24021
47

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

"A C A T L A N"

INTELIGENCIA ARTIFICIAL Y SISTEMAS EXPERTOS

T E S I N A

QUE PARA OBTENER EL TITULO DE

LICENCIADO EN MATEMATICAS APLICADAS Y COMPUTACION

P R E S E N T A :

MIREYA SANCHEZ TOVAR

ASESOR: GUSTAVO GUDIÑO RAMIREZ

MEXICO, D. F.

ABRIL 2003

A

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

ÍNDICE

CAPÍTULO IV. REPRESENTACIÓN DE PROBLEMAS DE INTELIGENCIA ARTIFICIAL

CAPÍTULO V. USO Y MODIFICACIÓN DEL CONOCIMIENTO

CAPÍTULO VI. APLICACIONES ESPECÍFICAS

JUSTIFICACIÓN Y OBJETIVO

Desarrollar el temario de Inteligencia Artificial de la Licenciatura de Matemáticas Aplicadas y Computación facilitará la búsqueda de información a los alumnos interesados en las áreas de Inteligencia Artificial y Sistemas Expertos, de una forma comprensible y fácil de entender incluso a personas que aún no se encuentren familiarizadas con el tema.

INTRODUCCIÓN

"Hacer que una máquina piense" ha sido el sueño histórico del hombre que hasta ayer parecía imposible.

Hoy, debido al avance excepcional de la tecnología informática y computacional durante los últimos 50 años, cada día deja de parecer sólo un tema sacado de una película de ficción para acercarse más a nuestra realidad presente y futura: simular el comportamiento del cerebro humano por medio de una computadora; en donde las entradas del interlocutor se realizan a través de un teclado o un dispositivo similar y las salidas las escribe la propia computadora en el monitor, simulando todo ello a los órganos sensoriales del oído y el habla, de tal forma que podemos dialogar con la computadora como si lo estuviéramos haciendo con una persona.

El objetivo de este trabajo, es presentar de una manera no exhaustiva algunos de los temas relacionados a la Inteligencia Artificial y sobre todo, a una de sus grandes aplicaciones: los Sistemas Expertos.

Pero, ¿qué es la Inteligencia Artificial? A pesar de no existir una definición exacta acerca de IA, dentro del Capítulo I *(Introducción a la Inteligencia Artificial)* se otorgarán algunas definiciones de Inteligencia Artificial según los diferentes "padres" de dicha ciencia, además de que se hablará de su campo de acción y sobre la construcción de sistemas basados en IA, así como de las herramientas necesarias para realizar dicha construcción.

Una de las herramientas principales para la construcción de sistemas de IA son los lenguajes. En el Capítulo II *(Lenguajes de Programación)* se hablará de los distintos tipos de lenguajes que existen en la actualidad, sus principales características, sus ventajas y sus desventajas; y también de dos de los principales lenguajes que se utilizan para la creación de sistemas de IA: LISP y PROLOG; sus características más importantes y pequeños programas para ejemplificar el modo de funcionamiento del lenguaje.

Una de las áreas de IA que más auge ha tenido, es el área de los Sistemas Expertos, por lo que el Capítulo III *(Sistemas Expertos)* estará dedicado a ellos. Se dará una definición de SE, y se mencionarán sus principales características, objetivos y ventajas; también se verá la estructura general de un SE, mostrando los aspectos más importantes de cada una de sus partes; además de que se verá la comparación entre

un SE y un programa tradicional. Finalmente se dará a conocer el ciclo de vida de un SE y los pasos necesarios para la construcción del mismo.

Como se verá, para la construcción de cualquier SE es fundamental adquirir el conocimiento necesario que se representará en dicho sistema, por lo cual, el Capítulo IV *(Representación de problemas de Inteligencia Artificial)* tratará primeramente el tema de la ingeniería del conocimiento y de la adquisición de éste por parte del ingeniero del conocimiento; después se hablará de cómo los humanos procesamos la información que nos rodea, y de los distintos métodos que existen para resolver y representar los problemas de IA.

El Capítulo V *(Uso y modificación de conocimiento)* hablará de las características generales de las técnicas de aprendizaje y la resolución general de problemas, principalmente mediante técnicas de inferencia deductivas, haciendo énfasis en dos de las técnicas principales: encadenamiento hacia delante y encadenamiento hacia atrás.

Finalmente, dentro del Capítulo VI *(Aplicaciones específicas)* se presentarán los diferentes tipos de aplicaciones de los SE que son utilizados en la actualidad, así como sus características, además de que se mostrarán las aplicaciones y desarrollos de dichos sistemas en México.

CAPÍTULO I.
INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL (IA).

Definición de Inteligencia Artificial

La Inteligencia Artificial es la parte de las Ciencias de la Computación que se ocupa del diseño de sistemas inteligentes, esto es, sistemas que exhiben características que asociamos con la inteligencia en las conductas humanas.
Feigenbaum y Barr.

El estudio de cómo lograr que las computadoras realicen tareas que por el momento, los humanos hacen mejor.
E. Rich - Knight, 1991.

La rama de la Ciencias de la computación que se ocupa de la automatización de la conducta inteligente.
Luger y Stubblefield, 1993.

Es la Ciencia e Ingeniería de hacer máquinas inteligentes (especialmente programas). Esto está relacionado a la tarea de usar computadoras para entender la inteligencia humana, pero IA no tiene que limitarse a métodos que son biológicamente observables.
J. Mc Carthy, 1998.

La IA comprende el estudio y creación de sistemas computarizados que manifiestan cierta forma de inteligencia: sistemas que aprenden nuevos conceptos y tareas, que pueden razonar y derivar conclusiones útiles acerca del mundo que nos rodea, sistemas que pueden comprender un lenguaje natural o percibir y entender una escena visual, y sistemas que realizan otro tipo de actividades que requieren de inteligencia humana.

La IA es una ciencia que trata de la comprensión de la inteligencia y del diseño de máquinas inteligentes, es decir, el estudio y la simulación de las actividades intelectuales del hombre (manipulación, razonamiento, percepción, aprendizaje, creación).
P. H. Winston.

La IA es el estudio de las computaciones que permiten percibir, razonar y actuar. Es un campo de estudio que busca explicar y emular el comportamiento inteligente en términos de procesos computacionales.
R. J. Schalkoff.

Estudia las representaciones y procedimientos que automáticamente resuelven problemas usualmente resueltos por humanos.
R. E. Bellman.

A pesar de la diversidad de conceptos propuestos para la IA, en general todos coinciden en que la IA trata de alcanzar inteligencia a través de la computación. Todo programa de computación, requiere de una representación de cierta entidad y de un proceso para su manipulación.

Desde el punto de vista de los objetivos, la IA puede considerarse en parte como ingeniería y en parte como ciencia:

- Como ingeniería, el objetivo de la IA es resolver problemas reales, actuando como un conjunto de ideas acerca de cómo representar y utilizar el conocimiento, y de cómo desarrollar sistemas informáticos.

- Como ciencia, el objetivo de la IA es buscar la explicación de diversas clases de inteligencia, a través de la representación del conocimiento y de la aplicación que se da a éste en los sistemas informáticos desarrollados.

Campo de acción de la IA

La IA engloba diferentes subáreas. Las principales subáreas existentes son, entre otras, el reconocimiento de voz o de patrones, la demostración automática de teoremas, el procesamiento del lenguaje natural, la visión artificial, y los sistemas expertos.

En capítulos posteriores se tratarán cada uno de estos temas, haciendo énfasis en el que puede tener mayor número de aplicaciones prácticas: el de los Sistemas Expertos, siendo dichas aplicaciones de utilidad en temas tan variados que pueden ir desde la medicina hasta la enseñanza.

Además, un campo particular de la Inteligencia Artificial es la Representación del Conocimiento, cuyos objetivos son el desarrollo de modelos adecuados para la adquisición y manipulación del conocimiento.

Construcción

Para construir un sistema de IA, es necesario:
1- Definir el problema formalmente con precisión.
2- Analizar el problema.
3- Representar el conocimiento necesario para resolver el problema.

4- Elegir la mejor técnica que resuelva el problema y aplicarla.

Definición formal del problema. Hay problemas que por ser artificiales y estructurados son fáciles de especificar (por ejemplo el ajedrez). Otros problemas naturales, como por ejemplo la comprensión del lenguaje, no son tan sencillos de especificar.
Para producir una especificación formal de un problema se deben definir:

- Espacio de estados válidos.
- Estado inicial del problema.
- Estado objetivo o final.
- Reglas que se pueden aplicar para pasar de un estado a otro.

Todos estos temas serán definidos en el Capítulo IV (Representación de problemas de IA) en el apartado de Métodos de IA para resolver problemas.

Análisis del problema. Luego de definir el problema formalmente, el segundo paso en la resolución es el análisis del mismo. A fin de poder elegir el método más apropiado para resolver un problema particular, es necesario analizar distintas cuestiones que afectan a la definición del mismo y a las características de la solución deseada. Existen varias preguntas a responder:
1. ¿Puede descomponerse el problema en subproblemas más pequeños?
2. ¿Pueden deshacerse pasos inadecuados hacia la solución?
3. ¿Es predecible el universo del problema?
4. ¿Una solución es buena de manera absoluta o relativa?
5. ¿La solución deseada es un estado o la ruta hacia un estado?
6. ¿El conocimiento se necesita para resolver el problema o para restringir la búsqueda de la solución?
7. El programa que soluciona el problema ¿busca sólo la solución o necesita interactuar con una persona?

1. ¿Puede descomponerse el problema en subproblemas más pequeños?
Algunos problemas pueden descomponerse en subproblemas independientes, de manera que encontrar una solución global es la composición de soluciones particulares. Por ejemplo en la resolución de integrales, una integral puede descomponerse por partes, y resolver las partes simples directamente o descomponerlas recursivamente.
Por otra parte, existen otros problemas que no pueden descomponerse y componer la solución a partir de las soluciones parciales de sus partes. Por el contrario, una solución necesita considerar globalmente el problema.

2. ¿Pueden deshacerse pasos inadecuados hacia la solución?

Algunos problemas permiten deshacer uno o varios pasos hacia una solución una vez realizados. En este aspecto, existen tres categorías en las que puede dividirse un problema:

- Recuperables: en un punto dado es posible deshacer todos los pasos inadecuados hacia la solución.
- No recuperables: en un punto dado no es posible deshacer ningún paso realizado. Por ejemplo en una partida de ajedrez no se puede volver atrás una vez movidas las piezas. En estos problemas el sistema debe esforzarse en la toma de decisiones pues éstas son irrevocables. Algunos usan una planificación en la que se analiza por adelantado una secuencia de pasos antes de realizar el primer paso para descubrir a donde conduce.
- Ignorables: en un punto dado es posible ignorar los pasos realizados hasta el momento y comenzar de nuevo con una nueva solución. Por ejemplo un demostrador de teoremas puede abandonar una demostración basada en un lema dado y comenzar nuevamente. Estos problemas se resuelven con estrategias de control sencillas que nunca vuelven hacia atrás.

3. ¿Es predecible el universo del problema?

Los problemas pueden ser de:

- Consecuencia cierta: es posible planificar una secuencia de movimientos estando seguros del resultado a obtener. Se puede realizar una planificación para generar operadores que garanticen llegar a la solución.
- Consecuencia incierta: no es posible planificar con certeza pues no se sabe que ocurrirá luego del siguiente movimiento. Sin embargo, se puede realizar una planificación para generar operadores que tengan una buena probabilidad de llegar a la solución.

Los problemas más difíciles de resolver son los no recuperables de consecuencia incierta.

4. ¿Una solución es buena de manera absoluta o relativa?

La solución de un problema puede consistir en encontrar:

- Algún camino: sólo importa encontrar una solución sin importar si existen otros caminos que conducen a la solución. Generalmente se resuelven con heurísticas. Por ejemplo programa de respuestas a preguntas.
- El mejor camino: importa encontrar la ruta más corta hacia la solución. Son problemas más complicados de computar. Algunos requieren una búsqueda más exhaustiva que usando heurísticas.

5. ¿La solución deseada es un estado o la ruta hacia un estado?

La solución de un problema puede consistir en encontrar:

- Un estado final: no es necesario el registro del proceso, sólo importa arribar a la solución final. Por ejemplo interpretar texto.

- Una ruta hacia un estado final: se necesita dar el camino seguido desde el estado inicial al estado final.

6. ¿El conocimiento se necesita para resolver el problema o para restringir la búsqueda de la solución?

El conocimiento puede emplearse para:

- Reconocer la solución: se necesita gran cantidad de conocimiento acerca del problema para poder encontrar una solución. Por ejemplo comprensión de texto.
- Acotar la búsqueda: la solución básica puede encontrarse con poco conocimiento, pero para restringir el árbol de búsqueda y encontrar la solución de manera más eficiente es necesario contar con más conocimiento. Por ejemplo en el ajedrez se necesita básicamente poco conocimiento para conocer los movimientos legales y un mecanismo sencillo de búsqueda. Pero dado que para aumentar la eficiencia de la búsqueda ésta debe restringirse, se necesita conocimiento de heurísticas de buenas estrategias y tácticas para jugar.

7. El programa que soluciona el problema ¿busca sólo la solución o necesita interactuar con una persona?

Con respecto a la relación programa-usuario, existen dos tipos de programas que solucionan el problema:

- Solitarios: reciben como entrada el problema y dan como salida la solución. No importa el razonamiento que haya seguido la máquina para encontrar la solución.
- Conversacionales: existe una comunicación hombre-máquina de manera que el usuario puede ayudar a la máquina o la máquina puede informar al usuario durante la búsqueda de la solución. Para que esta comunicación sea posible debe existir una correspondencia entre el razonamiento seguido por la máquina y la forma de razonamiento humano. Por ejemplo en un sistema experto de diagnóstico médico, el usuario no aceptaría el veredicto de una máquina si no puede comprender el razonamiento que la llevó a él.

Herramientas de la IA

Una decisión fundamental al definir un problema es decidir qué tan bien modelarlo. Muchas veces se dispone de la experiencia que ayuda a escoger el mejor paradigma. Por ejemplo, la experiencia sugiere que una nómina se elabora mejor con un procedimiento de programación convencional; también sugiere que es preferible usar un paquete comercial, si está disponible, en lugar de escribir uno desde cero. Una guía general para seleccionar un paradigma es considerar primero el más tradicional: la programación convencional. La razón para esto es la vasta experiencia que tenemos con la programación convencional y la amplia variedad de paquetes comerciales

disponibles. Si un problema no puede solucionarse eficazmente con la programación convencional, entonces hay que mirar hacia los paradigmas no convencionales, como la IA.

Al escoger un lenguaje, una pregunta básica debe ser si el problema exige más conocimiento o inteligencia. Los sistemas expertos dependen de una gran cantidad de conocimiento especializado o experiencia para resolver un problema, mientras que la IA enfatiza un método para la solución de problema. Es común que los sistemas expertos dependan de la correspondencia de patrones en un dominio de conocimiento restringido para guiar su ejecución, mientras que la IA suele concentrarse en la búsqueda de paradigmas en dominios menos restringidos.
Entre las principales herramientas que existen para la construcción de programas de IA tenemos: Lenguajes, Herramientas y Shells.

Lenguaje

Es un traductor de comandos escrito con una sintaxis específica. Un lenguaje para sistemas expertos, también proporciona un mecanismo de inferencia que ejecuta las instrucciones del lenguaje. Dependiendo de la forma en que esté implantado, el mecanismo de inferencia puede proporcionar encadenamiento hacia atrás, adelante o ambos. Las preguntas acerca del tiempo de desarrollo, la conveniencia, la conservación, la eficiencia y la velocidad, determinan en qué lenguaje debe estar escrito el software.

Herramienta

Es un lenguaje adicionalmente asociado con programas de utilerías para facilitar el desarrollo, la depuración y el uso de los programas de aplicación. Los programas de utilerías pueden incluir editores de texto e imágenes, depuradores, administradores de archivos e incluso generadores de código. Algunas herramientas pueden incluso admitir el uso de paradigmas diferentes, como el encadenamiento hacia adelante y hacia atrás en una aplicación.

En algunos casos, una herramienta puede integrarse con todos sus programas utilitarios en un solo ambiente, para presentar al usuario una interfaz común. Este método minimiza la necesidad de que el usuario abandone el ambiente para ejecutar una tarea. Por ejemplo, una herramienta simple tal vez no proporcione medios para la administración de archivos y, por tanto el usuario tendrá que salir de la herramienta para aplicar comandos del sistema operativo. Un ambiente integrado permite un fácil intercambio de datos entre varios programas utilitarios dentro del mismo ambiente.

Algunas herramientas ni siquiera requieren que el usuario escriba algún código; en cambio, la herramienta permite al usuario introducir conocimiento mediante el ejemplo, a partir de tablas u hojas de cálculo, y genera el código apropiado por sí misma.

Shell

Herramienta con propósitos especiales, diseñada para cierto tipo de aplicaciones en las que el usuario sólo debe proporcionar la base de conocimientos. El ejemplo clásico de esto es el shell de EMYCIN (MYCIN vacío), éste se hizo al eliminar la base de conocimientos médico del sistema experto MYCIN (MYCIN fue uno de los primeros sistemas expertos desarrollados, y se verá con mayor detalle en el Capítulo VI Aplicaciones Específicas, dentro del apartado de Medicina.

MYCIN estaba diseñado como un sistema de encadenamiento hacia atrás para diagnosticar enfermedades. EMYCIN se creó por la simple eliminación del conocimiento médico y pudo usarse como un shell que contenía conocimiento acerca de otros tipos de sistemas consultivos que utilizan él encadenamiento hacia atrás. El shell EMYCIN demostró que podía volver a utilizarse el software esencial de MICYN como mecanismo de inferencia y la interfaz del usuario. Este fue un paso muy importante en el desarrollo de la tecnología para los modernos sistemas expertos, porque significó que no era necesario construir desde cero cada nueva aplicación.

CAPÍTULO II.
LENGUAJES DE PROGRAMACIÓN.

Tipos de lenguajes

En la actualidad existe una gran cantidad de lenguajes y herramientas para programar computadoras. En muchos de ellos se ha trabajado para la realización de proyectos de IA como lo son LISP y PROLOG.

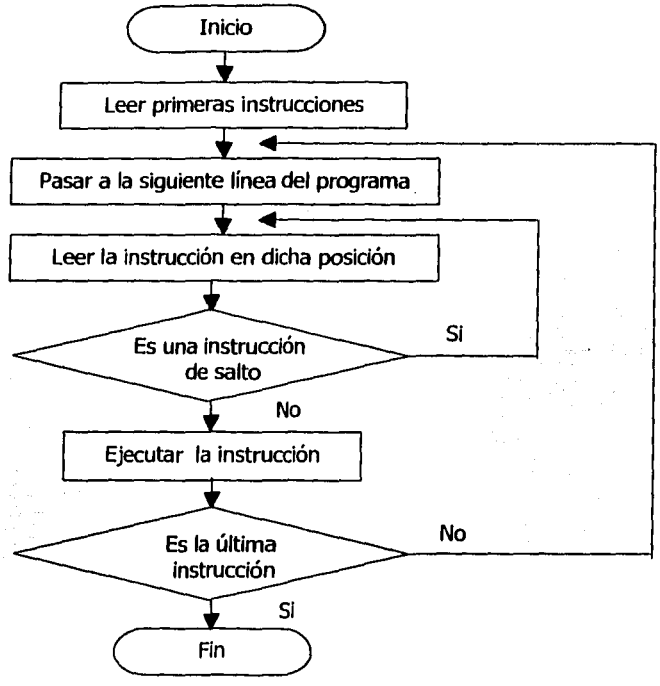
Las tareas de programación que hay que realizar en las aplicaciones de la IA son las siguientes:

1. Sistemas de tratamiento simbólico.
2. Sistemas de representación del conocimiento.
3. Estructuras de control o motor de inferencia.
4. Sistemas de entrada/salida.
   - Diálogo.
   - Justificación.
   - Mantenimiento.
   - Explicación.
   - Presentación.

Los diferentes lenguajes se pueden clasificar de acuerdo a la información que manipulan, principalmente se dividen en: Imperativos, Funcionales, Orientados a objetos y Declarativos.

Imperativos

Son aquellos en los que el control del programa pasa a la siguiente línea del programa salvo que se le ordene lo contrario. El programador debe indicar en el programa el flujo de ejecución de las instrucciones y el funcionamiento de la computadora es claro para el programador.



Ventajas desde el punto de vista de IA:

TESIS CON
FALLA DE ORIGEN

- Flexibilidad total.
- Conocimiento muy generalizado de los lenguajes.
- Eficacia máxima.

Desventajas:

- Desarrollos muy largos y costosos.
- Aplicaciones desarrolladas poco utilizables en otras aplicaciones.
- Programas difíciles de leer o modificar.

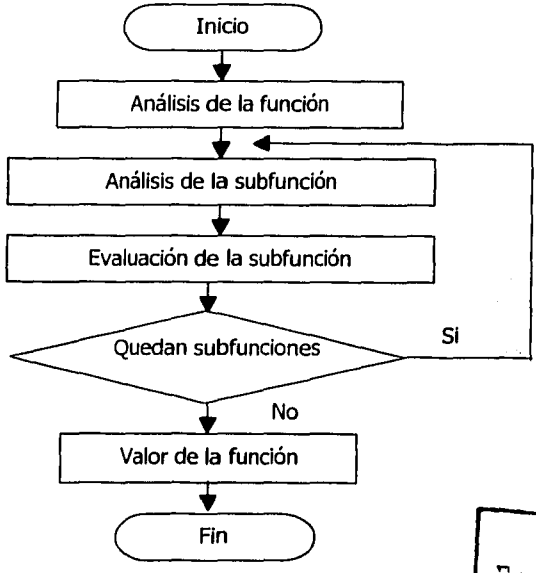
Ejemplos: Pascal, C, Basic.

Funcionales

Son aquellos en los que el flujo del programa viene marcado por las necesidades que aparecen al evaluar una función. El programador solamente tiene que indicar el orden

de evaluación de las funciones, pero no es necesario que se preocupe por indicar su posición física dentro del programa.
En los lenguajes funcionales, el control de tipo secuencial ha sido sustituido por cuatro pasos de evaluación similares a los que se utilizan al evaluar una función matemática. Dicho proceso consiste en:

1. Análisis de la función a evaluar.
2. Búsqueda de la primera subfunción.
3. Evaluación de la primera subfunción.
4. Retorno a la función anterior o valor final si se encuentra en la primera función.



TESIS CON
FALLA DE ORIGEN

Ventajas:

- Poseen sistemas de tratamiento simbólico.
- Sencillez en la construcción de motores de inferencia.
- Sencillo de aprender y utilizar.

Desventajas:

- Es caro por los requerimientos mínimos que necesita para funcionar.
- Poco eficaz sobre arquitecturas tradicionales.
- Entornos cerrados.
- Pueden crecer con facilidad por lo que hay muchas versiones.

Ejemplos: LISP y LOGO.

Orientados a Objetos

Se caracterizan porque no existe distinción entre los procedimientos y los datos. Los programas están formados por los objetos y cada objeto interpreta el mensaje que le llega. Un objeto es la particularización de una clase, heredando en este proceso las propiedades de la clase, molde o prototipo que lo ha originado. Debido a este proceso generativo de los objetos, estos están estructurados jerárquicamente en clases y subclases. Los objetos tienen una serie de propiedades que les caracterizan y se llaman atributos.

Ventajas:

- El sistema de control es fácilmente convertible en un motor de inferencia.
- La tarea de construir una aplicación se reduce notablemente si se hace uso de un lenguaje orientado a objetos.
- Permiten que un sistema evolucione y que puedan introducirse cualquier tipo de cambios mientras el sistema funciona.

Desventajas:

- Poco eficaz ya que por ejemplo, las variables que no están declaradas indican que en todo momento se debe de comprobar el tipo de variables que intervienen en el proceso.
- Filosofía de programación completamente nueva para IA.
- Es fundamental la experiencia y la capacitación para la producción de este tipo de sistemas por transferencia de experticia. No es posible adquirir experticia en una nueva tecnología de manera individual. Esto hace más difícil la adopción de este tipo de lenguajes.
- Dependencia del lenguaje. A pesar de la portabilidad conceptual de los objetos en un sistema orientado a objetos, en la práctica existen muchas dependencias.
- Determinación de las clases. Una clase es un molde que se utiliza para crear nuevos objetos. En consecuencia es importante crear el conjunto de clases adecuado para un proyecto. Desafortunadamente la definición de las clases es más un arte que una ciencia.
- Performance. En un sistema donde todo es un objeto y toda interacción es a través de mensajes, el tráfico de mensajes afecta el rendimiento. A medida que la tecnología avanza y la velocidad de microprocesamiento, potencia y tamaño de la memoria aumentan, la situación mejorará; pero en la situación actual, un diseño de una aplicación orientada a objetos que no tiene en cuenta el rendimiento no será viable comercialmente.

Ejemplo: Java

Declarativos

Son aquellos en los que solamente hay que indicarle al programa el objetivo que se desea demostrar, especificando en el programa el universo sobre el cual demostrarlo y las reglas que se pueden utilizar, es decir, se especifica el "qué" pero no el "cómo".
Para que el lenguaje sea capaz de afrontar este problema, es imprescindible que incorpore un motor de inferencia que controle el proceso de demostración.
Son lenguajes muy compactos al no necesitar alguna estructura de control.

Los lenguajes declarativos cuentan con:

1. Un sistema de representación de conocimiento en forma de reglas de producción.
   Ejemplo: Si (antecedentes) Entonces (consecuentes).
2. Un motor de inferencia que es el propio intérprete o compilador, basado en la unificación, la búsqueda en profundidad y la marcha atrás.
3. Un sistema sencillo de diálogo.

Características:

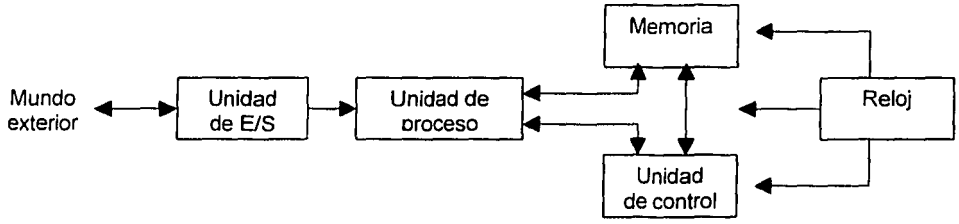
- Los lenguajes declarativos no se basan en la máquina Von Newman [1] sino en modelos matemáticos.

