístico es aumentar la eficiencia del proceso de razonamiento ofreciendo guías. Tal orientación tiene el efecto de reducir el espacio que realmente debe ser explorado antes de encontrar una solución. Esta solución permite resolver problemas más complejos dentro de un tiempo determinado.

---

[2] Inteligencia Artificial
Rich, E. Gustavo Gilli, 1983

La heurística se aplica ante la ausencia de mecanismos de control más precisos. Se centra en la identificación de soluciones que son satisfactorias. Como cualquier regla empírica resulta aceptable en la mayoría de los casos.

**Búsquedas**

Al intentar resolver cualquier tipo de problema se debe suponer que existe una solución, sí el problema está bien definido se puede especificar el porqué de la solución y como obtenerla. Pero en el planteamiento de un problema no hay ninguna implicación lógica que garantice la existencia de esa solución, sin embargo saber que existe una solución en un espacio de búsqueda determinado puede contribuir a la planeación y ejecución de la estrategia oportuna. Es importante limitar lo más posible el espacio de búsqueda, pero sin excluir la solución de este; además de establecer un planteamiento sistemático que garantice que las diferentes opciones son consideradas eficientemente y que ninguna de ellas es ignorada.

Existen diferentes tipos de búsquedas, algunas de la más utilizadas son la búsqueda en profundidad y la búsqueda en amplitud. En la primera el programa explora las ideas iniciales hasta sus límites, antes de considerar cualquier alternativa; en la segunda se selecciona una o dos opciones, ignorando las restantes, y las examina en detalle.

En la búsqueda en profundidad la ruta seleccionada se explora exhaustivamente, ignorándose las opciones restantes hasta haber llegado a un conclusión. Alcanzada dicha etapa se vuelve al último punto de toma de decisión y se investiga la opción siguiente. Este enfoque es ventajoso cuando la estructura o árbol no es muy compleja, pues se puede perder mucho tiempo si la solución está en uno de los últimos nodos.

La búsqueda en amplitud explora todos los nodos al mismo nivel antes de proceder a analizar nodos más avanzados de la estructura. En algunas estructuras arbóreas muy amplias las estrategias de amplitud puede resultar antieconómica en tiempo y en recursos. Las técnicas de

profundidad como de amplitud son métodos exhaustivos, considerados inadecuados para estructuras arbóreas de alto nivel de complejidad.

Se pueden utilizar algunas técnicas heurísticas para superar algunos problemas planteados en las estrategias de búsqueda para estructuras arbóreas. "La teoría de búsqueda heurística incluye la demostración matemática de que un procedimiento heurístico que use una función de evaluación numérica encuentra una solución por el camino más corto, supuesta cierta relación entre el nodo solución y el nodo en que se aplica la función. Un procedimiento de este tipo garantiza el camino más corto al mismo tiempo que evita procedimientos exhaustivo de búsqueda. Es decir, es a un mismo tiempo algorítmico y heurístico"[3]

En una búsqueda, la información heurística se puede aplicar de diferentes modos:

- Para decidir el nodo siguiente que se ampliará
- Para decidir la eliminación (poda) de ciertos nodos del árbol en el que se realiza la búsqueda.

Es útil ampliar el nodo que parece más prometedor para comenzar la búsqueda a partir de la mejor opción. Este enfoque está basado en la función evaluativa que permite estimar las probabilidades de un nodo.

La teoría de la búsqueda trata de la exploración y desarrollo de árboles y gráficos. Un árbol es un tipo especial de gráfico con un nodo superior y ramificaciones en o nodos predecesores (estructura de árbol invertido). En cada nodo hay que tomar la decisión acertada a fin de llegar a una solución en un mínimo de tiempo. Las tareas de búsqueda consisten en la generación de soluciones potenciales y en su correspondiente verificación. En inteligencia artificial, la computadora crea expresiones simbólicas y las modifica secuencialmente hasta satisfacer las

---

[3] Introducción a la Inteligencia Artificial
Simons, G.L. Ediciones Díaz Santos, 1987

condiciones. El sistema se comporta inteligentemente en el sentido que selecciona información del entorno del problema y la utiliza para guiar el proceso de búsqueda, evitando, en un contexto ideal aquellas exploraciones que no conducen a resultados positivos y que, por tanto, suponen una pérdida de tiempo.

## 1.4 Aplicaciones de inteligencia artificial.

La inteligencia artificial puede aplicarse a una gran variedad de problemas. Cualquier problema que no se preste a un solución algorítmica, puede ser resuelto basándose en alguno de los diferentes campos que comprende la inteligencia artificial. Los algoritmos necesitan datos precisos para resolver los problemas, muchos problemas no numéricos que contienen ambiguedades y datos inciertos no se ajustan a este tipo de procesos. Existen muchas situaciones en el mundo real que están desorganizadas, son imperfectas o de las que se carece de información completa. La IA puede tratar este tipo de problemas, dando soluciones satisfactorias. Las siguientes son algunas categorías de la inteligencia artificial.

**Sistemas expertos**

Un sistema experto reúne el conocimiento de uno o más especialistas en un campo específico del conocimiento, en un paquete de software que puede funcionar en casi cualquier tipo de computadora. Aunque el campo del conocimiento sea complejo, se pueden lograr los mismos resultados con un sistema experto que los que obtendría un especialista con el mismo problema. Disponer de un sistema experto permite realizar gran parte del trabajo de un especialista.

**Procesamiento de lenguaje natural.**

Por lenguaje natural se entienden los distintos lenguajes humanos, escritos o hablados. Los programas de procesamiento de lenguaje natural aceptan como entrada datos en lenguaje natural, los procesan y dan el resultado en lenguaje natural.

Las computadoras tiene muchos lenguajes en distintos niveles. En el nivel más bajo trabajan en binario, después se desarrollaron lenguajes ensambladores y de alto nivel para facilitar la programación, es necesario aprender a programar en esos lenguajes de computación si se desea trabajar con ellos.

Hablar en nuestro propio idioma con una computadora es una característica deseable, que permitiría trabajar de forma amigable con la computadora y poder realizar un trabajo útil. Los programas de procesamiento de lenguaje natural proporcionan esta característica.

Estos programas constan de una base de datos de conocimientos, reglas de producción y programas de control. La base de datos consiste en un diccionario de palabras que la computadora entiende. Cuando se introduce una frase en lenguaje natural, el programa de inferencia la examina para determinar si entiende las palabras introducidas. Si es así inicia la acciones deseadas.
Se han hecho grandes avances en los últimos años en el desarrollo de programas que entiendan frases complejas en lenguajes naturales.

**Reconocimiento de voz.**

Muy relacionado con los sistemas de procesamiento del lenguaje natural, los sistemas de reconocimiento de voz proporcionan un forma nueva de usar las computadoras. Los sistemas de procesamiento de lenguaje natural pueden interpretar textos que se tecleen. En el caso de sistemas de reconocimiento de voz las computadoras entiende la voz humana. El sistema de datos de entrada es un micrófono que produce una señal eléctrica analógica que será transformada en números binarios que puede interpretar un programas en inteligencia artificial. Al igual que otros programas en IA se utilizan técnicas de búsqueda y comparación de patrones. La entrada binaria se compara con patrones de entrada anteriores que forman parte de la base de conocimientos. Si la entrada oral coincide con los patrones predefinidos, se produce el reconocimiento y la comprensión.

**Visión por computadora.**

Los sistemas de visión por computadora utilizan las técnicas de IA para analizar e interpretar información visual se recoge mediante cámaras de televisión que se codifica en señales binarias que representan la imagen visual, la cual se almacena en la memoria del ordenador, donde podrá ser manipulada mediante los programas de IA.

Estos programas utilizan técnicas de búsqueda y comparación de patrones para ayudar a identificar los objetos y sacar información de los datos visuales. Se utiliza un base de conocimientos que contiene formas y patrones conocidos con los que los programas de inferencia efectúan comparaciones. Cuando se produce un reconocimiento el objeto queda identificado y comprendido.
Los sistemas de visión tienen muchas aplicaciones. La mayoría se utilizan para inspección y control en las fábricas, donde pueden reemplazar a los trabajadores humanos.

**Robótica.**

El campo de la inteligencia artificial intenta imitar capacidades humanas en especial la mente. El campo de la robótica intenta imitar las capacidades físicas de los humanos. Un robot es un dispositivo electromecánico que intenta duplicar algunas funciones de la anatomía humana. El dispositivo más utilizado en robótica es una máquina que simula ciertas funciones de un brazo y una manos humanos. Conocidos como manipuladores, estos robots son muy utilizados en procesos de fabricación, y son utilizados también en ambientes peligrosos.

La mayoría de robots están controlados mediante un dispositivo electrónico o circuitos integrados, se programan con algoritmos que indica a la extremidad exactamente como tiene que moverse. Un problema de este método es que el brazo por ejemplo hace exactamente lo que se le dice, y nada más. Los elementos que va a manipular deben estar perfectamente alineados con respecto al brazo, ya que este no puede tomar decisiones y modificar su funcionamiento para adaptarse a condiciones no preestablecidas.

La forma de dar a los brazos de un robot mayor utilidad consiste en dotarlo de sensores e inteligencia. Proporcionando al robot la capacidad de definir su posición o de ver el objeto con el que se está trabajando, este puede adaptarse a las situaciones que se van planteando. Combinando los sensores con la inteligencia artificial, se crea un robot inteligente capaz de adaptase a su entorno.

**Otras aplicaciones.**

Una aplicación menos desarrollada de la Inteligencia Artificial es el aprendizaje por computadora, los investigadores están intentando lograr que las computadoras aprendan con su experiencia o de los datos que reciben. El aprendizaje es una capacidad clave de los seres humano y los investigadores en inteligencia artificial buscan una forma eficaz de imitarla.

Los métodos de Inteligencia artificial se están utilizando para crear una forma de enseñanza asistida por computadora. Mediante la pantalla y el teclado el estudiante mantiene comunicación con la computadora.

Otra aplicación es la programación automática, que se aplican las técnicas de la inteligencia artificial a la tarea de crear software. Estos programas ayudan a simplificar y acelerar el proceso de programación.

# Capítulo 2

## Sistemas expertos

### 2.1 Antecedentes de los sistemas expertos

En la década de los cincuentas apareció un interés especial por parte de los pedagogos y psicólogos, por encontrar los métodos generales para la solución de diversos tipos de problemas, con el fin de que estos métodos se pudieran enseñar a los estudiantes y con ello se mejorará su preparación. Se había ya observado en aquel entonces, que las personas aun conociendo toda la información necesaria para resolver correctamente un problema (definiciones, fórmulas etc.) son muchas veces incapaces de conseguirlo realizando con frecuencia razonamientos defectuosos.

Con la difusión e implantación de las primeras computadoras, en la segunda mitad de la década de los cincuenta, los estudios ya realizados en la resolución de problemas se intentaron trasladar a las computadoras. Pero surgieron nuevos problemas, como lo son: la representación de los conocimientos en la memoria de la máquina, la representación de las relaciones entre los conocimientos, etc.

En la década de los sesentas, coincidiendo con la segunda época de la inteligencia artificial, aparecen numerosos trabajos sobre el método general y universal de solución de problemas desarrollados sobre computadoras, de todos ellos el más famoso es el "General program solver" de Newell, Shaw y Simón, desarrollado en la Universidad de Carnege-Mellon en el año de 1957.

Uno de los problemas que surgieron en aquel entonces fue, la aparición de la explosión combinatoria en los cálculos exhaustivos que limitaba la profundidad de los mismos y el número de conocimientos que se podían procesar, es decir se calculaban todas las posibles soluciones

para luego elegir la óptima. Aparecen entonces los primeros algoritmos de poda (Algoritmo alfa-beta de John Mc Carthy 1961).

Se pretendió resolver un problema tan general y amplio con herramientas poco adecuadas. Tanto el hardware como el software estaban adaptados al cálculo numérico y no al campo simbólico, las computadoras eran lentas y tenían poco capacidad de memoria, se gastaban muchos recursos y estos eran caros, además los conocimientos lógico matemáticos necesarios para el desarrollos de estos aun no eran los adecuados.

En la década de los setenta coincidiendo con la tercera generación de computadoras, en la IA los planteamientos en el campo de la resolución de problemas cambiaban. No conociendo los mecanismos generales de resolución de problemas de la mente humana se pensó en simular los mismos para campos muy concretos del conocimiento. Es decir se imita la forma externa o comportamiento aparente, que es el enfoque opuesto a la línea de investigación de las redes neuronales.

El manejo eficaz de los conocimientos dio entonces sus primeros éxitos: los sistemas expertos. El precursor de los sistemas expertos actuales es el sistema DENDRAL (Universidad de Standford 1967), que incorporaba una gran cantidad de conocimientos que no estaba incluidos en el programa. Utilizaba para la representación del conocimiento las reglas de producción. El sistema era capaz de determinar la estructura química de un compuesto orgánico a partir de los resultados obtenidos mediante un espectrógrafo de masas. Un desarrollo mejorado del mismo todavía se utiliza en la industria mecánica.

De los sistemas expertos que se construyeron en esta época son famosos: PROSPECTOR (Standford Research Institute 1974) Sistema Experto en prospecciones mineras especialmente las petrolíferas que cuentan entre sus méritos el descubrimiento en 1980 en el estado de Washington de un importante yacimiento de molibdeno que fue posteriormente confirmado mediante prospecciones, y el MYCIN (Universidad de Standford 1977) Sistema experto en el diagnóstico y terapia de enfermedades infecciosas de origen bacteriano que contaba con unas 400 reglas.

En estos años de vida de los sistemas expertos se ha creado una nueva rama del saber: la ingeniería del conocimiento que es la parte de la inteligencia artificial que estudia los sistemas expertos o basados en el conocimientos y es un campo muy prometedor de la IA.

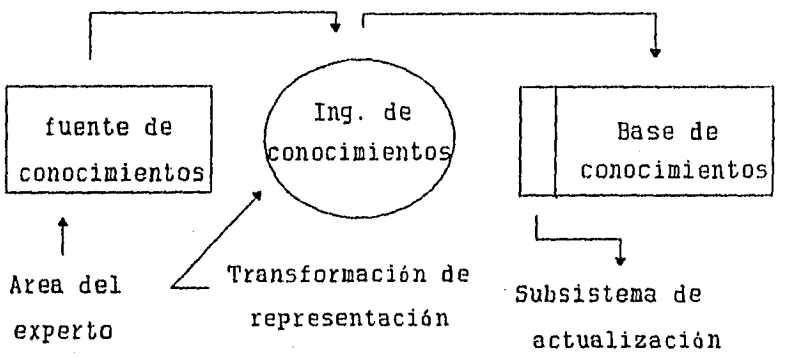



Fig. 2.1 Ingeniería del conocimiento

## 2.2 Primeros sistemas expertos

La tecnología representada por los actuales sistemas expertos es resultado de las técnicas en Inteligencia Artificial que han sido sujeto de intensas desarrollos desde 1950 en el área de los lenguajes de procesamiento simbólico como IPL, el primer lenguaje de procesamiento simbólico de listas, fue muy utilizado en los inicios de la IA. LISP actualmente es uno de los lenguajes más populares en IA fue desarrollado por John McCarthy en 1958.

El desarrollo los sistemas expertos actuales empieza a mediados de 1960. Varios sistemas fueron desarrollados entre 1965 y 1970, muchos de ellos con un campo muy limitado de aplicación.

## DENDRAL

El primer sistema experto fue desarrollado a principios de 1960 llamado **Dendral** en la Universidad de Stanford y ha influido en el diseño de la mayoría de sistemas expertos. Ayuda a los químicos en la identificación de la estructura molecular en sustancias desconocidas. Debido al gran número de posibles combinaciones de moléculas que puede formar una sustancia, los químicos tienen problemas a la hora de determinar la estructura exacta del compuesto a partir del análisis; empleando los datos que se obtienen en un espectrómetro de masas, las mediciones de resonancia magnética nuclear y los datos de laboratorio, Dendral puede identificar con rapidez y exactitud la estructura del compuesto.

## MACSYMA

Desarrollado en el MIT (Instituto Tecnológico de Massachusetts) a principios de los años sesentas, está diseñado para realizar manipulaciones simbólicas mediante expresiones matemáticas. Se siguió trabajando con él hasta principios de los ochenta. Es capaz de realizar simbólicamente cálculos diferenciales e integrales, así como complicadas expresiones algebraicas ayudando a reducirlas a una forma simple.

## MYCIN

Sin embargo no fue hasta 1970 que se inicio la investigación en gran escala. Los primeros sistemas expertos actuales se desarrollaron en el campo del diagnóstico médico. En la década de los setentas **MYCIN**, de la Universidad de Stanford; es un sistema para el diagnóstico y tratamiento de enfermedades infecciosas en la sangre. Fue uno de los primeros sistemas expertos en utilizar reglas de producción y métodos de inferencia con encadenamientos regresivos. También introdujo el concepto de factores de incertidumbre. Una de sus características principales es que su base de reglas está totalmente separada de su programa de inferencia. Este es uno de los sistemas expertos más estudiados y actualmente continua funcionando.

**PROSPECTOR**

Construido sobre la tecnología de MYCIN es un sistema diseñado para ayudar a los geólogos a encontrar yacimientos importantes. Su base de conocimientos que se complementa con reglas de producción y redes semánticas, contiene información sobre los depósitos y sobre la clasificación de distintos tipos de rocas y minerales. Dándole datos de acerca de una área en particular. Puede estimar probabilidades de encontrar distintos tipos de depósitos de minerales. Ha tenido un gran éxito en la localización de grandes depósitos de minerales valiosos.

**XCON**

Es un sistema experto diseñado para ayudar a los técnicos de Digital Equipment Corporation (DEC) a configurar sus sistemas de microcomputadoras. La serie VAX de DEC se encuentra disponible en una amplia gama de modelos con una gran cantidad de características y opciones que puede seleccionar el cliente. Dado el gran número de posibles combinaciones, el personal de DEC tenía problemas para configurar el sistema apropiado en el que todos los componentes funcionaran en conjunto. XCON genera automáticamente la configuración deseada a partir de los requisitos del cliente.

Inicialmente se denominó R1 y fue desarrollado en la Universidad Carnegi-Mellón a finales de los años setenta, siendo revisado y actualizado en DEC a principios de los años ochentas.

En años recientes los sistemas expertos han cubiertos más áreas del desarrollo humano como lo son los siguientes:

## 2.3 Conceptos fundamentales de los sistemas expertos

### Los expertos humanos

"Se denomina especialista en un campo (Experto) a una persona que tiene una gran cantidad de conocimientos en un área determinada del conocimiento. Un especialista adquiere sus conocimientos mediante aprendizaje formal y no formal, así como mediante la experiencia. Los especialistas pueden resolver problemas y tomar decisiones acertadas con rapidez"[4]

La importancia de los expertos son personas se debe a sus amplios conocimientos en un campo específico de la ciencia, estos conocimientos los han adquirido a lo largo de años de aprendizaje y de experiencia. Para resolver problemas en su campo de especialización utilizan estrategias diseñadas por ellos mismos con un alto grado de eficacia en un tiempo mínimo. Esta rama de la Inteligencia Artificial también denominada sistemas basados en el conocimientos ha tenido un gran desarrollo debido a las importantes aplicaciones que en la actualidad se han desarrollado. Estos sistemas dan solución a algunos a problemas como la necesidad de más especialistas en diferentes áreas del conocimiento. A continuación se muestra una definición de sistemas expertos dada por J.P. Sánchez y Beltrán para poder comprender un poco más el concepto de lo que es un Sistema Experto y su similitud con los expertos humanos.

