UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

ELEMENTOS PARA UN SISTEMA EXPERTO DE DIAGNOSIS PRELIMINAR DE FALLAS MAS COMUNES EN TUBERIA DE CALDERA

T E S I S

QUE PARA OBTENER EL TITULO DE

A C T U A R I O

P R E S E N T A :

ROBERTO ALDAVE MATAR

México, D. F. 1989

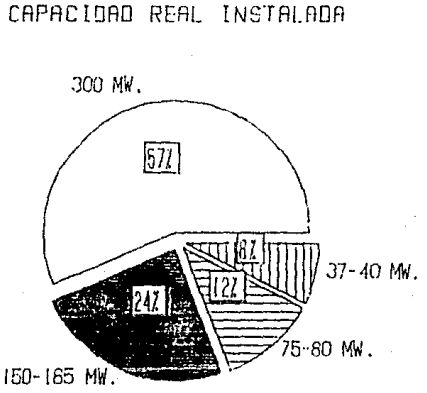
UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

Contenido

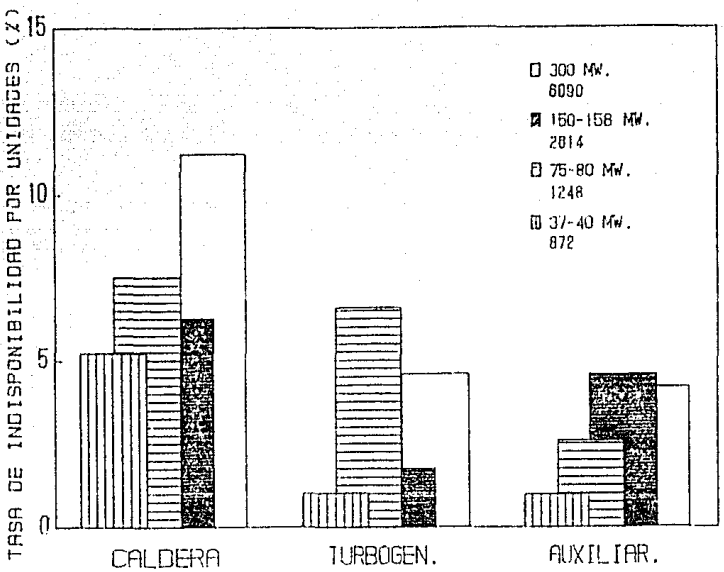
Capítulo 1

¿Para qué un Sistema Experto?

La Comisión Federal de Electricidad (CFE) reporta en su informe de labores 1987-1988 que aproximadamente el 60% de la energía eléctrica es generada por centrales termoeléctricas (CTE). Dichas centrales están formadas por unidades de diferentes capacidades de generación, en las cuales se lleva a cabo la conversión de la energía química almacenada en el carbón, combustóleo o gas a energía eléctrica, las unidades generadoras pueden ser clasificadas de acuerdo a su capacidad; así por ejemplo, hasta 1985 la capacidad real instalada en cada grupo de generación estaba distribuída como se muestra en la siguiente figura:



Los Informes Anuales de Disponibilidad e Indisponibilidad y sus Causas, elaboradas por la Gerencia de Generación y Transmisión de CFE, permiten conocer los principales eventos que conducen a que una unidad sea puesta fuera de servicio o generando menos de lo que debía. Así por ejemplo, la indisponiblidad durante el periódo 1980-1985 estuvo estimada en aproximadamente 7, 16, 12 y 20 % para los grupos de generación 37-40, 75-80, 150-185 y 300 MW, respectivamente. La distribución de la indisponibilidad entre los diferentes subsistemas que conforman una unidad se muestran a continuación.



Esta figura señala que el generador de vapor o caldera es el subsistema que contribuye mayoritariamente a la indisponibilidad de una unidad. Cabe señalar que, para satisfacer la demanda, es necesario substituir la energía no producida debido a fallas. Dicha energía de sustitución es generada por unidades denominadas de turbogas, cuya inversión inicial es más alta que para las unidades convencionales. Además, son más costosas de operar puesto que utilizan diesel, en tanto que las convencionales, como ya se mencionó, usan carbón, gas ó combustóleo.

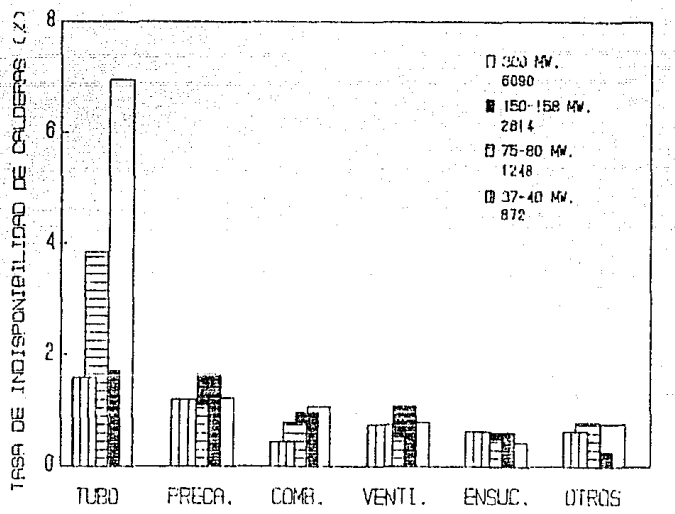
Para tener una idea del costo asociado a la sustitución, notamos que CFE proporciona la siguiente estimación porcentual de costos, en base a unidades convencionales de 350 MW:

COSTO UNITARIO DE GENERACION

CENTRAL	INVERSION	COMBUSTIBLE	OPER. Y MANT.	TOTAL
	Porcentaje	Porcentaje	Porcentaje	Porcentaje
Termo. conv				
(2 x 350)	100	100	100	100
(2 x 160)	120	104	169	109
(2 x 84)	141	107	262	118
(2 x 37.5)	167	112	446	130
Turbogas				
(Gas) (1x30)	263	137	781	177
Turbogas				
(Diesel)(")	270	178	781	210

Tomando en cuenta los costos actuales se estima que la indisponibilidad de una caldera, causada por la falla de un tubo cuesta aproximadamente 500 millones de pesos diarios. Además de las pérdidas económicas que puedan ser estimadas, existen otras que son más difíciles de cuantificar como lo serían aquellas que sufre un hospital o una industria por falta de energía.

Si ahora analizamos la distribución de la indisponibilidad del generador de vapor entre sus diferentes componentes es posible obtener la siguiente figura:



donde "OTROS" incluye componentes, tales como válvulas, colectores, ..., etc, cuya contribución individual es menor al 0.5 % . Esta figura indica que la falla de tubería es una de las causas principales de indisponibilidad.

Aún cuando la tubería es la componente que más fallas presenta, se ha encontrado que el número de mecanismos que pueden producir una falla no es grande, y que muchas de estas fallas son repetitivas. En efecto, al iniciar mi trabajo de tesis revisé aproximadamente 450 reportes sobre fallas analizadas en el Depto. de Combustibles Fósiles del Instituto de Investigaciones Electricas, registradas en el periódo 1981-1987, de los cuales 77 están relacionadas con tubería de generador de vapor. Aproximadamente el 59% de estas fallas ocurrieron en tubería de pared de agua. Las fallas en tubería de sobrecalentador fueron responsables del 24% de las fallas. Los tubos de recalentador y economizador contabilizaron el 12 y 5% respectivamente. Con la información extraída de los reportes generé una base de conocimiento en forma de reglas, para clasificar los mecanismos de falla, lo cual no implica que sean las únicas, ya que en muchas ocasiones los tubos fallados no son enviados para su análisis. Recientemente, fué publicado el "Manual para la Investigación y Corrección de Fallas de Tubería de Caldera" por el Electric Power Research Institute (EPRI), en el cual se describen 22 mecanismos de falla que se han detectado en centrales norteamericanas, así como también las posibles causas que los activan. Dado que muchas de estas causas están relacionadas con la operación de la unidad, es de esperarse, que con el envejecimiento de las mismas y con la degradación que esta sufriendo el combustible, el número de causas y mecanismos de fallas pueda incrementarse.

Para disminuir la ocurrencia de una falla repetitiva es importante determinar el mecanismo que la provoca, de tal manera que puedan tomarse las medidas apropiadas para evitar que se presenten las condiciones que activan dicho mecanismo. Los análisis que conducen a la determinación del mecanismo y las causas que provocaron la falla de un tubo son realizados en laboratorios especializados y en la mayoría de los casos, el tiempo involucrado en el análisis, incluyendo el tiempo de envío tanto del tubo como del reporte, resulta mayor que el tiempo requerido para la reparación y la puesta en servicio de la unidad con lo cual las medidas correctivas pueden no ser tomadas a tiempo. Por otra parte dichos análisis de fallas son conducidos por metalurgistas que son profesionistas que han recibido un entrenamiento adicional de aquel recibido por un ingeniero, un químico ó un físico; lo cual implica no solo una gran cantidad de conocimiento sino también un tiempo largo de entrenamiento.

Lo anterior indica que para disminuir la indisponibilidad de una unidad por falla de tubería, el ingeniero de planta requiere de una herramienta que le proporcione, en base a ciertas observaciones, como datos de operación, análisis simples, ..., etc, un mecanismo preliminar, sugiriéndole las posibles medidas correctivas que pudieran evitar la ocurrencia de la falla. Estas sugerencias, de preferencia debían ser del conocimiento del ingeniero de planta antes de llevar a cabo la reparación, ya que en muchas ocasiones la medida correctiva, o la verificación de la causa de falla, sólo puede lograrse con el generador fuera de servicio. Además, sería conveniente que existiera un mecanismo de retroalimentación el cual permitiera seleccionar la medida correctiva más adecuada de acuerdo a las condiciones de la falla.

Por lo tanto, el propósito de esta tesis es el desarrollar algunas herramientas que conformen el sistema antes mencionado. Para este propósito, en el siguiente capítulo se discutirán algunas alternativas concluyéndose que, a largo plazo, un sistema experto es lo más conveniente. Posteriormente en el Capítulo III, se muestra, a través de ejemplos, la implementación del módulo de adquisición de conocimiento desarrollado en este trabajo, el cual permitirá generar reglas a partir del conocimiento en los reportes de análisis de falla.

Capítulo 2

¿Qué es un Sistema Experto?

2.1 Introducción

Para presentar información acerca de la problemática de fallas en tubería de caldera que pueda ser fácilmente usada por personal de planta para determinar rapidamente un mecanismo de falla, el sistema desarrollado en esta tesis contempló los siguientes aspectos.

1. Se requiere de evidencia, la cual consiste de un conjunto de hechos acerca de las condiciones del tubo fallado, obtenida mediante una interacción entre el usuario y el sistema.
2. Se necesita un conocimiento específico en términos de relaciones entre cada tipo de falla y las condiciones en las que se presenta cada una.
3. Contar con un procedimiento que haga corresponder su conclusión, diagnóstico, a una serie de respuestas dadas por el usuario.
4. En el caso de que no se haya brindado un diagnóstico, debido a que el conocimiento actual del sistema es insuficiente, el sistema debe dar la facilidad de adquirir más información sobre el tubo fallado de tal manera que sea capaz de relacionarla con un conjunto de información que sustenta el conocimiento específico y poder generar posiblemente nuevo conocimiento.
5. Además, si se llegó a un diagnóstico el sistema debe ser capaz de dar consejos acerca de que medidas correctivas serán tomadas para prevenir la futura ocurrencia del mismo tipo de falla.

La consideración de no utilizar elementos proporcionados por sistemas de cómputo tradicionales se debe a que emplean programas organizados jerárquicamente. Una dificultad de este tipo de programas se presenta al modificar la base del conocimiento particular, ya que podrían requerirse de cambios extensivos a varios de los programas existentes, estructuras de datos y organizacion de subrutinas [16]. El diseño presente en los sistemas expertos es mucho más modular, y cambios a la base de conocimiento pueden hacerse de una manera relativamente independiente.

2.2 Sistemas Expertos

Los sistemas expertos (SE) son programas de cómputo que pueden en algún sentido razonar, esto es, tratan de emular el proceso de razonamiento de un experto humano con el objeto de resolver problemas de manera inteligente en una área específica del conocimiento.

Al hablar de inteligencia en un sistema experto es conveniente mencionar los siguientes aspectos:

1. Los expertos humanos son personas que por sus años de experiencia en una especialidad dada, producen resultados de alta calidad en un tiempo mínimo, haciendo uso de heurísticas y modelos de inferencia de alto nivel. Hay que resaltar que a estas personas son las que trata de emular un SE [13].

2. Cuando uno se refiere al conocimiento en un sistema experto, debe entenderse como la información que el sistema necesita antes de mostrar un comportamiento inteligente, dicho comportamiento radica en el hecho de que el sistema debe de construir su solución selectiva y eficientemente de un espacio de alternativas [13].

El conocimiento que manejan los expertos humanos es de dos tipos: uno que puede llamarse público y otro privado. El primero es el conocimiento que podemos encontrar en los libros o cualquier material impreso o de otro tipo y es aquel que pertenece al dominio público. El segundo, es el conocimiento adquirido por el especialista de una manera empírica y a través de su experiencia cotidiana al ejercer su disciplina. Este conocimiento no se haya en los libros ni es del dominio público, esto es precisamente lo que uno está pagando cuando se consulta a un especialista, pues bien en el trabajo de SE el esclarecimiento y reproducción de tal conocimiento son consideradas las tareas centrales.

Así podemos decir que los SE manipulan conocimiento para resolver de manera eficiente y selectiva un problema en un área particular del conocimiento humano, el cual consiste de información acerca de un dominio particular, comprensión de los problemas del campo o área y una pericia o habilidad en resolver algunos de esos problemas.

2.3 Estructura de un Sistema Experto

2.3.1 Partes Fundamentales e Interfaces

Para describir la estructura de un sistema experto, conviene hacer notar que en la búsqueda de la solución de un problema específico, en general se encuentran involucrados dos clases de conocimiento, el particular y el general. La primera clase engloba el conocimiento representado por teorías, hipótesis, creencias y descripciones que de alguna manera intervendrán en la solución del problema específico; en tanto que la segunda engloba los métodos, reglas heurísticas, procedimientos, y las metareglas, es decir conocimiento acerca de reglas específicas en la base de conocimiento ó bien acerca de la estructura de la misma base, que permiten la manipulación e interpretación adecuada del conocimiento específico para llegar a una solución. En un sistema experto, la clase de conocimiento particular se encuentra contenido en lo que se llama *base de conocimiento*, en tanto

que la clase de conocimiento general forma lo que se conoce con el nombre de *máquina inferencial*. El alcance de un SE radica en el poder que la base de conocimiento conlleve para arribar a una solución; este poder consiste en que el conocimiento que se vaya incorporando a la base sea especializado y completo de manera tal que cuando sea manipulado por la máquina inferencial produzca una solución confiable en el menor tiempo.

Por otro lado, en la máquina inferencial se encuentra el control del programa; esto es, es la encargada de aplicar operadores para manipular la base de conocimiento de manera tal que cada aplicación a un hecho o evento es visto como una operación que se ejecuta cuando se intenta razonar o resolver un problema. En otras palabras la máquina inferencial decide cuál, cuándo y dónde aplicar un operador dependiendo del estado actual en el que se encuentra el proceso de búsqueda de solución.

El *contexto* es aquella parte del sistema donde se van colocando los hechos y evidencias acumulados durante una consulta. El contexto es utilizado por la máquina inferencial para llevar a cabo la toma de decisiones. Revisa qué información tiene, qué tan cierta es y qué se ha demostrado hasta ese momento.

Hay otras tres partes que constituyen a un SE, aunque no son fundamentales como las tres anteriores, sí son facilidades para ayudar al usuario en la interacción con el SE.

Una de estas partes es un *módulo para la adquisición de conocimiento* que consiste en una interfaz entre el ingeniero de conocimiento, persona encargada de extraer el conocimiento adecuado a un experto humano, y la base de conocimiento. De hecho es un editor altamente especializado cuya función principal es capturar la información exterior proporcionada por el ingeniero de conocimiento, para después almacenarla adecuadamente en la base de conocimento. Además permite modificar o eliminar la información que se encuentra en la base de conocimiento.

El *módulo de asistencia e instrucciones* es una interfaz que le da al usuario la facilidad de contar con información e instrucciones de ayuda en caso de requerirla en algún momento de la sesión de consulta, por ejemplo puede explicar al usuario porque una inferencia específica es hecha por la máquina de inferencias, o bien, explicar los hechos, reglas y procesos de decisión que fueron usados por la máquina de inferencias.

