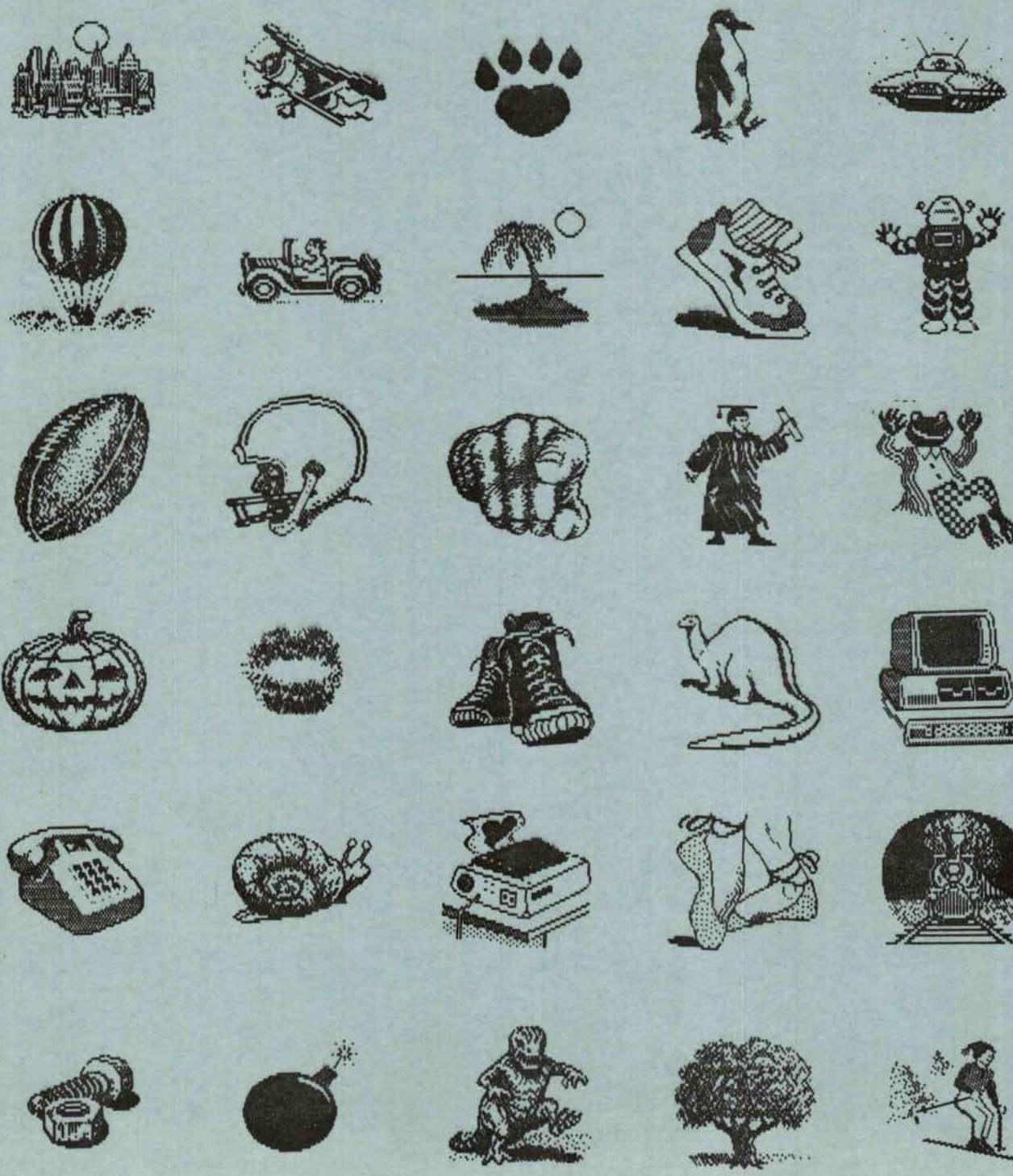
TRAMA

BOLETIN DE INTELIGENCIA ARTIFICIAL NUMERO 1 OCTUBRE 87



SUMARIO

DISEÑO DE EDICION : JAVIER SANCHEZ

EDITORIAL

A pesar de los problemas, siempre es posible salir adelante con perseverancia y decisión. Esto viene al caso de una demostración de esta tesis que el equipo que ha desarrollado la revista ha tenido ocasión de llevar a cabo. El tema es el ya tristemente famoso de las ayudas oficiales. Lo cierto es que nosotros nunca nos lo creímos mucho, y estaba en nuestra mente el llegar a la autonomía y autosuficiencia en el plano económico. Pero como dijo una vez el insigne filósofo Jose Luis Artola, nunca debe dejarse una tecla sin tocar. En eso estuvimos algún tiempo, pero en cuanto llegamos al párrafo donde dice "... por triplicado ... y en un plazo no superior a dos años recibirán un formulario en el que harán constar..." tuvimos que tomar una decisión ante una de las dos opciones siguientes:

1.-Desarrollar nuestro pobre conocimiento acerca de los vericuetos oficiales.

2.-Optar por una via de autofinanciación asaz inquietante.

El resultado de tan terrible disyuntiva la pudo apreciar el avisado lector que reparó en el precio de nuestro número anterior. Efectivamente, hemos optado por la independencia total,con los riesgos que conlleva, aunque eso sí, con mayor capacidad de crítica hacia todo lo establecido, sin temor a la "retirada" de una ayuda inexistente.

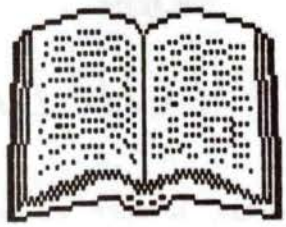
Por otro lado, y tenemos interés en que sea éste el tema clave, esperamos que el lector haya reparado también en que dedicamos el máximo esfuerzo a formar una revista de calidad, tanto en su contenido como en su presentación, con la ¿secreta? esperanza de verla un día en los kioskos ocupando un puesto honorable entre las revistas técnicas.

Reconocemos que el primer número tuvo una acogida que rebasó nuestras más pesimistas previsiones y esperamos que la tirada (muy superior) de éste siga siendo insuficiente para cubrir la demanda. Es una especie de opción de revalorización de coleccionista como premio a los que han prestado su apoyo pecuniario a estos siempre heróicos primeros números.

Y ya sin más preámbulo, os cedo el resto del ejemplar. Estoy seguro de que lo disfrutareis casi tanto como yo.

Julian Inza



ENTREVISTA

Alejandro Vaca Berdayes.
Su mision ha sido la de coordinar el proyeccto de sistemas expertos, que actualmente esta desarrollando la C.T.N.E.. Es sobre este proyecto, sobre el que hablaremos durante la entrevista.

P:¿Cómo surge la idea de realizar un programa de sistemas expertos en Telefónica.?
R:En 1985 se empiezan a producir por parte de diferentes departamentos, contactos con centros que promueven la impartición de la inteligencia artificial, lo cual nos hace ver la necesidad de incorporar a la empresa esta nueva tecnologia. En el mes de octubre se firma el convenio entre Sperry y el ministerio de industria con el objetivo de instalar en nuestro pais su centro de I.A. para Europa. Esto hace ver la posibilidad de llegar a un acuerdo para la introducción de la I.A. en Telefónica. Dicho acuerdo se alcanza en Febrero de 1986.
P:¿Cuales son los planteamientos iniciales por parte de telefónica.?
R:Evidentemente el primer objetivo es evitar la introducción de forma puntual y dispersa de estas nuevas tecnologias por parte de los diferentes departamentos interesados en el tema. Se trata de dar un primer empujón global, bastante ambicioso y que reuna todos los esfuerzos.
P:Una vez alcanzado este punto,¿cuál es el siguiente paso a dar.?
R:Una vez admitida la necesidad de un programa de sistemas expertos, se realiza una amplia prospección dentro de la Compañia para la detección de los posibles temas a abordar. Como conclusión de este trabajo se obtienen 20 posibles puntos de aplicación para este proyecto. Por supuesto es imposible realizar los 20 proyectos a la vez, por lo que se hace una primera selección que reduce los proyectos iniciales a 4. Estos cuatro proyectos piloto son los siguientes: Sistema de Gestión Dinámica de Red. Monitor de Aplicaciones Informáticas. Sistema de Analisis de la Demanda de Servicios. Sistema de Gestión de Deuda Externa.
P:¿En que fase de desarrollo se encuentra actualmente el proyecto.?
R:Básicamente el momento actual del proyecto es el de la construcción del equipo que va a realizar estos trabajos, buscando las empresas o universidades que puedan colaborar. Tambien se esta intentando buscar la forma de irradiar a toda
la compañia el tema de la I.A.. En el tema de la informática suele ocurrir que cada área organizativa de la propia entidad genera sus propias herramientas y cubre sus necesidades con esas herramientas. Cuando la organización se quiere dar cuenta es muy difícil la integración de las distintas herramientas generadas. Esto es lo que se pretende evitar.
P:¿Cuál será el costo de estos cuatro proyectos.?
R:Primero hay que tener en cuenta que este proyecto constituye la entrada en un nuevo campo, lo que obliga a la obtención de un éxito inicial, que dé confianza a la empresa. Por otro lado se trata de temas que a pesar de no exigir un elevado número de personas trabajando, si imponen un largo periodo de desarrollo. Estas dos razones nos han inducido a seleccionar de entre los cuatro proyectos señalados, únicamente dos para un desarrollo inicial con mayores garantias de éxito. Los proyectos seleccionados han sido los dos primeros, quedando los otros dos pendientes de las disponibilidades materiales con que se cuente.
P:Pero este éxito, ¿cómo se mide?. ¿Cuál es la rentabilidad de un S.E.?
R.Evidentemente la rentabilidad vendrá en función de la ubicación del propio sistema. No hay una norma general que indiqué cuándo un S.E. es rentable y cuándo es preferible la utilización de expertos humanos, ésto debe estudiarse en cada caso particular.
P:¿Qué plazo de finalización tienen estos proyectos.?
R:Un proyecto de este tipo tiene una fecha de iniciación y otra de puesta en marcha, a partir de ese momento el proyecto puede empezar a funcionar, lo cual no quita para que posteriormente se pueda seguir modificando y perfeccionando. Ocurre igual que si contrataras a una persona, primero la tienes en periodo de practicas y a

P:Cambiando de tema, ¿qué condiciones debe tener un tema para poder ser abordable por un S.E.?
R:Las fronteras son realmente difusas. En principio debe ser tan estrecho como para poder hacerlo, ya que queremos pasar el conocimiento nuestro a una máquina y si el tema es demasiado amplio acabaremos por perdernos. Pero también debe ser lo suficientemente ancho como para que tenga interés su realización. Una vez que se ha realizado una primera fase del proyecto y que ésta se ha puesto a funcionar, siempre estamos a tiempo de seguir desarrollando y ampliando nuestra idea inicial para asi conseguir un mayor campo de trabajo, pero como ya he dicho no conviene excederse en los planteamientos iniciales. Esa ha sido una de nuestras preocupaciones en este estudio.
P:¿Podrías profundizar un poco más en alguno de los dos proyectos seleccionados.? ¿Por ejemplo en el monitor de aplicaciones informaticas.?
R:Bueno éste era uno de los posibles temas seleccionados. concretamente en este caso, se trata de emplear la I.A. para la creación de prototipos previamente a la realización de los mismos. En cualquier proyecto que se este realizando y por supuesto también en un proyecto de tipo informático, se parte de un diseño general y luego se va avanzando hasta llegar a su total desarrrollo. Cuando lo has desarrollado por completo y llegas abajo, te das cuenta de elementos que no casan, de incompatibilidades de duplicidades y de toda una serie de problemas. Resulta que tu ya has desarrollado todo eso, destinándole una gran cantidad de recursos y te ves en la necesidad de replanteártelo. La I.A. hace posible la creación de unos prototipos muy fácilmente desarrollados y en muy corto tiempo que te permiten conocer los resultados finales de tu diseño sin necesidad de construirlo. Volviendo a lo que decíamos antes sobre la mayor o menor amplitud de un proyecto, en este caso por ejemplo hemos partido de la idea de utilizar nuestro monitor para diseños informáticos pero nada nos impide que una vez alcanzado este primer objetivo y puesto en funcionamiento el monitor se pueda hacer una segunda fase que permita su empleo en el diseño de circuitos o de redes telefónicas. Es evidente que si hubieramos tratado de conseguir ésto como objetivo inicial tendríamos muchas más posibilidades de fracasar que de esta otra manera.
P:Bueno vamos ha hacer las dos últimas preguntas para finalizar esta conversación. En primer lugar una pregunta para predecir el futuro: ¿se llegará en nuestro pais a alcanzar algo que podríamos denominar la red integrada de sistemas expertos.? ¿Podría llegar a pedir a la C.T.N.E. en lugar de un servicio de teletexto, un servicio de sistemas expertos en el que yo pudiese encontrar ayuda sobre un tema que a mi me interese.?
R:Esta claro que si lo que ocurre es que en estos momentos la tecnología que tenemos disponible no nos permite alcanzarlo. Aparte de la necesidad de esa tecnología también sería necesario el tiempo que nos permitiése ir cubriendo todos los posibles temas de interés, pero sin ninguna duda se va a llegar a eso. La I.A. permite por primera vez la trasposición del conocimiento a una máquina y la realización de inferencias a partir de esos conocimientos, este hecho es tan importante que no sólo va a revolucionar el mundo de la informática sino también el mundo económico y social. Hace falta tiempo, pero igual que ahora tenemos bases de datos relacionadas entre sí, llegaremos a tener bases de conocimientos que se interrelacionen. Primero habrá que crear estas bases dentro de parcelas pequeñas para luego irlas ampliando hasta cubrir todos los campos. Esto requerirá algún tiempo tanto por tecnología como por necesidades económicas pero sin duda se alcanzará.
P:Los planteamientos que se han estado comentando sobre este proyecto son absolutamente ambiciosos. Parece lógico preguntarse si realmente va a existir la infraextructura y el apoyo tanto humano como económico para conseguir sacarlos adelante y llegar hasta las últimas consecuencias.
R:Como ya he dicho antes nuestra compañía está muy interesada en todo lo que es la nueva tecnología y es una empresa con vocación de estar en punta. Recientemente hemos realizado unas jornadas y alguien que creo recordar era precisamente de la escuela de Telecomunicaciones decía que Telefónica era más que una empresa, como el Barça. Efectivamente nuestra vocación no se para en ser una simple empresa dedicada a dar un servicio público como es el teléfono. Nuestra idea va más allá y en ese sentido nosotros tenemos una ilusión muy grande por lo que son las nuevas tecnologías y en este caso concreto por meter la I.A dentro de la casa. En ese sentido la empresa va a facilitar todos los medios necesarios para seguir adelante con ello, los resulados obtenidos tendrán su influencia pero sin duda alguna existe una sensibilidad muy importante por estos temas.

* * * * * * * * * *

Alejandro Vaca Berdayes.

Concedió esta entrevista al grupo de I.A. de la rama de estudiantes del I.E.E.E. con motivo de la conferencia-coloquio que dió en la E.T.S.I. de TELECOMUNICACION durante la celebración del SATELEC 87.

Es jefe de la Sección de Mecanización, Departamento de Promoción y Desarrollo Industrial de la C.T.N.E.

continuación empieza a trabajar, pero esa persona seguirá aprendiendo con el desempeño de su labor. Esto mismo es aplicable a un S.E..

P:Entonces, ¿cuál es este primer plazo, cuando empezaran a considerarse operativos los sistemas?

R:Nos gustaria obtener una primera respuesta en doce meses.

P:¿Qué tipo de ayuda habéis recibido o pensáis recibir de otras empresas o instituciones.?

R:La idea inicial es buscar la colaboración de alguna empresa que tenga ya productos dentro del campo de la I.A., asi como contar con la Universidad tanto nacional como extranjera. De esta forma la empresa nos va a proporcionar una cierta garantia de obtener resultados positivos. Por otro lado la colaboración con la universidad va a permitir no descolgarnos de las constantes innovaciones que se producen dentro de este campo.

