2ej. 11

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

SOLUCION DE UN PROBLEMA DE ASIGNACION DE RECURSOS UTILIZANDO UN SISTEMA EXPERTO

T E S I S

QUE PARA OBTENER EL TITULO
DE INGENIERO EN COMPUTACION

P R E S E N T A N

GUILLERMO FUENTES HERNANDEZ
FRANCISCO JAVIER LAGUNES TOLEDO

DIRECTOR DE TESIS
ING. JORGE GIL MENDIETA

México, D. F. Diciembre de 1987

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

PROLOGO

Lo que nació como resultado de un requisito para obtener el título a nivel licenciatura de la carrera de Ingeniero en Computación, se ha convertido en nuestro primer paso hacia la industria del software. Nuestros objetivos antes tan vastos como nuestro panorama, se han focalizado en los sistemas expertos. Nuestro desarrollo posterior (estudios de posgrado, especializaciones, etc...), se harán alrededor de este tema con la idea de formar una empresa con un nivel digno de desarrollo tecnológico. Estamos concientes que no es tarea fácil; pero hasta la fecha hemos ido concretizando las metas planeadas, siendo la titulación una de ellas; hecho que marca la terminación de una larga carrera de estudio y el comienzo de otra.

El porque de la elección de este tema estriba en dos aspectos fundamentales. El primero es que pertenece al área de software, única parte de la ciencia de la computación en donde creemos que podemos hacer algo competitivo a nivel mundial, ya que el hardware del primer mundo está años luz de nuestras posibilidades tecnológicas y económicas. El segundo aspecto es porque, simple y llanamente, nos gustan e interesan los sistemas expertos.

Esperamos que este trabajo de tesis pueda servir a los interesados en iniciarse en esta diciplina, como a nosotros ya lo ha hecho, lo que nos hace sentir satisfechos. No nos queda mas que agradecer infinitamente la guía y ayuda desinteresada de nuestro director de tesis, un hombre que por su conocimiento forma parte importante de la familia computacional en México, el Ing. Jorge Gil Mendieta, actual secretario académico del Instituto de Investigaciones en Matemáticas Aplicadas y Sistemas (IIMAS).

Guillermo Fuentes Hernández.
Francisco Javier Lagunes Toledo.

INTRODUCCION

La inteligencia artificial, dentro del inmenso avance de la computación y la electrónica, es un campo abierto al desarrollo de múltiples aplicaciones. Desde simples problemas de inducción hasta robótica y bioingeniería pueden ser resueltos (por lo menos parcialmente), mediante el uso de técnicas de inteligencia artificial. Todo aquel proceso que involucre al conocimiento en forma dinámica es susceptible a usar estas.

Dentro de las ramas de la inteligencia artificial encontramos a los sistemas expertos, que usando un conjunto de técnicas de la primera, nos permiten la solución de problemas específicos a semejanza de como lo haría un humano experto.

En la actualidad los sistemas expertos han tenido cierto auge, sobre todo en aplicaciones muy complejas, como podría ser el control de una planta nuclear. En el presente trabajo se ha elegido un problema específico de asignación de recursos, utilizando una forma de solución novedosa, que es la incorporación de experiencia y conocimiento bajo técnicas de sistemas expertos. El sistema está realizado en los lenguajes de programación turbo-prolog y turbo-pascal, para un ambiente de computadoras personales.

Los capítulos de la tesis están estructurados de manera jerárquica, haciendo mención en el primer capítulo de los métodos que utiliza la inteligencia artificial para la solución de problemas, para lo cual se discute la definición, análisis y técnicas de solución de los problemas en general. Se continúa, en el capítulo dos, con el planteamiento específico de las fases necesarias para la construcción de un sistema experto. Estas fases incluyen la definición del problema que se pretende solucionar, los objetivos que deberá cumplir el sistema, la adquisición del conocimiento necesario y la estructuración de este conocimiento dentro de la computadora por medio de la lógica de predicados o la de posibilidades.

El tercer capítulo es el desarrollo de un sistema experto de acuerdo a lo planteado en el capítulo dos, detallando paso a paso cada una de la fases expuestas.

Por último se presenta un bósquejo de la actividad actual sobre inteligencia artificial a nivel mundial, resaltando los principales grupos de trabajo con sus respectivas tendencias de desarrollo.

Se concluye este trabajo con tres apéndices que incluyen al código fuente del sistema experto, un ejemplo de su funcionamiento y una tabla con los principales sistemas expertos actualmente en uso.

CAPITULO 1

INTRODUCCION A LOS SISTEMAS EXPERTOS

1.1) DEFINICION DE PROBLEMAS

Lo que quizá es la parte medular de toda la ciencia aplicada es la técnica de modelar la realidad, es decir, su representación matemática-teórica; de tal manera que manejando en forma parcial las variables que intervienen en un proceso, podamos hacer una captación momentánea de ésta realidad para así poder entenderla y modificarla.

La definición de un problema es la obtención de un modelo que nos permita manejar éste de una manera más sencilla, ya que si partimos del significado de "definición", el diccionario nos dice: "...enunciación de las cualidades y características de una cosa...", a lo que podríamos reforzar diciendo que también se enmarca dentro de los límites de alguna parte del conocimiento específico del que se trate; mientras que "problema" lo encontramos enunciado como sigue: "...cuestión que se trata de resolver sistemáticamente...", a lo que nosotros dejaríamos como situaciones a las que debemos darle solución para poder así llegar a un objetivo. De todo esto podemos concluir que la definición del problema es la representación cualitativa-cuantitativa, dentro de un marco definido, de una situación reflejo de la realidad que será representada en un modelo.

De aquí se desprende que en cuanto mejor esté definido un problema, más fácil será encontrar su solución. Esta acción en contados casos la encontramos sencilla, en los más se torna complicada. Para la materia en que nos atañe (los sistemas expertos), en donde se tocan todas las caras del conocimiento, la labor puede tornarse realmente difícil si no se hace un estudio previo de todas las formas de representación del conocimiento que se han desarrollado hasta el momento; y aunque éste estudio sólo verá una parte de estas técnicas, es bueno recalcar que cada una tiene características que la van a hacer útil para ciertos problemas y para otros no. La técnica universal no se ha desarrollado aún (si es que existe). En nuestro caso usaremos la técnica de definición de un problema como una búsqueda en el espacio de los estados, siendo ésta muy socorrida dentro de la inteligencia artificial.

La representación del espacio de los estados forma la base de virtualmente la mayoría de los métodos de inteligencia artificial usados en la actualidad. Su estructura corresponde a la de la solución del problema en dos formas importantes:

1) Se permite que una definición formal de un problema convierta alguna situación dada en una situación deseada, usando un conjunto de operaciones permisibles.

2) Permite también definir el proceso de solución de un problema particular como una combinación de técnicas conocidas (representadas como una regla definiendo un simple paso en el espacio), y busca la técnica general de exploración del espacio, tratando de encontrar algún camino del estado actual al estado destino. La búsqueda es un proceso muy importante en la solución de problemas difíciles para los cuales pocas técnicas directas son posibles de utilizar.

Ahora, para definir formalmente un problema, es necesario realizar las siguientes acciones:

1) Definir un espacio de estados que contenga todas las posibles configuraciones de los objetos, y quizá algunas imposibles. Se puede definir todo el espacio sin enumerar todos los estados que contiene, es decir con notación implícita.

2) Especificar uno o más estados iniciales.

3) Especificar uno o más estados, que podrían ser aceptados como soluciones del problema, llamados estados destino.

4) Especificar un conjunto de reglas que describan las acciones (operadores) disponibles. Estas se pueden generar a través de las siguientes cuestiones:

   ¿Qué hechos no se asumen en la definición de los estados, pero están presentes en el problema?

   ¿Qué tan generales deben ser la reglas?

   ¿Qué tanto del trabajo de resolución del problema puede ser preresuelto y dado en reglas?

El problema puede ser resuelto usando las reglas en combinación con una apropiada estrategia de control, moviéndose en el espacio del problema en un camino que lleva del estado inicial al estado objetivo. Esto es, el proceso de búsqueda es fundamentalmente el proceso de solución del problema. Esto, sin embargo, no significa que otros métodos directos no puedan ser aprovechados.

1.2) ANALISIS DE PROBLEMAS

Puesto que la búsqueda forma el núcleo de muchos procesos inteligentes, es útil estructurar los programas de inteligencia artificial de una forma que facilite describir estos procesos. Estas estructuras son provistas por los sistemas de producción.

Un sistema de producción consiste de:

a) Un conjunto de reglas formadas por un lado derecho (un patrón), que determina la aplicabilidad de la regla, y un lado izquierdo, que indica la acción a ser realizada si la regla es aplicada.

b) Una o más bases de datos que contengan toda aquella información apropiada para la tarea en particular. Algunas partes de la base de datos pueden ser permanentes, mientras que otras pueden pertenecer solamente a la solución de un problema específico. Esta información puede ser estructurada de cualquier modo apropiado, siendo ésta de acuerdo a la naturaleza del problema.

c) Una estrategia de control que especifique el orden en el cual las reglas serán comparadas con la base de datos y la manera de resolver los conflictos que se presenten cuando varias reglas puedan ser utilizadas a la vez.

Surge ahora la pregunta de cómo saber que regla es la que se aplicará en el siguiente paso en la búsqueda de la solución del problema, ya que varias reglas pueden ser utilizadas en el estado actual. Las estrategias de control fungen entonces un papel muy importante; pues su eficiencia se verá reflejada en la rapidéz de solución del problema. Sus características principales son las siguientes:

1) Una buena estrategia de control debe causar movimiento, es decir, que no produzca situaciones de no cambio, lo que llevaría a no poder solucionar el problema.

2) Que sea sistemática, para que cubra la necesidad de movimiento global (a lo largo de varios pasos), y movimiento local (durante un mismo paso).

Para resolver problemas difíciles eficientemente, es necesario comprometer los requerimientos de movilidad y sistematicidad junto con la construcción de una estructura de control que nos garantice encontrar una solución aceptable. Es ahí donde la técnica heurística hace su aparición, siendo definida como una técnica que mejora la eficiencia de los procesos de búsqueda, sacrificando la demanda de perfección. Presenta por una parte excelentes resultados en la elección de buenas direcciones hacia la solución; pero es pésima al dirigir en situaciones de no movilidad. Usando técnicas heurísticas apropiadas se pueden obtener buenas (no óptimas) soluciones en problemas difíciles. Hay muchas técnicas heurísticas de propósito general, útiles para todos los dominios y en adición se pueden crear para propósitos especiales que explotan dominios específicos del conocimiento y así resolver problemas particulares.

El hecho de que estas técnicas nos van a evitar el caer en problemas como la explosión combinatorial (es decir cuando las posibles soluciones se producen en forma factorial), hace vasta su aplicabilidad; además de que raramente se necesita

la solución óptima, ya que con una aproximación se pueden obtener excelentes resultados. Se sabe que la gente al resolver problemas no busca soluciones óptimas, sino que se satisfagan sus necesidades. Sin embargo, las aproximaciones producidas por la heurística pueden no ser buenas para las situaciones extremas, aunque estas son difíciles que se presenten en el mundo real.

Para la elección del método o combinación de métodos para resolver problemas particulares, es necesario analizar el problema desde varias dimensiones:

1) Ver la posibilidad de descomponer el problema en subpartes simples y pequeñas, pudiendo aplicar recursividad ó subrutinas.

2) ¿Pueden algunos pasos de la solución ser ignorados o anulados si se prueba su inutilidad? Siendo importante reconocer tres posibles situaciones a presentarse:

   2.1) Pasos ignorables: cuando los pasos pueden no tomarse en cuenta.

   2.2) Pasos recuperables: cuando los pasos de solución pueden ser desconocidos.

   2.3) Pasos irrecuperables: cuando los pasos de solución no pueden ser desconocidos.

3) ¿Se puede predecir el universo del problema? Hay problemas en los que se puede saber exactamente qué sucederá con una acción determinada, es decir, la salida es conocida dada cualquier entrada, por lo que se puede planear el construir caminos que nos lleven a la solución; mientras que para problemas con salida incierta el realizar planes de solución resulta difícil, puesto que no sabemos con exactitud el estado de problema en el siguiente paso, por lo que hay que hacer uso de la probabilidad y los posibles caminos aumentan en forma exponencial.

4) ¿Es evidente que la solución correcta no necesita comparación con todas las demas posibles soluciones?, o dicho de otra forma, la solución es absoluta o relativa. Una solución a un problema puede contar con todas las características para ser en sí una de ellas; pero eso no significa que sea la mejor, por lo que la heurística no puede ser usada para los casos en donde se busque la mejor solución, y entonces se deben usar métodos mas sofisticados para encontrarla .

5) ¿Es la base de conocimientos para resolver el problema internamente consistente? Esto significa que los conocimientos contenidos en la base son coherentes y nos llevan a una solución correcta, puesto que una base de conocimientos no consistente nos puede llevar a resultados "logicamente correctos" pero incoherentes.

Los sistemas no consistentes tendrán que ser ayudados por el exterior.

6) ¿Es necesaria una gran cantidad de conocimiento para resolver la problemática, o sólo para comprimir la búsqueda?. Para algunos casos las reglas de conocimiento nos mostrarán el camino para encontrar la solución, junto con mecanismos de control, estrategias y tácticas. En otros las reglas unicamente nos irán enmarcando el resultado dentro del universo.

7) ¿Es necesaria la interacción entre la computadora y la persona para resolver el problema, o puede hacerlo la computadora por si sola? Debemos distinguir entre dos tipos de problemas: solitarios, en los que la computadora nos dará una descripción del problema, producirá una respuesta sin comunicación intermedia y no nos explicará el razonamiento del proceso. Conversacionales, en los cuales habrá comunicación intermedia entre la persona y la computadora, proveyéndose asistencia o informacion adicional a la máquina, a la persona o a ambas.

Hasta aquí se han examinado un conjunto de características para distinguir distintas clases de problemas, y se ha dicho que los sistemas de producción es una buena herramienta para describir las operaciones que nos lleven a la solución de un problema. Estos sistemas pueden poseer una serie de características que nos vislumbren la manera en la que pueden ser implementados. Es aquí donde introducimos las siguientes definiciones de sistemas de producción:

a) Sistema de producción monotónico.
Es aquel en el cual la aplicación de una regla nunca prevee la última aplicación de otra regla, que podría haber sido aplicada al mismo tiempo en que la primera regla fue seleccionada.

b) Sistema de producción parcialmente conmutativo.
Es aquel que tiene la propiedad de que si la aplicación de una secuencia particular de reglas transforma un estado X en uno Y, entonces cualquier permutación de esas reglas, que sean permitidas, realiza la misma transformación.

c) Sistema de producción conmutativo.
Es aquel que tiene las dos características anteriores.

Ahora es interesante conocer la relación entre los tipos de problemas y los sistemas de producción, para entonces resolverlos de la mejor manera posible. Para cualquier problema existen una infinidad de sistemas de producción que nos ayudan a encontrar soluciones, algunos más eficientes que otros. En el sentido formal no es posible encontrar tal relación, debido a que todos los problemas pueden ser resueltos con todo tipo de relaciones, pero en la práctica si pueden encontrarse, lo que nos permitirá hallar la solución

más eficientemente. Algunas de las relaciones más comunes son:

a) Sistemas de producción monotónicos parcialmente conmutativos son útiles para resolver problemas ignorables, en los cuales una formulación natural nos mostrará los pasos que pueden ser ignorados en la solución. Son problemas en los que por general se obtienen nuevas cosas, cambiando otras antiguas y son del tipo de los ignorables, por ejemplo, prueba de teoremas. Este tipo de sistemas de producción son útiles también por su capacidad de no necesitar el paso anterior cuando ya se ha encontrado un camino correcto a la solución. Esto resulta principalmente ventajoso para sistemas que necesitan ver hacia atrás para llegar al estado destino, en este caso, las bases de datos no necesitan ser restauradas, incrementándose la eficiencia.

b) Sistemas de producción no monotónicos parcialmente conmutativos se caracterizan porque los cambios ocurridos pueden ser invertidos, no siendo crítico el orden de las operaciones.

c) Sistemas de producción parcialmente conmutativos son significativos para las implementaciones en donde es necesario se vigile la duplicación de estados durante el proceso de búsquedas por motivos de economía.

d) Sistemas de producción no parcialmente conmutativos útiles en aquellos problemas en los que ocurren cambios irreversibles, por ejemplo, para problemas químicos.

1.3) TECNICAS DE SOLUCION DE PROBLEMAS (1)

La solución de un problema es un proceso de búsqueda de un camino que nos lleve al estado destino. Existen infinidad de estos procesos, entre los que se encuentran variedades de la búsqueda heurística. Estas técnicas son llamadas métodos débiles (Weaks methods) debido a su característica de que aunque pueden ser utilizados para cualquier tarea en particular o en cualquier dominio del problema, su eficacia en la mayoría de las veces es altamente dependiente de la forma en que explotan el dominio específico de conocimiento fuera y dentro de ellos, siendo, por ejemplo, incapaces de vencer la explosión combinatorial en la que caen estos procesos de búsqueda. A pesar de ello, estos métodos continúan siendo el esqueleto de la inteligencia artificial actual.

Cada proceso de búsqueda puede ser visto como una travesía de una gráfica dirigida, en la cual cada nodo representa un estado del problema y cada arco una relación entre los estados representados por los nodos que conecta. Los procesos de búsqueda deben encontrar un camino a través de la gráfica comenzando en un estado inicial y concluyendo en uno o más estados finales. Esta gráfica debe ser conformada en

principio a partir de las reglas que definan movimientos válidos dentro del espacio del problema. Sin embargo, en la práctica no es necesario esto, ya que en lugar de contruir la gráfica explícita y buscar en ella, la mayoría de los programas de búsqueda representan a la gráfica implícita en las reglas, generando explícitamente solo aquellas partes que se ha decidido explorar.

Entre las características de los procesos de búsqueda, se encuentra la dirección en el que estos deben proceder: Hacia adelante, a partir del estado inicial, o hacia atrás, a partir del estado destino. Esto es debido a que el modelo de sistemas de producción de los procesos de búsqueda puede ser visto como procesos simétricos hacia atrás o hacia adelante. En el razonamiento hacia adelante, el lado izquierdo (las precondiciones), estan en correspondencia contra el estado actual, y el lado derecho (los resultados), son usados para generar nuevos nodos a representar hasta que el objetivo es encontrado. En el razonamiento hacia atrás, el lado derecho es puesto en correspondencia con el estado actual y el lado izquierdo es usado para generar nuevos nodos, representando nuevos estados finales a ser alcanzados; ésto continúa hasta que uno de estos estados se encuentra marcado como un estado inicial. Para elegir cual de los métodos hay que usar existen tres factores que nos pueden ayudar:

1) Observar si hay más estados iniciales o finales, debido a que es más fácil comenzar del que tiene menor número, para encontrar a los que se dirige, aumentando así la eficacia de la búsqueda.

2) Verificar en que dirección el factor de ramificación (es decir el número promedio de nodos que pueden ser alcanzados directamente de un nodo simple) es mayor. Debemos proceder en la dirección donde éste sea menor.

3) Si el proceso entablará comunicación con el usuario para justificar las razones de su procedimiento, entonces debemos elegir la dirección que se acerque más al camino que escogería el usuario.

4) Es posible tambien usar una técnica llamada bidireccional, en donde se comienza desde el estado inicial y el final simultáneamente, hasta que los dos caminos se encuentren, pero esta técnica presenta casos de aumento exponencial de los nodos a cada paso, por lo que puede ser poco efectiva.

Pasamos ahora a la topología de los procesos de búsqueda. Un camino simple para implementar una estrategia de búsqueda es por medio de un árbol. Cada nodo de éste es expandido por las reglas de producción, que van a formar un conjunto de nodos sucesivos, pudiendo cada uno de estos ser expandidos hasta que alguno represente a una solución. No obstante, con frecuencia resulta que un mismo nodo es generado varias veces durante el proceso, formando parte de varios caminos, el proceso entonces comienza de nuevo cada vez que esto sucede

porque no se utilizan adecuadas técnicas de control y se degenera el árbol en una gráfica arbitrariamente dirigida. Este problema puede ser resuelto convirtiéndolo en una gráfica dirigida, que difiere del árbol en que varios caminos pueden llegar juntos a un mismo nodo. Un procedimiento de búsqueda de árbol puede ser convertido en una búsqueda de gráfica, modificando la acción de generación de nodos en cada paso, aplicando las siguientes reglas:

1) Examinar el conjunto de nodos que han sido creados para ver si el posible nuevo nodo existe.
2) Si no existe, adicionarlo a la gráfica.
3) Si existe:
   3.1) Establecer la liga entre el nodo idéntico y desechar el nuevo.
   3.2) Para el caso de estar buscando la mejor solución, entonces habrá que hacer una comparación entre los caminos y desechar el que presente la peor opción, usando criterios específicos para cada problema.

Aparece entonces el problema de los ciclos en la búsqueda gráfica, siendo los ciclos caminos en los que los nodos aparecen más de una vez. Esto dificulta la forma de probar que la búsqueda gráfica pueda terminar.
Tratar a los procesos de búsqueda como una gráfica dirigida reduce el esfuerzo que se invierte en buscar por los mismos caminos varias veces, como sucede en las búsquedas de árbol, pero se requiere esfuerzo adicional cada vez que un nodo es generado y se investigar si existe o no.
Los procedimientos de búsquedas gráficas son útiles especialmente para relaciones con sistemas de producción parcialmente conmutativos (donde no importa el orden de aplicación de las reglas).

Hasta ahora hemos representado a la búsqueda de la solución como el proceso de movimiento a través de una gráfica ó un árbol donde cada nodo representa un punto en el espacio del problema; surge ahora el cómo representar un nodo individual. Para problemas complejos la situación se torna complicada y debemos entonces centrarnos haciendo las siguientes observaciones:

1) ¿Cómo pueden ser representados los objetos y los hechos individuales?
2) ¿Cómo pueden ser combinados las representaciones de los objetos individuales para que representen un estado completo del problema?
3) ¿Cómo pueden las secuencias de los estados del problema, que surgen en los procesos de búsqueda, ser representados eficientemente?

Las primeras dos cuestiones pertenecen al campo de representación del conocimiento, que analizamos en el capítulo dos. Existen unas pocas técnicas con amplia aplicabilidad que pueden ser útiles, como es el caso de la lógica de predicados, centrada en la idea de que el

conocimiento puede ser representado como un conjunto de objetos, cada uno con atributos propios y un conjuntos de relaciones con otros objetos.

La tercera pregunta adquiere particular importancia en el contexto de los procesos de búsqueda. Supóngase que representamos cada nodo como una lista de características que lo describen, pero, ¿qué pasa si durante el proceso de búsqueda cada descripción es muy larga?, entonces cada descripción será representada de nuevo para cada nodo, por lo que pronto estaremos fuera de nuestra capacidad de memoria. Además gastaremos todo nuestro tiempo creando y actualizando estos nodos. El problema de resolver cuales características son diferentes en cada uno de los nodos es realmente complicado. Todo este problema de representación de los nodos, tanto los que cambian y los que no, es conocidos como el problema de enmarcación (frame problem). En algunos casos sólo en las partes difíciles son representados todos los datos para facilitar el manejo al tener mas información, en otros sólo los que tienen cambio son considerados como los importantes. En realidad no hay una respuesta simple para los problemas de representación del conocimiento ó del problema de enmarcación, sino unicamente técnicas que nos facilitarán la estructuración del dominio del problema.

1.3.1) APAREO.

Hemos descrito los proceso de solución de problemas usando la búsqueda como la aplicación de reglas apropiadas a problemas de estado individuales para generar nuevos estados, en los cuales las reglas puedan ser aplicadas, y así encontrar una solución. Se ha sugerido que las búsquedas más eficientes envuelven el escoger entre las reglas posibles la mejor y dirigirnos con ella a la solución. Para hacer esto se requiere algún tipo de apareo entre el estado actual y las precondiciones de las reglas, siendo esto crítico para los sucesos de los sistemas basados en reglas. A continuación se muestran algunas opciones de apareo.

1) INDEXACION.

Un camino para seleccionar la regla a aplicar es hacer una búsqueda simple a través de todas las reglas, comparando cada una de las precondiciones del estado actual y extraer la que sea igual, pero se presentan dos problemas:

a) Para problemas muy complejos es necesario aplicar gran cantidad de reglas. La búsqueda a través de todas ellas en cada paso podría ser muy ineficiente.

b) No es siempre obvio saber si las precondiciones de las reglas son satisfechas por un estado en particular.

Para algunos casos es muy fácil enfrentar el primero de los problemas. En lugar de buscar a través de las reglas, usamos al estado actual como índice dentro de las reglas y seleccionamos la pareja inmediatamente. Desafortunadamente el

esquema simple de indexación sólo trabaja cuando las precondiciones de las reglas tienen la forma de las configuraciones de tablero, por ejemplo, el juego de ajedrez. Los procesos de apareo son sencillos sólo en el caso de sacrificar la generalidad en las declaraciones de las reglas, que como se ha visto es mejor escribirlas en la forma más general posible. Es decir que la indexación no es útil siempre que las precondiciones de las reglas son declaradas como predicados de alto nivel. En muchos sistemas de pruebas de teoremas, por ejemplo, las reglas son indexadas por los predicados que contiene, por eso todas las reglas que podrían ser aplicadas para probar un hecho en particular, pueden ser rápidamente accesadas. A pesar de estas y otras limitaciones de esta aproximación, la indexación es una forma eficiente para la operación de sistemas basados en reglas.

2) APAREO CON VARIABLE

Algunas veces el problema de seleccionar la regla aplicable se presenta más difícil que el sólo hecho de encontrar la forma de ignorar el grueso de las reglas e ir inmediatamente a aquella que podría ser apropiada. Puede no ser sencillo examinar un problema en particular y un estado del problema dado y determinar si la precondición de la regla es satisfecha. El problema surge aquí al igual que en la indexación, donde las precondiciones no se definían como descripciones exactas de situaciones particulares, siendo que describen propiedades de compleja variación que las situaciones deben tener. Muchas veces esto arroja que el descubrir si hay una relación entre una situación en particular y las precondiciones de una regla dada, implica un proceso de búsqueda significante. De un sencillo tipo de apareo no literal podemos requerir búsquedas extensas, surgidas éstas cuando los patrones contienen variables. Otros dos problemas pueden surgir del apareo no literal. El primero es que es usualmente importante grabar no sólo que una pareja fué encontrada entre un patrón y una descripción de estado, sino que también esas uniones fueron realizadas durante el proceso de apareo de modo que estas puedan ser usadas en la parte de la regla que denota acción. El segundo problema a considerar en el apareo no literal es que una regla simple puede relacionar el estado del problema actual en más de una forma, esto nos dirige a varias alternativas de acción de la derecha de la regla. Otra cosa importante es el hecho de que el número de estados que pueden ser generados como subsecuentes de un estado en particular, no esta dado por el número de reglas que pueden ser aplicadas, sino más bien por el número de caminos de todas las reglas que pueden ser aplicadas.

3) APAREO COMPLEJO Y APROXIMADO

Procesos más complejos de apareo son requeridos cuando las precondiciones de una regla especifica propiedades que no están declaradas explícitamente en la descripción del estado actual. En este caso un conjunto aparte de reglas deben ser usadas para describir cómo de algunas propiedades se pueden

inferir otras. De igual forma procesos de apareo más complejos son requeridos cuando las reglas deben ser aplicadas sí las precondiciones se relacionan aproximadamente a la relación actual. Esto sucede con frecuencia para los casos que envuelven descripciones físicas del mundo. El apareo por aproximación es particularmente difícil debido a que incrementamos la tolerancia permitida en el apareo, incrementando también el número de reglas para realizarlo, esto conlleva a un aumento en los procesos de búsqueda. Hay tambien que reconocer que el apareo por aproximación nunca será superior al apareo exacto en situaciones como la comprensión de discursos, donde éste da como resultado tratar de aparear sin que haya reglas, lo que hace los procesos de búsqueda realmente azarosos. Para algunos problemas, casi toda la acción se encuentra en el apareo de las reglas con el estado del problema. Una vez realizado, muy pocas reglas son aplicadas por lo que permanecer en la busqueda es trivial.

Aunque los procesos de apareo en realidad no presentan gran problema la mayoría de las veces, hay que recordar el problema de enmarcación (frame problem). Una forma de manejarlo es no almacenar las descripciones de los estados completamente para cada nodo, sino almacenar sólo los cambios del nodo previo. Sí esto se realiza, el proceso de apareo debe ser modificado para escudriñar hacia atrás, a partir de un nodo, sus predecesores, buscando los hechos referidos.

El resultado de los procesos de apareo es una lista de reglas cuyo lado izquierdo ha sido relacionado con la descripción del estado actual, junto con las variables que hayan sido generadas por el proceso, siendo el trabajo de búsqueda decidir en que orden serán aplicadas las reglas; pero algunas veces es útil incorporar algunas de esas decisiones dentro del proceso de apareo. Esta fase del proceso es llamado resolución del conflicto. Por ejemplo, suponga que algunas de las reglas de un sistema trata con situaciones que son casos especiales de las situaciones cubiertas por otras reglas. Esta reglas deben por lo general tener mayor prioridad que cualquiera más general. El propósito de dicha especificidad es dar el tipo de conocimiento que aplica el experto al solucionar problemas sin utilizar procesos de búsqueda. Sí nosotros consideramos todas las reglas que relacionamos, entonces la suma de todas aquellas de proposito especial incrementarán la busqueda en vez de decrementarla. Para prevenir ésto hay que construir un deshechador de reglas más generales durante el proceso: para lograrlo hay pocas maneras de hacerlo, entre ellas tenemos las dos siguientes:

1) Sí el conjunto de precondiciones de una regla contiene todas las precondiciones de otra y algo más, entonces la primera regla es mas general que la segunda.

2) Sí las precondiciones de una regla son las mismas que de otra, excepto que en en el primer caso las variables son especificadas y en el segundo las mismas son constantes, entonces la primera regla es mas general que la segunda.

La forma exacta de adoptar estos criterios depende de la forma en que esten escritas las precondiciones.

Otra forma en la cual el proceso de apareo puede aligerar el peso en los mecanismos de búsqueda, es ordenar las relaciones encontradas basándose en la importancia de los objetos. Por ejemplo, en la entablación de diálogos con el computador, se le puede dar rango de importancia a cada palabra y a partir del peso de cada una continuar el diálogo por el rumbo donde adquiere más importancia (ver lógica de posibilidades, capitulo dos).

Existe también otro modo de utilizar la prioridad de apareo manejándolo como una función de la posición de los objetos relacionados en la descripcion del estado actual. La forma en que el apareo interactúa con los procesos de búsqueda depende de la estructura de ambos, pero es también cierto que en el primer proceso se utiliza cierta clase de información que es útil en el segundo, por lo que una comunicación entre ambos sería muy recomendable.

Cuando el proceso de apareo en un sistema basado en reglas no es directo, éste puede requerir alguna búsqueda. Los mismos procedimientos de búsqueda usados en el último nivel de solución del problema pueden usarse en el apareo y ésto, eventualmente usándolo en forma recursiva, puede terminar en un proceso directo de apareo.

1.3.2) FUNCIONES HEURISTICAS

Se tiene definida a la heurística como una técnica que auxilia en el descubrimiento de las soluciones de los problemas. Hay técnicas heurísticas con un vasto campo de aplicación, así como otras que representan a campos específicos del conocimiento que son relevantes en la solución de algunos problemas. Hay dos formas en cuyo dominio específico la información heurística puede ser incorporada dentro de un procedimiento de búsqueda basado en reglas:

1) En las reglas mismas. Por ejemplo, las reglas de un sistema de juego de ajedrez deben no solo describir el conjunto de movimiento permitidos, sino también los movimientos tácticos.

2) Como una función heurística que evalúa problemas individuales de los estados y determina que es lo más deseable a hacer.

Ahora podemos definir una función heurística como aquella que mapea las descripciones del estado del problema con alguna medida de deseabilidad, usualmente representada con números. ¿Cuáles aspectos del estado del problema son considerados?, ¿cómo son evaluados y que peso es dado a aspectos individuales?. Esto se escoge de tal forma que el valor de la función heurística en un nodo dado en el proceso de búsqueda dá una buena estimación de la posibilidad de que el nodo está en un camino deseable hacia la solución.

Funciones heurísticas bien diseñadas juegan un papel muy importante en la guía para que el proceso de búsqueda llegue eficientemente a la solución. Algunas veces funciones heurísticas muy simples pueden dar un buen estimado de si un camino es bueno o no. En otras situaciones pueden usarse funciones heurísticas más complejas. Hay que remarcar el hecho de que en ocasiones el valor alto en las funciones heurísticas puede indicar una buena posición, mientras que en otras un bajo valor puede representar una situación ventajosa. No importa entonces la forma en que la función esté definida, ya que el programa puede minimizar ó maximizar el valor.

El propósito de una función heurística es el guiar al proceso de búsqueda en la dirección más recomendable, sugiriendo qué camino seguir cuando más de uno está disponible. Lo más correcto de una funcion heurística es estimar los méritos reales de cada nodo en la búsqueda del árbol ó gráfica, el más directo será la solución del proceso. En un caso extremo, la función heurística podría hacer que no necesitemos búsqueda alguna. El sistema podría moverse directamente a la solución. Pero para muchos procesos el costo de computación en dicha función sobrepasa al valor que se ahorra en el proceso de búsqueda. Después de todo, sería posible tener una función heurística perfecta realizando una búsqueda completa para el nodo en cuestión, siempre y cuando apunte hacia una buena solución. Por lo general, hay una relación entre el costo de evaluación de una función heurística y el ahorro en tiempo de búsqueda que la función provee.

