UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

Unidad Académica de los Ciclos Profesional
y de Posgrado del Colegio de
Ciencias y Humanidades

"EDA: Un sistema tutorial inteligente en el área de estructuras de datos".

T E S I S

Para obtener el título de:

MAESTRO EN CIENCIAS DE LA COMPUTACION

P R E S E N T A:

ARTURO OBREGON SANCHEZ

Director: DR. SERGIO MARCELLIN JACQUES

MEXICO, D. F. **1989**

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

CONTENIDO

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCION

Es tradicional que en las normas generales empleadas en la enseñanza y el aprendizaje, los profesores ofrezcan e impartan la instrucción a sus alumnos por medio de cursos, pláticas informales, notas escritas en el pizarrón, materiales audiovisuales como películas, discos, acetatos y diapositivas. Estos materiales facilitan y ayudan la labor de enseñanza del profesor.

Los estudiantes trabajan "por su cuenta" cuando leen un libro de texto, resuelven problemas, redactan informes, consultan trabajos o libros de la biblioteca o practican en laboratorios o talleres. La interacción entre el maestro y sus alumnos y entre los mismos estudiantes toma generalmente la forma de discusión, a base de preguntas y respuestas.

Estos tres procedimientos - impartir, estudiar independientemente e interacción entre maestro y discípulos - son los métodos didácticos considerados como fundamentales.

Al impartir o explicar una clase el profesor, muestra, presenta, demuestra o desarrolla el tema o materia objeto de la enseñanza, a un grupo de estudiantes, da por supuesto que todos los estudiantes están adquiriendo el mismo grado de conocimiento, al mismo nivel de compresión y al mismo tiempo.

Sin embargo, sabemos que no es así en la realidad. Cada estudiante asimila a su ritmo particular y a su paso o grado de comprensión. Ahora, ¿ qué sucede cuando por alguna razón el alumno se ausenta de clase ? ¿ Se le proporciona alguna alternativa a este tipo de estudiantes para no perder el "hilo" de la clase ?

El aprendizaje por medio de una máquina fue implementado y expandido por Skinner en Harvard, en la década de los cincuentas. Primeramente se tuvo la idea de Instrucción Programada, en donde el material didáctico era dividido en módulos, los cuales se presentaban en un orden específico. Después de la presentación de un módulo, se hacía una pregunta al alumno. Si la respuesta que se proporcionaba era correcta entonces se pasaba a presentar el siguiente módulo, de lo contrario se presentaba el módulo nuevamente, a manera de repaso.

Los primeros sistemas de Instrucción Programada presentaban unicamente textos. Posteriormente se aplicaron películas y diapositivas. Desgraciadamente, las máquinas para la enseñanza en los años cincuentas y sesentas, no llegaron a ser populares dadas las carencias de equipo y material.

Al llegar las computadoras se tuvo un dispositivo con mayor flexibilidad para la Instrucción Programada. Fué entonces que nació lo que hoy se conoce como Educación Asistida por Computadora. EAC es en general, un sistema de instrucción individualizada que utiliza un programa de computadora como medio esencial para el aprendizaje. La idea central es usar a la máquina como una herramienta auxiliar presentando textos y gráficas en una secuencia determinada e interactuando con el alumno a través de preguntas y respuestas. Desafortunadamente, este tipo de instrucción no prolifero como se esperaba por razones como las siguientes:

a) Utilizar a la computadora únicamente como un dispositivo de instrucción programada ha resultado relativamente caro.

b) Implica una politica de instrucción individualizada y no grupal.

c) La producción de este tipo de material implica una especialización en la materia por parte del programador.

d) Dada la cantidad de computadoras diferentes, no se habia tenido material estándar altamente comercial.

A partir de la década de los ochentas, con la proliferación de microcomputadoras, da inicio a sistemas de inteligencia artificial aplicados a la elaboración de sistemas educativos. A estos sistemas, se les conoce como Sistemas Tutoriales Inteligentes.

Estos sistemas aplican principios de inteligencia artificial, generando diálogos en lenguaje natural, haciendo uso de métodos de inferencia y utilizando estructuras y tecnicas de representación de conocimiento, tanto de la materia que se va a enseñar como del conocimiento del estudiante.

En México, la idea de utilizar a la computadora como auxiliar en el proceso de enseñanza-aprendizaje, ha captado la atención tanto de instituciones educativas como de investigación, quienes vislumbran, cada uno desde su particular punto de vista, el uso de la computadora como un elemento de gran apoyo y utilidad en la educación, siendo cada dia más los trabajos desarrollados y las investigaciones referentes a este campo de la inteligencia artificial.

El presente trabajo de tesis tiene los siguientes objetivos principales:

a) Definir claramente el concepto de sistema experto, describiendo todos y cada uno de sus componentes.

b) Mostrar las características, elementos, ventajas y desventajas de los sistemas tutoriales inteligentes, especificando el vínculo que existe con los sistemas expertos.

c) Crear un sistema tutorial inteligente en el área de estructuras de datos, concretamente en el tema de algoritmos de ordenamiento y búsqueda.

Para lograr los objetivos anteriormente señalados, el desarrollo de la tesis quedó estructurado en 6 capítulos que a continuación se describen:

Capítulo I.- Inteligencia Artificial y Sistemas Expertos.

El capítulo I define el concepto de Inteligencia Artificial, sus objetivos centrales y algunas razones por las que complementa a otras áreas.

Se incluye una reseña histórica que abarca desde su origen en 1956 a la fecha, diferenciando tres épocas:

a) Etapa inicial.- (1956 a 1970) Que comprende la creación de técnicas básicas para representar el comportamiento inteligente. En esta época se desarrollaron dos líneas de investigación: Los métodos de búsqueda heurística y los métodos de deducción automática.

b) Etapa de prototipos.- (1970 a 1981) En la que se desarrollaron proyectos más complejos pero todavía limitados a centros de investigación. Los proyectos más importantes fueron: Programas inteligentes controladores de robots, sistemas de comprensión en lenguaje natural, sistemas expertos, desarrollo de la programación lógica y optimización operativa de los entornos LISP.

c) Etapa de difusión.- (1981 a la fecha) En la que se desarrollan proyectos comerciales por parte de empresas privadas. Las aplicaciones más importantes son aquellas que giran alrededor de los sistemas basados en conocimiento. Se incluye una relación de las cinco empresas más importantes en los Estados Unidos que ofrecen soluciones comerciales.

En materia de sistemas expertos, se incluye la definición de experto (de carne y hueso), sus características y el tipo de conocimiento que maneja. Como sistema experto se incluye una definición, una arquitectura típica; el término de ingeniero del conocimiento y la diferencia entre sistemas expertos y programas convencionales.

Capítulo II.- Conocimiento

Este capítulo inicia a partir de la descripción filosófica del conocimiento, tocando cinco partes fundamentales: La posibilidad, el origen, la esencia, las formas y el problema de la verdad. Dentro de este contexto formal ubicamos la parte que a la Inteligencia Artificial le interesa, siendo ésta la que se refiere a la naturaleza del conocimiento debido a su carácter pragmático, en donde la finalidad no es descubrir verdades teóricas sino actuar en la realidad.

Se explica la conexión natural que existe entre las bases de datos y las bases de conocimientos y la separación en dos ramas principales de las técnicas para representar el conocimiento: procedurales y declarativas. Se incluyen las características de cada una de ellas.

Posteriormente se describen en forma detallada las técnicas más comunes de representación de conocimiento: Autómatas finitos, programas, lógica simbólica, sistemas basados en reglas de producción, redes semánticas, marcos y guiones.

Se complementa el capítulo con otras representaciones no convencionales que se basan en el reconocimiento de patrones y no en el razonamiento. Estas técnicas son el algoritmo ID3 y el clasificador Holland.

Capítulo III.- Aprendizaje.

El capítulo III comienza con definición de aprendizaje y una breve discusión de la relación que existe entre la adquisición de conocimientos y su almacenamiento mediante una representación.

Para describir el aspecto humano de aprendizaje y poder contestar a la pregunta ¿Cómo se adquiere el conocimiento? se incluye una descripción del aprendizaje infantil, el aprendizaje adulto y el aprendizaje mediante tutorías.

El aprendizaje en un sistema experto incluye la descripción del proceso de aprendizaje y una descripción de las distintas clases de aprendizaje.

a) Aprendizaje por memorización.
b) Aprendizaje por instrucción o algorítmica.
c) Aprendizaje por analogía.
d) Aprendizaje a partir de ejemplos y contraejemplos.
e) Aprendizaje a partir de ensayo y prueba.

Capitulo IV.- Sistemas Expertos.

Este capitulo inicia con una clasificación del camino de novato a experto. Se identifican cinco etapas: Novato, principiante avanzado, competencia, perito y experto. Se describen en forma detallada las caracteristicas de cada una de estas etapas.

En materia de sistemas expertos se incluye:

- Una definición (integrada de un conjunto de nueve definiciones, mismas que se incluyen en el anexo 1).
- Los objetivos comunes a todos los sistemas expertos.
- Los elementos básicos que conforman a los sistemas expertos:

  a) Base de conocimientos.
  b) Interfaz con el usuario.
  c) Interfaz con el experto.
  d) Mecanismo de razonamiento.
  e) Espacio de trabajo.

- Una clasificación basada en su tipo de función:

  a) Interpretación.
  b) Predicción.
  c) Diagnósticos.
  d) Diseño.
  e) Planeación.
  f) Monitoreo
  g) Depuración.
  h) Reparación.
  i) Instrucción.
  j) Control.

  Los mecanismos de razonamiento más comunes agrupados en siete modalidades:

  a) Elección de una dirección de solución.- Encadenamiento hacia adelante, encadenamiento hacia atrás, encadenamiento hacia adelante y hacia atrás y manejo de eventos.
  b) Razonamiento en presencia de incertidumbre.- Procedimientos numéricos y revisión de credibilidad.
  c) Búsqueda en un espacio pequeño.
  d) Búsqueda en un espacio grande.- Generación y prueba jerárquica y líneas múltiples de razonamiento.

e) Métodos para manejar un espacio grande por transformación de espacios.- Rompimiento del problema (Secuencia fija de subproblemas, el mínimo compromiso, propagación de restricciones, conjeturas y razonamientos plausibles), refinamiento jerárquico y resolución jerárquica dentre de subespacios que colaboran.
f) Métodos para manejar un espacio grande por desarrollo de alternativas o espacios adicionales.- Empleo de modelos múltiples y meta-razonamientos.
g) Tratamiento con el tiempo.- Cálculo situacional y planificación con restricciones de tiempo.

En base a una investigación bibliográfica se presentan en el anexo 2 un total de 18 diagramas de sistemas expertos.

Capítulo V.- Sistemas ICAI.

En el capítulo V, en materia de sistemas ICAI se incluye:

- Su definición.
- Las áreas de estudio que conforman a este tipo de sistemas.
- Sus componentes principales:

  a) Módulo experto.
  b) Modelo del estudiante.- De sobreposición y depuración.
  c) Módulo tutorial.- Socrático y de entrenamiento.

- Una clasificación de los tipos de sistemas ICAI:

  a) Tutores mezclados con iniciativas.
  b) Tutores de entrenamiento.
  c) Tutores de diagnóstico.
  d) El concepto de micromundos.
  e) Sistemas expertos.

- Una reseña histórica que abarca desde sus orígenes en los años 20's a la fecha. Esta reseña concluye con una tabla que muestra los desarrollos ICAI más importantes.
- Una propuesta acerca de como deben ser los sistemas ICAI del futuro.

En base a una investigación bibliográfica se presentan en el anexo 3 un total de 7 diagramas de sistemas ICAI.

Capitulo VI.- Descripción del sistema EDA.

En este capitulo se describe el proyecto EDA que se presenta como un sistema tutorial inteligente en el área de estructuras de datos; en él se incluye:

- Las consideraciones generales entorno al proyecto:

  a) Estrategia de aprendizaje.
  b) EDA: Un sistema experto o un sistema ICAI ?

- La descripción funcional del sistema EDA:

  a) Módulo de datos,
  b) Módulo de ordenamiento,
  c) Módulo de búsqueda,
  d) Módulo de algoritmo,
  e) Módulo de simulación, y,
  f) Módulo de animación gráfica.

- Los detalles técnicos de su implantación.

El trabajo termina con comentarios acerca de la experiencia adquirida durante el desarrollo de este trabajo y con sugerencias a posibles extensiones del sistema EDA.

Se incluyen todas las referencias bibliográficas de los trabajos consultados.

CAPITULO I

INTELIGENCIA ARTIFICIAL Y SISTEMAS EXPERTOS

Introducción

El objetivo que se persigue en este primer capítulo es dar un breve panorama histórico de la técnicas básicas y realizaciones en el campo de la Inteligencia Artificial a lo largo de sus casi treinta años de historia, así como, definir los conceptos fundamentales de Inteligencia Artificial y Sistemas Expertos.

1. ¿Qué es Inteligencia Artificial?

Según Winston [WINSTON 1984], la Inteligencia Artificial es:

> "el estudio de las ideas que permiten a las computadoras ser inteligentes".

También enfatiza que "los objetivos centrales de la Inteligencia Artificial son el hacer a las computadoras más útiles, así como comprender los principios que hacen posible la inteligencia".

¿Pero qué es inteligencia? ¿Es la habilidad de razonar? ¿Es la habilidad de adquirir y aplicar conocimiento? ¿Es la habilidad de percibir y manipular cosas en el mundo físico?

Seguramente todas estas habilidades forman parte de lo que se conoce como inteligencia, pero no podemos decir que sean todas. Una definición en el sentido usual resulta imposible, dado que la inteligencia es una mezcla de muchas habilidades sobre todo en la representación y el procesamiento de información.

La perspectiva de la Inteligencia Artificial complementa las perspectivas de la psicología, lingüística y filosofía. Algunas razones son:

- El estudio del conocimiento por computadora ayuda al

pensamiento. Trabajar con computadoras ha conducido a un nuevo lenguaje, que sirve para pensar cómo hacer y describir cosas. El uso metafórico y análogo de los conceptos involucrados ha conllevado a un nivel de pensamiento superior: un pensamiento acerca del pensamiento.

- Desarrollar una teoría por computadora permite detectar errores conceptuales y casos extraordinarios que por lo general escapan aún hasta del más meticuloso investigador. Los principales obstáculos frecuentemente surgen como si nunca se hubieran reconocido como problemas ántes de iniciar el ciclo de experimentación.

- Los desarrollos en computadora cuantifican los requerimientos de tareas. Una vez que un programa desarrolla una tarea específica, por medio de instrucciones externas al programa se puede obtener cuanto procesamiento de información requiere la misma.

- Los programas de computadora exhiben paciencia ilimitada y no requieren alimentación. Además, es sencillo privar a un programa de computadora de una pieza de conocimiento con el fin de probar qué tan importante es esa pieza realmente. Una de las principales ventajas de la simulación por computadora es que se puede modificar arbitrariamente las variables que en el mundo real es imposible controlar.

No es lo mismo que "las computadoras sean inteligentes" a que "las computadoras simulen inteligencia". La Inteligencia Aritificial estimula a la gente que quiere descubrir los principios que todos los procesadores de información inteligentes deban explotar.

Consecuentemente, no existe obsesión alguna con imitar la iteligencia humana ni, por otra parte, prejuzgar en contra del uso de métodos que aplica la inteligencia humana. En su lugar, existe un punto de vista que genera nuevas metodologías e induce nuevas teorías.

Como resultado de este nuevo punto de vista se han generado nuevas ideas acerca de como ayudar a la gente a ser "más inteligente". Así como el conocimiento psicológico acerca de cómo los humanos procesan información puede llegar a crear computadoras "inteligentes", las teorías derivadas del uso de computadoras, frecuentemente sugieren métodos para educar mejor a la gente. Dicho de otro modo, la metodología involucrada en crear programas "inteligentes" puede transferirse para hacer a la gente más "lista".

2. Reseña Histórica

El origen de la Inteligencia Artificial (nombre debido a John McCarthy) se remonta a la conferencia sobre teoría informática que tuvo lugar en 1956 en el Darmouth College. En ella aparecieron sistemas con capacidad para desarrollar juegos (juego de damas de Samuels) y demostrar teoremas (The logic theorist de Newell y Simon) creándose el nombre de Inteligencia Artificial y el concepto de Sistemas Inteligentes.

Aunque el ritmo de desarrollo de la Inteligencia Artificial ha sido inferior a lo previsto, se han cumplido algunas de las expectativas de estos primeros momentos; de hecho, en el año de 1981 en la Conferencia Mundial de Vancouver sobre Inteligencia Artificial, se celebró un homenaje a los pioneros de aquella conferencia de Darmouth con motivo de que la Inteligencia Artificial había cumplido 25 años. En esta conferencia se contó con cerca de 2000 asistentes presentándose trabajos tanto de software como de hardware.

En el desarrollo de conocimientos que constituyen hoy la Inteligencia Artificial se pueden diferenciar tres épocas:

- Etapa primaria.- Que comprende la creación de técnicas básicas para representar el comportamiento inteligente tanto a nivel de métodos como de lenguajes. Este período abarca de 1956 a 1970.
- Etapa de prototipos.- En la que se desarrollan proyectos más complejos pero todavía limitados a centros de investigación. Este segundo período abarca de 1970 a 1981.
- Etapa de difusión industrial.- Se desarrollan proyectos comerciales por parte de empresas privadas. Este tercer período inicia a partir de 1981.

A continuación se comentan las características de cada etapa.

2.1. Etapa Inicial

En esta época, con el objeto de representar el comportamiento inteligente, se desarrollaron dos líneas de investigación:

- Los métodos de búsqueda heurística.

- Los métodos de deducción automática.

Con los métodos de búsqueda heurística la resolución de un problema se concibe como la búsqueda, en un espacio de estados posibles, de la secuencia de operaciones de transformación que conducen de la situación inicial o dato del problema a una situación que cumple con las condiciones exigidas como solución del problema; la conducción de este proceso de búsqueda se realiza con criterios restrictivos (heurísticos) basados en el conocimiento sobre el tema.

Los métodos de búsqueda heurística se utilizan también para la descomposición de problemas en otros más simples de forma que, partiendo de una formulación del problema, una búsqueda sistemática encuentra el conjunto de subproblemas cuya resolución asegura la del problema inicial, y cada uno de los subproblemas es resoluble por las operaciones elementales dadas.

La formalización de los procesos de búsqueda heurística se realizó por Newell Shaw y Simon en las denominadas reglas de producción, constituidas por pares situación-acción:

- Características de situación del universo en que se describe el problema que verifican el antecedente.

- Conjunto de operaciones básicas a realizar.

La dinámica de aplicación de un conjunto de reglas consiste en un ciclo repetitivo, constituido de dos pasos:

- Paso de selección de posibles acciones, para las que se comparan los antecedentes de reglas y el estado actual del problema, de manera que se obtiene el conjunto de reglas tales que se cumple la condición en el estado intermedio de resolución del problema que se trata.

- Paso de decisión y acción en donde se aplica la regla más conveniente según el criterio heurístico seleccionado, modificándose el estado actual.

En las técnicas de deducción automática se modela el conjunto de condiciones descriptivas de un problema y los mecanismos de transformación mediante fórmulas de cálculo de predicados. Un problema es resoluble si sus especificaciones son deducibles de las premisas del problema, en cuyo caso la descripción del

proceso de deducción es el método de resolución del problema.

Una de las líneas de investigación más activas de esta etapa fue la utilización de los resultados teóricos de la lógica de primer orden obtenidos hasta los años treinta. Como consecuencia de este proceso, Robinson en 1965 formuló la regla universal de inferencia de resolución y unificación, que permitía reducir el problema de deducción automática a una búsqueda de resoluciones entre cláusulas.

Como métodos intermedios entre ambas técnicas aparecen los basados en redes semánticas. Fueron introducidas en 1968 por Quillian para representar las relaciones formuladas con predicados mediante gráficas con arcos etiquetados. Los métodos de resolución de problemas basados en esta representación son procesos de búsqueda con restricciones asociadas a los tipos de arco de la red.

Finalmente, en esta etapa inicial se integra definitivamente el lenguaje LISP como vehículo de formulación tanto de representaciones como de procesos inteligentes. Este lenguaje creado por John McCarthy, en su primera versión se basa en una estructura de lista tanto para la información como para el programa, da manera que un proceso podría generar un programa y evaluarlo directamente. LISP que no fue adoptado por ninguna marca de computadoras, se concibió casi en la misma época que FORTRAN y se ha mantenido exclusivamente para la comunidad de especialistas en Inteligencia Artificial hasta la fecha.

En esta época las aplicaciones desarrolladas fueron fundamentalmente juegos y rompecabezas, ya que las limitaciones de equipo y la prioridad dada a la investigación de métodos aconsejaba este tipo de aplicaciones cuyo papel era principalmente ilustrativo.

2.2. Etapa de Prototipos

A principios de los setenta, cuando ya se contaba con una serie de técnicas probadas, se desarrollaron una serie de proyectos de mayor alcance que los realizados hasta el momento con objeto de estudiar las posibilidades de aplicación industrial. Los proyectos más importantes fueron:

a) Programas inteligentes controladores de robots.- Es decir, sistemas de programación de robots a partir de las especificaciones generales de las tareas a realizar. Se desarrollaron proyectos de este tipo en el Instituto de Stanford, el Instituro Tecnológico de Massachusetts y la Universidad de Edimburgo. Son notables los desarrollos de los sistemas STRIPS y NOAH, válidos tanto para programación de

robots como para asesorar a personas en el desarrollo de tareas.

b) Sistemas de comprensión en lenguaje natural.- Se desarrollaron diversos proyectos capaces de entender subconjuntos del lenguaje natural. Estos sistemas cuyo objetivo es ponderar la ambigüedad de las afirmaciones del lenguaje de las personas produce representaciones precisas para su proceso por computadora. Hendrix las clasifica en dos niveles:

   - Sistemas de interfaz con bases de datos, que requieren manejar los conceptos y estructuras utilizados por dichas bases. Utilizando una representación de redes semánticas estructuradas fue posible realizar sistemas con un nivel de comportamiento aceptable. Ejemplo de este tipo de sistemas son: LADDER, EUFID, LUNAR.

   - Sistemas que incluyen un modelo lógico del mundo, que les permite completar los aspectos ambiguos del lenguaje mediante razonamientos sobre este modelo. De ellos se han realizado sistemas experimentales como es el sistema SHRDLU que opera en un micromundo de objetos geométricos al cual el sistema es capaz de contestar las preguntas en lenguaje natural, deduciendo las respuestas o detectando las contradicciones y ambiguedades.

c) Sistemas Expertos.- A partir de 1977 se comenzaron a desarrollar programas sofisticados de computación que manipulan conocimientos de expertos para resolver de una manera eficiente problemas de una área especifica, tal como lo hacen los expertos humanos. Los sistemas expertos son creados para actuar como asistentes "inteligentes" en los procesos de toma de decisión. La descripción de este tipo de sistemas se profundiza en la sección tres de este capitulo.

d) Desarrollo de la programación lógica.- La programación lógica se creó a partir de los trabajos de Colmerauer, Roussel y Kowalski que condujeron a la primera versión del lenguaje PROLOG en 1975. De hecho, los resultados de la etapa inicial en el área de deducion automática hicieron posible la construcción de sistemas que, a partir de la formulación del conocimiento base para la resolucion de un problema en forma de cláusulas y la formulación de un pregunta-problema como conjunción de condiciones a satisfacer, contestan con respuestas que se obtienen a partir de las especificaciones del problema.

   PROLOG representa un cambio cualitativo en el enfoque de la programación que pasa de expresar el "cómo" resolver un problema (lenguajes procedurales) a expresar el "qué"

(conocimiento para resolver un problema).

e) Optimización operativa de los entornos LISP.- A partir del concepto inicial de LISP se desarrollaron entornos de diseño específicos basados en la experiencia con aplicaciones. Los más importantes son INTERLISP y MACLISP. El rasgo más característico de esta optimización es la construcción en 1979 por el MIT de un chip con las funciones LISP básicas que estiman será la base para la construcción de una línea específica de computadoras: las máquinas LISP.

2.3. Etapa de Difusión Industrial

Actualmente, la característica principal de esta etapa es la aceptación de los proyectos de Inteligencia Artificial como una actividad informática profesional. A partir de 1981 las aplicaciones basadas en estas técnicas no unicamente son desarrolladas por centros de investigación (como Stanford, MIT, Carnegie Mellon, Rutgers, etc.) sino también por empresas profesionales.

A finales de 1984 ya existían 151 empresas en el mundo dedicadas al diseño de sistemas de Inteligencia Aritificial y tres compañías que desarrollaban hardware basado en las máquinas LISP.

Las aplicaciones más importantes son aquellas que giran alrededor de los sistemas basados en conocimiento. En esta etapa se están desarrollan actividades que tienden a:

- Crear entornos de programación que faciliten la construcción y validación de bases de conocimientos.

- Aplicación de estas bases de conocimiento para la resolución de problemas tanto administrativos como técnicos.

- Interfaces en lenguaje natural que permita la fácil relación de los expertos con la base de conocimiento.

Aunque todavía a la espera de una implantación generalizada de la robótica industrial, se está trabajando en la creación de sistemas de robóts inteligentes, capaces de captar y manipular datos a partir de mecanismos de visión y sensores y autoprogramar la secuencia de movimientos para realizar una tarea a partir de especificaciones sencillas.

Finalmente, se puede mencionar que actualmente (1988) las cinco compañías más importantes en los Estados Unidos que ofrecen

soluciones comerciales son [KOSLOV 1988]:

- AION.- Esta empresa ubicada en Palo Alto (CA) ofrece su sistema de desarrollo Arion enfocado a sistemas mainframe de IBM.

- FIRST CLASS EXPERT SYSTEMS.- Ubicada en Wayland (MA) y ofrece sistemas de desarrollo "1st Class" basados en computadoras personales.

- GOLD HILL COMPUTERS.- Ubicada en Cambridge (MA) y ofrece herramientas de desarrollo de sistemas expertos basadas en el lenguaje de programación LISP enfocado a computadoras personales.

- INTELLIGENT TECHNOLOGY GROUP.- Ubicada en Pittsburgh (PA) y ofrece servicios de procesamiento de datos a corporativos basado en la técnica de sistemas expertos.

- NEURON DATA.- Ubicada en Palo Alto (CA) que ofrece su producto Nexpert para la Macintosh, computadoras personales IBM, DEC's VAX, Apollo, Sun y estaciones de trabajo Hewlett-Packard.

3. Sistemas Expertos

El concepto de Sistemas Expertos fue introducido en una conferencia sobre Inteligencia Artificial en 1977 por Feigenbaum, con la siguiente frase:

> "El poder de un sistema experto deriva del conocimiento que posee y no de los formalismos o mecanismos de inferencia que emplea" [HAYES-ROTH 1983].

Esta frase indicó un cambio importante en la perspectiva de muchos investigadores del área de Inteligencia Artificial. El primer grupo de investigadores estaba dominado por una ingenua creencia de que, con pocas leyes de razonamiento acoplado a computadoras poderosas, se produciría un desempeño experto y superhumano. La experiencia demostró que esta estrategia para la resolución de problemas de propósito general no servía para resolver la mayoría de los problemas complejos. Como reacción a estas limitaciones, muchos investigadores comenzaron a trabajar en problemas de aplicaciones estrechamente definidas.

A mediados de los 70's, los pocos investigadores que reconocieron

el papel central del conocimiento en estos sistemas, iniciaron una serie de esfuerzos para desarrollar teorías comprensivas sobre la representación del conocimiento y su asociación con sistemas de propósito general. Estos esfuerzos tuvieron un éxito limitado porque el "conocimiento" como blanco de estudio era muy amplio y difuso. Sin embargo, la lección aprendida de estas experiencias fue, finalmente, lo que expresó Feigenbaum: "El conocimiento de los expertos provee la clave para lograr un desempeño experto, miéntras que la representación del conocimiento y sus esquemas de inferencia proveen los mecanismos para su uso" [HAYES-ROTH 1983].

¿Qué es un experto? ¿Cuales son sus características? ¿Qué tipos de conocimientos maneja?

Los expertos son personas que resuelven problemas específicos; sus características principales son:

- Tienen conocimientos profundos en su materia.
- Tienen una amplia experiencia en la solución de problemas referentes a dicha materia.
- Llegan a conclusiones, a través de razonamientos, utilizando el conocimiento que poseen o que hayan adquirido.
- Pueden dar una explicación de dichas conclusiones, así como de su comportamiento.

El conocimiento del experto, presenta dos aspectos:

a) Conocimiento formal: Basado en leyes conocidas de la naturaleza (por ejemplo las leyes de la física de Newton) y en métodos formales de razonamiento. Este tipo de conocimiento también es llamado conocimiento público y usualmente está incluido en los libros de texto relativos al tema.

b) Conocimiento informal: Según el juicio, la opinión o el criterio del experto; consta de sugerencias y de cálculos aproximados. Este tipo de conocimiento también es llamado conocimiento privado o heuristico y usualmente se acumula con la experiencia.

Se presume que el mejor experto es aquél que posee y emplea el conjunto más rico de conocimientos tanto formales como heuristicos.

Los Sistemas Expertos son "programas que utilizan tanto el conocimiento formal como el informal para resolver problemas" [MARIK 1987]. Esta es la razón por la cual ellos pueden alcanzar

un comportamiento similar al de los expertos humanos.

Para el desarrollo de un sistema de este tipo, es necesario que el conocimiento sea suministrado por expertos humanos en la materia en colaboración con una persona familiarizada con la técnicas de sistemas expertos. A esta persona se le conoce como Ingeniero del Conocimiento. Todo el conocimiento expresado de manera que pueda ser utilizado por el programa, constituye en la computadora un archivo, que es llamado la base de conocimientos. Al conjunto de procedimientos de razonamiento que se aplican a la base de conocimientos se le denomina la máquina de inferencia.

La arquitectura típica de los sistemas expertos es muy heterogénea; en forma general podemos decir que constan de las siguientes partes:

a) Base de conocimientos, la cual contiene el saber específico en la disciplina de la cual el sistema es experto. Consiste en un conjunto de hechos (datos) y de reglas programadas. No contiene información específica de un problema en particular. El conocimiento plasmado en la base juega el papel más importante en la calidad y habilidad experta basada en una computadora.

b) Interfaz con el usuario, la cual permite aceptar y reconocer un lenguaje de comandos en forma natural y los traduce en instrucciones y datos para que el sistema experto trabaje.

c) Interfaz con el experto, la cual permite captar información del exterior (proporcionada por el experto) e introducirla en forma adecuada en la base de conocimiento.

d) Mecanismo de razonamiento, también llamada máquina de inferencia, el cual permite controlar el sistema. En este módulo se interpretan las reglas contenidas en la base de conocimientos.

e) Espacio de trabajo, en la cual se almacenan resultados, hipótesis y decisiones intermedias, así como el estado en que el problema se encuentra en un momento dado.

¿En qué difieren los sistemas expertos de los programas convencionales?

Los programas convencionales pueden ser divididos en:

a) Un algoritmo, en el cual está contenido todo el conocimiento formal.

b) Datos, a los cuales se les aplica el algoritmo.

c) Mecanismos de entrada y salida, que contienen la información sobre la forma de comunicación con el usuario y la forma de accesar los datos.

Estos programas sólo pueden dar respuestas a problemas para los cuales están especificamente programados. Si el programa necesita ser modificado para incluir nueva información, el programa entero debe ser examinado. Y, para cumplir el fin para el cual fue programado, debe ejecutar una pre-determinada secuencia fija de instrucciones.

Un sistema experto independiza los procedimientos para la solución de problemas de los datos respectivos. Las modificaciones se realizan alterando la base de conocimientos sin afectar la estructura del programa completo. Estos sistemas seleccionan los medios y hechos para obtener una respuesta adecuada a una situación especifica.

CAPITULO II

CONOCIMIENTO

Introducción

La naturaleza del conocimiento e inteligencia ha sido discutida por psicólogos, filósofos, lingüistas, educadores y sociólogos por cientos de años. Debido a que la metodología de investigación en Inteligencia Artificial necesita diseñar programas que muestren un comportamiento "inteligente", los investigadores han adoptado una posición pragmática en lo que se refiere al conocimiento.

El objetivo de este capitulo es analizar el elemento principal de los sistemas expertos - el conocimiento-. Nuestro estudio inicia a partir de la descripción filosófica del conocimiento, tocando cinco puntos fundamentales: La posibilidad, el origen, la esencia, las formas y el problema de la verdad. Dentro de este contexto formal ubicamos la parte que a la Inteligencia Artificial le interesa, siendo esta la que se refiere a la naturaleza del conocimiento.

Una vez delimitado el alcance de la Inteligencia Artificial se hace una síntesis de los modelos de representación de conocimiento más comunes que se utilizan actualmente y se comenta la liga existente con los sistemas para el manejo de bases de datos.

1. Definición Filosófica de Conocimiento

El conocimiento se define como:

> "la acción y efecto de aprender mediante la actividad intelectual, la realidad individual y concreta y las relaciones existentes entre las cosas o conceptos" [SALVAT 1981].

Como todos nuestros conocimientos se formulan en juicios, y éstos tienen sus fuentes ya sea en la percepción sensible o en ciertos axiomas que suponemos evidentes, es preciso averiguar si

efectivamente nuestras percepciones corresponden a la realidad y si los axiomas coinciden también con la realidad. La rama de la filosofía que aspira a aclarar estas cuestiones es la teoría del conocimiento o Gnoseología.

Una de las más graves y profundas cuestiones que se plantea la filosofía es la que se refiere al conocimiento humano. Como nuestro conocimiento es conocimiento de objetos, es decir de cosas que parecen existir independientemente de nosotros, se trata de averiguar si es posible alcanzar a conocerlos en realidad, o bien si existen límites para dicho conocimiento.

Otra cuestión es la que se refiere al origen, o sea a la fuente de nuestro conocer. ¿Deriva el conocimiento de la experiencia o se origina en la razón?

Una tercera cuestión se refiere a la esencia del conocimiento. Se trata de la relación sujeto y objeto. ¿Existe realmente el objeto o es el sujeto quien determina el objeto?

Un cuarto problema es el que se plantea con respecto a las formas del conocimiento. ¿Además del conocimiento racional hay otra clase de conocimiento, un conocimiento intuitivo?

Por último se plantea la cuestión del criterio de verdad. Se trata de tener un signo que nos diga en un caso concreto si un conocimiento que poseemos es verdadero o no.

1.1. Posibilidad del Conocimiento

Este problema se refiere al valor de nuestro conocimiento. Se trata de saber si es posible conocer algo y si ese conocimiento tiene algún valor, se dan seis posturas fundamentales:

a) El dogmatismo; Está posición que es la más antigua, afirma la posibilidad del conocimiento. Considera que el contacto entre el sujeto y el objeto es real, o sea el sujeto es capaz de aprender el objeto. No se preocupa todavía de examinar el poder cognoscitivo de la inteligencia.

b) El escepticismo; Esta doctrina niega que el sujeto pueda aprender el objeto, y tener por consiguiente conocimiento de él. Como extiende la duda a todos los conocimientos, resulta que es preciso abstenerse de formular cualquier juicio. El fundador de esta doctrina fué Pirrón de Elis.

c) El subjetivismo y el relativismo; Estas dos tendencias derivan del escepticismo. Miéntras el escepticismo enseña que no existe un conocimiento verdadero y cierto, el subjetivismo y el relativismo sostienen que hay una verdad pero que su

validez es limitada. Para el subjetivismo la verdad es algo que depende totalmente del individuo de su estructura psicológica. Para el relativismo la verdad es relativa, en vez de radicar en la naturaleza del sujeto depende de factores externos como la época, la influencia del medio, el círculo cultural o político, etcétera.

d) El pragmatismo; Para el pragmatismo lo verdadero significa lo útil, lo valioso, lo que fomenta la vida. La finalidad de la inteligencia no es descubrir verdades teóricas sino actuar en la realidad. De manera que será verdad todo aquello que le permita actuar con congruencia en la vida, todo aquello que le resulte útil y provechoso especialmente para la vida social.

e) El criticismo; Es una posición intermedia entre el dogmatismo que tiene una fé ciega en la razón y el escepticismo que niega toda posibilidad de un conocimiento verdadero. El criticismo examina el poder cognoscitivo mismo. Investiga las fuentes del conocimiento y distingue entre los problemas que puede resolver y aquellos otros que están fuera de su alcance y que por lo tanto permanecen sin solución. El fundador de esta doctrina es Manuel Kant quien considera que lo único que conocemos del mundo son las representaciones, las meras apariencias de los objetos, es decir los "fenómenos" que nuestra inteligencia ordena en el espacio y en el tiempo [FINGERMANN 1976].

f) El positivismo; El positivismo sostiene que existen límites para nuestro saber como para nuestro no-saber. Limita el valor del conocimiento al campo de la experiencia, es decir a los fenómenos y a sus relaciones. Restringe su acción a las ciencias positivas considerando solamente los hechos, puesto que no podemos conocer la esencia de las cosas. El fundador de esta doctrina fué Augusto Comte.

1.2. El Origen del Conocimiento

Se nos plantea aquí cuál es el factor determinante en el conocimiento si la experiencia o la razón, es decir cuales son las fuentes de donde derivan nuestros conocimientos. Las doctrinas más importantes son:

a) El racionalismo; Esta doctrina encuentra en el pensamiento, en la razón, la fuente más importante y más decisiva para el conocimiento. Sólo los juicios fundados en el pensamiento poseen necesidad lógica y validez universal, es decir que el juicio formulado debe ser un resultado forzoso, una consecuencia necesaria y válida en todas partes y para todos los seres racionales. Esta postura ha sido defendida por: Platón, Sócrates, Descartes y Leibniz.

b) El empirismo; El empirismo sostiene la tesis de que la única fuente del conocimiento humano es la experiencia. Todas nuestras ideas y conocimientos aún los más abstractos y generales, derivan de la experiencia. Defienden esta postura John Locke y David Hume. Existe una variante defendida por el filósofo francés Condillac, quién transformó el empirismo en sensualismo porque sólo admite la percepción exterior mediante los sentidos negando la existencia interna.

c) El intelectualismo; Esta dirección trata de conciliar el racionalismo con el empirismo. Considera que ambos factores participan en la producción del conocimiento, sin embargo el conocimiento arranca de la experiencia aunque la razón intervenga. Esta doctrina fue sostenida por Aristóteles y Santo Tomás de Aquino.

d) El apriorismo; Para el apriorismo las únicas fuentes del conocimiento son la experiencia y la razón. Afirma que nuestro conocimiento presenta ciertos conocimientos a "priori" que no derivan de la experiencia. Esta doctrina tiene como fundador a Manuel Kant.