P:Hablando de innovaciones, ¿a qué nivel esta la C.T.N.E. y en general nuestro pais en lo que se refiere a I.A. dentro de Europa.?

R:En los congresos en que yo he participado, una gran parte de las ponencias presentadas correspondian al mundo universitario, hay pocas experiencias en lo que se refiere a realizaciones ya concluidas. En nuestro pais por ejemplo, nos encontramos con IBERIA que tiene en la actualidad un S.E. en funcionamiento y otro en preparación. En general dentro del contexto europeo no nos encontramos muy retrasados. Todavia estamos a tiempo si nos ponemos a trabajar ya a traves de empresas como la Banca que tambien esta realizando programas de este tipo, o de

cualquier otra que todavia no se haya introducido en el tema. Dentro del terreno del Software, si se ponen los recursos adecuados podemos llegar a ser un pais en punta del panorama internacional.

P:Volviendo a lo que se refiere al proyecto de Telefónica, ¿se ha empleado algun tipo de referencia de proyectos similares realizados en otros paises.?

R:Realmente no hemos conseguido ninguna referencia anterior en trabajos de este tipo. Hemos tenido conocimiento de que quizá en Gran Bretaña existiera un proyecto similar pero no hemos podido obtener ninguna información al respecto. Nos hemos limitado a seguir una metodología que considerábamos válida para nuestro caso ya que careciamos de experiencias en las que basarnos.

P:¿Cómo ve el futuro de los sistemas expertos tanto en Telefónica como en en el resto de las empresas que anteriormente ha citado.?

R:La irradiación de estos temas tanto en Telefónica como en el resto de las empresas se va a realizar aunque los primeros intentos sean malos. A pesar de que los dos proyectos de los que estamos hablando no obtengan los resultados esperados esta nueva tecnología va a ser asimilada por las empresas, más pronto que tarde. Nosotros estamos convencidos de que esta iniciativa va a ser beneficiosa no sólo para la empresa sino también para el pais y por ello no hay perspectivas de desánimo. Es más, esta ilusión que nosotros tenemos estamos tratando de contagiarla a muchas empresas y a la administración pública que también está desarrollando tres proyectos de S.E.. En realidad se está desarrollando más de lo que parece, pero nosotros seguimos manteniendo el cliché de pais retrasado tecnológicamente,aunque en este tema en concreto ésto no sea tan cierto.

P:Bueno, y ahora la pregunta capciosa que siempre acaba saliendo en conversaciones sobre este tema. ¿No cabe la posibilidad de que estos sistemas expertos usurpen los puestos de trabajo que actualmente estan ocupando los expertos humanos?

R:Para contestar a esta pregunta emplearé un ejemplo que me parece bastante ilustrativo. Cuando se inició la andadura de la telefonia, las conexiones entre dos abonados se realizaban a través de una operadora. En ese nivel de tecnología y con la cantidad de abonados que existen en la actualidad, sería imposible realizar todas las conexiones, no habría suficiente gente en la tierra para atenderlo. Es evidente que el nivel de tecnología ha avanzado. Cuando se produjo la sustitución de los operadores por las centrales automáticas se podia haber planteado una pregunta similar sobre la posibilidad de que la nueva tecnología restara puestos de trabajo. Yo pienso que la sociedad humana tiene capacidad para absorver esos avances tecnológicos y que dichos avances son positivos.

P:¿Se ha realizado algún estudio más profundo sobre estas consecuencias.?

R:Más que enfocado a las posibles influencias negativas, FUNDESCO ha realizado un estudio en el que se hace patente la falta de personal preparado en tecnologías de la información y que no va a ser capaz de cubrir la demanda que se producirá en un futuro no demasiado lejano, dentro de estas tecnologías.

ARTICULO:

Inteligencia artificial en la informática de gestión

Dos áreas de interés en Inteligencia Artificial prometen cambiar radicalmente los usos del ordenador y los modos de acceso al mismo. La gestión del conocimiento humano que realizan los Sistemas Expertos y el canal de acceso a la "infoesfera" que proprociona el reconocimiento del lenguaje natural, recuerdan cada vez más el semblante de los ordenadores de las películas de ciencia ficción. ¡Lo sorprendente es que se trata de ciencia-ficción a nivel de PC!

Hasta la aparición del ordenador personal, la informática de gestión estaba dominada por la omnipresencia del lenguaje Cobol. Realizar la más mínima operación financiera requería comprar un ordenador de respetables dimensiones en lo físico y económico, y programar la aplicación concreta... ¡cómo no, en Cobol! Con el advenimieto del ordenador personal las cosas cambiaron bastante. No sólo era posible resolver tareas cotidianas con este tipo de ordenadores o pequeñas redes de ellos, sino que empezaron a surgir paquetes destinados a resolver la más particular de las aplicaciones. Los escasos medios iniciales de almacenamiento masivo de datos dieron paso a soportes magnéticos de alta capacidad y seguridad. Comenzaba a surgir la ifgura del ordenador como esclavo tonto al que encomendar las tareas más laboriosas y tediosas.

Pero gestionar una gran cantidad de información lleva consigo un esfuerzo mental considerable. El problema es más cualitativo que cuantitativo: Tan importante o más es almacenar mucho, como recuperar la información de forma sencilla. Esta es una máxima de primer orden en una ciencia que se ha propueto hacer partícipe a

* * * * * * * * * *

toda persona de las facilidades que es capaz de ofrecer. Mientras la capacidad mental del "homo informaticus" no crezca a mayor ritmo, la única forma de llevar a la práctica este objetivo es permitirle la interacción con la máquina en el lenguaje que mejor conoce: el lenguaje natural.

No es exagerado afirmar que el gran motor de la avanzada sociedad actual es la economía. La creciente

complejidad de la esfera económica y la cada vez mayor cantidad de informacion que genera y es preciso procesar, se presta especialmente bien a un tratamiento informático. Los Sistemas Expertos permiten vislumbrar la mutación que convertirá al esclavo tonto de los años 60 y 70 en esclavo inteligente de los 90, capaz de poner en manos de las personas que han de tomar decisiones delicadas los consejos y análisis que hasta ahora son realizados por equipos de especialistas.

A continuación se dará un repaso a los más avanzados programas que, más o menos dentro del ámbito de la Inteligencia Artificial y centrados en los dos aspectos mencionados de los Sistemas Expertos y reconocimiento del lenguaje natural están empezando a hacer realidad las ideas reflejadas en los párrafos anteriores. Ha de tenerse en cuenta que mucho de lo que aquí se diga es aplicable a otros campos de aplicación profesional, pero estimamos que centrándonos en lo económico ayudará a que muchos lectores se sientan identificados con los problemas que vamos a esbozar.

De la hoja de cálculo al programa inteligente.

Si hay un producto informático que ha adquirido plena popularidad en estos años ha sido la hoja de cálculo. Desde la aparición de VisiCalc en 1978, hasta los paquetes integrados como Lotus 1-2-3 o Symphony en los que aparecen las últimas versiones de esta aplicación los usuarios que han rentabilizado las filas, columnas y fórmulas de este tipo de programas se cuentan por millares.

La idea original de los padres de VisiCalc -Dan Bricklin, estudiante de economía en Harvard por aquel entonces, y Bob Frankstone, amigo del primero -, era la creación de un programa para realizar previsiones financieras con la ayuda del cual fuera posible contestar a preguntas del tipo ¿que pasaría si...?, objetivo plenamente conseguido a la vista de las cifras de ventas.

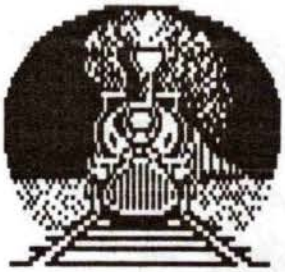
Si ya se dispone de un programa capaz de explorar el futuro,¿qué necesidad se tiene de buscar nuevas soluciones? La pregunta tiene una respuesta evidente al analizar las limitaciones de las hojas de cálculo .Para su utilización,es necesario tener un modelo potente del sistema que se está estudiando,en el que sea posible identificar una serie de variables cuyas fluctuaciones definan por completo los posibles estados del sistema.Una hoja de cálculo es simplemente el instrumento que permite plasmar este esquema sobre la pantalla del ordenador,del que se aprovecha su enorme capacidad de cálculo para realizar las cuentas que de otra forma habría que hacer a mano.

Ya se ha mencionado la enorme complejidad de los sistemas económicos.El desarrollo de un modelo que los represente es una tarea de gigantes en cuanto el caso en estudio comienza a alejarse de lo trivial.Frente a la naturaleza puramente estática de las hojas de cálculo,los Sistemas Expertos poseen el calificativo de dinámicos.Estos programas manipulan la información que se les suministra para dar resultados que sorprenden a profesionales con abundanmte experiencia y,lo que es más importante,son capaces de llegar a conclusiones concretas y de sugerir soluciones a problemas que hasta el momento eran de difícil e incluso imposible tratamiento informático.

Siempre refiriéndonos al campo económico,los ordenadores han sido hasta la fecha excelentes herramientas para el proceso de información regular y perfectamente estratificada en grupos.Fuera de esto,se convertían en gigantescos inútiles.Con la llegada de los Sistemas Expertos es posible atacar problemas que se caracterizaban por su elasticidad y falta de fronteras bien definidas,justo el tipo de problemas que se encuentran más a menudo y que son más difíciles de resolver.

El experto entra en escena.

En la actualidad es posible encontrar ya un buen número de aplicaciones que pueden recibir el calificativo de inteligencia artificial,capaces de brindar una apreciable ayuda en la toma de decisiones.Una consulta a este tipo de aplicaciones toma el mismo cariz que si la realizara un auténtico experto en la materia :Al proponer ciertos hechos o situaciones,el ordenador genera un consejo y es capaz de explicar su linea de razonamiento.Este tipo de programas



no tienen su campo de actuación restringido a lo puramente económico,sino que es posible utilizarlos en cualquier tarea de cierta complejidad y cuya resolución pueda ajustarse al esquema mencionado.

Los productos que se encuentran en el mercado pueden dividirse en dos grupos,dependiendo de cómo adquieren el conocimiento que les permite llegar a sus conclusiones.En el primero de ellos,tal conocimiento se adquiere mediante la introducción de una serie de reglas del tipo SI...ENTONCES,en las que se encuentra codificado el saber que se posee sobre un área determinada.En el segundo,el usuario presenta al sistema una serie de ejemplos y será el programa el que deduzca la regla asociada a ellos.Tras una serie de sesiones de aprendizaje,nos encontramos en la misma situación que con el primero de los tipos.

Los sistemas basados en ejemplos son más apropiados para aquellas situaciones en las que factores y situaciones son estables y repetitivos,apareciendo sólo cambios en los valores de las variables que definen el sistema,y existiendo,además,un reducido número de posibles resultados finales.Por contra,los sistemas basados en reglas se prestan a resolver situaciones con espacio de solución más amplios y en las que los factores cambien temporalmente;puede afirmarse,en definitiva,que poseen una potencia y flexibilidad mayores.

Esta última afirmación no significa que haya que elegir siempre un programa basadoen reglas.En primer lugar, hay muchos problemas sencillos que pueden ser resueltos por programas que aprenden de ejemplos.Además, la tarea de extraer el conocimiento de un experto en un área determinada y codificarlo en forma de reglas del tipo comentado, no es ni mucho menos una empresa trivial en la mayoría de los casos.Las ayudas que proporcionan los programas basados en la exposición de ejemplos pueden resultar muy provechosas, aunque tampoco sea nada trivial la selección de una serie de ejemplos significativos.

Dígaselo en lenguaje natural.

El otro gran campo donde la inteligencia artificial tiene mucho que decir, en uyn entorno económico y de gestión, se concreta en el reconocimiento del lenguaje natural.

De forma simultánea al crecimiento de una base de datos aparecen problemas asociados a su gestión.Estos problemas casi nunca están relacionados con limitaciones físicas de capacidad de memoria, los cuales se resuelven normalmente con inversiones económicas adicionales para aumentar las posibilidades hardware del sistema.Las auténticas complicaciones emergen del lado del software.

Tener almacenados cientos de miles de datos sólo es interesante si el acceso a los mismos se puede realizar de forma fácil y rápida.En muchas ocasiones, el rendimiento de una base de datos es sólo una fracción de lo que en realidad podría dar de sí, debido a que son muy pocos los usuarios dispuestos a aprender las sutilezas del lenguaje de acceso a la misma.Además, este tipo de lenguajes suelen caracterizarse por obligar al usuario a tener en la mente ciertos conocimientos de la estructura interna del sistema de almacenamiento, una necesidad que aleja al usuario casual del acceso a la información.

El reconocimiento del lenguaje natural es uno de los campos de actuación de la I.A. que más

atención recibe por parte de los especialistas en bases de datos.Si algún día fuera posible realizar las consultas a una de tales bases en los mismos términos que se realiza una llamada telefónica al servicio de información se podría decir que la informática estaba lista para inundar realmente la vida cotidiana de las personas.Aunque a los familiarizados con programas como dBase III les parezca de lo más natural expresar cosas como "LIST ALL LIBROS,AUTOR,EDITOR,AÑO FOR AÑO>1980", es muy seguro que la mayoría de los habitantes del planeta están dispuestos a jurar que no se expresan a diario de esta forma.

Desgraciadamente, los muchos años de investigación en este aspecto y los escasos resultados obtenidos ponen de manifiesto la complejidad del problema.Pero esto no es óbice para que existan en el mercado un buen número de productos que permiten la comunicación con el ordenador en unos términos más flexibles que pulsar teclas de función o escribir crípticos comandos.

El acceso a bases de datos es sin duda uno de los principales campos de interés para los expertos en reconocimiento del lenguaje natural.No cabe duda que cualquier aplicación informática, desde un procesador de textos a una hoja electrónica, se revalorizaría enormemente si fuera capaz de ofrecer un interfaz de estas características al usuario.

Los programas que proporcionan interfaces en lenguaje natural, tanto para bases de datos como para aplicaciones más convencionales, se pueden dividir en dos tipos, denominados controlados por menú y controlados por comando.

En los primeros se presenta al usuario una serie de ventanas en las que en cada instante van apareciendo los distintos términos que se pueden escoger para componer un comando.El programa más conocido dentro de este grupo es "NaturalLink" de Texas Instruments, del cual existen versiones diseñadas para funcionar de forma conjunta con programas tan populares como Lotus 1-2-3,dBase III, WordStar, MultiMate, MultiPlan e incluso con el propio sistema operativo MS-DOS.

Cuando el producto de Texas está en acción, en la parte superior de la pantalla aparece el mensaje "I want to ..." y debajo de él una serie de ventanas que contienen las palabras y frases con las que componer la petición.Por medio de las teclas de cursor o de un ratón es posible desplazarse de una a otra ventana escogiendo los "pedazos" de comando necesarios.A medida que éste se va construyendo, aparece escrito en la parte superior de la pantalla. Una vez que se ha completado,la simple pulsación de una tecla visualiza el comando que en situaciones normales se habría

tecleado,procediéndose acto seguido a su ejecución.

En los programas dirigidos por comandos, el enfoque es completamente distinto.En estos casos el usuario se enfrenta a una pantalla en blanco sobre la que puede realizar cualquier tipo de pregunta de la forma más natural. Evidentemente este es el tipo de aplicación que caeria de lleno en lo que verdaderamente se entiende por reconocimiento del lenguaje natural, pero el éxito parece estar más del lado de las aplicaciones dirigidas por menús, para las que el calificativo de "lenguaje natural" es, cuando menos, de delicada aplicación.Ello se debe en gran medida a factores psicológicos:el usuario se siente muy desasistido cuando se enfrenta al frio resplandor de una pantalla vacia o en la que tan sólo aparece el mensaje "Digame qué desea que haga". Por contra, en un sistema basado en menús, el usuario puede navegar libremente por todas las opciones que tiene a su alcance cualquiera de ellas.Además, se trata de un excelente sustitutivo de la lectura de los manuales, ya que no es necesario memorizar una larga serie de letras para especificar distintas opciones de un mismo comando... Todas ellas aparecen en pantalla.

