T E S I S D O C T O R A L

USO DE UN MODELO DE APRENDIZAJE PARA UN SISTEMA COMPLEJO DE DIAGNOSTICO INDUSTRIAL CON LIMITACION TEMPORAL

por

Jesús CARDEÑOSA LERA

Ingeniero aeronaútico por la E.T.S.I. Aeronaúticos de Madrid.

Presentado en la

FACULTAD DE INFORMATICA

de la

UNIVERSIDAD POLITECNICA DE MADRID

para la obtención del

Grado de Doctor en Informática

MADRID, DICIEMBRE DE 1987

USO DE UN MODELO DE APRENDIZAJE PARA UN SISTEMA COMPLEJO DE DIAGNOSTICO INDUSTRIAL CON LIMITACION TEMPORAL

RESUMEN

Tras una introducción a la necesidad de estudio en este tema, se muestran las distintas ramas que los investigadores van siguiendo en la actualidad, exponiéndose las diferencias entre el diagnóstico médico y el industrial, así como la necesidad de estructurar el conocimiento del problema diagnóstico. La aproximación a tiempo real como objetivo y la definición de los sistemas complejos caracterizan el problema propuesto, como método de resolución del problema del diagnóstico, bajo condiciones de tiempo limitado en la respuesta. Como resolución a este problema, se proponen una serie de procedimientos integrados que permiten dar una respuesta según el tiempo disponible y que se resumen en:

- Procedimiento de construcción de un árbol de fallos a partir del conocimiento en forma de reglas.
- Procedimientos de depuración estructural del árbol de fallos.
- Nuevo procedimiento de construcción del conjunto de Conjuntos Mínimos, puerta a puerta del árbol.

- Método de resolución de incertidumbre en los Conjuntos Mínimos en base a parámetros de fiabilidad para el caso de tiempo suficiente.
- Definición del concepto de Conjunto Virtual como procedimiento de resolución de tipo estructural, del problema del diagnóstico.
- Método de resolución de incertidumbre en Conjuntos Virtuales basado en parámetros de fiabilidad.

Los métodos propuestos permiten, desde la detección de inconsistencias en el conocimiento, hasta la posibilidad de diagnóstico incompleto, pero seguro, cuando el tiempo es insuficiente, como caracterización del problema del diagnóstico en emergencias.

USE OF A LEARNING MODEL FOR A COMPLEX INDUSTRIAL DIAGNOSIS SYSTEM, WHEN THE TIME IS LIMITED

ABSTRAC

After introducing the necessity of studying this subject, we present the different approaches currently followed by researchers and the differencies between medical and industrial diagnosis as well as the need to structure the knowledge of the diagnosis problem. The approach to real time as an objective and the definition of complex systems characterize the proposed problem as a method to solve the diagnosis problem under limited time conditions in response. To solve this problem a series of integrated procedures are proposed which allow a response to be given according to the available time. These can be summarized as follows:

- Construction procedure of a failure tree starting from knowledge in rule-ways.
- Structural debugging procedures of a failure tree.
- New construction procedure of the set of minimal sets, gate-to-gate of the tree.
- Method of solving the uncertainty in the Minimal Sets based on reliability parameters in the case of sufficient time.
- Definition of the Virtual Set concept as a solving procedure of structural type in the diagnosis problem.
- Method of solving the uncertainty in Virtual Sets according to reliability parameters.

The proposed methods permit not only to detect inconsistencies in the knowledge but also to make an incomplete but safe diagnosis when time is insufficient. These can be considered the specific features of the diagnosis problem in emergencies.

INDICE

PAG.

FE DE ERRATAS

PAG. 127- LINEA 14.....debe poner :(conjunción lógica)

PAG. A1-1; debe añadirse al final una regla más, que sería:

D ——————→ E

T I T U L O

"USO DE UN MODELO DE APRENDIZAJE PARA UN SISTEMA COMPLEJO DE DIAGNOSTICO INDUSTRIAL CON LIMITACION TEMPORAL.

I N T R O D U C C I O N

El objeto de esta Tesis es centrar uno de los problemas candentes en el mundo industrial, clasificarlo, estudiar el estado actual de las investigaciones y proponer un método de mejora, en la dirección que se explica brevemente a continuación.

En los grandes sistemas industriales que se denominan complejos, se determinan normalmente unos comportamientos desde el diseño denominados como de alta fiabilidad. Normalmente, la fiabilidad, queda definida en base a una necesidad de seguridad, ya sea porque la prestación para la que han sido diseñados es de vital importancia política ó económica, ó bien por incidir su funcionamiento, -ó su mal funcionamiento-, en un grupo humano más ó menos numeroso.

Estos sistemas se denominan complejos en base a una serie de características que se pueden resumir en:

* El conjunto de sistemas y subsistemas que los componen es llevado y atendido por equipos humanos muy diferenciados.

* El número de componentes, decenas de miles, es tan grande que resulta inviable tener un control sobre todos ellos de forma directa.

Ejemplos de estos sistemas pueden ser las plantas productoras de energía eléctrica (térmicas ó nucleares), las plantas de la industria química y el campo aeronaútico, principalmente.

Estos sistemas, una vez construidos, funcionan dentro de los parámetros de diseño, gracias a los operadores que las vigilan y atienden. Sin embargo, la propia complejidad de los sistemas por ellos vigilados, hace que, con cierta frecuencia, se produzcan fallos de componentes ó subsistemas completos.

En caso de uno de estos fallos, la misión del operador es actuar sobre el sistema según su criterio y decisión, en base a su formación y documentación.

En sistemas, como por ejemplo, el caso nuclear ó incluso el aeronaútico y dada la gran responsabilidad de las decisiones que han de tomarse, el "stress" al que el operador se encuentra sometido es el que consigue que en casos de emergencia grave, las decisiones basadas en su razonamiento-diagnóstico de la falla, son erróneas en un 50-70% en el caso nuclear y llegan al 90% en el caso aeronaútico.

Esta situación, ha obligado a instituciones como el EPRI y a partir del importante accidente de la isla de las Tres Millas (Harrisburg) a iniciar y patrocinar investigaciones para corregir esta situación.

Las investigaciones tienen un ámbito de aplicación mundial y en ellas ha entrado por la puerta grande la Inteligencia Artificial,(en lo sucesivo I.A.). Estas investigaciones tienen como objetivo casi mayoritario los sistemas de ayuda a la decisión oyéndose de forma insistente, dos conceptos:

- Diagnosis de fallos.
- Tiempo Real.

Quizás sea en el campo del diagnóstico donde la I.A. , haya logrado casi desde sus comienzos, éxitos más espectaculares y donde quizás tenga los futuros.

Hay que diferenciar diagnóstico médico del industrial; cuestión importante al partirse de supuestos bien distintos.

Nos queda por aclarar brevemente que se entiende por Tiempo Real.

Parece claro que en el campo industrial y ante una emergencia grave, son necesarias decisiones de una prontitud extrema. Es más, el colapso mental que sufre el operador en algunas situaciones hace que su indecisión ante una acción a tomar sea tan grave como el tomarla errónea, incluso a veces más. El operador en esos momentos, querría tener en sus manos algo que le ayudase a tomar alguna decisión, ni siquiera la ideal, pero que dejase el sistema en mejor situación. Para ello, necesita saber exactamente ó al menos aproximadamente, qué es lo que ha pasado. Y necesita saberlo con extremada prontitud. Aún hoy , se mal recuerdan las casi dos horas que se tardó en llegar a una conclusión en la Isla de las Tres Millas. La eficiencia es pues el objetivo. Y las investigaciones actuales van en ese sentido. Conseguir sistemas de diagnóstico más eficientes.

La frase Tiempo-Real se está utilizando para etiquetar sistemas que son simplemente más eficientes que los hasta ahora existentes. Pero una decisión en 10 segundos puede ser suficiente en un sistema y no serlo en otro. Por lo tanto "Tiempo-Real" es un objetivo al que hay que tender, pero al cual se sabe que no se va a llegar nunca de forma estricta.

En consecuencia, "tiempo real" puede sustituirse por "tiempo suficiente" y en ese sentido se propone un método que es capaz de emitir decisiones, aceptando el error, pero en el tiempo suficiente, ó al menos disponible y sabiendo que en ese tiempo el error es mínimo. Este es un objetivo que supone una aportación decisiva al problema de la eficiencia en el diagnóstico.

Se centra el problema en los sistemas industriales antes reseñados, aprovechando métodos muy específicos del entorno industrial así como de investigadores cuya actividad se centra en el campo de la I.A.

Quizás el problema sea la gran diversificación, y la falta de comunicación de las personas dedicadas a estos asuntos. Se puede, sin embargo, afirmar que la propuesta que se hace en esta Tesis, no ha tenido precedente hasta el momento, cuestión que puede corroborarse, a la vista de la muy reciente documentación a la que se hace referencia.

Finalmente se explicará mediante unos ejemplos la forma procedimental de este método aún con las limitaciones de espacio derivadas del hecho de ser aplicado sobre lo que en uno de los sistemas, que son objeto de este trabajo, sería tan sólo un subsistema de los que realmente podrían ser atacados con él.

El problema y consiguiente método que se propone es específico, pero dotado de la suficiente estructura como para ser de general aplicación en una gran parte de las necesidades de diagnóstico en sistemas complejos. Propuesto este método quedan abiertas varias líneas de investigación, unas , para su probable mejora, y otras muchas para llegar a un logro más lejano que es la predicción sistematizada de fallos, y que se indicará al final de este trabajo.

CAPITULO SEGUNDO

E S T A D O D E L A C U E S T I O N

1.-INTRODUCCION

El problema que se va a tratar de describir en su evolución, desde su planteamiento,-que puede atribuirse a los años 50-,hasta el momento actual, tiene tres características diferenciadoras pero muy integradas en el campo de aplicación industrial.

Una primera sería el concepto de diagnóstico. Aunque de forma general puede decirse que es la averiguación de la causa de un fallo en un determinado sistema en estudio ú observación, se verá más adelante como existen características diferenciadoras entre el diagnóstico mèdico y el industrial, que son sin duda parte de las causas que hacen que los métodos no sean siempre transportables.

Muchos investigadores en I.A. atacan este problema ya en MYCIN [MELL-84], que sin duda supone una estructuración del conocimiento diagnóstico, permitiendo además un tipo de razonamiento bajo incertidumbre muy clásico en medicina.

El diagnóstico industrial nace, sin embargo, como necesidad de estudio cuando los sistemas a estudiar, adquieren un volumen tanto cualitativo como cuantitativo, bastante considerable.

Esta va a ser la segunda característica diferenciadora, es decir, el hecho de tener que realizar la labor del diagnóstico en sistemas complejos. La caracterización de un sistema complejo no está aún bien determinada, pero como tal fenómeno, ya ha habido estudios en ese sentido. Más adelante se volverá con detalle a comentar esto. Tan solo decir en principio que cuando Basden [BASD-84] comenta sobre las aplicaciones de los Sistemas Expertos, (en lo sucesivo SS.EE.) establece unas limitaciones sobre lo adecuado de éstos para resolver problemas, dando una medida de la complejidad por el número de reglas; en su artículo nos dice que los SS.EE. no son adecuados para problemas complejos, midiendo estos como aquellos que requieran más de 10.000 reglas. Sin embargo, hoy con XCON y XSEL este número se supera. No parece una medida muy adecuada de la complejidad pues tan solo nos da una limitación cuantitativa sin pensar en la estructura del conocimiento contenida en ese conjunto de reglas.

La posibilidad de descomposición del problema y tratamiento del mismo por medio de creación de contextos ú otras formas, hace que el número de reglas, no sea una buena medida de la complejidad de un problema.

Hay además una importante cuestión a considerar. El estudio y tratamiento de sistemas complejos, al menos en el mundo industrial, es atacado en la actualidad por medio de métodos específicos, pero que quedándose cortos aún, se está recurriendo de forma sistemática a los métodos empleados en I.A. Dicho de otra manera, se está recurriendo a la I.A. precisamente para resolver este tipo de problemas, al no dar más de sí los métodos convencionales.

Queda la tercera característica diferenciadora de nuestro problema. La constricción temporal. Siempre es buena la eficiencia, en la resolución de cualquier problema. En un problema de diagnóstico, más aún si cabe. Sin embargo en medicina por ejemplo, la necesidad de efectuar análisis y por tanto el tiempo empleado en ellos, hace que la constricción temporal en cuanto a la respuesta, no sea apremiante la mayor parte de las veces.

En el campo industrial y en ciertas áreas (centrales nucleares, industria química, industria aeronaútica) en las que la pérdida de tiempo en la diagnosis de un fallo puede originar una rápida degradación del resto del sistema, el tiempo es un factor clave.

El accidente de la Central Nuclear de la Isla de las Tres Millas, con una tardanza de algo más de 2 horas en saber qué era exactamente lo que pasaba, hizo entrar al sistema en situación de irreversibilidad.

La constricción temporal es en sí una necesidad. Pero es precisamente en estos tipos de sistemas donde actúa como fuerte elemento de "stress" sobre el siempre presente operador.

Es decir, estos sistemas deben su actuación ante fallos a la acción de un ser humano, ante quien dicha constricción actúa como un elemento importante, generador de fallos de tipo humano.

Se han hecho estudios [HAUP-86] en ese sentido, evaluando la fiabilidad del operador en función del tiempo disponible para la respuesta, según la gravedad aparente del evento acaecido, e introduciendo una probabilidad adicional de fallo por este concepto, en la total del sistema.

En los momentos en que un fallo es detectado por el operador, el saber su causa para poner remedio a tiempo, produce un "vuelco" en la mente del operador.

Es en esos momentos, cuando el operador necesitaría saber "algo", aunque sea poco, de lo que puede estar ocurriéndole al sistema. Y ese "algo" necesita saberlo desde el primer momento. En ese sentido el "diagnosticador automático" actúa como un sistema de ayuda a la decisión, y esa es quizás la última característica del problema que se va a estudiar. Un claro ejemplo es el aumento de la eficacia en la labor del operador, cuando en los estudios previos del S.E. "Smokey" se comprobó que el mero hecho de "ralentizar" las señales de llegada a las consolas mediante una cinta de video, disminuía de tal forma el "stress" del operador que sus actuaciones aumentaban el porcentaje de éxitos del 50% al 90%. Este es quizás uno de los SS.EE. que atacó frontalmente el problema del "stress" del operador ante una avalancha de información.[XERO-86].

Este planteamiento de "tomar la mejor decisión en el mínimo tiempo", ó al menos, "la mejor decisión en cada momento" es ya propuesto por la industria de estas ramas principalmente, como un problema de Tiempo Real asimilando este concepto a eficiencia, pero resolviendo el problema de forma exhaustiva.

Aquí el planteamiento difiere ligeramente de éste, dado que al acotar el tiempo disponible, se renuncia a la exhaustividad. A cambio, acotamos el abanico de posibilidades de fallo ante las que el operador debe tomar su decisión.

Este problema queda bastante bien definido por Michalski [MICH-86], y, de alguna manera, es atacado por Haddaway [HADD-86] basándose en un modelo de la "Lógica de Precisión Variable".

Diagnóstico, sistema complejo y constricción temporal, definen de forma integrada un problema muy específico y que en el campo industrial adquiere relevancia especial.

De cara a su resolución ó propuesta de solución hay que hacer mención a dos cuestiones adicionales. Una de ellas sería la adquisición y representación del conocimiento. El conocimiento de un sistema industrial tiene dos fuentes:

* El diseño, mediante el cual, se predicen comportamientos deseados, así como consecuencias de comportamientos de componentes que originan fallo.

* La experiencia, basada en comportamientos de sistemas similares.

En base a estas fuentes, se establecen unas reglas de comportamiento del sistema.

¿Pero, cómo se representan?

En esto hay gran variación de cara al diagnóstico e incluso corrección de defectos de diseño; en este tipo de industria se representan por medio de árboles de fallos de los que se hablará extensamente más adelante.

La otra cuestión es el aprendizaje; cuestión que está presente en estos sistemas y que aún no tiene una clara metodología.

De nuevo, el hecho de la complejidad industrial caracteriza este hecho particular.

Mientras en otros campos se estudia el aprendizaje como un determinante del cambio del razonamiento de un experto en base a la experiencia [KOLO-84] debido sin duda al tratamiento individualizado de los expertos, en el área industrial, el aprendizaje es necesario para ir aumentando la eficiencia del sistema de diagnóstico y a la par, ir reduciendo la complejidad.

Esta, viene dada en una primera medida, por la absoluta imposibilidad de encontrar un único experto que conozca el sistema de forma completa, y lo que es peor , ni siquiera a base de formación.

El aprendizaje aquí, no es entonces un problema de incorporación de nuevas reglas a la base de conocimiento sino de cómo utilizar ese conocimiento, pues en contra de lo que dice Kolodner [KOLO-84] no siempre servirá para usar mejor el que ya se tenía.

Las líneas actuales seguidas son:

* Generalización de similaridades.
* Aprendizaje reconducido.

La primera se basa en crear conceptos en base a similaridades entre hechos descritos por otros conceptos y ha sido inicialmente implementada en CYRUS (Kolodner,1983) e IPP (Lebowitz, 1983).

La segunda, basada en anotar los fallos ó excepciones y establecer la diferencia con los hechos normales, fue propuesta por Schank en 1982 e implementada por Riesbeck en 1982 en ALFRED.

En principio es compatible con el sistema complejo y los métodos convencionales seguidos hasta ahora, la idea del aprendizaje incremental por fallo dirigido; se formula una hipótesis de fallo y tras corroborarla, se corrige y se incorpora a la memoria.

Sin embargo, en el sistema industrial complejo, están por desarrollar estos conceptos de cara a una metodología eficaz.

Las tres principales lineas de investigación actual son:

* Integración de nuevos hechos en estructuras preexistentes de conocimiento.[SHAN-82]

* Formación de nuevos conceptos por inducción.[CARB-83].

* Formación de nuevos conceptos por analogías.[CARB-83].

Riesbeck, da una buena panorámica de esta problemática en [RIES-84].

El ataque a este problema en el campo industrial que se está estudiando, se hace de forma preferente en base a estadísticas. De alguna manera sigue una línea de información al sistema ya clásica,-pero sin duda adaptada a una normativa técnica-, y, por otro lado, necesaria para la configuración de datos de fiabilidad. En ese sentido, se utiliza con cierta frecuencia la teoría subjetiva de la probabilidad [PARR-81], [MART-82], en la que la clave es el empleo del teorema de Bayes y su interpretación como instrumento para combinar de forma coherente, observaciones nuevas con el estado de conocimiento anterior a dicha observación. Una buena descripción de estos métodos se puede encontrar en [HAUP-86]. Se ha demostrado que si la muestra es grande, los resultados obtenidos se acercan a los que da el método de máxima verosimilitud de Fisher [RIOS-67].

Sin embargo, surge un nuevo problema derivado del hecho de que los sistemas industriales que se están estudiando son denominados muchas veces como de alta-fiabilidad, es decir, donde sus componentes principales no fallan prácticamente nunca, con lo que las muestras siempre serán pequeñas.

Este problema en su estado actual es descrito en [ABBO-85] en el que la ayuda (no sustitución) al operador, se basa en diagnóstico ante nuevos eventos y aprendizaje por incorporación al arbol de fallos, modificando el mismo. Esto supone que el árbol anterior, ó no está completo, ó no está bien construido, y refleja sin duda la enorme dificultad que tiene su construcción en un sistema complejo.

Queda pues hecha mención a una de las lineas de investigación a todos los niveles, pero de forma muy especial en los sistemas complejos, como es el aprendizaje, que sigue abierta y sin solución clara para muchos problemas presentes hoy. Una buena descripción del estado actual del tratamiento estadístico de estos problemas puede encontrarse en [APOS-79-A] y [APOS-79-B].

En los siguientes epígrafes, vamos a una detallada descripción del problema en su momento actual tomando como base,las tres características diferenciadoras ya descritas anteriormente.

2.- CARACTERIZACION DEL PROBLEMA

2.1.-EL PROBLEMA DIAGNOSTICO

De forma simple y sin matizar, el problema del diagnóstico no es otro que dado un fallo en un sistema, encontrar la causa. La segunda fase sería el tratamiento (medicina) ó los normales de actuaciones ante fallos (industria).

Lo que parece claro es que gran parte de los problemas de diagnóstico no tienen una única causa, pero hoy por hoy se tiende a restringir el campo de los posibles fallos y mediante diversos métodos, resolver esta incertidumbre hasta tomar una decisión en cuanto a la causa de fallo más probable.

Dos de los aspectos que son objeto de investigación y que constituyen aún líneas abiertas son:

* Identificación de causas comunes a fallos múltiples y simultáneos.

* Identificación de causas múltiples a un fallo determinado.

Esta última es tratada de forma bastante clara en la industria por medio del concepto de Conjunto Mínimo [HAUP-86] y la primera, de aplicación en medicina, es tratada por Kleer y Reggia [KLEE-86], [REGG-84]. En este sentido Kleer, propone un modelo de aplicación en electrónica basado en el contraste de diferencias en los comportamientos del sistema con un modelo de funcionamiento correcto. Es decir, la detección de discrepancia. Lo que supone utilización de conocimiento profundo.

La localización del fallo concreto, se haría después en base a introducir una señal de referencia y ver cual es la salida. Este conjunto de actuaciones se denomina "test" y viene descrito en una de sus formas en [CHAN-84-B] en base a la aplicación hecha por Davis en 1983 en el sistema MDX.[CHAN-83].

El mayor inconveniente que tiene en este sistema es lo aducido al campo de aplicación, limitado a sistemas eléctricos ó electrónicos, donde es posible introducir esa señal de referencia.

Otros sistemas, como pueden ser los de tipo dinámico (hidraúlicos por ejemplo), no permiten la introducción de esas señales de referencia.

Kleer, sin embargo propone unas lineas que no son nuevas, pero no por ello, menos útiles, para el estudio de múltiples fallos simultáneos, donde el espacio de "candidatos" (causas de los fallos), crece exponencialmente [KLEE-86]. El, denomina "SINTOMA" a una diferencia entre lo predecido y lo observado aceptando que un síntoma dice que uno ó más componentes han fallado. Asimismo, llama "CONFLICTO" a un conjunto de supuestos que soportan un síntoma y plantea como objetivo de su método, la identificación de conflictos mínimos, es decir, conflictos que no tienen un subconjunto que también sea conflicto. Finalmente, genera "candidatos", es decir, hipótesis de causas de fallo.

Esta idea ya se puede encontrar como "conjunto de separación" y "conjunto mínimo" de árboles de fallos en 1972 [FUSS-72].

En la misma línea que Kleer y casi simultáneamente se encuentra a Reggia [REGG-86] ampliando el modelo "Set-Covering" ya propuesto por él mismo en 1983 [REGG-84] y basado en el concepto de "mínimo conjunto de recubrimiento", concepto exactamente igual que el denominado por Kleer "conflicto mínimo".

Más adelante se explicará este método por considerar que es un modelo de resolución del problema del diagnóstico con fallos múltiples, de los más importantes y mejor estructurados hasta el momento.

Como características principales del problema del diagnóstico, se destacan las siguientes, que posteriormente se comentan en detalle:

a) Presentación e identificación de síntomas.
b) Identificación de posibles causas.
c) Establecimiento de relaciones entre causas y síntomas.
d) Búsqueda de conjuntos "causa del fallo".
e) Resolución de incertidumbre.

2.1.1. Presentación e identificación de síntomas.

Aquí de nuevo, se encuentra la diferencia de concepto y de tratamiento, entre el diagnóstico industrial y el médico. Normalmente en la industria, la palabra síntoma, no tiene sentido en la mayoría de los casos. Esto es porque dado que las tolerancias en las medidas suelen estar bien definidas, lo normal es que si se está dentro de tolerancia, no hay fallo y si se está fuera ya es fallo. En ese sentido, un síntoma es ya de por sí un fallo. Quizás poco importante por sí solo, pero que combinado con otros comportamientos puede ser causa de fallos de mayor rango. De una forma sutil, se está diciendo que el tratamiento industrial de este problema está basado en una jerarquización de lo que se consideran fallos. De esta forma el fallo a investigar es el de mayor rango y es predefinido por los diseñadores de estos sistemas. Las causas serán fallos de menor rango.