El último es un *módulo de interfaz* entre el usuario y el SE. Su función es la de aceptar y reconocer un lenguaje de comandos del usuario y traducirlos a instrucciones para un SE.

2.4 Construcción de un Sistema Experto

La acumulación y codificación de conocimiento es uno de los aspectos más importantes de un sistema experto [21].

El conocimiento para un sistema experto puede adquirirse de varias maneras, cada una involucra transferir la pericia de una fuente de conocimiento a un programa, la cual se necesita para alcanzar una alta ejecución en la resolución del problema en un área particular o dominio. La pericia que se intenta elucidar en este trabajo de tesis consistirá en establecer un conjunto de relaciones entre tipos de fallas y condiciones del tubo fallado, acerca de la problemática de fallas en tubería de caldera. Se menciona en [13], que la fuente puede ser ya sea un experto, o cualquier otra mediante la cual pudiese haber adquirido su conocimiento el experto, por ejemplo artículos

especializados, revistas etc. Este proceso de transferencia y transformación de conocimiento de una fuente a un programa, el cual es conocido como *adquisición de conocimiento* puede ser realizado por un ingeniero de conocimiento (IC) o un programa. En el primer caso, el IC entrevista a uno o más expertos del dominio para extraer su conocimiento; primeramente definen el problema a ser atacado, descubren los conceptos básicos involucrados y desarrollan reglas que expresan las relaciones existentes entre los conceptos, también especifican estrategias o reglas heurísticas que constituirían el conocimiento general para resolver problemas. Después el IC debe de elegir el lenguaje o la herramienta para representar el conocimiento formalizado del área, en estructuras de datos conceptualmente simples dentro del esquema o marco del lenguaje [13], de tal manera que pueda ser manipulado e interpretado por las estrategias que constituyen el conocimiento general para resolver problemas.

En cuanto a la adquisición del conocimiento por medio de un programa, este puede ser de diferentes tipos de acuerdo al grado de automatización presente. Lo anterior será tratado más adelante en el capítulo III, en un intento por describir las características contenidas en el sistema.

2.5 Representación del Conocimiento

La representación del conocimiento es una de las áreas más importantes a considerar en la construcción de un SE. La forma que se elija para describir las propiedades de los objetos y eventos involucrados en el área de interés debe de considerar el grado de detalle de estos conceptos, ya que de esto depende en gran parte la eficiencia de un SE, al manipular estas estructuras para procedimientos inferenciales, de interpretación etc.

Como se mencionó en la introducción se cuenta con un conjunto de relaciones que asocian tipos de falla con características visuales del tubo fallado. A continuación daremos un ejemplo a manera de ilustrar el tipo de las asociaciones anteriores,

La falla es termofluencia si:

- La localización del tubo es tubería de sobrecalentador.
- El tubo presenta una deformación abombada.

La cual constituye una regla de la forma si p entonces q, donde p es el antecedente y q el consecuente. Los sistemas expertos en los cuales el conocimiento del área es representado por un conjunto de reglas llamadas producciones, se les conoce precisamente con el nombre de sistemas basados en reglas. Dichas producciones son probadas contra un conjunto de hechos o conocimiento de la situación actual. En nuestro caso esta información es obtenida a través de un proceso interactivo con el usuario. Un manejador de reglas se encarga de determinar qué regla satisface los hechos, y ejecuta la acción determinada en su conclusión, más adelante ilustraremos con ejemplos, como trabaja el mecanismo de control que manipula a nuestro conjunto de reglas.

El conocimiento presente podría ser llevado sin mayores problemas a una estructura de datos sencilla soportada por algún lenguaje, sólo hay que observar que cada condición de la regla se puede poner en términos de un calificador o atributo y su respectivo valor, por ejemplo un calificador sería la deformación y su valor respectivo presente en la regla, sería abombado.

En PROLOG, lenguaje elegido para la construcción del presente sistema, se podría tener una estructura de datos como la siguiente:

```
falla(termofluencia,   [a(posición del tubo,sobrecalentador),
                        a(deformación,abombado)])
```

el cual es un objeto llamado falla que representa a la regla, mostrándonos la relación existente entre su conclusión, termofluencia, y las condiciones en las que se presenta, representadas por los elementos de la lista. Cada elemento de la lista expresa la relación entre un atributo o calificador y su valor.

No es la única estructura que podría utilizarse para representar el concepto anterior, por ejemplo una manera alternativa sería,

```
falla(termofluencia,   [sobrecalentador,abombado])
      atributo         ([posición del tubo,sobrecalentador])
      atributo         ([tipo de deformación,abombado])
```

2.6 Estrategias de Control

La otra parte importante de un SE, como se indicó anteriormente es su motor o máquina de inferencias, la cual basa su funcionamiento en algunas de las estrategias de razonamiento que ya existen. La idea en una estrategia de control es la de alcanzar una meta aplicando una sucesión adecuada de operadores a una situación inicial del dominio. Cada aplicación de un operador modifica la situación de alguna manera [3]. Hay un esquema general para las estrategias de control donde al aplicarse un operador se tiene una descripción general de la situación o estado, el cual es planteado como un proceso de búsqueda, cuyo objetivo es aplicar operadores hasta encontrar alguna sucesión que produzca un estado que satisfaga una condición de meta. En este contexto una meta es una descripción de la solución desada, y el conjunto de estados posibles que llevan de una condición inicial a una condición de meta es visto como el espacio de búsqueda.

Los estados en el espacio de búsqueda frecuentemente son representados por nodos. Cada nodo es producido por la aplicación de un sólo operador, representándose esta operación como una liga o arco entre los nodos. Subsecuentes aplicaciones de operadores producen sucesores de esos nodos y así sucesivamente. A este tipo de estructura se le conoce con el nombre de gráfica. La resolución del problema es llevada buscando a través del espacio posible de soluciones a un nodo que satisfaga una condición de meta. A este proceso se le conoce como búsqueda en gráficas.

Existen un tipo especial de gráficas llamadas árboles, donde cualquier nodo tiene a lo más un antecesor, dentro de los cuales hay un nodo llamado raíz que no tiene antecesor y que describe la situación inicial, además existen un conjunto de nodos terminales los cuales no tienen sucesores. Como puede verse los árboles son el resultado de búsquedas en gráficas.

El problema de producir un estado que satisfaga una condición de meta puede ahora ser formulado como el problema de buscar una gráfica para encontrar un nodo cuya descripción de estado asociado satisfaga la meta [3].

Si bien vimos a los árboles como el resultado de búsquedas en gráficas, también hay árboles de búsqueda. La diferencia principal es que los últimos árboles mencionados son explícitos, es decir, se construyen cuando la búsqueda procede; los únicos nodos incluídos son aquellos que tienen

asociada una sucesión de nodos conectados por arcos desde el nodo raíz a él, y que acaban de ser desarrolladas en el proceso de búsqueda.

En contraste la gráfica a ser buscada es ordinariamente no explícita, la cual podría pensarse de tal manera de tener un nodo para cualquier estado posible en el espacio de búsqueda, y tal que haya una sucesión de arcos y nodos que lo conecten con el nodo raíz.

Hay varias maneras por las cuales puede ser conducida la búsqueda a una solución, dependiendo de la orientación que se le dé en el árbol, ya sea aplicando el operador a una situación actual del problema, o a una condición de meta. A continuación se mencionan a dos de ellas, y se dan sus características generales.

2.6.1 Razonamiento hacia Adelante

En este mecanismo de razonamiento los operadores se aplican a un nodo que describa una situación actual del problema, produciéndose entonces una situación diferente o modificada. El objetivo es llevar la situación adelante de su configuración actual a una que satisfaga una meta. En el capítulo III daremos un ejemplo para ilustrar este concepto, cuando describamos un programa de inducción referido a una modalidad de adquisición de conocimiento.

2.6.2 Razonamiento hacia Atrás

En este tipo de estrategia se tiene otro tipo de operador el cual es aplicado a un nodo que describe una situación de meta, en vez de aplicarlo a la situación actual del problema. De esta manera, la búsqueda es llevada a otro nodo que puede verse como otra submeta, la cual se espera sea más fácil de resolver que la anterior, la búsqueda continúa de esta misma manera hasta encontrar un nodo que junto con los anteriores describan condiciones suficientes para satisfacer la condición de meta inicial.

A continuación daremos el siguiente ejemplo:

Consideremos un caso sencillo donde se tiene una sola regla en la base de conocimiento, la cual es representada por el siguiente formato :

falla(termofluencia, [a(posición del tubo,sobrecalentador),
a(deformación,abombado),a(fractura,boca de pescado)])

Una estrategia de razonamiento hacia atrás empezaría con lo que quiere probar, es decir con la causa de falla termofluencia.

Como primer paso selecciona dicha regla, y decide que debe establecer la posición del tubo, tipo de deformación y tipo de fractura para concluir la causa de falla. Se liga cada nodo representando a los atributos con la causa de falla, como podemos apreciar en la figura 2.1.

Como segundo paso el mecanismo encuentra que la posición del tubo es sobrecalentador, y trata de establecer interactuando con el usuario, si es cierto o no que el tubo fallado está en tubería de sobrecalentador. Se pone una liga explícita entre los nodos que representan al calificador posición del tubo y su valor respectivo sobrecalentador.

En el paso 3, el mecanismo encuentra que el tipo de deformación es abombado, pone una liga explícita entre los nodos representando al tipo de deformación y abombado, trata de establecer de manera análoga al paso 2 la validez de esta liga.

En el paso 4 una vez más se ligan los nodos asociados a las descripciones de tipo de fractura y su valor boca de pescado y, se establece si el tipo de fractura es boca de pescado.

Si se determinó la validez de cada uno de los conceptos involucrados, se concluirá que la causa de falla es termofluencia. En caso contrario el conocimiento presente en la base de conocimiento es insuficiente para llegar a una conclusión.

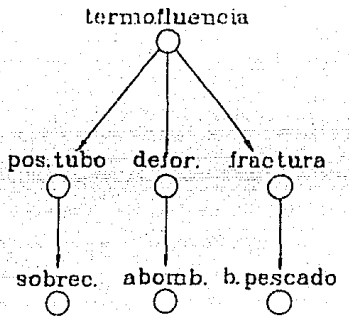


Fig. 2.1 Un ejemplo para ilustrar el proceso de busqueda hacia atrás.

2.7 Lenguajes para construir Sistemas Expertos

Para construir e implementar un sistema basado en conocimiento, se tiene que hacer una selección cuidadosa del lenguaje o herramienta de programación más adecuada para llevarlo a cabo. Estas herramientas se pueden agrupar bajo tres categorías.

2.7.1 Lenguajes Imperativos

Los programas elaborados en este tipo de lenguajes instruyen a la máquina la serie de pasos a seguir para resolver un problema. Esto se debe a que la base de diseño de estos lenguajes son las computadoras convencionales.

Estos lenguajes son entonces principalmente aplicados a problemas que requieren de un algoritmo para su solución, por ejemplo en áreas científicas, estadísticas, matemáticas. Entre los lenguajes de este tipo se enuentran FORTRAN, PASCAL, C.

2.7.2 Lenguajes Funcionales

Los lenguajes de este tipo están basados en propiedades de las funciones matemáticas. La esencia de programación de estos lenguajes es combinar funciones para producir funciones más potentes. Esta característica es importante, ya que a partir de un objeto de datos simple y uniforme, por ejemplo listas o arreglos, se pueden construir estructuras de datos, las cuales sirven como argumentos de las funciones, las cuales al aplicarse producen otras estructuras, teniéndose de esta manera un manejo de memoria dinámico; creándose o eliminándose objetos de datos. Como ejemplo de este tipo de lenguajes esta LISP (*List Programming*), cuya única estructura de datos son las listas.

Que un lenguaje proporcione un manejo dinámico de memoria es importante para la construcción de un SE, en vista de que la máquina inferencial de un SE, como se mencionó, revisa una memoria de trabajo o contexto en la cual se van acumulando los hechos de una consulta. Lo anterior hace posible generar nuevo conocimiento a partir de la nueva experiencia presentada, si se cuenta con una máquina inferencial para estos fines.

2.7.3 Lenguajes Declarativos

La escencia de los lenguajes declarativos, esta más en describir hechos conocidos y relaciones acerca de un problema, que en preescribir la sucesión de pasos tomados por una computadora para resolver el problema.

Entre los lenguajes declarativos encontramos a PROLOG (*PROgramming in LOGic*), el cual posee una estructura de datos uniforme llamada *term*, con la cual se construye todo dato, al igual que un programa. Se mencionó en la sección anterior que en Lisp la única estructura de datos es la lista, sin embargo en Prolog, la lista es simplemente un tipo particular de estructura.

En Prolog también se forma una base de datos dinámica, formada por aplicaciones de estructuras llamadas predicados a los hechos.

Un programa en Prolog consiste de un conjunto de cláusulas. Cada cláusula es un hecho acerca de la información dada o una regla que especifica como la solución puede relacionar o ser inferida por los hechos. Programar en PROLOG consiste en:

- Declarar un conjunto de claúsulas (hechos o reglas) que nos describen como se relacionan los objetos.
- Preguntar acerca de estos objetos y sus relaciones.

Por otra parte hay una estrategia poderosa y muy frecuentemente usada en los procesos de búsqueda en árboles, por las máquinas inferenciales de SE's llamada *recursividad*, no todos los lenguajes soportan este mecanismo, tal es el caso de FORTRAN lenguaje imperativo, sin embargo Lisp y Prolog soportan tal característica.

Una estrategia de búsqueda muy útil también construida internamente en PROLOG es el *backtracking*. El backtracking es importante para implementar búsquedas en árboles, ya que por ejemplo, cuando la búsqueda procede a través del espacio de búsqueda, se van ligando explícitamente los nodos por medio de arcos; si en algún momento del proceso se establece la no validez de un nodo,

el backtracking le permite a Prolog regresar automáticamente a un nodo antecesor para intentar encontrar una solución alternativa, de esta manera no se sigue extendiendo la rama a través del nodo inválido, en este sentido la búsqueda se hace más óptima. Este hecho será ilustrado en el capítulo III cuando se describa a una componente de la máquina inferencial del sistema desarrollado en la tesis.

Lisp no tiene inmerso un mecanismo de backtracking, aunque se podría programar, sin embargo Prolog nos ahorra ese trabajo.

El lenguaje elegido para el desarrollo del presente sistema fué Prolog, ya que posee estructuras y mecanismos que lo hacen ser un lenguaje de más alto nivel que Lisp para implementar sistemas expertos.

2.7.4 Escenarios de Sistemas Expertos con dominio independiente

Estos escenarios, también llamados shells o esqueletos de sistemas expertos, proporcionan un sistema constructor con un motor de inferencias al cual se le pueden añadir las bases de conocimiento sobre temas específicos. Estos sistemas también proporcionan un editor para adquisición de conocimiento y un módulo de asistencia para simplificar la construcción de los sistemas expertos. Como puede verse su única limitante es que el usuario sólo puede manejar problemas que pueden ser representados en la forma que tenga ya definida el escenario, reduciendo grandemente las opciones de diseño del constructor del sistema experto. Como ejemplos de estos lenguajes están, EMYCIN, EXPERT, KAS.

2.7.5 Lenguajes de Representación de Propósito General

Los lenguajes de representación de propósito general son lenguajes más especializados que los de manipulación simbólica como Lisp o Prolog. Han sido desarrollados para facilitar la implementación de una amplia gama de problemas en el espacio de *derivación-formación*. Proveen mayor control sobre el acceso y búsqueda de información que los que proporcionan el anterior tipo de lenguajes, pero podrían resultar más difíciles de usar [13]. Estos lenguajes de propósito general varian en gran manera según la extensión de su generalidad y flexibilidad, como ejemplos Knowledge Kraft, LOOPS, KEE, ART.

De estas dos últimas clasificaciones diremos que uno de los objetivos del trabajo es desarrollar un shell propio a mediano plazo con ciertas características que se describirán en el siguiente capítulo, siendo Prolog el lenguaje adecuado para estos fines.

2.8 Aplicaciones de Sistemas Expertos

En la referencia [2] se menciona, que se ha propuesto que los tipos de problemas que un sistema experto puede resolver, caen dentro de dos posibles clases: problemas de *derivación* y problemas de *formación*. La mayor parte de los problemas del mundo real caen dentro de estas dos categorías y, posiblemente sean una combinación de ambas.