### Definición de Sistema Experto

"Un sistema experto o sistema basado en el conocimiento, es un conjunto de programas de computadora capaces, mediante la aplicación de conocimientos, de resolver problemas en un área determinada del conocimiento o saber y que normalmente requerían de la inteligencia humana"[5]

---

[4] A fondo los sistemas expertos
Frenzel, Louis E. Jr. Editorial Anaya, 1989

[5] Introducción a la Inteligencia Artificial
Simons, G.L. Ediciones Díaz Santos, 1987

Otra definición de un Sistema Experto tomada del libro de David W. Roltson es la siguiente:

"Un S.E es un aplicación computacional para la solución de complicados problemas que de otro modo requerirían una gran experiencia humana. Así estos Sistemas Expertos simulan el procesos de razonamiento humano aplicando específicamente conocimiento e inferencias"[6]

Roltson muestra algunas características que un sistema experto ideal debe incluir, estas se muestran a continuación:

- Extenso conocimiento de un campo específico de interés humano
- Aplicación de técnicas de búsqueda
- Análisis heurístico
- Capacidad para inferir nuevos conocimientos del conocimiento ya existente en el
- Procesamiento simbólico
- Habilidad para explicar su propio razonamiento

Como se puede ver un sistema experto no se comporta como un experto humano pues no se conocen aun los procesos mentales que se ponen en funcionamiento en el hombre cuando se trata de resolver un problema y mucho menos cual es el fundamento de su inspiración. Es por eso que lo que realmente hace un S.E. es simular estos procesos, mediante programas de computadora que manejan un lenguaje simbólico que es muy adecuado para este tipo de procesos. Realiza estos razonamientos de una forma mejorada ya que el S.E. tiene como características, el explicar porque procede de una u otra forma y el justificar los resultados obtenidos.

---

[6] Principles of Artificial Intelligent and Expert System Development
Roltson, David. McGraw-Hill, 1990

| | Sistema experto | Experto humano |
|---|---|---|
| conocimiento | adquirido | adquirido +innato |
| adquisición conocimiento | teórico | teórico+práctico |
| campo | único | múltiple |
| explicación | siempre | a veces |
| limitación en capacidad | sí | sí, no evaluable |
| reproducible | sí, idéntico | no |
| vida | infinita | finita |

Fig. 2.2 Cuadro comparativo entre sistemas expertos y expertos humanos

**Características de un sistema experto.**

Las principales características de un sistema experto son:

- Contienen extensos conocimientos de un campo específico del saber.
- Llegan a la solución de un problema en base a un razonamiento propio.
- Aprenden de sus experiencias, aumentando ahí su base de conocimientos

Los sistemas expertos en general no son capaces de obtener el conocimiento por si solos mediante la práctica por lo tanto no son realmente expertos, es más correcto denominarlos Sistemas basados en el conocimiento. Un Sistema Experto reúne el conocimiento de uno o más especialistas en un campo específico del saber en un programa de computadora. Se genera un estructura en la cual el conocimiento debe introducirse antes de intentar hacer algún tipo de consulta en el sistema. Hay que representarlo tanto sobre el papel como en el propio sistema experto, y por último este debe ser aplicable a la solución de los problemas, esta solución debe poder ser comunicada y explicada al usuario del sistema.

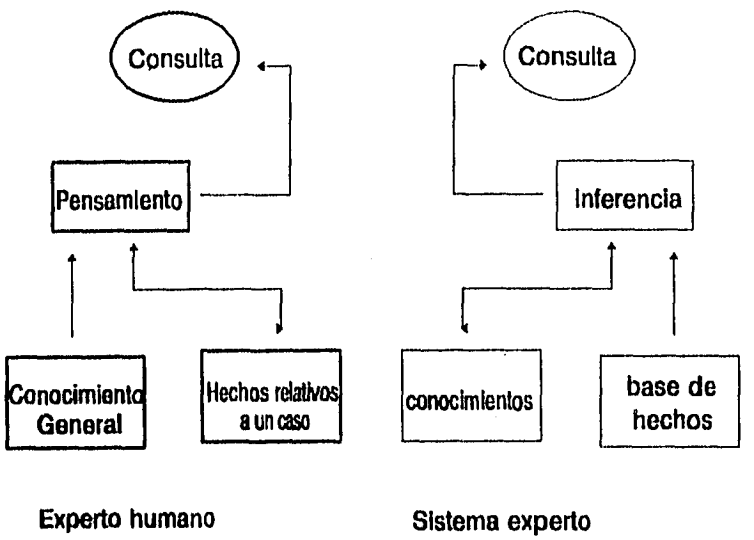



Fig. 2.3 Comparación de procesos entre un experto humano y un sistema experto

Algunas de las ventajas principales de los Sistemas Expertos son el mejorar la productividad ya que ponen una gran cantidad de conocimientos al alcance de cualquier no especialista para que este pueda aplicarlos cuando los necesite, evitando así esperas innecesarias o errores que podrían resultar muy costosos.

Un sistema Experto tiene por razones físicas una limitación en el conocimiento que puede contener, no solamente por el tamaño de la memoria en la que se almacena, sino también por el tiempo requerido para su procesamiento, por lo que es solamente un experto en un campo muy reducido del saber aunque cada día aumenta su capacidad. Estos sistemas permiten conservar el conocimiento de un especialista, así aun cuando este desaparezca sus conocimientos (quizás también los de otros) si han sido incorporados a un sistema de este tipo no se perderán. Por otra parte ayudan a mejorar la capacidad de aprendizaje, ya que una persona que utiliza regularmente un sistema experto para resolver problemas a la larga aprenderá conceptos y aplicaciones de la información que en el se encuentra.

Sus funciones básicas son la resolución de un problema en base al conocimiento que contiene de un área específica y la comunicación de este conocimiento al usuario. En un sistema experto es el mismo conocimiento el que resuelve y explica. La estrategia general de resolución es realmente el control del sistema que se denomina motor de inferencia.

El conocimiento en un sistema experto tiene que ser flexible, es decir, poder modificar el conocimiento que tiene almacenado sin que ello afecte el resto del sistema. En un tipo de programación estructurada (procesamiento numérico) es imposible que el programa funcione únicamente con parte del algoritmo o con los datos incompletos. En un sistema experto esto no sólo tiene que ser posible sino es así como debe poder funcionar, con bases de conocimiento que nunca están completas en las cuales el conocimiento va aumentando poco a poco con el transcurso del tiempo. Esto hace que las funciones de control (motor de inferencia), los datos o hechos (base de hechos) y el conocimiento sean completamente independientes. El conocimiento en un S.E. debe estar en el software e ir creciendo conforme sea consultado, ya que debe ser capaz de resolver cualquier problema que se plantee dentro de su campo. No puede tener almacenadas todas las soluciones a todos los posibles problemas (de no ser así no podría resolver nuevos problemas) por lo tanto no es necesario especificar el procedimiento que debe seguir en cada caso para obtener la solución, debe ser declarativo.

Si bien los problemas a resolver nunca son completamente numéricos o completamente simbólicos, los sistemas expertos emplean el razonamiento simbólico frente al procesamiento numérico o algoritmo que utilizan los programas tradicionales.

## 2.4 Estructura de los sistemas expertos

Por lo general en un sistema experto los datos están agrupados en lo que se denomina Base de hechos, los algoritmos no existen y en su lugar se utilizan sistemas de representación del conocimiento de tipo declarativo que forman la base del conocimiento, el control es independien-

te y se denomina motor de inferencia, la entrada y salida de los datos es similar a los programas tradicionales.

Los elementos que conforman un Sistema Experto son:

A) El motor de inferencia (búsqueda de información)
B) La base de conocimientos (Representación del conocimiento)
C) La base de hechos (información)
D) Los módulos de comunicación :

El módulo de consulta o del usuario
El módulo de trabajo o del experto

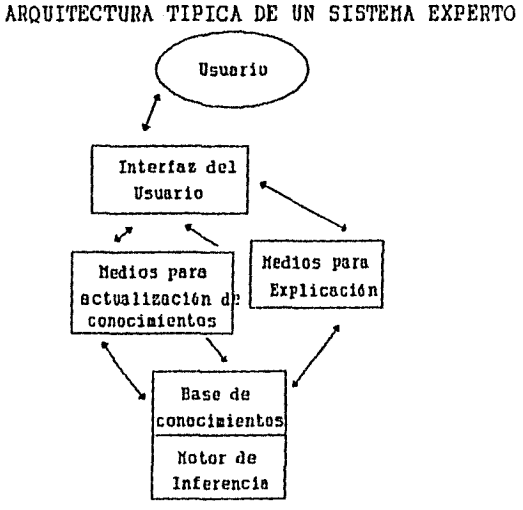



Fig. 2.4 Estructura de un sistema experto

**A) El motor de inferencia.**

Es realmente el sistema de control del sistema que construye de una forma dinámica el camino para llegar a la solución.

El motor de inferencia selecciona, decide, interpreta y aplica el conocimiento de la base de conocimientos, sobre la base de hechos con el fin de obtener la solución buscada. Este programa funciona con los datos suministrados por el usuario para recorrer la base de conocimientos. Controla todo el proceso, sus dos funciones básicas son inferencia y control. La inferencia se refiere al examen de las reglas a ejecutar de los programas de comparación de patrones, mientras que el control se refiere a la secuencia en la que se examinan las reglas.

Evalúa y selecciona todo el conocimiento a emplear, comprueba que el conocimiento sea aplicable, lo aplica, comprueba una condición final y controla las reglas activas.

"En general un motor de inferencia viene definido por las siguientes características:

- El lenguaje con el que ha sido escrito
- La velocidad de trabajo (el número de inferencias por segundo)
- Las estrategias que utiliza
- El sistema de elección del conocimiento
- La posibilidad de utilizar metaconocimientos.
- El orden de la lógica que emplea.
- El método de evaluación."[7]

El motor de inferencia en un sistema experto, es independiente del conocimiento y de los hechos, una variación en cualquiera de las bases puede significar una variación en el resultado,

---

[7] Sistemas expertos una metodología de programación
Sánchez y Beltrán. J.B. Macrobit, 1990

pero no variará el control y por lo tanto éste permanecerá libre de errores que pudieran ser producidos debido a cambios en la misma.

Un mismo motor de inferencia puede ser la base de varios S.E. en diferentes campos del conocimiento.

**Tipos Búsquedas**

Un motor de inferencia puede utilizar dos estrategias de búsqueda de soluciones.

**a) La búsqueda no ordenada de soluciones**

Aleatoria. En la que se recorre de forma exhaustiva la base de conocimientos, es válida cuando esta es pequeña y la velocidad de búsqueda es alta.

Heurística. En la que se divide la base de conocimientos en módulos y solamente se busca en alguna de estas partes según lo indique el metaconocimiento, este método de búsqueda ahorra gran cantidad de tiempo.

**b) La búsqueda ordenada de soluciones**

Se basa en el encadenamiento del conocimiento cuando este está representado en forma de reglas. Este encadenamiento se consigue haciendo que el consecuente o acción de una regla se convierta en el antecedente de la siguiente regla a considerar.

El encadenamiento hacia adelante, guiado por los datos o deductivo consiste en enlazar los conocimientos a partir de datos iniciales con el fin de obtener la solución de un problema. Este proceso de encadenamiento genera nuevos hechos y existen dos formas de tratarlos: en profundidad cuando un hecho en cuanto se genera se introduce en la base de hechos, o en

anchura cuando no se incorporarán a la base de hechos hasta que se ha terminado de aplicar la base de conocimientos.

El encadenamiento hacia atrás, guiado por los objetivos o inductivo, consiste en comprobar que el objetivo es cierto en base a unos hechos que forman el universo del sistema y en base a conocimientos.

**B) La base de conocimientos**

Es la parte de un S.E. que contiene el conocimiento del dominio en el cual es competente el programa (un campo o especialidad). Este debe representarse con el fin de que pueda incluirse en el sistema. Este representación debe poder manipularse sin dificultad, una unidad de conocimiento podrá ser modificada o excluída si que esto afecta el resto de la base de conocimientos o el sistema experto.
Debe contener un proceso de justificación de las soluciones y explicación de los procesos. Debe permitir establecer relaciones entre los conocimientos.

La mayoría de los sistemas expertos utilizan reglas de producción, por lo que también puede ser llamada base de reglas. Algunos sistemas expertos utilizan también redes semánticas y marcos de referencia.

**Reglas de producción.** "Son sentencias constituidas que tienen dos partes que contienen una pequeña cantidad de conocimiento."[8]

**Redes semánticas.** "Es una representación gráfica del conocimiento que ilustra las relaciones entre objetos. Este conocimiento se representa mediante nodos y arcos."[8]

---

[8] Los sistemas expertos en la empresa
Benchimol, Guy. Macrobite, 1988

Marcos."Son esquemas para representar el conocimiento, se divide en elementos llamados ranuras, estos contienen atributos de los objetos que se describen."[8]

Otra manera de representar el conocimiento es el sistema de lógica formal, el más empleado es la lógica de predicados de primer orden

**C) La base de hechos**

La base de hechos es el conjunto de información que forman el universo del sistema experto y en base a ellos y mediante la base de conocimientos se llega a la solución.

La base de hechos denominada a veces base de datos global o memoria de trabajo, es la parte de la memoria de la computadora que se emplea para mantener un registro de los datos recibidos, conclusiones intermedias y datos generados. La máquina de inferencia utiliza la base de datos como una memoria intermedia para almacenar lo que sucede en el sistema.

Los datos iniciales se almacenan en esta base. A medida que se van examinando reglas, se van almacenando las conclusiones derivadas de éstas en la base de datos. La máquina de inferencia utiliza estas conclusiones intermedias como nuevos datos para buscar nuevos patrones. Al final de un trabajo la base de datos contiene la cadena de hechos completa, que incluye no sólo los datos iniciales, sino también el camino para llegar a la conclusión del problema a resolver.

**D) Los módulos de comunicación**

**a) Módulos del experto.** Este realiza la adquisición del conocimiento y el mantenimiento del mismo, permite validarlo y depurarlo y además de configurar el sistema en base a los requerimientos del usuario.

b) **Módulo del usuario.** Este debe permitir el diálogo con el usuario de forma sencilla. Entre sus funciones más importantes están, la entrada de datos, salida de explicaciones y justificaciones, salida de soluciones.

## 2.5 La informática convencional frente al desarrollo de sistemas expertos.

La inteligencia artificial y los sistemas expertos son colecciones de programas que permiten a las computadores imitar la forma de razonamiento humano, es decir pensar, razonar y aplicar los conocimientos que tienen almacenados. Las computadoras como un conjunto de componentes electrónicos de propósito general fueron diseñadas para procesar de alguna forma la información, este procesamiento es básicamente numérico. Las computadoras por sí mismas no tienen algún tipo de aplicación, no funcionan si no cuentan con programas adecuados con los que hacerlo y explotar al máximo sus capacidades.

La mayoría de las computadoras usan lo que se denomina software convencional estos son programas que resuelven problemas matemáticos, manejo de bases de datos, hojas electrónicas de cálculo, procesamiento de texto o generación de gráficos.

Existe otro tipo de programas para computadora que es el que utiliza la inteligencia artificial, este tipo de programación o enfoque se denomina "Procesamiento simbólico" que les permite simular la capacidad de razonar del ser humano y llegar a las mismas conclusiones a las que llegaría alguna persona experta en algún campo específico.

En la mayoría de las computadoras el procesamiento de los datos se realiza mediante un código de programación llamado algoritmo, en este se integran las instrucciones de los pasos que se deben seguir para llegar al objetivo para el cual fue realizado ese programa. El algoritmo está integrado por una secuencia de instrucciones, estas deben realizarse en el orden en que fueron escritas siguiendo el flujo de las estructuras de control determinadas por este tipo de

programación. La computadora recibe la secuencia de instrucciones que debe realizar paso a paso en el algoritmo, este define en forma clara las acciones que se deben seguir para resolver el problema dado para el que fue diseñado este programa.

Para resolver un problema, los programadores analizan en primer lugar el problema, para determinar exactamente que datos de entrada necesitará el programa y que información de salida debe obtenerse y desarrollan entonces un algoritmo que procese los datos de entrada para producir la información de salida. El algoritmo es un procedimiento claramente definido que realiza las operaciones deseadas, una vez definido el programador debe proceder a traducirlo a una lista secuencial de instrucciones definidos en el lenguaje de programación en el que se ha decidido codificarlo. El siguiente paso consiste en generar el código binario mediante las herramientas adecuadas, es decir traducirlo a un lenguaje entendible para el procesador es decir, en lenguaje de máquina.

Los programas escritos en forma convencional almacenan los datos (hechos, figuras etc.) en archivos, los datos pueden ser manipulados y recuperables. Con estos realizan todo tipo de cálculos como lo son operaciones matemáticas, resolución de fórmulas etc. Una vez generado el código el lenguaje de programación utilizado debe poder traducirlo a un lenguaje que la máquina entienda directamente, también la traducción de datos de una forma a otra es una función que realizan este tipo de programas. El ordenamiento de datos seleccionados o la reestructuración de los mismos así como el manejo de la edición de datos como lo son la eliminación, cambio o adición. En base a estructuras de control determinadas se decide entre alternativas y para llegar a conclusiones. Pueden llevar a cabo el control de dispositivos internos y externos del equipo.

La inteligencia artificial o también llamada informática simbólica, utiliza un enfoque totalmente diferente para resolver problemas, el conocimiento se representa simbólicamente (un símbolo puede ser una letra, una palabra o un número que se emplea para representar objetos, acciones y sus relaciones) que pueden representar cualquier tipo de cosas incluyendo personas y lugares. acontecimientos e ideas. Los símbolos pueden almacenarse en la memoria de la computadora en

forma de caracteres o cadenas de caracteres ASCII. Mediante el uso de símbolos se pueden crear "Bases de conocimientos" que establezcan diferentes hechos acerca de objetos, acciones o procesos y sus relaciones.

| | Sistema Experto | Programa Tradicional |
|---|---|---|
| Conocimientos | En programa independiente | En programa y circuitos |
| Tipo de datos | Simbólicos | Numéricos |
| Resolución | Heurística | Combinatoria |
| Def. problema | Declarativa | Procedimental |
| Control | Independiente no secuencial | Dependiente secuencial |
| Conocimientos | Precisos | Imprecisos |
| Modificaciones | Frecuentes | Raras |
| Explicaciones | Sí | No |
| Solución | Satisfactoria | Optima |
| Justificación | Sí | No |
| Resolución en: | Area limitada | Especifica |
| Comunicación | Independiente | En programa |

Fig. 2.5 Comparación entre un programa tradicional y un sistema experto

Una vez creada la base de conocimientos, se debe diseñar un método para utilizarla. Básicamente se necesita un programa que use el conocimiento para razonar y pensar en como resolver un problema particular. Esta clase de programas se denominan generalmente programas de inferencia o motor de inferencia y están diseñados para tomar decisiones y juicios basados en los datos simbólicos de la base de conocimientos. Los programas de inferencia aceptan preguntas del exterior acerca de un problema particular e intentan aplicar los conocimientos de que disponen para resolverlo.