1.3. La Esencia del Conocimiento

Este problema se refiere a la realidad de nuestros conocimientos, a la realidad del mundo exterior. Nos preguntamos ahora cuál es el factor - objeto, sujeto - determinante en el conocimiento. Pueden darse dos tipos de soluciones:

1) Premetafísicas; En las que nada se dice sobre el carácter ontológico del sujeto y del objeto; dentro de este grupo tendríamos:

   a) El objetivismo; Que sostiene la determinación del sujeto por parte del objeto. El sujeto reproduce las cualidades del objeto. Este se encuentra como algo situado en el exterior frente a la conciencia del sujeto. Esta doctrina es defendida por Platón y Edmundo Husserl.

   b) El subjetivismo; Que sostiene que el sujeto determina al objeto. No se trata de un sujeto individual, concreto, sino de un sujeto superior trascendente (por ejemplo Dios). Esta doctrina es defendida por San Agustín, Hermann Cohen y Plotino.

2) Metafísicas; En las que interviene el carácter ontológico del objeto; en este grupo destacan:

a) El realismo; Según el cual existen cosas reales independientes de la conciencia que las conoce. Le parece natural considerar que las cosas que están fuera de nosostros y nosostros mismos, son tales como los percibe la conciencia. A esta postura se le conoce como realismo ingenuo. El realismo critico considera que las propiedades o cualidades de las cosas tales como los colores, los sabores, los sonidos, etc., sólo existen en nuestra conciencia y surgen cuando determinados estímulos actúan sobre nuestros órganos sensoriales. Las percepciones sólo son fenómenos de nuestra conciencia y dependen de la naturaleza de ésta. Defienden esta postura Demócrito, Aristóteles, Galileo, Descartes, Hobbes y Locke.

b) El idealismo; Sostiene la tesis de que no existen cosas reales independientemente de la conciencia que conoce. La única realidad son nuestras percepciones, nuestra conciencia con sus contenidos, por esta razón se llama también a esta posición conciencialismo o idealismo subjetivo (psicológico). El idealismo objetivo ó lógico toma como punto de partida la objetividad de la ciencia. Esta objetividad se hace manifiesta en la conciencia como una serie de pensamientos, de juicios, de raciocinios y no por un cúmulo de procesos psicológicos. Defienden a esta postura Berkeley, Fichte y Hegel.

c) El fenomenalismo; Esta doctrina trata de conciliar las dos tendencias opuestas, la del realismo y la del idealismo. El fenomenalismo sostiene que no conocemos las cosas como son en sí sino como se nos aparecen. Considera que existen cosas reales pero que no podemos conocer su esencia, su natualeza íntima. Lo único que nos es dado conocer son los fenómenos, es decir contenidos de conciencia. El fundador de esta doctrina es Manuel Kant.

1.4. Las Formas del Conocimiento

Aquí se trata de saber cuáles son las especies de conocimiento, es decir en qué forma conocemos un objeto.

Todos o casi todos nuestros conocimientos son adquiridos en forma mediata mediante operaciones lógicas. Según los intuicionistas, hay una intuición racional, otra emocional y otra volitiva, cuyos órganos de conocimiento son respectivamente, la razón, el sentimiento y la voluntad. En los tres casos hay un conocimiento inmediato de un objeto. De manera que si consideramos en cada objeto tres aspectos, la esencia, la existencia y el valor, tendremos tres clases de intuicion de un objeto: para la intuición de la esencia servirá la razón, para la intuición de la

existencia la voluntad y para la intuición de los valores la emoción.

Algunos autores admiten la intuición pero sólo para los valores éticos o estéticos. No niega que muchas veces las hipótesis y las teorías científicas, lo mismo que los sistemas metafísicos, descansan en intuiciones que se adelantan a la penosa investigación que avanza paso a paso. Pero toda intuición por más luminosa que sea, debe resistir para su validez la prueba de la inteligencia. En conclusión, las intuiciones sólo podrán suministrar ciertos datos para el conocimiento, pero será menester que estos datos sean elaborados e interpretados por el pensamiento.

1.5. El Problema de la Verdad

Este problema nos aparece ligado al de la esencia del conocimiento. Es evidente que sólo hay conocimiento cuando éste es verdadero; un conocimiento falso es un contrasentido.

La verdad siempre se expresa en un juicio; por consiguiente sólo los juicios pueden ser verdaderos o falsos. Los objetos no son ni verdaderos ni falsos; son reales, ideales o imaginarios. Ahora bien, ¿cuándo es verdadero un conocimiento? Cuando existe una concordancia entre el conocimiento y el objeto.

Emparejada a esta cuestión aparece otra denominada criterio de la verdad, esto es, aquella que se pregunta por la certeza de un conocimiento. Para Kant el criterio de verdad debe ser universal, aplicable "a todos los conocimientos sin distinción de objetos" , y no puede tener un carácter únicamente formal. A este criterio se le llama de Asentimiento Universal.

Resulta pues, de todo lo que dejamos expuesto [FINGERMANN 1976]:

1o. Que el conocimiento, representación interior en el que conoce es siempre formado en razón y supuesto de lo cognoscible.

2o. Que el sujeto no crea, funda o pone por sí lo conocido, ni su presencia, sino que son recibidos por aquél.

3o. Que los elementos receptivos son la base ó la materia sobre la cual obra el sujeto cuando piensa, dando forma a la representación intelectual.

4o. Que el sujeto colabora con la presencia de lo cognoscible á la formación del conocimiento.

5o. Que el conocimiento sólo es resultado de la actividad en la forma determinada en que el sujeto asimila la presencia de lo

cognoscible, forma que es por lo mismo susceptible de error aunque sea éste siempre rectificable, merced á la presencia constante del objeto.

6o. Que el conocimiento es como penetración de lo receptivo con lo activo ó de la materia (lo objetivo) con la forma (lo subjetivo).

Desde el punto de vista de la Inteligencia Artificial lo que nos interesa es el carácter pragmático del conocimiento, en donde la finalidad no es descubrir verdades teóricas sino actuar en la realidad.

En el próximo subíndice, se analizarán las herramientas más representativas que la Inteligencia Artificial ha desarrollado para poder representar el conocimiento.

2. Modelos de Representación de Conocimiento

Los modelos en el área de bases de datos son instrumentos que permiten describir una realidad en forma organizada y estructurada, captando parte del significado o semántica de los datos.

En su concepción más general, un modelo de representación de conocimiento es cualquier estructura de trabajo, en la cual se puede almacenar y recuperar cualquier información acerca del mundo [ALLEN 1987].

Tradicionalmente, el manejo de datos y los sistemas para el manejo de bases de datos (SMBD) surgieron como una forma de controlar datos dentro de una organización. En su momento la investigación en bases de datos respondió con el desarrollo de varios modelos básicos, a necesidades como:

- Compartir información entre varios usuarios.
- Evitar redundancia e inconsistencia en los datos.
- Mantener en forma eficiente grandes volúmenes de datos.
- Incrementar la confiabilidad de los datos a través de la implantación de mecanismos de integridad.

Estos modelos básicos permitieron organizar los datos en una

forma rigurosa y bien definida, los más conocidos son:

a) **Modelo Jerárquico**, en donde los datos están organizados en una estructura de árbol.

b) **Modelo de Red**, en donde los datos están interconectados vía ligas que forman gráficas dirigidas.

c) **Modelo Relacional**, en donde los datos se organizan en tablas.

Con el incremento en la demanda de sistemas orientados al usuario (casuales y experimentados), se desarrollaron nuevas tendencias fuera del ámbito de los modelos tradicionales, entre las que tenemos:

- La incorporación de más semántica en los modelos de bases de datos.
- El desarrollo de mejores medios ambientes para el usuario que incluyan interfaces más amigables y soporten diferentes vistas del contenido y organización de los datos. Para ello los modelos de bases de datos incluyeron el concepto de metaconocimiento, es decir información en la base sobre la base de datos misma.

Ambas tendencias han conducido al desarrollo de SMBD más inteligentes y es aquí donde los investigadores en bases de datos han empezado a reconocer la conexión natural que existe entre las bases de datos y las bases de conocimiento y aprovechar los logros de las investigaciónes en Inteligencia Artificial que sobre los modelos de representación de conocimiento se han realizado para enriquecer su área.

Aunque hay una gran variedad de herramientas para la representacion de conocimiento todas comparten propiedades comunes. Todas hacen distinción entre proposiciones - aquellas cosas que pueden ser verdaderas o falsas - y términos - aquellas cosas que representan objetos; tanto objetos físicos (mesa, silla, etc) como objetos ilusorios (ideas, eventos ó tiempo) -.

Partiendo de esto, las representaciones divergen mayormente en los tipos de predicados que soportan, los tipos de comportamiento inferencial que definen y los métodos por los cuales se completa la inferencia. El problema de representar el conocimiento es complejo y constituye aún un campo de investigación abierto.

Para empezar, es preciso reconocer los distintos tipos de conocimiento: hay conocimientos descriptivos que corresponden tanto a hechos del dominio de experiencia (conceptos y relaciones), como a datos del problema concreto a resolver; y hay

conocimientos normativos (llamados también procedimentales), referentes a los procedimientos que se utilizan para deducir otros hechos, y éstos a su vez, pueden ser de tipo táctico (reglas) o de tipo estratégico (que reglas aplicar en cada momento).

Tradicionalmente, las técnicas para representar el conocimiento han sido separadas en dos ramas principales: procedurales y declarativas.

La disputa acerca de los méritos relativos de las representaciones procedurales contra las representaciones declarativas fué una batalla importante en la historia de la Inteligencia Artificial, de la cual se desarrolló mucha de la teoría actual de representación de conocimiento [WINOGRAD 1975].

Los proceduralistas hicieron énfasis en lo directo de la línea de inferencias hechas por sus sistemas (usando heurística para dominios específicos evitando así razonamientos irrelevantes), así como la facilidad de codificación y entendimiento del propio proceso de razonamiento.

Las principales ventajas de las representaciones procedurales son:

- Es fácil representar el conocimiento de cómo hacer cosas.
- Es fácil representar conocimiento que no se ajusta en la mayoría de los esquemas declarativos simples (por ejemplo, raciocinios probabilísticos).
- Es fácil representar conocimiento heurístico de cómo hacer cosas eficientemente.

Los declarativistas hicieron énfasis en la flexibilidad y economía de sus esquemas, así como de la totalidad y certeza de sus deducciones y acerca de la facilidad para modificar sus sistemas.

Las principales ventajas de las representaciones declarativas son:

- Cada hecho necesita ser almacenado una sola vez, sin importar el número de formas diferentes en las que puede ser usado.
- Es fácil añadir nuevos hechos al sistema sin cambiar hechos ni procedimientos previos.

A pesar de que en retrospectiva estas posturas parecen haber sido

escogidas arbitrariamente entre otros posibles esquemas de representación, la batalla procedural-declarativa tuvo su importancia. Más que resuelta, esta controversia fué disuelta y el resultado fué un mucho mayor respeto por la importancia de la representación de conocimiento en el trabajo actual en Inteligencia Artificial. Hoy en día es comunmente aceptado que en la mayoría de los casos existe una necesidad de ambos tipos de información; de hecho, gran parte de las representaciones emplean una combinación de ambos [AIKINS 1983].

Las técnicas de representación de conocimiento más comunes son: [PINSON 1981].

1) Autómatas Finitos
2) Programas
3) Lógica Simbólica
4) Sistemas basados en reglas de producción
5) Redes Semánticas
6) Marcos
7) Guiones

Los métodos 1 y 2 son de representación procedural; estos métodos marcaron el inicio de la investigación de modelos para representación de conocimiento y, hoy en día, practicamente no se utilizan, por lo que unicamente citaremos las referencias mostradas por Pinson. Autómatas Finitos [FISHER 70] y Programas [NILSSON 1971].

Los métodos 3, 4, y 5 son de representación declarativa, y los métodos 6 y 7 son de representación mixta. Estas técnicas de representación son las más comunes que se utilizan en la actualidad, por lo que se hará una descripción más amplia de cada una de ellas.

2.1. Lógica Simbólica

Es bien sabido que la lógica, como herramienta para el análisis del comportamiento racional tiene una historia milenaria. Los trabajos más importantes para la formalización de la lógica clásica se realizaron durante la segunda mitad del siglo pasado y la primera de éste. Cabe destacar los trabajos de Boole sobre lógica de proposiciones y Fregge sobre la lógica de predicados [CUENA 1986].

La lógica de proposiciones (o de enunciados) maneja variables proposicionales (a,b,c...) así como las relaciones entre ellos. Estas proposiciones compuestas se logran mediante las siguientes conectivas:

Y	denotado como	AND, ^, &
O	denotado como	OR, V
No	denotado como	NOT, ¬, ~
IMPLICA	denotado por	IF-THEN, ⊃, =>
EQUIVALENTE	denotado por	IF-AND-ONLY-IF, ≡

Unas reglas de formación de enunciados válidos definen rigurosamente el lenguaje y se puede construir un sistema axiomático mediante la definición de axiomas y reglas de transformación.

Con la lógica de proposiciones se pueden formalizar muchos de los procesos racinales mediante la utilización de un sistema de reglas de inferencia.

El resultado de evaluar una proposición sencilla o compuesta, siempre es verdadero o falso, de aquí se pueden generar lo que todos conocemos como tablas de verdad.

P	Q	P^Q	PVQ	¯P	P=>Q	P≡Q
F	F	F	F	V	V	V
F	V	F	V	V	V	F
V	F	F	V	F	F	F
V	V	V	V	F	V	V

donde, "F" es falso y "V" verdadero.

El mecanismo de inferencia que se utiliza en la lógica de proposiciones se base en las siguientes reglas, donde los símbolos "P", "Q" y "R" representan fórmulas.

a) Modus Ponens
dado P, P => Q, deriva Q.

b) Modus Tollen
dado ¯Q, P => Q, deriva ¯P.

c) Silogismo Hipotético
dado P => Q, y Q => R, deriva P => R

d) Silogismo Disyuntivo
dado P V Q, y ¯P, deriva Q

e) Conjunción
dado P y Q — deriva P ^ Q

f) Adición
dado P ó Q — deriva P V Q

La lógica de predicados es una extensión de la lógica de proposiciones, en ella se introducen los elementos necesarios para tratar con razonamientos donde intervienen propiedades de individuos y relaciones entre ellos.

Un sistema que utiliza el cálculo de predicados diferencia los siguientes elementos:

a) Términos, los cuales pueden ser: constantes o variables.

   Por ejemplo, "x" podría ser la variable "personas" y "Juan" una constante.

b) Predicados, que expresan una propiedad de alguna variable o una relación entre dos o más variables.

   Por ejemplo, si definimos el predicado "alto", "alto(x)" sería la representación de "el individuo x es alto".

   Si "MAQ" es el predicado de "es más alto que", "MAQ(Juan,Luis)" representa "Juan es más alto que Luis".

c) Funciones, que permiten representar transformaciones.

   Por ejemplo, sean las funciones "padre(x)" y "madre(x)" y el predicado "casado(x,y)". La expresión "casado(padre(x),madre(x))" representa el hecho de que la madre y el padre del individuo x esten casados.

   Es importante notar que, mientras que la evaluación de una función da como resultado un elemento del universo, la evaluación de un predicado en lógica binaria es verdadero o falso.

d) Conectivas, u operadores lógicos, como por ejemplo: Y, O, NO, etc.

e) Cuantificadores, de existencia (Existe,) y universalidad (Para Todo,).

El mecanismo de inferencia que utiliza la lógica de predicados se basa en las reglas definidas en la lógica de proposiciones, además de aquellas que manejan los cuantificadores.

Los cuantificadores, los operadores, las variables, los predicados y las reglas para poner en marcha el proceso de inferencia, constituyen el cálculo de predicados de primer orden. Ciertas expresiones que requieren cuantificar los predicados o utilizarlos como argumentos, conducen a la lógica de predicados de orden superior.

La idea de tomar a la lógica de predicados de primer orden como un lenguaje de programación condujo a la definición del lenguaje PROLOG, en donde el programador sólo tiene que especificar los hechos y las reglas que definen el dominio del conocimiento sin preocuparse de los procesos de deducción automática [KOWALSKI 1974].

Para mostrar el poderío de este modelo, ilustraremos la definición de "conocimiento proposicional" según Israel Scheffler [SCHEFFLER 1979].

Definición.

X conoce que Q
si y sólo si

I) X cree que Q

II) X tiene evidencia adecuada que Q

III) Q es verdadero

En ésta definición, las tres condiciones enunciadas -- condición de creencia, condición de evidencia y condición de verdad -- definen conjuntamente el conocimiento y corresponde al conocimiento fuerte. El conocimiento débil se define con las condiciones de creencia y verdad.

Resumiendo, podemos decir que la resolución de problemas cuya naturaleza es deductiva, su tratamiento es viable mediante la lógica simbólica, sin embargo la mayoría de los problemas reales son de naturaleza inductiva, por ejemplo la interpretación de conocimientos para el diagnóstico de un padecimiento que implica conocimientos inciertos.

Un ejemplo de sistemas que utilizan la representación en base a la lógica de predicados es el sistema SIR [RAPHAEL 1968].

Para información adicional, consultar [NILSSON 1971],

[KOWALSKI 1974] y [EMDEN 1976].

2.2. Sistemas Basados en Reglas de Producción

Hace más de 40 años que Post (1943) propuso el formalismo de reglas de producción como mecanismo computacional de carácter general. También lo encontramos en los algoritmos de Markov y en la lingüística por Chomsky (1957), que utiliza reglas de reescrituración en el reconocimiento sintáctico de frases de lenguaje natural [MARTINEZ 1987].

Una regla de producción es un par ordenado (A,B), que puede representarse en el lenguaje de la lógica de proposiciones como A -> B. Los elementos del par reciben distintos nombres: antecedente y consecuente, premisa y conclusión o condición y acción.

La sintaxis o escritura de las reglas es diversa y está en función del proceso de selección y de ejecución de las mismas. Esta sintaxis es reconocida por el intérprete que accesa las relgas, que son en sí una cadena de símbolos; algunos sistemas aceptan también el uso de variables.

El intérprete funciona siguiendo una regla de inferencia bien conocida en lógica, la llamada regla de modus ponens: del hecho A y de la regla A -> B, se infiere la conclusión B.

Desde el punto de vista funcional el proceso de inferencia opera en dos fases: una de reconocimiento y otra de acción. La primera se subdivide a su vez en: seleccionar las reglas pertinentes y en resolver el conflicto de resolución en caso de que exista más de una regla aplicable.

Algunos de los criterios que se utilizan para determinar cual acción ejecutar cuando existe un conflicto de resolución son:

- Establecer orden en los datos,
- Clasificar las reglas por prioridad de ejecución,
- Ejecutar la regla más recientemente instanciada,
- Aplicar metareglas.

En la segunda fase se ejecutan las acciones establecidas por las reglas durante el proceso de inferencia. La selección de las reglas depende de la situación en curso de tratamiento. Si se consideran los hechos establecidos y se verifican con las reglas, se dice que el razonamiento del sistema es hacia adelante, ya que

de cumplirse el miembro izquierdo de la regla, se ejecuta la parte derecha de ésta. Por el contrario, si se consideran las metas posibles a alcanzar, es decir el miembro derecho de la regla y se verifican sólo las reglas que concluyen esas metas, se trata de razonamiento hacia atrás.

Una de las características sobresalientes de los sistemas basados en reglas de producción es su potencialidad de aprendizaje; es decir que a partir de su base de conocimientos inicial, el sistema es capaz de generar o simplificar las reglas de inferencia que rigen su comportamiento, pero no en forma automática.

Este esquema es la base de MYCIN [SHORTLIFFE 1976] - sistema experto desarrollado en Stanford en 1976 para el diagnóstico y tratamiento de enfermedades infecciosas en la sangre -, que, desde el punto de vista de lógica formal, utiliza lógica de proposiciones con algunas extensiones que lo aproximan a la lógica de predicados (existencia de predicados predefinidos y ciertas posibilidades de cuantificar) y a la lógica multivaluada (factores de certidumbre).

Un ejemplo de sistemas que utilizan el modelo de reglas de producción es el sistema EURISKO [LENAT 1983].

En suma, los sistemas basados en reglas de producción permiten representar los conocimientos en forma modular y uniforme; son útiles en los casos donde se detecta y trata con una gran cantidad de estados independientes; en sistemas cuyos objetivos son amplios y cuyas acciones son de corto alcance y de toma de decisiones súbita. La mayoría de los sistemas expertos comerciales hoy en día utilizan esta herramienta para la representación de los conocimientos.

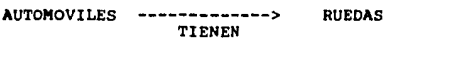
Para ampliar detalles sobre la filosofía y la teoría de los sistemas basados en reglas de producción, consultar [POST 1943], [DAVIS 1977] y [SUWA 1984].

2.3. Redes Semánticas

Este modelo de representación de conocimiento fue introducido por Ross Quillian en 1968, originalmente fue desarrollado como un modelo psicológico de la memoria asociativa humana conocido como MEMORIA SEMANTICA [QUILLIAN 1968].

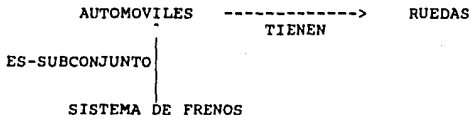
Esta representación se compone de un conjunto de nodos relacionados entre sí por medio de ligas o arcos; cada nodo corresponde a: un atributo, un estado, una entidad o un evento. Las ligas o arcos que van de un nodo a otro representan relaciones entre nodos.

Consideremos por ejemplo:



donde AUTOMOVILES Y RUEDAS son nodos representando conjuntos ó conceptos y TIENEN es el nombre del arco especificando su relación. Entre las posibles interpretaciones de esta red, está la siguiente oración: TODOS LOS AUTOMOVILES TIENES RUEDAS.

La habilidad de apuntar directo a los hechos relevantes es particularmente notable en los arcos ES-UN y ES-SUBCONJUNTO, las cuales establecen una propiedad inherente de jerarquía en la red. Por ejemplo:



De aquí puede interpretarse que como el SISTEMA DE FRENOS es un subconjunto de los AUTOMOVILES, y los AUTOMOVILES TIENEN RUEDAS, el SISTEMA DE FRENOS TIENE RUEDAS también. La interpretación de las estructuras de la red depende solamente del programa que las manipula, es decir no hay convención alguna del significado. De ahí que las inferencias que se derivan de la manipulación de la red no son necesariamente válidas.

Una característica clave de las redes semánticas es que las asociaciones importantes se pueden hacer explícitamente. Hechos relevantes acerca de un objeto pueden ser inferidos de los nodos a los cuales están directamente ligados, sin necesidad de buscar en grandes bases de datos.

Podría considerarse la teoría y el modelo de redes semánticas como una teoría computacional del entendimiento verbal superficial en los humanos. Si consideramos los nodos como estructuras que usamos para conceptos verbales, las relaciones semánticas representarían el concepto lingüístico del pensamiento y juntos describen eventos en lenguaje natural. [NORMAN 1971].

Un ejemplo de sistema que utiliza este tipo de representación es PROSPECTOR [DUDA 1979] - sistema experto en el tema de Exploración Mineral - .

Para información adicional sobre formas de uso de las redes semánticas, consultar [FINDLER 1979] y [SCHUBERT 1976].

2.4. Marcos (Frames)

Este modelo de representación de conocimiento fue propuesto en 1975 por Minsky [MINSKY 1975]. En este modelo se parte de que existe una gran evidencia de que el ser humano no analiza nuevas situaciones partiendo de la nada, ni tampoco construye nuevas estructuras de conocimiento para describirlas. En su lugar, parece que tenemos en la memoria una gran colección de estructuras que representan nuestas experiencias previas con objetos, lugares y situaciones. Para analizar una nueva experiencia evocamos las estructuras apropiadas que tenemos almacenadas y las llenamos con detalles del evento actual.

Un marco es una estructura de trabajo general que describe objetos, la cual puede ser adaptada a una situación especifica cambiando ciertos detalles. Consiste de una colección de "slots" o casillas que describen aspectos de los objetos. Asociado a cada casilla puede haber un conjunto de condiciones que deben ser cumplidas por quien quiera llenarlas.

Las caracteristicas generales de este modelo de representación de conocimiento son:

a) Cada objeto tiene asociado un conjunto de atributos que lo caracterizan.

b) Cada atributo tiene asociado un valor, el cual a su vez puede ser otro objeto.

c) Los atributos-valor corresponden a:

- El nombre de un procedimiento especifico; ejecutable automáticamente o mediante requisición.
- Apuntadores a otro objeto.
- Un valor calculado durante el proceso de ejecución.
- Un valor asignado por omisión.
- Una constante.

De esta forma la representación mediante objetos combina el aspecto procedural y declarativo, lo cual constituye una

aportación más a las ya efectuadas por las reglas semánticas.

Por supuesto que antes de que un marco pueda ser utilizado, debe verificarse que sea aplicable a la situación actual. Minsky propone que alguna parte de la evidencia sea usada para hacer una selección inicial del marco candidato; el marco se instancia creando una estructura específica que describa la situación actual. Este marco deberá contener algunas casillas para las cuales algunos valores serán asignados.

El estado de un objeto definido por el sistema es:

a) Activo, en el caso de que se encuentre presente en la lista de hipótesis en curso de tratamiento, ya sea para confirmarlo o eliminarlo.

b) Semiactivo, en el caso de que las hipótesis sean sugeridas vía diversas alternativas, pero sin suficiente "peso" para ser considerdas todavía.

c) Inactivos, cuando el objeto se ha eliminado o jamás será instanciado para ser considerado.

La estructura de control varía según el caso específico, sin embargo algunos criterios seguidos son:

- Ordenar datos.
- Poner en marcha ciertas hipótesis asociadas a situaciones precisas.
- Ordenar las hipótesis.
- Verificar la hipótesis más relevante.
- Instanciar la hipótesis más recientemente ejecutada.

En resumen, los marcos como las redes semánticas, son estructuras de propósito general en los cuales es posible representar conjuntos particulares de conocimiento dentro de un dominio específico. Los detalles de la operación de un sistema basado en marcos contendrán, así como el tipo de razonamiento que el sistema usará durante su actuación.

Un ejemplo de sistemas que utilizan este modelo de representación es el sistema AM [LENAT 1976] y el sistema GUS [BOBROW 1977].

Para información adicional, consultar [MINSKY 1975], [GOLDSTEIN 1979] y [WINOGRD 1975].

2.5. Guiónes (Scripts)

Este modelo de representación de conocimiento fue introducido en 1975 por Schank y Abelson como una propuesta de especialización de la teoría de marcos. [SCHANK 1977].

Un guión es una estructura que describe una secuencia estereotípica de eventos dentro de un contexto en particular y está formado por un conjunto de casillas (slots). Asociada a cada casilla puede haber información acerca del tipo de valores que puede contener, así como un valor que será usado en caso de que ninguna información este disponible.

Como podemos notar, está descripción es muy similar a la de un marco dada en la sección anterior y hasta este nivel de detalle son idénticas. Debido al papel tan especializado que tiene el guión es posible hacer observaciones más precisas acerca de su estructura.

Los componentes más importantes de un guión son:
(ver figura no. 2.1)

a) Condiciones de entrada.- Estas condiciones en general deben ser satisfechas antes de que los eventos descritos en el guión puedan ocurrir.

b) Resultados.- Son condiciones que generalmente serán ciertas después de que los eventos descritos en el guión hayan ocurrido.

c) Propiedades.- Son casillas que representan a los objetos involucrados en los eventos del guión. La presencia de estos objetos puede ser inferida aún cuando no sean mencionados explícitamente.

d) Roles.- Son casillas representando a las personas involucradas en los eventos descritos en el guión. Al igual que en las propiedades, la presencia de estas personas puede ser inferida sin necesidad de que sean mencionadas explícitamente.

e) Rutas.- Dentro de un guión pueden existir varias secuencias de eventos que comparten varias escenas, mientras que las restantes pueden ser opcionales o excluyentes. A estas secuencias de escenas se les llaman rutas.

f) Escenas.- Una escena es una secuencia determinada de eventos que ocurren. Los eventos están representados en forma de dependencias conceptuales [RIESBECK 1975].

GUION : RESTAURANTE RUTA : CAFETERIA PROPIEDADES : mesas menú C=comida cuenta dinero ROLES : H=cliente M=mesera CO=cocinero CA=cajero D=dueño CONDICIONES DE ENTRADA : H tiene hambre H tiene dinero RESULTADOS : H tiene menos dinero D tiene más dinero H no tiene hambre H está satisfecho (opcional)	Escena 1 : LLEGADA H PTRANS H hacia resturante H ATTEND ojos a las mesas H MBUILD donde sentarse H PTRANS H hacia la mesa H MOVE H sentado Escena 2 : ORDENAR (menú en la mesa) H PTRANS menu a H (go to *e2.1) (la mesera trae el menú) M PTRANS M hacia la mesa M ATRANS menu a H *e2.1 H MTRANS lista-comida a CP(H) *e2.2 H MBUILD escoge C H MTRANS señal a M M PTRANS M hacia la mesa H MTRANS orden a M M PTRANS M a CO M MTRANS (ATRANS C) a CO (CO MTRANS 'no C' a M) M PTRANS M a H M MTRANS 'no C' a H OR (go to *e2.2),(go to *e4.2) CO DO script-preparar-comida Escena 3: COMER *e.3 CO ATRANS C a M M ATRANS C a H H INGEST C OR (go to *e2.2),(go to *e4.1) Escena 4: *e4.1 M MOVE (escribir cuenta) M PTRANS M a H M ATRANS cuenta a H H ATRANS propina a M H PTRANS H a CA H ATRANS dinera a CA *e4.2 H PTRANS H afuera del restaurante

Figura 2.1. Componentes de un Guión.

A pesar de que los guiones son estructuras menos generales que los marcos y éstas no se ajustan a la representación de cualquier tipo de conocimiento son muy útiles porque, en el mundo real sí se ajustan los patrones a las secuencias de los eventos. Estos patrones surgen debido a las relaciones causales entre los eventos y estos a su vez, descritos en un guión forman una gigantesca cadena causal.

El inicio de la cadena es el conjunto de condiciones de entrada que permiten que los primeros eventos de los guiones ocurran y el fin de la cadena es el conjunto de resultados que pueden a su vez, permitir que otros eventos o secuencias de eventos (otros guiones) ocurran.

Un ejemplo de sistemas que emplean el modelo de guiones y que además utilizan la teoría de dependencias conceptuales es el sistema ESPIA [MACHADO 1985].

Para mayor información sobre guiones, consultar [SCHANK 1977] y [SCHANK 1981].

2.6. Otras Representaciones

Mucho de la investigacion en Inteligencia Artificial a la fecha, ha estado concentrada en la representación de conocimiento de tal forma que pueda ser adquirida, almacenada y utilizada por computadora. Sin embargo, existen algoritmos que permiten obtener conocimientos directamente de un conjunto de datos.

Estos sistemas expertos no convencionales se basan en el reconocimiento de patrones en lugar de razonamiento.

2.6.1. Algoritmo "ID3"

El algortimo ID3 originalmente desarrollado por J. Ross Quinlan [QUINLAN 1983] es el método más comunmente usado en sistemas expertos comerciales que utilizan métodos de inducción para generar reglas.

La complejidad de los problemas del mundo real frecuentemente dificultan el diseño de un conjunto detallado de reglas. (Ver Sistemas Basados en Reglas de Producción). En algunas áreas la cantidad de información que se requiere para dar una solución es prohibitivamente grande, en otras el conocimiento no está lo suficientemente definido como para ponerlo en reglas.

El algoritmo funciona de la siguiente forma: la información a

clasificar se organiza en una tabla, a cada renglón se le llama "ejemplo" y a cada columna se le llama "atributo". A la primera columna se le llama "atributo de clase" y la meta es determinar la relación existente entre el "atributo de clase" y los valores de otros atributos.

El primer paso consiste en seleccionar uno de los atributos para que sea el punto de inicio o nodo raiz del árbol. Una vez seleccionado este atributo, se particiona el conjunto de ejemplos (renglones de la tabla) en tablas más pequeñas cada una conteniendo ejemplos con el mismo valor del atributo seleccionado. Si las instancias del "atributo de clase" no son las mismas para cada una de estas tablas más pequeñas se selecciona un segundo atributo que viene a ser otro nodo del árbol y se repite esta operación hasta que el "atributo de clase" este clasificado.

En las figuras 2.2, 2.3 y 2.4, se muestra un ejemplo de como trabaja un árbol de clasificación.

UTILIDAD	ANTIGUEDAD	COMPETENCIA	TIPO
Baja	Vieja	No	Software
Baja	Media	Si	Software
Alta	Media	No	Hardware
Baja	Vieja	No	Hardware
Alta	Nueva	No	Hardware
Alta	Nueva	No	Software
Alta	Media	No	Software
Alta	Nueva	Si	Software
Baja	Media	Si	Hardware
Baja	Vieja	Si	Software

Figura 2.2 Conjunto de ejemplos.

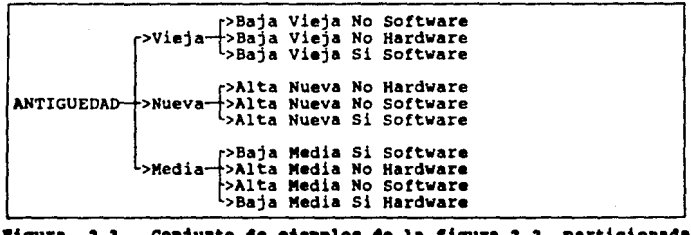


Figura 2.3. Conjunto de ejemplos de la figura 2.2 particionada en el atributo ANTIGUEDAD.

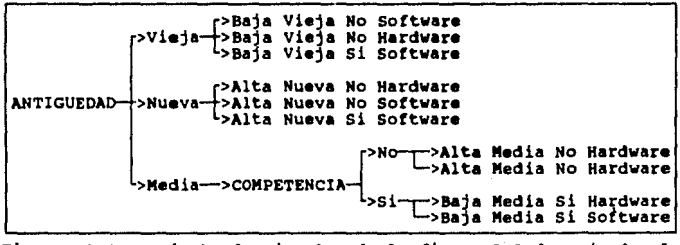


Figura 2.4 Conjunto de ejemplos de la figura 2.3 después de la segunda partición en el atributo COMPETENCIA.

A partir de este árbol, se pueden generar un conjunto de reglas if-then siguiendo cada ruta desde el nodo raíz hasta los nodos terminales. Cada regla es una serie de condiciones que consiste de parejas atributo-valor seguidas por una conclusión que contiene la clase y el correspondiente valor de la clase.

Por ejemplo, siguiendo las figuras 2.2, 2.3 y 2.4, sus reglas serían:

```
If ANTIGUEDAD es vieja
then UTILIDAD es baja

If ANTIGUEDAD es nueva
then UTILIDAD es alta

If ANTIGUEDAD es media
and COMPETENCIA es no
then UTILIDAD es alta

If ANTIGUEDAD es media
and COMPETENCIA es si
then UTILIDAD es baja
```

Figura 2.5 Reglas producidas del árbol de clasificación.

Para mayor información sobre este algortimo, consultar [QUINLAN 1983] y [THOMPSON 1986].

2.6.2. Clasificador Holland

La literatura de sistemas expertos muestra que la mayoría de los programas existentes funcionan como clasificadores. Un conjunto de estímulos del medio ambiente (características, síntomas, etc) son clasificados dentro de una o más categorías. Por ejemplo, los diagnósticos médicos se desarrollan asignando a un conjunto de síntomas una enfermedad específica, es decir se clasifican por el tipo de enfermedad. Este algoritmo de reconocimiento de patrones fue propuesto en la Universidad de Michigan por John Holland [HOLLAND 1986].

El algoritmo funciona de la siguiente forma: La información acerca de un conjunto de condiciones se codifica en una cadena de bits, en donde, cada bit representa una característica específica que, generalmente es de naturaleza binaria (existe o no existe). Esta cadena de bits la cual puede ser de 10 ó 12 bits para problemas simples o de varias cientos de bits para problemas complejos, se le conoce como "mensaje". Un patrón de características en particular se le conoce como un "clasificador".

Para desarrollar una tarea como encontrar a una persona específica en un evento deportivo, se deberá crear un "clasificador" que especifique entre aquellos posibles, los atributos que el individuo posea. De igual forma, se deberá crear un "mensaje" por cada individuo en el evento deportivo especificando los atributos que cada uno posee. Aquel "mensaje" que sea lo más parecido al "clasificador" será el individuo en cuestión.

Aunque conceptualmente es simple, existen algunas características a considerar:

- Se debe especificar el conjunto global de atributos. Esta definición generalmente requiere un conocimiento de experto.

- La frase "muy similar" frecuentemente representa un problema. Existe la estrategia de clasificar los bits en tres tipos: Tipo "A", aquellos que representan una características esencial o crítica; Tipo "B", todas aquellas características que son muy recomendables más no críticas y Tipo "C", aquellas características que sirven como diagnóstico cuando están presentes y no tienen importancia si están ausentes. Un "clasificador" es "muy similar" a un "mensaje" cuando posee todas las características de Tipo "A" y el mayor número de características Tipo

"B" y "C".

Un ejemplo de sistemas que utilizan este algoritmo es el sistema COPDAB [SCRODT 1986].

Para mayor información sobre este algoritmo, consultar [HOLLAND 1986] y [FREY 1986].

CAPITULO III

APRENDIZAJE

Introducción

Una de las manifestaciones de la inteligencia - considerada por algunos la "esencia" de la inteligencia - es la habilidad de aprender.