Esta jerarquización, [CHAN-84-B] induce pues a pensar, en la existencia de una estructura.

En medicina, la identificación de síntomas abarcaría los "sensores humanos", (dolor, inmovilidad..,) y los análisis y observaciones hechas por el médico. Precisamente en este campo es donde la estructuración y jerarquización sea más deficiente y quizás por innecesaria.En ese campo, queda más patente que un síntoma es tan solo eso, un síntoma, y no en sí un fallo. El modelo entonces, puede ser diferente y quizás más eficiente, a pesar del mayor desconocimiento del cuerpo que tiene un médico que el de un ingeniero sobre la máquina. Desconocimiento lógico por otra parte, dado que el médico estudia un sistema que él no ha diseñado, por lo que de alguna manera, el cuerpo humano es para el médico, como una caja negra de la que deduce cosas por su observación tanto física como de comportamientos.

En principio, en la industria, un síntoma-fallo es detectado por la señal de un sensor, que indica que el parámetro por él medido se encuentra fuera de tolerancia.

En sistemas complejos se recurre a la combinación de señales provenientes de diferentes sensores creando así un parámetro, que es el que indica por así decirlo, un síntoma-fallo de un subsistema. Normalmente la identificación de un fallo global de un sistema ó presentación de "caída" de una función crítica es verificada en último término por el operador.

Este sistema origina muchos problemas derivados del fallo del sensor en sí, (cuestión por la cual rara vez el sensor es único), evitándose en buena parte por sistemas de sensores para una sola medida, llegándose así a estimar si lo que falla es la medida ó el sensor[HAUP-86].

Otro problema complejo derivado de la existencia de sensores, es cuando las señales de varios de ellos, indican en su combinación un fallo imposible, (por ejemplo, que una bomba esté funcionando y simultáneamente no lo esté). Este problema queda bien descrito por Chandrasekaran en [CHAN-85] con propouesta concreta mediante planteamientos de I.A. de cara a su resolución.

Asimismo Hashemi ataca este problema en un caso de aplicación en centrales nucleares [HASH-87] utilizando el lenguaje CSRL, diseñado para SS.EE. de diagnosis [BYLA-83].

Como se ve, éste epígrafe es un problema bastante complejo de estudiar y tratar, sobretodo en los sistemas industriales y susceptible sin duda de mejora. La complejidad de los sistemas de que se está hablando, hace que el número de grupos de sensores sea alto (unos 600-800 en una central nuclear), lo suficiente, por un lado, para simplificar la tarea del diagnóstico y, por otro, para hacerla más difícil en su interpretación. Más adelante se verá la importancia de una señal de sensor en la simplificación drástica del conjunto de "conjuntos mínimos". Sin embargo, esta es una característica de los sistemas complejos.

Este problema en medicina es diferente. La detección de los síntomas, es la única fuente de información ante un posible fallo.Y el síntoma es deducido de los comportamientos.

El síntoma, por otro lado, y como ya se ha comentado, raramente se confunde con la causa, pues de alguna forma la combinación de varios síntomas constituye en sí la causa ó causas de su presentación.

En ese sentido, las relaciones causales entre síntomas y causas son de un solo nivel. El método "Set-Covering" así lo muestra. La estructura causal del diagnóstico médico parece ser simple en cuanto a niveles, pero compleja en cuanto a que las causas de un síntoma ó conjunto de síntomas son varias.

El diagnóstico industrial, parte de un solo "síntoma" ó fallo principal, ó caída de función crítica, y estudia todas las causas posibles en estructuras de cierto número de niveles. (Arboles de fallos). Precisamente el problema de la detección de los fallos es estudiado en 1985 por la Japan Atomic Energy Research Institute [YOSH-85] en el S.E. denominado DISKET, como sistema de diagnóstico.

Posteriormente este S.E. fué ampliado con la posibilidad de detección de fallos antes de activarse las alarmas [BERG-87].

Estos sistemas parten de una clasificación de los fallos, así como una jerarquización de las hipótesis de fallo; Esta jerarquización viene implícitamente exigida por la existencia de varios niveles. La representación de niveles se hace en este caso, con sólamente tres, en un árbol de sucesos. La base de conocimientos está constituida por tres tipos de reglas y por definiciones. Las reglas tienen adjudicado un peso, por medio de un "factor de certeza" (FC) con valores entre -1 y 1, utilizándose sin embargo estos FC como probabilidades y por medio del teorema de Bayes.

El método es relativamente simple pero acertado. Hay que tener en cuenta, que su función, dado que solo tiene tres niveles, es la localización de fallos de un nivel alto.

Un comentario importante, [BERG-87], es ver que cuando sistemas como MYCIN, EXPERT,etc, son aplicados a los sistemas industriales, se encuentran con graves limitaciones, estableciendo como causa principal el hecho de que los anteriores están construidos para sistemas estáticos, mientras que los sistemas industriales son generalmente dinámicos.

Y efectivamente esa es una diferenciación muy clara y quizás de alto nivel entre los sistemas médicos y los industriales.

2.1.2. -Identificación de posibles causas.

En principio y para un médico, existe mayor claridad por lo ya dicho, ante esta cuestión. Es decir, la mejor diferenciación entre síntomas y enfermedades, hace que pueda colocar a un lado los síntomas y a otro las enfermedades.

Esta cuestión, puede parecer obvia, pero en los sistemas industriales, por lo ya explicado antes, no es tan fácil.

Puede producirse la paradoja de que, por ejemplo, en un sistema de bombeo en el que una válvula colocada a la salida de la bomba, falla por cierre, acabe fallando la bomba precisamente porque en principio la válvula sigue funcionando, aunque la válvula esté cerrada, siendo como son en principio sucesos independientes.

En los sistemas industriales puede ocurrir pues, que no todas las causas de un fallo son "malignas", por decirlo de alguna forma, sino que lo "maligno", es su presentación junto con otros acontecimientos y precisamente estos.

Esta idea queda muy bien resuelta en detalle por los "Conjuntos Mínimos" de un arbol de fallos, basados en la simultaneidad de ciertos acontecimientos, basados a su vez, en la situación bivalente de un determinado componente, es decir, fallo ó no-fallo.

El problema del diagnóstico busca normalmente una única causa. Kleer sin embargo, [KLEE-86] se plantea el caso de múltiples fallos simultáneos. Y lo mismo se plantea Reggia, [REGG-86] como ya se comentó anteriormente.

Sin embargo, el estado de múltiples fallos simultáneos no deja de ser un cuadro determinado y específico, de forma que sea todo ese conjunto el suceso no deseado.

Queda claro, que el espacio de candidatos ó hipótesis de fallo, es mayor en caso de múltiples fallos simultáneos que en caso de uno solo, pero hecha esta salvedad cuantitativa, no deja de tener al final un conjunto de candidatos, del cual habrá que extraer finalmente el más probable. Este cuadro pues, no difiere especialmente del de un solo fallo, hecha la salvedad cuantitativa.

Reggia [REGG-86] estudia, sin embargo, este problema, añadiendo además una hipótesis de verosimilitud que le permiten llegar al "conjunto de recubrimiento" ("set-covering") más probable, siendo éste una hipótesis de fallo. Este modelo formal de diagnóstico, basado en la "Parsimonius-covering theory" [PENG-86] ha sido estudiado en los últimos cinco años.

Este modelo se basa en las asociaciones causales entre "desórdenes ó enfermedades" (d) y "manifestaciones ó síntomas" (m), como elemento central de la base de conocimientos de diagnóstico. De esta forma el tipo más simple de problema diagnóstico, es definido como:

una cuaterna : $P = \{ D, M, C, M+ \}$

donde $D = \{d_1,\ldots,d_n\}$ = conjunto no vacío de desórdenes (finito).

$M = \{m_1,\ldots,m_k\}$ =conjunto no vacío de manifestaciones (finito).

$C \subseteq D \times M$ es una relación con $|$dominio(C)=D
$|$rango(C)=M

$M+ \subseteq M$; (M+ = manifestaciones en un caso concreto).

La relación C capta la noción intuitiva:

$\{ d_i,m_j\} \in C$

si "el desorden d_i puede causar el síntoma m_j", lo que no implica que m_j ocurra siempre que d_i esté presente.

A partir de aquí se definen dos funciones:

$\forall\ m_j \in M$ Causas$(m_j) = [\ d_i\ /\ \{d_i,m_j\} \in C\ ]$

$\forall\ d_i \in D$ Efectos$(d_i) = [\ m_j\ /\ \{d_i,m_j\} \in C\ ]$

Entonces un conjunto de desórdenes $D_I \subseteq D$ se dice que es un RECUBRIMIENTO de un conjunto de manifestaciones $M_J \subseteq M$ si:

$$M_J \subseteq \text{Efectos}\ (D_I)$$

Bien, mediante un determinado algoritmo ya expuesto por Reggia en su modelo "Set-Covering", [REGG-84], llega a determinar el conjunto D_I, aplicando el criterio de mínima cardinalidad. Dicho algoritmo es parcialmente criticable por contener aspectos arbitrarios.

En un modelo posterior, [REGG-86] mejora de alguna forma el razonamiento seguido. En éste, considera que una hipótesis de diagnóstico debe ser un RECUBRIMIENTO de M+. Por el principio de "parsimonia" ú "Occam's Razor" se adopta un primer criterio de plausibilidad: "un RECUBRIMIENTO simple es preferible a uno complejo". Y de esta forma, llega a que el conjunto de todos los RECUBRIMIENTOS simples de M+ es la solución de un problema dado.

Pero, ¿Qué es un RECUBRIMIENTO simple? ¿ En base a qué se define la naturaleza de la simplicidad?

El propio Reggia da cuatro criterios, que son:

* Restricción al desorden único:Es decir, un RECUBRIMIENTO D_I de M+ es una hipótesis plausible si solo contine un único desorden.

Nosotros decimos simplemente que no hay motivo para considerar que por tener un único desorden pueda ser una hipótesis más plausible.

* Minimalidad: Un RECUBRIMIENTO D_I de M+ es una hipótesis de fallo si tiene el mínimo cardinal de todos los RECUBRIMIENTOS de M+. Esto ya quedó expuesto con el modelo "set-covering" siendo entonces el único criterio de simplicidad seguido. Pero tampoco da una razón para ello.

* No-redundancia:Un RECUBRIMIENTO D_I de M+ es una hipótesis si no tiene subconjuntos que contengan a M+.

* Relevancia:Un RECUBRIMIENTO D_I de M+ es una hipótesis si solo contiene desórdenes que estén en Causas(M+).

Asimismo Reggia, comenta las características de estos cuatro criterios. Relativo al "desorden único" dice que es muy restringido en cuanto a su aplicación, pues cuando existen fallos múltiples y simultáneos, no cubre las necesidades del problema.

Más adelante, se dirá algo más sobre este criterio en comparación con otros, aunque sí se puede decir ahora que es un criterio inconsistente con la estructura de resolución del problema diagnóstico. Es sin duda, cómodo, pero no una clara justificación como para que constituya un criterio general.

Relativo al criterio de minimalidad, en cuanto al RECUBRIMIENTO de mínimo cardinal, como idea, se le encontrará más adelante justificación, pero precisamente en este modelo, es quizás donde su aplicación presente más problemas en cuanto a su plausibilidad. Reggia corrobora esta idea, pero con otros motivos, pues ya introduce la resolución de la incertidumbre presentada al comparar RECUBRIMIENTOS de distinto cardinal y distintos d_i.

Peng [PENG-86-A] señala además dificultades computacionales para la aplicación de este criterio, que no concreta y que no se ven con claridad.

El criterio de relevancia pierde el de parsimonia/plausibilidad y, finalmente, Reggia señala que la linea más adecuada a seguir es el criterio de no-redundancia.

Este es uno de los sistemas de resolución del problema-diagnóstico mejor construidos. De todas formas, aunque sus autores lo catalogan como de caracter general, sigue teniendo un claro origen en el diagnóstico médico. Este, es reconocible por la existencia de un solo nivel en el razonamiento entre causas (D) y efectos (M).

En el area industrial ya se encuentran enseguida y en variados casos, estructuras de varios niveles como base del razonamiento diagnóstico [HAJE-87] en los que se utilizan procedimientos de estructuración en árboles de eventos.

Sin embargo, y aunque se volverá después sobre ello, uno de los métodos,- hoy por hoy el más completo-,más utilizado en el area industrial es el árbol de fallos [HAUP-86]. El árbol de fallos representa el conjunto de conexiones lógicas entre los componentes de un sistema técnico y el suceso no deseado dentro del mismo. Estas conexiones son de naturaleza determinística y, conducen a resultados probabilísticos solo cuando se utilizan probabilidades para la descripción del comportamiento de los componentes. De dicho arbol de fallos pueden llegar a obtenerse todas las causas posibles del fallo del sistema y que vienen representadas por los "conjuntos minimos". Es decir, detectado un fallo evidente ó principal, que puede asumirse a un cuadro de manifestaciones, se puede llegar a saber mediante el estudio del arbol de fallos, cuales son todas las causas posibles que pueden originar este fallo. El deducir cual de ellas es la causante, es en el caso de un sistema complejo, una labor larga y en muchos casos extremadamente laboriosa. Si a esto se añade una extremada necesidad de eficiencia, se encuentra con la raiz del problema que esta tesis pretende atacar.

Baste decir en este apartado que la idea de "conjunto mínimo", y de "conjunto de separación" deducible del árbol de fallos, recoge en un solo concepto , la idea subyacente que existe en los cuatro criterios descritos y señalados por Reggia.

Ampliando esta idea hay que definir previamente qué es un Conjunto Mínimo (en lo sucesivo C.M.).

Un C.M. es el conjunto de estados de componentes elementales del sistema, necesarios y suficientes, para que se produzca el "Suceso no Deseado" (en lo sucesivo SND). Hay que hacer la precisión de que el conjunto de C.M. que llevan a un SND, en un sistema complejo, puede tener varios millones de C.M. [ZIPF-82].

Asímismo, precisar que ese conjunto de condiciones necesarias y suficientes está formado por sucesos elementales,que en un momento dado se dan de forma simultánea aunque hayan tenido lugar de forma individual a lo largo del tiempo.

A estos sucesos elementales que no necesariamente indican situación de fallo, sino más bien indican estado actual de un componente, utilizando la lógica binaria de fallo ó no-fallo, se les denomina "Sucesos Básicos"(en lo sucesivo S.B.). En ese sentido, si un C.M. de un determinado sistema es $(x_1,\bar{x}_2)$, los sucesos básicos son x_1 y $\bar{x}_2$ y pueden indicar que x_1 falla y x_2 no fallo, pero que esta situación origina el fallo del sistema. De esta forma un C.M. representa un "modo de fallo" del sistema.

Si la realidad del fallo es que se produce el conjunto $(x_1,\bar{x}_2,x_3)$ a éste se le llama "Conjunto de Separación" (en lo sucesivo C.S.) y en la resolución del problema diagnóstico no se tiene en cuenta pues contiene al C.M. $(x_1,\bar{x}_2)$ y éste es suficiente para que exista fallo, de forma que el hecho de que además falle x_3, no modifica el estado del sistema. En este sentido, y dado que el estudio del arbol de fallos, en principio, da los C.S. en base a las conexiones lógicas entre los S.B., la primera labor que se realiza es la extracción de los C.M. desde el conjunto de C.S.

Los C.M. son entonces aquellos "modos de fallo" que estén contenidos en un C.S. De esta forma se procede a eliminar todos los C.S. que contengan a otro C.S., quedando al final tan solo los C.S. de los que se puede afirmar su incontención en otro C.S.,aunque pueden tener intersección distinta del vacío. Por ejemplo si los C.S. de un sistema son:

$(C.S.)_1 = (x_1, \bar{x}_2, x_3)$

$(C.S.)_2 = (x_1, \bar{x}_2)$

$(C.S.)_3 = (x_1, x_4)$

queda claro que el $(C.S.)_1$ contiene a $(C.S.)_2$, por lo que se elimina $(C.S.)_1$, y nos quedan como C.M.:

$(C.M.)_1 = (x_1, \bar{x}_2)$

$(C.M.)_2 = (x_1, x_4)$

cuya intersección es no vacia pero que constituyen dos modos de fallo diferentes. Es decir, el sistema falla porque lo hace x_1 y no x_2 ó bien porque lo hacen x_1 y x_4.

La resolución de esta incertidumbre entre uno de los dos modos de fallo ó incluso el hecho de que tengan lugar los dos simultáneamente, es el problema objeto de esta tésis.

La forma de encontrar los C.M. quedará descrita en su estado actual en el apartado 2.1.1.4.

2.1.3. -Establecimiento de relaciones entre causas y síntomas.

Parece claro que entre causas y síntomas unas están relacionadas y otras no. Establecer las relaciones cuando existan es tarea del experto y la forma de acotar estas relaciones necesita de una determinada metodología.

Es momento de destacar dos métodos ya nombrados. El "Set-Covering" de Reggia [REGG-84], [REGG-86] y los Árboles de fallos, de masiva utilización en ciertas áreas [VESE-81] y cuyas diferencias más evidentes son precisamente en el área de aplicación. ("Set-Covering en medicina"), (Arboles de fallos en industria); en el grado de implantación, (los árboles de fallos son frecuentisimos en sistemas complejos), y en que los árboles de fallos no sólo contienen relaciones causa-efecto, sino jerarquización y estructura.

Como ya se ha visto en el apartado anterior, una (quizás de las más significativas) de las diferencias más importantes en la forma de atacar el diagnóstico médico ó el industrial, era que en este último y en problemas complejos, se establecía una jerarquía de las hipótesis de fallo y, por tanto, se dotaba al problema de una estructura de las relaciones causales entre los fallos y sus causas.

La vaguedad aparente de muchos fenómenos físicos y la forma de razonamiento cualitativo basado en jerarquizar los conceptos, es un problema del máximo interés y estudio en la actualidad. Sistemas como el FOG expuesto por Raiman [RAIM-86] son ejemplos de sistemas orientados a representar y estructurar ese tipo de vaguedad aparente y conocimiento intuitivo que puede ser utilizado, en el razonamiento cualitativo. Pero relativo a la diferencia antes aludida, es Chandrasekaran quien nos habla del conocimiento representado en la estructura misma [CHAN-84-A].

El concepto de estructura a que alude, tiene que ver con la idea de "profundidad" y "superficie" en los sistemas de la que ya habla Hart (1982) ó Michie (1982) en la forma de "camino alto" ó "camino bajo".

En ese sentido, los sistemas de superficie son los más adecuados, para pares de datos patrón-decisión, con estructuras de control simples. Un ejemplo de estos sistemas sería el MYCIN [SHOR-76].

La diferencia con los "sistemas profundos", es que estos pueden resolver problemas de mucha más complejidad que los de superficie.

Resumiendo estas ideas, Chandrasekaran dice que, "entre el sistema de patrones y reglas y el conocimiento profundo, existe un conocimiento base de la estructura de resolución del problema, que se encuentra compilada en él". Usando esta estructura se gana en eficiencia. [CHAN-84-A].

Consecuencia lógica de esto, es que un sistema complejo es dificilmente estudiable si no se hace a partir de su estructura. De no hacerlo así, en el mejor de los casos, la eficiencia pasa a reducirse de tal forma que hace inutil la búsqueda.

Insistiendo en la idea de estructuración del problema, Davis [CHAN-84-B] describe estos conceptos en su propuesta de estudio de perturbaciones en sistemas electrónicos, enfocando la tarea como un proceso de razonamiento, desde un procedimiento hacia la estructura, ó dicho de otra forma, desde los comportamientos erróneos hacia la estructura de fallo. Apunta asímismo el hecho de que no hay una buena teoría de diagnosis para ello, y sitúa como uno de los objetivos iniciales a cubrir, el de obtener un lenguaje para describir la estructura, entendiendo por ésta, la información sobre la interconexión de módulos. Como objetivo, el hecho de encontrar un lenguaje, no parece el más importante, si Davis se está refiriendo a un lenguaje genérico y válido para el diagnóstico general, toda vez que, como ya hemos visto, distintos problemas de diagnóstico tienen distintas representaciones estructurales, ó mejor dicho, unos tienen estructura y otros no.

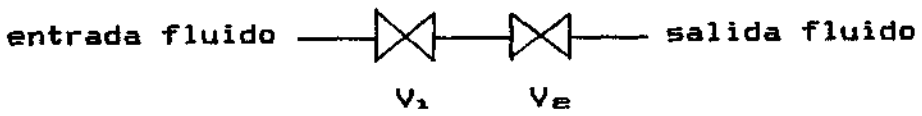
Sin embargo parece que como objetivo en un tipo de problema es una cuestión, más que conveniente, de una imperante necesidad.

De alguna manera, Reggia [REGG-84] está induciendo la idea de estructura, cuando en su modelo atribuye a cada "desorden" activo, dos números de clasificación en orden a situación y nivel. Sin embargo, en la ampliación de su modelo [REGG-86] ya descrito, establece relaciones causales pero sin una estructura de niveles. Y se entiende de esto, que siente la necesidad de estructurar por niveles, pero pensando siempre en el tipo de diagnóstico médico.

Llegados a este punto, es posible creer que el árbol de fallos recoge todas las ideas anteriormente descritas. A primera vista un árbol de fallos puede parecer un grafo Y/O dado que sus nodos son puertas tipo AND y OR. Sin embargo, la diferencia más clara está en el objetivo de su utilización. Los grafos Y/O, son tipos de representaciones que permiten reducir los costes de una búsqueda [HOME-86] en base a determinadas técnicas ó algoritmos de búsqueda [PAZO-80], [NILS-81].

El árbol de fallos que básicamente ya ha sido descrito anteriormente representa la estructura de conexiones lógicas de cara al fallo, de los componentes de un sistema.

Un ejemplo de un sistema simple sería: "Un sistema de dos válvulas colocadas en serie"



Se define el suceso no deseado (SND) como interrupción de flujo a la salida. El arbol de fallos sería:

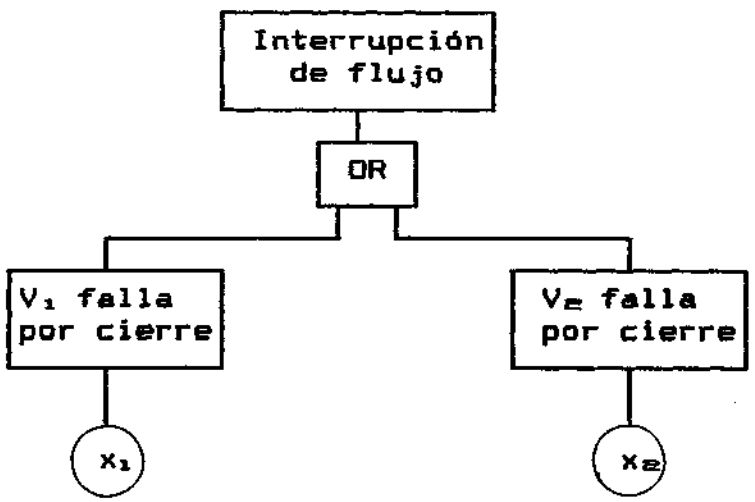


Fig.2.1

donde: SND ≡ Interrupción de flujo.

x_1≡ SB de etiqueta " V_1 falla por cierre".

x_2≡ SB de etiqueta " V_2 falla por cierre".