Evidentemente, el usuario experto se sentirá muy atado por un entorno que le obliga a iniciar un largo viaje por un sistema de ventanas cuando lo que simplemente desea es un COPY *.* B:.Por tal razón, este tipo de programas suelen permitir opcionalmente la vuelta a un entorno de comandos mediante la simple presión de una tecla.

Conclusiones.

La I.A. tiene mucho que aportar en un entorno complejo como es el de la toma de decisiones empresariales o de cualquier otro tipo.Al comentar lo que son capaces de hacer los Sistemas Expertos actuales por la comunidad empresarial, es obligado hacer referencia a nuevoes entornos

de lenguaje natural que permitan mejorar las relaciones entre este colectivo, que en general no tiene por qué ser experto en la utilización de ordenadores, y las aplicaciones informáticas más corrientes:Desde el acceso a una base de datos, hasta el envio de un fichero a través de linea telefónica, pasando por el mantenimiento de los archivos en disco.

Todas las aplicaciones comentadas -Sistemas Expertos basados en reglas o en ejemplos,e inerfaces de lenguaje natural- se encuentran disponibles sobre PCs.Su calidad y facilidad de manejo varia enormemente de unas a otras, en especial en lo que respecta a los Sistemas Expertos, campo en el que es posible encontrar desde excelentes herramientas de

desarrollo,muy orientadas a los profesionales de la informática, hasta pequeños y económicos sistemas basados en ejemplos que pueden ser utilizados sin más conocimientos informáticos que saber por dónde se enciende el ordenador.

Igual ocurre en el terreno de los interfaces de lenguaje natural.El mercado ofrece tanto auténticos paquetes integrados como simples intérpretes para suavizar el acceso a una base de datos.La evaluación de este tipo de software es una tarea que está en muy estrecha relación con la complejidad de la aplicación a resolver.

Aunque todavia en estado embrionario, Sistemas Expertos y reconocimiento del lenguaje natural son dos disciplinas de la ingenieria del software que están creando una nueva forma de interaccionar entre el hombre y la máquina.Sólo juntas serán capaces de hacer realidad los vaticnios de la Sociedad de la Información,cuyo comienzo nos ha tocado vivir.

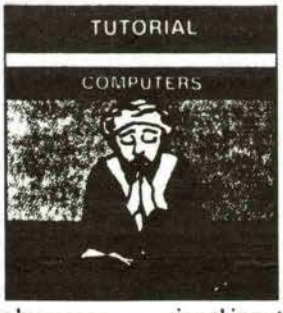
Autor: Angel Martinez

* * * * * * *

INTERNACIONAL

Software 'doctor' prescribes remedies

Through expert systems, artificial intelligence can diagnose ailments in diverse equipment—from mainframe computers to oil drilling rigs



At 4:00 a.m., the disk subsystem of a critical mainframe computer breaks down. Within a few minutes, the computer operator has reached the manufacturer's expert troubleshooter by phone at home. A short exchange yields the probable cause of the failure, and the operator begins to take corrective action. Scenes like this have been common for years, but they may become rare. With software techniques now being put into practice, no one need be awakened from a sound sleep for an expert consultation.

Knowledge-based systems—programs that operate by reasoning from a set of facts and rules—can quickly isolate the cause of a failure and recommend fixes. Such systems are primed with the most likely failure modes of equipment, the underlying causes of given symptoms, and similar information. Troubleshooting systems, which have been developed by dozens of companies, range from software to improve the performance of computer systems to programs that tell oil well crews how to restart stalled drilling rigs.

In theory, troubleshooting programs can be written for any system that humans can diagnose. Researchers are working on systems that will not need to be primed with knowledge about how failure occurs. Instead, they will be able to deduce failure modes and symptoms from a description of the system's design and intended function.

For the present, however, knowledge engineers must construct troubleshooting programs by encoding diagnostic techniques into a form that computers can execute. These techniques are obtained from domain experts—either expert troubleshooters or equipment designers. A few companies sell software to ease the task of building such systems, but the field and the market are still emerging.

Troubleshooting based on artificial intelligence is an outgrowth of early work at universities on expert systems for diagnosing human illnesses as well as machine faults. Programs such as Stanford University's Mycin, Puff, and Oncocin reason about infection, respiratory problems, and cancer, respectively, based on rules connecting symptoms to causes, and the EL program, developed at the Massachusetts Institute of Technology by Gerry Sussman, was used to analyze electronic circuits. Although troubleshooting software based on the principles elucidated in this university work can be applied to a wide range of problems, it also has limitations. Chief among them is that the connection between symptoms and causes is based on empirical evidence rather than an underlying model of how a system functions. Troubleshooting software may thus make obvious mistakes when faced with unfamiliar problems or incomplete evidence.

Troubleshooting software runs from the simple to the inordinately complex, but the design principles are the same: with a series of symptoms as evidence, the program uses some form of IF-THEN rules to determine what is wrong. Some symptoms suggest specific faults, while others rule them out. Troubleshooting software can narrow the range of components that may have failed by calling for more tests. Programs can engage in forward chaining—reasoning from facts and rules to conclusions—or backward chaining—from tentative hypotheses to supporting facts. Often both kinds of logic are used.

For example, if an oscilloscope doesn't display a trace properly, the possible sources of the fault range from the power supply (and the wall plug) to the signal inputs. Knowing that the scope is getting power would rule out many faults by forward chaining. A troubleshooting system might then select as a working hypothesis that the input section is faulty. Chaining backward through a rule like "If a good signal is fed into the input and does not appear on the display, then the input section may be faulty," the software might recommend a simple test to confirm or disprove its hypothesis.

Many troubleshooting systems in use by industry are relatively simple: a few hundred rules or so, compared with several thousand rules in the most complex programs—although rule counts may vary by a factor of 10 depending on the language used, just as lines of code do in conventional software. Some programs, in fact, start out essentially as rule-based translations of the service instructions for a system. So what incentive is there to take something that a person can do perfectly well and turn it over to a machine? Developers of troubleshooting systems offer several explanations, chief among them increased reliability, consistency, and availability.

Even though technicians may generally make the right decisions in troubleshooting a system, they occasionally make mistakes; troubleshooting software may reduce those errors. Because the troubleshooting program encodes knowledge in computer-readable form, it can speed up the diagnostic process and improve the productivity of both expert and novice technicians.

Paul Wallich Associate Editor

Defining terms

Domain expert: a person whose knowledge is tapped to produce the rules that guide a knowledge-based system.
Inference engine: software for drawing conclusions from facts expressed in machine-readable form.
Knowledge-based system: software that uses a series of empirical rules about a topic, coupled with an inference procedure, to draw conclusions from an input set of facts.
Knowledge-based system shell: a program to speed the development of knowledge-based systems; it usually contains an inference engine, a rule editor for entering new knowledge, and various methods of knowledge representation for encoding facts about objects in the domain.
Knowledge engineer: a person, often a programmer, who works with a domain expert to encode knowledge; some knowledge-based system shells may enable domain experts to act as knowledge engineers.

The automated approach can also make sure that all the technicians using it take the same actions in response to a problem. If one technician's approach is demonstrably better, then it can be incorporated in the software for all to use. Researchers have found differences of 10:1 or more in the performance of nominally competent people performing certain tasks.

Because the hardware, software, and engineering time required to develop troubleshooting software are fairly costly, these techniques have been used mainly thus far to diagnose high-priced systems: large computer systems, steam turbine generators, telephone networks, and so on. Because of the intimate design knowledge required for troubleshooting, suppliers of a system usually develop its troubleshooting software, although a customer with sufficient expertise and need could also build the program. Thus far, third-party efforts have been relatively rare.

Knowledge-based systems in action

One of the first industrial applications of knowledge-based systems for troubleshooting was General Electric Corp.'s CATS (computer-aided troubleshooting system), designed to improve the maintenance of diesel locomotives. The system codified the expertise of a GE troubleshooter with decades of experience. A 500-rule prototype implemented in the Forth language on a PDP-11 minicomputer was put into trial use in mid-1983 at GE's own repair shops and at two customer locations.

GE has capitalized on this early work to build other troubleshooting systems. Last spring the company won a U.S. Air Force contract to develop a knowledge-based system for maintaining and repairing GE TF-34 jet engines on A-10 ground-attack planes. In addition to helping less-skilled mechanics diagnose engine problems and make repairs, the project also calls for the software to reason about engine conditions from sensor readings and the maintenance history of other engines. If that part of the development effort is successful, it may be possible to inspect and replace jet engine parts as they become worn rather than at fixed intervals, as is done now.

Not surprisingly, quite a number of knowledge-based systems have been developed to troubleshoot and tune computer systems. Digital Equipment Corp., for example, reported as early as 1982 on a rule-based system for diagnosing failed circuit boards in disk drives. Based on the symptoms uncovered, the software chose tests that would eliminate the largest number of candidate failures.

The software also allowed for input from the technician doing the troubleshooting. If the technician, perhaps because of prior experience, thought a particular part was at fault, the troubleshooting program could make tests that would immediately confirm or disprove the hypothesis.

AT&T Bell Laboratories was another early industrial user of knowledge-based systems. Its ACE (automated cable expertise) program is used by several local telephone companies to manage the maintenance and repair of multiple-pair telephone cables. ACE analyzes reports of trouble from field service technicians to determine when a particular cable is deteriorating and is in need of repair. ACE receives its reports from a problem database maintained by the local phone company using it, and sends out its recommendations via electronic mail to the maintenance supervisors responsible for the cables in question. This off-line approach differs from that of most troubleshooting programs, which require a technician to enter information directly into the program, for immediate feedback.

```
IF   there is a voltage across the light socket when
       the switch is turned on
     AND the light bulb does not light
THEN the light bulb is burned out (.80)

IF   there is a voltage across the light socket when
       the switch is turned on
     AND the light bulb does not light
THEN the light bulb is not properly screwed in (.30)

IF   the light bulb rattles when shaken
     AND the light bulb does not light
THEN the light bulb is burned out (.99)
```

IF-THEN rules are a commonly used representation for encoding knowledge in artificial-intelligence systems. Such rules can be used either to draw conclusions from facts or to determine what tests should be made to see if a hypothesis is true. The certainty factors following each conclusion are used to support further reasoning but do not represent probabilities.

Troubleshooting power

Westinghouse Electric Co. has developed a rule-based system for diagnosing steam turbine generators that uses more than 8500 rules to analyze instrument readings and determine whether generators are being operated correctly or whether they should be brought off-line to repair problems. In some cases repairs can be delayed until the generator is taken off-line for scheduled maintenance.

The software, called Gen-aid, is in use on seven generators at Texas Utilities Generating Co. and on one in New York state. A utility in Florida is scheduled for connection in early 1987. Although the full-blown version has been operational for only about a year, prototypes have been in operation since mid-1984, and the system has already diagnosed a number of faults, according to Avelino Gonzalez, a systems engineer at the Westinghouse diagnostic center in Orlando, Fla.

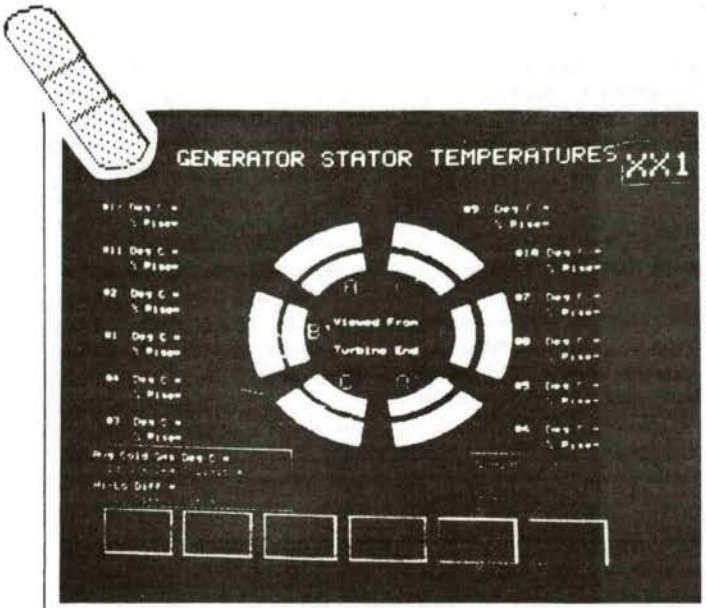
For example, Gonzalez said, in early 1985 the diagnostic system reported that, based on its sensor readings, some strands had probably broken in the stator windings of a generator. The unit was taken out of service and repaired within four days. If the generator had continued to operate until it exceeded limits on current flow or temperature, the damage would have been more extensive, requiring weeks or months to repair, he said.

Gen-aid uses about 250 sensor inputs—far more than a human operator could monitor simultaneously—including vibration, temperature, and chemical data. Chemical sensors sample the water that is turned to steam and passed through the turbine, and they also check the air passing through the generator. Increased levels of certain ions may signal that the insulation on the generator's windings is decomposing under high temperature, for example.

The sensor readings are collected by a minicomputer—a DEC Microvax II—and any data outside a narrow band of nominal operating conditions is relayed to the diagnostic computer in Orlando. The Orlando computer then checks whether the IF sides of any of the rules in its knowledge base match the newly reported conditions; if they do, the program reasons forward to find the most likely cause of the problem and a recommended solution. The recommendation is then passed by personnel at Orlando to the operators of the power plant, who can decide either to implement it or to wait for further developments. Automatic transmission of recommendations is expected to be implemented by summer 1987, Gonzalez said.

According to Jean Chess, a senior engineer at Westinghouse's research and development center in Pittsburgh, one of the most sophisticated parts of the software is not the rules for diagnosing the generator itself, but those for making sure that all the sensors are in good working order. If they are not, the program either ignores them or recalibrates their data. For example, if a thermocouple fails, it usually just gives an off-scale reading, while a chemical sensor may drift over time. By cross-checking the chemical sensor's output against the output of other sensors and against information derived from fundamental principles of turbine generator operation, the program can correct the readings.

The troubleshooting system is derived from a knowledge-based system shell developed by Westinghouse in cooperation with researchers from Carnegie-Mellon University in Pittsburgh. The shell—a program that simplifies the development of knowledge-



Westinghouse Electric Co. (photos)

A Westinghouse Electric Co. expert system monitors the health of steam turbine generators. In addition to detecting generator faults and recommending fixes, it can also display conditions within the generator on a touch-sensitive color CRT.

based systems by providing a syntax for expressing knowledge—is called the Process Diagnosis System (PDS). It is specialized for monitoring the health of continuous processes, rather than making judgments based on a single snapshot in time.

Although the shell for the troubleshooting software was originally developed in Lisp, a language used for many artificial-intelligence programs, the production version of the package is written in the C language for speed, Chess said. As new features are developed, the program code for them is converted from Lisp to C by a set of automatic conversion programs. Updating is a continuous process, Chess noted.

"There are several people in Orlando who work full time writing rules," she said. "A system like this is never quite done."

The shell game

While rule-based software can be written in any computer language, development is much easier with so-called expert-system shells. These programs contain tools for creating and editing rules and for doing forward- and backward-chaining inferences, so the knowledge engineer or domain expert need only encode the requisite knowledge. For example, a typical shell would contain algorithms for interpreting IF-THEN rules; without a shell, those algorithms would have to be written from scratch.

The first step in developing a knowledge-based system, once the scope of the project has been fleshed out, is to build an initial knowledge base. A knowledge engineer may build up the base by extracting rules from texts such as maintenance manuals, or an expert troubleshooter may solve several test cases while explaining the reasoning behind each conclusion or choice of tests.

As shells become easier to use, domain experts may act as their own knowledge engineers, writing down the rules they use and entering them into the shell. Once the knowledge has been entered, the prototype troubleshooting system can take on a few test cases. Usually it makes serious mistakes, highlighting situations where rules conflict or do not cover special cases. In some cases, mistakes occur because the methods that manuals or engineers claim should be used for troubleshooting are not correct.