1.3.3) ESTRATEGIAS BASICAS DE BUSQUEDA

Siendo la búsqueda heurística una herramienta poderosa para la solución de problemas difíciles, la estrategia usada es con frecuencia crítica para determinar con que tanta efectividad será resuelto un problema. La elección de la estrategia correcta para un problema particular depende en gran parte de sus características. Siendo los métodos débiles (weak methods) de propósito general, no van a presentar solución para todos los problemas por ellos mismos, sino que son presentados como esqueleto para, usando conocimiento específico, ser organizados y explotados. A continuación se presenta una tabla con los más importantes. No pretendemos explicarlos sino unicamente dar una descripción somera de ellos.

1.3.3.1) BUSQUEDA A PROFUNDIDAD (DEPTH FIRST SEARCH)

Esta estrategia es la mas simple de todas las aproximaciones y consiste de los siguientes pasos:

1) Generar una solución posible. Para algunos problemas estos significa la generación de puntos particulares en el espacio del problema, en otros significa generar un camino a partir del estado inicial.

2) Probar por comparación si es una solución del nodo marcado ó del nodo final del camino marcado con el conjunto de estados finales aceptados.

3) Si la solución ha sido encontrada, concluye la búsqueda, de otro modo, comenzar de nuevo.

Si la generacion de posibles soluciones es hecha sistemáticamente, este procedimiento encontrará la solución si ésta existe, desafortunadamente si el espacio del problema es muy grande, nos puede ocupar un largo tiempo encontrar el resultado. Este algoritmo es un procedimiento de búsqueda inductivo, hay que generar las soluciones completas antes de ser probadas. La forma más simple para implementar procedimientos sistemáticos es tratándolo como una búsqueda de árbol inductiva (depth-first), con recorrimiento hacia atrás (backtracking).

1.3.3.2) BUSQUEDA POR ESCALAMIENTO (HILL CLIMBING)

El procedimiento de escalamiento es una variante del anterior en el cual, lo obtenido durante el procedimiento de prueba, es utilizado durante la parte de generación para encontrar la dirección en que se moverá dentro del espacio de los estados. Esto se logra aumentando durante la parte de prueba una función heurística que dará un estimado de que tan cerca se encuentra el estado actual del estado objetivo. Se tiene la ventaja sobre el anterior de utilizar menos recorridos para encontrar el camino a la solución.

1.3.3.3) BUSQUEDA POR AMPLITUD (BREADTH FIRST SEARCH)

En este proceso, todos los nodos de un árbol son revisados antes de continuar en el siguiente nivel. Es fácil considerar a este proceso como de búsqueda de arbol; pero puede tranformarse en un proceso de gráfica dirigida por medio del método anteriormente visto.

Este método garantiza el encontrar una solución, si es que existe, dado que existe un número finito de ramas del árbol, y que serán revisadas todas y cada una de ellas. Por el mismo razonamiento sabemos que la primera solución encontrada será la más corta (aunque esto no significa que sea la mejor). Existen tres problemas principales en este tipo de búsqueda:

1) Se requiere gran cantidad de memoria, pues el número de nodos aumenta factorialmente.
2) Requiere gran cantidad de procesamiento.
3) Operaciones irrelevantes ó redundantes son realizadas.

Este tipo de búsqueda es inapropiada para los casos en los que se hallan muchos caminos a la solución y estos son muy largos.

1.3.3.4) BUSQUEDA MEJORADA (BEST FIRST SEARCH)

Este proceso es una combinación de los dos anteriores. A cada paso de éste, seleccionaremos el nodo más promisorio de los que hallamos generado hasta entonces, aplicando una apropiada función heurística en cada uno de ellos, expandiendo entonces el nodo elegido usando las reglas para generar a sus sucesores. Si alguno de ellos es una solución, se termina la búsqueda, si no, todos los nodos generados son adicionados al conjunto general de nodos eligiendo el mejor y continuando con el proceso.
Existen dos variantes de ésta búsqueda, una es por las llamadas gráficas-o (or-graphics), y la otra por la técnica llamada de agenda.

Existe otro tipo de estructura que también es útil para la representación de soluciones que ya se había mencionado antes, la gráfica tipo árbol y-o (and-or tree graphic). Es especial para aquellas situaciones que pueden ser resueltas por medio de la descomposición de problemas en un conjunto de subproblemas, en donde todos y cada uno de ellos deben de ser solucionados de antemano, para así llegar al nodo destino. Existen métodos específicos para solucionar este tipo de formación, uno de ellos es el llamado algoritmo AO*.

1.3.3.5) BUSQUEDA POR SATISFACCION OBLIGADA (CONSTRAINT SATISFACTION)

Muchos problemas de inteligencia artificial pueden ser resueltos usando este procedimiento cuyo objetivo es el de descubrir algún estado del problema que satisfaga un conjunto dado de obligaciones. Los diseños así tratados no deben tener límites fijos en lo que respecta al uso de tiempo, costo y materiales. Los problemas así manejados no requieren de un nuevo proceso de búsqueda, pueden utilizarse para éste fin cualquiera de las estrategias de búsqueda hasta aquí expuestas. Sin embargo, para la estructura de estos problemas, es útil el aumentar la descripción del estado del problema con una lista de obligaciones que cambian de acuerdo a la parte del problema a resolverse, y junto con éste aumentar también el mecanismo de búsqueda para así manejar esta lista. Así entonces, el proceso simple a la solución se convierte en dos búsquedas concurrentes, una de la cuales opera en el espacio del problema de la lista de obligaciones y el otro en el espacio original.

1.3.3.6) BUSQUEDA POR ANALISIS DE SIGNIFICADOS FINALES (MEANS ENDS ANALISYS)

Este proceso se centra en las diferencias entre el estado actual y el estado objetivo. Una vez aislada la diferencia, un operador que pueda reducirla debe ser encontrado. Puede ser quizá que ninguno pueda ser aplicado, entonces se genera un subproblema a solucionar. Puede ser entonces que la reducción no nos dé exactamente el estado destino, para lo cual se genera un segundo subproblema para subsanar esta diferencia. Este método puede ser aplicado recursivamente.

Hasta aquí se han visto algunos de los llamados métodos débiles (weaks methods), los cuales pueden ser aplicados en una variedad inmensa de casos. Cada uno, como se vió, tiene características que lo hacen propio para cada tipo de problema. En algunos casos estos métodos pueden ser combinados en problemas realmente difíciles y separarlos en subrutinas independientes.

(1) Rich E., 1983.
Randall D., 1982.

CAPITULO
2

ESTRUCTURA BASICA DE UN SISTEMA EXPERTO

En el siglo 17 el hombre soñaba con volar, veía a las aves y se preguntaba el como podría imitarlos. Comenzó a inventar aparatos y dispositivos con alas que podían moverse para simular el aleteo de los pájaros. Sobra mencionar que cada individuo utilizando un armazón con alas, y moviendo frenéticamente los brazos de arriba a abajo después de haberse arrojado a un precipicio, terminaba gravemente lesionado.

Posteriormente se dió cuenta de que para poder volar, no necesariamente tenía que imitar todos los movimientos de aleteo de las aves, y en la actualidad tenemos estupendos aviones que cumplen con su ojetivo sin necesidad de aleteo alguno.

La moraleja de esta pequeña historia con relación a los sistemas expertos, es el cuestionarse si estos deben tratar de duplicar los mecanismos de la inteligencia humana, ó es suficiente el que sólo produzcan un comportamiento inteligente para aconsejar a los humanos expertos ó enseñar su conocimiento a los neófitos.

En este capitulo analizaremos los componentes que forman a un sistema experto partiendo de las siguientes dos definiciones:

Sistema (desde el punto de vista computacional):

> Programa de computadora compuesto de uno o varios módulos, cada uno de los cuales realiza una función especifica pero se relaciona con otros módulos del sistema.

Experto (desde el punto de vista humano):

> Persona con amplios conocimientos sobre un tema que analiza y resuelve problemas sobre ese tema.

De estas dos definiciones podemos extraer varias interrogantes:

a) ¿Qué tipo de problemas resuelven los expertos?
b) ¿Cómo se efectúa el análisis y solución de un problema?
c) ¿Qué implica el tener "amplios conocimientos"?
d) ¿Qué tipo de módulos serán los de un sistema experto?
e) ¿Cuáles son las funciones que tienen que realizar estos módulos?

La figura 2.1 es un esquema de la estructura básica de un sistema experto. Podemos apreciar elementos internos y externos. El experto humano que soluciona problemas junto con el usuario que interactúa con el sistema forman los elementos externos de la estructura. La base de conocimientos, la máquina de inferencias, la estrategia de control y la interface hombre-máquina son los elementos internos.

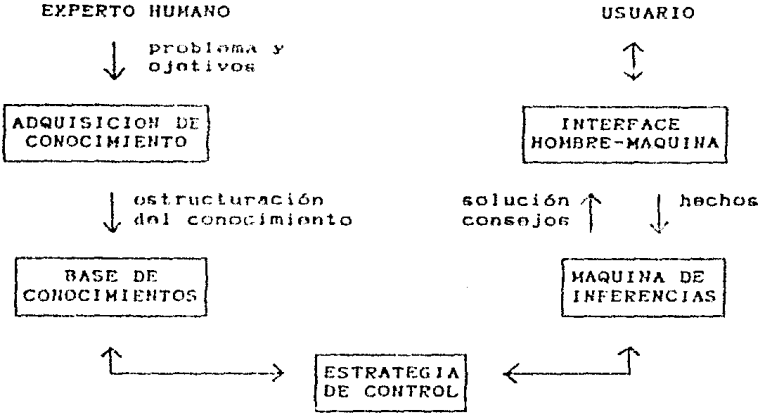


Figura 2.1: Estructura básica de un sistema experto

En base a este esquema discutiremos el tipo de problemas viables a ser resueltos por un sistema experto, así como los objetivos que se pretenden conseguir durante la solución del problema, y se detallan los cuatro módulos básicos de un sistema experto con las funciones que desempeñan cada uno de ellos.

2.1) El problema y los objetivos

Antes de diseñar un sistema debemos contar con un problema al cual queramos resolver. Para efecto de un sistema experto, el tipo de problema a solucionar se enmarca dentro de un esquema experto. ¿Cómo reconocer a este tipo de problemas? Sugerimos analizar algunas características con las que los podemos identificar, ya sea por una o varias de ellas:

CARACTERISTICAS DE LOS PROBLEMAS EXPERTOS

- No existe una fórmula mágica para solucionarlo. Problemas donde no es posible ir a un libro para seleccionar un algoritmo único de solución al problema

planteado. Por ejemplo los procesos legales, análisis climatológico, solución de integrales.

- La solución no es única.
  Problemas con la característica de que su solución abarca varias respuestas y la dificultad estriba en seleccionar la más adecuada. Por ejemplo los problemas de ruta crítica, asignación de recursos, topografía de circuitos.

- La información para atacar el problema es incierta.
  Aquí caen los problemas relacionados con hechos que no es posible saber de antemano como la inflación, mercadotecnia, toma de decisiones; los cuales hay que atacar en base a probabilidades.

- Hay expertos reconocidos sobre el tema.

- La habilidad para atacar el problema es difícil para los neófitos.

- Escasean los expertos para un dominio en particular.

- La solución del problema no requiere sólo del sentido común para resolverlo, involucra conocimiento.

- El dominio del problema es bien pagado.

Habiendo escogido un problema experto sobre el cual queremos hallar soluciones ó consejos, viene ahora la fase de imitación del experto humano. Consiste en identificar los objetivos que se plantean los expertos al analizar y resolver los problemas. Hay varios objetivos que componen las características de los humanos expertos.

CARACTERISTICAS DE HUMANOS EXPERTOS

- Explican lo que hacen.
  Pueden decir el porque atacan un problema de cierta manera y porque consideran que una solucion es mejor o más confiable que otra.

- Juzgan la confiabilidad de sus propias conclusiones.
  Establecen grados de certeza sobre los resultados que obtienen.

- Reconocen cuando se hallan confundidos.
  No se quedan en un ciclo infinito si ya no saben como seguir atacando el problema ó cuando necesitan información adicional para resolverlo.

- Se comunican con otros expertos.
  Transfieren y asimilan nuevo conocimiento de sus colegas expertos para ampliar su dominio del problema y volverse más expertos.

- Aprenden de la experiencia.
  Este es el proceso que les hace volverse más expertos y es

el conocimiento que por lo general no puede encontrarse en los libros.
La gente aprende de varias maneras: por prueba y error, por ejemplos, por analogía, por memorización, por razonamiento de lo general a lo particular y viceversa. Además muchas veces resuelven un problema buscando alguna semejanza con otro problema similar.

- Cambian de punto de vista para resolver un problema.
Tienen varias maneras de atacar un problema para resolverlo cuando el camino de solución que escogieron no los está conduciendo a los resultados deseados.

Conjuntando el análisis de características de problemas y humanos expertos, un sistema experto contempla en su diseño los siguientes objetivos:

OBJETIVOS DE LOS SISTEMAS EXPERTOS

- Resolver el problema para el cual fué diseñado.
Un sistema experto está limitado a un dominio específico, por lo que se debe de establecer cuál es el rango de problemas incluidos en este dominio y el sistema deberá proporcionar la solución o soluciones más confiables.

- Explicar el resultado o sus conclusiones.
Nos debe poder mostrar la lógica que siguió para llegar al resultado o resultados finales de una manera comprensible para el usuario.

- Aprender
Adquirir nuevo conocimiento sobre el tema para el que fué diseñado, para lo cual su diseño debe de prever su crecimiento incremental.

- Interpretar el conocimiento.
Lo que hace poderoso a un sistema experto no es unicamente la cantidad de información que maneje, sino la interpretación que le dé a la misma, para lo cual un sistema experto debe de contar con un buen módulo de razocinio, inferencias o conjeturas.

- Representar el conocimiento
Estructurar el conocimiento de la manera más adecuada dependiendo de las características del problema que se pretende solucionar, para que el sistema pueda llegar rapidamente a conclusiones.

En resumen podemos decir que los primeros dos aspectos fundamentales para la elaboración de un sistema experto son la definición de un problema, enmarcando las características del mismo junto con la solución a la que tenemos que llegar, así como el detalle de los objetivos que deseamos cumpla nuestro sistema.

2.2) INGENIERIA DEL CONOCIMIENTO

El conocimiento es la parte medular de todo sistema experto y todo su funcionamiento gira en torno a él. Este conocimiento es de dos tipos. El primero es el de los hechos del dominio del problema y es el que comunmente se encuentra en los libros porque es algo ya establecido. El segundo se denomina conocimiento heurístico, que es el conocimiento de la práctica y buen juicio dentro de un campo. Este conocimiento es el que se obtiene de la experiencia a través de los años y es el que hace a una persona volverse experta.

Una vez que se tiene el problema a resolver junto con los objetivos que satisfacen la solución, es labor del ingeniero del conocimiento la adquisición de estos dos tipos de conocimiento, para posteriormente convertirlos en estructuras de datos que puedan ser procesadas por la computadora a través de la máquina de inferencias y la estrategia de control.

Una de las fases más difíciles de la construcción de un sistema experto es la extracción de conocimiento a un experto humano, ya que no todos los expertos pueden explicar facilmente su manera de resolver un problema.

Aún más, los expertos son por lo general personas muy ocupadas y su tiempo vale oro, por lo que como primer paso, el ingeniero del conocimiento debe persuadirlo de que vale la pena el invertir su valioso tiempo en la construcción del sistema.

Algunos ingenieros han elaborado sistemas expertos para la adquisición automática de conocimiento, como el de EURISKO, desarrollado por Lenat [Lenat, 1983], pero sus resultados todavía no logran el que este proceso sea totalmente satisfactorio, por lo tanto, mientras no se logre una extracción óptima del conocimiento, la mayoría de los ingenieros del conocimiento sigue la siguiente metodología de trabajo [Forsyth, 1984]:

1) Estructurar el dominio del problema encontrando un buen modelo del proceso de solución del problema, simulando el razonamiento del experto.

2) Producir un prototipo del modelo que funcione, tomando en cuenta que puede no trabajar bien al primer intento, pero a sabiendas de que será más sencillo el corregirlo posteriormente.

3) Seguir un ciclo de prueba, depuración y refinamiento del modelo hasta que su respuesta sea satisfactoria y cumpla con los objetivos del problema.

La justificación de esta metodología se fundamenta en las conclusiones de Weiss [Weiss & Kulikowski, 1983] sobre el diseño de sistemas expertos. Menciona que los humanos expertos encuentran más fácil el criticar (constructivamente)

un sistema en producción que decir lo que se necesita, o sea que funciona como un marco de referencia, al cuál se le va agregando nuevo conocimiento sobre el ya implementado. También asevera que el prototipo del sistema mantiene interesado al experto porque ve que ya existe algo en producción y, por lo tanto, no están perdiendo el tiempo.

Ya se ha mencionado que un sistema experto esta basado en una gran cantidad de conocimiento sobre un área en especial; es importante que se tomen en cuenta las consecuencias de trabajar con conocimiento antes de elaborar un sistema que funcione en torno a él. Hay varios principios que los ingenieros del conocimiento utilizan para el diseño de un sistema experto.

1) El conocimiento adquirido puede ser incompleto o inexacto. La parte más difícil de las tareas del ingeniero del conocimiento es la extracción del conocimiento a un humano experto. Una vez que se consigue este objetivo, el ingeniero no puede tener la certeza de que este conocimiento funcionará bien para todos los problemas del dominio en cuestión, ya que él no es experto en ese tema, por lo que es recomendable el realizar pruebas al sistema durante su desarrollo y no esperar hasta la terminación del mismo.

2) La información de hechos puede ser mal suministrada. Esto sucede cuando el usuario se halla confundido en la calidad y cantidad de hechos que debe suministrar al sistema para que su problema sea resuelto. Un mal suministro de esta información tiene como consecuencia una resolución pobre del problema, por lo que el ingeniero del conocimiento debe cerciorarse que las solicitudes de información sistema-usuario no sean confusas.

3) El sistema puede ampliar su conocimiento. Al igual que los expertos humanos que se actualizan continuamente, el sistema experto debe diseñarse para ser flexible, es decir, que permita la incrementación o substitución de su conocimiento por otro más actualizado o novedoso.

4) Considerar el razocinio con datos inciertos. Tener presente que algunos usuarios no dispondrán de cierto tipo de información cuando el sistema la solicite, o simplemente no se sepa. Es una buena medida el considerar que hacer cuando se presente esta situación. El siguiente es un ejemplo de este tipo de situación.

Sistema: Deme el horario disponile del profesor Martínez para impartir la clase de compiladores

Usuario: No lo sé

Sistema: En ese caso puedo tomar tres alternativas:

1) Descartar al profesor Martínez para impartir

la materia de compiladores.
2) Asignar al profesor Martínez horario de tiempo completo.
3) Tomar en cuenta al profesor Martínez con horario de tiempo completo sí ya no hay disponibles otros profesor que impartan la materia de compiladores.

Cuál es su sugerencia?

Usuario: La 3.

El ingeniero del conocimiento posee ahora la información para desarrollar los cuatro módulos básicos del sistema:

1) La base de conocimientos.
2) La máquina de inferencias o conjeturas.
3) La estrategia de control.
4) La interface hombre-máquina.

2.3) REPRESENTACION, MANEJO Y CONTROL DEL CONOCIMIENTO

Las personas se vuelven expertas porque tienen conocimiento sobre un dominio almacenado en algún lugar de su cerebro y saben además la literatura que deben consultar para ayudarse a resolver un problema.
Un sistema experto cuenta con su propio almacén de información: la base de conocimientos.

El ingeniero del conocimiento estructura todo el conocimiento extraído de los libros y del experto dentro de estructuras de datos asequibles por la computadora para formar la base de conocimientos. Existiendo diversas formas de estructurar la información, es su labor el seleccionar la mejor para almacenar el conocimiento de acuerdo a las características del problema que se intenta solucionar.

Hay dos modos muy utilizados para estructurar al conocimiento, la lógica de predicados y la lógica de posibilidades. En ambas se pueden codificar tanto hechos como conocimiento en forma de reglas. Una regla es la consecuencia que ocurre cuando se presentan varios hechos y hay que tomar alguna decisión al respecto, o dicho de otra manera, es la heurística de qué hay que hacer con los hechos cuando se presenten durante el ciclo de solución del problema.
La principal diferencia entre la lógica de predicados y la de posibilidades es que la primera asevera los hechos y reglas como siempre ciertos, mientras que en la de posibilidades tanto hechos como reglas están sujetos a una probabilidad de certeza.

En las dos secciones siguientes se explican ambas lógicas mediante un ejemplo para su mejor comprensión.

2.3.1) LOGICA DE PREDICADOS

Supongamos los siguientes hechos:

1) La computación es una licenciatura
2) Guillermo estudia computación
3) Javier estudia computación
4) Computación se imparte en la UNAM
5) Guillermo es pasante
6) Javier no es pasante

Además de los hechos anteriores tenemos el siguiente conocimiento:

7) Si alguien estudia una licenciatura tiene que asistir a una universidad que imparta esa licenciatura

8) Si alguien estudia una licenciatura y es pasante, esa persona trabaja.

9) La persona que trabaja gana dinero

La lógica de predicados nos permite estructurar la información dada en lenguaje natural en una manera estandarizada, que para los hechos y el conocimiento de nuestro ejemplo es:

1) La computación es una licenciatura
licenciatura(Computación)

Este predicado está formado por dos elementos, el primero es el nombre del predicado (licenciatura), que hace mención al tipo de información que estamos codificando; luego se hace referencia a los argumentos del predicado (Computación). Los argumentos son los encargados de darle algun valor al predicado encadenándolos con uno o varios objetos.

2) Guillermo estudia computación
estudia(Guillermo,Computación)

Aquí tenemos dos argumentos que encadenan a dos objetos diferentes, Guillermo y Computación.
El orden de aparición de los argumentos no es importante siempre y cuando sepamos la relación que existe entre ellos. En este predicado pusimos primero al sujeto que estudia (Guillermo) y después la materia (Computación), pero bien se pudo haber declarado: estudia(Computación,Guillermo), ya que sabríamos que Computación no estudia a Guillermo.

3) Javier estudia computación
estudia(Javier,Computación)

4) Computación se imparte en la UNAM
imparte(Computación,UNAM)

5) Guillermo es pasante
pasante(Guillermo)

6) Javier no es pasante

-pasante(Javier)

Un símbolo usado en la lógica de predicados es el "-", el cuál indica negación de un predicado.

Para la conversión del conocimiento a predicados utilizaremos la siguiente simbología:

¥x = para toda x
Єx = existe x
U = ó (or)
∩ = y (and)
-> = implica qué ó tiene como consecuencia qué

7) Si alguien estudia una licenciatura, tiene que asistir a una universidad que imparte esa licenciatura
¥xЄyЄz estudia(x,y) ∩ imparte(y,z) -> asiste(x,z)

8) Si alguien estudia una licenciatura y es pasante, esa persona trabaja.
¥xЄy estudia(x,y) ∩ pasante(x) -> trabaja(x)

9) La persona que trabaja gana dinero
¥x trabaja(x) -> ganadinero(x)

Nuestro conocimiento (hechos y reglas) queda representado por los nuevo predicados anteriores que se muestran en la tabla 2.2

Pensemos un momento como podríamos explotar este conocimiento para generar nuevo conocimiento. De la información anterior podemos saber directamente si por ejemplo, Javier estudia computación; pero ¿qué sucede si queremos saber si Guillermo trabaja?, entonces ya no es tan directa la solución. El objetivo de encontrar nuevo conocimiento a partir de premisas dadas es trabajo de la máquina de inferencias, que analizamos en otra sección; aquí lo importante es que el conocimiento ha quedado codificado en una forma tal, que podemos generar nuevo conocimiento a partir de las reglas número siete, ocho y nueve o podemos saber la relación que guardan los objetos entre sí.

TABLA 2.2
CONVERSION DE ORACIONES A PREDICADOS

1) La computación es una licenciatura
   licenciatura(Computación)

2) Guillermo estudia computación
   estudia(Guillermo,Computación)

3) Javier estudia computación
   estudia(Javier,Computación)

4) Computación se imparte en la UNAM
   imparte(Computación,UNAM)

5) Guillermo es pasante
   pasante(Guillermo)

6) Javier no es pasante
   -pasante(Javier)

7) Si alguien estudia una licenciatura, tiene que asistir a
   alguna universidad que imparta esa licenciatura
   ∀x∈y∈z estudia(x,y) ∩ imparte(y,z) -> asiste(x,z)

8) Si alguien estudia una licenciatura y es pasante, esa
   persona trabaja
   ∀x∈y estudia(x,y) ∩ pasante(x) -> trabaja(x)

9) La persona que trabaja gana dinero
   ∀x trabaja(x) -> ganadinero(x)

2.3.1.1) La base de conocimientos

Los predicados anteriores necesitan almacenarse de otra manera en la computadora, de tal forma que queden eliminados los símbolos ∀, ∈ y ->. La justificación se basa en que el proceso de inferencias sobre predicados se complica cuando hay que identificar cuantificadores universales (∀x), cuantificadores existenciales (∈x) e implicaciones (->).

Los predicados, entonces, se convierten a claúsulas en forma normal conjuntiva por medio del algoritmo desarrollado por Davis en 1960 [Rich, 1983].

ALGORITMO DE CONVERSION DE PREDICADOS A LA FORMA NORMAL CONJUNTIVA

1) Dado un predicado que contenga implicaciones (->).

   sustituir: expresión.1 -> expresión.2
         por: -expresión.1 U expresión.2
   ejemplo: ∀x∈y∈z estudia(x,y) ∩ imparte(y,z) -> asiste(x,z)
   queda: ∀x∈y∈z -[estudia(x,y) ∩ imparte(y,z)] U asiste(x,z)

2) Aplicar las negaciones a cada componente del predicado, utilizando los teoremas de deMorgan:

2.1) substituir: -[expresión.1 ∩ expresión.2]
por: -expresión.1 ∪ -expresión.2

2.2) substituir: -[expresión.1 ∪ expresión.2]
por: -expresión.1 ∩ -expresión.2

2.3) substituir: -∀x expresión.1
por: ∈x -expresión.1

2.4) substituir: -∈x expresión.1
por: ∀x -expresión.1

ejemplo: ∀x∈y∈z -[estudia(x,y) ∩ imparte(y,z) ∪ asiste(x,z)
queda: ∀x∈y∈z -estudia(x,y) ∪ -imparte(y,z) ∪ asiste(x,z)

3) A cada componente del predicado asignarle variables únicas que no entren en conflicto con otras variables del mismo predicado.

substituir: ∀x expresión.1(x) ∩ ∀x expresión.2(x)
por: ∀x expresión.1(x) ∩ ∀y expresión.2(y)

Esto es permitido ya que la variable cuantificada actúa sólo sobre el cuerpo de la expresión que le sigue hasta que aparezca otra variable cuantificada.

4) Mover todos los cuantificadores al lado izquierdo del predicado sin cambiar su orden. Este movimiento es admisible ya que en el paso tres eliminamos los conflictos con variables idénticas y entonces nuestro predicado presenta la forma:

∀x∈y∈z -estudia(x,y) ∪ -imparte(y,z) ∪ asiste(x,z)

Que se conoce como forma normal prenex, en la cuál se distinguen dos partes: una llamada prefijo formada por todos los cuantificadores y otra llamada matriz, formada por el resto del predicado.

5) Eliminar los cuantificadores existenciales
La presencia de un cuantificador existencial en un predicado refleja una relación de que existe algún ó algunos valores de la variable existencial que provocan que el postulado sea verdadero ó falso. Es necesario entonces el controlar la dependencia que guarda un predicado considerado como regla sobre los predicados considerados como hechos.

ejemplo:
∀x∈y∈z -estudia(x,y) ∪ -imparte(y,z) ∪ asiste(x,z)

Aquí todas las claúsulas del predicado contienen una variable existencial, pero sólo estudia(x,y) e

imparte(y,z) son claúsulas que tienen su contraparte de predicados en forma de hechos, lo cual quiere decir que hay que buscar estos predicados en la base de conocimientos cuando se aplique esta regla para encontrar los valores de "x", "y" y "z" que satisfagan a la regla. La manera de conseguir lo anterior es marcando el nombre de las claúsulas involucradas con un prefijo único que indique que tienen su contraparte en forma de predicados de hechos. Por lo tanto:

substituir:
¥x€y€z -estudia(x,y) U -imparte(y,z) U asiste(x,z)
por: ¥x -bhestudia(x,y) U -bhimparte(y,z) U asiste(x,z)

El prefijo bh indica "busca hecho" y para todas las claúsulas que lo contengan la máquina de inferencias buscará su contraparte.

6) Eliminar los cuantificadores universales.

substituir:
¥x -bhestudia(x,y) U -imparte(y,z) U asiste(x,z)
por: -bhestudia(x,y) U -imparte(y,z) U asiste(x,z)

Esto es posible hacerlo por que queda implícito que una variable universal puede tomar cualquier valor.

7) Poner el prefijo "bh" a las claúsulas con cuantificadores universales que pertenezcan a los hechos.
No todos los hechos están identificados por la presencia de cuantificadores existenciales, sino por los universales. Localizar estas claúsulas y agregarles el prefijo "bh" (busca hecho).

Por ejemplo, nuestro predicado número ocho contiene la claúsula pasante(x), siendo "x" una variable universal pero referenciada la claúsula dentro de los predicados de hechos.

8) Transformar el predicado en un conjunto de disyunciones, utilizando la propiedad distributiva.

substituir: [expresión.1 ∩ expresión.2] U expresión.3
por: [expresión.1 U expresión.3] ∩ [expresión.2 U expresión.3]

Aquí termina el proceso de conversión de predicados a la forma normal conjuntiva. La conversión completa de los predicados de nuestro ejemplo se muestra en la tabla 2.3

TABLA 2.3
CONVERSION DE PREDICADOS A FORMA NORMAL CONJUNTIVA

1) licenciatura(Computación)
   licenciatura(Computación)

2) estudia(Guillermo,Computación)
   estudia(Guillermo,Computacin)

3) estudia(Javier,Computación)
   estudia(Javier,Computación)

4) imparte(Computación,UNAM)
   imparte(Computación)

5) pasante(Guillermo)
   pasante(Guillermo)

6) -pasante(Javier)
   -pasante(Javier)

7) ∀x∈y∈z estudia(x,y) ∩ imparte(y,z) -> asiste(x,z)
   -bhestudia(x,y) U -bhimparte(y,z) U asiste(x,z)

8) ∀x∈y estudia(x,y) ∩ pasante(x) -> trabaja(x)
   -bhestudia(x,y) U -bhpasante(x) U trabaja(x)

9) ∀x trabaja(x) -> ganardinero(x)
   -trabaja(x) U ganardinero(x)

Estos predicados en forma normal conjuntiva son los que forman ya nuestra base de conocimientos. En la siguiente sección se discute el método de inferencia "Resolución", con el cual podemos trabajar cualquier base de conocimientos estructurada en forma de predicados en forma normal conjuntiva.

2.3.1.2) La máquina de inferencias

La estrategia de inferencias que utilizaremos para el manejo de conocimiento estructurado como predicados en forma normal conjuntiva se le conoce con el nombre de Resolución [Rich, 1983].

Supongamos que uno de los objetivos de nuestro pequeño sistema experto es demostrar si Guillermo gana dinero, esto es: ganardinero(Guillermo) es falso o verdadero.

La resolución intenta demostrar un objetivo por contradicción, esto es que para nuestro objetivo tratará de demostrar: -ganardinero(Guillermo). El algoritmo de resolución es el siguiente:

ALGORITMO DE INFERENCIAS: RESOLUCION

LLamemos G al objetivo que se intenta demostrar.
1) Negar G y convertir el predicado a forma normal conjuntiva.

2) Repetir hasta que se satisfaga G y el remanente quede vacío ó G no pueda ser demostrada.
   2.1) SI G contiene más de una cláusula, escoger una de ellas para formar un nuevo objetivo G y al resto llámese remanente.
   2.2) Buscar un predicado en la base de conocimiento que contenga la negación de G y llámese P a ese predicado. P puede ser una regla o hecho, a menos que G contenga el prefijo "bh", en cuyo caso P debe ser un hecho.
   Si no se halla P, el objetivo es falso o no puede ser demostrado.
   2.3) Unificar los argumentos de G y P. La unificación es como sigue:
      2.3.1) Tomar el argumento ónsimo de G y P. LLámense G(n) y P(n).
      2.3.2) Si G(n) y P(n) son constantes, se unifican si son iguales. Si son diferentes regresar al paso 2.2.
      2.3.3) Si G(n) es una constante y P(n) una variable, substituir P(n) por G(n) en todo P y en el remanente, si éste es diferente de vacío.
      2.3.4) Si G(n) es una variable y P(n) es una constante, substituir G(n) por P(n) en todo P y en el remanente si éste es diferente de vacío.
      2.3.5) Si G(n) y P(n) son variables, se unifican automaticamente.
      2.3.6) Si todavía quedan argumentos por unificar, regresar al paso 2.3.
   2.4) Una vez unificados todos los argumentos, realizar la operación G = P - G + Remanente, es decir, eliminar G del predicado y sumar los objetivos restantes incluidos en el remanente.
   2.5) Si G queda vacío, entonces se satisface el objetivo como verdadero, si no es vacío, regresar al paso 2.1.

Mostraremos a continuación el conjunto de inferencias que se efectúan para G = ganardinero(Guillermo).

1) Negar G y convertir el resultado a forma normal conjuntiva.
   G = -ganardinero(Guillermo)
   Remanente = vacio.

El siguiente paso es encontrar una cláusula dentro de los predicados de la base de conocimiento que contenga: ganardinero(x), donde x es una variable.