Normalmente, para la descripción de los estados de los componentes, se utilizan variables booleanas de forma que:

$$a_n = \begin{cases} 1 & \text{si el componente funciona.} \\ 0 & \text{si el componente falla.} \end{cases}$$

$(n = 1,....,N)$

Siendo N el número total de componentes del sistema. El estado del sistema se indica por medio de otra función booleana:

$$\Phi(a_1,.....,a_n) = \begin{cases} 1 & \text{si el sistema funciona.} \\ 0 & \text{si el sistema falla.} \end{cases}$$

Esta "función de estructura" representa la estructura a la que antes hacíamos mención de forma que su valor depende de los SB y, sobretodo, del SND.

Se dice con esto, que un mismo sistema físico, puede tener con los mismos componentes, distintos SND. según la función que se estudie, siendo las diferentes estructuras de fallo representadas por distintos árboles de fallos y, en consecuencia, distintas funciones de estructura [HAUP-86].

Recientemente [CALD-79], [CALD-80] se ha propuesto una extensión del algebra de Boole que permite tratar componentes y sistemas con más de dos estados (p.ej.: válvulas medio abiertas). El problema en la aplicación de estos sistemas, estriba en la dificultad de conseguir probabilidades para estados intermedios de componentes.

El estado de componentes y función de estructura utilizan una representación en lógica positiva, sin embargo, se utiliza más la lógica negativa:

$$x_n = 1 - a_n = \begin{cases} 1 \text{ si el componente falla.} \\ 0 \text{ si el componente funciona.} \end{cases}$$

Y la función de estructura:

$\Theta(x_1,\ldots,x_n) = 1 - \Phi(a_1,\ldots,a_n)$ que es

igual a | 1 si el sistema falla.
| 0 si el sistema funciona.

y es la que se adoptará en el estudio de nuestro problema.

Una observación más es el hecho de que en general la función de estructura tiene la propiedad de la monotonia, resultante del hecho de que un sistema en estado de fallo,no suele empezar a funcionar cuando falla otro componente más. Sin embargo, se comienza a prestar atención a funciones de estructura no monótonas, originadas, cuando en un árbol coexisten SB y sus negaciones. Una forma de atacar este problema es descomponiendo la función de estructura no monótona en subfunciones monótonas independientes [CHU-80].

Aunque volveremos a hablar de árboles de fallos y los procedimientos a ellos asociados, en el siguiente apartado, de lo ya comentado podemos extraer las siguientes cuestiones:

Una primera sería el hecho tan comentado antes del conocimiento "contenido" en la estructura del problema.

Y efectivamente así es, puesto que además de las reglas, que serían la representación del conocimiento de "superficie" que en el ejemplo de la Fig. 2.1 serían:

SI "V_1 falla por cierre" ENTONCES "interrupción de flujo".

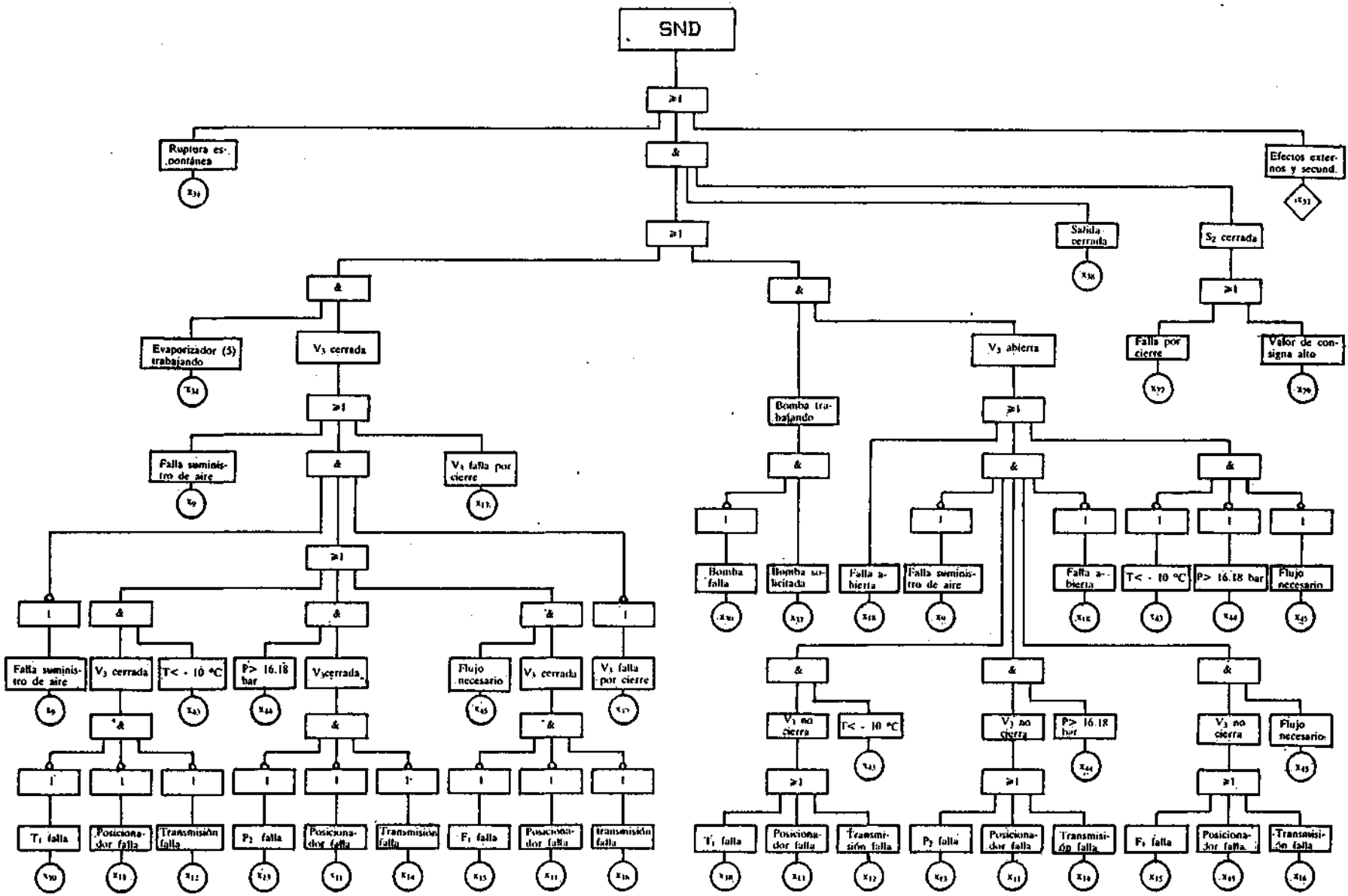
SI "V_2 falla por cierre" ENTONCES "interrupción de flujo".

El árbol sin embargo muestra la estructura de la concurrencia de fallos de componentes a fallo del sistema. Esa estructura sería el conocimiento "profundo".

Una segunda cuestión a hacer notar, es que precisamente esta estructura dota de consistencia a una serie de reglas (ó conocimiento) de cara a un SND. Y sería una buena herramienta para el ingeniero de conocimiento el dotarle de la capacidad de extraer de un gran volumen de reglas, aquellas que tienen entre sí una "consistencia" ó estructura y que le permitiese construir esa estructura.

Esta cuestión será atacada en la parte de resolución del problema objeto de esta tesis.

En un ejemplo como el anterior la estructura es enormemente simple. Pero observando el árbol de la página siguiente que corresponde a la ruptura de un subsistema real, que seguiría siendo el de un problema no complejo. En este árbol se observa ya un hecho también comentado antes, es decir, la existencia de niveles ó jerarquización de fallos. De nuevo lo único que añadido es que los niveles ó jerarquías están, de alguna forma implícitos en el conocimiento superficial y si el sistema es complejo, resulta difícil extraer la estructura sin una metodología. La utilización de la estructura del árbol de fallos sin incorporar metodologías de I.A. queda descrita por Roman [ROMA-87] en estudios para el EPRI.


SND
≥1
Ruptura espontánea
x36
&
Efectos externos y secund.
x33
≥1
Salida cerrada
x38
S2 cerrada
≥1
Falla por cierre
x37
Valor de consigna alto
x39
&
Evaporizador (5) trabajando
x34
V3 cerrada
≥1
Falla suministro de aire
x9
&
V3 falla por cierre
x17
≥1
1
Falla suministro de aire
x9
&
V3 cerrada
T< - 10 °C
x43
1
&
T1 falla
x30
Posicionador falla
x31
Transmisión falla
x32
&
P> 16.18 bar
x44
V3cerrada
&
P2 falla
x29
Posicionador falla
x31
Transmisión falla
x34
&
Flujo necesario
x45
V3 cerrada
&
F1 falla
x35
Posicionador falla
x31
transmisión falla
x36
1
V3 falla por cierre
x17
&
Bomba trabajando
&
1
Bomba falla
x30
Bomba solicitada
x37
V3 abierta
≥1
1
Falla abierta
x18
&
V3 no cierra
T< - 10 °C
x43
≥1
T1 falla
x30
Posicionador falla
x31
Transmisión falla
x32
1
Falla suministro de aire
x9
&
V3 no cierra
P> 16.18 bar
x44
≥1
P2 falla
x29
Posicionador falla
x31
Transmisión falla
x34
1
Falla abierta
x18
&
V3 no cierra
Flujo necesario
x45
≥1
F1 falla
x35
Posicionador falla
x31
Transmisión falla
x36
1
T< - 10 °C
x43
1
P> 16.18 bar
x44
1
Flujo necesario
x45

Asimismo queda ya descrita la importancia de la utilización de este tipo de estructura en sistemas de ayuda al operador en los términos ya descritos, de este problema, en referencias de aplicaciones de la ANS (American Nuclear Society) [SPUR-85], [GUAR-85], sobretodo en las áreas de respuesta a accidentes, que llevan implicito un diagnóstico previo. La estructura del árbol es conocida, fácilmente computable y evita los problemas derivados de los bucles en grafos cerrados [GRAH-82].

Como resumen de este epigrafe, puede decirse que el establecimiento de relaciones entre causas y síntomas, ó bien, entre fallos de "alto nivel" y de "bajo nivel" en un sistema complejo, necesita, por cuestiones de eficiencia preferentemente, el moverse en metodologías que puedan tratar estructuras que representan el tipo de conocimiento "profundo". Y que en el problema del diagnóstico al menos, una de esas estructuras es el árbol de fallos, al que hasta el momento, no se han aplicado metodologías de I.A., salvo en casos muy aislados.

Ya en la actualidad, aparte de esos casos, se desarrolla el sistema TEST por medio del tratamiento de este tipo de estructura por parte de Carbonell [CARB-87] y sobretodo en aplicaciones de sistemas complejos, que necesitan por sus propias características, algo más que una base de conocimiento basada en reglas.

2.1.4. -Búsqueda de conjuntos causa-efecto

Es decir, y a falta de mejor descripción posterior, búsqueda de "conflictos" según Kleer; de "set-covering" según Reggia ó de "conjuntos mínimos", según la metodología de los árboles de fallos. Conceptualmente son casi lo mismo, pero diferentes según lo comentado en el apartado anterior.

Se profundizará más en este tema y en estos dos métodos, más que significativos en el momento actual, de diferentes áreas de aplicación.

Causa-efecto en un problema de diagnóstico, donde el efecto es un síntoma (medicina) ó un fallo de nivel más alto (industria), es buscar el conjunto de fenómenos,que actuando solos ó junto a otros originan un suceso no deseado. Ya centrados en el modelo "set-covering" de Reggia [REGG-84], [REGG-86], y el de "conjuntos mínimos" ó " conjuntos de separación", extraibles de un árbol de fallos [HAUP-86], se comenta a continuación hasta qué grado representan conceptos parecidos.

El algoritmo mediante el cual, Reggia extrae los CONJUNTOS DE RECUBRIMIENTO parte del concepto de generador. Por éste entiende una colección de conjuntos de desarreglos ó fallos concurrentes, que representan explícitamente un conjunto de hipótesis de fallo y que pueden ser utilizados para generarlos. En el algoritmo presentado en el sistema D descrito por Reggia, el concepto FOCUS representa el conjunto de generadores ó hipótesis de fallos concurrentes.

La idea de generador antes expuesta es muy similar a la idea de conjunto de "Conjuntos de separación".

Sin embargo se observa que el algoritmo para formarlos lleva implícita una interesante idea, pero de nuevo encontramos que todos los generadores se encuentran al mismo nivel, lo que para un problema complejo, es poco adaptable.

Por otro lado, hay una parte del algoritmo que, al menos en principio, resulta arbitraria, y es el hecho de que en un determinado momento de la aplicación del algoritmo, debe reestructurar el conjunto FOCUS y, por medio de la llamada a un procedimiento, se crea un nuevo conjunto de generadores, basado en que la cardinalidad de cualquier hipótesis de fallo debe ser exactamente un grado mayor que la antigüa.

Así pues pasa en un momento dado, de haber una hipótesis de fallo, a haber dos, de aquí a tres y así sucesivamente. Nos preguntamos por qué precisamente un grado mayor y no más, ó dejarlo en un solo grado. Reggia no justifica esta decisión y tampoco le encontramos una clara explicación.

Buscando una explicación a esto, se ha intentado resolver esa aparente arbitrariedad con criterios razonados que difieren del seguido por Reggia.

Así se llega a obtener un FOCUS de un solo grado, más restringido que el propuesto por él, pero contenido en el de él y, por tanto, más idóneo para encontrar antes una hipótesis de fallo única. No es objeto describir aquí esta cuestión con más detalle, pues insistimos en el hecho de que el sistema propuesto, al carecer de niveles y, por tanto, de estructura al menos parcialmente, no se adapta por lo ya comentado al problema complejo, objeto de esta tesis. Sin embargo, para el tipo de problema planteado por Reggia, es un método como ya se ha comentado muy bien estructurado.

En el análisis del árbol de fallos no se presenta este problema. De nuevo es la estructura la que nos facilita la obtención del conjunto de "conjuntos de separación" que como ya se ha explicado constituyen la totalidad de los modos de fallo del sistema resultado, de aplicar la estructura sobre el conjunto S.B.

Posteriormente, de ahí se extraen los C.M.

Como ya se vió, había cuatro criterios expuestos por Reggia y ya comentados de cara a encontrar de forma plausible un conjunto de hipótesis de fallo.

El criterio del desorden único estaría representado por los conjuntos mínimos de un solo S.B. No parece un criterio aceptable puesto que puede tener una "fuerza" ó frecuencia de presentación muy baja y es tan consistente como decir que se escoge por ejemplo, el conjunto D_I de máxima cardinalidad. Por prácticamente la misma razón, se rechaza la idea de mínima cardinalidad ó minimalidad.

El criterio de relevancia es obvio, y en un problema concreto exigir simplemente que en una hipótesis de fallo no haya "desórdenes" ó "sucesos básicos", que no originen ese fallo, no aporta al problema ninguna clasificación.

De nuevo se encuentra que en un árbol de fallos, exigir el criterio de relevancia es inútil, puesto que la propia estructura, al generar los C.S. según los criterios de Reggia todos son relevantes.

Finalmente, queda el criterio de no-redundancia, que además Reggia reconoce como el más plausible, y de mayores posibilidades. Una hipótesis de fallo es no-redundante cuando es un RECUBRIMIENTO de M+, pero no tiene subconjuntos que cubran a M+. Este es un criterio de alguna forma similar a la idea de "Conjunto de separación". Aceptando la idea de RECUBRIMIENTO, si éste es no-redundante constituye un conjunto de condiciones suficientes para ser hipótesis de fallo, puesto que en el modelo "Conjunto de Recubrimientos",("Set-Covering") los $d_i \in D_I$ "pueden ser causas de los $m_j \in M_J$ pero no necesariamente lo son". La diferencia con un "conjunto de separación", es que éste contiene una serie de estados de componentes que constituyen en sí un modo de fallo.

Es decir, el conjunto de S.B. que pertenece a un C.S. constituye por la actuación conjunta de todos ellos, una hipótesis de fallo, mientras en un conjunto no-redundante puede haber algún d_i que no lo sea, y, sin embargo, estar en el RECUBRIMIENTO. Esto en el C.S. no ocurre.

Sin embargo, la gran similitud de estos conceptos estriba en el hecho de que los elementos de un C.S. constituyen igualmente un conjunto de condiciones suficientes que determinan un modo de fallo.

Sin embargo, en la idea de "Conjunto Mínimo" ya explicada, un C.M. determina un modo de fallo en sí, pero además constituye en sí un conjunto de condiciones necesarias y suficientes, para que constituyan una hipótesis de fallo. Esta exigencia "completa" de alguna forma la idea de no-redundancia, pero por su propia definición contiene en sí misma la de minimalidad y la de relevancia.

Por este motivo y en el estudio que se plantea en este trabajo, se establece que para estudiar el problema diagnóstico en sistemas complejos, es necesaria una estructura. De cara a la diagnosis industrial (al menos), se acepta como idónea la del árbol de fallos, y, finalmente, como conjuntos causa-efecto, aceptamos la de "conjunto mínimo".

Establecidas estas premisas en cuanto al estudio del problema diagnóstico, se puede pasar a comentar como se resuelve la incertidumbre implícita en todo problema diagnóstico derivado de la existencia de varias (en sistemas complejos, muchísimas) hipótesis de fallo.

2.1.5. -Resolución de incertidumbre.

Porque las metodologías del problema del diagnóstico, consiguen no poco con la acotación de los conjuntos, -ó al menos su descripción-, que constituyen hipótesis de fallo. Sin embargo, al final de este proceso, hay que tomar una decisión entre varias posibilidades. Aún en el caso de los árboles de fallos cuya cabecera ó raiz es un único "suceso no deseado", ó "fallo de mayor rango", pueden en sistemas complejos presentarse incluso centenares de miles de "conjuntos mínimos"[ZIPF-82].

Se hablará sobre los diferentes métodos utilizados en I.A. para la resolución de este problema. Sin embargo cabe decir aquí, que salvo un intento de clasificación de los mismos por parte de Heckerman [HECK-86], se echa de menos una guía ó metodología que permita una clasificación de los mismos y adecuabilidad a los diferentes tipos de problemas a tratar.

Creemos que es buen momento para intentarlo y es una línea abierta a la investigación, que permita clasificar el problema de la resolución de la incertidumbre.

La incertidumbre se encuentra ligada a todo problema de diagnóstico, y básicamente no es por otra cosa más que porque nunca hay un caso exáctamente igual a otro. El razonamiento-diagnóstico, puede llegar a ser ordenado, estructurado y similar, pero siempre para llegar a una serie de posibilidades de entre las cuales alguien decide que es una en concreto.

Sólo sistemas muy simples permiten hacer una inferencia directa de la causa del fallo. Sin embargo, esas inferencias simples están obedeciendo a una ordenación secuencial ó jerarquización de hipótesis de fallo. El razonamiento diagnóstico es de tipo secuencial, lo que implica en sí una jerarquización. El problema surge cuando a la vez que la secuencialidad se dan paralelismos en cada nivel de la jerarquía. Esta es la representación en un árbol de fallos.

La combinación de paralelismo y simultaneidad es la que empieza a definir la complejidad de un problema y de alguna forma, empieza a crear incertidumbre. Aceptado esto, se llega a un punto del problema. Ya se tienen una serie de hipótesis de fallo. Hay que elegir una. ¿Cuál?

En sistemas de los ya calificados como de "superficie", se utiliza el método de patrones, con la limitación de servir tan solo para problemas simples.

Aún así, los métodos más utilizados se basan en estimadores. Métodos que se van a comentar a continuación:

* Probabilidades simbólicas.

Son utilizadas por Reggia [REGG-84] en su modelo "Set-Covering". Indican la estimación subjetiva, no numérica de las observaciones. Naturalmente la dificultad estriba en transmitirlas, puesto que se acaban cuantificando conceptos.

Conceptos como:

- " x " frecuentemente origina " y ".
- " x " puede originar " y ".
- " x " nunca se presenta asociado a " y ".
- " x " normalmente se presenta asociado a " y ".
- " x " es raro que se presente asociado a " y ".
- " x " solo ocurre si " y ".

no dejan de describir una frecuencia perfectamente representable de forma numérica. El decir como dice Reggia, que las probabilidades exactas en diagnóstico, no tienen aplicación; Esto parece contradictorio y arriesgado, pues entonces ¿Qué sentido tienen los cinco estimadores utilizados por él en su modelo y que se exponen a continuación?

A = Siempre (al que se atribuye el nº 4)

B = Alta verosimilitud (" " " " " nº 3)

M = Media " (" " " " " nº 2)

L = Baja " (" " " " " nº 1)

N = Nunca " (" " " " " nº 0)

números que se utilizan para evaluar, finalmente, cúal de los generadores tiene más peso, y a los que llama probabilidades simbólicas.

Esto es una costumbre muy generalizada, pero no se ve diferencia con las probabilidades sin adjetivo más que en una cuestión de escala.

Sin embargo, Reggia corrige su método, aceptando de alguna forma la aproximación alternativa de plausibilidad, por medio del cálculo de probabilidad objetiva, utilizando la teoría de la probabilidad [REGG-86]. La base utilizada es asociar a cada $d_i \in D_I$ una probabilidad p_i, y a cada asociación causal (d_i, m_j) un "peso" ó "fuerza causal" $c_{i,j}$ que representa la frecuencia con la que d_i causa m_j.

La escala es: $0 < p_i < 1$

$0 < c_{i,j} < 1$

La suposición adicional de que los desórdenes son independientes entre sí, las fuerzas causales constantes, y el hecho de que ninguna manifestación ocurre sin causa, completan el cuadro en el que utilizando el teorema de Bayes se llega a la hipótesis más plausible.

* Método de Dempster-Shafer.

Se admite que la regla de Dempster-Shafer (en lo sucesivo D-S), es una generalización de la teoría de probabilidades.[DEMP-68], [SHAF-76], [SHAF-86]. En este sentido, y combinando la teoría de la evidencia [GORD-85] y la de probabilidad bayesiana [PEAR-86], se ha hecho una extensión de la teoría de D-S,[YENG-86], de cara al tratamiento de un espacio jerárquico de hipótesis asimilando los FC (factores de certeza) de MYCIN, con el "bpa" (probabilidad básica asignada) de D-S sin una justificación formal [GORD-85].

La extensión propuesta por Yeng, acaba asimilando el "mapping" multivaluado, hacia un multiconjunto probabilístico donde finalmente desarrolla un modelo de razonamiento a partir de D-S, consistente con el teorema de Bayes, con asunción de independencia condicional. De alguna forma, D-S, con las extensiones posteriores, es asimilable a sistemas bayesianos, siendo en todo caso un buen sistema para expresar la ignorancia.

Al nivel de aplicación y como extensión de la teoría de la probabilidad Bayesiana de cara al problema diagnóstico, se encuentra el sistema INSPECTR que necesita de la metodología de D-S para expresar el conocimiento ó desconocimiento de la ocurrencia de un evento cuya expresión formal es una regla. En cualquier caso, sigue asumiendo la jerarquización de hipótesis mutuamente excluyentes [KESS-87].

Un SE de diagnóstico industrial, bastante completo y que utiliza un sistema similar conceptualmente al de D-S es TURBOMAC, donde ha de representarse de alguna forma la ignorancia sobre un problema. Este SE clasifica (jerarquiza) hasta 142 tipos de problemas manejando más de 9700 reglas. [PARG-87]

Sin extenderse demasiado se puede decir que D-S como extensión de la teoría de la probabilidad Bayesiana ó bien ambos sistemas integrados, quizás sea un sistema -aún hoy de poca implantación-, de los que mejor se adapten a los problemas de diagnóstico a los sistemas complejos.

Sigue habiendo, sin embargo, gran diversidad en el tratamiento de la incertidumbre, ya sea dependiendo del problema, ó simplemente del equipo que lo trate. Lo que sí parece comprobarse por las experiencias en este campo, es que es un buen sistema para el tratamiento de hipótesis jerarquizadas, es decir , de problemas dotados de estructura por niveles.

