A. Graficación

I. Aspectos Generales

1. Introducción a la graficación

- **Definición**

El uso adecuado y provechoso de la tecnología han hecho de la computadora un dispositivo poderoso para producir imágenes en forma rápida y económica. Actualmente en todas las áreas es posible aplicar gráficas por computadora con algún objetivo, por ello se ha generalizado la utilización de gráficas por computadora. De igual modo las gráficas por computadora se utilizan de manera rutinaria en diversas áreas, como en la ciencia, ingeniería, empresas, industria, gobierno, arte, entretenimiento, publicidad, educación, capacitación y presentaciones gráficas.

Las computadoras se han convertido en una herramienta poderosa para producir imágenes, interpretar información o mejorar la calidad de visualización de las mismas en forma rápida y económica. Debemos aclarar que los métodos que se utilizan en las gráficas por computadora y en procesamiento de imágenes tienen características similares pero no son iguales es decir, las dos áreas realizan, en forma fundamental operaciones distintas. Las herramientas para graficación por computadoras, se utilizan para crear una o más imágenes. Por otro lado, en el procesamiento de imágenes se aplican técnicas para modificar o interpretar imágenes existente como fotografías y rastreos de televisión. El intéres por los métodos de tratamiento o procesamiento digital de imágenes deriva de dos áreas principales:

Objetivo:
Estudiar los principios y metodologías necesarias para la representación, manipulación y despliegue de figuras e imágenes en dos y tres dimensiones, considerando los dispositivos de hardware con características específicas para procesos de graficación.

- Aplicaciones

Diseño Asistido por Computadora

El método de diseño asistido por computadora, conocido por lo general como CAD (computer assisted design), ahora se utilizan de forma rutinaria en el diseño de construcciones, automóviles, aeronaves, embarcaciones, naves espaciales, computadoras, telas y muchos productos. En el caso de algunas aplicaciones de diseño, los objetos se despliegan primero en forma de armazón mostrando la forma general y sus características internas. Los despliegues del armazón permiten ver a los diseñadores con rapidez los efectos de ajustes interactivos para diseñar formas. Regularmente, los paquetes de software de aplicaciones de CAD ofrecen al diseñador un entorno con ventanas múltiples; estas diversas ventanas desplegables muestran secciones amplificadas de vistas de diferentes objetos. Los circuitos y las redes para comunicaciones, abastecimientos de agua y otros servicios públicos se construyen a través de la colocación repetida de algunas formas gráficas. Las formas usadas en un diseño representan los diversos componentes del circuito o de la red. Con el paquete de diseño se ofrecen formas estándar para circuitos eléctricos, electrónicos y lógicos. Para otras aplicaciones, un diseñador puede crear símbolos personalizados empleados necesariamente para construir la red o el circuito. Así, se diseña el sistema colocando sucesivamente los componentes en el esquema, con el paquete de gráficas ofreciendo de manera automática las conexiones entre los componentes. Esto permite al diseñador experimentar rápidamente con esquemas de circuitos alternativos para reducir al mínimo el número de componentes o el espacio para el sistema. Con frecuencia se utilizan las animaciones en las aplicaciones del CAD. Las animaciones en tiempo real que emplean despliegues de armazones en

un monitor de video son útiles para probar el comportamiento de un vehículo o un sistema. Cuando no desplegamos objetos con superficies presentadas, pueden realizarse con rapidez los cálculos correspondientes a cada segmento de la animación para así crear un movimiento suave de tiempo real en la pantalla. Igualmente, los despliegues de armazones permiten al diseñador ver el interior del vehículo y observar el comportamiento de los componentes internos durante el movimiento. Las animaciones en entornos de realidad virtual se utlizan para determinar la forma como influyen ciertos movimientos en los operadores de vehículos. Por ejemplo, el operador de un tractor con ayuda de un dispositivo montado sobre la cabeza que presenta una vista estereoscópica del cucharón del cargador frontal o del retro excavador, manipula los controles como si se encontrara en el asiento del tractor. Esto permite al diseñador explorar diversas posiciones del cucharón o del retro excavador que pudieran obstruir la visión del operador. Cuando los diseños de objetos están completos o casi completos, se aplican modelos de iluminación realista y presentaciones de superficies para producir despliegues mostrando la apariencia del producto final. También se crean despliegues realistas para la publicidad de automóviles y otros vehículos mediante efectos especiales de iluminación y escenas de fondo. El proceso de manufactura también se asocia con la descripción por computadora de objetos diseñados para automatizar la construcción del producto. Por ejemplo, se puede convertir el esquema de un tablero de circuitos en una descripción de los procesos individua-les necesarios para elaborar el esquema. Algunas partes mecánicas se frabican por medio de la descripción de cómo se deben formar las superficies con herramientas. Luego, se ajustan las herramientas controladas de manera numérica para fabricar la parte de acuerdo con estos planos de construcción. Los arquitectos utilizan métodos gráficos interactivos para proyectar plantas arquitec-tónicas donde se muestra la disposición de habitaciones, ventanas, escaleras, anaqueles, barras de cocina y otras características de la construcción. A partir del displiegue del plano de una construcción en un monitor de video, un diseñador eléctrico puede experimentar con instalaciones para cableado, conexiones eléctricas y sistemas de alarma de incendios. Del mismo modo, aplicando paquetes para el esquema de instalaciones se determina la utilización del espacio en una oficina o en una planta de fabricación. Despliegues realistas de diseños arquitectónicos permiten a los arquitectos y a sus clientes estudiar la apariencia de una construcción particular o de un grupo de ellas, como un campus universitario o un complejo industrial. Con los sistemas de realidad virtual, los diseñadores pueden simular un "recorrido" por las habitaciones o alrededor de construcciones para apreciar mejor el efector general de un diseño particular. Además de presentar despliegues de fachadas realistas, los paquetes de CAD para arquitectura ofrecen medios para experimentar con planos interiores tridimensionales y la iluminación. Muchas otras clases de sistemas y productos se diseñan usando ya sea paquetes de CAD generales o software de CAD desarrollado en forma especial.

Arte por Computadora

Los métodos de gráficas por computadora se utilizan en forma generalizada tanto en aplicaciones de bellas artes como en aplicaciones de arte comercial. Los artistas utilizan una variedad de métodos computacionales, incluyendo hardware para propósitos especiales, programas artísticos de brocha de pintar del artista (como Lumena), otros paquetes de pintura (como PixelPaint y SuperPaint), software desarrollado de manera especial, paquetes de matemática simbólica (como Mathematica), paquetes de CAD, software de edición electrónica de publicaciones y paquetes de animaciones que proporcionan los medios para diseñar formas de objetos y especificar movimientos de objetos. La idea básica del programa paintbrush (brocha de pintar) permite a los artistas "pintar" imágenes en la pantalla de un monitor de video. En realidad, la imagen se pinta por lo general de manera electrónica en una tableta de gráficas (digitalizador) utilizando un estilete, el cual puede simular diferentes trazos, anchuras de la brocha y colores. Los creadores de bellas artes emplean diversas tecnologías de computación para producir imágenes. Con el propósito de crear pinturas el artista utiliza una combinación de paquetes de modelado tridimensional, diagramación de la textura, programas de dibujo y software de CAD. En un ejemplo de "arte metamático" un artista utilizó una combinación de funciones matemáticas, procedimientos fractales, software de Matematica, impresoras de chorro de tinta y otros sistemas con el fin de crear una variedad de formas tridimensionales y bidimensionales, al igual que pares de imágenes estereoscópicas. Otro ejemplo arte electrónico creado a partir de relaciones matemáticas es la obra

de un compositor que está diseñada en relación con las variaciones de la frecuencia y otros parámetros en una composición musical para producir un video el cual integra patrones visuales y auditivos. También se aplican estos métodos en el arte comercial para crear logotipos y otros diseños, distribuciones de página que combinan texto y gráficas, anuncios publicitarios por televisión y otras áreas. Para muchas aplicaciones de arte comercial (y películas, al igual que otras aplicaciones), se emplean técnicas fotorrealistas para presentar imágenes de un producto. Las animaciones también se utilizan con frecuencia en publicidad y los comerciales de televisión se producen cuadro por cuadro, donde cada cuadro del movimiento se presenta y graba como un archivo de imagen. Se simula el movimiento al mover ligeramente las posiciones de los objetos con respecto a las del cuadro anterior. Una vez presentados todos los cuadros de la secuencia de animación, se transfieren a película o se almacenan en un búfer de video para hacer una reproducción. Las animaciones en película requieren 24 cuadros por cada segundo de la secuencia de animación. Si se reproduce en un monitor de video, se requieren de 30 cuadros por segundo. Un método común de gráficas que se utilizan en muchos comerciales es la transformación (morphing), donde se transforma un objeto en otro (metamorfosis). En televisión se ha empleado para transformar una lata de aceite en un motor de automóvil, un automóvil en un tigre, un charco en una llanta y el rostro de una persona en otro.

Aplicaciones.

La graficación por computador se emplea hoy en día en varias áreas de la industria, los negocios, el gobierno y el entretenimiento. La lista de aplicaciones es enorme y crece con rapidez a medida que los computadores con capacidades graficas se convierten en artículos de consumo. A continuación se verán algunas de estas áreas:

a. **Interfaces con el usuario:** Si ha trabajado con Machintosh o con un computador personal compatible con IBM que ejecute Windows usted es prácticamente un avezado usuario de los gráficos. Después de todo la mayoría de las aplicaciones que se ejecutan en computadores personales o estaciones de trabajo tienen interfaces con el usuario que se apoyan en sistemas de ventanas para administrar múltiples actividades simultaneas, así como seleccionar elementos de los menús, iconos y objetos de pantalla. Los programas de procesamiento de texto, hoja de calculo y compuedición son aplicaciones típicas que aprovechan estas técnicas de interfaz con el usuario.

b. **Graficación interactiva en los negocios, la ciencia** y **la tecnología:** Esta es la siguiente aplicación más usual de los gráficos es para crear gráficos bidimensionales y tridimensionales de funciones matemáticas, físicas y económicas; histogramas y diagramas de barras y circulares; diagramas de programación de actividades; diagramas de inventario y producción. Todos estos gráficos se usan para presentar de manera significativa y concisa las tendencia y los patrones extraídos de los datos, para así esclarecer fenómenos complejos y facilitar la toma de decisiones informada.

c. **Cartografía:** La graficación por computador se emplea para producir representaciones precisas y esquemáticas de fenómenos geográficos y de otro tipo, a partir de datos de mediciones. Como ejemplos están los mapas geográficos, meteorológicos, de contorno y de densidad de población.

d. **Medicina:** La graficación por computador tiene una función cada vez más importante en campos como la medicina de diagnostico y la planificación de cirugías. En el segundo caso, los cirujanos usan los gráficos como auxiliares para guiar instrumentos y determinar precisamente donde hay que eliminar tejidos enfermos. p .e **MRI** magnetic resonance imaging) imágenes por resonancia magnética.

e. **Bosquejo** y **diseño asistido por computador:** En el diseño asistido por computador (CAD computer-aided design), los usuarios emplean la graficación interactiva para diseñar componentes y sistemas de dispositivos mecánicos, eléctricos y electromecánicos y electrónicos, incluyendo estructuras como edificios, carrocerías de automóviles, cascos de aviones y barcos, pastillas de circuitos integrados a escala muy grande (VLSI, very large scale integration) y redes telefónicas y de computadoras. Por lo general, el punto central de la atención es la interacción con un modelo basado en computador del componente o sistema que diseña. Pero en ocasiones el usuario sólo quiere producir dibujos precisos de los componentes y el montaje, como en el caso de bosquejos en linea o los planos arquitectónicos.

f. **Sistemas multimedia:** La graficación por computador tiene una función crucial en la rápidamente creciente área de los sistemas multimedia. Como su nombre lo indica, los multimedia comprenden más de un medio de comunicación. En estos sistemas se emplean convencionalmente texto, gráficos y sonido, pero pueden existir muchos mas.

g. **Simulación** y **animación para visualización científica** y **entretenimiento:** Los filmes de animación generados por computador y las presentaciones del comportamiento variable en el tiempo de objetos reales y simulados constituyen una herramienta cada vez más común para la visualización científica y del campo de la ingeniería. Podemos usar estos gráficos para estudiar entidades matemáticas abstractas o modelos matemáticos de fenómenos como el flujo de fluidos, la relatividad, las reacciones nucleares y químicas, la función de los órganos y el sistema fisiológico, y la deformación de estructuras mecánicas sujetas a diferentes tipos de carga. Otra área de la tecnología avanzada es la producción de efectos especiales de gran elegancia para filmes. Existen mecanismos complejos que permiten modelar los objetos y representar las sobras.

II. Hardware para graficación

1. Medios

Dispositivos de entrada y de salida

Teclados

Un teclado alfanumérico en un sistema de gráficas se utiliza sobre todo como un dispositivo para capturar cadenas de texto, es eficiente para capturar datos no gráficos. Las teclas de control de cursor y las de función
son características regulares en los teclados de uso común. Las teclas de función permiten que los usuarios capturen operaciones empleadas con frecuencia con un solo golpe de tecla y las teclas de control de cursor
pueden emplearse para seleccionar objetos desplegados o posiciones de coordenadas al poner en posición el cursor de la pantalla. Otros tipos de posicionamiento del cursor, como la bola palmar o la palanca de control, se incluyen en algunos teclados. A menudo se incluye un teclado numérico en el teclado para la captura rápida de datos numéricos.

Bola palmar y esfera de control

Como su nombre indica, la bola palmar es una bola que se puede hacer girar con los dedos o la palma de la mano, para producir movimiento en el cursor de la pantalla. Con potenciómetros que se conectan a la bola, se mide la cantidad y la dirección de la rotación. Este dispositivo es de posicionamiento bidimensional. Una esfera de control ofrece seis grados de libertad, y en realidad no se mueve.

Guante de datos

El guante está construido con una serie de sensores que detectan los movimientos de la mano y los dedos. Se emplea un acoplamiento electromagnético entre antenas transmisoras y antenas receptoras para
proporcionar información acerca de la posición y la orientación de la mano. Cada una de las antenas transmisoras y receptoras puede estar estructurada con un conjunto de tres retículas mutuamente perpendiculares que forman un sistema de coordenadas cartesianas. La entrada del guante de datos
puede emplearse para poner en posición y manipular objetos en una escena virtual.

Arquitecturas para graficación (aceleradores, *buses*, coprocesadores, etcétera)

El marco conceptual de alto nivel que se presenta en el siguiente diagrama nos puede servir para describir casi cualquier sistema gráfico interactivo. En el nivel de hardware (que no se presenta explícitamente en el diagrama), un computador recibe su entrada de dispositivos interactivos y envía las imágenes a dispositivo de presentación. El software tiene tres componentes. El primero, el programa de aplicación, crea, almacena y lee del segundo componente, el modelo de aplicación, que representa los datos u objetos que se mostrarán en la pantalla. El programa de aplicación también maneja las entradas de usuario. Este programa produce vistas enviando al tercer componente, el sistema gráfico, una serie de mandatos gráficos de salida que contienen una descripción geométrica de qué se verá y los atributos que indican cómo deben aparecer los objetos. El sistema gráfico es el responsable de la producción real de la imagen a partir de las descripciones detalladas, así como de pasar las entradas de usuario al programa de aplicación para que se procesen.

Así el sistema gráfico es un intermediario entre el programa de aplicación y el hardware de presentación, que lleva a cabo una transformación de salida de los objetos en el modelo de aplicación a una vista del modelo. Simétricamente, efectúa una transformación de entrada de las acciones del usuario a entradas para el programa de aplicación, por medio de las cuales la aplicación hará cambios al modelo o a la imagen. La tarea fundamental del diseñador de un programa de la graficación interactiva es especificar qué clases de elementos de datos u objetos se generarán y representarán pictóricamente y como interactuan el usuario y el programa de aplicación para crear y modificar el modelo y su representación visual. La mayor parte de las tareas del programador tiene que ver con la creación y edición del modelo, asi como con el manejo de la interacción con el usuario, no con la creación de las vistas, ya que esta tarea es realizada por el sistema gráfico.

III. Conceptos, técnicas y algoritmos

1. Herramientas

Librerías y paquetes para gráficas (tipos y características)

El primer paquete grafico se llama **SRGP** (simple raster graphics package, paquete gráfico simple de barrido) es independiente del dispositivo y explota las capacidades de las tramas. El repertorio de primitivas (líneas, rectángulos, círculos, elipses y cadenas de texto) de **SRGP** es similar al del popular paquete gráfico de trama Macintosh QuickOraw y al del paquete Xlib de Xwindow System. Por otra parte, las características de manejo de interacción de SRGP constituyen un subconjunto

de las SPHIGS, el paquete gráfico de mayor nivel que se usa para presentar primitivas tridimensionales, SPHIGS es un dialecto simplificado del paquete gráfico estándar PHIGS (programmer's hierarchical interactive graphics system, sistema gráfico interactivo jerarquico para programadores), diseñado tanto para hardware de barrido como vectorial.

El segundo paquete de graficación se llama PHIGS (programmer's hierarchical interactive graphics system, paquete grafico interactivo jerárquico para programadores) implanta una especificación diseñada como estándar por un organismo normativo oficial a nivel internacional o nacional.

2. Modelado

Software para modelado (R)

Alias Wavefront's Maya - Es quizá el software más popular en la industria, por lo menos hasta 2003. Es utilizado por muchos de los estudios de efectos visuales más importantes en combinación con Renderman, el motor de rénder fotorrealista de Pixar. Última versión a Septiembre de 2003: Maya 5.
Discreet's 3D Studio Max - Originalmente escrito por Kinetix (una división de Autodesk) como el sucesor de 3D Studio. Kinetix luego se fusionó con la última adquisición de Autodesk, Discreet Logic. La versión actual (a Febrero de 2003) es la 5.1. Es el líder en el desarrollo de 3D en la industria de juegos y usuarios hogareños.
Newtek's Lightwave 3D - Fue originalmente desarrollado por Amiga Computers a principios de la década de 1990. Más tarde evolucionó en un avanzado y muy usado paquete de gráficos y animación 3D. Actualmente disponible para Windows, Mac OS y Mac OS X. La versión actual es 7.5. El programa consiste en dos componentes: Modelador y Editor de escena. Es el favorito entre los entusiastas, y es utilizado en muchas de las mayores productoras de efectos visuales como Digital Domain.
Avid's Softimage XSI - El contrincante más grande de Maya. En 1987, Softimage Inc, una compañía situada en Montreal, escribió Softimage|3D, que se convirtió rápidamente en el programa de 3D más popular de ese período. En 1994, Microsoft compró Softimage Inc. y comenzaron a reescribir SoftImage|3D para Windows NT. El resultado se llamó Softimage|XSI. En 1998 Microsoft vendió Softimage a Avid. La versión actual (hasta mediados de 2003) es 3.5.

3. **Síntesis de imagen**

Software para síntesis

Como se ha podido apreciar, dicha actividad se circunscribe en un orden intelectual, técnico y manual, aspectos indivisibles a la hora de planificar sus trabajos.

Aunar criterios establecidos desde el ámbito cultural, para confrontarlo con los aportes tecnológicos y proyectarlos a un objetivo en común, serán los requisitos básicos, a fin de establecer un nuevo aporte en esta materia y delimitar sus alcances.

A su vez, es preciso tomar conciencia de que cada programa específico de diseño requiere de una formación integral, que ayudan a comprender la esencia misma de esta disciplina, con nuevos criterios que consolidan, aún más, su autonomía.

Es posible establecer una analogía con las principales relaciones entre las funciones operativas y un uso consensual del sistema.

PARATEXTO VIRTUAL	PARATEXTO GRÁFICO	VISIÓN CRÍTICA
FUENTES	Aplicación de estilo, tamaño y nombre de las fuentes, así como también los efectos.	Destacar títulos, palabras claves, organización discursiva de citas, etc.
BORDES Y SOMBREADOS	Permiten la utilización de cuadros sobre la base de estilos predefinidos.	Remarcar carátulas, separación de párrafos, encabezados, organización de tablas, etc.
VIÑETAS / NÚMEROS	Aplicación de fuentes especiales y símbolos.	Destacar elementos en una dispersión genérica, o bien, en una sucesión jerárquica.
COLUMNAS	Se establecen como espacios interdependientes (ancho y separación) para la lectura del texto.	Para el sistema occidental, un texto se lee desde la primer columna de arriba hacia abajo y se pasa a la siguiente hasta llegar al final.
TABLAS	Generan espacios independientes de identificación global.	El usuario establece un modo de organización discursiva independiente para su posterior aplicación de formatos.
TÍTULOS/ RÓTULOS	Genera una aplicación práctica de convenciones que permiten cambiar fuentes y posición de títulos básicos.	Los títulos se pueden identificar en un orden jerárquico (principal y accesorios)
ÍNDICE	Mediante la aplicación de títulos, se puede generar un índice automático.	Sugieren dos modelos analíticos de índice: el onomástico y el general.
DIBUJOS	Se pueden incluir dibujos en un texto de párrafo o de imagen.	Se sugieren los modos ClipArt establecidos como metaarchivos de Windows (.wmf).
IMÁGENES	Selección de imágenes de ClipArt y personales para su inclusión en el texto (formatos: JPG, BMP, etc.)	Una imagen tiene expresa significación con el contenido textual.
OTRAS FUNCIONES COMPLEMENTARIAS	Se establecen las funciones de ortografía, sinónimos, guardar, salir, copiar y pegar, dividir ventanas, entre otras.	Permiten al usuario una perfección de su actividad y el ahorro de tiempo (copiados intensos de párrafos, reproducción de imágenes, organizar una disposición de varios archivos, trabajar

		alternadamente, etc.)
CÓDIGO ASCII	Ubicación de la tecla Alt con una combinación numérica. Se utiliza en el caso de no hallar algún signo o letra en el teclado.	Incorporación de caracteres especiales: ñ, Ñ, ¿,¡, etcétera.
ENCABEZADOS Y **PIE DE PÁGINA**	Habilita un nuevo espacio de escritura (fuera del área de página).	Permite crear un texto repetitivo para otras páginas que organizan un contenido sistemático con funciones automáticas (rótulos, números de página, fecha y hora, etc.)
NÚMERO DE PÁGINA	Incluye en las páginas una sucesión de números en forma automática.	Disposición de los elementos, tanto en el encabezado como en el pie de página.
TEXTO DE PÁRRAFO	Organización de un texto base de acuerdo con una estructura convencional.	Incorporar guiones, sangrías, alineaciones, etc.
TEXTO ARTÍSTICO	Incorporar un texto base con relación a un efecto imagen	Cambios globales de disposición y de efectos. Ejemplo: WordArt.
LETRA CAPITAL	Ampliar la primera letra de un párrafo.	Cambio de fuente y su ubicación en el párrafo.
GUIONES	Distribución de funciones básicas y de efectos.	Galería de efectos Preset y Scripts.
COLORES Y **RELLENOS**	Aplicación de colores básicos y estructuras de rellenos interactivos.	Modos de enunciación disponibles: tramas, degradados y formas complejas de combinación.
EFECTOS COMUNES DE TEXTO	Escribir un texto de párrafo con modos de visión unívoca y alternada.	Se pueden establecer: negrita, subrayada, tachada, subíndice, superíndice, etcétera. Por ejemplo: m^2, H_2O, **texto**, entre otros.
EFECTOS ESPECIALES	Organización de elementos sobre la base de un determinado modo de visión.	Incorporar efectos en 2 y 3 dimensiones, de imagen, etc.