Existen literalmente docenas de respuestas a la pregunta ¿qué es el aprendizaje? Sin embargo, no hay una sola definición completa o convincente de aprendizaje. Ni siquiera se puede decir con seguridad que exista una definición que la mayor parte de las personas involucradas con el aprendizaje encuentre aceptable. No obstante la gran cantidad de formas en que se define el aprendizaje, existe cierto grado de acuerdo en cuanto a lo que implica. En casi todas las definiciones de aprendizaje se mencionan tres conceptos: cambio, conducta y experiencia.

El aprendizaje produce un cambio en el interior del organismo. Se desconoce la naturaleza exacta de éste cambio. Puede ser anatómico, químico, eléctrico o una combinación de los anteriores. Como regla general, podemos decir que el aprendizaje implica cambios estables.

El aprendizaje tiene una profunda influencia sobre la conducta del ser humano. De hecho, los organismos modifican sus conductas antiguas y adquieren nuevas, principalmente por medio del aprendizaje.

Hemos hablado de que el aprendizaje implica cambios, pero ¿qué es lo que da lugar a estos cambios? La respuesta es la experiencia. La experiencia es el factor más importante aunque no el único que interviene en el aprendizaje. Naturalmente, ciertas experiencias tienen efectos que no pueden considerarse como aprendizaje.

Como definición hemos escogido la siguiente,

> "el aprendizaje es un cambio relativamente estable, sin especificar, dentro del organismo que posibilita un cambio en la conducta; se debe a la experiencia y no puede explicarse en términos de reflejos, instintos, maduración o influencia de la fatiga, de las enfermedades o drogas" [CHANCE 1981].

Las computadoras están "aprendiendo a aprender" mediante programas heurísticos y adaptables que, desde un punto de vista conceptual, permiten a las máquinas aprender de la experiencia.

Cualquier sistema de aprendizaje está relacionado con la adquisición de conocimientos y su almacenamiento mediante una representación u otra. A nivel máquina, el modo en que los conocimientos se memorizan afecta directamente al modo en que tiene lugar el aprendizaje, y por supuesto, determina si éste tiene lugar o no. Por ejemplo, si un sistema utiliza funciones evaluativas para controlar una búsqueda, entonces el aprendizaje implica la adquisición de mejores funciones de evaluación; pero si se utilizan reglas de producción, el aprendizaje significa la generación de nuevas reglas capaces de organizar nuevos tipos de comportamiento.

El objetivo de este capítulo es describir ambos aspectos del aprendizaje. Por un lado el aspecto humano, es decir, cual es el tipo de aprendizaje que se espera del alumno y cuales son sus características. Por otro lado, una breve descripción de los diferentes tipos de aprendizaje que se utilizan en los sistemas expertos.

1. ¿Cómo se adquiere el Conocimiento?

El aprendizaje infantil se logra mediante actividades de exploración que hace el niño de su entorno, intentando activamente descubrir las relaciones entre las acciones y los resultados. Es decir, es una situación exploratoria de ensayo y error.

El aprendizaje adulto tiene a veces estas mismas características, pero más a menudo consiste en la adquisición de nuevo conocimiento o nuevas habilidades bajo la guía de un instructor o un libro. El aprendizaje mediante libros o mediante instructor no requiere ya de los dificultosos esfuerzos exploratorios para determinar la relación entre la situación ambiental y las acciones: el papel del instructor o del libro es indicar las cuestiones relevantes y guiar la adquisición de conocimiento importante.

La mayor parte del conocimiento adulto se puede caracterizar como un intento por acoplar el nuevo conocimiento que se presenta con las estructuras de la memoria presentes: usar los esquemas existentes en la memoria para orientar la formación de nuevos esquemas. Aquí lo más importante es darse cuenta de la tremenda cantidad de conocimiento que se requiere para que una persona aprenda temas nuevos. De ahí que, al hablar de Sistemas Expertos, sea muy difícil que este tipo de herramientas pretendan aumentar los conocimientos de una persona, sino más bien debemos

considerarlos como elementos de apoyo para facilitar el manejo de los mismos.

Pensemos en el mucho tiempo y esfuerzo que lleva aprender un tema. Casi cualquier tema que vale la pena aprender lleva años aprenderlo bien. Una buena manera de caracterizar temas semejantes es la existencia de expertos: siempre que un tema es lo suficientemente rico o importante como para que en él haya expertos practicantes, entonces habrá una situación en la que se requerirán muchos años para alcanzar un conocimiento experto.

Pensemos en cualquier tema: historia, mecánica de automóviles, matemáticas, béisbol, lenguas, incluso computación. Todos estos temas requieren años de estudio para dominarlos. Los temas difieren en los tipos de destrezas involucradas: algunos requieren movimientos grácile y controlados de las manos, otros requieren buena capacidad de memoria o de razonamiento abstracto, otros requieren juicios espaciales, y otros aún requieren de coordinación corporal. Pero todos requieren grandes cantidades de conocimiento organizado de maneras que hagan rapidamente accesibles los conocimientos pertinentes, y ello con eficiencia y en los momentos adecuados. El secreto de aprender materias complejas está en la organización adecuada de la información en la memoria.

Consideremos lo que conlleva llegar a conocer una materia. Un libro, por ejemplo, requiere claramente una cantidad considerable de tiempo para leer. Sin embargo, el libro no es más que la superficie. Por un lado, el lector no podría empezar el libro sin haber ya adquirido una base razonable de conocimientos sobre el mundo, la ciencia, las matemáticas, el arte y la música. Aunque una persona generalmente cree que en estos temas está "flojo", probablemente los ha estudiado en la escuela durante varios años.

Consideremos cualquier deporte o ejecución. Los concertistas profesionales aprenden su instrumento a edades muy tempranas: bastante antes de la adolescencia, y practican habitualmente cuatro o cinco horas al día, todos los días, durante el resto de sus vidas. Los atletas profesionales han dedicado cantidades de tiempo similares, y también deben practicar continuamente, o de lo contrario pierden sus habilidades.

La historia es la misma en todos los temas complejos: para dominar temas complejos se requieren miles y miles de horas de aprendizaje y práctica. De hecho, no hay indicios de que uno alcance nunca una ejecución óptima: no importa cuánto tiempo hayamos empleado en aprender, con más estudio y aprendizaje podemos mejorar la ejecución.

Un conjunto común de métodos de instrucción exige que el aprendiz desarrolle estrategias e hipótesis sobre el material por aprender. Esto es especialmente cierto en los sistemas llamados aprendizaje por descubrimiento o diálogo socrático, métodos de aprendizaje mediante tutorías. Aquí el instructor responde las

preguntas, o bien con preguntas o con demostraciones, y a los estudiantes se les enseña a responder sus propias preguntas, a descubrir por sí mismos las relaciones importantes entre los materiales que están aprendiendo. Con una orientación sabia por parte del instructor, los estudiantes "accidentalmente" dan con (y más importante, advierten) la información crítica. Otros métodos de enseñanza se dirigen más a la pura presentación del conocimiento, la práctica en el empleo de ese conocimiento y la comprobación para asegurar que el conocimiento se ha adquirido realmente y que se puede usar en las ocasiones adecuadas.

La mayor parte del aprendizaje en los adultos se puede caracterizar diciendo que consiste en muchos pasos pequeños, cada uno de los cuales es simple y directo, en donde surgen complicaciones y dificultades inevitables por dos razones:

1. El conocimiento nuevo tiene que ser integrado apropiadamente en las estructuras de la memoria del conocimiento previamente adquirido.

2. Tiene que haber un conjunto apropiado de estructuras de conocimiento y de memoria como base para adquirir cualquier conocimiento particular nuevo [LINDSAY 1983].

Muchas de las dificultades para aprender cosas nuevas se pueden atribuir al fracaso en uno de estos dos aspectos del aprendizaje. Cuando se considera que algo es difícil o complejo, o quizá ininteligible, generalmente significa que está presente la razón 2. Cuando parece que tiene sentido, pero luego no está disponible para su uso en el momento adecuado, falla generalmente la razón 1: el nuevo conocimiento no estaba integrado apropiadamente con el conocimiento previo. Aunque se haya adquirido la información, si no está disponible en el momento deseado es como si esa información nunca hubiera existido.

Los estudios sobre el proceso de aprendizaje - especialmente el aprendizaje que tiene lugar en la adquisición de materias complejas - está en sus inicios. En parte, porque los desarrollos recientes en el estudio de la memoria no han hecho todavía mucho impacto en los estudios del aprendizaje. Los estudios del aprendizaje de materias complejas son a su vez complejos, y no se ha adelantado mucho en el desarrollo de herramientas experimentales y analíticas para la tarea.

El estudio de temas complejos se puede caracterizar mejor probablemente diciendo que consiste en tres tipos de operaciones sobre las estructuras de la memoria: acreción, restructuración y sintonización. Básicamente si consideramos la memoria humana organizada en esquemas de memoria, entonces el conocimiento que se adquiere debe o bien acoplarse en los esquemas ya existentes o bien en esquemas de nueva adquisición. Para aprender un tema, tiene que haber inicialmente una cantidad básica de información

acerca de él. Esta información consta de los hechos y material de fondo para el tema; es gradualmente acumulado o acrecido en las estructuras de la memoria.

Cuando las materias por aprender son nuevas, los esquemas de la memoria existentes serán inadecuados para caracterizar la información nueva acumulada, de modo que ha de tener lugar una reestructuración de los esquemas. La reestructuración es probablemente el proceso fundamental del aprendizaje de nueva información. Los viejos esquemas se modifican o reestructuran, y crean otros nuevos. Luego, la información que ya existía en la memoria se debe acoplar con las estructuras recientemente creadas, y la nueva información puede entrar directamente en los nuevos esquemas. Es improbable que los esquemas recientemente estructurados sean completamente apropiados para el material: tienen que ser ajustados o sintonizados para que satisfagan las condiciones exactas de uso. Para la mayoría de los temas complejos, las fases del aprendizaje: acreción, reestructuración y sintonización, probablemente ocurren una y otra vez, quizá no cesan nunca durante los largos años que lleva aprender cualquier materia compleja.

Una vez considerado el aspecto del aprendizaje en el humano, en la siguiente sección describiremos el proceso de aprendizaje en los Sistemas Expertos.

2. El Aprendizaje en un Sistema Experto

El proceso de aprendizaje en un sistema experto comprende la adquisición de conocimientos, así como su aplicación y el desarrollo de experiencias con la modificación del comportamiento [MARTINEZ 1986]. Actualmente, está organizado en tres aspectos:

- El desarrollo y el análisis de programas de cómputo que incrementa la eficiencia de ejecución de un conjunto de tareas.
- La simulación de los procesos de aprendizaje humano.
- El análisis teórico sobre los métodos posibles de aprendizaje y de los algoritmos empleados, independientemente de la disciplina a tratar.

Los programas computacionales que aprenden se clasifican en función de:

- La estrategia de aprendizaje empleada.

- La representación del conocimiento o de la experiencia adquirida.

- El campo de aplicación de que se trata.

La estrategia de aprendizaje está relacionada con la cantidad de inferencia ejecutada, a partir de la información disponible. Así, se diferencian distintas clases de aprendizaje:

1. Aprendizaje por memorización, es la adquisición directa de nuevos conocimientos. Por ejemplo: un programa de cálculo que desarrolla una tarea específica y en la cual todos los conocimientos que emplea deben estar explícitamente ya que no realiza inferencias.

2. Aprendizaje por instrucción o algorítmica, consiste en transformar la información desde un lenguaje fuente a una representación interna y posteriormente interpretarla en función de la utilización requerida.

3. Aprendizaje por analogía, aplica los conocimientos ya utilizados, en otras situaciones similares, creando así otros conocimientos y adquiriendo experiencia. Un ejemplo de este tipo de aprendizaje es la reescrituración de un programa para ejecutar funciones similares o con la misma complejidad de resolución.

4. Aprendizaje a partir de ejemplos y contraejemplos; en él se deducen conceptos a partir de la generalizaciones y simplificaciones entre otras acciones. La cantidad de inferencia a ejecutar es considerable.

5. Aprendizaje a partir de ensayo y prueba, así como el descubrimiento efectuado por el programa mismo. Por ejemplo: los programa que establecen teorías que implantan criterios de clasificación en función de jerarquías. Esta clase de aprendizaje, se basa fundamentalmente en la observación realizada por el programa; ésta puede ser pasiva o activa. Pasiva, si sólo realiza clasificaciones. Activa, si se perturba el entorno; retroalimentando las observaciones hechas y reformulando las hipótesis. La cantidad de inferencia efectuada por el programa es mayor que en cualquier otro tipo de aprendizaje y en consecuencia, el más importante.

Los conocimientos adquiridos por el programa pueden ser entre

otros:

- Reglas que rigen el comportamiento.
- Reglas descriptivas de objetos físicos
- Reglas heurísticas para la resolución de problemas.
- Reglas de clasificación taxonómicas.

La representación de estos conocimientos, es a través de:

- Reglas de Producción.
- Arboles de decisión.
- Gramáticas formales.
- Gráficas y redes de estereotipos y esquemas de jerarquías o taxonomías de objetos.

  Descritos en el capítulo dos de la tesis.

Para mayor información sobre el tema aprendizaje en sistemas expertos, consultar [CHARNIAK 1985].

CAPITULO IV

SISTEMAS EXPERTOS

Introducción

La reproducción por modelos, mediante mecanismos de inteligencia artificial y ciencia del conocimiento, han recorrido muchos caminos en sus primeros treinta años, pero todos convergen a una tesis central:

> "la comprensión, la solución de problemas y todas las demas funciones de la inteligencia, incluso el aprendizaje dependen crucialmente del saber, de los conocimientos".

Es preciso conocer primero para poder comprender más tarde. Incluso es necesario saber primero para poder saber más tarde.

La investigación en inteligencia artificial se ha desplazado hacia el saber especifico en las últimas dos décadas. El proyecto que inicio este desplazamiento fue DENDRAL [HAYES-ROTH 1983], un sistema experto que permitió deducir estructuras químicas a partir de los datos disponibles de los fisicoquímicos. Este proyecto que inició en 1965, demostró que para que funcionara de un modo práctico, sencillo y especializado, el sistema debería tener conocimientos considerables de fisicoquímica, es decir poseer grandes cantidades de conocimiento especifico.

La utilidad de los sistemas expertos cubre ya más de una docena de campos, aunque quizá el grupo más numeroso se centre en la medicina. Los primeros esfuerzos para construir sistemas expertos han proporcionado un importante fundamento intelectual y un conjunto útil de herramientas para cierto tipo de trabajos, aunque de momento son esfuerzos limitados.

El objetivo de este capitulo es desarrollar el concepto de experto y describir en general los sistemas expertos (definición, tipos y componentes). Se presenta un panorama de los desarrollos actuales, resaltando los mecanismos de razonamiento que emplean.

1. El Camino de Novato a Experto

¿Cómo es posible que los expertos en un área en particular, alcancen tan altos niveles de desempeño? ¿Cómo se diferencian de aquellos que únicamente son competentes o son considerados peritos ?

Con estas inquietudes en mente, la psicología cognitivista está tratando de convertirse en experto en el tema de expertos. El trabajo pionero en esta área se inició hace 20 años con el estudio de jugadores expertos de ajedrez. Se había considerado que la habilidad de pensar muchas jugadas adelante y analizar las implicaciones de cada movimiento, era la diferencia entre novatos y expertos. A mediados de los 60s, el psicólogo Adriaan de Groot mostró que ni los jugadores expertos ni los novatos piensan más de unos cuantos movimientos adelante. La gran diferencia entre los dos, es la habilidad experta de recordar temporalmente distintas posiciones del tablero (considerando juegos reales) en menos de cinco segundos.

William Chase y Herbert Simon de la Universidad de Carnegie-Mellon en Pittsburgh continuaron esta investigación a principios de los 70s. Ellos demostraron que un experto de ajedrez es aquella persona que después de años de experiencia, puede reconocer tanto como 100,000 posiciones significativas y dar la mejor respuesta a cada una de ellas.

Resumiendo los resultados más recientes en la investigación de expertos los psicólogos hacen notar lo siguiente: [TROTTER 1986]

- El ser experto parece ser muy específico, es decir el ser experto en un dominio no garantiza el ser experto en otros.
- Los expertos desarrollan la habilidad de percibir grandes patrones de información significativos y además lo hacen de tal manera que parece ser casi intuitiva.
- Esta extraordinaria habilidad depende de la naturaleza y organización de su conocimiento.

Miéntras algunos investigadores están enfocados acerca de cómo un experto organiza y utiliza su conocimiento, otros están enfocados a analizar el largo y algunas veces difícil camino que sigue una persona de novato a experto. Los hermanos Hubert L. Dreyfus y Stuart E. Dreyfus de la Universidad de Berkeley en California en su libro más reciente "Mind Over Machine" clasifican este proceso en cinco etapas:

a) Novato; Durante esta primera etapa de adquisición de nuevas habilidades, el novato aprende a reconocer varios hechos objetivos y características relevantes de esa habilidad, así

como reglas para decidir como actuar con esos hechos y características.

Estos hechos y reglas son "libres de contexto", es decir son definidas en una forma clara y ojetiva, de tal forma que el novato puede reconocerlas y aplicarlas dependiendo de la situación en que estas ocurren.

Por ejemplo, una persona que está aprendiendo a manejar se le dice a que velocidad cambiar las velocidades, o cual es la distancia aconsejable con respecto al automóvil de enfrente a cualquier velocidad. Estas reglas básicas ignoran el contexto no tomando en cuenta situaciones como densidad de tráfico o paradas de emergencia.

b) Principiante avanzado; Cuando el novato adquiere experiencia en situaciones reales, su desempeño se incrementa a un nivel aceptable. A través de experiencia práctica en situaciones concretas, y notando la similitud con situaciones extrañas, el principiante avanzado aprende a reconocer y a tratar con hechos y elementos no definidos previamente, así como a aplicar reglas más sofisticadas.

Por ejemplo, un principiante avanzado aprende a usar el sonido del motor para decidir cuando hacer un cambio de velocidades, o a distinguir entre un conductor ebrio de otro.

c) Competencia; Con experiencia, el aprendiz comienza a reconocer más y más situaciones libres de contexto. Para evitar el ser saturado por este cúmulo de información, el futuro experto adopta una visión jerárquica para la toma de decisiones escogiendo un plan para organizar las situaciones concentrándose únicamente en los elementos más importantes. Esta persona simplifica e incrementa su desempeño y gradualmente crece en competencia.

Por ejemplo, un conductor competente que requiere cruzar la ciudad a toda velocidad, no sigue reglas e incluso evade las leyes y toma en cuenta situaciones como distancia o tráfico.

d) Perito; Para esta persona, usualmente muy involucrada en su habilidad, los elementos importantes residen en su memoria miéntras que los no importantes se guardan en el subconciente y temporalmente se olvidan. Así como las situaciones cambian, las diferentes características pueden tomar importancia. Simplemente las cosas suceden, aparentemente porque el perito ha experimentado situaciones similares en el pasado y recuerda los planes que anteriormente generó. Este tipo de personas aún piensan en forma analítica pero en algunas situaciones responden en forma intuitiva reconociendo de una manera sencilla porque ellos ven similitudes con experiencia previas.

Por ejemplo, un conductor perito intuitivamente reduce la velocidad al aproximarse a una curva en un día lluvioso, el concientemente decide desacelerar o aplicar los frenos.

e) Experto; El experto aplica reglas que forman parte integral de su persona, de sus actitudes; las tiene asimiladas. Actúa en forma natural y casi siempre acierta. Cuando falla, frecuentemente es porque está en competencia con otro experto. Su habilidad ha llegado a ser por mucho una parte de ellos que ya no se preocupan por ella, como si fuera de su propio cuerpo.

Por ejemplo, un conductor experto se ha convertido con su automóvil en uno solo, sabe cuando se requiere reducir la velocidad y lo hace sin comparar alternativas.

2. Sistemas Expertos

Los sistemas expertos son programas para resolver problemas que son sustancialmente generales, admitiendo en principio dificultad y requiriendo de experiencia. Farreny los define de la siguiente forma: [FARRENY 1986].

> Los sistemas expertos son programas destinados a remplazar o a asistir al hombre en campos donde se requiere de una expertez humana, cuyas características son:
>
> - Ser insuficientemente estructurada para constituir un método de trabajo.
>
> - Estar sujeta a revisiones o complementaciones (según la experiencia acumulada).

Partiendo de esta definición podemos asignar tres objetivos comúnes a todos los sistemas expertos:

1) Capturar facilmente el conjunto de las unidades del conocimiento.

2) Explotar el conjunto de las unidades del conocimiento.

3) Actualizar y revisar las unidades del conocimiento.

Las unidades del conocimiento son reglas que describen una posible etapa del razonamiento de los expertos.

En si, un modelo de la experiencia de los mejores practicantes de una área se puede pensar como el conocimiento que se ejecuta a un nivel dado, más la utilización de procedimientos inferenciales. Otras definiciones de sistemas expertos se encuentran en el Anexo 1.

El conocimiento de un sistema experto consiste de hechos y heurística. Los hechos constituyen el cuerpo de la información que es compartido ampliamente, públicamente disponible y del cual existe una concordancia general entre los expertos en el campo.

El buen nivel de desempeño del sistema experto está en función del tamaño y la calidad de la base de conocimientos que ésta tiene.

3.1. Tipos de Sistemas Expertos

La mayoría de las aplicaciones en la ingeniería de conocimiento caen en la siguiente clasificación genérica basada en su tipo de función: [HAYES-ROTH 1983]

a) Interpretación, infiriendo descripciones de situaciones a través de sensores de datos.

   Esta categoría incluye entendimiento de lenguaje, análisis de imagenes, interpretación de señales o dilucidar estructuras químicas y muchas clases de análisis inteligente. Un sistema de interpretación explica los datos observados asignándoles un significado simbólico para describir la situación o el estado actual del sistema.

b) Predicción, infiriendo las consecuencias de situaciones dadas.

   Esta categoría incluye pronósticos del clima, predicciones demográficas o de tráfico, estimaciones de cosechas o pronósticos militares. Un sistema de predicción típicamente emplea un modelo dinámico paramétrico con valores de parámetros ajustados a una situación dada. Las consecuencias inferidas del modelos forman las bases para las predicciones. Si se ignora la probabilidad de las estimaciones, los sistemas de predicción pueden generar un gran número de posibles escenarios.

c) Diagnósticos, infiriendo un mal funcionamiento de un sistema por observación.

Esta categoría incluye diagnóstico médico, electrónico, mecánico o de programación entre otros. Los sistemas de diagnóstico típicamente relacionan las irregularidades de un comportamiento con sus posibles causas, usando una de dos técnicas. Un método esencialmente utiliza una tabla de asociación entre comportamientos y diagnósticos. El otro método combina el conocimiento del diseño del sistema con el conocimiento de defectos potenciales en el diseño, implantación o componentes para generar posibles malos funcionamientos en forma consistente con las observaciones.

d) Diseño, configurando objetos bajo restricciones.

Los sistemas de diseño desarrollan configuraciones de objetos que satisfagan las restricciones de los problemas de diseño. Esos problemas incluyen diseño de circuitos impresos, diseño de construcciones e inclusive problemas de costos. Los sistemas de diseño generan descripciones de objetos con varias relaciones con otros y verifican que estas configuraciones cumplan con las restricciones establecidas.

e) Planeación, diseñando acciones.

Estos sistemas se especializan en problemas de diseño concernientes a objetos que desarrollan funciones. Incluyen programación automática en robots, proyectiles, rutas, comunicaciones, experimentos y problemas de planeación militar. Los sistemas de planeación emplean modelos de comportamiento para inferir los efectos de actividades planeadas.

f) Monitoreo, comparando observaciones del comportamiento de un sistema en características que parezcan ser cruciales para que un plan sea exitoso.

Estas características cruciales o puntos vulnerables corresponden a defectos potenciales en el plan. Generalmente los sistemas de monitoreo identifican estos puntos vulnerables en dos tipos. Uno tipo de vulnerabilidad corresponde a condiciones asumidas cuya violación nulificaría el plan en forma racional. Otro tipo de vulnerabilidad surge cuando algunos efectos potenciales del plan violan una restricción de planeación. Muchos sistemas de monitoreo por computadora existen para plantas nucleares, tráfico aéreo, etc., aunque no existe en realidad un sistema experto de este tipo.

g) Depuración, prescribiendo remedios a malos funcionamientos.

Estos sistemas utilizan las capacidades de planeación, diseño y predicción para crear especificaciones o recomendaciones para corregir un problema diagnosticado. Existen sistemas de depuración para programación por computadora en la forma de bases de conocimiento inteligentes y editores de texto, pero ninguno califica como sistema experto.

h) Reparación, ejecutando un plan para administrar un remedio para algún problema diagnosticado.

Estos sistemas incorporan las capacidades de depuración, planeación y ejecución. Los sistemas asistidos por computadora se utilizan en el campo de los automóviles, aviones y mantenimiento de equipos de cómputo entre otros. Los sistemas expertos apenas están ingresando a este campo.

i) Instrucción, diagnosticando y depurando el comportamiento de estudiantes.

Típicamente estos sistemas empiezan construyendo una descripción hipotética del conocimiento del estudiante que se interpreta como el comportamiento del estudiante. Posteriormente se diagnostican las debilidades en su conocimiento y se identifica un remedio apropiado. Finalmente se planea un tutorial que interactue con el estudiante para cubrir sus necesidades de conocimiento.

j) Control, interpretando, prediciendo, reparando y monitoreando el comportamiento de sistemas.

Un sistema experto de control gobierna el comportamiento en conjunto de un sistema. Para hacer esto, el sistema debe interpretar en forma repetitiva la situación actual, predecir el futuro, diagnosticar las causas de los problemas, formular un plan que remedie la situación y monitorear su ejecución para asegurar el éxito de la operación en su conjunto. El tipo de aplicaciones incluye el control de tráfico aéreo, control de misiles e inclusive manejo de negocios entre otros.

2.2. Arquitectura de Sistemas Expertos

A pesar de la diversidad de arquitecturas que se manejan alrededor de los sistemas expertos (ver anexo 2), es posible distinguir los elementos básicos comunes que los forman:

a) Base de conocimientos, la cual contiene el saber especifico en la disciplina de la cual el sistema es experto. Consiste en un conjunto de hechos (datos) y de reglas programadas. No contiene información especifica de un problema en particular. El conocimiento plasmado en la base juega el papel más importante en la calidad y habilidad del sistema.

   Una descripción completa de los modelos para representar conocimiento se incluye en el capitulo dos de esta tesis.

b) Interfaz con el usuario, la cual permite aceptar y reconocer un lenguaje de comandos en forma natural y los traduce en instrucciones y datos para que el sistema experto trabaje.

c) Interfaz con el experto, la cual permite captar información del exterior (proporcionada por el experto) e introducirla en forma adecuada en la base de conocimiento.

d) Mecanismo de razonamiento, también llamada máquina de inferencia, el cual permite controlar el sistema. En este módulo se manejan las reglas contenidas en la base de conocimientos. Dada la importancia de este componente se hará una descripción más amplia en la siguiente sección.

e) Espacio de trabajo, en la cual se almacenan resultados, hipótesis y decisiones intermedias, así como el estado en que el problema se encuentra en un momento dado.

2.3. Mecanismos de Razonamiento

El mecanismo de razonamiento, al cual se hace referencia también como "maquina de inferencia" o "estructura de control", es un conjunto de rutinas y estructuras tanto para la producción como para el control de decisiones, así como para el manejo de preguntas al usuario. En esta parte del sistema experto no debe contener, por lo menos teoricamente, información acerca del dominio especifico del problema y debe contar con cierta independencia para permitir un grado de generalidad.

A pesar de lo anterior, la base de conocimientos y el mecanismo de razonamiento están intimamente ligados, dado que la primera presenta la organización del conocimiento en base al cual el segundo va a trabajar. Existe entonces una estrecha relación entre la organización del sistema experto y su control.

Los mecanismos de razonamiento no pueden ser completamente independientes del tipo de problemas a resolver, sin embargo,

estas rutinas de razonamiento preferentemente no deben ser tan especializadas, pues no podrían entonces aplicarse a otro tipo de problemas.

El razonamiento del sistema experto debe ser guiado de alguna manera, con el fin de proporcionar una secuencia de respuestas válidas y coherentes al usuario, presentando un medio ambiente aceptable. Para tal guía es necesario un conjunto de estrategias de control, las cuales son:

- Estrategias de razonamiento
- Estrategias de explicación
- Estrategias de interpretación

El poderío de un sistema experto podría medirse de acuerdo a que tan correctamente son aplicadas las técnicas de razonamiento sobre el conocimiento almacenado. El éxito del sistema radicará en encontrar una buena respuesta a un problema en base a los recursos con que se disponga. La eficiencia en la búsqueda de soluciones afecta directamente al éxito del sistema.

En [HAYES-ROTH 1983] se presenta un conjunto de factores que ayudan a la eficiencia de un sistema experto, así como algunos que incrementan la dificultad del problema. Los factores que mejoran la eficiencia son:

- Un conocimiento almacenado que sea aplicable y correcto.
- La eliminación de alternativas de solución que lleven al sistema a un "callejón sin salida", o sea a ninguna solución.
- La no redundancia en los procesos y en los datos.
- El incremento de rapidez en las operaciones.
- La existencia de fuentes de conocimiento.
- El razonamiento a varios niveles de abstracción.

Los factores que incrementan la dificultad del problema a resolver son:

- El almacenamiento de conocimiento y datos erróneos.
- Los cambios en los datos, que corresponden al carácter dinámico de la situación del problema.

- El número de posibilidades a evaluar.
- Los procedimientos complejos que implican el desechar una posibilidad.

El sistema experto para un problema debe buscar una solución. Esta tarea le corresponde directamente al mecanismo de razonamiento. El acceso directo a una solución única no es posible en la mayoría de los casos. Generalmente son utilizadas técnicas de búsqueda, así como generadores de soluciones. Estos últimos permiten al sistema tener una visión general de todas las posibles soluciones al problema, permitiéndole probar cada una de ellas hasta encontrar la más apropiada.

Entre los métodos de búsqueda de soluciones, el más simple es el de analizar todas las alternativas una por una. A este método se le denomina de "fuerza bruta". El problema con este método es la existencia, en algunas ocasiones, de espacios de búsqueda de gran tamaño. Otro método de búsqueda es el llamado "de poda", con el cual se eliminan algunas alternativas, mejorando así la eficiencia con respecto al caso anterior.

Entre los métodos de búsqueda más comunes, encontramos los siguientes:

1) Elección de una dirección de solución.

a) Encadenamiento hacia adelante.- Cuando se tienen un conjunto de datos o de ideas básicas como punto de partida, el encadenamiento hacia adelante resulta una técnica natural para direccionar las soluciones de los problemas. Esta metodología ha sido utilizada por sistemas expertos en el área de análisis de datos, diseño, diagnóstico y formación de conceptos.

   Un ejemplo ilustrativo es el sistema AM [LENAT 1976], que descubre conceptos matemáticos en base a la tares de formación de conceptos. El sistema usa encadenamiento hacia adelante iniciando el proceso con ideas elementales de la teoría de conjuntos. Construye un espacio de búsqueda con todas las posibles conjeturas que pueden ser generadas de esas ideas elementales. Elige las conjeturas más interesantes y prosigue en una línea de razonamiento.

b) Encadenamiento hacia atrás.- Esta aproximación es aplicable cuando nuestro punto de partida es una meta o hipótesis. Planificación es un buen ejemplo para este tipo de aproximación, debido a que la función del planificador es construir un plan para poder alcanzar las metas deseadas. Un planificador considerará sus metas sin el consumo de recursos excesivos o la violación de restricciones. Si hay una meta en conflicto, el planificador tiene que establecer prioridades.

Un ejemplo ilustrativo es el sistema NOAH [SACERDOTI 1977]. Este sistema de planificación asigna un orden en el tiempo a los operadores de un plan. Los planes individuales se extienden en paralelo para que interactúen las submetas. Inicialmente se asigna un orden parcial de tiempo para los operadores y cuando se observan interferencias entre los planes parciales de submetas, se ajusta el orden de los operadores para poder resolver dichas interferencias.

c) Encadenamiento hacia adelante y hacia atrás.- Cuando el espacio de búsqueda es grande, la técnica de doble búsqueda suele ser eficiente. El método consiste en tomar un estado inicial y las metas o hipótesis siguiendo un proceso de convergencia para poder igualar las soluciones en un punto intermedio. El método que se utiliza es parecido al de relajación. Esta aproximación también es muy útil cuando el espacio de búsqueda puede dividirse jerárquicamente. En tales casos, la búsqueda se combina apropiadamente en términos de "Top down" y "Bottom up". Una búsqueda de tal naturaleza se aplica en forma particular a problemas complejos, incorporando además incertidumbre.

En ejemplo ilustrativo de esta técnica es el sistema HERSAY II [ERMAN 1980]. Este sistema cuya función es interpretar señales con el propósito de entender el habla, divide el problema jerárquicamente desde arriba en diferentes niveles, con oraciones en el tope y parámetros que miden señales en la parte inferior. El sistema hace un procesamiento "Top down" y "Bottom up" con una aproximación de relajación para ampliar y combinar los candidatos parciales.

Los datos frecuentemente están llenos de errores y tienen ruido, de esto se desprende lo siguiente:

- Los intérpretes deberán enfrentarse con información parcial.
- Para cualquier problema dado, los datos parecen ser contradictorios. El interpretador debe ser apto para suponer cuales datos son creíbles.
- Cuando los datos no son confiables, la interpretación no será confiable. Para la credibilidad es importante identificar en donde la información es incierta o incompleta y donde las suposiciones se han hecho.
- Las cadenas de razonamiento pueden ser largas y complicadas. Es provechoso estar dispuesto a aclarar como las especificaciones están sustentadas por la evidencia.

d) Manejo de eventos.- Esta aproximación para encontrar soluciones de problemas es parecida al encadenamiento hacia adelante excepto que se basa en el estado actual del problema. La técnica va tomando las etapas subsecuentes en base a nuevos datos o en respuesta a un cambio de situación. La técnica de manejo de eventos es muy apropiada para operaciones en tiempo real, tal como monitoreo o control.

Un ejemplo ilustrativo es el sistema VM (Ventilator Manager) [CHERNIACK 1977], cuya función es interpretar en tiempo real el significado clínico de los datos que arroja un sistema de monitoreo fisiológico de aliento.

2) Razonamiento en la presencia de incertidumbre.

En muchos casos las soluciones de los problemas se conducen en presencia de incertidumbre en los datos o en el conocimiento. Para este tipo de problemas es posible utilizar técnicas numéricas, o también, las incertidumbres pueden ser manejadas con una aproximación de la forma de regreso de rastro. El razonamiento en la presencia de incertidumbre sucede en ejemplos típicos de diagnóstico y análisis de datos.

a) Procedimientos numéricos.- Los procedimientos numéricos han sido ideados para manejar evidencias que pueden ser combinadas. Los sistemas que utilizan esta aproximación manejan factores de certidumbre relacionados con probabilidades que usan rangos entre 0 y 1 para indicar la intensidad de la evidencia.

Los sistemas expertos cuya función consiste en el diagnóstico son algunos ejemplos que usan la técnica mencionada. Esto se debe a que los juicios de razonamiento dados en términos de reglas de condición-conclusión van acompañados de una estimación de incertidumbre.

Un caso concreto de sistema experto de diagnóstico es MYCIN [SHORTLIFFE 1976), que tiene como proposito el diagnosticar infecciones bacterianas y recomendar la terapia en base a antibióticos.

b) Revisión de la Credibilidad o "Mantenimiento de la Verdad".- Con frecuencia, la credibilidad y las líneas de razonamiento evolucionan en base a la información que se va tomando para el problema. Cuando la información es parcial o errónea completamente, el sistema incurre en contradicciones y trae como consecuencia malas conclusiones, debiendo de haber un proceso para retractarse. Para facilitar esto, es necesario mantener un registro en la base de datos de la credibilidad y su justificación. Usando esta aproximación es posible explotar las redundancias en los datos experimentales para

mantener la verdad y así incrementar la confiabilidad del sistema.

3) Búsqueda en un espacio pequeño.

En general existen una gran cantidad de problemas que han sido enfocados en un contexto de espacio pequeño de búsqueda. En tales casos, una línea de razonamiento sencilla es suficiente de tal forma que no es necesario un retroceso. Estas técnicas han sido utilizadas en áreas de diseño, diagnóstico y análisis, y en la mayoría de los casos han usado una aproximación directa de búsqueda exaustiva. Esta no es un área natural para la IA, pero el uso de sistemas declarativos puede ser interesante desde el punto de vista de Ingeniería de Software.

4) Búsqueda en un espacio grande.

a) Generación y prueba jerárquica.- El razonamiento de eliminación exhaustiva que puede ser apropiado para un espacio pequeño de búsqueda es ineficiente en espacios grandes. A menudo la búsqueda en un espacio de estados se formula en términos de "Generación y Prueba" que es equivalente a un razonamiento por eliminación. En este tipo de metodología el sistema genera soluciones plausibles y un probador "podará" las soluciones que fallan de acuerdo a un criterio apropiado.

Una aproximación de generación y prueba jerárquica puede ser muy efectiva si se tiene la capacidad de evaluar soluciones candidatas que se especifican parcialmente. En tales casos es conveniente una poda temprana de ramas completas que representan clases enteras de soluciones asociadas con sus especificaciones parciales. De esta manera, es posible reducir masivamente la búsqueda requerida.

La generación y prueba jerárquica es una metodología apropiada para muchos problemas de interpretación y diagnóstico que incluyen una gran cantidad de datos. En tales casos, los generadores tienen que ser ideados para que puedan particionar el espacio de solución para admitir una temprana poda y así obtener las soluciones deseadas.

Como un caso concreto de aplicación tenemos el sistema GA1 [STEFIK 1978], cuya función es la interpretación de datos sobre las medidas de elementos moleculares para inferir estructuras moleculares completas.

b) Líneas múltiples de razonamiento.- Esta metodología se utiliza para ampliar la cobertura de una búsqueda incompleta. En este caso, los programas de búsqueda pueden decrementar las oportunidades de descartar una buena solución de evidencia débil para llevar un número limitado de soluciones en

paralelo, miéntras se aclara cual de las soluciones es la mejor.