Predicado No. 9: -trabaja(x) U ganardinero(x)

Los argumentos "Guillermo" y "x" se unifican realizando la substitución P= -trabaja(Guillermo) U ganardinero(Guillermo).

Unificados los argumentos G = P - G + Remanente:
G = -trabaja(Guillermo)

Remanente = vacío.

Buscamos la claúsula: trabaja(Guillermo) en la base de conocimientos:

Predicado No. 8: -bhestudia(x,y) U -bhpasante(x) U trabaja(x)

Unificamos los argumentos subetituyendo "x" por "Guillermo":
P = -bhestudia(Guillermo,y) U -bhpasante(Guillermo) U trabaja(Guillermo)

G = -bhestudia(Guillermo)
Remanente= -bhpasante(Guillermo)

Buscamos el hecho estudia(Guillermo) ya que G contiene al prefijo "bh":

Predicado No. 2: estudia(Guillermo).

Los argumentos son constantes e iguales, por lo tanto se unifican.

G = -bhpasante(Guillermo).
Remanente = vacío.

Predicado No. 5: pasante(Guillermo).

Los argumentos son constantes e iguales, por lo tanto se unifican.

G = vacío.
Remanente = Vacío.

Como G y Remanente son vacíos, el objetivo se satisface: ganardinero(Guillermo) es verdadero.

Está comprobado [Rich, 1983], que el algoritmo de resolución demostrará un objetivo como verdadero, si la base de conocimientos contiene la información necesaria para hacerlo. Sin embargo, en bases de conocimiento muy grandes, el proceso de demostración puede tomar mucho tiempo, o si el objetivo a demostrar es falso y se cuenta con la información para demostrarlo hay que realizar dos pasadas al algoritmo. La estrategia de control para controlar el algoritmo de resolución de una manera óptima se presenta a continuación.

2.3.1.3) La estrategia de control

Este módulo tiene como propósito el minimizar el tiempo de búsqueda de predicados cuyos argumentos puedan ser unificados, demostrar un objetivo como falso si se cuenta con la información para hacerlo y "memorizar" la secuencia de rastreo de predicados para explicaciones posteriores de como se demostró el objetivo.

En la estrategia de control se cumplen los siguientes

objetivos:

1) Indexar todos los nombres de claúsulas apuntando a todos los predicados que las contengan en la base de conocimiento, junto con una indicación de si está negada o no. Esto facilita el rastreo de predicados.

2) Ir eliminando las claúsulas que se vayan unificando de la base de conocimiento para que ya no puedan participar en unificaciones o búsquedas posteriores.

3) Para reconocer a un objetivo como falso, se toma el objetivo proporcionado por el usuario y no se le aplica la negación. Entonces si el algoritmo logra la unificación de tal manera que al final resulte G y Remanente = vacío, el objetivo es falso, en caso contrario, como inicialmente no fué verdadero, el sistema no puede sacar ninguna conclusión sobre el objetivo.

2.3.1.4) La interface hombre-máquina

Con la interface hombre-máquina estamos interactuando con el sistema ya sea para solicitar consejos o para introducir información.

La manera más sencilla de solicitar consejo sobre un objetivo en particular, es el darle al sistema el objetivo como predicado en forma normal conjuntiva, ya que es con lo que la máquina de inferencias trabaja. El problema de usar este método radica en que es muy engorroso para el usuario el convertir cada objetivo que desea a la forma normal conjuntiva, y si el objetivo es muy complejo, puede cometer equivocaciones a la hora de hacer la conversión.

La primera característica de nuestra interface es proveerla con un mecanismo que pueda aceptar predicados en su forma normal conjuntiva o en forma natural. Si el predicado se introduce en forma natural, habrá que implementar el algoritmo de conversión a forma normal conjuntiva dentro de nuestro sistema.
La segunda característica consiste en simplificar el lenguaje de los predicados lo más que podamos hacia el lenguaje humano. Esto es importante, porque de otra manera el usuario está sujeto a emplear los simbolos de la lógica de predicados y los nombres de las claúsulas tal como se encuentran en la base de conocimientos.

Para ilustrar esto supongamos que el objetivo de nuestro usuario es el conocer todos los estudiantes que trabajan y estudian Computación. En forma de predicado natural esto es:

$\forall x$ trabaja(x) $\wedge$ estudia(x,Computacion)

El usuario podría introducir en vez de este predicado, algo así como: trabajan y estudian Computación.

Es responsabilidad entonces de la interface hombre-máquina

el convertir esta oración a predicado natural. La conversión puede llevarse a cabo manteniendo una base de datos con todas las posibles palabras que hagan referencia a las claúsulas de la base de conocimientos. Así pues, podemos tener:

Para trabaja(x) -> trabajador, trabajan, trabajo, empleo, empleado, etc...

Para estudia(x,y) -> estudiantes, estudiar, estudian, etc...

La idea es el relacionar cada palabra de la oración proporcionada por el usuario con una claúsula de nuestra base de conocimientos para poder hacer la conversión a predicado en forma normal conjuntiva.
Lo ideal de todo sistema experto es el contar con un intérprete perfecto del lenguaje humano, pero mientras esto no sea una realidad, lo más que podemos hacer por el usuario es facilitarle las cosas hasta donde nos sea posible.

2.3.2) Lógica de posibilidades.

La lógica de posibilidades (fuzzy logic) fué inventada por Lotfi Zadeh en 1965. Nos permite razonar bajo incertidumbre, donde los hechos y las reglas de inferencia ó ambas no son 100% confiables.

Supongamos que que nos topamos con el problema de dar un diagnóstico sobre la falla de un automóvil, para lo cuál contamos con el siguiente conocimiento:

1) El 30% de los vehículos que llegan a nuestro taller necesitan afinación.

2) El 40% de los vehículos que llegan a nuestro taller necesitan ajustar la bomba de gasolina.

3) Un carburador desreglado indica necesidad de afinar con probabilidad de 50% y de no afinar con probabilidad de 20%.

4) Una entrada excesiva de aire al carburador indica necesidad de afinar en un 90% y de no afinar en un 15%.

5) Un consumo excesivo de gasolina indica necesidad de ajustar la bomba de gasolina en un 70% y de no ajustar en un 25%.

6) Si el motor se para al desacelerar hay que ajustar la bomba de gasolina en un 70% de los casos y no hay que ajustarla en el 30%.

Este conocimiento lo dividimos en diagnóstico y síntomas con sus respectivas probabilidades de acuerdo a la tabla 2.2., donde:

P[DIAG] = probabilidad de que se presente el síntoma dado que el diagnóstico asociado sea acertado.

P[-DIAG] = probabilidad de que se presente el síntoma dado que el diagnóstico asociado no sea acertado.

TABLA 2.2

DIAGNOSTICOS Y SINTOMAS PARA UN AUTOMOVIL

DIAGNOSTICOS

DIAGNOSTICO	CASOS ANTERIORES DETECTADOS	NUMERO DE SINTOMAS ASOCIADOS AL DIAGNOSTICO
Afinación	30%	2
Ajuste bomba de gasolina	40%	2

SINTOMAS

NUMERO DE SINTOMA	DESCRIPCION	P[DIAG]	P[-DIAG]	DIAGNOSTICO ASOCIADO
1	carburador desreglado	50%	20%	Afinación
1	carburador desreglado	75%	20%	Ajuste bomba de gasolina
2	exceso de aire en carburador	90%	15%	Afinación
3	consumo excesivo de gasolina	70%	25%	Ajuste bomba de gasolina
4	paro de motor al desacelerar	70%	30%	Ajuste bomba de gasolina

2.3.2.1) La base de conocimientos

La estructuración de la base de conocimientos se realiza sobre los siguientes dos formatos:

1) Formato de Diagnóstico:

D,P[D],N,S(1),P[S(1)/D],P[S(1)/-D],....,S(N),P[S(N)/D], P[S(N)/-D]

D = nombre del diagnóstico.
P[D] = probabilidad de que el diagnóstico D se presente en este automóvil, dado que se ha presentado en otros vehículos.

N = número de síntomas aplicables que hacen que el diagnóstico D sea acertado.
S(N) = síntoma número N.
P[S(N)/D] = probabilidad de que el síntoma número N sea observado dado que el diagnóstico D sea acertado.
P[S(N)/-D] = probabilidad de que el síntoma número N sea observado dado que el diagnóstico D no sea acertado.

2) Formato de Síntomas:

NSINT,NOMBRE,PREGUNTA

NSINT = número de síntoma.
NOMBRE = nombre del síntoma.
PREGUNTA = texto interrogando sobre la observación del síntoma.

Ahora estructuremos el conocimiento de nuestro ejemplo dentro de estos dos formatos utilizando la tabla 2.2.

1) Formato de diagnóstico:
1.1) AFINACION,0.3,2,1,0.5,0.2,2,0.9,0.15
El nombre del diagnóstico es AFINACION, tiene 30% de posibilidad de presentarse en cualquier vehículo y son dos los síntomas asociados al diagnóstico. El primer síntoma es el número 1 y presenta 50% de probabilidad de ser observado si el automóvil necesita afinación y 20% de ser observado si no se necesita afinar. El segundo síntoma es el número 2 y presenta 90% de probabilidad de ser observado para que el automóvil necesite revisión de bomba de gasolina y 15% si no necesita la revisión.

1.2) REVISAR-BOMBA-GAS,0.4,3,2,0.75,0.2,4,0.7,0.25,5,0.7,0.3

2) Formato de síntomas:
2.1) 1,CAR-DES,¿Está desreglado el carburador?
Es el síntoma número uno llamado CAR-DES,(carburador desreglado). Su pregunta asociada es si el carburador está desreglado.

2.2) 2,EX-AIRE-CAR,¿Hay entrada de aire en exceso al carburador?
2.3) 3,CON-EXC-GAS,¿Consume gasolina en exceso?
2.4) 4,PAR-MOTOR-DES,¿Se detiene el motor al desacelerar?

La base de conocimiento así formada puede contener tantos diagnósticos como se deseen, con su correspondiente relación de síntomas y posibilidades de ocurrir.

2.3.2.2) La máquina de inferencias

Las inferencias son efectuadas utilizando los teoremas de Bayes sobre probabilidades.

TEOREMAS DE PROBABILIDAD

$$1)\ P(D/S) = \frac{P(D \cap S)}{P(S)}$$

$$2)\ P(S/D) = \frac{P(S \cap D)}{P(D)}$$

P(D/S) = probabilidad de que el diagnóstico D sea acertado dado que se observa el síntoma S.
P(D ∩ S) = probabilidad de que el diagnóstico D sea acertado y se observe el síntoma S.
P(S) = probabilidad de que se observe el síntoma S.
P(S/D) = probabilidad de que se observe el síntoma S, dado que el diagnóstico D es acertado.
P(S ∩ D) = P(D ∩ S)
P(D) = probabilidad de que el diagnóstico D sea acertado.

Ahora combinemos las fórmulas 1 y 2 para obtener:

$$3)\ P(D/S) = \frac{P(S/D)*P(D)}{P(S)}$$

Pero 4) P(S) = P(S/D)*P(D)+P(S/-D)*P(-D)

donde P(S/-D) = probabilidad de que se observe el síntoma S, dado que el diagnóstico D no sea acertado.
P(-D) = 1-P(D) = probabilidad de que el diagnóstico no sea acertado.

Por lo tanto, substituyendo esta relación en 2:

$$5)\ P(D/S) = \frac{P(S/D) * P(D)}{P(S/D)*P(D)+P(S/-D)*[1-P(D)]}$$

Haciendo la relación de esta fórmula con la información que tenemos en la base de conocimientos, podemos usarla como sigue:

La ecuación debe aplicarse sobre todos los síntomas observables del automóvil recursivamente para todas las hipótesis de diagnóstico que incluyan a los síntomas.

Por ejemplo:

Información observable:
1) El carburador está desreglado.
2) La entrada de aire al carburador es excesiva.
3) El motor se detiene al desacelerar.

Hipótesis # 1: Afinar vehículo.
Primer síntoma aplicable a la hipótesis: carburador

desreglado.

P(D/S) = probabilidad de que el automóvil necesite afinación dado que se observa el carburador desreglado.
P(S/D) = probabilidad de que el carburador esté desreglado dado que se necesite afinar = 0.5

P(D) = probabilidad de que se requiera afinación = 0.3
P(S/-D) = probabilidad de que el carburador esté desreglado dado que no se necesita la afinación = 0.2.

Entonces:

$$P(D/S) = \frac{0.5 * 0.3}{0.5 * 0.3 + 0.2 * (1 - 0.3)} = 0.5172 = 51.72\%$$

Para el segundo síntoma (entrada excesiva de aire al carburador), realizamos la asignación P(D) = P(D/S) = 0.5172, la cual obtuvimos como resultado del primer síntoma.

P(D/S) = probabilidad de que se necesite afinación dado que la entrada de aire al carburador es excesiva.
P(S/D) = 0.9
P(D) = 0.5172
P(S/-D) = 0.15

$$P(D/S) = \frac{0.9 * 0.5172}{0.9 * 0.5172 + 0.15 * (1 - 0.5172)} = 0.8654 = 86.54\%$$

El tercer síntoma no es aplicable a nuestra hipótesis, por lo que nuestro primer diagnóstico es que se necesita la afinación con un 86.54% de certeza.

Como se puede apreciar, la probabilidad de que el automóvil necesite de afinación se incrementó al aplicar la ecuación recursivamente con el segundo síntoma. Este incremento se debe a la substitución de la probabilidad del diagnóstico [P(D)] original (0.3) por el obtenido con el primer síntoma (0.5172), y así conforme el número de síntomas observables aumenta, la probabilidad de que el diagnóstico sea certero aumenta ó disminuye.

Hipótesis # 2: ajustar bomba de gasolina.
Primer síntoma aplicable a la hipótesis: carburador desreglado.

P(D/S) = probabilidad de que se necesite ajuste de bomba dado que el carburador está desreglado.
P(S/D) = 0.75
P(D) = 0.4
P(S/-D) = 0.2

$$P(D/S) = \frac{0.75 * 0.4}{0.75 * 0.4 + 0.2 * (1 - 0.4)} = 0.7143 = 71.43\%$$

Segundo síntoma aplicable: consumo excesivo de gasolina.

P(D/S) = probabilidad de que se necesite ajuste de bomba dado que el vehículo consume gasolina en exceso.
P(S/D) = 0.7
P(D) = 0.7143
P(S/-D) = 0.25

$$P(D/S) = \frac{0.7 * 0.7143}{0.7 * 0.7143 + 0.25 * (1 - 0.7143)} = 0.875 = 87.50\%$$

Como ya no hay más hipótesis de diagnóstico por resolver, las conclusiones del sistema son:

1) Diagnóstico más certero: Ajustar bomba de gasolina, certeza = 87.50%.
2) Alternativa: Afinación, certeza = 86.54%.

2.3.2.3) La estrategia de control.

Lo siguiente que necesita nuestro sistema experto es un mecanismo de control para determinar cuándo una hipótesis de diagnóstico puede considerarse certera o no certera sin necesidad de revisar todos los síntomas observables pendientes.

Esto es útil cuando tenemos síntomas que hacen que sea necesario dar un diagnóstico sin importar el que otros síntomas puedan o no ser observados.
Iniciemos nuestra estrategia de control haciendo las siguientes consideraciones:

1) Establecer un valor de aceptación de la hipótesis [P(D/S)]. Este puede ser digamos VA = 0.9 (90%).

2) Establecer un valor de rechazo de la hipótesis. Podemos emplear VR = 0.1 (10%).

3) Definir P(máxima) como la máxima probabilidad que puede tomar nuestra hipótesis si todos los demás síntomas aplicables dentro de nuestra base de conocimiento fueran observables.

4) Definir P(mínima) como la mínima probabilidad que puede tomar nuestra hipótesis si todos los demás síntomas aplicables dentro de nuestra base de conocimientos no son observables.

La estrategia de control trabaja de la siguiente manera:

1) Tomar un síntoma observable y calcular P(D/S) para cada diagnóstico donde aparezca el síntoma.

2) Calcular P(máxima) y P(mínima) con los síntomas restantes en la base de conocimiento para cada diagnóstico del paso 1.

3) Encontrar el mayor P(mínima) del conjunto generado en el paso 2, llamémoslo Mmin.

4) Sí alguno de los P(máxima) es mayor que Mmin, significa que con más evidencia podemos dar un mejor diagnóstico, por lo que hay que regresar al paso 1.

5) No habiendo P(máxima) mayor a Mmin, entonces podemos concluir las inferencias y decir que el diagnóstico más acertado es el que haya generado este Mmin, ya que el valor de las demás hipótesis con toda la evidencia a su favor, no sobrepasan a este Mmin, que tiene toda la evidencia en contra.

6) Además de sugerir éste diagnóstico (Mmin) como el más recomendable, también podemos sugerir otros diagnósticos de acuerdo a las siguientes categorías:

   6.1) Hipótesis que hayan obtenido un P(mínima) mayor a VA (valor de aceptación).
   6.2) Hipótesis con P(mínima) menor a VA y P(máxima) mayor a VR (valor de rechazo).

7) Cuando P(máxima) es menor a VR, entonces no podemos inferir nada seguro sobre la validez de la hipótesis.

2.3.2.4) La interface hombre-máquina.

Estamos ahora en la parte de introducir información al sistema para consultarlo sobre las fallas que observamos en nuestro vehículo. Esta consulta podemos llevarla a cabo de dos maneras. La primera es el encadenamiento hacia atrás (backward chaining), en la cuál el usuario va introduciendo síntomas observables de su automovil, y el sistema sacará todas las inferencias posibles sobre los diagnósticos que prevean ese síntoma hasta que haya algun valor de probabilidad certero para un diagnóstico cualquiera.

La otra manera de interactuar con el sistema es utilizando el encadenamiento hacia adelante (forward chaining), en la cuál es el sistema experto el que hace las preguntas partiendo de la premisa que conoce cuáles síntomas son los más frecuentes que se presentan. El encadenamiento hacia adelante es útil cuando el usuario que hace la consulta no sabe por donde comenzar para solucionar la falla de su vehículo, es decir, no conoce los síntomas de su automóvil o conoce tantos qué no sabe cuál es más relevante que otro.

El siguiente algoritmo es una mezcla de encadenamiento hacia adelate y hacia atrás. El algoritmo se encarga de escojer la mejor pregunta relacionada con un síntoma al usuario:

1) Extraer de la base de conocimientos la probabilidad apriori de cada diagnóstico P(D). Guárdense estas probabilidades en un arreglo PD(i).

2) Para cada síntoma de la base de conocimiento, óbtenganse su valor preferencial (VP).
   VP(j) = Σ P[S(j)]
   La sumatoria es para i = 1 hasta número de diagnósticos.
   P[S(j)] = probabilidad de que el síntoma j se observe en la hipótesis i.
   P[S(j)] = P[S(j)/H] * P[H] + P[S(j)/-H] * P[-H] (fórmula 4)

3) De todos los VP(j) obtenidos, escoger aquel que tenga el mayor valor e interrogar al usuario sobre éste síntoma, que es el más probable de presentarse. La respuesta del usuario debe estar en el rango [0,1], indicando un cero que el síntoma es falso, un uno verdadero y 0.5 incertidumbre.

4) Calcular P[H(i)/R] utilizando como P[S/R] el valor proporcionado por el usuario, para todas las hipótesis donde se aplique el síntoma S:
   P[H(i)/R] = P[H(i)/S] * P[S/R] + (1 - P[H(i)/S]) * (1 - P[S/R])

5) Seguir la estrategia de control de la sección anterior a partir del paso número dos para checar si ya se puede inferir el diagnóstico correcto.
   En caso contrario proseguir con 6.

6) Recalcular los valores preferenciales substituyendo la probabilidad P[H(i)] por P[H(i)/R] calculada en el paso 4, y regresar al paso 3.

Apliquemos este algoritmo a nuestro sistema experto y veamos los resultados:

1) Probabilidad apriori de diagnósticos:
   P[AFINACION] = 0.3
   P[REVISAR-BOMA-GAS] = 0.4

2) Valor preferencial de síntomas:
   VP(CAR-DES) = [0.5 * 0.3 + 0.2 * (1 - 0.3)] + [0.75 * 0.4 + 0.2 * (1 - 0.4)] = 0.71
   VP(EX-AIRE-CAR) = 0.9 * 0.3 + 0.15 * (1 - 0.3) = 0.375
   VP(CON-EXC-GAS) = 0.7 * 0.4 + 0.25 * (1 - 0.4) = 0.43
   VP(PAR-MOT-DES) = 0.7 * 0.4 + 0.3 * (1 - 0.4) = 0.46

3) El valor preferencial mayor corresponde al síntoma CAR-DES, que es el síntoma número uno, por lo que tomamos la pregunta almacenada en el formato de este síntoma para desplegársela al usuario: Está desreglado el carburador?
   Supongamos que nuestro usuario contesta que sí en un 90%.
   P[CAR-DES] = 0.9

4) Calcular P[H(i)/R] para cada hipótesis:

   4.1) P[AFINAR/CAR-DES] = 0.5172 * 0.9 + 0.4828 * 0.1
                          = 0.5138

4.2) P[REVISAR-BOMBA-GAS/CAR-DES] = 0.7143 * 0.9 + 0.2857 * 0.1 = 0.6714

5) Aplicando la estrategia de control:
Para AFINAR:
P(máxima) = 0.8638 * 1 + 0.1362 * 0 = 0.8638
Para REVISAR-BOMBA-GAS:
P(máxima) = 0.9303
Las P(mínima) son las probabilidades obtenidas en el paso 4:
Para AFINAR:
P(mínima) = 0.5138
Para REVISAR-BOMBA-GAS:
P(mínima) = 0.6714

Como alguna P(máxima) es mayor que la mayor de las P(mínima), volvemos a calcular los valores preferenciales sobre los síntomas que no se le han preguntado al usuario:

VP(EX-AIRE-CAR) = 0.9 * 0.5138 + 0.15 * (1 - 0.5138) = = 0.5354
VP(CON-EXC-GAS) = 0.7 * 0.6714 + 0.25 * (1 - 0.6714) = = 0.5521
VP(PAR-MOT-DES) = 0.7 * 0.6714 + 0.3 * (1 - 0.6714) = = 0.5686
Así que la siguiente pregunta es la relacionada con el paro del motor al desacelerar.

Este proceso continúa hasta que obtengamos una probabilidad de diagnóstico de acuerdo a nuestros criterios mínimos de aceptación.

CAPITULO 3

CONSTRUCCION DE UN SISTEMA EXPERTO

Varios autores han formulado respuestas a la pregunta: ¿qué es un sistema experto?. Esta rama de la inteligencia artificial ha cobrado gran interés, el cual se puede medir por el nivel de publicaciones especializadas. Algunas de las respuestas son:

1) "Un sistema experto es un programa que se comporta como un humano experto de tal manera que produzca acciones útiles." [Prendergast, 1985].

2) "Un sistema experto es un intento de identificar, formalizar, codificar y usar el conocimiento de humanos expertos como la base para programas más eficientes." [Winston, 1985].

3) "Un sistema experto no es más que una colección de hechos y reglas con las cuales podemos encontrar soluciones 'educadas' sobre diversos problemas de diversas ciencias." [Jenkins, 1986].

4) Una definición mas formal, tendiendo a la estandarización de lo que es un sistema experto, fué dada por el grupo especialista de la sociedad de computación británico (British Computer Society's Specialist Group):

"Un sistema experto es la incorporación a una computadora de una base de conocimientos de una habilidad experta, en una forma tal que el sistema pueda ofrecer consejos inteligentes o tomar alguna decisión inteligente sobre el procesamiento de una función. Una característica adicional deseable, que muchos considerarían fundamental, es la capacidad del sistema, al requerírsele, de justificar su propia línea de razonamiento, de una manera directamente inteligible para el usuario. El estilo adoptado para cumplir con estos requisitos es la programación basada en reglas".

En nuestro intento por construir un sistema experto seleccionamos primero un problema a resolver de acuerdo a las características de los problemas expertos enunciadas en el capítulo dos. Posteriormente solicitamos la ayuda de un experto humano que tuviera el conocimiento y la experiencia necesarias para solucionar el problema planteado, comenzamos entonces la etapa de adquisición de conocimiento para formar la base de conocimiento de nuestro sistema y la estrategia de control. Con estos elemento a la mano diseñamos los cuatro módulos básicos del sistema (base de conocimientos, máquina de inferencias, estrategia de control e inteface hombre-máquina), para posteriormente desarrollarlos en los

lenguajes de programación turbo pascal y turbo prolog.

3.1) LA PLANEACION

3.1.1) DEFINICION DEL PROBLEMA Y OBJETIVOS

La concepción del sistema experto comienza al plantear un problema y su solucion o soluciones. El problema que hemos seleccionado es el de la asignación o unificación de recursos.

DEFINICION DEL PROBLEMA: Dado un conjunto de recursos libres y una serie de relaciones entre ellos, aplicar reglas de asignación de recursos para lograr la unificación entre un recurso y otro.

Un recurso libre es un objeto, ya sea físico ó conceptual, con ciertos atributos que le dan algún valor. Se le llama recurso libre a aquel que no ha sido asignado a otro recurso y por lo tanto podemos disponer de él. Esta asignación se lleva a cabo tomando en cuenta las relaciones que existen entre los recursos libres y seleccionando la manera en que queremos unificarlos.

Tomemos por ejemplo los recursos tiempo, programadores y computadoras, que por si solos son recursos libres. Para unificar estos recursos queremos asignar un tiempo de uso de computadora a los programadores y este objetivo marca la relación existente entre estos recursos. La unificación se lleva a cabo al decidir cuanto tiempo se asigna a cual trabajador y en que computadora.

Este problema posee dos características enunciadas en el capítulo dos sobre problemas expertos: su solución no es única y además puede obtenerse de varias maneras.
El objetivo de nuestro sistema será lograr la unificación de recursos de acuerdo a la siguiente hipótesis.

HIPOTESIS DE UNIFICACION: Un objeto O1 ocupa un espacio físico E durante un tiempo T si y solo si no existe algún otro objeto O2 ocupando el mismo espacio físico E durante el mismo tiempo T ó una fracción de T.

Esta hipótesis es nuestro punto de partida para la selección de recursos, relaciones entre ellos y reglas de asignación o unificacion de los mismos.

3.1.2) ADQUISICION DE CONOCIMIENTO

El problema experto ya fué planteado así como el objetivo que se desea conseguir a través del sistema. Falta ahora identificar concretamente los elementos que forman la definición del problema. Nos ocuparemos de un problema especial de asignación de recursos donde definimos los recursos libres, sus relaciones y reglas de asignación.

Supongamos que somos miembros de alguna asociación

profesional de ingenieros cuyos dirigentes están planeando en organizar un congreso sobre diversos temas relacionados con su area de trabajo. El comité organizador del congreso se está integrando y somos invitados a formar parte de él para desarrollar alguna actividad. Se nos informa que el congreso contempla la invitación de socios dedicados a exponer conferencias, así como la participación del público en general que asista a la conferencia. Se planea llevar a cabo una campaña publicitaria para conocer las inquietudes de las personas interesadas en asistir al congreso y así determinar el número de conferencias que se impartirán, la duración de las mismas, los temas y el número de asistentes esperados al evento.

El presidente del comité organizador nos asigna la tarea de escoger un lugar donde se lleve a cabo el congreso, así como efectuar la calendarización de las conferencias para un número máximo de días que puede durar el congreso. Solicitamos el consejo de una persona que anteriormente ha resuelto este problema con la organización de una convención el año anterior.

El objetivo es estudiar a esta persona y conseguir que su conocimiento y experiencia nos sirvan de modelo para construir un sistema experto que realice la calendarización como la haría esta persona. Para iniciar el experto nos aconseja una serie de reglas a seguir.

REGLAS PARA RECABAR HECHOS

1) Seleccionar por lo menos dos centros de convenciones donde pueda efectuarse el congreso.

2) Recabar la siguiente información concerniente a los salones de cada centro de convenciones:

   2.1) Nombre del salón.
   2.2) Capacidad para diferentes tipos de amueblado.
   2.3) Horarios disponibles para los días en que planea efectuarse la convención.
   2.4) Salones contiguos con posibilidad de hacer uno solo con mayor capacidad.

3) Recabar la siguiente información sobre las conferencias que se planean impartir:

   3.1) Nombre de la conferencia.
   3.2) Tema al cual pertenece la conferencia.
   3.3) Asistencia esperada a la conferencia.
   3.4) Duración de la conferencia.
   3.5) Mobiliario deseado para la conferencia.

De lo aquí expuesto podemos dar ahora una definición concreta del problema que resolverá nuestro sistema experto.

DEFINICION DEL PROBLEMA: Dado un conjunto de recursos libres (conferencias, salones y tiempo), y una serie de relaciones

entre ellos (asistencia a la conferencia, capacidad de salones. duración de la conferencia, disponibilidad de salones), aplicar las reglas de asignación aprendidas para lograr la calendarización del congreso.

HIPOTESIS DE UNIFICACION: Una conferencia C1 ocupa un salón S durante un tiempo T si y sólo si no existe alguna otra conferencia C2 ocupando el mismo salón S durante el mismo tiempo T ó una fracción de T.

Ahora bien, la estrategia de control coordina un método o reglas de asignación, que el experto nos explica a través de un ejemplo.

ESTRATEGIA DE CONTROL

1) Ordenar las conferencias por temas y asistencia a las mismas.

2) Ordenar los salones por disponibilidad y capacidad.

3) Evitar en lo posible que las conferencias sobre un mismo tema se traslapen dentro del mismo horario el mismo día. Esto se realiza para que las personas que asistan al evento puedan asistir al mayor número posible de exposiciones sobre el tema de su mayor interés.

4) Una vez que a un salón se le haya asignado un tipo de mobliario, este salón no podrá cambiar de mobiliario durante ese día, porque el estar cambiando de mobiliario a un salón requiere de personal que lo haga y de gasto excesivo de tiempo mientras se amuebla. La experiencia aconseja amueblar todos los salones antes de comenzar las conferencias del día y mantenerlos así durante el resto del día.

5) Asignar primero las conferencias que sobrepasen en asistencia la capacidad de todos los salones de manera individual, tomando en cuenta a los salones que puedan unirse. De acuerdo a su experiencia nos explica que cuando una conferencia requiere el juntar dos salones contiguos para satisfacer la demanda de asistencia, es más probable encontrar estos salones que dispongan del mismo mobiliario y horario desde el principio de la asignación, que cuando ya se le han asignado otras conferencias.

6) Considerar un rango de aceptación para la capacidad de los salones. Por ejemplo, si a una conferencia van a asistir 65 personas y existe un salón con capacidad para 70 asistentes, la conferencia se puede asignar a este salón utilizando un rango de aceptación de colocar 5 personas más en ese salón. Esto no perjudica la asignación debido a que el número de asistentes esperados es sólo una estimación.

Los recursos libres de conferencias y salones que vamos a unificar son los siguientes:

RECURSO CONFERENCIAS

NOMBRE	TEMA	ASISTENCIA ESPERADA	DURACION DESEADA	MOBILIARIO DESEADO
Sensores	I.A.	65	1 hr.	Auditorio
Visión	I.A.	140	2 hrs.	Auditorio
Prolog	Lenguaje	75	1 hr. 30'	Auditorio
Aprendizaje	I.A.	45	2 hrs.	Escuela

RECURSO SALONES

NOMBRE	MOBILIARIO	CAPACIDAD	DISPONIBILIDAD		
			DIA	HORA INICIAL	HORA FINAL
Salon-1	Auditorio	100	1	11 am.	14 pm.
Salon-1	Escuela	50	1	11 am.	14 pm.
Salon-2	Auditorio	70	1	9 am.	14 pm.
Salon-2	Escuela	35	1	9 am.	14 pm.
Salon-3	Auditorio	90	1	9 am.	14 pm.
Salon-3	Escuela	45	1	9 am.	14 pm.

Además el centro de convenciones permite juntar los salones uno y dos para formar uno más grandes llamado salon-12.

Establecidos los recursos libres, se decide efectuar la calendarización para una duración del congreso de un día y el rango de aceptación para la capacidad de los salones (estrategia de control número 6), se establece en 10%.

Las reglas de asignación de nuestro experto combinadas con la estrategia de control son las siguientes:

REGLAS DE ASIGNACION

1) Aplicar la estrategia de control número 1: ordenar las conferencias por temas y asistencia en forma descendente.

NOMBRE	TEMA	ASISTENCIA ESPERADA	DURACION DESEADA	MOBILIARIO DESEADO
Aprendizaje	I.A.	45	2 hrs.	Escuela
Sensores	I.A.	65	1 hr.	Auditorio
Visión	I.A.	140	2 hrs.	Auditorio
Prolog	Lenguajes	75	1 hr. 30'	Auditorio

2) Aplicar la estrategia de control número 2: ordenar los salones de manera ascendente de acuerdo a su capacidad y disponibilidad.

NOMBRE	MOBILIARIO	CAPACIDAD	DISPONIBILIDAD		
			DIA	HORA INICIAL	HORA FINAL
Salon-2	Escuela	35	1	9 am.	14 pm
Salon-3	Escuela	45	1	9 am.	14 pm
Salon-1	Escuela	50	1	11 am.	14 pm
Salon-2	Auditorio	70	1	9 am.	14 pm
Salon-3	Auditorio	90	1	9 am.	14 pm
Salon-1	Auditorio	100	1	11 am.	14 pm

3) Aplicar la estrategia de control número 5: asignar primero las conferencias cuya asistencia sobrepasen la capacidad de los salones de acuerdo a la siguiente regla de asignación:

3.1) Buscar conferencia cuya asistencia sobrepase a la capacidad de los salones y obtenga su asistencia, mobiliario y duración. En caso de no hallar una conferencia con esta característica, pasar a la regla 4.