Los programas de inferencia manipulan la información simbólica almacenada en la base de conocimientos mediante un proceso de búsqueda y comparación de patrones. Estos programas reciben algunos datos iniciales que les proporcionan información suficiente para iniciar los procesos de tratamientos. Utilizando estos datos iniciales, exploran la base de conocimientos en

busca de los datos apropiados. La búsqueda continua hasta que se encuentra una solución, en la búsqueda inicial puede resultar la aparición de un dato que lleva a otro proceso de búsqueda y así sucesivamente. El programa de inferencia realiza series de búsquedas que se encadenan, simulando un proceso de razonamiento lógico.

Todos los programas de inteligencia artificial utilizan este sistema de búsqueda y comparación de patrones para encontrar conexiones y relaciones. Este proceso puede resolver el problema satisfactoriamente; en algunos casos, sin embargo, no se puede resolver sino se dispone de suficientes conocimientos o datos de entrada, sin estos la información simbólica puede no ser capaz de resolver los problemas. Esto contrasta con los programas de informática convencional, en los que un algoritmo siempre produce un resultado si se dispone de datos iniciales. Sin embargo cuando se dan suficientes datos de entrada y se dispone de una buena base de conocimientos la informática simbólica se pueden obtener soluciones muy satisfactorias.

Los programas de inferencia se implementan con algoritmos que definen las técnicas de búsqueda y comparación de patrones que se usarán en la base de conocimientos. Son estas técnicas las que resuelven el problema no los algoritmos.

El software convencional realiza una forma de tratamiento de datos denominada procedural en la que los algoritmos detallan específicamente los procedimientos usados para resolver los problemas. El software de la inteligencia artificial, incluyendo los sistemas expertos, no realiza tratamiento procedural. El programa contiene el "qué" del problema, pero no el "cómo". El qué son los conocimientos y los datos de entrada. Pero no existe ningún procedimiento de manipulación donde se detalle paso a paso las actividades a seguir. Mediante procesos de búsqueda y comparación, los programas de inferencia llegan a su propia conclusión.

# CAPITULO 3

## Representación del conocimiento

Diferentes tipos de representación del conocimiento son usados en el diseño de sistemas expertos, aquí trataremos algunas como lo son el cálculo de predicados, las reglas de producción, los marcos y las redes semánticas.

### 3.1 La lógica y el cálculo de predicados

La lógica está fuertemente relacionada con la validez de los argumentos, esto es con los métodos para determinar si algunas conclusiones se pueden deducir correctamente a partir de hechos supuestos. La lógica es de gran importancia para la programación ya que un programa es realmente un conjunto de enunciados cuasi-lógicos que son procesados para llegar a una conclusión.

La representación del conocimiento y las inferencias que pueden realizarse con base en la lógica para derivar nuevo conocimiento o información son una herramienta muy útil en el desarrollo de sistemas expertos. El razonamiento del sentido común es el tipo de inferencia que las personas utilizan en situaciones ordinarias, es muy difícil de implementar en las computadoras.

Uno de los métodos más frecuentemente utilizados para dibujar inferencias es la lógica deductiva que ha sido usada desde tiempo antiguos para determinar la validez de un argumento. Un argumento lógico es un grupo de oraciones en las cuales la última se justifica en base a las previas, en el llamado encadenamiento de razonamientos, un tipo de argumento lógico es el silogismo, por ejemplo:

premisa: Alguien que puede programar es inteligente

premisa: Miguel puede programar

Conclusión : Entonces, Miguel es inteligente

En un argumento, las premisas son usadas como evidencia sobre la cual se pueda sustentar la conclusión, las premisas son también llamadas antecedentes y la conclusión es llamada consecuente; la característica esencial de la lógica deductiva es que una conclusión verdadera sigue a premisas verdaderas, una línea es normalmente dibujada para separar las premisas de la conclusión como se muestra abajo aunque no es necesario.

El argumento podría ser escrito de la siguiente manera:

Alguien que puede programar es inteligente
Miguel puede programar
∴ Miguel es inteligente

En donde los tres puntos ∴ significan "por lo tanto"

Los silogismos son frecuentemente utilizados porque pueden ser expresados en términos de reglas IF THEN por ejemplo el silogismo anterior quedaría así:

IF Alguien que puede programar es inteligente
Miguel puede programar
THEN Miguel es inteligente

En general, un silogismo es cualquier argumento deducido en forma válida teniendo dos premisas y una conclusión, las premisas y las conclusiones son definidas como oraciones categóricas de las siguientes cuatro formas.

| Forma | Esquema | Significado |
|---|---|---|
| A | Toda S es P | Afirmación universal |
| E | Ninguna S es P | Negación universal |
| I | Alguna S es P | Afirmación Particular |
| O | Alguna S no es P | Negación Particular |

Categorías de oraciones

Este esquema de datos muestra la forma lógica de un silogismo.

Premisa mayor: Todo M es P
Premisa menor: Todo S es M
Conclusión: Todo S es P

Este es un silogismo estructurado en un forma estándar, el sujeto es el objeto el cual está siendo descrito, el predicado describe algunas propiedades del sujeto, por ejemplo la siguiente oración

Todas las microcomputadoras son computadoras

El sujeto es "microcomputadoras" y el predicado es "computadoras"

En la oración:

Todas las microcomputadoras con 8 megabytes son computadoras con gran memoria

El sujeto es "microcomputadora con 8 megabytes" y el predicado es "computadoras con gran memoria".

La forma de las oraciones categóricas han sido identificadas desde tiempos antiguos por las letras A,E,I, y O. La A e I indican afirmaciones, mientras que la E y O son negaciones. El verbo "es" llamado cópula significa conexión entre las dos partes de la oración. El tercer término del silogismo, M, es llamado término medio y es común en ambas premisas, es importante porque un silogismo es definido como una conclusión que no puede ser inferida desde una premisa sola.

Como un ejemplo se muestra el siguiente argumento

Toda A es B
Toda B es C
∴Toda A es B

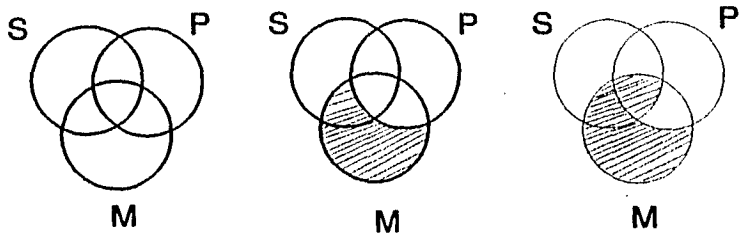
Este no es un silogismo válido porque se llega a la conclusión en base solamente a la primer premisa.

El cuantificador describe la parte de la clase que es incluida, los cuantificadores "Todo" y "No" son llamados cuantificadores universales por que se refieren a clases enteras. El cuantificador "Algunos" es llamado cuantificador universal porque se refiere sólo a una parte de la clase.

Existen procedimientos de decisión estos son métodos que permiten establecer la validez de un silogismo. Estos procedimientos de decisión son métodos mecánicos o algoritmos los cuales determinan automáticamente la validez de un silogismo. Los procesos de decisión para proposiciones consiste en construir tablas de verdad y examinarlas por medio de tautologías.

Estos procedimientos pueden ser hechos utilizando diagramas de Venn representados por círculos traslapados.

La premisa mayor es mostrada en la figura. La sección rayada de la M indica que no hay elementos en esa porción. En la C, la menor premisa esta incluida por rayados que no contienen elementos, desde la C se puede ver que la conclusión es falsa aunque algunos S están en P.




A)diagrama de Venn B)Después de la premisa mayor C)Después de la premisa menor

Fig. 3.1

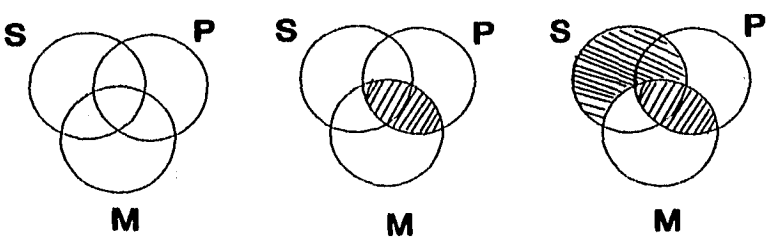
Ahora tomando en cuenta el siguiente silogismo:

Ningún M es P

Todo S es M

∴ Ningún S es P

Este es un silogismo válido como se muestra a continuación:




A)diagrama de Venn B)Después de la premisa mayor C)Después de la premisa menor

Fig. 3.2

## REGLAS DE INFERENCIA

Aunque los diagramas de Venn son un procedimiento de decisión de silogismos, no es convenientes utilizarlos para argumentos más complejos porque serían muy difíciles de leer. La lógica proposicional ofrece otras formas para describir argumentos, por ejemplo el siguiente argumento proposicional.

Si hay electricidad la computadora podrá funcionar

hay electricidad
∴ la computadora podrá funcionar

Este argumento puede ser expresado en un sentido formal utilizando letras para representarlo como sigue:

A=ahí hay electricidad
B=la computadora podrá funcionar
los argumentos se escribirían así

A -> B
A
∴B

Argumentos como este aparecen frecuentemente. Un esquema general para representar argumentos de este tipo es:

p -> q
p
∴q

En donde p y q son variables lógicas las cuales pueden representar cualquier argumento. El uso de variables en lógica proposicional permite tipos más complejos de oraciones que las cuatro formas de los silogismos A, E, I, O. El esquema de inferencias de esta forma proposicional es llamado principalmente modus ponens.

El modus ponens es un caso especial de un silogismo lógico, es importante porque esta forma la base de la base de reglas de los sistemas expertos. Esta proposición compuesta p -> q, corresponde a la regla mientras que la p corresponde al patrón que debe igualar el antecedente para que la regla sea satisfecha.

Sin embargo , la condicional p -> q no es exactamente equivalente a una regla porque la condicional es una definición lógica definida por una tabla de verdad y ahí están muchas posibles definiciones de la condicional.

Generalmente se utilizan letras mayúsculas en la teoría lógica como A,B, C para representar proposiciones constantes como lo es en "Sí hay electricidad" y letras minúsculas como p, q, r para representar variables lógicas.

La tabla de verdad del modus ponens es mostrada a continuación, esta es una tautología, puesto que los valores de los argumentos se muestran en la columna derecha y todos son verdaderos no importado los valores de estas premisas. En la tercera cuarta y quinta columnas los valores verdaderos son escritos con ciertos operadores como lo son el -> y and (^). Estos son llamados conectores principales porque ellos conectan los dos partes principales de una proposición compuesta.

| p | q | p -> q | (p->q) ∧ p | (p->q) ∧p->q |
|---|---|---|---|---|
| T | T | T | T | T |
| T | F | F | F | T |
| F | T | T | F | T |
| F | F | T | F | T |

Tabla de verdad para modus ponens

Existen otras tablas de verdad a las que se aplican diferentes reglas de inferencia, esta se da como ejemplo debido a la facilidad que da para su comprensión.

Las reglas de inferencia pueden ser aplicadas a argumentos con más de dos premisas.

**EL CALCULO DE PREDICADOS**

El cálculo de predicados consiste en una relación única entre algunos objetos representados dentro de un paréntesis y la relación entre estos que se encuentra fuera del paréntesis.

Es un (Victor, Ingeniero en computación)

Tales estructuras forman las sentencias básicas del cálculo de predicados, los nombres de relación se denominan "predicados" y los objetos son conocidos como "argumentos. El orden que deben tener los argumentos debe ser determinado por el programador y ser seguido de una manera constante.
Un predicado puede tener cualquier número de argumentos:

trabaja (Victor,UNISYS, Ingeniero en computación)
computadora (descompuesta)

Las proposiciones individuales (atómicas), tienen sólo un predicado y el argumento que las relacionan estas pueden combinarse para formar proposiciones compuestas mediante el uso de conectores lógicos como lo son "y" , "o", no e implica que puede leerse como si entonces. Con estas estructuras es posible representar un gran número de relaciones como por ejemplo

es un (Raúl, Ingeniero en software) y trabaja(Raúl,IBM)

En el cálculo de predicados existe una estructura adicional llamada cuantificador se utilizan para indicar cuántas particularizaciones de variables necesitan tomar el valor de "verdad" para que toda la proposición sea "verdad". Hay dos tipos de cuantificadores el cuantificador universal y el cuantificador existencial. Con la cuantificación universal todas las sustituciones de la variable entre paréntesis por elementos de algún dominio de aplicación, deben tomar el valor de elementos "de verdad" , con el cuantificador existencial solamente algunas de las sustituciones necesitan tomar dicho valor.

"Todos los Licenciados en Informática son programadores"
**(X)(Licenciados en Informática(X)->programador (X))

"Algunos licenciados en informática son instructores"
*(X)(Licenciados en Informática(X)->instructores(X))

Estos cuantificadores pueden mezclarse en una misma expresión, el orden en que se introduzcan las variables afectadas pueden cambiar el significado.

Con la mecánica de representación del cálculo de predicados, es posible expresar de una forma estándar enunciados muy complejos.

La función de los conectores lógicos y de los cuantificadores en la construcción de proposiciones simples (atómicas) se rige por reglas perfectamente definidas llamadas "de formación". Con estas, aun cuando cambie la estructura de las proposiciones se conservará el significado de las

mismas. A las proposiciones correctamente estructuradas se les conoce tambien como "fórmulas bien conformadas".

Algunos de los cuantificadores que utiliza son los siguientes:

| Operador | Significado |
|---|---|
| ^ | AND; conjunción |
| V | OR; disyunción |
| ~ | NOT; negación |
| ⇒ | If...then; condicional |
| ⇔ | si y sólo si; bicondicional |

La negación tienda la más alta precedencia que cualquier otro de los operadores y por ello no es necesario encerrarlo entre paréntesis. Esto es, una declaración como -p^q significa lo mismo que (~p)^q.

La condicional es análogo a la flecha de las reglas de producción en las que esta es expresada en la forma: if...then, por ejemplo:

if
 está lloviendo
then
 lleva un paraguas

puede ser representada en la forma

$$p \Rightarrow q$$

donde:

p = está lloviendo

q = lleva un paraguas

La lógica de predicados o cálculo de predicados de primer orden es un lenguaje formal en el que se pueden expresar una gran variedad de sentencias.

La capacidad generativa y evaluativa de la lógica está fuertemente relacionado con el método de sistema de producción, la regla si P entonces Q corresponde a una de las reglas de producción, mientras que la proposición singular P corresponde a un hecho ya declarado en la base de datos del sistema, la cual al ser comparada con la parte si de la regla produce la afirmación de Q como un nuevo hecho.

El principal problema en el cálculo de predicados es el hecho de que cuando una proposición es encontrada verdadera, lo seguirá siendo aun cuando se agregue más información que cambie significativamente para el razonamiento humano la veracidad del argumento (razonamiento monotónico). Muchos científicos encontrando estas dificultades buscaron otras técnicas de procesamiento más efectivas para algunos problemas específicos, con el fin de hacer más fácil la programación de estos sistemas basados en el conocimiento desarrollaron formalismos que no tienen tanto poder expresivo como la lógica pero son más tratables computacionalmente como lo fueron la aplicación de encadenamientos y búsquedas a sistemas como las redes semánticas, marcos.

Para lograr automatizar las demostraciones del cálculo de predicados es necesario decidir que regla aplicar, evitar la explosión combinatoria y permitir un razonamiento no monotónico, el lenguaje de programación PROLOG se desarrolló para lograr la automatización del cálculo de predicados, como un método de prueba de teoremas y una estrategia de control, esta técnica es llamada "demostración de teoremas por resolución" .

La técnica de resolución, soluciona los problemas de la selección de las reglas de inferencia y la generación de proppsiciones sin interés, reduciendo el procedimiento de demostración a la aplicación de una sola regla.

El proceso de control utilizado por PROLOG utiliza el concepto de búsqueda en el espacio de estados, las proposiciones dentro de un conjunto de cláusulas se pueden representar organizadas jerárquicamente (como un árbol Y/O). PROLOG utiliza la estrategia de recorrer el árbol en profundidad, comenzando por la parte izquierda, comienza a buscar una correspondencia de la submeta, cuando falla la búsqueda de una correspondencia con una conclusión el sistema efectúa una vuelta atrás (encadenamiento hacia atrás) hasta llegar a una nueva submeta, continúa el proceso hasta llegar a el objetivo.

## 3.2 Sistemas de producción

A principios de 1970, A. Newell y H.A. Simons introdujeron la notación de sistema de producción como un modelo psicológico del comportamiento humano. En este modelo, parte del conocimiento humano es representado en unidades separadas llamadas reglas de producción, estas unidades contienen información concerniente a las acciones que una persona tiene que realizar al percibir ciertos estímulos del medio ambiente. Las cuales afectan el como una persona conceptúa la realidad de su medio ambiente; por un lado porque el supuesto previo (información ya existente en su cerebro) tuvo que ser revisado, y por otro lado porque el nuevo fenómeno (hecho) tuvo que ser explicado (explicarse el porque de lo sucedido). En este modelo se distinguen dos mecanismos diferentes de almacenamiento para la información sensorial: memoria a corto plazo y memoria a largo plazo. La memoria a corto plazo contiene una cantidad limitada de información que rápidamente se pierde, esta corresponde a la parte de los sistemas de producción en el cual la entrada y los datos que se derivan de esta son guardados. La memoria a largo plazo es información almacenada permanentemente y corresponde a la base de reglas de un sistema de producción. Las reglas de producción fueron utilizadas en el sistema Dendral para predicciones sobre la estructura molecular de algunos compuestos. Muchos sistemas utilizan las

reglas de producción como un esquema para la representación del conocimiento, la adecuada utilización de los sistemas de producción para construir ciertos tipos de sistemas expertos depende no sólo de las reglas de producción en sí mismas, sino también del tipo de método de inferencia utilizado para las reglas basadas en este tipo de razonamiento.

Un sistema de producción trabaja en base a reglas de producción las cuales se encuentran en la base de reglas. Una regla de producción se compone de dos partes la parte izquierda que contiene una serie de elementos que describen las condiciones que deben ser verdaderas para que la regla sea aplicable y la parte derecha describe las acciones que se van a llevar a cabo si se cumplen las condiciones. Además de estas reglas un sistema de producción contiene la declaración y es en donde se encuentran las definiciones de los objetos a los que se hace referencia en las reglas. La base de reglas y la declaración de dominio conforman la base conocimientos de el sistema de producción.

Durante una consulta de la base de conocimientos de un sistema de producción, pueden realizarse diferentes operaciones con la información, generar más (adición de información), borrar o modificar como resultado de la aplicación de reglas de producción, por la entrada de datos del usuario al sistema o como resultado de búsquedas en la base de datos. Los hechos entran al sistema como conocimientos durante una consulta y son almacenados en una base de datos global llamado memoria de trabajo del sistema la cual va indicando el estado actual del sistema.

Los hechos son representados como variables, los cuales pueden ser tomados ya sea como un valor o como un grupo de valores. Un grupo o todas las variables definidas en un sistema de producción, juntas con sus posibles valores, presentan información importante en el campo de el sistema. Pueden ser utilizadas variable de valor simple que pueden tomar a lo más un valor constante en un tiempo, o variables de múltiples valores.