* FC.- Factores de certeza.

Todo estudioso de la I.A. asociará enseguida este concepto al sistema MYCIN de tratamiento de la incertidumbre. Sin embargo, aunque el término se sigue utilizando, muchas veces nada tiene que ver con aquel.

Esto es debido a que el "factor de certeza" ha pasado a ser un concepto que sirve para expresar de forma numérica dentro de un intervalo el grado de conocimiento ó desconocimiento sobre un evento.

No obstante, ese concepto "numérico", moviéndose en el intervalo [-1, 1] aún se encuentra, al igual que en MYCIN pero ligando valores de parámetros fisicos en los intervalos propios del parámetro con el intervalo [-1, 1] por medio de un isomorfismo [KITA-87].

Este caso se puede ver en el tratamiento de incertidumbre de un SE de aplicación industrial en la planta de Sunawaga en Japón para estudio de causas de emergencias y ayuda al operador. El sistema ha sido desarrollado en Prolog y está prevista su puesta en funcionamiento en septiembre de 1987.

En el sistema experto DISKET ya mencionado en el "meeting" de 1985 de la ANS [YOSH-85] como un clásico problema de diagnóstico de emergencias al igual que el anterior, se habla también del "factor de certeza". Sin embargo, hay ya algunas variaciones.

El concepto manejado es el siguiente: Los FC toman valores entre [-1, 0] ó [0, 1]. Si son positivos, expresan el incremento de certeza sobre una hipótesis en la parte "acción" de una regla, cuando la "condición" se cumple. Los valores negativos indican el incremento de la confianza en la negación de la hipótesis cuando la "condición" se cumple. Hasta aqui la semejanza con el concepto FC de MYCIN es muy alta.

El proceso de transmisión ó composición de la incertidumbre cuando n condiciones confluyen en la misma hipótesis, se basa en el CF_{sum} basado en el teorema de Bayes de esta forma:

$$CF_{sum} = 1 - (1 - CF_1)(1 - CF_2)...(1 - CF_n)$$

e incluso se hace notar que al final del proceso se suman todos cada uno con su signo [BERG-87].

Como se ve, los conceptos son similares a MYCIN pero luego y dependiendo de la aplicación, su uso se va desviando. Hay que hacer notar, sin embargo, que son los japoneses quienes hoy por hoy siguen en esa línea.

* Tratamiento probabilístico.

Uno de los SE más nombrados con tratamiento probabilistico es el PROSPECTOR. Sin embargo la teoría de la probabilidad y sus extensiones, forman un área de resolución de la incertidumbre en la industria de casi aplicación general. El motivo es que las normativas técnicas exigen en sus estudios un rigor en el tratamiento de la experiencia. El hecho de ser una teoría bien conocida, con una metodología más ordenada y consistente que otras, hace que la tendencia a su utilización vaya siendo paulatinamente mayor. Incluso se ha llegado a hacer una conjunción bastante bien estructurada entre el método de D-S y el probabilístico, hablándose de aquél como extensión de éste.

El SE TURBOMAC [PARG-87] utiliza el estudio frecuencial y estadístico de fenómenos para llegar a establecer probabilidades de sucesos. Luego construye los parámetros de incertidumbre según D-S.

El sistema PRISIMS [CAMP-85] hace un estudio sobre los riesgos de accidentes en plantas nucleares totalmente probabilístico.

Asimismo, el sistema CAPAM partiendo de probabilidades iniciales, acaba resolviendo la incertidumbre, añadiéndole un segundo nivel de acercamiento en base al estudio de probabilidades marginales [GUAR-85].

Una clara defensa de los sistemas Bayesianos soporta el sistema construido para estimar fallos de causa común que aún hoy por hoy sigue siendo una de las líneas abiertas de investigación de mayor interés en el área industrial al menos [KUJA-85].

Y como muestra del tratamiento probabilístico en tratamiento de árboles de sucesos, se encuentra el sistema SQUIMP [DIXO-85].

Ni qué decir tiene que los tratamientos actuales de árboles de fallos en sistemas simples, se realizan casi exclusivamente por aplicación de la teoría de la probabilidad [HAUP-86], [WARL-73].

Como colofón de este apartado, solo resta decir que aunque haya problemas en los que la aplicación estricta de la teoría de la probabilidad no ha sido suficiente, cuando ha sido dotada de extensiones coherentes, ha demostrado ser uno de los mejores métodos de tratamiento de la incertidumbre.

Quizás una de las extensiones de mayor aplicación en la industria pero que se comenta aparte es la teoría de la fiabilidad.

* Fiabilidad.

La teoría de la fiabilidad, como área de estudio, de tratamiento probabilístico, no se encuentra asociada en la actualidad a tratamientos sistematizados del problema del diagnóstico. El motivo no sería tanto de falta de adecuación, sino por lo relativamente reciente, del concepto de fiabilidad tal y como hoy se le entiende.

La fiabilidad nace como estudio de la sistematización de la seguridad de los sistemas industriales. Estos sistemas han sido casi de forma exclusiva los relacionados con la industria aeronaútica militar. Todo el movimiento previo de estudios y compendios de tasas de fallos, se realiza a finales de los años 50, donde las compras de equipo electrónico por parte de la Fuerzas Armadas USA están ya sujetas a especificaciones de fiabilidad. Una de las primeras fue la MIL-R-25717 en 1957. Posteriormente, aparecen el AFBM Exhibit 58-10 (1959) para misiles y la MIL-R-26674 (USAF) para aviones tripulados (1959). En ese mismo año, sale a la luz el primer compendio de tasas de fallo para la predicción de la fiabilidad en equipos electrónicos (como el RADC Reliability Notebook y el MIL-HBDK-217) en 1962.

Las primeras publicaciones datan de los años 60 [BAZO-61], y a final de esta década, la fiabilidad sale del ámbito espacial y militar a otras áreas.

La fiabilidad es un concepto aparecido principalmente por la complicación de los sistemas modernos [WARL-73]. En ese sentido cabe decir que el hacha de mano de silex del Paleolítico inferior poseía una fiabilidad 1. Es decir, éxito seguro.

Hoy día los sistemas son más perfectos y por tanto más complejos y el concepto de seguridad de funcionamiento en cada momento de los sistemas en estudio se realiza por medio de la fiabilidad.

Aunque sus àreas de aplicación son potencialmente todas, sigue habiendo una predominancia en las áreas aeronaúticas y centrales nucleares, es decir, que de alguna forma se le asocia con los sistemas complejos y dentro de estos con los árboles de fallos [WARL-73], [HAUP-86].

Una de las áreas donde la fiabilidad ha permitido estudios mucho mejor estructurados, ha sido el mantenimiento, que no es el objeto de ésta tesis, pero que, sin duda, es el paso siguiente a la resolución del problema del diagnóstico [PERE-86], [FAUS-86], [ILLA-86], [CUES-86].

En el área de los SSEE de diagnóstico, no aparece la fiabilidad como tal. Sin embargo, creemos que puede ser una incorporación importante de cara a evitar las "probabilidades subjetivas" que constituyen las "probabilidades a priori" de un determinado evento, dotándolas de una mayor rigor basado en la experiencia.

* Simulación.

La simulación ha sido y es utilizada para evaluaciones de tipo cualitativo cuando no se dispone de suficiente información cuantitativa en el estudio de fenómenos [RAIM-86]. Los sistemas de simulación como ayuda al diagnóstico, se han aplicado en la detección y evaluación de fallos, como elemento reproductor del funcionamiento del sistema y funcionando en paralelo con él, de forma que el sistema, al sufrir una desviación de sus parámetros en los rangos de diseño, queda detectada, por diferenciación con el parámetro esperado. En cualquier caso es aplicable al diagnóstico y detección de causas, con muchas limitaciones. Ahora bien, ya no los simuladores sino los juegos de simulación han servido tradicionalmente para predecir comportamientos [YAKO-77], [RIOS-67], [SPIE-69].

Sin embargo, en sistemas complejos y con tratamiento de árboles de fallos se utilizan métodos de simulación como el de Monte Carlo para estimar los C.M. (conjuntos mínimos) que aportan una mayor no-fiabilidad al sistema, [HAUP-86], lo cual es una forma de "extraer" del conjunto de C.M. los que aportan una mayor probabilidad de fallo y que, por tanto, constituyen hipótesis de fallo. Para un sistema complejo se estima como el mejor método, dada la casi imposibilidad computacional de forma eficiente de aplicar métodos analíticos. Naturalmente, el nivel de confianza depende del número de ensayos, que se eleva a varios millones, si se analiza un sistema muy fiable (con no-fiabilidad por debajo de 0,001) como centrales nucleares y satélites artificiales. Cifra de ensayos que resulta prohibitiva. Sin embargo, aún a riesgo de aumentar el error, se podría reducir el número de ensayos.

Este es pues un caso donde la simulación no pretende predecir el comportamiento del sistema sino obtener información sobre la estructura del sistema.

* Lógica difusa ó borrosa.

Este sistema basado en las reglas de Zadeh, ha sido utilizado en algunos sistemas para resolver el problema de incertidumbre [ZADE-79]. Con algunas variaciones sobre Zadeh, se permite en CIS que un sistema estructurado similar al MYCIN pero con un importante volumen de reglas (del orden de 100.000), llegue a resultados con una gran eficiencia [BLEL-86].

Sin embargo, y de forma más concreta, ya aparece el SE DICON de diagnosis y control de sistemas estructurados en niveles, y que utiliza la lógica difusa en la resolución de incertidumbre [CHRI-85].

En [ADYA-87] se encuentra una referencia muy reciente de la utilización de lógica difusa introducida en la función de predicción de valores iniciales, puesto que en este proyecto se la estima como un buen método para representar la intuición y el conocimiento cualitativo del operador. En la posterior simulación del sistema para encontrar su óptimo se ha demostrado, que la utilización de lógica difusa ha resultado más efectiva que otros sistemas.

Se piensa, sin embargo, que no se puede establecer una generalización pues la aplicación y el tipo de problema es muy particular.

* Lógica multivaluada.

Basada en el supuesto de que existen más estados de "verdadero" y "falso" ó "fallo" y "no-fallo", se realizan estudios en la actualidad de forma que puedan así representarse estados intermedios, no como coeficientes, ó valores dentro de un intervalo de los señalados, sino como parámetros nuevos e independientes.

No se ha llegado en la actualidad a resultados bien estructurados de este tipo, pero el problema diagnóstico, los ha atacado, tanto en el campo médico [HICK-85], como en el industrial. La dificultad de estos procedimientos tropieza, sin embargo, con la dificultad de conseguir probabilidades para estados intermedios de componentes, y los problemas de conocer la influencia de dichos estados sobre los parámetros del sistema analzado [HAUP-86], [CALD-79], [CALD-80].

* Teoría de la decisión.

Es precisamente Shortliffe quien en la actualidad propone métodos de la teoría de la decisión usando la base de reglas del MYCIN [SHOR-86].

Qué duda cabe que en principio, la teoría de la decisión sirve de forma genérica para resolver el problema de incertidumbre. Los métodos en los que se apoya, son en todo caso conocidos por mayor número de investigadores [SIMO-86]. Shortliffe en ese artículo se plantea su utilización de cara a establecer la bondad de una heurística para la mejor resolución de un problema, cuando se parta de planteamientos:

situación ——→ acción

Hace una salvedad al empleo de la teoría de la decisión en sistemas complejos por lo que supondría de costoso y poco eficiente computacionalmente, por lo que en principio parece que no serviría en este caso. Sin embargo, enseguida pasaremos a hablar de la complejidad de un sistema.

Los sistemas señalados y sus extensiones reflejan un ánimo de los investigadores de la I.A. en resolver el problema de la incertidumbre.

A cada nuevo tipo de problema se propone un nuevo método.

Resulta, sin embargo, significativo, un acercamiento general, hacia sistemas probabilísticos, ya sea, "puros" ó como extensiones del teorema de Bayes [SNOW-86] y extensiones de la regla ó método de D-S [SHAO-86].

Ante la gran diversificación de métodos se hace necesaria una guía para aplicar adecuadamente, tipos de resolución de incertidumbre suficientemente estructurados a cada tipo de problema.

Un estudio generalizado que puede ser un punto de partida es el mostrado por Heckerman [HECK-86] donde compara diferentes sistemas con la teoría de la probabilidad, en base a siete propiedades de las medidas de credibilidad y posteriormente deduciendo cuatro categorías de problemas no atacables por estrategias probabilísticas.

Quizás el próximo paso sea el clasificar los problemas de cara a elegir el método adecuado.

Esta tarea es probablemente compleja y constituye a nuestro juicio una linea abierta de investigación.

2.2.- SISTEMAS COMPLEJOS.

El hecho de la complejidad de un sistema ha estado presente especificamente en muchos problemas y en concreto en el problema diagnóstico. Si se hace mención explicita a él, no es por otro motivo, que por hacer que muchas de las heurísticas y metodologías empleadas, carecen de validez por su alto coste y baja eficiencia computacional.

Los problemas complejos, parece pues que necesitan metodologías que resuelvan el problema principal y además tengan en cuenta este fenómeno de la complejidad.

Ya Chandrasekaran habla de los procesos complejos cuando habla del "conocimiento profundo" ó "estructura", insistiendo en la necesidad,-al menos-, de simular estos procesos [CHAN-84-A]. Continuando en su tratamiento especifico para problemas complejos, describe como uno de ellos, el del diagnóstico en sistemas electrónicos [CHAN-84-B].

Asimismo Reggia y su modelo "set-covering" describe su método como adecuado para sistemas complejos [REGG-84].

De alguna manera, y aunque volveremos sobre él, Ratner define el sistema complejo como aquel que tiene una gran dificultad combinatoria, y propone el algoritmo "Joint" y el LPA* para resolverlos con eficiencia [RATN-86].

Asimismo en [SHOR-86] ya se vió como Shortliffe deducía que el empleo de la teoría de la decisión en sistemas complejos no era adecuado por su alto coste y baja eficiencia, y de forma similar Snow, [SNOW-86], nos asegura que los sistemas probabilísticos de resoluciób de incertidumbre son prohibitivos para sistemas grandes y complejos.

Para sistemas de representación de redes de grandes tamaños, Blelloch propone un método, el "Concurrent Inference System" (CIS) como sistema de reglas similar al Mycin pero con 100.000 reglas, con una gran eficiencia basada en una estructuración por niveles, que permite que una inferencia no "atraviese" más de 20 reglas ó niveles [BLEL-86].

De nuevo se ve cómo un problema complejo viene asociado en su tratamiento a una estructura jerarquizada. Asimismo, Hauptmans [HAUP-86] en su método de análisis del árbol de fallos indica, como el tratamiento de estructuras complejas, requiere métodos específicos, entre los que destacan:

- Simulación por Monte Carlo.
- Búsqueda analítica de los C.M. del árbol.
- Evaluación mediante reconocimiento de estructuras. [KOEN-74].

Sin embargo, se sigue hablando de sistemas complejos, asociándolos a grandes, sin a veces más referencias que el número de reglas que los componen.

En toda la bibliografía hasta ahora referenciada, sin embargo, nadie dice con claridad, qué es un sistema complejo. Parece claro, que un elevado número de reglas es característica de por sí para constituir un problema complejo. Pero puede haber problemas relativamente simples, con muchas reglas y otros de menor número de reglas y una estructura compleja. En este sentido cabe hacerse la pregunta ¿Cómo se define la complejidad? Gabriel [GABR-87] dá unas medidas que permiten conocer el grado de complejidad de un problema, como una medida matemática y compara el hecho de que una central nuclear tiene un grado aparente de complejidad como el de una computadora de tamaño medio y, sin embargo, la primera cuesta 10.000 veces más que la segunda. En base a esta diferencia de costes procede a un examen más detallado del problema.

En una primera aproximación, define el término "complejidad" como el montante de información necesaria para definir el producto ó proceso en términos de partes más elementales. A partir de aquí deduce que una medida razonable de la complejidad de una lista jerárquica de partes de un sistema, sería, el número de dígitos binarios necesarios en la especificación más detallada de la lista para ser capaces de distinguirla de otras listas.

La estimación de este concepto sería en base a una fórmula:

$$C = 0{,}5\ (N \cdot P) \cdot \log_e (N \cdot P)$$

donde N = Número de partes distintas en la planta.

P = Número de puertas en las que cada parte está presente.

En una primera aproximación estima C para una planta nuclear de tamaño medio en C ≈ 2.000.000 y en un VAX 11/780 en C = 512.000, que son en principio del mismo orden.

Sin embargo, en una planta nuclear existe un problema muy a tener en cuenta, de cara a la aplicación de los SS.EE. en estos problemas, y es el derivado de la eficacia en la comunicación de grandes grupos de expertos de diferentes disciplinas, que originan sus mucho más altos costes y asimismo las dificultades del ingeniero de conocimiento para extraer la información a estos.

Donald Michie [MICH-85] defiende, sin embargo, la entrada de la IA para la resolución de los problemas derivados de la complejiadd en los actuales sistemas, particularmente los nucleares, y propone una serie de especificaciones para el "software" que soporte la resolución de estos problemas.

Asimismo Kisner, [KISN-85], procede a una especificación mayor de cara a un proyecto que ataque el problema de forma integrada, flexible e inteligente. Ya en 1987, se encuentra una aplicación como interfaz de usuario inteligente para SSEE aplicados al estudio del mantenimiento y perturbaciones de turbinas de gas, como caso en que las tareas de los técnicos están caracterizadas por la interpretación de sintomatologia compleja [KOCH-86].

Como se vé, salvo la aportación de Gabriel, no hay una forma clara de distinguir un fenómeno complejo, fuera del ámbito industrial.

Sin duda, ésta constituye una de las líneas abiertas más claras de investigación, toda vez que el problema complejo no está tratado de forma general y quizás por ello y ante una carencia clasificatoria de los problemas en orden a su grado de complejidad, hoy por hoy, uno de los requerimientos expuestos por el Laboratorio de Inteligencia Artificial de la Universidad Politécnica de Madrid , para desarrollar un S.E. es que "la tarea no sea demasiado difícil" [PAZO-86]. Parece claro, sin embargo, en lo expuesto por Michie, que en el campo industrial y concretamente en el caso nuclear no solo se puede, sino que además, hoy por hoy, no existe otra alternativa.

2.3.- TIEMPO REAL

Esta expresión viene asociada a dispositivos computacionales en los que se presentan datos ó resultados de procesos en un tiempo suficiente como para que el destinatario de los mismos los pueda utilizar válidamente.

El objetivo de estos dispositivos es pues, la "máxima eficiencia". Los métodos para conseguirla son de dos tipos, e incluso mixtos. Uno es en base al propio "hardware" que ha de ser más y más potente, rápido y complejo conforme se atacan problemas más complejos.

Un ejemplo de SE en tiempo real atacado de esta forma, sería el mostrado por Koike [KOIK-86] que analiza los fallos de un sistema de producción de energía eléctrica de 12.000 MW, puesto definitivamente en servicio en 1987, por medio de un superminicomputador TOSBAC DS600/80 de 8 Mb de memoria principal, con una velocidad de proceso de 3,7 Mips.

Sin que esto pueda parecer demasiado grande se mostrarán otros, que utilizan dispositivos menos potentes , simplemente porque basan su rapidez, en una buena estructuración del problema y uso de heurísticas apropiadas. Se cree sin embargo, que sin abandonar el primer método, merece la pena volcarse de forma especial, en la solución "software", entre las que incluimos la IA.

El objetivo es sin duda la "eficiencia", pero ¿cuanta eficiencia? Esto es una forma de decir, que ciertas decisiones tienen un tiempo límite y paradójicamente esta necesidad se presenta en sistemas de gran complejidad. Ante esta situación, la necesidad, de forma genérica y, en el caso, de ayuda al operador se presenta de otra forma. Esta forma sería el diagnóstico de un sistema tan complejo como se quiera y "emitir diagnóstico aunque sea parcial, y por tanto, con asunción de error, pero dentro de los límites de tiempo preestablecido".

De alguna forma, este es el problema genérico que esta tesis pretende atacar.

Esta permanente alusión a la eficiencia ya viene reflejada por Chandrasekaran cuando habla de la "estructura" del conocimiento [CHAN-84-A]. Asímismo Ratner [RATN-86] propone, para resolver este tipo de problema dos algoritmos que combinan la idea de aproximación y búsqueda rápida, el JOINT y el LPA*, muy idóneos para problemas con límites de tiempo, basándose el LPA* (Local Path Algorithm) en una búsqueda local alrededor del objetivo, fijando un radio máximo.

De forma similar el algoritmo JOINT, divide el camino que parte de un nodo en segmentos de una longitud máxima, realizando luego la búsqueda local dentro de cada segmento, produciendo así una sucesión de subcaminos óptimos.

El sistema ATMS introducido por Kleer, es comentado por Morris [MORR-86] y ataca el problema de búsquedas a través de alternativas múltiples que modelizan acciones de cambios de estado en árboles estructurados.

Sin embargo ya anuncia que aquellos problemas que incluyen cambios temporales requieren algún mecanismo adicional.

El caso industrial es un representante clàsico de sistemas con cambios temporales (dinámicos) y no es por tanto atacable con herramientas tan potentes como las descritas.

Se han construido lenguajes específicos para estos problemas como el TCP [WILL-86], pero aún así, sigue habiendo grandes limitaciones.

Si a los cambios dinámicos se añade la extrema necesidad de eficiencia, parece un problema difícil de resolver, dado que extendida una hipótesis correspondiente al momento "t" puede no ser la idónea cuando se pretenda corroborar. En ese sentido, se han desarrollado trabajos para construir sistemas que detecten el fallo antes de que sea percibido ó antes de que un sistema tradicional de alarma lo detecte [BERG-87] basados en el diseño del sistema experto DISKET de diagnóstico ya planteado en 1985 [YOSH-85]. Este sistema utiliza las técnicas de simulación con simulador en paralelo con el sistema real de forma que detecta diferencias de comportamiento antes de que el fallo se produzca [WAKA-85].

Las técnicas de simulación son utilizadas en su vertiente estadistica para resolver de forma eficiente problemas de gran complejidad. [RIOS-67], [WARL-73], [HAUP-86]. De esta forma queda implícitamente definida una nueva medida del fenómeno complejo.

Es decir la resolución del problema diagnóstico y explicitación del fallo concreto que ha tenido lugar (ó de una hipótesis a verificar) en un tiempo predeterminado (que normalmente es el tiempo máximo en que el sistema no propaga los efectos del fallo, originando así fallos secundarios) un problema, sea cual sea, es complejo si no es localizable de forma exhaustiva dentro del tiempo predeterminado. Por tanto, hay que recurrir a métodos que inducen por sí mismos a un error.

Más que la expresión "tiempo real" pasamos a una necesidad, dentro de un "tiempo constreñido." El problema pues de la "constricción temporal" que no sólo se presenta en diagnóstico, es una característica del problema en estudio de esta tesis.

Una buena introducción al tema es hecha por Haddaway [HADD-86] basándose en la idea de una lógica, cuyo grado de certeza depende del tiempo disponible, lanzada por Michalsky y Winston y que denominan "Lógica de Precisión Variable" (VPL). El sistema de resolución de incertidumbre se basa en D-S [MICH-86].

Este problema y con esta dimensión integrada de toma de decisión en tiempo constreñido, tan solo encuentra eco en estos autores. Sin embargo, hay que hacer constar que la evaluación del error en un problema de diagnóstico complejo, está aún por definir en su parte cuantitativa. Es decir, en base a qué, cuando existen múltiples hipótesis, se puede valorar ese error. Por este motivo, en los sistemas industriales complejos, no se procede a una cuantificación del error sino a mostrar, unas recomendaciones jerarquizadas. Es el caso de la industria aeroespacial, donde se utiliza hoy por hoy de forma sistemática un simulador [SIEM-86].