5) Métodos para manejar un espacio grande por transformación de espacios.

a) Rompimiento del problema en subproblemas.- La aproximación de rompimiento de un problema en subproblemas puede ser manejado en dos clases. La primera corresponde a una aproximación que mantiene a los problemas sin interactuar. Este tipo de técnica es posible cuando las tareas que se realizan para alcanzar una meta no interactúan. Desafortunadamente son pocos los problemas del mundo real que se pueden clasificar bajo esta clase.

Los subproblemas que interactúan es una clasificación más interesante. En una gran mayoría de problemas complejos que se han desglosado en subproblemas, se ha visto que los subproblemas interactúan de tal forma que las soluciones válidas no pueden encontrarse independientemente. Sin embargo, tomando ventaja de los espacios pequeños de búsqueda, se han ideado algunas aproximaciones a lo anterior para poder tratar satisfactoriamente con tales interacciones.

1) Secuencia fija de subproblemas tal que no ocurra interacción.- Este tipo de técnica simplemente trata con la posibilidad de encontrar una partición ordenada para la cual no ocurran interacciones.

Un ejemplo ilustrativo es el sistema R1 [DERMOTT 1980], cuyo propósito es el de configurar sistemas de computación VAX. La tarea de configuración puede ser vista como una jerarquía de sub-tareas con una fuerte interdependencia temporal. R1 particiona la tarea de configuración en seis subtareas ordenadas y a cada subtarea se asocia un conjunto de reglas para cumplir con estas mismas. El sistema tiene alrededor de 300 reglas y una base da datos con la descripción de casi 400 componentes VAX.

2) El mínimo compromiso.- Esta técnica coordina la toma de decisiones con el aprovechamiento de información moviendo su enfoque de actividad del problema-solución entre los subproblemas disponibles. Las decisiones no son tomadas arbitrariamente o prematuramente sino son pospuestas mientras no exista bastante información.

El razonamiento basado en el principio de mínimo compromiso requiere de las siguientes disposiciones:

- La habilidad para conocer cuando hay bastante

información para tomar una decisión.

- La habilidad para suspender la actividad del problema-solución cuando la información no está disponible.

- La habilidad para moverse entre subproblemas reiniciando el trabajo con información que se vuelve disponible.

- La habilidad para combinar información de diferentes subproblemas.

Un ejemplo ilustrativo es el sistema MOLGEN [FEINGENBAUM 1981], cuyo propósito es diseñar experimentos de genética molecular.

3) Propagación de restricciones.- Esta técnica representa las interacciones por medio de restricciones. Las restricciones pueden ser visualizadas como descripciones parciales de entidades o como relaciones de submetas que deben ser satisfechas. La propagación de las restricciones es un mecanismo para mover información entre subproblemas. Por el hecho de introducir restricciones en lugar de elegir valores particulares, una solución esta dispuesta a seguir un estilo de mínimo compromiso del problema solución.

El sistema MOLGEN mencionado anteriormente, utiliza reglas para descubrir interacciones entre subproblemas vía la propagación de restricciones.

4) Conjeturas y razonamientos plausibles.- Las conjeturas es una parte inherente de búsqueda heurística, pero es particularmente importante con subproblemas que interactúan. En la aproximación de mínimo compromiso el proceso de solución deberá hacer un alto cuando tenga información insuficiente para decidir entre las elecciones que concurren. En tales casos, la conjetura heurística necesitará llevar el proceso de solución consigo. Si la conjetura es errónea, entonces el retroceso de dependencia directa puede ser usado para una recuperación eficiente.

Podemos mencionar algunas situaciones genéricas de razonamiento en las cuales las conjeturas son importantes:

- Muchas soluciones a problemas necesitan hacer frente con un conocimiento incompleto. El proceso puede ser incapaz de una mejor determinación de elección de la solución del problema.

- Un espacio de búsqueda puede ser completamente

denso en soluciones. Una conjetura es eficiente si las soluciones son abundantes, igualmente dispuestas y no hay necesidad de encontrar todas ellas.

- Algunas ocasiones se tiene una forma efectiva para convergencia de las soluciones, mejorando sistemáticamente la aproximación.

- La dificultad con conjeturas estriba en identificar conjeturas erróneas y la recuperación de esas eficientemente.

Uno de los sistemas diseñado para trabajar con conjeturas es el sistema EL [STALLMAN 1977], cuyo propósito es el análisis de estados estacionarios de circuitos de diodos, resistencias y transistores para determinar los valores de corrientes y voltajes.

b) Refinamiento jerárquico.- A menudo los aspectos más importantes de un problema son la abstracción y el desarrollo de soluciones alto nivel. Esta solución puede ser entonces refinada iterativamente de acuerdo a los detalles que se incluyen en el problema.

Esta técnica tiene muchas aplicaciones cuando el espacio de búsqueda en el nivel más alto es pequeño. El resultado de las soluciones en un alto nivel restringe la búsqueda a una porción más pequeña del espacio de búsqueda a nivel próximo inferior; de tal forma que cada nivel de solución puede ser prontamente encontrado. Este procedimiento es una técnica importante para prevenir explosiones combinatoriales en la búsqueda de una solución.

c) Resolución jerárquica dentro de subespacios que colaboran.- Algunos problemas pueden tener su espacio o solución jerárquica resueltó dentro de subespacios que contribuyen entre sí; es decir, los elementos de los espacios de alto nivel están compuestos de elementos de los espacio inferiores. Los subespacios heterogéneos resultantes son fundamentalmente diferentes del nivel tope del espacio solución. Sin embargo, las soluciones candidatos para cada nivel son muy útiles para restringir el rango de búsqueda a los niveles adyacentes. De esta forma estos candidatos actúan como una importante restricción para la explosión combinatoria.

6) **Métodos para manejar un espacio grande por desarrollo de alternativas o espacios adicionales.**

a) Empleo de modelos múltiples.- Frecuentemente la búsqueda para una solución utilizando un modelo global es muy difícil. El uso de modelos alternativos para el total o parte del problema puede simplificar grandemente la búsqueda.

El sistema SYN [KLEER 1980] es un ejemplo que combina los esfuerzos de modelos múltiples para emplear formas equivalentes de circuitos eléctricos.

b) Meta-razonamientos.- Es posible añadir extractos de espacios al espacio de búsqueda para así ayudar a decidir que hacer enseguida. Se puede pensar en ellos como estrategias y elementos tácticos que son elegidos entre varios métodos potenciales para decidir qué hacer en el nivel siguiente del problema.

Un ejemplo representativo es el sistema CRYSALIS [FEINGENBAUM 1981], cuya función es la interpretación automática de un mapa de densidad electrónica de proteínas, en donde para la hipótesis y pruebas de átomos y super-átomos utiliza meta-reglas con niveles de confianza asociados.

7) Tratamiento con el tiempo.

Poco se ha hecho en el área de sistemas expertos para tratar con el tiempo explícitamente. Los siguientes dos puntos son aproximaciones para tratar con el tiempo en términos de intervalo.

a) Cálculo Situacional.- El cálculo situacional fué una proposición en 1969 por Mc Carthy y Hayes para representar secuencias de acciones y sus efectos. Esta técnica usa el concepto de "situaciones" que cambian cuando acciones suficientes tienen que tomar lugar, o cuando nuevos datos indican un corrimiento situacional que es el apropiado. Las situaciones determinan el contexto para las acciones. Para describir una situación estereotipada nosotros podemos hacer uso de estructuras de datos para describir los cambios o la permanencia igual cuando las acciones son tomadas.

b) Planificación con restricciones de tiempo.- El sistema NOAH [SACERDOTI 1977] fue uno de los precursores de tipo de trazo en paralelo que trata con submetas interactuantes. El método de mínimo compromiso y encadenamiento hacia atrás inicialmente produce un orden parcial de operadores para cada plan. Cuando la interferencia entre planes con submetas sucede, el proyectista ajusta el orden de los operadores para resolver las interferencias y poder producir un plan paralelo final con un orden en el tiempo de los operadores.

El sistema DEVISER [FEINGENBAUM 1981] es una derivación

reciente del sistema NOAH que extiende su técnica de planificación en paralelo para tratar metas con restricción y duración de tiempo.

CAPITULO V

SISTEMAS ICAI

Introducción

Los sistemas ICAI (Intelligent computer-assisted instruction) que en español se traducen como programas de instrucción inteligentes asistidos por computadora, también se les conoce como sistemas tutoriales inteligentes. Estos sistemas se definen de la siguiente forma:

> "Son programas de computadora que utilizan técnicas de inteligencia artificial para ayudar a una persona a aprender" [KEARSLEY 1987].

Los sistemas ICAI se ubican en la intersección de las ciencias de la computación, la psicología cognitiva y la investigación educacional.

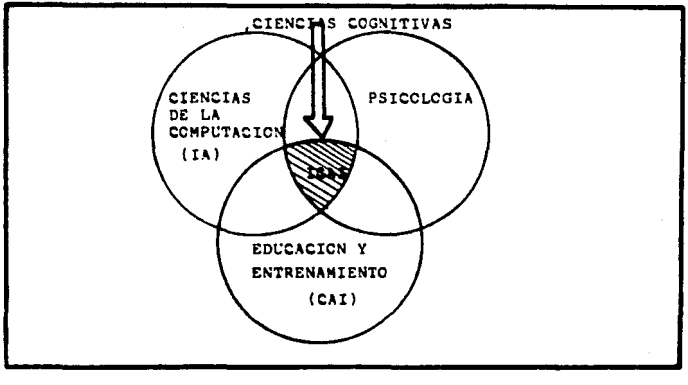


Figura 5.1. Areas que conforman los sistemas ICAI

El hecho de que la investigación ICAI se extienda sobre tres diferentes disciplinas, significa que hay diferencias muy grandes en terminología, metas de investigación, marcos teóricos y distinto énfasis entre investigadores, además de que estos requieren de un entendimiento mutuo de las tres disciplinas involucradas.

Por muchos años, los sistemas ICAI fueron confinados a laboratorios de investigación. Actualmente, con el crecimiento y la aceptación del campo de la inteligencia artificial, ha sido factible el introducir programas de IA en las áreas prácticas de entrenamiento y educación.

El objetivo de este capítulo es ofrecer un panorama general de los sistemas ICAI; definiendo los tipos de sistemas en que se clasifican, sus componentes y los desarrollos más representativos.

1. Componentes de un sistema ICAI

Los sistemas ICAI han tomado muchas formas pero esencialmente han separado los componentes principales de un mecanismo de instrucción, de tal forma que le permite al estudiante y al sistema, la flexibilidad de un ambiente de aprendizaje que se asemeja a lo que actualmente ocurre cuando el estudiante y el profesor se sientan juntos e intentan enseñar y aprender [ROBERTS 1983].

A pesar de la diversidad de arquitecturas que se manejan alrededor de los sistemas ICAI (ver anexo 3), es posible distinguir los elementos básicos comunes que los forman:

a) Módulo experto, el cual consiste de conocimientos sobre un dominio específico el cual el sistema intenta enseñar al estudiante.

   Este módulo se utiliza para generar el contenido instruccional, así como para evaluar el desempeño del estudiante. Los métodos de inteligencia artificial más representativos son las redes semánticas, sistemas de reglas de producción, representación procedural y la construcción de marcos o guiones. Para información detallada sobre estos modelos de representación de conocimiento, ver el capítulo dos de esta tesis.

b) Modelo del estudiante, el cual se utiliza para estimar el estado de conocimientos del estudiante y hacer hipótesis acerca de las estrategias de razonamiento utilizadas para alcanzar el estado actual de conocimientos.

Como la mayoría de los sistemas ICAI representan el conocimiento del estudiante como un subconjunto de la base de conocimientos de un sistema experto, el modelo se construye comparando el desempeño del estudiante con el desempeño de un sistema experto en la misma tarea. A esta técnica se le llama el "modelo de sobreposición" [CARR 1977].

Otra técnica es la de representar las tareas erróneas o mal aprendidas -las cuales no son subconjuntos de la base de conocimientos expertos - como variantes de este conocimiento. A esta técnica se le conoce como "modelo de depuración" [BROWN 1978].

El modelado del conocimiento del estudiante y el comportamiento de aprendizaje, básicamente utiliza dos rutinas:

- Trazar entre la estructura de conocimientos aquellas áreas en las que el estudiante a dominado o ha intentado aprender; y
- Aplicar patrones de reconocimiento al histórico de respuestas del estudiante para hacer inferencias acerca del entendimiento de la habilidad y el proceso de razonamiento utilizado para llegar a esa respuesta.

Para mantener el modelo del estudiante, existen cuatro fuentes principales de información [CLANCEY 1979]:

- El comportamiento del estudiante en la solución de problemas, observado por el sistema.
- Cuestionamientos directos al estudiante.
- Suposiciones basadas en la experiencia de aprendizaje del estudiante.
- Suposiciones basadas en alguna medida de la dificultad del material presentado.

Sin embargo, la mayoría de los sistemas utilizan unicamente las dos primeras fuentes de información para mantener el modelo del estudiante.

c) Módulo tutorial, el cual es un conjunto de especificaciones acerca de cual es el material que el sistema debe presentar y como y cuando debe hacerlo.

En los sistemas ICAI existentes, las estrategias de instrucción basicamente se representan por dos métodos:

- El método Socrático, el cual provee a los estudiantes con cuestionamientos para guiarlos a través de un proceso de depuración de sus propias mal interpretaciones. En el proceso de depuración, se asume que el estudiante razona acerca de lo que sabe y lo que no sabe y de esta forma modifica sus interpretaciones [CARBONELL 1970] y [STEVENS 1977].

- El método de entrenamiento, el cual provee a los estudiantes un ambiente en el cual los compromete en actividades como juegos por computadora con el fin de aprender habilidades relacionadas o habilidades generales para la solución de problemas. La meta del programa es tener a los estudiantes entretenidos y aprender como consecuencia de la diversión [BURTON 1979].

Los tres componentes no están totalmente desarrollados en cada sistema dado la complejidad y el tamaño de los sistemas ICAI. La mayoría de ellos se enfoca en el desarrollo de un solo componente de lo que constituye un sistema completo utilizable [CLANCEY 1979].

2. Tipos de sistemas ICAI

La figura 5.2 muestra graficamente los cinco tipos diferentes de sistemas existentes en la actualidad, los cuales son:

a) Tutores mezclados con iniciativas; Estos programas representan a los más antiguos sistemas ICAI, en donde, se compromete al estudiante a una conversación en dos sentidos y se intenta enseñar al estudiante vía el método Socrático con un descubrimiento guiado. Este tipo de sistemas se utilizan con mayor frecuencia en la enseñanza de habilidades conceptuales o procedurales. Un ejemplo representativo lo conforman los sistemas SCHOLAR [CARBONELL 1970] y SOPHIE [BROWN 1982].

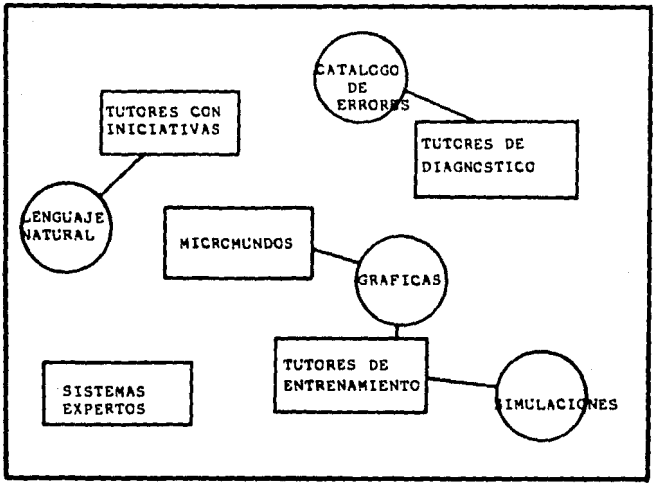


Figura 5.2. Tipos de sistemas ICAI

b) Tutores de entrenamiento; Un entrenador observa el desempeño del estudiante y lo provee de consejo que le ayuda a mejorar su desempeño. Los sistemas de entrenamiento se utilizan en programas del tipo solución de problemas como son simulaciones y juegos. Ejemplos de este tipo de sistemas son WEST [BURTON 1979] y WUSOR [GOLDSTEIN 1982].

c) Tutores de diagnóstico; Este tipo de sistemas intentan depurar el trabajo del estudiante. Los sistemas manejan un catálogo de errores que identifican los conceptos mal entendidos o faltantes que un estudiante puede tener al momento de resolver un problema. Los tutores de diagnóstico son apropiados para casi cualquier tipo de situación de resolución de problemas, aunque son sencillos de implantar para problemas con soluciones cerradas. Ejemplos de este tipo de sistemas son BUGGY [BROWN 1978] y PROUST [SOLOWAY 1983].

d) El concepto de micromundos; Este tipo de sistemas involucra el desarrollo de herramientas computacionales que le permiten al estudiante explorar dominios como la geometría, la física o la música. El mejor ejemplo conocido es LOGO [PAPERT 1980]. Este tipo de sistemas son los más cercanos al tradicional CAI.

e) Sistemas expertos; Este tipo de sistemas pueden ser utilizados como herramientas de trabajo y para proveer prácticas en la solución de problemas y toma de decisiones. Claramente se tiene un gran potencial de aplicaciones en el área de entrenamiento aunque el uso de sistemas expertos ya sea en entrenamiento o educación no ha sido muy difundido.

   La diferencia con los sistemas expertos convencionales es que los sistemas expertos ICAI incluyen un modulo adicional que contiene las reglas que controlan el proceso de enseñanza. Ver figura A3.3.

Al estudiar esta clasificación es importante tener en cuenta que cada tipo de sistema ICAI trata con un conjunto de elementos de la ciencia cognitiva e ignora los otros. Actualmente ningún tipo cubre todo lo concerniente a los sistemas ICAI. Sin duda alguna, en el futuro se generarán nuevos tipos de sistemas ICAI.

A pesar de sus grandes diferencias, la mayoría de los sistemas ICAI tratan con elementos y componentes similares. Por ejemplo, el elemento más importante en el desarrollo de un sistema ICAI es la forma apropiada de representar el conocimiento del que se trata. Otro elemento de gran importancia es como modelar el comportamiento actual del estudiante.

El compartir los conceptos y metodología proveen en los sistemas ICAI un objetivo común y una fundamentación teórica. Esta situación es análoga al período de la aviación en la que los aviones de propulsión a chorro sustituyeron a los aviones de hélice. La mayoría de los conocimientos acerca de volar fueron aún relevantes pero tuvieron que ser replanteados en términos de la nueva tecnología.

3. Desarrollo de los sistemas ICAI

Los sistemas ICAI tienen sus raíces durante los años 20's con el desarrollo de lo que se llamó "las máquinas de enseñanza". Estas máquinas fueron un intento de construir dispositivos interactivos de enseñanza. Skinner en los años 50's en sus trabajos sobre la

instrucción, aportó las bases de la metodología para la instrucción lineal programada tan difundida durante los años 50's y 60's. La instrucción programada fué el modelo para muchos esfuerzos iniciales en CAI (computer-assisted instruction).

Uno de los esfuerzos más representativos fué desarrollado en la Universidad de Stanford por el investigador Suppes [SUPPES 1968], en donde se demostró que en un ciclo regular añadiendo un poco de instrucción y sesiones de práctica con una computadora, se podría incrementar significativamente las habilidades del estudiante.

Durante los años 70's se desarrollaron varios proyectos tanto en universidades como institutos de entrenamiento (la mayoría se iniciaron en los sesentas). Entre los más significativos se encontraron SOLO, en la Universidad de Pittsburg, con el fin de enseñar el uso de computadoras como herramientas personales [DYWER 1974] y los sistemas PLATO y TICCIT que se desarrollaron con la finalidad de proporcionar sistemas de instrucción de bajo costo [BUNDERSON 1974].

Estos sistemas fueron probados en institutos y universidades y eventualmente se convirtieron en productos comerciales. Su principal beneficio fué el premitir a un gran número de investigadores el obtener experiencia práctica, así como, la apertura de una nueva era en los sistemas CAI con la aparición de las microcomputadoras [MERRIL 1980].

Otro desarrollo importante en los sistemas CAI se dio en la estructura de los componentes de instrucción. En sus primeros años, todos los componentes estaban combinados y almacenados en un mismo archivo. Esto implicaba que cualquier modificación al contenido o a las reglas de instrucción derivaba en practicamente un sistema nuevo. A principios de los 70's Seidel [SEIDEL 1971] desarrolló un proyecto para la armada americana llamado IMPACT en el cual el contenido de la materia (textos y gráficas) y las reglas de instrucción, estaban separados en diferentes archivos de datos. Asimismo, este sistema permitía que una variedad de actividades sucedieran en forma simultánea.

Aunque el prototipo del sistema IMPACT tenía la característica de una presentación sensible a las respuestas, el proceso de instrucción incluyendo la presentación de formatos y las interacciones entre la computadora y el estudiante, tenían que ser especificadas en un programa ántes de la instrucción, lo que traía un método muy rígido de enseñanza [STELZER 1972].

Con el fin de superar estas limitaciones (inherentes a las estructuras de formas pre-especificadas), aunado a la característica de generar nuevos problemas a partir de las combinaciones de diferentes elementos utilizando para ello técnicas modernas de representación de conocimiento, surgieron los sistemas ICAI (intelligent computer-assisted instruction).

El inicio de los sistemas ICAI lo marcó el sistema SCHOLAR desarrollado para enseñar geografía sudamericana [CARBONELL

1970]. La base de datos de SCHOLAR es compleja pero bien estructurada con la forma de una red de conceptos, datos y procedimientos. Los elementos de la red son unidades de información definiendo palabras y eventos en la forma de árboles multiniveles. El metodo de enseñanza es el diálogo Socrático. El sistema intenta primero diagnósticar los conceptos erróneos o mal interpretados y luego presenta material que forza al estudiante a ver sus propios errores. Los mecanismos de inferencia para contestar las preguntas de los estudiantes y evaluar sus respuestas son independientes del contenido de la red semántica y es aplicable a diferentes áreas de dominio.

Como extensión a SCHOLAR surge el sistema WHY [STEVENS 1977] diseñado para enseñar las causas del fenómeno de lluvia repentina, un complejo proceso geofísico que está en función de muchos factores no relacionados. WHY implanta el método tutorial socrático heurístico para describir las estrategias globales utilizadas por los seres humanos para guiar los diálogos.

El sistema SOPHIE [BROWN 1975], fué un intento por crear un "ambiente de aprendizaje reactivo" en donde el estudiante adquiere habilidades para la solución de problemas a base de probar sus propias ideas en lugar de recibir instrucciones por parte del sistema. SOPHIE incorpora un modelo del conocimiento del dominio junto con estrategias heurísticas para responder las preguntas de los estudiantes. SOPHIE permite a los estudiantes el tener una relación uno a uno con una computadora experta la cual les ayuda a generar sus propias ideas, experimentar con ellas y depurarlas cuando es necesario.

Los principios de SOPHIE se aplicaron para construir un modelo de diagnóstico para el aprendizaje de cuestiones básicas en la solución de problemas matemáticos conocido como BUGGY [BROWN 1978]. Este sistema provee los mecanismos para explicar el porque un estudiante estaba cometiendo un error en contrapartida de únicamente identificarlos y permitia una estrategia de enseñanza basada en un nuevo modelo conocido como de entrenamiento. Este modelo permitia identificar las estrategias de diagnóstico requeridas para inferir el conocimiento faltante del estudiante a partir del comportamiento observado.

Este modelo de entrenamiento apareció también en el sistema WEST [BURTON 1979], diseñado para enseñar la manipulación apropiada de expresiones aritméticas. Otro ejemplo lo tenemos con el sistema WUSOR [GOLDSTEIN 1977], diseñado para fomentar la habilidad del estudiante en el manejo de inferencias lógicas y probabilísticas a partir de cierta información básica.

Entre otros sistemas de diagnóstico encontramos a LMS (Leeds Modeling System) [SLEEMAN 1982] que intenta predecir el comportamiento del estudiante utilizando una representación basada en reglas de producción.

El sistema GUIDON [CLANCEY 1979], diseñado para enseñar la resolución de problemas de diagnóstico médico es diferente a

otros sistemas ICAI de diagnóstico en términos de una nueva estrategia en los diálogos entre el sistema y el estudiante. GUIDON utiliza una estrategia de enseñanza bien estructurada acorde a la última respuesta del estudiante, así como, preguntas y respuestas en forma repetitiva. Las reglas de enseñanza son organizadas en rutinas específicas, miéntras que las reglas para el diagnóstico médico son agrupadas jerarquicamente en un sistema aparte llamado MYCIN [SHORTLIFFE 1976].

GUIDON no tuvo éxito como sistema de enseñanza, sin embargo, mostró que una base de conocimientos de un sistema experto no podía ser utilizada como base de conocimientos de un sistema ICAI a menos que existiera un suplemento por otros niveles de conocimiento que ayudaran a explicar y organizar el conocimiento durante el proceso de enseñaza. Como resultado de estas investigaciones surge el sistema NEOMYCIN [CLANCEY 1981].

Otro avance importante en la aplicación de técnicas de IA en el campo de los sistemas ICAI fué el crear nuevos ambientes educacionales a través del control total de la experiencia aprendida por el estudiante. Como ejemplos de estos sistemas tenemos a LOGO [PAPERT 1980] y SMALLTALK [KAY 1977].

Un aspecto interesante de los sistemas tutoriales fué el desarrollo de características auto-mejorables. Este tipo de sistemas tenían dos grandes componentes: un programa de enseñanza adaptativo que se expresa en reglas de producción y un componente auto-mejorable que lleva a cabo cambios experimentales en las reglas de producción del programa de enseñanza. Este trabajo fué particularmente importante por su naturaleza adaptativa. Como ejemplos de esta línea de investigación tenemos el tutor cuadrático auto-mejorable [O'SHEA 1979] y el tutor de integracón simbólica [KIMBALL 1973].

La mayoría de los sistemas ICAI existentes se describen en [SLEEMAN 1982] a excepción de una línea de investigación basada en la teoría ACT [ANDERSON 1982]. ACT fué la base para el desarrollo de sistemas tutoriales para la enseñanza de LISP. Estos sistemas simulaban el comportamiento del estudiante utilizando un modelo cognitivo y no las técnicas de inteligencia artificial.

Se presenta un resúmen de los principales sistemas ICAI en la tabla 5.1 [FLETCHER 1984]. En ella se comparan los sistemas en términos de la materia que enseñan, los métodos para representar el conocimiento, el modelo del estudiante y la estrategia tutorial. La tabla incluye la referencia principal a cada sistema.

SISTEMA	MATERIA	BASE DE CONOCIMIENTOS	MODELO DEL ESTUDIANTE	MODULO TUTORIAL	REFERENCIA
SCHOLAR	Geografía	Redes Semánticas	Sobreposición con pesos relativos	Diálogo Socrático	Carbonell, 1970
WHY	Causas de las tormentas	Guiones	Identificación de malas interpretaciones	Diálogo Socrático	Stevens, 1982
INTEGRATE	Integración simbólica	Reglas de producción	Sobreposición	Ambiente reactivo con asesor	Kimball, 1982
SOPHIE	Localización de fallas electrónicas	Redes semánticas con simulador de circuitos	Sobreposición	Ambiente reactivo con interacciones guiadas	Brown, 1982
WEST	Expresiones aritméticas	Reglas de producción	Sobreposición	Ambiente reactivo con entrenador	Burton, 1979
BUGGY	Operaciones de restas	Red procedural	Identificación de malas interpretaciones	Ambiente reactivo con asesor	Brown, 1978
WUSOR	Relaciones lógicas	Red de gráficas genéticas	Sobreposición con pesos relativos	Ambiente reactivo con entrenador	Goldstein, 1982
EXCHECK	Lógica y teoría de conjuntos	Reglas de producción con intérprete de lógica	Sobreposición	Ambiente reactivo con asesor	Suppes, 1981

Tabla 5.1. Sistemas Tutoriales Inteligentes (resúmen).

SISTEMA	MATERIA	BASE DE CONOCIMIENTOS	MODELO DEL ESTUDIANTE	MODULO TUTORIAL	REFERENCIA
BIP	Programación en BASIC	Reglas de porducción	Sobreposición	Ambiente reactivo con red curricular y asesor	Barr, 1976
SPADE	Programación en LOGO	Reglas de producción	Sobreposición	Ambiente reactivo con entrenador	Miller, 1982
ALGEBRA	Algebra aplicada	Reglas de producción	Sobreposición	Ambiente reactivo con entrenador	Lantz, 1983
LMS	Procedimientos algebraicos	Reglas de producción	Reglas de producción de diagnóstico	Ambiente reactivo sin tutor	Sleeman, 1982
QUADRATIC	Ecuaciones cuadráticas	Reglas de producción auto mejorable	Sobreposición	Ambiente reactivo con asesor	O'Shea, 1982
GUIDON	Enfermedades infecciosas	Reglas de producción	Sobreposición con aplicación de probabilidad	Ambiente reactivo con interacciones estructurales	Clancey, 1982
PROUST	Programación en PASCAL	Redes semánticas	Identificación de malas interpretaciones	Ambiente reactivo con asesor	Soloway, 1983
STEAMER	Equipos de propulsión a vapor	Red procedural con modelo por dispositivo	Sobreposición con el modelo del dispositivo	Ambiente reactivo con asesor	Williams, 1981

Tabla 5.1. Sistemas Tutoriales Inteligentes (continuación).

4. Los sistemas ICAI del futuro

Al revisar los componentes de un sistema ICAI, así como su desarrollo histórico, es fácil observar que la mayoría de los sistemas ICAI existentes, han sido desarrollados para explorar la capacidad de las técnicas de inteligencia artificial en el proceso de instrucción en lugar de construir un sistema de instrucción efectivo.

Aunque los sistemas ICAI han tomado muchas formas distintas, dependiendo de las metas de desarrollo, las características de la base de conocimientos, el tipo de estudiantes y los métodos técnicos utilizados, sus características y procesos pueden ser representados con un modelo típico. El desempeño de un estudiante es evaluado comparándolo con un desempeño experto (simulado por computadora). Las diferencias entre estos desempeños se utiliza para inferir las causas que generan el problema en el estudiante (sus conceptos erróneos o malas interpretaciones) y definen sus necesidades de aprendizaje. El proceso interactivo entre la computadora y el estudiante (incluyendo la presentación de explicaciones para corregir las malas interpretaciones del estudiante) son determinadas por reglas tutoriales.

Este modelo típico sugiere que la mayoría de los sistemas actuales tienen varios inconvenientes como sistemas completos de instrucción, como los siguientes:

a) El proceso de evaluación del desempeño del estudiante es solamente dependiente del análisis de su respuesta a una pregunta dada. Esta evaluación es muy útil para diagnosticar sus problemas y definir sus necesidades de aprendizaje. Sin embargo, no es apropiado para medir un desempeño general en la tarea porque la evaluación de la respuesta está limitada a un aspecto específico de la misma. Además, no toma en cuenta la información disponible en la memoria a largo plazo del estudiante porque la evaluación se basa en el análisis de una respuesta inmediata. De ahí que los sistemas ICAI existentes tengan mecanismos de diagnóstico pero no componentes adecuados para la medición y evaluación.

b) Los sistemas ICAI existentes han ignorado muchas variables potencialmente importantes en el proceso de diagnóstico y prescripción confiando unicamente en la respuesta del estudiante a una pregunta dada. En la mayoría de las situaciones de aprendizaje, no se desarrolla un patrón confiable de la respuesta del estudiante hasta que éste ha obtenido un avance significativo en esa tarea dada.

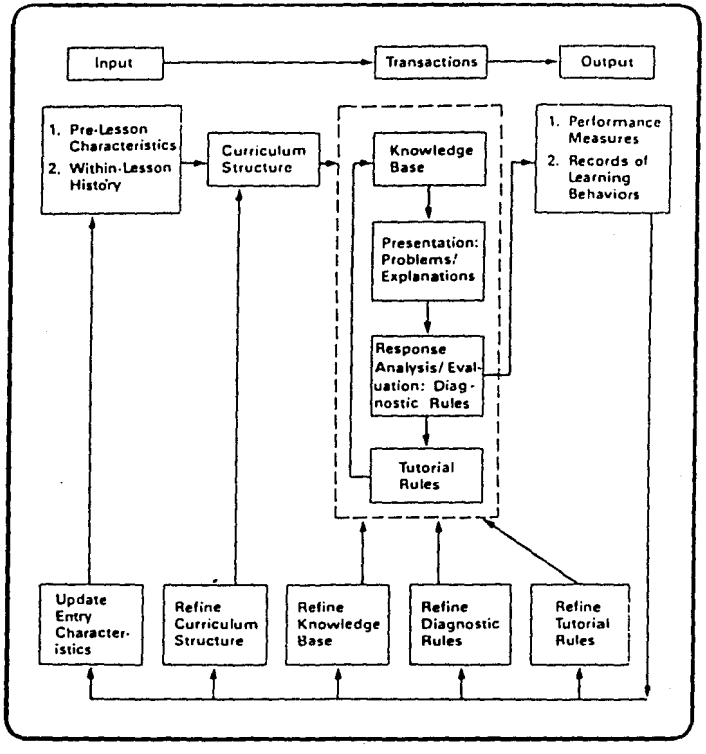


Figura 5.3. Modelo de Instrucción Adaptativa [PARK 1980].

c) La capacidad de generar nuevas pantallas de instrucción está limitada a un conjunto específico de unidades de información en la base de conocimientos. Muy pocos sistemas, si es que hay alguno, tienen la habilidad para refinar la estructura de la base de conocimientos y las reglas tutoriales y de diagnóstico.

Con el fin de sobreponer los inconvenientes anteriormente anotados en el desarrollo de sistemas futuros se presenta un modelo conceptual de instrucción adaptativa. (ver figura 5.3).

Este modelo divide el proceso de instrucción en tres escenarios: entrada, transacción y salida. El escenario de entrada basicamente consiste del análisis de las características iniciales del estudiante. Estas, incluyen no unicamente la información historia de sus respuestas sino también información acerca de las aptitudes del estudiante (por ejemplo, su habilidad intelectual, estilo de apredizaje, conocimiento anterior, etc.).

Estas características iniciales serían actualizadas con la información producida por el proceso de evaluación del desempeño, es decir, con la información del escenario de salida.

El escenario de transacción consiste de las interacciones entre el estudiante y el sistema. Con la base de las características iniciales del estudiante, el sistema selecciona problemas o explicaciones para su presentación; posteriormente el sistema evalúa la respuesta del estudiante al problema dado. La evaluación de la respuesta provee la información para diagnosticar las necesidades de aprendizaje y para medir el desempeño general en la tarea. Las necesidades de aprendizaje son inferidas de acuerdo a las reglas de diagnóstico. Finalmente, el sistema selecciona nuevas presentaciones y cuestionamientos para el estudiante acorde a las reglas tutoriales.

El escenario de salida consiste principalmente de la evaluacion del desempeño. Esta evaluación puede incluir no únicamente una evaluación general en una tarea dada, sino también un análisis completo del comportamiento de aprendizaje grabado en el sistema. Acorde a la evaluación de desempeño los componentes de instrucción son modificados o actualizados. Los componentes de instrucción a actualizar puede incluir contenidos en la base de conocimientos, estrategias de instrucción, reglas de diagnóstico o tutoriales, estructura del curso y características de entrada. Si el sistema no tiene capacidad para actualizar o modificar estos componentes en forma automática, debe permitir el desarrollar esa tarea en forma manual.

CAPITULO VI

DESCRIPCION DEL SISTEMA EDA

Introducción

En los capítulos anteriores se describió en forma detallada el contexto sobre el cual giran los desarrollos de sistemas ICAI. Los temas importantes son aprendizaje, conocimiento, sistemas expertos, sistemas ICAI e inteligencia artificial. El objetivo de este capítulo es describir el proyecto del sistema EDA, que se presenta como un sistema tutorial inteligente en el área de estructuras de datos.

1. Consideraciones Generales del Sistema EDA

1.1. Estrategia de Aprendizaje

Recordando lo expuesto en el capítulo III de la tesis en materia de aprendizaje, es importante hacer notar lo siguiente:

El aprendizaje adulto se logra mediante actividades de exploración, intentando en forma activa descubrir las relaciones entre las acciones y los resultados, pero más a menudo consiste en la adquisición de nuevo conocimiento bajo la guía de un instructor o libro. El papel del instructor es indicar las cuestiones relevantes y guiar la adquisición de conocimiento importante.

A partir de estos planteamientos el objetivo en materia de aprendizaje del sistema EDA quedó de la siguiente forma:

> Desarrollar un sistema tutorial que sirva como material de apoyo complementario a la enseñanza y que a su vez le permita al estudiante:
>
> . Desarrollar sus propias estrategias e hipótesis

del material a aprender.

. Descubrir por si mismos las relaciones importantes entre las acciones y los resultados.

. Tener una fuente de consulta que le indique las cuestiones relevantes y lo guie en la adquisición de conocimiento importante.

En ningún momento se pensó en utilizar como método de enseñanza la presentación de textos que conforman la materia a enseñar, la práctica en el empleo de ese conocimiento y la comprobación contra patrones de comportamiento establecidos para asegurar que éste se ha adquirido realmente. Esta estrategia considerada como tradicional, corresponde más bien a los inicios de los sistemas de Educación Asistida por Computadora.

La labor del profesor al impartir o explicar una clase de la materia objeto de la enseñanza es fundamental. El sistema EDA no pretende sustituir la labor del profesor, sino proporcionar una herramienta de apoyo que facilite y ayude su labor de enseñanza.

1.2. ¿ EDA: Un sistema experto o un sistema ICAI ?

Tomando como referencia lo expuesto en los capítulos II y III de la tesis, podemos mencionar que los seres humanos adquieren conocimiento en dos formas complementarias.

En la primera, los tópicos se estudian formalmente, ya sea en la escuela o a través de lecturas o libros de textos. Como resultado de este estudio el conocimiento se acumula en forma de definiciones, axiomas o leyes.