En la declaración de dominio se especifica de que tipo son las variables y constantes que se utilizarán.

D= sexo: femenino, masculino
edad: entero
molestia: fiebre, dolor abdominal,masa percibida, sonido abdominal
trastorno:tumor abdominal

De una base de conocimientos de un Sistema experto los siguientes hechos son elementos de una consulta específica.

sexo: femenino
edad: 27
molestia: fiebre, dolor abdominal,masa percibida
trastorno: tumor abdominal

Las reglas de producción son usadas para representar soluciones de problemas de un campo específico del conocimiento, las reglas heurísticas, muestran la analogía de las reglas de producción con la vida real de la siguiente forma:

Y si
condiciones que se deben cumplir
entonces
conclusiones a las que se llega

Ejemplo:

si
el paciente sufre de dolor abdominal, un sonido en el abdomen es percibido por auscultacion y una masa no identificada es encontrada en el abdomen por tacto.
entonces
el paciente tiene tumor abdominal*

---

* Nota: Los ejemplos de esta sección fueron tomados de libro de Peter Lucas.

Cada regla heurística debe ser transformada a regla de producción. Una regla de producción como un regla heurística contiene condiciones y conclusiones, pero a diferencia de una regla heurística es un enunciado formal, las condiciones y conclusiones contienen los siguientes elementos.

- valores constantes simbólicos y numéricos
- variables
- predicados y acciones

Diferentes tipos de sintaxis son utilizados para representar las reglas de producción. La siguiente es la utilizada en el libro de Peter Lucas en el cual el define:

"Una regla de producción es una oración que tiene la siguiente forma:

<regla de producción>::=**IF**<antecedente>**THEN**<consecuente> **FI**"[2]

Puede contener operadores como AND, OR, ALSO, etc.

Algunos predicados que pueden ser utilizados son los siguientes:

a) igual
b) diferente_de
c) menor_que
d) mayor_que
e) conocer
f) desconocer

---

[2] Principles of expert systems
Lucas, Peter. Addison-Wesley, 1991

y algunas acciones:

a) adicionar
b) borrar
c) modificar
d) calcular
e) escribir
f) llamar

Una condición es construida por un predicado y dos argumentos asociados: una constante y una variable, una condición expresa una comparación entre un valor específico (la constante) y el valor que la variable ha adoptado. El predicado después de ser evaluado regresa un valor cierto o falso.

F={edad=50, malestar= {dolor-abdominal, fiebre}

en donde edad es una variable de valor simple y malestar es una variable multivalor. Ahora se debe considerar la siguiente condición:

igual (malestar, dolor abdominal)

El predicado igual regresa un valor verdadero si el dolor abdominal es una de las constantes en el grupo de constantes utilizadas por la variable multivalor malestar; de lo contrario el valor devuelto es falso.
Tomando la siguiente condición para el anterior predicado.

igual(edad, 40)

En base al enunciado anterior el valor regresado para esta evaluación es que la edad es=50.

Una conclusión se construye con una acción y dos argumentos asociados, una acción puede ser considerada para operar en una variable.

La regla de producción quedaría así:

if

igual(malestar, dolor_abdominal) y

igual(auscultacion,sonido_abdominal) y

igual(tacto,masa_percibida)

then

adicionar(trastorno, tumor abdominal)

fi

En un sistema de producción los objetos son frecuentemente usados para explicar un grupo de propiedades las cuales son mencionadas en las reglas heurísticas. El conjunto de objeto_atributo_valor se denomina tupla. El aumentar los objetos en los predicados y las acciones permite expresar relaciones entre objetos y sus atributos.

<condición> ::=<predicado>(<objeto>,<atributo>,<valor>)
<conclusión>::=<acción> (<objeto>,<atributo>,<valor>)
Ejemplo:

If

objeto, atributo, valor

igual(paciente, malestar, dolor_pantorrilla) Y

igual(dolor, presenta, caminar) Y

igual(dolor, ausencia, descansar)

Then

adicionar(dolor, causa, aneurisma aórtico) además

adicionar(paciente, trastorno, aneurisma aórtico )

Fi

En la regla de producción anterior los objetos son paciente y dolor, los atributos del objeto dolor están integrados en la regla como presencia, ausencia y causa. Los atributos del objeto paciente son malestar y trastorno.

No siempre es posible expresar las relaciones entre los objetos y los atributos de una manera natural. Es deseable en un sistema experto tener estructurada la información (relaciones entre objetos). También sumándolo a la base de reglas se utiliza un esquema de objetos, este es aumentado en la declaración de dominio de la base de conocimientos. Un esquema de objetos define la interrelación entre objetos y sus atributos asociados.

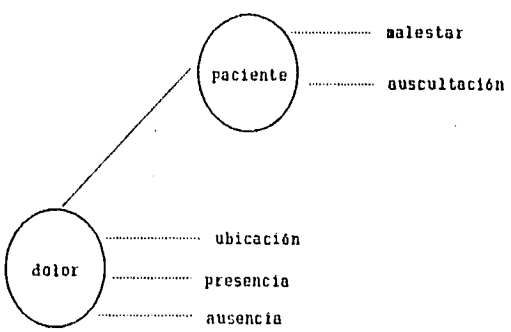



Fig. 3.3 Esquema de un objeto

La figura muestra una parte del esquema del objeto denominado sistema cardiovascular. En el objeto representado por una elipse, una línea simboliza la relación entre dos objetos, en este caso el objeto dolor es un subobjeto del objeto paciente, las líneas punteadas indican la relación entre un objeto y sus atributos asociados.

## 3.3 Redes semánticas

Una red semántica hace énfasis en la representación gráfica de las relaciones entre los elementos de un dominio. la estructura de una red semántica está conformada por nodos y arcos conectados entre sí, los nodos representan a los objetos físicos, conceptos o situaciones y los arcos o ligas las relaciones entre estos, las flechas muestran la dirección de las relaciones. Las relaciones proporcionan la estructura básica para organizar el conocimiento, sin ellas el conocimiento es una simple colección de hechos sin significado, además permiten que el conocimiento conforme una estructura con la cual otros conocimientos pueden ser inferidos.

La siguiente gráfica muestra la notación de redes semánticas utilizada por los sistemas expertos:

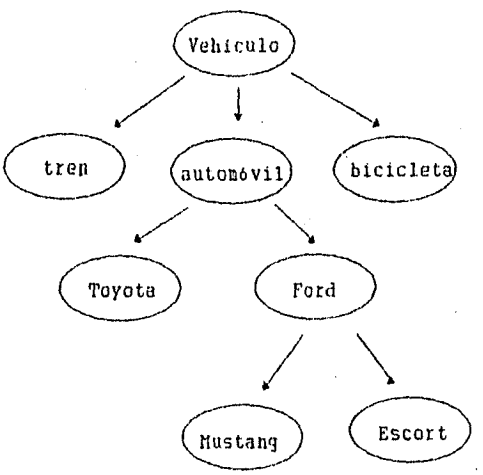



Fig. 3.4 Notación de una red semántica

En su modelo original Quillian trabaja con la memoria humana como una red asociada en la cual los conceptos son los nodos y las ligas forman la relación entre conceptos, un nodo es

estimulado en un inicio por una acción externa, este estimula nodos cercanos para conformar un patrón que se extiende conforme los nodos son estimulados, por ejemplo una persona conoce un millón de palabras pero sólo piensa en las que está leyendo en un escrito en un momento determinado.

Dos tipos de ligas son comúnmente usados IS-A ( es-un) y A-KIND-OF (del-tipo-de) estos se escriben comúnmente como ISA y AKO este tipo de ligas utilizan clases, una clase es un concepto matemático que se refiere a un grupo de objetos de un mismo tipo que tienen relación entre sí por ejemplo: avión, tren, automóvil. ISA se utiliza para relacionar los miembros específicos de una clase mientras que AKO relaciona una clase con otra.

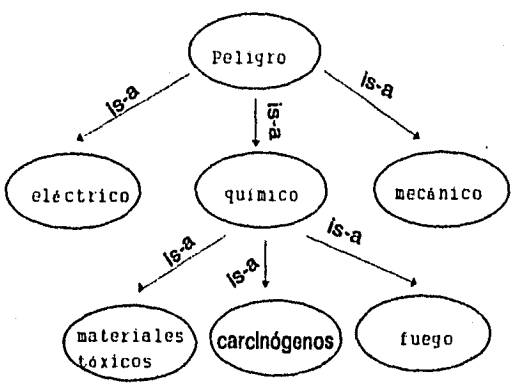



Fig. 3.5 Representación de la relación IS-A en una red semántica

Los objetos en una clase tiene uno o más atributos en común, cada atributo tiene un valor, la combinación de atributos y el valor son una propiedad, por ejemplo un dirigible tiene el atributo de tamaño, forma peso, color. El valor de la forma tiene la propiedad de ser elipsoidal.

Además de las características anteriormente mencionadas un concepto importante es el de la herencia, esta se deriva del uso de estructuras de conocimientos organizadas en forma jerárquica

por ejemplo: si sabemos que el dirigible de Goodyear es un dirigible, y también sabemos que los dirigibles tienen forma elipsoidal entonces el dirigible de goodyear es elipsoidal. La duplicación de características de un nodo a nodos descendentes es llamada herencia. A menos que se especifique lo contrario, se asume que los miembros de una clase heredarán todas las propiedades de su clase superior. La herencia es una herramienta muy usada en la representación del conocimiento por que elimina la necesidad de repetir características en común. Las ligas y la herencia dan un significado a el conocimiento representado.

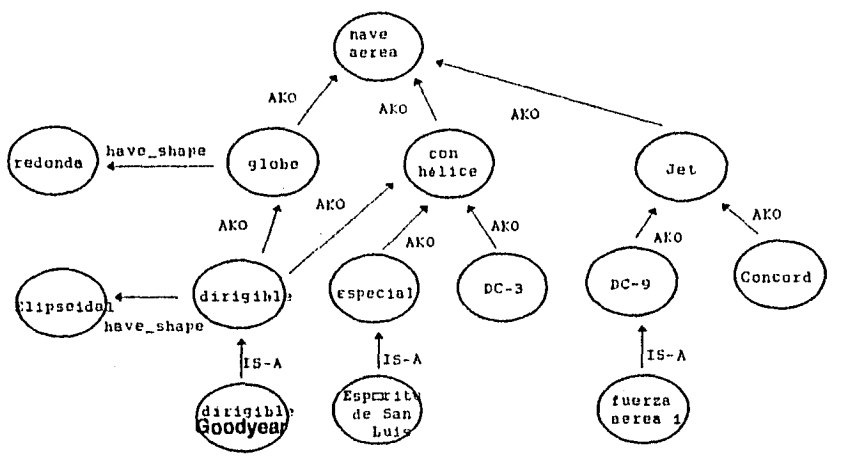



Fig. 3.6 Jerarquía y herencia en una red semántica

Pueden utilizarse tablas para nombrar nodos que contengan los campos objeto, atributo, valor que permiten construir una red semántica en base a ellos. Un esquema Objeto-atributo-valor puede ser usado para mostrar una red semántica. La representación en forma de tabla permite trasladar la red a código de computadora por reglas de inducción.

| OBJETO | ATRIBUTO | VALOR |
|---|---|---|
| manzana | color | rojo |
| manzana | tipo | macintosh |
| manzana | cantidad | 100 |
| uvas | color | rojo |
| uvas | tipo | semilla |
| uvas | cantidad | 500 |

Una tabla OAV (objeto-atributo-valor)

Una liga muy utilizada es la nombrada HAS-A (tiene-un) esta muestra que tiene atributos tiene un elemento como por ejemplo un globo tiene forma redonda (has a shape), es decir relaciona clases con subclases y es normalmente utilizada para relacionar un objeto con una parte de el.

automóvil HAS-A motor
automóvil HAS-A llantas
automóvil IS-A ford

Estos esquemas objeto-atributo-valor se utilizan especialmente para representar hechos y patrones cotejándolos con los antecedentes de una regla. Las redes semánticas como un sistema se conforman de nodos (objetos), atributos y valores conectados por una liga HAS-A e IS-A. Si sólo un objeto es representado y no es necesaria la herencia es suficiente su representación atributo-valor (AV) par.

Las redes semánticas tienen limitaciones al representar el conocimiento como lo son los nombres de las ligas, esto crea dificultades para entender cual es realmente el diseño de la red. Otro problema surge al nombrar a los nodos si un nodo es nombrado de alguna forma, puede tener varios significados o puede prestarse a confusión. En una red semántica para representar un conocimiento definitivo los nombres de sus nodos y ligas deben estar rigurosamente definidos.

Otro problema es la explosión combinatoria en cuanto a las búsquedas entre nodos especialmente si el resultado de la búsqueda es negativo, si una búsqueda produce un resultado negativo la búsqueda se realizará en muchos o todos los nodos de la red.

## 3.4 Marcos (frames)

Una red semántica es un ejemplo de una estructura de datos superficial porque todo el conocimiento en la red está en ligas y nodos. El término estructura de conocimiento es análogo a una estructura de datos en la cual están representados en una colección ordenada de conocimiento en vez de sólo datos, una estructura de conocimientos profunda como un conocimiento causal explica el porque ocurren las cosas, por ejemplo es posible construir un sistema experto médico con conocimiento superficial como el siguiente:

Si
    una persona tiene fiebre
entonces
    toma una aspirina

pero este sistema no contiene el conocimiento fundamental de bioquímica del cuerpo y el porque la aspirina puede bajar la fiebre. La regla misma regla podría definirse como:

Si
    una persona tiene un gato amarillo

entonces

toma un refrigerio.

En otras palabras el conocimiento de este sistema experto es superficial porque se basa en la sintaxis y no en la semántica, en donde hay dos palabras que pueden ser sustituidas por X o Y como en la siguiente regla

si

una persona tiene (X)

entonces

toma un (Y)

las letras (X) e (Y) no son variables en esta regla, pero representan cualquiera de dos palabras. Los doctores tiene conocimiento causal porque han tomado muchos cursos y tienen experiencia tratando personas enfermas, si el tratamiento que recetan no es adecuado, ellos pueden razonar acerca de él y encontrar una alternativa, en otras palabras los expertos saben cuando pueden romper las reglas.

Muchos tipos de conocimiento real no pueden ser representados por la estructura simple de redes semánticas, estructuras más complejas son necesarias para representar un conocimiento más profundo. En inteligencia artificial el término esquema es utilizado para describir una estructura del conocimiento más compleja que las redes semánticas. Las personas tienen estereotipos de conceptos en sus mentes por ejemplo al hablar de un animal muchas personas piensan en un perro. Un esquema conceptual es una abstracción de la realidad en la cual objetos específicos son clasificados por sus propiedades generales no incluyendo detalles exactos o características específicas de un individuo en particular. Un esquema tiene una estructura interna para sus nodos (las redes semánticas no) la etiqueta en una nodo de la red es todo el conocimiento que se tiene es como una estructura de datos en la ciencia de la computación en la cual la llave es además el único dato almacenado en el nodo, un esquema es como una estructura de datos en la cual los nodos contienen registros cada registro puede contener datos, registros o apuntadores para otros nodos.

Un tipo de esquema que es utilizado en muchas aplicaciones en inteligencia artificial es el frame, propuesto como un método para entender la visión, el lenguaje natural y otras áreas de la inteligencia artificial, los frames son estructuras para representar objetos en una situación usual (estereotipos). Los frames representan el conocimiento en tres dimensiones porque además de que los nodos tienen estructuras estas pueden ser simples u otros frames, su característica básica es que representan conocimiento limitado de un objeto, pero es conocimiento que siempre debe contener ese objeto. los slots y los fillers definen el objeto.

| Slots | Fillers |
|---|---|
| automotriz | General motors |
| modelo | Chevrolet caprice |
| año | 1979 |
| transmisión | automática |
| motor | gasolina |
| llantas | 4 |
| color | azul |

Frame de un automóvil

La principal importancia de los frames es la jerarquía que puede establecerse y la herencia. Con el uso de frames, jerarquías y herencia se pueden construir muy poderosos sistemas para la representación del conocimiento.

En particular, los sistemas expertos basados en frames son muy usados para la representación causal del conocimiento ya que su información es por causa y efecto.
Los fillers pueden ser valores como lo es una propiedad en el nombre del slot o un rango de

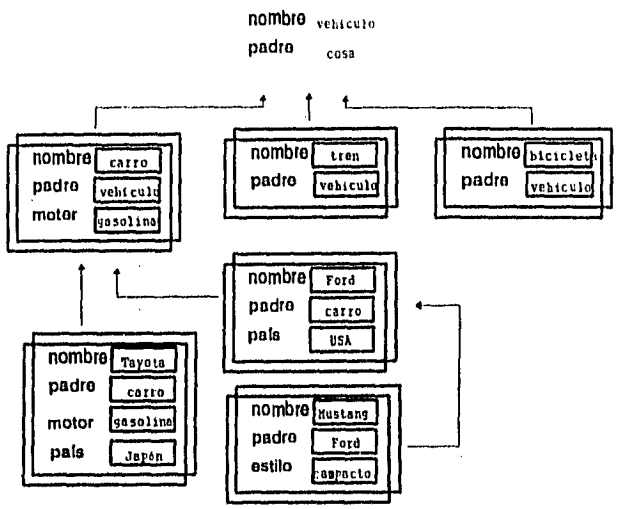



Fig. 3.7 Jerarquía y herencia en un frame

valores como en el tipo del slot, los slots pueden contener procedimientos fijos para si mismos llamados procedimientos unidos (procedural attachments) estos son generalmente como el si-necesito (if-needed) son procedimientos para ser ejecutados cuando un valor del filler es necesitado pero no esta inicialmente presente o el valor por default no es adecuado. Los defaults son de primera importancia en los frames porque ellos modelan algunos aspectos del frame corresponden a las expectativas de una situación que se puede construir basados en la experiencia, cuando se encuentra una nueva situación el frame más cercano es modificado para ayudar ajustándose a la situación. El pensamiento de las personas en una nueva situación no se inicia de la nada, los defaults y otros fillers son modificados, los defaults son utilizados para representar sentido común, es decir, el conocimiento que es generalmente utilizado en situaciones en las que información específica sea utilizada.

El tipo if-added (si-aumenta) es utilizado por procedimientos que se ejecutan cuando un valor es adicionado para un slot. En los tipos slot, el procedimiento if-added ejecuta otro proceso

llamado ADD-PROPERTY para aumentar un tipo de propiedad, si es necesario.
Un tipo if-removal (si-borra) es utilizado cuando un valor será borrado desde un slot, este tipo de procedimientos se usa si un valor es ya obsoleto.

| Slots | Fillers |
|---|---|
| Nombre | Propietario |
| especialización_de | un_tipo_de_objeto |
| Tipos | (carro, bote, casa)<br>si_necesario: proc.SUMAR_PROPIETARIO |
| Dueño | default:gobierno<br>si_necesario: proc.SUMAR_PROPIETARIO |
| Localización | (casa, trabajo, cambiante) |
| Status | (medio, pobre, bueno) |
| Garantía | (sí, no) |

Fig. 3.8 Frame con procedimiento

Como se vio en la figura anterior, el nodo superior en la jerarquía del frame muestra atributos que deben ser verdaderas en todos los elementos de esa categoría, mientras que en los niveles mas bajos se muestran características más específicas de un elemento. Los objetos que tienen características típicas a todos sus elementos son llamados prototipos.