1 En 1940 Von Newman configuró la arquitectura básica de las computadoras modernos, basada en los siguientes conceptos:

- programa almacenado

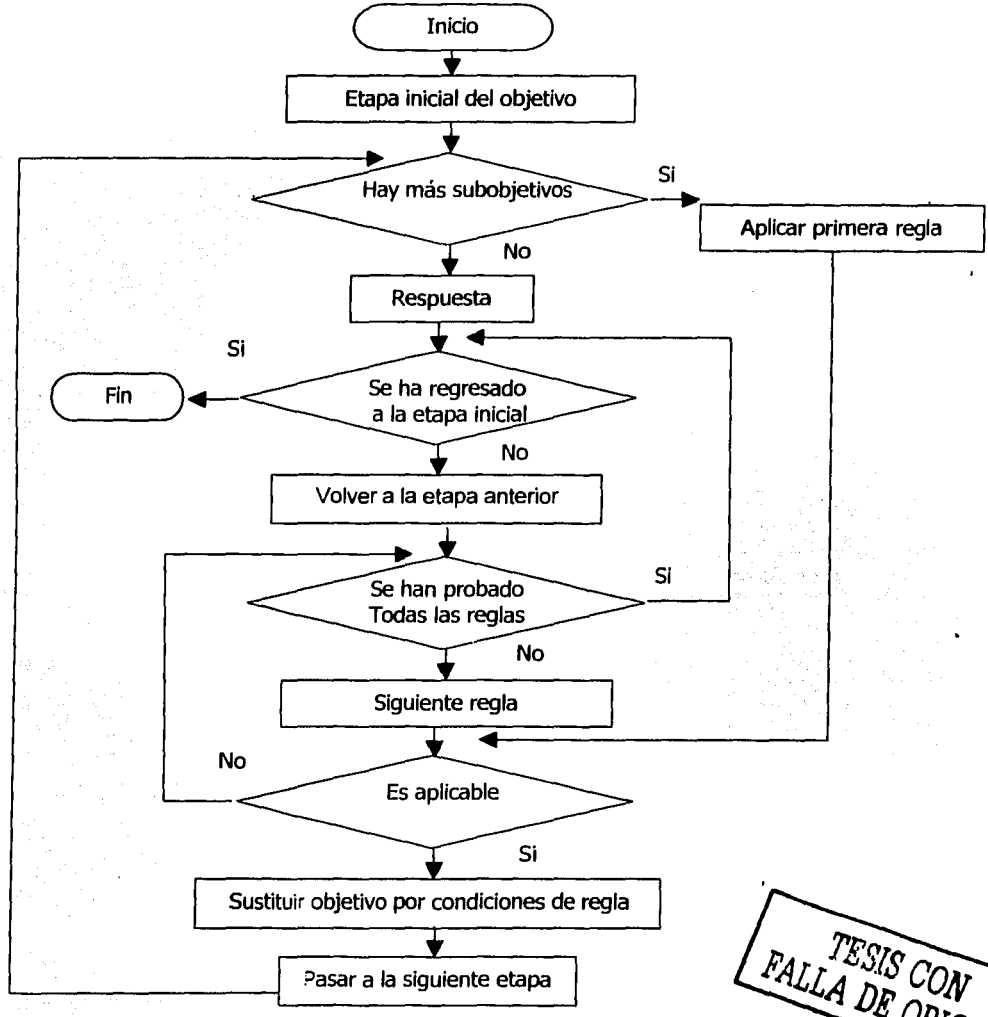
- ruptura de la secuencia de programa mediante la toma de decisiones.

Su diagrama de bloques es el siguiente:



TESIS CON
FALLA DE ORIGEN

- Los lenguajes declarativos, en contra de lo que hacen los imperativos, intentan ser referencialmente transparentes, es decir, la misma expresión siempre da los mismos resultados.
- En un lenguaje declarativo el control lo lleva la máquina, sin embargo en un lenguaje imperativo el control depende exclusivamente del programador.



TESIS CON FALLA DE ORIGEN

- Los lenguajes declarativos son independientes de la máquina, sin embargo los imperativos se basan en el lenguaje máquina en el que se apoyan, teniendo como instrucción principal la asignación.
- El concepto de variable en un lenguaje imperativo es un objeto cuyo valor puede cambiar en el tiempo, en cambio, en los lenguajes declarativos las variables son objetos cuyo valor no se conoce, y que, una vez que se le asocia un valor, conserva dicho valor hasta el final.

Ventajas::

- El motor de inferencia es fijo.
- El sistema de representación del conocimiento es único.
- La principal ventaja de los lenguajes declarativos es que son independientes de la máquina y, referencialmente transparentes.
- La cantidad de código que debemos escribir es menor, aunque la representación formal de estos problemas puede resultar, en ocasiones, poco evidente.
- Podemos desentendernos del control. Aunque aquí existe una limitación: no podemos hacerlo totalmente. Por ello, PROLOG nos proporciona formas, un tanto artificiales, de manejo de este control.
- Los datos pueden ser tanto de entrada como de salida y se pueden utilizar datos parcialmente construidos, es decir, podemos empezar a ejecutar sin contar con todos los datos.

Desventajas:

- Las operaciones de entrada/salida en estos lenguajes son algo complejas de realizar pero son posibles.
- Poco eficaz sobre arquitecturas tradicionales.
- Es poco fiable al no aparecer de manera explícita el comportamiento del programa pues el motor de inferencia es implícito al lenguaje.
- Es difícil representar la negación.

Ejemplos: PROLOG, OP5, Smalltaltk.

Programas LISP

El nombre LISP es la abreviatura de **List-P**rocessing (Procesamiento de Listas), ya que el LISP fue desarrollado para el procesamiento de listas. La lista es la estructura más importante de LISP. El lenguaje LISP fue diseñado ya a finales de los años 50 por McCarthy. A lo largo de los últimos años se han desarrollado muchos dialectos, por

ejemplo MACLISP, COMMONLISP, INTERLISP, ZETALISP, donde el COMMONLISP se está imponiendo cada vez más como estándar.

De acuerdo a los tipos de lenguajes, LISP es un lenguaje funcional, y como tal, todas sus construcciones son funciones en el sentido matemático del término. No hay instrucciones, ni siquiera una instrucción de asignación. Un programa funcional es una función que se define por composición de funciones más simples, para ejecutarlo, se aplica a los datos de entrada (parámetros de la función) y se obtiene un resultado (valor calculado de la función).

En LISP se dan los siguientes conceptos característicos:

Átomos: Es un dato elemental y es la unidad más pequeña del lenguaje. Los átomos pueden subordinarse a cualidades.

Listas: La estructura más importante es la lista. Es un conjunto de átomos o listas separadas por comas o espacios. Al primer elemento se le llama "nombre de la función", mientras que al resto se le llama "parámetros" o "atributos".

Funciones: Cada función LISP y cada programa LISP tiene estructura de lista. Los programas no pueden distinguirse sintácticamente de los datos. LISP ofrece sus propias funciones básicas.

Llamada a una función: (Nombre_funcion param1 param2)

Forma de Trabajo: LISP es un lenguaje funcional. Ofrece la posibilidad de realizar definiciones recursivas de funciones. La unión de procedimientos se realiza de forma dinámica, es decir en plena ejecución, y no como en otros lenguajes de programación. El sistema realiza automáticamente una gestión dinámica de memoria.

Ejemplo :

(A (B C) D) es una lista con tres elementos
A átomo
(B C) lista de átomos B y C
D átomo

También está permitida una lista vacía, "( )" ó " NIL" , que significa lo mismo.
Con esta estructura podemos configurar estructuras de cualquier complejidad, tan grandes como queramos.
Los átomos son números, cadenas de caracteres o símbolos. Un símbolo puede tener varios valores, al igual que una variable en otros lenguajes de programación, como por ejemplo un número, o también puede ser el nombre de una función, o incluso ambos.

Además a un símbolo pueden subordinarse cualidades, que además del valor del símbolo, contienen información adicional. Estas cualidades también reciben el nombre de atributos.

Componentes de un sistema LISP.

Un componente importante de un sistema LISP es la gestión dinámica de la memoria. El sistema administrará el espacio en la memoria para las listas en constante modificación, sin que el usuario lo deba solicitar. Libera los espacios de memoria que ya no son necesarios y los pone a disposición de usos posteriores. La necesidad de este proceso se deriva de la estructura básica de LISP, las listas, que se modifican de forma dinámica e ilimitada.

Existen varias razones por las que LISP es un buen lenguaje para sistemas de IA:

- Su principal estructura de datos es la lista, que es muy útil para representar la mayor parte del conocimiento que se una en los programas de IA.
- Puede representarse fácilmente una colección de hechos sobre un objeto individual en la lista de propiedades que se asocia con un átomo que representa el concepto. La lista de propiedades es una lista de pares atributo-valor.
- La estructura de control más natural es la recursión, que es apropiada para muchas tareas de resolución de problemas.
- La mayoría de las cosas no se ligan o fijan hasta el último momento posible. Por ejemplo, las listas no necesitan ser de un tamaño fijo.

Ejemplos de instrucciones en LISP:

' (quote)	Toma la lista como átomo y no como función
car	Da como resultado de su evaluación el primer elemento de una lista (car ' (uno dos tres)) Resultado: uno
cdr	Da como resultado de su evaluación el resto de la lista sin el primer elemento (cdr ' (uno dos tres)) Resultado: dos tres
cons	Añade un elemento al principio de una lista (cons ' (cero) ' (uno dos tres)) Resultado: cero uno dos tres
append	Une dos listas, una a continuación de la otra formando una sola (append ' (cero uno dos tres) ' (cuatro cinco)) Resultado: cero uno dos tres cuatro cinco
setq	Asigna valores a una variable (setq x 25) Resultado: x = 25

defun Define una función
(defun nombre_funcion (parámetro) (operaciones))

Ejemplos de funciones LISP

(+ 5 6) → 11
(+ 1 2 2) → 5

(defun cuadrado (X) (* X X)) (defun cubo (X) (* X X X))

(cuadrado 5) → 25 (cubo 5) → 125

Programas PROLOG

PROLOG es la abreviatura de **PRO**gramming **LOG**ic (programación Lógica), con lo que hacemos mención a la procedencia del lenguaje: Es una realización de lógica de predicados, como lenguaje de programación.
En la actualidad, el PROLOG se aplica como lenguaje de desarrollo en aplicaciones de Inteligencia Artificial en diferentes proyectos de Europa. En los Estados Unidos, el LISP está más extendido que el PROLOG. Pero para la mayoría de los terminales de trabajo de Inteligencia Artificial se ofrece también el PROLOG.

Definición de Hechos, Metas y Predicados en PROLOG

Para construir programas en PROLOG necesitamos convertir los conceptos expresados en lenguaje natural en un lenguaje basado en la lógica de primer orden, con el fin de obtener las cláusulas de Horn [2] tras el proceso de conversión adecuado, ya que PROLOG trabaja, precisamente, con este tipo de cláusulas. Básicamente, nuestro trabajo va a consistir en especificar adecuadamente los enunciados y reglas básicas para resolver un determinado problema de forma general. Después le plantearemos a PROLOG el conjunto de objetivos (problemas específicos para un problema general dado), que queremos que resuelva.

Los programas estarán formados por:
- La base de conocimientos: Hechos + Reglas de Inferencia.
- El conjunto de objetivos o metas.

2 La programación lógica está basada en un subconjunto del cálculo de predicados, incluyendo instrucciones escritas en formas conocidas como cláusulas de Horn. Este paradigma puede deducir

nuevos hechos a partir de otros hechos conocidos. Un sistema de cláusulas de Horn permite un método particularmente mecánico de demostración llamado resolución.

Un predicado especifica la relación existente entre los argumentos del mismo. El número de argumentos a los que se aplica dicho predicado se denomina paridad. Con un predicado podemos representar algo que sucede en el mundo real (hecho), o una regla (regla de inferencia), que nos permite deducir hechos que suceden en ese dominio mediante la aplicación de la misma.
La sintaxis de PROLOG para especificar un predicado es la siguiente:

nombre_predicado(arg1, arg2, etc. , argN).
(En el caso de que el predicado represente un hecho).

nombre_p1[([arg1], [arg2],etc., [argN])]:-
otro_p1[([arg1], [arg2],etc., [argN])],
etc.,
otro_pN[([arg1], [arg2], etc. , [argN])].
(En el caso de que el predicado represente una regla de inferencia).

Donde: :- Indica que el predicado consta de uno o más elementos
, Indica que continua otro elemento
. Indica el fin del predicado

Ejemplos:

```
abuela(X,Y):-
madre(X,Z),
madre(Z,Y).
```

```
factorial(N, R):-
N1 is N-1,
factorial(N1, Y),
R is N*Y.
```

Las reglas de inferencia permiten llevar a cabo la deducción de metas. Para que las reglas de inferencia sean aplicables, en general, deberá existir un conjunto de hechos sobre los que apoyarse para que las demostraciones se puedan realizar. Las metas representan los problemas específicos, basados en los problemas generales expresados en la base de conocimientos que deseamos que el demostrador automático de teoremas resuelva. El demostrador resuelve las metas comenzando por los predicados situados en la zona superior (de arriba a abajo), y para cada predicado el proceso de unificación se lleva a cabo de izquierda a derecha.

Mecanismos principales de PROLOG

Los mecanismos importantes del PROLOG son: recursividad, instanciación, verificación, unificación, y backtracking .

La Recursividad representa la estructura más importante en el desarrollo del programa. En la sintaxis del PROLOG no existen los bucles FOR ni los saltos; los bucles WHILE son de difícil incorporación, ya que las variables sólo pueden unificarse una sola vez. La recursión es más apropiada que otras estructuras de desarrollo para procesar estructuras de datos recursivas como son las listas y destacan en estos casos por una representación más sencilla y de mayor claridad.

La Instanciación es la unión de una variable a una constante o estructura. La variable ligada se comporta luego como una constante. Una variable está instanciada cuando existe un objeto determinado representado por la variable, en caso contrario, o cuando no se sabe lo que la variable representa, es no instanciada o libre.

La Verificación es el intento de derivar la estructura a comprobar de una pregunta desde la base de conocimientos, es decir, desde los hechos y reglas. Si es posible, la estructura es verdadera, en caso contrario es falsa.

La Unificación es el componente principal de la verificación de estructuras. Una estructura estará comprobada cuando puede ser unificada con un hecho, o cuando puede unificarse con la cabecera de una regla y las estructuras del cuerpo de dicha regla pueden ser verificadas. La unificación se realiza, para cada predicado, de izquierda a derecha, y para cada conjunto de predicados, de arriba a abajo. Se pueden unificar variables con constantes, siempre que la variable no esté instanciada. Si la variable está instanciada, el hecho de hacer unificación entre ambas se corresponde con la situación de unificación de dos constantes. Para que dos constantes se puedan unificar ambas han de ser iguales.

Ejemplos de unificación:

Variable no instanciada se intenta unificar con cualquier átomo
Resultado: ÉXITO
Ejemplo: X=5

Variable instanciada se intenta unificar con cualquier átomo distinto al valor que contiene
Resultado: FAIL
Ejemplo: X que contiene 5, X=6

Constante se intenta unificar con otra constante distinta
Resultado: FAIL

Ejemplo: 5=6

Expresiones se intentan comparar mediante símbolos de comparación que no "evalúan" y no coinciden totalmente, incluido el orden
Resultado: FAIL
Ejemplo: 5+3=3+5

Variable se intenta instanciar dos veces en la misma ejecución del programa. En el segundo intento de instanciación
Resultado: FAIL
Ejemplo: X=6, X=7

En resumen:
Una variable libre puede ser unificada por un término, pasando a ser una variable instanciada por los siguientes términos de la regla.
Una constante puede ser unificada con ella misma o con una variable libre.
Una variable particularizada puede unificarse con ella misma o con una variable libre.

El Backtracking es utilizado por PROLOG para resolver una meta propuesta. El procedimiento de backtracking consiste en generar un árbol de búsqueda de todas las posibles resoluciones que puede tener la meta en función de la base de conocimientos. Esto significa, que el algoritmo itera hasta que encuentra una solución. Cada vez que los predicados fallan y no son unificables se va generando una nueva rama hasta encontrar la solución deseada, de esta forma se va construyendo el árbol de búsqueda.

Puede ser que un problema se pueda resolver de varias formas. Es, por tanto, posible especificar que deseamos una nueva solución. El intérprete PROLOG ignora la solución encontrada hasta ahora y construye el árbol de búsqueda hasta generar una nueva solución o encontrar que ésta no existe.

Cuando un predicado se demuestra en función de una llamada a sí mismo, la llamada se convierte en un subobjetivo casi idéntico al objetivo a cumplir, salvo por el conjunto de datos sobre el que se aplica. Cuando se consigue la demostración de los subobjetivos generados en el proceso de recursión, se produce lo que se denomina *backtracking*, que funciona igual que en cualquier lenguaje.

No debemos de confundir recursividad con backtracking. Los predicados recursivos los diseñamos nosotros mientras que el procedimiento de backtracking es intrínseco al método de refutación que se usa para demostrar las metas.

Construcción de Bases de Conocimientos en PROLOG

La máquina virtual PROLOG toma como entrada la base de conocimientos expresada en forma clausal, el objetivo, también expresado en forma clausal, y genera una respuesta afirmativa en caso de que el objetivo se pueda demostrar aplicando el conocimiento almacenado en la base.
Resolver un problema, por tanto, es construir adecuadamente la base de conocimientos, y esta base expresada en lenguaje PROLOG junto con el conjunto de metas especificadas constituirán el programa.

Un programa PROLOG puede utilizar predicados recursivos, es decir se expresa en términos de sí mismo aplicado sobre un conjunto de datos distintos que tiende a convertirse en el conjunto de datos que satisface el caso o casos triviales del proceso de recursión. Por ejemplo, en el caso del problema del factorial, vamos calculando sucesivamente el factorial de un número decrementado del nivel anterior. Este número se aproxima cada vez más a 0, que es justamente el caso trivial de este proceso recursivo. Una vez que se alcanza este caso, las llamadas recursivas retornan los datos de salida y se produce la vuelta atrás.

Ejemplo:

```
factorial(0,1).
factorial(N, R):-
N1 is N-1,
factorial(N1, Y),
R is N*Y.
```

Igualmente, si ciertas realidades del mundo que se desea representar, se pueden expresar mediante una regla de inferencia en lugar de hacerlo con un conjunto de hechos, elegiremos la primera opción, de modo que en nuestra base de conocimientos, tendremos solamente los hechos que son necesarios para deducir otros que se pueden razonar a través de reglas.

La lógica se representa en forma de predicados. Estos predicados aparecen en tres formas distintas: como hechos, como reglas y como preguntas. La lógica formulada como hechos y reglas se define como base de conocimientos. A esta base de conocimientos se le pueden formular preguntas.

Consultas sobre la Base de Conocimientos

Las bases de conocimientos se construyen con el fin de que preguntemos al agente sobre nuevos hechos del mundo deducidos de dicha base. El agente debe

poseer un motor de inferencia para llevar a cabo el proceso de deducción de forma automática

El objetivo se plantea como un predicado nuevo basado en el conocimiento que tenemos. La máquina virtual o intérprete intentará unificar dicho predicado con los existentes en la base de conocimientos. Si puede unificarlo con alguno, nos dará una respuesta afirmativa. En caso contrario, nos proporcionará una respuesta negativa.

Cuando en la base de conocimientos sólo hay hechos

En este caso sólo tenemos un conjunto de predicados que expresan los hechos o afirmaciones que se producen en el mundo real. La forma de proceder de la máquina PROLOG, será por tanto, el intento de unificación con alguno de ellos.

predicado1(arg1, etc., argN).
predicado2(arg1, etc., argN).
etc.
predicadoM(arg1, etc., argN).

Cuando en la base de conocimientos hay hechos y reglas de inferencia no recursivas

Las reglas de inferencia permiten relacionar hechos o situaciones del mundo real para deducir otros hechos que, en principio, no son evidentes sin la utilización de dicha reglas.

Cuando en PROLOG tenemos una sentencia de la forma:
predicado(argumentos):-
predicado2(argumentos).

la máquina intenta unificar la parte izquierda de la regla mediante la demostración de la parte derecha. Dicha parte derecha se convierte en un subobjetivo a resolver. Las entradas y salidas se proporcionan a través de los argumentos, y se pueden utilizar variables locales o auxiliares para realizar cálculos que sólo afectan a esa parte derecha. De esta forma, en cada nivel de llamadas dichas variables locales funcionan del mismo modo que en cualquier lenguaje imperativo, ya que se consideran direcciones independientes unas de otras que sólo sobreviven en su nivel de llamada.

Estructura de un programa en Turbo PROLOG

domains	Establece el dominio de los argumentos de las reglas y los predicados. symbol — Grupo de caracteres que no contengan espacio ni caracteres especiales y comiencen con minúscula. string — Cadena de caracteres encerrada entre comillas dobles. integer — Número entero. real — Número real. char — Caracter ASCII encerrado entre apóstrofes.
predicates	Establece el nombre, tipo y número de argumentos de las reglas y los hechos.
clauses	Contiene las instrucciones, reglas y hechos que manipulan al programa.
goal	Se establece el objetivo a demostrar o verificar (sólo uno). Es la única parte opcional del programa.
/* */	Comentarios dentro del programa.
!	Instrucción de corte; una vez ésta encontrada, ya no realiza el backtraking para realizar otra búsqueda.
_	Variable anónima: no importa el valor del parámetro, siempre y cuando éste exista.
nl	Salto de línea.

Ejemplo: Programa que evalúa las raíces de una ecuación de segundo grado.

```
domains     var=real                          /* Define las variables */
predicates  resolver (var, var, var)          /* Define las funciones */
            revisa (var, var, var)
clauses           resolver (A, B, C) :-       /* Realiza el cálculo del determinante cuyos */
                  clearwindow,                /* parámetros son A, B y C */
                  D = B*B –4*a*C,
                  revisa (A, B, C), nl.       /* Manda llamar a la función revisa */
            revisa (_ , _ , D) :-
                  D<0,                  /* Caso 1 cuando el determinante es menor a 0 */
                  write ("Solución con números imaginarios"), ! .
            revisa ( A, B, C) :-
                  D = 0,                      /* Caso 2 cuando el determinante es cero */
                  X = - B / (2*A),            /* y sólo existe una solución */
                  write ("Una solución),
                  nl, write ("x= ", X) , ! .
```

```
revisa (A, B, C) :-                    /* Caso 3 cuando existen ambas raíces */
        raiz=sqrt (D),                 /* Calcula la raíz del determinante */
        X1 = (- B+ raiz) / (2*A),
        X2 = (- B - raiz) / (2*A),
        write ("Primera solucion x1 =", X1), nl,
        write ("Segunda solucion x2 = ", X"), nl.
```

Para resolver la ecuación: $2x^2 - 6x + 4 = 0$
goal: resolver (2, -6, 4).

```
resolver (2, -6, 4) :-                 /* A=2, B=-6, C=4 */
        D = 4,
        revisa (2, -6, 4).
revisa (_ , _ , 4) :-
        D < 0 ?                        /* No */
revisa (2, -6 , 4) :-
        D = 0 ?                        /* No */
revisa (2, -6 , 4) :-
        raiz = sqrt (4),               /* raiz = 2 */
        X1 = (6+2) / 2*2,              /* X1 = 2 */
        X2 = (6-2) /  2*2,             /* X2 = 1*/
```

Primera solucion x1 = 2
Segunda solucion x2= 1.

CAPÍTULO III.
SISTEMAS EXPERTOS (SE).

Los Sistemas Expertos son una expresión de los sistemas basados en el conocimiento. Con la aplicación de técnicas de Inteligencia Artificial finaliza la transición del procesamiento de datos al procesamiento de conocimientos.

Definición de Sistema Experto

Un sistema experto se define como un sistema basado en conocimiento que imita el pensamiento de un experto, para resolver problemas de un terreno particular de aplicación.

Un SE es una aplicación informática que simula el comportamiento de un experto humano en el sentido de que es capaz de decidir cuestiones complejas, y debe ser capaz de representar el conocimiento profundo de un campo en específico con el objetivo de utilizarlo para resolver problemas, justificar su comportamiento e incorporar nuevos conocimientos.

Una de las características principales de los sistemas expertos es que están basados en reglas, es decir, contienen conocimientos predefinidos que se utilizan para tomar todas las decisiones.

En teoría estos sistemas son capaces de razonar siguiendo los mismos pasos que seguiría un especialista (experto) en determinada materia (médico, matemático, biólogo, etc.) cuando resuelve un problema propio de su campo o de su disciplina. Por ello, el creador de un sistema experto tiene que comenzar por identificar y recoger del experto humano los conocimientos que este utiliza, pero sobre todo los conocimientos empíricos que se adquieren con la práctica.

Dado que los programas están basados en el conocimiento, un aspecto fundamental es la programación del conocimiento la cual hace uso de la representación explícita del conocimiento a usar por el sistema y de su interpretación y manipulación lógica por medio de métodos de inferencia que permiten deducir nuevo conocimiento a partir del que ya se dispone. Los sistemas expertos son máquinas que piensan y razonan como un experto lo haría en una cierta especialidad o campo. Por ejemplo, un sistema experto en diagnóstico médico requeriría como datos los síntomas del paciente, los resultados de análisis clínicos y otros hechos relevantes, y utilizando éstos, buscaría en una base de datos la información necesaria para poder identificar la correspondiente enfermedad.

Un Sistema Experto de verdad, no sólo realiza las funciones tradicionales de manejar grandes cantidades de datos, sino que también manipula esos datos de forma tal que el resultado sea inteligible y tenga significado para responder a preguntas incluso no completamente especificadas. Así, un sistema experto es un conjunto de programas de computadora que intenta imitar e incluso superar en algunas situaciones a un experto humano en un ámbito concreto de su actividad.

No pretende, en absoluto, reproducir el pensamiento humano, sino simplemente la pericia de un profesional competente (tengamos en cuenta que para construir un SE, se suele contar con grandes expertos en la materia que incorporan su conocimiento al sistema).

Esta pretensión es más sencilla ya que en algunos campos reducidos los expertos trabajan siguiendo reglas, aunque, generalmente, no sean conscientes de ello.

Características Generales

En los Sistemas Expertos:

- Su peritaje está restringido a un área específica (el dominio del problema).
- Existen capacidades de procesamiento numérico y simbólico.
- Hay capacidades para indicar cómo alcanzan sus conclusiones.
- Se usan conocimientos (almacenados o introducidos) para llegar a conclusiones.

Un SE debe ser capaz de:

- Interactuar eficazmente y en lenguaje natural con las personas.
- Manipular descripciones simbólicas y razonar sobre ellas.
- Razonar heurísticamente utilizando reglas que los expertos humanos consideran eficaces.
- Adquirir nuevos conocimientos y corregir o perfeccionar los que ya posea.
- Resolver problemas muy difíciles tan bien o mejor que un experto humano.
- Justificar sus conclusiones.
- Explicar por qué hacen preguntas cuando están intentando resolver un problema.
- Funcionar con datos erróneos o imprecisos.

Objetivos de los Sistemas Expertos:

- Supervisar. Comparar observaciones para determinar fluctuaciones.
- Diagnosticar. Inferir las causas del mal funcionamiento de un equipo a través de observaciones.