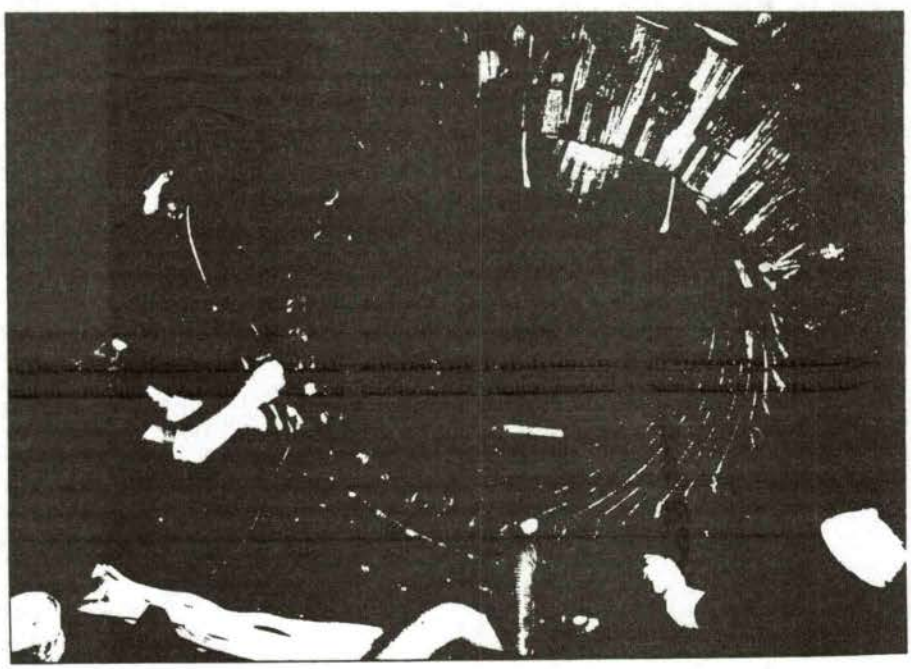
By tracing the reasoning of the knowledge-based system and asking the domain expert for clarification, the knowledge engineer can discover the rules the expert troubleshooter really uses. Additional test cases can then be posed to refine the knowledge base further. When the troubleshooting program correctly diagnoses all test cases, it is ready for field testing. Typically, more errors will be found, and the rules will be refined again. Eventually the system will correctly diagnose the vast majority of problems it is faced with, although additional special cases may appear indefinitely, just as they do to test the skills of human troubleshooters.

Although it is certainly simpler to write knowledge-based systems using shells instead of a programming language such as C or Fortran, generic knowledge-based system shells may require special extensions to suit them for troubleshooting. Shells specialized for troubleshooting, like that developed by Carnegie-Mellon and Westinghouse, offer an additional level of sophistication. Such shells may incorporate tools for reasoning about failures and the results of failures, so that the knowledge engineer or domain expert does not have to encode that knowledge explicitly—for example, the fact that a failed part will produce an abnormal output from normal input.

GE, for example, has refined the tools used to build its CATS system for locomotive troubleshooting into a diagnostic shell called GEN-X. The shell has been used for a number of projects within the company, and GE is considering licensing it to selected customers and eventually selling it as a product by itself.

GEN-X was designed with two major goals in mind: easing the knowledge-encoding task, so domain experts can enter much of their expertise without the help of a knowledge engineer, and improving the design of the user interface so technicians on the shop floor can use the system with a minimum of trouble.

Three separate forms of knowledge representation are used in GEN-X: rule tables, decision trees, and AND/OR trees. All map to the same underlying structure, but one or the other may make more sense for representing a particular portion of a problem. For example, a decision tree is well-suited to represent a set of sequential actions—such as performing diagnostic tests to home in on the cause of a failure—while a rule table could be more ef-



The steam turbine generators diagnosed by Westinghouse's 8500-rule monitoring system have many potential trouble spots—of which one is generator windings, which may fracture under stress or overheat. Two separate winding failures have been diagnosed by the monitoring software.

fective for representing a large set of rules to be applied to a single body of facts.

The shell also has provisions for attaching procedures to facts and conclusions in the knowledge base. Thus, if the application of a rule requires the value of a parameter like temperature or pressure, GEN-X could issue instructions to read the data from a sensor instead of just requesting its value from the user. Similarly a conclusion that a particular part had failed could trigger a videodisk sequence showing the part's location and the proper method for replacing it.

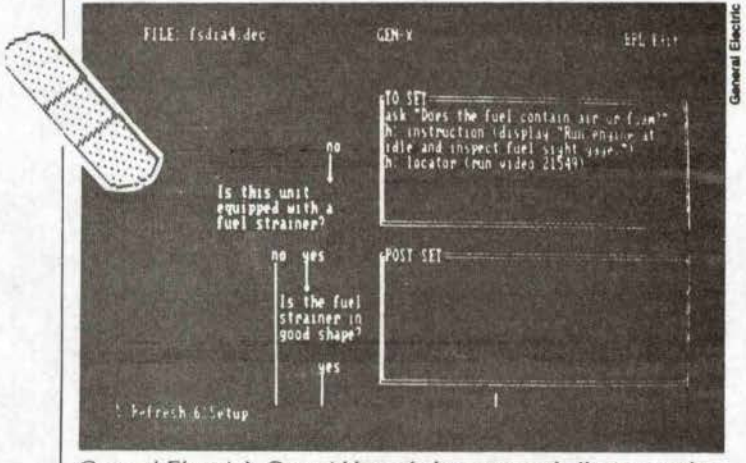
Other important issues to consider when looking at knowledge-based system shells are speed and memory use. While knowledge editing, simulation, tracing, and explanation facilities may be crucial to developing a good troubleshooting program, those features work best with the power of a large computer behind them. Since a troubleshooting program may end up on the shop floor in a relatively small, rugged package, it is important that it run adequately with limited computing resources.

Rules are not enough

Although rule-based systems are adequate for many kinds of troubleshooting, they also have limitations. For one, they do not incorporate comprehensive knowledge of how a system works, a major flaw that may prevent them from coping with unexpected failure modes or with multiple failures. In addition, if the system to be diagnosed is changed even slightly, the new behavior must be incorporated into new rules by a knowledge engineer working with the domain expert, in much the same way the troubleshooting knowledge was originally built up.

Researchers at several organizations, including the Massachusetts Institute of Technology, Xerox Corp., Stanford University, and Rutgers University, are working on techniques that will let a program deduce a system's function from its structure and the functions of its components, so that it can diagnose faults by comparing expected behavior with real behavior. Such programs would be able to troubleshoot a system based on its schematic, in much the same way that expert technicians do.

All the programs developed use some form of consistency checking—also known as truth maintenance or constraint propagation—to troubleshoot faulty systems. Since the function of each module in a circuit determines the relationship between its inputs and outputs, a program can reason about what signals should be appearing at which nodes in the circuit. For example, an adder's output will always be the sum of its inputs. If it isn't, then the adder has failed. Because the constraints relating outputs and inputs are logical rather than physical, they can be propagated backward as well as forward. Knowing that the output of an AND gate is a one, for example, constrains each of its inputs to be a one.



General Electric's Gen-aid knowledge-system shell program lets users represent knowledge in the form they find most convenient. Shown here is a decision tree for diagnosing diesel locomotives, along with the actions the program will take at one step in the tree.

Constraint propagation for troubleshooting works by tracing circuit constraints forward from the inputs and backward from the outputs until the values predicted at a node conflict. If the outputs of a circuit imply one value at an internal node but the inputs imply another, then the constraints must be violated somewhere along the way. By checking values at other nodes within the circuit, a troubleshooting program can determine which module has failed. Automatic test equipment, of course, can perform the same function, but only for a single system; a constraint-based troubleshooting program has the potential to diagnose any system that can be described to it.

Good troubleshooting calls for more than the ability to diagnose faults when given complete information about a system's condition. Since troubleshooters generally start with incomplete data about a malfunction, it is critical for programs to be able to decide which tests will furnish the most information about failed components. This decision-making process is complex.

For example, an unreliable component—one that a troubleshooter would want to test first—might be so deeply imbedded in a system that many tests would be required to determine unambiguously that it had failed. A troubleshooting program must thus determine which tests will most likely yield useful information, by balancing the distinguishing power of each test against the reliability of the components it covers. Even a test that could rule out dozens of failure candidates might not be much use if all the candidate components are known to be highly reliable. On the other hand, a test that eliminated only one failure candidate might be useful if that candidate failed often. Armed with a description of circuit modules and their interconnections, the program can determine which tests check which components. A database of the failure rates of parts contains information for the other side of the balancing act.

Programs that troubleshoot circuits based on analysis of their functional blocks and the connections between them are powerful in many ways. However, their idealized approach to describing circuits has its drawbacks, such as assuming that wires and devices are always unidirectional, with information flowing only from inputs to outputs. As many circuit designers know, violation of such assumptions is the most difficult to track down.

One solution is a hierarchical approach to troubleshooting: considering modules and their connections at an idealized level as much as possible, but moving to more detailed descriptions when necessary. Randall Davis, a professor of computer science at MIT, has worked to develop systems that can go all the way from a top-level block diagram of a digital circuit down to diagnosing a faulty pin in a backplane connector.

Extending the troubleshooting function

Researchers are also working to develop software to help debug systems while they are still in the design stage. One way is to use constraint propagation as a form of simulation to see if a design will actually perform its intended functions. By including timing as a constraint, for example, a debugging system can tell whether a circuit will run at its rated speed. Software being developed at Rutgers by a group including computer science professors Tom Mitchell and Van Kelly can also determine how robust a circuit is—that is, how much its performance would have to degrade before it would go beyond its voltage or timing margins.

The Rutgers effort includes a module that helps modify old designs to meet new needs. When old subsystems are replaced by new ones, or other design parameters are changed, this software can tell which design constraints have been violated and indicate ways that the design can be altered to make it functional.

Although researchers are extending the reach of troubleshooting programs, and some rule-based packages have already found their way into production use, there are still failures that defy automated analysis. One that is hard for even the most

(A) IF power-on
AND button-pushed
THEN sequencer-activated

(B) IF sequencer-on
THEN motor-activated

(C) IF motor-on
AND belt-tight
THEN drum-spinning

(D) IF panel-light-on
THEN power-on

(E) IF [X]-activated
AND NOT ([X]-on)
THEN [X]-failed

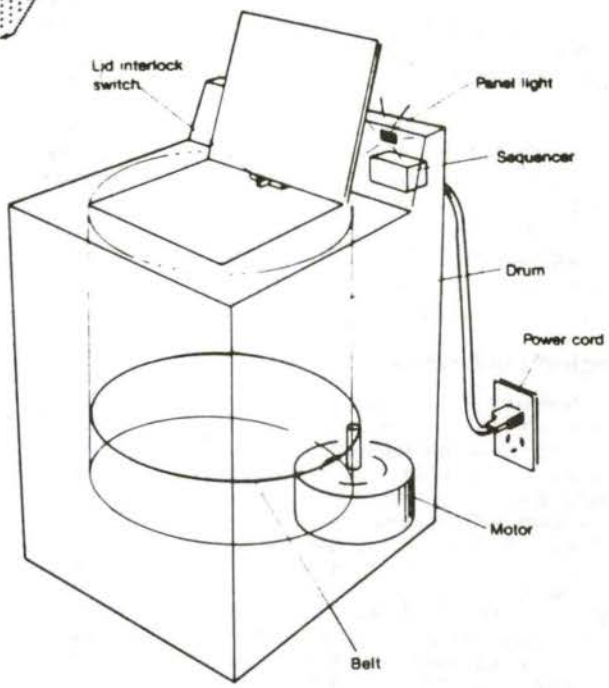
Visible data:
Button-pushed
Drum-spinning
Sequencer-on
Panel-light-on
Invisible data:
Belt-tight
Motor-on

In this portion of a hypothetical rule-based system for diagnosing washing machine faults, rules indicate which parts depend on others for their function. It may be necessary to chain back through several rules to find all the conditions that one part depends on. If the start button has been pushed, but the washing machine drum is not spinning, then any number of things could be wrong.

A troubleshooting system would try to narrow down the failure by chaining back to find possible faults, then looking for visible information that would confirm or disprove its hypotheses. For example, if sequencer-on is false, then either power-on is false (the machine is not plugged in) or the sequencer is broken. The troubleshooting system looks for a rule telling it how to find whether power-on is true or false and finds (D). Since panel-light-on is a piece of visible data, the system can ask whether the panel light is on without incurring the cost of disassembling the washing machine. If the panel light is on, then the power is on and the sequencer is faulty.

In this example, it is quite possible that the troubleshooting system will exhaust its possibilities from the visible data without reaching a conclusion. In that case it will have to ask that the washing machine be disassembled so that additional information can be gathered. The structure of the inference engine—the software that interprets the rules and decides which one to apply next—must be set up so that rules depending on simple tests are applied before those depending on more complex tests are invoked.

A more flexible inference engine will use so-called meta-rules rather than hard-coded program logic to determine which rules should be applied when. This does not affect the end result, but may make the troubleshooting system easier to develop.



The troubleshooting system here suffers from a number of shortcomings. Not only does it fail to deal with malfunctions such as a broken water hose or ill-fitting lid, but it also makes a number of dangerous assumptions that are implicitly encoded in its rules. For example, since most washing machines have fairly complex wiring, power present at one location need not imply power at all locations. In addition, the assumption that connections between modules will not fail is faulty, since a broken wire between the sequencer and the motor could prevent the motor from turning on even though neither the motor nor the sequencer was faulty.

Additional rules or clauses can be added to handle such problems, or if certain failures are considered unlikely enough, they may be ignored, just as human troubleshooters often ignore improbable failure modes. The system could also be redesigned so that the connections between different parts of the washing machine were represented explicitly rather than being implicit in the rules. —*P.W.*

sophisticated systems to track is the intermittent fault. Because it does not always occur, it requires reasoning about system behavior over periods of time, something that knowledge-based systems generally do only in relatively crude fashion.

Another problem whose full understanding still eludes researchers is time-related behavior. For example, faults may be triggered (or revealed) only by particular sequences of events, such as a message arriving at a network controller before it processes the previous one, or a particular series of bit patterns that might cause a disk controller to fail. Dealing with such failures requires techniques for representing both sequence—the order of events—and duration, or length of time between events.

Putting troubleshooting expertise to use

Certain features appear common to most current applications of troubleshooting software. Among them are the high cost of downtime for the equipment being diagnosed, a scarcity of human experts, and a relatively well-defined problem. Power plants and oil rigs are examples of operations where downtime can cost hundreds of thousands of dollars per day; troubleshooting software costing half a million dollars could pay off very quickly. Less expensive systems can justify troubleshooting software only if there are enough of them to make the total cost savings large.

On the other hand, simpler, less costly tools for building rule-based programs are beginning to become available to run on personal computers, and the cost of building relatively small troubleshooting systems with them could be fairly low, assuming a domain expert is available. For simpler troubleshooting programs, the required expertise may be simply a codified version of existing repair manuals.

A scarcity of human experts for troubleshooting a particular piece of equipment adds to the cost of downtime by increasing the time needed for repair. However, in some cases troubleshooting programs may help to standardize and improve repair jobs even if many qualified people are available to perform them. The speed of computers and their tolerance for monotonous jobs may also make troubleshooting programs effective for monitoring applications where it would not be practical to use people, no matter how many had the expertise. Westinghouse's Gen-aid is one example of this kind of application.

The third feature of most troubleshooting programs, limited scope, is probably the most pervasive. Research programs may be able to diagnose malfunctions in several different systems when given a representation of the design of each system, but production troubleshooting packages are specialized for a single system, be it a digital data network, a locomotive, or a gas turbine. Because

each system has limited scope, the system designers may help build the troubleshooting programs—essentially as part of the documentation effort—or a systems integrator may construct the troubleshooting software for a particular customer or configuration. A large customer acting as integrator may also undertake the effort.

What issues should the designer of a troubleshooting system take into account? The most obvious are whether the job can be done, and at a reasonable cost. If troubleshooting software does not solve a large enough portion of the problem at hand, it is useless regardless of cost, but even if it does, the computing resources and time required to run it may make the software approach unfeasible.