Los frames fueron originalmente creados para representar un conocimiento estereotipado, los cuales tienen bien definidos los rasgos distintivos de una clase y muchos de estos están rasgos están definidos por default en los slots, una dificultad en ellos aparece al intentar determinar las características de algunos elementos que contemplan excepciones un ejemplo muy utilizado en estos casos es el siguiente:

la característica de vuelo en los animales esta limitada para las aves, un pinguino es un ave pero no vuela; esta excepción no está prevista al hablar de frames que contienen atributos específicos de una clase, en este caso la herencia no será valida para un pingüino.

# CAPITULO 4

## Métodos de inferencia

En gran parte de los sistemas expertos el problema de control es muy complejo, estos son llamados en algunas ocasiones como "búsqueda en el espacio de estados". En la resolución de este tipo de sistemas no existe un método único o camino para llegar a una solución, es necesario probar por diferentes caminos posibles, intentando evaluar en cada etapa la posibilidad de éxito.

La estructura de control o métodos de inferencia de un sistema establece la estrategia para seleccionar el camino a seguir, por ejemplo: una estrategia simple implicaría un examen exahustivo de la información. A continuación se muestran algunos métodos que utiliza la estructura de control para llegar a una adecuada solución sin tener que recorrer todo la base de conocimientos.

### 4.1 Inferencias sencillas árboles, redes y gráficas

Un árbol es una estructura jerárquica que se compone de nodos y ramas,los nodos contienen información o conocimiento, y las ramas que unen los nodos también son llamadas ligas.

En un árbol orientado, el nodo raíz es el más alto en la jerarquía y las ramas son las partes más bajas, un árbol puede ser considerado un tipo especial de red semántica en el cual cada nodo excepto la raíz tiene exactamente un padre y cero o más hijos. Los árboles binarios generalmente tienen 0, 1 o 2 ramas por nodo, este tipo de árbol tiene a lo más dos hijos por nodo y sus hijos son llamados hijo izquierdo ó hijo derecho.

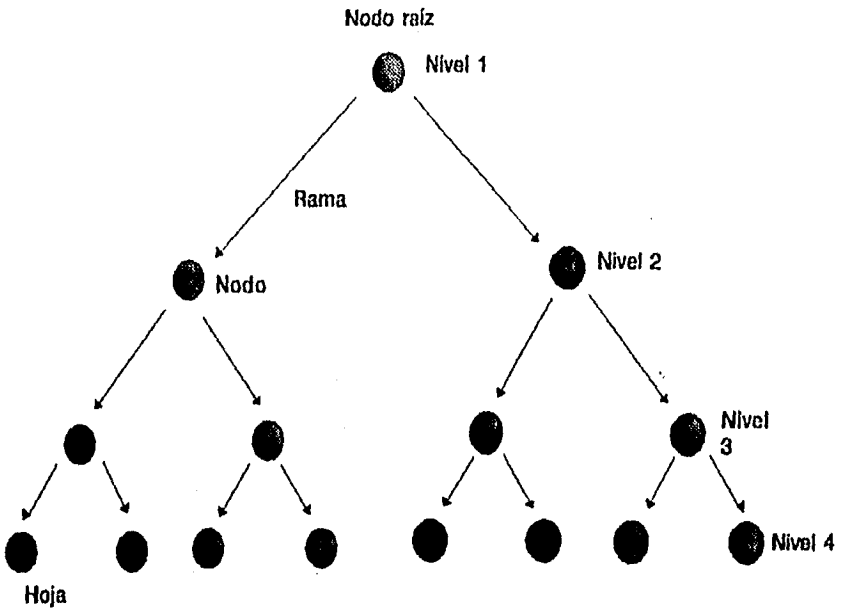



Fig. 4.1

En los árboles binarios existe solo una secuencia o ruta desde la raíz a cualquier nodo y no es posible moverse en contra de esta ruta.

Los árboles son un caso especial de una estructura matemática general llamada gráfica. Una gráfica puede tener cero o más ligas entre nodos y no existe alguna distinción entre padres e hijos, un ejemplo de una gráfica es un mapa en el cual las ciudades son nodos y las ligas son los caminos, las ligas pueden tener flechas y direcciones asociadas a ellas y un peso que caracteriza algunos aspectos de las ligas, una analogía de esto es una calle de un sentido con pesos límites que indiquen cuantos carriles hay en la avenida. Los pesos en las gráficas pueden ser un tipo de información, sí la gráfica representa la ruta de una aerolínea los pesos pueden ser millas entre ciudades, costo de vuelo, consumo de combustible o cualquier otro tipo de información relacionada.

Las gráficas y las redes son utilizadas para clasificar objetos por su forma jerárquica de los padres sobre los hijos, otra de sus aplicaciones es la toma de decisiones y estos son llamados árboles de decisión o redes de decisión, puede ser utilizado el término estructura para llamar a los árboles o redes. Una estructura de decisión es un esquema para representar el conocimiento y un método de razonamiento acerca de este conocimiento, la clasificación de animales es un buen ejemplo en el cual existe una estructura de red en la que los nodos contienen preguntas, las ligas las respuestas sí o no a las preguntas y las ramas almacenan el nombre del animal del que se está hablando.

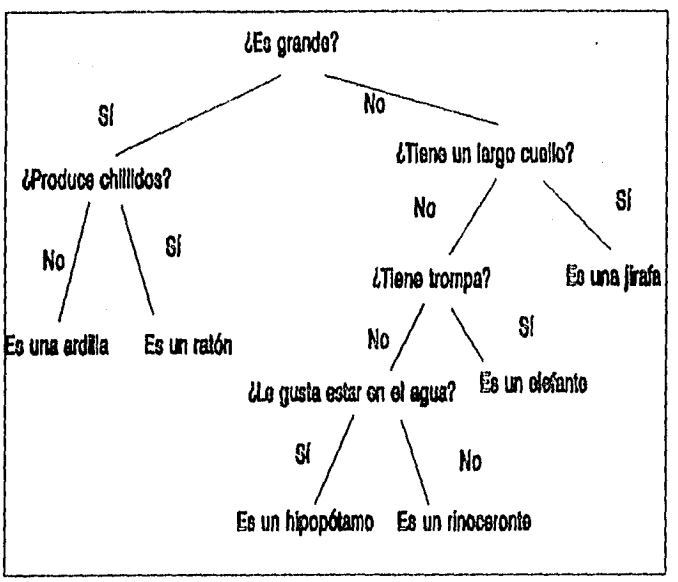



Fig. 4.2

Si las decisiones son binarias, entonces un árbol binario es fácil de construir y muy eficiente, cada pregunta va hacia abajo un nivel en el árbol, una pregunta puede decidir una de dos posibles respuestas, si un árbol binario es construido tal que todos sus ramas sean respuestas y todos los primeros nodos hacia abajo son preguntas, puede tener un máximo de $2^N$ respuestas por **N**.

Otra característica muy utilizada en los árboles de decisión es el poder autoaprender, si la solución (conjetura) es errónea, un procedimiento puede ser llamado para ejecutar una nueva búsqueda, y llevar a cabo una correcta clasificación de preguntas y de respuestas para las repuestas "si" o "no". Un nuevo nodo, ligas y ramas pueden ser creados dinámicamente y adicionados a los árboles.

Las estructuras de decisión pueden ser mecánicamente traducidas a reglas de producción, en una a lo ancho de la estructura y generando reglas IF-THEN-ELSE para cada nodo, por ejemplo

si la preguntas es = "¿Es muy grande? y la respuesta = "no"
entonces se pregunta := ¿grazna?

si la pregunta es = "¿Es muy grande? y la respuesta = "si"
entonces la pregunta := ¿tiene una nariz muy grande?

y así se continua con los demás nodos. Un nodo hoja puede generar una respuesta a partir de una pregunta, apropiados procedimientos podrían además hacer una búsqueda desde los datos de entrada de un usuario y construir nuevos nodos si fuera erróneo.

Las gráficas pueden ser aplicadas para muchos problemas prácticos, un método muy usado para describir el comportamiento de un objeto es definir una gráfica llamado espacio de estados, un estado es una colección de características las cuales pueden ser usadas para definir el estatus o estado de un objeto. El espacio de estados es un grupo de estados mostrando la transición entre ellos que un objeto puede experimentar, una transición toma un objeto desde un estado a otro.

El diagrama de estados es muy usado para describir la solución de un problema, en este tipo de aplicaciones, se puede pensar en el estado de espacios como un espacio de problemas en donde algunos estados corresponden estados intermedios en la solución de problema y algunos estados corresponden a las respuestas. Para la solución de un problema existen múltiples estado meta o de éxito que corresponden a las posibles soluciones, encontrando la solución para un problema

en un espacio de estados de un problema, envuelto en ello encontramos una ruta válida desde un inicio (oración del problema) a un éxito (respuesta).

El árbol de decisión de los animales puede ser visto como un problema de espacio en donde las respuestas si/no a preguntas determinadas son estados de decisión.

Un ejemplo de un problemas de espacio ocurre en el problema de el mono y los plátanos, este consiste en dar instrucciones a un mono diciéndole como tomar algunos plátanos colgados del techo. los plátanos están fuera de su alcance, dentro de la habitación hay un sillón y una escalera. La configuración del estado incial es el mono en el sillón.

Las instrucciones son:

1. Baja del sillón
2. Muévete hacia la escalera
3. Mueve la escalera bajo el lugar donde están los plátanos
4. Sube la escalera
5. Toma los plátanos

Las instrucciones variarán dependiendo de la configuración inicial del mono, sillón y escalera ya que pueden darse estados iniciales diferentes.

Aunque este problema parezca fácil de resolver para el ser humano, lleva un número considerable de razonamientos. Una aplicación práctica de los sistemas de razonamiento como este, es dar instrucciones a un robot concernientes a la solución de una tarea.

Otra aplicación muy usada de las gráficas es explorar rutas para encontrar soluciones a problemas.

La figura anterior muestra una red simple para el problema del viaje del vendedor, en este

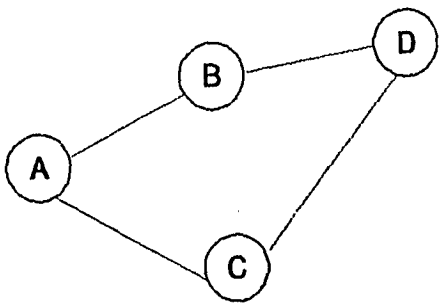



Fig. 4.3

ejemplo, se asume que el problema es encontrar una ruta desde un nodo A el cual visita todos los otros nodos, como es usual en el problema del viaje del vendedor que un nodo no pueda ser visitado dos veces.

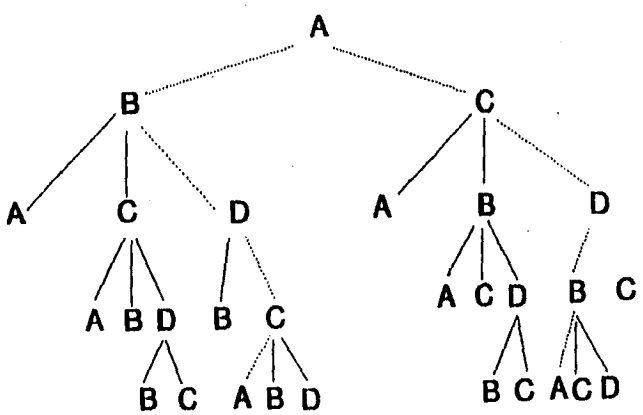



Fig. 4.4

Esta figura muestra todas las posibles rutas desde un estado inicial A a cualquier nodo en forma

de árbol, las rutas correctas ABDCA y ACDBA son mostradas con líneas punteadas en la gráfica.

Dependiendo de los algoritmos de búsqueda, la exploración de rutas para encontrar una correcta puede envolver una considerable cantidad de caminos hacia atrás, por ejemplo la ruta ABA podría ser la primera búsqueda sin éxito y después regresar a B, el camino CA, CB, CDB y CDC podría ser una búsqueda sin éxito hasta que el primer camino correcto ABDCA sea encontrado.

Muchos tipos de sistemas expertos utilizan el encadenamiento hacia atrás para solucionar problemas, los cuales tratan de resolver subdividiéndolos en pequeños subproblemas y resolviéndolos. Un tipo árbol o red que es muy usado en la representación de problemas con encadenamiento hacia atrás son los arboles AND-OR

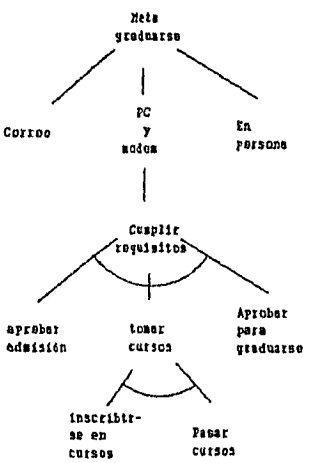



Fig. 3.5

La figura muestra un ejemplo de una red AND-OR para llegar a la meta de graduarse en alguna carrera, para realizar esta meta se puede ir a el colegio en persona o a través de cursos por

correspondencia, con cursos por correspondencia el trabajo puede ser mejorado ya sea por asignaturas por correo.o electrónicamente utilizando un módem o una computadora en casa.

El orden para satisfacer requerimientos para el grado, puede ser subdividido en tres submetas que deben ser cumplidas, a)aplicar para ser admitido 2)tomar cursos y 3) aplicar para graduarse, en la figura existe un arco que atraviesa las ligas desde las metas para satisfacer los requerimientos para este árbol de submetas, el arco representa el conector AND, que indica que esta meta puede ser satisfecha sólo si todos los requerimientos de esta submeta son satisfechos, las metas sin el arco como lo son correo, computadora y modem, y en persona son nodos OR en los cuales cumpliendo una de estas submetas puede lograr un grado en el colegio.

## 4.2 Encadenamiento hacia adelante y hacia atrás

Un grupo de múltiples inferencias que conectan un problema con sus soluciones es llamado encadenamiento. Existen dos tipos de encadenamientos el encadenamiento hacia adelante y el encadenamiento hacia atrás. En el primero la búsqueda o recorrido va desde el problema a la solución; también podría definirse como el razonamiento desde los hechos a las conclusiones, a las cuales se llega desde los hechos. Por otra parte el encadenamiento hacia atrás se da cuando un recorrido va desde una hipótesis de la posible solución hacia los hechos sobre los cuales se sustenta esta. Otra forma para describir este encadenamiento es en términos de metas a las cuales se llega a través de submetas.

El encadenamiento puede ser expresada en términos de inferencia por ejemplo si se tiene la regla de modus ponens del siguiente tipo:

$$\begin{array}{ll} p \rightarrow & q \\ & p \\ \hline \therefore & q \end{array}$$

Un ejemplo de una inferencia en base a un encadenamiento es la siguiente:

elefante (x) -> mamífero (x)
mamífero (x) -> animal (x)

Estas reglas pueden ser usadas en un encadenamiento causal para una inferencia hacia adelante como se muestra en la siguiente gráfica en la cual se deduce que Claudio es un animal, en base al encadenamiento que se muestra a continuación, dando como resultado que Claudio es un elefante.

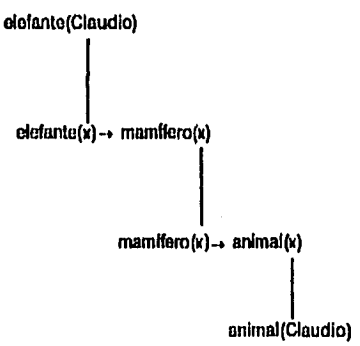



Fig. 4.6 Encadenamiento hacia adelante

En la siguiente figura se muestra el encadenamiento causal que es representado por una secuencia de barras,| conectando el consecuente de una regla con el antecedente de la siguiente, una barra además indica la unificación de variables con hechos, por ejemplo la variable (x) en el predicado elefante(x) debe estar unido con el hecho elefante(Claudio) antes de que la regla elefante pueda ser aplicada. El encadenamiento causal es una secuencia de implicaciones y unificaciones.

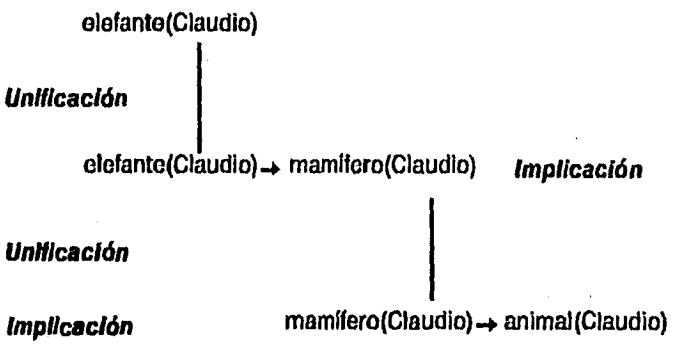



Fig. 4.7 Encadenamiento causal hacia adelante

Un encadenamiento hacia atrás es el proceso inverso. El problema principal o central del encadenamiento hacia atrás es encontrar una cadena que ligue la evidencia con la hipótesis de solución, el hecho: elefante (Claudio) es llamado evidencia en un encadenamiento hacia - atrás para indicar que sobre estos se sustenta la hipótesis.

Como ejemplo de un encadenamiento hacia-atrás y de encadenamiento hacia-adelante, Joseph Gitarro presenta la siguiente situación: "suponga que va manejando y repentinamente ve una patrulla con las luces y la sirena encendidas. Por encadenamiento hacia-adelante usted puede inferir que este policía lo busca o a alguien más, este es el acto inicial que soporta estas dos posibles conclusiones. Si la patrulla se coloca a la derecha atrás de usted o le hace algún tipo de señal, la siguiente inferencia es que ellos quieren que se detenga, tomando este como un trabajo de hipótesis, se puede utilizar un encadenamiento hacia-atrás para razonar el porque. Algunas posibles hipótesis intermediarias son manejar a alta velocidad, malfuncionamiento del

equipo, un vehículo robado, ahora usted examina la evidencia que pueda soportar una de estas hipótesis bajo un encadenamiento hacia-atrás o mediante conclusiones intermedias. Quizá fue la lata de cerveza que tiró por la ventana, o manejar a 100 km/hr en una zona con máxima velocidad de 30 km/hr, o el mal funcionamiento de una luz o la placa de su auto lo identifica como robado, en este caso cada parte de la evidencia soporta una hipótesis intermedia y todas ellas pueden ser verdad. Algunas o todas estas hipótesis intermedias son trabajos de razonamiento del porque la policía lo busca."

El encadenamiento hacia-atrás en el ejemplo anterior consiste en determinar en base a supuestos intermedios el por que se le indica detenerse(hipótesis), algunos de estos supuestos se mostraron anteriormente alguno de ellos o algunos pueden ligarse para llegar a la conclusión.