- Corregir. Prescribir remedios para un mal funcionamiento.
- Reparar. Ejecutar un plan para solucionar un problema.
- Instruir. Diagnosticar, corregir y evaluar los hábitos de los estudiantes.
- Controlar. Interpretar, predecir, reparar, y monitorear la conducta de sistemas.
- Predecir. Inferir posibles consecuencias de situaciones dadas.
- Interpretar. Inferir descripciones de situaciones a partir de datos sensoriales.
- Diseñar. Configurar objetos dentro de restricciones.
- Planificar. Desarrollar guías para acción.
- Clasificar. Establecer categorías para conjuntos de criterios.

Beneficios de los Sistemas Expertos

- Servir de auxiliares inteligentes para personas tomando decisiones o resolviendo problemas.
- Mejorar la consistencia y calidad del trabajo realizado.
- Aumentar el rendimiento profesional en términos de rapidez para alcanzar soluciones.
- Altos retornos por la inversión.
- Capturar la "experiencia" de una compañía para usarla en adiestramientos de las generaciones actuales y futuras.

Ventajas de los Sistemas Expertos

- Reemplazar al experto.
- Resolver problemas para los que no existe un modelo matemático adecuado o su solución es muy compleja.
- Preservar el conocimiento de expertos y hacerlo accesible a más personas.
- Capacidad de explicar al usuario el proceso de razonamiento para llegar a los resultados.

Estructura de un Sistema Experto

Básicamente, un SE está compuesto por:

- La base de conocimientos.
- El motor de inferencia.
- Interfaz del usuario.

Los participantes de un SE: experto del dominio e ingeniero de conocimientos se detallan en la capítulo siguiente (Representación de problemas de IA).

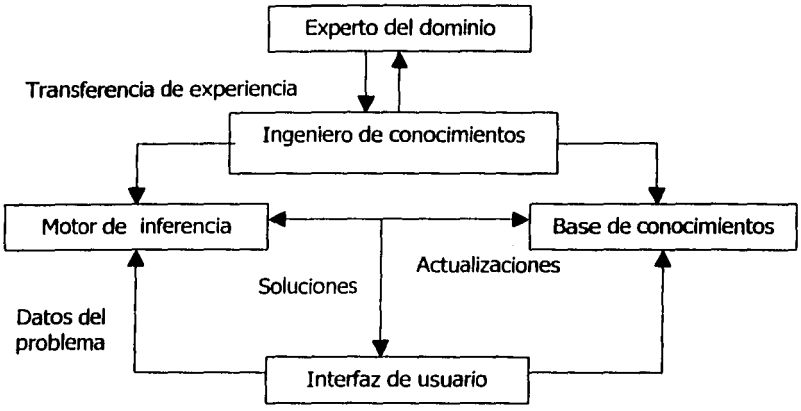


Figura1. Estructura de una Sistema Experto.

TESIS CON
FALLA DE ORIGEN

La base de Conocimientos ("knowledge base, KB")

Un SE posee el conocimiento del experto humano convenientemente formalizado y estructurado; esto es lo que se conoce como Base de conocimientos. Está constituida por la descripción de los objetos y las relaciones entre ellos, así como de casos particulares y excepciones.

Algunos sistemas basados en el conocimiento incluyen metaconocimiento o conocimiento sobre el conocimiento, que es la capacidad para buscar en la base de conocimientos y abordar la resolución del problema de una manera inteligente usando diferentes estrategias para la resolución con sus condiciones particulares de aplicación. Es decir, se trata de definir criterios mediante los cuales el sistema decide la estrategia de búsqueda a utilizar en función de unos datos iniciales.

La KB contiene los conocimientos relativos al dominio que el SE usará para alcanzar sus conclusiones y utiliza formalismos de representación para codificar los conocimientos. También contiene los hechos, relaciones, reglas relevantes al dominio del SE e idealmente, captura las reglas usadas por las personas expertas en ese dominio.
La representación del conocimiento en la KB tiene que ser: sencilla, independiente, fácil de modificar, transparente, relacional y potente.

Los pasos para construir una buena base de conocimientos son los siguientes:

1. Definir acerca de lo que se va a hablar. El conocimiento del dominio debe ser suficiente para poder definir objetos y hechos.
2. Escoger un vocabulario para predicados, funciones y constantes. Traducir los conceptos importantes del nivel del dominio a nombres de nivel de lógica.
3. Codificar todo el conocimiento general relativo al dominio.
4. Codificar una descripción de un caso específico del problema. Obtención de oraciones relativas a casos específicos de conceptos.
5. Hacer consultas al procedimiento de inferencia y obtener respuestas.

El Motor de Inferencia ("inference engine")

También llamado intérprete de reglas, es un módulo que se encarga de las operaciones de búsqueda y selección de las reglas a utilizar en el proceso de razonamiento. Por ejemplo, al tratar de probar una hipótesis dada, el motor de inferencia irá disparando reglas que irán deduciendo nuevos hechos hasta la aprobación o rechazo de la hipótesis objetivo.

Definiciones:

- Es la unidad lógica con la que se extraen conclusiones de la base de conocimientos, según un método fijo de solución de problemas que está configurado imitando el procedimiento humano de los expertos para solucionar problemas.
- Es la parte usada para realizar las deducciones necesarias para alcanzar las conclusiones.
- Es un programa que usa los distintos hechos y reglas para alcanzar una conclusión.
- El motor de inferencia es el medio por el cual se controlan y aplican los conocimientos.
- Son mecanismos de inferencia que permiten que el sistema razone a partir de los datos y conocimientos de entrada para producir los resultados de salida. Estos mecanismos indican el orden en el que el sistema realiza los pasos de razonamiento de aceptar entradas y producir salidas.

El funcionamiento del motor de inferencia se resume en los siguientes pasos:
1. Evaluación. Se selecciona el conocimiento a emplear.
2. Comprobación. Se indica si el conocimiento es aplicable.
3. Ejecución. Se aplica el conocimiento.
4. Controlar las reglas activas. Mediante la resolución de conflictos (primera regla aplicable, regla más prometedora, etc.).

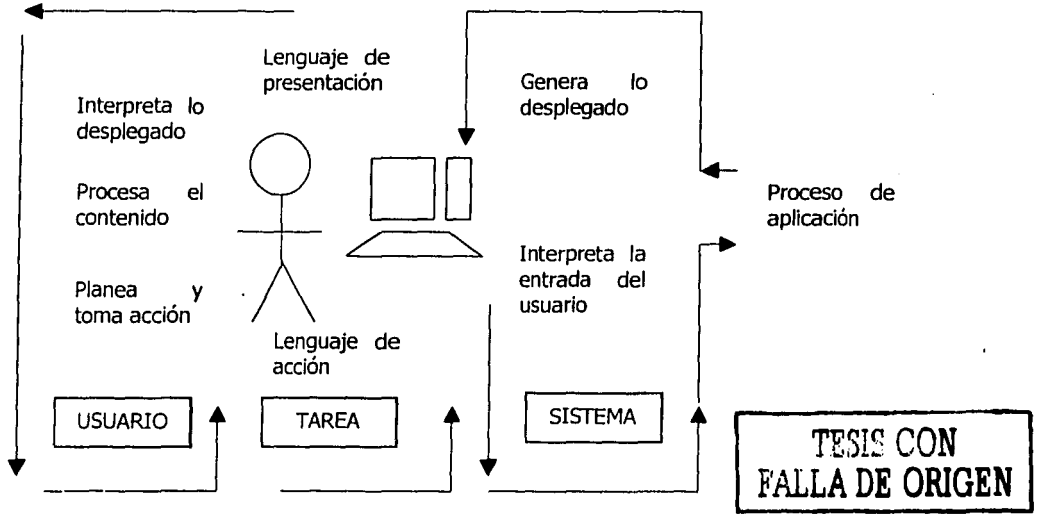
La interfaz

Es aquella parte que sirve para la comunicación entre el usuario y el programa.

Los datos desplegados en la máquina proveen un contexto para la interacción y dan instrucciones de acción para el usuario, asumiendo que el usuario sabe como interpretar lo desplegado. El usuario formula una pregunta, los datos pasan al sistema a través de la interfaz. La calidad de una interfaz desde el punto de vista del usuario depende en lo que ve o siente; el usuario debe saber entender lo que está sintiendo y qué acciones puede o debe hacer para obtener los resultados que necesita. El proceso consta de los siguientes elementos:

1. Lenguaje de presentación. Es la información desplegada para el usuario y puede ser mostrada como menús, ventanas o texto. Puede ser estático o dinámico, numérico o simbólico y puede aparecer como voz o impreso.
2. Reacción del usuario. El usuario interpreta lo desplegado, procesa el contenido y planea la acción.
3. Lenguaje de acción. El usuario puede seleccionar un tema de un menú, contestar una pregunta, mover una ventana o teclear un comando.
4. Computadora. Interpreta la entrada del usuario, ejecuta la tarea y genera básicamente un lenguaje de presentación (salida de la computadora).

El proceso anterior se encuentra esquematizado en la Figura 2.



TESIS CON
FALLA DE ORIGEN

Figura 2. Interfaz de un Sistema Experto.

Estilos de interfaces

Se refiere a la comunicación interactiva entre el usuario y la computadora. Determina cómo la información es desplegada en el monitor y cómo es introducida en la computadora.

Comparación de los modos de interfaces

INTERFAZ / DIMENSIONES	MENU	FORMAS	COMANDOS	OBJETOS	PREGUNTAS/ RESPUESTAS
VELOCIDAD	Lento	Moderado	Rápido	Lento	Lento
PRECISIÓN	Libre de errores	Moderado	Muchos errores	Libre de errores	Moderado
TIEMPO DE ENTRENAMIENTO	Corto	Moderado	Largo	Corto	Corto
PREFERENCIA	Muy alta	Baja	Con entrenamiento	Alta	Alta
PODER	Bajo	Bajo	Muy alto	Moderado-Alto	Moderado
FLEXIBILIDAD	Limitada	Muy limitada	Muy alta	Moderada-Alta	Alta
CONTROL	El sistema	El sistema	El usuario	Usuario, sistema	Sistema

Tabla1. Comparación de Interfaces.

Comparación entre un sistema experto y un programa tradicional

Un programa tradicional puede esquematizarse de la siguiente manera:

Datos + Algoritmo + control + Entradas / Salidas = **Programa tradicional**

Mientras que un sistema experto estaría definido de la siguiente forma:

Base de Hechos + Base de Conocimientos + Motor de Inferencia + Entradas / Salidas = **Sistema Experto**

Del esquema se desprende que la base de hechos es en un sistema experto, lo que los datos son en un programa tradicional.

De la misma manera la base de conocimientos reemplaza al algoritmo.

Y el motor de inferencia es el programa.

CON FALLA DE ORIGEN

Ciclo de vida de un Sistema Experto

El modelo del ciclo de vida utilizado con éxito en varios proyectos de sistemas expertos es el modelo lineal, adaptado de Bochsler. Este ciclo de vida está formado por varias etapas, que van de la planeación a la evaluación de sistemas, y describe el desarrollo del sistema hasta el punto en que se evaluaran sus capacidades funcionales. El ciclo de vida mostrado podría considerarse un circuito del modelo en espiral, donde cada etapa esta formada por tareas. Quizá no sean necesarias todas las tareas para una etapa, en especial una vez que el sistema se encuentra en mantenimiento y evolución; en cambio, las tareas tienen la intención de servir como una composición de todas las tareas para el ciclo de vida entero, desde el concepto inicial hasta el retiro del software. Las tareas también dependerán del tipo exacto de aplicación que se está construyendo, sólo deben considerarse como una guía y no como requisitos absolutos que deben realizarse para completar cada etapa.

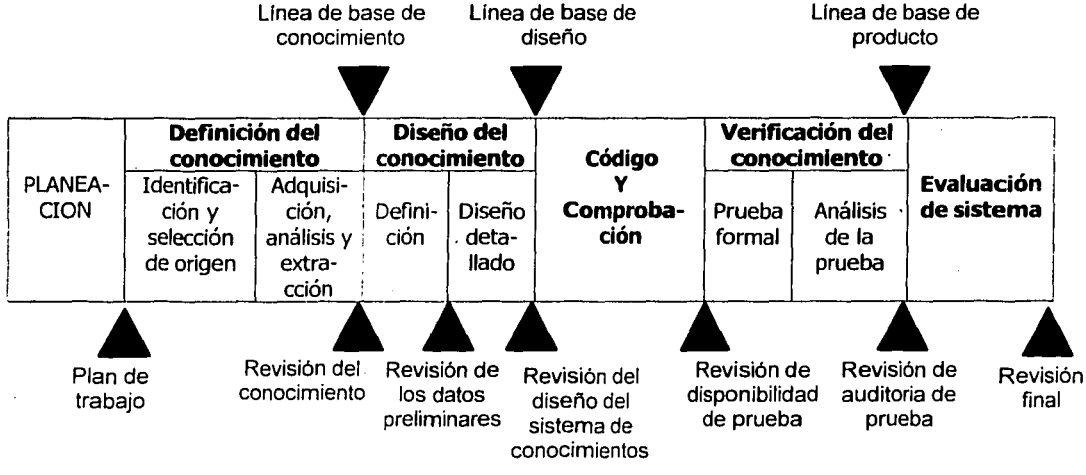


Figura 3. Modelo lineal del ciclo de vida para desarrollar sistemas expertos.

Construcción de un Sistema Experto

TESIS CON
FALLA DE ORIGEN

Los pasos a seguir para la Construcción de un SE son los siguientes:

a) Estudio de la demanda.

Una orientación sobre algunas de las razones que hacen pensar en la idoneidad de un SE para la resolución de un problema son:

- Si la experiencia es importante y es escasa o se pierde.
- Si la experiencia humana se precisa cada vez en más lugares a la vez.
- Si la experiencia se precisa en entornos hostiles.
- Si la variación de los problemas es tan rápida que no permite la formación de los expertos humanos.
- Si el encontrar soluciones aunque estas no sean óptimas es beneficioso.

También hay que determinar el costo del SE, que se desglosa en:

- Costo de desarrollo.
- Costo de explotación: computadoras utilizadas y de medios de intercomunicación (red local, módems, etc.), licencias de utilización de los lenguajes y herramientas utilizados, cursos de formación de los operadores del sistema, personal que manejará el SE.
- Costo de mantenimiento: actualización de las bases de conocimiento, actualización de los sistemas operativos, lenguajes y herramientas, mantenimiento y reparación de las computadoras.

Todos estos costos van a determinar desde un principio el tipo de "software" y de "hardware" que pueden emplearse y con ello hay que decidir la viabilidad económica del proyecto.

b) Análisis del problema.

En esta fase del proyecto hay que determinar primeramente la idoneidad de la metodología de SE para la resolución del problema, después es necesario delimitar el campo en el que se desea la construcción de un SE y la tarea o tareas que se desea que realice el SE (interpretación, diagnóstico, etc.), y por último determinar los dominios (pueden requerirse uno o más dominios de conocimiento), la cantidad de conocimientos y el tipo de conocimientos necesarios en cada dominio (concretos / generales o metaconocimientos, cierto / incierto, algorítmico, heurístico, imperativo o procedimental / declarativo, teóricos , prácticos, exactos y aproximados, etc.).

Requisitos:

- No debe requerir el sentido común ni la inspiración para la resolución de los problemas planteados (jeroglíficos y acertijos, etc.), por el contrario debe requerir conocimiento, juicio y experiencia.
- Requiere la intervención de un experto para su resolución (asesoría de inversiones). Lógicamente si no requiere a una persona experta, puede ser por dos motivos o que el problema es trivial y por tanto existen otras formas más sencillas de resolverlo (ordenación alfabética de nombres) o sea complejo y ya esté resuelto de otra forma (resolución de ecuaciones de grado n) o en forma de S.E (determinación de la fórmula de un

compuesto orgánico), en cuyo caso habría que estudiar la idoneidad de las soluciones ya existentes.
- El problema debe tener solución, es decir que no sea imposible, y ésta reporte beneficios a un conjunto de usuarios.
- El conocimiento empleado no está contenido en libros o manuales o si lo está, su manejo y relación no es trivial (legislación, medicina, etc.). Es decir que no exista un método definido y único que solucione el problema.
- La forma de actuar no es rígida ni preestablecida (reparación de averías).
- No existen ecuaciones o algoritmos que definen su comportamiento en un tiempo finito ni modelos estadísticos que se adapten, y si no es así se necesita un seguimiento (explicación y justificación).
- Debe ser un problema concreto, con el fin de que el conocimiento requerido para su resolución sea limitado, es decir con fronteras definidas, y reducido (diagnóstico de las averías de un determinado motor).
- Está definida, o es fácilmente definible, una nomenclatura que permita describir y representar el problema.
- No debe ser ni excesivamente complejo, pues el costo de desarrollo sería muy grande, ni trivial pues no sería rentable el desarrollo.

Por otra parte el SE debe ser el único sistema viable de resolución, pues si existen otros seguramente estén más desarrollados lo que facilitaría su resolución al aplicar técnicas más conocidas. Además, debe resultar más barato ya que no tiene sentido que por la novedad se gaste más dinero del imprescindible.

En esta fase deben de quedar definidos los objetivos que debe satisfacer el producto final, como puede ser: tiempo de respuesta, sistemas y aplicaciones con las que debe comunicarse, etc.

c) Elección de la fuente de conocimientos.

Es necesario contar con un experto o grupo de expertos que estén dispuestos a aportar sus conocimientos. Estos expertos deben de ser reconocidos como tales al menos por el grupo de usuarios.

d) Preselección del soporte.

En este punto hay que optar por elegir los lenguajes, herramientas, etc. para su implantación y el ordenador de forma conjunta.

Deben considerarse el tipo de razonamiento a utilizar (exacto o aproximado, etc.), el sistema de representación (reglas de producción, marcos, etc.) y el tiempo de respuesta (tiempo real o tiempo diferido, etc.).

e) Obtención del conocimiento.

El conocimiento debe extraerse del experto humano y formalizarse a fin de que sea completo, sin ambigüedades y después representarlo en el sistema elegido.

Este punto es el que consume los mayores esfuerzos del desarrollo ya que por una parte es largo y tedioso, y por otra parte una incorrección en la base de conocimientos invalidaría el SE.

Siete son los pasos a cubrir en esta etapa:

1. Familiarización del ingeniero del conocimiento con el problema (mediante la lectura de libros, revistas, etc.) y su entorno (términos utilizados).
2. Delimitar el problema.
3. Determinar entre los expertos disponibles los más idóneos para la aportación del conocimiento (cantidad y calidad de sus conocimientos, grado de dedicación al desarrollo, interés mostrado) facilidad de expresión y de comunicación, reconocimiento de los usuarios finales como expertos válidos.
4. Motivar a los expertos en el desarrollo del SE desde el comienzo de su colaboración hasta el final de la misma.
5. Plantear de forma general la resolución del problema, que se hace en base a las aportaciones de un grupo de expertos; a este paso también se le conoce como la obtención de los conocimientos preliminares.
6. Complementar la base de conocimientos según el esquema definido en el paso anterior; se hace en base a los conocimientos de un número reducido de expertos (típicamente uno), también se conoce a este paso como la obtención del conocimiento con detalle.
7. Completar y corregir la base de conocimientos sobre el soporte del SE. Este paso, se realiza normalmente sobre el prototipo, ya que muchas veces parte del conocimiento ha sido transferido de un sistema de representación a otro, por parte del ingeniero de conocimiento con el fin de unificarlos.

Existen herramientas de ayuda al ingeniero del conocimiento en su tarea como es el programa TEIRESIAS que refina la base de conocimientos, localizando y depurando los errores (reglas incompletas, inconsistencias, etc.). Muchas de las herramientas y lenguajes cuentan con herramientas potentes para la detección y corrección de errores en la KB existiendo incluso editores de KB.

Por último, debemos señalar que la estrategia de extracción del conocimiento depende del software elegido.

Este punto se continua por lo menos hasta la realización del sistema operacional, pudiendo prolongarse con una menor dedicación durante toda la vida del SE.

f) Selección del soporte.

En este punto de nuevo se elige el soporte, que no tiene porque ser el mismo que en el punto precedente ya que ahora contamos con más parámetros como son.

- El tipo de motor de inferencia.
- La capacidad que requiere la base de conocimientos.
- Necesidades de acceso del SE a programas ya desarrollados (ejemplo base de datos).
- Número de usuarios y modos de acceso de éstos, tanto en su forma física (directo, red local, etc.) como en los privilegios de acceso (experto, usuario, etc.).
- Las interfaces con el experto y el usuario.
- Formas de actualización de la base de conocimientos.

g) Construcción del prototipo.

El prototipo del SE sirve para evaluar por parte de los expertos la validez del mismo desde el punto de vista del conocimiento y no de la herramienta.

Muchos de los SE que se han desarrollado han terminado su vida en este punto, pues su objetivo era mostrar la viabilidad de las técnicas de los SE para la resolución de ciertos tipos de problemas.

h) Validación del prototipo.

Este es un punto delicado pues dado que el número de problemas que pueden plantearse en teoría es ilimitado no pueden probarse todos ellos; por otra parte al no ser la resolución determinística de las pruebas realizadas, no se puede asegurar cual va a ser el funcionamiento en otros casos.

Las validaciones por lo general consisten en enfrentar al SE contra el experto humano, el cual estudia el comportamiento del mismo frente a una gama de problemas comparando la resolución que da con la que él realizaría. La palabra que más espera oír el ingeniero del conocimiento del experto humano es "se me parece".

i) Construcción de un modelo operacional.

En este punto puede ser necesario el cambio del soporte físico del SE, tanto el software como el hardware con el fin de adaptarse a los requerimientos expresados por el usuario, ambos tienen que estar además completamente integrados. Aparte de la traslación del SE en esta fase debe cuidarse en especial la protección del flujo normal del programa frente a malas operaciones por parte del usuario, las entradas y salidas de los datos de forma que sean lo más sencillas posibles, la optimización del consumo de recursos de la computadora, etc.

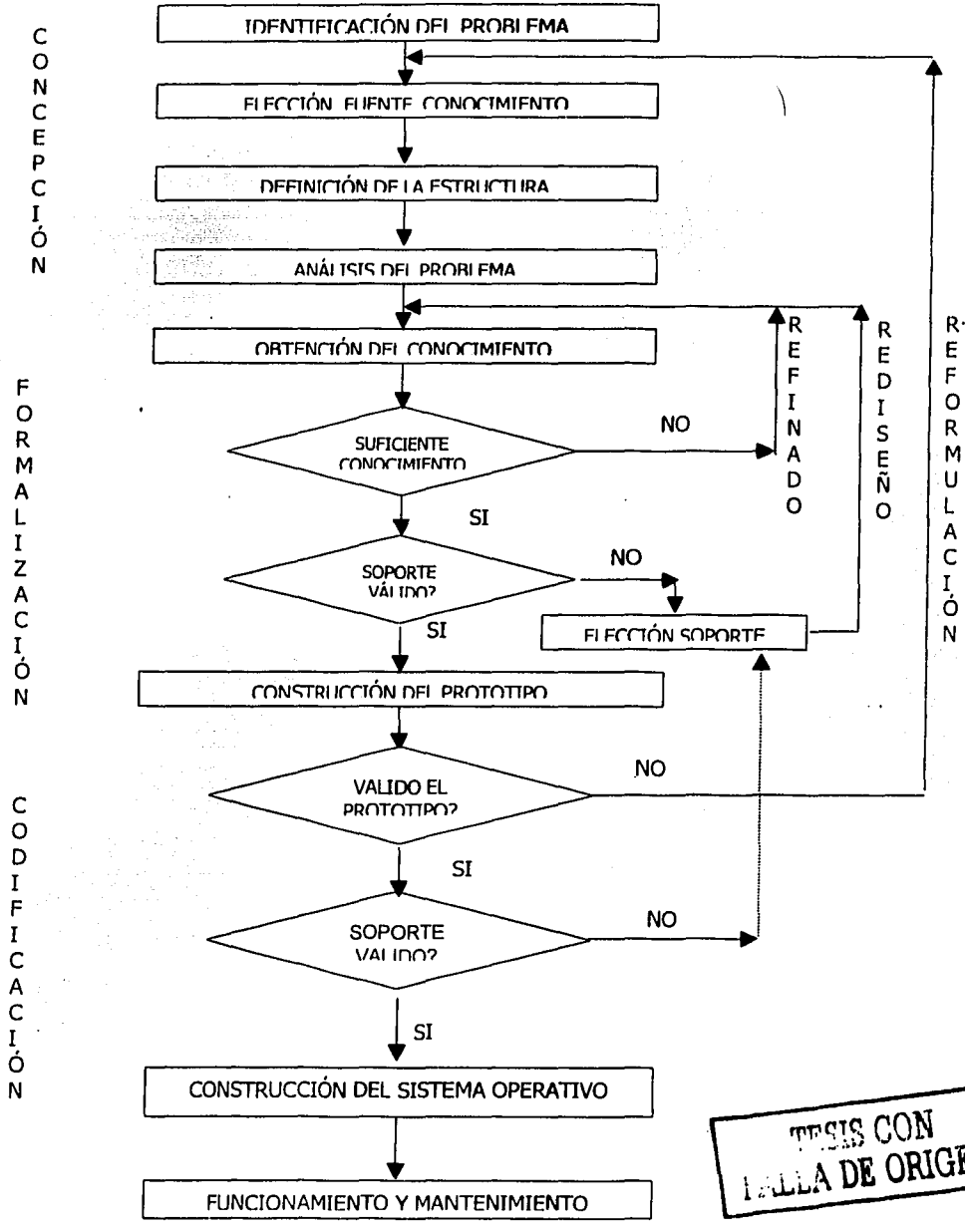
Hay que confeccionar en esta fase toda la documentación del SE como manuales.

j) Mantenimiento.

A todo SE debe mantenérsele el conocimiento que contiene con el fin de que esté "al día" y considere la experiencia.

Un SE es un sistema "vivo".

Podemos resumir los pasos para la construcción de un SE en el siguiente diagrama:



TESIS CON
FALLA DE ORIGEN

Figura 4. Construcción de un SE.

CAPÍTULO IV.
REPRESENTACIÓN DE PROBLEMAS DE INTELIGENCIA ARTIFICIAL.

Ingeniería del conocimiento

Para la construcción de sistemas basados en estructuras de datos y en procedimientos para calcular funciones a partir de ellos es preciso contar con:

1. Paradigmas de representación del conocimiento que sean capaces de incorporar en su estructura un grado de complejidad suficiente para la clase de problemas a resolver por el sistema.
2. Procedimientos de interpretación que sean capaces de obtener las respuestas según modelos de razonamiento apoyados en la representación anterior.
3. Métodos efectivos de adquisición del conocimiento.

Al conjunto de técnicas, métodos personales, instrumentación y criterios para la adquisición de dicho conocimiento se le llama **Ingeniería del conocimiento**; dichos criterios son:

- Constructividad. Se deben incorporar conceptos suficientes de forma de entender a los expertos del tema, además de construirse de manera incremental, es decir, por la progresiva incorporación de conocimiento.
- Operatividad. Deben de ser evaluables por un procedimiento efectivo y con el máximo contenido teórico para resolver la clase de problemas que debe de resolver.