4. Técnicas de Animación

Software para animación

Algunas aplicaciones: típicas de la animación generada por computadora son el entretenimiento (películas y dibujos animados), publicidad, estudios científicos y de ingeniería, capacitación y educación. A pesar de que tendemos a considerar que la animación implica movimientos de objetos, el termino animación por computadora por lo regular se refiere a cualquier secuencia de tiempo de cambios visuales en una escena. Además de cambiar las posiciones de los objetos con traslaciones y rotaciones, una animación por computadora podria desplegar variaciones de tiempo en el tamaño, el color, la transparencia o la textura de la superficie de los objetos. Con frecuencia, las animaciones publicitarias realizan la transición de la forma de un objeto en otra.

Este planteamiento estándar para dibujos animados se aplica también en otras aplicaciones de la animación, aunque hay muchas aplicaciones especiales que no siguen esta secuencia. Las animaciones por computadora de tiempo real que producen los simuladores de vuelo, por ejemplo, despliegan secuencias de movimiento en respuesta a las especificaciones de los controles de la aeronave. Y aplicaciones de visualización se generan mediante las soluciones de los modelos numéricos. Para una animaci6ncuadro por cuadro, se genera y almacena por separado cada cuadro de la escena.1Luego, pueden grabarse los cuadros en la película desplegarse de manera consecutiva en modo de "pista de tiempo real".

El guión es una descripción de la acción. Define la secuencia de movimiento como un conjunto de eventos básicos que deben ocurrir. De acuerdo con el tipo de animación que se debe producir, el guión podría consistir en un conjunto de borradores o ser una lista de las ideas básicas para el movimiento.

Un cuadro clave es un diseño detallado de la escena en un momento determinado de la secuencia de animación. En cada cuadro clave se sitúa cada objeto de acuerdo con el tiempo para ese cuadro. Algunos cuadros clave se seleccionan en posiciones extremas en la acción; otras se espacian de modo que el intervalo de tiempo entre cuadros clave no sea muy extenso. Para movimientos complicados, se especifican mas cuadros clave que para movimientos simples con variaciones lentas.
Los *cuadros tntermedios* son los cuadros intermedios entre los cuadro clave.
El numero de cuadros intermedios que se necesita se determina por los medios que se van a utilizar para desplegar la animación. La película requiere 24 cuadros por segundo. Por lo regular, los intervalos de tiempo para el movimiento se establecen de modo que haya de tres a cinco cuadros de intermedio por cada par de cuadros clave

Animaciones de rastreo

En sistemas de rastreo, podemos generar animación de tiempo real en aplicaciones limitada al utilizar operaciones de rastreo. Como estudiamos un método simple para la traslación en el plano de xy es transferir un bloque rectangular de valores de píxel de una posición a otra. También es sencillo realizar rotulaciones bidimensionales en múltiplos de 90°, aunque podemos girar bloques rectangulares de píxeles a través de ángulos arbitrarios al emplear procedimientos de antilias.

LENGUAJES DE ANIMACION POR COMPUTADORA
El diseño y el control de las secuencias de animación se manejan con un conjunto de rutinas de animación. Con frecuencia, para programar las secuencias de animación se utiliza un lenguaje de uso común, como C, Lisp, Pascal o FORTRAN, pero se han desarrollado varios lenguajes de animación especializados. Las funciones de animación incluyen un editor de gráficas, un generador de cuadros clave, un generador de cuadros intermedios y rutinas gráficas estándar. El editor de gráficas nos permite diseñar y modificar las formas de los objetos al utilizar superficies de spline, métodos de geometría sólida constructiva u otros esquemas de representación

Una tarea común en la especificación de una animación es la descripción de la escena, la cual incluye el posicionamiento de objetos y fuentes de luz, define los parámetros fotométricos (intensidades de las fuentes de luz y propiedades de la iluminación de la superficie) y establece los parámetros de la cámara (posición, orientación y características del lente). Otra función estándar es la especificación de la acción. Esta función comprende la distribución de las trayectorias de movimiento para los objetos y la cámara. Se requieren las rutinas gráficas estándar, transformaciones de vista y de perspectiva, así como transformaciones geométricas, a fin de generar movimientos de los objetos como una función de especificaciones de aceleraciones o de trayectoria cinemática, identificación de superficies visibles y las operaciones de presentación de superficies.
Los sistemas de cuadro clave son lenguajes especializados para animación que se diseñan sólo para generar los cuadros intermedios a partir de los cuadros clave que el usuario especifica.

SISTEMAS DE CUADRO-CLAVE

Generamos cada conjunto de cuadros intermedios a partir de la especificación de dos (o más) cuadros clave. Las trayectorias del movimiento se pueden dar con una descripción cinemática como un conjunto de curvas spline o los movimientos pueden ser con base en las características físicas al especificar las fuerzas que actúan sobre los objetos que se deben animar.
Para escenas complejas, separamos los cuadros en componentes u objetos individuales que se llaman diapositivas de celuloide, un término para hacer referencia a la animación de dibujos. Dadas las trayectorias de la animación, podemos interpolar las posiciones de objetos individuales entre cualquiera de dos tiempos.
Con transformaciones de objetos complejas, las formas de los objetos pueden cambiar con el tiempo. Como ejemplos, podemos citarlas telas, rasgos faciales, detalles amplificados, formas en evolución, objetos que explotan o se desintegran y la transformación de un objeto en otro. Si todas las superficies se describen con enlaces de polígonos, entonces el número de aristas por polígono puede variar de un cuadro al siguiente. Por tanto, el número total de segmento de línea puede cambiar en cuadros diferentes.

Transformación (Morphing)
La conversión de las formas de los objetos de una a otra forma se denomina transformación (motphing), que es un término para expresar una metamorfosis. Se pueden aplicar métodos de transformación en cualquier movimiento o transición que implique un cambio de forma.
Al dar dos cuadros clave para la transformación de un objeto, primero ajustamos la especificación del objeto en uno de los cuadros, de modo que el número de aristas de polígono (o el número de vértices) sea igual para los dos cuadros

B. INTELIGENCIA ARTFICIAL

I. Representación del conocimiento.

1. Lógica

- **Proposicional (sintaxis y semántica)**

Definiciones de Inteligencia Artificial

Una buena definición de IA es algo elusiva y controversial, fundamentalmente porque la inteligencia humana no está completamente entendida. Cada libro de texto en IA propone una definición que enfatiza las diferentes perspectivas que, cada autor cree, encierra el campo. A continuación se transcriben algunas de ellas:

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

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

La IA es el estudio de las computaciones que permiten percibir, razonar y actuar.

La IA es un campo de estudio que busca explicar y emular el comportamiento inteligente en términos de procesos computacionales.

La IA estudia las representaciones y procedimientos que automáticamente resuelven problemas usualmente resueltos por humanos

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

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

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

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

Para usar la IA se requiere una comprensión básica de la forma en que se puede representar el conocimiento y de los métodos que pueden utilizar o manipular ese conocimiento.

La lógica proposicional es la más antigua y simple de las formas de lógica. Utilizando una representación primitiva del lenguaje, permite representar y manipular aserciones sobre el mundo que nos rodea. La lógica proposicional permite el razonamiento, a través de un mecanismo que primero evalúa sentencias simples y luego sentencias complejas, formadas mediante el uso de conectivos proposicionales, por ejemplo Y (AND), O (OR). Este mecanismo determina la veracidad de una sentencia compleja, analizando los valores de veracidad asignados a las sentencias simples que la conforman.

Una proposición es una sentencia simple que tiene un valor asociado ya sea de verdadero (V), o falso (F). Por ejemplo:

Hoy es Viernes

Ayer llovió

Hace frío

La lógica proposicional, permite la asignación de un valor verdadero o falso para la sentencia completa, no tiene facilidad par analizar las palabras individuales que componen la sentencia. Por este motivo, la representación de las sentencias del ejemplo, como proposiciones, sería:

hoy_es_Viernes

ayer_llovió

hace_frío

La proposiciones pueden combinarse para expresar conceptos más complejos. Por ejemplo:

hoy_es_Viernes y hace_frío.

A la proposición anterior dada como ejemplo, se la denomina **fórmula bien formada** (*well-formed formula*, **wff**). Una fórmula bien formada puede ser una proposición simple o compuesta que tiene sentido completo y cuyo valor de veracidad, puede ser determinado. La lógica proposicional proporciona un mecanismo para asignar valores de veracidad a la proposición compuesta, basado en los valores de veracidad de las proposiciones simples y en la naturaleza de los conectores lógicos involucrados.

Los conectadores básicos de la lógica proposicional, se dan en la Tabla 4.1. Las tablas de verdad para las operaciones básicas, se muestran en la Tabla 4.2.

NOMBRE	CONECTOR	SÍMBOLO
Conjunción	AND	^
Disyunción	OR	v
Negación	NOT	~
Implicación	If-Then	=>
Equivalencia	Igual	=

Tabla 4.1 *Conectores básicos de la lógica proposicional*

p	***q***	**Disyunción** ***p* v *q***	**Conjunción** ***p* ^ *q***	**Negación** ***~p***	**Implicación** ***p* => *q***	**Equivalencia** ***p* = *q***
V	V	V	V	F	V	V
V	F	V	F	F	F	F
F	V	V	F	V	V	F
F	F	F	F	V	V	V

Tabla 4.2 *Tablas de verdad para operadores lógicos*

El conectador de implicación, puede ser considerado como un condicional expresado de la siguiente forma:

Si A => B va a ser verdadero,

entonces toda vez que A sea verdadero, B debe ser siempre verdadero.

Para los casos en los cuales *A* es falso, la expresión *A => B*, es siempre verdadera, independientemente de los valores lógicos que tome *B*, ya que el operador de implicación no puede hacer inferencias acerca de los valores de *B*.

Existen varias equivalencias en lógica proposicional, similares a las del álgebra Booleana. Estas se dan en la Tabla 4.3.

DENOMINACIÓN	**REPRESENTACIÓN LÓGICA**
Leyes Equipotenciales	A => B = ~A v B A ^ ~A = F A v ~A = V
Leyes Conmutativas	A ^ B = B ^ A A v B = B v A
Leyes Distributivas	A ^ (B v C) = (A ^ B) v (A ^ C) A v (B ^ C) = (A v B) ^ (A v C)
Leyes Asociativas	A ^ (B ^ C) = (A ^ B) ^ C A v (B v C) = (A v B) v C
Leyes Absortivas	A ^ (A v B) = A A v (A ^ B) = A
Leyes de DeMorgan	~(A ^ B) = ~A v ~B ~(A v B) = ~A ^ ~B

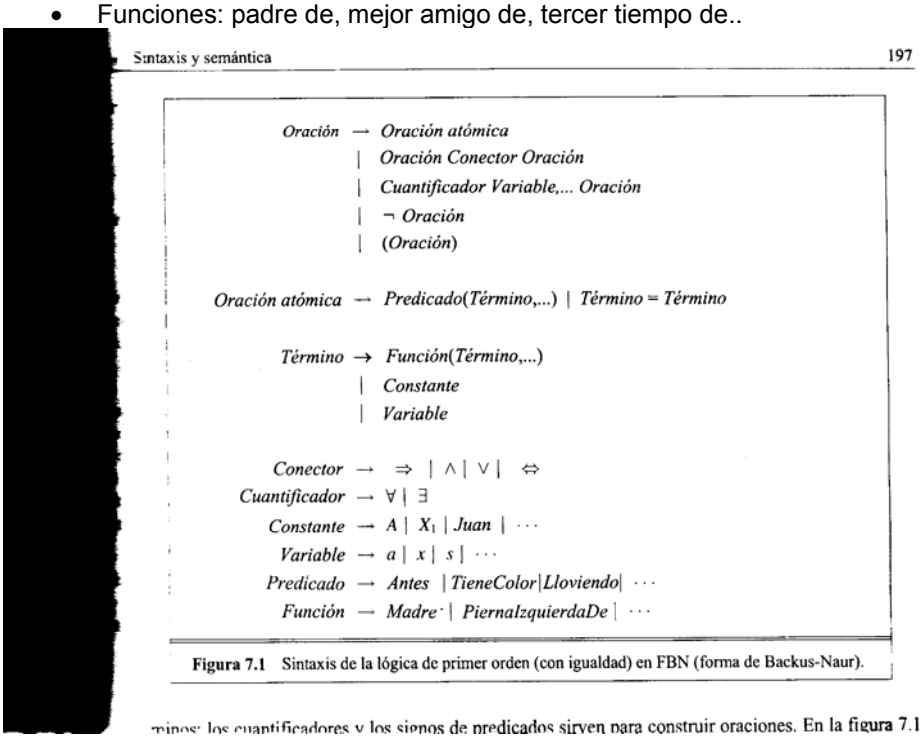
Tabla 4.3 *Equivalencias en lógica proposicional*

- **De primer orden (sintaxis y semántica)**

En donde se presenta una lógica suficiente para conseguir agentes que se basan en el conocimiento.

La logica de primer orden considera que el mundo esta constituido por objetos, es decir, entes con identidades individuales y propiedades que los distinguen de otros objetos. Entre estos objetos, existen diversos tipos de relaciones. Algunas de estas son las funciones: relaciones en que a una entrada corresponde un solo valor.

- Objetos : gente, casas, números, teorías, siglos...
- Relaciones: hermano de, mayor que, dentro de....
- Propiedades: rojo, redondo, de varios pisos.....
- Funciones: padre de, mejor amigo de, tercer tiempo de..



Sintaxis y semántica 197

Oración → *Oración atómica*
| *Oración Conector Oración*
| *Cuantificador Variable,... Oración*
| ¬ *Oración*
| (*Oración*)

Oración atómica → *Predicado*(*Término*,...) | *Término* = *Término*

Término → *Función*(*Término*,...)
| *Constante*
| *Variable*

Conector → ⇒ | ∧ | ∨ | ⇔
Cuantificador → ∀ | ∃
Constante → *A* | X_1 | *Juan* | ···
Variable → *a* | *x* | *s* | ···
Predicado → *Antes* | *TieneColor* | *Lloviendo* | ···
Función → *Madre* | *PiernaIzquierdaDe* | ···

Figura 7.1 Sintaxis de la lógica de primer orden (con igualdad) en FBN (forma de Backus-Naur).

minos: los cuantificadores y los signos de predicados sirven para construir oraciones. En la figura 7.1

• Como representación del conocimiento

Dado que el conocimiento es importante y primordial para el comportamiento inteligente, su representación constituye una de las máximas prioridades de la investigación en IA. El conocimiento puede ser representado como imágenes mentales en nuestros pensamientos, como palabras habladas o escritas en algún lenguaje, en forma gráfica o en imágenes, como cadenas de caracteres o colecciones de señales eléctricas o magnéticas dentro de un computador. En nuestro estudio de IA, consideraremos las representaciones escritas y sus correspondientes estructuras de datos utilizadas para su almacenamiento en un computador. La forma de representación que se escoja dependerá del tipo de problema a ser resuelto y de los métodos de inferencia disponibles.

Una representación del conocimiento puede ser un esquema o dispositivo utilizado para capturar los elementos esenciales del dominio de un problema. Una representación manipulable es aquella que facilita la computación. En representaciones manipulables, la información es accesible a otras entidades que usan la representación como parte de una computación.

Debido a la variedad de formas que el conocimiento puede asumir, los problemas involucrados en el desarrollo de una representación del conocimiento son complejos, interrelacionados y dependientes del objetivo. El términos generales, se debe tratar que el conocimiento esté representado de tal forma que:

- Capture generalizaciones.
- Pueda ser comprendido por todas las personas que vayan a proporcionarlo y procesarlo.
- Pueda ser fácilmente modificado.
- Pueda ser utilizado en diversas situaciones aún cuando no sea totalmente exacto o completo.
- Pueda ser utilizado para reducir el rango de posibilidades que usualmente debería considerarse para buscar soluciones.

El conocimiento declarativo puede ser representado con modelos relacionales y esquemas basados en lógica. Los modelos relacionales pueden representar el conocimiento en forma de árboles, grafos o redes semánticas. Los esquemas de representación lógica incluyen el uso de lógica proposicional y lógica de predicados.

Los modelos procedimentales y sus esquemas de representación almacenan conocimiento en la forma de cómo hacer las cosas. Pueden estar caracterizados por gramáticas formales, usualmente implantadas por sistemas o lenguajes procedimentales y sistemas basados en reglas (sistemas de producción).

Las representaciones declarativas son usualmente más expansivas y costosas, en el sentido que la enumeración puede ser redundante e ineficiente. Sin embargo, la modificación de las representaciones declarativas es usualmente muy fácil; simplemente se agrega o se elimina conocimiento. Las representaciones procedimentales, en cambio, pueden ser más compactas, sacrificando flexibilidad. Representaciones prácticas pueden incluir elementos tanto declarativos (listado de hechos conocidos), como procedimentales (un conjunto de reglas para manipular los hechos).

2. Reglas

• Representación

Los sistemas basados en reglas son los más comúnmente utilizados. Su simplicidad y similitud con el razonamiento humano, han contribuido para su popularidad en diferentes dominios. Las reglas son un importante paradigma de representación del conocimiento.

Las reglas representan el conocimiento utilizando un formato ***SI-ENTONCES*** (***IF-THEN***), es decir tienen 2 partes:

- La parte ***SI*** (***IF***), es el antecedente, premisa, condición o situación; y
- La parte ***ENTONCES*** (***THEN***), es el consecuente, conclusión, acción o respuesta.

Las reglas pueden ser utilizadas para expresar un amplio rango de asociaciones, por ejemplo:

> ***SI*** *está manejando un vehículo* ***Y*** *se aproxima una ambulancia,* ***ENTONCES*** *baje la velocidad* ***Y*** *hágase a un lado para permitir el paso de la ambulancia.*
>
> ***SI*** *su temperatura corporal es de 39 ºC,* ***ENTONCES*** *tiene fiebre.*
>
> ***SI*** *el drenaje del lavabo está tapado* ***Y*** *la llave de agua está abierta,* ***ENTONCES*** *se puede inundar el piso.*

Inferencia Basada en Reglas

Una declaración de que algo es verdadero o es un hecho conocido, es una **afirmación** (*fact*). El conjunto de afirmaciones se conoce a menudo con el nombre de **memoria de trabajo o base de afirmaciones**. De igual forma, al conjunto de reglas se lo denomina **base de reglas**.

Un sistema basado en reglas utiliza el *modus ponens* para manipular las afirmaciones y las reglas durante el proceso de inferencia. Mediante técnicas de búsqueda y procesos de unificación, los sistemas basados en reglas automatizan sus métodos de razonamiento y proporcionan una progresión lógica desde los datos iniciales, hasta las conclusiones deseadas. Esta progresión hace que se vayan conociendo nuevos hechos o descubriendo nuevas afirmaciones, a medida que va guiando hacia la solución del problema.

En consecuencia, el proceso de solución de un problema en los sistemas basados en reglas va realizando una serie de inferencias que crean un sendero entre la definición del problema y su solución. Las inferencias están concatenadas y se las realiza en forma progresiva, por lo que se lo que se dice que el proceso de solución origina una **cadena de inferencias**.

Los sistemas basados en reglas difieren de la representación basada en lógica en las siguientes características principales:

- Son en general no-monotónicos, es decir hechos o afirmaciones derivadas, pueden ser retractados, en el momento en que dejen de ser verdaderos.

Pueden aceptar incertidumbre en el proceso de razonamiento.

• Encadenamiento hacia delante y hacia atrás

El Proceso de Razonamiento

El proceso de razonamiento en un sistema basado en reglas es una progresión desde un conjunto inicial de afirmaciones y reglas hacia una solución, respuesta o conclusión. Como se llega a obtener el resultado, sin embargo, puede variar significativamente:

- Se puede partir considerando todos los datos conocidos y luego ir progresivamente avanzando hacia la solución. Este proceso se lo denomina *guiado por los datos* o de **encadenamiento progresivo** (*forward chainning***)**.
- Se puede seleccionar una posible solución y tratar de probar su validez buscando evidencia que la apoye. Este proceso se denomina *guiado por el objetivo* o de **encadenamiento regresivo** (*backward chainning*).

Razonamiento Progresivo

En el caso del razonamiento progresivo, se empieza a partir de un conjunto de datos colectados a través de observación y se evoluciona hacia una conclusión. Se chequea cada una de las reglas para ver si los datos observados satisfacen las premisas de alguna de las reglas. Si una regla es satisfecha, es ejecutada derivando nuevos hechos que pueden ser utilizados por otras reglas para derivar hechos adicionales. Este proceso de chequear reglas para ver si pueden ser satisfechas se denomina *interpretación de reglas*.

La interpretación de reglas es realizada por una máquina de inferencia en un sistema basado en conocimiento. La interpretación de reglas, o inferencia, en el razonamiento progresivo involucra la repetición de los pasos que se indican en la siguiente figura.