Para visualizar estos encadenamientos establecer una ruta a través del espacio del problema en el cual el estado intermedio corresponde a las hipótesis intermedias en el encadenamiento hacia-atrás o conclusiones intermedias bajo encadenamiento hacia-adelante, es de gran ayuda.

| **Encadenamiento hacia adelante** | **Encadenamiento hacia-atrás** |
|---|---|
| Planeación y control | Diagnóstico |
| Presente a futuro | Presente a pasado |
| Antecedente a consecuente | Consecuente a antecedente |
| Maneja datos, razonamiento abajo-arriba | Maneja metas, razonamiento arriba-abajo |
| Encuentra que soluciones siguen a los hechos | Encuentra que hechos pueden soportar la hipótesis |
| Búsqueda en amplitud | Búsqueda en profundidad |
| Antecedente determina la búsqueda | Consecuente determina la búsqueda |
| No facilita la explicación | Facilita de explicación |

Algunas caracterfsticas de los encadenamientos

Las características de esta tabla deben ser tomados sólo como una guía. Esta puede ser utilizada para hacer diagnósticos en un sistema de encadenamiento hacia adelante y de planeación en uno de encadenamiento hacia atrás.

En particular la explicación es fácil en el encadenamiento hacia atrás porque el sistema puede explicar que meta se está tratando de satisfacer, el encadenamiento hacia-adelante no es fácil de explicar porque las submetas no son pueden ser explicadas se sabe solamente que son descubiertas.

El encadenamiento hacia adelante también es llamado razonamiento abajo-arriba porque este razonamiento desde el bajo nivel de evidencia, hechos para el alto nivel esta basado en los hechos.

Un importante aspecto para obtener evidencia es hacer las preguntas adecuadas, estas permiten aumentar la eficiencia al determinar la respuesta correcta. Un requerimiento obvio para esto es que el sistema experto podría solo hacer preguntas referentes a la hipótesis que se está tratando de probar, evitando hacer preguntas no necesarias y sólo hacerlas acerca del tema, reduce el costo en tiempo y dinero para obtener las respuestas que contesten estas preguntas. Esta evidencia puede ser acumulada para ser utilizada en otros casos similares y evitar repetir cuestionamientos.

Idealmente el sistema experto debe permitir al usuario información voluntariamente acerca del tema para lograr procesos más veloces de encadenamiento hacia-atrás y haciendo el sistema más conveniente para el usuario. Este tipo de evidencia permite al sistema avanzar algunas ligas en el encadenamiento causal.

El encadenamientos hacia adelante y hacia atrás, son dibujados por diagramas de árbol debido a la simplicidad que estos para ser comprendidos. Una buena aplicación para encadenamiento hacia-adelante ocurre si el árbol es amplio y no muy profunda. Esto es porque el encadenamiento hacia adelante facilita una búsqueda a lo ancho, porque la búsqueda por conclusiones va nivel

por nivel, en contraste el encadenamiento hacia-atrás facilita una búsqueda en profundidad, un buen árbol para búsqueda en profundidad es angosto y profundo.

Cuando se llevan a cabo encadenamientos en estructuras de reglas de producción, la estructura de las reglas determina la búsqueda de la posible solución, es decir, la activación de una regla depende de los patrones que esta regla diseñe, los patrones en el lado derecho de la regla determinan si la regla puede llegar a ser activada por lo actos, las acciones del lado izquierdo de la regla determinan los actos que son acertados y como afectan a otras reglas. Una situación análoga existe para el encadenamiento hacia atrás excepto que estas hipótesis son usadas en vez de reglas, claro una hipótesis intermedia puede ser una regla siendo usada en este como consecuente en vez un antecedente.

Como un ejemplo considere las siguientes reglas:

IF A THEN B
IF B THEN C
IF C THEN D

Si el hecho A es dado y la máquina de inferencia es diseñada para emparejar hechos contra antecedentes, entonces los hechos intermedio B y C pueden ser correctos y D la conclusión también lo será, este proceso corresponde a un encadenamiento hacia adelante.

En contraste si el hecho (actualmente hipótesis) D es acertada y la máquina de inferencia compara actos contra consecuencias, el resultado corresponde a un encadenamiento hacia atrás

El encadenamiento hacia atrás puede ser realizado en un sistema de encadenamiento hacia adelante y viceversa, por ejemplo las reglas previas pueden ser escritas como

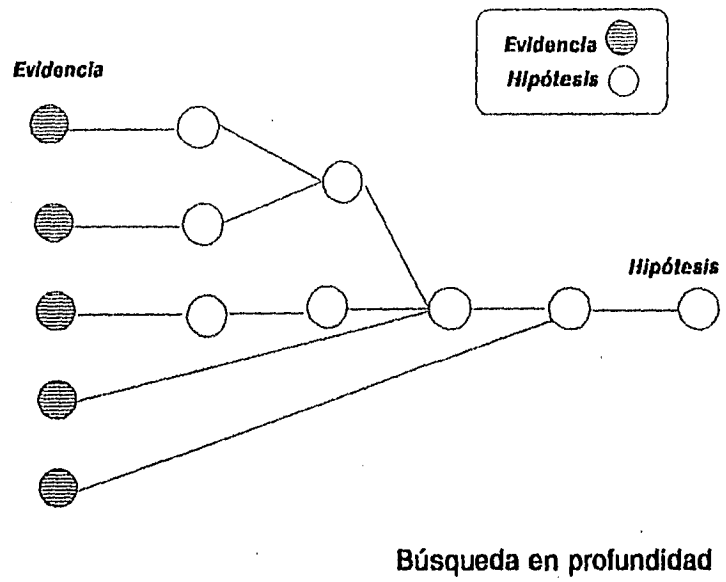



Fig. 4.8 Encadenamiento hacia atrás

IF D THEN C
IF C THEN B
IF B THEN A

Ahora C y B son considerados como submetas o hipótesis intermedias que deben ser satisfechas para satisfacer la hipótesis D. La evidencia A es un hecho que indica el fin de la creación de submetas, si este es un hecho A, entonces D es soportado y considerado cierto bajo esta inferencia de encadenamiento hacia atrás. Si no es A, entonces la hipótesis D no puede ser soportado y se considera falsa.

"Una dificultad con este tipo de inferencias es la eficiencia, un sistema de encadenamiento hacia atrás facilita la búsqueda en profundidad, mientras que un sistema de encadenamiento hacia

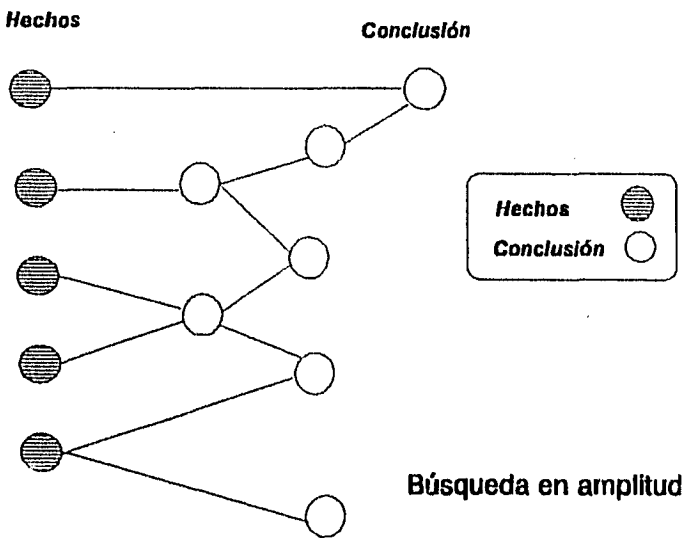



Fig. 4.9 Encadenamiento hacia adelante

adelante facilita un búsqueda en amplitud"[1]. Sí se lleva a cabo una aplicación con encadenamiento hacia atrás en un sistema de encadenamiento hacia adelante y viceversa el sistema no será eficiente en la búsqueda de una solución. La segunda dificultad es conceptual, el conocimiento tomado del experto tendrá que ser alterado para conocer las demandas de la máquina de inferencia. Por ejemplo, la máquina de inferencia de un encadenamiento hacia adelante iguala las reglas de el antecedente mientras que en un encadenamiento hacia atrás se igualan al consecuente. Esto es si el conocimiento del experto es naturalmente un encadenamiento hacia atrás tendrá que ser totalmente reestructurado para adaptarlo a un modo de encadenamiento hacia adelante y viceversa.

---

[1] Expert systems principles and programing
Giatarrano, Joseph, PWS-Kent Publishing Co.

# CAPITULO 5

## Herramientas para el diseño de sistemas expertos

### 5.1 Lenguajes

En los inicios de la ingeniería del conocimiento se desarrollaron importantes sistemas expertos como los fueron DENDRAL, MACSYMA, y MYCIN, estos fueron escritos en un lenguaje de programación llamado LISP que es un lenguaje muy importante para el desarrollo de aplicaciones en inteligencia artificial.

En la actualidad existen herramientas que proporcionan ayuda en cuanto a diseño de sistemas expertos a los ingenieros en conocimiento, estas herramientas son llamadas shells. Algunos de estos también fueron escritos en LISP o PROLOG que es otro de los lenguajes pioneros para el desarrollo de aplicaciones en inteligencia artificial.

Tiempo después se desarrollaría otro lenguaje también muy importante para generar este tipo de aplicaciones como OPS5 con el cual se han desarrollado los sistemas. como lo son XCON Y XSEL.

Las herramientas para desarrollo de sistemas expertos como lo son LISP, PROLOG y OPS5 obedece a varias razones como lo son el que los desarrolladores necesiten características específicas en la aplicación, que el lenguaje sea capaz de proporcionar algunas de las más importantes son una eficiente manipulación de símbolos y soportar el crecimiento del programa.

Existen máquinas LISP y PROLOG estas han sido especialmente diseñadas para trabajar en base

a los requerimientos de estos lenguajes ya que su tipo de programación no es la programación estructurada clásica sino programación orientada a símbolos como se ha venido estudiando.

Desarrollos en lenguajes como C y Pascal tienden a ser más eficientes en computadoras personales que aplicaciones desarrollas en lenguajes de inteligencia artificial.

**LISP (LISt Processor)**

LISP fue el primer lenguaje de procesamiento simbólico fue desarrollado en 1958 en el Instituto de Tecnología de Massachusetts por John McCarty. Inicialmente se desarrollo como un lenguaje de programación con el que se lograra crear aplicaciones que fueran capaces de pensar.

La característica principal de LISP a la cual debe gran parte de su fama es la capacidad de manipulación de símbolos, por lo cual es el lenguaje con el que se han desarrollado muchas aplicaciones en inteligencia artificial.

Algunas características de LISP es el uso de listas, es decir, colecciones de campos contenidos en paréntesis para representar asociaciones entre símbolos.

Por ejemplo:
(+ 54 267)
(IRENE RAUL RODOLFO RAYMUNDO)
(VEHICULOS (AVION(DC9 DC10) AUTOMOVIL (FORD RENAULT)

En LISP el tamaño y forma de una lista puede cambiar continuamente sin tener algún tipo de efecto en la ejecución del programa.

La estructura básica de lisp se compone de combinaciones de los siguientes elementos: átomos, listas y procedimientos.

El átomo es la estructura más pequeña puede ser un número o un símbolo (símbolos porque representan algo en particular)

Ejemplos

| | | |
|---|---|---|
| 33.6 | RAUL | A |
| 13 | ES-MENOR-QUE | + |

Los procedimientos están incluidos en las listas para determinar qué hacer con los átomos son el primer elemento en cada lista, los elementos siguientes son los argumentos del procedimiento.

Existen dos tipos de procedimientos en LISP:

A) PRIMITIVOS. estos son dados por el lenguaje como: +, -, EQUAL

B) Procedimientos definidos por el programador a través de la función DEFUN (DEFinit FUNction)

Ejemplo:
(DEFUN MITAD(N) (QUOTIENT N 2)

Otras características de este lenguaje son la recursividad, intérpretes interactivos, el tratamiento de programas y los ambientes para programar.

**PROLOG (PROgramming in LOGic)**

Fue desarrollado en Francia en 1973 por Alain Colmerauer y su equipo de desarrollo en la universidad de Marsella. Fue utilizado en el procesamiento del lenguaje natural, más adelante se empleo para desarrollar aplicaciones en inteligencia artificial por su facilidad para manipular de símbolos.

El átomo es la estructura más pequeña puede ser un número o un símbolo (símbolos porque representan algo en particular)

Ejemplos

| | | |
|---|---|---|
| 33.6 | RAUL | A |
| 13 | ES-MENOR-QUE | + |

Los procedimientos están incluidos en las listas para determinar qué hacer con los átomos son el primer elemento en cada lista, los elementos siguientes son los argumentos del procedimiento.

Existen dos tipos de procedimientos en LISP:

A) PRIMITIVOS. estos son dados por el lenguaje como: +, -, EQUAL

B) Procedimientos definidos por el programador a través de la función DEFUN (DEFinit FUNction)

Ejemplo:
(DEFUN MITAD(N) (QUOTIENT N 2)

Otras características de este lenguaje son la recursividad, intérpretes interactivos, el tratamiento de programas y los ambientes para programar.

**PROLOG (PROgramming in LOGic)**

Fue desarrollado en Francia en 1973 por Alain Colmerauer y su equipo de desarrollo en la universidad de Marsella. Fue utilizado en el procesamiento del lenguaje natural, más adelante se empleo para desarrollar aplicaciones en inteligencia artificial por su facilidad para manipular de símbolos.

PROLOG soporta la representación lógica del conocimiento, los programadores describen sus problemas en términos de hechos y reglas a los que PROLOG busca las posibles soluciones.

Un programa en prolog se compone de hechos, reglas y preguntas, los hechos se utilizan para representar la existencia de relaciones entre los objetos por ejemplo la sentencia "Raúl es dueño de la casa" se compone de dos objetos "Raúl" y "casa" y la relación entre ellos es "dueño"

dueño (raul, casa)

Una pregunta en PROLOG va precedida por el símbolo de interrogación ?- . Utilizando el ejemplo anterior

?- dueño (raul, casa)

Para dar una respuesta PROLOG debe hacer una búsqueda en la base de conocimientos hasta encontrar el hecho que se empareje con la pregunta.

Una regla en PROLOG tiene la forma: F1 :-F2 el símbolo :- significa "Si" y F1, F2 son los hechos. F1 es llamado la cabeza de la regla y describe el hecho que la regla está tratando de establecer, F2 es llamado el cuerpo de la regla y representa la meta que se intenta satisfacer si la cabeza es cierta.

Por ejemplo:

hermana(eva, miguel):-
femenino(eva)
padres(miguel, P)
padres(eva, P)

Lo cual significa:

Eva es hermana de Miguel si
Eva es mujer y
los papas de Eva son P y
los papas de Miguel son P

PROLOG trabaja con diferentes tipos de estructuras de datos como lo son: estructuras, constantes (átomos y enteros) y listas.

Las estructuras son objetos que contienen colecciones de objetos a los que se llama componentes.

Una lista es una colección de elementos como lo son los átomos y estructuras, colocados bajo un cierto orden entre corchetes.

Entre algunas de las características más importantes de PROLOG se encuentra su mecanismo de control este se compone de encadenamiento hacia atrás y corte. Mediante este encadenamiento prolog puede regresar a la submeta anterior, después de darse cuenta que el paso siguiente no cumple la submeta regresa a la anterior y a partir de esta busca una nueva solución. El corte es utilizado por el programador para detener el encadenamiento hacia-atrás por alguna circunstancia como lo podría ser dejar libres espacios de memoria que una búsqueda exhaustiva podría ocupar.

**OPS5 (Official Production System)**

Este lenguaje trabaja mediante la representación del conocimiento basado en reglas, incorpora un patrón de igualación y un intérprete que trabaja en base a encadenamiento hacia adelante, además de otras herramientas de programación.

Este lenguaje fue diseñado para el desarrollo de sistemas en base a reglas de producción y su máquina de inferencia trabaja en base a encadenamiento de abajo hacia arriba. OPS5 tiene más

parecido con un lenguaje de programación que con un shell de sistema experto, no contiene facilidades de explicación, ni una interfaz para el usuario. Este lenguaje ha sido usado con mucho éxito para construcción de sistemas expertos.

Fue desarrollado en la Universidad de Carnegie-Mellon. En el diseño y desarrollo de sistemas de producción, entre otros por C.L. Forgy, J. McDermott, A. Newell y M. Rychener. Algunos de los lenguajes y sistemas que surgieron de este largo período de investigación fueron: PSG, PSNLST, y OPS. OPS experimentó un número de cambios que llevaron al lenguaje OPS5 propiamente dicho. El lenguaje es principalmente utilizado para construir s.e. y se volvió popular después de su exitosa aplicación en el desarrollo del s.e. XCON. Se aconseja sin embargo, utilizar el lenguaje en el desarrollo de otro tipo de aplicaciones dentro del campo del tratamiento simbólico también (symbolic processing).

La mayor parte de un programa OPS5 es el establecimiento de reglas de producción que constituyen la sección de producción del programa. Estas reglas de producción ó "productions" como son llamadas en el contexto OPS5 emplean una representación "object-attribute-value" los objetos y atributos que se mencionan en las reglas de producción tienen que ser declarados en la sección de declaración (declaration section) del programa. Una declaración objeto se forma de la instrucción "literalize" seguida del nombre del objeto declarado y de todos sus atributos asociados. El nombre de un objeto es también llamado "class name".

"Durante la ejecución de un programa OPS5, un objeto puede volverse "instantiated" o sea, una copia o "instantiation" de un objeto puede ser creada y en la cual constantes han sido introducidas (fill in) para sus atributos. Las variables (o valores) asignado(a)s a los atributos son ya sea constantes ó el valor especial "nil" que indica que el valor (o variable) del atributo es desconocido. Se distinguen dos tipos diferentes de constantes: constantes numéricas llamadas simplemente "numbers" y constantes simbólicas llamadas "atoms". La "instantiation" creada después es insertada en la memoria de trabajo o "working memory" como es llamado en OPS5. Tan pronto como la instantiation de un objeto se encuentra en memoria de trabajo se le da el

nombre de "working-memory-element" (w.m.e.). A cada w.m.e. se le asigna un entero positivo llamado "time tag" por medio del cual los w.m.e. son identificados sin confusión."[1]

## 5.2 Shells

Existen diferentes tipos de herramientas para la construcción de sistemas expertos, entre estas existen los lenguajes de programación orientados a programación en inteligencia artificial como lo son LISP, PROLOG, OPS5 y los lenguajes tradicionales como lo son C y PASCAL, además de estos otras herramientas muy utilizadas en la actualidad por su eficiencia en la construcción de sistemas expertos,y las facilidades que presentan son los denominados SHELLS, este software contiene los componentes que permiten diseñar y construir sistemas expertos.

Como se ha tratado en capítulos anteriores los componentes básicos de un sistema experto son: la máquina de inferencia, la base de conocimientos, los sistemas de interface con el usuario, las facilidades de explicación y las facilidades para la adquisición de conocimientos. Cuando un sistema experto es completado estos cinco componentes trabajan en forma conjunta para llegar al objetivo por el cual fue creado.

Los componentes de los shells varían de acuerdo al objetivo de su diseño etc. pero todos contienen estos elementos:

1. Ayudas para los desarrollos de los ingenieros en conocimientos
2. Ayudas para la adquisición de conocimientos
3. Herramientas para la administración del conocimiento
4. Máquinas de inferencia
5. Interfaces usuario_sistema

---

[1] Expert systems for experts
Parsaye, Kamran. Wiley Inter-Science, 1990

6. Facilidades de explicación
7. Interfaces de lenguaje

El desarrollo de ayudas para los ingenieros en conocimiento proveen las herramientas necesarias para construir sistemas expertos que cubran los objetivos para los que fueron creados. Estas ayudas incluyen herramientas para dar formato a pantallas, como lo es la integración de atributos como el color, presentación etc. El desarrollo de ayudas gráficas, y editores con diferentes tipos de propósitos. por ejemplo los que ayudan en la creación, borrado o modificación de capacidades, como lo es el texto en el que se encuentran las recomendaciones adecuadas después de una consulta.