CONFERENCIA : Vision
ASISTENCIA-CONF : 140
MOBILIARIO-CONF : Auditorio
DURACION-CONF : 2 hrs.

3.2) Buscar dos salones que puedan ser unidos y amueblados de acuerdo al mobiliario de la conferencia y obtenga la capacidad del salón unido.

salon-12 = salon-1 + salon-2

MOBILIARIO-SALON: ambos de tipo auditorio
CAPACIDAD-SALON : 100 + 70 = 170

Si ASISTENCIA-CONF > CAPACIDAD-SALON + RANGO DE ACEPTACION (de acuerdo a la regla de asignación número 4), la conferencia no puede asignarse a este salón, por lo tanto regresar a 3.1.

3.4) Obténgase la disponibilidad de los salones juntados así como el horario común de ambos.

[salon-1]
DIA: 1 HORA-INICIAL-1: 11 am. HORA-FINAL-1: 14 pm.

[salon-2]
DIA: 1 HORA-INICIAL-2: 9 am. HORA-FINAL-2: 14 pm.

[horario común]
DIA: 1 HORA-INICIAL: 11 am. HORA-FINAL: 14 pm.

Si DURACION-CONF > HORA-FINAL - HORA-INICIAL ó este horario lo ocupa ya otra conferencia del mismo tema (estrategia de control número 3), regresar a 3.1.

3.5) Asignar la conferencia al salón encontrado con un horario de HORA-INICIAL a HORA-INICIAL + DURACION-CONF.

Se asigna la conferencia Visión al salón unido salon-12 con mobiliario tipo auditorio y en horario de 11am a 13pm.

3.6) Eliminar de la tabla de salones los mobiliarios de los salones asignados que sean diferentes al mobiliario de la conferencia (de acuerdo a la estrategia de control número 4), y ajustar su disponibilidad eliminando el horario asignado.

NOMBRE	MOBILIARIO	CAPACIDAD	DISPONIBILIDAD		
			DIA	HORA INICIAL	HORA FINAL
Salon-3	Escuela	45	1	9 am.	14 pm
Salon-2	Auditorio	70	1	13 pm.	14 pm
Salon-2	Auditorio	70	1	9 am.	11 am
Salon-3	Auditorio	90	1	9 am.	14 pm
Salon-1	Auditorio	100	1	13 pm.	14 pm

3.7) Regresar a 3.1.

4) Asignar las conferencias a salones individuales de acuerdo

a la siguiente regla de asignación:

4.1) Buscar conferencias que puedan asignarse a salones individuales y obtener su asistencia, mobiliario y duración. Si ya no hay conferencias, la asignación ha terminado, pasar a la regla 5.

CONFERENCIA : Aprendizaje
ASISTENCIA-CONF : 45
MOBILIARIO-CONF : Escuela
DURACION-CONF : 2 hrs.

4.2) Buscar un salón que satisfaga el mobiliario solicitado por la conferencia y obtenga su capacidad. Si no hay salón con el mobiliario deseado la conferencia no puede asignarse, por lo tanto regresar a 4.1.

SALON : salon-3
MOBILIARIO-SALON: escuela
CAPACIDAD-SALON : 45

Si ASISTENCIA-CONF > CAPACIDAD-SALON + RANGO DE ACEPTACION, la conferencia no puede asignarse a este salón, por lo tanto regresar a 4.2.

4.3) Obténgase la disponibilidad del salón.

[salon-3
DIA: 1 HORA-INICIAL: 9 am. HORA-FINAL: 14 pm.

Si DURACION-CONF > HORA-FINAL - HORA-INICIAL o este horario lo ocupa ya otra conferencia del mismo tema, la conferencia no puede asignarse a este salón, por lo tanto regresar a 4.2.

4.4) Asignar la conferencia al salón encontrado con un horario de HORA-INICIAL a HORA-INICIAL + DURACION-CONF.

Se asigna la conferencia Aprendizaje al salon-3 con mobiliario tipo escuela y en horario de 9 am. a 11 am.

4.5) Eliminar de la tabla de salones los mobiliarios del salón asignado que sean diferentes al mobiliario de la conferencia (de acuerdo a la estrategia de control número 4), y ajustar su disponibilidad eliminando el horario asignado.

NOMBRE	MOBILIARIO	CAPACIDAD	DISPONIBILIDAD		
			DIA	HORA INICIAL	HORA FINAL
Salon-3	Escuela	45	1	11 am.	14 pm
Salon-2	Auditorio	70	1	13 pm.	14 pm
Salon-2	Auditorio	70	1	9 am.	11 am
Salon-1	Auditorio	100	1	13 pm.	14 pm

4.5) Regresar a 4.1.

Repitiendo esta regla para las conferencias que quedan por asignar (Sensores y prolog), la asignación es como sigue.

CONFERENCIA : Sensores
ASISTENCIA-CONF : 65
MOBILIARIO-CONF : Auditorio
DURACION-CONF : 1 hr.

SALON : salon-2
MOBILIARIO-SALON: auditorio
CAPACIDAD-SALON : 70

ASISTENCIA-CONF < CAPACIDAD-SALON + RANGO DE ACEPTACION

[disponibilidad salon-2]
DIA: 1 HORA-INICIAL: 13 pm. HORA-FINAL: 14 pm.

¿Por qué se escogió esta disponibilidad y no la de 9 am a 11 am para el salon-2? El experto no explica que cuando hallemos al mismo salón con dos horarios disponibles, tomemos el que más se ajuste a la duración de la conferencia, para evitar la fragmentación de este salón en muchos horarios pequeños; además, de haber escogido el segundo horario de este salón, éste se traslaparía con una de las conferencias ya asignadas.

DURACION-CONF <= HORA-FINAL - HORA-INICIAL.

Se asigna la conferencia Sensores al salon-2 con mobiliario tipo auditorio y en horario de 13 pm. a 14 pm.

NOMBRE	MOBILIARIO	CAPACIDAD	DISPONIBILIDAD		
			DIA	HORA INICIAL	HORA FINAL
Salon-3	Escuela	65	1	11 am.	14 pm
Salon-2	Auditorio	70	1	9 am.	11 am
Salon-1	Auditorio	100	1	13 pm.	14 pm

CONFERENCIA : Prolog
ASISTENCIA-CONF : 75
MOBILIARIO-CONF : Auditorio
DURACION-CONF : 1 hr. 30'

SALON : salon-2
MOBILIARIO-SALON: auditorio
CAPACIDAD-SALON : 70

ASISTENCIA-CONF < CAPACIDAD-SALON + RANGO DE ACEPTACION (75 < 70 * 1.1 = 77).

[disponibilidad salon-2]
DIA: 1 HORA-INICIAL: 9 am. HORA-FINAL: 11 am.

DURACION-CONF <= HORA-FINAL - HORA-INICIAL.

Se asigna la conferencia Prolog al salon-2 con mobliario tipo auditorio y en horario de 9 am. a 10:30 am.

NOMBRE	MOBILIARIO	CAPACIDAD	DISPONIBILIDAD		
			DIA	HORA INICIAL	HORA FINAL
Salon-3	Escuela	65	1	11 am.	14 pm
Salon-2	Auditorio	70	1	10:30 am.	11 am
Salon-1	Auditorio	100	1	13 pm.	14 pm

5) Ordenar la calendarización efectuada por horarios.

DIA	HORA	CONFERENCIA	ASIS-TENCIA	SALON	MOB	CAPA-CIDAD
1	9:00-10:30	Prolog	75	salon-2	Aud	70
1	9:00-11:00	Aprendizaje	45	salon-3	Esc	45
1	11:00-13:00	Visión	140	salon-12	Aud	140
1	13:00-14:00	Sensores	65	salon-2	Aud	70

Como puede apreciarse en la calendarización definitiva, las tres conferencias del tema de Inteligencia Artificial no se traslapan en sus horarios, dando así oportunidad a los participantes al congreso, interesados en este tema, de asistir a todas ellas.

3.2) EL DISEÑO

Recabada toda la información y el conocimiento necesarios para solucionar el problema, en esta fase diseñamos los cuatro módulos básicos del sistema experto. Comenzamos con la estrategia de control para dar los lineamientos que seguirán las reglas de asignación del sistema, para lograr la asignación óptima de recursos. Posteriormente se presenta la base de conocimientos con los hechos y las reglas de asignación que deben estar presentes en el sistema.

Al llegar a la interface hombre-máquina se explica la manera en que el usuario entabla el diálogo con el sistema, ya sea para introducir información de hechos ó solicitar la asignación de sus recursos.

3.2.1) LA ESTRATEGIA DE CONTROL

Para llevar a cabo la estrategia de control sugerida por nuestro experto, tenemos que realizar algunas acciones antes de poner en marcha al sistema. Las acciones previas a la operación del sistema son: el ordenamiento de las conferencias en grupos de temas y dentro de cada grupo su clasificación ascendente por asistencia a las conferencias. También hay que ordenar los salones por su capacidad en forma ascendente.

Para estos ordenamientos aprovechamos la utilería de SORT que provee el sistema operativo MS-DOS y este se encarga de cumplir con esta actividad.

Las reglas de asignación son ejecutadas de acuerdo a la siguiente estrategia:

1) Solicitar la duración del congreso en días y el rango de aceptación para la capacidad de los salones.

2) Investigar cuales conferencias deben ser asignadas a salones dobles (regla de asignación # 10).

3) Seleccionar un tema de conferencias.

4) Para conferencias con necesidad de salones dobles y evitando el traslape de horarios, se aplica lo siguiente:

   4.1) Asignar conferencias a salones dobles (regla de asignación # 1).
   4.2) Eliminar otros mobiliarios para los salones asignados (regla de asignación # 7).
   4.3) Eliminar el horario asignado de la disponibilidad de los salones asignados (regla de asignación # 6).
   4.4) Suspender temporalmente la disponibilidad de otros salones para el mismo horario asignado, para evitar el traslape de conferencias a un mismo horario (regla de asignación # 8).
   4.5) Si hay más conferencias por asignar regresar a 4.1.

5) Para conferencias con necesidad de salones individuales y evitando el traslape de horarios, se aplica lo siguiente:

   5.1) Asignar conferencias a salón individual (regla de asignación # 2).
   5.2) Eliminar otros mobiliarios para el salón asignado (regla de asignación # 7).
   5.3) Eliminar el horario asignado de la disponibilidad del salón asignado (regla de asignación # 6).
   5.4) Suspender temporalmente la disponibilidad de otros salones para el mismo horario asignado, para evitar el traslape de conferencias a un mismo horario (regla de asignación # 8).
   5.5) Si hay más conferencias por asignar regresar a 5.1.

6) Eliminar los salones que hayan quedado con disponibilidad nula (regla de asignación # 3).

7) Restaurar la disponibilidad de los salones que fueron suspendidos temporalmente para que puedan participar en la siguiente asignación (regla de asignación # 4).

8) Unir la disponibilidad de los salones con fragmentos continuos para obtener una disponibilidad secuencial (regla de asignación # 5).

9) Ordenar la disponibilidad dividida de cada salón de menor a mayor (regla de asignación # 9).

10) Para las conferencias con necesidad de salones dobles y que quedaron sin asignar, se aplica lo siguiente:

   10.1) Asignar conferencias a salones dobles (regla de asignación # 1).
   10.2) Eliminar otros mobiliarios para los salones asignados (regla de asignación # 7).
   10.3) Eliminar el horario asignado de la disponibilidad de

los salones asignados (regla de asignación # 6).
10.4) Si hay más conferencias por asignar regresar a 10.1.

11) Para las conferencias con necesidad de salones individuales y que quedaron sin asignar, se aplica lo siguiente:

11.1) Asignar conferencias a salón individual (regla de asignación # 3).
11.2) Eliminar otros mobiliarios para el salón asignado (regla de asignación # 7).
11.3) Eliminar el horario asignado de la disponibilidad del salón asignado (regla de asignación # 6).
11.4) Si hay más conferencias por asignar regresar a 11.1.

12) Regresar al paso 1.

Veamos ahora como están formados los hechos y reglas en la base de conocimientos.

3.3.2) LA BASE DE CONOCIMIENTOS

Los hechos o información conocida del sistema incluye a las conferencias y salones con los siguientes argumentos:

[CONFERENCIAS]

1) Nombre de la conferencia.
2) Subtema a que pertenece la conferencia.
3) Tema a que pertenece la conferencia.
4) Mobiliario deseado para la conferencia.
5) Duración deseada de la conferencia.
6) Asistencia esperada a la conferencia.
7) Bandera de solicitud de salon doble o individual.

[SALONES]

1) Nombre del salón.
2) Tipo de mobiliario.
3) Capacidad del salón para cada tipo de mobiliario.
4) Disponibilidad:
   4.1) Día disponible.
   4.2) Hora inicial disponible.
   4.3) Número de horas disponibles a partir de la hora inicial.
5) Bandera de disponibilidad suspendida temporalmente.

La estructuración de estos hechos en una base de datos que pueda ser utilizada por el sistema, se realiza utilizando cláusulas del lenguaje de programación prolog, las cuales quedan con el siguiente formato:

Cláusulas para conferencias:

conferencia(NT,AS,DU,MO,TC,SC,NC,SD)

donde:

NT = Número del tema al cual pertenece la conferencia.
AS = Asistencia esperada a la conferencia.
DU = Duración de la conferencia.
MO = Mobiliario deseado para la conferencia.
TC = Tema de la conferencia.
SC = Subtema de la conferencia.
NC = Nombre de la conferencia.
SD = Bandera que indica si la conferencia requiere de salón doble (se actualiza a la hora de ejecución del sistema).

Cláusulas para salones:

1) salon(DIA,CAPACIDAD,MOBILIARIO,NOMBRE,SUSPENCION).
2) h_salon(DIA,HORA_INI_DISP,DISPONIBILIDAD,NOMBRE,SUSPENCION).
3) j_salon(NOMBRE_SALON,NOMBRE_SALON,NOMBRE_SALON)

donde:

DIA = Número de día (dentro de la duración del congreso), en que se encuentra disponible este salón.
CAPACIDAD = capacidad del salón (número de personas), para el mobiliario especificado en MOBILIARIO.
NOMBRE = Nombre del salón.

HORA_INI_DISP = hora a partir de la cual el salón se encuentra disponible dentro de DIA.
DISPONIBILIDAD = Número total de horas disponibles (consecutivas) para el salón.
SUSPENCION = Bandera de suspención temporal de disponibilidad.

En la claúsula de salon( ), queda asentada la capacidad del salón dependiendo del tipo de mobiliario.

En la claúsula de h_salon( ) se incluye la disponibilidad del salón por día y hora de inicio disponible y la claúsula j_salon( ) nos indica los salones que pueden juntarse para formar uno con mayor capacidad.

El conocimiento incluido queda representado por las siguientes reglas de asignación:

Regla de Asignación # 1: asignación de conferencia a salones dobles.

La regla es verdadera si se cumplen las siguientes condiciones:

1.1) Existen dos salones que puedan ser unidos.
1.2) El día disponible de ambos salones es el mismo y cae dentro de la duración del congreso.
1.3) La conferencia a ser asignada pertenece al tema en proceso, el mobiliario que solicita lo poseen los salones disponibles y la asistencia esperada puede ser cubierta por las capacidades conjuntas de los

salones.

1.4) Existe un horario común disponible para ambos salones que satisface la duración de la conferencia.

Regla de Asignación # 2: asignación de conferencias a salones individuales.

La regla es verdadera si se cumplen las siguientes condiciones:

1.1) Existen algún salón disponible.
1.2) El día disponible del salón cae dentro de la duración del congreso.
1.3) La conferencia a ser asignada pertenece al tema en proceso, el mobiliario que solicita lo posee el salón disponible y la asistencia esperada puede ser cubierta por la capacidad del salón.
1.4) El horario disponible del salón satisface la duración de la conferencia.

Regla de asignación # 3: eliminar de la base de datos salones cuya disponibilidad sea nula.

La regla es verdadera si existe algún salón cuya disponibilidad haya quedado en ceros debido a asignaciones anteriores.

Regla de asignación # 4: Restaurar los horarios de salones para que puedan participar en asignaciones posteriores.

La regla se cumple si existe algún salón cuya disponibilidad haya sido suspendida temporalmente para evitar el traslape de horarios.

Regla de asignación # 5: juntar horarios fragmentados de un mismo salón que sean consecutivos.

La regla se cumple si en un momento dado encontramos en la base de datos un salón, con por lo menos dos horarios fragmentados de la siguiente manera:

NOMBRE	DIA DISPONIBLE	HORA INICIAL	HORA FINAL
salon-1	1	9:00 am.	9:30 am.
salon-1	1	9:30 am.	11:00 am.

En este caso el objetivo de la regla es unir los fragmentos para que la base de datos contenga:

NOMBRE	DIA DISPONIBLE	HORA INICIAL	HORA FINAL
salon-1	1	9:00 am.	11:00 am.

Regla de asignación # 6: Eliminar de la base de datos el horario asignado del salón o salones asignados a la conferencia.

La regla se cumple si se ha efectuado una asignación conferencia-salón-horario.

Regla de asignación # 7: Eliminar otros mobiliario para un salón de la base de datos.

La regla se cumple si se ha efectuado una asignación conferencia-salón-horario.

Regla de asignación # 8: suspender temporalmente la disponibilidad de salones para el horario asignado a una conferencia.

La regla se cumple si se ha efectuado una asignación conferencia-salón-horario y se desea evitar el traslape de horarios.

Regla de asignación # 9: ordenar la disponibilidad de los salones de manera ascendente.

La regla se cumple si en la base de datos se nos presenta el siguiente caso:

NOMBRE	DIA DISPONIBLE	HORA INICIAL	HORA FINAL
salon-1	1	9:00 am.	12:00 am.
salon-1	1	16:00 pm.	17:00 pm.

El objetivo de la regla es acomodar este salón de la siguiente manera:

NOMBRE	DIA DISPONIBLE	HORA INICIAL	HORA FINAL
salon-1	1	16:00 pm.	17:00 pm.
salon-1	1	9:00 am.	12:00 am.

ya que hay menor disponibilidad de 16:00 pm. a 17:00 pm. (1 hora) que de 9:00 am. a 12:00 am. (3 horas).

Regla de de asignación # 10: Identificar las conferencias que necesiten de salones dobles para ser asignadas.

La regla se cumple si la asistencia de una conferencia no puede ser cubierta por ninguno de los salones de la base de datos de manera individual.

3.2.3) LA MAQUINA DE INFERENCIAS: PROLOG

Hemos diseñado ya dos de los cuatros módulos básicos de nuestro sistema experto: la base de conocimientos y la estrategia de control. Para que el sistema pueda procesar los hechos y aplicar las reglas de asignación, es necesario el módulo de inferencias ó conjeturas.

El lenguaje de programación Prolog posee la característica

de ser un lenguaje declarativo. Con esto se quiere decir que al codificar un programa, no hay que decirle a Prolog cómo debe de hacer las cosas, sino nada más qué es lo que debe hacer y Prolog se encarga de los procedimientos para llevarlas a cabo.
Las declaraciones de Prolog se dividen en dos grupos: hechos y reglas. Los hechos se codifican en forma de claúsulas con el formato:

```
objeto(atributo_1,atributo_2,...,atributo_n )
```

y las reglas tienen la forma de:

```
Si [antecedente] entonces [consecuente].
```

En Prolog puede haber tantas reglas como se deseen. Se establece un objetivo, se dan los hechos y Prolog aplica las reglas contenidas en su base de conocimientos para tratar de satisfacer el objetivo.

Una regla de asignación codificada en Prolog es por ejemplo:

```
asigna si
/*1*/    conferencia(NOMBRE_CONF,MOBILIARIO,ASISTENCIA) y
/*2*/    salon(NOMBRE_SALON,MOBILIARIO,CAPACIDAD) y
/*3*/    ASISTENCIA <= CAPACIDAD y
/*4*/    escribe(NOMBRE_CONF,MOBILIARIO,ASISTENCIA,
          NOMBRE_SAL,CAPACIDAD).
```

La información de hechos se codifica de la siguiente manera:

```
conferencia("Sensores","Auditorio",65)
conferencia("Prolog","Escuela",40)
salon("Salon-1","Auditorio",70)
salon("Salon-1","Escuela",35)
salon("Salon-2","Auditorio",100)
salon("Salon-2","Escuela",50)
```

En la regla de asignación estamos declarando qué es lo que queremos hacer, pero no estamos diciendo cómo se va a hacer. Es tarea de Prolog el tratar de satisfacer la regla.

El funcionamiento de Prolog está basado principalmente en dos aspectos:

1) La unificación de claúsulas y argumentos.
2) La búsqueda hacia atrás (backtraking).

Veamos el funcionamiento de Prolog utilizando la regla de asignación y los hechos declarados líneas arriba:

Prolog encuentra dentro de la regla la claúsula:

```
conferencia(NOMBRE_CONF,MOBILIARIO,ASISTENCIA)
```

Esta claúsula se llama "conferencia" y tiene tres argumentos

ó características: NOMBRE_CONF, MOBILIARIO y ASISTENCIA.

Para unificar esta claúsula, Prolog busca en la base de datos alguna claúsula que se llame también "conferencia", con valores para los argumentos NOMBRE_CONF, MOBILIARIO y ASISTENCIA.
Buscando secuencialmente, la primera claúsula que cumple con estos requisitos es:

```
conferencia("Sensores","Auditorio",65)
```

Entonces Prolog unifica los argumentos asignando el valor a las variables:

```
NOMBRE_CONF = Sensores
MOBILIARIO  = Auditorio
ASISTENCIA  = 65
```

Como Prolog tuvo éxito en la unificación de "conferencia", pasa el control a la siguiente claúsula de la regla:

```
salon(NOMBRE_SALON,MOBILIARIO,CAPACIDAD)
```

Solo que a la variable MOBILIARIO ya se le asignó valor en la claúsula anterior, por lo que en realidad Prolog tratará de unificar la claúsula:

```
salon(NOMBRE_SALON,"Auditorio",CAPACIDAD)
```

y se va a buscar a la base de datos la primera claúsula con la que se pueda unificar:

```
salon("Salon-1","Auditorio",70)
```

La unificación produce:

```
NOMBRE_SALON = Salon-1
CAPACIDAD    = 70
```

La tercera parte de nuestra regla es una comparación entre los valores de dos argumentos:

```
ASISTENCIA <= CAPACIDAD
```

o tomando los valores ya asignados a las variables:¿65 <= 70?

Como la comparación es verdadera se pasa el control a la última aseveración de la regla que es:

```
escribe(NOMBRE_CONF,MOBILIARIO,ASISTENCIA,NOMBRE_SAL,
        CAPACIDAD)
```

en cuyo caso se imprimen los recursos asignados o unificados:

NOMBRE	MOBILIARIO	ASISTENCIA	SALON	CAPACIDAD
Sensores	Auditorio	65	Salon-1	70

Comienza ahora el proceso de búsqueda hacia atrás (backtracking), en donde Prolog regresa a la última claúsula que unificó, para tratar de obtener más unificaciones que también satisfagan a la regla: por lo tanto, en nuestra regla "asigna", Prolog buscan otras claúsulas de salones que se pueda unificar con:

```
salon(NOMBRE_SALON,"Auditorio",CAPACIDAD)
```

Encuentra a la claúsula:

```
salon("Salon-2","Auditorio",100)
```

y realiza las unificaciones:

```
NOMBRE_SALON = Salon-2
CAPACIDAD    = 100
```

Como si se cumple la comparación ASISTENCIA <= CAPACIDAD (65 <= 100), Prolog también escribe esta asignación:

NOMBRE	MOBILIARIO	ASISTENCIA	SALON	CAPACIDAD
Sensores	Auditorio	65	Salon-2	100

Nótese que esta nueva unificación no alteró el valor de los argumentos de la claúsula "conferencia".

Como ya no existen otras claúsulas que satisfagan a:

```
salon(NOMBRE_SALON,"Auditorio",CAPACIDAD)
```

Prolog realiza una nueva búsqueda hacia atrás, pero esta vez sobre la claúsula "conferencia", y averigua que existe:

```
conferencia("Prolog","Escuela",40)
```

por lo que comienza a aplicar de nuevo la claúsula de salón:

```
salon(NOMBRE_SAL,"Escuela",CAPACIDAD)
```

sobre: salon("Salon-1","Auditorio",70)

Estas dos claúsula no pueden unificarse, ya que la "conferencia" tiene especificado un mobiliario "Escuela" y la claúsula hallada define un mobiliario "Auditorio". Prolog entonces toma la siguiente claúsula:

```
salon("Salon-1","Escuela",35)
```

La cual si cumple con las reglas de unificación:

```
NOMBRE_SALON = Salon-1
CAPACIDAD    = 70
```

En resumen, ésta regla de asignación genera las siguientes unificaciones:

CONFERENCIA	MOBILIARIO	ASISTENCIA	SALON	CAPACIDAD
Sensores	Auditorio	65	Salon-1	70
Sensores	Auditorio	65	Salon-2	100
Prolog	Escuela	40	Salon-2	50

La conclusión de todo este proceso, es que Prolog evalúa una regla, aplicando todas las posibles combinaciones que pueda hallar en su base de conocimientos, para satisfacer a la regla, y aquí es donde se ve la importancia de la estrategia de control, ya que es ésta la que coordina cuales de estas combinaciones son las que llevan al sistema experto a la solución del problema.

3.2.4) LA INTERFACE HOMBRE-MAQUINA

La interface tiene por objeto la captura de la información para las dos bases de datos utilizadas por el sistema. Cada una contiene datos sobre diferentes objetos. En una está todo lo relacionado a las conferencias y en la otra lo de los salones disponibles en cada hotel o centro de convenciones.

La información para estas dos bases de datos es almacenada en tres archivos independientes, a partir de los cuales se generan las bases de datos con la sintaxis utilizada por Prolog vista anteriormente. Toda esta transformación se hace de manera transparente para el usuario por medio de un programa traductor.

Los registros de cada archivo estan formados por las siguientes estructuras de datos:

REGISTRO DE CONFERENCIAS

Datos que contiene:	Tipo de variable
Clave de conferencia	ENTERA
Asistencia esperada	ENTERA
Duración	REAL
Tipo de mobiliario	ALFANUMERICA
Clave de tema	ENTERA
Clave de subtema	ENTERA
Nombre de la conferencia	ALFANUMERICA

REGISTRO DE SALONES

Datos que contiene	Tipo de variable
Nombre	ALFANUMERICA
Capacidad de acuerdo a mobiliario	VECTOR ENTERO[4]
Hora de inicio	VECTOR REAL[10]
Horas disponibles	VECTOR REAL[10]

REGISTRO DE PAREJAS DE SALONES CON POSIBILIDAD DE UNIRSE Y FORMAR UNO SOLO.

Datos que contiene	Tipo de variable
Clave de salón_1	ENTERA
Nombre de salon_1	ALFANUMERICA
Clave de salón_2	ENTERA
Nombre de salón_2	ALFANUMERICA
Clave de salón_12	ENTERA
Nombre de salón_12	ALFANUMERICA

También son utilizados dos archivos de datos auxiliares que contienen los temas y subtemas de las conferencias a exponer, que podrá ser alimentado en el mismo programa de captura. Ambos registros son iguales.

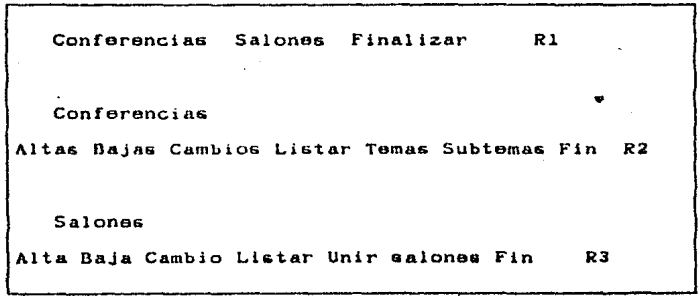
REGISTRO PARA TEMAS Y SUBTEMAS

Datos que contiene:	Tipo de variable:
Clave de tema	ENTERA
Nombre de tema	ALFANUMERICA

El método para captura usado es el de pantalla/menú, por lo que resulta sumamente sencillo su uso. A continuación se explica, paso a paso, su funcionamiento y restricciones. (Para más información de esta interface en cuanto a su programación ver apéndice número 3).

La explicación se hará de la forma como se van presentando las pantallas a manera de manual de usuario.

PANTALLA PRINCIPAL



Objetivo: Elección de la función que se desea realizar.

Respuestas válidas R1,R2,R3: Primer caracter del nombre de cada función.

Procedimiento al que conduce:

R1 : Movimientos sobre el archivo de salones o conferencias de acuerdo a elección.

R2 : Funciones de mantenimiento sobre el archivo de conferencias, mantenimiento a los archivos de temas y subtemas.

R3 : Funciones de mantenimiento sobre el archivo de salones, captura de las pareja de salones con posibilidad de unión.

PANTALLA GENERAL CONFERENCIA.

Clave de conferencia	R1
Nombre	R2
Clave de tema	R3
Clave de subtema	R4
Duración	R5
Mobiliario	R6
Asistencia	R7

Objetivo : Máscara utilizada para los procesos de alta baja y cambio, por medio de la cuzl haremos los movimientos sobre las variables del registro del archivo de conferencias.

Respuestas válidas:

R1 : Entera, es la clave numérica asociada a la conferencia, por medio de la cual será reconocida.

R2 : Alfanumérica, con longitud de 50 caracteres.

R3 : Entera, es la clave del tema al que pertenezca la conferencia. Tema previamente capturado en el archivo de temas.

R4 : Real, la duración en horas.minutos de la conferencia.

R5 : Entero, tipo de mobiliario necesario de acuerdo a las siguientes claves:

1) escuela 2) auditorio 3) comité 4) alimentos

R6 : Entero, asistencia esperada a la conferencia.

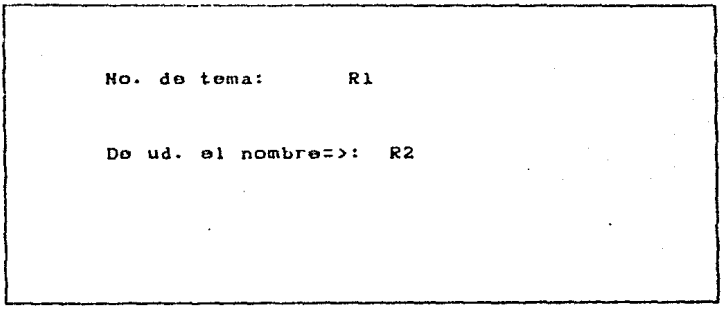
Adecuaciones de acuerdo a la función elegida:

Para altas: El cursor se posiciona en las celdas para cada variable secuencialmente en cuanto la variable anterior se halla capturado correctamente.

Para bajas: Al elegir esta opción, aparece en pantalla el cuestionamiento acerca de que si es el registro a borrar, respondiendo con un si o no de acuerdo a cada situación.

Para cambios: Se pregunta la clave de la variable que se desea cambiar, a lo que se responde con el caracter en mayúscula en el nombre del campo.

PANTALLA DE TEMAS



Objetivo: Máscara utilizada para la captura de temas y subtemas, unicamente es posible dar altas y bajas.

Respuestas válidas:

R1 : Entera, clave de tema o subtema que podrá ser capturada en la parte de conferencias.

R2 : Alfanumérica, descripción del tema.

Adecuaciones de acuerdo a la función elegida:

Ninguna, pantalla única, si la clave de conferencia ya está ocupada pregunta si desea darse de baja.

PANTALLA PRINCIPAL (AL ELEGIR FUNCION DE SALONES).

```
   Conferencias  Salones  Finalizar          S


   Conferencias

Altas Bajas Cambios Listar Temas Subtemas Fin


   Salones

Alta Baja Cambio Listar Unir salones Fin    A
CENTRO DE CONVENCIONES [8 CARACTERES]=>     R1
```

Objetivo: Indicar qué archivo para los salones será utilizado ya que se puede, con la misma conferencia, hacer varias pruebas con diferentes centros de convenciones.

Respuestas válidas:

R4 : Alfanumérica, nombre del lugar con una máximo de ocho caracteres.

PANTALLA DE SALONES

```
clave del salón:          R1
Nombre:                   R2
Capacidad de acuerdo
al mobiliario
ESCUELA:                  R3
AUDITORIO:                R4
COMITE:                   R5
ALIMENTOS:                R6
Horario general:
DE INICIO:                R7
DISPONIBLE:               R8


     DIA RELATIVO:        R9
     HORA INICIAL:        R10
     DISPONIBLE:          R11
```

Objetivo: Máscara utilizada para el mantenimiento y captura del archivo de salones. El cursor se posiciona en cada celda correspondiente a cada una de las variables del registro de salones.

Respuestas válidas:

R1 : Entera, clave numérica del salón para su manejo

sencillo.

R2 : Alfanumérica, nombre asociado al salón.

R3 : Entera, capacidad usando mobiliario tipo escuela.

R4 : Entera, capacidad usando mobiliario tipo auditorio.

R5 : Entera, capacidad usando mobiliario tipo comité.

R6 : Entera, capacidad usando mobiliario tipo comedor.

R7 : Real, Horario de inicio de actividades. Formato Horas.mins.

R8 : Real, Total de horas disponibles. Formato horas.mins.

R9 : Entero. Para el caso de que algún día de duración de la conferencia sea diferente y debido a que todos son inicializados con los datos anteriores (R7,R8), se ha agregado esta posibilidad. El día relativo se refiere a su posición relativa considerando al primer día de conferencias como día uno.

R10 : Real, misma que R7.

R11 : Real, misma que R8.