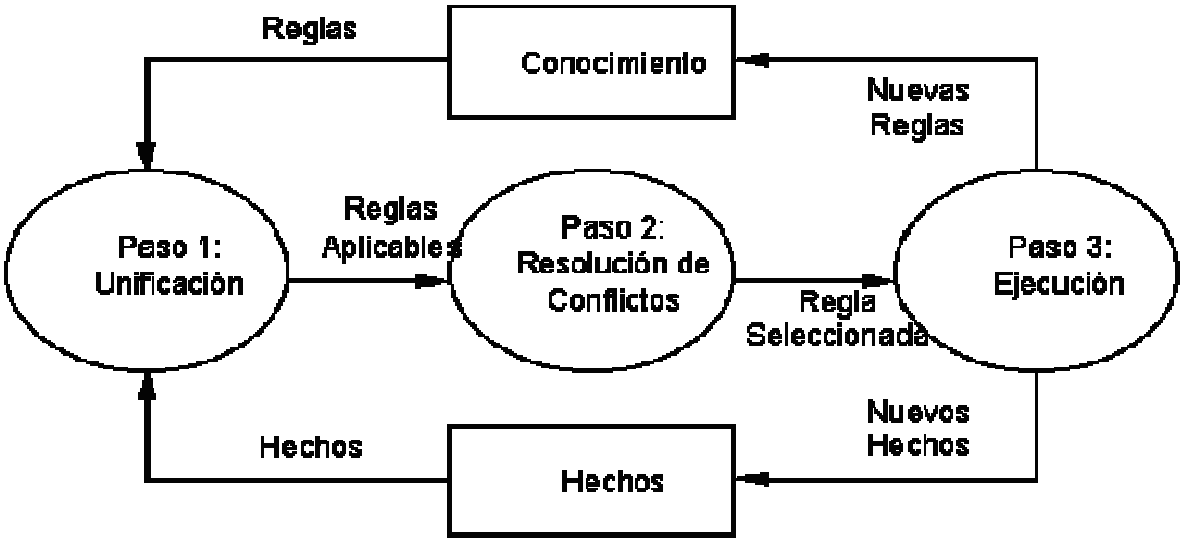


Figura 4.1 *Proceso de Razonamiento Progresivo*

1. **Unificación (*Matching*).-** En este paso, en las reglas en la base de conocimientos se prueban los hechos conocidos al momento para ver cuáles son las que resulten satisfechas. Para decir que una regla ha sido satisfecha, se requiere que todas las premisas o antecedentes de la regla resuelvan a verdadero.
2. **Resolución de Conflictos.-** Es posible que en la fase de unificación resulten satisfechas varias reglas. La resolución de conflictos involucra la selección de la regla que tenga la más alta prioridad de entre el conjunto de reglas que han sido satisfechas.
3. **Ejecución.-** El último paso en la interpretación de reglas es la ejecución de la regla. La ejecución puede dar lugar a uno o dos resultados posibles: nuevo hecho (o hechos) pueden ser derivados y añadidos a la base de hechos, o una nueva regla (o reglas) pueden ser añadidas al conjunto de reglas (base de conocimiento) que el sistema considera para ejecución.

En esta forma, la ejecución de las reglas procede de una manera progresiva (hacia adelante) hacia los objetivos finales.

Un conjunto de aplicaciones adecuadas al razonamiento progresivo incluye supervisión y diagnóstico en sistemas de control de procesos en tiempo real, donde los datos están continuamente siendo adquiridos, modificados y actualizados. Estas aplicaciones tienen 2 importantes características:

1. Necesidad de respuesta rápida a los cambios en los datos de entrada.
2. Existencia de pocas relaciones predeterminadas entre los datos de entrada y las conclusiones derivadas.

Otro conjunto de aplicaciones adecuadas para el razonamiento progresivo está formado por: diseño, planeamiento y calendarización, donde ocurre la síntesis de nuevos hechos basados en las conclusiones de las reglas. En estas aplicaciones hay potencialmente muchas soluciones que pueden ser derivadas de los datos de entrada. Debido a que estas soluciones no pueden ser enumeradas, las reglas expresan conocimiento como patrones generales y las conexiones precisas entre estas reglas no pueden ser predeterminadas.

Razonamiento Regresivo

El mecanismo de inferencia, o interprete de reglas para el razonamiento regresivo, difiere significativamente del mecanismo de razonamiento progresivo. Si bien es cierto ambos procesos involucran el examen y aplicación de reglas, el razonamiento regresivo empieza con la conclusión deseada y decide si los hechos que existen pueden dar lugar a la obtención de un valor para esta conclusión. El razonamiento regresivo sigue un proceso muy similar a la búsqueda primero en profundidad.

El sistema empieza con un conjunto de hechos conocidos que típicamente está vacío. Se proporciona una lista ordenada de objetivos (o conclusiones), para las cuales el sistema trata de derivar valores. El proceso de razonamiento regresivo utiliza esta lista de objetivos para coordinar su búsqueda a través de las reglas de la base de conocimientos. Esta búsqueda consiste de los siguientes pasos:

1. Conformar una pila inicialmente compuesta por todos los objetivos prioritarios definidos en el sistema.
2. Considerar el primer objetivo de la pila. Determinar todas las reglas capaces de satisfacer este objetivo, es decir aquellas que mencionen al objetivo en su conclusión.
3. Para cada una de estas reglas examinar en turno sus antecedendentes:
    a. Si todos los antecedentes de la regla son satisfechos (esto es, cada parámetro de la premisa tiene su valor especificado dentro de la base de datos), entonces ejecutar esta regla para derivar sus conclusiones. Debido a que se ha asignado un valor al objetivo actual, removerlo de la pila y retornar al paso (2).
    b. Si alguna premisa de la regla no puede ser satisfecha, buscar reglas que permitan derivar el valor especificado para el parámetro utilizado en esta premisa.
    c. Si en el paso (b) no se puede encontrar una regla para derivar el valor especificado para el parámetro actual, entonces preguntar al usuario por dicho valor y añadirlo a la base de datos. Si este valor satisface la premisa actual entonces continuar con la siguiente premisa de la regla. Si la premisa no es satisfecha, considerar la siguiente regla.

Si todas las reglas que pueden satisfacer el objetivo actual se han probado y todas no han podido derivar un valor, entonces este objetivo quedará indeterminado. Removerlo de la pila y retornar al paso (2). Si la pila está vacía parar y anunciar que se ha terminado el proceso.

El razonamiento regresivo es mucho más adecuado para aplicaciones que tienen mucho mayor número de entradas, que de soluciones posibles. La habilidad de la lógica regresiva para trazar desde las pocas conclusiones hacia las múltiples entradas la hace más eficiente que el encadenamiento progresivo.

Una excelente aplicación para el razonamiento regresivo es el diagnóstico, donde el usuario dialoga directamente con el sistema basado en conocimiento y proporciona los datos a través del teclado. Problemas de clasificación también son adecuados para ser resuelto mediante el razonamiento regresivo.

- ## Resolución de conflictos

El resultado del proceso de emparejamiento es una lista cuyos antecedentes han emparejado la descripción del estado actual con cualesquiera que sean los enlaces de variables que se generaron en el proceso de emparejamiento. El metodo de búsqueda debe decidir el orden en que se aplicarán las reglas. Pero a veces resulta útil incorporar alguna parte de esta decisión en el proceso de emparejamiento. Esta parte del proceso de emparejamiento se denomina resolución de conflictos

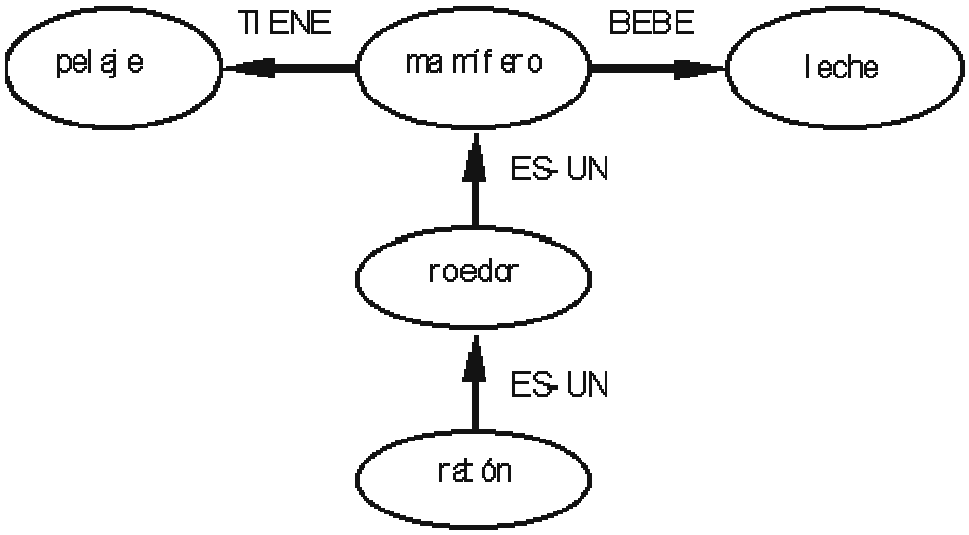
Existen tres enfoques básicos al problema de resolución de conflictos en un sistema de producción:

- Asignar una preferencia basada en la regla que se emparejó
- Asignar una preferencia basada en los objetos que se emparejaron
- Asignar una preferencia basada en la acción que llevará a cabo la regla emparejada.

3.Redes semánticas, frames y scripts

Las ***redes semánticas***, fueron originalmente desarrolladas para representar el significado o semántica de oraciones en inglés, en términos de objetos y relaciones. Actualmente, el término ***redes asociativas*** es más ampliamente utilizado para describirlas ya que no sólo se las usa para representar relaciones semánticas, sino también para representar asociaciones físicas o causales entre varios conceptos u objetos.

Las redes asociativas se caracterizan por representar el conocimiento en forma gráfica. Agrupan una porción de conocimiento en dos partes: objetos y relaciones entre objetos. Los objetos se denominan también nodos (elementos del conocimiento) y las relaciones entre nodos se denominan enlaces o arcos. Cada nodo y cada enlace en una red semántica, deben estar asociados con objetos descriptivos.



Fragmento de una red asociativa.

Son muy apropiadas para representar conocimiento de naturaleza jerárquica. Su concepción se basa en la asociación de conocimientos que realiza la memoria humana. Las principales aplicaciones son: comprensión de lenguaje natural, bases de datos deductivas, visión por computadora, sistemas de aprendizaje.

Ventajas de las Redes Asociativas

Las redes asociativas tienen dos ventajas sobre los sistemas basados en reglas y sobre los basados en lógica:

1. Permiten la declaración de importantes asociaciones, en forma explícita y sucinta.
2. Debido a que los nodos relacionados están directamente conectados, y no se expresan las relaciones en una gran base de datos, el tiempo que toma el proceso de búsqueda por hechos particulares puede ser significativamente reducido.

Desventajas de las Redes Asociativas

Entre las desventajas de las redes asociativas, se pueden mencionar:

1. No existe una interpretación normalizada para el conocimiento expresado por la red. La interpretación de la red depende exclusivamente de los programas que manipulan la misma.
2. La dificultad de interpretación a menudo puede derivar en inferencias inválidas del conocimiento contenido en la red.
3. La exploración de una red asociativa puede derivar en una explosión combinatoria del número de relaciones que deben ser examinadas para comprobar una relación

Representación mediante Plantillas (frames)

Una plantilla (*frame*) es una estructura de datos apropiada para representar una situación estereotípica. Las plantillas organizan el conocimiento en objetos y eventos que resultan apropiados para situaciones específicas. Evidencia psicológica sugiere que la gente utiliza grandes plantillas para codificar el conocimiento de experiencias pasadas, o conocimiento acerca de cosas que se encuentran comúnmente, para analizar y explicar una situación nueva en su cotidiana actividad cognoscitiva.

Una *plantilla* representa un objeto o situación describiendo la colección de atributos que posee. Están formadas por un nombre y por una serie de campos de información o *ranuras* (*slots*). Cada *ranura* puede contener uno o más *enlaces* (facets). Cada *enlace* tiene un valor asociado. Varios *enlaces* pueden ser definidos para cada *ranura*, por ejemplo:

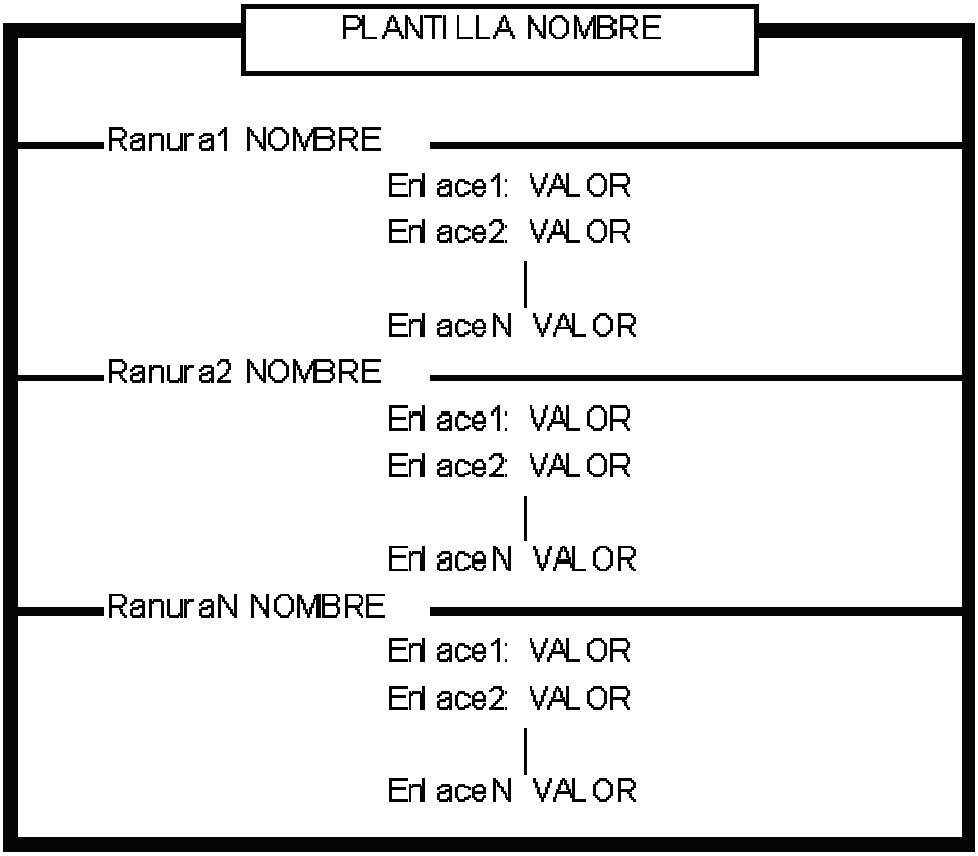
- *Rango*.- El conjunto de posibles valores para la ranura.
- *Valor*.- El valor de la ranura.
- *Default*.- El valor a ser asumido si no se especifica alguno.

Además los *enlaces* pueden ser procedimientos que residen en la base de datos y están aguardando para ser utilizados cuando se los necesite. Entre los más comunes se pueden mencionar:

- *Si-Necesitado*.- Procedimiento(s) para determinar el valor actual de una ranura.
- *Si-Agregado*.- Procedimiento(s) a ejecutarse cuando un valor es especificado para una ranura.
- *Si-Modificado*.- Procedimiento(s) a ejecutarse si el valor de una ranura es cambiado.

A estos procedimientos también se los denomina *demons* y representan un concepto poderoso en las plantillas, esto es, la habilidad de combinar conocimiento procedimental dentro de la estructura de conocimiento declarativo de la plantilla. Esto sugiere que una plantilla puede ser un medio poderoso de representación del conocimiento, especialmente si se la incorpora en una red de plantillas.

En la siguiente figura se muestra una representación abstracta de una plantilla.



Representación abstracta de una plantilla.

Se pueden establecer ciertas similitudes entre un sistema basado en plantillas y un sistema de bases de datos. Aparentemente los dos representan "datos" (a través de las ranuras de una plantilla y de los campos de una tabla de datos), sin embargo las plantillas representan en realidad conocimiento, mientras que las bases de datos representan sólo datos. La investigación que se realiza actualmente en bases de datos está examinando la posibilidad de aplicarlas a la representación del conocimiento, incorporando herencia y *demons* (***Bases de Datos Inteligentes***), similar a lo que se tiene en sistemas basados en conocimiento.

Ventajas de las Plantillas

Las ventajas que se pueden establecer para los sistemas basados en plantillas son las siguientes:

1. Facilidad de proceso guiado por las expectativas. Un sistema basado en plantillas, mediante los *demons* es capaz de especificar acciones que deben tener lugar cuando ciertas condiciones se han cumplido durante el procesamiento de la información.
2. El conocimiento que posee un sistema basado en plantillas es significativamente más estructurado y organizado que el conocimiento dentro de una red asociativa.
3. Las plantillas pueden ser estructuradas de tal forma que sean capaces de determinar su propia aplicabilidad en determinadas situaciones. En el caso de que una plantilla en particular no sea aplicable, puede sugerir otras plantillas que pueden ser apropiadas para la situación.
4. Se puede fácilmente almacenar en las ranuras valores dinámicos de variables, durante la ejecución de un sistema basado en conocimiento. Esto puede ser particularmente útil para

aplicaciones de simulación, planeamiento, diagnóstico de problemas o interfaces para bases de datos.

Desventajas de las Plantillas

Las principales desventajas que se pueden establecer para la representación del conocimiento mediante plantillas, son:

1. Dificultad de representar objetos que se alejen considerablemente de estereotipos.
2. No tiene la posibilidad de acomodarse a situaciones u objetos nuevos.
3. Dificultad para describir conocimiento heurístico que es mucho más fácilmente representado mediante reglas.

Representación mediante Objetos (scripts)

Los objetos, son similares a las plantillas. Ambos sirven para agrupar conocimiento asociado, soportan herencia, abstracción y el concepto de procedimientos agregados. La diferencia radica en los siguiente:

1. En las plantillas, a los programas y a los datos se los trata como dos entidades relacionadas separadas. En cambio en los objetos se crea una fuerte unidad entre los procedimientos (*métodos*) y los datos.
2. Los *demons* de las plantillas sirven sólo para computar valores para las diversas ranuras o para mantener la integridad de la base de conocimientos cada vez que una acción de alguna plantilla, afecta a otra. En cambio, los *métodos* utilizados por los objetos son más universales ya que proporcionan cualquier tipo general de computación requerida y además soportan encapsulamiento y polimorfismo.

Un objeto es definido como una colección de información representando una entidad del mundo real y una descripción de cómo debe ser manipulada esta información, esto es los *métodos*. Es decir, un objeto tiene un nombre, una caracterización de clase, varios atributos distintivos y un conjunto de operaciones. La relación entre los objetos viene definida por los *mensajes*. Cuando un objeto recibe un *mensaje* válido, responde con una acción apropiada, retornando un resultado.

NOMBRE OBJETO	Limpiador Izquierdo
UN-TIPO-DE	Limpiador
ATRIBUTOS	Hecho de metal y caucho Longitud: 14 pulgadas Color: negro y plateado Localización: inferior izquierda Función: remover humedad de parabrisa
OPERACIONES	Activado: se mueve en arco sobre el parabrisa repetidamente de izquierda a derecha Desactivado: se mueve a posición de reposo

Descripción tipo objeto para un limpia parabrisas.

Ventajas de la Representación mediante Objetos

Los objetos, como forma de representación del conocimiento ofrece las siguientes ventajas:

1. Poder de abstracción.
2. Encapsulamiento o capacidad de esconder información.
3. Herencia, es decir pueden recibir características o propiedades de sus ancestros.
4. Polimorfismo, que permite crear una interfaz común para todos los diversos objetos utilizados dentro del dominio.
5. Posibilidad de reutilización del código.
6. Mayor facilidad para poder trabajar eficientemente con sistemas grandes.

Desventajas de la Representación mediante Objetos

Las desventajas son similares a las que se indicaron para las plantillas:

1. Dificultades para manejar objetos que se alejan demasiado de la norma.
2. Dificultades para manejar situaciones que han sido encontradas previamente

- **Representación**

En general, una *representación* es un conjunto de convenciones sobre la forma de describir algún tipo de cosa. El hallar una representación apropiada es una parte fundamental de la resolución de un problema.

El principio de la representación establece que:

> ***Una vez que un problema es descrito mediante una buena representación, el problema está casi resuelto*.**

Descripciones Explícitas

La descripción explícita de una buena representación está caracterizada por los siguientes aspectos importantes:

- Hace explícitos los objetos y las relaciones de importancia: de una sola mirada se puede apreciar lo que sucede.
- Pone de manifiesto las restricciones inherentes al problema.
- Agrupa los objetos y las relaciones.
- Suprime los detalles insignificantes.
- Es transparente: se puede entender lo que representa.
- Es completa: contiene todo lo que es necesario que debe expresar.
- Es concisa: expresa todo lo necesario con eficacia.

Las representaciones tienen cuatro ingredientes fundamentales:

- El **léxico**, que determina los símbolos que están permitidos en el vocabulario de la representación.
- Una parte **estructural** que describe las restricciones sobre la forma en que los símbolos pueden ordenarse.

- Una parte **operativa** que especifica los procedimientos de acceso que permiten crear descripciones; así como la forma de modificarlas y utilizarlas para responder preguntas.
- Una parte **semántica** que establece una forma de asociar el significado con las descripciones.