Las herramientas de inducción, los debuggers, los editores de la base de conocimientos, los intérpretes y las herramientas que revisan la consistencia son ejemplos de muchas de las herramientas que deben ser incluidas en los shells para ayudar en el desarrollo de bases de datos de conocimientos. Existen editores que revisan la existencia de posibles errores en la sintaxis, gramática y estructura de la base de conocimientos, otros encuentran el error y dan una explicación al usuario de porque esta mal. Los programas diseñados para revisar por completo el trabajo de la misma forma que un editor, estos programas revisan si faltan datos en las entidades en la base de conocimientos, revisan consistencia y redundancia en los datos por ejemplo buscan conflictos de información.

Como nos hemos dado cuenta la adquisición del conocimiento es considerada la parte crítica en el desarrollo de los sistemas expertos, los editores del conocimiento y las herramientas de inducción son ejemplos de ayudas para la adquisición del conocimiento en un sistema experto. Los editores son una clase especial que apoyan a los constructores de sistemas expertos en la creación y de la base de conocimientos con datos, estos frecuentemente son configurados para aceptar datos que se adapten a la forma en que se basará la representación del conocimiento.

Las herramientas de inducción son capaces de desarrollar datos para la base de conocimientos directamente desde la fuente de información como lo son las bases de datos en computadoras,

registros manuales etc. utilizando estas herramientas más de la mitad de reglas de un sistema experto pueden ser desarrolladas automáticamente reduciendo el tiempo de generación.

De un buen diseño de la interface usuario-sistema depende la aceptación que el sistema experto pueda obtener del usuario esta interface puede incluir generadores de menú, módulos de lenguaje natural y módulos para presentaciones gráficas, estas facilidades permiten al ingeniero en conocimientos generar interfaces que el usuario podrá manejar intuitivamente.

Las herramientas para la administración del conocimiento aceptan datos desde el constructor del sistema experto, los almacena, los administra para que la máquina de inferencia o las facilidades de explicación puedan utilizarlo.

La máquina de inferencia es un componente fundamental del sistema experto bajo desarrollo en muchos casos estos componentes proporcionan todo el control lógico que el sistema experto requiere en otros casos una pequeña porción de los requerimientos lógicos del sistema pueden ser programados por el ingeniero en conocimientos y cargados en la base de conocimientos.

Las facilidades de explicación permiten que el ingeniero en conocimientos incluya en su sistema el componente explicativo de el porque se llego a tal conclusión, dado que el usuario final puede no ser experto en el tema a tratar. Tomando esto en consideración, algunas formas de explicación son incluidas.

El lenguaje de interface permiten que un sistema experto pueda comunicarse con un programa externo. Estas interfaces generan un conducto mediante el cual los datos pueden ser pasados entre el sistema experto y otros programas.

Los Shells contienen un grupo limitado de formas para la representación del conocimiento y motores de inferencia que controlen la parte lógica del sistema. La base de conocimientos es la parte del sistema experto que contiene datos acerca de un tema específico. El domino del

conocimiento puede ser organizado de varias formas, reglas de producción, marcos, redes semánticas, lógica de primer orden etc.

Algunos componentes y herramientas de los shells para Sistemas expertos son desarrollados con el propósito de soportar la creación de sistemas expertos, otros son desarrollados para ayudar a los usuarios en la utilización de su sistema, aun otros son desarrollados para soportar ambos el ambiente de desarrollo y la aplicación en si.

# ANEXO

## Ejemplo práctico: elaboración de un prototipo

Así como en el desarrollo de sistemas convencionales, también la Inteligencia Artificial cuenta con etapas para el desarrollo de sistemas basados en el conocimiento. Estás son: analizar el problema, diseñar el sistema, construir un prototipo y verificar el prototipo de modo progresivo. Estos estados son equivalentes a las fases clásicas de desarrollo. Las variaciones principales son a nivel de detalle.

| Estados AI | Etapas clásicas de desarrollo |
|---|---|
| Analizar el problema | Plan |
| Diseñar el sistema | Diseño |
| Construir un prototipo | Editar, codificar y compilar |
| Verificar el prototipo | Probar |

En la etapa de análisis se elige un problema, se determina sí el problema sería solucionable mediante técnicas de inteligencia artificial, y se determina que tipo de conocimiento es necesario y quien podría proporcionarlo de modo eficaz. En una fase intermedia entre el análisis y el diseño, el ingeniero en conocimientos adquiere el conocimiento que requiere el programa.

Durante la fase de diseño se organiza el saber del modo que más se asemeja a la clasificación del mismo que el especialista humano desarrolla durante su trabajo y crea una estructura básica para fundamentar el sistema, una estructura típica es la jerárquica. Por ejemplo en diagnóstico

de averías de coches pueden estructurarse los problemas de acuerdo al lugar en el que está localizada la avería, en caso de ser necesario, el especialista, y a su vez el ingeniero de conocimientos subdividirá aun más la estructura jerárquica de diagnóstico de averías hasta alcanzar finalmente suficiente nivel de detalle como para determinar que el problema.

Durante la etapa siguiente se efectúan dos tipos de tarea, en una de ella los diseñadores del sistema colocan información detallada de la aplicación para cada una de las etapas de la jerarquía. Dependiendo del sistema y de sus métodos de representación del conocimiento, dicha información detallada puede incluir diagnóstico u otras reglas de aplicación, características de las estructuras físicas, valores aceptables que los usuarios finales pueden introducir y otra información necesaria para el normal funcionamiento del sistema. Esta información se escribe en el lenguaje de programación que proporcione la herramienta de diseño. Los conocimientos de estos detalles permiten que el sistema experto avance a través de la jerarquía siguiendo la estrategia de razonamiento que utilice la herramienta (encadenamiento hacia atrás, adelante o una combinación de ambos) hasta concluir en el diagnóstico específico que se señala en la parte inferior de la jerarquía.

El segundo tipo de información introducida durante la fase de construcción es la información de control. Esta se encuentra en las reglas que controlan la consulta del usuario final real, controlando el orden en el que se deben buscar ambas hipótesis siguiendo las diferentes líneas de razonamiento y haciendo conocer al sistema cuando se ha resuelto el problema.

Las reglas de control son más sofisticadas y más difíciles de usar correctamente que las reglas sobre los conocimientos del tema de la aplicación, que son del tipos Si el coche no arranca Entonces verificar si tienen gasolina.

En la etapa de verificación del programa, el diseñador del sistema prueba los conocimientos de éste con varios problemas del tipo de los cuales el sistema tiene la capacidad de resolver. Si la ejecución de los resultados de prueba produce resultados erróneos, hay que consultar con el especialista humano, normalmente el puede explicar las razones del por que el sistema da estos

resultado y recomendar los conocimientos que hay que añadir al sistema para que no repita las mismas equivocaciones. Estos nuevos conocimientos se codifican como nuevas reglas y se añaden al sistema, que se verifica de nuevo con más casos de prueba.

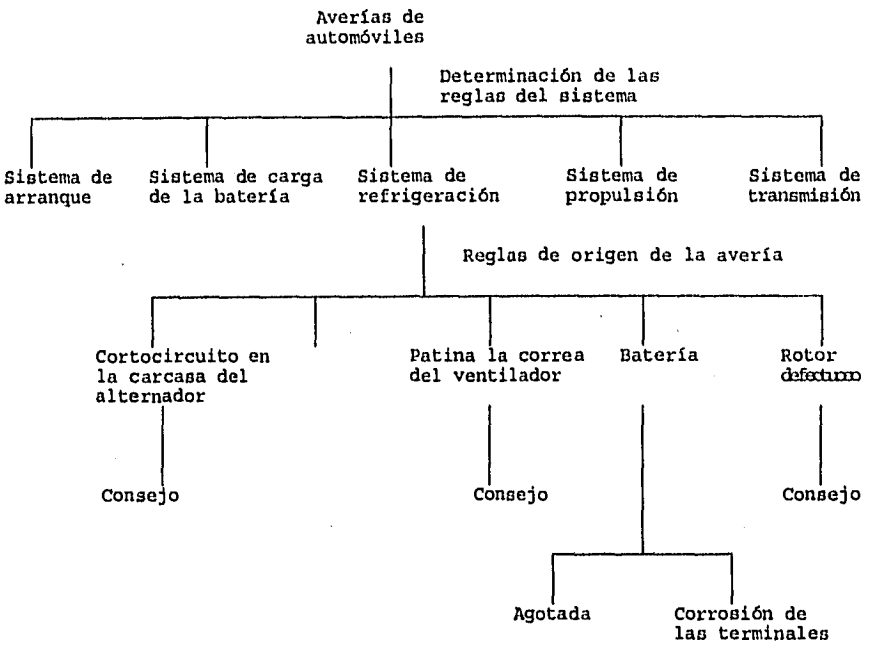
Cuando reiteradamente se han hecho suficientes pruebas, modificaciones o adiciones graduales de conocimientos, se consigue un sistema basado en el conocimiento que consistentemente proporciones respuestas, consejos y diagnósticos acertados en una gran variedad de casos y, consecuentemente, puede ser verificado en el campo. La verificación en el campo descubre gradualmente nuevos casos que el sistema no puede resolver adecuadamente, y también nuevos errores que no se habían advertido previamente.

El manejo de los nuevos casos y errores requiere la adición de más reglas. De este modo el sistema madura gradualmente hasta el punto de poder resolver casi todos los problemas a los que normalmente se enfrenta el especialista humano.

## CONSTRUCCION DE UN PROTOTIPO

En esta parte explicaremos como construir, verificar y modificar un prototipo de un sistema basado en el conocimiento usando herramientas de desarrollo de aplicaciones de Inteligencia Artificial basadas en computadoras personales. La herranienta que mostraremos se llama M.1 y permitirá sus correspondientes metodologías de desarrollo.

En este ejemplo se construirá un sencillo prototipo de sistemas basados en el conocimiento, denominado system.car para diagnostico de averías en automóviles. La figura siguiente muestra la estructura de system.car.




Estructura básica de un sistema experto de diagnóstico de averías en un automóvil

El sistema basado en el conocimiento así construido atenderá una consulta del usuario y le preguntará sobre la sintomatología del coche. En base a los síntomas, sugerirá el sistema en que posiblemente se encuentra la avería y las pruebas a efectuar sobre los diversos componentes del sistema en cuestión. Dependiendo de los resultados de la prueba, sugerirá la causa del problema y ofrecerá una acción correctiva.

Si asumimos que se ha terminado el análisis y el diseño del sistemas basados en el conocimiento, para construirlo, la primera tarea que tiene que realizar el usuario de la herramienta es

desarrollar una base de conocimientos. En el M.1 para este propósito se puede utilizar un editor de textos estándar.

El primer punto necesario en una base cognitiva M.1 es un objetivo. En un sistema de encadenamiento regresivo tal como el M.1, el objetivo primario o de nivel superior es normalmente de tipo muy general; por ejemplo, proporcionar consejo al usuario. Durante una consulta, el sistemas basados en el conocimiento busca cualquier regla que concluya que el consejo a dar al usuario es tal y tal. Tal y tal puede ser una variable que tiene vigencia en los consejos relativos a la conclusión de diagnóstico del sistemas basados en el conocimiento, determinada en la parte inferior de la jerarquía del sistemas basados en el conocimiento y como resultado de la conclusión de varias reglas.

Durante la ejecución, M.1 intenta las diferentes reglas que concluyen las causas hipotéticas de una avería a fin de ver qué requisitos de ellas se pueden satisfacer. Las partes condicionales se satisfacen en función de los resultados de las pruebas, síntomas y el sistema que manifiesta la avería. Mediante encadenamientos regresivos, M.1 explora la estructura jerárquica buscando reglas que concluyan los síntomas apreciados en el sistema que se está diagnosticando, hasta que finalmente por encadenamiento regresivo alcanza un punto suficientemente alto en la jerarquía como para satisfacer las precondiciones, y por tanto, las conclusiones de una vía jerárquica. De este modo, M.1 verifica que la causa hipotética y el consejo al usuario son correctos.

Para simplificar la explicación, el prototipo inicial ilustrativo de un sistema de diagnóstico de automóviles será muy reducido, y su objetivo será determinar la causa de una avería en el sistema de carga de la batería. Este microsistemas basados en el conocimiento contendrá un objetivo, una regla, dos preguntas y dos conjuntos de valores aceptables como respuestas a las preguntas. Entonces el sistema se ampliará para mostrar cómo se puede construir un sistema mayor y más sofisticado.

El primer paso para construir la base cognitiva system.car es definir un objetivo. Para este microsistema, el objetivo definido es la causa de la avería en el sistema de carga de la avería.

Por tanto, el diseñador del sistema escribe:

> > add meta = causa-falla-sistema-encendido.

La palabra <<add>> es el mandato de M.1 para añadir una entrada a la base de conocimientos. Cada entrada debe terminar con un punto para indicar el final de una entrada al M.1.

Ahora que el sistema de base de conocimientos tiene un objetivo, necesita algunas reglas para ayudarle a conseguir dicho objetivo. El especialista en automóviles que proporciona el saber para system.car ha determinado dos pruebas, una prueba denominada del osciloscopio y la otra, prueba de arco de campo, que se pueden efectuar sobre el sistema de carga de batería para determinar si hay o no hay un corto en la carcasa del alternador. Por tanto, el diseñador del sistema codifica este conocimiento en forma de regla:

```
> > add if resultado-prueba-osciloscopio =
        linea-recta-and resultado-prueba-arc=amarillo
        then causa-falla-en-encendido  =
        alternador-tiene-corto
```

Bajo petición, M.1 listará las entradas en la base cognitiva desarrollada hasta ahora. Para comprender la elevada interactividad de las herramientas de desarrollo de sistemas de base cognitiva, cuan rápidamente se pueden construir los prototipos del sistema, y cuan fácilmente se pueden añadir conocimientos incrementalmente, ahora ya podríamos ejecutar este sistemas basados en el conocimiento de diagnóstico de automóviles, pese a que sólo tiene dos entradas en su base cognitiva. Utilizará su única regla para intentar satisfacer su objetivo. La herramienta M.1 proporciona un formato por omisión para preguntar a los usuarios sobre los resultados de las pruebas, atributos, objetivos u observaciones necesarias para satisfacer las condiciones de una regla cuando el sistema no puede inferir la información de otras reglas o hechos. El formato por omisión es < < ¿Cuál es el valor de (lo que aparezca después de la palabra *si*)? > >

| Eventos | Conclusiones |
|---|---|
| Buscando resultado-de-prueba-osciloscopio<br>Encontró resultado-de prueba-osciloscopio<br>buscando resultado-prueba-arc | resultado-prueba-osciloscopio<br>= línea-recta |

| Razonamiento | Opciones |
|---|---|
| Invocando kb-2<br>Si resultado-de-prueba-osciloscopio<br>= linea-recta<br>and resultado-prueba-arc = amarillo<br>entonces causa-falla-en-encendido = alternador-tiene-corto | Cualquier<br>expresión |

En la figura se muestra la ejecución del sistema de una regla, un objetivo. Conforme el sistema se ejecuta, M.1 lleva a cabo la consulta del usuario en la mitad inferior de la pantalla, mientras que su entorno de depuración interactiva muestra determinados hechos relevantes en cuatro ventanas ubicadas en la parte superior de la pantalla. La pantalla de eventos muestra eventos significativos u ocurrencias tales como resultados o atributos que M.1 ha encontrado o está buscando. La ventana de razonamiento muestra los razonamientos del M.1 conforme busca una conclusión. Su razonamiento consiste en reglas actualmente invocadas, o aquellas que ya han alcanzado el éxito. La ventana de conclusiones muestra las conclusiones que M.1 ha alcanzado basándose en inferencias o en respuestas dadas a los usuarios. La ventana de opciones muestra las respuestas que M.1 acepta de los usuarios. La ejecución del sistema revela que en base a la evidencia disponible, la carcasa del alternador tiene un corto. Puesto que la regla 2 de entrada a la base de conocimientos no especifica un valor de certidumbre en su conclusión, se asume un factor de certidumbre del 100%.

Durante la consulta, si el usuario ha preguntado "por qué" el sistema le ha hecho determinada pregunta, M.1 le respondería que necesita la respuesta como ayuda para determinar si la regla invocada es aplicable en la consulta que se está realizando. Además, conforme se ejecuta el sistema M.1, la ventana de razonamiento cambia continuamente para mantener al usuario informado de lo que el sistema está haciendo y por qué.

Es útil hacer más preguntas que las que permite el formato por omisión, a fin de que la base de conocimientos pueda preguntar a los usuarios sobre los resultados de una prueba o los síntomas de una avería, además es práctico identificar ante el sistema un restringido grupo de respuestas apropiadas o "valores legales" que el sistema puede aceptar de los usuarios. Aunque esto no es esencial para el propio sistema, le previene de fallar en el caso de que el usuario de alguna respuesta incongruente, o alguna sobre la que el programa no tenga la correspondiente regla. La prueba de infalibilidad tiene lugar cuando el sistema verifica la respuesta del usuario respecto a una lista de opciones posibles sin aceptar otras repuestas. Una ventaja adicional de introducir valores legales para diferentes preguntas es que durante la consulta, cuando M.I pregunta a los usuarios, les muestra en la pantalla las respuestas adecuadas que pueden seleccionar. En la figura siguiente se muestra una ejecución del sistema modificado system.car con las frases relativas a las preguntas del programador y las restricciones sobre las respuestas de los usuarios.

| Eventos | Conclusiones |
|---|---|
| Buscando resultado-de-prueba-arc<br>Encontró resultado-de prueba- arc<br>Encontró causa-falla-en-encendido | causa-falla-en-encendido<br>= alternador-tiene-corto |

| Razonamiento | Opciones |
|---|---|
| Invocando kb-2<br>Si resultado-de-probar<br>= linea-recta<br>and resultado-prueba-arc = amarillo<br>entonces causa-falla-en-encendido = alternador-tiene-corto | ninguno<br>azul<br>amarillo |

Ejecute una prueba oscilatoria estándar. ¿Qué tipo de patrón desea ver?

> > linea-indicadora

Encienda el swicht y desconecte el alternador. ¿Qué tipo de parpadeo observa?

> > Amarillo

causa-falla-en-encendido = alternador-tiene-corto

(100%) porque kb-2

### Adición de reglas

Evidentemente hay muchas causas posibles de averías en el sistema de encendido. Por lo tanto el sistema deberá contener tantas reglas como posibilidades, preguntas y valores legales permisibles para cada regla.

El especialista que proporciona los conocimientos a system.car sabe tres causas de avería en el sistema de carga: la correa del ventilador patina, la armadura tiene corto y está en corto el rotor. También sabe por experiencia que puede diferenciar entre distintas causas: observar la correa del ventilador, examinar los resultados de algunas pruebas estándar, y examinar algunas piezas y conexiones eléctricas y mecánicas del coche.