2.8.1 Problemas de Derivación

En los problemas de derivación la solución existe en la base de conocimiento, y la solución a este tipo de problemas implica la identificación de la ruta de solución adecuada por parte del mecanismo de inferencias.

Dentro de estos tipos de problemas podemos encontrar:

(a) *interpretación* (datos ⇒ significado)

(b) *diagnósis* (síntomas ⇒ fallas)

(c) *reparación* (fallas ⇒ solución)

(d) *monitoreo* (señales ⇒ interpretación y alarmas)

(e) *control* (señales ⇒ regulación)

2.8.2 Problemas de Formación

En los problemas de formación las condiciones del problema se expresan como restricciones a cumplir por la solución. El mecanismo de inferencia debe generar o construir una solución usando el conocimiento contenido en la base de conocimiento.

Dentro de los problemas de formación se encuentran

(a) *plancación* (objetivo ⇒ secuencia de acciones)

(b) *diseño* (requerimientos ⇒ sistema u objetivo)

El módelo lógico estará determinado por la aplicación específica, así por ejemplo para el problema de interpretación, dados los datos actuales presentes en el problema; se intentará encontrar un conjunto de condiciones que los satisfagan lo cual nos llevará a tener un significado de la información anterior, teniéndose en este caso un razonamiento hacia adelante.

Capítulo 3

Lo que hice para el Sistema Experto

3.1 Introducción

En este capítulo intentaremos describir la estructura del sistema desarrollado en este trabajo de tesis. Para tener un marco de desarrollo de sus características, nos referiremos más acerca del proceso de adquisición de conocimiento.

Mencionábamos en la sección 2.4 , que la translación del conocimiento de una fuente a un programa podría realizarse a través de un ingeniero de conocimiento o un programa. La primera de estas modalidades ya fué ahí discutida. Respecto a la segunda el presente sistema cuenta con dos programas, dependiendo si la fuente de conocimiento es un experto humano o bien datos iniciales o empíricos, y que constituyen nuestro primer acercamiento al modo de adquisición de conocimiento mediante un programa. A continuación presentaremos a dos de sus posibles alternativas.

En la primera de ellas, el programa es un editor a través del cual se intenta una conversación más directa del experto humano con el sistema experto. Su función principal es capturar el conocimiento del experto en forma de reglas y almacenarlas adecuadamente en un archivo. Para facilitar la captura de la información debe tener una amplia capacidad de dialogar con el experto. Además puede mostrar información sobre la estructura de las reglas.

El programa con que cuenta actualmente el sistema es sencillo, y será descrito brevemente en la próxima sección.

Por otra parte, para que una persona llegué a ser experta en algún área particular del conocimiento se necesita que vaya adecuando su conocimiento con relación a las experiencias que resulten del ejercicio diario de su disciplina. En la referencia [13], se dice, "hay razón para esperar que un programa de inducción pudiera construir una base de conocimiento para un sistema experto de manera similar".

El programa de inducción con que cuenta el sistema desarrollado en esta tesis, produce reglas directamente de datos, utilizando una función que emplea el algoritmo ID3, el cual es frecuentemente utilizado por SE's comerciales . Conviene hacer notar que se necesita un mecanismo que dé soporte a la credibilidad de la información inmersa en este tipo de adquisición de conocimiento, al menos en nuestro programa, para evitar conclusiones inconsistentes.

3.2 Interacción y Descripción de los Elementos del Sistema

3.2.1 Interacción

En la Figura 3.1, están presentes los elementos que conformarían lo que consideramos como un SE completo. Los programas representados por rectángulos en líneas punteadas no se han desarrollado todavía, sin embargo, se presentan con las ya existentes para tener una concepción más clara y amplia del sistema propuesto.

Como se ilustra en la Figura hay dos posibles entradas al sistema. En caso de que la persona sea un experto, entonces el sistema captura la información y la almacena en la base de conocimiento BC "A". Aquí se emplea el programa que llamaremos Captura I, el cual ofrece además la facilidad de modificar esta base de conocimiento; aumentando o borrando una regla. Entre otras cosas verifica que dos reglas con conclusiones diferentes no tengan asociadas el mismo conjunto de condiciones.

Por otro lado, si la persona es un usuario entonces el sistema entiende que realizará una consulta a la base de conocimiento BC "C" a través del programa Encuentra, el cual intentará dar un diagnóstico siempre y cuando la base de conocimiento no este vacía. Si hubo diagnóstico, el siguiente paso se realiza mediante el programa Verifica, el cual trata de corroborar el diagnóstico dado por el programa Encuentra, presentando transparencias, o fotos que muestren gráficamente las características de un tubo fallado, de acuerdo a las respuestas dadas por el usuario en su consulta a la base de conocimiento; otra alternativa es hacer preguntas extras, en base a la trayectoria de la solución encontrada, para reforzar la caracterización del tubo fallado.

Si no se verifica el diagnóstico, entonces la acción tomada es la misma que se tendría en caso de que el programa Encuentra no hubiera dado un diagnóstico, la cual es preguntar al usuario si tiene mayor información sobre el tubo fallado, que pudiera ser procesada a la base de conocimiento BC "B". En caso afirmativo se captura la información a través del programa Captura II, luego pasa está a través del programa Soporte, que se describe más adelante, el cual ajusta una base de datos tomando en cuenta la nueva evidencia o información, antes de ser alimentada al programa Genera, el cual produce reglas directamente de los datos reajustando la base de conocimiento BC "B". Hasta aquí se tienen dos bases de conocimiento, las cuales deben ser comparadas y contrastadas por el programa Completez para verificar que no existen inconsistencias y redundancias entre ellas; produciéndose con esto una base de conocimiento final BC "C" . En el caso de no contar con más información entonces se termina con la sesión al sistema.

Si el programa Verifica proporcionó un diagnóstico, entonces el sistema debe tener la habilidad de explicar su propio mecanismo de razonamiento acerca de como llegó a su conclusión, ofreciendo de esta manera la posibilidad de entrenamiento a través del programa Enseñanza. Después de esto se pregunta si se desea otro diagnóstico o si se tiene información sobre otro tubo fallado,en caso de no darse ninguna de estas alternativas se finaliza la sesión.

Si relacionamos la arquitectura presente en el esquema de la Fig. 3.1, con la estructura clásica de un SE se puede hacer la siguiente asociación:

Hay presentes en el sistema dos módulos de adquisición de conocimiento, el primero dado a través del programa Captura I, y el otro por por medio del subesquema que encierra a los programas, Captura II, Soporte, a la base de datos y a Genera. De lo anterior resultan dos bases de conocimiento intermedias hacia una final.

Los componentes de la máquina inferencial son el programa Encuentra que consulta la base de conocimiento. El programa Verifica y Enseña actuarían como el módulo de asistencia. El programa Soporte le permite a una componente de la máquina inferencial, el programa Genera, trabajar con datos incompletos y/o inconsistentes. Por último Completez implementaría la base final a partir de las intermedias.

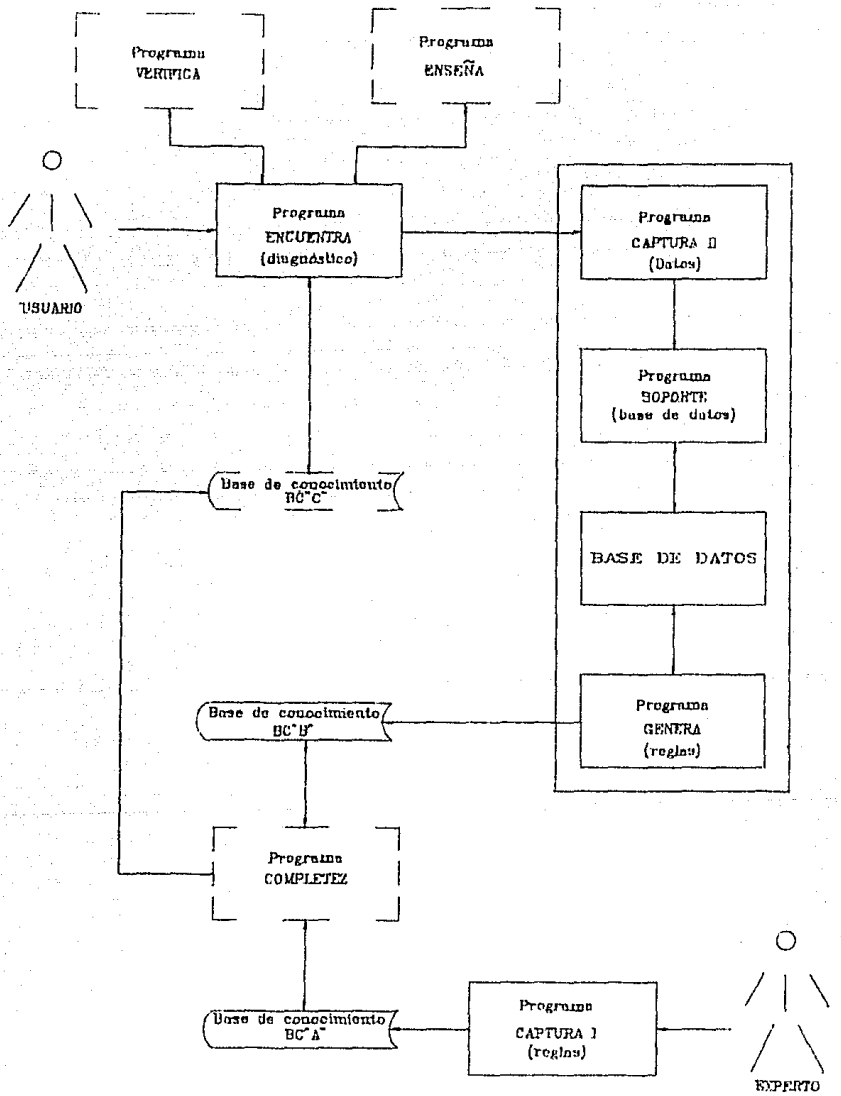


Fig. 3.1 Interacción de los elementos del sistema experto propuesto.

3.2.2.1 Base de Conocimiento

Como se mencionó anteriormente y como se puede observar en la Fig. 3.1, la base de conocimiento es implementada en los posibles procesos que ha continuación se describen.

Proceso A.

En este proceso el conocimiento es dado por el experto del área en forma de reglas. La única función del sistema es capturarlo y almacenarlo adecuadamente, a través del programa Captura I a una base de conocimiento, que llamamos BC "A". La manera en que se representó el conocimiento del experto en esta parte se ejemplifica como sigue:

```
arco_dbase  (n("la_posición_del_tubo","pared_de_agua"),
            ["pared_de_agua"])
arco_dbase  (n("tipo_de_deformación","elongado"),
            ["presenta_elongación"])
arco_dbase  (n("tipo_de_fractura","boca_de_pescado"),
            ["abertura_ancha","a_lo_largo_del_tubo"])
arco_dbase  (n("tipo_de_bordes","bordes_delgados"),
            ["el_espesor_de_la_pared_del_tubo_esta_adelgazado"])
arco_dbase  (n("tipo_de_bordes","bordes_delgados"),
            ["en_forma_de_cuchilla","adelgazados"])
      falla (["sobrecalentamiento",
            "pared_de_agua","elongado",
            "boca_de_pescado","bordes_delgados"])
arco_dbase  (n("tipo_de_deformación","ligero_abombamiento"),
            ["presenta_un_ligero_abombamiento",
            "en_forma_de_chipote"])
arco_dbase  (n("tipo_de_fractura","picaduras"),
            ["presenta_picaduras"])
arco_dbase  (n("tipo_de_fractura","picaduras"),
            ["pequeñas_perforaciones_tipo_poro"])
      falla (["corrosión_cáustica",
            "pared_de_agua","ligero_abombamiento","picaduras"])
```

en el cual encontramos a dos reglas. Cada regla se describe a través de un objeto llamado falla, al cual lo componen , como primer elemento la causa de falla, es decir, la conclusión de la regla, y como segundo elemento una lista de características, es decir hechos o condiciones bajo las cuales se presenta la falla, que vienen a ser las condiciones de la regla.

Hay que observar que cualquier característica asociada a una falla tiene un calificador o atributo. Esta relación es expresada por un objeto llamado arco_dbase a través de su primer componente, por ejemplo para la característica boca de pescado presente en la falla sobrecalentamiento se tiene

arco_dbase(n(tipo_de_fractura,boca_de_pescado)), [abertura_ancha,a_lo_largo_del_tubo])

donde n(tipo_de_fractura,boca_de_pescado)

expresa la relación atributo-valor, es decir, el tipo de fractura es el atributo de la característica o valor: boca de pescado. La lista presente en la segunda componente describe la relación anterior permitiendo una descripción mas completa de la misma. La lista de cada arco_dbase se empleará para hacer preguntas al usuario acerca de la verificación de los hechos o características del tubo fallado para intenter dar un diagnóstico; haciendo corresponder sus respuestas a alguna falla presente en las conclusiones de las regla. Este proceso será descrito más adelante en este capítulo.

Proceso B.

Aquí el sistema captura la información, consistente de reportes por medio del programa Captura II. Con dicha información se procede a ajustar una base de datos, a través del programa Soporte, el cual se describe en la próxima sección, para ser alimentada a un proceso inferencial que produce reglas. Conformándose una segunda base de conocimiento que llamamos BC "B".

El procesamiento en forma de reglas de datos iniciales es llevado a cabo por el programa Genera, empleando una potente función discriminadora para generar reglas directamente de datos. En el CAPITULO IV, nos ocuparemos de describir detalladamente este algoritmo, y comprobar su formulación.

Para ilustrar la base BC "B" consideremos el siguiente conjunto de reglas procesadas por Genera:

atributos	([”la_posición_del_tubo”, ”pared_de_agua”,”sobrecalentador”])
atributos	([”tipo_de_deformación”,”elongado”,”abombado”])
lis_nom	([”la_posición_del_tubo”,”tipo_de_deformación”])
falla	(”sobrecalentamiento”,[”pared_de_agua”,”elongado”])
falla	(”corrosión_cáustica”,[”pared_de_agua”,”abombado”])
falla	(”termofluencia”,[”sobrecalentador”])

Teniéndose un conjunto de reglas representadas por medio de objetos llamados falla. La conclusión de la regla está presente en su primer componente, seguida de una lista de características del tubo fallado, que son las condiciones de la regla, pero las relaciones entre calificadores y características no se referencian una por una como en el caso anterior, en cambio, un atributo relaciona a todas las características correspondientes, por ejemplo:

atributos(la_posición_del_tubo,pared_de_agua,sobrecalentador])

expresa que la posición del tubo califica al valor pared de agua, característica de la fallas sobrecalentamiento y corrosión cáustica, y al valor sobrecalentador, característica de la falla termofluencia.

Las diferencias en la estructura del conocimiento de ambas bases se ven más claras, si transladamos ambas estructuras a un árbol. En esta translación se está poniendo la estructura inmersa en las reglas en términos de las alternativas posibles en cada estado del problema de búsqueda de una solución. Los dos árboles son mostrados en las figuras 3.2, y 3.3.

Así se tiene que la siguiente regla presente en el ejemplo de BC "A" :

falla(sobrecalentamiento,
[pared_de_agua,elongado,boca_de_pescado,bordes_delgados])

- arco_dbase(n(la_posición_del_tubo,pared_de_agua), [pared_de_agua])
- arco_dbase(n(tipo_de_deformación,elongado), [presenta_elongación])
- arco_dbase(n(tipo_de_fractura,boca_de_pescado), [abertura_ancha,a_lo_largo_del_tubo])
- arco_dbase(n(tipo_de_bordes,bordes_delgados), [el_espesor_de_la_pared_del_tubo esta_adelgazado]) arco_dbase(n(tipo_de_bordes,bordes_delgados), [en_forma_de_cuchilla,adelgazados])

es representada por el árbol en la Fig. 3.2.

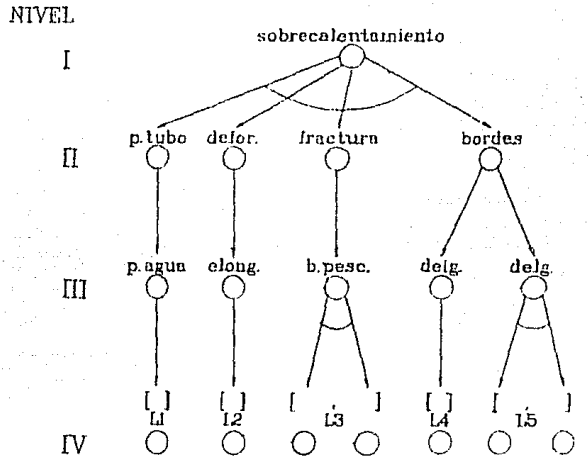


Fig. 3.2 Arbol que representa a una regla en BC "A" generado por un proceso de búsqueda. ([ ] indica la lista descriptiva de las relaciones atributo–valor presentes en los niveles II y III).