Para ilustrar el problema de la representación, considérese el siguiente ejemplo:

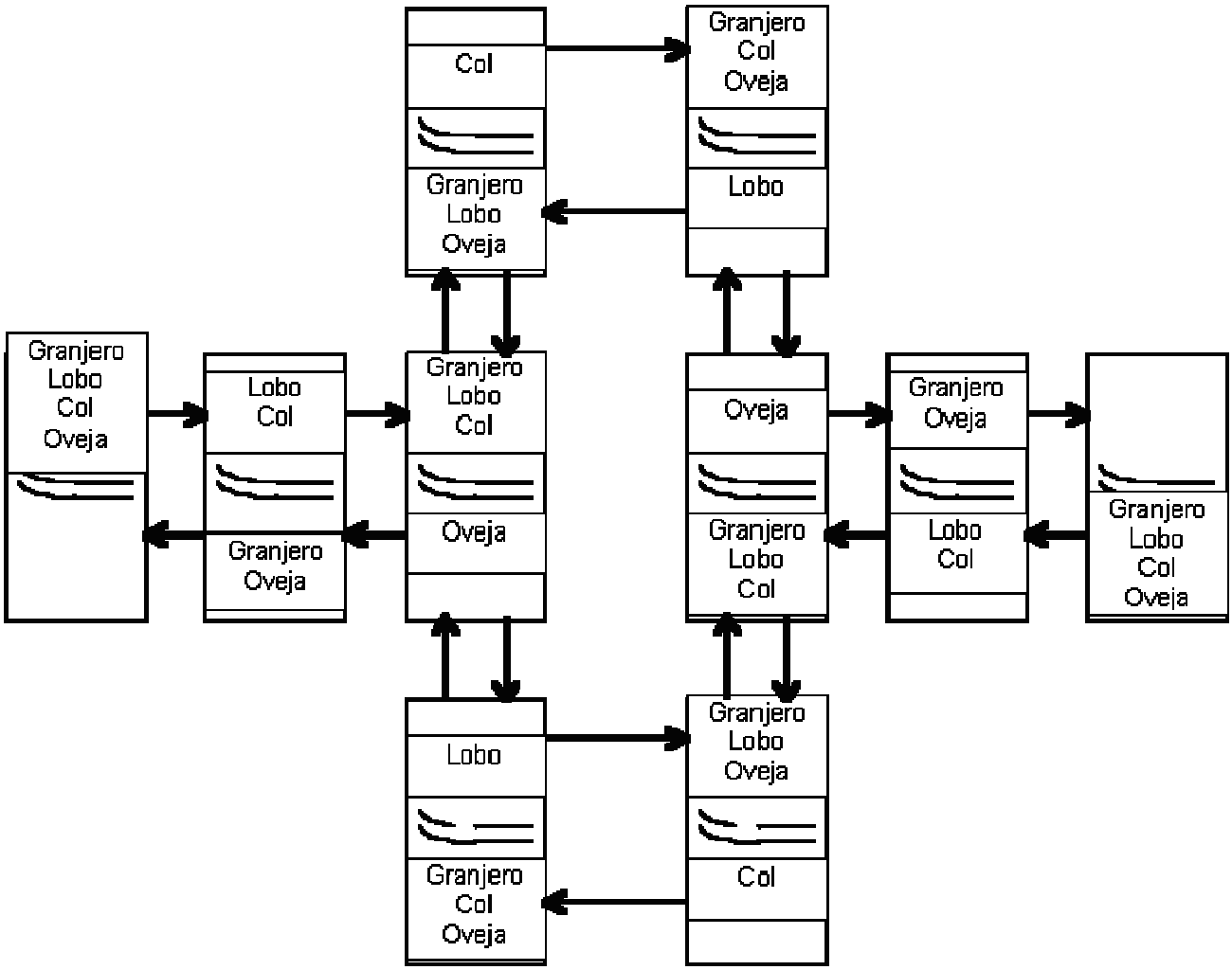
> *Un granjero desea cruzar un río llevando consigo un lobo silvestre, una oveja y una carga de col. Por desgracia su bote es tan pequeño que sólo puede transportar una de sus pertenencias en cada viaje. Peor aún, si no vigila al lobo, puede comerse a la oveja y si no cuida la col, puede comerse la oveja; de modo que no puede dejar al lobo solo con la oveja, ni a la oveja sola con la col. ¿ Cómo puede hacer para cruzar el río sin contratiempos?*

La representación utilizada en el problema del granjero, es un ejemplo de ***red semántica***. Desde el punto de vista del **léxico**, las *redes semánticas* están formadas por **nodos**, que representan objetos; **enlaces**, que representan relaciones entre objetos; y, **etiquetas** de enlace, que denotan relaciones particulares.

Desde el punto de vista **estructural**, los nodos están conectados entre sí por enlaces etiquetados. En los diagramas, los nodos aparecen como círculos, elipses o rectángulos; y, los enlaces como flechas que apuntan de un nodo a otro.

Desde la perspectiva de la **semántica**, el significado de los nodos y sus enlaces, depende de la aplicación.

Desde la perspectiva operativa, los procedimientos de acceso son, en general, cualquiera de los siguientes: constructores, destructores, lectores y escritores. Las redes semánticas usan constructores para formar nodos y enlaces; lectores para responder preguntas acerca de estos; escritores, para modificar nodos y enlaces; y, ocasionalmente, destructores, para borrar nodos y enlaces.



Representación del problema del granjero, el lobo, la oveja y la col.

Las especificaciones dadas por una red semántica, son los suficientemente explícitas, como para ser utilizadas en representaciones de problemas de IA.

Existen diversos grupos de familias de representaciones, que se derivan de las redes semánticas: árboles semánticos, árboles de búsqueda, árboles de juegos, árboles de decisión, sistema de plantillas, red de propagación de valores, red de restricción aritmética, red de contracción, red de intervalos, espacio de estados, etc.

Una representación basada en una red semántica, en determinados problemas, puede generar *árboles* o *grafos*. El *árbol* es un tipo de red semántica que no contiene lazos, en cambio un *grafo* es una estructura en la que pueden haber varios de ellos.

Desde el punto de vista de los procesos de búsqueda, los árboles son preferibles a los grafos, ya que no hay que preocuparse por la posibilidad de quedar atrapados en algún lazo.

Las técnicas de solución de problemas en IA, en general, incorporan un proceso de búsqueda. Todo proceso de búsqueda puede ser visualizado como el recorrido por un árbol en el que cada

nodo representa un estado y cada rama representa las relaciones entre los estados cuyos nodos conecta.

En general, las reglas contienen en forma implícita el árbol, y se genera en forma explícita sólo aquellas partes que se decide explorar. Las principales diferencias que pueden aparecer en las diferentes técnicas de búsqueda, son:

- La dirección en la cual se conduce la búsqueda (hacia adelante o hacia atrás).
- La estrategia de control, o forma de seleccionar las reglas que pueden ser aplicables. Los principales requerimientos de una buena estrategia de control son: que cause desplazamiento en el espacio de estado; y, que sea sistemático.
- La forma de representar cada nodo del proceso de búsqueda (representación del conocimiento).

Muchas veces, tratar el proceso como búsqueda en un grafo en lugar de una búsqueda en un árbol, puede reducir el esfuerzo que se gasta en explorar senderos, esencialmente iguales, varias veces. Sin embargo, los requisitos asociados, son:

- Cada vez que se genere un nodo se debe chequear para ver si ha sido generado antes.
- Se deben introducir procedimientos especiales para que la búsqueda no quede atrapada en algún lazo.

4.inteligencia artificial distribuida

Sistemas de razonamiento distribuido

Hasta ahora, en todas las explicaciones sobre sistemas de resolución de problemas hemos centrado en el diseño de sistemas simples. En este apartado se inteneta llegar más allá viendo los *sistemas de razonamiento distribuidos.* Se definen como sistemas compuestos por un conjunto de módulos separados (llamados a menudo agentes, ya que se espera que cada módulo actúe como una entidad resolutora de problemas por derecho propio) y un conjunto de caminos de comunicación entre ellos.
Esta definición, da una idea muy vaga de lo que son. Admite sistemas largo de un espectro que va desde sistemas estrechamente emparejados, en los el mecanismo de control está completamente centralizado y la base de conocimientos está compartida, hasta aquellos en los que el control y el conocimiento está totalmente distribuido. De hecho y como es lógico, la mayoría de los sistemas de razonamiento distribuido reales se sitúan en algún lugar intermedio. Esta definición también incluye sistemas que están distribuidos en diversos niveles de granularidad (como los sistemas conexionistas en los que los nodos individuales no desarrollan el razonamiento en el mismo sentido en que se ha venido utilizando el término).

Para la mayoría de los diferentes tipos de aplicaciones, los sistemas de razonamiento distribuido presentan ventajas importantes respecto a los grandes sistemas monolíticos. Dichas ventajas incluyen:

Modularidad del Sistema: Es más fácil construir y mantener un conjunto de módulos cuasi independientes, que uno enorme.
Eficiencia: No todo el conocimiento es utilizado para todas las tareas. modularizándolos, se enfocan los esfuerzos de los sistemas resolutores de proti por caminos que resultan más adecuados.
Arquitecturas de computación rápida: A medida que los resolutores de problemas se hacen más complejos, también necesitan ciclos mas complejos. Aunque las máquinas continúen haciéndose mas rápidas, los verdaderos aumentos de velocidad empiezan a no ser debidos a un procesador simple con una amplia memoria asociada, sino a un grupo de procesadores más pequeños, cada de los cuales tiene su propia memoria. Los sistemas distribuidos estan mas capacitados para

utilizar dichas arquitecturas.
Razonamiento heterogéneo: Las técnicas de resolución de problemas y los formalismos para la representación de conocimientos que, por un lado, puede no resultar mejores para trabajar con una parte de un problema, pueden no resultar adecuados para trabajar con otra parte del problema.
Perspectivas múltiples: El conocimiento requerido para resolver un determinado problema puede no residir en la cabeza de un única persona. Resulta muy dificil conseguir que varias personas construyan una única base de conocimiento coherente y, en ocasiones, incluso resulta imposible ya que sus modelos del dominio resultan, de hecho, inconsistentes.

Problemas distribuidos: Algunos problemas son distribuidos por naturaleza. Por ejemplo, puede existir diferente información disponible en cada uno, en diferentes localizaciones físicas.

Confiabilidad: Si un problema se distribuye a través de agentes de diferentes sistemas, la resolución del problema puede continuar, incluso si uno de los sistemas fallase.

Una arquitectura para razonamiento distribuido debe proporcionar:

1. Un mecanismo para asegurar que las actividades de los diferentes agentes del sistema estan coordinados de manera que el sistema global resolutor de problemas alcance su(s) objetivo(s).
2. Una estructura de comunicación que permita el paso de información en ambas direcciones, entre los agentes.
3. Versiones distribuidas de las técnicas de razonamiento necesarias. Estos mecanismos deben diferir de sus duplicados monolíticos, ya que se supone que deben operar con un conjunto de bases de conocimiento locales antes que con un conocimiento global, que es de esperar que posea un conjunto de propiedades globales (como la consistencia).

- **Arquitecturas/esquemas de control (de central hasta distribuido)**

La definición de un sistema de razonamiento distribuido es aquel compuesto de un juego de modulos separados (usualmente llamados agentes desde que se espera que cada modulo actue como una entidad que resuelve problemas por su propio derecho) y un juego de caminos de comunicación entre ellos. Esta definición es intencionalmente vaga, la cual admite que los sistemas dondequiera a lo largo de un espectro que los ranquea desde sistemas ajustados en los cuales existe un mecanismo de control centralizado y una base de conocimiento compartida para aquellos en los cuales el control y el conocimiento estan completamente distribuidos, de hecho, por supuesto, la mayoria de los sistemas de razonamiento distribuidos se encuentran en algun lugar intermedio. Esta definición tambien incluye a los sistemas que estan distribuidos en distintos niveles de granularidad, aunque no se intenta incluir sistemas con muy fina granularidad (tales como sistemas de conección en los cuales los nodos individuales no realizan razonamiento en el mismo sentido en que se a utilizado este termino).

Para muchos tipos de aplicaciones, los sistemas de razonamiento distribuido tienen muchas ventajas sobre los sistemas monolíticos, estas ventajas pueden incluir:

1. **Modularidad del sistema:** Es mas facil construir y mantener una colección de modulos independientes que uno grande.
2. **Eficiencia:** No todo el conocimiento es necesario para realizar todas las tareas, si lo modularizamos, ganamos la habilidad de enfocar los esfuerzos del sistema en la resolución del problema de maneras que pueden resultar mas útiles
3. **Rápida arquitectura de computadoras:** Como resolvedoras del problema se pone mas complejo, ellas necesitan mas y mas ciclos, aunque las maquinas continúan haciéndose mas rápidas, los verdaderos aceleramientos provienen no de un simple procesador con una gran memoria asociada, pero para los clusters de los procesadores pequeños, cada

uno con su propia memoria, los sistemas distribuidos tienen mejores posibilidades de explotar ese ti 00 de arquitectura

4. **Razonamiento heterogéneo:** Las técnicas de solución de problemas y los formalismos de la representación del conocimiento que son mejores para trabajar en una parte del problema puede no ser la mejor manera de trabajar en alguna otra parte.
5. **Múltiples perspectivas:** El conocimiento requerido para solucionar un problema puede no residir en la cabeza de una sola persona. Es muy difícil hacer que mucha gente construya una sola coherente base de conocimiento y algunas veces es imposible por que sus modelos de dominio son inconsistentes.
6. **Problemas distribuidos:** Algunos problemas están inherentemente distribuidos, por ejemplo, puede haber diferentes datos disponibles en todas las diferentes distribuciones físicas.
7. **Contiabilidad:** Si un problema es distribuido entre agentes en diferentes sistemas, la solución de problemas puede continuar incluso si un sistema falla.

Una arquitectura para el razonamiento distribuido puede proporcionar:

1. Un mecanismo para asegurar que las actividades de varios agentes en el sistema están coordinadas así que el sistema de resolución de problemas logre su meta de manera global.

2. Una estructura de comunicación que permita que la información circule hacia delante y hacia atrás entre los agentes o módulos.

3. Versiones distribuidas de las técnicas de razonamiento necesarias. Estos mecanismos es probable que difieran de sus contrapartes monolíticas ya que ellos presumiblemente operan en un juego de bases del conocimiento locales en vez de una global una que pueda presumirse que posee un juego de propiedades global (así como consistencia).

- **Arquitectura de pizarrón (R)**

La idea detrás del acercamiento de pizarrón en simple, el sistema completo consiste en:

1. Un juego de módulos independientes llamados fuentes del conocimiento (KS's knowledge sources) los cuales contienen el dominio del conocimiento especifico del sistema.
2. Un pizarron, el cual es la estructura de la información compartida por la cual las bases del conocimiento se comunican entre si.
3. Un sistema de control, el cual determina el orden en que las bases del conocimiento operaran en las entradas del pizarrón.

Para ver como estas piezas trabajan juntas vamos a ver al sistema HEARSAY-II las fuentes del conocimiento corresponden a los niveles de conocimiento acerca del lenguaje (silabas, palabras, frases y oraciones), y las tareas están a discusión. El pizarrón contiene hipótesis acerca de interpretaciones en cada uno de estos niveles. El control se realiza por una fuente especializada del conocimiento que razona acerca de estos factores como costo de ejecución y posibilidades de lograr un resultado.

Cuando una fuente del conocimiento es activada (como se describe abajo) esta examina el contenido actual de el pizarron y aplica su conocimiento ya sea para crear una nueva hipótesis y

escribirla en el pizarron o para modificar una existente, Aunque la ejecución del sistema HEARSAR-II completo consiste en la ejecución asíncrona de la colección de las fuentes del conocimiento, la ejecución de una fuente individual es un proceso secuencial. Una vez que la fuente del conocimiento es activada se ejecuta sin ser interrumpida hasta que termina.

Las hipótesis en el pizarron estan organizadas a lo largo de dos dimensione: nivel (desde el pequeño, hipótesis de bajo nivel acerca de sonido individuales a los largos, alto nivel acerca del significado de una oración completa) y tiempo La meta del sistema es la de crear un sola hipótesis que represente la solución a un problema.

• Sistemas multiagentes

La forma más sencilla de razonamiento distribuido es aquella que se efectúa por medio de un agente simple que:

1. Descompone el objetivo en subobjetivos, y
2. Asigna los subobjetivos entre el resto de agentes.

Este tipo de razonamiento se denomina normalmente *planificación multiagente.* El primer paso, el de la descomposición del problema, es esencialmente el mismo que se llevaba a cabo para sistemas de planificación con un único agente. En teoría, el resultado de la descomposición es un conjunto de subproblemas mutuamente independientes. Sin embargo, esto no siempre es posible, así que se deberán usar algunas de las técnicas que se vieron en el Capítulo 13.

Una vez que se realiza la descomposición, los subproblemas se deben emparejar con los agentes disponibles, para su posterior ejecución. En este punto, la planificación difiere de la que se hace con un único agente en lo siguiente:

- A menos que todos los agentes esclavos sean idénticos, el agente maestro debe tener acceso a los modelos de las capacidades de los diferentes esclavos. Estos modelos hacen posible diferenciar qué tareas son más apropiadas para cada diferente agente esclavo.
- Incluso si todos los agentes esclavos son idénticos, el agente maestro debe hacer balance para asegurarse de que el objetivo final se completa tan pronto como sea posible
- Una vez que las tareas han sido distribuidas, es necesario alcanzar la sincronización entre los esclavos, a menos que todas las tareas sean completamente

Todo el sistema consta de:

- Un conjunto de módulos independientes denominados fuentes de conocimiento (o KSs), que contienen el conocimiento del dominio específico del sistema.
- Una pizarra, que es la estructura de datos compartida a través de la que se comunican entre sí las fuentes de conocimiento.
- Un sistema de control, que determina el orden en que las fuentes de conocimiento operarán sobre las entradas de la pizarra.

Para ver cómo funcionan juntas todas estas piezas, se echará un vistazo al sistema HLARSAY-II. Aquí, el KSs corresponde a IOS niveles del conocimiento que se tiene del habla, del lenguaje (silabas, palabras, frases y oraciones), y el problema que se está tratando. La pizarra contiene hipótesis acerca de las interpretaciones en cada uno de esos niveles. El control se lleva a cabo mediante una fuente especializada de conocimiento que razona acerca de factores como el coste de la ejecución y la probabilidad de alcanzar un resultado satisfactorio.

Cuando se activa un KS (como se describe más adelante), éste examina los contenidos de la

pizarra en ese determinado momento, y aplica su conocimiento tanto para crear una nueva *hifrótesis* y escribirla en la pizarra, como para modificar una ya existente. Aunque la ejecución de todo el sistema HEARSAY-II consiste en la ejecución a síncrona de un conjunto de KSs, la ejecución de un único KS es un proceso secuencial. Una vez que el KS es activado, se ejecuta sin interrupción hasta el final.

Las hipótesis de la pizarra se desarrollan a lo largo de dos dimensiones: nivel (que va desde pequeñas hipótesis de bajo nivel relativas a sonidos individuales, hasta hipótesis de alto nivel acerca del significado de una secuencia completa) y tiempo (correspondiente al análisis de los períodos de emisión). El objetivo del sistema es crear una única hipótesis que represente una solución al problema. Para el sistema HEARSAY-II, tal solución podría ser, por ejemplo, una interpretación aceptable de una emisión completa.

II. Aprendizaje

1. simbólico

• aprendizaje de árboles de decisión

La inducción mediante arboles de decisión es a la vez una de las modalidades mas sencillas y mejores del algoritmo de aprendizaje. Constituye un adecuado medio para el area del aprendizaje inductuivo y su implantación es fácil. Primero explicaremos el elemento de ejecución.

Los árboles de decisión como elementos de desempeño

Un árbol de decisión toma como entradas objetos o situaciones caracterizados mediante un conjunto de propiedades; el árbol entrega a la salida una "decisión" sí o no. Es decir, los árboles de decisión representan tunciones booleanas. Aunque también es posible representar tunciones con un mayor rango de salidas, por razones de sencillez nos limitaremos al caso booleano. Los nodos internos del árbol corresponden a una prueba del valor de una de las propiedades y las ramas del nodo son identificadas mediante los posibles valores de la prueba. En los nodos hoja del árbol se especifica el valor booleano que hay que producir en caso de llegar a una hoja determinada.
Por ejemplo, considere el problema de decidir si está dispuesto a esperar a que le asignen una mesa en un restaurante. El objetivo reside aquí en aprender una definición del predicado de la meta[4] *Esperará,* definición expresada mediante un árbol de decisión.[5] Al plantear lo anterior como un problema de aprendizaje, primero hay que especificar de qué propiedades o *atributos* se dispone para describir los ejemplos del dominio.[6] Supóngase que elegimos la siguiente lista de atributos;

1. *Alternativa:* Si algun restaurante cercano ofrece una alternativa adecuada.
2. *Bar:* Si el restaurante tiene un área de bar cómoda en donde esperar la asignación de la mesa.
3. *Vier/Sáb:* Es válido en viernes y sábados.
4. *TenerHambre:* 5i tenemos hambre.
5. *Clientes:* Cuántas personas hay en el restaurante (los valores son *Ninguno, Algunos, EstáLleno).*
6. *Precio:* El rango de los precios del restaurante (5, $5, sss).
7. *Lluvia:* Si está lloviendo atuera.
8. *Reservación:* Si se hizo una reservación.
9. *Tipo:* El tipo de restaurante (francés, italiano, tailandés o de hamburguesas).

10. *TiempoEsperaEstimado:* Tiempo de espera estimado por el anfitrión (0-10 minutos, 10-30, 30-60)

2.probabilistico

- **clasificador bayesiano**

La técnica más antigua y mejor definida para manejar la incertidumbre es la ***Regla de Bayes***, la misma que está basada en la teoría clásica de la probabilidad. Las hipótesis son más o menos probables dependiendo de las posibilidades de los hechos o evidencias que las sostienen. La probabilidades se calculan en base a la fórmula general de la probabilidad condicionada de ***Bayes*** o alguna transformación de la misma.

El procedimiento para el modelo probabilístico es el siguiente:

- El factor de un conjunto de condiciones unidas por el operador lógico **Y** (**AND)** es igual al producto de cada una de las evidencias que intervienen.
- El factor de un conjunto de condiciones unidas por el operador lógico **O** (**OR)** es igual al complementario del producto de los complementarios de cada una de las evidencias que intervienen.
- Para el cálculo del coeficiente de la regla se aplica la ***Regla de Bayes***:

$$P(C_i \ / \ H) = \frac{P(H \ / \ C_i) \ . \ P(C_i)}{\sum_j (P(H \ / \ C_j) \ . \ P(C_j))}$$

Donde, ***P*** es la probabilidad, ***C*** son las conclusiones o resultados, ***H*** son los hechos o evidencias, ***i*** es una conclusión determinada, ***j*** es una variable que va de **1** al número de conclusiones posibles. Para aplicar esta fórmula, las conclusiones deben ser excluyentes y completas.