Durante la ejecución del sistema ampliado, M.1 verifica cada una de las reglas en orden consecutivo. Por ejemplo, invocará primero la regla 2 de la base de conocimientos, y ello le llevará a observar que los resultados de la prueba del osciloscopio es una línea recta, pero que en la prueba del arco no se produce "chispa". Por tanto, la regla 2 de la base de conocimientos falla. A continuación, M.1 invoca la regla 7 de la base. Si el usuario dice que la correa del ventilador no está rota, la regla 7 de la base falla, y se invoca la regla 10 de la base.

Las preguntas y los valores legales no tienen porqué estar en orden numérico. Por el contrario, una vez que se invoca una regla y M.1 comienza a verificar sus precondiciones, M.1 busca otras entradas de la base de conocimientos que concluyen valores válidos para las precondiciones que se están verificando o que contienen las expresiones de las precondiciones. Por ejemplo, cuando verifica la condición "Si el resultado de la prueba del osciloscopio = línea recta", M.1 encontrará la respuesta preguntando por la prueba del osciloscopio y las entradas que contienen los valores legales correspondientes a los resultados de la prueba del osciloscopio. También encontrará cualesquiera otras reglas que concluyan los resultados de esta prueba.

M.1 no procede a verificar la precondición siguiente o regla siguiente hasta que ha obtenido toda la información que necesita de estas entradas interrelacionadas de la base de conocimientos, ya

sea tomando nota de los hechos, infiriendo información, o preguntando al usuario. Esta persistencia en un tema antes de pasar al siguiente, aumenta la eficiencia y se asemeja al modo en que los seres humanos efectúan diagnósticos, especialmente en las grandes aplicaciones, donde hay que dilucidar entre muchas posibles averías en muchos subsistemas. Generalmente, los seres humanos no manejan muchos sistemas diferentes, solicitan los resultados de las pruebas, o hacen preguntas fortuitas y aleatorias.

**Como se controla una consulta**

Conforme se añaden más reglas a las base de conocimientos, resulta cada vez más ineficiente comprobar cada regla relevante o irrelevante. Por ejemplo, si se ha determinado que existe un problema en el sistema de enfriamiento, no tiene sentido repasar todas las reglas o comprobar todas las piezas del sistema de transmisión y del motor. Para evitar reciclar todas las reglas irrelevantes, system.car determina primero los síntomas iniciales y concreta en qué sistema, probablemente, está la avería. Finalmente, no sólo determina la causa de la avería, sino que también muestra las acciones correctivas aconsejables.

Para garantizar que los procesos de diagnóstico se efectúan en el orden mencionado. los usuarios escriben reglas de control complementarias de las reglas de decisión. A diferencia de las reglas de decisión, las reglas de control no guardan ninguna decisión, las reglas de control no guardan ninguna relación con el tema o materia de la consulta. Las reglas de representación que muestran saludos o consejos a los usuarios, así como las reglas que dicen al sistema que ha terminado una consulta son ejemplos de reglas de control.

Para aumentar la eficacia conforme se va ampliando system.car, se escribe una nueva versión con una sola regla de control que gestiona la consulta en cooperación con los objetivos y subobjetivos del sistema que se generan durante la ejecución del mismo. un objetivo de nivel superior es el objetivo global de un sistema cognitivo. Un objetivo de nivel superior puede ser finalizar la consulta y, en consecuencia, el sistema efectúa todas las tareas de diagnóstico necesarias para finalizar la consulta.

El objetivo de nivel superior de system.car, enumerado a continuación, es proporcionar consejo al consumidor:

meta = consejo-al-cliente

La única regla de control de system.car, que concluye en un consejo al cliente, es la siguiente:

Si sistema = X
    Y causa-falla-en-X = Y
    Y recomendación-para-Y = Z
Entonces consejo-al-cliente = Z

A diferencia de las reglas de decisión que hemos visto hasta ahora, esta regla contiene las variables: X, Y, y Z. X significa cualquier sistema. Y se puede sustituir por cualquier avería en el sistema asignado a X. Z representa una recomendación o consejo al cliente, relativa al tipo de fallo representado por Y.

Par concluir el consejo apropiado para el cliente, system.car debe satisfacer las tres precondiciones de la regla de control, en el orden en que aparecen. Por tanto, la regla de control determina la secuencia de las principales etapas en el proceso de diagnóstico. Cada una de las tres precondiciones: determinación del sistema, localización de la causa de la avería y dar la recomendación consiguiente. Son realmente subobjetivos de system.car. Para satisfacerlos, system.car recicla regresivamente las reglas de la base de conocimientos que concluyen en los subobjetivos del sistema.

**Depuración**

En este punto es importante advertir que, aunque las reglas se pueden añadir fácilmente, de modo incremental, es impreciso aseverar que la adición de una nueva regla no afecta al resto de la base de conocimientos.

Los mecánicos de coches saben que si un coche no arranca, el problema puede ser originado por una batería defectuosa. Sin embargo varias consultas a system.car no diagnosticaron satisfactoriamente un coche que no arrancaba; el sistema nunca sugirió que una batería defectuosa podía ser la causa de la avería.

Sofisticadas interfases interactivas de usuario y herramientas de depuración son herramientas esenciales para entornos de Inteligencia Artificial, y M.1 no es ninguna excepción. Las reglas de M.1 y otras entradas a la base de conocimientos se indexan automáticamente conforme van entrando en el sistema. Consecuentemente, una exploración de las reglas relevantes de la base de conocimientos de system.car se puede obtener pidiendo al depurador de M.1 que muestre en la pantalla todas las reglas que concluyen que la causa de la avería es una batería defectuosa. La pantalla revela que no hay ninguna regla que concluya que la causa de la avería es una batería defectuosa. Por lo tanto, el diseñador del sistema añade la regla siguiente:

kb-21: Si nivel-de-carga = bajo
Y batería-pierde-carga
Entonces causa-de-falla-en sistema-carga = batería-mala

Se reejecuta la consulta con system.car pero éste todavía no concluye que la causa de la avería en una batería en mal estado. Por lo tanto, el diseñador del sistema rexamina las precondiciones de la regla que se acaba de añadir. La primera precondición o prueba es "Si nivel-de-carga = bajo".

Bajo petición, el depurador de M.1 muestra el valor de cualquier expresión para una consulta determinada, y representa también todas las reglas y hechos en la base de conocimientos que contienen información o concluyen un valor para la expresión en cuestión.

En esta consulta, el depurador del sistema muestra que la expresión "nivel-de-carga" no tiene asignado ningún valor. Consecuentemente, no hay reglas o preguntas relativas a nivel de carga.

Por tanto, se necesita otra regla que concluya que el nivel de carga es bajo. En consecuencia, se añade la regla kb-22 al final de la base.

kb-22: Si cuando-el-motor-está-apagado-voltaje-batería = bajo
Or pH-batería = alto
Entonces nivel-de-carga = bajo

Pero para concluir la segunda condición de la regla kb-21 se necesita todavía otra regla:

kb-23: Si corriente-carga-lenta-continua = baja
Entonces batería-pierde-carga

Ampliación del sistema

Se deben hacer más adiciones y cambios a la base de conocimientos antes de que system.car pueda diagnosticar una batería defectuosa. Hay que hacer varias preguntas sobre la carga después de la carga lenta continua. Además, se necesitan valores legales para cada una de dichas respuestas.

La adición de reglas para diagnosticar baterías defectuosas afecta también a entradas anteriores a la base de conocimientos. Por ejemplo, el diseñador del sistema debe dar a system.car algunas pistas sobre potenciales problemas de la batería y asegurarse de que la batería forma parte del sistema de carga. Para proporcionar esta información, se necesita modificar la regla kb-16 que concluye que el problema es el sistema de carga.

El mecánico dice que el chasquido que oye cuando arranca el coche es un síntoma de los problemas de la batería. Sabiendo esto, el diseñador del sistema pide a M.1 que muestre la regla kb-16 y la modifiquen para comprender las relaciones entre los chasquidos que se oyen durante la prueba de arranque y una avería en el sistema de arranque. La regla modificada dice lo siguiente:

kb-16: Si el síntoma=coche-no-arranca
Y resultados-prueba-de-arranque=chasquidos
Entonces sistema=sistema-arranque(80% seguridad)

Un análisis de la pregunta relativa a la prueba de arranque muestra que la pregunta anterior a la base de conocimientos continúa siendo apropiada para contribuir a la inclusión de nueva información. No obstante, la pregunta no puede aceptar aun el resultado, "chasquido". Por lo tanto el diseñador del sistema debe modificar los valores legales que relacionan el resultado del arranque de modo que el programa pueda aceptar "chasquido" como una respuesta y mostrarla como una opción posible entre las varias que aparecen en la ventana de opciones.
Por lo menos, system.car tiene en su base de conocimientos una completa vía jerárquica que corresponde a la estructura de la base de conocimientos que culmina e el diagnóstico de una batería defectuosa, así como todas las preguntas relacionadas y valores legales apropiados para el adecuado manejo de las respuestas. Para añadir características extra hay otros aspectos que los diseñadores de sistemas deben realizar para hacer a system.car lo más semejante posible a las capacidades de un mecánico automótriz.

El usuario puede desconocer la respuesta algunas preguntas del mecánico. En estas condiciones, si los mecánicos no pueden evaluar u observar fácilmente la información requerida, pueden encontrar un modo u otro de continuar su diagnóstico, pese a esta falla. Por ejemplo, además de la regla kb-22, system.car puede tener una segunda regla que concluya también que el nivel de carga es bajo:

kb-24: Si corriente-carga = baja
Entonces nivel-carga = bajo

Si el usuario responde "desconocido" a las preguntas sobre voltaje y pH de la batería de la kb-22, M.1 busca y encuentra kb-24. Si conoce la corriente de carga, puede usar kb-24 en vez de kb-22 para concluir o determinar el nivel de carga.

Como alternativa el usuario puede proporcionar una regla sobre el procedimiento a seguir si se sabe la respuesta. Tal regla puede proporcionar varios diagnósticos posible, y asignar determinado factor de certidumbre (cf) a cada uno de ellos: Este factor de certidumbre no equivale realmente a los factores de aceptación o confianza que se utilizan en el cálculo de probabilidades y estadísticas. Es, por el contrario, una cuantificación de la certidumbre que el especialista humano tiene en su diagnóstico, en base a su experiencia.

Por ejemplo, en una consulta, un usuario puede saber que la corriente de carga de la batería es baja, pero puede ignorar o carecer del tiempo necesario para comprobarlo, si la batería puede sustentar determinada corriente de carga. La regla kb-25, mostrada a continuación, puede manejar satisfactoriamente cualquiera de estos casos.

kb-25: Si se-desconoce-tiene-carga-la-batería
Entonces causa-avería-en-sistema-encendido =
batería-dañada (cf 40%)
Y causa-avería-en-sistema-encendido =
problema-regulador-voltaje (cf 20%)
Y causa-avería-en-sistema-encendido =
alternador-no-carga-batería (cf 20%)

Si durante una consulta un usuario responde "desconocido" a una pregunta sobre "corriente de carga después de carga lenta continua", system.car buscará kb-25 que concierne a esta repuesta desconocida. Las conclusiones en kb-25 se pasan a kb-23, que proporcionará al usuario todas las causas posibles de avería del automóvil de kb-25, junto con sus correspondientes factores de certidumbre. En cierto modo, esta respuesta es equivalente al modo en que el mecánico invoca sus propios factores de certidumbre cuando le responde a un cliente sobre las posibles fallas del automóvil.

# CONCLUSIONES

En los inicios de la inteligencia artificial se pensó llegar a métodos generales en la solución de problemas, intentando trasladar los estudios ya realizados en la solución de problemas a medios computacionales, pero ni el hardware ni el software estaban posibilitados en esos momentos para trabajar en el campo simbólico, además al darse cuenta de que no podía llegarse a una solución general basándose en el como el hombre solucionaba los problemas se intento imitar solo la forma externa de estos mecanismos, fragmentando el estudio en diferentes campos de las capacidades humanas.

Uno de estos campos de desarrollo es el de los sistemas expertos, área a la que se le ha llamado científicamente ingeniería del conocimiento, en la que en este se intenta en base a un sistema computacional llegar a las conclusiones a las que llegaría un experto al hacerle algún tipo de cuestionamiento en un tema específico del que es experto.

Un sistema experto no se comporta como un experto humano, ya que se desconocen los procesos mentales que se realizan al llevar acabo una consulta, pero el resultado final de la consulta a un sistema experto en la mayoría de las veces llega ha ser tan bueno como el del experto humano teniendo la capacidad de explicar su procedimiento y justificar sus resultados.

Algunas de las características principales de los sistemas expertos son: los extensos conocimientos que en el están contenidos de un campo específico del conocimiento humano, el llegar a la solución de problemas en base a un razonamiento propio y el aprender de su experiencia aumentando así sus conocimientos.

En un sistema experto los datos que se ingresan al generarse la consulta, así como los datos de los razonamientos intermedios están agrupados en la base de hechos, los algoritmos como son

conocidos en un sistema de programación estructurada no existen y en su lugar se utilizan sistemas de representación como los son las reglas de producción, los marcos, .Y sistemas de control mediante los cuales se realizan las búsquedas de soluciones, que son independientes de los conocimientos, y que se denominan motores de inferencia. Esta representación del conocimiento debe permitir, que la modificación de alguna unidad de conocimiento no afecte el resto del sistema.

Los conocimientos en este tipo de sistemas se representan simbólicamente, los programas de inferencia manipulan esta información almacenada en la base de conocimientos mediante un proceso de búsqueda y comparación de patrones. Reciben los datos iniciales mediante interfases que contienen y les permiten comunicarse con el usuario y este su vez con el sistema, utilizando esta información exploran la base de conocimientos en busca de la solución apropiada, el programa de inferencia realiza series de búsquedas que se encadenan simulando un proceso de razonamiento lógico.

Los programas de inferencia se implementan con algoritmos que definen las técnicas de búsqueda y comparación de patrones que se usarán en la base de conocimientos, son estas técnicas las que resuelven el problema no los algoritmos con los que son generados.

Otra característica muy importante de este tipo de sistemas es el funcionar con bases de conocimientos que nunca están completas, en las cuales el conocimiento va aumentando con el transcurso del tiempo (con el uso), debido a esto las funciones de control, los datos o hechos y el conocimiento son completamente independientes.

A lo largo del desarrollo del campo de la ingeniería en conocimientos, se han diseñado diferentes medios para representar el conocimiento que debe estar contenido en el. Estos han evolucionando desde sus inicios con el estudio de la lógica de predicados, los diferentes mecanismos de búsqueda, representaciones básicas como lo son las reglas de producción, la utilización de marcos que son sistemas de representación utilizados mayormente en la generación de sistemas expertos de una mayor dimensión.

Pero el desarrollo de este campo no se detiene en este tipo de representaciones, sigue avanzando en el intento de llegar a un tipo de razonamiento que se asemeje cada vez más al razonamiento humano generándose nuevos métodos de representación y búsqueda con el diseño de nuevos sistemas o mejorando los procesos ya existentes.

Se han dado también grandes aportaciones en el área de diseño y desarrollo de sistemas expertos pasando de lenguajes convencionales a lenguajes especialmente diseñados para el campo de la inteligencia artificial. Continuando con este tipo de aportaciones surgieron sistemas para la generación de sistemas expertos llamados shells, este tipo de herramientas ahorra un gran tiempo de programación permitiendo lograr un mejor diseño, además de ser herramientas amigables en su utilización. Existen diferentes tipos de shells tanto para el diseño y programación de pequeños sistemas expertos, como para sistemas de mayor tamaño.

Como ya se mencionó este es un campo en constante desarrollo debido a la importancia que ha tomando en muchos y muy variadas ramas de la actividad humana, integrándose cada día nuevas técnicas para mejora su diseño y funcionamiento, gran parte de la importancia de esta rama de la inteligencia artificial radica en el poder consultar conocimientos necesarios y especializados en un área teniendo la seguridad que información proporcionada es óptima para la solución de un problema. Los conocimientos expertos de un campo del conocimiento pueden ser proporcionados a un gran número de personas que no necesitan tener grandes conocimientos del tema o del área informática, para poder consultar esta información, tomando encuentra las facilidades que presentan para ser utilizados, la confiabilidad en sus resultados así como para explicar paso a paso el razonamiento utilizado para llegar a una conclusión especifica.

La ingeniería del conocimiento es un claro ejemplo de como el hombre evoluciona y va generando nuevos sistemas que le permitan facilitar diversas actividades, pero sobre todo los sistemas expertos al poder recopilar información sobre un campo específico y razonar con base en esta información, permiten acrecentar de una manera rápida información que tomaría a una persona años aprender y comprender. De está manera apoya tanto, la toma de decisiones como en la generación, acumulación y desarrollo de mayor conocimiento.

El desarrollo de la inteligencia artificial continua, con el surgimiento de nuevas técnicas o mejorando las ya existentes. Quizá el objetivo que determinó Alan Turing, en su famoso "Test de Turing", en el que propone la construcción de una máquina con inteligencia artificial capaz de alcanzar niveles de aprendizaje similares a las de un ser humano aun sea lejano y difícil de alcanzar.

Pero como se ha visto, no es necesario llegar a esta ambiciosa meta para alcanzar los enormes beneficios de los desarrollos y aplicaciones actuales en AI. La investigación y los modelos de AI aplicados a la solución de problemas prácticos ya están dando frutos, y encontramos diferentes aplicaciones de los sistemas basados en el conocimiento en diferentes áreas del conocimiento humano, como lo son: la medicina, finanzas, la industria del software, etc. Y con posibilidades razonables de ser utilizados y aplicados en otras tantas, en un futuro inmediato.

Aun cuando esta rama puede considerarse en sus inicios, ya que es relativamente nueva como área de desarrollo, la constante investigación en la misma ha permitido desarrollar sistemas más eficientes, pero sobre todo el hecho de que sea un área que continua evolucionando rápidamente deja abierta la posibilidad de algún día poder contar con sistemas capaces de hacer realidad el un objetivo ambicioso: el que sean capaces de pensar y generar conocimiento por si mismos.

# BIBLIOGRAFÍA


Benchimol, Guy
Levine, Pierre
Pomerol, Jean-Charles
Los sistemas expertos en la empresa
Macrobite, 1988
197 p.

Frenzel, Louis E. Jr.
A fondo los sistemas expertos
Editorial Anaya, 1989
213 p.

Giarratano, Joseph
Riley Gary
Expert Systems Principles and Programing
PWS-KENT Publishing Company, 1989.
632 p.

Lucas, Peter
Van Der Gaag, Linda
Principles of Expert System
Addison-Wesley, 1991.
598 p.

Nilsson, Nil J.
Principios de Inteligencia Artificial
Ediciones Díaz Santos, 1987
422 p.

Rauch, Wendy B.
Aplicaciones de la inteligencia artificial en la actividad empresarial, la ciencia y la industria. (fundamentos - aplicaciones)
Ediciones Díaz Santos, 1989

685 p.

Rich, E.
Inteligencia Artificial
Gustavo Gilli, 1983
446 p.

Roltson, David W.
Principles of Artificial Intelligent and Expert System Development
McGraw-Hill, 1990
261 p.

Sánchez y Beltrán, J.B.
Sistemas expertos una metodología de programación
Macrobit, 1990
261 p.

Simons, G.L.
Introducción a la inteligencia artificial
Ediciones Díaz Santos, Madrid 1987.
267 p.