En la segunda, el conocimiento se adquiere por experiencia o por enseñanza de un tutor. El tutor (o la experiencia), usualmente enseña al aprendiz la utilización de reglas para analizar tareas o resolver problemas, de esta forma el aprendiz se comporta como si conociera las teorías generales cuando en realidad aplica teorías específicas dentro de un dominio.

Los investigadores en inteligencia artificial se refieren a la heurística y a las teorías específicas dentro de un dominio como un "conocimiento superficial". La mayoría de los sistemas expertos utilizan este conocimiento que generalmente es suficiente para resolver un problema. Los principios básicos y las teorías generales que un experto utiliza cuando se enfrenta a problemas realmente difíciles se conoce como un "conocimiento profundo" [HARMON 1985].

Este conocimiento profundo, desde el punto de vista de la psicologia, corresponde a un esquema de memorización, el cual tiene las siguientes caracteristicas:

a) La velocidad de respuesta es muy importante, es decir, es más importante la velocidad de respuesta que su exactitud.
b) Los pequeños errores no tienen grandes consecuencias.
c) La lectura de instrucciones interfiere con el desempeño de la tarea.
d) El tipo de aplicación requiere de respuestas memorizadas, es decir, las tareas no implican un proceso complejo de toma de decisiones.

La aplicación del conocimiento profundo, el cual se deriva de la educación, provee un punto de vista muy amplio de un área y sus inter-relaciones con otras areas.

El sistema EDA, conforma su base de conocimientos con definiciones y principios básicos en un marco de teoria general orientada a un esquema de memorización. En él, se pretende que el alumno razone -- mediante la presentación de textos guiados y la ejecución de herramientas de experimentación -- acerca de los principios de operación de los algoritmos de ordenamiento y búsqueda y no que funcione como herramienta de consulta que provea de mecanismos para la toma de decisiones.

Sin embargo, es factible como extensión al sistema, el generar un módulo experto orientado a entrenamiento para la selección de algoritmos bajo condiciones reales de operación.

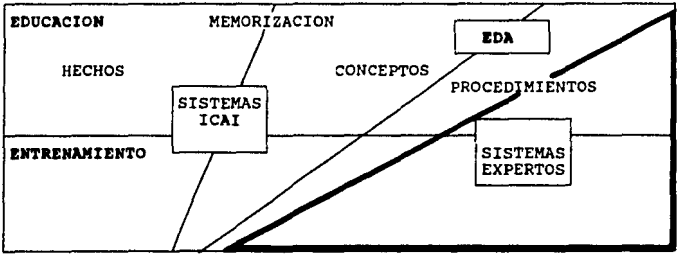


Figura 6.1. Relación de los sistemas ICAI y Expertos por área de aplicación.

La figura 6.1 tomada de la fuente [HARMON 1985] nos muestra la relación entre los sistemas ICAI y los sistemas expertos en función de las área de entrenamiento y educación. En esta figura, ubicamos al sistema EDA.

Tomando en cuenta los componentes que conforman a los sistemas tutoriales inteligentes (ver capitulo V), el sistema EDA quedó estructurado de la siguiente forma:

a) Componente experto.- Debido a la complejidad del material a presentar, el alto volumen de información y la necesidad de mantener un orden lógico en su presentación, optamos por utilizar el modelo de representación de conocimiento de redes semánticas.

   La información correspondiente a cada algoritmo quedó estructurada en bloques de información de 15 líneas. El número de bloques por algoritmo varía de acuerdo a su complejidad. Cada bloque corresponde a un nodo de la red y el arco que liga a dos nodos corresponde a la acción de una tecla.

b) Componente tutorial.- El cual es un conjunto de especificaciones acerca de cual debe ser el material a presentar; como y cuando hacerlo.

   EDA utiliza una estrategia mixta; por un lado emplea un método socrático para indicarle al alumno las cuestiones relevantes y lo guie en la adquisición de conocimiento importante (esta guia va mezclada con la presentación del conocimiento del módulo experto) y por el otro se genera un "ambiente de aprendizaje reactivo".

   EDA estructura su ambiente de aprendizaje reactivo dándole la libertad al alumno de seleccionar los algoritmos con los que desea trabajar, que experimentos quiere llevar a cabo, que características deben tener los datos, etc...

   Este ambiente le permite desarrollar sus propias estrategias e hipótesis del material a aprender y descubrir por si mismo las relaciones importantes entre las acciones y los resultados.

c) Modelo del estudiante.- El modelo del estudiante nos permite estimar el grado de avance del estudiante en la adquisición de conocimiento. En este trabajo no se desarrolló ya que el ambiente de aprendizaje reactivo le da toda la libertad al alumno de escoger el material a aprender y esto hace que sea muy complejo estimar el grado de avance.

   Por ejemplo, si un estudiante decide comenzar el curso estudiando el algoritmo de Quick sort, no es posible saber si el estudiante lo hizo porque le interesa profundizar en ese

algoritmo en particular y ya conoce los otros o simplemente le llamó la atención el nombre.

Es importante hacer notar que si el alumno no conoce de métodos de ordenamiento puede emplear la estrategia sugerida por el método socrático.

2. Sistema EDA

2.1. Descripción Funcional

El desarrollo de aplicaciones en computadoras personales, se ha caracterizado en últimas fechas por el uso de una interfaz hombre-máquina muy amigable con el usuario. Esta interfaz utiliza colores, ventanas y una opción de ayuda en línea en todo momento. EDA emplea una interfaz de este tipo.

El menú principal esta formado por tres módulos generales que son: el módulo de datos, el módulo de ordenamiento y el módulo de búsqueda. La figura 6.2 muestra un diagrama a bloques del sistema EDA.

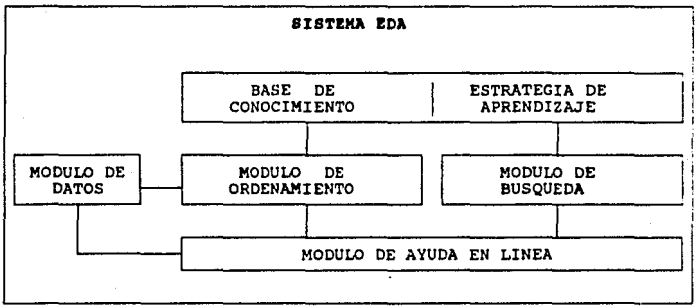


Figura 6.2. Diagrama a bloques del sistema EDA.

2.1.1. Módulo de Datos

El módulo de datos está ligado al módulo de ordenamiento. Su finalidad es conformar el vector de datos con el cual los algoritmos de ordenamiento van a trabajar.

El usuario tiene la libertad de seleccionar el número de datos (el rango va de 5 a 300 elementos) y las características de los mismos (definidos por el usuario, ya ordenados, aleatorios, semiordenados).

2.1.2. Módulo de Ordenamiento

La finalidad de este módulo es permitirle al alumno seleccionar alguno de los algortimos de ordenamiento que incluye el sistema EDA.

Los algoritmos con los que opera este módulo son:

a) En el grupo de intercambio.- Burbuja, shaker sort y quick sort.

b) En el grupo de inserción.- Directa, binaria y shell.

c) En el grupo de selección.- Directa y heap sort.

Al seleccionar alguno de los algoritmos, el alumno tiene la opción de ejecutar una de las siguientes tres operaciones: Algoritmo, simulación o animación gráfica. Estas operaciones se describirán a detalle posteriormente.

La última opción de este menú, llamada comparativo, le permite al alumno ejecutar la operación de animación gráfica en todos los algoritmos con los mismos datos. Al final el sistema muestra una tabla resúmen indicando para cada algoritmo el tiempo requerido para su ejecución y un índice con respecto al más rápido de todos.

2.1.3. Módulo de Búsqueda

La finalidad de este módulo es permitirle al alumno seleccionar uno de los algortimos de búsqueda incluido en EDA.

Los algoritmos con los que opera este módulo son:

a) En búsqueda por comparación de llaves.- Secuencial, auto-organizable y binaria.

b) En búsqueda por transformación de llaves.- Cuadrado medio, doblados, división y corrimientos.

c) En el manejo de colisiones.- Lineal, cuadrático, doble hash, cubetas y encadenamiento fusionado.

Al seleccionar cualquiera de los algoritmos, el alumno tiene la opción de ejecutar alguno de las siguientes operaciones: Algoritmo o simulación. Estas operaciones se describirán a detalle posteriormente.

2.1.4. Módulo de Algoritmo

El módulo de algoritmo tiene la finalidad de mostrar al alumno la información de la base de conocimientos referente a un algoritmo en particular.

El texto para cada algoritmo se compone de los siguientes módulos:

a) Presentación general del algoritmo en cuestión; en el algoritmo burbuja se incluye también una presentación general del tema métodos de ordenamiento, y en el primer algoritmo de cada grupo del módulo de búsqueda se incluye una introducción general al tema correspondiente.

b) El código fuente desarrollado en lenguaje Pascal o en su defecto pseudocódigo.

c) Comentarios sobre su operación (cuando es eficiente, cuando no, caso crítico, tipo de operaciones involucradas, etc..).

d) La secuencia de teclas de función necesaria para observar la animación del algoritmo en pantalla (en los algoritmos de ordenamiento) o la simulación (en los algoritmos de búsqueda).

2.1.5. Módulo de Simulación

La finalidad de este módulo es presentarle al alumno paso a paso la secuencia de operaciones que se llevan a cabo en cada uno de los algoritmos.

La simulación puede ejecutarse en dos modos diferentes, total o parcial. En el modo total, todas las operaciones que se llevan a cabo en la simulación, aparecen a manera de comentarios en la parte inferior de la ventana. En el modo parcial, unicamente se muestra los números o registros involucrados en las operaciones sin hacer referencia al tipo de operación. El sistema da la facilidad de cambiar el modo de operación al oprimir cualquier tecla.

La simulación en los algoritmos de ordenamiento se lleva a cabo con un vector de 8 números. Este vector constituye los datos para la simulación y es el mismo para todos los algoritmos.

La simulación en los algoritmos de búsqueda se lleva a cabo por medio de dos tablas. Una tabla de 7 llaves que constituyen los datos de la simulación y una tabla de 15 registros. Esta tabla de registros nos permite hacer búsquedas de las llaves en los algoritmos de búsqueda o nos permite insertar estas en los algoritmos de hash. En la simulación de los algoritmos para el manejo de colisiones, se incluye otra variable que es el factor de carga. Dependiendo del factor de carga seleccionado se generan más o menos registros ocupados en la tabla de registros.

2.1.6. Módulo de Animación Gráfica

El módulo de animación gráfica unicamente está disponible para los algoritmos de ordenamiento. Esto se debe a que estos algoritmos tienen caracteristicas que son dificiles de comprender (por ejemplo recursividad, intercambios excesivos, manejo de arboles, etc.), y la aplicación de esta técnica nos permite facilitarle al estudiante el entendimiento de estas caracteristicas que muchas veces pasan desapercibidas.

Su finalidad es mostrarle al alumno de una forma gráfica el comportamiento de cada algoritmo. Se utiliza una representación de barras verticales en donde cada barra representa a un elemento del vector de datos y el tamaño de la barra a su valor numérico.

En los algoritmos de ordenamiento, lo más costoso en términos de tiempo de procesador son los intercambios. Para que el alumno pudiera observar como algunos algoritmos realizan muchos

intercambios, éstos se representan como una barra de otro color.

Al finalizar la animación, el sistema muestra el tiempo que se requirió para ordenar el vector de datos.

2.2. Implantación

Con el fin de hacer accesible el sistema EDA a la mayor cantidad posible de alumnos de las carreras relacionadas con la computación, aunado a la proliferación de equipos microcomputadores en la instituciones de enseñanza superior y centros de instrucción, seleccionamos una computadora PC compatible con IBM.

Como método para atraer la atención del alumno en las relaciones importantes entre las acciones y los resultados se utiliza el manejo de colores. Esto implica que una restricción en el uso de este programa es la disponibilidad de una interfaz CGA (Color Graphics Adapter).

El sistema ocupa un diskette de 5 1/4 pulgadas y por ello no requiere del uso de un disco duro. La cantidad mínima de memoria RAM para ejecutarlo es de 128 KB.

El sistema fué desarrollado en lenguaje Pascal (Turbo Pascal versión 4.0) en aproximadamente 3000 líneas de código. No se utilizó como librería ningún paquete comercial.

En el anexo 4 se muestran algunas pantallas de la interfaz hombre-máquina del sistema EDA.

En el anexo 5 se incluye la base de conocimientos del sistema EDA.

CONCLUSIONES

El objetivo de este capítulo es puntualizar la experiencia adquirida durante el desarrollo de este trabajo y sugerir posibles extensiones del sistema EDA.

1. Experiencia Adquirida

Como resultado del presente trabajo se obtuvo una fuente de consulta general que esclarece los conceptos de:

- Inteligencia Artificial,
- Conocimiento,
- Aprendizaje,
- Sistemas Expertos y
- Sistemas ICAI.

Para cada uno de estos temas se incluyen las referencias más importantes. Se realizó una extensa consulta bibliográfica para lograr una síntesis general del estado del arte existente a la fecha.

Se puso en práctica estos conceptos con el desarrollo de un sistema tutorial inteligente en el área de estructuras de datos al que llamamos sistema EDA.

Entre sus características más importantes podemos mencionar las siguientes:

- Ser una herramienta de apoyo que facilita y ayuda la labor de enseñanza en particular para el área de informática.
- Permitirle al estudiante desarrollar sus propias estrategias e hipótesis del material a aprender, así como, descubrir por sí mismo las relaciones importantes entre las acciones y los resultados.
- Ser una fuente de consulta que indica las cuestiones relevantes en materia de algoritmos de ordenamiento y búsqueda.

Un sistema tutorial inteligente requiere de tiempo, experiencia, maduración y experimentación por lo que consideramos que el sistema EDA está en la etapa de prototipo.

En el desarrollo de este tipo de sistemas es importante involucrar desde la fase de diseño a expertos en el tema a impartir y a los psicólogos o pedagogos expertos en materia educacional, esto implica que la aplicación de técnicas de IA deben estar más enfocada hacia la efectividad instruccional, es decir, estas deberán ser seleccionadas acorde a las necesidades de instrucción.

Para que esta área de la inteligencia artificial incremente su importancia, los esfuerzos deberán seguir enfocándose a la utilización de equipos microcomputadores.

2. Futuros Desarrollos

A partir de toda la documentación consultada y como resultado del desarrollo del sistema EDA, propongo las siguientes vías de desarrollo entorno a EDA:

- Llevar a cabo un experimento controlado en salones de clase con el fin de medir la efectividad del sistema EDA como herramienta de apoyo a la enseñanza.

- Desarrollar un módulo que permita medir el grado de avance del estudiante; sería conveniente involucrar a psicólogos en este desarrollo. Una característica deseable sería que este modelo fuera capaz de descubrir inconsistencias en la base de conocimientos del alumno.

- Ampliar el espectro del contenido aplicando EDA a otros temas. Por ejemplo, análisis numérico, investigación de operaciones, finanzas, compiladores, etc.

- Incluir como un subíndice del menú de ordenamiento (algoritmo, simulación y animación gráfica) la información referente a la definición de los algoritmos de ordenamiento en términos de tipos de datos abstractos.

- Desarrollar un módulo de ordenamiento orientado a

los métodos externos. Este módulo formaría parte del menú principal de EDA y podría contemplar los métodos de mezcla (natural, balanceada por múltiples caminos, polifase, cascada y oscilante).

- Desarrollar un sistema experto para la simulación de casos reales. Este sistema experto podría generar escenarios a partir de condiciones reales en forma aleatoria. Las variables a modificar podrían ser el volumen de datos (poco volumen para métodos internos y grandes volumenes para métodos externos), las características de los datos (casi ordenados, aleatorios, etc.), el tipo de las llaves (numéricas, alfabéticas, etc.), el lenguaje del que se dispone, el número de veces a ejecutar el ordenamiento, etc.

En el presente trabajo se instrumentó un sistema EDA básico que permitirá continuar los desarrollos futuros de herramientas y apoyar la enseñanza en las aulas de las universidades del país.

REFERENCIAS

[AIKINS 1983]
"Prototypical Knowledge for Expert Systems"
Janice S. Aikins
Artificial Intelligence 20, North-Holland, 1983

[ALLEN 1987]
"Natural Language Understanding"
James Allen
The Benjamin/Cummings Publishing Company, Inc., 1987

[ANDERSON 1982]
"Acquisition of cognitive skill"
Anderson, J. R.
Psychological Review, 89, 1982

[AYALA 1985]
"Uso de sistemas expertos en la educación"
Gerardo Ayala
Facultad de Ingeniería, UNAM, 1985

[BEGG 1987]
"Authoring Systems for ICAI"
Iain M. Begg & Ian Hogg
Artificial Intelligence & Instruction, Kearsley 1987

[BIC 1986]
"Learning from AI: New Trends in Database Technology"
Lubomir Bic, Jonathan P. Gilbert
University of California, Irvine, IEEE, Mar 1986

[BOBROW 1977]
"GUS: A frame-driven dialog system"
D.G. Bobrow, R.M. Kaplan, D.A. Norman, H. Thompson and T. Winograd
Artificial Intelligence, Vol. 8 no 1., Feb 1977

[BROWN 1975]
"SOPHIE. A step towards a reactive learning environment"
Brown, J. S., & Burton, R. R., & Bell, A. G.
International Journal of Man-Machine Studies, 7, 1975

[BROWN 1978]
"Diagnostic models for procedural bugs in basic mathematical skills"
Brown, J. S., & Burton, R. R.
Cognitive Science, 2, 1978

[BROWN 1982]
"Pedagogical, natural lenguage and knowledge engineering in SOPHIE I, II and III"
Brown, J. S., & Burton, R. R., & de Kleer, J.
Intelligent Tutoring Systems, Sleeman 1982

[BROWN 1987]
"Algorithm Animation"
Marc H. Brown
ACM Distinguished Dissertations, MIT Press, 1987

[BUNDERSON 1974]
"The design and production of learned-controlled courseware for the TICCIT system"
Bunderson, C. V.
International Journal of Man-Machine Studies, 6, 1974

[BURTON 1979]
"An investigation of computer coaching for informal learning activities"
Burton, R. R., & Brown, J.S.
International Journal of Man-Machine Studies, 11, 1979

[CARR 1977]
"Overlays: A theory of modeling for computer aided instruction"
Carr, B., & Goldstein, I.P.
AI Lab., Massachusetts Institute of Technology, memo 406, 1977

[CARBONELL 1970]
"AI in CAI: An artificial intelligence approach to computer assisted instruction"
Carbonell, J.R
IEEE Transactions on Man-Machine Systems, 11, 1970

[CHANCE 1981]
"Aprendizaje y Conducta"
Paul Chance
Editorial Tecnos, S.A., Madrid, 1981

[CHAPA 1984]
"Arquitectura de Sistemas Expertos"
Chapa V. S.
Instituto Politécnico Nacional, 1984

[CHARNIAK 1985]
"Introduction to Artificial Intelligence"
Eugene Charniak and Drew McDermott
Addison-Wesley, 1985

[CHERNIACK 1977]
"Pulmonary Function Testing"
Cherniack, R.N.
Saunders, Philadelphia, 1977

[CLANCEY 1979]
"Tutoring rules for guiding a case method dialogue"
Clancey, W. J.
International Journal of Man-Machine Studies, 11, 1979

[CLANCEY 1981]
"NEOMYCIN: Reconfiguring a rule-based expert system for application to teaching"
Clancey, W. J., & Letsinger, R.
Proceedings of the Seventh IJCAI, 1981

[COLLINS 1985]
"Introduction to Knowledge Base Systems"
R. A. Frost Collins
Academic Press, New York, 1985

[COTTON 1976]
"Models of learning"
Cotton, J. W.
Anuual Review of Psychology, 27, 1976

[CUENA 1986]
"Inteligencia Artificial: Sistemas Expertos"
J. Cuena, G. Fernández, R. López y M. Verdejo
Alianza Editorial, Madrid, 1986

[DAVIS 1977]
"Production rules as a representation for a knowledge-based consultation program"
R. Davis, B. Buchanan and E. Shortiffe
Artificial Intelligence, Vol. 8 no 1., Feb 1977

[DERMOTT 1980]
""R1: A rule-based configurer of computer systems"
Mc Dermott J.
Computer Science Dept., Carnegie-Mellon University, April 1980

[DOLK 1984]
"Knowledge Representation for Model Management Systems"
Dolk D., Konsynski B.
IEEE Transactions of Software Engineering, Nov 1984

[DUDA 1979]
"Model Design in the Prospector Consultant System for Mineral Exploration"
R. Duda, J. Gaschnig, P. Hart
Expert System for the Microelectronic Age
D. Michie Edinburgh University Press, 1979

[DYWER 1974]
"Heuristic strategies for using computers to enrich education"
Dywer, T. A.
International Journal of Man-Machine Studies, 6, 1974

[EMDEN 1976]
"The Semantics of Predicate Logic as a Programming Language"
M.H. Emden and R.A. Kowalski
ACM 23, no. 4., 1976

[ERMAN 1980]
"The HEARSEY-II speech-undestanding systems integrating knowledge to solve uncertanty"
Erman L.D., Hayes-Roth., Lesser V.R. y Reddy D.R.
ACM, Computing Surveys 12, 1980

[FARRENY 1986]
"Les Systemes Experts principes et éxemples"
H. Farreny
Chadues Editions, Novembre 1986

[FEIGENBAUM 1981]
"The Handbook of Artificial Intelligence"
Feigenbaum, E.A. y Barr, A.
Los Altos Cal, Vol 1, 1981

[FEIGENBAUM 1982]
"Knowledge Engineering for the 1980's"
Feigenbaum, E.A.
Computer Science Dept., Stanford University, 1982

[FINDLER 1979]
"Associative Networks: Representation and use of Knowledge by Computers"
Findler, N.V.
Academic Press, New York, 1979

[FINGERMANN 1976]
"Filosofia"
Gregorio Fingermann
Editorial "El Ateneo", Buenos Aires, 1976

[FISHER 1970]
"Computer structures for programming languages"
D.A. Fisher
Computer Science Dept., PhD thesis, Pittsburg Carnegie Mellon Univ., 1970

[FLETCHER 1984]
"Intelligent instructional systems in training"
Fletcher, J. D.
Applications in artificial intelligence, Princenton, 1984

[FLORES 1985]
"Incertidumbre y Sistemas Expertos"
Flores R. J.
Instituto Politécnico Nacional, 1985

[FREY 1986]
"A bit-mapped classifier"
Peter W. Frey
Byte, Nov 1986

[GOLDSTEIN 1977]
"The computer as coach: An athletic paradigm for intellectual education"
Goldstein, I.P., & Carr, B.
Proceedings of the Annual Meeting of the Association for Computer Machinery, Seattle, Washington, 1977

[GOLDSTEIN 1979]
"Using frames in scheduling"
I.P. Goldstein and B. Roberts
Artificial Intelligence: An MIT Prespective, Vol. 1, 1979

[GOLDSTEIN 1982]
"The genetic graph: A representation of the evolution of procedural knowledge"
Goldstein, I.P.
Intelligent Tutoring Systems, Sleeman 1982

[GUENTHNER 1986]
"A theory for the representation of knowledge"
Franz Guenthner, Hubert Lehmann, Wolfgang Schonfeld
IBM J RES DEVELOP Vol. 30 no. 1, Jan 1986

[GUZMAN 1984]
"Sistemas Expertos y sus aplicaciones"
Guzmán A. A.
Instituto Politécnico Nacional, 1984

[HARMON 1985]
"Expert Systems: Artificial Intelligence in business"
Harmon P., & King, D.
John Wiley & Sons., New York, 1985

[HAYES-ROTH 1983]
"Building Expert Systems"
Hayes-Roth, Waterman, Lenat
Addison-Wesley Publishing Co., 1983

[HAYES-ROTH 1984]
"The knowledge-based expert system: A tutorial"
Frederick Hayes-Roth
IEEE, September 1984

[HOLLAND 1986]
"Escaping Brittleness: The Possibilities of General Purpouse Learning Algortihms Applied to Parallel Rule-Based Systems"
Holland, J
Machine Learning II, Los Altos Ca., Morgan Kaufmann Publ., 1986

[KAY 1977]
"Personal dynamic media"
Kay, A. & Goldberg, A.
IEEE Computer. March 1977

[KEARSLEY 1987]
"Artificial Intelligence & Instruction. Applications and Methods"
Greg P. Kearsley
Addison Wesley, Jun 1987

[KIMBALL 1973]
"Self-optimizing computer-assisted tutoring: Theory and practice"
Kimball, R.
Institute of Mathematical Studies in the Social Science, Psychology and Education Series, Technical Report No. 206, Stanford, CA, 1973

[KINNUCAN 1984]
"Computers that think like experts"
Paul Kinnucan
High Technology Business, Jan 1984

[KLEER 1980]
"Propagation of constraints applied to circuit synthesis"
Kleer, de J y Sussman, G.J.
Circuit Theory Appl., 8, 1980

[KNUTH 1973]
"The Art of Computer Programming. Vol.3 Sorting and Searching"
Donald E. Knuth
Addison-Wesley Publishing Co., 1973

[KOSLOV 1988]
"Rethinking Artificial Intelligence"
Alex Koslov
High Technology Business, May 1988

[KOWALSKI 1974]
"Predicate logic as a programming language"
Kowalski, R.A.
North Holland Publisching Co., Amsterdam, 1974

[LENAT 1976]
"An artificial intelligence approach to discovery in mathematics as heuristic search"
Lenat D.B.
SAIL AIM-286 A.I. Lab. Stanford University, 1976

[LENAT 1983]
"EURISKO: A program that learns new heuristics and domain concepts"
Lenat D.B.
Artificial Intelligence 21, 1983

[LINDSAY 1983]
"Introducción a la Psicología Cognitiva"
Peter H. Lindsay y Donald A. Norman
Editorial Tecnos S.A., Madrid, 1983

[MACHADO 1985]
"ESPIA: Entendimiento Simbólico Por medio de la Inteligencia Artificial"
Machado A.,García H. y Barrera E.
Tesis de Licenciatura, Universidad de la Americas, Puebla 1985

[MAHER 1984]
"Tool and techniques for knowledge based expert systems for engineering design"
M.L. Maher, D. Sriram y S. J. Fenues
Adv. Eng. Software, Vol. 6 No. 4, 1984

[MARCELLIN 1975]
"Un Nuevo Enfoque a la Enseñanza por Computadora"
Sergio Marcellín Jacques
Facultad de Ciencias, UNAM, 1975

[MARIK 1987]
"Introduccion a las técnicas de Sistemas Expertos"
Vladimir Marik
Primer Curso Internacional de Sistemas Expertos, Nov 1987

[MARTINEZ 1986]
"El Aprendizaje en un Sistema Experto"
Dra. Ana Ma. Martínez Enríquez
Ponencia presentada en MEXICON 86, 26-30 Oct., 1986

[MARTINEZ 1987]
"Representación de Conocimientos y su Mecanismo de Inferencia"
Dra. Ana Ma. Martínez Enríquez
Primer Curso Internacional de Sistemas Expertos, Nov 1987

[MERRIL 1980]
"TICCIT"
Merril, M. D. & Schneider, E. W. & Fletcher, K. A.
Englewoods Cliffs, NJ, Educational Technology, 1980

[MINSKY 1975]
"A framework for representing knowledge"
M. Minsky
The Psychology of Computer Vision, McGraw-Hill, 1975

[NAWROCKI 1980]
"Computer-based maintenance training in the military"
Nawrocki, L. H.
Human detection and diagnosis of system failures, New York, 1980

[NAYLOR 1983]
"Build your own expert system"
Naylor, C.
Sigma Technical Press, Cheshire, U.K., 1983

[NILSSON 1971]
"Problem solving methods in Artificial Intelligence"
N.J. Nilsson
McGraw-Hill, 1971

[NORMAN 1971]
"A System for Perception and Memory"
Norman, D.A.
Models of Human Memory, Academic Press, New York, 1971

[NORTON 1985]
"The Peter Norton Programmer's Guide to the IBM PC"
Peter Norton
Microsoft Press, 1985

[O'SHEA 1979]
"A self-improving quadratic tutor"
O'Shea, T.
International Journal of Machine Studies, 11, 1979

[PAPERT 1980]
"Mindstorms"
Papert, S.
New York, Basic Books, 1980

[PARK 1980]
"Adaptive design strategies for selecting number and presentation order of examples in coordinate concept acquisition"
Park, O. & Tennyson, R.D.
Journal of Educational Psychology, 72, 1980

[PARK 1983]
"Computer-based instructional systems for adaptive education: A review"
Park, O. & Tennyson, R.D.
Contemporary Education Review, 2, 1983

[PINSON 1981]
"Représentation des Connaissances dans les Systémes Experts"
Suzanne Pinson
R.A.I.R.O Informatique/Computer Science, Vol. 15 no 4., 1981

[POST 1943]
"Formal reductions of the general combinatorial decision problem"
Post, E.
American Journal of Mathematics 65, 1943

[QUINLAN 1983]
"Learning Efficient Clasification Procedures and Their Application to Chess end Games"
Quinlan, J. Ross
In Machine Learning An Artificial Intelligence Approach, Tioga Publishing co., 1983

[QUILLIAN 1968]
"Semantic Memory"
Quillian M.R.
Semantic Information Processing, Cambridge Ma., 1968

[RAPHAEL 1968]
"SIR: A computer program for semantic information"
Raphael, B.
Semantic Information Processing, Cambridge Ma., 1968

[RIESBECK 1975]
"Conceptual Analysis"
C. Riesbeck
Conceptual Information Processing, North Holland Publishing, Amsterdam, 1975

[ROBERTS 1983]
"Intelligent computer-assisted instruction: And explanation and overview"
Roberts, F.C., & Park, O.
Educational Technology, 23, 1983

[SACERDOTI 1977]
"A Structure for Plans and Behavior"
Sacerdoti, E.D.
Elsevier, Nuew York, 1977

[SALVAT 1971]
Enciclopedia Salvat
Salvat Editores, S.A, Barcelona, 1971

[SCHANK 1977]
"Scripts, Plans, Goals and Understanding"
Schank R.C. y Abelson R.P.
Laurence Erlbaum Assoc. Hillsdale, New Jersey, 1977

[SCHANK 1981]
"Inside Computer Understanding"
Schank R.C. y Riesbeck C.K.
Laurence Erlbaum Assoc. Hillsdale, New Jersey, 1981

[SCHEFFLER 1979]
"Las Condiciones del Conocimiento. Una introducción a la Epistemología y a la Educación"
Israel Scheffler
Instituto de Investigaciones Filosóficas, UNAM No.29, 1979

[SCHRODT 1986]
"Predicting International Events"
Philip A. Schrotd
Byte, Nov 1986

[SCHUBERT 1977]
"Extending the expressive power of Semantic Networks"
Schubert, L.
Artificial Intelligence, Vol. 7 no. 2, 1976

[SEIDEL 1971]
"Current status of computer-administered instruction work under project IMPACT"
Seidel, R.J
Educational Technology, Vol. 11, 1971

[SHORTLIFFE 1976]
"Computer Based Medical Consultations MYCIN"
E. Shortliffe
Elsevier Computer Sc. Library N.J USA, 1976

[SLEEMAN 1982]
"Intelligent tutoring systems"
Sleeman, D., & Brown, J.S.
Academic Press, New York, 1982

[SOLOWAY 1983]
"PROUST: Knowledge-based program debugging"
Elliot Soloway & W. Lewis Johnson
Proceedings Eighth International Software Engineering Conference, Orlando Fa., 1983

[STALLMAN 1977]
"Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis"
Stallman, R.M. y Sussman, G.J.
Artificial Intelligence 9, 1977

[STANDISH 1980]
"Data Structures Techniques"
Thomas A. Standish
Addison Wesley, 1980

[STEFIK 1978]
"Inferring DNA Structures from segmentation data"
Stefik, M.J.
Artificial Intelligence 11, 1978

[STELZER 1972]
"Project IMPACT software documentation: Overview of the computer-administrated instruction subsystem"
Stelzer, J., & Garneau, J.
Technical Report 72-21, Alexandria VA: Human Resources Research Organization, 1972

[STEVENS 1977]
"The goal structure of a Socratic tutor"
Stevens, A. L., & Collins, A.
Proceedings of the Annual Meeting of Man-Machine Studies, Seattle, Washington, 1977

[SUPPES 1968]
"Computer-Assisted Intruction: The 1965-66 Stanford Aritmetic Program"
Suppes, P., & Jerman, M., & Brian, D.
Academic Press, New York, 1968

[SUWA 1984]
"Completness and Consistency in a Rule-Based System"
M. Suwa, A.C. Scott and E.H. Shortiffe
Rule-Based Expert Systems, Addison-Wesley Publishing Co., 1984

[THOMPSON 1986]
"Finding rules in data"
Beverly Thompson and William Thompson
Byte, Nov 1986

[TROTTER 1986]
"The Mystery of Mastery"
Robert J. Trotter
Psychology Today, July 1986

[WALKER 1986]
"Knowledge systems: Principles and practice"
Adrian Walker
IBM J RES DEVELOP Vol. 30 no.1, Jan 1986

[WALLACH 1987]
"Development Strategies for ICAI on Small Computers"
Bret Wallace
Artificial Intelligence & Instruction, Kearsley 1987

[WENGER 1987]
"Artificial Intelligence and Tutoring Systems"
Etienne Wenger
Morgan Kaufmann Publishing Inc., 1987

[WINOGRAD 1975]
"Frame Representation and Declarative-Procedural Controversy"
Winograd T.
Representation and Understanding, Academic Press, New York, 1975

[WINSTON 1984]
"Artificial Intelligence"
Patrick Henry Winston
Addison-Wesley Publishing Company Inc., 1984

[WIRTH 1976]
"Algorithms+Data Structures=Programs"
Niklaus Wirth
Prentice Hall Inc., 1976

ANEXO 1

DEFINICIONES DE SISTEMA EXPERTO

[PINSON 1981]

"Se puede definir un sistema experto en forma funcional como un sistema que permite la resolución de problemas en un dominio específico (medicina, geología, química, etc...) utilizando una base de conocimientos adquirida de los expertos del dominio y un mecanismo de razonamiento característico de los expertos".

[FEINGENBAUM 1982]

"Un sistema experto es un programa de computadora inteligente que utiliza conocimientos y procedimientos de inferencia para resolver problemas que son bastante difíciles, requiriendo de una experiencia humana significativa para su solución".

[NAYLOR 1983]

"Se considera que un sistema experto es la incorporación en una computadora de un componente basado en el conocimiento que se obtiene a partir de la habilidad de un experto, de forma tal que el sistema pueda dar consejos inteligentes o tomar decisiones inteligentes.... Una característica adicional deseable, y que para muchos es fundamental, es que el sistema sea capaz, bajo demanda, de justificar su propia línea de razonamiento de una forma inmediatamente inteligible para el que lo usa".

[GUZMAN 1984]

"Un sistema experto es un programa que exhibe conocimientos profundos en un área limitada del saber".

[CHARNIAK 1985]

"Un sistema experto es un programa de aplicación de inteligencia artificial basado en reglas para desarrollar una tarea que requiere expertismo".

[FLORES 1985]

"Un sistema experto es un programa de computadora 'Inteligente', capaz de resolver problemas reales tan complejos que requieran la intervención de una persona calificada como experto en el área".

[CUENA 1986]

"Un sistema experto es un sistema informático que incorpora, en forma operativa, el conocimiento de una persona experimentada, de forma que es capaz de responder como esta persona, como de explicar y justificar sus respuestas".

[FARRENY 1986]

Los sistemas expertos son programas destinados a remplazar o a asistir al hombre en campos donde se requiere de una expertez humana, cuyas características son:

- Ser insuficientemente estructurada para constituir un método de trabajo.
- Estar sujeta a revisiones o complementaciones (según la experiencia acumulada).

[MARIK 1987]

"Los sistemas expertos son programas sofisticados de computación que manipulan conocimientos de expertos para resolver eficiente y efectivamente problemas de un área especifica".

ANEXO 2

DIAGRAMAS DE SISTEMAS EXPERTOS

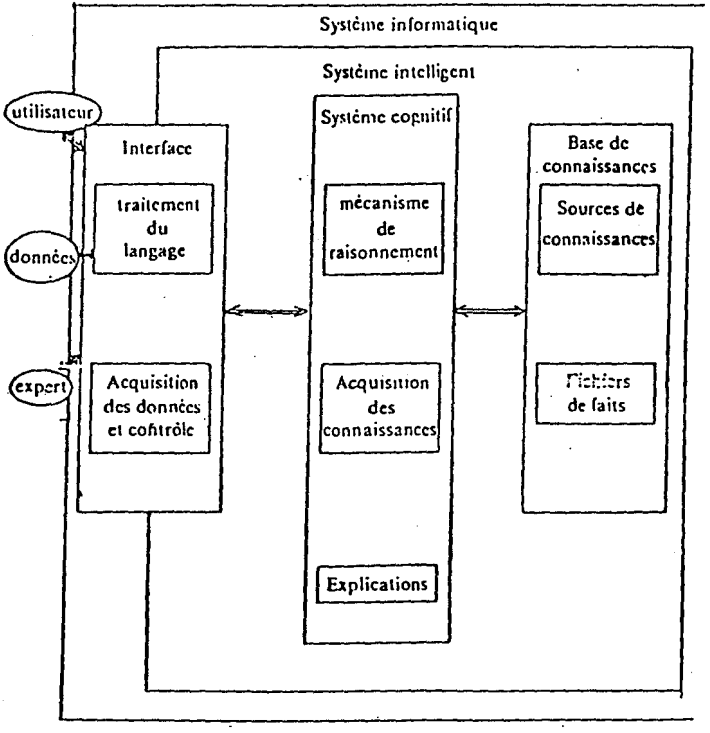


Figura A2.1 "Structure d'un système expert" [PINSON 1981].