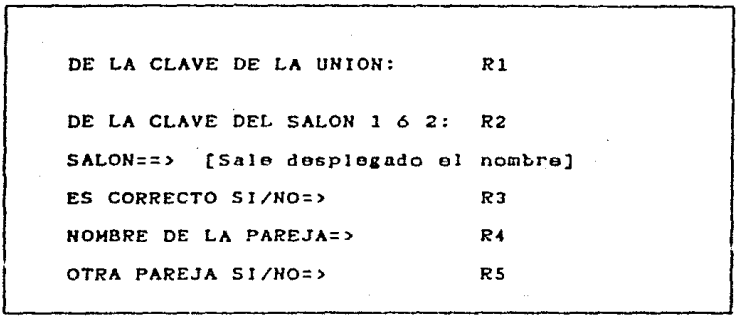
Adecuaciones de acuerdo a la función elegida:

Para altas: El cursor se posiciona secuencialmente en cuanto la variables inmediata anterior haya sido capturada correctamente.

Para bajas: Aparece la información que se pretende borrar cuestionando acerca de que si es la que se desea inicializar en ceros.

Para cambios: Se pregunta la clave de la variable a cambiar, a lo que se responde con el caracter en mayúscula en el nombre del campo. Se procede de acuerdo a la función de altas.

PANTALLA PARA LA UNION DE SALONES.



Objetivo: Máscara para mantenimiento del archivo de salones con posibilidad de unir. Es posible tener unicamente las funciones de alta y baja.

Respuestas válidas:

R1 : Entera, clave de la unión.

R2 : Entera, clave de uno de los salones a unir.

R3 : Alfanumérica, afirmación o negación.

R4 : Alfanumérica, nombre a la pareja unida.

R5 : Alfanumérica, afirmación o negación.

Adecuaciones de acuerdo a la función elegida.

Para altas: El cursor se posiciona secuencialmente en cuanto la variables inmediata anterior halla sido capturada correctamente.

Para bajas: En caso de que la clave para unión haya sido ya dada de alta el programa lo avisará preguntando si se desea dar de baja.

CAPITULO

LA INDUSTRIA DEL CONOCIMIENTO

La importancia de los sistemas expertos comenzó a notarse en 1980; año en que se efectuó la primera conferencia internacional sobre inteligencia artificial. El comité organizador, la asociación americana para inteligencia artificial, esperaba la participación de cientos de personas y agrupaciones, pero la asistencia llegó a miles, incluyendo a hombres de ciencias, reporteros y capitalistas de riesgo, cuyo propósito era la comercialización de los sistemas expertos.

Pero no toda la inteligencia artificial se refiere a los sistemas expertos, hay otras áreas, como lo es el reconocimiento de formas y el entendimiento del lenguaje humano que acaparan la atención mundial. La tendencia es que todo razonamiento, entendimiento, solución de problemas y aprendizaje es pensado ahora en construirse fundamentándose en conocimiento, ya que este permite la generación de acciones inteligentes.

La siguiente generación de computadoras empleará las técnicas de inteligencia artificial para procesar la información contenida en ellas por medio de acciones inteligentes; no puede afirmarse todavía que las computadoras serán inteligentes por sí mismas, ya que el problema de hacer máquinas creativas deriva de la naturaleza de la inteligencia en el ser humano y hasta ahora ninguna ciencia ha descubierto como se lleva a cabo este fenómeno en la mente de las personas.

Hay gran actividad a nivel mundial por el desarrollo de supercomputadoras y de software basado en técnicas de inteligencia artificial; el reto lo lanzó Japón en 1982 al comenzar formalmente su proyecto de la quinta generación. Como respuesta a la actividad japonesa Europa, Gran Bretaña y Estados Unidos entraron a la competencia.

JAPON: LA QUINTA GENERACION

En 1982, el ministerio japonés de industria y comercio internacional (MITI: ministery of international trade and industry), puso en marcha un proyecto a diez años para desarrollar la quinta generación de computadoras. Este proyecto lleva el nombre de ICOT, qué es el instituto para la generación de nueva tecnología computacional (institute of new-generation computer technology). Los sistemas que se desarrollan en ICOT estarán orientados hacia el procesamiento de conocimiento, ya sea para desarrollar sistemas expertos,

reconocimiento de formas, solución de problemas y entendimiento del lenguaje humano. El software empleado tiene que ser de un alto nivel lógico, para lo cuál Japón comenzó a utilizar el lenguaje de programación Prolog (Programming in Logic), aunque el producto final será un lenguaje llamado HIMIKO para la programación lógica de las nuevas computadoras.

El hardware necesario para implementar el software desarrollado va a ser diferente a la arquitectura Von Newman actualmente empleada, siendo la orientación de éste la construcción de máquinas con capacidad de procesamiento en paralelo. A estas computadoras los japoneses las han denominado "sistemas de procesamiento de información y conocimiento" (KIPS: knowledge and information processing systems). Su velocidad de proceso será cuantificada en términos del número de inferencias lógicas por segundo que puedan efectuar (LIPS: logical inferences per second) y su objetivo es lograr velocidades cercanas a los gigalips, siendo:

1 GIGALIPS = 1,000,000,000 LIPS y
1 LIPS = entre 100 y 100,00 instrucciones de máquina por segundo (MIPS: machine instructions per second).

GRAN BRETAÑA

A raíz del anuncio japonés de su quinta generación de computadoras, Gran Bretaña respondió con un proyecto preliminar a cinco años, que de cumplir con los objetivos planteados, se extenderá otros cinco años más. El grupo inglés es encabezado por el investigador británico John Alvey y contempla los siguientes desarrollos:

1) Ingeniería de software.
2) Supercomputadoras con un muy alta escala de integración.
3) Sistemas basados en conocimiento.
4) Interfaces hombre-máquina.

En este proyecto participa el gobierno, la industria y las universidades, cuyos objetivos son:

1) Ingeniería de software:
Desarrollo de procedimientos y técnicas para lograr un ambiente de soporte a la programación con el establecimiento de un centro de producción de software y uno de control de calidad.

2) Supercomputadoras:
Pretende entrar al mercado mundial de la tecnología VLSI (muy alta escala de integración), desarrollando nuevas herramientas de diseño asistido por computadora (CAD: computer aided design) para la creación de microcircuitos.

3) Sistemas basados en conocimiento:
Es la contraparte de los KIPS japoneses que los ingleses llaman "sistemas basados en información y conocimiento"

(IKBS: information and knowledge base sistems) en los cuales se desarrollarán los siguientes temas:

a) Representación del conocimiento.
b) Solución de problemas.
c) Desarrollo de bases de datos inteligentes.
d) Técnicas de adquisición de conocimiento.
e) Desarrollo de un prototipo de máquina IKBS.

4) Interface hombre-máquina:

La comunicación hombre-máquina se desarrolla en las siguientes etapas:

a) Diseño de diálogos.
b) Modos de entrada/salida y su eficiencia.
c) Identificación de las habilidades humanas para entablar una comunicación para que los nuevos sistemas puedan ser utilizados como tutores o consultores.
d) Análisis del comportamiento humano al solucionar problemas complejos.
e) Investigar a que nivel una computadora deberá imitar el comportamiento humano al solucionar problemas.
f) Procesamiento de imágenes y sonidos.

EUROPA

Conforme pasan los años, los países de la comunidad económica europea se han vuelto cada vez más dependientes de Japón y Estados Unidos con la importación de chips y software de aplicación. Por lo mismo estos países se están quedando atrás en el desarrollo tecnológico y nadie les asegura que no habrá una guerra comercial que los deje sin la nueva tecnología, es por ello que decidieron entrar a la competencia y planear su propia estrategia.

En julio de 1983 se formó el programa estratégico europeo para la investigación de tecnología en la información (ESPRIT: european strategic program for research in information technology); proyecto planeado a seis años con los siguientes objetivos:

1) Microelectrónica avanzada.
2) Tecnología de software.
3) Procesamiento avanzado de información.
4) Automatización de oficinas.
5) Sistemas de intercambio de información.

El proyecto contempla la participación de universidades, la industria y el gobierno de los países miembros de la comunidad económica europea, incluyendo a Gran Bretaña, aunque este país cuenta con su propio proyecto de desarrollo tecnológico.

ESTADOS UNIDOS

La respuesta de Estados Unidos no ha sido a nivel

nacional, es decir, no han formulado un proyecto estratégico a largo plazo como el de los japoneses, británicos o europeos. Sin embargo, la actividad en norteamérica es fuerte a través de universidades e industrias. La mayor aportación a la inteligencia artificial la generan la universidades de Standford y el Instituto Tecnológico de Masachussets, cuyos objetivos incluyen la elaboración de sistemas expertos y la aplicación de las técnicas de inteligencia artificial a la solución de problemas industriales.

Por lo que respecta a las industrias, estas incluyen a los laboratorios Bell, Digital Equipment Corporation (DEC), Fairchild, American Telegraph and Telephon (ATT), Hewlett Packard, Machine Intelligent Corporation y Texas Instruments entre otras.

La participación del gobierno se enfoca principalmente al desarrollo de sistemas expertos para la industria militar y por supuesto, IBM no está con los brazos cruzados esperando el éxito o fracaso japonés; se sospecha que ha estado trabajando ya desde hace tiempo en su propio proyecto de quinta generación.

APENDICE 1: MERCADO ACTUAL DE LOS SISTEMAS EXPERTOS

A continuación se presenta una tabla de sistemas expertos en uso actualmente (1). Se presenta el nombre del sistema, el área ó ciencia para el cual fué desarrollado y su aplicación.

Area	Sistema	Utilidad
Administración	KM-I	Asesoría en la administración.
	Rabbit	Ayuda a usuarios a formular preguntas a una base de datos.
	Sin Nombre	Asesoría para el desarrollo de proyectos de alto riesgo en el campo de la construcción.
	Callisto	Modela, controla, monitorea, calendariza y administra proyectos a largo plazo.
Bioingeniería	Molgen	Planeación de experimentos de análisis estructural y síntesis del DNA.
Computación	Dart	Diagnóstico de fallas de computadora.
	R1 ó Xcon	Configura arquitecturas de computadoras VAX.
	Spear	Diagnóstico de fallas de computadora.
	Xsel	Asistencia para seleccionar una configuración de computadora.
	Sin Nombre	Diagnóstico de fallas de computadoras VAX.
	Pro-App	Asistencia en el diseño y depuración de software.
	PSI	Desarrolla programas sencillos basado en su descripción por medio de oraciones.

Area	Sistema	Utilidad
De uso general	Age	Ayuda en el desarrollo de sistemas expertos.
	AL/X	Ayuda a codificar el conocimiento de un experto.
	Emycin	Sistema básico de inferencias.
	Expert	Sistema básico de inferencias aplicado a medicina y exploración de petróleo.
	Kepe	Sistema para la representación de una base de conocimientos.
	KS-300	Diagnóstico y asesoría industrial
	MRS	Representación de conocimiento y estrategia de control para solucionar problemas.
	Teiresas	Transfiere conocimiento del experto a la computadora y guía en la adquisición de nuevas reglas.
Educación	Guidon	Enseñanza por medio de una selección de preguntas de acuerdo al nivel del estudiante.
	Sin Nombre	Enseñanza de lenguajes de programación a estudiantes.
Exploración	Dipmeter Advisor	Analiza información de posibles yacimientos de petróleo.
	Drilling Advisor	Diagnóstico de problemas de perforación y su corrección.
	Hydro	Consejos para solucionar problemas de abastecimiento de agua.
	Prospector	Evalúa posibles yacimientos de minerales.
	Waves	Análisis de información sísmica para la industria petrolera.
Genética	Genesis	Planeación de experimentos genéticos.

Area	Sistema	Utilidad
Ingeniería	Eurisko	Diseño de microcircuitos. El sistema aprende de sus descubrimientos.
	KBVLSI	Diseño de circuitos VLSI.
	Sacon	Asiste a ingenieros a tomar la mejor estrategia de análisis en problemas estructurales.
	Sin Nombre	Control de reactores nucleares.
	Sin Nombre	Diagnóstico de errores de fabricación de circuitos integrados.
Leyes	LDS	Modela el proceso de decisión de abogados y aconseja sobre la legislación.
	Taxman	Asesora la estrategia financiera de una compañía para conseguir sus objetivos.
Medicina	Abel	Diagnóstico de desórdenes de base ácida electrolítica.
	Caduceus	Diagnóstico para medicina interna
	Casnet	Consulta de tratamientos para varios posibles diagnósticos.
	Mycin	Diagnóstico de meningitis e infecciones sanguíneas.
	Oncocin	Consulta sobre tratamiento de quimio-terapia.
	Puff	Analiza posibles desórdenes pulmonares.
	VM	Monitoreo de pacientes en cuidado intensivo.

Area	Sistema	Utilidad
Militar	Airplan	Tráfico militar alrededor de un portaaviones.
	HASP SIAP	Rastreo e identificación de barcos usando señales de sonar.
	Tatr	Bombardeo de objetivos desde el aire.
	Sin Nombre	Análisis de comunicaciones militares.
Química	Dendral	Interpreta datos de espectrómetros de masas y determina la estructura molecular y atómica.
	Secs	Asistencia en la planeación de síntesis orgánica.

(1) Fuente:

Edward A. Feigenbaum, "The fifth generation", 1983.
Patrick H. Winston, "Commercial uses of A.I.", 1985.

APENDICE 2: LISTADO DEL SISTEMA EXPERTO

A continuación se presenta el listado completo del sistema experto para la solución de un problema de asignación de recursos. El primer listado corresponde al programa en lenguaje de programación turbo prolog que es el que da la solución al problema. El segundo listado es en pascal y corresponde a la interface hombre-máquina para la captura de la base de datos del sistema.

```
[TURBO PROLOG]

code=2000
domains

tema_conf,subt_conf,nom_conf,mobi,nom_sal=symbol
dia,quorum,capacidad,cve_sal,cve_conf,asi_conf=integer
duracion,disponible,hora=real
file=arch_asigna;conf_msg;sal_msg

database

conferencia(cve_conf,quorum,duracion,mobi,tema_conf,subt_conf
            ,nom_conf,asi_conf)
salon(dia,capacidad,mobi,nom_sal)
h_salon(dia,hora,disponible,nom_sal,cve_sal)
j_salon(nom_sal,nom_sal,nom_sal)
tema_conf(cve_conf)

predicates

inicia
menu_uno
menu_dos(string)
repite
menu_principal
asigna(string,integer,integer)
recursos(string,integer)
quita_mob(mobi,nom_sal,dia)
quita_hora(dia,hora,hora)
minimo(hora,hora,hora)
maximo(hora,hora,hora)
imprime(string,string)
asigna_st(integer,cve_conf,integer)
asigna_ct(integer,integer)
borra_jun(dia,hora,disponible,nom_sal)

goal

inicia.

clauses

repite.
repite if repite.
```

```
inicia if
    system("BORRA"),menu_principal.
menu_principal if
    makewindow(1,7,7,"ASIGNACION DE RECURSOS",0,0,6,80),
    write("Archivo de conferencias: "),
    readln(ARCH_CONF),
    write("Archivo de salones: "),
    readln(ARCH_SAL),
    existfile(ARCH_CONF),
    existfile(ARCH_SAL),
    write("cargando base de datos, espere por favor..."),
    consult(ARCH_SAL),
    consult(ARCH_CONF),
    menu_uno,!.
menu_principal if
    write("Archivo      inexistente    o    base    de    datos
    inconsistente"),nl,
    write("Oprima <RETURN> para proseguir"),readchar(_),
    menu_principal.
menu_uno if
    repite,
    makewindow(2,7,7,"",6,0,10,80),
    write("A = Asignar recursos"),nl,
    write("R = Recursos no asignados"),nl,
    write("F = Fin de asignacion"),nl,
    write("Opcion:
       "),readln(OPCION),upper_lower(OPCION_U,OPCION),
    menu_dos(OPCION_U),fail.
menu_dos("A") if
    write("Duracion del congreso: "),readint(DIAS_CONG),
    write("Tolerancia en capacidad de salones (en porcentaje):
       "),readint(TOL),
    asigna("A",DIAS_CONG,TOL).
menu_dos("R") if
    write("Duracion del congreso: "),readint(DIAS_CONG),
    recursos("R",DIAS_CONG).
menu_dos("F") if
    removewindow,removewindow,exit.

/* SEPARA CONFERENCIAS QUE NECESITAN SALONES COMPARTIDOS */
asigna("A",DIAS_CONG,TOL) if
    conferencia(CVE_CONF,QUORUM,DURACION,MOBI,TEMA,SUBT_CONF,
       NOM_CONF,0),
    write(NOM_CONF),nl,
    salon(DIA,CAPA,MOBI,_),
    DIA <= DIAS_CONG,
    conferencia(CVE_CONF,QUORUM,DURACION,MOBI,TEMA,SUBT_CONF,
       NOM_CONF,0),
    QUORUM <= CAPA*(1+TOL/100),
    retract(conferencia(CVE_CONF,QUORUM,DURACION,MOBI,TEMA,
       SUBT_CONF,NOM_CONF,0)),
    assertz(conferencia(CVE_CONF,QUORUM,DURACION,MOBI,TEMA,
       SUBT_CONF,NOM_CONF,1)).
asigna("A",_,_) if
    existfile("RECURSOS.ASG"),
    openappend(arch_asigna,"RECURSOS.ASG").
asigna("A",_,_) if
```

```
    not(existfile("RECURSOS.ASG")),
    openwrite(arch_asigna,"RECURSOS.ASG").
asigna("A",_,_) if
    makewindow(3,7,7,"ASIGNACION",0,0,24,80),
    write("DIA     HORA          NOMBRE                    ASI
       MOBILIARIO "),
    write("SALON"),nl.
asigna("A",DIAS_CONG,TOL) if
    tema_conf(CVE_CONF),
    asigna_st(DIAS_CONG,CVE_CONF,TOL).
asigna("A",DIAS_CONG,TOL) if
    asigna_ct(DIAS_CONG,TOL).
asigna("A",_,_) if
    writedevice(screen),nl,write("Oprima <RETURN> o 'P' para
       imprimir "),
    readln(IMP),upper_lower(IMP_U,IMP),removewindow,
    shiftwindow(2),imprime("A",IMP_U).

/* RECURSOS NO ASIGNADOS */
recursos("R",_) if
    openwrite(conf_nsg,"CONFEREN.NSG"),
    openwrite(sal_nsg,"SALONES.NSG"),
    makewindow(3,7,7,"RECURSOS NO ASIGNADOS",0,0,24,80),
    writedevice(screen),
    write("CVE     CONFERENCIA            ASISTENCIA  DURACION
       MOBILIARIO"),nl,
    conferencia(CVE_CONF,QUORUM,DURACION,MOBI,TEMA_CONF,
       SUBT_CONF,NOM_CONF,_),
    writedevice(screen),
    writef("%2%2%-20%5%4%8%5.2%5%-15",CVE_CONF," ",NOM_CONF,"
       ",QUORUM," ",DURACION," ",MOBI),nl,
    writedevice(conf_nsg),
    writef("%2%2%-20%2%-20%2%-20%5%4%8%5.2%5%-15",CVE_CONF,"
    ",TEMA_CONF," ",SUBT_CONF," ",NOM_CONF," ",QUORUM,"
    ",DURACION," ",MOBI),nl.
recursos("R",DIAS_CONG) if
    writedevice(screen),nl,
    write("DIA      HORA          NOMBRE SALON          MOBILIARIO
       "),
    write("CAPACIDAD"),nl,
    salon(DIA,CAPACIDAD,MOBI,NOM_SAL),
    DIA <= DIAS_CONG,
    h_salon(DIA,HORA_INI,DISPONIBLE,NOM_SAL,1),
    HORA_FIN = HORA_INI + DISPONIBLE,
    salon(DIA,CAPACIDAD,MOBI,NOM_SAL),
    DIA <= DIAS_CONG,
    h_salon(DIA,HORA_INI,DISPONIBLE,NOM_SAL,1),
    HORA_FIN = HORA_INI + DISPONIBLE,
    writedevice(screen),
    writef("%2%2%5.2%1%5.2%2%-20%2%-15%2%4",DIA,"
       ",HORA_INI,"-",HORA_FIN," ",NOM_SAL," ",MOBI,"
       ",CAPACIDAD),nl,
    writedevice(sal_nsg),
    writef("%2%2%5.2%1%5.2%2%-20%2%-15%2%4",DIA,"
    ",HORA_INI,"-",HORA_FIN," ",NOM_SAL," ",MOBI,"
    ",CAPACIDAD),nl.
recursos("R",_) if
```

```
    writedevice(screen),nl,write("Oprima <RETURN> o 'P' para
       imprimir "),
    readln(IMP),
    upper_lower(IMP_U,IMP),removewindow,shiftwindow(2),
    imprime("R ",I MP_U).

/* ASIGNACION DE SALONES COMPARTIDOS SIN TRASLAPE*/
asigna_st(DIAS_CONG,CVE_CONF,TOL) if
    j_salon(NOM_SAL1,NOM_SAL2,NOM_SAL3),
    salon(DIA,CAPA1,MOBI,NOM_SAL1),
    salon(DIA,CAPA2,MOBI,NOM_SAL2),
    DIA <= DIAS_CONG,
    CAPA = CAPA1 + CAPA2,
    conferencia(CVE_CONF,QUORUM,DURACION,MOBI,TEMA_CONF,
       SUBT_CONF,NOM_CONF,0),
    QUORUM > CAPA1*(1+TOL/100), QUORUM > CAPA2*(1+TOL/100),
    QUORUM <= CAPA*(1+TOL/100),
    h_salon(DIA,HORA_INI1,DISP1,NOM_SAL1,1),
    h_salon(DIA,HORA_INI2,DISP2,NOM_SAL2,1),
    conferencia(CVE_CONF,QUORUM,DURACION,MOBI,TEMA_CONF,
       SUBT_CONF,NOM_CONF,0),
    maximo(HORA_INI1,HORA_INI2,HORA_INI),
    HORA_FIN1 = HORA_INI1 + DISP1,
    HORA_FIN2 = HORA_INI2 + DISP2,
    minimo(HORA_FIN1,HORA_FIN2,MINIMO),
    MINIMO > HORA_INI,
    DURACION <= MINIMO - HORA_INI,
    HORA_FIN = HORA_INI + DURACION,
  writedevice(screen),
  writef("%2%1%5.2%1%5.2%1%2%1%-20%1%4%1%-12%1%-20",DIA,"
     ",HORA_INI,"-", HORA_FIN," ",CVE_CONF,"/",NOM_CONF,"
     ",QUORUM," ",MOBI," ",NOM_SAL3),nl,
    writedevice(arch_asigna),
    writef("%3%1%5.2%1%5.2%1%2%1%-20%1%-20%1%-20%1%4%2%-15%1%
       -20% 1%4 ",DIA, " ",HORA_INI,"-",HORA_FIN,"
       ",CVE_CONF,"/",TEMA_CONF," ",SUBT_CONF," ",
       NOM_CONF," ",QUORUM," ",MOBI," ",NOM_SAL3," ",CAPA),nl,
    retract(conferencia(CVE_CONF,QUORUM,DURACION,MOBI,
       TEMA_CONF,SUBT _CONF,NOM_CONF,0)),
    writedevice(screen),
    quita_mob(MOBI,NOM_SAL1,DIA),
    quita_mob(MOBI,NOM_SAL2,DIA),
    quita_hora(DIA,HORA_INI,HORA_FIN),
    QUITA_HORA = DURACION,
    borra_jun(DIA,HORA_INI,QUITA_HORA,NOM_SAL1),
    borra_jun(DIA,HORA_INI,QUITA_HORA,NOM_SAL2).

/* ASIGNACION DE SALONES INDIVIDUALES SIN TRASLAPE*/
asigna_st(DIAS_CONG,CVE_CONF,TOL) if
    salon(DIA,CAPACIDAD,MOBI,NOM_SAL),
    DIA <= DIAS_CONG,
    conferencia(CVE_CONF,QUORUM,DURACION,MOBI,TEMA_CONF,
    SUBT_CONF,NOM_CONF,1),
    QUORUM <= CAPACIDAD*(1+TOL/100),
    h_salon(DIA,HORA_INI,DISPONIBLE,NOM_SAL,1),
    conferencia(CVE_CONF,QUORUM,DURACION,MOBI,TEMA_CONF,
    SUBT_CONF,NOM_CONF,1).
```

```
    DURACION <= DISPONIBLE,
    HORA_FIN = HORA_INI + DURACION,
    writedevice(screen),
    writef("%2%1%5.2%1%5.2%1%2%1%-20%1%4%1%-12%1%-20",DIA,"
        ",HORA_INI,"-", HORA_FIN," ",CVE_CONF,"/",NOM_CONF,"
        ",QUORUM," ",MOBI," ",NOM_SAL),nl,
    writedevice(arch_asigna),
    writef("%3%1%5.2%1%5.2%1%2%1%-20%1%-20%1%-20%1%4%2%-15%1%
        -20% 1%4 ",DIA, " ",HORA_INI,"-",HORA_FIN,"
        ",CVE_CONF,"/",TEMA_CONF," ",SUBT_CONF," ",
        NOM_CONF," ",QUORUM," ",MOBI," ",NOM_SAL,"
        ",CAPACIDAD),nl,
    RESTA = DISPONIBLE - DURACION,
    retract(conferencia(CVE_CONF,QUORUM,DURACION,MOBI,
    TEMA_CONF,SUBT _CONF,NOM_CONF,1)),
    retract(h_salon(DIA,HORA_INI,DISPONIBLE,NOM_SAL,1)),
    asserta(h_salon(DIA,HORA_FIN,RESTA,NOM_SAL,1)),
    quita_mob(MOBI,NOM_SAL,DIA),
    quita_hora(DIA,HORA_INI,HORA_FIN).

/* ELIMINA SALONES CON DISPONIBILIDAD NULA */
asigna_st(_,_,_) if
    h_salon(DIA,HORA_INI,DISP,NOM_SAL,CVE_SAL),
    DISP = 0,
    retract(h_salon(DIA,HORA_INI,DISP,NOM_SAL,CVE_SAL)).

/* REASIGNA SALONES CON HORARIOS PARA SIGUIENTE TEMA */
asigna_st(_,_,_) if
    h_salon(DIA,HORA_INI,DISP,NOM_SAL,0),
    retract(h_salon(DIA,HORA_INI,DISP,NOM_SAL,0)),
    asserta(h_salon(DIA,HORA_INI,DISP,NOM_SAL,1)).

/* JUNTA SALONES CON HORARIOS CONTIGUOS */
asigna_st(_,_,_) if
    h_salon(DIA,HORA_INI1,DISP1,NOM_SAL,1),
    h_salon(DIA,HORA_INI2,DISP2,NOM_SAL,1),
    HORA_INI1 + DISP1 = HORA_INI2,
    DISP = DISP1 + DISP2,
    retract(h_salon(DIA,HORA_INI1,DISP1,NOM_SAL,1)),
    retract(h_salon(DIA,HORA_INI2,DISP2,NOM_SAL,1)),
    assertz(h_salon(DIA,HORA_INI1,DISP,NOM_SAL,1)).
asigna_st(_,_,_) if
    h_salon(DIA,HORA_INI1,DISP1,NOM_SAL,1),
    h_salon(DIA,HORA_INI2,DISP2,NOM_SAL,1),
    HORA_INI2 + DISP2 = HORA_INI1,
    DISP = DISP1 + DISP2,
    retract(h_salon(DIA,HORA_INI1,DISP1,NOM_SAL,1)),
    retract(h_salon(DIA,HORA_INI2,DISP2,NOM_SAL,1)),
    assertz(h_salon(DIA,HORA_INI2,DISP,NOM_SAL,1)).

/*   ACOMODO   DE   HORARIOS   DE   SALONES   POR   DISPONIBILIDAD
    ASCENDENTE */
asigna_st(DIAS_CONG,_,_) if
    h_salon(DIA,_,DISP1,NOM_SAL,1),
    DIA <= DIAS_CONG,
    h_salon(DIA,HORA_INI2,DISP2,NOM_SAL,1),
    DISP1 > DISP2,
```

```
    retract(h_salon(DIA,HORA_INI2,DISP2,NOM_SAL,1)),
    asserta(h_salon(DIA,HORA_INI2,DISP2,NOM_SAL,1)).

/* ASIGNACION DE SALONES COMPARTIDOS CON TRASLAPE*/
asigna_ct(DIAS_CONG,TOL) if
    j_salon(NOM_SAL1,NOM_SAL2,NOM_SAL3),
    salon(DIA,CAPA1,MOBI,NOM_SAL1),
    salon(DIA,CAPA2,MOBI,NOM_SAL2),
    DIA <= DIAS_CONG,
    CAPA = CAPA1 + CAPA2,
    conferencia(CVE_CONF,QUORUM,DURACION,MOBI,TEMA_CONF,
       SUBT_CONF,NOM_CONF,0),
    QUORUM > CAPA1*(1+TOL/100), QUORUM > CAPA2*(1+TOL/100),
    QUORUM <= CAPA*(1+TOL/100),
    h_salon(DIA,HORA_INI1,DISP1,NOM_SAL1,1),
    h_salon(DIA,HORA_INI2,DISP2,NOM_SAL2,1),
    conferencia(CVE_CONF,QUORUM,DURACION,MOBI,TEMA_CONF,
    SUBT_CONF,NOM_CONF,0),
    maximo(HORA_INI1,HORA_INI2,HORA_INI),
    HORA_FIN1 = HORA_INI1 + DISP1,
    HORA_FIN2 = HORA_INI2 + DISP2,
    minimo(HORA_FIN1,HORA_FIN2,MINIMO),
    MINIMO > HORA_INI,
    DURACION <= MINIMO - HORA_INI,
    HORA_FIN = HORA_INI + DURACION,
    writedevice(screen),
    writef("%2%1%5.2%1%5.2%1%2%1%-20%1%4%1%-12%1%-20",DIA,"
       ",HORA_INI,"-", HORA_FIN," ",CVE_CONF,"/",NOM_CONF,"
       ",QUORUM," ",MOBI," ",NOM_SAL3),nl,
    writedevice(arch_asigna),
    writef("%3%1%5.2%1%5.2%1%2%1%-20%1%-20%1%-20%1%4%2%-15%1%
       -20% 1%4 ",DIA, " ",HORA_INI,"-",HORA_FIN,"
       ",CVE_CONF,"/",TEMA_CONF," ",SUBT_CONF," ",
       NOM_CONF," ",QUORUM," ",MOBI," ",NOM_SAL3," ",CAPA),nl,
    retract(conferencia(CVE_CONF,QUORUM,DURACION,MOBI,
       TEMA_CONF,SUBT_CONF,NOM_CONF,0)),
    quita_mob(MOBI,NOM_SAL1,DIA),
    quita_mob(MOBI,NOM_SAL2,DIA),
    QUITA_HORA = DURACION,
    borra_jun(DIA,HORA_INI,QUITA_HORA,NOM_SAL1),
    borra_jun(DIA,HORA_INI,QUITA_HORA,NOM_SAL2).

/* ASIGNACION DE SALONES INDIVIDUALES CON TRASLAPE*/
asigna_ct(DIAS_CONG,TOL) if
    salon(DIA,CAPACIDAD,MOBI,NOM_SAL),
    DIA <= DIAS_CONG,
    conferencia(CVE_CONF,QUORUM,DURACION,MOBI,TEMA_CONF,
       SUBT_CONF,NOM_CONF,1),
    QUORUM <= CAPACIDAD*(1+TOL/100),
    h_salon(DIA,HORA_INI,DISPONIBLE,NOM_SAL,1),
    conferencia(CVE_CONF,QUORUM,DURACION,MOBI,TEMA_CONF,
    SUBT_CONF,NOM_CONF,1),
    DURACION <= DISPONIBLE,
    HORA_FIN = HORA_INI + DURACION,
    writedevice(screen),
    writef("%2%1%5.2%1%5.2%1%2%1%-20%1%4%1%-12%1%-20",DIA,"
       ",HORA_INI,"-", HORA_FIN," ",CVE_CONF,"/",NOM_CONF,"
```

```
      ",QUORUM," ",MOBI," ",NOM_SAL),nl,
   writedevice(arch_asigna),
   writef("%3%1%5.2%1%5.2%1%2%1%-20%1%-20%1%-20%1%4%2%-15%1%
      -20% 1%4 ",DIA," ",HORA_INI,"-",HORA_FIN,"
      ",CVE_CONF,"/",TEMA_CONF," ",SUBT_CONF," ",
      NOM_CONF," ",QUORUM," ",MOBI," ",NOM_SAL,"
      ",CAPACIDAD),nl,
   RESTA = DISPONIBLE - DURACION,
   retract(conferencia(CVE_CONF,QUORUM,DURACION,MOBI,
      TEMA_CONF,S UBT _CONF,NOM_CONF,1)),
   retract(h_salon(DIA,HORA_INI,DISPONIBLE,NOM_SAL,1)),
   asserta(h_salon(DIA,HORA_FIN,RESTA,NOM_SAL,1)),
   quita_mob(MOBI,NOM_SAL,DIA).

/* ASIGNACION DE SALONES COMPARTIDOS CON TRASLAPE A CUALQUIER
   CONFERENCIA*/
asigna_ct(DIAS_CONG,TOL) if
   j_salon(NOM_SAL1,NOM_SAL2,NOM_SAL3),
   salon(DIA,CAPA1,MOBI,NOM_SAL1),
   salon(DIA,CAPA2,MOBI,NOM_SAL2),
   DIA <= DIAS_CONG,
   CAPA = CAPA1 + CAPA2,
   conferencia(CVE_CONF,QUORUM,DURACION,MOBI,TEMA_CONF,
      SUBT_CONF,NOM_CONF,_),
   QUORUM > CAPA1*(1+TOL/100), QUORUM > CAPA2*(1+TOL/100),
   QUORUM <= CAPA*(1+TOL/100),
   h_salon(DIA,HORA_INI1,DISP1,NOM_SAL1,1),
   h_salon(DIA,HORA_INI2,DISP2,NOM_SAL2,1),
   conferencia(CVE_CONF,QUORUM,DURACION,MOBI,TEMA_CONF,
      SUBT_CONF,NOM_CONF,0),
   maximo(HORA_INI1,HORA_INI2,HORA_INI),
   HORA_FIN1 = HORA_INI1 + DISP1,
   HORA_FIN2 = HORA_INI2 + DISP2,
   minimo(HORA_FIN1,HORA_FIN2,MINIMO),
   MINIMO > HORA_INI,
   DURACION <= MINIMO - HORA_INI,
   HORA_FIN = HORA_INI + DURACION,
   writedevice(screen),
   writef("%2%1%5.2%1%5.2%1%2%1%-20%1%4%1%-12%1%-20",DIA,"
      ",HORA_INI,"-", HORA_FIN," ",CVE_CONF,"/",NOM_CONF,"
      ",QUORUM," ",MOBI," ",NOM_SAL3),nl,
   writedevice(arch_asigna),
   writef("%3%1%5.2%1%5.2%1%2%1%-20%1%-20%1%-20%1%4%2%-15%1%
      -20% 1%4 ",DIA," ",HORA_INI,"-",HORA_FIN,"
      ",CVE_CONF,"/",TEMA_CONF," ",SUBT_CONF," ",
      NOM_CONF," ",QUORUM," ",MOBI," ",NOM_SAL3," ",CAPA),nl,
   retract(conferencia(CVE_CONF,QUORUM,DURACION,MOBI,
      TEMA_CONF,SUBT_CONF,NOM_CONF,0)),
   quita_mob(MOBI,NOM_SAL1,DIA),
   quita_mob(MOBI,NOM_SAL2,DIA),
   QUITA_HORA = DURACION,
   borra_jun(DIA,HORA_INI,QUITA_HORA,NOM_SAL1),
   borra_jun(DIA,HORA_INI,QUITA_HORA,NOM_SAL2).

/* ELIMINA SALONES CON DISPONIBILIDAD NULA */
asigna_ct(_,_) if
   h_salon(DIA,HORA_INI,DISP,NOM_SAL,CVE_SAL),
```

```
    DISP = 0,
    retract(h_salon(DIA,HORA_INI,DISP,NOM_SAL,CVE_SAL)).

/* ELIMINA HORARIOS DE SALONES COMPARTIDOS */
borra_jun(DIA,HORA_INI,DURACION,NOM_SAL) if
    h_salon(DIA,HORA_INI,DURACION,NOM_SAL,0),
    retract(h_salon(DIA,HORA_INI,DURACION,NOM_SAL,0)).
borra_jun(_,_,_,_).

/* ELIMINA OTROS MOBILIARIOS DE UN MISMO SALON */
quita_mob(MOBI,NOM_SAL,DIA) if
    salon(DIA,CAPA,O_MOBI,NOM_SAL),
    MOBI <> O_MOBI,
    retract(salon(DIA,CAPA,O_MOBI,NOM_SAL)).
quita_mob(_,_,_).

/* ELIMINA HORARIOS TRASLAPADOS PARA UN MISMO TEMA DE SALONES
    INDIVIDUALES*/
quita_hora(DIA,HORA_INI,HORA_FIN) if
    h_salon(DIA,OHORA_INI,DISPONIBLE,NOM_SAL,1),
    DISPONIBLE <>0, OHORA_FIN = OHORA_INI + DISPONIBLE,
    OHORA_INI < HORA_INI, OHORA_FIN > HORA_INI,  OHORA_FIN  <=
    HORA_FIN,
    USA = HORA_INI - OHORA_INI,
    NOUSA = OHORA_FIN - HORA_INI,
    retract(h_salon(DIA,OHORA_INI,DISPONIBLE,NOM_SAL,1)),
    asserta(h_salon(DIA,OHORA_INI,USA,NOM_SAL,1)),
    asserta(h_salon(DIA,HORA_INI,NOUSA,NOM_SAL,0)).
quita_hora(DIA,HORA_INI,HORA_FIN) if
    h_salon(DIA,OHORA_INI,DISPONIBLE,NOM_SAL,1),
    DISPONIBLE <>0, OHORA_FIN = OHORA_INI + DISPONIBLE,
    OHORA_INI < HORA_INI, OHORA_FIN > HORA_FIN,
    USA1 = HORA_INI - OHORA_INI,
    USA2 = OHORA_FIN - HORA_FIN,
    NOUSA = HORA_FIN - HORA_INI,
    retract(h_salon(DIA,OHORA_INI,DISPONIBLE,NOM_SAL,1)),
    asserta(h_salon(DIA,OHORA_INI,USA1,NOM_SAL,1)),
    asserta(h_salon(DIA,HORA_FIN,USA2,NOM_SAL,1)),
    asserta(h_salon(DIA,HORA_INI,NOUSA,NOM_SAL,0)).
quita_hora(DIA,HORA_INI,HORA_FIN) if
    h_salon(DIA,OHORA_INI,DISPONIBLE,NOM_SAL,1),
    DISPONIBLE <>0, OHORA_FIN = OHORA_INI + DISPONIBLE,
    OHORA_INI = HORA_INI, OHORA_FIN <= HORA_FIN,
    retract(h_salon(DIA,OHORA_INI,DISPONIBLE,NOM_SAL,1)),
    asserta(h_salon(DIA,OHORA_INI,DISPONIBLE,NOM_SAL,0)).
quita_hora(DIA,HORA_INI,HORA_FIN) if
    h_salon(DIA,OHORA_INI,DISPONIBLE,NOM_SAL,1),
    DISPONIBLE <>0, OHORA_FIN = OHORA_INI + DISPONIBLE,
    OHORA_INI = HORA_INI, OHORA_FIN > HORA_FIN,
    USA = OHORA_FIN - HORA_FIN,
    NOUSA = HORA_FIN - OHORA_INI,
    retract(h_salon(DIA,OHORA_INI,DISPONIBLE,NOM_SAL,1)),
    asserta(h_salon(DIA,HORA_FIN,USA,NOM_SAL,1)),
    asserta(h_salon(DIA,OHORA_INI,NOUSA,NOM_SAL,0)).
quita_hora(DIA,HORA_INI,HORA_FIN) if
    h_salon(DIA,OHORA_INI,DISPONIBLE,NOM_SAL,1),
    DISPONIBLE <>0, OHORA_FIN = OHORA_INI + DISPONIBLE,
```

```
    OHORA_INI > HORA_INI, OHORA_FIN <= HORA_FIN,
    retract(h_salon(DIA,OHORA_INI,DISPONIBLE,NOM_SAL,1)),
    asserta(h_salon(DIA,OHORA_INI,DISPONIBLE,NOM_SAL,0)).
quita_hora(DIA,HORA_INI,HORA_FIN) if
    h_salon(DIA,OHORA_INI,DISPONIBLE,NOM_SAL,1),
    DISPONIBLE <>0, OHORA_FIN = OHORA_INI + DISPONIBLE,
    OHORA_INI  >  HORA_INI,  OHORA_INI < HORA_FIN, OHORA_FIN >
    HORA_FIN,
    USA = OHORA_FIN - HORA_FIN,
    NOUSA = HORA_FIN - OHORA_INI,
    retract(h_salon(DIA,OHORA_INI,DISPONIBLE,NOM_SAL,1)),
    asserta(h_salon(DIA,HORA_FIN,USA,NOM_SAL,1)),
    asserta(h_salon(DIA,OHORA_INI,NOUSA,NOM_SAL,0)).
quita_hora(_,_,_).

/* ENCUENTRA EL MINIMO DE DOS CANTIDADES */
minimo(DATO1,DATO2,MINIMO) if
    DATO1 <= DATO2,
    MINIMO = DATO1.
minimo(DATO1,DATO2,MINIMO) if
    DATO2 < DATO1,
    MINIMO = DATO2.

/* ENCUENTRA EL MAXIMO DE DOS CANTIDADES */
maximo(DATO1,DATO2,MAXIMO) if
    DATO1 >= DATO2,
    MAXIMO = DATO1.
maximo(DATO1,DATO2,MAXIMO) if
    DATO2 > DATO1,
    MAXIMO = DATO2.

/* IMPRESION DE RECURSOS */
imprime("A",IMP) if
   closefile(arch_asigna),
   IMP = "P",
   system("IMP-ASG").
imprime("R",IMP) if
   closefile(conf_nsg),
   closefile(sal_nsg),
   IMP = "P",
   system("IMP-NASG").
```