A pesar que el método de Bayes es mucho más desarrollado que otros métodos para manejar incertidumbre, no deja de tener ciertas dificultades prácticas:

1. Requiere de una gran cantidad de datos probabilísticos para construir una base de conocimientos. Por ejemplo, si un sistema de diagnóstico posee ***p*** conclusiones detectables y ***q*** características observables relevantes, requiere un mínimo de (***p * q + p***) valores probabilísticos, asumiendo que: todas las conclusiones son mutuamente excluyentes, las características son condicionalmente independientes para cada conclusión, y que todas las características son valores verdaderos. Caso contrario, se requeriría de un número significativamente mayor que el indicado.
2. Los tamaños de la muestra para obtener las probabilidades condicionales deben ser lo suficientemente grandes, como para que las probabilidades obtenidas sean exactas y significativas.
3. A menudo las relaciones entre la hipótesis y la evidencia son importantes para determinar la forma en que la incertidumbre será manejada. Al reducirse estas asociaciones a simples números, remueve información relevante que podría utilizarse para razonar con éxito acerca de las incertidumbres.
4. La reducción de dichas asociaciones a números también elimina la posibilidad de utilizar este conocimiento en otras tareas.

III. Búsqueda y juegos

1. conceptos básicos

- **representación de espacio de estados**

EL MÉTODO DE ANÁLISIS DE MEDIOS Y METAS

El estado de un sistema es una descripción que basta para determinar el futuro. En un espacio de estados, cada nodo representa un estado, y cada enlace una posible transición de un paso de un estado a otro:

Un **espacio de estados** es una representación, o sea una red semántica en la que
los nodos representan estados;
los enlaces representan transiciones entre estados.

En consecuencia un espacio de estados es miembro de la familia de redes semánticas de representaciones que se introdujo.

En el contexto de la resolución de problemas, los estados corresponden al lugar en que usted está o debería estar en el proceso de resolver un problema. De ahí que el estado actual sea el lugar en donde usted se encuentra; el estado meta, el lugar al que desea llegar; y el problema consiste en hallar una sucesión de transiciones que conduzca del estado inicial al estado meta.

En lo que resta de esta sección, usted aprenderá acerca del análisis de medios y metas, un método estándar para seleccionar transiciones. También podrá conocer una forma popular de instrumentar el análisis de medios y metas mediante una tabla sencilla.

2. sin información

- **búsqueda en profundidad y a lo ancho**

Algoritmo Primero en Profundidad (DEPTH-FIRST)

1. Si el estado inicial es el objetivo, salir y retornar éxito.
2. Sino, haga lo siguiente hasta que se obtenga señal de éxito o fracaso:
    a. Genere un sucesor E del estado inicial. Si no hay más sucesores, retorne con señal de fracaso.
    b. Llame recursivamente al algoritmo, esta vez con E como el estado inicial.
    c. Si la señal es éxito, retorne, de otra manera, continúe en este lazo

Algoritmo Primero a lo Ancho (BREATH-FIRST)

1. Crear una variable NODE_LIST y ponerla al estado inicial.
2. Hasta que se encuentre el objetivo o hasta que NODE_LIST esté vacía haga lo siguiente:
    a. Remover el primer elemento de NODE_LIST, y llamarlo E. Si NODE_LIST estuvo vacía, salir.
    b. Para cada forma en que cada regla puede ajustarse al estado descrito en E, haga lo siguiente:
        i. Aplicar la regla para generar un nuevo estado.
        ii. Si el nuevo estado es un estado objetivo, salir y retornar este estado.
        iii. Sino, añada el nuevo estado al final de NODE_LIST.

3. Con información

- ## Hill climbing, best-first,beam search y A*

Ascenso a Colina (Hill Climbing)

Es una variante del algoritmo de búsqueda de generación y prueba. Del procedimiento de prueba existe una realimentación que ayuda al generador a decidirse por cual dirección debe moverse en el espacio de búsqueda. En estos procesos se abandona la búsqueda si no existe un estado alternativo razonable al que se pueda mover.

Los algoritmos de **ascenso a colina** son típicamente **locales**, ya que deciden qué hacer, mirando únicamente a las consecuencias inmediatas de sus opciones. Puede que nunca lleguen a encontrar una solución, si son atrapados en estados que no son el objetivo, desde donde no se puede hallar mejores estados, por ejemplo:

1. **Un máximo local**, que es un estado mejor que sus vecinos pero no es mejor que otros que están algo más alejados.

2. **Una meseta**, es un espacio de búsqueda en el que todo un conjunto de estados vecinos tienen igual valor.

3. **Un risco**, que es un tipo especial de máximo local, imposible de atravesar con movimientos simples.

Hay algunas formas que pueden ayudar a solventar estos problemas, aunque no existe garantía:

1. Para evitar **máximos locales**, regresar a un estado anterior y explorar en una dirección diferente.

2. Para casos de **mesetas**, dar un salto grande en alguna dirección y tratar de encontrar una nueva sección del espacio de estado.

3. Para los **riscos**, aplicar dos o más reglas, antes de realizar una prueba del nuevo estado, esto equivale a moverse en varias direcciones a la vez.

Los algoritmos de ascenso a colina, a pesar de explorar sólo un paso adelante, al examinar el nuevo estado pueden incluir una cierta cantidad de información global codificada en la **función objetivo** o **función heurística**.

Primero el Mejor (Best-First)

Este algoritmo, combina las ventajas de los algoritmos primero en profundidad y primero en amplitud. Sigue un sendero a la vez, pero puede cambiarse a otro sendero que parece más prometedor que el que está siguiendo.

En este sentido, puede considerarse que es un algoritmo que realiza su proceso de búsqueda en un **grafo** de tipo **O**, ya que todos sus ramales representan una alternativa de solución. Para su operación, el algoritmo necesita dos listas de nodos y una función heurística que estime los méritos de cada nodo que se genere:

33

1. **ABIERTOS** - Es una variable que contiene los nodos que han sido generados. La función heurística ha sido aplicada a ellos, pero todavía no han sido examinados, es decir no se han generado sus sucesores. **ABIERTOS** puede considerarse como una **COLA DE PRIORIDADES** en la que los elementos con mayor prioridad son los que tienen los valores más prometedores, dados por la función heurística.

2. **CERRADOS** - Es una variable que contiene los nodos que han sido examinados. Es necesario tener esta información, para que la búsqueda sea en un grafo y no en un árbol.

3. **FUNCIÓN HEURÍSTICA** - Permite que el algoritmo busque primero por senderos que son o parecen más prometedores.

Para muchas aplicaciones, es conveniente definir esta función ***f'***, como la suma de dos, que se las llamará ***g*** y ***h'***. La función ***g*** es una medida del costo de llegar desde el nodo inicial al nodo actual. La función ***h'*** es una estimación del costo adicional para llegar desde el nodo actual al estado objetivo. Aquí es donde se explota el conocimiento que se dispone sobre el dominio del problema.

Es decir, la función combinada ***f'*** representa una estimación del costo de llegar desde el estado inicial hasta el estado objetivo, siguiendo el sendero que ha generado el nodo actual. Si el nodo actual ha generado más de un sendero, el algoritmo deberá dejar registrado sólo el mejor.

El algoritmo, en la forma que fue formulado, se aplica a grafos. Puede ser simplificado para aplicarse a árboles, si no se preocupa de comprobar si un nuevo nodo esta en ABIERTOS o en CERRADOS. Esto aceleraría la generación de nodos y la búsqueda, para casos en que es poco probable que se repitan nodos.

Usualmente, el costo de ir de un nodo a su sucesor, g, se fija en una constante igual 1, cuando se desea encontrar un sendero a la solución, que involucre el menor número de pasos. Si por el contrario nos interesa encontrar el camino menos costoso y algunos operadores cuestan más que otros, se asume un valor de g, que refleje esos costos. Un valor de g igual a cero significaría que simplemente nos interesa llegar a alguna solución, de cualquier manera.

Si h' es un estimador perfecto de h, hará que A* converja inmediatamente al objetivo, sin búsqueda. Mientras mejor sea h', más cerca se estará de alcanzar esta aproximación directa. Si ***h'*** vale cero, la búsqueda será controlada por ***g***. Si el valor de ***g*** es también cero, hará que la búsqueda sea aleatoria. Si el valor de ***g*** es siempre 1, hará que la búsqueda sea primero en anchura. Para los casos en que ***h'*** no sea perfecto ni cero, y nunca llega a sobrestimar el valor de ***h***, el algoritmo $\mathbf{A}^*$ está garantizado que encontrará un sendero óptimo a un objetivo, en caso de que exista solución. Cuando un algoritmo garantiza el encontrar una solución óptima, si esta existe, se dice que es **admisible**.

Teorema de Admisibilidad

Si ***h'*** nunca sobrestima a ***h*** entonces A* es admisible. La única manera de garantizar que ***h'*** nunca sobrestime a ***h*** es haciéndolo cero, pero con ello estamos en la búsqueda primero en anchura, es decir el algoritmo es admisible, pero no eficiente. Pero existe un corolario que puede ser de utilidad en casos prácticos:

COROLARIO: ***Si h' muy rara vez sobrestima h por más de entonces el algoritmo A* rara vez encontrará una solución cuyo costo sea mayor más que en que el costo de la solución óptima.***

Bajo ciertas condiciones, el algoritmo **A*** puede ser óptimo, en el sentido de generar el menor número de nodos posibles, en el proceso de encontrar una solución, pero bajo otras condiciones puede no serlo.

Algoritmo Guiado por Agenda

Una agenda es una lista de tareas que un sistema podría realizar. Asociado con cada tarea están, usualmente, dos cosas: una lista de razones de por qué se propone la tarea (justificaciones); y, un valor representando el peso de la evidencia que sugiere que la tarea podría ser útil.

El algoritmo A*

El algoritmo de búsqueda del primero mejor descrito antes es una simplificación dc un algoritmo denominado A~, que fue introducido por primera vez por Hart *et al.* (1968; 1972). Este algoritmo utiliza la mismas funciones fi, g y *h',* así como las listas *ABIERTOS* Y *CERRADOS,* que ya se han desaito.

Algoritmo: A*

1. Empezar con *ABIERTOS* conteniendo sólo el nodo inicial. Poner el valor g de ese nodo a O, su valor *h'* al que corresponda, y su valor fi a *h'* + O, es decir, a h^1. Inicializar *CERRADOS* como una lista vacía.
2. Repetir el siguiente procedimiento hasta que se encuentre un nodo objetivo: si no existen nodos en *ABIERTOS,* informar del fallo. En caso contrario, tomar aquel nodo de *ABIERTOS* que tenga el menor valor de fi. Llamarle *MEJORNODO.* Quitarlo de *ABIERTOS* y colocarlo en *CERRADOS.* Mirar si *MEJORNODO* es un nodo objetivo, si lo es, salir e informar de la solución (bien *MEJORNODO* silo único que se desea es el nodo, o todo el camino empleado en unir el estado inicial y *MEJORNODO* si se está interesado en la ruta.). En caso contrario, generar los sucesores de *MEJORNODO,* pero sin colocar *MEJORNODO* apuntando a ellos (antes debe mirarse si alguno de ellos ya ha sido generado). Para cada *SUCESOR,* hacer lo siguiente:

a) Poner a *SUCESOR* apuntando a *MEIORNODO.* Estos enlaces hacia atrás hacen posible que se recupere el camino una vez que se ha encontrado la solución.

b) Calcular *g(SUCESOR) = g(MEIORNODO)* + el coste de ir desde *MEIORNODO* hasta *SUCESOR.*

c) Mirar si *SUCESOR* está contenido en *ABIERTOS* (es decir, si ya ha sido generado pero no procesado). Si es así, se denomina a ese nodo *VIEJO.* Puesto que este nodo ya existe en el grafo, se puede desechar *SUCESOR,* y añadir *"lElO* a la lista de los sucesores de *MEIORNODO.* Ahora debe decidirse si el enlace paterno de *VIEJO* debe apuntar a *MEJORNODO.* Deberia ocurrir así si el camino que se ha encontrado hasta *SUCESOR* es de menor coste que el mejor camino actual hasta *VIEJO* (puesto que *SUCESOR* y *VIEJO* son, en realidad, el mismo nodo). Para ver si cuesta menos llegar hasta *VIEJO* a través de su padre actual o hasta *SUCESOR* a través de *MEJORNODO,* se comparan sus valores g. Si *VIEJO* tiene menor (o igual) coste, no es necesario hacer nada. Si *SUCESOR* tiene menor coste, entonces se hace que el enlace paterno de *VIEJO* apunte hacia *MEJORNODO,* y que se grabe el nuevo camino óptimo en *g(VIEJO)* y que se actualice f^1*(VIEJO).*

d) Si *SUCESOR* no se encontraba en *ABIERTOS,* mirar si se encuentra en *CERRADOS.* Si es así, llamar *VIEJO* al nodo de *CERRADOS* y añadirlo

a la lista de sucesores de ***MEJORNODO.*** Mirar si el nuevo camino es mejor que el viejo tal y

como se hizo en el paso *2(c)* y actualizar apropiadamente el enlace paterno y los valores de g y f'. Si se acaba de encontrar un camino mejor a ***VIEJO,*** se debe propagar la mejora a los sucesores de *VIEJO.* Esto es algo delicado: ***VIEJO*** apunta a sus sucesores. Cada sucesor, a su vez, apunta a sus sucesores, y así sucesivamente hasta que cada rama termina en un nodo que, o bien está en *ABIERTOS* o no tiene sucesores. Por tanto, para propagar el nuevo coste hacia abajo, se puede hacer una búsqueda transversal en profundidad del árbol, empezando en *VIEJO,* cambiando el valor g de cada nodo (y por tanto su valor f'), terminando cada rama cuando se alcanza o bien un nodo sin sucesores, o bien un nodo para el que ya se ha encontrado un camino equivalente o mejor.[4] Es fácil examinar esta condición. El enlace paterno de cada tiodo apunta hacía atrás a su mejor predecesor conocido. Conforme hay una propagación a un nodo siguiente, se debe comprobar si su predecesor apunta al nodo del cual se viene. Silo hace así, la propagación debe continuar. Si no, en su valor de g ya aparece el mejor camino del que forma parte y, por lo tanto, la propagación se puede detener en este punto. Pero es posible que al propagar el nuevo valor de g hacia abajo, el camino que se esté siguiendo pueda volverse mejor que el camino a través del antecesor actual. Por tanto, deben compararse los dos. Si el camino a través del antecesor actual todavía es mejor, debe detenerse la propagación. Si el camino a través del cual se propaga es ahora mejor, se inicializa el antecesor y se continúa con la propagación.

e) Si *SUCESOR* no estaba ya en *ABIERTOS* o en *CERRADOS,* ponerlo en *ABIERTOS* y añadirlo a la lista de sucesores de *MEJORNODO.* Calcular *f'(SUCESOR) = g(SUCESOR) + h'(SUCESOR).*

Se pueden hacer algunas interesantes observaciones sobre este algoritmo. La primera de ellas hace referencia al papel que desempeña la función g. Esta función permite hacer la elección del nodo que se va a expandir a continuación sobre la base, no sólo de la bondad que el nodo en sí mismo pueda indicar (medida por *h'),* sino sobre la base de la bondad del camino que ha conducido a ese nodo. Con la incorporación de g en f' no siempre se elige como el siguiente nodo a expandir aquel que parece más cercano al objefivo. Esto es útil si el camino elegido tiene relevancia. Si sólo es relevante alcanzar el objetivo sea de la forma que sea, se puede definir g como O, de forma que siempre se elige el nodo que parezca más cercano al objetivo Erial. Si lo que se desea es encontrar un camino con el menor número de pasos posibles, se hace que el coste aplicado para pasar de un nodo a otro sea constante.

4. Juegos

- **Mínimax**

El procedimiento de búsqueda mínimas es un procedimiento de búsqueda en profundidad, de profundidad limitada. Se describio brevemente anteriormente

La idea consiste en comenzar en la posición actual y usar el generador de movimientos plausibles para generar un conjunto de posiciones sucesivas posibles. Entonces se puede aplicar la función de evaluación estática a esas posiciones y escoger simplemente la mejor. Después de hacer esto, puede llevarse hacia atrás ese valor hasta la posición de partida para representar nuestra evaluación de la misma. La posición de partida es exactamente tan buena para nosotros como la posición generada por el mejor movimiento que podamos hacer a continuación. Aquí se va a suponer que la función de evaluación estática devuelve valores elevados para indicar buenas situaciones para nosotros, de manera que nuestra meta es *maximizar* el valor de la función de evaluación estática de la siguiente posición del tablero.

En la Figura 12.1 se muestra un ejemplo de esta operación. Supone una función de evaluación estática que devuelve valores entre -10 y 10, donde 10 indica una victoria para nosotros, -10 una victoria para el oponente y O un empate. Puesto que nuestra meta es maximizar el valor de la

función heurística, escogemos movernos a B. Al llevar hacia atrás el valor de B hasta A, podemos concluir que el valor de A es 8, puesto que sabemos que se puede llegar a una posición con un valor de 8.
Puesto que sabemos que la función de evaluación estática no es completamente precisa, nos gustaría llevar la búsqueda más allá de una sola capa. Esto podría ser muy importante, por ejemplo, en un juego de ajedrez donde estemos en medio de un intercambio de piezas. Después de nuestro movimiento, podría parecer que nuestra situación es muy buena, pero, si mirásemos un movimiento más allá, veríamos que una de nuestras piezas también es capturada, por lo que la situación no era tan favorable como en principio parecía. Por tanto, nos gustaría prever qué es lo que sucederá en cada una de las nuevas posiciones del juego del siguiente movimiento realizado por el oponente. En lugar de aplicar la función de evaluación estática a cada una de las posiciones que acabamos de generar, aplicamos a cada una de ellas el generador de movimientos plausibles, generando un conjunto dc posiciones posteriores para cada una de ellas. Si queremos paramos aht en una previsión de dos capas, podríamos aplicar la función de evaluación estática a cada una de dichas posiciones, tal y como muestra la Figura 12.2.
Pero ahora debemos tener en cuenta el hecho de que el movimiento que se realiza a continuación debe elegirlo el oponente, y por tanto, deberá propagarse hacia atrás al siguiente nivel. Supongamos que realizamos el movimiento B. Entonces el oponente debe elegfr entre los movimientos E, F y O. El objetivo del oponente es *minimizar* el valor de la función de evaluación, por lo que puede esperarse que elija moverse a F. Esto significa que si hacemos el movimiento B, la verdadera posición en la que desembocamos en el siguiente movimiento es muy mala para nosotros. Esto es cierto aunque el nodo E represente una posible configuración que fuese buena para nosotros. Pero ya que no nos toca a nosotros mover en este nivel, no podremos elegirlo. La Figura 12.3 muestra el resultado de propagar los nuevos valores hacia la raíz del árbol. En el nivel representado por la elección del oponente, se elige el menor valor y se propaga hasta la raíz. En el nivel que representa nuestra elección se ha elegido el nivel máximo.

Una vez que se han propagado hacia atrás los valores del segundo nivel, resulta claro que el movimiento correcto que podemos realizar en el primer nivel, dada la información disponible, es C, puesto que no hay nada que el oponente pueda hacer ahí para producir un valor peor que -2. Este proceso puede repetirse para tantos niveles como lo permita el tiempo disponible, y las evaluaciones más precisas que se produzcan pueden usarse para elegir el movimiento correcto en el nivel más alto. La alternancia de maximización y minimización en capas alternas cuando las evaluaciones se envían de regreso a la raíz, se corresponde con las estrategias opuestas que siguen los dos jugadores y que da a este método el nombre de minimax.
Una vez descrito informalmente el funcionamiento del procedimiento minimax, vamos a describirlo con precisión. Es un procedimiento directamente recursivo que se basa en dos procedimientos auxiliares específicos del juego que se está jugando:

1. GENMOV(Posición, *Jugador):* El generador de movimientos plausibles, que devuelve una lista de nodos que representan aquellos movimientos que podría realizar *Jugador* en *Posición.*
2. *EST~CA(Posidón, Jugador):* La función de evaluación estática que devuelve un número que representa la bondad de *Posición* desde el punto de vista de *Jugador*[2].

Al igual que en cualquier programa recursivo, un punto critico en el diseño de MINI~ItAN es cuándo parar la recursión y llamar simplemente a la función de evaluación estática. Existen diversos factores que pueden influenciar dicha decisión. Estos incluyen:

- ¿Ha ganado algún bando?
- ¿Cuántas capas se han explorado hasta el momento?
- ¿Cuán prometedor es este camino?
- ¿Cuánto tiempo nos queda?
- ¿Cuán estable es la configuración?

Para el procedimiento general MINIMAX que se explica a continuación, se hará uso de una función, BASTANTE-PROFUNDO, que suponemos evaluará todos esos factores y devolverá

CIERTO si debe pararse la búsqueda en el nivel actual y FALSO en caso contrario. Esta sencilla implementación de BASTANTE-PROFUNDO toma dos valores, *Posición* y *Profundidad.* Ignorará su parámetro *Posición* y simplemente devolverá CIERTO si su parámetro *Profundidad* excede a una constante de corte.
Un problema que surge al definfr MINIMAX como procedimiento recursivo es que debe devolver dos resultados en vez de uno:

- El valor propagado del camino elegido
- El camino mismo. Se devuelve el camino entero aunque, probablemente, sólo se necesite en realidad el primer paso del mismo.

Supondremos que MINIMAX devuelve una estructura que contiene ambos resultados y que tenemos dos funciones, VALOR y CAMINO que extraen cada uno de los componentes por separado.
Puesto que el procedimiento MINIMAX se define como una función recursiva, debe especificarse también cómo debe llamarse inicialmente. Toma tres parámetros, una situación del tablero, la profundidad actual de búsqueda y el jugador que mueve. Así pues, la primera llamada para calcular el mejor movimiento en la posición AC"JAL debería ser

MINIMAX(ACTUAL, O, JUGADOR-UNO)

si mueve JUGADOR-UNO, o

MINIMAX(ACTUAL, O, JUGADOR-DOS)

si es JUGADOR-DOS el que mueve.