Funciones de la ingeniería del conocimiento:

- Adquisición del conocimiento.
- Representación del conocimiento.
- Validación del conocimiento.
- Explicación del conocimiento.
- Inferencias.
- Mantenimiento.
- Procesos para generar aplicaciones de IA.

El ingeniero del conocimiento

En el desarrollo de un SE existen cuatro tareas muy diferenciadas que en principio deben desarrollar cuatro personas o grupos diferentes dado lo diverso de las tareas a realizar.
Las personas que intervienen son:

- El experto.
- El ingeniero del conocimiento.
- El programador.
- El ingeniero informático

El experto: es la persona o grupo de personas que proporcionan los conocimientos sobre el campo de actuación del SE y validan el funcionamiento del mismo desde el punto de vista de su aptitud para desarrollar las tareas para las que se ha desarrollado.

El ingeniero del conocimiento: es la persona que aplica el conocimiento y métodos que permiten adquirir y representar el conocimiento del experto en un sistema informático.
Es sin duda alguna la persona clave en el desarrollo, y existe ya un perfil que debe reunir el ingeniero de conocimiento para que su labor sea útil:

- Espíritu abierto: que le permita la comprensión de las exposiciones de los expertos, el ser consciente de su ignorancia sobre el campo del experto y el saber tratar a personas de muy diversos campos y niveles de formación.
- Conocimientos multidisciplinarios: con el fin de poder abordar problemas de dominios muy diversos y que lo mantengan al día tanto a nivel de software como de hardware.
- Curiosidad intelectual: que le lleve a profundizar en los problemas y en las soluciones de los mismos.
- Organizado: con el fin de poder manejar grandes cantidades de informaciones muy diversas.

El Programador: es el encargado de transferir el conocimiento obtenido a un lenguaje o entorno de desarrollo para un ordenador en concreto. Indudablemente debe conocer los lenguajes y entornos de programación de SE más usuales.

El ingeniero informático: es el encargado de elegir o desarrollar el software y el hardware adecuado en cada caso. Debe estar al día de los últimos avances en este campo.

Esta estructura es completamente teórica, ya que al momento de emprender el desarrollo de un SE se empieza por descubrir la carencia de técnicos que reúnan estos conocimientos, por lo que hay que proceder a su formación lo que alenta el desarrollo

e induce a que se unan las tareas de ingeniero del conocimiento, ingeniero informático y programador.

Procesamiento humano de la información. (Modelo Newell-Simon).

Allen Newell y Herbert Simon [3] proponen un modelo del problema humano que hace una analogía entre el proceso de una computadora y el proceso humano de la información. El procesamiento humano de la información consiste en los subsistemas perceptivo, cognoscitivo y motor; los cuales son mejor representados en el siguiente diagrama:

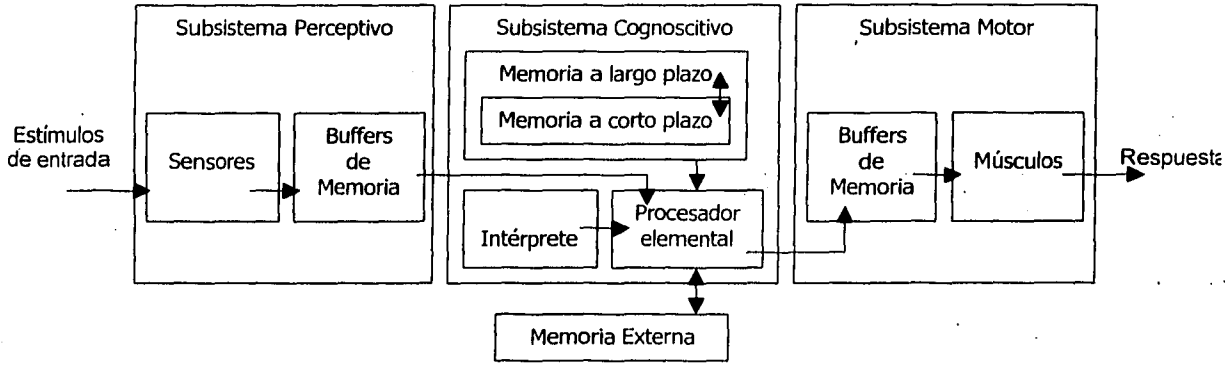


Figura 5. Procesamiento humano de la Información.

Subsistema Perceptivo.

Los estímulos externos son la entrada del sistema humano de procesamiento de la información. Estos estímulos se introducen a través de sensores, como los ojos o los oídos. El subsistema perceptivo esta formado por esos sensores y por memorias internas (buffers) que almacenan brevemente la información entrante mientras espera ser procesada por el subsistema cognoscitivo.

TESIS CON
FALLA DE ORIGEN

3 Herbert Simon y Allen Newell, ambos pioneros científicos de la computación y fundadores de la IA como parte de las ciencias de la computación. Ellos escribieron que: "etc. existe ahora en el mundo máquinas que piensan, que aprenden y que crean. Más aún, su habilidad para hacer estas cosas va a incrementarse rápidamente hasta en el futuro visible que el rango de problemas que puedan manejar será coextenso con el rango al cual la mente humana ha sido aplicada".

Subsistema Cognoscitivo

Memoria a corto plazo. Poco después de que la información sensorial queda almacenada, el procesador cognoscitivo transfiere una parte de ella a una memoria de trabajo a corto plazo. No todos los preceptos de las memorias intermedias se codifican. Los sentidos están continuamente depositando una gran cantidad de información en las memorias intermedias. El sistema cognoscitivo se encarga de hacer la selección y proceso de codificación.

El procesador cognoscitivo como la unidad central de proceso (CPU) de una computadora, funciona cíclicamente sacando información de las memorias sensoriales intermedias y transfiriéndola a la memoria de trabajo.

En las tareas más sencillas (por ejemplo, encender/apagar la luz) el sistema cognoscitivo sirve simplemente como un canal para transferir información de las entradas sensoriales a las salidas motoras.

Las tareas más complejas involucran más información (por ejemplo, el aprender un lenguaje de programación nuevo), para llevar a cabo estas tareas, el procesador cognoscitivo cuenta con el apoyo de un segundo sistema de memoria, la memoria a largo plazo.

Memoria a largo plazo. La memoria a largo plazo esta formada por una gran cantidad de símbolos almacenados, junto con un sistema de indexación complejo. La memoria es una vasta red de racimos. El aprendizaje y el recuerdo se dan conforme se van èstableciendo y revisando los enlaces entre racimos. Conforme vamos creciendo agrupamos más y más informaciones sobre conceptos cada vez más abstractos. Conforme la activación se propaga hacia nuevos racimos, los previamente activados se van haciendo menos accesibles. Esto se debe a los limitados recursos del procesador cognoscitivo humano.

Subsistema Motor.

Tras explorar y buscar en las memorias, normalmente se envía información al sistema motor. Los procesadores motores provocan acciones en los músculos y en otros sistemas internos. Ello, a su vez, tiene como consecuencia alguna actividad observable.

Adquisición del conocimiento

Existen cinco tipos de adquisición de conocimiento que son: declarativo, procedimental, semántico, episódico y metaconocimiento.

Declarativo. Es una representación descriptiva del conocimiento. Nos indica cómo son los hechos, este conocimiento se expresa como un conjunto de resultados. El experto expresa sus afirmaciones y sus asociaciones. Este tipo de conocimiento se considera como superficial o de nivel básico ya que la información experta sólo se verbaliza. El conocimiento declarativo es esencialmente importante en la etapa de la adquisición del conocimiento.

Procedimental. Este conocimiento considera la forma en la cual una situación ocurre bajo diferentes circunstancias. Este conocimiento incluye sentencias paso a paso e instrucciones del tipo "hasta que", e involucra respuestas automáticas a estímulos, también nos indica cómo usar el conocimiento y cómo hacer inferencias.

Semántico. Este conocimiento refleja la estructura cognitiva que involucra el uso de la memoria a largo plazo y utiliza: palabras o símbolos significativos (sintaxis y semántica) y sus reglas de uso (símbolos especiales como * / + -), palabras o símbolos asociados y sus interrelaciones (estructuras de control), y algoritmos para manipular símbolos, conceptos y relaciones.

Episódico. Es aquel conocimiento que puede considerarse autobiográfico ya que es información experimental organizada como un caso o como un episodio, aunque esto se realiza en la memoria de largo plazo, regularmente es clasificado en tiempo y espacio.

Metaconocimiento. Es el conocimiento acerca del conocimiento, es decir, se refiere al conocimiento de operación del sistema basado en el conocimiento, esto es, acerca de su habilidad para razonar. Son las técnicas y procedimientos para manipular el conocimiento.

Proceso de adquisición del conocimiento

El proceso general de adquisición del conocimiento puede dividirse en cinco etapas:

1. **Identificación**. Durante esta etapa, el problema y la mayoría de sus características se definen o se identifican. El problema puede dividirse en subproblemas (si es necesario), los participantes (expertos, usuarios, programadores) son identificados y los recursos son bosquejados. El ingeniero del conocimiento aprende acerca de la situación y entre todos los participantes determinan el propósito de la aplicación.

2. **Conceptualización**. Si el conocimiento importante a la situación de decisión está completamente diversificado, entonces es necesario determinar los conceptos y las relaciones a utilizar. Estas y otras cuestiones son contestadas durante la conceptualización ya que las siguientes preguntas son contestadas en esta etapa: ¿qué información debe utilizarse y cómo puede representarse en la base de conocimientos?, ¿son las reglas de producción un buen medio de representación?, ¿cómo puede extraerse el conocimiento?.

3. **Formalización**. El conocimiento adquirido debe ser representado en la base de conocimientos, la manera en la cual el conocimiento es representado y organizado debe determinar la metodología de adquisición. En esta etapa , la adquisición del conocimiento se mezcla con la representación y varios elementos de software y hardware son analizados. Esta etapa es muy difícil porque incluye la obtención del conocimiento de los expertos.

4. **Implementación**. Esta etapa involucra la programación del conocimiento tomando en cuenta que es posible una adquisición adicional o realizar cambios. En esta etapa es desarrollado un prototipo que resuelva el problema.

5. **Prueba**. En esta etapa el ingeniero de conocimientos pone a prueba al sistema, aplicándolo a diversos ejemplos. Los resultados son mostrados al experto y las reglas que manejan el conocimiento son revisadas si es necesario. En otras palabras, es verificada la validez del conocimiento.

Métodos de IA para resolver problemas

Representación de espacios de estado

El método de resolución de cualquier problema usando IA involucra tres grandes elementos: Espacio de estado del problema, Estados metas u objetivos y Operadores o reglas; representados en la Figura 3.

Espacio de estados. Es donde se define la situación del problema y las condiciones existentes.

- *Estado*. Son valores instantáneos de las variables y las condiciones existentes, son alternativas potenciales de solución de problemas. Todos los estados son únicos. Un estado es la representación de un problema en un instante dado. Para definir el espacio de estados no es necesario hacer una enumeración exhaustiva de todos los estado válidos, sino que es posible definirlo de manera más general.

- *Estado inicial.* Consiste en uno o varios estados en los que puede comenzar el problema.

Metas u objetivos. Son aquellos estados que se desea alcanzar y que representan la respuesta final o la solución. El estado objetivo consiste en uno o varios estados que se consideran como solución aceptable.

Operadores o reglas. Son los procedimientos para cambiar de un estado a otro; el operador recibe el proceso por el cual alguna acción es seleccionada para cambiar del estado inicial a otro que esté más cercano de la meta. Los operadores mueven el problema de un estado al siguiente siguiendo una guía o una estrategia de control hasta que la meta es alcanzada. Las reglas describen las acciones u operadores que posibilitan un pasaje de estados. Una regla tiene una parte izquierda y una parte derecha. La parte izquierda determina la aplicabilidad de la regla, es decir, describe los estados a los que puede aplicarse la regla. La parte derecha describe la operación que se lleva a cabo si se aplica la regla, es decir, como obtener el estado sucesor.

Por ejemplo, en el problema de jugar al ajedrez:

- El espacio de estados son la totalidad de tableros que se puede generar en un juego de ajedrez.
- El estado inicial es el tablero de 8 x 8 donde cada celda contiene un símbolo de acuerdo a las piezas situadas en la posición de apertura.
- El objetivo o estado final se define como cualquier posición de tablero en la que el contrario no puede realizar ningún movimiento legal y su rey esté amenazado.
- Las reglas son los movimientos legales, que pueden describirse mediante un patrón, una parte con la posición actual de tablero y otra parte que describe el cambio que debe producirse en el tablero.

Dado que escribir exhaustivamente todas las reglas es imposible prácticamente, (en el ejemplo, escribir todas las posiciones de tablero), las reglas deben escribirse de la manera más general posible.

La representación como espacio de estados forma parte de la mayoría de los métodos de IA. Su estructura se corresponde con la resolución de problemas porque:

- permite definir formalmente el problema, mediante la necesidad de convertir una situación dada en una situación deseada mediante un conjunto de operaciones permitidas;
- permite definir el proceso de resolución de un problema como una combinación de técnicas conocidas y el proceso de búsqueda (la técnica general de

exploración del espacio intenta encontrar alguna ruta desde el estado actual hasta un estado objetivo).

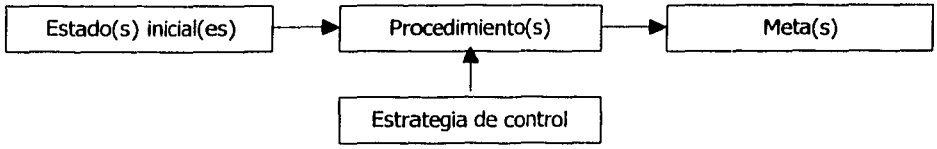


Figura 6. Representación de espacios de estado.

Métodos de búsqueda en un espacio de estados

Los problemas de IA pueden resolverse con el uso de reglas en combinación con una estrategia de control para trasladarse a través del espacio de estados hasta encontrar un camino desde el estado inicial hasta el estado final. Se elige una regla entre aquellas cuya parte izquierda concuerda con el estado actual. Se aplica la regla elegida realizando el cambio de estado tal como se describe en la parte derecha de la regla. Si el nuevo estado es estado objetivo o final se ha encontrado la solución. En caso contrario se continúa con la aplicación de reglas al nuevo estado.

Una estrategia de control especifica el orden en el que se deben aplicar las reglas, así como también la forma de resolver conflictos cuando es posible aplicar más de una regla. Para que una estrategia de control sea válida debe cumplir con dos requisitos:

- Causar cambios: las estrategias de control que no causan cambios de estado nunca alcanzan la solución. Un ejemplo de estrategia de control que no causa cambios es seleccionar siempre la primera regla aplicable de la lista de reglas definidas.

- Ser sistemática: las estrategias de control que no son sistemáticas pueden utilizar secuencias de operaciones no apropiadas varias veces hasta alcanzar la solución. Un ejemplo de estrategia de control no sistemática es seleccionar la regla a aplicar al azar. Esta estrategia puede encontrar la solución eventualmente, pero luego de haber realizado varios pasos innecesarios e incluso haber vuelto varias veces al mismo estado.

En caso de no contar con una aproximación directa al problema, el proceso de búsqueda resulta fundamental en la resolución del mismo.

TESIS CON
FALLA DE ORIGEN

Los algoritmos son estrategias de control sistemáticas. Todos se basan en considerar un árbol de estados cuya raíz es el estado inicial, y en cada nivel se hallan los estados sucesores correspondientes.

A continuación se muestra una tabla comparativa de los métodos de búsqueda principales:

MÉTODO		PROCESO DE BÚSQUEDA	PRUEBA	TIPO DE SOLUCIÓN
Métodos ciegos	Búsqueda completa	Todas las soluciones posibles son encontradas	Las comparaciones terminan cuando todas las alternativas son revisadas	Óptima
	Búsqueda parcial	Sólo algunas soluciones son encontradas	Comparaciones hasta encontrar alguna solución	La "mejor" entre las soluciones encontradas
Métodos heurísticos		Se analizan sólo soluciones prometedoras	Se detiene cuando la solución es "buena"	Buena solución

Tabla 2. Métodos de Búsqueda en un espacio de estados.

Métodos ciegos exhaustivos

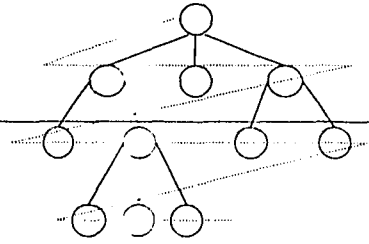
También llamados métodos sin información ya que en ellos, el agente sólo puede diferenciar un estado que es meta de uno que no lo es. No posee información respecto a cuántos pasos necesita dar o a qué distancia está de la meta; en estos métodos, la búsqueda de la solución se realiza analizando todos y cada uno de los posibles caminos de solución. Las estrategias de búsqueda sin información se diferencian en el orden en que se expanden los nodos, y existen básicamente dos métodos: búsqueda en amplitud y búsqueda en profundidad.

En ambos tipo de búsqueda, ésta se realiza partiendo del nodo raíz, de arriba hacia abajo y de izquierda a derecha.

Búsqueda en amplitud

En esta búsqueda, iniciando en el nodo raíz:

1. Se aplica a cada nodo todas las posibles reglas que se puedan aplicar a dicho nodo.
2. Antes de realizarse un siguiente nivel de nodos, se debe verificar que todos los nodos de un nivel sean expandidos o analizados.
3. Este proceso continua hasta obtener todos los nodos terminales o nodo solución.



TESIS CON
FALLA DE ORIGEN

Figura 7. Búsqueda en amplitud.

Ejemplo: Dado el árbol siguiente, donde B y L son los dos únicos nodos meta y A es el nodo inicial, se indica el orden en que los nodos son visitados siguiendo el método de búsqueda en amplitud.

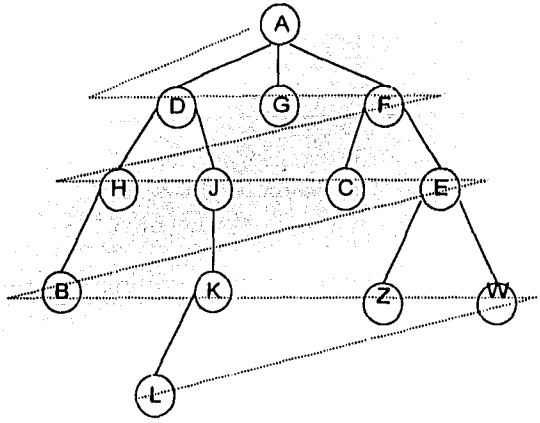


Figura 8. Ejemplo de Búsqueda en amplitud.

El orden de los nodos es: A, D, G, F, H, J, C, E, **B**, K, Z, W, **L**

Búsqueda en profundidad

En este método, al iniciar se le aplica al nodo raíz cualquier posible regla para generar un nodo del siguiente nivel:

1. A este nodo se le aplica cualquier regla posible para generar un nodo del siguiente nivel.
2. Este proceso continua hasta que ya no sea posible aplicar otra regla, obteniendo un nodo "no solución" o un nodo solución.
3. Para generar los siguientes nodos, nos regresamos al primer nodo en el cual sea posible aplicar otra regla.
4. El proceso continua hasta generar todas las soluciones.

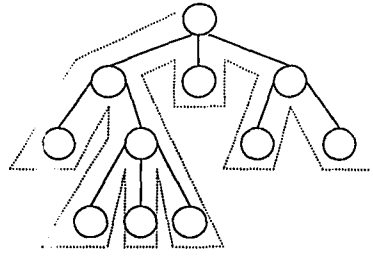


Figura 9. Búsqueda en profundidad.

TESIS CON
FALLA DE ORIGEN

Ejemplo: Dado el árbol siguiente, donde B y L son los dos únicos nodos meta y A es el nodo inicial, se indica el orden en que los nodos son visitados siguiendo el método de búsqueda en profundidad.

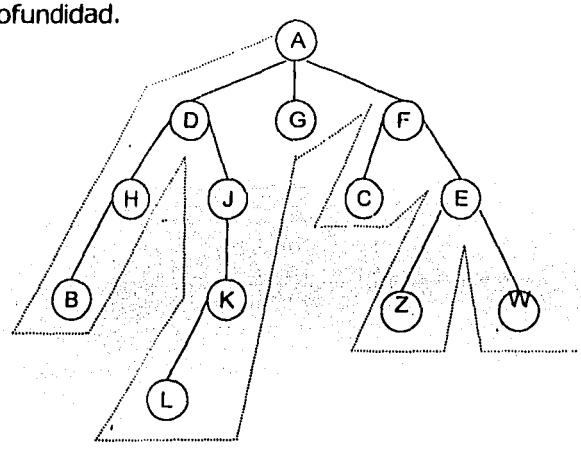


Figura 10. Ejemplo de Búsqueda en profundidad.

El orden de los nodos visitados es: A, D, H, **B**, J, K, **L**, G, F, C, E, Z, W

Métodos ciegos parciales

Se basan en los métodos ciegos exhaustivos, existiendo así dos principales: primero en amplitud y primero en profundidad.

Primero en amplitud (Breadth-First Search)

En este método se aplica el proceso de búsqueda en amplitud hasta obtener una "primera" solución:

1. A cada nodo (comenzando por el nodo raíz) se le aplican todas las reglas (operadores) para generar los nodos del siguiente nivel.
2. Una vez generados, se comprueba nodo por nodo si alguno de ellos es solución del problema. De ser así, el proceso termina.
3. En caso contrario se generan los nodos del siguiente nivel.

Este tipo de búsqueda visita cada nodo del árbol por niveles, es decir, visita todos los nodos de un nivel antes de visitar los del siguiente.
El algoritmo para este método es:

TESIS CON
FALLA DE ORIGEN

```
Lista_nodos = [estado_inicial];
Mientras Not Vacia(lista_nodos)
    estado_actual = lista_nodos.primero;
    Si EstadoFinal(estado_actual) entonces
            Terminar;
        Sino
            lista_reglas = ReglasAplicables (estado_actual);
            Mientras NOT Vacia(lista_reglas)
                    estado_sucesor = AplicarRegla (lista_reglas);
                    lista_nodos = lista_nodos + [estado_sucesor];
            Fin Mientras;
        Fin Sino;
Fin Mientras;
```

Con la búsqueda en amplitud se asegura que una vez alcanzada una solución no existe otra ruta hacia la solución que tenga menor cantidad de pasos. Una desventaja de este algoritmo es que para alcanzar una solución de n pasos, debe haber explorado todo el espacio de estados hasta ese nivel.

Ejemplo: En el siguiente árbol, encontrar una ruta cuyo valor sea mayor o igual a 50 mediante el método de primero en amplitud.

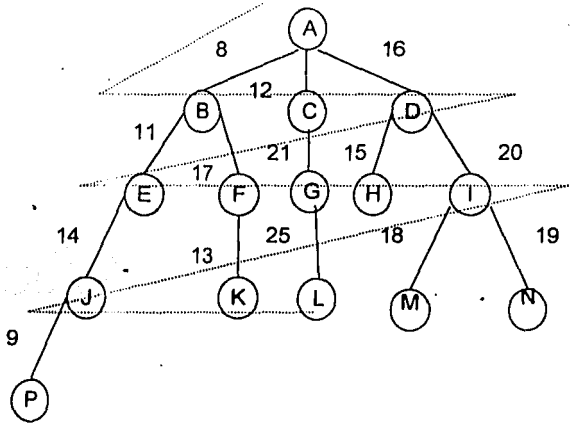


Figura 11. Ejemplo de Primero en amplitud.

Los nodos generados por el método son: A, B, C, D, E ,F, G, H, I, J, K, L (total=12)
Y la ruta elegida es: A, C, G ,L (12 + 21 + 25 = 58)

TESIS CON
FALLA DE ORIGEN

Primero en profundidad (Depth-First Search)

Se aplica el método de búsqueda en profundidad hasta obtener una "primera" solución:

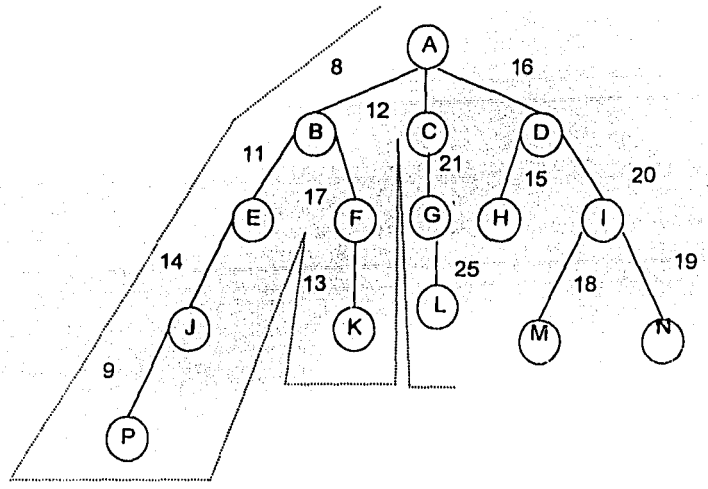
1. A un nodo (comenzando por el nodo raíz) se le aplica una regla para generar un nodo del siguiente nivel.
2. Una vez generado, se comprueba si dicho nodo es una solución, en caso afirmativo, el proceso termina.
3. Si el nodo no es solución, entonces se genera el siguiente nodo, repitiendo el proceso hasta obtener la primera solución.

Este algoritmo de búsqueda continúa por una rama del árbol hasta encontrar la solución o decidir terminar la búsqueda por esa dirección (por llegar al estado final, por tener un largo de ruta que supera una cota máxima determina, por haber llegado a un estado ya visitado, etc.). Al fracasar una ruta, se realiza un backtracking o vuelta atrás, continuando la exploración en el paso inmediatamente anterior.

```
Función Buscar (estado_actual) devuelve Boolean
Comienzo
    Si EstadoFinal(estado_actual) entonces
            Devolver TRUE;
    Sino
            exito = FALSE;
            lista_reglas = ReglasAplicables (estado_actual);
            Mientras NOT exito AND NOT Vacia(lista_reglas)
                    estado_sucesor = AplicarRegla (lista_reglas);
                    exito = Buscar (estado_sucesor);
            Fin Mientras;
            Devolver exito;
        Fin Sino;
Fin Buscar;
```

Con la búsqueda en profundidad no es necesario tener almacenado todo el espacio de estados, sino sólo el camino que se está explorando. Puede encontrar la solución sin tener que explorar gran parte del espacio de estados. Como desventajas de este algoritmo se señala que puede seguir una ruta infructuosa durante muchos pasos, y además la primera solución que encuentra puede distar mucho de ser la solución de mínima cantidad de pasos.

Ejemplo: En el siguiente árbol, encontrar una ruta cuyo valor sea mayor o igual a 50 mediante el método de primero en profundidad.



TESIS CON
FALLA DE ORIGEN

Figura 12. Ejemplo de Primero en profundidad.