Los tres programas de la interface hombre-máquina son:

1) CAPTURA.PAS : carga de datos al sistema.
2) TRACON.PAS : conversión de datos de conferencias a predicados en sintaxis de prolog.
3) TRASAL.PAS : conversión de datos de salones a predicados en sintaxis de prolog.

```
PROGRAM CAPTURA ; (*INPUT/OUTPUT*)
CONST
LIMTEM : INTEGER= 100;
LIMSTE : INTEGER= 300;
LIMDEL : INTEGER= 1000;
LIMSAL : INTEGER= 50;
LINCON : INTEGER= 200;
LIMUSA : INTEGER= 25;
TYPE
ST12   = STRING[12];
ST8    = STRING[8];
ST10   = STRING[10];
ST50   = STRING[50];
ST35   = STRING[35];
ST2    = STRING[2];
ST15   = STRING[15];
ST20   = STRING[20];
ENT    = INTEGER;
BOL    = BOOLEAN;
REA    = REAL;
CAR    = CHAR;
ARRE10= ARRAY[1..10] OF REAL;
ARRE4  = ARRAY[1..4]  OF INTEGER;
ARRE2  = ARRAY[1..2] OF INTEGER;
CONFER = RECORD
         CVECON : ENT;
          QUORUM : ENT;
          DURACIO : REA;
          MOBIL_C : ST10;
          TEMA    : ENT;
          SUBTEMA: ENT;
          NOMBRE : ST50;
          END;
NOMTEM = RECORD
          NUMTEM : ENT;
          NOMTEMA   : ST50;
          END;
NOMSTE = RECORD
          NUMSTE : ENT;
          NOMSUB : ST50;
          END;
SALON  = RECORD
          S_NOM  : ST15;
          S_MOB  : ST15;
          S_CAP  : ENT;
          S_UIN  : ARRE10;
          S_DIS  : ARRE10;
          S_TMO  : ARRE4;
          S_DIR  : ENT;
```

```
            END;
UNESAL = RECORD
            USAL1  : ENT;
            USAL1N : ST50;
            USAL2  : ENT;
            USAL2N : ST50;
            USALI  : ST50;
            END;
VAR
   DDUR  : REA;
   BAND1 : BOL;
   OPCI  : CAR;
   ARUSA : FILE OF UNESAL;
   VARUSA: UNESAL;
   ARSAL : FILE OF SALON;
   VARSAL: SALON;
   ARCON : FILE OF CONFER;
   VARCON : CONFER;
   ARTEM  : FILE OF NOMTEM;
   VARTEM : NOMTEM;
   ARSTE  : FILE OF NOMSTE;
   VARSTE : NOMSTE;
FUNCTION EXICON(NOMARC : ST35):BOOLEAN;
VAR
   FIL : FILE;
BEGIN
   ASSIGN(FIL,NOMARC);
   (*$I-*) RESET(FIL); (*$I+*)
   EXICON:=(IORESULT=0);
   CLOSE (FIL);
END;
PROCEDURE PINCUA;
VAR
   CON1,CON2  : ENT;
BEGIN
  TEXTCOLOR(1);
   GOTOXY(1,1);
   WRITE('┌──────────────────────────────────────────────────────────────────────────────');
   WRITE('──────────────────────────────┐');
   CON1:=2;
   CON2:=1;
   WHILE CON1<25 DO
   BEGIN
      CON2:=1;
      GOTOXY(CON2,CON1); WRITE('│');
      CON2:=79;
      GOTOXY(CON2,CON1); WRITE('│');
      CON1:=CON1+1;
   END;
   GOTOXY(1,25);
   WRITE('└──────────────────────────────────────────────────────────────────────────────');
   WRITE('──────────────────────────────');GOTOXY(79,25); WRITE('┘');
END;
PROCEDURE BCP(X,Y,L : ENT);
BEGIN
   WHILE L>0 DO
   BEGIN
```

```
        GOTOXY(X,Y);
        WRITE(' ');
        X:=X+1;
        L:=L-1;
    END;
END;
PROCEDURE PANGRAL;
BEGIN
     PINCUA;
    TEXTCOLOR(4);
    GOTOXY(4,8);   WRITE('Conferencias');
    GOTOXY(30,8);  WRITE('Salones');
    GOTOXY(46,8);  WRITE('Finalizar');
    GOTOXY(4,14);  WRITE('CONFERENCIAS:');
    GOTOXY(4,16);  WRITE('Alta');
    GOTOXY(10,16);WRITE('Baja  Cambio  Listar');
    GOTOXY(31,16);WRITE('Temas  Subtemas Finalizar');
    GOTOXY(4,20);  WRITE('SALONES:');
    GOTOXY(4,22);  WRITE('Alta');
    GOTOXY(10,22);WRITE('Baja  Cambio  Listar');
    GOTOXY(31,22);WRITE('Unir salones  Finalizar');
END;
PROCEDURE BTL(X,Y,L,NL  : ENT);
BEGIN
    WHILE NL>0 DO
    BEGIN
        BCP(X,Y,L);
        NL:=NL-1;
        Y:=Y+1;
    END;
END;
PROCEDURE TIPMOB;
BEGIN
    GOTOXY(4,15);WRITE('Tipos de mobiliario :');
    GOTOXY(4,16);WRITE('1) escuela 2) auditorio');
    GOTOXY(4,17);WRITE('3) comite  4) alimentos');
END;
PROCEDURE PANALC;
BEGIN
    PINCUA;  TEXTCOLOR(4);
    GOTOXY(4,4);    WRITE('CLAVE DE CONFERENCIA: ');
    GOTOXY(4,5);    WRITE('Nombre: ');
    GOTOXY(4,7);    WRITE('clave de Tema: ');
    GOTOXY(4,9);    WRITE('clave de Subtema: ');
    GOTOXY(4,11);   WRITE('Duracion: ');
    GOTOXY(4,14);   TIPMOB;
    GOTOXY(4,18);   WRITE('Mobiliario: ');
    GOTOXY(4,22);   WRITE('Asistencia ');
END;
PROCEDURE DECHOR(VAR HORA :REAL);
BEGIN
    HORA:=FRAC(HORA)*0.6+INT(HORA);
END;
PROCEDURE PANDC(VARCON : CONFER);
VAR
    DDUR : REA;
BEGIN
```

```
      WITH VARCON DO
      BEGIN
         GOTOXY(30,5); WRITE(NOMBRE);
         GOTOXY(30,7); WRITE(TEMA);
         GOTOXY(30,9); WRITE(SUBTEMA);
         DDUR:=DURACIO;
         DECHOR(DDUR);
         GOTOXY(30,11); WRITE(DDUR:5:2);
         GOTOXY(30,16); WRITE(MOBIL_C);
         GOTOXY(30,22); WRITE(QUORUM);
      END;
   END;
   PROCEDURE HORDEC(VAR HORA : REAL);
   BEGIN
      HORA:=(FRAC(HORA)/0.6)+INT(HORA);
   END;
   PROCEDURE AVISO;
   BEGIN
      CLRSCR;
      PINCUA;
      TEXTCOLOR(3);
      GOTOXY(10,8);
      WRITE('PARA ESTE PROCESO NECESITA PREVIAMENTE EL CONOCER');
      GOTOXY(10,9);
      WRITE('LA DIRECCION DE LA CONFERENCIA DESEADA.SI AUN NO-');
      GOTOXY(10,10);
      WRITE('LA TIENE PIDA UN LISTADO DE ELLAS CON LA OPCION L');
      GOTOXY(10,11);
      WRITE('DE ESTE MENU.');
   END;
   PROCEDURE APEEXC(VAR DIR  : ENT);
   BEGIN
      ASSIGN(ARCON,'CONFER.DAT');
      RESET(ARCON);
      DIR:=0;
   END;
   PROCEDURE APENEXC(VAR DIR: ENT);
   BEGIN
      ASSIGN(ARCON,'CONFER.DAT');
      REWRITE(ARCON);
      DIR:=0;
      WITH VARCON DO
      BEGIN
         TEMA:=0;
         SUBTEMA:=0;
         QUORUM:=0;
         MOBIL_C:=' ';
         DURACIO:=0;
         NOMBRE:=' ';
         CVECON:=0;
         WHILE DIR<LIMCON DO
         BEGIN
            SEEK (ARCON,DIR);
            WRITE (ARCON,VARCON);
            DIR:=DIR+1;
         END;
      END;
```

```
END;
PROCEDURE APENCO;
BEGIN
   ASSIGN(ARTEM,'TEMA.DAT');
   RESET(ARTEM);
   ASSIGN(ARSTE,'SUBTEMA.DAT');
   RESET(ARSTE);
END;
PROCEDURE ACEENT(MEMBRETE :ST35; VAR VARENT :ENT);
VAR
 OK   :  BOL;
BEGIN
   OK:=FALSE;
   WHILE NOT OK DO
   BEGIN
      WRITE (MEMBRETE);
      (*$I-*) READ(VARENT);(*$I+*)
      OK:=(IORESULT=0);
   END;
END;
PROCEDURE ACEREA(MEMBRETE : ST35; VAR VARREA  : REA);
VAR
OK   :  BOL;
BEGIN
   OK:=FALSE;
   WHILE NOT OK DO
   BEGIN
      WRITE (MEMBRETE);
      (*$I-*) READ (VARREA); (*$I+*)
      OK:=(IORESULT=0);
   END;
END;
PROCEDURE ACECAR (MEMBRETE : ST35; VAR OPCI   : CHAR);
VAR
   BANSIG   : BOL;
BEGIN
   BANSIG:=FALSE;
   WHILE NOT BANSIG DO
   BEGIN
      WRITE (MEMBRETE);
      (*$I+*) READ(OPCI); (*$I-*)
      BANSIG:=(IORESULT=0);
      IF LENGTH(OPCI) < 1 THEN BANSIG:=FALSE;
   END;
END;
PROCEDURE ACENTC(X,Y,L :ENT; VAR VARENT :ENT);
VAR
 OK   :  BOL;
BEGIN
   OK:=FALSE;
   WHILE NOT OK DO
   BEGIN
      BCP(X,Y,L);
      GOTOXY(X,Y);
      (*$I-*) READ(VARENT);(*$I+*)
      OK:=(IORESULT=0);
   END;
```

```
END;
PROCEDURE ACEREC(X,Y,L: ENT; VAR VARREA  : REA);
VAR
OK   :  BOL;
BEGIN
   OK:=FALSE;
   WHILE NOT OK DO
   BEGIN
      BCP(X,Y,L);
      GOTOXY(X,Y);
      (*$I-*) READ (VARREA); (*$I+*)
      OK:=(IORESULT=0);
   END;
END;
PROCEDURE ACECAC (X,Y,L: ENT; VAR OPCI   : CHAR);
VAR
   BANSIG   : BOL;
BEGIN
   BANSIG:=FALSE;
   WHILE NOT BANSIG DO
   BEGIN
      BCP(X,Y,L);
      GOTOXY(X,Y);
      (*$I+*) READ(OPCI); (*$I-*)
      BANSIG:=(IORESULT=0);
      IF LENGTH(OPCI) < 1 THEN BANSIG:=FALSE;
   END;
END;
PROCEDURE C_TEMA(VAR  VARCON : CONFER);
VAR
BAN2     :  BOL;
RESP     : ST2;
DTEM     : ENT;
BEGIN
   WITH VARTEM,VARCON DO
   BEGIN
         BAN2:=TRUE;
         WHILE BAN2 DO
         BEGIN
            GOTOXY(30,7); ACEENT(' ',TEMA);
            IF (TEMA<LIMTEM) AND (TEMA>=0) THEN
            BEGIN
               SEEK(ARTEM,TEMA);
               READ(ARTEM,VARTEM);
               IF NUMTEM>0 THEN
               BEGIN
                  GOTOXY(35,7); WRITE(NOMTEMA);
                  GOTOXY(35,8);WRITE('¿ES CORRECTO?[NO/RET]=>');
                  READLN (RESP);
                  IF RESP<> 'NO' THEN BAN2:=FALSE
                  ELSE
                  BEGIN
                     BCP(30,7,5);
                     BCP(30,8,30);
                  END;
               END
               ELSE
```

```
                BEGIN
                   GOTOXY(50,8);WRITE('TEMA NO DADO DE ALTA');
                   DELAY(LIMDEL);
                   GOTOXY(50,8);WRITE('                    ');
                   BCP(30,7,5);
                   BCP(30,8,30);
                END
             END
             ELSE
             BEGIN
                GOTOXY(50,8); WRITE('TEMA FUERA DE RANGO');
                DELAY(LIMDEL);
                GOTOXY(50,8); WRITE('                    ');
                BCP(30,7,5);
                BCP(30,8,30);
             END;
          END;
    END;
END;
PROCEDURE C_STEM(VAR VARCON : CONFER);
VAR
   BAN2:BOL;
   DSTE : ENT;
   RESP : ST2;
BEGIN
   WITH VARSTE,VARCON DO
   BEGIN
      BAN2:=TRUE;
         WHILE BAN2 DO
         BEGIN
            GOTOXY(30,9); ACCENT(' ',SUBTEMA);
            IF (SUBTEMA<LIMSTE) AND (SUBTEMA>=0) THEN
            BEGIN
               SEEK(ARSTE,SUBTEMA);
               READ(ARSTE,VARSTE);
               IF SUBTEMA>0 THEN
               BEGIN
                  GOTOXY(35,9); WRITE (NOMSUB);
                  GOTOXY(35,10);WRITE('¿ES CORRECTO?[SI/NO]=>');
                  READLN(RESP);
                  IF RESP<>'NO' THEN BAN2:=FALSE
                  ELSE
                  BEGIN
                     BCP(30,9,30);
                     BCP(30,10,30);
                  END;
               END
               ELSE
               BEGIN
                  GOTOXY(50,10); WRITE('SUBTEMA NO DADO DE ALTA');
                  DELAY(LIMDEL);
                  GOTOXY(50,10); WRITE('                    ');
               END;
            END
            ELSE
            BEGIN
                 GOTOXY(50,10);WRITE('SUBTEMA FUERA DE RANGO');
```

```
                    DELAY(LIMDEL);
                END; GOTOXY(50,10):WRITE('                    ');
            END;
    END;
END;
PROCEDURE C_MOBI(VAR VARCON : CONFER);
VAR
AUX    : ENT;
BAN3   : BOL;
BEGIN
    WITH VARCON DO
    BEGIN
    BAN3:=TRUE;
        WHILE BAN3 DO
        BEGIN
            ACENTC(30,18,5,AUX);
            BAN3:=FALSE;
            CASE AUX OF
                1 : MOBIL_C:='ESCUELA';
                2 : MOBIL_C:='AUDITORIO';
                3 : MOBIL_C:='COMITE';
                4 : MOBIL_C:='ALIMENTOS';
                ELSE
                BEGIN
                    WRITE ('ERROR EN CODIGO');
                    BAN3:=TRUE;
                    DELAY(LIMDEL);
                END;
            END;
            BOP(30,16,15);
            GOTOXY(30,16); WRITE(MOBIL_C);
        END;
    END;
END;
PROCEDURE CAMCON;
VAR
RESP             : ST2;
BAN1,BAN2,BAN3   : BOL;
CARA             : CAR;
DIR1             : ENT;
BEGIN
  AVISO: GOTOXY(10,15);
  WRITE('DESEA CONTINUAR CON EL PROCESO DE CAMBIO NO/RET=>');
  READ (RESP);
  BAN3:=TRUE;
  WHILE BAN3 DO
  BEGIN
   IF RESP <>'NO' THEN
   BEGIN
       IF EXICON('CONFER.DAT') THEN
       BEGIN
           APEEXC(DIR1);
           APENCO;
           BAN1:=FALSE;
           WITH VARCON DO
           BEGIN
```

```
      CLRSCR;
      PANALC;
      GOTOXY(4,4); WRITE('CLAVE DE CONFERENCIA=>');
      WHILE NOT BAN1 DO
      BEGIN
         ACENTC (30,4,5,DIR1);
         IF (DIR1>0) AND (DIR1<LIMCON)  THEN
         BEGIN
         (*$I-*) SEEK(ARCON,DIR1);
         READ(ARCON,VARCON); (*$I+*)
         IF DURACION=0 THEN
         BEGIN
           GOTOXY(50,4); WRITE('NO DADA DE ALTA');
           DELAY(LIMDEL);
           BCP(50,4,15);
         END
         ELSE
            BAN1:=TRUE;
         END
         ELSE
         BEGIN
            GOTOXY(50,4); WRITE('CVE FUERA DE RANGO');
            DELAY(LIMDEL); BCP(50,14,20);
         END;
      END;
         PANDC(VARCON);
         BAN2:=TRUE;
         WHILE BAN2 DO
         BEGIN
            GOTOXY(50,22);WRITELN ('[ F ]  FIN');
            GOTOXY(30,24);ACECAR ('TECLEE PRIMERA LETRA=>',CARA);
            CASE CARA OF
               'N'   :  BEGIN
                           BCP(30,5,50);
                           GOTOXY(30,5);READ(NOMBRE);
                        END;
               'T'   :  C_TEMA(VARCON);
               'S'   :  C_STEM(VARCON);
               'F'   :  BAN2:=FALSE;
               'A'   :  ACENTC(30,22,8,QUORUM);
               'D'   :  BEGIN
                           ACEREC(30,11,10,DURACION);
                           HORDEC(DURACION);
                        END;
               'M'   :  C_MOBI(VARCON);
               ELSE
                        BEGIN
                           GOTOXY(50,24);WRITE('CLAVE ERRONEA');
                           DELAY (LIMDEL);BCP(50,24,18);
                        END;
            END;
            SEEK(ARCON,DIR1);
            WRITE(ARCON,VARCON);
         END;
      END;
END
ELSE
```

```
      BEGIN
          WRITELN ('ARCHIVO NO EXISTE');
          DELAY (LIMDEL);
      END;
     GOTOXY(20,24);
     WRITE ('OTRO MOVIMIENTO DE CAMBIO?[NO/RET]=>');
     READ(RESP);
     IF RESP='NO' THEN BAN3:=FALSE;
      CLOSE(ARTEM);
      CLOSE(ARSTE);
      CLOSE(ARCON);
      END
      ELSE
         BAN3:=FALSE;
      END;
      CLRSCR;
END;
PROCEDURE CRETEM;
VAR
CONT : ENT;
BEGIN
    ASSIGN(ARTEM,'TEMA.DAT');
    REWRITE(ARTEM);
    CONT:=0;
    WHILE CONT<LIMTEM DO
    BEGIN
       WITH VARTEM DO
       BEGIN
          NUMTEM:=0;
          NOMTEMA:='  ';
          SEEK(ARTEM,CONT);
          WRITE(ARTEM,VARTEM);
          CONT:=CONT+1;
       END;
    END;
    CLOSE(ARTEM);
END;
PROCEDURE CRESTE;
VAR
CONT : ENT;
BEGIN
    ASSIGN(ARSTE,'SUBTEMA.DAT');
    REWRITE(ARSTE);
    CONT:=0;
    WHILE CONT<LIMSTE DO
    BEGIN
       WITH VARSTE DO
       BEGIN
          NUMSTE:=0;
          NOMSUB:='  ';
          SEEK(ARSTE,CONT);
          WRITE(ARSTE,VARSTE);
          CONT:=CONT+1;
       END;
    END;
    CLOSE(ARSTE);
END;
```

```
PROCEDURE CAPTEM;
VAR
   DIR          : ENT;
   BAN2,
   BAN1,BAND    : BOL;
   RESP         :ST2;
BEGIN
   ASSIGN(ARTEM,'TEMA.DAT');
   RESET (ARTEM);
   BAND:=TRUE;
   BAN2:=TRUE;
   WHILE BAND DO
   BEGIN
      WITH VARTEM DO
      BEGIN
         BAN1:=TRUE;
         CLRSCR;
         WHILE BAN1 DO
         BEGIN
            BAN2:=TRUE;  PINCUA;   TEXTCOLOR(4);
            GOTOXY(15,8); WRITE('NO. DE TEMA=>');
            ACENTC(28,8,10,DIR);
            IF (DIR<LIMTEM) AND (DIR>0) THEN
            BEGIN
            SEEK(ARTEM,DIR);
            READ(ARTEM,VARTEM);
            IF NUMTEM=0 THEN
            BAN1:=FALSE
            ELSE
            BEGIN
               GOTOXY(30,15);WRITE('TEMA YA EXISTENTE');
               DELAY(LIMDEL); BCP(30,15,30);
               GOTOXY(30,13);WRITE('NOMBRE=>',NOMTEMA);
               GOTOXY(15,11);WRITE('DESEA DARLO DE BAJA[SI/RET]=>');
               READ(RESP);
               IF RESP='SI'THEN
               BEGIN
                  NUMTEM:=0;
                  NOMTEMA:='          ';
                  SEEK(ARTEM,DIR);
                  WRITE(ARTEM,VARTEM);
                  BAN2:=FALSE;
               END;
               BCP(15,11,35);
               BCP(28,8,5);
               BCP(30,13,50);
             END;
           END;
         END;
         IF BAN2 THEN
         BEGIN
         GOTOXY(15,16); WRITE ('DE UD. EL NOMBRE=>');
         READ (NOMTEMA);
         NUMTEM:=DIR;
         SEEK (ARTEM,NUMTEM);
         WRITE(ARTEM,VARTEM);
         END
```

```
      END;
      READLN(RESP);WRITE ( 'OTRO MOVIMIENTO[NO/RET]=>');
      IF RESP='NO' THEN BAND:=FALSE;
      BCP(28,8,5);
      BCP(15,16,49);
   END;
   CLRSCR;
   CLOSE(ARTEM);
END;
PROCEDURE CAPSTE;
VAR
   DIR          : ENT;
   BAN2,
   BAN1,BAND    : BOL;
   RESP         :ST2;
BEGIN
   ASSIGN(ARSTE,'SUBTEMA.DAT');
   RESET (ARSTE);
   BAND:=TRUE;
   CLRSCR;
   WHILE BAND DO
   BEGIN
      WITH VARSTE DO
      BEGIN
         BAN1:=TRUE;
         WHILE BAN1 DO
         BEGIN
            PINCUA; TEXTCOLOR(3);
            GOTOXY(15,8); WRITE('NO. DE SUBTEMA=>');
            ACENTC(32,8,5,DIR);
            IF DIR<LIMSTE THEN
            BEGIN
            SEEK(ARSTE,DIR);
            READ(ARSTE,VARSTE);
            IF NUMSTE=0 THEN
            BEGIN
            BAN1:=FALSE ;
            NUMSTE:=DIR;
            END
            ELSE
            BEGIN
               GOTOXY(30,15);WRITE('SUBTEMA YA EXISTENTE');
               DELAY(LIMDEL); BCP(30,15,30);
               GOTOXY(30,13); WRITE('NOMBRE=>',NOMSUB);
               GOTOXY(15,11);WRITE('DESEA DARLO DE BAJA[SI/RET]=>');
               READ(RESP);
               IF RESP='SI'THEN
               BEGIN
                  NUMSTE:=0;
                  NOMSUB:='          ';
                  SEEK(ARSTE,DIR);
                  WRITE(ARSTE,VARSTE);
                  BAN2:=FALSE;
               END;
               BCP(15,11,35);
               BCP(33,8,5);
```

```
                BCP(30,13,50);
               END;
             END;
           END;
          GOTOXY(15,16); WRITE ('DE UD. EL NOMBRE=>');
          READ(NOMSUB);
          SEEK (ARSTE,NUMSTE);
          WRITE(ARSTE,VARSTE);
       GOTOXY(15,20); WRITE ( 'OTRO MOVIMIENTO[NO/RET]=>');
       READ(RESP);
       IF RESP='NO' THEN BAND:=FALSE;
       BCP(15,20,40);
       BCP(33,20,50);
       BCP(32,8,5);
       END;
   END;
   CLRSCR;
   CLOSE(ARSTE);
END;
PROCEDURE TEMA;
BEGIN
   IF EXICON('TEMA.DAT') THEN
      CAPTEM
   ELSE
   BEGIN
      CRETEM;
      CAPTEM;
   END;
END;
PROCEDURE SUBTEM;
BEGIN
   IF EXICON('SUBTEMA.DAT') THEN
      CAPSTE
   ELSE
   BEGIN
      CRESTE;
      CAPSTE;
   END;
END;
PROCEDURE APECTS;
BEGIN
   ASSIGN(ARCON,'CONFER.DAT');
   RESET(ARCON);
   ASSIGN(ARTEM,'TEMA.DAT');
   RESET(ARTEM);
   ASSIGN(ARSTE,'SUBTEMA.DAT');
   RESET(ARSTE);
END;
PROCEDURE PANLIS;
BEGIN
    CLRSCR; PINCUA; TEXTCOLOR(3);
    GOTOXY(4,1);   WRITE('CLAVE');
    GOTOXY(10,1); WRITE('Nombre de conferencia');
    GOTOXY(60,1); WRITE('Tema');
    GOTOXY(66,1); WRITE('Subtema');
    GOTOXY(10,2); WRITE('Duracion');
    GOTOXY(25,2); WRITE('Asistencia');
```

```
    GOTOXY(40,2); WRITE('Mobiliario');
END;
PROCEDURE LISCON;
VAR
X,Y,
CONTE,
ANTEM       : ENT;
RESP,NADA        : ST2;
BAN1,BAN2   : BOL;
BEGIN
   APECTS;
   WITH VARCON,VARTEM,VARSTE DO
   BEGIN
   CLRSCR;
   PANLIS;
   BAN2:=TRUE;
   WHILE BAN2 DO
   BEGIN
      X:=4;
      Y:=4;
      CONTE:=0;
      WHILE NOT EOF(ARCON) DO
      BEGIN
         READ (ARCON,VARCON);
         CONTE:=CONTE+1;
         IF (QUORUM>0) AND (TEMA<LIMTEM) AND (SUBTEMA<LIMSTE) THEN
         BEGIN
            SEEK (ARTEM,TEMA);
            READ (ARTEM,VARTEM);
            SEEK (ARSTE,SUBTEMA);
            READ (ARSTE,VARSTE);
            DDUR:=DURACIO;
            DECHOR(DDUR);
            GOTOXY(4,Y);WRITE(CVECON);
            GOTOXY(10,Y);WRITE(NOMBRE);
            GOTOXY(60,Y);WRITE(NUMTEM);
            GOTOXY(66,Y);WRITE(NUMSTE);
            Y:=Y+1;
            GOTOXY(10,Y);WRITE(DDUR:5:2);
            GOTOXY(25,Y);WRITE(QUORUM);
            GOTOXY(40,Y);WRITE(MOBIL_C);
            Y:=Y+1;
            IF Y>23 THEN
            BEGIN
                GOTOXY(30,24); WRITE('PARA CONTINUAR TECLEE RETURN');
                READ(NADA);
                CLRSCR;
                PANLIS;
                Y:=4;
            END;
         END;
      END;
      BAN2:=FALSE;
   END;
   END;
   CLOSE(ARCON);
   CLOSE(ARTEM);
```

```
   CLOSE(ARSTE);
   CLRSCR;
END;
PROCEDURE BAJCON;
VAR
RESP            : ST2;
BAN1,BAN2,BAN3  : BOL;
CARA            : CAR;
DIR1            : ENT;
BEGIN
  AVISO;
  WRITE('DESEA CONTINUAR CON EL PROCESO DE BAJA NO/RET=>');
  READ (RESP);
  BAN3:=TRUE;
  WHILE BAN3 DO
  BEGIN
   IF RESP <>'NO' THEN
   BEGIN
      IF EXICON('CONFER.DAT') THEN
      BEGIN
         APEEXC(DIR1);
         APENCO;
         BAN1:=FALSE;
         WITH VARCON DO
         BEGIN
         CLRSCR;
         PANALC;
         GOTOXY(4,4); WRITE('CLAVE DE CONFERENCIA=>');
         WHILE NOT BAN1 DO
         BEGIN
            ACENTC (30,4,5,DIR1);
            (*$I-*) SEEK(ARCON,DIR1);
            READ(ARCON,VARCON); (*$I+*)
            IF DURACIO=0 THEN
            BEGIN
              GOTOXY(50,4); WRITE('NO DADA DE ALTA');
              DELAY(LIMDEL);
              BCP(50,4,15);
            END
            ELSE
               BAN1:=TRUE;
         END;
            PANDC(VARCON);
            GOTOXY(30,24); WRITE(' ES ESTA LA QUE DESEA BORRAR SI/RET>'
            READ(RESP);
            IF RESP='SI' THEN
            BEGIN
                  BTL(30,2,49,23);
                  CVECON:=0;
                  TEMA:=0;
                  SUBTEMA:=0;
                  QUORUM:=0;
                  MOBIL_C:=' ';
                  DURACIO:=0;
                  NOMBRE:=' ';
                  SEEK(ARCON,DIR1);
                  WRITE(ARCON,VARCON);
```

```
            END;
        END;
      END
      ELSE
      BEGIN
          WRITELN ('ARCHIVO NO EXISTE');
          DELAY (LIMDEL);
      END;
      BCP(30,24,40);
     GOTOXY(30,24); WRITE ('OTRO MOVIMIENTO DE BAJA?[SI/NO]=>');
     READ(RESP);
     IF RESP='NO' THEN BAN3:=FALSE;
      CLOSE(ARTEM);
      CLOSE(ARSTE);
      CLOSE(ARCON);
      END
      ELSE
         BAN3:=FALSE;
      END;
      CLRSCR;
END;
PROCEDURE ALTCON;
VAR
DIRA ,
DIR,DTEM,DSTE,AUX   : ENT;
BAN1,BAN2,BAN3,BAN4:BOL;
RESP                : ST2;
BEGIN
    DIR:=0;
    IF  EXICON('TEMA.DAT')=FALSE THEN CRETEM;
    IF  EXICON('SUBTEMA.DAT')=FALSE THEN CRESTE;
    IF EXICON('CONFER.DAT') THEN
       APEEXC(DIR)
    ELSE
       APENEXC(DIR);
    CLOSE(ARCON);
    APENCO;
    BAN1:=TRUE;
    CLRSCR;
    PANALC;
    WHILE BAN1 DO
    BEGIN
       WITH VARCON,VARTEM,VARSTE DO
       BEGIN
          APEEXC(DIR);
          BAN4:=TRUE;
          WHILE BAN4 DO
          BEGIN
             ACENTC(30,4,5,CVECON);
             DIRA:=CVECON;
             IF (CVECON>=0) AND (CVECON<=LIMCON) THEN
             BEGIN
                SEEK(ARCON,CVECON);
                READ(ARCON,VARCON);
                CVECON:=DIRA;
                IF QUORUM=0 THEN
                     BAN4:=FALSE
```

```
            ELSE
            BEGIN
               GOTOXY(50,4); WRITE('DADO DE ALTA');
               DELAY(LIMDEL);
               GOTOXY(50,4); WRITE('              ');
               GOTOXY(30,4); WRITE('      ');
            END
         END
         ELSE
         BEGIN
           GOTOXY(50,4); WRITE('CVE FUERA DE RANGO');
           DELAY(LIMDEL);
           GOTOXY(50,4); WRITE('                    ');
           GOTOXY(30,4); WRITE('      ');
         END;
      END;
   IF DIRA>0 THEN
   BEGIN
     GOTOXY(30,5); READ(NOMBRE);
     BAN2:=TRUE;
      WHILE BAN2 DO
      BEGIN
         ACENTC(30,7,5,TEMA);
         IF (TEMA<LIMTEM) AND (TEMA>0) THEN
         BEGIN
            SEEK(ARTEM,TEMA);
            READ(ARTEM,VARTEM);
            IF NUMTEM>=0 THEN
            BEGIN
               GOTOXY(35,7); WRITE(NOMTEMA);
               GOTOXY(35,8);WRITE('¿ES CORRECTO?[NO/RET]=>');
               READ(RESP);
               IF RESP<> 'NO' THEN BAN2:=FALSE
               ELSE
               BEGIN
                  BCP(30,7,5);
                  BCP(30,8,30);
               END;
            END
            ELSE
            BEGIN
               GOTOXY(50,8);WRITE('TEMA NO DADO DE ALTA');
               DELAY(LIMDEL);
               GOTOXY(50,8);WRITE('                      ');
               BCP(30,7,5);
               BCP(30,8,30);
            END
         END
         ELSE
         BEGIN
            GOTOXY(50,8); WRITE('TEMA FUERA DE RANGO');
            DELAY(LIMDEL);
            GOTOXY(50,8); WRITE('                     ');
            BCP(30,7,5);
            BCP(30,8,30);
         END;
      END;
```

```
    BAN2:=TRUE;
    WHILE BAN2 DO
    BEGIN
       ACENTC(30,9,5,SUBTEMA);
       IF (SUBTEMA<LIMSTE) AND (SUBTEMA>=0) THEN
       BEGIN
          SEEK(ARSTE,SUBTEMA);
          READ(ARSTE,VARSTE);
          IF NUMSTE>=0 THEN
          BEGIN
             GOTOXY(35,9); WRITE (NOMSUB);
             GOTOXY(35,10);WRITE('¿ES CORRECTO?[SI/NO]=>');
             READ(RESP);
             IF RESP<>'NO' THEN BAN2:=FALSE
             ELSE
             BEGIN
                BCP(30,5,35);
                BCP(30,10,30);
             END;
          END
          ELSE
          BEGIN
             GOTOXY(50,10); WRITE('SUBTEMA NO DADO DE ALTA');
             DELAY(LIMDEL);
             GOTOXY(50,10); WRITE('                        ');
          END;
       END
       ELSE
       BEGIN
            GOTOXY(50,10);WRITE('SUBTEMA FUERA DE RANGO');
            DELAY(LIMDEL);
            GOTOXY(50,10);WRITE('                        ');
       END;
    END;
DDUR:=0;
WHILE DDUR=0 DO
BEGIN
  ACEREC(30,11,5,DDUR);
  IF DDUR>24 THEN DDUR:=0;
  HORDEC(DDUR);
  DURACIO:=DDUR;
END;
BAN3:=TRUE;
WHILE BAN3 DO
BEGIN
    ACENTC(30,18,5,AUX);
    BAN3:=FALSE;
    CASE AUX OF
       1 : MOBIL_C:='ESCUELA';
       2 : MOBIL_C:='AUDITORIO';
       3 : MOBIL_C:='COMITE';
       4 : MOBIL_C:='ALIMENTOS';
       ELSE
       BEGIN
          GOTOXY(50,18); WRITE('ERROR EN CODIGO');
          BAN3:=TRUE;
          DELAY(LIMDEL);
```

```
                GOTOXY(50,18); WRITE('                ');
              END;
           END;
        END;
           BAN3:=TRUE;
           WHILE BAN3 DO
           BEGIN
              ACENTC(30,22,7,QUORUM);
              IF QUORUM>0 THEN BAN3:=FALSE;
           END;
        END;
        GOTOXY(30,24);WRITE('SE GRABA PANTALLA NO/RET=>');
        READ(RESP);
        IF RESP<>'NO' THEN
        BEGIN
           GOTOXY(60,24); WRITELN('REGISTRO GRABADO');
           DELAY(LIMDEL);
           SEEK(ARCON,DIR);
           WRITE(ARCON,VARCON);
        END;
        BCP(30,24,46);
      END;
        GOTOXY(30,24);WRITE('OTRA ALTA NO/RET=>');
        READ(RESP);
        DIR:=DIR+1;
        IF RESP='NO' THEN
        BAN1:=FALSE
        ELSE
        BTL(30,2,49,23);
     CLOSE(ARCON);
     END;
     CLOSE(ARTEM);
     CLOSE(ARSTE);
     CLRSCR;
  END;
  PROCEDURE CONFER_P;
  VAR
    BAN1 : BOL;
    OPCI : CAR;
  BEGIN
     BAN1:=TRUE ;
     WHILE BAN1 DO
     BEGIN
       PANGRAL;
       ACECAC(61,16,5,OPCI);
       CASE OPCI OF
          'A' : ALTCON;
          'B' : BAJCON;
          'C' : CAMCON;
          'L' : LISCON;
          'T' : TEMA;
          'S' : SUBTEM;
          'F' : BAN1:=FALSE;
          ELSE
          BEGIN
             GOTOXY(61,18);WRITE('ERROR EN CLAVE');
             DELAY(LIMDEL);BCP(61,18,20);
```

```
            END;
         END;
      END;
   END;
   PROCEDURE CRESAL(RESP  :  ST20);
   VAR
      CONT : ENT;
   BEGIN
      WITH VARSAL DO
      BEGIN
         CONT:=1;
         ASSIGN(ARSAL,RESP);
         REWRITE(ARSAL);
         S_DIR:=0;
         S_MOB:=' ';
         S_NOM:=' ';
         S_CAP:=0;
         CONT:=1;
         WHILE CONT<=10 DO
         BEGIN
            S_HIN[CONT]:=0;
            S_DIS[CONT]:=0;
            IF CONT<= 4 THEN S_TMO[CONT]:=0;
            CONT:=CONT+1;
         END;
         CONT:=0;
         WHILE CONT<= LIMSAL DO
         BEGIN
            SEEK(ARSAL,CONT);
            WRITE(ARSAL,VARSAL);
            CONT:=CONT+1;
         END;
         CLOSE(ARSAL);
      END;
   END;
   PROCEDURE APESAL( RESP :  ST12);
   BEGIN
      IF EXICON(RESP)=FALSE THEN CRESAL(RESP);
      ASSIGN(ARSAL,RESP);
      RESET(ARSAL);
   END;
   PROCEDURE PANSAL;
   BEGIN
      PINCUA;  TEXTCOLOR(5);
      GOTOXY(4,3); WRITE('Clave del Salon:');
      GOTOXY(11,5);WRITE('Nombre :');
      GOTOXY(4,7); WRITE('Capacidad de acuerdo ');
      GOTOXY(4,8); WRITE('al mobiliario');
      GOTOXY(10,9); WRITE('ESCUELA:');
      GOTOXY(10,10); WRITE('AUDITORIO:');
      GOTOXY(10,11); WRITE('COMITE:');
      GOTOXY(10,12); WRITE('ALIMENTOS:');
      GOTOXY(4,14); WRITE('Horario general:');
      GOTOXY(9,15); WRITE('DE INICIO:');
      GOTOXY(9,16); WRITE('DISPONIBLE:');
   END;
   PROCEDURE LISAC(VARSAL: SALON);
```

```
BEGIN
WITH VARSAL DO
BEGIN
GOTOXY(30,5); WRITE(S_NOM);
GOTOXY(30,9); WRITE(S_TMO[1]);
GOTOXY(30,10); WRITE(S_TMO[2]);
GOTOXY(30,11); WRITE(S_TMO[3]);
GOTOXY(30,12); WRITE(S_TMO[4]);
GOTOXY(30,15); WRITE(S_HIN[1]:5:2);
GOTOXY(30,16); WRITE(S_DIS[1]:5:2);
END;
END;
PROCEDURE PANESAL;
BEGIN
   GOTOXY(4,21); WRITE('DIA RELATIVO :');
   GOTOXY(4,22); WRITE('HORA INICIAL :');
   GOTOXY(4,23); WRITE('HORA DISP :');
END;
PROCEDURE CSNOM(VAR SNOM  :  ST15);
VAR
BAN1 : BOL;
BEGIN
   BAN1:=TRUE;
   WHILE BAN1 DO
   BEGIN
   BAN1:=FALSE;
   GOTOXY(30,5);READ(SNOM);
   IF LENGTH(SNOM)=0 THEN BAN1:=TRUE;
   END;
END;
PROCEDURE CSTMO(VAR STMO : ARRE4;VAR SCAP : ENT; VAR SMOB : ST15);
VAR
   BAN2,
   BAN1  :  BOL;
   OPCI  :  ENT;
BEGIN
   BAN1:=TRUE;
   WHILE BAN1 DO
   BEGIN
      ACENTC(30,9,10,STMO[1]);
      ACENTC(30,10,10,STMO[2]);
      ACENTC(30,11,10, STMO[3]);
      ACENTC(30,12,10,STMO[4]);
      SCAP:=99;
      IF (STMO[1]+STMO[2]+STMO[3]+STMO[4])>0 THEN