Algoritmo: MININIAX (Posición, Profundidad, Jugador)

1. Si BASTANTE-PROFUNDO(Posición, *Profundidad),* entonces devolver la estructura:
VALOR = *EST~CAQosición, Jugador);* CAMINO = nada
Esto indica que no existe ningún camino a partir de este nodo y que su valor es el que determine la función de evaluación estática.
2. En caso contrario, generar otra capa del árbol llamando a la función GEN-*MOV(Posición, Jugado~* y poniendo en SUCESORES la lista que devuelva.
3 Si SUCESORES está vacío, entonces no puede realizarse ningún movimiento, por lo que se devuelve la misma estructura que se habría devuelto si BASTANTE-PROFUNDO hubiese sido verdadero.
4. Si SUCESORES no está vacío, entonces examinar cada elemento por orden y grabar el mejor. Esto se hace de la siguiente forma:
Inicializar MEJOR-RESULTADO al mínimo valor que pueda devolver ESTÁTICA. Se actualizará para reflejar el mejor tanteo que pueda obtenerse de un elemento de SUCESORES.
Para cada elemento SUC de SUCESORES, hacer lo siguiente:
a) Actualizar RESULT-SUC a

MINIMAX(SUC, *Profundidad* + 1, *CONTRARIO(Jugador))*[3].

Esta llamada recursiva a MIMMAX realizará realmente la exploración de SUC.
b) Actualizar NUEVO-VALOR a -VALOR(RESULT-SUC). Esto hará que se reflejen los méritos de la posición desde la perspectiva opuesta a la del siguiente nivel inferior.

*c)*Si NUEVO-VALOR> MEJOR-RESULTADO, entonces hemos encontrado un sucesor que es mejor que todos los hallados hasta el momento. Grabarlo haciendo lo siguiente:
i. Actualizar MEJOR-RESULTADO a NUEVO-VALOR.
ii. El mejor camino conocido va ahora desde ACTUAL a SUC, y desde ahí siguiendo el

camino apropiado desde SUC, tal como se determine llamando recursivamente a MINIMAX. Por tanto, actualizamos MEJOR-CAMINO con el resultado de añadir SUC a CAMINO(RESULT-SUC).
5. Una vez examinados todos los sucesores, conocemos el valor de Posición al igual que el camino a tomar a partir de él. Por tanto, se devuelve la estructura
VALOR = MEJOR-RESULTADO CAMINO = MEJOR-CAMINO

Cuando se devuelve la primera llamada a MIINIMAX, el primer elemento de CAMINO es el mejor movimiento a partir de ACTUAL. Para ver cómo funciona este procedimiento, el lector deberia trazar su ejecución para el árbol del juego.
El procedimiento MINIMAX que acaba de describirse es muy simple. Pero su rendimiento puede mejorarse de forma impoflante mediante algunos refinamientos. En las siguientes secciones se ve algunos de ellos.

- **ADICIÓN DE LA PODA ALFA-BETA**

Recuerde que el procedimiento minimax es un proceso primero en profundidad. Cada camino se explora tan lejos como lo permita el tiempo, la función de evaluación estática se aplica a las posiciones del juego en el último paso del camino, y el valor debe entonces propagarse hacia atrás en el árbol de nivel en nivel. Una de las ventajas de los procedimientos primero en profundidad es que a menudo puede mejorarse su eficiencia usando técnicas de ramificación y acotación, en las que pueden abandonarse rápidamente aquellas soluciones parciales que son claramente peores que otras soluciones conocidas. Describimos una aplicación directa de esta técnica en el problema del viajante de comercio del Apartado 2.2.1. Para este problema, lo único que se necesitaba era recordar la longitud del mejor camino que se había encontrado hasta el momento. Si otro camino parcial sobrepasaba esa cota, se abandonaba. Pero, así como fue necesario modificar ligeramente nuestro procedimiento de búsqueda para tratar los jugadores maximizador y minimizador, también es necesario modificar la estrategia de ramificación y acotación para incluir dos acotaciones, una para cada uno de los jugadores. Esta estrategia modificada se denomina *poda alfa-beta.* Requiere el mantenimiento de dos valores umbral, uno que representa la cota inferior del valor que puede asignarse en último término a un nodo maximizante (que llamaremos *alfa)* y otro que represente la cota superior del valor que puede asignarse a un nodo minimizante (que llamaremos *beta).*
Para ver cómo funciona el procedimiento alfa-beta, consideremos el ejemplo mostrado. Después de analizar el nodo F sabernos que el oponente tiene garantizado un resultado de -5 o menos en C Q>uesto que el oponente es el jugador minimizador). Pero también sabemos que tenemos un resultado seguro de 3 o más en el nodo A, donde podemos llegar si movemos a B. Cualquier otro movimiento que produzca un resultado de menos de 3 es peor que el movimiento a B, y podemos ignorarlo. Con Sólo examinar F podemos asegurar que un movimiento en C es peor (será menor o igual que -5), sin necesidad de mkar el resultado del nodo G. Así pues, no necesitamos en absoluto explorar el nodo O. Naturalmente, puede parecer que la poda de un nodo no justifica el gasto de grabar los límites y examinarlos, pero si estuviésemos explorando un árbol de seis niveles, entonces no sólo habríamos eliminado un nodo único, sino un árbol entero de tres niveles de profundidad.

IV. Lenguajes del area

1. funcionales

Tradicionalmente **LISP** y **PROLOG** han sido los lenguajes que se han utilizado para la programación de SBC. Estos lenguajes ofrecen características especialmente diseñadas para manejar problemas generalmente encontrados en IA. Por este motivo se los conoce como ***lenguajes de IA***.

Una de las principales características que comparten los lenguajes LISP y PROLOG, como consecuencia de su respectiva estructura, es que pueden ser utilizados para escribir programas capaces de examinar a otros programas, incluyendo a ellos mismos. Esta capacidad se requiere,

por ejemplo, para hacer que el programa explique sus conclusiones. Esto sólo puede hacerse si el programa tiene la capacidad de examinar su propio modo de operación.

- **LISP**

Su nombre se deriva de **LIS**t **P**rocessor. LISP fue el primer lenguaje para procesamiento simbólico. John McCarthy lo desarrolló en 1958, en el Instituto de Tecnología de Massachusetts (MIT), inicialmente como un lenguaje de programación con el cual los investigadores pudieran implementar eficientemente programas de computadora capaces de razonar. Rápidamente LISP se hizo popular por su capacidad de manipular símbolos y fue escogido para el desarrollo de muchos sistemas de IA.

Existen varios dialectos de LISP en el mercado informático, sin embargo COMMON LISP puede considerarse como el estándar *de facto*. Actualmente, LISP se lo utiliza en varios dominios que incluyen la escritura de compiladores, sistemas para diseño VLSI, sistemas para diseño mecánico asistido por computadora (AUTOCAD), animaciones gráficas y sistemas basados en conocimiento.

2. Lógicos

- **Clausulas de Horn**

Las cláusulas de Horn (instrucciones ejecutables de PROLOG) tienen el siguiente aspecto:

```
hija (*A, *B) <- mujer (*A), padre (*B, *A).
```

que podría leerse así: "A es hija de B si A es mujer y B es padre de A".

Obsérvese que, en PROLOG, el símbolo <- separa la conclusión de las condiciones. En PROLOG, las variables se escriben comenzando con un asterisco. Todas las condiciones deben cumplirse simultáneamente para que la conclusión sea válida. Por tanto, la coma que separa las distintas condiciones es equivalente a la conjunción copulativa (en algunas versiones de PROLOG se sustituye la coma por el símbolo &). La disyunción, en cambio, no se representa mediante símbolos especiales, sino definiendo reglas nuevas, como la siguiente:

```
hija (*A, *B) <- mujer (*A), madre (*B, *A).
```

que podría leerse así: "A es hija de B si A es mujer y B es madre de A".

- **PROLOG**

PROgramming in **LOG**ic (PROLOG), es otro de los lenguajes de programación ampliamente utilizados en IA. PROLOG fue desarrollado en Francia, en 1973 por Alain Colmenauer y su equipo de investigación en la Universidad de Marseilles. Inicialmente fue utilizado para el procesamiento de lenguaje natural, pero posteriormente se popularizó entre los desarrolladores de aplicaciones de IA por su capacidad de manipulación simbólica. Utilizando los resultados del grupo francés, Robert Kowalski de la Universidad de Edimburgo, en Escocia, desarrolló la teoría de la programación lógica. La sintaxis propuesta por Edimburgo, se considera el estándar de facto del PROLOG.

A partir de 1981 tuvo una importante difusión en todo el mundo, especialmente porque los japoneses decidieron utilizar PROLOG para el desarrollo de sus sistemas de computación de quinta generación. Actualmente existen varios dialectos del PROLOG para diferentes plataformas.

Lenguajes del área

Rápido Prototipaje

Puede utilizarse la flexibilidad y el poderío de LISP, PROLOG o de otras herramientas de desarrollo para crear rápidamente un prototipo funcional del sistema final deseado. Este permitirá realimentar en forma temprana aspectos de profundidad y tipo de conocimiento, de las necesidades de los usuarios, así como ayudará a verificar la validez de las decisiones tomadas durante la etapa de diseño. Con esto, si se presenta un *dilema de cambio*, su impacto será mínimo debido a lo temprano de su ocurrencia.

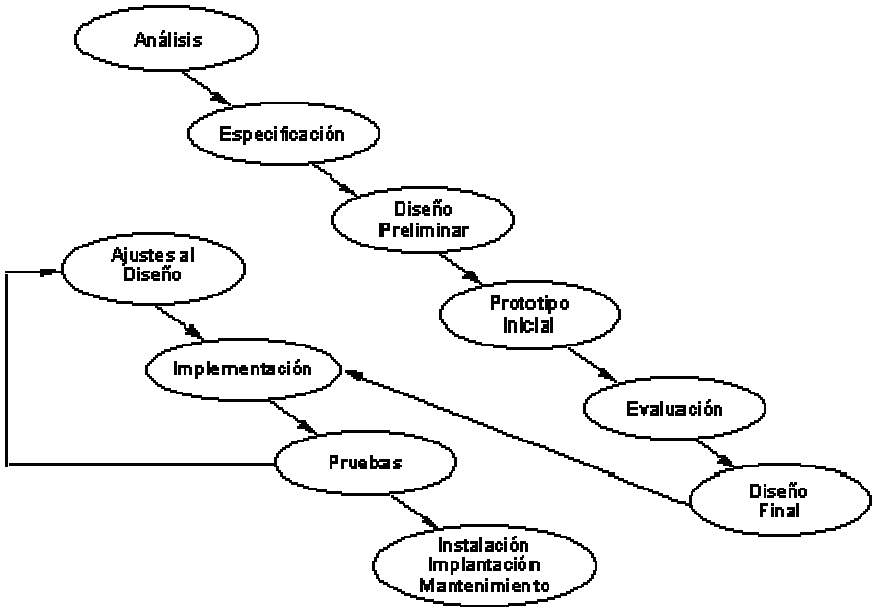


Figura 6.2. *Ciclo de vida de un Sistema Basado en Conocimiento*

En la Fig. 6.2, se puede apreciar que el desarrollo del prototipo inicial utiliza un ciclo de vida modificado del que normalmente se emplea en ingeniería de software. Las fases de análisis y especificación deben realizarse teniendo en cuenta el sistema completo, pero el diseño y la implementación del prototipo son realizados de una manera rápida y preliminar. Esto proporciona un sistema funcional que puede ser evaluado para obtener la realimentación necesaria. El prototipo inicial puede ser desechado una vez que se ha obtenido la realimentación esperada o puede ser mejorado en forma incremental para constituirse en un subsistema del SBC final. En la mayoría de casos se opta por desecharlo debido a las dificultades que existen para modificarlo y acomodar todas las ideas generadas durante su desarrollo y evaluación. En general, es más fácil empezar un nuevo proceso de desarrollo, siguiendo las diferentes etapas del ciclo de vida con mayor detalle.

Desarrollo Incremental

Una vez que el prototipo del sistema ha sido evaluado, se realiza su diseño final y se lo somete a un ciclo continuo de modificaciones para que mejore sus características y su calidad. A este

proceso se conoce como desarrollo incremental. La estrategia que se sigue, puede ser definida como:

> ***Un proceso iterativo de extracción, representación y confirmación del conocimiento en un limitado subconjunto del dominio del problema con el objetivo de construir en forma incremental el SBC en segmentos autocontenidos***.

El desarrollo incremental se centra en dos conceptos:

Dividir para conquistar, donde un segmento de conocimiento manejable pero completo, es seleccionado y desarrollado.

Desarrollo iterativo, donde el concepto de dividir para conquistar es aplicado iterativamente sobre los diferentes segmentos que conforman el problema completo.

En consecuencia, el desarrollo incremental involucra varios ciclos de recopilación del conocimiento de los expertos, incorporación del conocimiento en el sistema, revisión de la implementación resultante con los expertos y corrección de los problemas encontrados. A través de la progresiva incorporación de módulos independientes al sistema en desarrollo, un SBC puede ser rápidamente puesto en funcionamiento, aunque sea en forma parcial. En cambio un programa convencional, debido a su naturaleza procedimental, en general debe ser completamente implementado, antes de ser utilizado.

IV. Aplicaciones

1. sistemas basados en conocimientos

Los métodos generales desarrollados para la resolución de problemas y técnicas de búsqueda al inicio de la era de la IA demostraron no ser suficientes para resolver los problemas orientados a las aplicaciones, ni fueron capaces de satisfacer los difíciles requerimientos de la investigación. A este conjunto de métodos, procedimientos y técnicas, se lo conoce como **Inteligencia Artificial Débil**. La principal conclusión que se derivó de este trabajo inicial fue que los problemas difíciles sólo podrían ser resueltos con la ayuda del conocimiento específico acerca del dominio del problema.

La aplicación de estas ideas dio lugar al desarrollo de los denominados **Sistemas Basados en Conocimiento** (***K****nowledge* ***B****ased* ***S****ystems*) y al aparecimiento de la **Ingeniería Cognoscitiva**, como una rama de la **IA**, que estudia los sistemas basados en el conocimiento. La definición de un sistema basado en conocimiento puede ser simplemente la siguiente:

> Es un sistema computarizado capaz de resolver problemas en el dominio en el cual posee conocimiento específico.

La solución es esencialmente la misma que hubiera dado un ser humano confrontado con idéntico problema, aunque no necesariamente el proceso seguido por ambos puede ser igual.

El simple concepto dado, puede causar confusión ya que muchos sistemas basados en programas convencionales podrían ser incorrectamente categorizados como sistemas basados en conocimiento. Esta inconsistencia puede ser aclarada, sobre la base de tres conceptos fundamentales que distinguen a los sistemas basados en conocimiento de los programas algorítmicos convencionales y de los programas generales basados en búsqueda (IA débil):

1. La separación que existe entre el conocimiento, y la forma cómo éste es utilizado.
2. El uso de conocimiento específico de un determinado dominio.

3. La naturaleza heurística, antes que algorítmica del conocimiento utilizado.

Arquitectura (R)

La naturaleza de la resolución de problemas durante la primera década de la investigación en inteligencia artificial residía en un mecanismo de búsqueda de propósito general en el que se entrelazaban pasos de razonamiento elementales para encontrar soluciones completas.

A estos procedimientos se les ha denominado métodos débiles, debido a que la información sobre el dominio con que cuentan es débil. La única forma de remediarlo es recurrir al uso del conocimiento adecuado a pasos de razonamiento más amplios, y resolver casos recurrentes en limitadas áreas de la experiencia. Podría afirmarse que para resolver un problema prácticamente es necesario saber de antemano la respuesta respectiva.

Sistemas basados en el conocimiento: ¿clave del poder? (1969-1979)

La naturaleza de la resolución de problemas durante la primera década de la investigación en lA residía en una mecanismo de búsqueda de propósito general en el que se entrelazaban pasos de razonamiento elementales para encontrar asi soluciones completas. A estos procedimientos se les ha denominado **métodos débiles,** debido a que la información sobre el dominio con que cuentan es débil. En el caso de muchos dominios complejos resulta también que su rendimiento es débil. La única forma de remediarlo es recurrir al uso del conocimiento adecuado a pasos de razonamiento más amplios, y resolver casos recurrentes en limitadas áreas de la experiencia. Podría afirmarse que para resolver un problema prácticamente es necesario saber de antemano la respuesta respectiva.
[31 programa DENDRAL (Buchanan *el* al, 1969) constituye uno de los primeros ejemplos de este enfoque. Fue diseñado en Stanford, donde Ed Feigenbaum (anteriormente discipulo de Herbert Simon), Bruce Buchanan (filósofo convenido en científico de la computación) y Joshua Lederberg (genetista ganador del premio Nobel) colaboraron en la solución del problema de inferir una estructura molecular a partir de la información proporcionada por un espectrómetro de masas. El programa se alimentaba con la fórmula elemental de la molécula (p. e., $C6H_13N0_2$), el espectro de masas informaba sobre las masas de los diversos fragmentos de la molécula que se producían después de ser bombardeada con un haz de electrones. Por ejemplo, un espectro de masa con un pico en *ni* 15, correspondería a la masa de un fragmento metilo (CH_3).
La primera versión del programa generaba todas las posibles estructuras que correspondieran a la fórmula, luego predecía el espectro de masa que se observaría en cada caso, y comparaba éstos con el espectro real. Como era de esperar, pronto el método anterior resulté inviable para el caso de moléculas de tamaño decente. Los investigadores del DENDRAL consultaron con químicos analíticos; sc dieron cuenta de que éstos trabajaban buscando patrones bien conocidos de picos en el espectro que sugerian estructuras comunes en la molécula. Por ejemplo, para identificar el subgrupo de las cetonas **(C** --O) se empleó la siguiente regía:

si hay dos picos en y Y X2 tales que
(a) xj + X2 M + 28 (Mes la masa de toda la molécula);
(b) xi 28 es un pico alto
(c) **X7** - 28 es un pico alto
(d) por lo menos una de las xi Y x2 es alta. **por lo tanto** existe un subgrupo cetona
Luego de determinar que la molécula contiene una subestructura en particular, la cantidad de posibles candidatos se reduce considerablemente. El equipo DENDRAL concluyó que el nuevo sistema ofrecía muchas posibilidades debido a que:

Toda la información teórica necesaria para resolver estos problemas se correlacionó, a partir de su forma generM, en el [componente predicho por el espectro] ("primeros principios"), con formas eficientes especiales ("recetas de cocina"). (Feígenbaum *et al,* 1971.)

La trascendencia de DENDRAL Ño la de ser el primer sistema *de conocimiento intensivo* que lograba fúncionar: sus conocimientos se basaban en importantes cantidades de regías para propósitos espectales. En sistemas diseñados posteriormente se incorporó también lo fúndamental de lo propuesto para el EC de McCarthy: la nítida separación del conocimiento (en forma de reglas) y la parte correspondiente al razonamiento.
Teniendo presente esta lección, Feigenbaum y otros, en Stanford, dieron inicio al Proyecto de Programación Heurística, PPH (HPP, Heurístie Programming Project), dedicado a determinar el grado con el que la nueva metodología de los **sistemas expertos** podía aplicarse a otras áreas de la actividad humana. El siguiente esflierzo de consideración se dio en el área del diagnóstico médico. Feigenbaum, Buchanan y el doctor Edward Shortliffe diseñaron el programa MYCIN, para el diagnóstico de las infecciones sanguíneas. Con un respaldo de 450 reglas, MYCIN era capaz de hacer diagnósticos tan buenos como los de un experto y, desde luego, mejores que los de un médico recién graduado. Se distinguía del DENDRAL principalmente en dos aspectos. En primer lugar, a diferencia de las reglas del DENDRAL, no se contaba con un modelo teórico desde el cual pudieran deducirse las reglas de MYCIN. Era necesario obtenerlas a partir de amplias consultas con los expertos, quienes las habían obtenido de su experiencia directa en diversos casos. En segundo lugar, las re~as deberían reflejar la incertidumbre inherente al conocimiento médico. En MYCIN se contaba con un cálculo dc incertidumbre denominado factores **de certeza** (véase el capitulo 14), que al parecer (en esa época) correspondía muy bien a la manera como las médicos ponderaban las evidencias al hacer un diagnóstico.
A éste siguieron otros procedimientos para la realización de diagnósticos médicos. En la universidad Rutgers, el proyecto de Saul Amarel *Computers in Biomedicine* (Las computadoras en la biomedicina) emprendió la ambiciosa tarea del diagnóstico de enfermedades tomando como base el conocimiento explícito de los mecanismos causantes del proceso de la enfermedad. Entretanto, grupos numerosos en el MIT y en el New England Medical Center se afanaban por encontrar un método para producir diagnósticos y tratamientos con base en las teorías de la probabilidad y de la utilidad. Su objetivo era construir sistemas que produjeran recomendaciones médicas óptimas probables. En Stanford, el método de usar reglas propuestas por los médicos gozó al principio de mayor aceptación. Sin embargo, otro sistema de razonamiento probabilista, PROSPECTOR (Duda *el al.,* 1979), provocó mucha publicidad al recomendar la realización de perforaciones de exploración en una ubicación geológica en donde se encontraron importantes depósitos de molibdeno.
La importancia del conocimiento se demostró también en d área de la comprensión del lenguaje natural. No obstante que el sistema SxiwnU de Winograd para la comprensión del lenguaje natural había suscitado mucho entusiasmo, su dependencia del análisis sintáctico provocó algunos de los mismos problemas que habían surgido en el trabajo con la traducción automatizada. Si bien era capaz de resolver los problemas de ambigúedad y de entender los pronombres utilizados, esto era posible gracias a que se le había diseñado específicamente para un área: el mundo de los bloques. Fueron varios los investigadores que, como Eugene Chamiak, estudiante becado del posgrado de Winograd en el MIT, opinaron que para una sólida comprensión del lenguaje era necesario contar con un conocimiento general sobre el mundo y del método general necesario para el uso de tal conocimiento.