El nodo raíz de este árbol representa a la condición de meta que describe la conclusión *sobrecalentamiento* de la regla. Este nodo es tipo "Y", es decir, para arribar a la conclusión el sistema debe de verificar la validez de todos sus nodos sucesores. Estos nodos, en el nivel II, que representan a los distintos atributos que califican a las características del tubo fallado, son del tipo "O" , lo cual quiere decir que para la satisfacción de un nodo "O" en este nivel es suficiente la realización de cualesquiera de sus nodos sucesores en el nivel III. Por lo que respecta a los nodos en el tercer nivel, que identifican a las características del tubo fallado son nodos tipo "Y", indicando que para la satisfacción de uno de estos nodos es necesaria la realización de todos sus nodos sucesores, los cuales describen a la característica del tubo .

De la presentación anterior en términos de árboles de la estructura de conocimiento en BC "A", podemos concluir a partir de la relación entre atributos y sus correspondientes características, expresada en los niveles II y III, que es posible describir la relación entre un atributo y su característica correspondiente, de varias maneras, a través de sus posibles nodos alternativos en el nivel III, y que son descritos en el nivel IV, por medio de sus listas asociadas. Por ejemplo, para la relación tipo de bordes-delgados en la Figura anterior, hay dos maneras análogas o sinónimas de describir a bordes_delgados a través de las listas asociadas presentes en el nivel IV. Digo que son análogas ya que el tipo de nodo asociado a tipo de bordes es "O". Lo anterior permite tener una mayor posibilidad de descripción al utilizar esta estructura.

Para la regla asociada a sobrecalentamiento en el ejemplo de la base BC "B" se tiene el siguiente árbol en la Fig. 3.3.

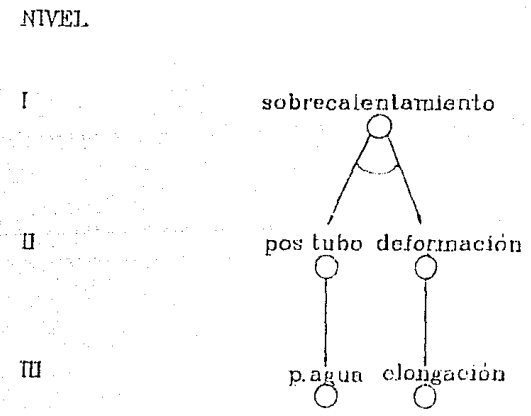


Fig. 3.3 Arbol que representa a una regla en BC "B" generado por un proceso de búsqueda.

Los árboles generados por el tipo de reglas presente en la base BC "B", como se puede observar en la Fig. 3.3, son de 3 niveles, a diferencia de los 4 niveles que son generados en BC "A". La ausencia del nivel IV en estos árboles implica que hay una menor descripción de las relaciones entre calificadores y características del tubo fallado, de hecho la descripción es dada por la relación misma. Sin embargo en esta base se ha eliminado redundancia a través del algoritmo que procesa datos empíricos en reglas, este hecho será presentado más adelante en el capítulo IV cuando se hable acerca de la implementación del algoritmo.

La ventaja entonces de la base BC "A" sobre la base BC "B" se manifiesta en una mayor descripción de las características del tubo fallado, sin embargo presenta una mayor redundancia, debida a que el conocimiento del experto es capturado directamente y de inmediato se almacena, sin haber un proceso de refinamiento en el sistema para la base BC "A".

Proceso C.

Hasta este punto hemos hecho referencia a dos bases de conocimiento, las cuales son intermedias hacia una base de conocimiento final BC "C", que sería implementada por el programa Completez. De contar con él, se compararían las bases BC "A" y BC "B" resultando la base BC "C" sin inconsistencias y redundancias, que debe ser la de consulta. Dicho programa no se encuentra actualmente en el sistema, ya que es un tema actualmente en desarrollo a nivel mundial en el área de sistemas expertos y bases de datos. Existen diferentes enfoques a este respecto: encontrándose contradicciones en algunas de ellas. Para una mayor discusión y acercamiento puede consultarse la siguiente bibliografía [8],[9].

Actualmente el sistema permite consultar por un lado la base BC "A", si la entrada al sistema es por el módulo del experto. Asimismo también puede consultar la base BC "B" si la entrada al sistema es por el módulo del usuario. En la próxima sección se hablará acerca de estos programas de consulta a las bases de conocimiento BC "A" y BC "B".

Si bien el sistema, en este momento no es capaz de comparar las dos bases intermedias de conocimiento BC "A" y BC "B", hay que hacer notar que el sistema realiza cambios en el conocimiento contenido en la base BC "B", ya que al llegar nueva información al sistema, la cual puede ser inconsistente o incompleta; el sistema es capaz de razonar con ella, al ser ajustada la base de datos por el programa Soporte, y en consecuencia reajustar sus reglas a través del programa Genera. Por lo que toca a la base BC "A" el experto tiene la opción de modificarla agregando o borrando una regla.

3.2.2.2 Componentes de la Máquina Inferencial

Dentro de la máquina inferencial del actual sistema se distinguen varios procedimientos, entre estos se encuentran los programas Encuentra, Genera, Soporte . El programa Genera como ya señalamos merece atención aparte en el siguiente capítulo.

El procedimiento Soporte proporciona al sistema la capacidad de manejar la información adquirida a través de Captura II, y que le es proporcionada cuando no se produjo un diagnóstico ó, si este se produjo, el usuario tiene información sobre otro tipo de falla. Ya que es realmente imposible predeterminar la cantidad correcta de información y conocimiento requeridas para realizar un diagnóstico, este procedimiento es necesario para mantener credibilidad a la información contenida en la base de datos (BD), la cual soporta al programa Genera para arribar a sus conclusiones. Para ilustrar a BD a lo largo de todos los casos mencionados adelante, consideraremos la siguiente base de datos inicial, refiriéndonos a ella como BDI, la cual será ajustada de acuerdo a si la información proporcionada por el usuario es completa, incompleta y/o inconsistente; resultando una base actualizada que llamaremos BD-A.

Por otra parte, el conjunto de números dispuestos en renglón son índices que identifican la posición inicial de cada objeto respecto al número de bytes acumulados a su inicio en la base. Este archivo de índices le sirve al compilador turbo Prolog para dar de baja a estructuras de datos de BD que ya no interesan actualmente, lo cual es representado asignando un -1 a su índice asociado, de esta manera sólo se leerán estructuras con índices positivos en la siguiente consulta al sistema.

(BASE DE DATOS INICIAL)
BDI

```
  0      clase     (["causa","sobrecalentamiento","termofluencia"])
 55      causa     (["sobrecalentamiento","termofluencia"])
102   atributos    (["la_posición_del_tubo","pared_de_agua", "sobrecalentador"])
173   atributos    (["tipo_de_deformación","elongado","abombado"])
231   atributos    (["tipo_de_fractura","boca_de_pescado"])
282   atributos    (["tipo_de_bordes","bordes_delgados","bordes_gruesos"])
348    lis_nom     (["la_posición_del_tubo","tipo_de_deformación",
                     "tipo_de_fractura","tipo_de_bordes"])
441      dato      ([ ],"única","sobrecalentamiento",["pared_de_agua","elongado",
                                                     "boca_de_pescado",
                                                     "bordes_delgados"])
545      dato      ([ ],"única","termofluencia",{"sobrecalentador","abombado",
                                                 "boca_de_pescado",
                                                 "bordes_gruesos"})
```

Si observamos, en BDI estan presentes dos reportes o datos capturados a través de Captura II, representados por objetos llamados dato. Cada dato contiene información acerca del tipo de falla en su tercer componente, y una lista de características descriptivas en la cuarta componente. La primera y segunda componente sólo tienen significado para el proceso interno del programa Genera, por lo que bien pueden ser ignorados en la descripción del proceso Soporte. Cualquier característica asociada a una falla tiene un calificador que al igual que en BC "B", están referenciadas por medio de los objetos llamados atributos. La estructura lis_nom nos muestra en una lista a los atributos diferentes inmersos en los datos; mientras clase y causa nos muestran a los distintos tipos de fallas presentes en los datos.

El procedimiento Soporte, como primer paso verifica si el número de variables o atributos , que contiene la información recibida para caracterizar al tubo fallado es igual, menor o mayor al número de variables contenida en la información que conforma la BD.

Caso a) Si el número de variables es igual, y los valores de las variables asociados a la información proporcionada, ya están presentes en BD, entonces el siguiente paso es verificar que la lista de valores de las variables proporcionada, no sea la misma lista que describa otro tipo de falla presente en BD, ya que tener esta situación en BD representa un conflicto. A una lista junto con un tipo de falla diré que es un elemento de BD.

a1) Si existe un conflicto, el programa Soporte no permite la entrada a BD de la información proporcionada, además borra de BD los elementos conflictivos y, en consecuencia, puede también borrar valores de variables contenidos en BD para mantener la completez de la base.

Ej 1: Observando a BD-A:

BD-A

-1	clase	(["causa","sobrecalentamiento","termofluencia"])
-1	causa	(["sobrecalentamiento","termofluencia"])
-1	atributos	(["la_posición_del_tubo","pared_de agua", "sobrecalentador"])
-1	atributos	(["tipo_de_deformación","elongado","abombado"])
231	atributos	(["tipo_de_fractura","boca_de_pescado"])
-1	atributos	(["tipo_de_bordes","bordes_delgados","bordes_gruesos"])
348	lis_nom	(["la_posición_del_tubo","tipo_de_deformación", "tipo_de_fractura","tipo_de_bordes"])
-1	dato	([ ],"única","sobrecalentamiento",["pared_de_agua","elongado", "boca_de_pescado", "bordes_delgados"])
545	dato	([ ],"única","termofluencia",["sobrecalentador","abombado", "boca_de_pescado", "bordes_gruesos"])
645	atributos	(["la_posición_del_tubo","sobrecalentador"])
700	atributos	(["tipo_de_deformación","abombado"])
747	atributos	(["tipo_de_bordes","bordes_gruesos"])
795	causa	(["termofluencia"])
821	clase	(["causa","termofluencia"])
-1	dato	([ ],"única","termofluencia",["pared_de_agua","elongado", "boca_de_pescado", "bordes_delgados"])

vemos que el reporte capturado es,

dato(termofluencia,
[pared_de_agua,elongado,boca_de_pescado,bordes_delgados])

que resulta ser inconsistente con,

dato(sobrecalentamiento,
[pared_de_agua,elongado,boca_de_pescado,bordes_delgados])

de BDI, no permitiéndose la entrada de este elemento, lo cual es representado asignando un -1 al índice asociado a este elemento en BD-A. La misma asignación se hace al dato asociado a sobrecalentamiento, trayendo como consecuencia la actualización de los atributos, posición del tubo, tipo de deformación, tipo de bordes en sus valores asociados en sus listas, y también la de los objetos que representan a las causas de fallas; marcando con -1 sus índices respectivos, y abriendo nuevos objetos en BD-A: a saber:

atributos([la_posición_del_tubo,sobrecalentador])
atributos([tipo_de_deformación,abombado])
atributos([tipo_de_bordes,bordes_gruesos])
causa([termofluencia])
clase([causa,termofluencia])

a2) En caso de no existir un conflicto entonces la información proporcionada es agregada en

BD.

Ej 2: La siguiente base BD-A resulta de agregar el elemento,

dato(sobrecalentamiento,
[sobrecalentador,elongado,boca_de_pescado,bordes_delgados])

al no crear conflicto con ningún elemento de BDI, además como podemos observar todos sus valores de atributo estaban ya en BDI.

BD-A

0	clase	(["causa","sobrecalentamiento","termofluencia"])
55	causa	(["sobrecalentamiento","termofluencia"])
102	atributos	(["la_posición_del_tubo","pared_de_agua", "sobrecalentador"])
173	atributos	(["tipo_de_deformación","elongado","abombado"])
231	atributos	(["tipo_de_fractura","boca_de_pescado"])
282	atributos	(["tipo_de_bordes","bordes_delgados","bordes_gruesos"])
348	lis_nom	(["la_posición_del_tubo","tipo_de_deformación", "tipo de_fractura","tipo_de_bordes"])
441	dato	([ ],"única","sobrecalentamiento",["pared_de_agua","elongado", "boca_de_pescado", "bordes_delgados"])
545	dato	([ ],"única","termofluencia",["sobrecalentador","abombado", "boca_de_pescado", "bordes_gruesos"])
645	dato	([ ],"única","sobrecalentamiento",["sobrecalentador","elongado", "boca_de_pescado", "bordes_delgados"])

a3) Por otra parte, si existe al menos un valor de variable en la lista de la información proporcionada que no esté contenida en BD, entonces esta información no puede crear conflictos en BD y se agrega automáticamente a BD, para conformar otro elemento en BD.

Ej 3: A BDI se agregó el elemento,

dato(corrosión_cáustica,
[pared_de_agua,ligero_abombamiento,picaduras,bordes_delgados])

el cual tiene dos nuevos valores de atributo, uno, ligero abombamiento asociado a tipo de deformación, y otro, picaduras correspondiente a tipo de fractura, además de una nueva causa de falla, corrosión cáustica, resultando entonces la base BD-A; actualizada con los nuevos valores de atributo, y tipo de falla en sus respectivos objetos.

BD-A:

```
-1      clase      (["causa","sobrecalentamiento","termofluencia"])
-1      causa      (["sobrecalentamiento","termofluencia"])
102     atributos  (["la_posición_del_tubo","pared_de_agua", "sobrecalentador"])
-1      atributos  (["tipo_de_deformación","elongado","abombado"])
-1      atributos  (["tipo_de_fractura","boca_de_pescado"])
282     atributos  (["tipo_de_bordes","bordes_delgados", "bordes_gruesos"])
348     lis_nom    (["la_posición_del_tubo","tipo_de_deformación",
                     "tipo_de_fractura","tipo_de_bordes"])
441     dato       ([ ],"única","sobrecalentamiento",["pared_de_agua","elongado",
                                                      "boca_de_pescado",
                                                      "bordes_delgados"])
545     dato       ([ ],"única","termofluencia",["sobrecalentador","abombado",
                                                 "boca_de_pescado",
                                                 "bordes_gruesos"])
645     atributos  (["tipo_de_deformación","elongado","abombado",
                     "ligero_abombamiento"])
725     atributos  (["tipo_de_fractura","boca_de_pescado","picaduras"])
788     causa      (["sobrecalentamiento","termofluencia","corrosión_cáustica"])
856     clase      (["causa","sobrecalentamiento","termofluencia",
                     "corrosión_cáustica"])
932     dato       ([ ],"única","corrosión_cáustica",["pared_de_agua",
                                                      "ligero abombamiento",
                                                      "picaduras",
                                                      "bordes_delgados"])
```

Caso b) Si el número de variables proporcionada es menor, y tal que todas están presentes en BD, lo mismo que los valores de estas variables entonces se completa la lista de variables en la información proporcionada, de tal manera , que una variable ausente es considerada irrelevante, por lo que es necesario incluirla con sus n valores posibles a la información proporcionada; conformándose n-listas cada una con un valor diferente de variable irrelevante e información proporcionada. Si existe otra variable irrelevante digamos con m valores, entonces cada n-lista formada; conforma a su vez m listas cada una con un valor diferente de esta otra variable irrelevante , teniéndose de esta manera *mn* listas. Este proceso se continua de la misma manera hasta terminar con la última de las variables irrelevantes en la información proporcionada. Entonces se verifica inconsistencias para cada lista formada lo cual se reduce a los subcasos a1) y a2) al estar ya cada lista completa, con la salvedad de que si se presenta un conflicto , solamente no es permitida la entrada de la lista conflictiva a BD ya que posiblemente la inconsistencia no fué producida directamente por el usuario.

Ej 4: El elemento capturado fué,

dato(daño_x_hidrógeno,[pared_de_agua,*,*,*])

Los asteriscos nos indican los atributos marcados como irrelevantes en el proceso de captura de la información. Lo anterior lo podemos notar al haberse generado 2 x 1 x 2 = 4 datos para daño por hidrógeno en BD-A, al tener los atributos, tipo de deformación, tipo de fractura y tipo de bordes 2, 1 y 2 valores respectivamente.