Aún hoy, sin embargo, la línea principal de ataque al problema de la eficiencia es simplemente esa, es decir, lograr sistemas de la "máxima eficiencia" y es en el campo nuclear donde se buscan estos sistemas con mayor ímpetu.

Como resumen de lo expuesto hasta ahora podemos referenciar el sistema LMI, que reproduce de alguna forma la problemática y la forma de ser atacada en la actualidad, de los SSEE en entornos de tiempo real. [MOOR-85].

Este sistema ha sido diseñado para operar en una máquina Lisp y asume que pueden llegar señales de 20.000 puntos de alarma. Para el sistema LMI ha sido utilizada la máquina LAMBDA.

Finalmente, se tratará de referenciar adecuadamente un aspecto que podría tratarse de forma separada,pero que en este caso se encuentra íntimamente ligada a la condición de "máxima eficiencia".

Se trata del error humano.

Tanto en las centrales nucleares, como en los sistemas de seguimiento de satélites, así como, el manejo de una aeronave, están confiados a personas, y por normativa legal no se dejan a la suerte de sistemas automáticos. Se le presupone al operador mucha mayor capacidad, flexibilidad, y maniobrabilidad que al más pefeccionado sistema [MARI-86]. Por esta misma razón, la carga sicológica del operador ante una emergencia es un importante factor de "stress" que origina que algunas de sus decisiones sean fatalmente erróneas.

El factor principal que induce al error humano es, sin duda, la necesidad de dar una respuesta extremadamente rápida. Es decir, se le exige además de un buen criterio, una "máxima eficiencia".

En [OKAM-85] se describe un sistema basado en interfaz-usuario de presentación del estado del sistema como importante ayuda al operador y que reduce considerablemente la probabilidad de error humano según queda recogido en la norma NUREG/CR-1278,2254 [SWAI-83]. Como queda recogido en este artículo, una de las futuras líneas a atacar es la combinación de un sistema de presentación en pantalla, simple y fácil de interpretar, conteniendo sólo lo estrictamente necesario, con sistemas adecuados de diagnóstico. Este problema queda de alguna forma, atacado en esta tesis.

El proceso de decisión del operador, y que ha de realizar en un tiempo de "máxima eficiencia" podemos encontrarlo descrito con cierto detalle en [TANA-85] y básicamente consta de las siguientes fases:

* Detección de síntomas ó condiciones anómalas.
* Identificación del estado del sistema.
* Interpretación de la situación ó diagnosis.
* Determinación de las causas.
* Acciones a tomar.

y todo esto en un sistema complejísimo y en tiempos extraordinariamente cortos que en el mejor de los casos no debe llegar a media hora.

En la actualidad la tendencia es considerar que el error humano es inevitable, expresando la participación del operador como un elemento más del sistema. Todo sistema eficaz de ayuda al operador, reduce la probabilidad de fallo del mismo. De esta forma y aún asumiendo que el comportamiento del operador, no se puede estandarizar como el de un componente técnico, se tiende a tratar los actos humanos de la misma forma que los componentes [SWAI-83]. Entonces, los errores humanos se definen como acciones fuera de tolerancia [HAUP-86].

Uno de los cinco tipos de errores que da la clasificación es el "error de tiempo" definido como el hecho de que una persona no consigue realizar una tarea ó parte de ella dentro del intervalo previsto, es decir, la realiza demasiado tarde ó demasiado pronto.

Finalmente, comentar que se ha descrito el comportamiento del operador de una central nuclear, mediante una distribución exponencial igual que los componentes y que este tipo de estudio ha permitido introducir el "fallo del operador" en una acción determinada en el árbol de fallos del sistema, de cara a conocer la fiabilidad global del sistema, con este nuevo parámetro. Esta forma de estudiar el problema, hace que el árbol de fallos se vuelva mucho más complejo por lo que se sigue tendiendo a eximir al mismo de este parámetro y crear sistemas cada vez más eficaces de ayuda al operador.

CAPITULO TERCERO

3.1- DETERMINACION DEL PROBLEMA GLOBAL EN SU APLICACION INDUSTRIAL

Como ya se ha comentado en el capítulo anterior, el marco de trabajo es el calificado como de ayuda al operador en emergencias. En una situación así, el operador, ayudado por estos sistemas, debe en un sistema complejo, y en un tiempo extremadamente corto recibir la suficiente información para determinar la causa del fallo. En las referencias señaladas en la bibliografía, puede verse como problema genérico de más acuciante necesidad de resolución en la actualidad.

Como forma de representación y buena descripción del problema diagnóstico, se encuentran:

* Modelo "Conjunto de recubrimiento".
* Arbol de fallos.

Como forma de resolución de incertidumbre y por encontrarse implícito en la normativa existente en la actualidad en áreas ya aceptadas y no relacionadas en principio con la I.A., asi como su extendida aplicación en modelos de mantenimiento, destacamos la fiabilidad.

El método actual consiste básicamente en encontrar el C.M. que más aporta a la no-fiabilidad del sistema en ese momento.

En la bibliografía reseñada, tan sólo en un artículo, se mencionan los Conjuntos Mínimos extraibles del árbol de fallos para sistemas complejos [RODR-86]. En este artículo, se presenta una metodología para construir bases de conocimientos de reglas a partir de un árbol de fallos, en base al método simple:

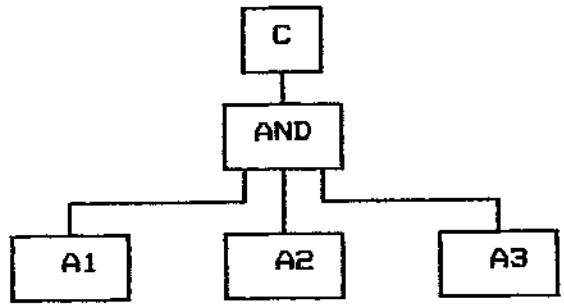


Fig. 3.1

De esta puerta AND se deduce la regla:

IF A1 and A2 and A3 THEN C.

De esta otra puerta OR:

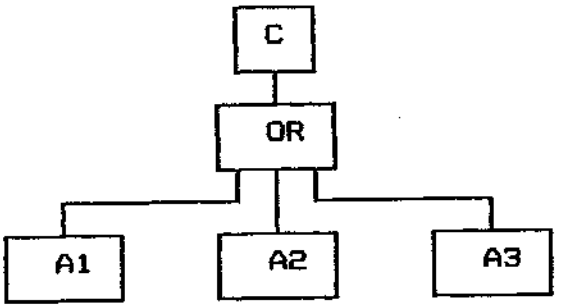


Fig. 3.2

se deducen las reglas:

IF A1 THEN C

IF A2 THEN C

IF A3 THEN C

Y, posteriormente, se trata como un sistema de producción.

Sin embargo, y como ya se comentó, la representación del árbol de fallos es una estructura que favorece implícitamente, la expresión de un conocimiento más profundo.

No existe una gran coordinación en esta aplicación industrial, entre los técnicos convencionales de estudio del diagnóstico y los que lo hacen bajo el aspecto y metodologías de la I.A. y es difícil encontrar bibliografía y documentación que aúnen ambos métodos. Entre la escasa existente, se puede encontrar [WEIS-84], [SCHW-84], [APOS-78].

Otro aspecto interesante a destacar, son las llamadas "funciones críticas", y no nombradas explícitamente hasta ahora, siendo de utilización general en plantas nucleares. Una "función crítica" es un suceso no deseado de importancia vital para el funcionamiento de la planta. Dicho de otra forma, es un fallo global de gran magnitud.

En la actualidad, el número de éstas, oscila entre cuatro y nueve, y la primera labor del operador en caso de emergencia es localizar qué función crítica ha fallado. Hecho esto, el operador pasa a corregir síntomas sin ver en principio cuales son las causas de los mismos, ayudado por el sistema CFMS (Critical Function Monitoring System) desarrollado por Combustion Engineering Inc.(C-E)[NEUS-87].

El tiempo medio que tarde el operador en corregir una desviación de una función crítica es de unos 300 Sg, [EPRI-85-A] si es que realmente se corrige. Es decir, en realidad no se va a las causas puesto que, hoy por hoy, supone un importante problema computacional.

En la consideración hecha hasta ahora del problema, baste decir, que una "función crítica" sería el SND (suceso no deseado) ó cabecera de un apreciablemente grande árbol de fallos. La búsqueda posterior en él es un proceso que supone en la realidad, varias horas, hasta encontrar la hipótesis de fallo más probable.

Cabe destacar el problema de ir a los síntomas y no a las causas, como el causante de situaciones de desconcierto y tremenda carga sicológica al operador, cuando el sistema no responde al "tratamiento" aplicado, y es ese el momento en el cual empieza el problema en estudio en esta tesis.

La importancia del problema y la aceptación definitiva de las metodologías de la I.A. para resolverlo, llevan al EPRI a confeccionar unas especificaciones de las herramientas de I.A. para esta aplicación [EPRI-85-B].

Las de más uso en la actualidad e incluso un método de selección de las mismas es encontrable en [CLEL-87], [LIDS-87], así como una buena representación del papel del operador en este proceso.

Resultado de la intensa investigación en este sentido, son el apreciable número de herramientas y SS.EE. desarrollados en sistemas eléctricos hasta el momento, y que viene explicada y referenciada en [SUN-87].

El número de trabajos allí referenciados (algo menos de 30) es aún pequeño; hay que tener en cuenta las 72 aplicaciones potenciales específicas de los SS.EE. y otros sistemas basados en el conocimiento, en este area de la industria, [ROMA-87]. Diez de éstas, corresponden al área del diagnóstico.

Este estado actual de los trabajos en este sector, viene sin duda empujado con fuerza, desde que en septiembre de 1985 se celebró el "Topical Meeting on Computer Applications for Nuclear Power Plant Operation and Control" en Washington y en la que estuvieron presentes los organismos correspondientes de USA, Europa, Japón y Canadá.

Tan sólo un trabajo presentado en 1987, muestra una unificación del tratamiento diagnóstico por medio de la utilización del árbol de fallos como herramienta convencional existente y su plena integración en un S.E. [NEUS-87]. Se hablará más adelante de este trabajo, pues siendo interesante por su integración, al no utilizar los Conjuntos Mínimos, realiza inferencias no necesarias.

El problema actual pues, ya no es en sí la forma de resolverlo, sino la falta de integración entre áreas como la I.A. y otras preexistentes y que son perfectamente capaces de integrarse.

En la resolución del problema propuesto en esta tesis, se plantea esa integración y se cree resolverla en buena medida.

3.2.- ACOTACION DEL PROBLEMA

Después de todo lo anteriormente comentado, parece claro que se está en:

* Ambiente de emergencia.
* Sujeto: Operador.
* Necesidad: Respuesta rápida.

Esta respuesta rápida, supone una plena identificación del fallo. Esta labor se realiza ya sea con cierta lentitud por la complejidad del sistema, ó bien con un riesgo de error debido a la fuerte carga de "stress" sobre el operador, causada por el abanico de decisiones que ha de tomar después y que se esquematiza en la figura 3.3.

En un sistema industrial complejo, del que se toma como referencia las centrales nucleares, aunque los mètodos son generales, se escoge como método de representación a nivel primario, el árbol de fallos, por las siguientes razones:

* Adecuación a un entorno ya construido.
* Utilidad del árbol de fallos para otras tareas,ya en la actualidad (modificaciones y mejoras en diseño, aumento de la fiabilidad del sistema...,)
* Facilidad de conversión del árbol de fallos en un sistema de producción.
* Mecanismos derivados del mismo para conocer la fiabilidad de un sistema.

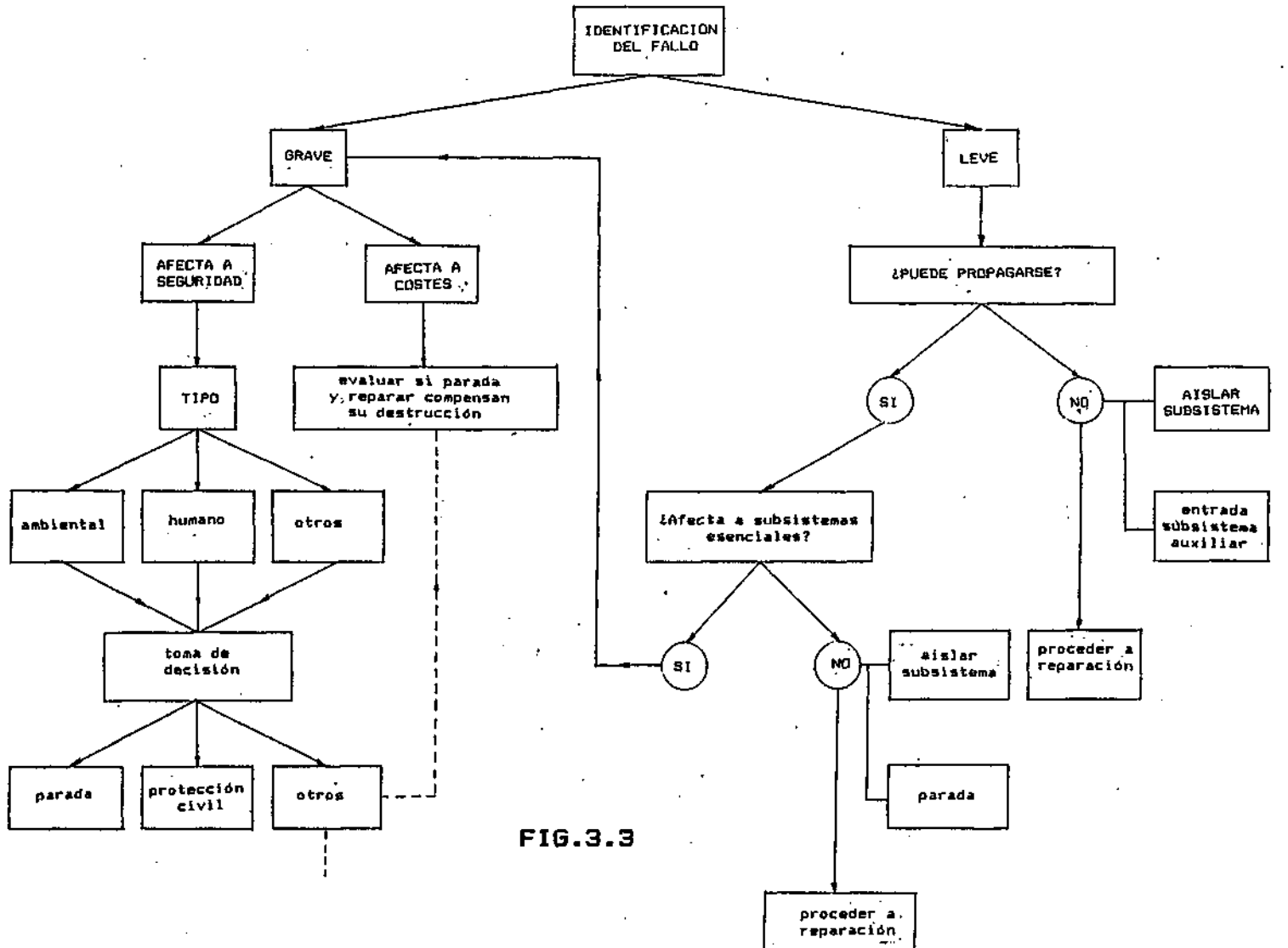


FIG.3.3

NOTA: Esta figura tan solo pretende esquematizar el complejo abanico de decisiones que pesan sobre la mente del operador ante una correcta ó no, identificación del fallo.

Escogido pues el árbol de fallos, se hace la precisión de utilizar dos únicos estados para los componentes (lógica bivaluada) por no haber datos en la actualidad que permitan utilizar la lógica multivaluada. Se parte asimismo de una función de estructura ya en forma multilineal, es decir, eliminadas las idempotencias $\Rightarrow (x_m)^n = x_m$, $n \neq 0$ derivadas del hecho de estar manejando variables booleanas.

Esto se hace de cara a introducir la fiabilidad de los componentes, para acabar conociendo la fiabilidad (ó no-fiabilidad) del sistema [HAUP-86].

Con esto se dice implícitamente que, por razones de adecuación a sistemas existentes y, por otro lado, por cuestiones de eficacia probada, se escoge como parámetro de certidumbre la fiabilidad (probabilidad de que un componente ó subsistema funcione hasta el momento t), pues refleja tanto las características de fabricación como su edad.

La fiabilidad de un componente ó subsistema queda perfectamente reflejada por la "tasa de fallos" $\theta(t)$, que varía en función del tiempo y forma la llamada "curva de la bañera", dada por la figura 3.4.

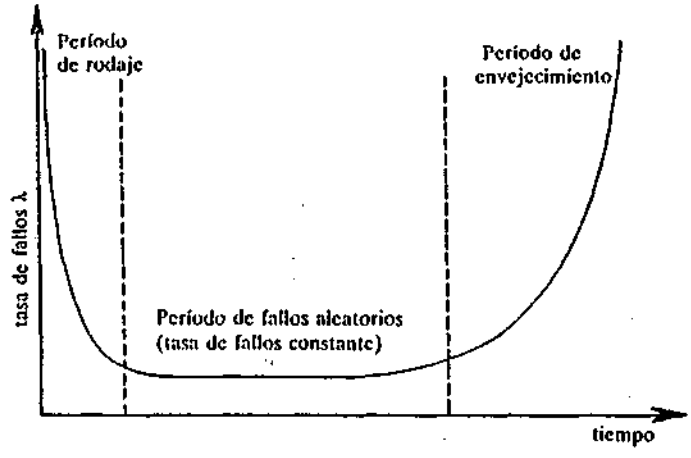


FIG. 3.4: Forma general de la tasa de fallos en función del tiempo ("curva de bañera")

Este estudio se centra en el período en el que la "tasa de fallos" es constante. La "tasa de fallos" es el inverso del "tiempo medio hasta el fallo" (MTBF) [WARL-73]. La distribución utilizada en este período es la exponencial por su facilidad de manejo matemático y su buena representación del mismo. La fiabilidad se refleja pues por la expresión:

$$F(t) = e^{-\theta t}$$

y la no-fiabilidad por:

$$F(t) = 1 - e^{-\theta t}$$

dado su caracter probabilístico.

La obtención de datos de fiabilidad es uno de los problemas más acuciantes en este momento, dado que las instalaciones tienen tanto a nivel de componentes como de sistema completo una altísima fiabilidad, de forma que a veces (muchas), y debido a las políticas de mantenimiento preventivo se sustituyen sin que se les conozca fallo.

Esto hace que este parámetro se estime en base a datos de componentes similares en otras instalaciones, y en caso de haber fallo, se reconstruye la función de fiabilidad por medio de una función de aprendizaje basada en el teorema de Bayes y ya comentada anteriormente.

Se parte pues en este estudio, de datos de fiabilidad para cada S.B. (tasas de fallo) ya conocidos.

El hecho de que la fiabilidad de un componente, ó subsistema cambie con el tiempo, es el causante de que el C.M. más probable como causa del fallo hoy, pueda no serlo mañana. Por este motivo, la resolución de incertidumbre debe hacerse cada vez que se detecte la "caida" de la función crítica.

La actualización de los datos de fiabilidad, se realiza por medio de reuniones periódicas de los técnicos. (Normalmente se corrigen tasas de fallos). Sin embargo, el hecho de que las políticas de mantenimiento adjudiquen un tope de horas de funcionamiento a los componentes, hace que la edad real de los mismos no quede reflejada de forma individual para cada uno de ellos.

Según la política de mantenimiento seguida, se adjudica al componente reparado ó sustituido la edad del antigüo, con lo que se falsea la fiabilidad del sistema, aumentándola, pero luego llegado el "tiempo medio entre revisiones" se sustituye, con lo que aumentan enormemente los costes de mantenimiento.

En cualquier caso, de cara al diagnóstico y aceptando la fiabilidad, el falseamiento de sus datos, no favorece al diagnóstico ni siquiera aumentándola, por lo que en el sistema propuesto se parte de las edades reales de los componentes.

En otro orden de cosas y toda vez que los C.M. representan "modos de fallo", el problema del diagnóstico se enfocaría como, encontrar en el momento que se detecta el SND, aquel C.M. que tenga una no-fiabilidad más alta.

Los problemas que surgen son los siguientes:

* El número de C.M. se eleva a centenares de miles e incluso varios millones.
* Si el criterio es la fiabilidad, por cual de ellos empieza a estimarse.

En la actualidad existen varios programas para generar los C.M. Hauptmanns propone como uno de los más utilizados el programa ARBOL [HAUP-79], [HAUP-86].

El inconveniente que existe, es que al utilizar una representación matricial, hasta que el proceso no ha terminado, no se dispone de información utilizable.

Existen unos 15 programas más, ya sean analíticos ó de simulación por Monte Carlo, obteniéndose según el método, parámetros de fiabilidad ó C.M.

El mayor problema es que en árboles de sistemas complejos, la deducción de todos los C.M. es computacionalmente prohibitiva, por lo que se recomienda que para estos se utilice el método de Monte Carlo , donde su mayor inconveniente es que no los saca todos, sino tan sólo aquellos que aportan más, a la no-fiabilidad del sistema. El problema adicional es que este método se basa en la calidad de los datos de fiabilidad, que no siempre existe. En ese sentido, es mejor el método analítico que se basa en el estudio de la estructura del sistema.

No existe un criterio claro sobre qué método hay que aplicar en cada caso.

Una última cuestión que es importante considerar es el tiempo empleado en la confección de un árbol de fallos, dada la complejísima interrelación entre los diversos equipos que trabajan en ello, las inconsistencias de conocimiento entre unos y otros, las complejas disposiciones físicas de los componentes y la definición del SND, hacen que hoy, el coste de confección del árbol de fallos de una planta nuclear sea del orden de 25 años-hombre. Cualquier método que ayude a reducir este tiempo sería de importante consideración.

3.3.- HIPOTESIS DE TRABAJO

En este apartado se pretende mostrar, los supuestos de los que se parte, antes de describir el planteamiento del problema.

Los supuestos ó hipótesis principales son:

* Sistemas complejos.

* Determinación de modos de fallo, es decir, diagnóstico.

* Resolución del conflicto, con tiempo restringido.

Ambito de partida: Sistemas industriales.

Modo de representación inicial: Arbol de fallos.

Sistema de resolución de incertidumbre: Basado en parámetros de fiabilidad.

Se supone además:

* Actualización permanente de la edad de los componentes.

* No existen sensores por debajo del SND.(Es el caso más desfavorable).

* Se supone conocida la "función crítica".(Es decir, se conoce el SND).

* Se supone cierta la corroboración de la caída de la función crítica.

Bajo estas hipótesis se pasa ya a plantear el problema a resolver, compuesto de una serie de subproblemas, que se describen a continuación, justificando su necesidad.

3.4.-PLANTEAMIENTO DEL PROBLEMA

A.- Construcción de árbol de fallos a partir de reglas y adjudicación de niveles a las puertas.

Dado el gran coste que actualmente tiene la construcción de un árbol de fallos, se propone un procedimiento que permita, a partir de reglas individualizadas con ciertas condiciones, deducir el árbol de fallos, determinando además, cuáles son los S.B., cuales son puertas y cual sería la cabecera del árbol, deducible de dichas reglas, amén de adjudicarle simultáneamente niveles a dicha estructura.

Este sistema, al integrar las reglas en un árbol, permitirá detectar inconsistencias de las reglas en cuanto que no se forme un árbol único ó, por el contrario, construir varias estructuras que permitan deslindar el conocimiento de alto nivel (estructural) de un conjunto de reglas no ordenadas.

Este procedimiento ha de permitir, por tanto, la integración de conocimiento de equipos muy diferenciados.

Las reglas habrán de cumplir tres condiciones:

* Ser de la forma < condición > ——→ < situación >.

* Tener tan sólo una < situación >.