         BAN1:=FALSE
      ELSE
         BEGIN
            GOTOXY(50,12);WRITE('MOVIMIENTO NO ACEPTADO');
            DELAY(LIMDEL); BCP(50,12,23);
         END;
   END;
END;
PROCEDURE CSHIN(VAR SHIN:ARRE10; VAR SDIS : ARRE10);
VAR
   RESENT  : ENT;
   RESREA  : REA;
```

```
   RESRE2   : REA;
   BAN1     : BOL;
   DIR      : ENT;
   RESP     : ST2;
BEGIN
   BAN1:=TRUE;
   WHILE BAN1 DO
   BEGIN
      RESENT:=10;
      ACEREC(30,15,10,RESREA);
      ACEREC(30,16,10,RESRE2);
      IF (RESREA<RESRE2) AND (RESREA>0) AND (RESRE2>0) THEN
         BAN1:=FALSE
      ELSE
      BEGIN
         GOTOXY(40,16):WRITE('ALGUN DATO EN CERO');
         DELAY(LIMDEL);BCP(40,16,20) ;
      END;
      RESRE2:=RESRE2-RESREA;
   END ;
   GOTOXY(8,22);
   WRITE('HASTA ESTE MOMENTO TODOS LOS DIAS HAN SIDO INICIALIZADOS');
   GOTOXY(8,23);
   WRITE('CON LOS DATOS ANTES DADOS.SI DESEA ALGUNA MODIFICACION');
   GOTOXY(8,24);
   WRITELN('INDIQUELA');
   DIR:=1;
   WHILE DIR<=10 DO
   BEGIN
        SHIN[DIR]:=RESREA;
        SDIS[DIR]:=RESRE2;
        DIR:=DIR+1;
   END;
   BAN1:=TRUE;
   WHILE BAN1 DO
   BEGIN
      GOTOXY(30,24);WRITE('DESEA CORRECCION NO/RET=>');
      READLN(RESP);
      BTL(2,20,75,5);
      IF RESP='NO' THEN
         BAN1:=FALSE
      ELSE
         BEGIN
            PANESAL;
            RESENT:=11;
            WHILE RESENT>10 DO
            BEGIN
               ACENTC(30,21,1,RESENT);
               IF RESENT=0 THEN RESENT:=11;
            END;
            ACEREC(30,22,1,SHIN[RESENT]);
            ACEREC(30,23,1,SDIS[RESENT]);
         END;
    END;
END;
PROCEDURE LISALU(DIR   :  ENT);
VAR
```

```
      CON2,
      CONT   :  ENT;
      RESP   :  CAR;
   BEGIN
      WITH VARSAL DO
      BEGIN
         SEEK(ARSAL,DIR);
         READ(ARSAL,VARSAL);
         IF S_CAP>0 THEN
         BEGIN
            CLRSCR;
            PINCUA; TEXTCOLOR(5);
            GOTOXY(10,5);WRITE ('NO. DE SALON=>',S_DIR);
            GOTOXY(10,6);WRITE('NOMBRE=>',S_NOM);
            GOTOXY(10,7);WRITE('CAPACIDAD POTENCIAL CON MOBILIARIO TIPO:')
            GOTOXY(10,8);WRITE('ESCUELA=>',S_TMO[1]);
            GOTOXY(10,9);WRITE('AUDITORIO=>',S_TMO[2]);
            GOTOXY(10,10);WRITE('COMITE=>',S_TMO[3]);
            GOTOXY(10,11);WRITE('ALIMENTOS=>',S_TMO[4]);
            CONT:=1;  CON2:=12;
            WHILE CONT<=10 DO
            BEGIN
               DECHOR(S_HIN[CONT]);
               DECHOR(S_DIS[CONT]);
               GOTOXY(10,CON2);
ISP=>',        WRITE('HORA INICIAL DIA:',' ',CONT,'=>',S_HIN[CONT]:5:2,' D
               S_DIS[CONT]:5:2);
                CONT:=CONT+1;  CON2:=CON2+1;
            END;
            GOTOXY(20,24);WRITE('PARA CONTINUAR TECLEE RETURN');
            READLN(RESP);
         END;
      END;
   END;
   PROCEDURE LISAL;
   VAR
    CONT   :  ENT;
   BEGIN
      CONT:=0;
      WHILE CONT<=LIMSAL DO
      BEGIN
         LISALU(CONT);
         CONT:=CONT+1;
      END;
      CLRSCR;
   END;
   PROCEDURE BASAL(RESP1 : ST12);
   VAR
      CON1,
      CONT    :  ENT;
      BAN2,
      BAN1    :  BOL;
      RESP    :  ST2;
   BEGIN
     BAN2:=TRUE;
     CLRSCR;
     WHILE BAN2 DO
```

```
BEGIN
 PANSAL;
 APESAL(RESP1);
 BAN1:=TRUE;
 WHILE BAN1 DO
 BEGIN
    ACENTC(30,3,5,CONT);
    IF (CONT<=LIMSAL) AND (CONT>0) THEN
    BEGIN
       SEEK(ARSAL,CONT);
       READ(ARSAL,VARSAL);
       WITH VARSAL DO
       BEGIN
          IF S_CAP>0 THEN
             BAN1:=FALSE
          ELSE
          BEGIN
             GOTOXY(50,3);WRITE('NO DADO DE ALTA');
             DELAY(LIMDEL); BCP(50,3,20);
          END;
       END;
    END
    ELSE
    BEGIN
       GOTOXY(50,3):WRITE('CLAVE ERRONEA');
       DELAY(LIMDEL); BCP(50,3,15);
    END;
 END;
 LISAC(VARSAL);
=>');  GOTOXY(25,24);WRITE('TODA INFORMACION SERA BORRADA ¿CORRECTO SI/RET?
 READ(RESP);  BCP(25,24,54);
 IF RESP='SI' THEN
 BEGIN
    BTL(30,2,49,23);
    SEEK(ARSAL,CONT);
    READ(ARSAL,VARSAL);
    WITH VARSAL DO
    BEGIN
       S_DIR:=0;
       S_MOB:=' ';
       S_NOM:=' ';
       S_CAP:=0;
       CON1:=1;
       WHILE CON1<=10 DO
       BEGIN
          S_HIN[CON1]:=0;
          S_DIS[CON1]:=0;
          IF CON1>=4 THEN S_TMO[CON1]:=0;
          CON1:=CON1+1
       END;
    END;
    SEEK(ARSAL,CONT);
    WRITE(ARSAL,VARSAL);
 END;
 GOTOXY(25,24):WRITE('OTRA BAJA NO/RET=>');
 READ (RESP);BCP(25,24,34);
 IF RESP='NO' THEN BAN2:=FALSE;
```

```
    CLOSE(ARSAL);
  END;
  CLRSCR;
END;
PROCEDURE ALSAL(RESP1:ST12);
VAR
   BAN1 :  BOL;
   BAN2 :  BOL;
   DIR  :  ENT;
   RESP :  ST2;
BEGIN
   WITH VARSAL DO
   BEGIN
      BAN1:=TRUE;
      WHILE BAN1 DO
      BEGIN
         APESAL(RESP1);
         BAN2:=TRUE;
         CLRSCR;
         PANSAL;
         WHILE BAN2 DO
         BEGIN
            ACENTC(30,3,5,DIR);
            IF (DIR<=LIMSAL) AND (DIR>0) THEN BAN2:=FALSE;
         END;
         SEEK(ARSAL,DIR);
         READ(ARSAL,VARSAL);
         IF S_DIR =0 THEN
         BEGIN
            S_DIR:=DIR;
            CSNOM(S_NOM);
            CSTMO(S_TMO,S_CAP,S_MOB);
            CSHIN(S_HIN,S_DIS);
            SEEK(ARSAL,DIR);
            WRITE(ARSAL,VARSAL);
         END
         ELSE
         BEGIN
            GOTOXY(45,24);WRITE('DADO DE ALTA');
            DELAY(LIMDEL);BCP(45,24,14);
         END;
         GOTOXY(30,24);WRITE('OTRA ALTA NO/RET=>');
         READ(RESP);
         IF RESP='NO' THEN BAN1:=FALSE;
         CLOSE(ARSAL);
      END;
   END;
   CLRSCR;
END;
PROCEDURE CASAL(RESP1:ST12);
VAR
   BAN3,
   BAN1 :  BOL;
   BAN2 :  BOL;
   DIR  :  ENT;
   RESP :  ST2;
   OPC1 :  CAR;
```

```
BEGIN
   WITH VARSAL DO
   BEGIN
      CLRSCR;
      PANSAL;
      BAN1:=TRUE;
      WHILE BAN1 DO
      BEGIN
         APESAL(RESP1);
         BAN2:=TRUE;
         WHILE BAN2 DO
         BEGIN
            ACENTC(30,3,5,DIR);
            IF (DIR<=LIMSAL) AND (DIR>0) THEN BAN2:=FALSE;
         END;
         SEEK(ARSAL,DIR);
         READ(ARSAL,VARSAL);
         GOTOXY(5,18); WRITE('Finalizar');
         IF S_CAP >0 THEN
         BEGIN
            LISAC(VARSAL);
            BAN3:=TRUE;
            WHILE BAN3 DO
            BEGIN
               GOTOXY(30,20);ACECAR('OPCION [PRIMERA LETRA]=>',OPCI);
               CASE OPCI OF
               'N'    :     CSNOM(S_NOM);
               'M'    :     CSTMO(S_TMO,S_CAP,S_MOB);
               'H'    :     CSHIN(S_HIN,S_DIS);
               'F'    :     BAN3:=FALSE;
                ELSE          BEGIN
                               GOTOXY(40,20);WRITE('ERROR EN CLAVE');
                               DELAY(LIMDEL);BCP(40,20,20);
                               BAN3:=TRUE;
                               END;
                END;
            END;
            SEEK(ARSAL,DIR);
            WRITE(ARSAL,VARSAL);
         END
         ELSE
         BEGIN
            GOTOXY(30,24);WRITE('NO DADO DE ALTA');
            DELAY(LIMDEL);BCP(30,24,30);
         END;
         GOTOXY(30,24);WRITE('OTRO CAMBIO NO/RET=>');
         READLN(RESP); BCP(30,24,34);
         IF RESP='NO' THEN BAN1:=FALSE;
         CLOSE(ARSAL);
      END;
   END;
   CLRSCR;
END;
PROCEDURE APEUSA;
VAR
   CONT  : ENT;
BEGIN
```

```
WITH VARUSA DO
BEGIN
IF EXICON('USAL.DAT') THEN
BEGIN
   ASSIGN(ARUSA,'USAL.DAT');
   RESET(ARUSA);
END
ELSE
BEGIN
   ASSIGN(ARUSA,'USAL.DAT');
   REWRITE(ARUSA);
   CONT:=0;
   WHILE CONT<=LIMUSA DO
   BEGIN
      USAL1:=0;
      USAL1N:=' ';
      USAL2:=0;
      USAL2N:=' ';
      USALI:=' ';
      SEEK(ARUSA,CONT);
      WRITE(ARUSA,VARUSA);
      CONT:=CONT+1;
   END;
   CLOSE(ARUSA);
   ASSIGN(ARUSA,'USAL.DAT');
   RESET(ARUSA);
END;
END;
END;
PROCEDURE APESAU(RESP: ST12);
BEGIN
   ASSIGN(ARSAL,RESP);
   RESET(ARSAL);
END;
PROCEDURE LISUSA(VARUSA : UNESAL);
BEGIN
  CLRSCR; PINCUA;
   WITH VARUSA DO
   BEGIN
        GOTOXY(10,20);WRITE('S1=>',USAL1,' ',USAL1N);
        GOTOXY(10,21);WRITE('S2=>',USAL2,' ',USAL2N);
        GOTOXY(10,22);WRITE('IDENTIFICADOR=>',USALI);
   END;
END;
PROCEDURE SALUNE(RESP12 : ST12);
VAR
ARAL  : ARRE2;
RESP1,
RESP   :  ST2;
BAN2,
BAN3,
BAN4,
BAN1   :  BOL;
RESDIA,
RESPE,
CONM,
CONT,
```

```
DIRS,
COND,
CONT1,
CONT2   :  ENT;
BEGIN
   APESAU(RESP12);
   WITH VARUSA,VARSAL DO
   BEGIN
      CLRSCR; PINCUA;
      GOTOXY(10,10);
      WRITE('ANTES DE CONTINUAR NECESITA LA CLAVE DE LOS SALONES');
      GOTOXY(10,11);WRITE('A UNIR, DESEA CONTINUAR[NO/RET]=>');
      READ(RESP);
      CLRSCR;
      IF RESP='NO' THEN
      BEGIN
         GOTOXY(30,13);WRITE('CONSULTE LISTADO DE SALONES');
         DELAY(LIMDEL); CLRSCR;
      END
      ELSE
          BEGIN
             BAN2:=TRUE;
             WHILE BAN2 DO
             BEGIN
                APEUSA;
                BAN3:=TRUE;
                BAN4:=TRUE;
                WHILE BAN3 DO
                BEGIN
                PINCUA;
                GOTOXY(10,5);
                WRITE('DE LA CLAVE DE LA UNION=>');
                ACENTC(40,5,5,DIRS);
                IF (DIRS>0) AND (DIRS<LIMUSA) THEN
                BEGIN
                   PINCUA; TEXTCOLOR(3);
                   SEEK(ARUSA,DIRS);
                   READ(ARUSA,VARUSA);
                   IF USAL1=0 THEN BAN3:=FALSE
                   ELSE
                   BEGIN
                      GOTOXY(30,6); WRITE('DADO DE ALTA');
                      DELAY(LIMDEL); BCP(30,6,20);
                      LISUSA(VARUSA);
                      GOTOXY(10,7);
                      WRITE('DESEA DARLO BAJA SI/RET=>');
                      READ(RESP1);
                      IF RESP1='SI'THEN
                      BEGIN
                          USAL1:=0;
                          USAL2:=0;
                          USAL1N:=' ';
                          USAL2N:=' ';
                          USALI:=' ';
                          SEEK(ARUSA,DIRS);
                          WRITE(ARUSA,VARUSA);
                          CLOSE(ARUSA);
```

```
            BAN4:=FALSE;
            BAN3:=FALSE;
          END;
          CLRSCR;
      END;
END
ELSE
BEGIN
      GOTOXY(30,6); WRITE('FUERA DE RANGO');
      DELAY(LIMDEL); BCP(30,6,20);
END;
END;
IF BAN4 THEN
BEGIN
    CONT1:=1;
    WHILE CONT1<=2 DO
    BEGIN
      BAN1:=TRUE;
      WHILE BAN1 DO
      BEGIN
         GOTOXY(10,10);
         WRITE('DE UD. LA CLAVE DEL SALON ' ,CONT1,'=>')
         ACENTC(40,10,10,RESPE);
         ARAL[CONT1]:=RESPE;
         IF (ARAL[CONT1]>0) AND (ARAL[CONT1]<=LIMSAL) TH
         BEGIN
           SEEK(ARSAL,ARAL[CONT1]);
           READ(ARSAL,VARSAL);
           IF S_CAP>0 THEN
           BEGIN
             GOTOXY(10,12);WRITE('SALON=>',S_NOM);
             GOTOXY(10,13);WRITE('ES CORRECTO SI/NO=>');
             READ(RESP);
             IF RESP='SI' THEN
             BEGIN
               IF CONT1=1 THEN
               BEGIN
                  USAL1:=ARAL[CONT1];
                  USAL1N:=S_NOM;
               END
               ELSE
               BEGIN
                 USAL2:=ARAL[CONT1];
                 USAL2N:=S_NOM;
               END;
               BAN1:=FALSE;
               CONT1:=CONT1+1;
            END;
            END
            ELSE
            BEGIN
              GOTOXY(50,13);WRITE('NO DADO DE ALTA');
              DELAY(LIMDEL);BCP(50,13,20);
            END;
         END
         ELSE
         BEGIN
```

```
                                GOTOXY(50,13);WRITE('FUERA DE RANGO');
                                DELAY(LIMDEL);BCP(50,13,20);
                              END;
                          END;
                       END;
                       GOTOXY(10,15);
                       WRITE('NOMBRE DE LA PAREJA=>');
                       READ(USALI);
                       SEEK(ARUSA,DIRS);
                       WRITE(ARUSA,VARUSA);
                       CLOSE(ARUSA);
                    BAN1:=TRUE;
                    WHILE BAN1 DO
                    BEGIN
                       GOTOXY(10,20);WRITE('OTRA PAREJA SI/NO=>');
                       READ(RESP);
                       BCP(10,20,40);
                       IF RESP='SI' THEN  BAN1:=FALSE
                       ELSE
                       BEGIN
                               IF RESP='NO' THEN
                               BEGIN
                                  BAN1:=FALSE;
                                  BAN2:=FALSE;
                               END;
                       END;
                    END;
                 END;
              END;
           END;
           CLOSE(ARSAL);
        END;
        END;
        PROCEDURE SALON_P;
        VAR
           BAN1   :  BOL;
           OPC1   :  CAR;
           RESP1  : ST8;
           RESP   : ST12;
           CON    :  ENT;
        BEGIN
           BAN1:=TRUE;
           WHILE BAN1 DO
           BEGIN
              BAN1:=FALSE;
              CON:=1;
              WHILE CON<=8 DO
              BEGIN
                 RESP1[CON]:=' ';
                 CON:=CON+1;
              END;
=>');         GOTOXY(4,24);WRITE('CENTRO DE CONVENCIONES [MAXIMO 8 CARACTERES]=
              GOTOXY(49,24);READ(RESP1);
              IF LENGTH(RESP1)=0 THEN BAN1:=TRUE;
              RESP:=RESP1+'.DAT';
           END;
           BAN1:=TRUE;
```

```
    WHILE BAN1 DO
    BEGIN
       PANGRAL;
       ACECAC(61,22.4,OPCI);
       CASE OPCI OF
          'A' :  BEGIN
                   ALSAL(RESP);
                 END;
          'B' :  BEGIN
                   BASAL(RESP);
                 END;
          'C' :  BEGIN
                   CASAL(RESP);
                 END;
          'L' :  BEGIN
                   APESAL(RESP);
                   LISAL;
                   CLOSE(ARSAL);
                 END;
           'U' : SALUNE(RESP);
          'F' :  BAN1:=FALSE;
          ELSE
                 BEGIN
                   GOTOXY(61,24):WRITE('ERROR EN CLAVE');
                   DELAY(LIMDEL):BCP(61,24,20);
                   CLOSE(ARSAL);
                 END;
       END;
    END;
END;
BEGIN (* RUTINA PRINCIPAL *)
    CLRSCR;
    PANGRAL;
    BAND1:=TRUE;
    WHILE BAND1 DO
    BEGIN
        ACECAC(61,8.5,OPCI);
        CASE OPCI OF
        'C'  :  CONFER_P;
        'S'  :  SALON_P ;
        'F'  :  BAND1:= FALSE;
        ELSE
           BEGIN
             GOTOXY(61,10);WRITE('ERROR EN CODIGO');
             DELAY (2000); BCP(61,10,20);
           END;
        END;
     END;
     CLRSCR;
END.
```

```
PROGRAM TRACON: (*INPUT/OUTPUT*)
CONST
LIMCON   :  INTEGER=100;
LIMTEM   :  INTEGER=50;
TYPE
ENT      = INTEGER;
BOL      = BOOLEAN;
REA      = REAL;
CAR      = CHAR;
ST10     = STRING[10];
ST2      = STRING[2];
ST4      = STRING[4];
ST5      = STRING[5];
ST09     = STRING[9];
ST15     = STRING[15];
ST20     = STRING[10];
ST50     = STRING[50];
ST80     = STRING[80];
CONFER   = RECORD
             CVECON   :  ENT;
             QUORUM   :  ENT;
             DURACIO  :  REAL;
             MOBIL_C  :  ST10;
             TEMA     :  ENT;
             SUBTEMA  :  ENT;
             NOMBRE   : ST50;
             END;
NOMTEM   =   RECORD
             NUMTEM   : ENT;
             NOMTEMA  : ST50;
             END;
NOMSTE   =   RECORD
             NUMSTE   : ENT;
             NOMSUB   : ST50;
             END;
VAR
ARCON    :   FILE OF CONFER;
VARCON   :   CONFER;
ARTEM    :   FILE OF NOMTEM;
VARTEM   :   NOMTEM;
ARSTE    :   FILE OF NOMSTE;
VARSTE   :   NOMSTE;
ARTXT    :   TEXT;
VC       :   ARRAY[1..105] OF CHAR;
VC1      :   ARRAY[1..79] OF CHAR;
VECCVE   :   ST2;
VECQUO   :   ST4;
VECDUR   :   ST5;
VECMOB   :   ST10;
VECTEM   :   ST50;
VECSUB   :   ST50;
VECNOM   :   ST50;
BAN1     :   BOL;
CONT.
CONT1.
CONT2    :   ENT;
PROCEDURE BORVAR;
```

```
VAR
   CONT   : ENT;
BEGIN
   VECCVE[1]:=' ';
   VECCVE[2]:=' ';
   CONT:=1;
   WHILE CONT<=4 DO
   BEGIN
      VECQUO[CONT]:=' ';
      VECDUR[CONT]:=' ';
      CONT:=CONT+1;
   END;
   VECDUR[5]:=' ';
   CONT:=1;
   WHILE CONT<=10 DO
   BEGIN
      VECMOB[CONT]:=' ';
      CONT:=CONT+1;
   END;
   CONT:=1;
   WHILE CONT<=50 DO
   BEGIN
      VECTEM[CONT]:=' ';
      VECSUB[CONT]:=' ';
      VECNOM[CONT]:=' ';
      CONT:=CONT+1;
   END;
END;
PROCEDURE APEAR;
BEGIN
   ASSIGN(ARTEM,'TEMA.DAT');
   RESET(ARTEM);
   ASSIGN(ARSTE,'SUBTEMA.DAT');
   RESET(ARSTE);
   ASSIGN(ARCON,'CONFER.DAT');
   RESET(ARCON);
   ASSIGN(ARTXT,'BASCON.TXT');
   REWRITE(ARTXT);
END;
PROCEDURE FINCUA;
VAR
   CON1,CON2   :  INTEGER;
BEGIN
   GOTOXY(9,9);
   WRITE('┌──────────────────────────────────────────────────────────────────┐');
   WRITE('└──────────────────┘');
   CON1:=10;
   CON2:=1;
   WHILE CON1<21 DO
   BEGIN
      CON2:=9;
      GOTOXY(CON2,CON1); WRITE('│');
      CON2:=73;
      GOTOXY(CON2,CON1); WRITE('│');
      CON1:=CON1+1;
   END;
   GOTOXY(9,21);
```

```
    WRITE('└──────────────────────────────────────────────────────────────────────────────┘');
    WRITE('──────────────────┘ ');
END;
PROCEDURE BORRA;
VAR
CONT3   :   ENT;
BEGIN
    CONT3:=1;
    WHILE CONT3<=105 DO
    BEGIN
        VC[CONT3]:=' ';
        CONT3:=CONT3+1;
    END;
    CONT3:=1;
    WHILE CONT3<=79 DO
    BEGIN
        VC1[CONT3]:=' ';
        CONT3:=CONT3+1;
    END;
END;
BEGIN (*RUTINA PRINCIPAL*);
    CLRSCR;
    PINCUA;
    WINDOW(10,10,70,20);
    TEXTCOLOR(2);
    GOTOXY(1,1);
    APEAR;
    BORRA;
    BORVAR;
    WITH VARCON,VARSTE,VARTEM DO
    BEGIN
        CONT:=0;
        WHILE CONT<= LIMCON DO
        BEGIN
            SEEK(ARCON,CONT);
            READ(ARCON,VARCON);
            IF DURACIO > 0 THEN
            BEGIN
                WRITELN(NOMBRE);
                SEEK(ARTEM,TEMA);
                READ(ARTEM,VARTEM);
                SEEK(ARSTE,SUBTEMA);
                READ(ARSTE,VARSTE);
                VC[1]:='c';
                VC[2]:='o';
                VC[3]:='n';
                VC[4]:='t';
                VC[5]:='e';
                VC[6]:='r';
                VC[7]:='e';
                VC[8]:='n';
                VC[9]:='c';
                VC[10]:='i';
                VC[11]:='a';
                VC[12]:='(';
                STR(TEMA:2,VECCVE);
                VC[13]:=VECCVE[1];
```

```
VC[14]:=VECCVE[2];
VC[15]:=',';
STR(QUORUM:4,VECQUO);
VC[16]:=VECQUO[1];
VC[17]:=VECQUO[2];
VC[18]:=VECQUO[3];
VC[19]:=VECQUO[4];
VC[20]:=',';
CONT1:=21;
CONT2:=1;
STR(DURACIO:5:2,VECDUR);
WHILE CONT1<=25 DO
BEGIN
   VC[CONT1]:=VECDUR[CONT2];
   CONT1:=CONT1+1;
   CONT2:=CONT2+1;
END;
VC[26]:=',';
VC[27]:='"';
CONT1:=28 ;
CONT2:=1;
VECMOB:=MOBIL_C;
WHILE CONT1<=37 DO
BEGIN
  VC[CONT1]:=VECMOB[CONT2];
  CONT1:=CONT1+1;
  CONT2:=CONT2+1;
END;
VC[38]:='"';
VC[39]:=',';
VC[40]:='"';
VECTEM:=NOMTEMA;
CONT1:=41;
CONT2:=1;
WHILE CONT1<=60 DO
BEGIN
   VC[CONT1]:=VECTEM[CONT2];
   CONT1:=CONT1+1;
   CONT2:=CONT2+1;
END;
VC[61]:='"';
VC[62]:=',';
VC[63]:='"';
VECSUB:=NOMSUB;
CONT1:=64;
CONT2:=1;
WHILE CONT1<=78 DO
BEGIN
   VC[CONT1]:=VECSUB[CONT2];
   CONT1:=CONT1+1;
   CONT2:=CONT2+1;
END;
VC[79]:='"';
VC[80]:=',';
VC[81]:='"';
CONT1:=82;
CONT2:=1;
```

```
            VECNOM:=NOMBRE;
            WHILE CONT1<= 97 DO
            BEGIN
               VC[CONT1]:=VECNOM[CONT2];
               CONT1:=CONT1+1;
               CONT2:=CONT2+1;
            END;
            VC[98]:='"';
            VC[99]:=',';
            VC[100]:='0';
            VC[101]:=')';
            WRITELN(ARTXT,VC);
            BORRA;
            BORVAR;
         END;
         CONT:=CONT+1;
      END;
      BORRA;
      BORVAR;
      CONT:=0;
      WHILE CONT<= LIMTEM DO
      BEGIN
         SEEK(ARTEM,CONT);
         READ(ARTEM,VARTEM);
         IF NUMTEM > 0 THEN
         BEGIN
            WRITE(NOMBRE);
            VC1[1]:='t';
            VC1[2]:='e';
            VC1[3]:='m';
            VC1[4]:='a';
            VC1[5]:='_';
            VC1[6]:='c';
            VC1[7]:='o';
            VC1[8]:='n';
            VC1[9]:='f';
            VC1[10]:='(';
            STR(NUMTEM:2,VECCVE);
            VC1[11]:=VECCVE[1];
            VC1[12]:=VECCVE[2];
            VC1[13]:=')';
            WRITELN(ARTXT,VC1);
            BORRA;
            BORVAR;
         END;
         CONT:=CONT+1;
      END;
      CLOSE(ARTXT);
   END;
   WINDOW(1,1,80,25);
   CLRSCR;
END.
```

```
PROGRAM TRASAL; (*INPUT/OUTPUT*)
CONST
LIMSAL   :  INTEGER=50;
LIMUSA   :  INTEGER=25;
LIMDEL   :  INTEGER=1000;
TYPE
ENT     = INTEGER;
BOL     = BOOLEAN;
REA     = REAL;
CAR     = CHAR;
ST10    = STRING[10];
ST08    = STRING[8];
ST2     = STRING[2];
ST4     = STRING[4];
ST5     = STRING[5];
ST09    = STRING[9];
ST15    = STRING[15];
ST20    = STRING[20];
ST50    = STRING[50];
ST80    = STRING[80];
ARRE10  = ARRAY[1..10] OF REAL;
ARRE4   = ARRAY[1..4] OF INTEGER;
ARRE2   = ARRAY[1..2] OF INTEGER;
ARRM    = ARRAY[1..4] OF ST10;
SALON   = RECORD
            S_NOM   :  ST15;
            S_MOB   :  ST15;
            S_CAP   :  ENT;
            S_HIN   :  ARRE10;
            S_DIS   :  ARRE10;
            S_TMO   :  ARRE4;
            S_DIR   :  ENT;
            END;
UNESAL =   RECORD
             USAL1   : ENT;
             USAL1N  : ST50;
             USAL2   : ENT;
             USAL2N  : ST50;
             USALI   : ST50;
             END;
VAR
ARSAL   : FILE OF SALON;
VARSAL  : SALON;
ARUSA   : FILE OF UNESAL;
VARUSA  : UNESAL;
ARTXT   :  TEXT;
VC      :  ARRAY[1..70] OF CHAR;
VM      :  ARRM;
ARAL    :  ARRE2;
RESP,
VECCVE  :  ST2;
VECDUR  :  ST4;
VECMOB  :  ST10;
VECHOR  :  ST5;
VECNOM  :  ST50;
VECTEM  :  ST15;
BAN2,
```

```
BAN1    :  BOL;
RESDIA,
CONM,
CONT,
COND,
CONT1,
CONT2   :  ENT;
FUNCTION EXICON(NOMARC : ST20):BOOLEAN;
VAR
  FIL   : FILE;
BEGIN
   ASSIGN(FIL,NOMARC);
   (*$I-*) RESET (FIL); (*$I+*)
   EXICON:=(IORESULT=0);
   CLOSE(FIL);
END;
PROCEDURE PINCUA;
VAR
   CON1,CON2   :  INTEGER;
BEGIN
   GOTOXY(9,9);
   WRITE('┌──────────────────────────────────────────────────────────────────────');
   WRITE('──────────┐');
   CON1:=10;
   CON2:=10;
   WHILE CON1<21 DO
   BEGIN
      CON2:=9;
      GOTOXY(CON2,CON1); WRITE('│');
      CON2:=72;
      GOTOXY(CON2,CON1); WRITE('│');
      CON1:=CON1+1;
   END;
   GOTOXY(9,21);
   WRITE('└──────────────────────────────────────────────────────────────────────');
   WRITE('──────────┘');
END;
PROCEDURE IMOB;
BEGIN
   VM[1]:='ESCUELA';
   VM[2]:='AUDITORIO';
   VM[3]:='COMITE';
   VM[4]:='ALIMENTOS';
END;
PROCEDURE APEAR;
VAR
RES : ST08;
RES1 : ST20;
BAN1 : BOL;
BEGIN
   RES:=' ';
   BAN1:=FALSE;
   WHILE NOT BAN1 DO
   BEGIN
      WRITE('DE UD EL NOMBRE DEL HOTEL O CC=>');
      READLN(RES);
      RES1:=RES+'.DAT';
```

```
      BAN1:=EXICON(RES1);
   END;
   ASSIGN(ARSAL,RES1);
   RESET(ARSAL);
   ASSIGN(ARTXT,'BASAL.TXT');
   REWRITE(ARTXT);
   ASSIGN(ARUSA,'USAL.DAT');
   RESET(ARUSA);
END;
PROCEDURE LISTAS;
VAR
   CONT : ENT;
BEGIN
   WITH VARSAL DO
   BEGIN
      CONT:=1;
      WHILE CONT<=LIMSAL DO
      BEGIN
         SEEK(ARSAL,CONT);
         READ(ARSAL,VARSAL);
         IF S_CAP>0 THEN  WRITELN('SALON=>',CONT,' ',S_NOM);
         CONT:=CONT+1;
      END;
   END;
END;
PROCEDURE ACEENT(MEMBRETE :ST50; VAR VARENT : ENT);
VAR
   OK  : BOL;
BEGIN
   OK:=FALSE;
   WHILE NOT OK DO
   BEGIN
      WRITE(MEMBRETE);
      (*$I-*)READLN(VARENT); (*$I+*)
      OK:=(IORESULT=0);
   END;
END;
PROCEDURE BORRA;
VAR
CONT3  :  ENT;
BEGIN
   CONT3:=1;
   WHILE CONT3<=70 DO
   BEGIN
      VC[CONT3]:=' ';
      CONT3:=CONT3+1;
   END;
END;
PROCEDURE BORVEC(VAR VECTEMA  : ST15);
VAR
   CONT  :  ENT;
BEGIN
   CONT:=1;
   WHILE CONT<=15 DO
   BEGIN
      VECTEMA[CONT]:=' ';
      CONT:=CONT+1;
```

```
    END;
    CONT:=1;
    WHILE CONT<=10 DO
    BEGIN
       VECMOB[CONT]:=' ';
       CONT:=CONT+1;
    END;
    CONT:=1;
    WHILE CONT<=50 DO
    BEGIN
       VECNOM[CONT]:=' ';
       CONT:=CONT+1;
    END;
END;
PROCEDURE TRAAD;
VAR
    CONT   : ENT;
    BAN1   :  BOL;
BEGIN
    WITH VARUSA DO
    BEGIN
    CONT:=0;
    WHILE CONT<LIMUSA DO
    BEGIN
    SEEK(ARUSA,CONT);
    READ(ARUSA,VARUSA);
    IF (USAL1>0) OR (USAL2>0) THEN
    BEGIN
    BORRA;
    WITH VARSAL DO
    BEGIN
       WRITELN(S_NOM);
       VC[1]:='J';
       VC[2]:='_';
       VC[3]:='6';
       VC[4]:='a';
       VC[5]:='1';
       VC[6]:='o';
       VC[7]:='n';
       VC[8]:='(';
       VC[9]:='"';
       SEEK(ARSAL,USAL1);
       READ(ARSAL,VARSAL);
       VECTEM:=S_NOM;
       CONT1:=10;
       CONT2:=1;
       WHILE CONT1<= 24 DO
       BEGIN
          VC[CONT1]:=VECTEM[CONT2];
          CONT1:=CONT1+1;
          CONT2:=CONT2+1;
       END;
       BORVEC(VECTEM);
       VC[25]:='"';
       VC[26]:=',';
       VC[27]:='"';
       SEEK(ARSAL,USAL2);
```

```
        READ(ARSAL,VARSAL);
        VECTEM:=S_NOM;
        CONT1:=28;
        CONT2:=1;
        WHILE CONT1<=42 DO
        BEGIN
            VC[CONT1]:=VECTEM[CONT2];
            CONT1:=CONT1+1;
            CONT2:=CONT2+1;
        END;
        BORVEC(VECTEM);
        VC[43]:='"';
        VC[44]:=',';
        VC[45]:='''';
        VECTEM:=USALI;
        CONT1:=46;
        CONT2:=1;
        WHILE CONT1<=60 DO
        BEGIN
            VC[CONT1]:=VECTEM[CONT2];
            CONT1:=CONT1+1;
            CONT2:=CONT2+1;
        END;
        BORVEC(VECTEM);
        VC[61]:='"';
        VC[62]:=')';
        WRITELN(ARTXT,VC);
        END;
        END;
        CONT:=CONT+1;
    END;
    END;
END;
BEGIN (*RUTINA PRINCIPAL*);
    CLRSCR;
    PINCUA; GOTOXY(11,11);
    WINDOW(10,10,70,20);
    TEXTCOLOR(3);
    APEAR;
    IMOD;
    BORRA;
    BORVEC(VECTEM);
    BAN2:=TRUE;
    WHILE BAN2 DO
    BEGIN
        BAN2:=FALSE;
        (*ACEENT('DE UD EL NUMERO MAXIMO DE DIAS=>',RESDIA);*)
        RESDIA:=10;
        IF (RESDIA=0) OR (RESDIA>10) THEN
        BEGIN
            WRITELN('NUMERO ERRONEO');
            DELAY(LIMDEL);
            BAN2:=TRUE;
        END;
    END;
    WITH VARSAL DO
    BEGIN
```

```
CONT:=0;
WHILE CONT<= LIMSAL DO
BEGIN
   SEEK(ARSAL,CONT);
   READ(ARSAL,VARSAL);
   BORRA;
   IF S_CAP > 0 THEN
   BEGIN
    WRITELN(S_NOM);
    CONM:=1;
    WHILE CONM<=4 DO
    BEGIN
     IF S_TMO[CONM]>0 THEN
     BEGIN
      COND:=1;
      WHILE COND<= RESDIA DO
      BEGIN
      VC[1]:='&';
      VC[2]:='a';
      VC[3]:='l';
      VC[4]:='o';
      VC[5]:='n';
      VC[6]:='(';
      STR(COND:2,VECCVE);
      VC[7]:=VECCVE[1];
      VC[8]:=VECCVE[2];
      VC[9]:=',';
      CONT1:=10;
      CONT2:=1;
      STR(S_TMO[CONM]:4,VECDUR);
      WHILE CONT1<=13 DO
      BEGIN
         VC[CONT1]:=VECDUR[CONT2];
         CONT1:=CONT1+1;
         CONT2:=CONT2+1;
      END;
      VC[14]:=',';
      VC[15]:='"';
      CONT1:=16 ;
      CONT2:=1;
      VECMOB:=VM[CONM];
      WHILE CONT1<=25 DO
      BEGIN
        VC[CONT1]:=VECMOB[CONT2];
        CONT1:=CONT1+1;
        CONT2:=CONT2+1;
      END;
      VC[26]:='"';
      VC[27]:=',';
      VC[28]:='"';
      VECTEM:=S_NOM;
      CONT1:=29;
      CONT2:=1;
      WHILE CONT1<=43 DO
      BEGIN
         VC[CONT1]:=VECTEM[CONT2];
         CONT1:=CONT1+1;
```

```
            CONT2:=CONT2+1;
          END;
          BORVEC(VECTEM);
          VC[44]:='"';
          VC[45]:=')';
          WRITELN(ARTXT,VC);
          BORRA;
          COND:=COND+1;
        END;
        end;
        CONM:=CONM+1;
    END;
    END;
    CONT:=CONT+1;
    BORRA;
END;
BORRA;
CONT:=0;
WHILE CONT<= LIMSAL DO
BEGIN
    SEEK(ARSAL,CONT);
    READ(ARSAL,VARSAL);
    BORRA;
    IF S_CAP > 0 THEN
    BEGIN
        WRITELN(S_NOM);
        COND:=1;
        WHILE COND<= RESDIA DO
        BEGIN
        VC[1]:='h';
        VC[2]:='_';
        VC[3]:='s';
        VC[4]:='a';
        VC[5]:='l';
        VC[6]:='o';
        VC[7]:='n';
        VC[8]:='(';
        STR(COND:2,VECCVE);
        VC[9]:=VECCVE[1];
        VC[10]:=VECCVE[2];
        VC[11]:=',';
        CONT1:=12;
        CONT2:=1;
        STR(S_HIN[COND]:5:2,VECHOR);
        WHILE CONT1<=16 DO
        BEGIN
            VC[CONT1]:=VECHOR[CONT2];
            CONT1:=CONT1+1;
            CONT2:=CONT2+1;
        END;
        VC[17]:=',';
        CONT1:=18;
        CONT2:=1;
        STR(S_DIS[COND]:5:2,VECHOR);
        WHILE CONT1<=22 DO
        BEGIN
            VC[CONT1]:=VECHOR[CONT2];
```

```
                CONT1:=CONT1+1;
                CONT2:=CONT2+1;
            END;
            VC[23]:=',';
            VC[24]:='"';
            VECTEM:=S_NOM;
            CONT1:=25;
            CONT2:=1;
            WHILE CONT1<=39 DO
            BEGIN
                VC[CONT1]:=VECTEM[CONT2];
                CONT1:=CONT1+1;
                CONT2:=CONT2+1;
            END;
            BORVEC(VECTEM);
            VC[40]:='"';
            VC[41]:=',';
            VC[42]:='1';
            VC[43]:=')';
            IF S_HIN[COND]>0 THEN
            WRITELN(ARTXT,VC);
            BORRA;
            COND:=COND+1;
            END;
        END;
        CONT:=CONT+1;
        BORRA;
    END;
    BORRA;
    TRAAD;
    CLOSE(ARTXT);
  END;
END.
```