En Yale, Roger Sehank (un lingúista convertido en investigador de lA) reforzó lo anterior al afirmar: "No existe eso que llaman sintaxis", lo que provocó el enojo de muchos língilistas pero sirvió para iniciar un útil debate. Schank y sus estudiantes diseñaron una serie de programas (Schank y Abelson, 1977; Sehank y Ríesbeck, 1981; Dyer, 1983) cuyo objetivo era la comprensión del lenguaje natural. El foco de atención estaba menos en el lenguaje *per se* y más en los problemas vinculados a la representación y razonamiento del conocimiento necesario para la comprensión del lenguaje. Entre los problemas estaba el de la representación de situaciones estereotipadas (Cullingford, 1981), la explicación de la organización de la memoria humana (Rieger, 1976; Kolodner, 1983) y la comprensión de planes y objetivos (Wilensky, 1983). William Woods (1973)

construyó el sistema LuNAR, que permitió a los geólogos hacer preguntas en inglés en relación con las muestras de roca que trajera la misión lunar del Apolo. LuNAR file el primer programa de lenguaje natural utilizado por personas distintas al autor del sistema para realizar una tarea. A partir de entonces se han empleado diversos programas de lenguaje natural como interfaces con las bases de datos.
El generalizado crecimiento de las aplicaciones para problemas del mundo real provocó el respectivo aumento en la demanda de esquemas de representación del conocimiento que sí fáncionaran. Se diseñó una considerable cantidad de diversos lenguajes de representación; algunos basados en la lógica (por ejemplo el lenguaje Prolog gozó de mucha aceptación en Europa, aceptación que en Estados Unidos file para la familia del PLANNER). Otros, con base en la noción de marcos de Minsky (1975), se decidieron por un enfoque un poco más estructurado al recopilar información sobre un objeto en particular y tipos de eventos y organizando estos tipos en grandes jerarquías taxonómicas, similares a las de la biología.

MYCIN (1976) Es un programa de diagnostico médico. Los factores de certeza fueron inventados para utilizarse por primera vez en el sistema experto MYCIN para la medicina, diseñado para funcionar como una solución de ingeniería y como un modelo de raciocinio humano en condiciones de incertidumbre.

MYCIN fue creado para el diagnostico de las infecciones sanguíneas, con un respaldo de 450 reglas era capaz de hacer diagnósticos tan buenos como los de un experto y mejores que los de un medico recién graduado. MYCIN tenia dos aspectos característicos, el primero las reglas era necesario obtenerlas a partir de amplias consultas con los expertos, quienes las habían obtenido de su experiencia directa en diversos casos, y en segundo lugar las reglas debían reflejar la incertidumbre inherente al conocimiento medico, MYCIN contaba con un calculo de incertidumbre denominado factores de certeza, que al parecer correspondía muy bien a la manera como los médicos ponderaban las evidencias al hacer un diagnostico.

Prospector (1979) provoco mucha publicidad al recomendar la realización de perforaciones de exploración en una ubicación geológica en donde se encontraron importantes depósitos de molibdeno, en este sistema se utilizo un sistema basado en reglas, en el que estas eran justificadas suponiendo una independencia global.

Dendral (1969) constituye uno de los primeros ejemplos del enfoque de los sistemas basados en conocimiento. Soluciono el problema de inferir una estructura molecular a partir de la información proporcionada por un espectometro de masas. El equipo DENDRAL concluyó que el nuevo sistema ofrece muchas posibilidades debido a que toda la información teórica necesaria para resolver estos problemas se correlaciono a partir de su forma general, en el [componente predico por el espectro] ("primeros principios"), con formas eficientes especiales.

La trascendencia de DENDRAL fue la de ser el primer sistema de conocimiento intensivo que lograba funcionar, sus conocimientos se basaban en importantes cantidades de reglas para propósitos especiales.

CONVINCE (1983) Primer sistema experto en donde fueron utilizadas redes de creencia.

MUNIN (1989) Sistema para el diagnostico de desórdenes neuromusculares.

PATHFINDER (1991) Utilizado en patoloqía

- **Tipos (diagnóstico, diseño, etcétera) (R)**

Tipos de sistemas basados en conocimiento

1. Diseño
1. Diagnostico

2. Lógico
3. Médico
5. Comercial
6. Con incertidumbre
7. Con razonamiento de metanivel

3.Herramientas de desarrollo

- ***Shells* de sistemas expertos (R)**

Son sistemas híbridos llamados conchas de sistemas expertos, los cuales proveen aun ambiente completo (relativamente hablando) para la construcción de sistemas expertos basados en el conocimiento.

Inicialmente cada sistema experto que era construido era creado de marcos, usualmente en LISP, pero después de esto muchos sistemas han sido construidos de esta manera, se volvió claro que estos sistemas muy seguido tenían mucho en común en especial desde que los sistemas eran construidos como un conjunto de representaciones declarativas (en su mayoría reglas) combinadas con un interprete para estas representaciones, era posible separar el interprete del dominio especifico del conocimiento y después crear un sistema que pudiera ser usado para construir nuevos sistemas expertos mediante la adición de nuevo conocimiento correspondiente al nuevo problema. Los interpretes resultantes son llamados conchas.

Ahora existen varios conchas a la venta que sirven como la base para muchos sistemas expertos que se están construyendo. Estas conchas proveen de mucha flexibilidad en la representación del conocimiento y en el razonamiento. Estas generalmente soportan reglas, marcos, sistemas de mantenimiento de la verdad y una gran variedad de sistemas de razonamiento.

Las primeras conchas de sistemas expertos proveían mecanismos para la representación del conocimiento, y explicación. Después, herramientas para la adquisición del conocimiento se le agregaron, pero como experiencia el uso de estos sistemas para resolver problemas del mundo real fue creciendo se volvió claro que las conchas de los sistemas expertos necesitaban hacer algo mejor, necesitaban hacer mas fácil la integración de sistemas expertos con otros tipos de programas. Los sistemas expertos no pueden operar como una aspiradora, en mayor medida de la que los hombres pueden. Necesitaban acceso a las bases de datos corporativas y este acceso necesita ser controlado de igual manera como se hace para otros sistemas. Una de las principales características de una concha debe ser el proveer una interfase fácil de usar entre el sistema experto que esta escrito con la concha y una mas grande, debe contar con un ambiente de programación mas convencional.

- ***Lenguajes de Programación***

En principio, cualquier lenguaje de programación puede ser utilizado. Siendo así de amplio el espectro del cual se puede escoger un lenguaje para programar un SBC, se debe considerar como factor importante de decisión, la extensión en la cual el lenguaje cubre o se adecua a los requerimientos de diseño. Atendiendo a la forma de estructurar sus instrucciones, se los puede dividir en:

- IMPERATIVOS: *PASCAL, C/C++*.
- FUNCIONALES: *LISP*.

- DECLARATIVOS: *PROLOG, CHIP, OPS5*.
- ORIENTADOS A OBJETOS: *SmallTalk, Hypercard, CLOS*.

OPS5

Official **P**roduction **S**ystem **5** (OPS5), es un lenguaje para ingeniería cognoscitiva que soporta el método de representación del conocimiento en forma de reglas. Incorpora un módulo unificador, un intérprete que incluye un mecanismo de encadenamiento progresivo, y herramientas para edición y depuración de los programas. OPS5 es un miembro de la familia de lenguajes de programación desarrollados en la Universidad Carnegie - Mellon. Varias compañías han desarrollado implementaciones comerciales de OPS5, para diferentes plataformas.

Para que un sistema computacional actúe como un verdadero experto, es deseable que reúna, en lo posible, lo más importante de las características de un experto humano, esto es:

- Habilidad para adquirir conocimiento.
- Fiabilidad, para poder confiar en sus resultados o apreciaciones.
- Solidez en el dominio de su conocimiento.
- Capacidad para resolver problemas.

Dada la complejidad de los problemas que usualmente tiene que resolver un SE, puede existir cierta duda en el usuario sobre la validez de respuesta obtenida. Por este motivo, es una condición indispensable que un SE sea capaz de explicar su proceso de razonamiento o dar razón del por qué solicita tal o cual información o dato.

Tareas

Entre las principales tareas que un SE, puede realizar, se pueden mencionar las siguientes:

- Interpretación: Análisis y síntesis (PROSPECTOR)
- Diagnóstico (MYCIN)
- Reparación, corrección o terapia
- Control: Tiempo real y tiempo diferido
- Educación.
- Simulación, pronóstico o predicción
- Supervisión
- Planificación
- Diseño

Campos de Aplicación

Múltiples son los campos de aplicación de los SE:

- Medicina
- Finanzas y Gestión
- Industria
- Electrónica, informática y telecomunicaciones
- Militar
- Educación
- Transportes
- Aeronáutica.
- Agricultura.
- Arqueología.
- Derecho
- Geología.
- Química.
- Ventas.

47

Perspectivas Futuras

En unos pocos años, se prevé que los SE tendrán una mayor difusión, se abaratará su costo, su programación y utilización serán más fáciles. Los SE estarán embebidos en diversas aplicaciones, especialmente en software de uso general, como el producido por Microsoft; y, en sistemas de supervisión y control.

En el campo de la investigación, se dará énfasis al desarrollo de SE con capacidad de aprendizaje o SE con sentido común. La aplicación de SE en simulación y control, permitirá conocer, de una manera idónea, el comportamiento de sistemas y la forma de optimizar procesos

- **Ambientes de programación (R)**

1. Programación orientada a objetos.
2. Programación dinámica.
3. Programación evolutiva.
4. Programación funcional.
5. Programación lógica inductiva ILP (inductive logical programming).

C. Interaccion humano computadora
I Aspectos Generales

1. Introducción a la interacción humano-computadora

- Orígenes y definiciones

La interacción humano-computadora, o “human-computer interaction” (HCI), se encarga de analizar y estudiar la relación que se presenta entre el ser humano con la computadora y sus sistemas, de tal forma que puedan diseñarse y desarrollarse las actividades conjuntas de forma productiva y segura, participando en el análisis y diseño de toda clase de sistemas, que pueden ser los que involucran el control de procesos y plantas productivas o generadoras de energía, el control de tráfico aéreo, los sistemas de cómputo en las oficinas, e incluso hasta los juegos de video.

Las computadoras se comenzaron a popularizar en la década de 1950, pero resultaban extremadamente complejas, de costo sumamente elevado y ocupaban grandes espacios, por lo que sólo podían ser utilizadas por especialistas en contadas universidades y centros de investigación. Sin embargo, con los avances tecnológicos, la miniaturización de los circuitos y el desarrollo de los chips de silicio, las computadoras se volvieron más confiables y fáciles de utilizar, además de reducir dramáticamente sus costos y el espacio que requerían.

Todos estos cambios llevaron a la aparición de las computadoras personales en la década de los setentas, mismas que fueron mejorando para popularizarse y provocar un cambio radical en todas las actividades del ser humano durante los siguientes años. La creación de las primeras computadoras personales representa un hecho de gran importancia para el estudio de HCI debido a que se le otorgó la capacidad de interactuar con las computadoras a una gran cantidad de individuos, y no solo a expertos programadores.

Para que las computadoras y los sistemas sean ampliamente aceptados y puedan ser efectivamente utilizados, es necesario que sean diseñados para desarrollar las tareas requeridas, de acuerdo a las capacidades y habilidades de los usuarios a quienes se destina. El equipo y los sistemas deben suplir las carencias del usuario, desarrollar las tareas peligrosas, tediosas o repetitivas que resulten inadecuadas para desarrollar por parte del humano, y al mismo tiempo debe ser tan amigable con el usuario, que este pueda operarla sin tener que conocer y comprender los intrincados procesos eléctricos, magnéticos, lógicos y matemáticos que se llevan a cabo en su

interior. No puede existir un solo diseño de computadora y sistemas que sea adecuado para todos, ya que cada usuario presenta diferentes capacidades, habilidades y niveles de conocimiento.

Se puede decir que el objetivo principal del estudio de HCI es producir sistemas seguros, útiles, efectivos, eficaces y funcionales. En base a la utilidad y funcionalidad, surge el concepto de "usabilidad", que está relacionado con crear sistemas fáciles de aprender y fáciles de usar.

El estudio de la HCI involucra muchas áreas del conocimiento humano, entre las que destacan:

Ciencias de la computación

Ingeniería

Psicología cognitiva

Psicología social y organizacional

Ergonomía y factores humanos

Inteligencia artificial

Lingüística

Sociología

Antropología

Filosofía

II Aspectos Humanos

1. Características, capacidades y limitaciones cognitivas y preceptúales

- Memoria y percepción

El proceso de la memoria supone retener y recordar conocimientos o informaciones pasadas. No se puede desligar memoria y percepción porque esas informaciones nos llegan por los sentidos. A partir de ahí son procesadas y codificadas y pasan a formar parte de la memoria. El ser humano trabaja desde distintos tipos de memoria:

Memoria inmediata sensorial: consiste en retener la información en unas décimas de segundo. Es la memoria que se usa cuando vemos una película, oímos una conversación por la calle u ojeamos escaparates.
Memoria a corto plazo: la información percibida llega a ser almacenada y la retenemos durante más tiempo que la memoria sensorial. Es la memoria práctica que se observa, por ejemplo, al recordar un número de teléfono o repasar la agenda de las cosas que tenemos que hacer inmediatamente.
Memoria a largo plazo: es la que generalmente se entiende por memoria en sí. Aquí se encuentra toda la información que queramos retener y para almacenarla necesitamos muchas repeticiones.

En el ámbito escolar, la memoria sigue teniendo detractores, pues con frecuencia se asocia el uso de memoria con alumnos poco inteligentes.
Sin embargo, no siempre es así. Para explicar esta diferencia se puede hablar de dos categorías:

Memoria repetitiva: se utiliza para acumular datos desconectados.
Memoria comprensiva: establece asociaciones entre las informaciones nuevas percibidas y los conocimientos previos. Este proceso de asociaciones estimula nuestra mente hacia esquemas de conocimientos cada vez más complejos.

Fases de la memorización
Una vez que percibimos algo, comienza un proceso que conduce a la memorización de esos datos. Este ciclo comprende las siguientes fases:

Comprensión: supone la observación a través de los sentidos y el entendimiento tanto de las partes como del todo.
Fijación: se adquiere con la repetición. Es imprescindible fijar antes de recordar una información que nos interesa
Conservación: esta fase está en función del interés, la concentración y el entrenamiento dependerá el modo en que se memoriza.
Evocación: significa sacar al plano de la conciencia los conocimientos almacenados.
Reconocimiento: consiste en la interrelación de los conocimientos nuevos y previos.

Psicología Cognitiva.

- Capacidades y limitaciones del usuario.
- Interacción en grupo (distributed cognition).
- Estructura y funciones de la organización.
- Apoyo a la toma de decisiones.

- **Procesos cognitivos**

"Modos en que trabajan nuestras mentes, como influyen en nuestra comprensión del mundo, recuerdos, percepciones, pensamientos, emociones y motivaciones guían nuestras acciones".
Actuaremos con nuestra base de cómo creemos que es el mundo.

La perspectiva cognitiva

El marco dominante que ha caracterizado la IHC ha sido cognitivo.

Cognición:
Proceso a través del cual se adquiere conocimiento incluye: entender, recordar, razonar, atención estar conciente, adquirir capacidades, crear nuevas ideas.

“Procesos referentes a nuestra interacción con los objetos del mundo que nos rodea y el cómo conocemos.”

Incluye:

- Comprensión
- Memoria
- Razonamiento
- Atención
- Conciencia (estar conciente de algo)
- Adquirir habilidades

- Crear nuevas ideas

Uno de los principales objetivos de IHC ha sido entender y representar como humanos interactúan con los sistemas en términos de cómo se transmite el conocimiento entre los dos

¿Cómo los humanos logran sus objetivos?

- Esta actividad orientada a metas comprende diferentes tareas cognitivas, entre ellas el procesamiento de la información.

- Ergonomía

La participación de la ergonomía en HCI se enfoca hacia la obtención de la mejor forma en que pueden diseñarse y utilizarse efectiva y eficientemente las computadoras, así como a la proposición de principios, guías, métodos y herramientas que permitan desarrollar y mejorar los métodos existentes de interacción con los sistemas de cómputo.

III Interfaces

1. Diseño de Interfaces

- Principios y guías de diseño

El diseño de interfaces es una disciplina que estudia y trata de poner en práctica procesos orientados a construir la interfaz más usable posible, dadas ciertas condiciones de entorno(5).

El entorno dentro del cual se inscribe el diseño de una interfaz y la medida de su usabilidad, está dado por tres factores:

1. Una **persona**.
2. Una **tarea**.
3. Un **contexto**.

El diseño de interfaces pertenece a un campo mayor del conocimiento humano, de origen altamente interdisciplinario, llamado **Human Computer Interaction**

Existen principios relevantes para el diseño e implementación de IU, ya sea para las IU gráficas, como para la Web.
Anticipación.

Las aplicaciones deberían intentar anticiparse a las necesidades del usuario y no esperar a que el usuario tenga que buscar la información, recopilarla o invocar las herramientas que va a utilizar.
(Para ver el gráfico faltante haga click en el menú superior "Bajar Trabajo")

Autonomía
La computadora, la IU y el entorno de trabajo deben estar a disposición del usuario. Se debe dar al usuario el ambiente flexible para que pueda aprender rápidamente a usar la aplicación. Sin embargo, está comprobado que el entorno de trabajo debe tener ciertas cotas, es decir, ser explorable pero no azaroso.

Es importante utilizar mecanismos indicadores de estado del sistema que mantengan a los usuarios alertas e informados. No puede existir autonomía en ausencia de control, y el control no puede ser ejercido sin información suficiente. Además, se debe mantener información del estado del sistema en ubicaciones fáciles de visualizar.

Percepción del Color
Aunque se utilicen convenciones de color en la IU, se deberían usar otros mecanismos secundarios para proveer la información a aquellos usuarios con problemas en la visualización de colores

Valores por Defecto
No se debe utilizar la palabra "Defecto" en una aplicación o servicio. Puede ser reemplazada por "Estándar" o "Definida por el Usuario", "Restaurar Valores Iniciales" o algún otro término especifico que describa lo que está sucediendo. Los valores por defecto deberían ser opciones inteligentes y sensatas. Además, los mismos tienen que ser fáciles de modificar.

Consistencia
Para lograr una mayor consistencia en la IU se requiere profundizar en diferentes aspectos que están catalogados en niveles. Se realiza un ordenamiento de mayor a menor consistencia:

1. Interpretación del comportamiento del usuario: la IU debe comprender el significado que le atribuye un usuario a cada requerimiento. Ejemplo: mantener el significado de las los comandos abreviados (shortcut-keys) definidos por el usuario.
2. Estructuras invisibles: se requiere una definición clara de las mismas, ya que sino el usuario nunca podría llegar a descubrir su uso. Ejemplo: la ampliación de ventanas mediante la extensión de sus bordes.
3. Pequeñas estructuras visibles: se puede establecer un conjunto de objetos visibles capaces de ser controlados por el usuario, que permitan ahorrar tiempo en la ejecución de tareas específicas. Ejemplo: ícono y/o botón para impresión.
4. Una sola aplicación o servicio: la IU permite visualizar a la aplicación o servicio utilizado como un componente único. Ejemplo: La IU despliega un único menú, pudiendo además acceder al mismo mediante comandos abreviados.
5. Un conjunto de aplicaciones o servicios: la IU visualiza a la aplicación o servicio utilizado como un conjunto de componentes. Ejemplo: La IU se presenta como un conjunto de barras de comandos desplegadas en diferentes lugares de la pantalla, pudiendo ser desactivadas en forma independiente.
6. Consistencia del ambiente: la IU se mantiene en concordancia con el ambiente de trabajo. Ejemplo: La IU utiliza objetos de control como menúes, botones de comandos de manera análoga a otras IU que se usen en el ambiente de trabajo.
7. Consistencia de la plataforma: La IU es concordante con la plataforma. Ejemplo: La IU tiene un esquema basado en ventanas, el cual es acorde al manejo del sistema operativo Windows.

La inconsistencia en el comportamiento de componentes de la IU debe ser fácil de visualizar. Se debe evitar la uniformidad en los componentes de la IU. Los objetos deben ser consistentes con su comportamiento. Si dos objetos actúan en forma diferente, deben lucir diferentes. La única forma de verificar si la IU satisface las expectativas del usuario es mediante testeo.

Eficiencia del Usuario
Se debe considerar la productividad del usuario antes que la productividad de la máquina. Si el usuario debe esperar la respuesta del sistema por un período prolongado, estas pérdidas de tiempo se pueden convertir en pérdidas económicas para la organización. Los mensajes de ayuda deben ser sencillos y proveer respuestas a los problemas. Los menúes y etiquetas de botones deberían tener las palabras claves del proceso.

Ley de Fitt
El tiempo para alcanzar un objetivo es una función de la distancia y tamaño del objetivo. Es por ello, que es conveniente usar objetos grandes para las funciones importantes.

Interfaces Explorables
Siempre que sea posible se debe permitir que el usuario pueda salir ágilmente de la IU, dejando una marca del estado de avance de su trabajo, para que pueda continuarlo en otra oportunidad. Para aquellos usuarios que sean noveles en el uso de la aplicación, se deberá proveer de guías para realizar tareas que no sean habituales.

Es conveniente que el usuario pueda incorporar elementos visuales estables que permitan, no solamente un desplazamiento rápido a ciertos puntos del trabajo que esté realizando, sino también un sentido de "casa" o punto de partida.

La IU debe poder realizar la inversa de cualquier acción que pueda llegar a ser de riesgo, de esta forma se apoya al usuario a explorar el sistema sin temores.

Siempre se debe contar con un comando "Deshacer". Este suprimirá la necesidad de tener que contar con diálogos de confirmación para cada acción que realice en sistema.
El usuario debe sentirse seguro de poder salir del sistema cuando lo desee. Es por ello que la IU debe tener un objeto fácil de accionar con el cual poder finalizar la aplicación.

Objetos de Interfaz Humana
Los objetos de interfaz humana no son necesariamente los objetos que se encuentran en los sistemas orientados a objetos. Estos pueden ser vistos, escuchados, tocados o percibidos de alguna forma. Además, estos objetos deberían ser entendibles, consistentes y estables.

Uso de Metáforas
Las buenas metáforas crean figuras mentales fáciles de recordar. La IU puede contener objetos asociados al modelo conceptual en forma visual, con sonido u otra característica perceptible por el usuario que ayude a simplificar el uso del sistema.

Curva de Aprendizaje
El aprendizaje de un producto y su usabilidad no son mutuamente excluyentes. El ideal es que la curva de aprendizaje sea nula, y que el usuario principiante pueda alcanzar el dominio total de la aplicación sin esfuerzo.

Reducción de Latencia
Siempre que sea posible, el uso de tramas (multi-threading) permite colocar la latencia en segundo plano (background). Las técnicas de trabajo multitarea posibilitan el trabajo ininterrumpido del usuario, realizando las tareas de transmisión y computación de datos en segundo plano.