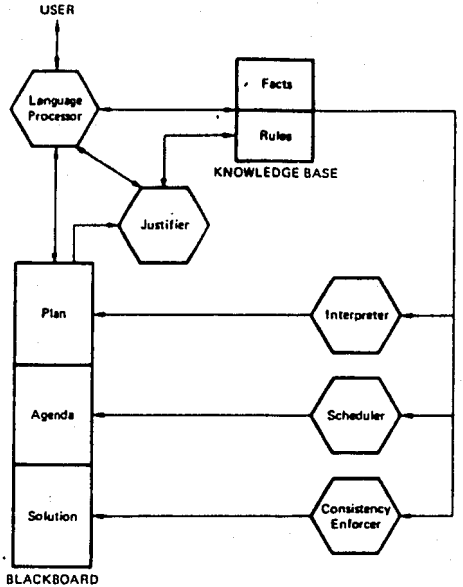
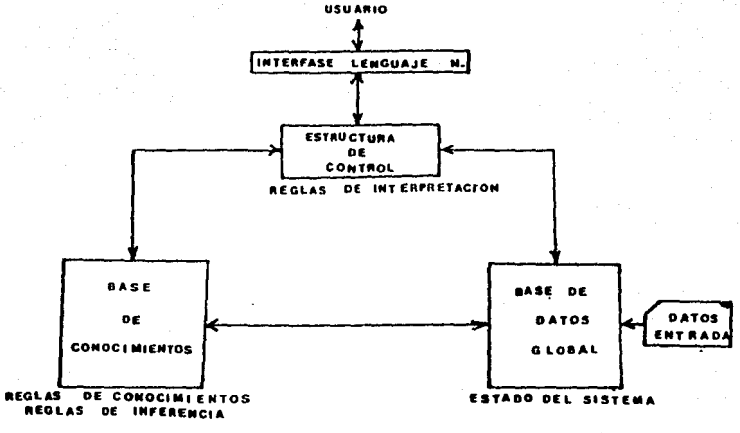


Figura A2.2 "Anatomy of an ideal expert system" [HAYES-ROTH 1983].



ESTRUCTURA BASICA DE UN SISTEMA EXPERTO

Figura A2.3 "Estructura básica de un sistema experto" [CHAPA 1984].

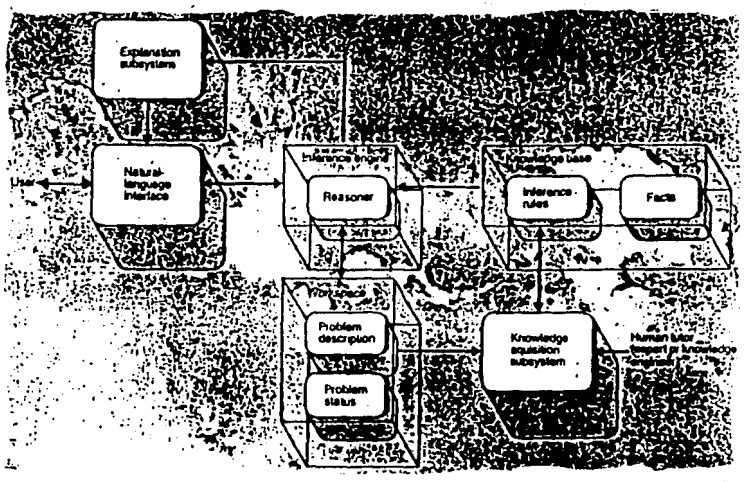


Figura A2.4 "Expert system of today" [KINNUCAN 1984].

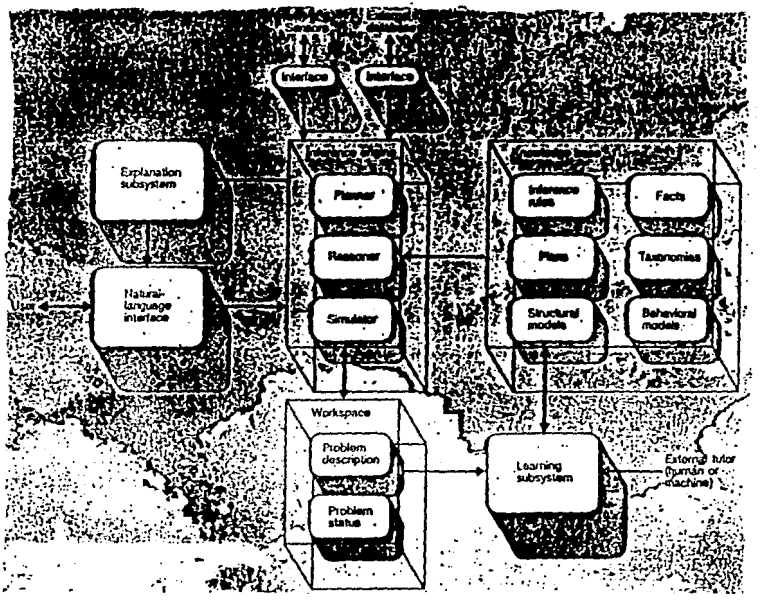


Figura A2.5 "Expert system of the future" [KINNUCAN 1984].

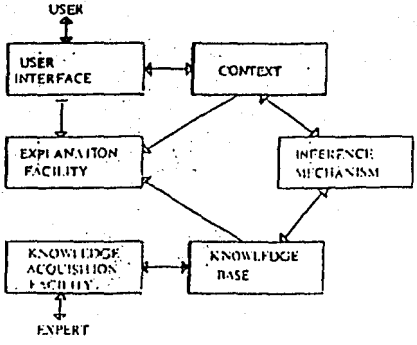


Figura A2.6 "A schematic view of a complete Knowledge-Based Expert System" [MAHER 1984].

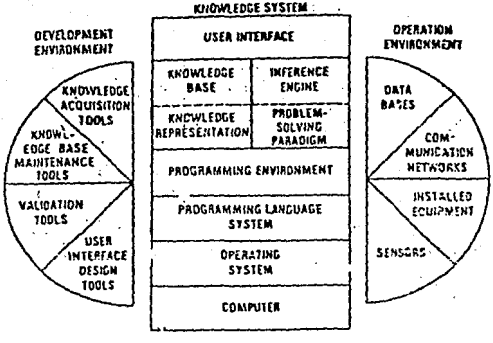


Figura A2.7 "Major components of a contemporary knowledge system" [HAYES-ROTH 1984].

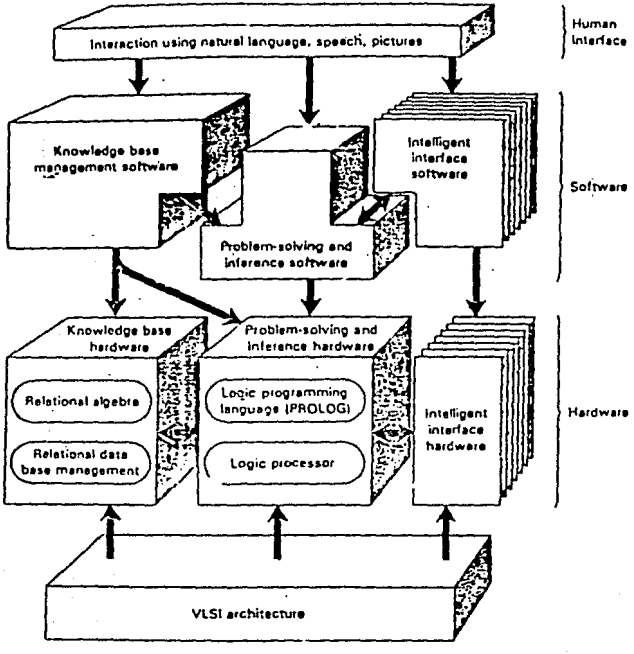


Figura A2.8 "Fifth-Generation Computer Systems" [FEINGENBAUM 1984].

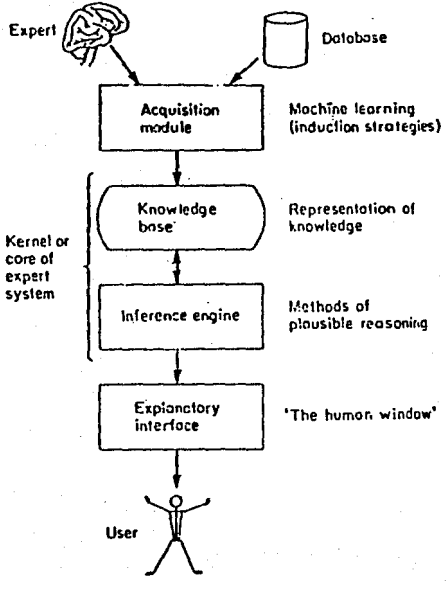


Figura A2.9 "A typical expert system" [FORSYTH 1984].

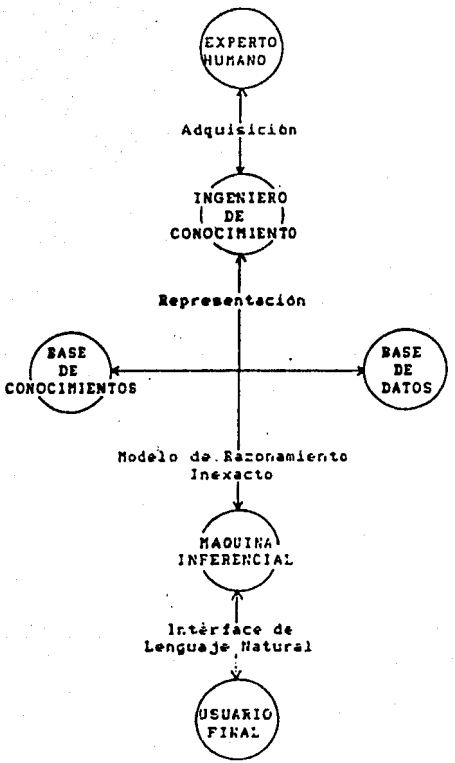


Figura A2.10 Componentes de un sistema experto [FLORES 1985].

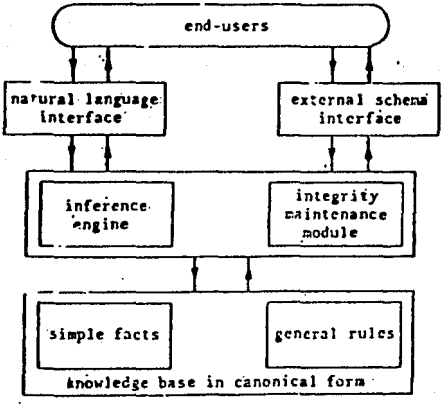


Figura A2.11 "A simple architecture for an unsophisticated knowledge base system" [COLLINS 1985].

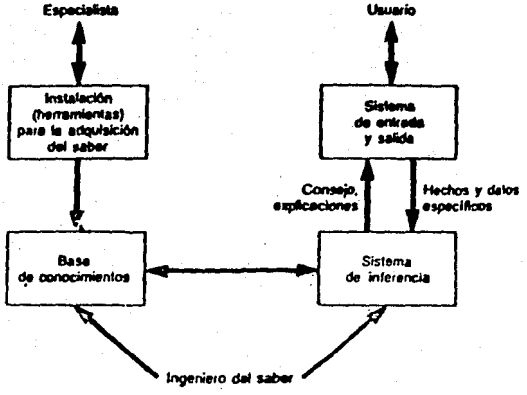


Figura A2.12 Estructura básica de un sistema especializado [FEINGENBAUM 1986].

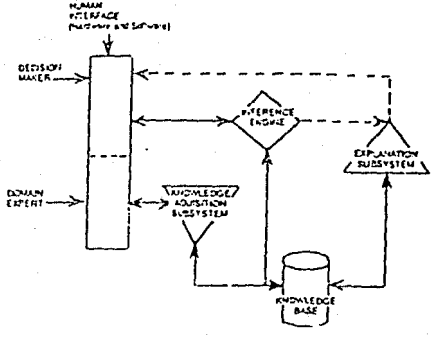


Figura A2.13 'Components of an Expert System' [KEIM 1986].

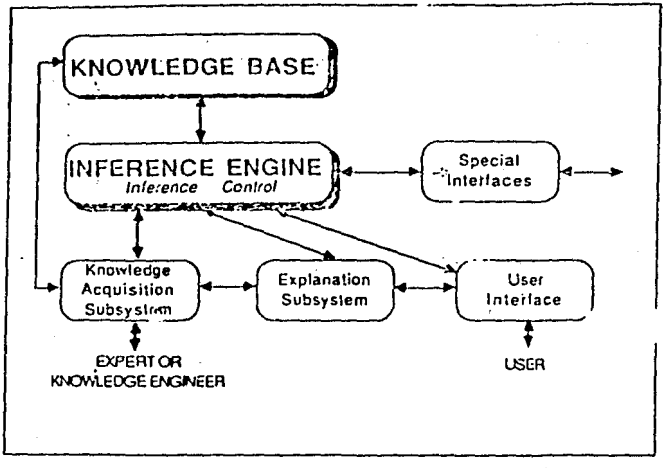


Figura A2.14 'The architecture of an expert system' [HARMON 1986].

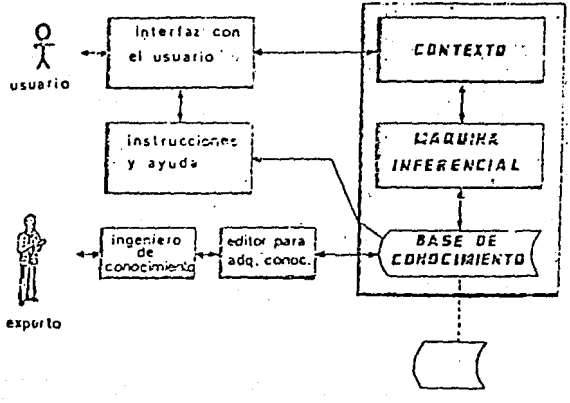


Figura A2.15 Componentes de un sistema experto [SALDAÑA 1986].

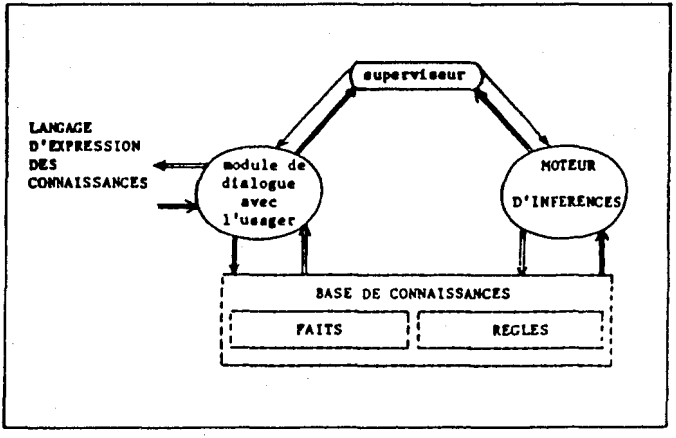


Figura A2.16 "Organisation de principe d' un systéme-expert" [FARRENY 1986].

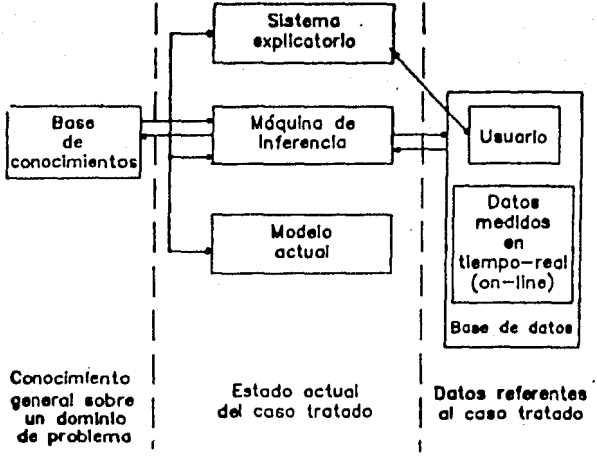


Figura A2.17 Sistema experto de diagnóstico [MARIK 1987].

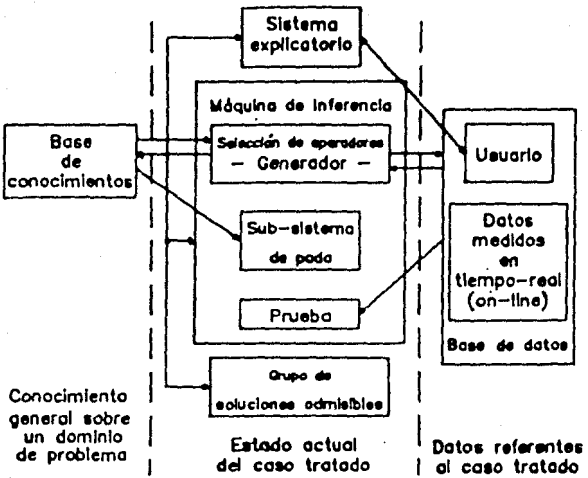


Figura A2.18 Sistema experto de planeación [MARIK 1987].

ANEXO 3

DIAGRAMAS DE SISTEMAS ICAI

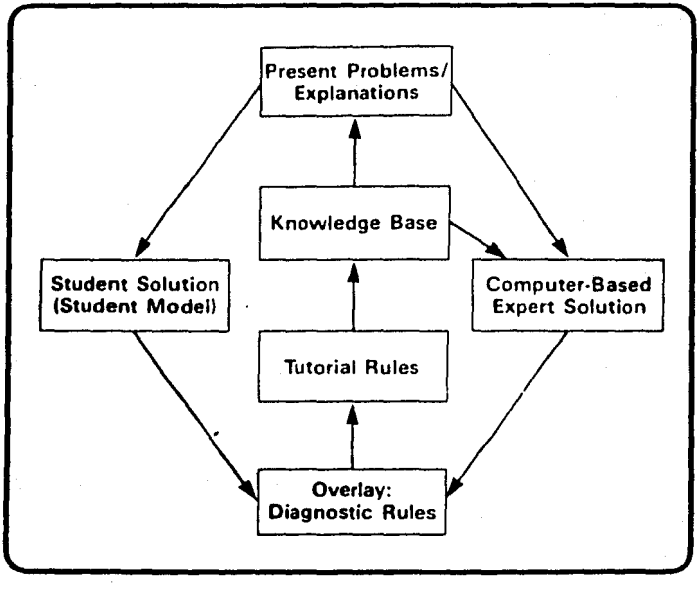


Figura A3.1 **"A General Model of ICAI" [PARK 1980].**

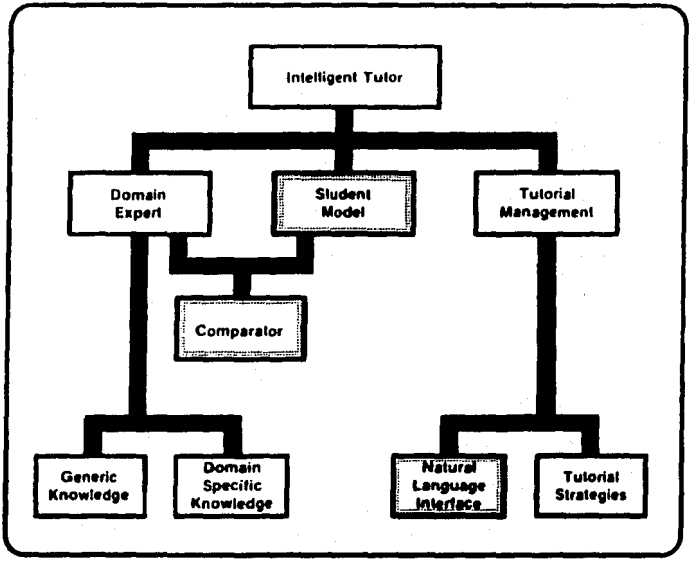


Figura A3.2 **"Components of an ICAI System" [HAWROCKI 1980].**

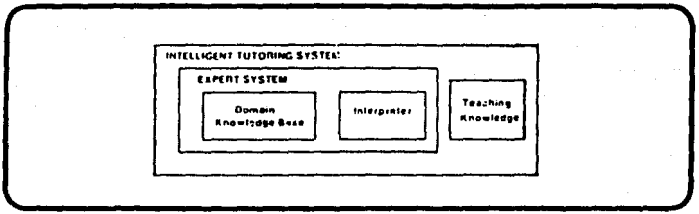


Figura A3.3 **"Components of an Intelligent Tutoring System" [CLANCEY 1981].**

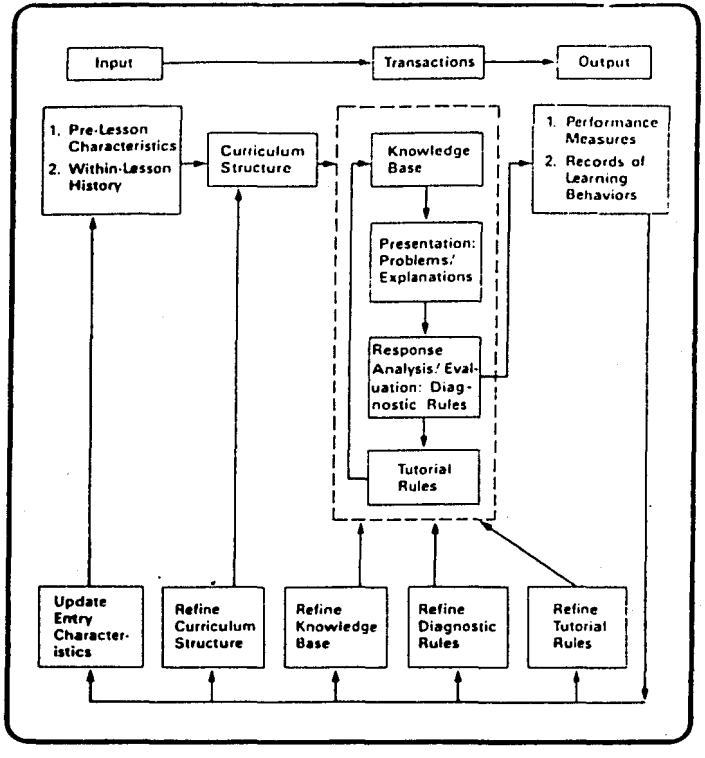


Figura A3.4 "A model of Adaptive Instruction" [PARK 1983].

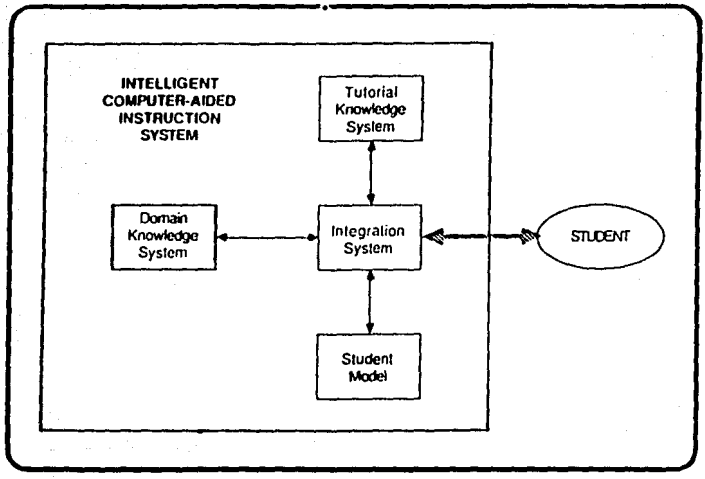


Figura A3.5 **"Architecture of an ICAI System" [HARMON 1985].**

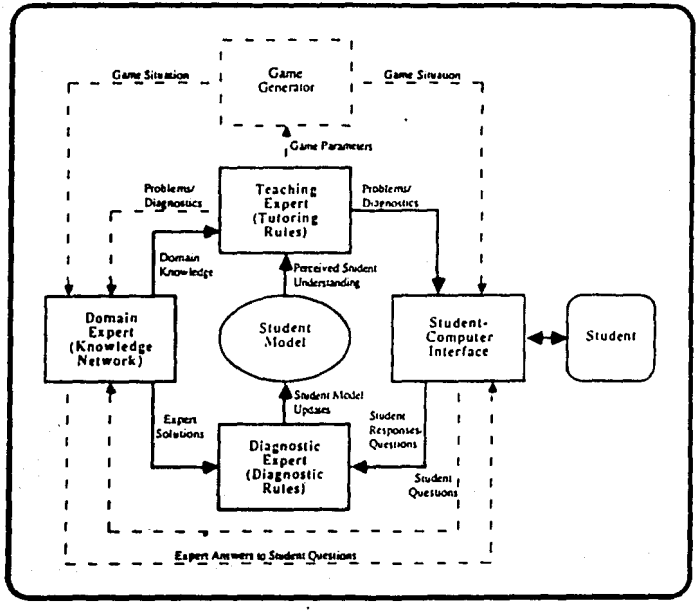


Figura A3.6 "Major Components of Typical ICAI Programs" [WALLACH 1987].

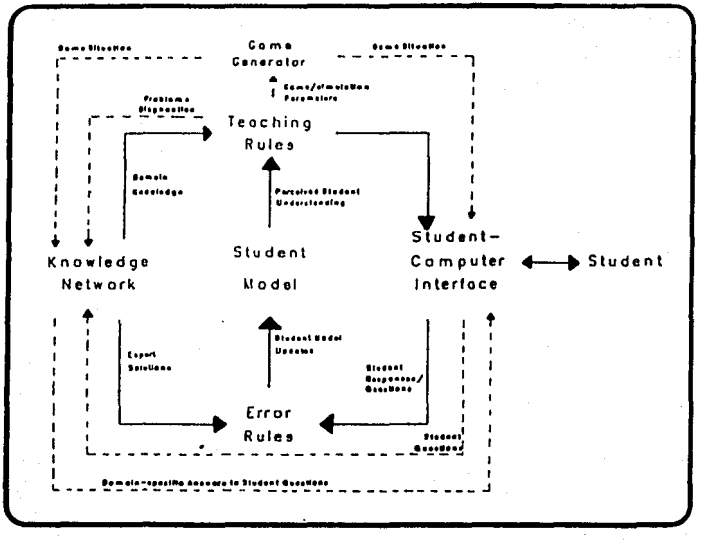
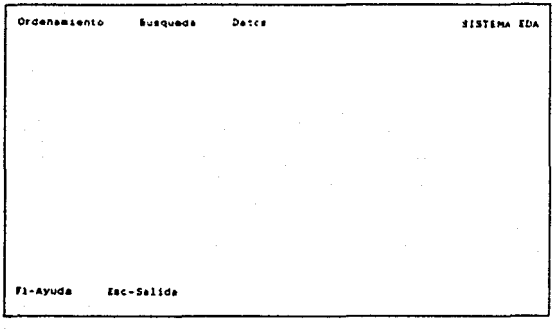


Figura A3.7 "Model of ICAI Environment" [BEGG 1987].

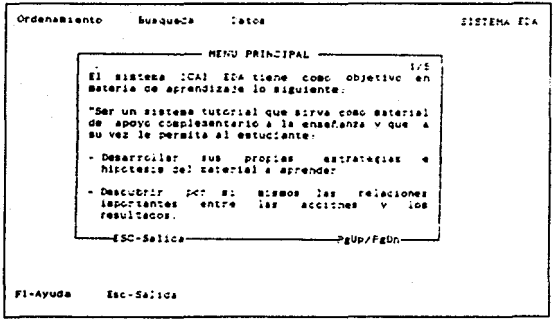
ANEXO 4

INTERFAZ HOMBRE-MAQUINA DEL SISTEMA EDA

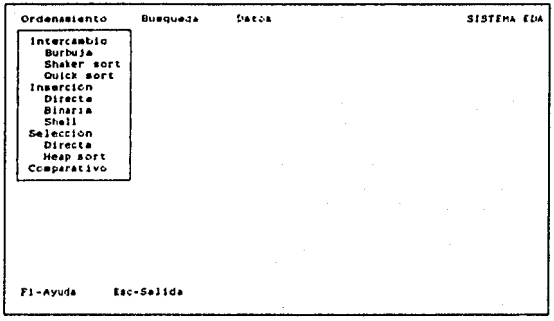
Menú principal (Módulos: Ordenamiento, búsqueda y datos).



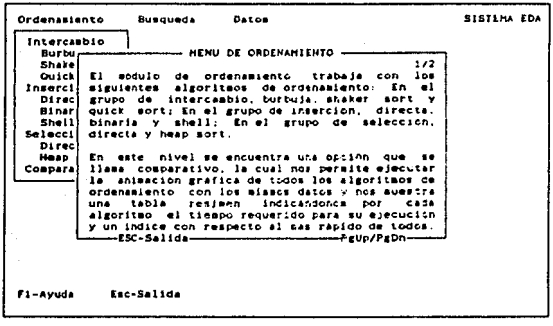
Ayuda menú principal.



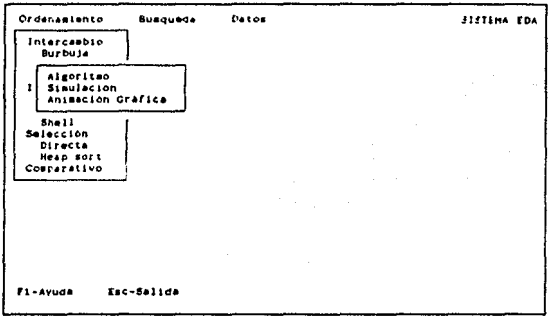
Menú de ordenamiento.



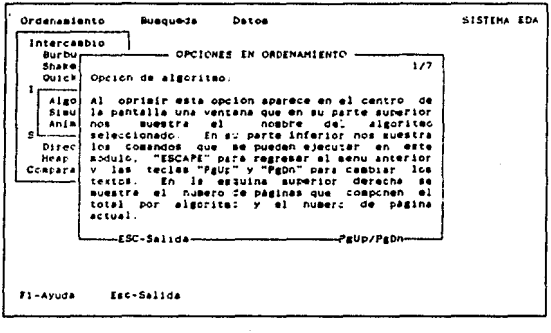
Ayuda menú de ordenamiento.



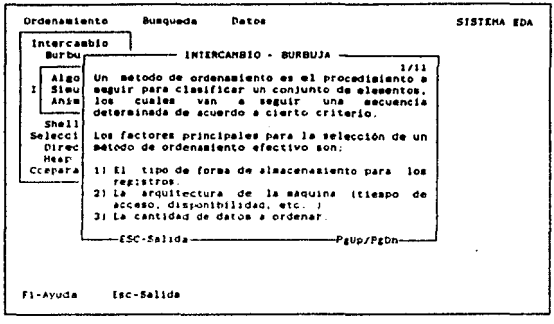
Menú de opciones en ordenamiento.



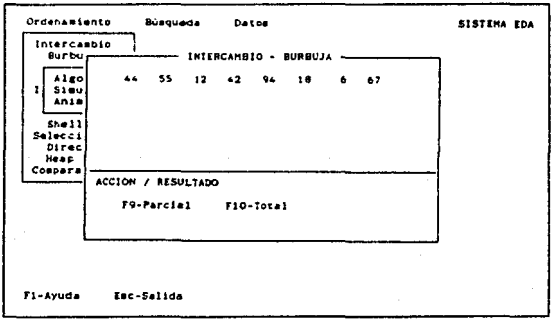
Ayuda menú de opciones en ordenamiento.



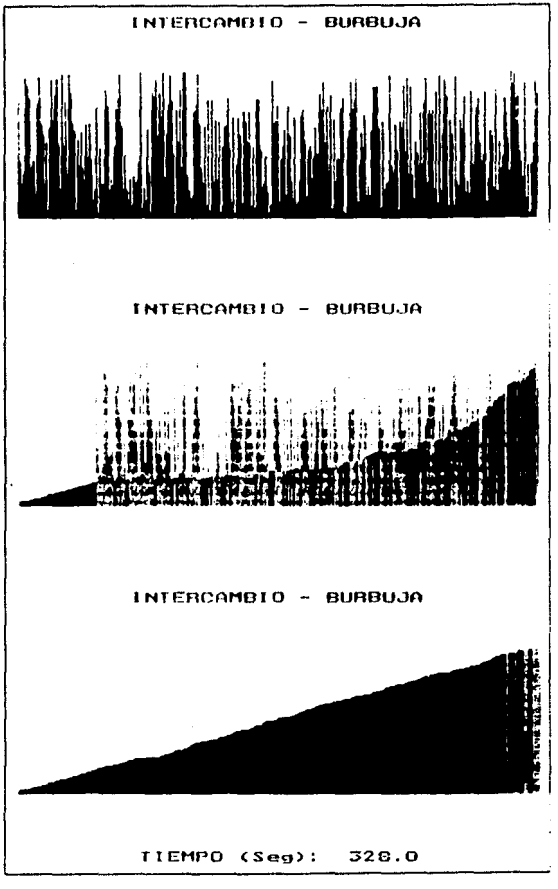
Opción de algoritmo (Módulo de ordenamiento).

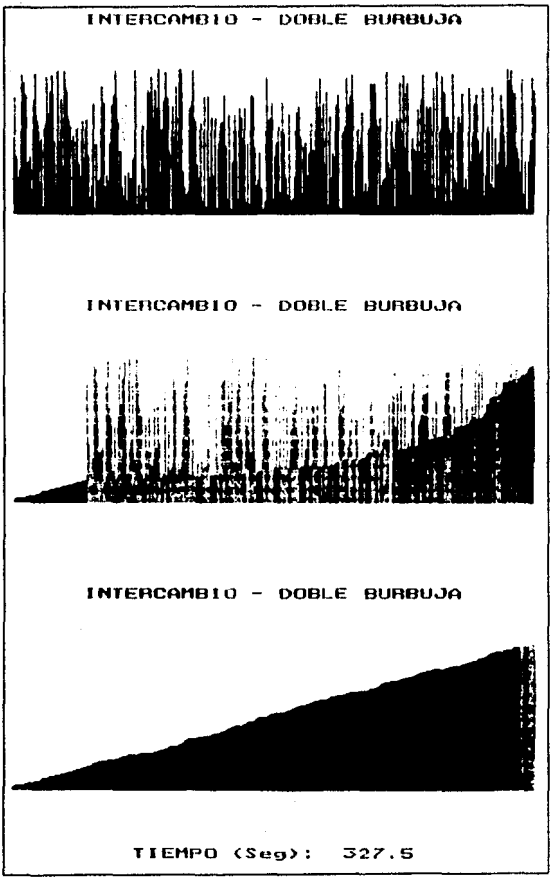


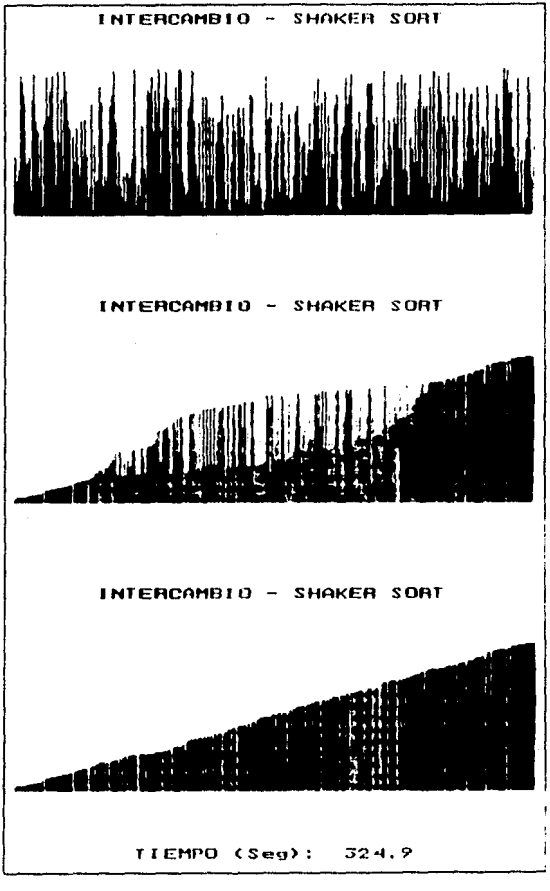
Opción de simulación (Módulo de ordenamiento).

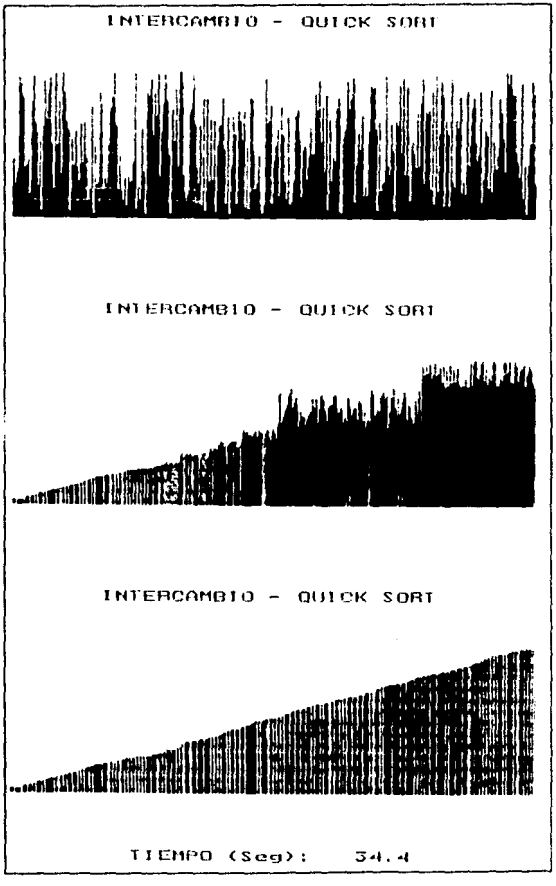


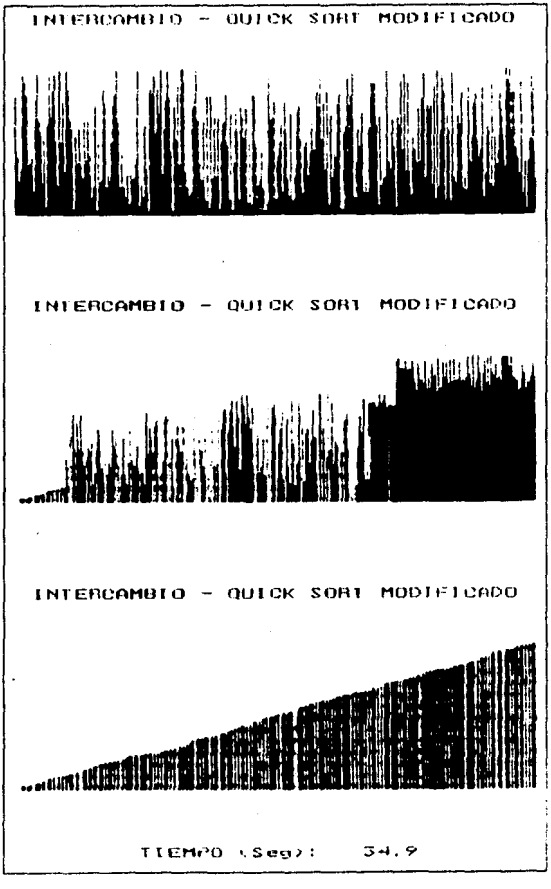
Opción de animación gráfica (Módulo de ordenamiento).