APENDICE 3: Asignación por medio del sistema experto

Este es un ejemplo de la solución propuesta por el sistema experto para la asignación de una serie de conferencias a salones de un centro de convenciones.

Se presentan los recursos conferencias y salones que se introducen al sistema, su representación en forma de claúsulas para su interpretación por medio de Prolog, así como el reporte que genera el sistema.

RECURSO CONFERENCIAS

No	NOMBRE	TEMA	SUB-TEMA	ASIS-TENCIA	DURA-CION	MOBILIARIO
01	Eficiencia dinamica de sistemas	01		30	3	comité
02	Ingenieria en sistemas de potencia	01		800	1	auditorio
03	Administración de sistemas de potencia	01		15	2	comité
04	Administración de sistemas VAR	01	01	20	2:30	comité
05	Torres, poleas y conductores	02		1200	3	auditorio
06	Métodos analiticos por computadora	03		50	4	comité
07	Planeción de sistemas	01		50	3	escuela
08	Sistemas eléctricos de potencia I	01		70	4	escuela
09	Diseño de subestaciones	01	02	20	3	comité
10	ESMOL I	02		55	3	auditorio
11	ESMOL II	02		30	2:30	escuela
12	Departamento PES	01		40	3	escuela.
13	Sistemas de distribución	02		30	3	comité
14	Switchoo	02		12	2	comité

No	NOMBRE	TEMA	SUB-TEMA	ASIS-TENCIA	DURA-CION	MOBILIARIO
15	Control automático de plantas	01	01	15	3:30	comité
16	Sistemas harmónicos de potencia	02		25	2	comité
17	Sistemas de seguridad	01		25	3	comité
18	NESC	01	02	6	2:30	comité
19	Controles de estabilidad	01		10	3:00	comite
20	Administración en demanda	01		15	3:00	escuela
21	Administración en demanda	01		35	3	auditorio
22	Comida coordinadores	01		32	1:30	alimentos
23	Planeación de sistemas	01	03	10	3	comité
24	Avances en tecnología T&D I	02		30	1	comité
25	Estándares T&D	02		435	1:30	escuela
26	Desarrollo energético	01	02	15	2	comité
27	Sistemas eléctricos de potencia II	01		600	3	auditorio
28	Microprocesadores en subestaciones	03		100	3	auditorio
29	Avances en tecnlogia T&D II	02		300	1	auditorio
30	Poder nuclear	01	02	10	3:30	comité

RECURSO SALONES

NOMBRE	MOBILIARIO	CAPACIDAD	DISPONIBILIDIDAD		
			DIA	HORA INICIAL	HORA FINAL
Molino Rey I	Auditorio	920	T	11:00 am	17:00 pm
	Escuela	480	T	11:00 am	17:00 pm
	Alimentos	660	T	11:00 am	17:00 pm
Molino Rey II	Auditorio	720	T	11:00 am	17:00 pm
	Escuela	360	T	11:00 am	17:00 pm
	Alimentos	440	T	11:00 am	17:00 pm
Oaxaca A	Auditorio	100	T	9:00 am	14:00 pm
	Escuela	50	T	9:00 am	14:00 pm
	Comité	25	T	9:00 am	14:00 pm
Oaxaca B	Auditorio	100	T	9:00 am	14:00 pm
	Escuela	50	T	9:00 am	14:00 pm
	Comité	25	T	9:00 am	14:00 pm
Taxco	Auditorio	70	T	10:00 am	17:00 pm
	Escuela	40	T	10:00 am	17:00 pm
	Comité	20	T	10:00 am	17:00 pm
Del Sol	Auditorio	550	T	10:00 am	17:00 pm
	Escuela	300	T	10:00 am	17:00 pm
	Alimentos	350	T	10:00 am	17:00 pm

RECURSO SALONES UNIDOS

NOMBRE	MOBILIARIO	CAPACIDAD	DISPONIBILIDIDAD		
			DIA	HORA INICIAL	HORA FINAL
Oaxaca A-B =	Auditorio	200	T	9:00 am	14:00 pm
Oaxaca A	Escuela	100	T	9:00 am	14:00 pm
Oaxaca B	Alimentos	50	T	9:00 am	14:00 pm
Molino Rey I-II	Auditorio	1640	T	11:00 am	17:00 pm
Molino Rey I +	Escuela	840	T	11:00 am	17:00 pm
Molino Rey II	Alimentos	1100	T	11:00 am	17:00 pm

Una vez efectuada la captura de los recursos, estos se convierten a predicados en sintaxis de prolog invocando al comando TRADUCE desde el MS-DOS.

A continuación se llama al sistema experto con el comando PARUSE, el cual presenta la siguiente pantalla:

```
-------------------------ASIGNACION DE RECURSOS-------------------------
Archivo de conferencias: TESIS.CON
Archivo de salones: TESIS.SAL
cargando base de datos, espere por favor...
```

```
A = Asignar recursos
R = Recursos no asignados
F = Fin de asignación
Opción: A
Duración del congreso: 3
Tolerancia en capacidad de salones (en porcentaje): 5
```

El usuario introduce al sistema los dos archivos donde están almacenados los recursos conferencias y salones (TESIS.CON y TESIS.SAL). Después de la carga de estos recursos (base de datos), se selecciona la opción A para comenzar la asignación o la R, para ver los recursos que no han sido unificados. En ambos casos el sistema solicita la duración deseada para el congreso en número de días.

La salida generada por el sistema experto para este ejemplo es la siguiente:

RECURSOS ASIGNADOS

DIA	HORA	TEMA	SUBTEMA	NOMBRE	ASIS	MOBILIARIO	SALON	CAPA
1	9.00-12.00	1/SISTEMAS ELECTRICOS		EFICIENCIA DINAM	30	COMITE	OAXACA A-B	50
1	10.00-13.00	1/SISTEMAS ELECTRICOS		ADMINISTRACION E	15	ESCUELA	DEL SOL	300
1	11.00-14.00	2/TRANSMISION Y DISTRI		TORRES POLEAS Y	1200	AUDITORIO	MOLINO REY I-II	1640
1	12.00-14.50	1/SISTEMAS ELECTRICOS	DISTRIBUCION	NECC	6	COMITE	TAICO	20
1	14.00-16.50	2/TRANSMISION Y DISTRI		ESMOL II	30	ESCUELA	DEL SOL	300
1	14.00-17.00	1/SISTEMAS ELECTRICOS		ADMINISTRACION D	350	AUDITORIO	MOLINO DE REY I	920
1	14.00-17.00	3/COMPUTACION		MICROPROCESADORE	100	AUDITORIO	MOLINO REY II	720
1	14.50-16.50	1/SISTEMAS ELECTRICOS		ADMINISTRACION D	15	COMITE	TAICO	20
2	9.00-10.00	2/TRANSMISION Y DISTRI		AVANCES EN TECNO	30	COMITE	OAXACA A-B	50
2	10.00-13.00	1/SISTEMAS ELECTRICOS		CONTROLES DE EST	10	COMITE	TAICO	20
2	10.00-13.00	1/SISTEMAS ELECTRICOS		DEPARTAMENTO PES	40	ESCUELA	DEL SOL	300
2	10.00-13.00	2/TRANSMISION Y DISTRI		SISTEMAS DE DIST	30	COMITE	OAXACA A-B	50
2	11.00-14.00	1/SISTEMAS ELECTRICOS		SISTEMAS ELECTRI	600	AUDITORIO	MOLINO REY II	720
2	13.00-16.00	1/SISTEMAS ELECTRICOS	PLANEACION	PLANEACION DE SI	10	COMITE	TAICO	20
2	13.00-16.00	2/TRANSMISION Y DISTRI		ESMOL I	55	ESCUELA	DEL SOL	300
2	16.00-17.00	1/SISTEMAS ELECTRICOS		INGENIERIA EN SI	800	AUDITORIO	MOLINO DE REY I	920
2	16.00-17.00	2/TRANSMISION Y DISTRI		AVANCES EN TECNO	300	AUDITORIO	MOLINO REY II	720
3	9.00-11.00	2/TRANSMISION Y DISTRI		SWITCHEO	12	COMITE	OAXACA A	25
3	9.00-12.50	1/SISTEMAS ELECTRICOS	ADMINISTRACION	CONTROL AUTOMATI	15	COMITE	OAXACA B	25
3	10.00-13.50	1/SISTEMAS ELECTRICOS	DISTRIBUCION	PODER NUCLEAR	10	COMITE	TAICO	20
3	11.00-13.00	2/TRANSMISION Y DISTRI		SISTEMAS ARMONIC	25	COMITE	OAXACA A	25
3	11.00-14.00	1/SISTEMAS ELECTRICOS		PLANEACION DE SI	50	ESCUELA	MOLINO REY II	360
3	13.00-14.50	2/TRANSMISION Y DISTRI		ESTANDARES IDI	435	ESCUELA	MOLINO DE REY I	480
3	13.50-15.50	1/SISTEMAS ELECTRICOS	DISTRIBUCION	DESARROLLO ENERG	15	COMITE	TAICO	20
3	15.50-17.00	1/SISTEMAS ELECTRICOS		COMIDA COORDINAD	32	ALIMENTOS	DEL SOL	350

CONFERENCIAS NO ASIGNADAS

DIA	TEMA	SUBTEMA	NOMBRE	ASISTENCIA	DURACION	MOBILIARIO
3	COMPUTACION		METODOS ANALITIC	50	4.00	COMITE
1	SISTEMAS ELECTRICOS	ADMINISTRACION	ADMINISTRACION D	20	2.50	COMITE
1	SISTEMAS ELECTRICOS	DISTRIBUCION	DISEÑO DE SUBEST	20	3.00	COMITE
1	SISTEMAS ELECTRICOS		SISTEMAS DE SEGU	25	3.00	COMITE
1	SISTEMAS ELECTRICOS		SISTEMAS ELECTRI	70	4.00	ESCUELA

SALONES NO ASIGNADOS

DIA	HORA	NOMBRE	MOBILIARIO	CAPACIDAD
1	10.00-12.00	TAXCO	COMITE	20
1	12.00-14.00	OAXACA A	COMITE	25
1	12.00-14.00	OAXACA B	COMITE	25
1	12.00-14.00	DEL SOL	ESCUELA	300
1	16.50-17.00	DEL SOL	ESCUELA	300
1	16.50-17.00	TAXCO	COMITE	20
2	11.00-16.00	MOLINO DE REY I	AUDITORIO	920
2	13.00-14.00	OAXACA A	COMITE	25
2	13.00-14.00	OAXACA B	COMITE	25
2	14.00-16.00	MOLINO REY II	AUDITORIO	720
2	15.00-17.00	DEL SOL	ESCUELA	300
2	16.00-17.00	TAXCO	COMITE	20
3	10.00-15.50	DEL SOL	ALIMENTOS	350
3	11.00-13.00	MOLINO DE REY I	ESCUELA	400
3	12.50-14.00	OAXACA B	COMITE	25
3	13.00-14.00	OAXACA A	COMITE	25
3	14.00-17.00	MOLINO REY II	ESCUELA	360
3	14.50-17.00	MOLINO DE REY I	ESCUELA	400
3	15.50-17.00	TAXCO	COMITE	20

BIBLIOGRAFIA

1) Jenkins, Richards A.
"Supercomputers of today and tomorrow",
Tabs Books Inc., 1986.

2) Lenat Douglas B.
"Building expert systems",
Addison-Wesley publishing,1983.

3) Forsyth, Richard.
"Expert systems, Principles and case studies",
Chapman and Hall computing,1984.

4) Sell, Peter S.
"Expert Systems, A practical introduction",
Halsted Press Book, 1985.

5) Rich, Elaine.
"Artificial Intelligence",
International Student Edition, 1983.

6) Feigenbaum Edward A.
"The Fifth Generation",
Addison-Wesley Publishing, 1983.

7) Winston Patrick H. y Prendergast Karen A.
"The AI business. Comercial uses of AI",
1985.

8) Lenat D.
"Eurisko",
1983.

9) Weiss S. y Kulikowski C.
"A practical guide to designing expert systems",
Chapman and Hall, 1984.

10) Soll Peter S.
"Expert Systems a Practical Introduction".
John Wiley and sons, 1985.

11) Knut W. Ernest.
"The brains of men and machines".
McGraw Hill, 1981.

12) Davis R., Lenat D.
"Knowledge-Based systems in artificial intelligence".
McGraw Hill, 1982.

13) Manual de Turbo Prolog.

14) memorias de
"The Second Conference on A. I. Applications".
1985.

15) "How to Make Expert Systems in your Microcomputer".
Tabs Books Inc., 1986.