Las inconsistencias se verifican para cada dato, así el siguiente elemento:

dato([daño_x_hidrógeno,
[pared_de_agua,elongado,boca_de_pescado,bordes_delgados]])

resultó ser el único conflictivo con BDI, y por lo tanto no se le permitió la entrada a BD-A. Además como daño por hidrógeno es una nueva causa de falla se actualizan sus objetos de referencia en BD-A, a saber:

causa([sobrecalentamiento,termofluencia, daño_x_hidrógeno])
clase([causa,sobrecalentamiento,termofluencia,daño_x_hidrógeno])

BD-A

-1	clase	(["causa","sobrecalentamiento","termofluencia"])
-1	causa	(["sobrecalentamiento","termofluencia"])
102	atributos	(["la_posición_del_tubo","pared_de_agua", "sobrecalentador"])
173	atributos	(["tipo_de_deformación","elongado","abombado"])
231	atributos	(["tipo_de_fractura","boca_de_pescado"])
282	atributos	(["tipo_de_bordes","bordes_delgados","bordes_gruesos"])
348	lis_nom	(["la_posición_del_tubo","tipo_de_deformación", "tipo_de_fractura","tipo_de_bordes"])
441	dato	([ ],"única","sobrecalentamiento",["pared de agua","elongado", "boca_de_pescado", "bordes_delgados"])
545	dato	([ ],"única","termofluencia",["sobrecalentador","abombado", "boca_de pescado", "bordes_gruesos"])
-1	dato	([ ],"única","daño_x_hidrógeno",["pared_de_agua","elongado", "boca_de_pescado", "bordes_delgados"])
747	causa	(["sobrecalentamiento","termofluencia","daño_x_hidrógeno"])
813	clase	(["causa","sobrecalentamiento","termofluencia", "daño_x_hidrógeno"])
887	dato	([ ],"única","daño_x_hidrógeno",["pared_de_agua","abombado", "boca_de_pescado", "bordes_delgados"])
989	dato	([ ],"única","daño_x_hidrógeno",["pared_de_agua","elongado", "boca_de_pescado", "bordes_gruesos"])
1090	dato	([ ],"única","daño_x_hidrógeno",["pared_de_agua","abombado", "boca_de_pescado", "bordes_gruesos"])

Si en la lista de variables proporcionada que están en BD, hay un valor nuevo para al menos una variable, entonces ninguna lista o elemento que se forman al completar la información creará conflicto con BD, por lo que automáticamente se agregan a BD.

Ej 5: El dato capturado fué

{pared_de_agua,*,ventana_abierta,*})

como puede observarse en BD-A a través de los 4 datos generados por la irrelevancia de 2 de sus atributos. Sin embargo como ventana abierta es un nuevo valor del tipo de fractura, ninguno de estos datos crea conflicto con BD1, por lo que son agregados automáticamente. Además se ajusta la lista de causas de falla de BD1, igual que el atributo de tipo de fractura.

BD-A

-1	clase	{"causa","sobrecalentamiento","termofluencia"}
-1	causa	{"sobrecalentamiento","termofluencia"}
102	atributos	{"la_posición_del_tubo","pared_de_agua", "sobrecalentador"}
173	atributos	{"tipo_de_deformación","elongado","abombado"}
-1	atributos	{"tipo_de_fractura","boca_de_pescado"}
282	atributos	{"tipo_de_bordes","bordes_delgados","bordes_gruesos"}
348	lis_nom	{"la_posición_del_tubo","tipo_de_deformación", "tipo_de_fractura","tipo_de_bordes"}
441	dato	{{},"única","sobrecalentamiento",{"pared_de_agua","elongado", "boca_de_pescado", "bordes_delgados"}}
545	dato	{{},"única","termofluencia",{"sobrecalentador","abombado", "boca_de_pescado", "bordes_gruesos"}}
645	atributos	{"tipo_de_fractura","boca_de_pescado", "ventana_abierta"}
714	causa	{"sobrecalentamiento","termofluencia","daño_x_hidrógeno"}
780	clase	{"causa","sobrecalentamiento","termofluencia", "daño_x_hidrógeno"}
854	dato	{{},"única","daño_x_hidrógeno",{"pared_de_agua","elongado", "ventana_abierta", "bordes_delgados"}}
956	dato	{{},"única","daño_x_hidrógeno",{"pared_de_agua","abombado", "ventana_abierta", "bordes_delgados"}}
1058	dato	{{},"única","daño_x_hidrógeno",{"pared_de_agua","elongado", "ventana_abierta", "bordes_gruesos"}}
1159	dato	{{},"única","daño_x_hidrógeno",{"pared_de_agua","abombado", "ventana_abierta", "bordes_gruesos"}}

Por otra parte si existe una variable nueva en la información proporcionada, es necesario primeramente completar los elementos de BD añadiendo la nueva variable así como su valor, de esta forma todas las variables de la información estan contenidas en BD; y se procede de la misma manera para completar la información proporcionada, y la verificación de inconsistencias dadas anteriormente en este caso b).

Ej 6: En BD-A observamos que el dato proporcionado fué

dato(daño_x_hidrógeno,[pared_de_agua,*,*,*,*,interno])

generándose 2x1x2=4 datos, que a excepción del ejemplo 4 dado anteriormente adicionó un nuevo atributo a la información capturada; teniéndose que ajustar todos los datos presentes en BDI con el valor interno de depósito. Se verifican inconsistencias para cada uno de los datos asociados a daño por hidrógeno. Se observa en BD-A, que se marcó con un −1 a:

dato(daño_x_hidrógeno,
[pared_de_agua,elongado,boca_de_pescado,bordes_delgados,interno])

el cual resulta ser inconsistente con,

dato(sobrecalentamiento,
[pared_de_agua,elongado,boca_de_pescado,bordes_delgados,interno])

Una vez más se actualizan en BD-A, las listas de causas de falla con daño por hidrógeno, además de la de lis_nom con su nuevo atributo , y se abre un nuevo objeto para este atributo.

```
 -1      clase      ([ "causa","sobrecalentamiento","termofluencia"])
 -1      causa      ([ "sobrecalentamiento","termofluencia"])
102      atributos  ([ "la_posición_del_tubo","pared_de_agua", "sobrecalentador"])
173      atributos  ([ "tipo_de_deformación","elongado","abombado"])
231      atributos  ([ "tipo_de_fractura","boca_de_pescado"])
282      atributos  ([ "tipo_de_bordes","bordes_delgados", "bordes_gruesos"])
 -1      lis_nom    ([ "la_posición_del_tubo","tipo_de_deformación",
                     "tipo_de_fractura","tipo_de_bordes"])
 -1      dato       ([ ],"única","sobrecalentamiento",[ "pared_de_agua","elongado",
                                                        "boca_de_pescado",
                                                        "bordes_delgados"])
 -1      dato       ([ ],"única","termofluencia",[ "sobrecalentador","abombado",
                                                   "boca_de_pescado",
                                                   "bordes_gruesos"])
645      lis_nom    ([ "la_posición_del_tubo","tipo_de_deformación",
                     "tipo_de_fractura","tipo_de_bordes",
                     "tipo_de_depósito"])
757      atributos  ([ "tipo_de_depósito","interno"])
800      dato       ([ ],"única","sobrecalentamiento",[ "pared_de_agua","elongado",
                                                        "boca_de_pescado",
                                                        "bordes_delgados",
                                                        "interno"])
914      dato       ([ ],"única","termofluencia",[ "sobrecalentador","abombado",
                                                   "boca_de_pescado",
                                                   "bordes_gruesos",
                                                   "interno"])
 -1      dato       ([ ],"única","daño_x_hidrógeno",[ "pared_de_agua","elongado",
                                                      "boca_de_pescado",
                                                      "bordes_delgados",
                                                      "interno"])
1136     causa      ([ "sobrecalentamiento","termofluencia","daño_x_hidrógeno"])
1202     clase      ([ "causa","sobrecalentamiento","termofluencia",
                     "daño_x_hidrógeno"])
1276     dato       ([ ],"única","daño_x_hidrógeno",[ "pared_de_agua","abombado",
                                                      "boca_de_pescado",
                                                      "bordes_delgados",
                                                      "interno"])
1388     dato       ([ ],"única","daño_x_hidrógeno",[ "pared_de_agua","elongado",
                                                      "boca_de_pescado",
                                                      "bordes_gruesos",
                                                      "interno"])
1499     dato       ([ ],"única","daño_x_hidrógeno",[ "pared_de_agua","abombado",
                                                      "boca_de_pescado",
                                                      "bordes_gruesos",
                                                      "interno"])
```

Si el número de variables es mayor entonces existe un atributo nuevo en la información proporcionada, y el siguiente paso es completar la información contenida en BD en relación a la proporcionada; proporcionando la nueva variable a cada elemento en BD. Teniéndose con esto que el número de variables tanto en la información proporcionada como en la contenida en BD es igual, con lo que se tiene de nueva cuenta el caso a), y se procede desde un principio para verificar si hay un conflicto o no.

Ej 7:

Observamos en BD-A que el dato capturado es:

dato(corrosión_de_baja_temperatura,
[economizador,elongado,pequeña_grieta,bordes_delgados,ranurada])

el cual contiene como valor de "apariencia de la superficie exterior a "ranurada". Lo cual trae como consecuencia el ajuste del conjunto de datos en BDI, aumentándoles este nuevo atributo.

El dato agregado además contiene a dos nuevos valores de atributos que ya estaban en BDI, no creando conflicto con BDI. Se actualizan los atributos adicionando sus nuevos valores, además de las listas asociadas a las causas de falla y se crea un objeto para el atributo nuevo, aunado al ajuste de lis_nom.

```
 -1      clase    (["causa","sobrecalentamiento","termofluencia"])
 -1      causa    (["sobrecalentamiento","termofluencia"])
 -1   atributos   (["la_posición_del_tubo","pared_de_agua", "sobrecalentador"])
173   atributos   (["tipo_de_deformación","elongado","abombado"])
 -1   atributos   (["tipo_de_fractura","boca_de_pescado"])
282   atributos   (["tipo_de_bordes","bordes_delgados", "bordes_gruesos"])
 -1     lis_nom   (["la_posición_del_tubo","tipo_de_deformación",
                    "tipo_de_fractura",
                    "tipo_de_bordes"])
 -1        dato   ([ ],"única","sobrecalentamiento",["pared_de_agua",
                                                     "elongado",
                                                     "boca_de_pescado",
                                                     "bordes_delgados"])
 -1        dato   ([ ],"única","termofluencia",["sobrecalentador",
                                                "abombado",
                                                "boca_de_pescado",
                                                "bordes_gruesos"])
645   atributos   (["la_posición_del_tubo","pared_de_agua",
                    "sobrecalentador","economizador"])
731   atributos   (["tipo_de_fractura","boca_de_pescado",
                    "pequeña_grieta"])
799     lis_nom   (["la_posición_del_tubo","tipo_de_deformación",
                    "tipo_de_fractura","tipo_de_bordes",
                    "apariencia_superf_ext"])
916   atributos   (["apariencia_superf_ext","ranurada"])
965        dato   ([ ],"única","sobrecalentamiento",["pared_de_agua",
                                                     "elongado",
                                                     "boca_de_pescado",
                                                     "bordes_delgados",
                                                     "ranurada"])
1080       dato   ([ ],"única","termofluencia",["sobrecalentador",
                                                "abombado",
                                                "boca_de_pescado",
                                                "bordes_gruesos",
                                                "ranurada"])
1191      causa   (["sobrecalentamiento","termofluencia",
                    "corrosión_de_baja_temperatura"])
1270      clase   (["causa","sobrecalentamiento","termofluencia",
                    "corrosión_de_baja_temperatura"])
1357       dato   ([ ],"única","corrosión_de_baja_temperatura", ["economizador",
                                                                 "elongado",
                                                                 "pequeña_grieta",
                                                                 "bordes_delgados",
                                                                 "ranurada"])
```

A continuación hablaremos del programa Encuentra, el cual puede consultar a la base BC

"A", o a la base BC "B". De hecho, como se ha ilustrado anteriormente a través de los ejemplos de árbol correspondientes, las reglas en BC "B" resultan ser en su estructura un caso particular de las de BC "A", al tener sólo tres niveles de arcos de tipo "Y", por lo que se ejemplificará entonces el mecanismo de control del programa Encuentra en base a un conjunto de reglas en BC "A".

Consideremos las suiguientes dos reglas, que conformen la base ejemplo para BC "A":

falla(sobrecalentamiento,[pared_de_agua,elongado, boca_de_pescado, bordes_delgados])

- arco_dbase(n([la_posición_del_tubo,pared_de_agua),
  [pared_de_agua])
- arco_dbase(n(tipo_de_deformación,elongado),
  [presenta_elongación])
- arco_dbase(n(tipo_de_fractura,boca_de_pescado),
  [abertura_ancha,a_lo_largo_del_tubo])
- arco_dbase(n(tipo_de_bordes,bordes_delgados),
  [el_espesor_de_la_pared_del_tubo_esta_adelgazado])
  arco_dbase(n(tipo_de_bordes,bordes_delgados),
  [en_forma_de_cuchilla,estan_adelgazados])

falla(corrosión_cáustica, [pared_de_agua,picaduras,ligero_abombamiento])

- arco_dbase(n(tipo_de_deformación,ligero,abombamiento),
  [presenta_un_ligero_abombamiento,en_forma_de_de_chipote])
- arco_dbase(n(tipo_de_fractura,picaduras),
  [picaduras])
  arco_dbase(n(tipo_de_fractura,picaduras),
  [pequeñas_perforaciones_tipo_poro])

Las reglas anteriores, se representarán a través de un árbol como resultado de una búsqueda en gráfica a manera de tener un conjunto de nodos o estados posibles, representando el conjunto total de alternativas disponibles para la búsqueda de una solución. Este marco nos servirá como una guía en la discusión del mecanismo de control presente en Encuentra, el cual va a ser ilustrado a través de un árbol de búsqueda que va a ser construído conforme se este realizando la consulta a la base, indicada por medio de flechas en la gráfica, Fig. 3.4.

Supongamos que se realiza una consulta a la base BC "A", para encontrar un posible diagnóstico preliminar en base a características observables de la muestra del tubo fallado. El mecanismo de control que actúa sobre la base BC "A" utiliza un mecanismo de razonamiento hacia atrás.

Al aplicarse el mecanismo al nodo raíz, entiende que debe satisfacer alguna regla. Selecciona la primera regla de la base, y genera una liga explícita a la causa de falla *sobrecalentamiento*; representando esta acción por un arco entre el nodo raíz y el nodo asociado a la causa sobrecalentamiento. Al estar en este momento en un nodo del primer nivel, el mecanismo está enterado de que es un nodo tipo "Y". Genera entonces una liga al nodo asociado al atributo de la primera característica, a saber, posición del tubo, el cual es un nodo tipo "O" al estar en el nivel II. Al intentar satisfacer

este nodo encuentra que la posición del tubo es pared de agua, y une estos dos conceptos a través de una liga entre el nodo de posición del tubo y el nodo de pared de agua. Para determinar esta relación debe preguntar si se verifica la lista descriptiva asociada a ella, presente en el nivel IV; se liga el nodo de pared de agua al nodo de dicha lista.

La pregunta se formula poniendo primero el atributo calificador y en seguida el primer elemento de la lista, de la siguiente manera:

la_posición_del_tubo: pared_de_agua

Supóngase que la respuesta dada por el usuario es afirmativa, y como la lista tiene un sólo elemento, entonces el nodo asociado a posición del tubo ha sido satisfecho.

Acto seguido encuentra que el segundo atributo a satisfacer es tipo de deformación y lo liga al nodo de elongado. Se repite de manera análoga el proceso de ligar respectivamente el nodo de tipo de deformación para determinar la validez del tipo de deformación elongado. Si nos fijamos en la gráfica nos damos cuenta que la respuesta dada es negativa, ya que el control regresa al nivel III ha determinar si existe un nodo alternativo para satisfacer al nodo de tipo de deformación; como no lo hay no se satisface el nodo de tipo de deformación en nivel II, y el control regresa al primer nivel de nodos, a seleccionar la siguiente regla a saber corrosión cáustica.

De la misma forma se liga el nodo raíz al nodo asociado a la causa de falla corrosión cáustica, y tratará de establecer la validez de sus nodos sucesores en el nivel II.