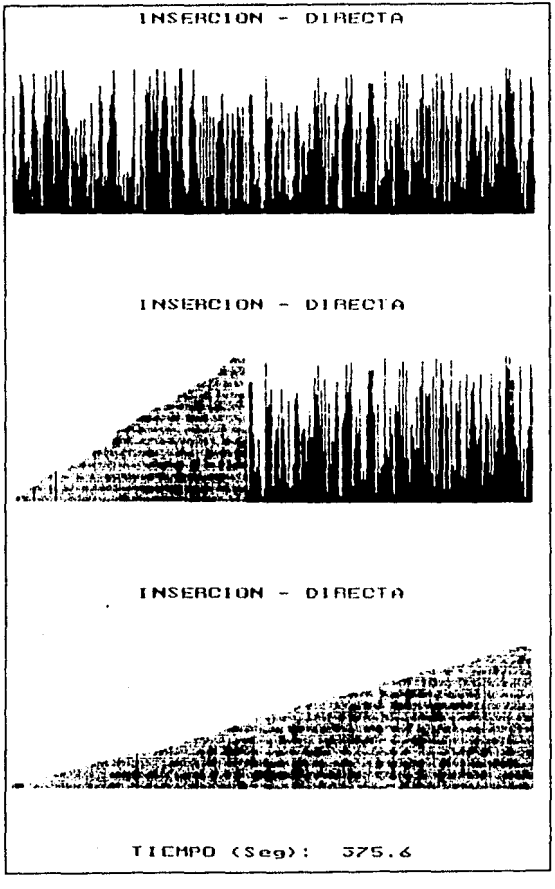


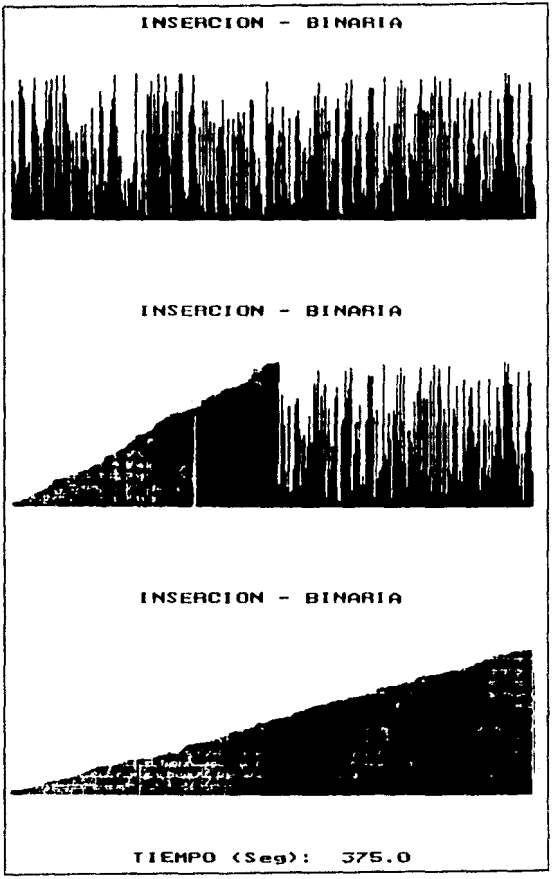

INTERCAMBIO - DOBLE BURBUJA
INTERCAMBIO - DOBLE BURBUJA
INTERCAMBIO - DOBLE BURBUJA
TIEMPO (Seg): 327.5

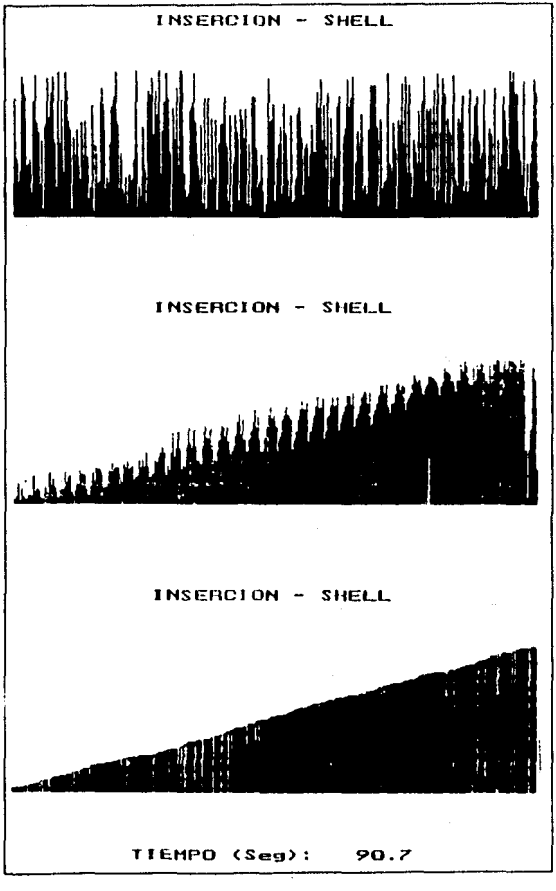

INTERCAMBIO - SHAKER SORT
INTERCAMBIO - SHAKER SORT
INTERCAMBIO - SHAKER SORT
TIEMPO (Seg): 324.9

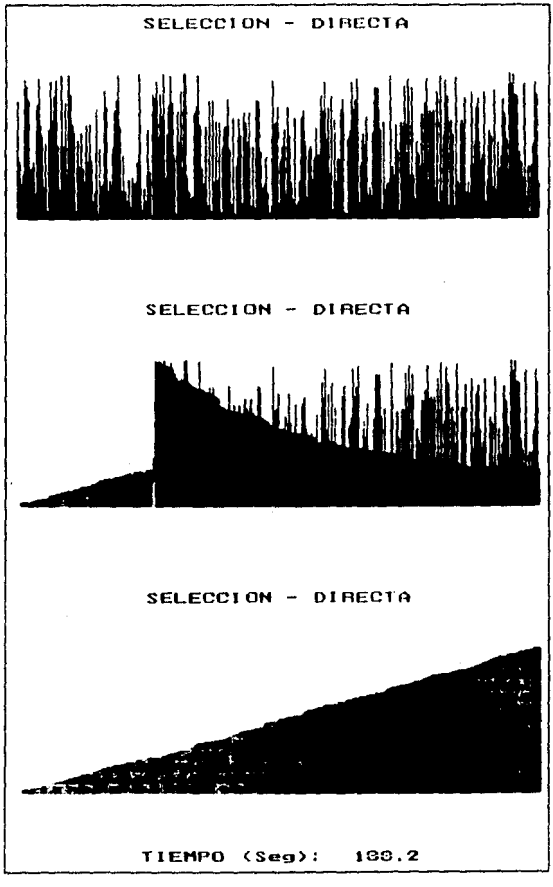

INTERCAMBIO - QUICK SORT
INTERCAMBIO - QUICK SORT
INTERCAMBIO - QUICK SORT
TIEMPO (Seg): 34.4

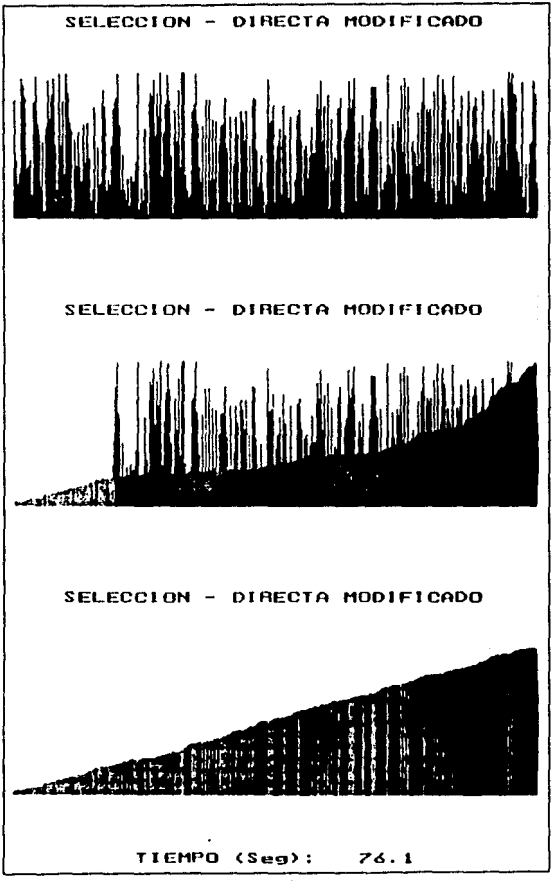

INTERCAMBIO - QUICK SORT MODIFICADO
INTERCAMBIO - QUICK SORT MODIFICADO
INTERCAMBIO - QUICK SORT MODIFICADO
TIEMPO (Seg): 34.9

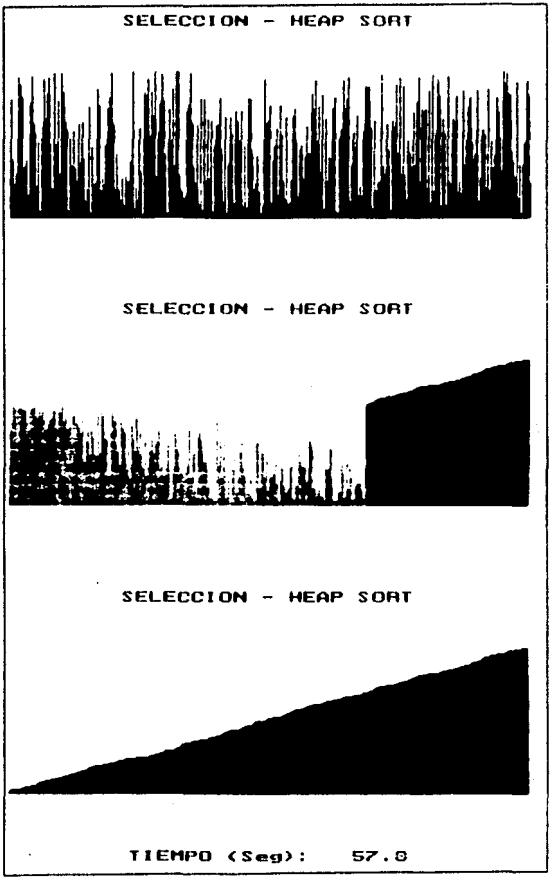

INSERCION - DIRECTA
INSERCION - DIRECTA
INSERCION - DIRECTA
TIEMPO (Seg): 375.6


INSERCION - BINARIA
INSERCION - BINARIA
INSERCION - BINARIA
TIEMPO (Seg): 375.0


INSERCION - SHELL
INSERCION - SHELL
INSERCION - SHELL
TIEMPO (Seg): 90.7


SELECCION - DIRECTA
SELECCION - DIRECTA
SELECCION - DIRECTA
TIEMPO (Seg): 188.2


SELECCION - DIRECTA MODIFICADO
SELECCION - DIRECTA MODIFICADO
SELECCION - DIRECTA MODIFICADO
TIEMPO (Seg): 76.1

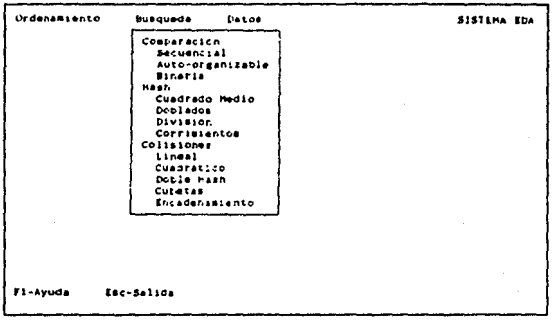

SELECCION - HEAP SORT
SELECCION - HEAP SORT
SELECCION - HEAP SORT
TIEMPO (Seg): 57.8

Cuadro comparativo.

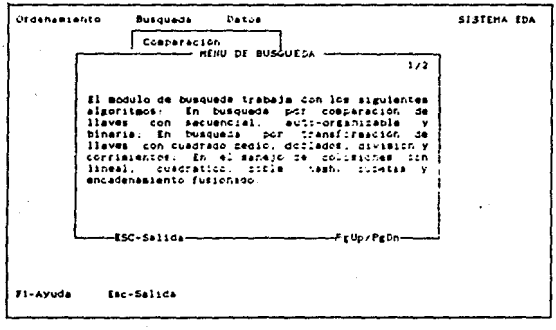
CUADRO RESUMEN

	Segs	Indice
INT - BURBUJA	303.67	30.06
INT - BURBUJA MOD.	303.02	30.00
INT - SHAKER SORT	302.81	29.98
INT - QUICK SORT	10.27	1.02
INT - QUICK SORT MOD.	10.10	1.00
INS - DIRECTA	375.80	37.21
INS - BINARIA	351.52	34.80
INS - SHELL	69.21	6.85
SEL - DIRECTA	142.20	14.04
SEL - DIRECTA MOD.	49.43	4.89
SEL - HEAP SORT	34.14	3.38

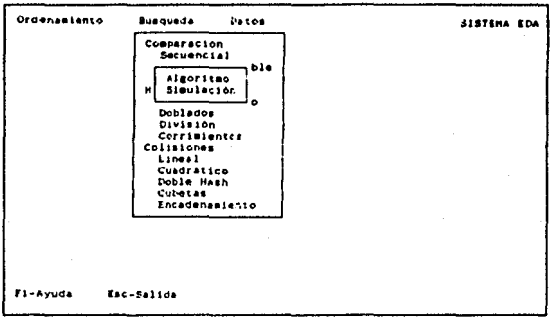
Menú de búsqueda.



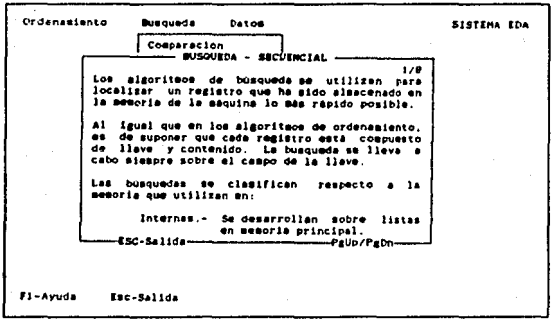
Ayuda menú de búsqueda.



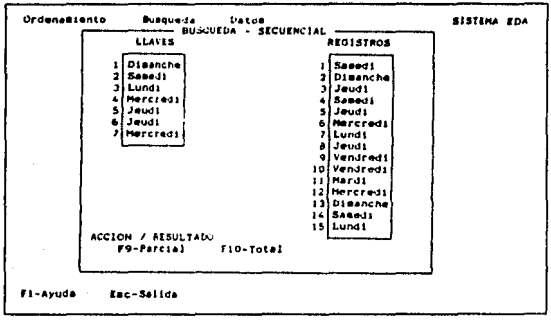
Menú de opciones en búsqueda.



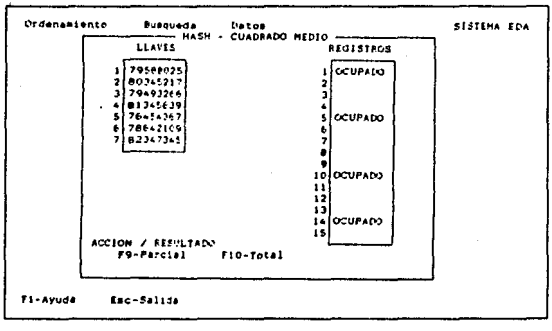
Opción de algoritmo (Módulo de búsqueda).



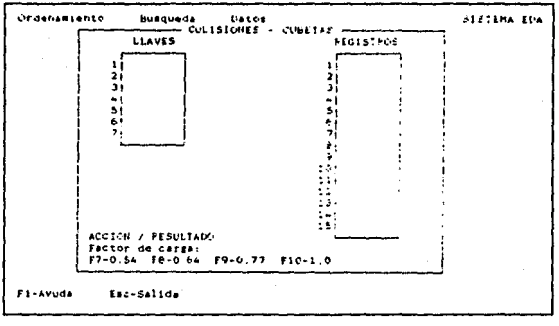
Opción de simulación búsqueda (Módulo de búsqueda).

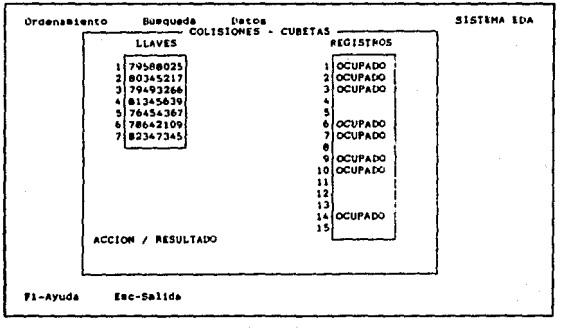


Opción de simulación hash (Módulo de búsqueda).

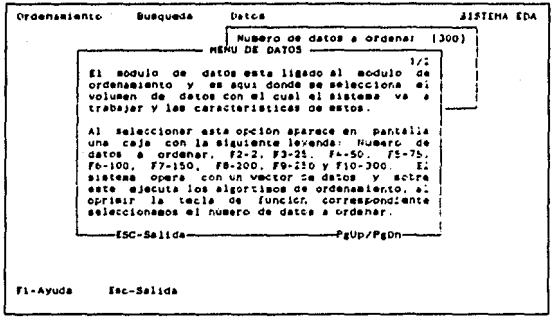


Opción de simulación colisiones (Módulo de búsqueda).

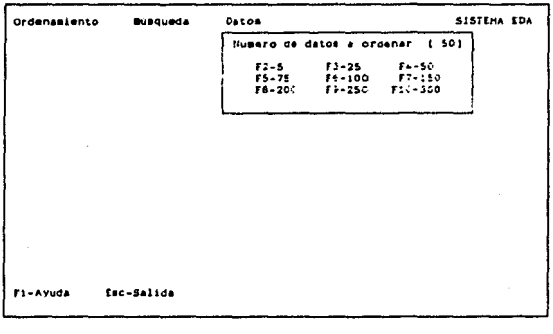


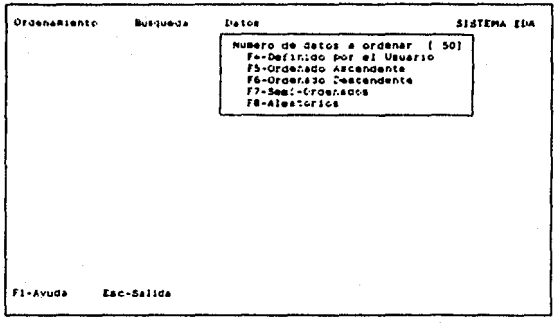


Ayuda módulo de datos.



Módulo de datos.




Ordenamiento
Busqueda
Datos
SISTEMA EDA
Numero de datos a ordenar [50]
F4-Definido por el Usuario
F5-Ordenado Ascendente
F6-Ordenado Descendente
F7-Semi-Ordenados
F8-Aleatorios
F1-Ayuda
Esc-Salida

ANEXO 5

BASE DE CONOCIMIENTOS DEL SISTEMA EDA

1. Módulo de Ordenamiento

BURBUJA

1/11

Un método de ordenamiento es el procedimiento a seguir para clasificar un conjunto de elementos, los cuales van a seguir una secuencia determinada de acuerdo a cierto criterio.

Los factores principales para la selección de un método de ordenamiento efectivo son:

1) El tipo de forma de almacenamiento para los registros.
2) La arquitectura de la máquina (tiempo de acceso, disponibilidad, etc.).
3) La cantidad de datos a ordenar.

BURBUJA

2/11

4) La situación de los datos, es decir, que tan ordenados están.
5) El modo de las llaves (binaria, decimal, alfabética).
6) La distribución de las llaves,

Criterios en los que se basa la eficiencia de un algoritmo:

1) El número de comparaciones de llaves.
2) El número de transferencias de registros.
3) El espacio extra de almacenamiento requerido.

BURBUJA

3/11

Según el tipo de memoria en que se efectúa el ordenamiento, los métodos se clasifican en:

a) Internos.- Cuando todos los datos a ordenar están en memoria principal (arreglos).

b) Externos.- Cuando los datos son tantos que no caben en memoria principal y se guardan en parte de la memoria secundaria (archivos).

Los métodos externos son mucho más lentos que los internos, ya que en estos se tiene que tomar en cuenta la transmisión de datos.

BURBUJA

4/11

Los métodos por intercambio transponen o intercambian sistematicamente pares de datos de tal forma que los elementos que se encuentran fuera de orden dejan de existir.

Ocupan la misma cantidad de memoria que la lista original. Este algoritmo lleva el nombre de burbuja ya que su flujo semeja la acción de una burbuja que emerge en un medio liquido.

El proceso inicia comparando los elementos XMAX y XMAX-1, donde XMAX es el último elemento dentro de un vector X. Si XMAX es menor que XMAX-1 se intercambian. Luego compara XMAX-1 y

BURBUJA

5/11

XMAX-2, si XMAX-1 es menor que XMAX-2 nuevamente los intercambia y así sucesivamente hasta comparar los datos 1 y 2.

Cuando esto ocurre el dato menor ha alcanzado su posición final (suponiendo que queremos ordenar el vector de menor a mayor). Este procedimiento se aplica sobre los datos comprendidos entre 2 y XMAX y como máximo se necesitan XMAX-1 pasadas.

Se considera una pasada cuando se han comparado todos los elementos del vector.

BURBUJA

El algoritmo es el siguiente: 6/11

```
PROCEDURE BURBUJA(X:ARR);
VAR
   I,J,K : INTEGER;
BEGIN
   FOR I:=1 TO XMAX-1 DO
      FOR J:=XMAX DOWNTO 2 DO
         IF X[J-1]>X[J] THEN
            BEGIN
               K:=X[J];
               X[J]:=X[J-1];
               X[J-1]:=K;
            END;
END; (* BURBUJA *)
```

BURBUJA

7/11

Existe también otro algoritmo llamado burbuja modificado el cual evita hacer comparaciones en la parte de la lista que ya fue ordenada.

BURBUJA

El algoritmo es el siguiente: 8/11

```
PROCEDURE BURBUJA_MODIFICADO(X:ARR);
VAR
   I,J,K : INTEGER;
BEGIN
   FOR I:=1 TO XMAX-1 DO
      FOR J:=XMAX DOWNTO 1+I DO
         IF X[J-1]>X[J] THEN
            BEGIN
               K:=X[J];
               X[J]:=X[J-1];
               X[J-1]:=K;
            END;
END; (* BURBUJA_MODIFICADO *)
```

BURBUJA

9/11

En ambos casos el algoritmo es muy ineficiente cuando la lista pasa de 10 ó 15 elementos, sin embargo es muy sencillo de programar.

Oprimiendo la tecla F8 generarás una lista de números en forma aleatoria.

Al oprimir la tecla F10, en el centro de la pantalla podrás observar la siguiente frase: F9-Burbuja y F10-Burbuja Modificado. Oprimiendo la tecla correspondiente podrás observar el algoritmo en pantalla. (Una vez que tengas el arreglo de datos en pantalla, no olvides oprimir la tecla RETURN para comenzar).

BURBUJA

10/11

En el caso de que el arreglo de datos a ordenar (para nuestro ejemplo en forma ascendente) ya se encuentre ordenado, el algoritmo desarrolla todas las comparaciones en las XMAX-1 pasadas pero no lleva a cabo ningún intercambio. Esto es importante dado que la cantidad de proceso requerido para las comparaciones es mínima comparada con la cantidad de proceso requerido cuando se involucran intercambios.

Oprime la tecla F5 para generar un vector de datos ordenado en forma ascendente y posteriormente oprime la tecla F10 para observar el comportamiento de los algoritmos bajo estas

BURBUJA

condiciones. 11/11

El caso mas crítico se obtiene cuando el arreglo de datos a ordenar (en nuestro ejemplo en forma ascendente) se encuentra ordenado a la inversa (en forma descendente). En esta situación el algoritmo lleva a cabo un intercambio en todas las comparaciones de las XMAX-1 pasadas.

Oprime la tecla F6 para generar un vector de datos ordenado en forma descendente y posteriormente oprime la tecla F10 para observar el comportamiento de los algoritmos bajo estas condiciones.

SHAKERSORT 1/6

Este algoritmo es muy similar al de la burbuja excepto que las pasadas se alternan, una de derecha a izquierda para colocar la llave menor con una de izquierda a derecha para colocar la llave mayor, y asi sucesivamente hasta colocar a todos los elementos del arreglo en el orden adecuado.

SHAKERSORT 2/6

El algoritmo es el siguiente:

```
PROCEDURE SHAKERSORT(X:ARR);
VAR
   J,K,L,R,M : INTEGER;
BEGIN
   L:=2; R:=XMAX; K:=XMAX;
   REPEAT
      FOR J:=R DOWNTO L DO
         IF X[J-1]>X[J] THEN
            BEGIN
               M:=X[J];
               X[J]:=X[J-1];
               X[J-1]:=M; K:=J;
            END;
```

SHAKERSORT 3/6

```
      L:=K+1;
      FOR J:=L TO R DO
         IF X[J-1]>X[J] THEN
            BEGIN
               M:=X[J];
               X[J]:=X[J-1];
               X[J-1]:=M;
               K:=J;
            END;
      R:=K-1;
   UNTIL L > R;
END; (* SHAKERSORT *)
```

SHAKERSORT

4/6

El algoritmo es muy ineficiente cuando la lista pasa de 10 ó 15 elementos.

Oprimiendo la tecla F8 generarás una lista de números en forma aleatoria.

Al oprimir la tecla F10 podrás observar el algoritmo en pantalla. (Una vez que tengas el arreglo de datos en pantalla, no olvides oprimir la tecla RETURN para comenzar).

SHAKERSORT

5/6

En el caso de que el arreglo de datos a ordenar (para nuestro ejemplo en forma ascendente) ya se encuentre ordenado, el algoritmo desarrolla todas las comparaciones en la primera pasada sin llevar a cabo ningún intercambio. El algoritmo detecta que no llevó a cabo ningún intercambio y termina. Estas características hacen que este algoritmo sea el más eficiente bajo estas circunstancias.

Oprime la tecla F5 para generar un vector de datos ordenado en forma ascendente y posteriormente oprime la tecla F10 para observar el comportamiento del algoritmo.

SHAKERSORT

6/6

El caso mas crítico se obtiene cuando el arreglo de datos a ordenar (en nuestro ejemplo en forma ascendente) se encuentra ordenado a la inversa (en forma descendente). En esta situación el algoritmo lleva acabo un intercambio en todas las comparaciones de todas las pasadas.

Oprime la tecla F6 para generar un vector de datos ordenado en forma descendente y posteriormente oprime la tecla F10 para observar el comportamiento del algoritmo bajo estas condiciones.

QUICKSORT

1/12

Este algoritmo fue desarrollado por C.A.R Hoare quien lo bautizó como QUICK SORT por ser este el método más rápido y eficiente de ordenamiento para arreglos.

El proceso inicia seleccionando el primer elemento del arreglo. A este elemento se le encuentra su posición final (cuando los elementos colocados a la izquierda de éste sean menores y a la derecha mayores).

Una vez encontrada la posición final de éste elemento, se particiona la lista de tal forma que se obtienen dos listas más pequeñas, una

QUICKSORT

2/12

conformada con los elementos de la izquierda y otra con los elementos de la derecha. Se repite el procedimiento en forma recursiva, primero con la lista de la izquierda y después con la lista de la derecha.

Al final llega el momento en que se tienen listas de tamaño 1 en un orden ascendente o descendente según sea el caso.

QUICKSORT

El algoritmo es el siguiente: 3/12

```
PROCEDURE QUICKSORT(X:ARR);
   PROCEDURE SORT(L,R:INTEGER);
   VAR
      I,J,V,W : INTEGER;
   BEGIN
      I:=L; J:=R; V:=X[I];
      REPEAT
         WHILE X[I] < V DO I:=I+1;
         WHILE V < X[J] DO J:=J-1;
         IF I<=J THEN
            BEGIN
               W:=X[I];
               X[I]:=X[J];
```

QUICKSORT

4/12

```
                X[J]:=W;
                I:=I+1;
                J:=J-1;
             END;
      UNTIL I>J;
      IF L<J THEN SORT(L,J);
      IF I<R THEN SORT(I,R);
   END; (* SORT *)
BEGIN
   SORT(1,XMAX);
END; (* QUICKSORT *)
```

QUICKSORT

5/12

Existe también otro algoritmo llamado quick sort modificado el cual en lugar de seleccionar el primer elemento de arreglo escoge un elemento intermedio.

El escoger un elemento intermedio en algunas ocaciones (no siempre) reduce la cantidad de proceso necesaria para colocar a ese elemento en su posición final ántes de particionar la lista.

QUICKSORT

El algoritmo es el siguiente: 6/12

```
PROCEDURE QUICKSORT_MODIFICADO(X:ARR);
   PROCEDURE SORT(L,R:INTEGER);
   VAR
      I,J,V,W : INTEGER;
   BEGIN
      I:=L; J:=R; V:=X[(L+R) DIV 2];
      REPEAT
         WHILE X[I] < V DO I:=I+1;
         WHILE V < X[J] DO J:=J-1;
         IF I<=J THEN
            BEGIN
               W:=X[I];
               X[I]:=X[J];
```

QUICKSORT 7/12

```
                X[J]:=W;
                I:=I+1;
                J:=J-1;
              END;
        UNTIL I>J;
        IF L<J THEN SORT(L,J);
        IF I<R THEN SORT(I,R);
    END; (* SORT *)
BEGIN
    SORT(1,XMAX);
END; (* QUICKSORT_MODIFICADO *)
```

QUICKSORT 8/12

En ambos casos el algoritmo es muy eficiente cuando la lista de elementos es muy grande. Si el número de elementos en la lista es pequeño el comportamiento del algoritmo es muy similar a cualquier otro.

Es importante hacer notar que no es restricción el programar este algoritmo en forma recursiva, sin embargo es esta caracteristica lo que lo hace tan eficiente.

QUICKSORT 9/12

Oprimiendo la tecla F8 generarás una lista de números en forma aleatoria.

Al oprimir la tecla F10, en el centro de la pantalla podrás observar la siguiente frase: F9-Quick Sort y F10-Quick Sort Modificado.

Oprimiendo la tecla correspondiente podrás observar el algoritmo en pantalla. (Una vez que tengas el arreglo de datos en pantalla, no olvides oprimir la tecla RETURN para comenzar).

QUICKSORT

10/12

En el caso de que el arreglo de datos a ordenar (para nuestro ejemplo en forma ascendente) ya se encuentre ordenado, el algoritmo requiere de menos intercambios para colocar al elemento seleccionado en su posición correcta dentro de la lista, por lo que el ordenamiento se desarrolla sumamente rápido. Es importante notar que bajo estas circunstancias el quick sort modificado es mas rápido que el quick sort normal.

Oprime la tecla F5 para generar un vector de datos ordenado en forma ascendente y posteriormente oprime la tecla F10 para observar

QUICKSORT

11/12

el comportamiento de los algoritmos bajo estas condiciones.

Cuando el arreglo de datos a ordenar (en nuestro ejemplo en forma ascendente) se encuentra ordenado a la inversa (en forma descendente), el algoritmo al buscar la posición final del primer elemento seleccionado, invierte el orden de todos los elementos del arreglo quedando éstos paracticamente ordenados.

Es importante notar que bajo éstas circunstancias el quick sort normal es más rápido que el quick sort modificado.

QUICKSORT

12/12

Oprime la tecla F6 para generar un vector de datos ordenado en forma descendente y posteriormente oprime la tecla F10 para observar el comportamiento de los algoritmos bajo estas condiciones.

INSERCION DIRECTA

1/5

Los métodos por inserción consisten en seleccionar un elemento del arreglo fuente y transferirlo dentro de una secuencia de destino insertándolo en el lugar apropiado.

INSERCION DIRECTA

El algoritmo es el siguiente: 2/5

```
PROCEDURE INSERCION_DIRECTA(X:ARR);
VAR
   I,J,K : INTEGER;
BEGIN
   FOR I:=2 TO XMAX DO
      BEGIN
         K:=X[I]; X[0]:=K; J:=I-1;
         WHILE K < X[J] DO
            BEGIN
               X[J+1]:=X[J];
               J:=J-1;
            END;
         X[J+1]:=K;
```

INSERCION DIRECTA

3/5

```
      END;
END; (* INSERCION_DIRECTA *)
```

El algoritmo es muy ineficiente cuando la lista pasa de 10 ó 15 elementos.

Oprimiendo la tecla F8 generarás una lista de números en forma aleatoria.

Al oprimir la tecla F10 podrás observar el algoritmo en pantalla. (Una vez que tengas el arreglo de datos en pantalla, no olvides oprimir la tecla RETURN para comenzar).

INSERCION DIRECTA

4/5

En el caso de que el arreglo de datos a ordenar (para nuestro ejemplo en forma ascendente) ya se encuentre ordenado, el algoritmo desarrolla por cada elemento a insertar todas las comparaciones. Sin embargo, dado que el elemento ya se encuentra en su posición final no desarrolla ningún intercambio.

Oprime la tecla F5 para generar un vector de datos ordenado en forma ascendente y posteriormente oprime la tecla F10 para observar el comportamiento del algoritmo.

INSERCION DIRECTA

5/5

El caso más crítico se obtiene cuando el arreglo de datos a ordenar (en nuestro ejemplo en forma ascendente) se encuentra ordenado a la inversa (en forma descendente). En esta situación el algoritmo lleva a cabo un intercambio en todas las comparaciones por cada elemento a insertar.

Oprime la tecla F6 para generar un vector de datos ordenado en forma descendente y posteriormente oprime la tecla F10 para observar el comportamiento del algoritmo bajo estas condiciones.

INSERCION BINARIA

1/7

Este método es una modificación del método de inserción directa. En él, se desarrolla una búsqueda binaria en el vector fuente para localizar el punto de inserción.

INSERCION BINARIA

El algoritmo es el siguiente: 2/7

```
PROCEDURE INSERCION_BINARIA(X:ARR);
VAR
   I,J,K,L,R,M : INTEGER;
BEGIN
   FOR I:=2 TO XMAX DO
      BEGIN
         K:=X[I]; L:=1; R:=I-1;
         WHILE L<=R DO
            BEGIN
               M:=(L+R) DIV 2;
               IF K<X[M] THEN R:=M-1
               ELSE L:=M+1;
            END;
```

INSERCION BINARIA 3/7

```
         FOR J:=I-1 DOWNTO L DO
            BEGIN
               X[J+1]:=X[J];
            END;
         X[L]:=K;
      END;
END; (* INSERCION_BINARIA *)
```

INSERCION BINARIA 4/7

El algoritmo es muy ineficiente cuando la lista pasa de 10 ó 15 elementos.

Oprimiendo la tecla F8 generarás una lista de números en forma aleatoria.

Al oprimir la tecla F10 podrás observar el algoritmo en pantalla. (Una vez que tengas el arreglo de datos en pantalla, no olvides oprimir la tecla RETURN para comenzar).

INSERCION BINARIA

5/7

En el caso de que el arreglo de datos a ordenar (para nuestro ejemplo en forma ascendente) ya se encuentre ordenado, el algoritmo desarrolla por cada elemento a insertar la búsqueda binaria para conocer su posición final. Sin embargo, dado que el elemento ya se encuentra en su posición final no desarrolla ningún intercambio.

Oprime la tecla F5 para generar un vector de datos ordenado en forma ascendente y posteriormente oprime la tecla F10 para observar el comportamiento del algoritmo.

INSERCION BINARIA

6/7

El caso más crítico se obtiene cuando el arreglo de datos a ordenar (en nuestro ejemplo en forma ascendente) se encuentra ordenado a la inversa (en forma descendente). En esta situación el algoritmo lleva a cabo por cada elemento a insertar, una búsqueda binaria de la posición final y un intercambio con todos elementos de la lista en su parte ya ordenada. (La posición final de cada nuevo elemento siempre es el inicio de la lista).

Oprime la tecla F6 para generar un vector de datos ordenado en forma descendente y posteriormente oprime la tecla F10 para observar

INSERCION BINARIA

7/7

el comportamiento del algoritmo bajo estas condiciones.

INSERCION BINARIA

5/7

En el caso de que el arreglo de datos a ordenar (para nuestro ejemplo en forma ascendente) ya se encuentre ordenado, el algoritmo desarrolla por cada elemento a insertar la búsqueda binaria para conocer su posición final. Sin embargo, dado que el elemento ya se encuentra en su posición final no desarrolla ningún intercambio.

Oprime la tecla F5 para generar un vector de datos ordenado en forma ascendente y posteriormente oprime la tecla F10 para observar el comportamiento del algoritmo.

INSERCION BINARIA

6/7

El caso más crítico se obtiene cuando el arreglo de datos a ordenar (en nuestro ejemplo en forma ascendente) se encuentra ordenado a la inversa (en forma descendente). En esta situación el algoritmo lleva a cabo por cada elemento a insertar, una búsqueda binaria de la posición final y un intercambio con todos elementos de la lista en su parte ya ordenada. (La posición final de cada nuevo elemento siempre es el inicio de la lista).

Oprime la tecla F6 para generar un vector de datos ordenado en forma descendente y posteriormente oprime la tecla F10 para observar

INSERCION BINARIA

7/7

el comportamiento del algoritmo bajo estas condiciones.

INSERCION SHELL

1/9

Este método también conocido como de inserción por decrementos disminuidos fue propuesto por D.L. Shell en 1959.

Primero, todos los elementos que están separados por 9 posiciones son agrupados y ordenados en forma separada. Esta acción es la primera pasada. Después, los elementos son reagrupados en grupos que están separados por 5 posiciones y son ordenados nuevamente. (2da. pasada).