* Tener una ó varias < condiciones > pero de tener varias, estar unidas tan sólo por "∧" (conjunción lógica).

Este procedimiento permitirá aplicar directamente al diagnóstico la estructura del árbol, fundamental para atacar el problema de la restricción temporal.

B.- Partiendo de un árbol de fallos dado, estructuración del mismo por niveles.

Esta cuestión es fundamental, pues dependiendo del tiempo disponible, la determinación de la hipótesis de fallo será a un nivel más alto (más error) ó más bajo (menos error).

C.- Depuración del árbol de fallos y renombramiento de puertas.

Depuración significa, eliminación de puertas redundantes. La presencia de éstas, ocurre por construirse los árboles siguiendo muy detalladamente la disposición física de los componentes. Sin embargo, de cara a la construcción de los C.S. a ese nivel, el hecho de haber una puerta OR como entrada de otra OR es redundante e innecesario, lo mismo con una puerta AND debajo de una AND.

Eliminadas estas puertas, el número de niveles en una rama disminuye y, por tanto, se procede a renombrarlas, es decir, adjudicarlas nuevos niveles.

Es importante hacer notar, que después de la depuración, el árbol que queda ya no refleja la disposición física de los componentes, pero sí, su estructura ante el fallo ó SND.

D.- Deducción de C.M. por niveles y estructurados.

Esta deducción es necesaria para el estudio de los modos de fallo de un subsistema, y no del completo (aunque podría serlo). Su estructuración es asímismo necesaria para estudiar los niveles del subárbol.

E.- Resolución del conflicto para obtener el C.M. menos fiable.

Obtenidos los C.M. en cabecera del árbol y según se los ha definido, es necesario saber cual es el menos fiable. Dado a veces el gran tamaño de estos C.M. y por una pura cuestión de eficiencia computacional, es necesario replantearse la compatibilidad entre independencia y simultaneidad en los S.B. que componen un determinado modo de fallo.

F.- Deducción de Conjuntos Virtuales (C.V.)

Queda desde aquí planteada la existencia de estos como aportación original necesaria para la resolución estructural del problema del diagnóstico, con constricción temporal. Su definición y utilización se mostrará en la resolución del problema.

G.- Resolución de incertidumbre en C.V.

Basada en los datos de fiabilidad a un nivel bajo pero atendiendo a la estructura del árbol, habrá que plantear el problema derivado del "tamaño" del desconocimiento del operador ante una decisión.

H.- Como caso particular se planteará la resolución global del problema, habiendo sensores por debajo del SND y comparación de este método con el utilizado en el S.E. construido por C-E (Combustion Engineering Inc)[NEUS-87].

CAPITULO CUARTO

"RESOLUCION DEL PROBLEMA"

4.1.- INTRODUCCION

El conjunto de métodos expuestos a continuación se proponen como resolución a los apartados A), B), C), D), E), F), G), del planteamiento del problema, y se exponen en ese mismo orden. Cada uno de estos, constituye un método de aplicación en si mismo, para resolver parcelas aisladas del tratamiento del problema diagnóstico en la industria, sin embargo, y aunque se exponen separadamente, la secuencia de los mismos hace que se vea como constituyen las distintas fases de un proceso más completo.

Los temas tratados y en los que se propone solución son pues los siguientes:

A) Construcción de un árbol de fallos a partir de reglas.
B) Estructuración de un árbol de fallos por niveles y representación del mismo.
C) Depuración del árbol de fallos y renombramiento de puertas.
D) Deducción de Conjuntos Minimos por niveles.
E) Resolución del conflicto para obtener el CM menos fiable.
F) Deducción de Conjuntos Virtuales.
G) Resolución de incertidumbre en CV.

En aras de la claridad en la exposición se denominarán las puertas con OR y AND. Sin embargo, la representación real sigue las normas de difusión internacional, según la tabla 4.1.

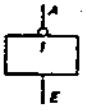
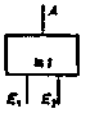
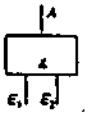
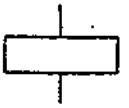
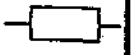
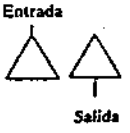
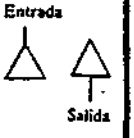
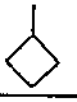
Significado	Símbolo	Símbolo	Símbolo
entrada estándar para un fallo primario (representa un suceso básico o primario)			
conexión lógica de negación; cuando se cumple E no se cumple A y viceversa			
puerta del tipo "o". A se cumple cuando se cumplen E_1 o E_2 o ambos (unión lógica)			
puerta del tipo "y". A se cumple únicamente cuando se cumplen E_1 y E_2 simultaneamente (intersección lógica)			
comentario			
puertas de transferencia que se emplean cuando un árbol de fallos se divide en varias partes			
entrada para un fallo secundario (fallo como consecuencia de otro fallo anterior)			

TABLA 4.1: Simbolos principales para la representación gráfica de árboles de fallos.

Durante la exposición, la representación de los C.S., en cada nivel se hará en forma de listas, denominando a cada suceso básico con la notación x_i (i = 1,2,.....) y siendo $\bar{x}_i$, la negación del mismo.

La representación de las puertas se indicará en el árbol primeramente con la identificación del tipo de puerta, seguida de dos números. El primero indicará el nivel donde se encuentra la misma, siendo el nivel 1 el de la puerta colocada inmediatamente debajo del SND, y el segundo el identificador de cada puerta dentro de su nivel.

Así pues, si por ejemplo en el nivel 3 existen dos puertas AND y una OR, su identificación será:
AND 3.1 ; AND .2 ; OR 3.1 ;

De esta forma el árbol de fallos que se muestra en la Fig. 4.1, queda como se muestra en la Fig. 4.2

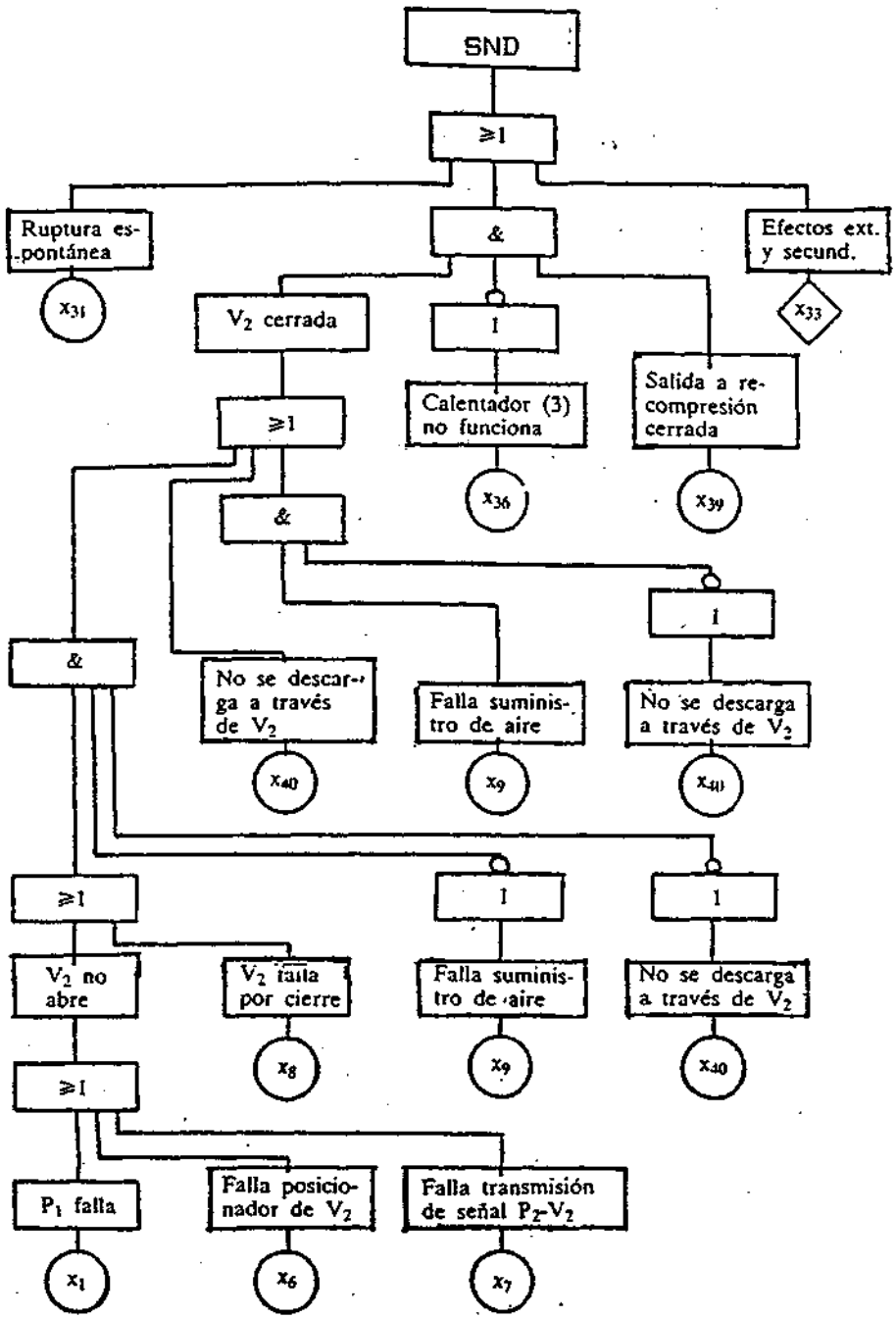


Fig. 4.1: Arbol de fallos con sistema de representación según normas.

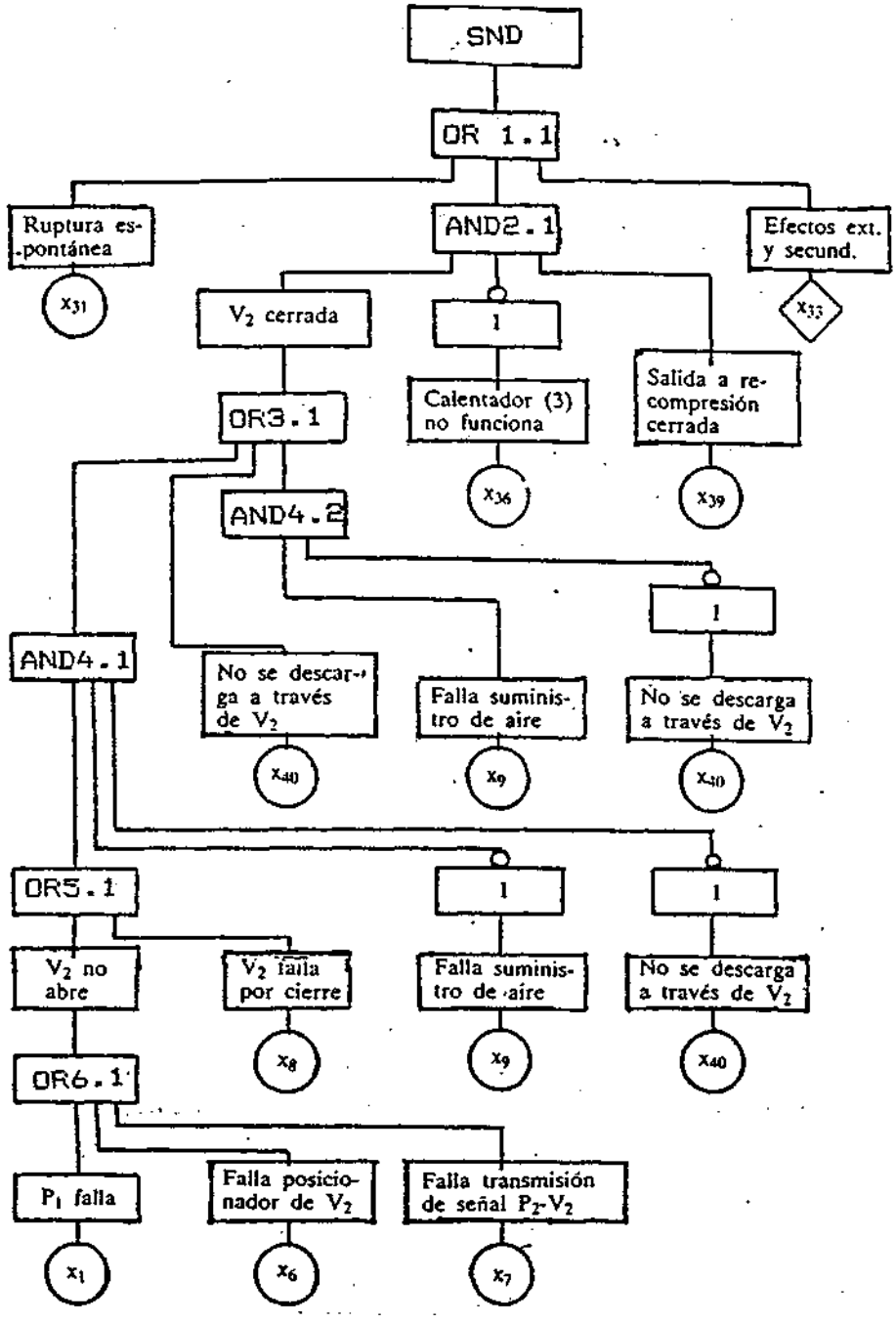


FIG.4.2: Arbol de fallos con la nueva denominación de puertas

4.2.- CONSTRUCCION DE UN ARBOL DE FALLOS A PARTIR DE REGLAS

Tal y como se expuso en el apartado A) del planteamiento del problema, todo método que ayude a la construcción de un árbol de fallos es importante de considerar. Vistas las ventajas del manejo de la estructura del árbol de fallos en cuanto a la eficiencia en el diagnóstico, se plantea un método para construirlo a partir de reglas simples que tan solo han de cumplir estas condiciones:

1) Ser de la forma <Condición> ⟶ <Situación>.

2) Tener tan solo una <Situación>.

3) Tener una ó varias <condiciones>, pero de tener más de una, estar unidas tan solo por " ∧ " (disyunción lógica).

En estas condiciones, cada regla puede representarse como una lista de dos ó mas elementos, en los que el situado a la derecha de todos es la <Situación>. En consecuencia, y por observación de un árbol de fallos se dan las siguientes características:

* Todas las <Situaciones> son puertas del árbol.

* La puerta del nivel 1 es aquella <Situación> que no sea además ninguna <Condición>.

* Las puertas OR son <Situaciones> que están más de una vez, es decir, en más de una regla.

* Las puertas AND son <Situaciones> que están una sola vez, es decir, en una sola regla, pero que han de tener forzosamente más de una <Condición>.

* Si existe una puerta AND que solo tiene una <Condición> es, ó bien una regla incompleta (habría que añadir alguna <Condición>), ó bien, es una regla redundante y por tanto eliminable.

* Si existe más de una <Situación> que no sea <Condición> de ninguna regla, indica la existencia de varios SND y por tanto de varios árboles de fallos.

4.2.1.- METODO DE SOLUCION PROPUESTO

En consecuencia de lo anterior se propone el siguiente procedimiento:

1) Se localiza a la derecha de todas las reglas la <Situación> que no se encuentre en ninguna parte izquierda.

2) Se ve si está más de una vez. Si lo está es puerta OR. Si no, pasar a 3).

3) ¿Tiene varios conceptos a su izquierda?

SI ——→es puerta AND.

NO ——→ es una puerta falsa. (eliminarla).

4) Si es puerta OR comprobar cúal de las partes <Condiciones> son también alguna <Situación>. Las que estén son puertas. Las que no, son S.B.

5) Si es puerta AND comprobar cúal de las <condiciones> son también <Situación>. Las que están son puertas. Las que no, son sucesos básicos.

6) Con cada una de las puertas deducidas se repite el proceso desde 2) y se aumenta en una unidad el identificador de nivel en cada puerta.

4.2.2.- VENTAJAS E INCONVENIENTES

Ventajas

* Capacidad de detección de todos los posibles SND implícitos en el conocimiento representado por las reglas.

* Capacidad para detectar inconsistencias ó conocimiento inútil, así como lagunas de conocimiento de cara a un SND.

* Gran facilidad para incorporar nuevo conocimiento.

* Total capacidad de estructuración y jerarquización del conocimiento de cara al diagnóstico.

Inconvenientes

* No apto en principio para sistemas cuyo conocimiento en cuanto a causas de fallos no esté completo.

4.3.- ESTRUCTURACION DE UN ARBOL DE FALLOS POR NIVELES Y REPRESENTACION DEL MISMO

4.3.1.- ESTRUCTURACION POR NIVELES

* Los niveles estarán adjudicados a las puertas, independientemente del tipo que sean.

* El nivel 1 corresponderá siempre a la puerta situada inmediatamente debajo del SND.

* En las distintas ramas del árbol, el nivel de cada puerta será una unidad mayor que la de la puerta a la que ésta concurra.

* La puerta que alcance el nivel más alto en todo el árbol, determinará el nivel del mismo, es decir, marcará la máxima profundidad del mismo.

4.3.2.- REPRESENTACION DEL ARBOL

* Se hará en forma de listas.
* En cada nivel habrá tantas listas como puertas pertenezcan al mismo.
* Cada una de estas listas, tendrá como primer elemento el identificador de la puerta (OR ó AND), seguida de los identificadores de puertas ó S.B. que concurran a ella en el árbol, desde el nivel inmediato inferior.

* Las puertas se identificarán con el identificador de tipo, seguidas de dos números. El primero de éstos indicará el nivel de la misma y el segundo la identificación de la puerta dentro de cada nivel.
* Los SB se identificarán con una letra (en esta tesis será una X), seguida de un número que representa la identificación del mismo dentro de todo el sistema.

4.3.3.- VENTAJAS E INCONVENIENTES

Ventajas

* La estructuración por niveles permite posteriormente la aportación progresiva de información en la tarea del diagnóstico, ya sea acotando el tiempo, ó bien, el nivel.
* La representación en forma de listas, permite, además del tratamiento de la información por lenguajes de tratamiento de listas, una gran claridad y pequeño espacio, así como una gran facilidad para la obtención analítica del conjunto de C.M.

Inconvenientes

* Difícil visualización de la estructura del árbol, por lo que esta representación debe ser utilizada tan solo como paso previo de los procedimientos propuestos a continuación.

4.4.- DEPURACION DEL ARBOL DE FALLOS Y RENOMBRAMIENTO DE PUERTAS

Frecuentemente, a una determinada puerta, le suceden, además de sucesos básicos, una ó más puertas del mismo tipo. La causa suele ser, ya sea la estricta formación del árbol según la disposición física de sus componentes, ó bien la colocación de un sensor. Sin embargo, esta situación y de cara a la extracción de los C.M. es simplificable. Sea el árbol de la Fig. 4.3:

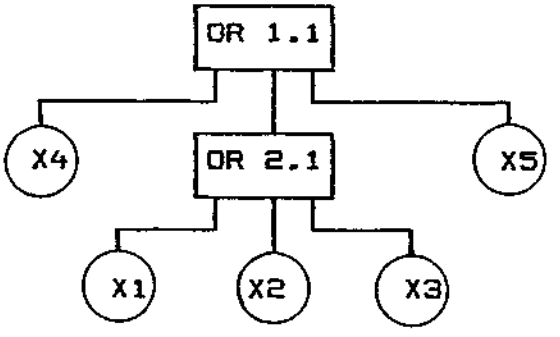


fig. 4.3

Los C.M. de este árbol son:

NIVEL 2

(X1) (X2) (X3) que son tres.

NIVEL 1

(X1) (X2) (X3) (X4) (X5) que son cinco.

que son los mismos que los de la siguiente disposición de la Fig. 4.4:

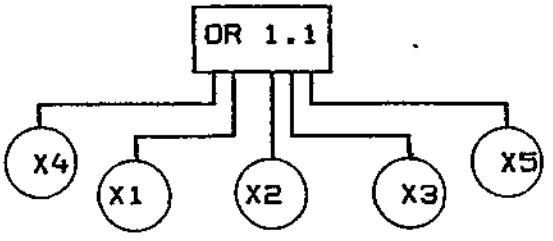


fig 4.4

quedándonos reducido el árbol a un solo nivel.

Con puertas AND ocurre lo mismo. Sea el árbol de la Fig. 4.5:

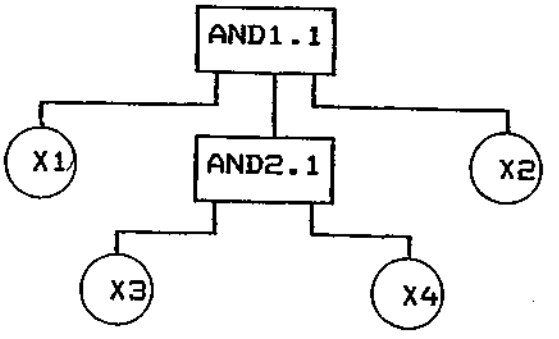


fig 4.5

Los C.M. de este árbol son:

NIVEL 2

(X3 X4) que es uno.

NIVEL 1

(X1 X2 X3 X4) que es uno.

que son los mismos que los de la configuración de la Fig. 4.6:

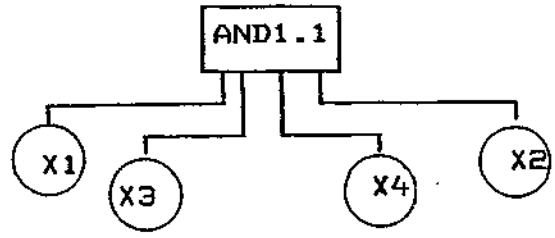


Fig. 4.6

Por tanto, se ve que de cara a la deducción de los C.M. del primitivo árbol, se llega más rápidamente procediendo a la simplificación del mismo eliminando niveles.

Al proceso descrito se le denominará depuración, y se propone a continuación.

4.4.1.- METODO DE SOLUCION PROPUESTO

1) Se revisa cada puerta comenzando por el nivel N-1 en un árbol de N niveles. Si en su contenido encuentra puertas del mismo tipo, las sustituirá por el estricto contenido de éstas, en el nivel anterior.
2) Terminado el proceso con el nivel N-1, se pasa a hacer lo mismo en el nivel N-2, y así sucesivamente hasta llegar al nivel 1.

3) Depurado el árbol en todos sus niveles, se aplica de nuevo el procedimiento de representación del mismo ya descrito, renombrando todas las puertas.

4.4.2.- VENTAJAS E INCONVENIENTES

Ventajas

* El árbol depurado es fiel representación de la estructura de los componentes de cara al fallo.

* El número de niveles del mismo, es siempre, menor ó igual que el del árbol sin depurar, con lo que el tiempo empleado en el diagnóstico será menor.

* De la relación entre el número de niveles del árbol depurado y sin depurar, podría establecerse una medida parcial de la complejidad del sistema.

Inconvenientes

* El árbol depurado ya no representa la estricta disposición física de los componentes del sistema.

4.5.- DEDUCCION DE CONJUNTOS MINIMOS POR NIVELES

4.5.1.- METODO PROPUESTO

Partiendo de un árbol depurado de N niveles, el proceso es el siguiente:

Nivel N

a) Se revisa el tipo de puerta de cada una de las del nivel N. En este nivel las puertas sólo contienen SB.

b) Si la puerta es OR formar tantos C.S. como S.B. haya.
Si la puerta es AND formar un solo C.S. que contiene a todos los S.B. que entren en ella.

Niveles N-1,N-2,.....,2,1

Dado que el árbol está depurado, ya no se necesita revisar el tipo de puertas puesto que si la del nivel N era OR, se sabe que la de nivel N-1 será AND y viceversa.

Ahora se encuentran puertas de tres tipos:

A) Que contengan sólo sucesos básicos.
B) Que contengan sólo otras puertas.
C) Mixtas.

Para ello se actuará de la siguiente forma:

c) Comprobar el tipo de puerta.

* Extraer puertas tipo A y sustituirlas por el resultado de aplicar el paso b) anterior.