A continuación, y según vemos en la figura a través de las flechas, el control hace primeramente la ligación de la posición del tubo hasta su lista de características asociada en el nivel IV, para preguntar acerca de esta relación, sin embargo antes de hacerla verifica que esta lista es la misma asociada para la posición del tubo de la falla *sobrecalentamiento.* Con esto la siguiente pregunta realizada es tipo_de_deformación: presenta_un_ligero_abombamiento por medio de su respectiva ligación; como se muestra en la gráfica, la respuesta es afirmativa. Enseguida pregunta acerca de si la fractura tiene también forma de chipote, ya que ligero abombamiento es un nodo tipo "Y"; una vez más la respuesta es positiva, y entonces el control encuentra el siguiente nodo asociado a corrosión cáustica en el nivel II, lo liga, y pregunta enseguida acerca del tipo de fractura "picaduras". La respuesta es negativa, entonces el control regresa al nivel III ,y encuentra un nodo alternativo para intentar satisfacer la relación tipo de fractura-picaduras, lo liga al tipo de fractura y se realiza la siguiente pregunta tipo_de_fractura: pequeños_agujeros_tipo_poro, la cual es positiva. Satisfaciéndose todos los nodos sucesores asociados a la causa de falla corrosión cáustica, por lo que concluye su diagnóstico en esta falla.

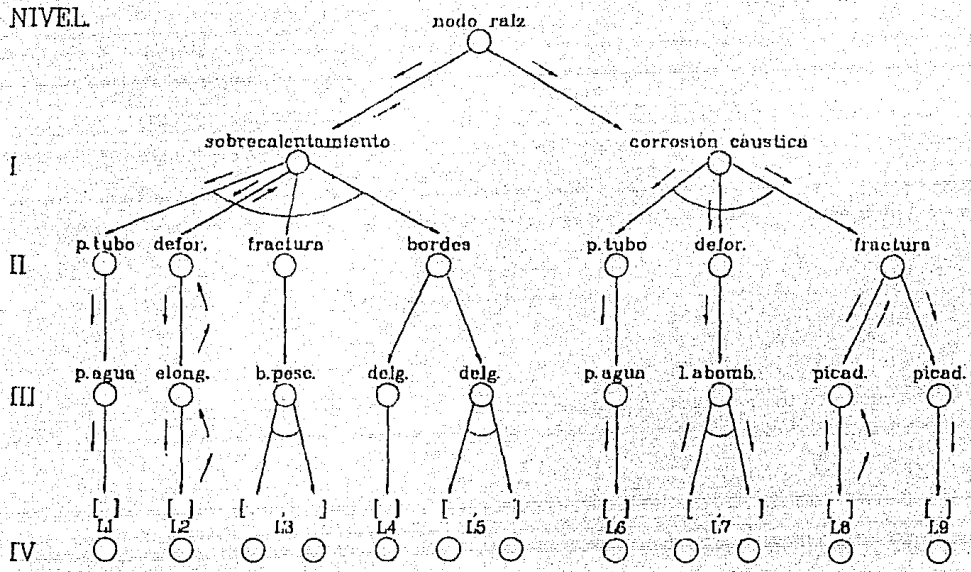


Fig. 3.4 Consulta a una base de conocimiento BC "A" con dos reglas, la cual es indicada por flechas a través del árbol.

Finalmente, mencionaremos a los Programas Verifica y Enseña, ausentes en este sistema, describiendo algunas de las características que se desea contengan.

En cuanto al programa Verifica, lo que se intenta es confirmar el diagnóstico sobre el tubo fallado, mostrando al usuario a través de un proyector fotos que permitan visualizar las características del tubo fallado de acuerdo a las respuestas dadas en el proceso de interacción. Este programa no se encuentra presente en el sistema, debido a que se requeriría un esfuerzo electrónico, el cual está fuera de los alcances de este trabajo de tesis, pero sería de gran importancia su uso, ya que reforzaría la objetividad, que debe de estar presente en el tratamiento del problema de diagnóstico.

El programa Enseña, tampoco se encuentra disponible en el sistema actual, pero es importante hacer notar que papel desempeña este mecanismo de razonamiento en un SE. Una vez que el sistema arribó a su conclusión, debe de tener la habilidad de explicar su propio mecanismo de razonamiento; esto es, debe de ser hábil en reconstruir las trayectorias de inferencias que debió de haber tomado para llegar a su conclusión. Esta habilidad es importante, ya que al ser capaz de explicar su razonamiento tiene la capacidad de entrenar a personas no expertas en el área. La razón por la cual no se implementó lo anterior en el presente trabajo de tesis, se debió a que los objetivos principales de este trabajo fueron enfocados hacia el proceso de adquisición de conocimiento del sistema.

Capítulo 4

Información: Como se mide

4.1 Introducción

Antes de proceder a la discusión de como clasificar datos eficientemente, analizaremos las características de éstos ; describiendo la manera en que el sistema captura la información contenida en ellos mediante Captura II. La captura de información se lleva a cabo de la siguiente manera:

a) Se presentan variables, a las cuales llamaremos atributos, en una ventana; esta presentación se hace de una variable a la vez, preguntándose si el atributo presente es relevante para la caracterización del tubo fallado.

b) Si la respuesta es afirmativa, se presentan en otra ventana una lista de valores de dicho atributo con la intención de que se elija uno. Los valores son utilizados para caracterizar específicamente una falla; por ejemplo, un atributo podría ser "deformación" cuyos posibles valores pueden ser "elongado", "abombado". Si el valor a ser proporcionado por el usuario no está presente en la ventana, hay una opción dentro de la misma para capturarlo y agregarlo a la lista de valores del atributo relevante. Si la respuesta fué negativa, o se concluyó el paso anterior, automaticamente se determina la relevancia del siguiente atributo.

c) El proceso se repite hasta agotar la lista de atributos inicialmente presente en el sistema. En caso de que el usuario deseé agregar un atributo adicional para caracterizar más su descripción sobre el tubo fallado, existe una opción mediante la cual se captura el atributo, y el valor asociado.

d) Finalmente, se muestra en una ventana una lista de causas, con la opción para agregar otra. Aquí conviene señalar nuevamente que el programa SOPORTE analizará la consistencia de la información proporcionada con la ya existente antes de agregarla a la base de datos, de manera tal que la elección podrá estar basada en un reporte o en la creencia del usuario.

En suma, para un dato o reporte se tienen: un subconjunto A_s, donde se encuentran los atributos relevantes, y el subconjunto V_{A_s}, cuyos elementos son los valores de los atributos seleccionados; asociado a ellos habrá un valor de causa c_i. Hay que señalar que si el usuario tomó la opción de agregar valores a las listas presentadas, entonces el sistema actualizará su base de datos de acuerdo como ya se discutió anteriormente.

4.2 Información

4.2.1 Cantidad de Información contenida en un Mensaje

Acabamos de ver cual es la estructura de un dato, por medio de la captura de ellos a través del programa Captura II. Qué es lo que lleva y cómo tener una clasificación eficiente de la información, es una pregunta que es atacada por una área de la matemática conocida como teoría de la información. A continuación se hablará acerca del espíritu de esta teoría.

Al recibir un mensaje, siempre se trata de atribuirle un significado, la teoría de información trata de cuantificarlo. Sabemos que pueden transmitirse mensajes a través de una cierta sucesión de símbolos, por ejemplo, en el alfabeto Morse se transmiten a través de dos símbolos, punto (.) y raya (-). De la misma manera se pueden generar mensajes de cualquier longitud conteniendo, uno o más símbolos.

Para atribuirle un significado a un mensaje de longitud N, es decir a una cierta sucesión de símbolos de tamaño N, debe de considerarse el número posible de mensajes que pudieron haber sido formados o mandados con esos símbolos, así la información que lleva un mensaje es relativa. Entre más grande sea el número de mensajes posibles que pudieron haber sido formados con esa sucesión de símbolos, la cantidad de información que es llevada por el mensaje recibido será mayor. Esto es, antes de recibir un mensaje, existirá una incertidumbre sobre la sucesión que será enviada de manera tal que la información puede ser medida a través de esta.

Alternativamente, la información puede ser relacionada con las probabilidades de obtención de cada uno de los símbolos que están presentes en la sucesión .

Ahora trataremos de formalizar la relación general entre la incertidumbre y la información llevada por cualquier tipo de mensaje .

En una situación inicial se tienen R_i posibles mensajes que pueden ser formados, y no se tiene ninguna información, $I_i = 0$. En una situación final se tienen $Rf = 1$ mensajes mandados ,y una información $I_f \neq 0$, de manera tal que $I(R)$, siendo I la función a ser determinada. Para esto, se desea que se puedan distinguir entre dos mensajes que llegan conjuntamente, para con esto poder medir la cantidad de información en el mensaje compuesto, en términos de la cantidad de información contenida en cada mensaje separadamente; es decir, se pide que la cantidad de información sea aditiva para eventos independientes, expresado en forma matemática tenemos:

Sean Ri1, Ri2 dos conjuntos independientes de posibles mensajes a ser enviados tales que

$$R = (Ri1)(Ri2)$$

se requiere que

$$I(Ri1Ri2) = I(Ri1) + I(Ri2)$$

Se puede demostrar [5] que la única función que satisface estas condiciones es:

$$I = K \ln(R)$$

donde K es una constante positiva arbitraria. En particular, $K = 1/\ln(c)$ de modo que

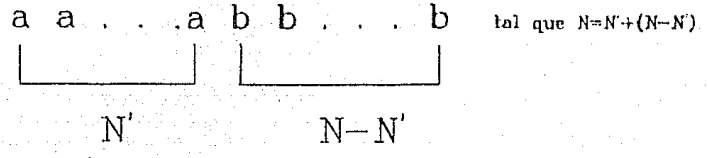
$$I = \log_c(R)$$

ya que

$$\log_c(x) = \frac{\log_d(x)}{\log_d(c)} = \log_c(d)\log_d(x)$$

es decir cualquier logaritmo de un número en una base c es proporcional al logaritmo del número en cualquier otra base d.

Para aplicar el resultado anterior al problema de determinar una clasificación de la información, supóngase que tenemos N datos sobre tubos fallados de los cuales N' son de un tipo de falla, y (N-N') son de otro tipo de falla. Los datos anteriores pueden formar una sucesión de tamano N (mensaje), del siguiente tipo:



Para atribuirle un significado a esta sucesión de reportes, pensemos en todos los mensajes o sucesiones de longitud N que pueden ser formados con N' de un tipo ,y N-N' de otro tipo de falla, esto no es otra cosa mas que el número de permutaciones entre N elementos, de los cuales N' son de un tipo, y N-N' son de otro tipo, por lo tanto,

el número de mensajes es:

$$\frac{N!}{N'!(N-N')!}$$

Este conjunto de mensajes representa R, al cual llamaremos espacio general de permutaciones, de manera tal que la cantidad de información contenida en un posible mensaje en este espacio es:

$$I = K \ln\left(\frac{N!}{N'!(N-N')!}\right) \ldots(1)$$

donde el argumento R también cuantifica la incertidumbre que hay antes del envío del mensaje. Si desarrollamos el logaritmo obtenemos:

$$\begin{aligned} I &= K[\ln N! - \ln(N'!(N-N')!)] \\ &= K[\ln N! - (\ln N'! + \ln(N-N')!)] \end{aligned}$$

y usando la aproximación de Stirling $\ln Q! = Q(\ln Q - 1)$, que es buena para $Q > 100$, entonces

$$I = K[N\ln(N-1) - (N'\ln(N'-1) + (N-N')\ln(N-N'-1))]$$

como $N = N' + (N - N')$ se obtiene

$$I = -K[N'\ln(N'/N) + (N-N')\ln((N-N')/N)]$$

Para obtener la información de los reportes por tipo de falla, es necesario dividir la expresión anterior entre el número de reportes de la sucesión,

$$\begin{aligned} i = I/N &= -K[(N'/N)\ln(N'/N) + (N \quad N')/N\ln((N - N')/N)] \\ &= -K[P\ln P + P'\ln P']...(2) \end{aligned}$$

donde: P es la probabilidad de obtener el primer tipo de falla
P' es la probabilidad de obtener el segundo tipo de falla
es decir $P = N'/N$ y $P' = (N - N')/N$.

Normalmente se elige $K = 1/\ln(2)$, y parece ser una elección natural, ya que por ejemplo si se tienen dos mensajes, y las probabilidades de obtener cada mensaje son iguales a 1/2, entonces

$$\begin{aligned} i &= -[1/2\log_2(1/2) + 1/2\log_2(1/2)] \\ &= 1/2\log_2(1/(1/2)) + 1/2\log_2(1/(1/2)) \\ &= 1/2\log_2(2) + 1/2\log_2(2) \end{aligned}$$

como el logaritmo de 2 en base 2 es igual a 1 se tiene

$$i = 1/2 + 1/2 = 1$$

y podríamos interpretar a i como el número de bits necesarios para distinguir entre los dos mensajes, de aquí su relación con la medida de bit usada en la computación, para almacenar información. Para una mayor profundidad y explicación de la relación existente entre la medida i, y la unidad bit utilizada en la computación, consultar la referencia [17], en donde se discute que pasa cuando se tienen 3 o más mensajes, teniendo distintas probabilidades asociadas.

Con esto hemos mostrado la equivalencia de relacionar la cantidad de información que lleva un mensaje con cualquiera de las dos maneras de cuantificar la incertidumbre; ya sea a través del número de mensajes posibles que pudieron haber sido mandados por una cierta sucesión de símbolos de longitud N, ó por medio de la probabilidad de obtener cada uno de estos símbolos.

Con la expresión (2), estamos clasificando los N reportes por tipos de falla, y solo podemos obtener la frecuencia con que se presenta cada falla, esto no nos ayuda para clasificarlas, por lo que es necesario incluir los distintos atributos para generar clasificaciones de los N datos de los cuales N' son de un tipo de falla y, N-N' son del otro tipo de falla.

Cada atributo genera una clasificación de los N reportes a través de sus valores, teniéndose un conjunto de mensajes. Cabe señalar que estos mensajes son distintos a los anteriores, ya que cada elemento de los nuevos es una falla con un solo atributo (con respecto del que se esta clasificando), y el valor asociado; por lo tanto la longitud sigue siendo la misma, pero los elementos al ser mas complejos pueden ser considerados como mensajes, que llamaremos submensajes. Entonces en vez de clasificar los reportes solo por tipos de falla, se clasificaran también por atributo simultaneamente; de manera tal que los submensajes permitan determinar la cantidad de información en los N reportes, esto es posible puesto que los submensajes relacionan tipos de falla, atributos y valores de atributo.

El resultado obtenido para las expresiones (1) y (2), y los que a continuación se desarrollarán son facilmente generalizables a m tipos de falla. Se considera el caso de solo dos tipos de fallas para facilitar la explicación.

4.2.2 Cantidad de Información contenida en un Atributo

Dados $N = N' + (N - N')$ reportes se tiene un conjunto A_t, el cual contiene solo a los atributos mencionados en los reportes, de modo que es un subconjunto del conjunto de atributos posibles A, y un conjunto cuyos elementos son los valores de atributo de cada elemento de A_t, al cual denotaremos por V_{A_t}.

Elijamos cualquier atributo de A_t , al cual denotaremos por A_t^* y tal que tenga asociada una lista de valores a_1^* , a_2^* ,..., a_l^*. Este atributo A_t^* genera una partición de los N reportes de la siguiente manera:

N_1: # de reportes con el valor a_1^*
N_2 : # de reportes con el valor a_2^*
$\vdots$
N_l : # de reportes con el valor a_l^*

$$\text{tal que } \sum_{i=1}^{l} N_i = N$$

Si observamos, este atributo caracteriza a cada subconjunto N' y (N-N') de reportes de N, donde N' es el número de reportes con el primer tipo de falla, y (N-N') es el número de reportes con el segundo tipo, de la siguiente manera :

N_1': reportes de N' con el valor a_1^*
$N_1 - N_1'$: reportes de N-N' con el valor a_1^*
$\vdots$
N_k' : reportes de N'con el valor a_k^*
$N_k - N_k'$: reportes de N-N' con el valor a_k^*
$\vdots$
N_l': reportes de N' con el valor a_l^*
$N_l - N_l'$: reportes de N-N' con el valor a_l^*

$$\text{tal que } N_i' + (N_i - N_i') = N_i \quad \forall_i = 1, ...l$$
$$\sum_{i=1}^{l} N_i' = N' , \quad \sum_{i=1}^{l} N_i - N_i' = N - N'$$

Los datos anteriores pueden ser puestos en forma matricial:

$$\begin{pmatrix} N_1' & N_1 - N_1' \\ \vdots & \vdots \\ N_l' & N_l - N_l' \\ N' & N - N' \end{pmatrix} \begin{matrix} N_1 \\ \vdots \\ N_l \\ N \end{matrix}$$

El análisis de la matriz se puede realizar en dos direcciones:

i) por renglón, estaríamos analizando la importancia que tiene el valor a_k^* sobre cada tipo de falla, en el caso de tomar el k-ésimo renglón;

ii) por columna , para considerar la distribución de los valores para cada tipo de falla.