Emerging applications

Troubleshooters in industry have begun to diagnose malfunctions with both large and small expert systems. A small sampling of projects reveals some implementations larger than General Electric's 700-rule diesel locomotive diagnostic system, but implementations with as few as 150 rules will successfully spot malfunctions. The projects surveyed below are only a sample of those in progress, and many more companies than those mentioned below are engaged in expert systems work.

• Texas Instruments Inc. in Austin has developed systems both for in-house use and for other companies on a consultant basis. In one project, TI implemented an expert system for the Campbell Soup Co. in Camden, N.J., to troubleshoot sterilizers—or cookers—in soup plants. If a sterilizer malfunctions, technicians must diagnose the problem quickly; otherwise they must discard the soup that is in the sterilizer. Because Campbell sterilizer expert Aldo Cimino will soon retire, an expert system was developed to codify Cimino's knowledge. The 151-rule system, written with TI's Personal Consultant development system, runs on a TI Professional Computer.

Within TI, a 1000-rule system helps maintain the epitaxial reactors that deposit thin, nearly perfect layers of semiconductor on silicon wafers.

• An expert system helps both users and technicians pinpoint problems in the RT PC computer marketed by IBM Corp. If the computer malfunctions, the program, which runs on an RT, asks the user or technician questions about the system's operation and selects appropriate test routines to pinpoint the problem. A technician can also request specific recommendations for repair. For example, if the program suspects a faulty computer memory, it can run a memory test and recommend replacing any defective chips. Written in the Pascal language, the 150-rule system is included with the RT PC.

• GTE Corp.'s Compass system interprets diagnostic messages issued by a GTE telephone switching system and suggests solutions to technicians. The program consists of 500 rules, 500 functions written in the Lisp language, and 1000 frames. (A frame contains information "slots" that the program can fill with actual or assumed data to predict what component of a switching system has failed.) With frames and IF-THEN rules, Compass can group many related malfunction reports and locate defective components. The program was written using the KEE (Knowledge Engineering Environment) development system from Intellicorp of Mountain View, Calif., and runs on a Xerox 1108 Lisp machine.

• Lockheed Corp. in Palo Alto, Calif., has been studying expert systems for troubleshooting satellites. One program developed by the company could help diagnose the reaction wheel assembly of the yet-to-be-launched Hubble Space Telescope. The reaction wheel spins the telescope to point in the desired direction. The 180-rule program was developed using LES (Lockheed Expert System). In addition, Lockheed has begun studying networked expert systems that could help diagnose problems on the proposed U.S. space station.

—Michael W. Lind

The user interface is also critical. Troubleshooting systems designed to be used on the shop floor, for example, must be written for a computer that can survive there, or for a terminal that will be linked to a computer elsewhere. If a terminal is used, response time becomes important.

If a troubleshooting system is going to be used by technicians unfamiliar with computer keyboards, then special input devices, like touch screens, function keys, or graphics tablets may be necessary.

But even in a less stringent environment like an electronics repair shop, designers must make sure a troubleshooting program is easy to use. User-interface programming typically accounts for close to 40 percent of the total program code in many knowledge-based systems, although this proportion may decrease when the number of rules exceeds several thousand.

Perhaps most important is the tone of the user interface; traditional computing is rife with stories of software packages that failed because their potential users rejected them, and since knowledge-based software has even more potential for infringing on human territory, the approach taken in implementing a user interface can determine success or failure. Reports from the field indicate that a management approach that gives users a strong voice in design and implementation is far more successful than one based on imposition from above.

Many troubleshooting systems developed so far act as assistants to people, rather than as authorities in their own right. For example, the Westinghouse Gen-aid program makes recommendations to utility plant operators rather than carrying out actions by itself. And troubleshooting software for disk drives developed by Digital Equipment Corp. is designed to take a seasoned technician's speculation about problem areas as a beginning for its own reasoning process.

Not only does this kind of secondary role for knowledge-based systems make them more acceptable to workers, but it also simplifies the task of the troubleshooting software designer. By relying on the wide range of a technician's knowledge, the designer has less data to encode in the program. And letting people decide how to implement the computer's conclusions provides an extra margin of safety against the potential for incorrect behavior that plagues all knowledge-based systems, particularly when they face problems of a kind they have not encountered before.

To probe further

An overview of knowledge-based software systems is given in "Expert systems, limited but powerful," by William Gevarter, *Spectrum*, August 1983, pp. 37–45. The July 1986 issue of *IEEE Computer* was devoted to various aspects of expert systems, and *IEEE Software* publishes frequent articles on the subject as well. In addition, the new *IEEE Expert* magazine concentrates entirely on knowledge-based systems.

Articles on knowledge-based troubleshooting can also be found in *AI* magazine, published by the American Association for Artificial Intelligence in Menlo Park, Calif., and in the proceedings of AAAI conferences, available from Morgan Kaufman Publishers, 95 First St., Los Altos, Calif. 94022. The proceedings of the most recent conference, held August 11–15, includes descriptions of such applications as test generation, semiconductor fabrication, glass annealing, naval gun mounts, and satellite attitude control. Additional conferences on knowledge-based systems are sponsored by the Association for Computing Machinery and the IEEE Computer Society.

Knowledge-based system shells, which can be used to build troubleshooting software although they are not optimized for the purpose, are available from a number of companies at prices ranging from under $100 to over $100 000 to run on anything from personal computers to mainframes and special-purpose Lisp-language hardware. Information on them is available from the vendors, which include Teknowledge Inc. of Palo Alto, Calif., Intellicorp of Mountain View, Calif., and Inference Corp. of Los Angeles. ◆

ARTICULO:

Procesos Inteligentes:
Su simulación

Creo que la expresión "Inteligencia Artificial" se emplea para designar sistemas informáticos que no merecen tal apelativo. Se mete en el mismo saco de la inteligencia artificial tanto a sistemas expertos como a otros que, si bien están, creo, menos desarrollados, tienen bases conceptuales de diseño diferentes de los primeros.

Un sistema experto no es inteligente. Es un complejo sistema de gestión de una base de datos con un conjunto de aditamentos que dan una gran flexibilidad -incluso capacidad de aprendizaje- a tales estructuras. El fundamento de su diseño estriva en realizar un mecanismo tal que, ante un conjunto de datos de un caso concreto presentados por el operador, obtenga una solución a ese problema especifico siguiendo o simulando los pasos totalmente trillados que un experto humano daria. Su funcionamiento responde al de un "empollón" clásico cuando resuelve un problema exactamente igual que los que ha hecho en otras ocasiones, sin pensar, mecánicamente. De hecho, la capacidad de aprendizaje mencionada antes consiste en la modificación de la base de datos con el uso (o su aplicación), y no en el tratamiento de esos datos de forma que no se haya considerado en su implementación. No pretendo quitar mérito al diseño de tales sistemas, cuya utilidad práctica no pongo en

duda, ya que descargan al operador del trabajo rutinario de resolver problemas, de forma análoga a cómo las calculadoras nos ahorran las operaciones aritméticas. Pero insisto en que llamar a eso inteligencia me parece excesivo.

Existen otros sistemas que ciertamente son, por ahora, nada útiles, porque me parece que están poco desarrollados; pero me merecerían el calificativo de inteligentes en mayor medida que los sistemas expertos. Me refiero a un tipo de algoritmos de tipo deductivo. El objetivo sería el siguiente: conseguir que un programa mantenga un monólogo mediante las reglas de la lógica y dentro de una gramática formal a definir, a partir de un conjunto de premisas, que se suponen válidas, obteniendo nuevas proposiciones que anuncian verdades no diferentes de las que ya están en aquellas premisas de partida, pero que por estar allí implícitamente no eran evidentes. No es el objetivo de este escrito entrar en detalles técnicos sobre la realización de estos sistemas, pero sí quiero indicar su principal ventaja e inconveniente. En primer lugar carecerían de la capacidad de aprendizaje, ya que son estructuras puramente formales con carencia total de contenido (lo que es aprendible) y que por tanto deben ser desarrollados completamente desde un principio. En segundo lugar permitirían explorar los resultados de un conjunto extenso y complejo de leyes y supuestos escritos en esa gramática a definir.

El meollo de la deducción, en el seno de una determinada gramática, y regida por las leyes de la lógica y del cálculo proposicional (como por ejemplo las matemáticas, u otras más simples), tiene como principal problema el volúmen de supuestos y caminos lógicos a seguir dadas las proposiciones de partida. Básicamente, en un sistema lógico cualquiera, si los contenidos están relacionados por leyes, es posible, desde un punto cualquiera, y mediante una cadena de razonamientos adecuados, llegar a cualquier otro punto; y viceversa, con cualquier razonamiento siempre se consiguen conclusiones dentro del sistema, y verdaderas. Cualquier parte de un sistema está contenida en cualquier parte de ese sistema, con lo cual la deducción explícita muestra cosas no evidentes, pero no dá conocimiento realmente nuevo. Un sistema deductivo lo que haría sería utilizar la más típica de las

características de un ordenador, el manejo de muchos elementos, analizando concienzudamente todas las posibilidades de razonamiento desde un punto de partida dado. En fin, la inteligencia estaría en definir qué gramática y qué leyes de la lógica; lo demás es mecánico y no aporta nada nuevo de verdad.

Quizá haya podido dar a entender que lo mecánico es lo no inteligente, pero no es así estrictamente. Lo no inteligente es lo que no aporta nada nuevo, lo que no propórciona información totalmente nueva, inexistente en el momento de crear el sistema informático en cuestión. Obsérvese que un sistema experto con capacidad de aprendizaje tampoco sería inteligente, porque, de verdad, lo nuevo que aprende, más que aprendido es enseñado. A mi modo de ver, la inteligencia consiste en inducir propiedades desde un conjunto de datos o de acontecimientos. Lo importante son las inferencias que se hacen en la realidad. Carece de importancia práctica el que esto se haga de una forma más o menos mecánica en el cerebro humano, pues si pretendemos que un ordenador simule la deducción, él si que lo deberá hacer de forma mecánica, que

al fin y al cabo es lo que es, una máquina. Por eso decía más arriba que la inteligencia no es lo no mecánico estrictamente. Desde este punto se abre un sinnúmero de ideas y discusiones sobre qué y cómo es la inteligencia, en concreto este proceso inductivo tan fundamental en la auténtica creación de la información, no de su tratamiento. Para concluir sólo expondré unas cuantas reflexiones que hago en relación con todo lo anterior.

Hay quien opina que la inteligencia es algo sutil e inasible. No lo creo. Lo que ocurre es que estamos rozando ahora, con este problema de la inteligencia artificial, algo muy distinto de todos cuantos problemas con los que se ha enfrentado la gente técnica. Siempre recurrimos, para solucionar

cosas, a las ciencias duras, matemáticas, física, química, etc., suponiendo que nada hay más aprovechable en otros ámbitos del saber humano por ambiguos, faltos de objetividad y de precisión, como la filosofía y la psicología, etc., y esto fue muy cierto en siglos pasados, pero no en éste. Filósofos de la ciencia, lógicos, filólogos, psicólogos de la percepción, etc., con retraso si, pero con eficiencia también, están llegando y han llegado en algunos temas a conclusiones tan fiables sobre la inteligencia y los procesos inteligentes como cualquier otra ciencia simplemente porque también ellos han adoptado el método científico. ¿Por qué no volver nuestra mirada tmbién en esa dirección?. Con seguridad que encontraremos cosas de una utilidad

práctica evidente, y tanto más por cuanto nunca se ha aplicado una mentalidad técnica y pragmática, como se supone que debe ser la nuestra, a estas áreas de conocimiento.

He hablado de filosofía y psicología, y nada de neurología y fisiología del sistema nervioso. Hay dos razones para ello. La primera es mi más absoluto desconocimiento del tema; la segunda, y objeto de este último apartado, una tendencia, sin fundamento, desde luego, a pensar que hasta ahora, me atrevería a decir, casi nunca hemos imitado a la Naturaleza para imitar sus efectos. Los aviones no baten las alas para volar como las aves, los impulsa una hélice o un reactor, algo absolutamente desconocido en zoología, etc. ¿Por qué ahora sí que habremos de imitar el sistema neuronal para hacer inteligencia artificialmente?. No es que desprecie estas ciencias, sino que

me parece perezosa la postura de algunos que dicen que mientras no se conozca plenamente el funcionamiento del cerebro no sabremos reproducirlo en silicio y no tendremos inteligencia artificial. No hay razón a favor ni en contra de ello, es sólo una impresión y como tal queda.

José Ramón Serrano Olmedo. 3º.

GENERACIONES

La sexta generación de ordenadores

Pese a que para muchos la Quinta Generación de Ordenadores está todavia por madurar, los japoneses ya nos están avisando de su próxima incursión en el futuro. Un nuevo enfoque con unos objetivos más que ambiciosos encuadrado en un marco interdisciplinar dará forma a la investigación en IA durante el próximo cuarto de siglo.

Si bien todavía no ha sido comunicado oficialmente por parte del gobierno japonés, corren rumores sobradamente fundados de que este país va a lanzar en un futuro próximo lo que se ha de llamar el programa cientifico de la frontera humana. Este programa se caracterizará por la fusión de investigaciones y desarrollos tanto en IA como en las funciones de los organismos vivos. Según las primeras filtraciones del documento que se espera haga público el mencionado gobierno, "el objetivo del programa es la creacion de un nuevo paradigma para la ciencia y la tecnologia capaz de la raza humana con la naturaleza".

Algunos detalles dan cuenta de la magnitud que puede adquirir la nueva empresa: La duración del programa se estima en unos veinte años, con un coste total de seis billones de de dólares, de los cuales 1.2 billones serán consumidos en las fases iniciales. Estas cifras pueden verse multiplicadas si el proyecto se convierte realmente en

un foro abierto de participación de muchos paises - en el que cada uno de ellos aportaria factor humano y económico - en lugar de convertirse en un motivo de mutuos recelos como está ocurriendo en la generación actual. Los japoneses hacen especial hincapié en el aspecto internacional al precisar que seria deseable que la tercera parte de los investigadores sean extranjeros y al emitir directrices de trabajo más de acuerdo con la espontaneidad de esta parte del mundo que con la rigidez nipona.

Echar la vista adelante para profetizar los resultados de la Sexta Generacion resulta tan dificil como hace unos años resultaba el mismo ejercicio con respecto a la generación actual. De hecho, apenas se tienen referencias sobre algun prototipo concreto de la "máquina de inferencia paralela", la cual constituía uno de los objetivos fundamentales de la Quinta Generación. La generación actual puede caracterizarse mejor por los avances en el software que los que han tenido lugar en hardware. En este sentido, el elemento que mejor la define es la difusión experimentada por los Sistemas Expertos, cuyas semillas ya estaban plantadas mucho antes de la idea japonesa.