Protección del Trabajo
Se debe poder asegurar que el usuario nunca pierda su trabajo, ya sea por error de su parte, problemas de transmisión de datos, de energía, o alguna otra razón inevitable.

Auditoría del Sistema
La mayoría de los navegadores de Internet (browsers), no mantienen información acerca de la situación del usuario en el entorno, pero para cualquier aplicación es conveniente conocer un conjunto de características tales como: hora de acceso al sistema, ubicación del usuario en el sistema y lugares a los que ha accedido, entre otros. Además, el usuario debería poder salir del sistema y al volver a ingresar continuar trabajando en lugar dónde había dejado.

 53

Legibilidad
Para que la IU favorezca la usabilidad del sistema de software, la información que se exhiba en ella debe ser fácil de ubicar y leer. Para lograr obtener este resultado se deben tener en cuenta algunas como: el texto que aparezca en la IU debería tener un alto contraste, se debe utilizar combinaciones de colores como el texto en negro sobre fondo blanco o amarillo suave. El tamaño de las fuentes tiene que ser lo suficientemente grande como para poder ser leído en monitores estándar. Es importante hacer clara la presentación visual (colocación/agrupación de objetos, evitar la presentación de excesiva información.

InterfacesVisibles
El uso de Internet, ha favorecido la implementación de interfaces invisibles. Esto significa que el usuario siempre ve una página específica, pero nunca puede conocer la totalidad del espacio de páginas de Internet. La navegación en las aplicaciones debe ser reducida a la mínima expresión. El usuario debe sentir que se mantiene en un único lugar y que el que va variando es su trabajo. Esto no solamente elimina la necesidad de mantener mapas u otras ayudas de navegación, sino que además brindan al usuario una sensación de autonomía.

- USABILIDAD

Definimos Usabilidad de un sistema o herramienta como una medida de su *utilidad*, *facilidad de uso*, *facilidad de aprendizaje* y *apreciación* para una tarea, un usuario y un contexto dado.

El peso relativo de cada una de estas medias está relacionado con el usuario, la tarea y el contexto. Por ejemplo, la facilidad de aprendizaje puede ser crucial para un producto y poco importante en otro.

Si bien no está incluida en la definición usual de usabilidad, se está comenzando a hablar de la evolución de estos factores a lo largo del tiempo. Por ejemplo, cómo una interfaz puede adaptarse al crecimiento en eficiencia y conocimientos de un usuario.

Utilidad	La utilidad es la capacidad que tiene una herramienta para ayudar a cumplir tareas específicas. Aunque esta afirmación parece obvia, es importante observar que una herramienta que es muy usable para una tarea, puede ser muy poco usable para otra, aún incluso si se trata de una tarea similar pero no idéntica. Un martillo y una maza son muy similares. Sin embargo, cada uno de ellos es adecuado para una tarea y muy poco usable para otras.
Facilidad de uso	La facilidad de uso está en relación directa con la eficiencia o efectividad, medida como velocidad o cantidad de posibles errores. Una herramienta muy fácil de usar permitirá a su usuario efectuar más operaciones por unidad de tiempo (o menor tiempo para la misma operación) y disminuirá la probabilidad de que ocurran errores. Ninguna herramienta o sistema es perfecto, pero una alta probabilidad de error puede llegar incluso a derivar en una imposibilidad de uso por falta de calificación, según cuáles sean los criterios para evaluar la herramienta o sistema[(3)]. Un caso especial de estas necesidades extremas son las herramientas de misión crítica como por ejemplo diagnóstico médico y aeronavegación. Son áreas que típicamente suelen requerir altísimos grados de eficiencia y precisión. Una falla en este tipo de aplicaciones puede tener serias consecuencias.

54

	Sin embargo, atención: la facilidad de uso no debe confundirse con la facilidad de aprendizaje.
Facilidad de aprendizaje	La facilidad de aprendizaje es una medida del tiempo requerido para trabajar con cierto grado de eficiencia en el uso de la herramienta, y alcanzar un cierto grado de retención de estos conocimientos luego de cierto tiempo de no usar la herramienta o sistema. Si bien la facilidad de aprendizaje suele tener una relación directa con la usabilidad, estrictamente hablando esto no necesariamente es así. La facilidad de aprendizaje debería ser una medida relativa, ya que hay sistemas muy complejos que no pueden ser aprendidos rápidamente. Que un software para control y monitoreo de maquinaria de producción requiera 6 meses de aprendizaje para un usuario típico, no quiere decir que es poco usable. Dada la complejidad del tema, difícilmente podría aprenderse en menos tiempo. Por lo tanto, lo importante es comparar entre varias posibles interfaces y ver cuál es la que requiere menos tiempo y/o queda mejor retenida. Si la versión siguiente, sin aumento en la complejidad del servicio brindado tomara 8 meses de aprendizaje, ahora sí estaríamos frente a un problema de usabilidad.
Apreciación	Es una medida de las percepciones, opiniones, sentimientos y actitudes generadas en el Usuario por la herramienta o sistema; una medida, si se quiere, de su seducción o elegancia. La apreciación es una medida menos objetiva que las anteriores, pero sin embargo, no menos importante. Lo importante de esta medida no es tenerla en forma absoluta sino, otra vez, compararla o analizarla en forma relativa. Esta comparación puede ser contra la competencia, contra la versión anterior del mismo producto, contra otras posibilidades que se estén tomando en cuenta. El otro punto importante respecto de la apreciación es tratar de analizar hasta donde “tiñe” el resto de las medidas. Un usuario al que no le “gusta” una interfaz puede generar mas errores, o tardar más en aprenderla. Debemos aprender a separar las medidas estrictas de las que puedan tener desviaciones debidas a una apreciación negativa.

Medidas de la usabilidad

Existen varios métodos para conocer la usabilidad de una herramienta o sistema:

1. Un análisis o evaluación heurística, o
2. un test de usabilidad.

Estos métodos, o enfoques, no son contrapuestos sino complementarios.

Estudios recientes en el área de Interfaces Humano–Computadora indican que los Tests de usabilidad muestran dónde están los problemas mientras que el análisis heurístico es más eficiente para proponer posibles soluciones.

1. Paradigmas de interacción

- **De lenguajes de comandos a manipulación directa (R, E)**

Lenguaje Natural

55

- El usuario teclea instrucciones en su idioma
- Ventaja: natural
- Problemas: Ambiguo, vago, demasiados caracteres, lento...
- Soluciones: Vocabulario controlado, abreviaturas ... --> ¡lenguaje de comandos!
- Metáfora: Autómata

Manipulación Directa

- Los objetos de interés son representados de manera visible
- Acciones rápidas, reversibles, incrementales sobre los objetos
- Ventaja: Intuitivo, sin errores, retroalimentación inmediata, sensación de control
- Metáfora: Manipulación de objetos y herramientas
- Limitante: Espacios grandes de información y acciones

Extensiones a manipulación directa

Manipulación directa, paradigma exitoso pero también con límitaciones:

a) Uso pleno de los sentidos

- rastreo ocular *(eye tracking)*
- retroalimentación de fuerza *(haptic interfaces)*
- visión esteroscópica (*ambientes virtuales*)
- interfaces olfativas (?)
- sonido, reconocimiento y síntesis de voz

b) Espacios de información de gran volumen, complejidad y dinamismo
visualización de información.

- Ambientes virtuales e inmersión

Definición de ambientes virtuales.

Las técnicas de ambientes virtuales, además conocidas como realidad virtual, han sido definidas por Kalawsky(1993) como un sistema donde una computadora genera un ambiente gráfico en tres dimensiones llamado *mundo virtual*, donde el usuario hace interacción con los objetos gráficos que residen en él, y al hacer esto, el usuario percibe una sensación de presencia dentro del mundo virtual, llamada *inmersión.* Para hacer esto posible, el usuario debe utilizar dispositivos de entrada/salida especiales. En la **Figura 1** se muestra a un estudiante utilizando un sistema de ambientes virtuales, el cual utiliza un "guante de datos", lentes para visión en estéreo, y una computadora personal (o estación de trabajo) con poder de despliegue gráfico considerable. La pantalla muestra un ejemplo de una proteína que reside en un archivo VRML. Los archivos VRML definen en sí el mundo virtual y los objetos gráficos que contiene éste. Estos archivos pueden desplegarse utilizando *plug-ins* especiales para visualizadores de páginas Web como Netscape y Explorer. Uno de los *plug-ins* más populares es Cosmo Player, de la empresa Platinum Software. Sin embargo, los archivos VRML pueden ser vistos prescindiendo de los dispositivos especiales para realidad virtual; simplemente se tiene que contar con una computadora personal con razonable poder de despliegue gráfico, un visualizador de páginas Web y el plug-in especial para despliegue de archivos VRML.

Los archivos VRML pueden ser creados a partir de archivos en formato PDB, utilizando ciertos programas convertidores especiales. Estos programas obtienen los valores de la estructura

molecular del archivo PDB y generan el código de la estructura tridimensional en formato VRML. Esto se realiza también automáticamente por medio de *plug-ins* especiales instalados en visualizadores de páginas Web.

Figura 1. Sistema de ambientes virtuales para visualizar archivos en formato VRML. (Foto: Cortesía del Laboratorio de Ambientes Virtuales, Universidad de Sussex.)

Moléculas virtuales en la enseñanza de la biología molecular

Con las técnicas de ambientes virtuales, los estudiantes pueden apreciar características abstractas de las moléculas, como lo es la visualización de estructuras tridimensionales. Además, la interacción se realiza utilizando dispositivos que permiten un control más preciso de la información en 3D, tales como los "guantes de datos" y apuntadores para 3D (García Ruiz, 1998). Hasta ahora se han utilizado solamente visualizadores de páginas Web y computadoras PC estándares con apuntadores convencionales (ratones) para enseñar biología molecular utilizando archivos VRML. Según análisis realizados por Youngblut(1998), los ambientes virtuales parecen facilitar la asimilación de conceptos científicos abstractos, y además los estudiantes pueden retenerlos en su memoria de largo plazo. Además, Youngblut comenta que los dispositivos especiales para ambientes virtuales serán cada vez más utilizados en las escuelas, principalmente debido a que los equipos son cada vez menos costosos, y más fáciles de instalar, utilizar y mantener.

Es importante mencionar que en ciertas circunstancias es preferible utilizar representaciones planas (en dos dimensiones) de la información química, tales como fórmulas, símbolos de reacciones y diagramas a seguir en los experimentos de laboratorio.

Algunas de las ventajas de usar archivos VRML son que los estudiantes pueden apreciar con gran nivel de detalle y desde cualquier ángulo la composición molecular; además, ellos pueden apreciar características físicas y químicas utilizando *metáforas visuales* implícitas en el archivo VRML. Por ejemplo, los átomos de oxígeno (un elemento altamente reactivo) se pueden mostrar en color rojo. Además, es posible transmitir vía Internet estructuras moleculares muy grandes en archivos relativamente pequeños (Brickmann y Vollhardt, 1995). Las bibliotecas digitales de moléculas virtuales han permitido a los estudiantes tener fácil acceso a información molecular organizada y automatizada Esta información ha sido también utilizada para desarrollar programas tutoriales y de modelado para la enseñanza de la biología molecular (Ferrin et al, 1992; Garcia Ruiz, en prensa).

- Navegación

Mecanismos de Navegación
Existen dos paradigmas de navegación:
El paradigma “Ir a” envía al usuario a otro contexto
El paradigma “Tráeme” ofrece al usuario información dentro del mismo contexto

Windows clásico

- - Pude ofrecer ambos paradigmas
- - Algunos botones producen navegación y otros no
- - La respuesta es inmediata
- - Mecanismos de navegación estándar definidos por guías de estilo

Web

- - Solo ofrece el paradigma "ir a", y el usuario puede perder la orientación
- - Orientado a la navegación
- - Respuesta menos inmediata condicionada por la latencia
- - Solo algunos (pocos) estándares definidos: Back, Forward.

- **Visualización (R, E)**

Visualización de Información

- Las bases de datos consisten de numerosos objetos con múltiples atributos
- Los usuarios están interesados sólo en un subconjunto
- Tipos de Datos
  - 1D: Lineales: Texto (memos, programas fuente, listas)
  - 2D: mapas geográficos, planos, tablas
  - 3D: Cuerpos, edificios, ...
  - Temporales: proyectos, historias
  - Multi-dimensionales: R^n, bases de datos relacionales
  - Árboles: taxonomías, directorios de archivos
  - Redes: relaciones complejas

Tareas del usuario:

- Ubicarse en vista panorámica
- Acercarse y alejarse (zoom)
- Filtrar
- Obtener detalles
- Relacionar
- Mantener una historia de acciones
- Extraer sub-colecciones

- **Agentes (R)**

Definición: Cualquier parte o proceso de la mente/cerebro que resulte suficientemente sencillo de entender en su tarea. Las interacciones entre grupos de estos agentes pueden llegar a producir fenómenos más difíciles de entender.

El primero de estos monigotes refleja la esencia de un agente genérico. Describiendo esta gráfica estamos definiendo, al mismo tiempo, a un agente genérico (el ser humano, el robot, el agente de soporte lógico (software) - agente llamado softbot, haciendo un mix de software y robot [en castellano el mix sería logibot] - son casos particulares de agentes. Las percepciones y las acciones del logibot digital son cadenas de ceros y unos.

El agente puede ser:

- autónomo:
  Un agente autónomo es un sistema anidado y parte integrante de un ambiente (environment) y que detecta o percibe (percepts) datos ambientales, momento a momento, y actúa sobre él con la intención de usar (actions) esos datos para su propia tarea (task) o *agenda*, afectando así lo que va a detectar en el futuro, sin intervención de terceras partes (basado en Franklin y Greasser, 1996).
- racional, que hace lo correcto, siendo "ideal" si hipoteticamente lo consigue del todo.
- inteligente, siendo aceptable cualquiera de las numerosas definiciones de inteligencia, por ejemplo, capaz de aprender/adaptivo.
- activo, o sea que tiene en cuenta la gama de acciones abiertas a la elección y sus recompensas o penalidades.
- pro-activo donde más allá de actuar en respuesta a su mundo, lo hace tomando la iniciativa.
- meta-intensivo (orientado a metas), que elige las acciones que lo conducen a la meta, que intenta lograr prioritariamente.
- modelo-intensivo, que tiene internalizado un cierto modelo del mundo y lo respeta en sus acciones.
- utilidad-intensivo, que se esfuerza por obtener un máximo de "felicidad".
- social-intensivo o de comunicación, capaz de comunicarse en algun lenguaje de comunicación para agentes comprensible para otros.
- de planificación, parecido al agente problema-intensivo (orientado a resolver problemas y tomar decisiones)
- *softbot* o sea robot basado en un software (como los robots bidimensionales de Beer que mimetizan a un invertebrado).
- reactivo cuando percibe perturbaciones en su mundo y responde a esa percepción de una manera actualizada (adaptada).
- reflejo o tropista, que responde de inmediato y en forma bien definida a un tropismo, a una percepción alarmante o beneficiosa (así un agente para el heliotropismo se orienta al sol).

Por "programa de agente" se entiende la vía que usa desde la percepción de su input y la acción de su output, pasando por estados internos de acuerdo con la función que cumple ese agente.

Las clases de Agentes

Tres vistas(opiniones) principales de agencia pueden ser distinguidas en la literatura. Primero, los informáticos e ingenieros de software han usado a agentes como una abstracción para conceptuar, diseñar y poner en práctica sistemas complejos. Esta clase de agentes se mencionará como *agentes de programador*. Segundo, los agentes pueden ser vistos como entidades que autónomamente emigran que actúan de parte de nodos de red en un ambiente distribuido. Esta clase será llamada *agentes de red*. Finalmente, han propuesto a agentes como una abstracción para usuarios finales para actuar recíprocamente con sistemas de ordenador. Esta vista(opinión) define la clase *agentes de usuario* llamados.

- **Agentes de Programador**

Los informáticos se ocupan del hardware complejo y entidades de software. Los mecanismos de abstracción solieron describir el comportamiento de estas entidades varían extensamente. Los organigramas, organigramas de datos y modelos de objeto han sido artículos *de mechanis*tic populares para visualizar la operación de autómatas. Como la complejidad de sistemas de software ha aumentado, *animistic* abstracciones han sido encontrados útil en la explicación del comportamiento de sistema. Considerando la familiaridad de los programadores con las entidades verdaderas mundiales que exponen *las intenciones*, atribuyendo la autonomía y la intención a componentes de sistema suministran un modo conveniente de describir, analizar y predecir la operación de sistemas enteros sin la remota preocupación(interés) sobre sus detalles internos.

 59

Esto es la perspectiva adoptada por Wooldridge y Jennings [1995a] para definir a un agente como *un sistema coherentemente descrito por la postura intencional* (en la referencia a Dennett [1987]). El énfasis aquí es en el modelado *de los procesos* realizados por el ordenador para la ventaja *del programador*. Como esta vista(opinión) madura, un acercamiento al desarrollo de software probablemente va a surgir bajo la bandera *de ingeniería de software orientada por agente*.

- **Agentes de Red**

La comunicación en ambientes calculadores distribuidos a menudo confía en *la llamada de procedimiento remota* (*RPC*) protocolos. En RPC, un proceso de cliente invoca una acción remota por enviando a un mensaje (la inclusión de argumentos de datos) a un proceso de servidor y bloques antes de que una respuesta sea recibida con los resultados de las operaciones solicitadas [RPC 1988]. En aquel punto, llámese los resultados son extraídos y los curriculums vitae de ejecución del llamador. El tráfico de red puede aumentar dramáticamente como el número de clientes y el número y la complejidad de aumento de tareas.

Conecte una red a agentes ofrecen una alternativa a RPC. La comunicación de ordenadores puede suministrar el código de los procedimientos para ser ejecutados por máquinas remotas, permitiendo al cómputo emigrar al sitio donde los datos residen. Los mensajes enviados a través de la red incluyen instrucciones para ser ejecutadas y datos que reflejan el estado corriente del procedimiento. Cada procedimiento de ejecución es considerado un agente autónomo que puede decidir emigrar a un anfitrión diferente en la red según sus exigencias de tarea. Un agente puede comenzar a ejecutar sobre una máquina y, cuando otros recursos son necesarios, el salto a la máquina que tiene(ha) aquellos recursos, siguen realizar su cómputo allí y, si necesitado, emigrar otra vez. Realizando su trabajo en un anfitrión de encubrimiento, el proceso de emigración es considerado como un agente que representa los intereses de la máquina de enviar. Los agentes de red también se han mencionado como *agentes móviles* [1996 Blanco], *agentes itinerantes* [el Ajedrez et al. 1995] *y agentes transportables* [Color gris et al. 1996]. También han llamado el modelo móvil basado por agente para la informática distribuida *el programa remoto* [1996 Blanco].

La realización a agentes móviles requiere que ordenadores participantes estén de acuerdo sobre el juego de las instrucciones que pueden ser enviadas sobre la red y ejecutadas por recibiendo a anfitriones. Estas instrucciones constituyen *un lenguaje de programación de agente móvil*. Un motor capaz de interpretar tal lengua debe residir sobre cada anfitrión participante. Un acercamiento natural al suministro de un ambiente seguro es de poner en práctica agencias como intérpretes capaces de supervisar instrucciones siendo(estando) ejecutadas por agentes e instalaciones que proveen de la migración de agente y la recepción. Si agentes móviles deben comunicar con el uno al otro, las agencias deben suministrar mecanismos para agentes para encontrar y cambiar datos también. Las puestas en práctica del modelo de agentes móvil solamente(justo) comienzan a aparecer. Los proyectos notables en el desarrollo de instalaciones de agente móviles incluyen *al Agente Tcl* [el Color gris 1995], *Ara* [Peine 1995], *TACOMA* [Johansen et al. 1995] *y Guión para la tele* [1996 Blanco]. En todos los casos, los motores que pueden ejecutar las declaraciones en una lengua interpretada han sido puestas en práctica y las declaraciones que permiten a programadores transparentemente especificar la migración de agente y las reuniones de agente son centrales construye.

- **Agentes de Usuario**

Los usuarios finales coherentemente han atribuido la autonomía y otros aspectos de agencia a sistemas de ordenador, hasta cuando esta vista(opinión) explícitamente no ha sido promovida por interfaces de usuario. Por ejemplo, en un experimento conducido por Friedman [1995], el ochenta y tres por ciento de los usuarios participantes atribuyó capacidades de toma de decisiones autónomas o intenciones a programas del ordenador. Un natural siguiente(próximo) intervienen el desarrollo de interfaces de ordenador humano debe hacer la abstracción de agente explícitamente disponible a los usuarios finales. Tal como los reveladores de software usan al programador y

conectan una red a agentes como abstracciones para enfrentarse con la complejidad de diseño de sistema, los usuarios finales mejor pueden ocuparse de la complejidad de sistema por viendo programas como animistic entidades

- **Interfaces multiusuario (*groupware*)**

Groupware: Software para grupos de usuarios

- Cambio de paradigma: de IHC a Interacción Humano-Humano mediada por computadora
- Dificultades adicionales al diseñar groupware
    - Aún no llegamos a conocer al individuo, menos al grupo
    - Roles múltiples
    - Factores sociales, motivacionales, políticos
    - Difícil observar grupos en acción
- Factores que hacen que el groupware falle:
    - Requiere de más trabajo y quien lo hace no es quien más se beneficia
    - Es "subversivo" (socava reglas y estructuras)
    - Requiere aun mayor flexibilidad (no manuales de procedimientos directamente, espacio para la improvisación)

Dimensiones de groupware

	mismo tiempo	**diferente tiempo**
mismo lugar	ej. juntas presenciales electrónicas	ej. trabajos por turno, salas de enfermeras
diferente lugar	ej. video conferencia, MUD's	ej. email, usenet BBSs.

Groupware: Algunos aspectos importantes

- Interfaz:
    - Representación de objetos compartidos
        - Metáforas
        - WYSIWIS estricto vs. relajado
        - Presencia y pertenencia al grupo (awareness) - *telepointers*
        - Contacto visual
- Desarrollo de groupware
    - Control de concurrencia
    - Ancho de banda
    - Toolkits especializados

 61