Los nodos generados por el método son: A, B, E, J, P, F, K, C, G, **L** (total=10)
Y la ruta elegida es: A, C, G ,L (12 + 21 + 25 = 58)

De acuerdo al ejemplo anterior, la búsqueda de primero en profundidad resultó menos costosa ya que sólo generó un total de 10 nodos, mientras que la búsqueda de primero en amplitud generó 12, a pesar de que ambas obtuvieron el mismo resultado, 58.

Representación reducida del problema

Métodos heurísticos

Además de las búsquedas ciegas, también existen búsquedas heurísticas (basadas en la experiencia).

Los métodos heurísticos se han diseñado para reducir el volumen de búsqueda para una solución. Cuando un problema está representado por un árbol de búsqueda, las búsquedas heurísticas tienden a reducir el tamaño del árbol al no utilizar nodos no vitales o trayectorias no prometedoras.

El objetivo es guiar al proceso de búsqueda en la dirección más provechosa sugiriendo el camino a seguir cuando hay más de una opción.

Ventajas:

- Tienen una gran flexibilidad, lo cual permite utilizarlos en problemas complejos o mal estructurados.
- Aunque un procedimiento de solución exacta puede existir, ésta puede ser algorítmicamente prohibitiva o difícil de presentar computacionalmente. En tales casos, los métodos heurísticos son mejores.
- El método heurístico suele ser fácil de entender para el tomador de decisiones, especialmente cuando está compuesto por un análisis cuantitativo y entonces las posibilidades de implementar la solución son altas.
- Un método heurístico puede usarse como parte de un procedimiento iterativo que garantice el encontrar una solución óptima.

Desventajas:

- La gran flexibilidad de los métodos heurísticos puede conducir a soluciones erróneas o fraudulentas.
- Ciertas heurísticas pueden contraponerse a otras que pudieran ser aplicadas a un mismo problema, lo cual genera confusión y desconfianza en dichos métodos.
- Las soluciones óptimas regularmente no son alcanzadas. Las diferencias entre la solución óptima y la generada heurísticamente pueden ser grandes, y con esto, las pérdidas potenciales pueden rebasar los beneficios.
- Las heurísticas no son algoritmos generales ya que por lo general son utilizadas para situaciones muy específicas. Por esta razón, es necesario elaborar programas especiales de computación para cada heurística.
- Las heurísticas pueden sacrificar la completitud, es decir, pueden pasar por alto una buena solución.

Las heurísticas pueden ser:

- Generales: son adecuadas para una amplia variedad de dominios. Por ejemplo la heurística del vecino más próximo (nearest neighbor) se aplica a muchos problemas combinatorios.
- De propósito especial: explotan el conocimiento específico de un dominio para resolver problemas particulares.

Las heurísticas se pueden incorporar a un proceso de búsqueda basado en reglas de dos maneras:

- Dentro de las mismas reglas. Por ejemplo en el ajedrez, las reglas pueden describir, además de los movimientos legales, también las buenas jugadas.
- Como una función que evalúa estados determinando su grado de "deseable". Esta función evalúa aspectos del problema dando pesos a aspectos individuales, de manera que el valor que devuelve es una estimación de que el nodo pertenece a la ruta que conduce a la mejor solución. Por ejemplo en el ajedrez

se puede tener una función a maximizar la cual devuelve el número de piezas de ventaja.

En conclusión, las heurísticas representan el conocimiento general y específico del mundo, que hace que sea abordable solucionar problemas complejos.

Uno de los métodos heurísticos más utilizados es el Método del Gradiente.

Método del gradiente

También llamado búsqueda en escalada o hill climbing. En este tipo de búsqueda, a cada nodo se le asocia un valor que nos dará idea de lo cerca que se encuentra el nodo meta. Dicho valor no será más que una estimación de la distancia real a la meta.
La incorporación de este tipo de información es normalmente a través de las llamadas "funciones de evaluación heurística", las cuales guiarán la búsqueda hacia aquellos caminos que se supone son más prometedores.
El método consiste en seguir el recorrido a través de aquellos nodos que nos conduzcan al objetivo deseado. No existe la posibilidad de retomar caminos abandonados, es decir, no se pueden reconsiderar las decisiones tomadas.
Para que el algoritmo pueda ser aplicado, la función heurística debe cumplir la condición de que para todo nodo visitado debe haber al menos un sucesor con un valor "más óptimo".

Ejemplo: En el siguiente árbol, encontrar una ruta cuyo valor sea mayor o igual a 50 mediante el método del gradiente.

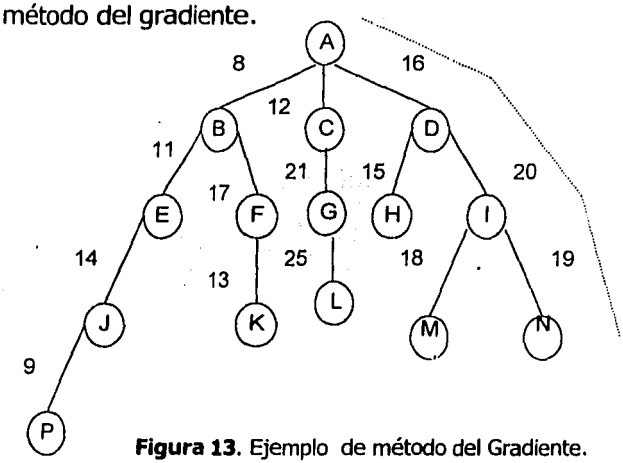


Figura 13. Ejemplo de método del Gradiente.

TESIS CON
FALLA DE ORIGEN

Los nodos generados y la ruta elegida por el método son:
A, D, I, N (16 + 20 + 19= 55)

En comparación con los métodos anteriores, éste último resulto "mejor" al generar sólo 4 nodos, aunque el valor obtenido fue menor (55, en comparación con 58).

Representación en forma de procedimientos de rutina

Muchos de los problemas de IA son demasiado complejos para ser resueltos mediante técnicas directas; es mejor intentar resolverlos por medio de métodos de búsqueda apropiados usando cualesquiera de las técnicas directas que estén a nuestra disposición para guiar la búsqueda, tales como: Redes Semánticas, Frames (Armazones) y Scripts (Guiones).

Redes Semánticas

Las redes semánticas son estructuras gráficas que codifican el conocimiento taxonómico de los objetos y sus propiedades.

Los nodos representan conceptos (objetos o sucesos).
Los arcos significan relaciones entre esos conceptos.

Existen dos tipos de nodos:
- Nodos etiquetados por constantes correspondientes ya sea a sus categorías taxonómicas o a sus propiedades.
- Nodos etiquetados por objetos constantes correspondientes a los objetos en el dominio.

Hay tres tipos de arcos conectando los nodos:
- Arcos de subcategoría.
- Arcos miembros.
- Arcos de función.

El objetivo inicial para el desarrollo de las redes semánticas fue el entender el lenguaje natural, más que la clasificación de datos. Otra característica de las redes semánticas es que existen tantas como las necesidades que han tenido diferentes investigadores en diferentes proyectos.

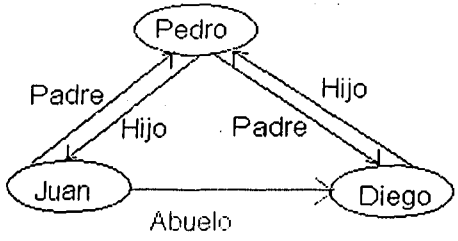
Se puede decir entonces que cualquier grafo en el cual los nodos se conecten por medio de arcos se le puede llamar red semántica, siempre que nodos y arcos estén etiquetados.

Para tener semántica en un grafo, se necesita definir cuidadosamente el significado de nodos y arcos, y cómo son usados.

Características de las Redes Semánticas.

a. Se introduce el concepto de *distancia semántica*, cantidad de arcos que separan un nodo de otro. En las redes semánticas la distancia puede ser importante, y es usada para localizar objetos poco o muy relacionados, dependiendo de la distancia. En algunos casos se puede disminuir la distancia agregando arcos con ese propósito.

Ejemplo:



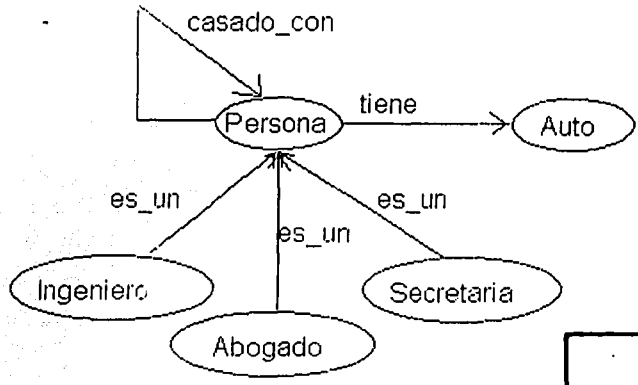
En este ejemplo se agregó un arco denominado *Abuelo*, para reducir la distancia entre *Juan* y *Diego*

b. En las redes semánticas también se tiene la idea de *partición*: es el contexto de una red, en el sentido de tener una *subred*, así para una tarea o trabajo específico sólo una parte de la red está disponible.

Esta facilidad resulta útil en el momento de realizar búsquedas, ya que se limita el espacio de búsqueda.

c. También se tiene la jerarquía de tipo (u objeto). Los tipos de jerarquías que se tienen en una red semántica son PARTE-DE y ES-UN. La existencia de una jerarquía implica que se permite la *herencia* donde un objeto que pertenece a una clase hereda todas las propiedades de la clase.

Ejemplo:



TESIS CON
FALLA DE ORIGEN

La herencia no sólo se refiere a la herencia de atributos y sus valores, sino que también se heredan los tipos de relaciones permitidas para esa clase, esto es, su comportamiento.

Esta estructura jerárquica permite a los nodos de niveles inferiores heredar propiedades de los nodos de niveles superiores.

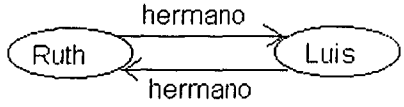
Descripción Formal de Red Semántica.

Estructuras.

La estructura de cualquier red semántica consiste de un grafo (Red). La forma en que se distingue entre las diferentes redes y arcos determina el tipo de red semántica, y así mismo el modelamiento de datos para el cual fue creada.

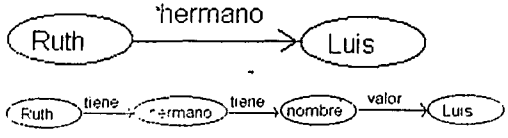
Se puede tener un modelo de datos en que los nodos representan cosas (instancias o valores de entidades) y los arcos pueden representar relaciones entre los nodos.

Ejemplo:



En lo antes dicho se tiene que una unidad de información puede ser considerada ya sea como una cosa (nodo) o hecho (arco). Aquí se procura cierta libertad de interpretación, respecto a qué son arcos y qué son nodos. Una forma de decidir si una unidad es cosa (nodo) o hecho (arco) es preguntarse si más adelante tiene que relacionarse con otra cosa (nodo). Si es así, entonces es nodo.

Ejemplo:



TESIS CON
FALLA DE ORIGEN

Los nodos pueden ser caracterizados de acuerdo a las cosas que representan. Una de estas caracterizaciones es la que establece que los nodos representan *conceptos, eventos, características* y *valores*.

a. Conceptos.

Son constantes para la realidad que se desea representar, y son utilizados para especificar valores (Ejemplo. Juan, la persona Juan).

b. Eventos.

Corresponden a acciones que ocurren en la realidad que se modela. Así, "Juan golpea a Pedro" se puede representar mediante un nodo "golpear" y dos nodos conceptos "Juan" y "Pedro".

Los arcos que conectan nodos eventos y nodos concepto corresponden a los roles que los conceptos juegan en el evento. En el ejemplo, "Juan" es el agente (el que produce el golpe) y "Pedro" el objetivo (el que recibe el golpe).

c. Características.

Son nodos que describen propiedades de un concepto. Corresponden a los atributos de un concepto. Ejemplo: el "peso" de Juan.

d. Valores.

Son nodos correspondientes a dominios de valores. Ejemplo: el peso de Juan es "48 Kg.". Corresponden a los valores que pueden tomar las características.

Ejemplo:

a. Juan golpea a Pedro.

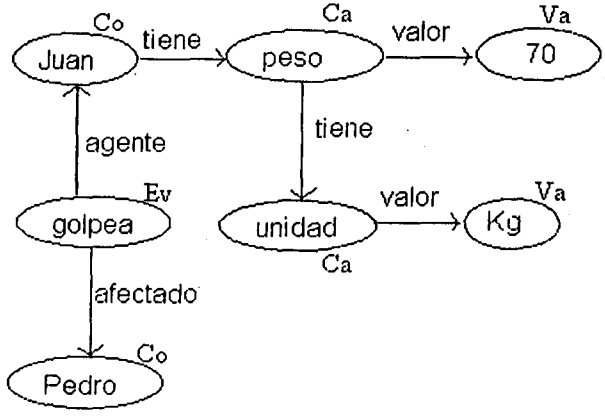
b. Juan pesa 70 Kg.

Conceptos (Co): Juan, Pedro

Eventos (Ev): golpea

Características (Ca): peso, valor

Valores (Va): 70, Kg.



Complementariamente se agrega la clasificación por CLASES. Así, Juan y Pedro son conceptos que pertenecen a la clase Persona, que es a su vez otro concepto. Para lograr esta pertenencia a clases se usa el arco ES_UN.

Además se tiene el arco PARTE_DE, que permite generar estructuras de composición.

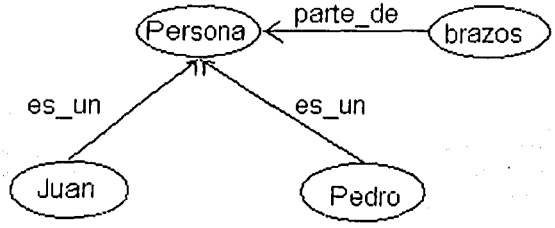
Ejemplo:

a. Pedro y Juan son personas.

b. Las personas tienen brazos.

TESIS CON
FALLA DE ORIGEN

Conceptos: Persona, Pedro, Juan, brazos.



Frames (Armazones)

Los armazones son mecanismos generales diseñados para la representación por computadora del conocimiento.

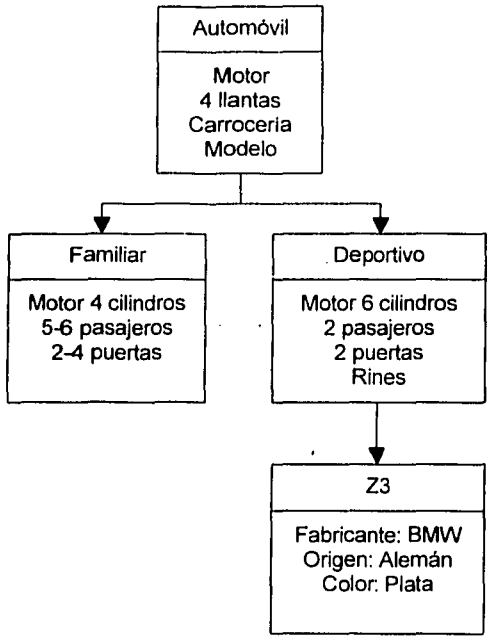
En cierto sentido, estas estructuras pueden verse usualmente como redes semánticas complejas, pero suelen tener gran parte de su estructura interna diseñada para hacerlas útiles en clases específicas de tareas de resolución de problemas.

También puede definirse como una estructura de datos que contiene todo el conocimiento acerca de un objeto.

En su forma más simple provee los medios para describir objetos de forma más completa por medio del almacenamiento y atributos particulares así como los valores de default.

TESIS CON
FALLA DE ORIGEN

Ejemplo: Frame de un Automóvil.



TESIS CON
FALLA DE ORIGEN

Nota.
Una de las características de los frames es que un frame puede identificar a toda una familia de objetos.

Definición. Consiste en una colección de ranuras que describen el aspecto de los objetos. Estas ranuras se llenan mediante otras ranuras que describen el aspecto de los objetos.

Asociado a cada ranura puede haber un conjunto de condiciones que deben cumplir la entidades que vayan a llenarla.

Cada ranura puede también llenarse con un valor por defecto, de forma que, en ausencia de información especifica sobre lo contrario, pueda suponerse lo que las cosas son usualmente.

En efecto, cada nodo correspondiente a un objeto o a una clase se convierte en un marco, que consta de una primera línea con el nombre del marco y una sucesión de líneas, llamadas "ranuras" (slots) con la sintaxis:

<ranura> ::= <nombre de relación>: <objeto relacionado> /
<nombre de relación>: <clase relacionada> /
<nombre de propiedad>: <valor de la propiedad> /
<nombre de propiedad>: (excep) <valor de la propiedad> /
<nombre de propiedad>: if_needed <procedimiento> /
<nombre de propiedad>: if_added <procedimiento>
<nombre de relación> ::= es_un / tipo_de / <relación específica de la aplicación>

Tipos de SLOTS en los Frames.

1. Hay SLOT para describir conocimiento *Declarativo* como lo es: peso de un objeto, la altura, la forma, el color, hobbies (leer, nadar, trotar)

2. Hay Slots para describir conocimiento *Procedural*, el cual se refiere a pequeños procesos que identifica a ciertas funciones. (Ejemplo Velocidad, aceleración, trabajo, calculo de energía, etc.).

3. Que identifican a Reglas.
   Ejemplo If maquina caliente THEN prender ventilador.

4. Un Slot puede indicar la conexión con otro FRAME.

5. Un Slot puede indicar la conexión con otros frames de representación de conocimiento como lo son con las Redes Semánticas.

Sistema de armazones.

Los armazones relacionados pueden agruparse juntos para formar un sistema de armazones.

Ejemplo.
Armazón que representa una vista de un cubo, tal como se ha indicado mediante el enlace ESUN desde el nivel superior del nodo a CUBO. En el siguiente nivel, cada nodo describe una cara del cubo. Los enlaces DEBE-SER que van de esos nodos a PARALELOGRAMO describen una restricción en los valores que pueden llenar la ranura representada por el nodo. Los enlaces que salen de estos nodos para especificar los nodos A, E y B, indican a los llenadores qué ranuras hay en una vista concreta.

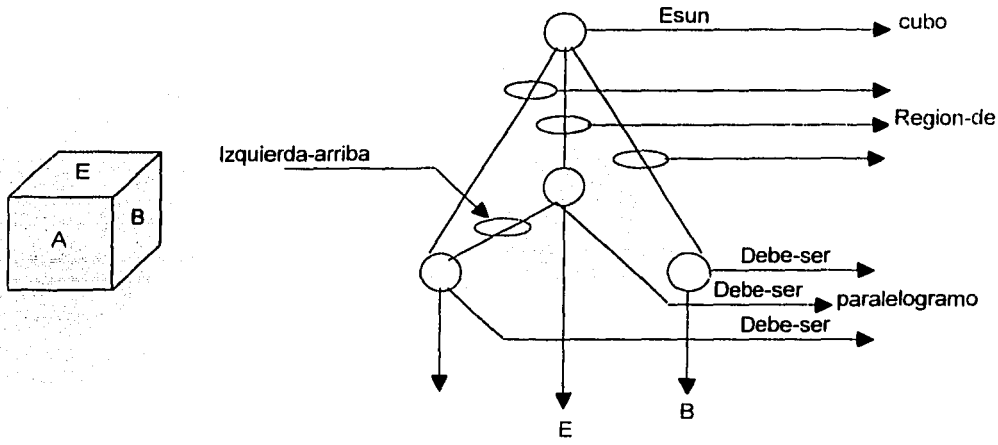


Figura 14. Ejemplo de un sistema de armazones que representa un paralelegramo.

Para muchas aplicaciones es importante poder considerar un objeto desde varios puntos de vista. Para ello se necesita un sistema de armazones. Un ejemplo de un sistema de armazones sencillo, es el siguiente:

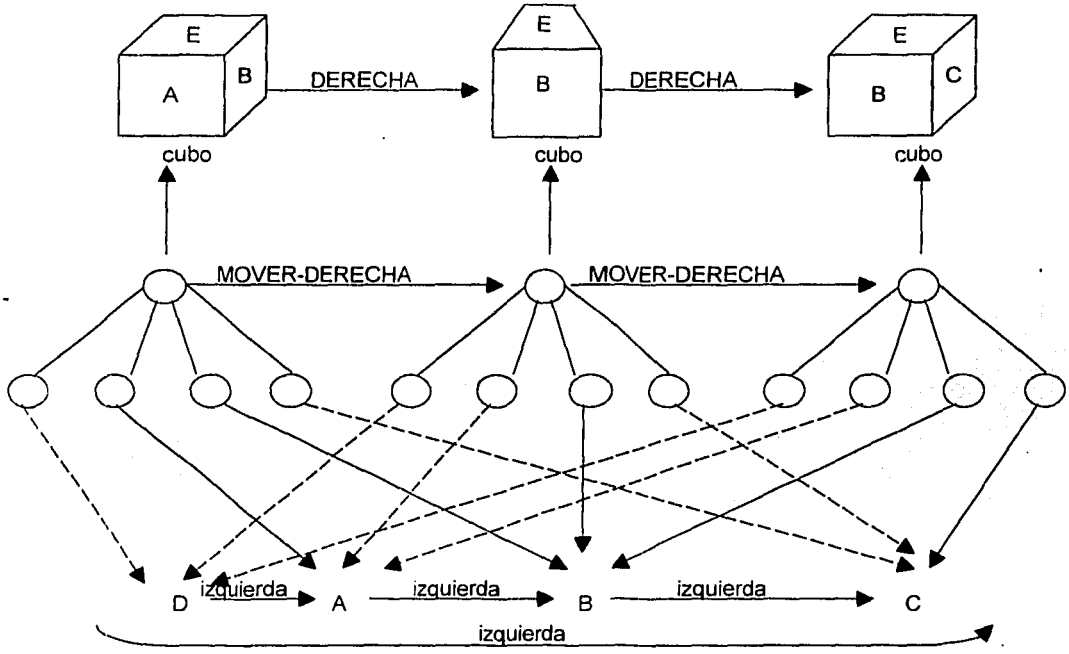


Figura 15. Ejemplo de un sistema de armazones que representa 3 distintas vistas de un cubo.

TESIS CON
FALLA DE ORIGEN

Consiste en tres armazones, cada uno representado una perspectiva de un cubo. Algunos de los detalles se han omitido en la figura 15 para mayor claridad, pero, naturalmente, aun están presentes en la estructura. Las líneas sólidas que parten de las ranuras que representan caras hacia las caras concretas que llenan esas ranuras indican caras visibles. Las líneas discontinuas indican caras que son invisibles desde ese punto de vista. Los enlaces entre armazones describen la relación entre las vistas representadas por dichos armazones.

Además de estos enlaces explícitos entre los armazones mismos, los armazones están conectados a través de los nodos compartidos que llenan sus ranuras. Estos nodos compartidos, representando cada uno una cara del cubo, representan descripciones de las caras desde puntos de vista independientes. Pero si las caras contienen modelos complejos, esos modelos se representarían por estructuras más complicadas, posiblemente otro conjunto de armazones.

Debidó a que cada ranura de un armazón puede llenarse con otra estructura de armazón, y puesto que una estructura dada puede llenar más de una ranura, no es necesario que haya almacenamiento redundante de la información común a muchos puntos de vista.

Razonamiento con el Conocimiento.

Los armazones son útiles por cuanto facilitan inferir hechos aún no observados sobre situaciones nuevas.
Esto lo facilita de diversas maneras:

- Contienen información.
- Contienen atributos.
- Describen ocurrencias típicas.

El proceso de ejemplificar un armazón en una situación concreta no suele proceder de forma gradual. Sin embargo, cuando el proceso tropieza con un obstáculo no suele ser necesario abandonar el esfuerzo y empezar de nuevo. En ese caso, pueden hacerse diversas cosas:

- Seleccionar los fragmentos y emparejarlos.
- Dar una excusa para el fallo.
- Referirse a enlaces entre armazones.
- Atravesar hacia arriba la estructura jerárquica.

Los armazones, al igual que las redes semánticas, son estructuras de propósito general en las que pueden incrustarse conjuntos concretos específicos del dominio.

Script (Guiones)

A continuación discutiremos una estructura de propósito más especial, la dependencia conceptual, en donde se definían primitivas específicas para usarse al construir representaciones individuales, donde podían surgir relaciones específicas entre los elementos de una representación. Estas construcciones específicas estaban hechas a la medida del problema de representar acontecimientos. Introducimos a continuación los armazones como una estructura de propósito general para representar grupos comunes de hechos.

Pero, al igual que en la representación de hechos individuales, los grupos de hechos pueden tener estructuras útiles de propósito especial que exploten propiedades específicas de su dominio restringido.

Tal estructura de propósito especial es el GUION.

Representación del conocimiento

Este tipo de Representación de conocimiento es utilizado para representar secuencias de EVENTOS de tipo:

- Historias.
- Dramas.
- Visitas al: Doctor, Restaurante, Estética.
- Viajes.

En los SCRIPTS o guiones intervienen elementos básicos como lo son:

1. Requerimientos de Entrada.
2. Roles o papeles de las personas o cosas que intervienen en el libreto.
3. Herramientas utilizadas por los diferentes participantes o actores.
4. Escenas.
5. Resultados.

Ejemplo: SCRIPT de una visita a un Restaurante de Servicio Rápido.

- Tipo de Restaurante:
  -Servicio Rápido.

- Requerimientos de Entrada:
  - Hay un cliente que tiene hambre.
  - El cliente tiene dinero.

- Herramientas que intervienen:
  - carro
  - dinero
  - bandeja
  - bote de basura
  - alimentos
  - catsup
  - servilletas
  - mostrador
  - palillos
  - sal
  - cuchillo
  - tenedor
  - cuchara
  - mesa

- Roles o Papeles:
  - Cliente.
  - Recepcionista .

- Escena 1: Entrada al Restaurante:
  - El cliente para el carro.
  - El cliente entra al Restaurante.
  - El cliente hace fila en el mostrador.
  - El cliente mira los distintos alimentos en la pared y decide cuales seleccionar.

- Escena 2: Ordenar:
  - El cliente le da la orden al recepcionista.
  - El recepcionista toma la orden y comienza a poner los alimentos a la bandeja.
  - El recepcionista entrega los alimentos e indica el total de la orden.
  - El cliente paga la orden.
  - El recepcionista recibe el dinero y da el cambio (si es necesario).

- Escena 3: Comer:
  - El cliente toma servilletas, cuchillo, tenedor, cuchara, la sal, palillos y catsup.
  - El cliente busca mesa.
  - El cliente consume los alimentos.

- Escena 4: Salida:
  - Recoge la basura de la mesa.
  - Vacía el contenido de la vasija en el bote de la basura.
  - Sale del restaurante.
  - Toma su carro.
  - Se va.