Cada elemento de la matriz representa una sucesión de reportes de un tipo de falla, y un valor de atributo, esto es un submensaje; así al considerarse el k-ésimo renglón completo se tendrá un

mensaje que caracterizará al k-ésimo valor de atributo, independientemente del tipo de falla.

Esta dirección solo dá la distribución de ese valor respecto a cada falla, pero no toma en cuenta su relación con los otros valores, por lo que es necesario considerar también la dirección columna. Si se considera solo la dirección columna, estarán involucrados submensajes entre los cuales se encuentra el asociado al k-ésimo renglón; así al considerarse la columna completa se tendrá un mensaje que caracterizará al tipo de falla independientemente de los valores de atributo.

Por lo tanto es necesario considerar conjuntamente las direcciones del k-ésimo renglón e i-ésima columna, con lo cual podrá determinarse la probabilidad de la intersección. Como se describirá mas adelante esta probabilidad junto con la probabilidad de obtener un valor de atributo, independiente del tipo de falla, proporcionarán la probabilidad condicional la cual estará relacionada con la cantidad de información.

Para determinar la probabilidad de la intersección hay que recordar que al considerar conjuntamente la j-ésima columna con el k-ésimo renglón se tendrá un mensaje, el cual caracteriza a la j-ésima falla con un valor de atributo a_k^*. Este mensaje es elemento del conjunto de mensajes, que resultan de permutar los reportes con el j-ésimo tipo de falla, y valor k-ésimo de atributo; teniéndose por lo tanto

$$\frac{N^{(j)}!}{N_k^{(j)}!} \quad permutaciones.$$

($N^{(j)} = N'$, en este caso, si consideramos al primer tipo de falla)

Con lo anterior se tiene una expresión para el valor de atributo a_k^* para un tipo de falla; para determinar la expresión en la cual no importe el tipo de falla, es necesario considerar conjuntamente la expresión anterior para todo tipo de falla.

$$\prod_{j=1}^{p} \frac{N^{(j)}!}{N_k^{(j)}!}$$

En particular, para este caso

$$\frac{N'!(N-N')!}{N_k'!(N_k-N_k')!}$$

La consideración conjunta de la k-ésima dirección renglón, y las direcciones columnas, conformará un estado, el cual denotaremos por:

$$\begin{bmatrix} N' & (N-N') \\ N_k' & (N_k-N_k') \end{bmatrix}$$

Dicho estado será la intersección del k-ésimo valor de atributo con cada tipo de falla, y generará un espacio de permutaciones . A este espacio le llamaremos el espacio de intersección del valor k-ésimo de atributo, con el conjunto $N', (N-N')$ de tipos de falla, y lo denotaremos por $C_{a_k^* \cap (N', (N-N'))}$.

Hasta este punto hemos derivado una expresión en donde, por un lado se permutan N' elementos de un tipo de falla, donde N_k' tienen valor k-ésimo de atributo. Lo mismo para el otro tipo de falla, donde hemos permutado $N-N'$ elementos, siendo $N_k - N_k'$ de ellos con el mismo k-ésimo valor del atributo.

Que tal si ahora nos cuestionamos sobre la permutación de estos N_k' y $N_k - N_k'$, pero solo entre ellos mismos, teniéndose en este caso un submensaje de longitud N_k. Esto no es otra cosa

que considerar la k-ésima dirección renglón. Y retomando el objetivo inicial, sería determinar la importancia que tiene el k-ésimo valor de atributo a_k^* sobre cada tipo de falla. Teniéndose

$$\frac{N_k!}{N_k'!(N_k - N_k')!} \qquad permutaciones$$

A este espacio de permutaciones , donde se está restringiendo a un subconjunto N_k de N, determinado por el k-ésimo valor de atributo, se le llamará espacio condicional al k-ésimo valor de atributo. Será denotado por C/a_k^*, y diremos que es generado por :

$$\begin{bmatrix} N_k \\ N_k' \end{bmatrix}$$

Hemos generado dos espacios que servirán para encontrar una medida formal para la cantidad de información contenida en un atributo. Notamos que ambos espacios de permutaciones fueron generados, por un lado, por una sucesión particular de N' y $(N - N')$ posiciones de N posibles en el caso de $C_{a_k^* \cap (N', (N-N'))}$; y, por otro de N_k posiciones de N posibles para el espacio C/a_k^*. Como la primera sucesión particular representa un orden en el cual llegarón los reportes, un orden diferente generará el mismo espacio, de tal manera que

$$\frac{N!}{N'!(N - N')!}$$

será el número de maneras de formar el espacio $C_{a_k^* \cap (N', (N-N'))}$, independientemente del orden de los reportes. Esto es importante, ya que servirá para obtener la probabilidad de intersección entre cualquier falla, y un valor de atributo.

Para cada una de estas sucesiones habrá $\frac{N'!(N-N')!}{(N_k'!(N_k-N_k')!)}$ permutaciones posibles, cuando se tome en cuenta el k-ésimo valor de atributo; de manera tal que

$$\frac{N!}{N'!(N - N')!} \qquad \frac{N'!(N - N')!}{N_k'!(N_k - N_k')!}$$

es el número total de permutaciones sin tomar en cuenta el orden de los reportes, con el valor k-ésimo de atributo.

Por otra parte para la sucesión particular, la cual depende del orden en que llegaron los reportes, en el espacio condicional, se tienen N_k reportes distinguibles de un total de N, con lo que hay:

$$\frac{N!}{N_k!}$$

maneras de generar el espacio C/a_k^* y para cada una de estas sucesiones hay $\frac{N_k!}{(N_k'!(N_k-N_k')!)}$ permutaciones entre los elementos con valor k-ésimo de atributo; de modo que

$$\left(\frac{N!}{N_k!}\right)\left(\frac{N_k!}{N_k'!(N_k - N_k')!}\right)$$

serán las permutaciones totales posibles para el estado condicional.

Pero las dos cantidades de casos posibles son iguales, ya que el orden en que se reciben los reportes en el espacio interseccion ha sido eliminado lo cual conduce a:

$$\frac{N!}{N_k'!(N_k - N_k')!} = \frac{N!}{N_k!}\,\frac{N_k!}{N_k'!(N_k - N_k')!}$$

Hemos logrado relacionar los espacios $C_{a_k^* \cap (N', (N-N'))}$ y C/a_k^* a través del número de maneras posibles en que pueden ser generados.

El factor

$$\mathcal{R}_{C/a_k^*} = \frac{N_k!}{(N_k'!(N_k - N_k')!)}$$

nos representa el número de mensajes dado el valor k-ésimo de atributo, y de acuerdo con (1), tenemos

$$CI(C/a_k^*) = K \ln \left[\frac{N_k!}{N_k'!(N_k - N_k')!} \right]$$

o de acuerdo con (2)

$$Ci(C/a_k^*) = -K \left[P_k \ln(P_k) + P_k' \ln(P_k') \right]$$

donde: P_k es la probabilidad de obtener la falla a dado a_k^*
P_k' es la probabilidad de obtener la falla b dado a_k^*

Por otra parte el factor

$$\mathcal{R}_{C_{a_k^* \cap (N', (N-N'))}} = \frac{N!}{N_k'!(N_k - N_k')!}$$

nos representa el número de mensajes formados con el valor k-ésimo de atributo y tipos de falla, sin importar el orden de los N reportes, y de acuerdo con (1), se tiene

$$CI(C_{a_k^* \cap (N', (N-N'))}) = K \ln \left[\frac{N!}{(N_k'!(N_k - N_k')!} \right]$$

o de acuerdo a la ecuacion (2)

$$Ci(C_{a_k^* \cap (N', (N-N'))}) = -K \left[P_{(N',k)} \ln(P_{(N',k)}) + P_{(N-N',k)} \ln(P_{(N-N',k)}) \right]$$

donde: $P_{(N',k)}$ es la probabilidad de obtener el primer tipo de falla con el valor k-ésimo de atributo
$P_{(N-N',k)}$ es la probabilidad de obtener el segundo tipo de falla con el valor k-ésimo de atributo

Por otra parte, si nos fijamos en el factor $N!/N_k!$ el cual nos representa el número de posiciones posibles para ser ocupadas por los N_k elementos distinguibles con valor k-ésimo de atributo del total N. Dicho factor puede considerarse como una ponderación sobre el valor k-ésimo. Por lo que puede interpretarse como la probabilidad de obtener un valor k-ésimo de atributo sin importar el tipo de falla. En consecuencia, se tendrá

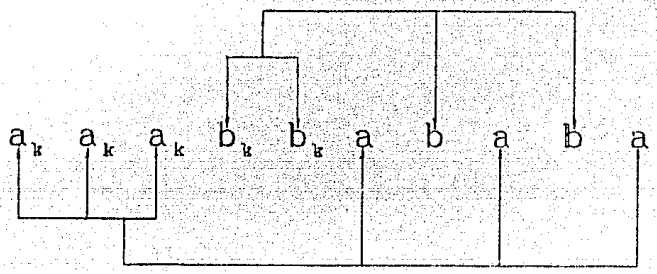
$$Ci(C_{a_k^* \cap (N', (N-N'))}) = P(a_k^*) Ci(C/a_k^*)$$

para el k-ésimo valor del atributo lo cual relaciona la cantidad de información en ambos espacios.

Se ha logrado determinar una expresión formal para la cantidad de información en el k-ésimo valor de un atributo. Para tomar en cuenta los valores restantes , hay que notar que estos valores son independientes entre si; por lo tanto, usando la fórmula de probabilidades totales se tiene que la cantidad de información contenida en un atributo es:

$$Ci(/A_i^*) = \sum_{k=1}^{l} Ci(C_{a_k^* \cap (N', (N-N'))})$$
$$= \sum_{k=1}^{l} P(a_k^*) Ci(C/a_k^*)$$

Para ilustrar los espacios anteriores daremos unos ejemplos, empezando por el espacio de intersección, consideremos la siguiente sucesión particular:

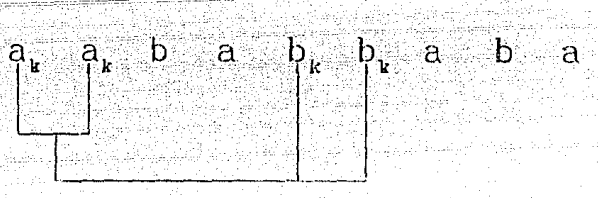


En este caso, tenemos una sucesión de $N = 10$ elementos, donde $N' = 6$ son de un tipo, digamos la falla a, y $N - N' = 4$ del otro tipo, la falla b. La k como subíndice de algunos elementos de la sucesión, señala que esos elementos tienen asociado el k-ésimo valor de atributo.

Entonces permutaremos los N elementos de la siguiente manera: de los $N' = 6$ elementos, observamos que $N_k' = 3$ tienen el valor a_k^* de atributo; entonces las permutaciones entre los $N' = 6$, serán formadas por $N_k' = 3$, y $N' - N_k' = 3$, que tienen el primer tipo de falla a. Pero si observamos los N_k' poseen su k-ésimo valor de atributo, y las restantes no poseen ese valor de atributo; no pudiendo distinguirlos, así que solo se suprimirán las repeticiones debidas a los N_k'. Con lo que se tendrán $N'!/(N_k')!$ permutaciones. Y para los $N - N' = 4$ elementos del segundo tipo de falla, se sigue un razonamiento similar, llegándose a la expresión $(N - N')!/(N_k - N_k')!$, y son las permutaciones de $(N - N')$ donde $N_k - N_k'$ son de un tipo de falla, y k-ésimo valor de atributo, y $(N - N') - (N_k - N_k')$ de la misma falla, pero sin importar su valor de atributo. Por lo que las permutaciones de N elementos, con las características anteriores es:

$$(N'!/N_k'!)((N - N')!/(N_k - N_k')!) = (6!/3!)(4!/2!)$$

Ahora para ilustrar el espacio condicional consideremos el siguiente ejemplo, sea la siguiente sucesión particular



El subíndice k me indica a los elementos de la sucesión que poseen el valor a_k^* de atributo. Donde se tienen $N = 9$ elementos, de los cuales $N' = 5$ tienen el tipo de falla a y $N'_k = 2$ tienen el valor k-ésimo de atributo; de los restantes, $N - N' = 4$ tienen el tipo de falla b y $N_k - N'_k = 2$ tienen el valor k-ésimo de atributo. Solo se permutarán elementos que tengan el mismo valor de atributo . Entonces la primera posición puede ser ocupada por 4 elementos, la segunda por 3, así hasta la cuarta que puede ser ocupada por 1 elemento, teniéndose $N_k!$ permutaciones, pero como N'_k son de un tipo de falla, y las restantes $N_k - N'_k$ del otro, entonces debemos quitar las repeticiones debidas al permutar entre si elementos de cada tipo de falla, con lo que se tendrán $N_k!/(N'_k!(N_k - N'_k)!) = 4!/(2!2!)$ permutaciones.

4.3 Clasificación eficiente de la información

Con la medida determinada en la sección anterior puede procederse para determinar una clasificación eficiente de la información. Como se está generando una clasificación de los N reportes por cada atributo, entre menor sea la cantidad de información (mayor probabilidad) asociada a algún atributo, querrá decir que hay una menor incertidumbre en la clasificación de los N reportes dada por ese atributo, por lo que para generar una clasificación eficiente de estos se eligirá entre los posibles atributos, aquel que tenga una menor incertidumbre de clasificación.

4.4 Implementación del Programa de Inducción

Consideremos la siguiente base de datos consistente de los siguientes reportes, capturados por Captura II:

```
clase     (["causa","termofluencia","sobrecalentamiento","corrosión_cáustica"])
causa     (["termofluencia","sobrecalentamiento","corrosión_cáustica"])
atributos (["la_posición_del_tubo","pared_de_agua", "sobrecalentador"])
atributos (["tipo_de_deformación","elongado","abombado",
           "ligero_abombamiento"])
atributos (["tipo_de_fractura","boca_de_pescado","picaduras"])
lis_nom   (["la_posición_del_tubo","tipo_de_deformación",
           "tipo_de_fractura"])
dato      ([ ],"única","termofluencia",["sobrecalentador","abombado",
                                        "boca_de_pescado"])
dato      ([ ],"única","sobrecalentamiento",["pared_de_agua","elongado",
                                             "boca_de_pescado"])
dato      ([ ],"única","corrosión_cáustica",["pared_de_agua",
                                             "ligero_abombamiento","picaduras"])
dato      ([ ],"única","sobrecalentamiento",["pared_de_agua","abombado",
                                             "boca_de_pescado"])
dato      ([ ],"única","sobrecalentamiento",["pared_de_agua","abombado",
                                             "picaduras"])
```

A continuación aplicaremos el algoritmo presente en el programa GENERA a la base anterior para ilustrarlo. Dicho algoritmo es implementado a través de un árbol utilizando una estrategia de control hacia adelante.

Como primer paso, selecciona un primer atributo, a saber, posición del tubo para clasificar el conjunto de reportes a partir de sus valores: sobrecalentador y pared de agua, generándose la clasificación mostrada en el árbol de la Figura 4.1.

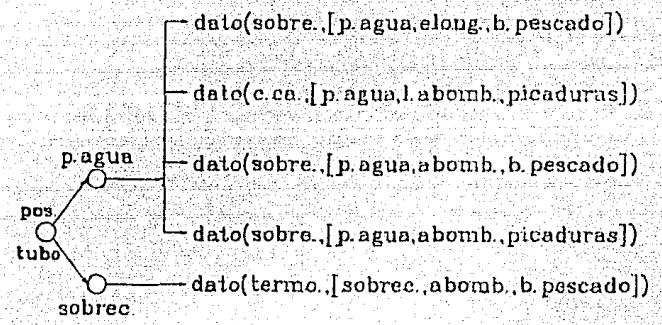


Fig. 4.1 Conjunto de datos particionados a través del atributo "posición del tubo".