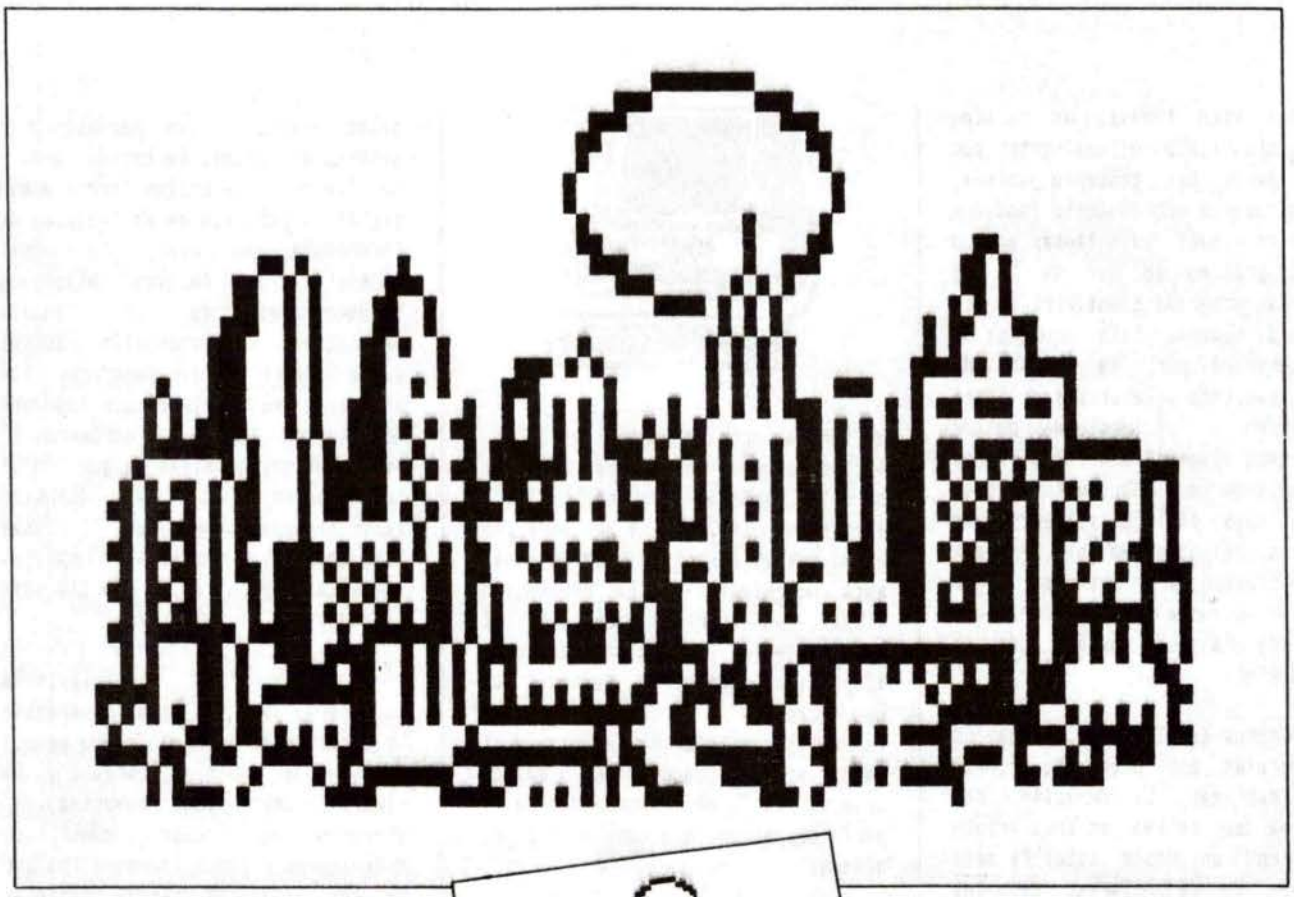
Algunos especialistas consideran que la Sexta Generación se caracterizará por la convergencia de estudios sobre el cerebro y la ciencia de los computadores, entrando en juego redes de ordenadores especializados a los que se les asignarán tareas concretas cuya resolución concurrente dará la solución global, de forma muy parecida a como actúa el cerebro. Para otros autores, la nueva

generación, aparte de poner un especial énfasis en el multiproceso a gran escala, se distinguirá de la actual en la importancia que tendrá la biologia en todo el diseño de las máquinas inteligentes, cubriendo el espectro que va desde los biochips hasta los ordenadores moleculares. En cualquiera de los casos, se pone de manifiesto la necesidad de un trabajo interdisciplinar que supere en mucho al existente en la acualidad.

Es posible que, al igual que ocurre en la generación actual, este objetivo tan audaz quede reducido finalmente a unas mejoras sustanciales en técnicas y productos precedentes. Los Sistemas Expertos actuales son capaces de competir con especialistas en determinadas materias, pero están todavia por solucionar problemas tan importantes como la ampliación de su campo de acción, el aprendizaje automático o la codificación del sentido común o de la creatividad. Cada uno de estos campos en si mismo da materia más que suficiente para ocupar los recursoso temporales, humanos y económicos asignados al programa de la Frontera Humana. Ni que decir tiene que otras ramas de la IA como son la robótica o el reconocimiento del lenguaja natural hablado y escrito apenas si se encuentran recién nacidas, y estarían dispuestas a reclamar para si los recursos antes citados. Por otra parte, todavia no se ha producido la verdadera explosión de la IA, una explosión que llenaria la sociedad de sistemas competentes de igual forma que la televisión inundó en su día miles de hogares en todo el mundo, y que probablemente tenga lugar con la nueva generación gracias a la difusión del ordenador personal y las redes de telecomunicación.

Discusiones aparte sobre los objetivos y su posible cumplimiento a la vista del estado del arte, lo cierto es que la Sexta Generación era algo inevitable desde el momento en que se empezó a hablar de la Quinta. El que con posterioridad lo que realmente deje tras de si un nombre con marcada intención comercial esté de acuerdo con los objetivos planteados en un primer momento carece de importancia si la investigación se ha dirigido en la dirección oportuna. La aparición de la máquina inteligente es sólo una cuetión de tiempo, no de nombres de proyectos.

Autor: Angel Martinez



NUESTRO SISTEMA EXPERTO

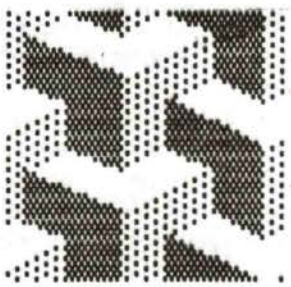
La justificación en un sistema experto

El módulo de justificación de un Sistema Experto es una de las partes más importantes de éstos. Su necesidad surge desde el mismo momento en que se realiza una consulta al sistema y al obtener una respuesta queremos saber cómo llegó el Sistema Experto a esa conclusión. Otra de las utilidades de un buen sistema de justificación, es que sirve para depurar el Sistema Experto cuando está en fase de prototipo avanzado (100-300 Reglas) o en su versión definitiva. Sin una herramienta de este estilo sería prácticamente imposible perseguir el razonamiento por una mesa de papeles, sin saber a ciencia cierta si lo que estamos haciendo con papel y lápiz es realmente seguir el camino que llevó a la máquina a la solución que nos ha dado.

Pese a la importancia de este tipo de sistemas, hay pocas personas en el mundo trabajando en estos temas, lo que provoca que los avances sean lentos, que no haya libros específicos sobre el tema y que si uno quiere adentrarse un poco en él, se vea obligado a perseguir revistas por todas las bibliotecas conocidas, y encontrarse continuamente trabajos no publicados de casi todas las universidades americanas.

El objetivo de este artículo es dar a conocer los tipos de justificación más usuales que nos podemos encontrar con un Sistema Experto, así como los problemas que podemos tener si pretendemos dotar a uno de ellos de un módulo justificador que se precie de serlo.

Hay tres tipos fundamentales de explicaciones:



1. Las que informaan sobre algo que ha sucedido, sin adentrarse en las razones que la produjeron. Este tipo de explicación sería la misma que la que nos da un compilador de un lenguaje cualquiera cuando tenemos un error:

130; EXPECTED ******

o más en relación con los Sistemas Expertos:

"He seguido la primera estrategia"

Es lo más sencillo que podemos encontrarnos, y desde luego no satisface, en general, la curiosidad del usuario acerca del camino seguido por la máquina para llegar a la conclusión que nos acaba de entregar.

2. Las que informan del camino seguido y las reglas aplicadas. Este es el tipo de explicación más comúnmente usado en los Sistemas Expertos, y dentro de él podemos toda una gama de profundidad en la explicación. Por ejemplo:

a. Aquellos que simplemente nos muestran el camino, dando las reglas, tal y como están escritas en el lenguaje de programación empleado, indicándonos por medio de unas claves cómo se produjo el encadenamiento.

b. Un siguiente paso consiste en los que nos entregan lo mismo, pero en lenguaje natural, con os que el usuario puede defenderse un poco mejor. Aquí tenemos nuevamente dos posibilidades: que la regla esté escrita en algún sitio ya en lenguaje natural, o que un generador de lenguaje natural la construya partiendo de la regla escrita en el código empleado para el sistema.

c. Una nueva mejora consiste en dar información acerca de la regla, como por ejemplo, quien nos dió la regla (para saber el grado de experiencia que poseia y poder modificarla, para poder cunsultar al experto...), para qué se emplea la regla (siempre en general), una somera explicación de la regla, en el caso de que no esté claro de qué antecedente se deduce el consecuente...

d. Cualquiera de las posibilidades anteriores ampliadas por un grafo que nos facilite la comprensión del encadenamiento.

3. Las que no sólo nos informan del camino seguiodo por el razonamiento, sino que también nos dicen porqué se eligió ese camino y no otros posibles, indicando cuales fueron los empezados y hubo que abandonar. Evidentemente éste es el más complicado de implementar y el autor desconoce si hay en el mercado algún S.E. que tenga unmódulo de justificación de este tipo.

Personalmente y con las herramientas, tanto de software como de hardware, que tenemos hoy, me inclino por realizar módulos del tipo 2.c, como los más factibles,

aunque me consta que se está investigando en el desrrollo de técnicas y herramientas que permitan, espero que en poco tiempo, la implementación en los Sistemas Expertos,de módulos del tercer tipo.

Para que el módulo justificador pueda ejercer su tarea cuando le sea requerida una explicación, es necesaria la existencia de una historia, la cual no es más que un "almacén" de datos en donde el Sistema Experto va guardandolos pasos que ha seguido hasta llegar a la solución. Sinesta historia es extraordinariamente dificil hacer una buena justificación. En ella se apuntan los objetivos principales, los subobjetivos en los que dividió los anteriores (y así para cada uno de ellos), las reglas que aplicó para demostrar estos objetivos, las

las variables a las que tuvo que dar valor al aplicar las reglas y estos valores, etc...

Son también muy interesantes los trabajos que se están desarrollando para que estas historias tengan todos los datos necesarios, y sólo los datos necesarios, así como sobre interpretaciones automáticas que puedan hacerse de ellas.

Conclusión:

La importancia de los sistemas de justificación en los Sistema Experto. es enorme, y es necesario que se vayan desarrollando herramientas que faciliten la tarea, haciéndo módulos justificadores "huecos", para que el Ingeniero del Conocimiento los rellene con las

reglas obtenidas de los expertos, (aunque esto podria hacerse automáticamente partiendo de las reglas de producción del sistema), y no se vea obligado a perder el tiempo en realozar un trabajo de programacion extra, que no le corresponde.

Bibliografía general sobre la documentación:

Estas referencias no pretenden ser nada, salvo unas pistas que el lector puede seguir para profundizar en el tema.

Buc 84 Rule based Expert Systems (Cap 18 y ss) Buchana y Shortlife,Eds. Adison-Wesley. 1984

Sch 84 A Computer-Based Dialysis Therapy advisor Schaffer,J.D.,Teschan.P.E., Caviedes,J.,Bourne,J.R. IEEE Transactions on bio-medical engineering,Feb 84 Pgs 255 - 259

Swa 81 Explaining and Justifing Expert Consulting Programs Swartout,W.R. Proceedings of the 7th I.J.C.A.I.,1981 Reimpreso en Readings in Medical A.I., Pgs 383 - 398

Swa 83 XPLAIN system for Creating and Explaining Expert Consulting Programs Swartout,W.R. A. Intelligence,21.1983 Pgs 285 - 325

Wie 80 BLAH: A System wich explains its reasoning Wiener,J.L. A. Intelligence,15.1980 Pgs. 19 - 48

THE PC-ERA

ELIZA

ELIZA es una simulación de un compañero de conversación relativamente coherente. El diálogo se efectúa en inglés por la simplicidad de conjugación de pronombres y verbos que presenta este idioma.

La primera versión de este programa apareció a mediados de los años sesenta y fue creada en LISP por JOSEPH WEINZENBAUM del M.I.T. Su conversión al lenguaje BASIC la realizó JEFF SHRAGER. La versión que aquí aparece es mucho más simple que la original; sin embargo, es más corta, de fácil comprensión, y su modificación es bastante sencilla.

La "DATA" de respuestas es la utilizada por STEVE NORTH en su versión para Microsoft BASIC.

El programa funciona de la manera siguiente: primero toma una frase del teclado, comprueba si ésta se repite o si contiene un mensaje de acabado. Después, busca alguna "palabra clave" que le indique qué tipo de respuesta es de esperar. A continuación conjuga el resto de la frase, es decir, intercambia las primeras y las segundas personas, (los pronombres principalmente, y algunos verbos). Por último, imprime la respuesta.

Para facilitar la comprensión de este programa adjunto una lista de variables y un comentario por líneas.

Si conseguís mejoras, nuevas versiones para otros lenguajes (por ejemplo PASCAL), o si tenéis alguna sugerencia, no dudéis en comunicárnosla. ¡¡ Espero que os guste !!

Variables:

N1:	Cantidad de palabras clave.
N2:	Cantidad de pares conjugados de palabras.
P(x):	Primera respuesta para la palabra clave número (x).
R(x):	Respuesta inmediata.
U(x):	Ultima respuesta.
I$:	Input (frase del usuario).
K$:	Palabra clave.
C$:	Palabra conjugada.
F$:	Respuesta.
P$:	Input previo.
K:	Número de palabra clave.
R$,S$,L,S,T:	Variables auxiliares.

Líneas:

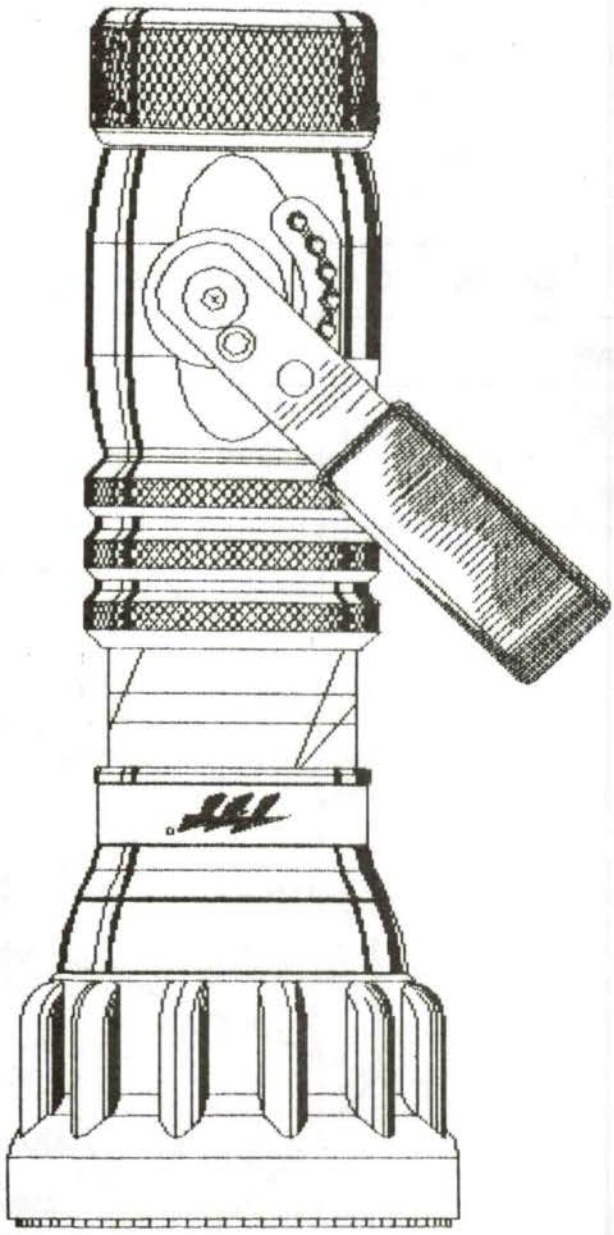
100,110	Inicialización.
120,130	Lectura de la frase.
140	Conversión a mayúsculas.
150,160	Comprobación del mensaje de acabado.
170	Chequeo de repetición.
180-260	Búsqueda de palabras clave dentro del input.
270-390	Conjugación del resto de la frase.
400-430	Respuesta y repetición.
1000-	Data - Palabras clave.
1100-	Data - Pares conjugados.
1200-	Data - Códigos.
1300-	Data - Respuestas.