- Resultados:
  - El cliente ya no tiene hambre.
  - El cliente tiene menos dinero.
  - El cliente esta satisfecho / el cliente no esta satisfecho.
  - El cliente tiene / no tiene dolor de estomago.
  - El recepcionista realizó su trabajo.

CAPÍTULO V.
USO Y MODIFICACIÓN DEL CONOCIMIENTO.

Programa que resuelva problemas generales (GPS)

Fue en los años 60 cuando Alan Newell y Herbert Simon, que trabajando la demostración de teoremas y el ajedrez por computadora logran crear un programa llamado GPS (General Problem Solver: solucionador general de problemas). Este era un sistema en el que el usuario definía un entorno en función de una serie de objetos y los operadores que se podían aplicar sobre ellos. Este programa era capaz de trabajar con las torres de Hanoi [4], así como con criptoaritmética [5] y otros problemas similares, operando, claro está, con microcosmos formalizados que representaban los parámetros dentro de los cuales se podían resolver problemas. Lo que no podía hacer el GPS era resolver problemas ni del mundo real, ni médicos ni tomar decisiones importantes. El GPS manejaba reglas heurísticas (aprender a partir de sus propios descubrimientos) que la conducían hasta el destino deseado mediante el método del ensayo y el error.

Se trataba de un programa, al que se le podían ofrecer pequeños problemas, y éste deberá describir todos los pasos que realiza hasta conseguir su fin, es decir, completar el problema positivamente.

4 El juego de las torres de Hanoi, o torres de diamante, es un juego oriental muy antiguo que hoy se conoce en todo el mundo. Consta de tres columnas y una serie de discos de distintos tamaños. Los discos están acomodados de mayor a menor en una de las columnas. El juego consiste en pasar todos los discos a otra de las columnas y dejarlos acomodados como estaban: de mayor a menor. Las reglas del juego son las siguientes: 1.- Sólo se puede mover un disco cada vez. 2.- Para cambiar los discos de lugar se pueden usar las tres columnas del juego; es decir que los distintos discos se pueden ir acomodando en las columnas según convenga. 3.- Nunca deberá quedar un disco grande sobre un disco chico.

5 La CriptoAritmética es un método que consiste en sustituir cada letra por un dígito, de modo que las cuentas (operaciones) sean correctas. A igual letra, igual dígito, y a distinta letra, distinto dígito. Como es habitual, los números no pueden tener ceros a izquierda. Un ejemplo de este tipo de problemas es:

SEND	Solución:	9 5 6 7
+MORE		1 0 8 5
MONEY		1 0 6 5 2

Se puso en práctica el análisis medios-fines, un principio de la retroalimentación de Wiener [6] llevado a un mayor nivel de abstracción. El análisis medios-fines consiste en detectar las diferencias entre un objetivo deseado y la situación actual y reducir después esas diferencias. También se aplicó por primera vez el Backtracking para probar si funciona y si no, volver atrás y probar otra cosa, que se convirtió desde aquel momento en una herramienta básica de la IA.

Las heurísticas.

GPS es un analizador de medios-fines: descrito el objetivo y la posición inicial, y dado un listado de los medios disponibles, GPS hace los ensayos necesarios y encontrar la manera de usar los medios hasta alcanzar el objetivo.
El método de ciego ensayo-error es un desesperado último recurso, más que una manera inteligente de resolver un problema.

GPS no es completamente ciego al aplicar el método de ensayo-error:

- No considera movimientos que le devuelven a la posición de partida
- Abandona cualquier cadena de movimientos que supera una longitud predefinida
- Tiene incorporadas instrucciones que le indican qué movimientos debe intentar primero cuando hay varios posibles: las heurísticas.

Un ejemplo.

Suponemos que GPS tiene que resolver los movimientos que un robot tiene que realizar para mover unos bloques de un lugar a otro. La lista de los medios disponibles para el robot contiene las operaciones de moverse, cogerunbloque, dejarunbloque, llevarunbloque, etc. Supongamos que el robot tiene en su brazo el bloque X y que tiene que coger el bloque Y. Antes de que cogerunbloque pueda emplearse sobre Y, el robot debe hacer dos cosas: dejar libre su brazo y moverse hasta donde esté Y.
¿Qué medio debe aplicar primero el robot: moverse o dejarunbloque? El programa puede contener una instrucción que manda que dejarunbloque debe intentarse primero, dado que si el robot tiene su brazo libre puede coger y apartar de su camino bloques que pueda encontrarse moviéndose hasta Y. Sin embargo, el intento de dejarunbloque debe abandonarse cuando dejar el bloque X obstruya el camino hacia Y. En estos casos, moverse debe intentarse primero.

6 Norbert Wiener fue uno de los primeros americanos en hacer observaciones en el principio de la teoría de la retroalimentación. El ejemplo más familiar de la teoría de la retroalimentación es el termostato: que controla la temperatura de un ambiente reuniendo la actual temperatura de la casa, comparándola con la temperatura deseada, y respondiendo mediante subida o bajada de temperatura. Muy importante en su investigación dentro del circuito de retroalimentación fue que Wiener teorizó que todos los comportamientos inteligentes son resultado de mecanismos de retroalimentación. Este descubrimiento influenció mucho el desarrollo temprano de la IA.

Los problemas de las heurísticas.

La tarea de construir rangos para las aplicaciones de los medios se puede convertir en un imposible si la lista de los medios disponibles es grande.
Las heurísticas son programadas en GPS: ¿quién resuelve entonces los problemas, la computadora o el programador?

La representación.

GPS puede solucionar únicamente problemas de la forma: ¿cómo alcanzar un cierto objetivo desde una determinada posición inicial?.

Un ejemplo.

Suponemos el siguiente problema: un hombre deja una cabaña en la cima de una montaña al mediodía y baja a otra cabaña en la base del monte. Al día siguiente, una mujer deja la cabaña, de nuevo al mediodía, de la base y sube hasta la cabaña de la cima. La cuestión es, ¿Hay algún momento, una hora x, tal que a la hora x en la segunda tarde, la mujer está exactamente en el mismo punto del camino que el hombre a la hora x de la primera tarde?
No hay forma natural de representar este problema en la forma: partiendo de la posición inicial utilizar los medios de la lista hasta alcanzar el objetivo. En este caso, ¿cuál es el objetivo?, ¿cuál es la posición inicial?, ¿cuáles son los medios?.

Los problemas de la representación.

Con frecuencia, la parte más difícil al resolver un problema es encontrar la manera adecuada de representarlo. Una vez encontrada la representación correcta, dar con la solución es sencillo. Por ejemplo, en el caso de los montañeros basta con considerar que las personas están bajando y subiendo el camino la misma tarde.

La filosofía de GPS era que las técnicas para solucionar un problema pueden separarse del conocimiento específico sobre el problema. GPS es general en el sentido de que no contiene ningún conocimiento específico sobre el problema.
La lección más importante que se puede sacar del fracaso de GPS es que los programas que no saben mucho no pueden hacer mucho.

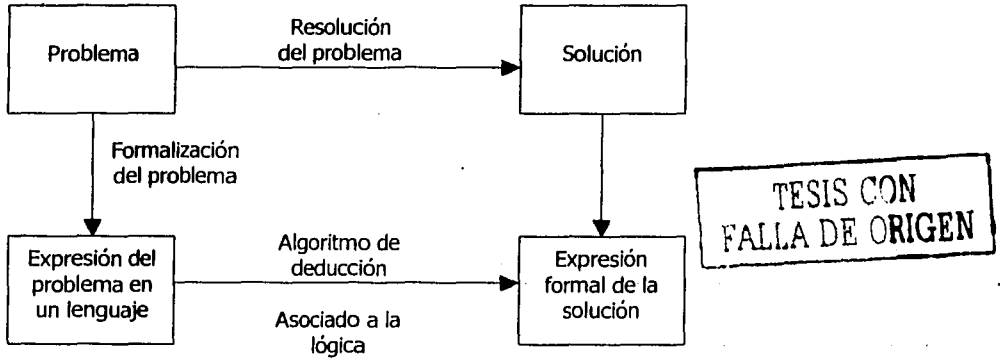
Los principales métodos del GPS conjuntan la heurística del análisis de medios y fines. Este tipo de análisis clasifica las cosas de acuerdo con su función, sus fines y medios para analizarlas.

Desventaja.
Aunque el análisis de medios y fines es útil, no define qué hay que hacer cuando existen varias acciones mediante las que se puede obtener lo mismo, o cuando no hay una acción que permita obtener lo que se desea.

El GPS se diseñó para que imitara los protocolos de resolución de problemas de los humanos. Dentro del reducido número de acertijos que podía manejar, la secuencia en la que el programa considera los subobjetivos y los posibles cursos de acción es semejante a la manera en la que los seres humanos abordan los mismos problemas, es decir, el GPS fue el primer programa que incorporó el enfoque del pensar como humano.

Deducción automática

Cuando utilizamos la lógica en inteligencia artificial, surge la necesidad de disponer de un algoritmo que pueda realizar automáticamente la deducción, es decir, resolver mecánicamente los problemas presentados en la lógica. Podemos representar lo anterior de la siguiente manera:



TESIS CON FALLA DE ORIGEN

Figura 16. Deducción automática.

Por problema entendemos toda situación que necesita una cierta inteligencia para comprenderla. En este proceso los aspectos lenguaje (para la expresión) y sistema deductivo (para la resolución) de la lógica juegan un papel determinante. Las soluciones del problema (la respuesta a la pregunta) se obtiene utilizando las propiedades del sistema deductivo.

La capacidad expresiva de la lógica puede determinarse a partir del tipo de preguntas que se pueden resolver, por ejemplo:

1. Preguntas que admiten una respuesta de tipo sí o no.
2. Preguntas cuya respuesta es una serie de acciones, eventualmente condicionales.
3. Preguntas que admiten un conjunto como respuesta.
4. Preguntas cuya respuesta es una explicación.

Aprendizaje e inferencia deductivo

La mecánica inferencial.

La llamada mecánica inferencial cubre un conjunto de nociones -deducción, inferencia, planificación, búsqueda, razonamiento (del sentido común), prueba, etc.- que, cual funciones, se aplican a la base de conocimientos (como argumento) generando nuevo conocimiento.

El motor de inferencia (tratado con detalle en el Capítulo III, Sistemas Expertos) alimentado por la base de conocimientos, construye dinámicamente el razonamiento, decidiendo qué reglas se activan y en qué orden.

Los mecanismos de razonamiento que utilizan reglas de producción son los dos básicos de la lógica formal: modus ponendo ponens y modus tollendo tollens.

Modus ponendo ponens: Afirmado el antecedente, se afirma el consecuente.

(A ^ (A -> B)) -> B

Modus tollendo tollens: Negado el consecuente, se niega el antecedente.

(´B ^ (A -> B)) -> ´A

Sea cual sea el modo de razonamiento utilizado, el ciclo de base del motor de inferencia comprende cuatro fases:

1. Fase de selección. Selección de un subconjunto de la base de hechos y de la base de reglas.
2. Fase de filtrado. El motor de inferencia compara la parte premisa de las reglas seleccionadas con los hechos de la base de hechos para determinar el conjunto de reglas aplicables.
3. Fase de resolución de conflictos o elección. El resultado de esta fase es la elección de la regla que efectivamente se va a aplicar. Esta fase expresa una estrategia que puede ser muy simple y sin relación con el contexto (la primera regla de la lista, la menos compleja, la menos utilizada), o compleja y que tiene en cuenta el contexto (la más prometedora, la más fiable, la menos costosa).
4. Fase de ejecución. Se ocupa de aplicar la regla elegida y consiste en añadir uno o varios hechos a la base de hechos.

El fin de este ciclo depende del modo de razonamiento utilizado.

Encadenamiento hacia adelante o dirigido por los datos

La llamada "técnica de encadenamiento hacia adelante" consiste en aplicar a la base de conocimientos (organizado en forma de reglas de producción), junto con otro conocimiento disponible, el esquema inferencial modus ponens. Esta estrategia se denomina "encadenamiento hacia adelante" o "razonamiento de datos dirigidos", porque comienza con los datos conocidos y aplica el modus ponens sucesivamente hasta obtener los resultados que se siguen. Las reglas se aplican "en paralelo", i.e., en cualquier iteración una regla toma los datos cuales eran al principio del ciclo, por lo tanto la base de conocimientos y el sistema no dependen del orden en el que las reglas son establecidas, almacenadas o procesadas.

Esta técnica suele utilizarse cuando la cantidad de datos es potencialmente muy grande, y resulta de interés algún conocimiento específico tomado en consideración (caso típico en los problemas de diagnóstico; MYCIN, por ejemplo). Esta técnica se corresponde con el método clásico en Lógica de la demostración de un teorema en un sistema axiomático. Debido al sistema axiomático (la base de conocimientos), la estrategia consiste en, dado un teorema (un objetivo), partir de él y tratar de encadenarlo (demostrarlo) en el sistema.

Se puede esquematizar de la siguiente forma:

- Toma de hechos iniciales
- Comienzo
- Fase de filtrado
    - Determinación del conjunto de reglas aplicables
    - Mientras que el conjunto de reglas aplicables no está vacío y el problema no está resuelto
        - Hacer
            - Fase de elección
            - Resolución de conflictos
            - Aplicar regla elegida
            - Modificar eventualmente el conjunto de reglas aplicables o base de hechos
        - Fin de hacer
    - Fin de mientras que
- Fin

La eficacia del motor de inferencia reside en la pertinencia de la decisión tomada (regla elegida) durante la fase de elección. La regla elegida condiciona la rapidez con la que el sistema llegará a la solución.

Ejemplo:

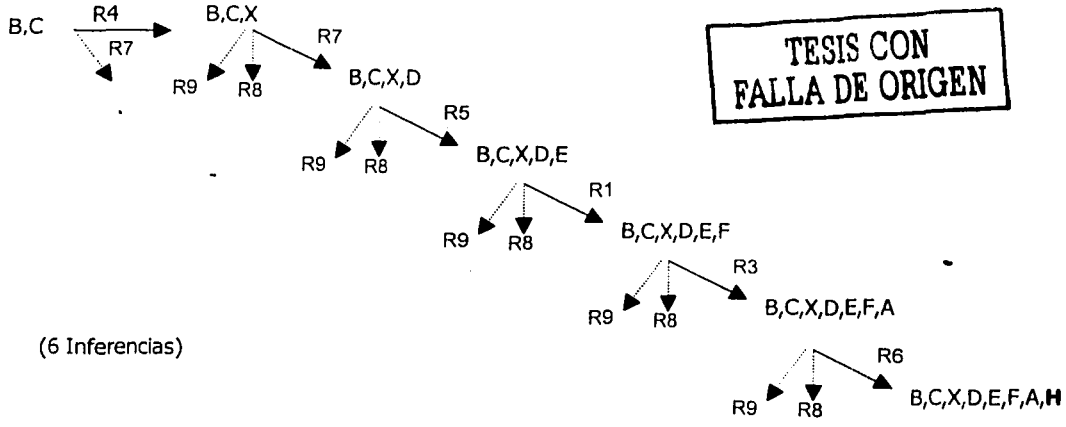
Sea la base de conocimientos:

R1: Si B y E	Entonces F	R6: Si A y X	Entonces H
R2: Si C y G	Entonces A	R7: Si C	Entonces D
R3: Si C y F	Entonces A	R8: Si X y C	Entonces A
R4: Si B	Entonces X	R9: Si X y B	Entonces D
R5: Si D	Entonces E		

Sea la base de hechos: **B, C**

Y el objetivo: **H**

1. Encadenamiento hacia adelante correspondiente a la primera regla aplicable en el orden de su numeración. Donde las líneas rígidas representan la regla aplicada, y las punteadas aquellas que también se pudieron aplicar.

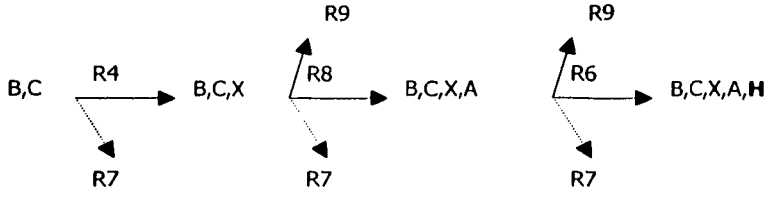


TESIS CON
FALLA DE ORIGEN

Figura 17. Primer resolución del ejemplo de encadenamiento hacia adelante.

ESTA TESIS NO SALE
DE LA BIBLIOTECA

2. Encadenamiento hacia adelante correspondiente a una estrategia que selecciona entre las reglas aplicables la que comporta mayor número de condiciones, empleándose en caso de igualdad la regla con un número más fiable.



(3 Inferencias)

TESIS CON
FALLA DE ORIGEN

Figura 18. Segundo resolución del ejemplo de encadenamiento hacia adelante.

Como podemos observar, de acuerdo a la estrategia de control que hayamos elegido depende el número de inferencias que utilicemos, y por lo tanto, el total de pasos necesarios para llegar a la solución. En este ejemplo, la primer estrategia elegida requirió de 6 inferencias y 8 pasos, y l asegunda estrategia, sólo 3 inferencias y 5 pasos.

Desventajas: esta forma de razonamiento posee diversos inconvenientes:

- El sistema activa todas las reglas aplicables, incluso aunque algunas no ofrezcan ningún interés.
- La base de hechos debe contener suficiente número de hechos iniciales para que el sistema pueda llegar a una solución.
- Los usuarios deben suministrar al SE toda la información que posee.
- En caso de rechazo, un solo hecho podría permitir llegar al objetivo, pero el usuario no está informado puesto que el conocimiento no es interactivo.
- Es necesario aplicar todas las reglas aplicables para deducir todo lo que se puede deducir.

Encadenamiento hacia atrás o dirigido por los objetos

La técnica del "encadenamiento hacia atrás" consiste en tratar de probar un dato (o conocimiento) enganchándolo en la base de reglas con el esquema de inferencia modus ponens, i. e., tomando al dato como un consecuente y buscando en la base de conocimientos el antecedente, a través de los pasos correspondientes.

El sistema parte del objetivo (o de una hipótesis del objetivo) y trata de volver a los hechos para demostrarlo.

Las reglas seleccionadas son las de la parte derecha consecuente, que corresponde al objetivo.

Las condiciones desconocidas (parte izquierda) subsisten mientras existan objetivos por demostrar.

Este proceso se repite hasta que todos los subobjetivos se hayan demostrado o se alcance el objetivo final, o hasta que no exista la posibilidad de seleccionar más reglas.

En cualquier caso, el sistema puede solicitar al usuario la resolución de uno o varios subobjetivos y el proceso comienza de nuevo.

El rechazo ocurre cuando el sistema no puede seleccionar reglas, ni plantear cuestiones al usuario (reglas insuficientes o incoherentes, o cuando el usuario no puede responder a las preguntas del SE).

Se puede esquematizar de la siguiente forma:

- Comienzo
- Fase de filtrado
- Si el conjunto de reglas seleccionadas está vacía
    - Entonces preguntar al usuario
- En caso contrario
    - Mientras que el objetivo no se haya resuelto y queden reglas seleccionadas
        - Hacer
            - Fase de elección
            - Añadir los subobjetivos correspondientes a la parte izquierda de la regla elegida
            - Si un objetivo no se ha resulto
                - Entonces resolverlo (desde el principio)
            - Fin si
        - Fin hacer
    - Fin mientras
- Fin si
- Fin fase
- Fin

La estrategia usada es muy simple, puesto que consiste en usar la primera regla aplicable, en su orden de numeración, para intentar a continuación, verificar uno tras otros los subobjetivos producidos (exploración con búsqueda en profundidad).

La exploración puede detenerse:

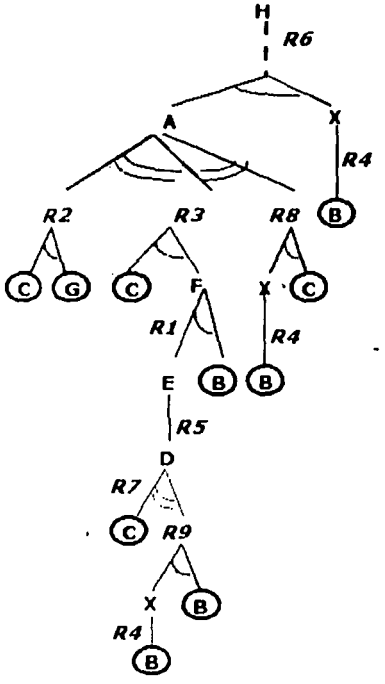
- Cuando el objetivo inicial se demuestra.
- Cuando se han explorado sin éxito todas las posibilidades.

Este modo de razonamiento consiste en construir un grafo Y/O, en donde

Los subobjetivos deben ser verificados.

Puede aplicarse una u otra regla.

Ejemplo: utilizando la base de conocimientos, la base de hechos y el objetivo, H, anteriores, aplicar el encadenamiento hacia atrás.



Donde:

H Es el objetivo.

Ri Son las reglas aplicables.

(i) Son los estados finales.

Obteniendo así, las bases de conocimientos:

1. **C, G y B**
2. **C y B**

TESIS CON
FALLA DE ORIGEN

Figura 19. Resolución del ejemplo mediante encadenamiento hacia atrás.

Ventajas

- El sistema plantea cuestiones únicamente cuando es necesario y después de haber explorado todas las posibilidades.
- El árbol de búsqueda es normalmente más pequeño que en el caso del encadenamiento hacia adelante.
- El proceso es interactivo.

Desventaja:

- Uno de los riesgos del encadenamiento hacia atrás es el de meterse en un bucle: para demostrar A hay que demostrar B, para demostrar B hay que demostrar A etc.

Estas dos formas de inferencia se corresponden con los dos métodos lógicos clásicos conocidos por varios nombres: método resolutivo / método compositivo; análisis / síntesis, etc. La distinción se basa en la relación direccional entre objetivos y datos. Y ambas formas pueden combinarse en el razonamiento. Cabe partir de un supuesto inicial, inferir una conclusión mediante un razonamiento hacia adelante y luego establecer un encadenamiento hacia atrás hasta encontrar los datos que confirman esa conclusión. A su vez, dentro de cada uno de estos tipos de razonamiento cabe distinguir "estrategias de solución", como las llamadas "búsqueda en profundidad" y "búsqueda en amplitud" que se vieron en el Capítulo IV, Representación de problemas de IA en el apartado de Métodos de Búsqueda en un espacio de estados.

CAPÍTULO VI.
APLICACIONES ESPECÍFICAS.

Como hemos visto, una de las principales aplicaciones de la Inteligencia Artificial son los Sistemas Expertos. Un SE es un término que ha ganado terreno y popularidad en los últimos tiempos ya que se refiere a algún tipo de programa de computación que emula la actuación de un experto humano en algún área específica. A continuación veremos algunas de sus aplicaciones.

Visión

La visión por computadora es una de las áreas de aplicación más complejas para las computadoras. Las tareas de la visión se ponen en ejecución en computadoras con el uso de diversas técnicas que se extienden del proceso de imagen, de la concordancia con el modelo o de la inteligencia artificial, y han conducido una parte significativa de la investigación en estas áreas así como en las tecnologías usadas para su puesta en práctica.

La visión artificial trata de traducir el mundo visual (que las personas interpretan como real) a un código binario de forma tal que facilite a un sistema robótico interactuar con su entorno. Este procesamiento se desarrolla en diferentes niveles: digitalización de la imagen, detección de contornos, reconocimiento de objetos y técnicas de interpretación.

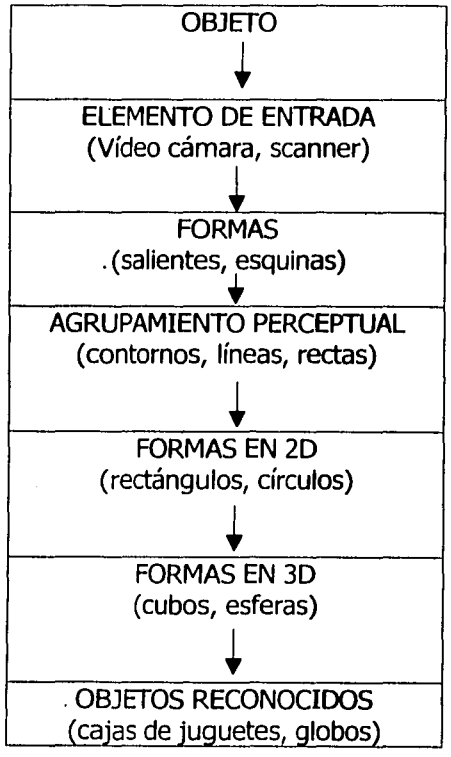
A pesar de tener resuelto el problema de imitar funcionalmente al ojo humano (con la invención de la cámara fotográfica y, posteriormente, de la cámara de vídeo), la interpretación de una imagen dista de ser un ejercicio trivial. En efecto, la tarea de transformar una imagen bidimensional en un objeto tridimensional identificado es uno de los retos más desafiantes que se hayan enfrentado. A fin de lograr una "comprensión" real de una escena y tener en cuenta el contexto, es necesario disponer en los sistemas computacionales -al igual que en el procesamiento del lenguaje- de un conjunto de conocimientos previos afín al campo de aplicación que se encara. De esta forma, por ejemplo en una fotografía satelital, un rectángulo podría interpretarse como una casa, pero también como un vehículo. Un conocimiento más profundo del campo de aplicación permite inferir que un pequeño rectángulo situado en medio de una superficie monótona tiene muchas probabilidades de ser un establecimiento rural.

La ventaja de éstos sistemas reside en su habilidad para realizar tareas repetitivas y tediosas con alto grado de eficiencia; características que los capacitan para la detección de fallas, análisis microscópico, control de calidad, selección de

componentes, etc. Así, por ejemplo, los sistemas actuales reconocen objetos artificialmente diseñados (piezas mecánicas, componentes electrónicos o planos de circuitos electrónicos) e, incluso, son capaces de distinguir los colores de pequeñísimas piezas iguales que se desplazan a velocidades imposibles de percibir por el ojo humano, así como medirlas con una precisión increíble o clasificarlas por color. Sin embargo, tienen dificultades cuando "ven" rostros, árboles, animales o paisajes.

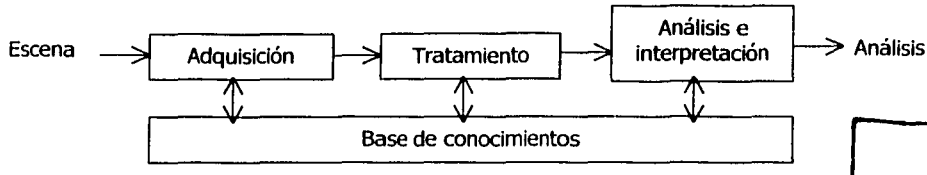
Los sistemas para visión perciben la forma, tamaño, localización o color de los objetos. Procesan e interpretan las imágenes para analizarlas e identificarlas. Los sistemas de visión de alto nivel usan técnicas de IA como razonamiento simbólico para facilitar las ambigüedades tales como: identificación de objetos según modelos, comparación de aspectos de los mismos con parámetros almacenados, etc. Estos modelos están equipados con técnicas heurísticas acerca del tamaño, forma y relaciones espaciales para su interpretación como imágenes.
Hay sistemas de visión de dos y tres dimensiones (2D o 3D); los primeros interpretan objetos o escenas como fotografías y los segundos objetos con dimensiones.