Calcula el desorden asociado al atributo "posición del tubo"

$$
\begin{aligned}
Ci(/\textit{posición del tubo}) &= -\left[\frac{4}{5}\left\{\frac{3}{4}\log_2\left(\frac{3}{4}\right)+\frac{1}{4}\log_2\left(\frac{1}{4}\right)\right\}+\frac{1}{5}\left\{1\log_2(1)\right\}\right] \\
&= -\left[\frac{4}{5}\left\{\frac{3}{4}(-.4150)+\frac{1}{4}(-2.0)\right\}+0\right] \\
&= -\frac{4}{5}(-.3112-.5) = -\frac{4}{5}(-.8112) \\
&= .649
\end{aligned}
$$

Igualmente genera los árboles para los atributos restantes: tipo de fractura y tipo de deformacion, como se muestran en las Figuras 4.2 y 4.3, y calcula sus desordenes.

$$
\begin{aligned}
Ci(/\textit{tipo de fractura}) &= -\left[\frac{3}{5}\left\{\frac{1}{3}\log_2\left(\frac{1}{3}\right)+\frac{2}{3}\log_2\left(\frac{2}{3}\right)\right\}\right] \\
&\quad -\left[\frac{2}{5}\left\{\frac{1}{2}\log_2\left(\frac{1}{2}\right)+\frac{1}{2}\log_2\left(\frac{1}{2}\right)\right\}\right] \\
&= -\left[\frac{3}{5}\left\{\frac{1}{3}(-1.5849)+\frac{2}{3}(-.5849)\right\}+\frac{2}{5}\left\{\frac{1}{2}(-1)+\frac{1}{2}(-1)\right\}\right] \\
&= -\left[\frac{3}{5}(-.5283-.3899)+\frac{2}{5}(-1)\right] \\
&= -\left[\frac{3}{5}(-.9182)-\frac{2}{5}\right] = .5509 - .4 \\
&= .9509
\end{aligned}
$$

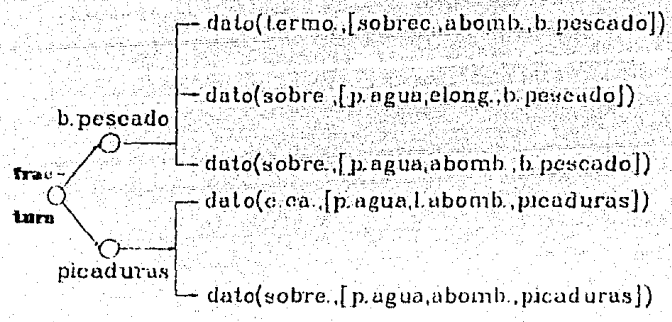


Fig 4.2 Conjunto de datos particionados a partir del atributo 'tipo de fractura

$$
\begin{aligned}
Ci(/\textit{tipo de deformación}) &= -\left[\frac{1}{5}\{1\log_2(1)\} + \frac{3}{5}\left\{\frac{1}{3}\log_2\left(\frac{1}{3}\right) + \frac{2}{3}\log_2\left(\frac{2}{3}\right)\right\}\right] \\
&\quad -\left[\frac{1}{5}\{1\log_2(1)\}\right] \\
&= -(0 - .5509 + 0) \\
&= .5509
\end{aligned}
$$

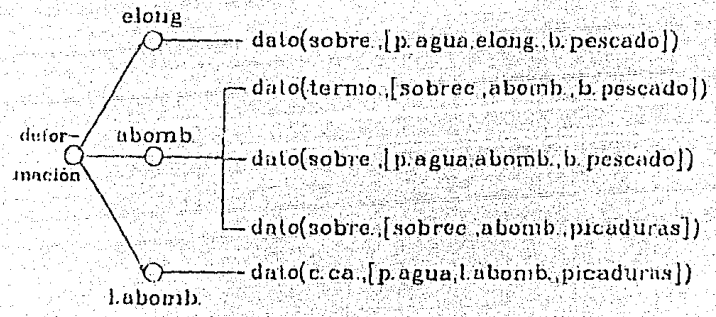


Fig 4.3 Conjunto de datos particionados a partir del atributo tipo de deformación"

De los atributos asociados a los 3 árboles anteriores selecciona al que clasificó mejor a la información, a saber deformación, para seguir extendiendo su árbol asociado hasta tener todas sus ramificaciones completas. Observamos en la Figura 4.3, que cuando el atributo deformación tiene el valor elongado, la causa es sobrecalentamiento, y cuando su valor es ligero abombamiento la causa es corrosión cáustica, por lo que no es necesario seguir clasificando a los subconjuntos de datos asociados. Decimos entonces que se tienen dos ramas completas del árbol. Sin embargo para el valor abombado se requiere una nueva partición de su subconjunto de elementos.

El mecanismo de control actuará entonces sobre el árbol asociado a deformación. Toma el nodo "elongado", y verifica si su rama asociada está completa. Como es el caso, pasa al siguiente nodo "abombado" para seguir expandiendo esta rama.

Ahora genera las ramificaciones para cada atributo distinto de tipo de deformación a partir del valor "abombado". De lo anterior se tiene la siguiente Figura 4.4.

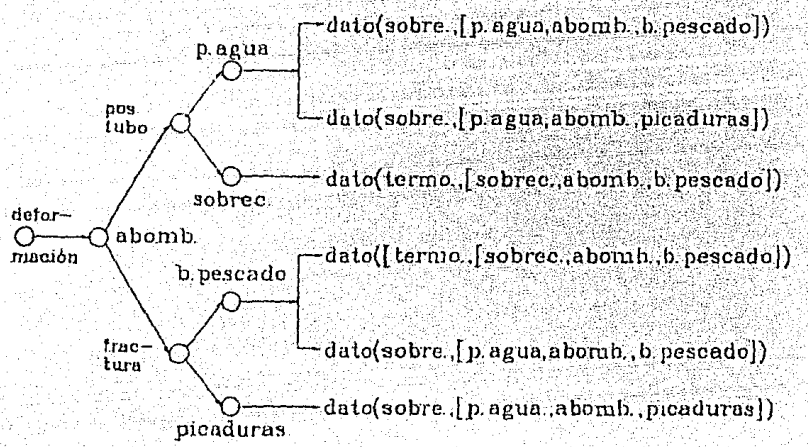


Fig. 4.4 Expansión de la rama de "abombado" a través de los atributos "posición del tubo" y "tipo de fractura".

De estas nuevas particiones a partir del nodo "abombado" calcula de la misma manera el desorden respectivo para cada atributo, resultando el atributo mínimo el correspondiente a posición del tubo. A continuación verifica si las ramas asociadas a los nodos de los valores pared de agua y sobrecalentador están completas, lo cual es afirmativo. Por último regresa al nodo "ligero abombamiento" que dejó pendiente, y como su rama esta completa se termina el proceso, ya que ahora todas las ramas del árbol generado a partir de tipo de fractura lo están.

Las reglas se producen siguiendo cada rama completa del árbol. Cada regla es una serie de condiciones de atributos y valores de atributos y una conclusión. A continuación se dá la salida del programa Genera, sobre el conjunto de reportes dispuestos inicialmente.

```
atributos  ([”la_posición_del_tubo”,”pared_de_agua”, ”sobrecalentador”])
atributos  ([”tipo_de_deformación”,”elongado”,”abombado”,
             ”ligero_abombamiento”])
atributos  ([”tipo_de_fractura”,”boca_de_pescado”,”picaduras”])
lis_nom    ([”la_posición_del_tubo”,”tipo_de_deformación”,
             ”tipo_de_fractura”])
falla      (”sobrecalentamiento”,[”elongado”])
falla      (”sobrecalentamiento”,[”abombado”,”pared_de_agua”])
falla      (”termofluencia”,[”abombado”,”sobrecalentador”])
falla      (”corrosión_cáustica”,[”ligero_abombamiento”])
```

CONCLUSIONES

Utilizar herramientas de sistemas expertos para brindar un diagnóstico de fallas en tubería de caldera resulta importante, ya que la confiabilidad de un diagnóstico radica en una base de conocimiento completa y especializada, lo cual involucra que el sistema vaya adecuando su conocimiento con la nueva evidencia o experiencia que vaya surgiendo en el transcurso de las consultas al mismo. Sistemas de cómputo tradicionales, informáticos, crean más problemas que soluciones al tratar de resolver el problema.

El lenguaje PROLOG resulta ser adecuado para el desarrollo de sistemas expertos, al menos de diagnóstico. La manera declarativa de plantear los problemas, sus mecanismos de control tanto internos como los que se implementan a partir de ellos, y las estructuras de datos para representar el conocimiento, facilitan el desarrollo de dichos sistemas.

Por lo que respecta al presente sistema, su característica principal es que puede generar nuevo conocimiento, en forma de reglas, a partir de datos empíricos. Además cuenta con un mecanismo de soporte que le permite manejar datos incompletos y/o inconsistentes para mantener credibilidad en la base de conocimiento.

El sistema ha sido evaluado con la base de conocimiento, que se generó en un principio hacia la investigación de la causa raíz, y con la captura de alrededor de 100 datos sobre tubos fallados, los cuales fueron procesados en forma de reglas . Ambas bases resultaron similares en términos generales, donde la diferencia principal está que en la base generada por el sistema hay menor redundancia y, esto es debido al método de aprendizaje presente, el cual reduce redundancia en la base, teniéndose entonces un tiempo de respuesta menor al utilizar esta base el programa Encuentra.

Por lo que respecta al programa Completez que contrastaría las bases anteriores, resultando una final, es importante de realizar su futura implementación, ya que el modelo lógico inmerso en este programa sería el fundamento para contrastar no solo dos bases de conocimiento sino múltiples bases, generadas de distintos métodos de aprendizaje, necesarios para tener distintos tipos de conocimiento y, llegar a una conclusión más real a través de consultar la base de conocimiento final.

También es necesario implementar un programa que proporcione las medidas correctivas, las cuales se podrían identificar con ayuda del programa Enseña.

Hay que señalar, que el sistema esta diseñado para tratar no solo la problemática de fallas en tuberías, ya que tiene la facilidad de capturar la información inicial acerca de cualquier otro problema en donde se pretenda obtener un diagnóstico.

Finalmente diremos, que el sistema con que se cuenta actualmente es un primer intento que se realiza; no ha sido utilizado ampliamente y, tampoco evaluado por un experto en el área. Su capacidad de respuesta estará en función de que empieza a trabajar con hechos reales y, sus alcances y limitaciones también serán determinadas en el transcurso de su utilización.

Si bien el sistema se encuentra en una etapa inicial, en la que no ha sido ampliamente usado, cabe esperar un comportamiento en relación a su implementación y diseño, por lo que vale la pena decir lo siguiente:

El lenguaje Prolog es una buena herramienta para la construcción de este sistema, ya que posibilita una gran facilidad en manejar la base de datos para mantener su credibilidad; asimismo su mecanismo de control de backtracking es útil para tomar los hechos relevantes y así evitar una búsqueda inecesaria por situaciones sin importancia, con lo que permite un procedimiento inferencial más óptimo presente en el programa Genera. La parte del cálculo númerico del programa Genera, puede ser mejorada utilizando menormente la inserción de estructuras de datos al manejar dinámicamente la memoria , cuestión que es importante de considerar si se cuenta con una disponibilidad de RAM no muy grande y, una base de datos suficientemente grande; el sistema actualmente se protege de esto refrescando la base de datos dinámica, es decir, limpiándola en pasos periódicos, determinados por la máquina inferencial del programa Genera de acuerdo a las hipótesis, acerca del estado actual de búsqueda de solución, sin embargo esta limpieza periódica consume tiempo de ejecución, por lo que se hace necesario implementarlo de otra manera.

La implementación del modelo lógico inmerso en el programa Soporte, se considera que es adecuada ya que esta utiliza apropiadamente el manejo de una recursividad eficiente con acumuladores, ahorrando espacio de memoria en el Stack, en la manipulación de la información.

Por otra parte el programa Encuentra esta limitado y solo puede operar a las reglas diseñadas en la forma de representación del conocimiento , haciendolas operativas a 4 niveles de nodos; el primer nivel dado por las conclusiones, que necesariamente son nodos tipo "Y", el segundo nivel de nodos, que representan a las condiciones de cada conclusión son nodos tipo "O", en el tercer nivel se tienen nodos sucesores de dada condición, siendo nodos tipo "Y", finalmente en el cuarto nivel se tienen nodos tipo "Y". Se podría llegar a ampliar el programa Encuentra para operar reglas a más niveles de nodos y, en cada nivel se pudieran tratar nodos de ambos tipos, a excepción del nivel de las conclusiones que son necesariamnete nodos tipo "Y", para poder representar un conocimiento más específico en forma de reglas.

En suma se cuenta con un sistema que puede ser optimizado en algunos aspectos y, ampliado respecto a las consideraciones hechas para hacer más eficiente su uso o empleo.

Para utilizar este sistema se requiere una máquina IBM AT ó XT, ó compatibles, con un mínimo de 640 kb de RAM.

BIBLIOGRAFIA

[1] Ashley,G.K., Duggan,P.E., Shor,S.W.W., Teller,A.,
Power Engineering 1988, Expert Systems Provide Help in Life Extension, Availability Improvement.

[2] Bañares,René,
Introducción a los Sistemas Expertos,
Memorias de los Seminarios Internos Organizados por el cómite de CAD/CAM del Instituto de Investigaciones Eléctricas, Mayo de 1988.

[3] Barr,Auron, Feigenbaum,Edward A.,
The Handbook of Artificial Intelligence
Volume I, II, III, Heuristech Press William Kaufmann, Inc., 1982.

[4] Clocksin,W.F., Mellish,C.S.,
Programming in Prolog,
Springer-Verlag, 1987.

[5] Crawford,Franzo H.,
Heat, Thermodynamics and Statistical Physics,
Harcourt,Brace and World,Inc., 1963.

[6] Forsyth,Richard,
Expert Systems: Principles and Case Studies,
Chapman and Hall, 1984.

[7] Ghezzi,Carlo, Jazayeri,Medi,
Programming Language Concepts,
John Wiley and Sons,Inc., 1982.

[8] Gottlob,Georg, Zicari,Roberto,
Closed World Databases Opened trough Null Vallues, 1988.

[9] Gutting,Ralf Hartmut, Zicari,Roberto, Choy,David M.,
An Algebra for Structured Office Documents, 1988.

[10] Hermann,Haken,
Synergetics, An introduction,
Springer-Verlag, 1983.

[11] Hamming,Richard W.,
Coding and Information Theory,
Prentice Hall,Inc., 1980.

[12] Harris,B.,
Theory of Probability,
Adisson-Wesley, 1966.

[13] Hayes-Roth, Frederick, Waterman, Donald A., Lenat, Douglas B.,
Building Expert Systems,
Addison-Wesley Publishing Company, Inc., 1983.

[14] Kolokouris, Angelos T.,
Machine Learning,
BYTE November 1986.

[15] Morales, Eduardo,
PROLOG (PROgramming in LOGic),
Memorias de los Sistemas Internos Organizados por el Cómite de CAD/CAM del Instituto de Investigaciones Eléctricas, Mayo de 1988.

[16] Nilsson, Nils J.,
Principles of Artificial Intelligence,
Tioga Publishing Company, Palo Alto, CA., 1980.

[17] Ramachandran, Bharath.
Information Theory,
BYTE December 1987.

[18] Saldaña, Aldana, Héctor,
Sistemas Expertos en Ingeniería,
I.P.N., Informe Técnico, 1986.

[19] Singh, G.P.,
Boiler-Tube Failure Prevention Using Expert System Aproach, 1987.

[20] Thompson, Beverly, Thompson, William,
Finding Rules in Data,
BYTE November 1986.

[21] Waterman, Donald A.,
A Guide to Expert Systems,
Addison-Wesley Publishing Company, Inc., 1986.

[22] Weiskamp, Keith, Hengl, Terry,
Artificial Inteligence Programming with Turbo Prolog, John Wiley and Sons, Inc., 1988.

[23] Weiss, Sholom M., Kulikowski, Casimir A.,
A Practical Guide to Designing Expert Systems,
Rowman and Allenheld Publishers, 1984.

[24] Weiss, Volker,
A Materials Failure Analysis Expert System Based on Logic Programming, 1987.

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

[26] *Turbo Prolog, Owner's Handbook*,
Borland International, Inc., 1986.