* Extraer puertas tipo B:

1) Si la puerta es OR, formar como C.S. todos los que concurran a ella provenientes del nivel inferior.

2) Si la puerta es AND, construir como nuevos C.S. todas las combinaciones posibles que se puedan formar, de manera que haya un solo C.S. del nivel inferior de cada una de las puertas del mismo. Su número será igual al producto de las cantidades de C.S. provenientes de las ramas del nivel inferior que concurran a ella.

3) Comprobar si:
- Algún nuevo C.S. contiene a otro.De ser así, eliminar el que contiene.Repetir este proceso hasta que ninguno esté contenido en otro.

- Algún C.S. contiene S.B. repetidos.De ser así y por la propiedad de idempotencia, dejar tan solo uno.
- Algún C.S. contiene a un S.B. y su negado. De ser asi, eliminarlo.(Monotonia).

Terminado este proceso, quedan los C.S. que son mínimos en cada puerta.

4) Contar los C.M. deducidos en cada puerta y archivar este dato.

* Extraer_puertas_tipo_C

5) Extraer las puertas contenidas y aplicar el mismo proceso con éstas que con las puertas tipo B.(Pasos 1,2,3).
6) Si la puerta es OR añadir los S.B. como nuevos C.S. al resultado del punto 5).
Aplicar de nuevo 3) y 4).
7) Si la puerta es AND, añadir los S.B. a todos los C.M. deducidos en 5).
Aplicar de nuevo 3) y 4).

d) Etiquetar con el nombre de la puerta los C.M. extraidos de cada una.

4.5.2.- CARACTERISTICAS DEL METODO PROPUESTO

* El método es del tipo analítico pues deduce la totalidad de los C.M.

* Al ser un método analítico solo se realiza una vez, teniéndose almacenados los C.M. de todas las puertas del sistema.

* El procedimiento expuesto, queda completado con algunas cuestiones operativas a nivel de programa y que se encuentran en el anexo 2 de esta tesis, como serían la colocación artificial de puerta OR al primer nivel en caso de serlo AND, tratamiento de subárboles repetidos, etc...

4.5.3.- VENTAJAS E INCONVENIENTES

Ventajas

* Se dispone de la información sobre cuáles son los C.M. en cada puerta, con lo que se puede aplicar el diagnóstico a subsistemas y no al sistema completo si así se quisiera.

* Al resultar muy idóneo para lenguajes de tratamiento de listas, hace que la utilización de los mismos permita el acceso directo a un C.M. (que es una lista de SB) ganando eficiencia y claridad en relación a métodos que utilizan representaciones matriciales que requieren una posterior interpretación.

* Permite una visualización puerta a puerta de la formación de los C.M.

Inconvenientes

* Al igual que en otros métodos actuales, se produce explosión combinatoria por lo que el tiempo empleado es grande. En las circunstancias de este trabajo no es un grave inconveniente, toda vez que, el proceso ha de realizarse una sola vez.

4.6.- RESOLUCION DEL CONFLICTO PARA OBTENER EL C.M. MENOS FIABLE

Como ya se dijo en su momento, cada C.M. representa un modo de fallo. Es decir, una causa que provoca el SND.

Saber entonces la causa del SND, supone ver cual de los C.M. aporta una mayor no-fiabilidad al sistema en ese momento.

Deducir esta cuestión en un árbol pequeño por medio de los métodos actuales no ofrece ningún problema de tiempo.

Sin embargo, en un sistema complejo donde en vez de algunos C.M., puede haber millones, el problema se convierte dentro de las limitaciones de tiempo en irresoluble.

Antes entonces de aplicar cualquier método que suponga una búsqueda exhaustiva ya no sólo del C.M. sino por medio de cálculos, cúal es el componente que más aporta a la no-fiabilidad del sistema, en la actualidad se recurre a la simulación por Monte Carlo. Aún asi y con sistemas muy complejos sigue siendo muy costoso computacionalmente.

Un sencillo sistema como el representado por el árbol de fallos que se muestra en la Fig. 4.7, ya tiene 61 C.M.

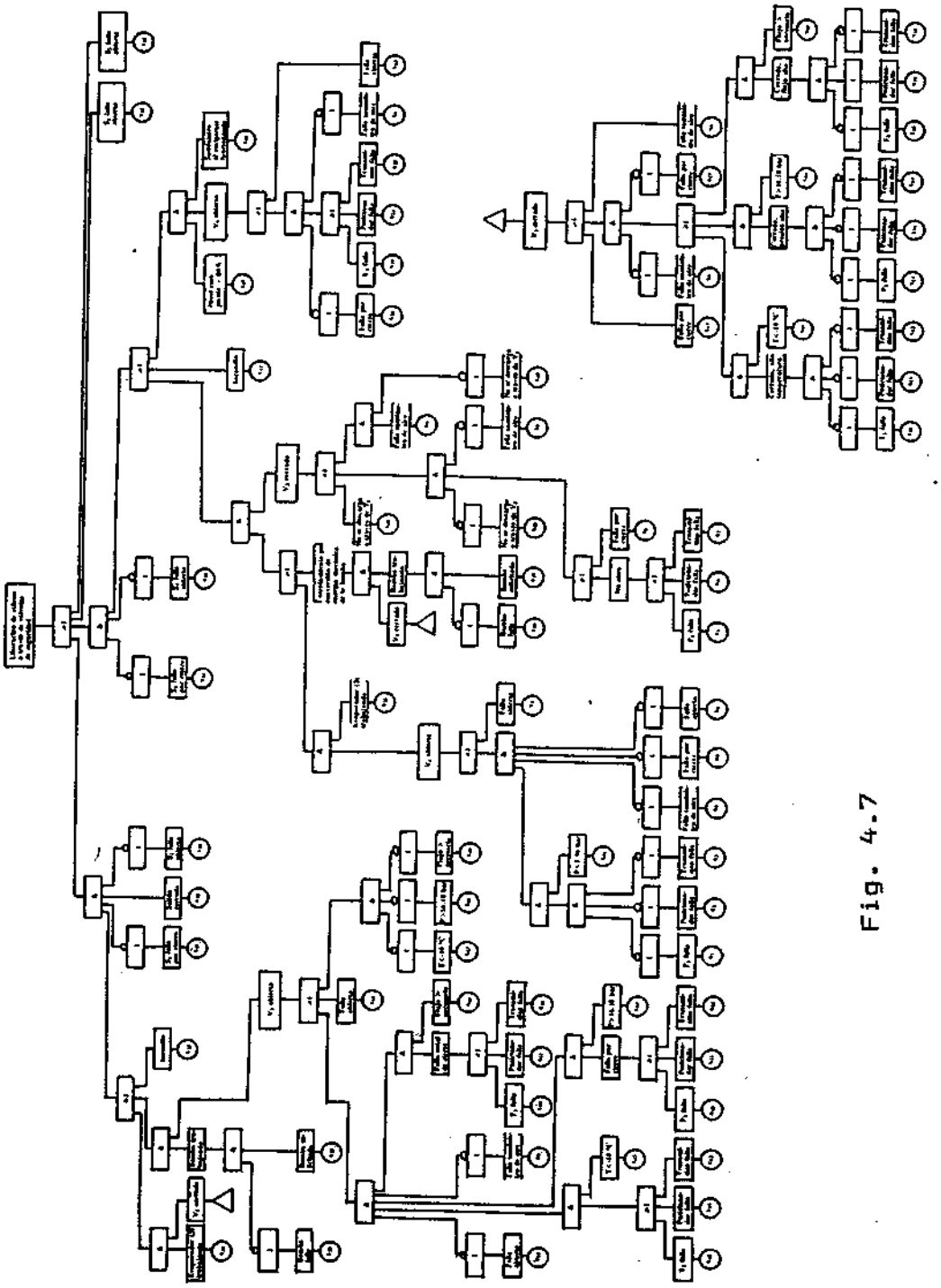


Fig. 4.7

Esta búsqueda, en la actualidad y en un sistema complejo, es muy poco eficiente computacionalmente, como ya se ha descrito anteriormente. Sin embargo, son los datos de fiabilidad los que obligan a hacer algún tipo de resolución de incertidumbre cada vez que se presente el SND, dado que dependen del tiempo. Y se necesita asimismo un método que permita simplificar este proceso, sin perder la veracidad del comportamiento del sistema.

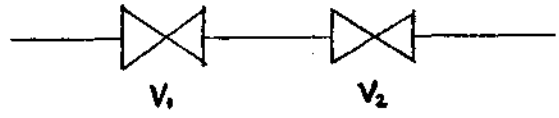
En este sentido se propondrá un método basado en una modificación de la transmisión y cálculo de probabilidad de fallo en árboles de fallos, que vamos a justificar previamente.

Los sistemas técnicos constituidos por componentes y subsistemas dispuestos en serie ó paralelo ó una combinación de ambos, tienen adjudicadas unas determinadas representaciones en términos de fiabilidad. En éstas, se permiten dos únicos estados, funcionamiento ó fallo. La representación en términos de fiabilidad no corresponde siempre a la disposición física, pues depende de la función a realizar or los componentes ó subsistemas.

Se dice entonces que una disposición en sentido de la fiabilidad es:

SERIE: Cuando todos sus componentes tienen que funcionar para cumplir la misión.
PARALELO: Cuando funciona si al menos un componente funciona.

Esto se ve mejor con un ejemplo. Sean 2 válvulas colocadas físicamente en serie con dos funciones diferentes.



Se vé como según la función a realizar, su disposición en sentido de la fiabilidad es diferente:

<u>1ª función</u> Paso de flujo de A a B.

Para que esto ocurra asi, tanto V_1 como V_2 deben de dejar pasar flujo, por tanto su disposición en términos de fiabilidad es SERIE.

2ª función Interrupción de flujo de A a B.

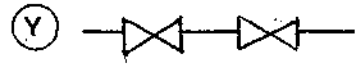
Para que esto ocurra basta con que una de ellas esté cerrada, por tanto su disposición en términos de fiabilidad es PARALELO.

Como sólamente se admiten dos estados (funcionamiento ó fallo), cada componente ó subsistema tiene asignada una probabilidad de funcionamiento "p". Por tanto la probabilidad de no-funcionamiento es q=1-p.

Véase entonces, cómo se pueden evaluar las probabilidades de funcionamiento de un sistema, conociendo las de sus componentes y posteriormente las de fallo.

PROBABILIDAD DE FUNCIONAMIENTO DEL SISTEMA CONOCIENDO LAS PROBABILIDADES DE FUNCIONAMIENTO DE SUS COMPONENTES

(Se suponen disposiciones en sentido de la fiabilidad)

SERIE : 

Tienen que funcionar V_1 y V_2. Suponiendo que p_1=0,7 y p_2=0,9 y aplicando la teoría de probabilidad a dos acontecimientos simultáneos e independientes, se tiene que la probabilidad de funcionamiento del sistema es:

$$p = p_1 \cdot p_2 = 0,7 \cdot 0,9 = 0,63$$

que es menor que la de su peor componente.

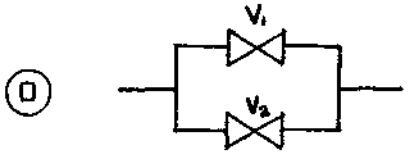
Aquí es donde se va a hacer la salvedad de la simultaneidad.

En los sistemas industriales la condición de simultaneidad es una cuestión discutible, dado que nunca un componente es exactamente igual a otro, aunque sus características sean las mismas. Considerar la simultaneidad como el hecho de que ambos en ese momento estén funcionando es lo mismo que considerar su fallo simultáneo. Los técnicos de sistemas saben muy bien que la simultaneidad en fallos, si de verdad es así, es síntoma de fallo de causa común y esa dependencia debe estar reflejada en el árbol de éxitos ó fallos, que se supone bien construido. Luego, de alguna forma, vemos que la simultaneidad e independencia en los sistemas industriales es un fenómeno casi imposible.

Considerando entonces la condición de independencia de sucesos con toda su potencia, vuélvase al ejemplo. Parece claro entonces que es más probable que V_2 funcione, que que lo haga V_1 en este instante, dado que su probabilidad de funcionamiento es mayor. Pero V_1 tiene una $p_1=0,7$ y por tanto la probabilidad de que el sistema funcione será como poco 0,7, y por tanto se acepta como probabilidad de funcionamiento del sistema :

$$p_s = \min \{ p_1, p_2 \}$$

PARALELO :



con $p_1=0,7$ y $p_2=0,9$

El supuesto de que se parte es que tanto V_1 como V_2 pueden realizar la función total, pues de lo contrario no sería una disposición paralelo.

De fallar en el instante "t", parece que antes fallará V_1, quedándo el componente V_2 en funcionamiento, que tiene la probabilidad mayor. Si se aplican las reglas para la probabilidad conjunta de 2 acontecimientos compatibles se tiene que la probabilidad de funcionamiento del sistema es:

$p_{\infty} = p_1 + p_2 - p_1 . p_2 = 0,7 + 0,9 - 0,63 = 0,97$

cuestión que contradice intuitivamente lo expuesto antes, pues en el mejor de los casos (que efectivamente falle primero V_1) la probabilidad de funcionamiento del sistema es 0,9.

Se acepta pues que la probabilidad de funcionamiento es:

$$p_{\infty} = \max \{ p_1 , p_2 \}$$

PROBABILIDAD DE FALLO DEL SISTEMA CONOCIENDO LAS PROBABILIDADES DE FALLO DE LOS COMPONENTES

Quizás esto interese más, puesto que el primer objetivo del diagnóstico es encontrar el fallo, y más aún, porque en el árbol de fallos se maneja este concepto. Se siguen suponiendo las disposiciones en sentido de la fiabilidad.

SERIE: La función es correcta si V_1 y V_2 realizan la función.

¿Cuándo falla? Si fallan ó V_1 ó V_2.

Con las anteriores probabilidades de funcionamiento, que eran: $p_1=0,7$ y $p_2=0,9$ se tiene que las correspondientes probabilidades de fallo son: $q_1=0,3$ y $q_2=0,1$ y entonces, si basta con que falle uno, parece claro que fallará primero el que tenga mayor probabilidad de fallo en el instante "t".

Se acepta pues que:

$$q_s = \max \{ q_1, q_2 \}$$

PARALELO: La función es correcta si V_1 ó V_2 realizan la función.

El fallo se produce si fallan V_1 y V_2. Con los mismos datos de antes, parece que el primero en fallar será V_1 y queda aún en servicio el sistema con $q_2=0,1$ que será su probabilidad actual de fallo, luego se acepta que:

$$q_s = \min \{q_1, q_2\}$$

PRUEBA DE CONSISTENCIA

Sistemas según el sentido de la fiabilidad.

SERIE: Dado que se ha aceptado que q=1-p para un solo componente lo mismo tendrá que ocurrir para un subsistema. Por tanto habrá de cumplirse que:

$$q_s = 1 - p_s$$

Acudiendo al ejemplo se ve que $p_s = \min\{0,7 \ , \ 0,9\} = 0,7$ y que $q_s = \max\{0,3 \ , \ 0,1\} = 0,3$ luego para el conjunto del sistema se cumple igualmente. Por inducción quedaría probado para "n" elementos.

PARALELO: Lo mismo. $p_s = \max\{p_1, p_2\} = \max\{0,7 \ , \ 0,9\} = 0,9$

$$q_s = \min\{q_1, q_2\} = \min\{0,3 \ , \ 0,1\} = 0,1$$

luego se cumple igualmente.

4.6.1.- METODO PROPUESTO

Consecuente con lo anterior se propone como método de cálculo de las probabilidades de funcionamiento del sistema, ó bien de fallo que nos permita deducir el conjunto mínimo menos fiable, el sistema de cálculo representado en la Fig. 4.9

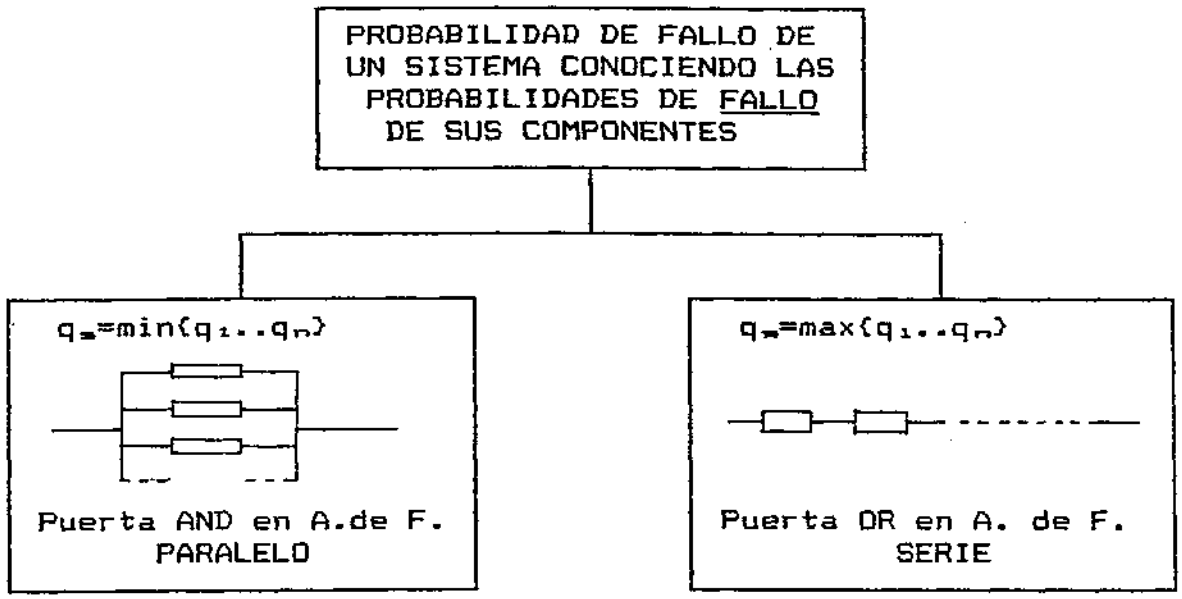
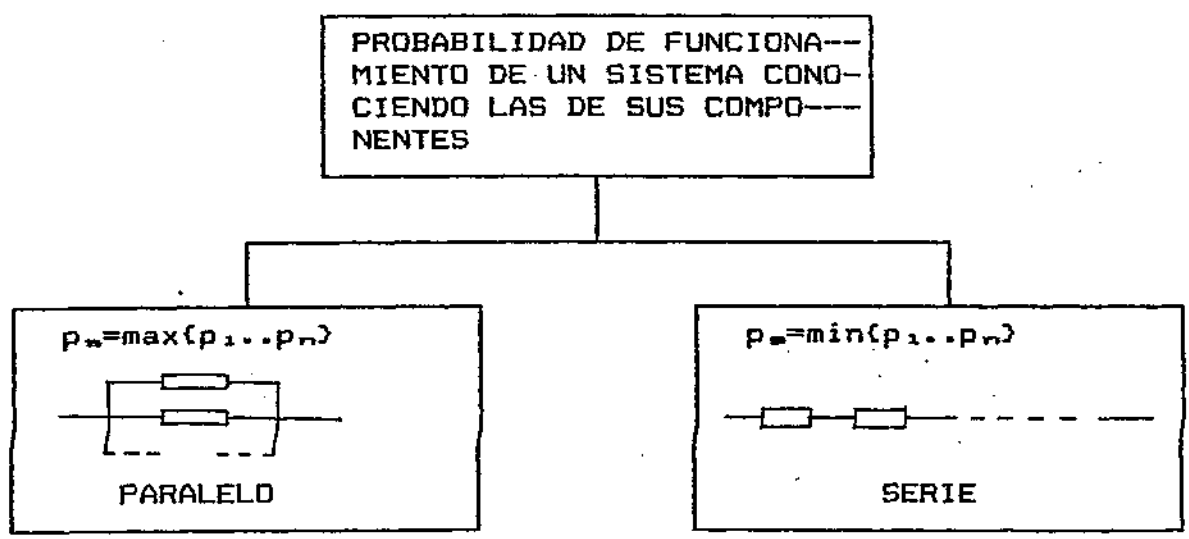

PROBABILIDAD DE FALLO DE UN SISTEMA CONOCIENDO LAS PROBABILIDADES DE FALLO DE SUS COMPONENTES
$q_s=\min(q_1..q_n)$
Puerta AND en A.de F.
PARALELO
$q_s=\max(q_1..q_n)$
Puerta OR en A. de F.
SERIE

Fig. 4.9


PROBABILIDAD DE FUNCIONA-- MIENTO DE UN SISTEMA CONO- CIENDO LAS DE SUS COMPO--- NENTES
$p_s=\max(p_1..p_n)$
PARALELO
$p_s=\min(p_1..p_n)$
SERIE

4.6.2.- VENTAJAS E INCONVENIENTES

Ventajas

* Buen reflejo de la realidad ante fallos denominados "simultáneos" que no lo son en realidad.
* Abarata los costes de mantenimiento preventivo al no obtenerse cifras tan altas de no fiabilidad.

Inconvenientes

* Los datos de fiabilidad deben ser muy precisos.

4.7.- DEDUCCION DE CONJUNTOS VIRTUALES

4.7.1. INFORMACION ESTRUCTURAL DEL ARBOL DE FALLOS

A pesar de la eficiencia del método de resolución propuesto en el apartado 4.5 el proceso de eliminación de incertidumbre en un árbol complejo sigue siendo de un apreciable coste computacional. Pues bien, ya que hemos estructurado el conocimiento en un árbol de fallos, se plantea una cuestión adicional:

¿Se puede en base al estudio únicamente de la estructura del árbol eliminar parte de la incertidumbre?

Se analizarán unas cuestiones previas, que acaban conformando parte, del procedimiento presentado en el anexo 2 de ésta tesis.

Se parte de un árbol ya depurado. Téngase una puerta OR con un sensor sobre ella:

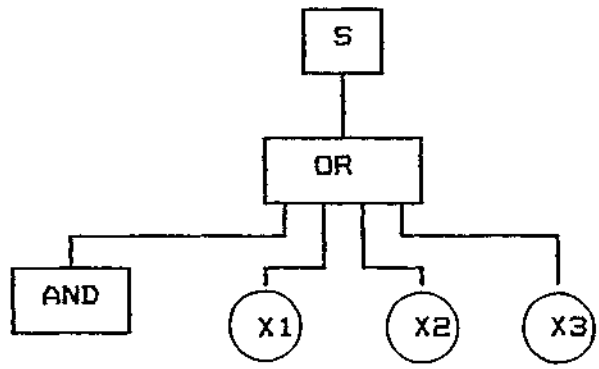


Fig. 4.7

Si el sensor indica FALLO, éste puede venir de cualquiera de los que están por debajo. Si indica NO FALLO dice que ninguna de sus ramas entrantes ha fallado. En otras palabras "SENSORES SOBRE PUERTAS "OR" DAN LA MAXIMA INFORMACION CON "NO FALLO". (No falla ninguna de sus entradas)". De la misma forma:

"SENSORES SOBRE PUERTAS "AND" DAN LA MAXIMA INFORMACION CON "FALLO" (Fallo de todas sus entradas)".

Puesto que el operador necesita conocer los fallos individualizados de S.B., se obtiene de forma inmediata un criterio para asegurar el fallo de un componente y desechar el de otros. Vease el ejemplo:

Sea un árbol que en un determinado nivel tiene la puerta [OR] con estos tres C.M.:

(X10 X1 X12 X43) (X13 X14 X15 X44)
(X11 X15 X16 X45)

y la puerta [AND] con:

(X11 X45) (X15 X45) (X16 X45)
además de otras puertas que no se consideran.

Se parte de la siguiente situación:

[OR] da señal de NO FALLO.
[AND] da señal de FALLO.

y además se detecta fallo en el SND.

La información estructural que estas señales dan es:

De la puerta OR se deduce que ninguno de sus C.M. ó modos de fallo parciales ha fallado, aunque puede estarlo alguno de sus componentes ó S.B.