Procesos básicos relacionados con un sistema de visión:



TESIS CON
FALLA DE ORIGEN

La visión artificial está integrada por una serie de etapas que esquemáticamente serían:



TESIS CON FALLA DE ORIGEN

Figura 20. Etapas de un sistema de visión por computadora

De estas etapas, las dos primeras se conocen relativamente bien por su aplicación directa en los procesos industriales, pero la tercera no; el problema radica en las ambigüedades inherentes a las escenas naturales, ya sea por el contexto o entorno en donde está la escena, la iluminación, los propios objetos, los sensores, las técnicas utilizadas o el dominio del problema.

Uno de los problemas importantes en el análisis de imágenes es el enorme cálculo necesario para escoger entre los posibles candidatos cuando los objetos pueden presentarse en cualquiera de sus posiciones espaciales. Este problema da lugar a la necesidad de la heurística para reducir la búsqueda, aunque no siempre es posible su utilización debido a que puede perderse información básica para interpretar la escena.

Ejemplos de Sistemas Expertos para Visión desarrollados en México:

- Construcción y ajuste de un modelo deformable de la mano en imágenes
  Desarrollado por: *Joaquin Peña Acevedo*

Lenguajes: C, GL y Open GL. Plataforma: Indy Silicon Graphics.
Partiendo de una colección de imágenes de manos, una caracterización de éstas por medio de un modelo distribuido de puntos (PDM), su mano promedio y del cálculo de sus principales modos de variación, es posible lograr que el modelo se deforme para ajustarse a la forma que tiene la proyección de una mano en una imagen. Esto tiene como finalidad que la computadora identifique una serie de ademanes que sirvan para establecer una comunicación visual con ella. 1999

- Detección de rasgos faciales y gestos en imágenes por medio de modelos deformables
  Desarrollado por: *Emilio Aguirre Altieri*

Lenguajes: C, C++, OpenGL. Plataforma: Unix, Windows 95/98.
Este procedimiento permite la creación de modelos compactos y flexibles, los cuales representan una clase de un conjunto de figuras por medio de: La posición media de un conjunto de puntos etiquetados y un pequeño número de modos de variación con

respecto a la media. Los puntos del modelo no requieren estar situados en los contornos de los objetos, éstos pueden representar rasgos internos, e inclusive sub-componentes de un ensamblado complejo. Este tipo de modelos puede ser utilizado en el análisis de imágenes en diferentes dominios o como clasificador (dado un ejemplo de una figura, se puede dar un estimado de que tan similar un ejemplo es a un miembro de una clase de figuras descritas por un modelo).

Ambos del Laboratorio Nacional de Informática Avanzada, A. C. LANIA.

- Reconocimiento visual de gestos dinámicos empleando redes bayesianas dinámicas
  Desarrollado por: Héctor Hugo Avilés Arriaga y Luis Enrique Sucar Succar .
  En el Instituto Tecnológico de Estudios Superiores Monterrey.

Comprensión del lenguaje natural

La comprensión del lenguaje natural por parte de las computadoras ha dejado de ser un tema de ciencia ficción tratado sólo en películas, para pasar a ser un tema actual de debate en los círculos científicos. Se han hecho infructuosos esfuerzos para que una computadora sostenga una conversación pero nunca se ha logrado que una persona confunda a un programa con un humano, circunstancia que según Alan Turing [7] nos indicaría que estamos tratando ya, no con una computadora, sino con una Inteligencia Artificial.

Lenguaje. Un lenguaje es la función que expresa pensamientos y comunicaciones entre la gente. Esta función es llevada a cabo por medio de señales y vocales(voz) y posiblemente por signos escritos(escritura).

Lenguaje Natural. El Lenguaje Natural (LN) es el medio que utilizamos de manera cotidiana para establecer nuestra comunicación con las demás personas. Este tipo de lenguaje es el que nos permite designar las cosas actuales y razonar acerca de ellas, fue desarrollado y organizado a partir de la experiencia humana y puede ser utilizado para analizar situaciones altamente complejas y razonar muy sutilmente. La riqueza de sus componentes semánticos da a los lenguajes naturales su gran poder expresivo y su valor como una herramienta para razonamiento sutil.

7 Puede considerarse a Alan Turing el padre de la IA, aunque este nombre no se utilizó hasta después de 1956, Turing estableció un nuevo paradigma cibernético y con ello amplió las fronteras de una ciencia abierta.
http://www.imagia.com.mx/hmm/va/Turing.htm

Por otro lado, la sintaxis de un LN puede ser modelada fácilmente por un lenguaje formal, similar a los utilizados en las matemáticas y en la lógica.
Otra propiedad de los lenguajes naturales es la polisemántica, es decir, la posibilidad de que una palabra tenga diversos significados.

En resumen, los lenguajes naturales se caracterizan por las siguientes propiedades:
1. Desarrollados por enriquecimiento progresivo antes de cualquier intento de formación de una teoría.
2. La importancia de su carácter expresivo debido grandemente a la riqueza del componente semántico (polisemántica).
3. Dificultad o imposibilidad de una formalización completa.

Lenguaje formal. Es aquel que el hombre ha desarrollado para expresar las situaciones que se dan en específico en cada área del conocimiento científico. Las palabras y oraciones de un lenguaje formal son perfectamente definidas (una palabra mantiene su mismo significado prescindiendo de su contexto o uso).
Los lenguajes formales son exentos de cualquier componente semántico fuera de sus operadores y relaciones. Los lenguajes formales pueden ser utilizados para modelar una teoría de la mecánica, física, matemática, o de otra naturaleza, con la ventaja de que en estos, toda ambigüedad es eliminada.

En resumen, las características de los lenguajes formales son las siguientes:
1. Se desarrollan de una teoría establecida.
2. Componente semántico mínimo.
3. Posibilidad de incrementar el componente semántico de acuerdo con la teoría a formalizar.
4. La sintaxis produce oraciones no ambiguas.
5. La importancia del rol de los números.
6. Completa formalización y por esto, el potencial de la construcción computacional.

Procesamiento computacional del Lenguaje Natural

Una meta fundamental de la Inteligencia Artificial, es la manipulación de lenguajes naturales usando herramientas de computación; en ésta, los lenguajes de programación juegan un papel importante, ya que forman el enlace necesario entre los lenguajes naturales y su manipulación por una máquina.

El procesamiento del lenguaje natural es un área de la IA que tiene que ver con los conocimientos y técnicas necesarios para que las computadoras puedan procesar el mismo lenguaje que usamos los humanos en el habla cotidiana.

Niveles del Lenguaje

Es necesario conocer los niveles del lenguaje natural, que son los siguientes: fonológico, morfológico, sintáctico, semántico, y pragmático.

Nivel Fonológico. Trata de cómo las palabras se relacionan con los sonidos que representan.

Nivel Morfológico. Trata de cómo las palabras se construyen a partir de unidades de significado más pequeñas llamadas morfemas, por ejemplo:

Rápida + Mente = Rápidamente

Nivel Sintáctico. Trata de cómo las palabras pueden unirse para formar oraciones, fijando el papel estructural que cada palabra juega en la oración y que sintagmas son parte de otros sintagmas.

Formas más utilizadas para representar información sintáctica:

a. Un conjunto explícito de reglas: reescritura de patrones, gramáticas de contexto libre aumentadas con restricciones, gramáticas de cláusulas, etc.

b. Redes de transición aumentadas (ATN) que expresan las reglas como cadenas de transiciones en autómatas recursivos.

Un ATN esta formada por un conjunto de autómatas. Cada autómata consta de un conjunto de nodos y de arcos con diferentes tipos de etiquetas:

- Analizar. Llamada al autómata
- Categoría. Comprobación de una palabra y avance sobre la cadena de entrada
- Pop. Fin y vuelta al punto donde se produjo la llamada.

Nivel Semántico. Trata del significado de las palabras y de cómo los significados se unen para dar significado a una oración, también se refiere al significado independiente del contexto, es decir de la oración aislada.

Hay diferentes formas de llevar a cabo el análisis semántico:

- Redes Semánticas.
- Teoría Conceptual.
- Frames. Describen clases de objetos en función de los aspectos de los mismos. Este método es una manera de organizar el conocimiento como una colección de características comunes al concepto, objeto, situación o sujeto.

Nivel Pragmático. Trata de cómo las oraciones se usan en distintas situaciones y de cómo el uso afecta al significado de las oraciones. Se suele reconocer un subnivel recursivo/discursivo, que trata de cómo el significado de una oración se ve afectado por las oraciones inmediatamente anteriores.

El componente pragmático es, finalmente, el que resuelve todas las referencias relativas al contexto, así como el que produce la interpretación del significado literal en términos del conocimiento sobre el dominio y las intenciones de los interlocutores.

Por una parte, es necesario relacionar las frases con el texto o diálogo donde se produce, para resolver pronombres, elipsis, frases incompletas, etc. Así, por ejemplo, en los diálogos A y B:

Diálogo A	Diálogo B
Dónde está el jamón?	Dónde está el jamón?
En el armario	En el armario
Sácalo	Ciérralo

Tenemos que:

1. *el jamón* designa a un objeto concreto, al igual que *el armario.*
2. *en el armario* es una frase incompleta ligada a la anterior, *el jamón esta en el armario.*
3. en A *lo* se refiere a *jamón*, mientras que en B se refiere a *armario.*

El problema de la ambigüedad

Uno de los grandes problemas del procesamiento de lenguaje natural, PLN, se produce cuando una expresión en LN posee más de una interpretación, es decir, cuando en el lenguaje de destino se le puede asignar dos o más expresiones distintas. Este problema de la ambigüedad se presenta en todos los niveles del lenguaje, sin excepción.

Ejemplo:

Juan vio a María, con el telescopio.

Juan vio a María con el telescopio.

En este ejemplo, la primera oración indica que Juan vio a María a través del telescopio, y en la segunda, Juan vio a que María traía un telescopio, pero para comprender exactamente el significado de estas oraciones, tendríamos que estar situados en el contexto completo de la oración o en la escena.

En apariencia este problema es demasiado sencillo, pero en realidad, es uno de los más complicados y que más complicaciones ha dado para que el PLN pueda desarrollarse por completo, ya que al presentarse en todos los niveles del lenguaje, se tienen que desarrollar programas (en lenguaje formal) para solucionarlos en cada caso.

Tipos de ambigüedades:

- Problemas de orden semántico.
- Omisiones de palabras que pueden ser rescatadas en el contexto en que se dan.
- La existencia de la polisemia (una misma palabra puede tener varios significados diferentes).
- La homografía (palabras diferentes pueden ser escritas de la misma manera).

El procesamiento del lenguaje es el que mejor demuestra, por ahora, los avances y las posibilidades de la IA.

Es también, por cierto, el que presenta el mayor interés para los comunicadores, ya que se relaciona directamente con el problema de la transmisión y del uso de la información.

Medicina

Las aplicaciones médicas han sido siempre uno de los campos de atención principales para la ingeniería. Desde el surgimiento de los sistemas informáticos, una de las primeras aplicaciones con tarjetas perforadas fue la automatización de historias clínicas.
Desde los inicios de la Inteligencia Artificial, se empezó a trabajar en diagnóstico automático y en la representación del conocimiento médico.

Cualquier relación de los sistemas expertos más conocidos (sobre todo, los primeros en recibir ese nombre) pretenden resolver problemas en algún área o aspecto de la medicina. Esto puede explicarse por el hecho de que el campo de la medicina es intelectualmente atractivo y desafiante para un informático ya que en la actividad

médica intervienen numerosos procesos de decisión bajo incertidumbre e imprecisión. Sin embargo, salvo muy pocas excepciones, los sistemas desarrollados no han pasado del entorno de investigación a la aplicación clínica rutinaria, a pesar de las evaluaciones muy favorables de muchos de ellos.

La pretensión del hombre de modificar su propia naturaleza con tecnologías actuales como la clonación, también hoy es posible gracias al desarrollo de la Informática.

Con la informática la ciencia cognitiva se ha orientado hacia la tecnología y ha nacido una nueva ciencia la inteligencia artificial (IA), que pretende explicar los procesos que son análogos al razonamiento humano y cuyos principios son:

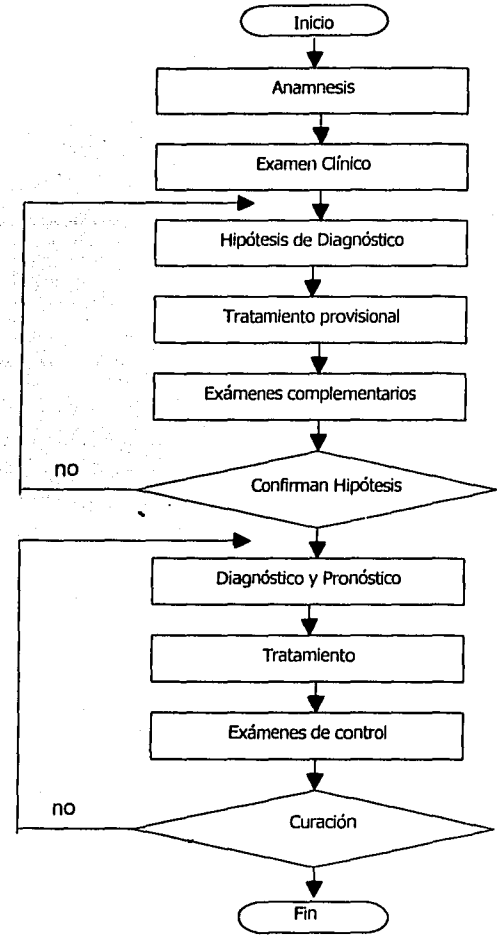
- Desarrollar comportamientos similares al humano.
- Utilizar técnicas de aprendizaje.
- Llegar a ser capaz de aprender.

En educación médica, hoy contamos con tecnologías como los simuladores, cada vez más sofisticados, y con posibilidades de entrenamiento en áreas como la cirugía, radioterapia y endoscopia.

La Informática y las telecomunicaciones en medicina dan origen a la "Telemedicina" con posibilidades tan sorprendentes como la de realizar cirugía entre dos grupos disciplinarios distantes a través de Internet.

En medicina la inteligencia artificial se ha convertido en una nueva especialidad facilitando a los médicos herramientas como los Sistemas Expertos y las Bases de Datos en línea, para ayudar en los diagnósticos.

La actividad médica puede descomponerse en tres partes: adquisición de datos del paciente, interpretación de esos datos para el establecimiento de un diagnóstico y de un pronóstico, y formulación y seguimiento de un plan terapéutico. Los datos incluyen características físicas, antecedentes familiares y personales y las llamadas manifestaciones: síntomas (sensaciones subjetivas comunicadas por el paciente: dolor, somnolencia, etc.), signos (hechos objetivos observados o médicos: temperatura, sonidos cardíacos, etc.) y resultados de laboratorio. Normalmente, estos datos no se adquieren de una sola vez, sino en el curso de unos procesos iterativos en los que se encuentran también actividades de diagnóstico y de tratamiento.

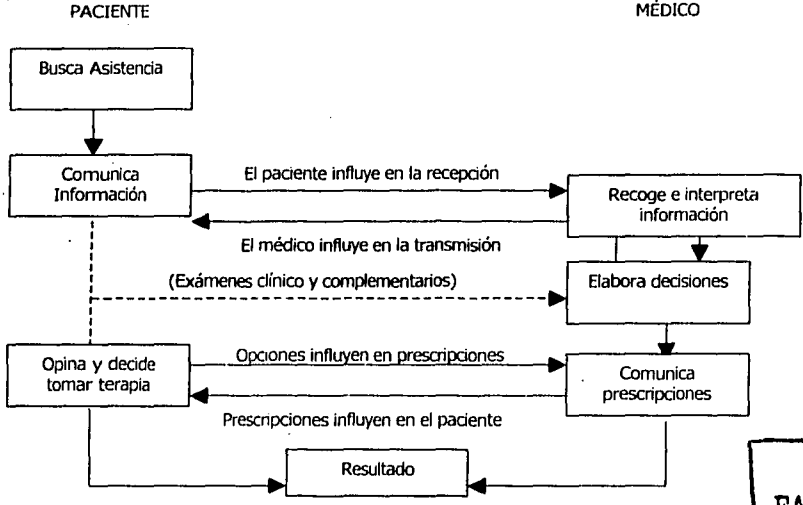


TESIS CON
FALLA DE ORIGEN

Figura 21. Procesos de la actividad médica.

El primer acto en la relación médico-paciente es el de la *anamnesis*, en el cual el médico adquiere los primeros datos en entrevista personal con el paciente, seguido de un examen clínico. Ya durante este acto el médico va manejando una hipótesis de diagnóstico, para las cuales pone en marcha sus mecanismos de conocimiento. Tales mecanismos incluyen tanto conocimientos explícitos o de libro como conocimientos implícitos, adquiridos mediante la experiencia profesional, que son los que distinguen al médico experimentado del principiante.

De la importancia de este primer acto cuenta el hecho de que durante el mismo quedan establecidos el setenta por ciento de los diagnósticos. Y la dificultad para analizarlo es doble: por una parte, el conocimiento que da la experiencia es difícilmente explicable; por otra: en la comunicación médico-paciente se establecen mecanismos de interacción que influyen en la calidad de la información que se recoge y en su interpretación.



TESIS CON
FALLA DE ORIGEN

Figura 22. Comunicación médico – paciente.

A principios de los años setenta, cuando empezaban a conocerse las primeras realizaciones fundamentadas en el paradigma de sistemas basados en conocimiento, se analizaban críticamente los sistemas de ayuda a las decisiones clínicas y se proponía aplicar técnicas basadas en razonamiento simbólico más que en un cálculo numérico para conseguir que los programas tuviesen características esenciales:

a) Conocimientos médicos, incluyendo reglas heurísticas y sentido común, en los que los médicos se apoyan para tomar sus decisiones.

b) Capacidad conversacional, tanto para captar el conocimiento de los expertos humanos como para comunicarse con los usuarios no informáticos.

c) Capacidad de explicación, para que el sistema pueda explicar a los usuarios sus líneas de razonamiento.

Desde entonces, prácticamente todos los trabajos sobre sistemas de ayuda a la toma de decisiones médicas se han orientado a la construcción de sistemas expertos.

Entre 1970 y 1980 se desarrolló MYCIN para consulta y diagnóstico de infecciones de la sangre. Este sistema introdujo nuevas características: utilización de conocimiento impreciso para razonar y posibilidad de explicar el proceso de razonamiento. Lo más importante es que funcionaba de manera correcta, dando conclusiones análogas a las que un ser humano daría tras largos años de experiencia. En MYCIN aparecen claramente diferenciados motor de inferencia y base de conocimientos. Al separar esas dos partes, se puede considerar el motor de inferencias aisladamente. Esto da como resultado un sistema vacío o shell. Así surgió EMYCIN (MYCIN Esencial) con el que se construyó SACON, utilizado para estructuras de ingeniería, PUFF para estudiar la función pulmonar y GUIDON para elegir tratamientos terapéuticos.

Ejemplos de sistemas expertos para medicina

SISTEMA	DOMINIO DE APLICACIÓN	MODELO
ABEL	Problemas electrolíticos	RS
AI/Rheum	Reumatología	RS
ANTICIPATOR	Prescripciones de antibióticos	RP
APES	Dermatología	RP
CAA	Electrocardiografía	RS
CADUCEUS	Medicina interna	RS Y E
CASNET	Oftalmología	RS
CENTAUR	Neumología	RP Y E
GAITSPERT	Neurología	RP
GUIDON	Enseñanza	RP
HEAMED	Psicofarmalogía	RP
INTERNIST	Medicina interna	RS Y E
IRIS	Oftalmología	RP, RS Y E
LOCALIZE	Neurología	RP
MEDIKS	Medicina general	RS
MYCIN	Enfermedades infecciosas	RP
NEOMYCIN	Enseñanza	RP
NEPHROS	Insuficiencia renal	RS
NEUROLOGISTIC	Neurología	RP
ONCOCIN	Oncología	RP Y E
PATREC	Colecistitis	E
PUFF	Neumología	RP
RADEX	Radiología	E
RX	Reumatología (inducción de conocimiento)	RP
SAM	Neurología	RP
SEEK	Reumatología (inducción del conocimiento)	RP Y RS
SPHINX	Sistema esencial	RP Y E
TOUBIB	Medicina general	RP Y E
VM	Monitorización respiratoria	RP

Tabla 3. Relación de sistemas expertos para medicina.

RP = Reglas de producción, RS = Redes Semánticas, E = Estructuras (frames)

En la tabla 3 se da una relación de sistemas expertos desarrollados para algún campo de la medicina. En la columna llamada "modelo" se indica la técnica básica adoptada en cada caso para al representación del conocimiento.

MYCIN, desarrollado en la Universidad de Stanford, es el sistema más conocido y el que más ha influido en el desarrollo de otros. Su arquitectura basada en reglas de producción con factores de certidumbre fue la base del sistema esencial EMYCIN con el que posteriormente se desarrollaron PUFF, SACON, CLOT, DART, etc.

Desde el punto de vista psicológico, se ha demostrado que las reacciones y de las actitudes de los médicos con respecto a las herramientas informáticas, en general, van siendo cada vez más positivas. Se ha visto que cualquier sistema informático, independientemente de que pueda funcionar muy bien, se verá indiscutiblemente rechazado, pero años más adelante se observan actitudes más favorables. Sin embargo, se ha propuesto que la informática tiene que considerarse como una ciencia básica de la medicina.

Las razones pragmáticas pueden resumirse en una muy simple: que los sistemas casi nunca se han diseñado para cubrir una necesidad real práctica. Las evaluaciones de varios sistemas (como PUFF, MYCIN, INTERNIST) coinciden en que alrededor del 80% de los evaluadores los califican positivamente, en el sentido de que funcionan al nivel del experto, pero no que sean de utilidad. Sugieren que la evaluación completa debería contemplar las siguientes fases:

1) Necesidad del sistema.
2) Funcionamiento al nivel de experto.
3) Facilidad de manejo (diálogos, tiempo de respuesta, etc.).
4) Aceptación (¿lo usan los médicos, aún cuando sea opcional?).
5) Impacto sobre la actividad de los médicos (¿ha cambiado su comportamiento?).
6) Impacto sobre la mejora de la calidad asistencial.
7) Mejora en la relación costo/eficacia de pruebas y terapias.

Finanzas y gestión

Es el segundo campo en importancia en cuanto a aplicaciones de los SE debido principalmente a las fuertes inversiones que han realizado las entidades financieras, bancarias y aseguradoras.

Aplicaciones potenciales de los sistemas expertos en finanzas:

Auditoría: Análisis de la materialidad y del riesgo, evaluación del control interno, planificación de la auditoría, evaluación de la evidencia, análisis de cuentas concretas, formación de opinión, emisión del informe, auditoría interna, auditoría informática, etc.

Contabilidad de costos y de gestión: Cálculo y asignación de costos, asignación de recursos escasos, control y análisis de desviaciones, planificación y control de gestión, diseño de sistemas de información de gestión, etc.

Contabilidad financiera: regulación legal, normas y principios contables, recuperación y revisión analítica de registros contables, diseño de sistemas contables, imputación contable, consolidación de estados contables, etc.

Análisis de estados financieros: Análisis patrimonial, financiero y económico de los estados contables, salud financiera de la empresa, cálculo e interpretación de porcentajes, cálculo y análisis de tendencias, etc.

Planificación financiera e industria de los servicios financieros: Planificación financiera corporativa, planificación financiera personal, análisis de inversiones, gestión de tesorería, mercado de valores, seguros, banca, concesiones de crédito, etc.

La mayoría de las aplicaciones de sistemas expertos desarrolladas en el campo del análisis de estados financieros están compuestas por dos módulos:

- Un módulo informático convencional, que realiza los cálculos por medio de la informática tradicional, o con la ayuda de una hoja de cálculo electrónica.
- Un módulo de sistema experto, que realiza los procesos de análisis e interpretación de los datos y de emisión del informe.

El tema más importante para el desarrollo de herramientas financieras con sistemas expertos es la adquisición de conocimientos específicos: que criterios serían los utilizados por un experto para tomar las mejores decisiones.

En este campo algunas otras aplicaciones y ejemplos son las siguientes:

- Análisis de mercados.
- Análisis de riesgos y tasación de seguros (ejemplo UNDERWRITING ADVISOR de Syntelligence, ASS.I.RI de Assicurazione Generali/1987).
- Aplicación de impuestos y tasas (ejemplo TAX GENUS 1987 de Apocalypse Systems).
- Asesoría jurídico y fiscal.
- Ayuda a la correcta realización de operaciones bancarias.

- Concesión de créditos y préstamos (ejemplo Letter of Credit Advisor de Bank of América empleando Expert Edge, Loan Risk Advisor de Frametec).
- Concesión de tarjetas de crédito (ejemplo Banco de Bilbao).
- Evaluación de riesgos de gestión de cartera (ejemplo Banco de Santander).
- Gestión del personal (ejemplo MYRIAM de EDF-GDF).
- Planes de inversión de capitales (ejemplo Planpower de Apex/1986 para la planificación financiera, Financial Advisor de Palladian Software).
- Planes de pensiones.
- Previsión de los tipos de interés.
- Previsión en las fluctuaciones en el mercado de divisas. (p.c. PANISSE del CEFI).
- Supervisión de los estados financieros.
- Valoración de la situación financiera de una empresa o cliente (ejemplo Director Ideal del Banco de Santander).
- Verificación de firmas.

Otras muchas aplicaciones se pueden encontrar si enfocamos la actividad financiera como una actividad industrial que hace un uso muy importante de medios electrónicos, etc.

Industria

La problemática de los SE industriales se centra en la necesidad de que los SE se comuniquen con dispositivos sensores, bases de datos, dispositivos de mando y accionamiento, etc., y todo ello en tiempo real.

De los distintos tipos de SE los más empleados en la industria son los de diagnóstico y reparación de averías tanto de productos como de maquinaria, por las siguientes razones:

- Los equipos y productos tienen una complejidad creciente.
- Los sistemas industriales se diseñan en la gran mayoría de las veces a prueba de fallos, lo que impide que aparezca dos veces el mismo fallo.
- Las reparaciones deben de realizarse sin interrumpir el servicio o en el menor tiempo posible, por las pérdidas que genera la inactividad.
- Presencia cada vez mayor de equipos específicos creados bajo demanda lo que dificulta su mantenimiento y reparaciones.