En la tercera pasada, los elementos son reagrupados en grupos que están separados por 3 posiciones y también se ordenan. Finalmente, en

INSERCION SHELL

2/9

la cuarta pasada, todos los elementos son ordenados de forma ordinaria. (1 a 1).

A la primera pasada se le llama Sort-9, a la segunda Sort-5, a la tercera Sort-3 y a la cuarta Sort-1.

Es recomendable que los incrementos sean potencias de 2, pero en algunos casos es más eficiente el utilizar otros incrementos.

La única condición es que el último incremento sea 1 y que cualquier incremento sea menor que su incremento anterior.

INSERCION SHELL

3/9

El ordenamiento por grupo se implanta como inserción directa y debe ser tan simple como se pueda, sin embargo cada grupo necesita tener su propia condición de salida (llamada centinela), de ahí que el vector de datos a ordenar se declare como:

```
X :  ARRAY [-9 .. XMAX] of item;
```

INSERCION SHELL

El algoritmo es el siguiente: 4/9

```
PROCEDURE SHELL(X:ARR);
CONST
   T  =  4;
VAR
   I,J,K,S,Y : INTEGER;
   M         : 1..T;
   H         : ARRAY [1..T] OF INTEGER;
BEGIN
   H[1]:=9; H[2]:=5; H[3]:=3; H[4]:=1;
   FOR M:=1 TO T DO
      BEGIN
         K:=H[M];
         S:=-K;
```

INSERCION SHELL

5/9

```
         FOR I:=K+1 TO XMAX DO
            BEGIN
               Y:=X[I]; J:=I-K;
               IF S=0 THEN S:=-K;
               S:=S+1;
               X[S]:=Y;
               WHILE Y<X[J] DO
                  BEGIN
                     X[J+K]:=X[J];
                     J:=J-K;
                  END;
               X[J+K]:=Y;
            END;
      END;
END; (* SHELL *)
```

INSERCION SHELL

6/9

El algoritmo es eficiente tanto para listas pequeñas como grandes y su comportamiento es aceptable independientemente de como sean los datos.

Oprimiendo la tecla F8 generarás una lista de números en forma aleatoria.

Al oprimir la tecla F10 podrás observar el algoritmo en pantalla. (Una vez que tengas el arreglo de datos en pantalla, no olvides oprimir la tecla RETURN para comenzar).

INSERCION SHELL

7/9

En el caso de que el arreglo de datos a ordenar (para nuestro ejemplo en forma ascendente) ya se encuentre ordenado, el algoritmo desarrolla los agrupamientos por cada centinela (4 en este caso) y al ordenar cada grupo por el método de inserción directa, dado que los elementos ya se encuentran en su posición final no desarrolla ningún intercambio.

Oprime la tecla F5 para generar un vector de datos ordenado en forma ascendente y posteriormente oprime la tecla F10 para observar el comportamiento del algoritmo.

INSERCION SHELL

8/9

El caso más crítico se obtiene cuando el arreglo de datos a ordenar (en nuestro ejemplo en forma ascendente) se encuentra ordenado a la inversa (en forma descendente). En esta situación el algoritmo lleva a cabo los agrupamientos por cada centinela y posteriormente los ordena con el método de inserción directa.

Como los grupos son pequeños, la cantidad de procesamiento necesaria para que cada elemento llegue a su posición final (dentro del agrupamiento por centinela) es relativamente poca.

INSERCION SHELL

9/9

Oprime la tecla F6 para generar un vector de datos ordenado en forma descendente y posteriormente oprime la tecla F10 para observar el comportamiento del algoritmo bajo estas condiciones.

SELECCION DIRECTA

1/9

Los métodos de selección, como su nombre lo indica, seleccionan del conjunto de datos al mayor o al menor (dependiendo del criterio) y lo excluye, y procede de forma similar con los restantes.

A este algoritmo, algunos autores como por ejemplo Knuth, lo llaman de inserción directa. Otros autores lo conocen como SINKING.

SELECCION DIRECTA

El algoritmo es el siguiente: 2/9

```
PROCEDURE SELECCION_DIRECTA(X:ARR);
VAR
   I,J,K : INTEGER;
BEGIN
   FOR I:=1 TO XMAX-1 DO
      FOR J:=I+1 TO XMAX DO
         IF X[J]<X[I] THEN
            BEGIN
               K:=X[I];
               X[I]:=X[J];
               X[J]:=K;
            END;
END; (* SELECCION_DIRECTA *)
```

SELECCION DIRECTA

3/9

Una modificación interesante de este algoritmo, se obtiene invirtiendo la secuencia de selección. El algoritmo tradicional barre de izquierda a derecha los elementos de la lista, pero si barremos de derecha a izquierda, al momento de ejecutar un intercambio, el elemento intercambiado queda casi en su posición final reduciendo considerablemente el número total de intercambios.

SELECCION DIRECTA

El algoritmo es el siguiente: 4/9

```
PROCEDURE SELECCION_DIRECTA_MODIFICADO(X:ARR);
VAR
   I,J,K : INTEGER;
BEGIN
   FOR I:=1 TO XMAX-1 DO
      FOR J:=XMAX DOWNTO I+1  DO
         IF X[J]<X[I] THEN
            BEGIN
               K:=X[I];
               X[I]:=X[J];
               X[J]:=K;
            END;
END; (* SELECCION_DIRECTA_MODIFICADO *)
```

SELECCION DIRECTA

5/9

Aunque el algoritmo no es muy eficiente, es tan sencillo de programar, que puede ser recomendable si la lista de elementos es pequeña.

Si la lista es grande, el algoritmo de selección directa modificado requiere de una cantidad menor de proceso que la mayoría de los algoritmos anteriormente mostrados a excepción del quicksort.

SELECCION DIRECTA

6/9

Oprimiendo la tecla F8 generarás una lista de números en forma aleatoria.

Al oprimir la tecla F10, en el centro de la pantalla podrás observar la siguiente frase: F9-Directa y F10-Directa Modificado.

Oprimiendo la tecla correspondiente podrás observar el algoritmo en pantalla. (Una vez que tengas el arreglo de datos en pantalla, no olvides oprimir la tecla RETURN para comenzar).

SELECCION DIRECTA

7/9

En el caso de que el arreglo de datos a ordenar (para nuestro ejemplo en forma ascendente) ya se encuentre ordenado, los algoritmos desarrollan unicamente las comparaciones sin llevar a cabo ningún intercambio.

Oprime la tecla F5 para generar un vector de datos ordenado en forma ascendente y posteriormente oprime la tecla F10 para observar el comportamiento de los algoritmos bajo estas condiciones.

SELECCION DIRECTA

8/9

Cuando el arreglo de datos a ordenar (en nuestro ejemplo en forma ascendente) se encuentra ordenado a la inversa (en forma descendente), el algoritmo de selección directa normal desarrolla por cada elemento de la lista una comparación e intercambios con todos los elementos de la lista hasta que cada elemento llega a su posición final.

El algoritmo de selección directa modificado requiere de una comparación y un intercambio por cada elemento de la lista, lo que lo hace muy eficiente bajo estas condiciones.

SELECCION DIRECTA

9/9

Oprime la tecla F6 para generar un vector de datos ordenado en forma descendente y posteriormente oprime la tecla F10 para observar el comportamiento de los algoritmos bajo estas condiciones.

HEAP SORT

1/17

El método de ordenamiento de TORNEO forma parte de los métodos de selección de tipo arboreo, y lleva este nombre por tener semejanza a la eliminación y calificación de competidores usado en los torneos.

En estos torneos existen rondas de eliminación/calificación, en las que los jugadores compiten por parejas. El ganador pasa a la siguiente ronda y el perdedor queda eliminado de la competencia.

El torneo termina cuando los competidores se reducen a sólo una pareja y al competir ésta,

HEAP SORT

2/17

sale el campeón del torneo.

La gráfica de torneo se puede representar como un árbol binario en el cual el nodo superior (raíz), representa al campeón.

Dado el siguiente vector de datos;

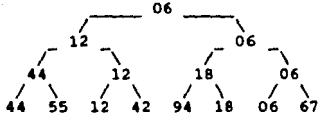
44 55 12 42 94 18 06 67

y considerando ganador al elemento menor, la primera ronda quedaría de la siguiente forma:

HEAP SORT

3/17

1era. Ronda

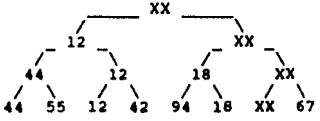


El valor de la llave raíz (campeón) pasará a ser el primer elemento del vector ordenado y su valor se cambia por un valor alto (XX, donde XX es infinito) para que en la siguiente

HEAP SORT

4/17

competencia quede en la raiz el siguiente valor menor del vector de datos.

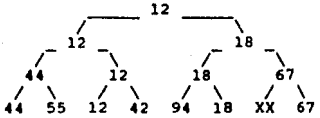


Una vez eliminado el elemento campeón, se reorganiza el árbol (2da. ronda), a partir del elemento que se eliminó.

HEAP SORT

5/17

2da. Ronda



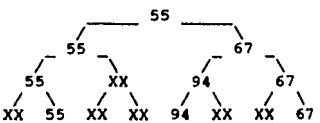
Este procedimiento se repite tantas veces como sea necesario para eliminar a todos los elementos de la competencia (XMAX-1 rondas).

Este método tiene la ventaja de evitar

HEAP SORT

6/17

comparaciones redundantes (-XX vs XX), pero sin embargo necesita memoria auxiliar en forma considerable y que muchas veces contiene nodos vacios.



HEAP SORT

J.W. Williams, propuso en 1964 un método con este principio pero que elimina la necesidad de utilizar una memoria auxiliar y nodos vacios. A este método se le llama HEAP SORT.

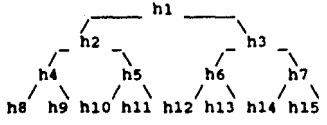
Un HEAP se define como una secuencia de llaves

h(l), h(l+1), ..., h(r)

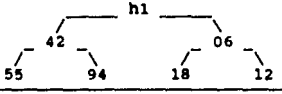
tal que h(i) <= h(2i)
h(i) <= h(2i+1)

para toda i = l..r/2, con un arreglo h visto como árbol binario.

HEAP SORT

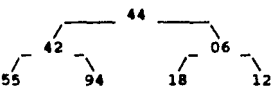


Si tenemos un árbol HEAP de 7 elementos como el siguiente:



HEAP SORT

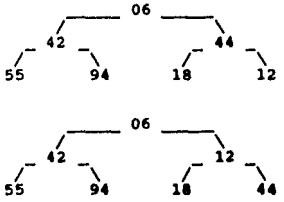
e insertamos una llave de valor 44, el árbol quedaria:



Pero para que sea un árbol HEAP se debe cumplir que sus hijos (del 44) sean mayores o iguales, por lo que debemos intercambiar los nodos hasta que se cumpla esta condición.

HEAP SORT

10/17



HEAP SORT

11/17

A este proceso de reacomodo se le llama SIFT y debe ejecutarse tantas veces como elementos vayamos a ordenar.

¿Como se desarrolla el proceso de oprdenamiento? Una vez formado el árbol HEAP con todos los elementos de la lista, se elimina el nodo raiz (campeón), se sustituye por infinito y se vuelve a ejecutar el proceso de reacomodo (SIFT) y asi sucesivamente hasta eliminar a todos los elementos de la competencia.

HEAP SORT

El algoritmo es el siguiente: 12/17

```
PROCEDURE HEAPSORT(X:ARR);
VAR
   L,R, K : INTEGER;

   PROCEDURE SIFT;
   LABEL
      13;
   VAR
      I,J : INTEGER;
   BEGIN
      I:=L;
      J:=2*I;
      K:=X[I];
```

HEAP SORT

13/17

```
      WHILE J<=R DO
        BEGIN
          IF J<R THEN
            IF X[J]<X[J+1] THEN J:=J+1;
          IF K>=X[J] THEN GOTO 13;
          X[I]:=X[J];
          I:=J;
          J:=2*I;
        END;
      13: X[I]:=K;
   END; (* SIFT *)
BEGIN
   L:=(XMAX DIV 2)+1;
   R:=XMAX;
```

HEAP SORT

14/17

```
   WHILE L>1 DO
      BEGIN
        L:=L-1;
        SIFT;
      END;
   WHILE R>1 DO
      BEGIN
        K:=X[1];
        X[1]:=X[R];
        X[R]:=K;
        R:=R-1;
        SIFT;
      END;
END; (* HEAPSORT *)
```

HEAP SORT

15/17

Este método no es muy recomendable para arreglos pequeños dado que los procesos de reacomodo consumen una cantidad considerable de proceso, sin embargo, en arreglos grandes se comporta de manera eficiente.

Oprimiendo la tecla F8 generarás una lista de números en forma aleatoria.

Al oprimir la tecla F10 podrás observar el algoritmo en pantalla. (Una vez que tengas el arreglo de datos en pantalla, no olvides oprimir la tecla RETURN para comenzar).

HEAP SORT

16/17

Si tomamos en cuenta que el algoritmo requiere formar un árbol HEAP a partir del arreglo de datos iniciales, el algoritmo se comporta de manera similar independientemente del orden inicial de los datos.

Es decir, la cantidad de proceso requerida para ordenar un arreglo es similar si el arreglo de datos a ordenar (para nuestro ejemplo en forma ascendente) ya se encuentre ordenado, o si el arreglo de datos se encuentra ordenado a la inversa (en forma descendente), o si el arreglo de datos tiene una distribución aleatoria.

HEAP SORT

17/17

Oprime la tecla F5 para generar un vector de datos ordenado en forma ascendente y posteriormente oprime la tecla F10 para observar el comportamiento del algoritmo.

Oprime la tecla F6 para generar un vector de datos ordenado en forma descendente y posteriormente oprime la tecla F10 para observar el comportamiento del algoritmo bajo estas condiciones.

3. Módulo de Búsqueda

BUSQUEDA SECUENCIAL 1/8

Los algoritmos de búsqueda se utilizan para localizar un registro que ha sido almacenado en la memoria de la máquina lo más rápido posible.

Al igual que en los algoritmos de ordenamiento, se supone que cada registro está compuesto de llave y contenido. La búsqueda se lleva a cabo siempre sobre el campo de la llave.

Las búsquedas se clasifican respecto a la memoria que utilizan en:

Internas.- Se desarrollan sobre listas en memoria principal.

BUSQUEDA SECUENCIAL 2/8

Externas.- Se desarrollan sobre archivos en memoria secundaria.

Por su operación se clasifican en:

Activas.- Modifican la lista o archivo.

Pasivas.- No modifican nada, sólo consultan.

El tipo de búsqueda seleccionado, dependerá del tiempo de búsqueda (que tan rápido queremos recuperar la información?) y las características de los datos (volumen, que tan ordenados estén,

BUSQUEDA SECUENCIAL

etc). 3/8

Existen dos tipos generales de búsqueda, por comparación y por transformación de llaves.

Los algoritmos de búsqueda por comparación de llaves tienen la característica común de realizar la comparación directa de la llave buscada con las llaves de los nodos en la estructura de datos (lista o archivo).

El algoritmo de búsqueda secuencial, también conocido como de fuerza bruta, consiste en el recorrido secuencial de principio a fin de los elementos de una lista.

BUSQUEDA SECUENCIAL

4/8

Es aplicable tanto a estructuras sin ordenar como ordenadas y es muy útil cuando la lista es pequeña y reside en memoria principal.

El algoritmo es el siguiente:

Dada una tabla de registros R1, R2, ..., Rn, cuyas respectivas llaves son K1, K2,...., Kn, este algoritmo busca un registro cuyo argumento vale K. Se asume que la tabla es mayor o igual a 1. N >= 1.

BUSQUEDA SECUENCIAL

5/8

```
A.  Inicio.-        I:=1;
B.  Compara.-       IF K=K(i) THEN OK en Reg # I;
C.  Incrementa.-    I:=I+1;
D.  Fin de
    Archivo.-       IF I<=N THEN GO TO B
                    ELSE No se encuentra;
```

Al oprimir la tecla F10 podrás observar el algoritmo en pantalla.

BUSQUEDA SECUENCIAL

6/8

Este algoritmo se puede mejorar para desarrollar una búsqueda más rápida, utilizando un registro adicional al final del archivo.

```
A.  Inicio.-      I:=1; K(N+1)=K;
B.  Compara.-     IF K=K(i) THEN GO TO D;
C.  Incrementa.- I:=I+1; GO TO B;
D.  Fin de
    Archivo.-     IF I<=N THEN OK en Reg # I
                  ELSE No se encuentra;
```

BUSQUEDA SECUENCIAL

7/8

Si la búsqueda fuera sobre una tabla ordenada, el algoritmo sería un poco más eficiente.
(K(i) <= K(i+1)).

```
A.  Inicio.-      I:=1; K(N+1)=XXX;
B.  Compara.-     IF K<=K(i) THEN GO TO D;
C.  Incrementa.- I:=I+1; GO TO B;
D.  Igualdad.-    IF K=K(i) THEN OK en Reg # I
                  ELSE No se encuentra;
```

donde XXX es infinito.

BUSQUEDA SECUENCIAL

8/8

En todos los ejercicios de simulación presentados en la sección de búsqueda, no se está tomando en cuenta el tiempo de ejecución de éstos, dado que todos los algoritmos dependen de la estructura inicial de los datos; la cual para efectos de simulación, se genera en forma aleatoria en todos los casos.

BUSQUEDA AUTO-ORGANIZABLE

1/3

Existe una regla cuando hacemos un análisis probabilistico de las búsquedas. Se llama la regla del 80-20, también conocida como la ley de Pareto.

De una tabla de búsqueda, la información útil ocupa el 20% de la tabla, mientras que el otro 80 por lo general no nos sirve.

BUSQUEDA AUTO-ORGANIZABLE

2/3

De esta regla, se derivó el siguiente algoritmo, que consiste en:

- Cuando se accesa un registro, se considera que éste pertenece a la parte útil (20%).
- Se transfiere ese registro de donde se encuentra al inicio de la tabla, con el fin de que cuando vuelva a ser búscado, el acceso sea más rápido.

El tipo de búsqueda es secuencial.

BUSQUEDA AUTO-ORGANIZABLE

3/3

Al oprimir la tecla F10 podrás observar el algoritmo en pantalla.

BUSQUEDA BINARIA

1/4

Este tipo de búsqueda parte del hecho de que la información tiene que estar ordenada y se basa en la bipartición repetida del intervalo de búsqueda.

Si se aplica este algortimo en una lista no ordenada, las llaves a buscar posiblemente nunca se encontrarian.

BUSQUEDA BINARIA

2/4

El algoritmo es el siguiente:

LI = Limite Inferior,
LD = Limite Superior,
K = Llave a buscar,
I = Indice donde se encontró, y,
X = Arreglo de llaves.

BUSQUEDA BINARIA

3/4

```
PROCEDURE BBINARIA(LI,LD,K,I);
VAR  H : INTEGER;
BEGIN
   H:=(LI+LD) DIV 2;
   IF  X[H]<>K THEN
      IF LD=LI THEN No existe
      ELSE IF K>H[K] THEN BBINARIA(H,LD,K,I)
           ELSE BBINARIA(LI,H,K,I)
    ELSE BEGIN
           I:=H;
           Si existe
         END;
END; (* BBINARIA *)
```

BUSQUEDA BINARIA

4/4

Al oprimir la tecla F10 podrás observar el algoritmo en pantalla.

HASH-CUADRADO MEDIO

1/4

Las búsquedas por transformación de llaves están basadas en la idea de calcular una dirección en forma directa, a partir de la modificación de la llave mediante una función.

A estas funciones de transformación se les llama funciones de dispersión (HASH).

Cuando a dos llaves diferentes se les aplica la transformación y la dirección de cada transformación es la misma, entonces tenemos un problema de colisión.

HASH-CUADRADO MEDIO

2/4

La función de dispersión debe ser rápida y debe evitar al máximo las colisiones.

El algoritmo del cuadrado medio (MID-SQUARE) consiste en tomar la llave, multiplicarla por si misma y seleccionar una parte de ella.

Ejemplos:

```
K  = 7521936
K^2= 56579521188096
        ###
Indice= 521
```

HASH-CUADRADO MEDIO

3/4

```
K  = 6879415
K^2= 47326350742225
          ###
Indice= 350

K  = 6879414
K^2= 47326336983396
          ###
Indice= 336
```

HASH-CUADRADO MEDIO

4/4

El tamaño máximo de mapeo está en función del número de dígitos seleccionados. En el caso de los ejemplos 10^3=1000.

Al oprimir la tecla F10 podrás observar el algoritmo en pantalla.

HASH-DOBLADOS

1/3

El algoritmo de doblados (FOLDING) consiste en tomar la llave y fraccionarla en partes. Se suman esas partes y se utilizan los dígitos menos significativos.

Ejemplo:

```
K = 923415728

k1= 923              923
k2=   415            415
k3=     728          728
                     ----
                     2066
```

HASH-CUADRADO MEDIO

2/3

Utilizando los tres digitos menos significativos el indice es igual a 066.

Si la llave es alfabética, a cada letra se le asigna un valor y a la posición relativa de cada simbolo un cierto peso.

Ejemplo:

```
      K = ARTURO
Peso = 1   1
         5   5
           5   5
             1   1
```

Valores

```
A = 1
R = 9
T = 2
U = 3
O = 6
```

HASH-CUADRADO MEDIO

3/3

```
A = 1 X 1 = 1
R = 9 X 5 = 45
T = 2 X 5 = 10
U = 3 X 1 = 3
R = 9 X 1 = 9
O = 6 X 5 = 30

  Indice = 98
```

Al oprimir la tecla F10 podrás observar el algoritmo en pantalla.

HASH-DIVISION

1/2

El algoritmo de división (DIVISION) consiste en tomar la llave y hacerle un división entera.

Ejemplo:

K = 50243

K DIV 100 = 502

Indice = 502

HASH-DIVISION

2/2

Al oprimir la tecla F10 podrás observar el algoritmo en pantalla.

HASH-CORRIMIENTOS

1/2

El algoritmo de corrimientos (SHIFTING) consiste en tomar de una llave de M digitos, N digitos (más o menos significativos).

Ejemplos:

```
K = 73214345          K = 73214345
        ###                   ###
Indice = 345          Indice = 732
```

Este algoritmo produce muchas colisiones.

HASH-CORRIMIENTOS

2/2

Al oprimir la tecla F10 podrás observar el algoritmo en pantalla.

Existen tantas funciones de dispersión como algoritmos en el mundo, dado que siempre es posible generar nuestra propia transformación.

Ejemplo:

```
Y = ((3*K^2+7)/(200*5) DIV 3);
```

COLISIONES-LINEAL

1/5

Cuando a dos llaves diferentes se les aplica la transformación y la dirección de cada transformación es la misma, entonces tenemos un problema de colisión.

Las colisiones se resuelven aplicando una segunda transformación a la llave, generando de esta forma una nueva dirección.

El factor de carga de una tabla se define como el número total de llaves a colocar entre el tamaño de la tabla.

FC = número de llaves / tamaño de la tabla

COLISIONES-LINEAL

2/5

Cuando decimos que una tabla tiene un factor de carga de 0.5 queremos decir que el número total de registros de la tabla es el doble que el número de llaves.

Ejemplo:

Número de llaves = 10
FC = 0.5

Tamaño de la tabla = 10/0.5 = 20

Si el factor de carga es cercano a uno, implica que el espacio disponible para resolver colisiones se reduce, lo que nos puede llevar a

COLISIONES-LINEAL

3/5

colisiones imposibles de resolver.

El algoritmo de manejo de colisiones tipo lineal consiste en moverse N posiciones adelante o atrás de la dirección obtenida por la función de dispersión.

Supone la existencia de una tabla circular.

Si N = 3, va saltando de 3 en 3 hasta encontrar un elemento en la tabla.

Ejemplo:

COLISIONES-LINEAL

K=10345232 4/5
Indice=3

Como está ocupado entonces
Indice=3+3=6

Como está ocupado entonces
Indice=6+3=9

Como está ocupado entonces
Indice=9+3=12

Si el registro 12 hubiera estado ocupado sería imposible resolver la colisión.

	Registros
1	
2	
3	Ocupado
4	
5	
6	Ocupado
7	
8	
9	Ocupado
10	
11	
12	10345232

COLISIONES-LINEAL

5/5

Al oprimir la tecla F10 podrás observar el algoritmo en pantalla.

COLISIONES-CUADRATICO

1/3

El algoritmo de manejo de colisiones cuadrático utiliza la siguiente fórmula para calcular la nueva dirección.

1er. intento l = l+1
2do. intento l = l+1+2
3er. intento l = l+1+2+3
4to. intento l = l+1+2+3+4
etc.....

donde l es el índice de la función de dispersión.

Supone la existencia de una tabla circular.

COLISIONES-CUADRATICO

```
K=10345232                                    2/3
Indice=3                                  Registros
                                        1 10345232
Como está ocupado entonces              2
Indice=3+1=4                            3 Ocupado
                                        4 Ocupado
Como está ocupado entonces              5
Indice=3+1+2=6                          6 Ocupado
                                        7
Como está ocupado entonces              8
Indice=3+1+2+3=9                        9 Ocupado
                                       10
Como está ocupado entonces             11
Indice=3+1+2+3+4=13                    12 Ocupado
Como la tabla es circular Indice=1
```

COLISIONES-CUADRATICO

3/3

Al oprimir la tecla F10 podrás observar el algoritmo en pantalla.

COLISIONES-DOBLE HASH

1/3

El algoritmo de manejo de colisiones doble hash es muy similar al de tipo lineal, la diferencia radica en que éste, utiliza una segunda función de hash para calcular el incremento. El valor de la segunda función debe ser mucho menor que el de la primera.

Supone la existencia de una lista circular.

Ejemplo:

COLISIONES-DOBLE HASH

K=10345232
Indice=3

Como está ocupado entonces
2do Hash=2; Indice=3+2=5

Como está ocupado entonces
Indice=3+2+2=7

2/3

	Registros
1	
2	
3	Ocupado
4	
5	Ocupado
6	
7	10345232
8	
9	Ocupado
10	
11	
12	Ocupado

COLISIONES-DOBLE HASH

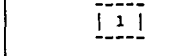
3/3

Al oprimir la tecla F10 podrás observar el algoritmo en pantalla.

COLISIONES-CUBETAS

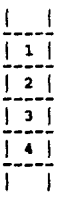
1/4

La función de hash que utiliza el algoritmo de manejo de colisiones de cubetas no nos proporciona el número de registro sino el de una cubeta.



COLISIONES-CUBETAS

2/4



La búsqueda se hace en forma secuencial dentro de la cubeta.

COLISIONES-CUBETAS

3/4

Ejemplo:

k:=	UNO	DOS	TRES	CUATRO	CINCO	SEIS	SIETE	OCHO
h(k)	1	2	2	1	2	1	3	1

1	UNO CUATRO SEIS OCHO	2	DOS TRES CINCO	3	SIETE

COLISIONES-CUBETAS

4/4

El problema es que hay cubetas que se llenan mucho (la cubeta número uno) y otras que se llenan poco (la cubeta número tres).

Al oprimir la tecla F10 podrás observar el algoritmo en pantalla.

COLISIONES-ENCADENAMIENTO FUSIONADO

1/3

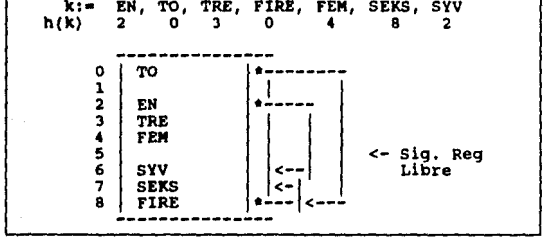
El algoritmo de manejo de colisiones de encadenamiento fusionado utiliza un apuntador al primer registro libre dentro de la estructura a partir de la parte inferior de la tabla.

Cuando se da una colisión, el elemento a insertar se coloca en el lugar a donde apunta esta variable y se crea un apuntador del registro ocupado a este nuevo registro.

La variable que apunta al primer registro libre se decrementa hasta encontrar el siguiente registro libre.

COLISIONES-ENCADENAMIENTO FUSIONADO

2/3



COLISIONES-ENCADENAMIENTO FUSIONADO

3/3

El apuntador al último registro libre se ilustrará en la simulación con un asterisco a la derecha del registro.

Al oprimir la tecla F10 podrás observar el algoritmo en pantalla.

3. Módulo de Ayuda

MENU PRINCIPAL

1/5

El sistema ICAI EDA tiene como objetivo en materia de aprendizaje lo siguiente:

"Ser un sistema tutorial que sirva como material de apoyo complementario a la enseñanza y que a su vez le permita al estudiante:

- Desarrollar sus propias estrategias e hipótesis del material a aprender.

- Descubrir por sí mismos las relaciones importantes entre las acciones y los resultados.

MENU PRINCIPAL

2/5

- Tener una fuente de consulta que le indique las cuestiones relevantes y lo oriente en la adquisición de conocimiento importante".

Para cumplir con este objetivo, el sistema EDA desarrolla un método de enseñanza conocido como socrático, el cual utiliza un texto para guiar al alumno. Este texto se encuentra mezclado con el tema que se imparte en el sistema tutorial.

MENU PRINCIPAL

3/5

EDA también constituye "un ambiente de aprendizaje reactivo" que le permite al alumno desarrollar sus propias estrategias e hipótesis del material a aprender y descubrir por sí mismo las relaciones importantes entre las acciones y los resultados.

EDA utiliza un sistema de ventanas en el cual un menú queda enmarcado por una caja. Los renglones de la caja representan las distintas opciones del menú. Una opción se selecciona oprimiendo la tecla "RETURN". La opción actual se identifica facilmente en la pantalla ya que se despliega con un color diferente.

MENU PRINCIPAL

4/5

Las teclas de flecha hacia arriba y flecha hacia abajo, nos permiten movernos dentro del menú; este menú utiliza una estructura de lista circular. Cada renglón dentro del menú puede ser seleccionado en una forma rápida oprimiendo la primera letra de cada renglón.

Cuando de un menú llamamos a otro, el segundo se encima en el primero y al salir de este, redibuja el primer menú. En todos los menús la tecla "F1" se utiliza como opción de ayuda y la tecla "ESCAPE" para regresar al menú anterior.

MENU PRINCIPAL

5/5

El menú principal utiliza las teclas de flecha hacia la derecha y flecha hacia la izquierda para moverse en el menú y las teclas "RETURN" y flecha hacia abajo para seleccionar la opción.

MENU DE ORDENAMIENTO

1/2

El módulo de ordenamiento trabaja con los siguientes algoritmos de ordenamiento: En el grupo de intercambio, burbuja, shaker sort y quick sort; En el grupo de inserción, directa, binaria y shell; En el grupo de selección, directa y heap sort.

En este nivel se encuentra una opción que se llama comparativo, la cual nos permite ejecutar la animación gráfica de todos los algoritmos de ordenamiento con los mismos datos y nos muestra una tabla resúmen indicándonos por cada algoritmo el tiempo requerido para su ejecución y un índice con respecto al más rápido de todos.

MENU DE ORDENAMIENTO

2/2

Al seleccionar cualquier algoritmo de ordenamiento aparece un menú con las siguientes opciones: algoritmo, simulación y animación gráfica.

Estas opciones se explican a detalle en el menú correspondiente.

ALGORITMO, SIMULACION, ANIMACION

1/7

Opción de algoritmo:

Al oprimir esta opción aparece en el centro de la pantalla una ventana que en su parte superior nos muestra el nombre del algoritmo seleccionado. En su parte inferior nos muestra los comandos que se pueden ejecutar en este módulo, "ESCAPE" para regresar al menú anterior y las teclas "PgUp" y "PgDn" para cambiar los textos. En la esquina superior derecha se muestra el número de páginas que componen el total por algoritmo y el número de página actual.

ALGORITMO, SIMULACION, ANIMACION

2/7

El texto por algoritmo se compone de los siguientes módulos:

a) Presentación general del algoritmo en cuestión; en el caso del algoritmo burbuja se incluye también una presentación general del tema métodos de ordenamiento;

b) El código fuente desarrollado en lenguaje pascal;

c) Comentarios sobre su operación (cuando es eficiente, cuando no, caso crítico, tipo de

ALGORITMO, SIMULACION, ANIMACION

3/7

operaciones involucradas, etc..) y la secuencia de teclas de función necesaria para observar la animación del algoritmo en pantalla. (Se prueban todos los casos; con datos aleatorios, con datos ya ordenados y con datos ordenados en forma contraria a como se quieren ordenar).

ALGORITMO, SIMULACION, ANIMACION

4/7

Opción de simulación:

Al oprimir esta opción aparece en el centro de la pantalla una ventana dividida en dos bloques principales. En la parte superior nos muestra el nombre del algoritmo seleccionado y un vector de 8 números. Este vector constituye los datos para la simulación y es el mismo para todos los algoritmos. La parte inferior se compone de dos líneas de texto en donde se despliega los comentarios sobre las operaciones que se están llevando acabo en la simulación.

ALGORITMO, SIMULACION, ANIMACION

5/7

La simulación puede ejecutarse en dos modos diferentes, total o parcial. En el modo total, todas las operaciones que se llevan acabo en la simulación, aparecen a manera de comentarios en la parte inferior de la ventana. En el modo parcial, unicamente se muestra los números involucrados en las operaciones sin hacer referencia al tipo de operación. El sistema da la facilidad de cambiar el modo de operación al oprimir cualquier tecla.

ALGORITMO, SIMULACION, ANIMACION

6/7

Opción de animación gráfica:

Al seleccionar esta opción, la computadora cambia de modo texto a modo gráfico. En la parte superior de la pantalla aparece el nombre del algoritmo a animar. Al centro de la pantalla se generan una serie de barras verticales en un color diferente al del fondo; cada barra representa a un elemento del vector de datos a ordenar y el tamaño de la barra a su valor numérico.

ALGORITMO, SIMULACION, ANIMACION

7/7

Desde el punto de vista del procesamiento de información, lo más costoso en términos de tiempo de procesador en los algoritmos de ordenamiento son los intercambios. Para poder observar como algunos algoritmos utilizan demasiados intercambios, éstos se representan como una barra de otro color.

Para iniciar la ejecución de la animación se debe oprimir la tecla "RETURN". Al finalizar la animación, el sistema muestra el tiempo que se requirió para ordenar el vector de datos.

MENU DE BUSQUEDA

1/2

El módulo de búsqueda trabaja con los siguientes algoritmos: En búsqueda por comparación de llaves con secuencial, auto-organizable y binaria; En búsqueda por transformación de llaves con cuadrado medio, doblados, división y corrimientos; En el manejo de colisiones con lineal, cuadrático, doble hash, cubetas y encadenamiento fusionado.

MENU DE BUSQUEDA

2/2

Al seleccionar cualquier opción de este menú aparece un menú con las siguientes opciones: algoritmo y simulación.

Estas opciones se explican a detalle en el menú correspondiente.

ALGORITMO/SIMULACION

1/5

Opción de algoritmo:

Al centro de la pantalla aparece una ventana que en su parte superior nos muestra el nombre del algoritmo seleccionado. En su parte inferior nos muestra los comandos que se pueden ejecutar en este módulo, "ESCAPE" para regresar al menú anterior y las teclas "PgUp" y "PgDn" para cambiar los textos. En la esquina superior derecha se muestra el número de páginas que componen el total por algoritmo y el número de página actual.

ALGORITMO/SIMULACION

2/5

El texto por algoritmo se compone de los siguientes módulos:

a) Presentación general del algoritmo en cuestión; en el caso del primer algoritmo de cada grupo se incluye también una introducción general al tema correspondiente.

b) El código fuente desarrollado en lenguaje pascal o en su defecto en pseudocódigo.

c) La secuencia de teclas de función necesaria para observar la simulación del algoritmo en pantalla.

ALGORITMO/SIMULACION

3/5

Opción de simulación:

Al oprimir esta opción aparece en el centro de la pantalla una ventana dividida en tres bloques principales. En la parte superior nos muestra el nombre del algoritmo a simular. Del lado izquierdo en forma aleatoria genera 7 llaves, estas llaves constituyen los datos para la simulación. Del lado derecho crea una tabla de 15 elementos. La parte inferior se compone de dos líneas de texto en donde se despliega los comentarios sobre las operaciones que se están llevando acabo en la simulación.

ALGORITMO/SIMULACION

4/5

La simulación puede ejecutarse en dos modos diferentes, total o parcial. En el modo total, todas las operaciones que se llevan acabo en la simulación, aparecen a manera de comentarios en la parte inferior de la ventana. En el modo parcial, unicamente se muestra los registros involucrados en las operaciones sin hacer referencia al tipo de operación. El sistema da la facilidad de cambiar el modo de operación al oprimir cualquier tecla.

ALGORITMO/SIMULACION

5/5

En la simulación de los algoritmos para el manejo de colisiones, se incluye otra variable que es el factor de carga (F7-0.54, F8-0.64, F9-0.77 y F10-1.0). Dependiendo del factor de carga seleccionado se generan más o menos registros ocupados en la tabla de registros.

MODULO DE DATOS

1/2

El módulo de datos esta ligado al módulo de ordenamiento y es aqui donde se selecciona el volumen de datos con el cual el sistema va a trabajar y las caracteristicas de estos.

Al seleccionar esta opción aparece en pantalla una caja con la siguiente leyenda: Número de datos a ordenar, F2-2, F3-25, F4-50, F5-75, F6-100, F7-150, F8-200, F9-250 y F10-300. El sistema opera con un vector de datos y sobre este ejecuta los algortimos de ordenamiento, al oprimir la tecla de función correspondiente seleccionamos el número de datos a ordenar.

MODULO DE DATOS

2/2

Una vez seleccionado el volumen de datos, aparece la siguiente leyenda: F4-Definidos por el Usuario, F5-Ordenado Ascendente, F6-Ordenado Descendente, F7-Semi-Ordenados y F8-Aleatorios.

Al oprimir la tecla de función correspondiente se definen las caracteristicas de los datos. Si la tecla de función oprimida es F4, el sistema le permite al usuario introducir sus propios datos.