LISTADO

```
100 N1=36 : N2=7 : DIM P(N1),R(N1),U(N1) : RESTORE 1210
110 FOR X=1 TO N1 : READ P(X),L : R(X)=P(X) : U(X)=P(X)+L-1 : NEXT X
120 PRINT "HELLO! I'M ELIZA.  WHAT'S YOUR PROBLEM?"
130 INPUT I$ : I$=" "+I$+"  "
140 REM
141 REM
142 REM  Si tu ordenador emplea letras minúsculas introduce aqui una
143 REM  subrutina que substituya  todas las letras minúsculas de la
144 REM  variable I$ por su correspondiente mayúscula.  Consulta el
145 REM  manual de tu ordenador y estudia las posibles alternativas.
146 REM
147 REM
150 FOR L=1 TO LEN(I$)-4 : IF MID$(I$,L,4)="SHUT" THEN PRINT "O.K."
                                                          : END
160 NEXT L
170 IF I$=P$ THEN PRINT "PLEASE DON'T REPEAT YOURSELF!" : GOTO 130
180 RESTORE 1010 : S=0
190 FOR K=1 TO N1 : READ K$ : IF S>0 THEN 230
200 FOR L=1 TO LEN(I$)-LEN(K$)+1
210 IF MID$(I$,L,LEN(K$))=K$ THEN S=K : T=L : F$=K$
220 NEXT L
230 NEXT K
240 IF S>0 THEN K=S : L=T : GOTO 270
250 IF S>0 THEN K=S : L=T : GOTO 270
260 K=36 : GOTO 400
270 RESTORE 1110
280 C$=" "+RIGHT$(I$,LEN(I$)-LEN(F$)-L+1)+" "
290 FOR X=1 TO N2 : READ S$,R$ : FOR L=1 TO LEN(C$)
300 IF (L+LEN(S$)>LEN(C$)) OR (MID$(C$,L,LEN(S$))<>S$) THEN 330
310 C$=LEFT$(C$,L-1)+R$+RIGHT$(C$,LEN(C$)-L-LEN(S$)+1) : L=L+LEN(R$)
320 GOTO 350
330 IF (L+LEN(R$)>LEN(C$)) OR (MID$(C$,L,LEN(R$))<>R$) THEN 350
340 C$=LEFT$(C$,L-1)+S$+RIGHT$(C$,LEN(C$)-L-LEN(R$)+1) : L=L+LEN(S$)
350 NEXT L : NEXT X
360 IF MID$(C$,2,1)=" " THEN C$=RIGHT$(C$,LEN(C$)-1)
370 FOR L=1 TO LEN(C$)
380 IF MID$(C$,L,1)="!" THEN C$=LEFT$(C$,L-1)+RIGHT$(C$,LEN(C$)-L) :
                                                          GOTO 380
390 NEXT L
400 RESTORE 1310 : FOR X=1 TO R(K) : READ F$ : NEXT X
410 R(K)=R(K)+1 : IF R(K)>U(K) THEN R(K)=P(K)
420 IF RIGHT$(F$,1)<>"$" THEN PRINT F$ : P$=I$ : GOTO 130
430 PRINT LEFT$(F$,LEN(F$)-1);C$ : P$=I$ : GOTO 130
1000 REM
1001 REM
1002 REM                PALABRAS « CLAVE »
1003 REM
1004 REM
1010 DATA "CAN YOU","CAN I","YOU ARE","YOU'RE","I DON'T","I FEEL"
1020 DATA "WHY DON'T YOU","WHY CAN'T I","ARE YOU","I CAN'T","I AM"
1030 DATA "I'M","YOU","I WANT","WHAT","HOW","WHO","WHERE","WHEN"
1040 DATA "WHY","NAME","CAUSE","SORRY","DREAM","HELLO","HI "
1050 DATA "MAYBE"," NO","YOUR","ALWAYS","THINK","ALIKE","YES"
1060 DATA "FRIEND","COMPUTER","NOENCONTRADONINGUNA"
1100 REM
1101 REM
1102 REM          PARES DE PALABRAS A CONJUGAR
1103 REM
1104 REM
1110 DATA " ARE "," AM "," WERE "," WAS "," YOU "," I "," YOUR "
1120 DATA " MY "," I'VE "," YOU'VE "," I'M "," YOU'RE "," ME "
1130 DATA " YOU "
1200 REM
1201 REM
1202 REM              CODIGOS DE RESPUESTA
1203 REM
1204 REM
1210 DATA 1,3,4,2,6,4,6,4,10,4,14,3,17,3,20,2,22,3,25,3,28,4,28,4,32
1220 DATA 3,35,5,40,9,40,9,40,9,40,9,40,9,40,9,49,2,51,4,55,4,59,4
1230 DATA 63,1,63,1,64,5,69,5,74,2,76,4,80,3,83,7,90,3,93,6,99,7,106
1240 DATA 7
1300 REM
1301 REM
1302 REM                   RESPUESTAS
1303 REM
1304 REM
1310 DATA "DON'T YOU BELIEVE THAT I CAN$"
1320 DATA "PERHAPS YOU WOULD LIKE TO BE ABLE TO$"
1330 DATA "YOU WANT ME TO BE ABLE TO$"
1340 DATA "PERHAPS YOU DON'T WANT TO$"
1350 DATA "DO YOU WANT TO BE ABLE TO$"
1360 DATA "WHAT MAKES YOU THINK I AM$"
1370 DATA "DOES IT PLEASE YOU TO BELIEVE I AM$"
1380 DATA "PERHAPS YOU WOULD LIKE TO BE$"
1390 DATA "DO YOU SOME TIMES WISH YOU WERE$"
1400 DATA "DON'T YOU REALLY$"
1410 DATA "WHY DON'T YOU$"
1420 DATA "DO YOU WISH TO BE ABLE TO$"
1430 DATA "DOES THAT TROUBLE YOU?"
1440 DATA "TELL ME MORE ABOUT SUCH FEELINGS."
1450 DATA "DO YOU OFTEN FEEL$"
1460 DATA "DO YOU ENJOY FEELINGS$"
1470 DATA "DO YOU REALLY BELIEVE I DON'T$"
1480 DATA "PERHAPS IN GOOD TIME I WILL$"
1490 DATA "DO YOU WANT ME TO$"
1500 DATA "DO YOU THINK YOU SHOULD BE ABLE TO$"
1510 DATA "WHY CAN'T YOU$"
1520 DATA "WHY ARE YOU INTERESTED IN WHETHER OR NOT I AM$"
1530 DATA "WOULD YOU PREFER IF WE WERE NOT$"
1540 DATA "PERHAPS IN YOUR FANTASIES I AM$"
1550 DATA "HOW DO YOU KNOW YOU CAN'T$"
1560 DATA "HAVE YOU TRIED?"
1570 DATA "PERHAPS YOU CAN NOW$"
1580 DATA "DID YOU COME TO ME BECAUSE YOU ARE$"
1590 DATA "HOW LONG HAVE YOU BEEN$"
1600 DATA "DO YOU BELIEVE IT IS NORMAL TO BE$"
1610 DATA "DO YOU ENJOY BEINGS$"
1620 DATA "WE WERE DISCUSING YOU, NOT ME."
1630 DATA "OH, I$"
1640 DATA "YOU'RE NOT REALLY TALKING ABOUT ME, ARE YOU?"
1650 DATA "WHAT WOULD IT MEAN TO YOU IF YOU GOT$"
1660 DATA "WHY DO YOU WANT$"
1670 DATA "SUPPOSE YOU SOON GOT$"
1680 DATA "WHAT IF YOU NEVER GOT$"
1690 DATA "I SOMETIMES ALSO WANT$"
1700 DATA "WHY DO YOU ASK?"
1710 DATA "DOES THAT QUESTION INTEREST YOU?"
1720 DATA "WHAT ANSWER WOULD PLEASE YOU THE MOST?"
```

```
1730 DATA "WHAT DO YOU THINK?"
1740 DATA "ARE SUCH QUESTIONS ON YOUR MIND OFTEN?"
1750 DATA "WHAT IS IT THAT YOU REALLY WANT TO KNOW?"
1760 DATA "HAVE YOU ASKED ANYONE ELSE?"
1770 DATA "HAVE YOU ASKED SUCH QUESTIONS BEFORE?"
1780 DATA "WHAT ELSE COMES TO MIND WHEN YOU ASK THAT?"
1790 DATA "NAMES DON'T INTEREST ME."
1800 DATA "I DON'T CARE ABOUT NAMES.  PLEASE GO ON."
1810 DATA "IS THAT THE REAL REASON?"
1820 DATA "DON'T ANY OTHER REASONS COME TO MIND?"
1830 DATA "DOES THAT REASON EXPLAIN ANYTHING ELSE?"
1840 DATA "WHAT OTHER REASONS MIGHT THERE BE?"
1850 DATA "PLEASE DON'T APOLOGIZE!"
1960 DATA "APOLOGIES ARE NOT NECESSARY."
1870 DATA "WHAT FEELINGS DO YOU HAVE WHEN YOU APOLOGIZE?"
1880 DATA "DON'T BE SO DEFENSIVE!"
1890 DATA "WHAT DOES THAT DREAM SUGGEST TO YOU?"
1900 DATA "DO YOU DREAM OFTEN?"
1910 DATA "WHAT PERSONS APPEAR IN YOUR DREAMS?"
1920 DATA "ARE YOU DISTURBED BY YOUR DREAMS?"
1930 DATA "HOW DO YOU DO?  PLEASE, STATE YOUR PROBLEM."
1940 DATA "YOU DON'T SEEM QUITE CERTAIN."
1950 DATA "WHY THE UNCERTAIN TONE?"
1960 DATA "CAN'T YOU BE MORE POSITIVE?"
1970 DATA "YOU AREN'T SURE?"
1980 DATA "DON'T YOU KNOW?"
1990 DATA "ARE YOU SAYING NO JUST TO BE NEGATIVE?"
2000 DATA "YOU'RE BEING A BIT NEGATIVE."
2010 DATA "WHY NOT?"
2020 DATA "ARE YOU SURE?"
2030 DATA "WHY NOT?"
2040 DATA "WHY ARE YOU CONCERNED ABOUT MY$"
2050 DATA "WHAT ABOUT YOUR OWN$"
2060 DATA "CAN'T YOU THINK OF A SPECIFIC EXAMPLE?"
2070 DATA "WHEN?"
2080 DATA "WHAT ARE YOU THINKING OF?"
2090 DATA "REALLY, ALWAYS?"
2100 DATA "DO YOU REALLY THINK SO?"
2110 DATA "BUT YOU ARE NOT SURE YOU$"
2120 DATA "DO YOU DOUBT YOU$"
2130 DATA "IN WHAT WAY?"
2140 DATA "WHAT RESEMBLANCE DO YOU SEE?"
2150 DATA "WHAT DOES THE SIMILARITY SUGGEST TO YOU?"
2160 DATA "WHAT OTHER CONNECTIONS DO YOU SEE?"
2170 DATA "COULD THERE REALLY BE SOME CONNECTION?"
2180 DATA "HOW?"
2190 DATA "YOU SEEM QUITE POSITIVE."
2200 DATA "ARE YOU SURE?"
2210 DATA "I SEE."
2220 DATA "I UNDERSTAND."
2230 DATA "WHY DO YOU BRING UP THE TOPIC OF FRIENDS?"
2240 DATA "DO YOUR FRIENDS WORRY YOU?"
2250 DATA "DO YOUR FRIENDS PICK ON YOU?"
2260 DATA "ARE YOU SURE YOU HAVE ANY FRIENDS?"
2270 DATA "DO YOU IMPOSE ON YOUR FRIENDS?"
2280 DATA "PERHAPS YOUR LOVE FOR FRIENDS WORRIES YOU.".
2290 DATA "DO COMPUTERS WORRY YOU?"
2300 DATA "ARE YOU TALKING ABOUT ME IN PARTICULAR?"
2310 DATA "ARE YOU FRIGHTENED BY MACHINES?"
2320 DATA "WHY DO YOU MENTION COMPUTERS?"
2330 DATA "WHAT DO YOU THINK MACHINES HAVE TO DO WITH YOUR PROBLEM?"
2340 DATA "DON'T YOU THINK COMPUTERS CAN HELP PEOPLE?"
2350 DATA "WHAT IS IT ABOUT MACHINES THAT WORRIES YOU?"
2360 DATA "SAY, DO YOU HAVE ANY PSYCHOLOGICAL PROBLEMS?"
2370 DATA "WHAT DOES THAT SUGGEST TO YOU?"
2380 DATA "I SEE."
2390 DATA "I AM NOT SURE I UNDERSTAND YOU FULLY."
2400 DATA "COME, COME, ELUCIDATE YOUR THOUGHT."
2410 DATA "CAN YOU ELABORATE ON THAT?"
2420 DATA "THAT IS QUITE INTERESTING."
```

Arturo Baeza



PASATIEMPOS

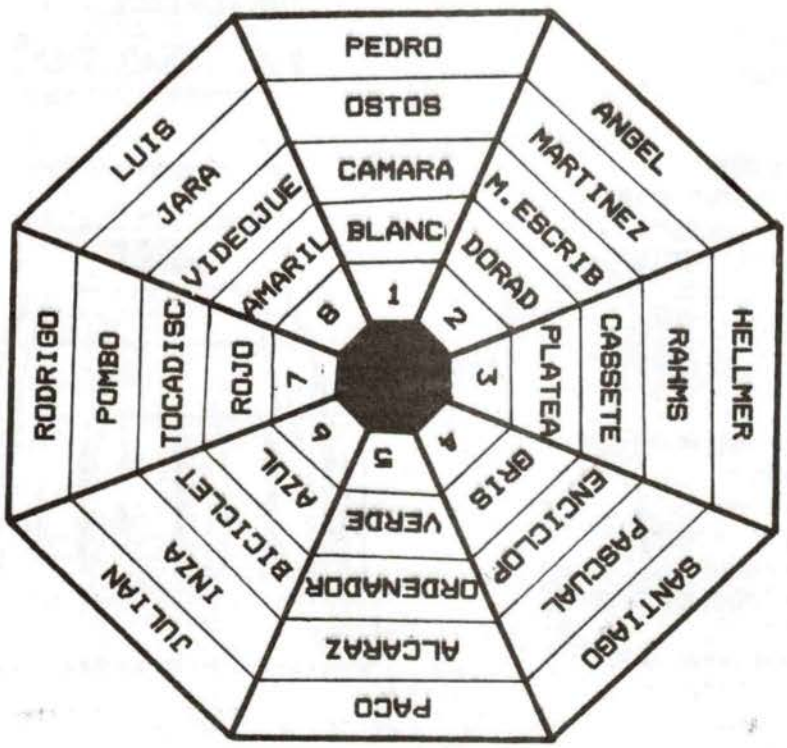
DADOS

MODO DE RESOLVERSE.-
Basandose en el juego de los dados cuyas equivalencias indicamos en la parte inferior, llegue a la obtencion de las puntuaciones que sacaron los jugadores de la partida que le ofrecemos.

VALORES:
As = 6 puntos
Rey = 5 puntos
Dama = 4 puntos
Jota = 3 puntos
Rojo = 2 puntos
Negro = 1 punto

	AS	K	Q	J	ROJ	NEG	TOTAL
OSCAR							
LUIS							
RAUL		20					

1.Oscar y Luis sacaron el mismo numero de Damas.
2.Luis puntuo en rojos igual que Raul en Negros.
3.Oscar saco cuetro Rojos.
4.El jugador que saco cuatro Damas obtuvo seis negros.
5.Luis y Raul lograron el mismo numero de Jotas.
6.El jugador que saco cuatro Negros obtuvo tres Reyes.
7.Raul totalizo 65 puntos.
8.Luis puntuo en Damas igual que Raul en Reyes.
9.El jugador que saco un Rojo obtuvo tres Jotas.
10.Oscar puntuo en negros igual que Raul en Rojos.
11.El jugador que saco cinco Jotas totalizo 57 puntos.
12.Todos los jugadores consiguieron los mismos Ases.
13.Raul saco un punto menos que Luis al final del juego
14.El jugador que no consiguio ningun Rey obtuvo dos Ases.