Vamos a ver algunas de las aplicaciones industriales de los SE:

- Diagnósticos de control de calidad (ejemplo los desarrollos de IGC y Labein).
- Detección y actuación en caso de alarmas y emergencias (ejemplo REACTOR de EG&G de Idaho.
- Planes de seguridad en plantas de energía nuclear.
- SMOKE de Univ. Carnegie Mellon para la detección y actuación en caso de incendio.
- LABEIN para el análisis de alarmas en redes eléctricas de alta tensión
- Configuración de equipos y sistemas bajo demanda.

El problema que surge es el gran número de configuraciones posibles y las interacciones que hay que considerar para cada uno de los componentes (ejemplo cada disco duro necesita un tipo de controlador específico). Así por ejemplo en el caso de un ordenador como el DPS90 puede tener unos 3500 componentes y cada uno tiene unos 479 tipos distintos; el número total de configuraciones posibles sería de 479 elevado a 3500 un número excesivamente grande para cualquier calculadora y para tenerla en la mente de cualquier ingeniero de sistemas, cuando el cliente le pide una configuración que cubra exactamente sus necesidades. El número de decisiones a tomar en un proceso de este tipo es de 500 a 800.

Este tipo de tareas es compleja (número de combinaciones posibles elevado no siendo todas ellas factibles, existencia de múltiples relaciones, limitaciones especificaciones del cliente a veces contradictorias o incompletas, especificaciones de los componentes, etc.), cara (por la complejidad y el tiempo requerido en el diseño y la comprobación del mismo) y el conocimiento empleado es incremental (aparición de nuevos componentes o variación de las especificaciones de los mismos).

Algunos ejemplos:

- SYSCON de Honeywell para la configuración de computadoras
- BEACON de Burrounghs para la configuración de computadoras desarrollado en Quintus Prolog.
- Configuraciones de centrales telefónicas a las normativas de cada administración por Alcatel.
- Generación de especificaciones y manuales de utilización, mantenimiento y reparación de sistemas fabricados bajo demanda.
- Control de procesos industriales (ejemplo fabricación de tubos de TV por Texas Instruments).
- Gestión óptima de los recursos (ejemplo Symple Factory de Allied Signal destinado a la planificación industrial, CONTROL ROUTING para la flota de aeronaves por parte de Iberia, ULISIS de la Dirección General de la Función Pública para la gestión del personal funcionario).

- Diagnóstico y reparación de averías, tanto dentro de la industria como de los productos manufacturados.

Ejemplos de SE para la Industria desarrollados en México:

- Sistemas Expertos de Planeación.
  Universidad Autónoma de La Laguna, Torreón, Coah.

- Aplicación de técnicas de inteligencia artificial en la modelación del conocimiento y monitoreo de su comportamiento en un proceso industrial multivariable.
  Instituto Tecnológico de Estudios Superiores Monterrey

- Sistema experto para el Instituto Mexicano del Petróleo (IMP), enfocado a la perforación de pozos petroleros.
  Nicolás Kemper Valverde

Electrónica, informática y telecomunicaciones

En la industria electrónica el tipo de SE más utilizados son sin duda alguna los de diseño, diagnóstico y reparación.

El uso de SE para el diseño se debe a la complejidad creciente de los circuitos y a gran número de parámetros a considerar en los mismos, siendo una necesidad la adopción de soluciones de compromiso en el diseño que deben ser explicadas al diseñador.

Veamos ahora una serie aplicaciones:

- Diseño de circuitos de alto grado de integración (ejemplo DAA de Universidad Carnegie Mellon desarrollado en OPS5 y PALLADIO de Universidad de Standford desarrollado con LOOPS).

- Sistemas inteligentes de autodiagnóstico contenidos (ejemplo PIES de Fairchild para el diagnóstico de averías en circuitos integrados, COMPASS de GTE desarrollado con KEE para el diagnóstico de averías en centrales de conmutación telefónica, IDEA de Pacific Belí sobre EXSYS para diagnóstico de avenas en centrales de conmutación telefónica, etc.).

- Configuración de equipos y sistemas.

- Control de redes de comunicación.

El objetivo es construir redes inteligentes es decir, que de alguna forma sean "conscientes" de las señales que conducen y en base a ello puedan optimizar tu funcionamiento. Para ello deben de incorporar conocimiento sobre la distribución de recursos, encaminamiento de las señales, etc. de forma que sean completamente abiertas y flexibles.

- Programación automática.
- Ajuste de equipos y sistemas.
- Optimización de programas de computadora.

Militar

Desde su origen la informática está presente en el campo militar pasando a ser uno de los factores claves en los desarrollos actuales, tanto más cuando el poder de las mismas es ya incluso excesivo para los fines que fueron desarrolladas.

La Inteligencia Artificial y los sistemas expertos están presentes en todos los grandes proyectos militares como por ejemplo el DARPA ("Defense Advanced Research Projects Agency" organismo creado en 1973 para la financiación de los proyectos con aplicaciones militares), Strategic Computing Program", "Compact Lisp Machine", "Pilot's Associate", etc. en los Estados Unidos de Norteamérica.

Dentro del campo militar existen muchas aplicaciones de los SE de entre todas ellas, y pese a las restricciones que existen en las informaciones sobre estos desarrollos, vamos a describir algunas:

- Elección inteligente de contramedidas electrónicas con el fin de obtener la máxima efectividad con unos recursos limitados.

El proceso que sigue el SE tiene tres fases la primera de ellas es la identificación de amenazas, la segunda la determinación de las prioridades de actuación sobre cada una de ellas y la tercera es la asignación de recursos, pasándose de nuevo a la primera fase.

Dentro de lo que es la identificación de las amenazas, existe un desarrollo que trata de la clasificación de las emisiones que no contiene comunicaciones, típicamente radares primarios. En orden de complejidad el SE identificaría la función del radar detectado, el tipo de radar y en que vehículo está ubicado.

Este SE también podría ser el núcleo del reconocimiento de blancos y el guiado de proyectiles.

- Guiado de vehículos y proyectiles de forma semiautomática (ejemplo PILOT'S ASSOCIATES de McDonnel Douglas, para la asistencia al piloto en aviones de combate) o automática de forma inteligente (ejemplo AI.V de Martín Marietta para el guiado de un vehículo terrestre autónomo, etc.).

- Planificación estratégica (ejemplo FRESH de Texas Instruments que planifica la estrategia y táctica de un combate naval,etc.) y de misiones tácticas en función de la topografía del terreno y de las defensas del enemigo (ejemplo KNOBS de MITRE Corp. planificación de misiones aéreas, TART de RAND Corp. planificación de ataques aéreos.

- Reconocimiento automático de blancos y valoración de los mismos.

- Reconocimiento de planes del enemigo.

- Interpretación de señales provenientes de sensores (ejemplo SUX de Pi Nii y Feigenbaum de la Universidad de Standford para la detección de submarinos, misiles de largo alcance, HASP de S.U. interpretación de señales estratégicas,etc.).

- Encaminamiento de los mensajes e informaciones en caso de conflicto, tiempos mínimos de comunicación.

- Estrategia a utilizar en las conversaciones de desarme (ejemplo SE POTITICS de Jaime Carbonell de la Universidad de Carnegie Mellon).

- Optimización de carga (ejemplo AALPS de la USAF).

Además se desarrollan SE de propósito menos específico como son el diagnóstico de averías (ejemplo IFL de McDonnell Douglas para el mantenimiento del helicóptero modelo Apache).

Junto a las características que se han señalado hay que destacar el problema que subyace en estos desarrollos en los que hay que elegir una solución de compromiso entre la exactitud de los resultados y la rapidez en obtenerlos.

Educación

Todos los SE son por si mismos herramientas útiles para la enseñanza, ya que explican los procesos que realizan y justifican su comportamiento. De la misma forma los SE de simulación lo son ya que enseñan el comportamiento de la realidad simulada, con la ventaja de ser ésta explicada.

Sin embargo, gracias a los SE se puede llegar mucho más lejos, ya que recordemos que los SE simulan la forma de resolver los problemas por parte de un experto; un experto en la educación es un pedagogo.

La creación de programas "inteligentes" de enseñanza no en cuanto a los contenidos sino en cuanto a la forma de enseñarlos ha sido un objetivo desde el inicio de la creación de programas educativos para computadora. El hecho de que los programas educativos hayan sido en mayor o menor medida "ignorantes" en cuestiones pedagógicas; y por tanto se limitaban a repetir una serie de pantallas hasta que el alumno conseguía acertar la respuesta adecuada, ha sido el motivo más importante de los recelos a los mismos por parte de los docentes.

Con todo esto los SE pueden:

- Diagnosticar las causas de los problemas de aprendizaje de un alumno. Esto es posible bien mediante la corrección de unos ejercicios o mediante una sesión interactiva con el SE por parte del alumno.

- Recomendar un tratamiento que podría llevar a cabo el maestro o directamente el propio SE.

Un SE completo en enseñanza tiene: un módulo de diagnóstico, un módulo de corrección o tratamiento y un tercer módulo que contenga la base de conocimiento del tema que debe de enseñar el SE.

El futuro de los SE en la enseñanza, no está tanto en la sustitución del maestro por un SE como en el desarrollo de SE de enseñanza por su complejidad (ejemplo lanzamiento espacial), su poco periodo de vida (ejemplo versiones de un sistema operativo de una computadora), su conocimiento creciente (ejemplo terapias de tratamiento de cáncer) hagan imposible la formación de los profesores o maestros que instruyan a los usuarios o alumnos. Además posibilitaría una difusión rápida de los contenidos, una actualización sencilla, una uniformidad de contenidos y criterios de enseñanza, una autoevaluación de las deficiencias de la "pedagogía" utilizada.

Algunos ejemplos de SE de enseñanza son:

- El DIPS de Good de la Universidad del estado de Florida para el diagnóstico e instrucción en la resolución de problemas (1986),.
- El GEO de la Universidad de Quebec para la enseñanza de la geografía canadiense,.
- SOPHIE de la Universidad de Standford para la enseñanza de la detección de averías en circuitos electrónicos,.
- TUTORING VMS de digital para la enseñanza del sistema operativo VAX/VMS.

Ejemplo de SE para la Educación desarrollado en México:

- TUTOREX [8], para el aprendizaje en distintas áreas del conocimiento, basado en la nueva tecnología computacional educativa (pedagogía, psicología, inteligencia artificial, habilidades del pensamiento y técnicas de aprendizaje e instrucción asistidas por computadora) capaz de emular a un tutor humano, y con características de:
  1. identificación del perfil de razonamiento del estudiante,
  2. análisis de lo que se sabe un estudiante y no sólo de las respuestas,
  3. identificación del modelo del estudiante y de las diferentes políticas de transmisión del conocimiento,
  4. empleo de diferentes niveles de tutoreo (helps), así como de estadísticas que identifiquen la ubicación del estudiante dentro del proceso enseñanza aprendizaje.

Sistemas Expertos en México

Además de los SE antes mencionados, en México existen algunos otros como:

- SE del Instituto de Biotecnología de la UNAM con la misión de estudiar organismos bacteriológicos, por Nicolás Kemper Valverde.

- SE de Búsqueda interactiva usado en la localización de etnias mexicanas, a solicitud del Instituto Nacional de Antropología e Historia (INAH).

8 http://www.ur.mx/principal/veritas/1999-2000/TemaAbajo15.htm

Principales centros de investigación y desarrollo de SE en México:

- Centro de Instrumentos de la Universidad Nacional Autónoma de México (Cinstrum-UNAM)
  http://www.cinstrum.unam.mx/Inteligentes/index.html
- Centro de Inteligencia Artificial del Tecnológico de Monterrey, Campus Monterrey
  http://www-cia.mty.itesm.mx/indexflash.html
- División Académica de Ingeniería del Instituto Tecnológico Autónomo de México
- Centro de Investigación en Tecnologías de Información y Automatización (CENTIA) de la Universidad de Las Américas
  http://mailweb.udlap.mx/~centia/
- Centro de Investigación Computacional (CIC)
  http://www.cic.ipn.mx/
- Centro de Investigación y de Estudios Avanzados (CINVESTAV) del Instituto Politécnico Nacional.
  www.cinvestav.mx
- Instituto de Investigaciones Eléctricas (IIE) de la Comisión Federal de Electricidad, la Universidad Veracruzana
  http://www.iie.org.mx/
- Instituto de Investigación en Matemáticas Aplicadas y Sistemas (IIMAS) de la Universidad Nacional Autónoma de México
  www.iimas.unam.mx
- Centro de Investigaciones Científicas y de Educación Superior de Ensenada (CICESE)
  www.cicese.mx
- Laboratorio Nacional de Informática Avanzada, AC
  www.lania.mx

CONCLUSIONES

Hemos revisado las técnicas principales de la Inteligencia Artificial, así como los requerimientos necesarios para la construcción de un Sistema Experto. Con lo anterior podemos decir que la IA nos puede ayudar a resolver problemas reales difíciles, a crear nuevas oportunidades en los negocios, la ingeniería y muchas otras áreas de aplicación.

Después de revisar cada uno de los capítulos, podemos establecer que siempre existirán dos aspectos fundamentales:

- Una forma de representación de conocimiento.
- Métodos de inferencia para resolución de problemas.

Una vez que se ha descrito un problema mediante la representación apropiada, el problema está casi resuelto.

La elección de la forma de representación de conocimiento es la que determina el tipo de métodos utilizados para la resolución de problemas. Entre los métodos que tratamos están:

- Métodos de búsqueda en un espacio de estado; los cuales otorgan al usuario todas las soluciones que pueden existir de un mismo problema.
- Métodos para la representación reducida del problema; los cuales sólo buscan las soluciones más óptimas.

En general, un mayor conocimiento implica menos búsqueda.

Un SE es un término que ha ganado terreno y popularidad en los últimos tiempos ya que se refiere a algún tipo de programa de computación que emula la actuación de un experto humano en algún área específica, que va desde la enseñanza hasta la medicina, pasando por los sistemas expertos para la industria y los negocios; también vimos sus aplicaciones principales así como las funciones que desempeñan, y de las cuales deriva su éxito.

Un Sistema Experto en sí no tiene verdadera Inteligencia Artificial; más bien, es un sistema basado en el conocimiento que, mediante el buen diseño de su base de información y un adecuado motor de inferencias para manipular dichos datos proporciona una manera de determinar resoluciones finales dados ciertos criterios.

Puede decirse, entonces que entre las necesidades urgentes que enfrentan los SE es el de determinar el alcance práctico del término, así como establecer estándares y normas de calidad contra las cuales se pueda evaluar a los presuntos "expertos"; pero sin olvidar que básicamente un SE:

- Descansa su poder en el uso de las heurísticas.
- Sus conocimientos son extraídos del experto humano.
- El razonamiento simbólico depende de la forma de representación del conocimiento.

Por conocimiento podemos decir que está formado por descripciones, relaciones y procedimientos, además de conceptos, restricciones y regulaciones del dominio del cual se está manejando; y donde las descripciones incluyen reglas acerca del cómo aplicarlas e interpretarlas en situaciones específicas; mientras que los procedimientos indican las operaciones necesarias para manipular los elementos anteriores.

Entonces, el proceso de adquisición del conocimiento es empleado para crear y actualizar la parte del SE que corresponde a la base de conocimientos, los cuales serán posteriormente interpretados para obtener una solución al problema.

Dicha base de conocimientos puede ser estructurada de varias maneras. Las que analizamos fueron:

- Redes Semánticas, que son redes que representan el conocimiento como una serie de relaciones entre hechos, sus características, clases generales y otros elementos. Generalmente se utilizan es sistemas basados en LISP (o lenguajes derivados de éste), que permiten el manejo de listas. En ocasiones se convierten en. cálculo de predicados para ser empleados en PROLOG.

- Marcos, que son estructuras que aprovechan el hecho de que diversas situaciones o problemas comparten una estructura que puede representarse como un esquema a ser llenado con datos específicos.

- Scripts, que son estructuras que nos permiten representar situaciones que podemos considerar como repetitivas, ya que siempre son realizadas de la misma forma.

Los mecanismos de razonamiento, llamados también máquinas de inferencia, emplean uno o varios enfoques o direcciones. Los más comunes son:

- Encadenamiento hacia adelante (forward), que parte de los hechos y avanza hacia las posibles soluciones.

- Encadenamiento hacia atrás (backward), que trata de hallar elementos que justifiquen una hipótesis.

En todo México, el uso de Sistemas Expertos en empresas es muy pobre. Las empresas siguen utilizando el viejo paradigma de atención a los usuarios de manera presencial ya sea personal o telefónica.

Básicamente, estos han sido los temas tratados en este trabajo, con los cuales hemos pretendido tocar los aspectos más importantes de una de las ramas de las Ciencias Computacionales que desde hace algunas décadas ha tenido un gran auge debido al ambicioso objetivo que pretende, crear sistemas que realicen tareas tan bien o mejor de lo que lo hace un humano:

la Inteligencia Artificial.

GLOSARIO

Conocimiento declarativo,
Memoria declarativa

Conocimiento de objetos y hechos. Tambien nombrada como memoria declarativa, la cual incluye el conocimiento sensorial. El conocimiento declarativo es esencial tanto para interpretar al mundo externo como tambien para ubicar su propio yo en contexto. La memoria declarativa es la que almacena conocimientos declarativos.

Conocimiento,
Representación del conocimiento

1. Se denomina conocimiento al resultado, por parte del humano, de la maduración semántica de la información y su comprensión experiencial.2:: En inteligencia artificial, una técnica, una manera consistente y útil de organizar la información en la computadora cuyo objetivo es facilitar su procesamiento. Se lo denomina KR (knowledge representation). Entre los esquemas de KR aparecen reglas de lógica simbólica, frames, redes semánticas y gráficos conceptuales. El tema general se denomina "modelado del conocimiento" (knowledge modeling).

Heurística

De acuerdo con ANSI/IEEE Std 100-1984, la heurística trata de métodos o algoritmos exploratorios durante la resolución de problemas en los cuales las soluciones se descubren por la evaluación del progreso logrado en la búsqueda de un resultado final. Se suele usar actualmente como adjetivo, caracterizando técnicas por las cuales se mejora en promedio el resultado de una tarea resolutiva de problemas (parecido al uso de "método óptimo").

Inducción y deducción

La inducción se refiere al movimiento del pensamiento que va de los hechos particulares a afirmaciones de carácter general.

La deducción es el método que permite pasar de afirmaciones de carácter general a hechos particulares. Proviene de deductio que significa descender.

Inferencia

Proceso mental por el cual se extraen conclusiones a partir de premisas más o menos explícitas, proceso resultante ya sea del "sentido común", ya sea de silogismos

informales y formales, o bien por aplicación de las reglas muy detallistas y cálculos de la inferencia estadística. Un ejemplo trivial de inferencia es el de escuchar decir: "Después de un debate cansador, el orador se dirigió a su silla" e inferir (llegar a la conclusión) que se sentó en ella. Este es un mecanismo de "cebado semántico", con pistas suficientemente claras para redondear el significado de una percepción más o menos hipotética. Con ese mecanismo no es "cuesta arriba" deducir lo no explicado en detalle, aclarar la hipótesis. Muy en general, tal inferencia es falsa. Hace falta que se señale que el orador se sentó. En caso contrario, la "inferencia" puede dejar de ser operativa. Por eso se suele señalar que no hay que atreverse a creer en la inferencia salvo si se la activa o ceba semánticamente con preguntas tales como "Adivine lo qué pasó después"

El justificativo es que el lenguaje acota o restringe la acción sólo débilmente, por lo cual es prudente terminar de escuchar lo que sucede a continuación.

Inteligencia

1. La inteligencia es la habilidad de razonar, deducir, inferir, adivinar y pedir perdón al equivocarse.

2. Inteligencia es capacidad de adaptación al entorno (Randall Beer) y tiene que ver con enfrentar alarmas y seudoalarmas.

3. (Allen Newell) La inteligencia se mide teniendo en cuenta el valor del cociente entre la cantidad de información ya existente revisada en una toma de decisiones y la información total guardada. En un ejemplo de este autor, un termostato automático tiene dos informaciones, calefaccionar si el objeto controlado está frío y no calefaccionar en el otro caso. Tiene dos reglas y las revisa antes de cualquier decisión. Tiene el máximo nivel de inteligencia, aunque sumamente reducida en su panorama. Newell espera algo similar de sistemas inteligentes más generalizados. Este enfoque combina en adecuadas proporciones el "case-based reasoning" o sea el razonamiento basado en casos o reglas ya memorizados ("preparación") con la habilidad combinatoria y creativa de factores ("deliberación").

Juicio.

El juicio es el acto intelectivo por el que se afirma o se niega algo de alguna cosa. En forma general, el juicio tiene el esquema "Sujeto es Predicado"; donde el verbo ser tiene una importancia particular, ya que envía siempre a la cuestión del ser de las cosas, al ámbito metafísico.

Es posible que se dé un juicio en forma negativa: "Sujeto no es Predicado"; por lo tanto, el juicio es un acto por el que se agregan o separan dos conceptos.

Lógica

Derivado del griego clásico logos (la razón, principio que gobierna al universo): un conjunto de reglas usadas para gestionar inferencias creíbles. Aristóteles recomienda una lógica dicotómica, verdadero-falso. Los filósofos orientales se inclinan más bien a usar una lógica difusa multi-valorada. Ambas técnicas se están usando para modelar los procesos cognitivos humanos en la computadora.

Modus ponens

Una regla de inferencia usada para probar la corrección o valor de verdad de proposiciones lógicas. Sean dos afirmaciones cualesquiera A y B. La forma de la regla es:

A=>B, Si A => B

Se sabe que si A es verdad, B tambien es verdad; además se sabe que A es verdad; con lo cual se infiere que B es verdad. Lo cual se escribe simbolicamente así: A => B, A. En un lenguaje más explícito se presenta una premisa 1 que es SI A, ENTONCES B acompañada de una premisa 2 que es A.

Modus tollens

Regla de inferencia usada para probar la corrección o el valor de verdad para proposiciones lógicas. Sean A y B dos afirmaciones genéricas cualesquiera. La forma de la regla puede abarcar A, B y otras más. Aquí se ejemplifica con A y B:

A=>B, Si ~B => ~A

En lenguaje más explícito se podría señalar una premisa 1 que es si A entonces B y una premisa 2 que es no B, entonces la conclusión es no A. Dicho de nuevo, se sabe que cuando A es verdad, B es verdad. Se sabe tambien que B no es verdad. Se puede inferir que A no es verdad.
Interesante es el caso simétrico de una doble negación, otra forma que contiene solamente A y B:

~A=>B, Si ~B => A

Razonamiento.

El razonamiento es una actividad psíquica de orden cognitivo por la que se asocia un sujeto a un predicado, cuyo nexo no es del todo evidente. Un ejemplo sería: "Los sistemas operativos que no se bloquean son sistemas robustos, es así que Linux no se me ha bloqueado, por lo tanto Linux es un sistema operativo robusto". Es decir, el razonamiento da como resultado un juicio, pero no en virtud de la evidencia inmediata entre el sujeto y el predicado, sino en razón de un nexo necesario que el intelecto aprehende por medio de la verdad supuesta de los juicios dados.

Relación causal

Un método para definir el conocimiento consiste en representar la causa y el efecto como entidades relacionadas. Una relación causal enlaza las causas con los efectos de forma comprensible como si fuera un teorema lógico (por ejemplo, si llueve, entonces las plantas van a crecer).

Sentido común

Se entiende que se trata de las habilidades mentales compartidas por la mayoría de la gente. Es lo opuesto al conocimiento del "experto" pero tiene sus propias dificultades, ya que requiere de conocimientos de muchos tipos. Se trata de una gran masa de material cognitivo prerrefinado y preanalítico, que le sirve a cada uno de base para conocimientos más refinados, analíticos, críticos y expertos. Cada uno de los componentes que forman el sentido común es en sí dudoso y discutible, pero la masa de todos ellos, que forma un conocimiento dogmático, es bastante menos dudoso porque dichos componentes se complementan mutuamente. (Minsky, 1988, p 327; Pepper,1966) .

Sintaxis y Semántica

Sintaxis es la forma de las expresiones, sentencias y unidades.
Semántica es el significado de estas expresiones, sentencias y unidades.

Taxonomía

"Taxonomía es el estudio teórico de la clasificación, incluyendo sus bases, principios, procedimientos y reglas" (Simpson, 1961).
"Taxonomía es la teoria y práctica de la clasificación de los organismos" (Mayr & Ashlock, 1991).

BIBLIOGRÁFIA

CASTILLO Enrique, ALVAREZ Elena. Sistemas Expertos, Editorial Parainfo, 1989

CHATAIN, Jean-Noel. Sistemas Expertos, Métodos y Herramientas , Editorial Parainfo

DIETER, Nebendahl, Sistemas Expertos, Parte I y II, Editorial Marcombo, 1991

GIARRATANO Joseph. Sistemas Expertos Principios y Programación, Editorial Thomson Editores, 1994

HARMON, Paul, KING, David. Sistemas Expertos: Aplicaciones de la Inteligencia Artificial en la actividad empresarial, 1988

LINDSAY, Susan. Practical Application of Expert Systems, Edit Wesley, Massachusetts, 1988

NEGRETE, José. De la Filosofía a la Inteligencia Artificial, Grupo Noriega Editores

NILSSON, Nils. Artificial Intelligence, A New Synthesis , Morgan Kaufmann Publishers, Inc., 2001

RICH, Elaine, Knight, Kevin. Inteligencia Artificial, Editorial Mac Graw Hill, Madrid, 1994

ROBINSON, Philliph R. Aplique Turbo Prolog, Editorial Mac Graw Hill

ROLSTON, David W. Principles of Artificial Intelligence and Expert Systems Development, Editorial Mac Graw Hill, 1990

RUSSELL, Stuart J., NORVING Peter. Inteligencia Artificial, un enfoque moderno, Editorial Prentice Hall Hispanoamericana, 1995

SÁNCHEZ Y BELTRÁN, Juan Pablo. Sistemas Expertos: Una metodología de programación, Editorial Macrobit, México, 1990

TURBAN, Efraim. Expert Systems and Applied Artificial Intelligence, Editorial MacMillan, USA, 1995

WINSTON, Patrick H. Inteligencia Artificial, Editorial Addison Wesley Iberoamericana, 1984

Inteligencia Artificial: Conceptos, técnicas y Aplicaciones, Serie Mundo Electróncio., Editorial Marcombo, Barcelona-México, 1987.

http://www.psycologia.com/articulos/ar-jsamper01.htm
Juan José Samper Márquez. Universidad de Granada

http://apolo.us.es/josera/prolog/1intro.html

http://www.fepafem.org/conferenciaint/viernes/myriamserrano.htm

http://www.unav.es/asignaturas/ia/tsld015.htm

http://www.control-automatico.net/recursos/articulos/art085.htm
Sergio Moriello, Red Internacional de Control Automático, Colombia.

http://www.lafacu.com/apuntes/ingenieria/sist_expe/
Joaquin Cárdenas Fernández

http://home.worldonline.es/jmariocr/index.htm

http://microelec.uab.es/~ferran/Tesis_abstract.html

http://www.uhu.es/nieves.pavon/pprogramacion/temario/tema1/tema1.html