SOLUCIONES
AL ANTERIOR

CURSO PROLOG

Antes de comenzar esta segunda parte del cursillo, hagamos un poco de memoria de lo visto hasta ahora:

Recordemos que un programa prolog estaba formado por una declaración de cláusulas y que, en el caso de TURBO-PROLOG era necesario además una previa definición de los predicados y de los dominios que se iban a utilizar, debido principalmente a que era compilado.

Además vimos que una cláusula podía ser una REGLA o un HECHO.

Un hecho era un predicado que sabemos que siempre se verifica para los valores constantes de sus argumentos. Si solo se dispusiera en prolog de los hechos, lo único que podríamos hacer con ellos sería consultarlos a modo de base de datos; en cambio las reglas aportan lo más destacado de prolog : su capacidad de inferir nuevo conocimiento a partir del ya conocido.

LAS REGLAS EN PROLOG

Una regla responde al siguiente lema: Un predicado llamado conclusión puede ser inferido, si se verifican a la vez una serie de predicados llamados premisas.

Veamos una regla escrita en prolog:

```
hermano(X,Y) if
  hombre(X) and
  progenitor(X,P) and
  progenitor(Y,P) and
  <>(X,Y).
```

Mediante esta regla se puede inferir una relacion hermano entre dos personas si se es capaz de inferir que X es hombre, y se encuentra un cierto P que sea progenitor (madre o padre) tanto de X como de Y, y claro está, que X sea distinto de Y.

Como se puede apreciar, el predicado que antes llamábamos conclusión es hermano(X,Y) y que el resto de los predicados son las premisas.

Tanto el predicado hombre como progenitor, han de poder verificarse en el programa, bien por cláusulas-hechos, bién como conclusiones de otras cláusulas, mientras que la relación <> es uno de los predicados predefinidos en el prolog, también conocidos como primitivas, que es cierto cuando los argumentos son distintos y

que se puede colocar de una manera más convencional como es X<>Y. Existen facilidades de este tipo para el resto de los signos de desigualdad <,>,>=,<=, y para las expresiones matemáticas como X=Y+Z, etc. no solo en este compilador sino en la mayoría de los traductores de prolog existentes.

Pongamos un programa completo que nos sirva para ilustrar los conceptos que se tratarán en este artículo.

```
        /* REGLAS */
          progenitor(X,P)      if
madre(X,P).
          progenitor(X,P)      if
padre(X,Y).

     hermano(X,Y) if
        hombre(X) and
        progenitor(X,P) and
        progenitor(Y,P) and
        X<>Y.

     hermana(X,Y) if
        mujer(X) and
        progenitor(X,P) and
        progenitor(Y,P) and
        X<>Y.
```

```
/*PROGRAMA ARTICULO II*/

domains
  persomas=symbol

predicates
  hombre(personas)
  mujer(personas)
  padre(personas,personas)
  madre(personas,personas)
progenitor(personas,personas)
  hermano(personas,personas)
  hermana(personas,personas)

clauses

      /*HECHOS*/
  madre(jose,rosa).
  madre(juan,rosa).

  padre(maria,luis).
  padre(juan,luis).

  hombre(juan).
  hombre(jose).
  hombre(luis).

  mujer(maria).
  mujer(rosa).
```

La separación en el programa de hechos y reglas no es necesaria, ya que podrían ir mezclados, aunque no se recomienda.

Lo que no es una casualidad es que todas las cláusulas que hacen verificar un mismo predicado, estén juntas (véase madre, padre, hombre, y progenitor), ya que esto es exigido por el compilador.

LA UNIFICACION O MATCHING

La unificación es el proceso por el cual las variables con que cuenta una cláusula adquieren un valor y además, es el mecanismo por el que se pueden verificar predicados.

1er tipo : CONSTANTE-CONSTANTE

Supongamos que se quiere averiguar, si el predicado hombre(jose) es cierto. Para

esto, el prolog intentará ver si existe alguna cláusula (hecho o regla), que permita verificar el predicado anterior.

La primera cláusula que encuentra es hombre(juan). Al comparar con hombre(jose) se da cuenta de que "juan" es distinto de "jose" y por tanto esta cláusula-hecho no le sirve por lo que continúa con la siguiente:hombre(jose); ahora al comprobar la coincidencia, concluye con que el predicado hombre(jose) es cierto.

A este proceso de comparación es lo que se llama unificación y por tanto se puede sacar como conclusión que un argumento constante se unifica consigo mismo.

2º tipo : CONSTANTE-VARIABLE

Supongamos que se quiere averiguar si el predicado progenitor(juan,rosa) es cierto.

El prolog, igual que antes, buscará en su coleccion de cláusulas, alguna con la que verificar la relación progenitor, encontrando en primer lugar:

progenitor(X,P) if madre(X,P).

Esta regla indica que existe una relación progenitor entre X y P si existe una relación madre entre X y P.

Pues bién, el prolog hace lo lógico: asocia a X el valor "juan" y a P el valor "rosa", quedando la regla anterior convertida en la siguiente:

progenitor(juan,rosa) if madre(juan,rosa).

Ahora el objetivo del prolog convertido en verificar la relación madre(juan,rosa), que siguiendo un proceso de unificación del tipo anterior queda realizado con la segunda claúsula, haciendo cierta la relación buscada.

En conclusión : Una constante se unifica con una variable, adquiriendo ésta el valor de la constante.

3er tipo : VARIABLE-CONSTANTE

Supongamos se quiere averiguar ahora si madre(X,Y) es cierto, donde X e Y son variables con valor no conocido.

El prolog, como en ocasiones anteriores, buscará entre las cláusulas, alguna que sea capaz de verificar una relación madre. La primera que encuentra es madre(juan,rosa), y como X e Y son variables, el prolog concluye con que si X fuera igual a "juan" e Y igual a "rosa", el predicado quedaría verificado, y por tanto, devuelve estos valores para X e Y.

Conclusión: Una variable se unifica con una constante, tomando la variable el valor de ésta.

4º tipo : VARIABLE-VARIABLE

Supongamos que se pretende verificar el predicado progenitor(juan,Prog), donde Prog es una variable de valor desconocido.

El prolog encontrará la regla: progenitor(X,P) if madre(X,P) e intentará mediante ésta, verificar el predicado anterior. Para esto, juan será unificado con X según el 2º tipo estudiado, y Prog al corresponderse con una variable hará que P sea a su vez, una variable de valor no conocido. Por tanto, la regla anterior queda convertida en la siguiente:

progenitor(juan,P) if madre(juan,P).

Ahora, el nuevo objetivo del prolog es madre(juan,P), el cual, mediante el 3er tipo de unificación se concluirá que es

cierto si P=rosa; esto implicará que progenitor(juan,Prog) es cierto si Prog adquiere el mismo valor que P, es decir, rosa.

Una variable cuyo valor en un momento dado, no se conoce se dice que está libre (free), y si éste es conocido, entonces se dice que está ligada (bound).

EVALUACION DE UNA REGLA.BACKTRACKING

Una vez comprendido el mecanismo de unificación; estudiemos cual es la mecánica general de evaluación de una regla en prolog.

El proceso es el siguiente:

Supongamos que se quiere evaluar un predicado determinado. Lo primero que hace el prolog es situarse en la primera cláusula cuyo predicado conclusión sea aquel que intenta, siguiendo el orden en que fueron escritas en el programa.

Una vez encontrada, unifica los argumentos de ambos predicados. Si es posible, entonces, si es un hecho concluye, y si es una regla, se dispone a hacer este mismo proceso con los predicados-premisas que componen la clausula y en el mismo orden en que fueron definidos en ésta.

Si no es posible, busca entonces una nueva cláusula capaz de inferir el predicado.

Este proceso se repite hasta que mediate una cláusula, el predicado quede verificado, o bien, se acaben las cláusulas para ese predicado, en cuyo caso devoverá FALSE.

Situémonos ahora en una regla, por ejemplo:

```
hermano(X,Y) if
  hombre(X) and
  progenitor(X,P) and
  progenitor(Y,P) and
  X<>Y.
```

Supongamos que se intenta probar el predicado hermano(jose,Hermano).

Por unificación la regla es ahora:

```
hermano(jose,Y) if
  hombre(jose) and
  progenitor(jose,P) and
  progenitor(Y,P) and
  jose<>Y.
```

Según lo anterior, lo primero en verificar es hombre(jose) que es unificado con la segunda cláusula que encuentra para este predicado; después intenta unificar P para el segundo predicado tomando P el valor "rosa"; para el tercer predicado se intenta progenitor(Y,rosa) resultando Y=jose y por tanto, cuando llegamos al último predicado nos queda:

jose<>jose -----> FALSO !!

¿Cómo soluciona esto el prolog? con el backtracking o marcha atrás.

Ya que ha resultado fallido un predicado premisa, replanteemos lo que se ha ido averiguando, y veamos si existen otras soluciones, empezando justo por lo último.

Como Y es lo ultimo hallado, se busca otra solución para Y; si ninguna hace que todas las premisas sean ciertas, entonces volvemos más atras, es decir, cambiando P; si aún así, queda infructuoso el proceso, entonces se abandonará la cláusula.

Afortunadamente se da el caso de que progenitor(Y,rosa) tiene una segunda solución : Y=juan, y como jose<>juan, el predicado hermano(jose,Y) es cierto si Y=juan.

El próximo articulo tratará de un apartado muy importante: el CUT.

Autor: Paco Alcaraz

BANDA DE VALENCIA

¿Qué es la Delegación de Alumnos? Un grupo de gente con mucha moral que creen que la situación del alumnado puede mejorar en esta Escuela.

Procuramos hacerlo lo mejor que podemos, nos enfrentamos a los profesores con todo el ímpetu que nos permite el saber que somos nosotros los que estamos por debajo. Es difícil, cuando no cruel, tratar de argumentar con gente que cree estar ya haciéndote un favor con solo escucharte. Es durísimo percibir con tanta claridad el desprecio por la docencia, la sensación de que cuanta más carrerilla cojamos, más gruesa será la pared contra la que chocaremos.

Aún así, no todo es llanto y rechinar de dientes: hay profesores

que nos tratan como personas, que buscan soluciones de compromiso para todas las partes, como la actual directiva de la que disfrutamos. Contamos con una infraestructura organizativa bien establecida, un escaso pero ni mucho menos tanto como solía ser, y el sentimiento de que la gente por lo menos sabe que tenemos un garito en el pasillo de los clubs, que existimos y que pueden acudir a nosotros para intentar que se les solucionen sus problemas.

Tomar decisiones tan delicadas y tan graves como las de las posturas a tomar ante las huelgas de profesores no es tarea fácil. Somos un colectivo hetereogéneo, y la polémica reina en cualquier Junta de Delegados. Quizás a veces adolezcamos de una cierta pasividad por no llegar a ponernos de acuerdo. Sin embargo la alegría de los al final supera a las decepciones. Hoy en día, gracias al buen entendimiento entre Delegación y Jefatura de Estudios, los exámenes suelen estar colocados de un modo relativamente cómodo para la mayoría. Existe un reglamento de revisión de exámenes, y una normativa sobre inasistencia, también de exámenes, que ha supuesto una clara mejora respecto a la situación anterior.

Desde aquí me gustaría hacer una llamada a que la cantera siga produciendo delegados preocupados. A ti que sólo ir a clase te insatisface, que la rutina diaria te deja inquieto, que quieres moverte, ven a Delegación, preséntate este año que viene en tu clase. Apúntate a este trabajo de hormiguita, para poco a poco establecernos como colectivo con derechos, con condiciones de vida y estudio dignas de esta Escuela.

Se hace lo que se puede...

Para este último tramo del curso tenemos previstas las siguientes actividades: (Infórmate en el club).

El último fin de semana de Abril (Días 24 y 25) Miembros del grupo de espeleología se unen al grupo de Standard Eléctrica de la misma especialidad (Uno de los mejores, si no el mejor), para aprender y perfeccionar conocimientos que serán transmitidos al resto del club. Viajan a la cueva de las mulas(Cuenca).

Esta aventura seguirá su desarrollo en mayo como avance y prerparación de una superaventura en una cueva de los Picos de Europa.

Durante el mes de mayo y aprovechando facilidades de la Comunidad de Madrid se realizarán actividades de senderismo (600 - 800 Pts.) y de escalada (1000Pts.) en la sierra de Madrid.

Por último y para los que quieran conocer Alemania este verano en un campo de trabajo, hemos contactado con una organización alemana y quizá alguno pueda ir...

ICARO
Club de nautas y aventuras.

DELEGACION DE ALUMNOS

Una nueva experiencia. Habrás pensado alguna vez que sería conveniente para ti realizar prácticas en empresas para completar tu formación profesional.

Nosotros hemos pensado ello somos ITEM-Colsulting. Tu también puedes ser ITEM-Consulting.

Te presentamos aquí unas ideas sobre nosotros con el objetivo fundamental de animarte a participar en este ambicioso proyecto:

Una Junior Empresa es ante todo una Asociación de alumnos regida por la ley 2248/1968.

Posee una estructura flexible al servicio de las empresas beneficiándose de la ayuda del medio universitario en cuyo seno se desarrolla y estando enteramente gestionada y animada por los estudiantes. Las Junior Empresas están articuladas alrededor de un Consejo de Administración y de una Junta Ejecutiva, elegidas, ambas, por Asamblea General y siendo su misión la de cumplir con la gestión y las funciones de representación y de decisión de la Asociación.

La Junta se compone generalmente de cuatro miembros:

- Un Presidente
- Un Tesorero
- Un Secretario General
- Un Vice-Presidente.

Las actividades de toda Junior Empresa son dirigidas por la Junta Ejecutiva, a través de la cual son solicitados, dentro de los medios profesionales de que se dispone, tipos muy variados de trabajos mediante los cuales los estudiantes tienen la oportunidad de aprender los diferentes aspectos de la vida en una empresa, desde la faceta de decisión hasta la de realización pasando por la estimación, la negociación y a veces por la prospectiva. Estos estudios o trabajos dan la oportunidad de ver aplicaciones muy concretas de las enseñanzas teóricas que se reciben en las aulas, como son:

-Estudios científicos de investigación
-Estudios científicos de aplicación
-Estudios de organización
-Estudios de marketing
-Estudios comerciales

Esta experiencias llevan a los alumnos a evolucionar dentro de un ambiente profesional que de otro modo les sería difícil conseguir mientras siguen, al mismo tiempo con sus estudios.

Cada miembro de la Junior Empresa se enfrenta a numerosos obstáculos que debe aprender a superar.

Problemas de comunicación, de responsabilidad, de conciencia profesional y técnicos son algunos de los problemas que deben ser salvados, tanto individualmente como a través de trabajo de grupo. En cierto modo se trata de una formación complementari a profesional.

Por sus actividades, la Junior Empresa representa un elemento vivo en el medio industrial y comercial. Sus miembros adquieren un

conocimiento real de las reglas del mercado. Supone, por tanto, una aproximación a los problemas concretos de la empresa. Pero además logra una sensibilización especial hacia los problemas humanos, desarrollando toda la eficacia personal que está contenida en sus miembros, los cuales deben tomar responsabilidades; lo que favorece al mismo tiempo sus facultades de adaptación. Por otra parte sus experiencias en la negociación y el trabajo en equipo les permite ser más rápidamente operacionales gracias a sus capacidades de relación.

Por último las Junior Empresas han sido hasta ahora cuna de creadores de empresas habiendo un alto número de estos entre los antiguos colaboradores.

Pues bien, ITEM-Consulting es la Junior Empresa de la E.T.S.I.T. y es miembro de la Confederación Nacional de Junior Empresas de España. Nuestros antecedentes se encuentran, por tanto, en el movimiento de las Junior Empresa francesas de gran auge e importancia en el país vecino, donde se cuentan con más de 100 Asociaciones reunidas en su C.N.J.E y con 15 años de experiencia. Solo en 1985 participaron de sus actividades alrededor de 15.000 estudiantes con un volumen de negocios cercano a los 1.000 Millones de Pesetas.

THE INSTITUTE OF
ELECTRICAL AND
ELECTRONICS
ENGINEERS, INC.

RAMA UNIVERSITARIA

ETSI TELECOMUNICACION
CIUDAD UNIVERSITARIA, S/N
28040 MADRID - ESPAÑA