En un intento de aproximarse más, se buscarían grupos de S.B. que estuviesen en varios C.M. En este caso, no los hay, es decir, la intersección es vacía por lo que se supondría que esos elementos no han fallado.

Sea la señal de FALLO en la puerta AND. Esta dice que ha fallado al menos uno de los tres C.M. La intersección de estos tres es el S.B. X45 y por tanto deducimos el fallo seguro del X45.

Esto y puesto que éste S.B. se encuentra en el C.M. proveniente de la puerta OR : (X11 X15 X16 X45) y se sabe que éste no es modo de fallo, dice al menos que ó X11 ó X15 ó X16 no han fallado.

Se deduce de aquí en un principio que CONVIENE COLOCAR SENSOR DE FALLO SOBRE PUERTAS "AND" TALES QUE TENGAN CONJUNTOS MINIMOS DE INTERSECCION NO VACIA.

Esto ocurrirá siempre que concurran a una puerta AND además de las puertas de tipo OR, algún suceso básico.

Este ó estos S.B. se convierten a ese nivel en condiciones necesarias (no suficientes) para que exista fallo. Este tipo de deducción puede hacerse en cada nivel, pero la labor hasta llegar al SND puede resultar costosísima.

El problema de esta tesis está planteado además como que no se tiene más sensor que el que indica la presencia del SND y esto es una información escasa.

Se ha justificado sin embargo, cómo la propia estructura del árbol, da una información que permite llegar a ciertas conclusiones. Esta puede resumirse en:

- Todo S.B. que concurra a una puerta AND se transforma en un fallo que afecta a todos los C.M. provenientes de niveles más bajos.

- Todo S.B. que concurra a una puerta OR, pasa a ser C.M. por si solo, en esa puerta.

Si se pudiera proceder a una exploración de todos los S.B. por niveles, empezando por el más alto, se tendría una información progresivamente mayor, que dependería del tiempo disponible.

Yendo del nivel más alto hacia el más bajo, se puede deducir incluso el C.M. con más probabilidad de fallo en ese instante, en el caso de tener tiempo suficiente.

Se define entonces:

Conjunto virtual: Es el conjunto de S.B. que son condiciones necesarias, no suficientes, en cada nivel, para que se produzca un determinado modo de fallo.

4.7.2..- Condiciones que debe cumplir el árbol

* Debe estar depurado.

* La puerta del nivel 1 debe ser siempre OR. De no serlo, se colocará en el árbol depurado una puerta OR ficticia que tenga como entrada la AND y otra entrada de un SB imposible. En el algoritmo presentado en el anexo 2 queda reflejada esta reconversión del árbol.

4.7.3.- Método propuesto de deducción de los C.V.

Nivel 1: Los SB que entran en la puerta son CV del primer nivel y a la vez mínimos.

Nivel 2: Los SB que entran en las puertas son CV del segundo nivel en cada una de ellas. El conjunto de CV está formado por los CM del primer nivel y los CV del segundo.

Nivel 2N+1 (puertas OR): Los SB que entran en esas puertas forman CM en ellas, por lo que de cara a la construcción de los CV en ese nivel, se combinan sus SB con todos los CV procedentes de la puerta inmediata superior a la que concurren.

Nivel 2N (puertas AND): Los SB que entran en esas puertas, se añaden a cada uno de los CV del nivel inmediato superior, formando así los nuevos CV de la puerta en estudio.

4.7.4.- VENTAJAS E INCONVENIENTES

Ventajas

* La deducción de los CV en cada nivel, permite aportar información progresiva al operador, sobre SB que son condiciones necesarias para el fallo desde los niveles más altos del árbol y que por tanto afectan a grandes cantidades de CM del mismo.

* La verificación por parte del operador de los SB que forman los CV y por su característica de ser condiciones necesarias, permite que, partiendo de un pequeño número de CV, pueda eliminarse toda una rama del árbol en cuanto alguno de los SB incorporados no se encuentre en el estado indicado, facilitando así la búsqueda.

* Puesto que la construcción de los CV se hace desde el nivel más alto y por tanto más sencillo, el método permite, ya sea ir aportando información progresiva, ó bien, acotar los tiempos de respuesta y aportar el máximo de información estructural a que se pueda haber llegado en ese período.

* Si se dispone del tiempo suficiente pueden llegarse a construir los CM del sistema, que no son otra cosa que los CV del último nivel de cada rama. Por este motivo, pueden llegar a detectarse múltiples causas del fallo posibilitando el estudio de zonas débiles del sistema.

* Dado que la información es estructural y por tanto se encuentra "compilada" en la estructura. el operador puede solicitar la información con la rapidez que él desee.

Inconvenientes

* Si los tiempos de acotación en la respuesta son muy cortos y el árbol es muy complejo, la información aportada es escasa, creando incertidumbre en el operador ante la gran cantidad de CM que tienen como condiciones necesarias, los CV disponibles en ese momento.

4.8.-RESOLUCION DE INCERTIDUMBRE EN CONJUNTOS VIRTUALES

Paliar el inconveniente señalado en el apartado 4.7.4.- es el objetivo y no sólo de tipo estructural. En la formulación de una hipótesis de fallo, ya sea total (CM) ó parcial (CV) debe reflejarse el estado del sistema, su edad, sus políticas de mantenimiento preventivo, etc... Un solo parámetro basta para ello y es la fiabilidad ó no-fiabilidad de los componentes del mismo, en el instante en que se ha detectado el SND.

Para llegar a la formulación de una sola hipótesis de fallo (un solo CM) y que se encontrará al final de una determinada rama, habrá que haber desechado todas las demás ramas.

Puesto que un CV es necesariamente parte de un CM ó modo de fallo, parece lógico darle el mismo tratamiento a las no-fiabilidades de sus SB que a los de los CM y ya propuesto en el apartado 4.6.1.- para estos.

Todo nuevo SB que entre en un CV modificará la probabilidad de fallo del CV sólo si es más baja que la de todos los anteriores, puesto que un CM refleja una disposición de cara al fallo en paralelo en sentido de la fiabilidad.

Sin embargo, la búsqueda de la hipótesis total de fallo ó CM se vé dificultada en un árbol complejo por la gran cantidad de CV que se forman. Por las propias características de construcción de los CV se sabe que en un determinado nivel, de cada uno de ellos "cuelgan", ó se generan, un número determinado de CM, número que se conoce, al haber sido deducidos los CM según el método propuesto en el apartado 4.2.1.

El número de modos de fallo que dependen de un CV hace que el operador tienda a pensar que el causante del fallo estará en aquella rama de la cual "cuelguen" más mínimos. De alguna manera, esta actitud, refleja la tendencia a pensar que el fallo estará en aquella rama del árbol que genere más modos de fallo.

4.8.1.- METODO PROPUESTO

1) El tratamiento de los CM y CV en cada nivel será el mismo.

2) En cada nivel y puerta y comenzando por la de nivel 1, cada CV tendrá asociados dos parámetros. Uno "Q" será el resultado de aplicar el método propuesto en 4.6.1.- a los SB que formen cada CV.

El otro, que se denominará "P" ó "peso" será:

$$P = \frac{n_1}{n_e}$$

siendo: n_1 = número de CM que dependen de ese CV.

n_e = número total de CM que dependen de la puerta a la que concurra ese CV.

3) La elección de la hipótesis de CV será aquella que presente una cifra más alta del producto:

$$H = P \cdot Q$$

4) H será el parámetro característico de todo CV ó mínimo en cada nivel. Si en algún momento un CV que ya sea mínimo tiene un H menor que algún mínimo aparecido en niveles anteriores, se volverá al que la tenga menor como nueva hipótesis e fallo.

5) Si en algún momento el operador detecta que alguno de los SB de los sucesivos CV elegidos en base al parámetro H no se encuentra en estado activo de cara al fallo, se tomará al siguiente CV en orden al parámetro H en las puertas más cercanas a él y que se encuentren sobre él.

4.8.2.- VENTAJAS E INCONVENIENTES

Ventajas

* El método permite establecer porcentajes que ayuden a la decisión del operador de cara a ver la fuerza de la elección hecha por el sistema.
* Es una buena combinación de la resolución estructural de la incertidumbre, junto a la derivada de la fiabilidad de los componentes.
* Representa bien la incertidumbre provocada por la existencia de subsistemas muy complejos que originan muchos modos de fallo.

* Es de gran eficiencia y fácil visualización siendo muy compatible con los actuales sistemas de monitorización.

Inconvenientes

* Requiere una puesta al día muy estricta de los datos de fiabilidad.

4.9.- RESUMEN DE LO PROPUESTO

El epígrafe 4.2.1.- propuesto, consiste en un método de construcción de árboles de fallos que siendo integrable con el resto de las propuestas, afecta sin embargo, a una problemática diferente, como es la construcción más eficaz y más eficiente de los árboles de fallos. Asimismo se propone como un método de detección de inconsistencias del conocimiento de cara al fallo.

Los epígrafes 4.3 al 4.8 inclusive, proponen metodologías que resuelven problemas concretos existentes en la actualidad. Sin embargo, constituyen en sí una metodología única, con total capacidad de integración de cara a la resolución del problema planteado en esta tesis.

Para atacar el problema planteado en esta tesis, se ha aplicado la técnica heurística de propósito general, "Divide et impera", es decir, descomponer el problema global en subproblemas a los que se ha dado solución a cada uno por separado.

En el anexo 1 se resuelve un ejemplo concreto, aplicando el conjunto de metodologías propuestas. Asimismo se incluye el comentario a que se hizo referencia en el apartado H) del planteamiento del problema.

En el anexo 2 se presentan los procedimientos y programas que constituyen en sí una herramienta de aplicación de la metodología propuesta.

El conjunto de métodos propuestos, constituyen de forma global una metodología de construcción a partir de conocimiento diagnóstico no estructurado, en otro estructurado y jerarquizado y posteriores métodos de tratamiento específico de la estructura creada para la resolución del problema del diagnóstico, al que se añaden las condiciones de tratar sistemas complejos y bajo la necesidad de solución extremadamente eficiente, como aportación original de esta tesis.

CAPITULO QUINTO

RESULTADOS, CONCLUSIONES Y FUTURAS LINEAS DE INVESTIGACION

5.1- RESULTADOS

El conjunto de métodos propuestos en esta tesis ha permitido construir una herramienta, de aplicación de los mismos, donde pueden manejarse las reglas que se deducen del árbol de fallos de cara a la formulación de una hipótesis de fallo según el tiempo disponible.Los programas y procedimientos que la constituyen se encuentran en el anexo 2.

Esta herramienta que consta de 12 ficheros realiza las siguientes funciones:

1) Procedimientos de construcción y manejo de árboles de fallos.

Su función es la correcta introducción de los datos, es decir, de la estructura del árbol, el cual se supone ya construido según los procedimientos actuales. Mediante una serie de funciones de ayuda al usuario y otras de caracter interno se construye la estructura del árbol, detectando si lo que se ha introducido es un árbol ó le falta alguna propiedad para serlo.

Igualmente permite introducir nueva información en un árbol ya existente ó modificar lo que habia.

2) Funciones de manejo de la estructura ARBOL y los NODOS del mismo, especialmente adaptadas a las características de los árboles de fallo.

Mediante estas funciones se ha conseguido identificar los nodos del árbol construido, de forma que, la estructura se adapte a las características específicas de un árbol de fallos.

3) Funciones para facilitar la utilización de las primitivas de manejo de la estructura árbol

Las funciones contenidas en este fichero son complementarias de las del anterior. Dispone además de otras funciones de ayuda, que permiten una mejor comunicación con el usuario. El objetivo de todas, es dotar de una gran flexibilidad a la estructura ya construida, así como de gran facilidad de acceso a la información en ella contenida, ya sea de tipo descriptivo ó estructural.

4) Procedimientos de ayuda al usuario de tratamiento de la estructura.

5) Menús de ayuda y procedimientos orientados a la introducción de datos por pate del usuario.

6) Procedimientos relacionados con la depuración y poda de árboles de fallos.

El conjunto de las funciones de este fichero realiza la labor de depuración ya descrita. Además de ésta se realizan otras, que consisten en la eliminación ó poda de ciertas puertas y SB en base a la información propiamente estructural del árbol. Mediante estas podas se genera una nueva estructura de la que ya han desaparecido aquellas combinaciones de SB que suponen la aparición de CS que requieren la aplicación de las propiedades de monotonía e idempotencia.

7) Procedimientos relacionados con el cálculo de CM.

Mediante este procedimiento se calculan los CM desde niveles inferiores al superior, de tal forma que son calculados puerta a puerta, a la vez que calcula el número de CM asociados a cada puerta así como el total del árbol.

8) Operaciones sobre conjuntos representados en forma de lista.

Mediante estos procedimientos se generan y eliminan los CS que no son CM en base a la información de tipo estructural.

9) Procedimientos para la realización de un diagnóstico del fallo detectado.

En este procedimiento se realiza propiamente el diagnóstico en base a la resolución de incertidumbre mediante parámetros de fiabilidad aplicados a los CV de cada nivel y de forma progresiva. Se añaden además unas funciones de ayuda al usuario que le permiten introducir información, así como extraerla, ya sea del árbol completo ó de un nodo aislado. El procedimiento permite al usuario:

* Solicitar información progresiva a medida que sea requerida.
* Esperar la salida progresiva de información.
* Acotar los tiempos de salida de información.

10) Procedimientos de manejo de los componentes del sistema.

Son una serie de ayudas que permiten la modificación de los datos, manejo de los tiempos de acotación,etc...

11) Procedimientos orientados a la realización de una simulación de funcionamiento del sistema para un caso real.

12) Primitivas básicas y funciones de manejo de la estructura utilizada para recordar la situación actual de suposiciones hechas por el sistema al realizar un diagnóstico.

5.2.- CONCLUSIONES

Los resultados obtenidos se adaptan a las necesidades expuestas en el planteamiento del problema y pueden resumirse en :

1) Resulta ser un procedimiento muy elástico y adaptable al usuario y a casos concretos muy diferentes, para la introducción y modificación de datos así como construcción de la estructura de manejo del árbol de fallos que permite una gran eficiencia en el proceso del diagnóstico.

2) El procedimiento de depuración del árbol simplifica enormemente la estructura para el cálculo posterior de CM.

3) El procedimiento de cálculo de los CM en la estructura ya depurada permite conocer los CM asociados a cada puerta del árbol.

4) El procedimiento de eliminación de incertidumbre ya sea de tipo estructural como basado en los datos de fiabilidad, permite reflejar el estado del sistema en cada momento.

5) Asímismo se tiene un procedimiento basado en la deducción de CV que, ya sea estructuralmente ó bien según los parámetros de fiabilidad, permite:

- Obtención progresiva de información ó formulación incompleta de hipótesis de fallo más probable.
- De disponer de tiempo suficiente, se consiga formular una hipótesis completa de fallo.
- Según las necesidades del usuario es posible acotar los tiempos de salida de la información, dando el máximo de información dentro de las cotas de tiempo señaladas.

Se puede deducir en conclusión que los procedimientos expuestos permiten mediante la construcción, estructuración, representación y depuración del conocimiento, primeramente, "compilar" el conocimiento en un árbol de fallos y reconocerlo como tal, de forma que es detectable el conocimiento inconsistente, ó bien la determinación y acotación de lagunas de conocimiento.

Esta estructura creada, consigue simplificar los sistemas complejos, permitiendo asimismo integrar conocimientos de equipos humanos muy diferenciados y que sin duda, son la representación cualitativa de un sistema complejo. Esta estructura permite igualmente y tras las labores de depuración una gran rapidez en las tareas del diagnóstico al tener jerarquizados los modos de fallo del sistema. Esta jerarquización es la que permite la aportación progresiva de la información en la tarea del diagnóstico, así como la detección de zonas débiles del sistema. De esta forma se aporta una información dinámica y progresivamente mayor con el paso del tiempo frente a la estática de tipo estructural, con lo que el sistema se adapta bien al caso concreto a tratar.

Se ha preferido no evaluar cuantitativamente la cantidad de información aportada según el tiempo transcurrido, dado que la información es de tipo cualitativo. Sin embargo, la estructura creada permitiría una evaluación de este tipo. Dado que la aportación de información es progresiva, una conclusión clara es que permite estimar el "hardware" óptimo para cada caso concreto.

Finalmente decir que, la elasticidad del sistema permite la contrastación de datos por el usuario y de no verificarse, proponer un nuevo modo de fallo, compatible con la información aportada por éste. En resumen, la herramienta resuelve el problema de la complejidad mediante la estructuración. El del diagnóstico mediante deducción estructural y parámetros de fiabilidad. El de la constricción temporal, en base a la creación y manejo del concepto de CV como estructura.

5.3.- FUTURAS LINEAS DE INVESTIGACION.

Se resumen a continuación las lineas abiertas de investigación a partir de los tres conceptos básicos que conforman esta tesis. Algunas han sido ya nombradas en el Estado de la Cuestión. Otras no. Globalmente, este tema, sigue siendo difícil de tratar y tiene por tanto muchas posibles vias de solución. Se destacan entonces las que se estiman como principales.

1) Queda por resolver, la definición de una clara función de aprendizaje conceptual en estructuras, siendo una de ellas el árbol de fallos. La identificación de estructuras y aplicación de unas a otras en temáticas diferentes integraría de alguna forma el más alto nivel del conocimiento diagnóstico.

2) Como ya se hizo mención, se estaba en la actualidad investigando en la clasificación de los métodos de resolución de incertidumbre. Se echa en falta sin embargo, una metodología de clasificación de los tipos de problemas de cara a encontrar el método de resolución de incertidumbre adecuado al problema.

3) Sigue estando sin resolver el problema planteado en esta tesis ni tan siquiera en su vertiente estructural, en cuanto al tratamiento del árbol de fallos con componentes de estados intermedios (no bivaluados).

4) Otra línea muy solicitada por los investigadores es la resolución del problema del diagnóstico, con fallos de causa común.

5) Consecuencia directa de esta tesis, es la construcción de un sistema de resolución de incertidumbre que parta de las cotas inferior y superior de ocurrencia de fallos, como sistema que reproduzca las condiciones extremas de dependencia total (causa común) e independencia total, de los sucesos acaecidos.

6) Otra línea abierta es aquella que mediante el estudio de la estructura pueda detectar inconsistencias en las lecturas de sensores y deducción del sensor fallado, con el objetivo de proporcionar una solución determinística.

7) Por lo ya comentado en esta tesis, queda aún por intentar la construcción de una estructura por niveles que permita resolver el diagnóstico médico de forma más eficiente.

8) Sigue abierta la línea de propuestas que a partir de la diagnosis industrial permita llegar a una formulación y metodologías, que hagan más barato y eficaz el mantenimiento preventivo.

9) Como parte del estudio de los estados no-bivaluados, se encuentra aún en fase de investigación. la diagnosis de los fallos secundarios.

10) Dado que el tratamiento actual de los árboles de fallo parte de la condición de monotonía en los estados de los componentes que forman un CM, queda aún por definir una metodología que aúne la condición de no-monotonía en un solo árbol.

B I B L I O G R A F I A

B I B L I O G R A F I A R E F E R E N C I A D A

[ABBO-85]
M.B.ABBOT; H.J. DE NORWALL; B.TOLLEY;
"THE FEASILILITY OF USING SIMPLE INTELLIGENT SYSTEMS AS AIDS TO PLANT OPERATORS".
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR POWER PLANT OPERATION AND CONTROL. 1985

[AOYA-87]
K.AOYAGI;I.SANO;
"A SCHEDULING SUPPORT SYSTEM BASED ON FUZZY INFERENCE FOR STARTUP OF FOSSIL POWER PLANTS".
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[APOS-78]
G.E.APOSTOLAKIS;S.L.SALEM;J.S.WU;
"CAT: A COMPUTER CODE FOR THE AUTOMATED CONSTRUCTION OF FAULT TREES".
EPRI-NP-705. 1978

[APOS-79-A]
G.APOSTOLAKIS;
"ASSESMENT OF THE FRECUENCY OF FAILURE TO SERAM IN LIGHT WATER REACTORS".
NUCLEAR SAFETY 30(6). 1979

[APOS-79-B]
G.APOSTOLAKIS; A.MOSLEH;
"EXPERT OPINION AND STATISTICAL EVIDENCE.-AN APPLICATION TO REACTOR CO
MELT FREQUENCY"
NUCLEAR SCI.ENG.70. 1979

[BASD-84]
A.BASDEN;
"EXPERTS SYSTEMS APPLICATIONS".
ACADEMIC PRESS-M.J.COOMBS. 1984

[BAZO-61]
I.BAZOVSKY;
"RELIABILITY THEORY AND PRACTICE".
PRENTICE-HALL INC. ENGLEWOOD CLIFFS,N.J. 1961

[BERG-87]
O.BERG;R.E.GRINI;M.YOKOBAYASHI;
"EARLY FAULT DETECTION AND DIAGNOSIS FOR NUCLEAR POWER PLANTS".
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[BLEL-86]
G.E.BLELLOCH;
"CIS:A MASSIVELY CONCURRENT RULE-BASED SYSTEM".
PROCEEDINGS OF FITH NATIONAL CONFERENCE OF A.I. 1986

[BYLA-83]
T.BYLANDER;S.MITTAL;B.CHANDRASERAKAN;
"CSRL: A LANGUAGE FOR EXPERT SYSTEMS FOR DIAGNOSIS".
WILLIAM KAUFMAN INC.LOS ALTOS C.A. 1983

[CALD-79]
L.CALDAROLA;
"FAULT TREE ANALYSIS WITH MULTISTATE COMPONENTS"
KfK 2761;EUR 5756e; 1979

[CALD-80]
L.CALDAROLA;
"GENERALIZED FAULT TREE ANALYSIS COMBINED WITH STATE ANALYSIS".
KfK 2530.EUR 5754e. 1980

[CAMP-85]
D.J.CAMPBELL; B.C.ELLISON; J.C.GLYNN; G.F.FLANAGAN;
"PLAN RISK STATUS INFORMATION MANAGEMENT SYSTEM".
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR
POWER PLANT OPERATION AND CONTROL. 1985

[CARB-83]
J.G.CARBONELL;R.S.MICHALSKY;T.M.MITCHELL;
"MACHINE LEARNING.AN ARTIFICIAL INTELLIGENCE APROACH".
BERLIN.SPRINGER-VERLAG. 1983

[CARB-87]
J.G.CARBONELL;
ESTADO ACTUAL DEL DESARROLLO DEL S.E. "TEST".
CURSO MASTER DE INGENIERIA DEL CONOCIMIENTO.FACULTAD DE INFORMATICA
DE LA UNIVERSIDAD POLITECNICA DE MADRID. 1987

[CAST-86]
L.CASTILLO; J.GONZALEZ-SUSTAETA; V.HUERTA; E.LIÑAN;
"A RULE TO NETWORK BASED EXPERT SYSTEM SHELL FOR POWER SYSTEM KNOWLEDGE
ENGINEERING".
INSTITUTO DE INVESTIGACIONES ELECTRICAS.MEXICO. 1986

[CHAN-83]
B.CHANDRASERAKAN;S.MITTAL;
"CONCEPTUAL REPRESENTATION OF MEDICAL KNOWLEDGE FOR DIAGNOSIS BY
COMPUTER: MDX AND RELATED SYSTEMS".
ED. M. YOVITS;ADVANCES IN COMPUTER. VOL 22; ACADEMIC PRESS. 1983

[CHAN-84-A]
B.CHANDRASEKARAN; S.MITTAL;
"DEEP VERSUS COMPILED KNOWLEDGE APPROACHES TO DIAGNOSTIC PROBLEM
SOLVING".
ACADEMIC PRESS-M.J.COOMBS. 1984

[CHAN-84-B]
B.CHANDRASEKARAN; R.DAVIS;
"REASONING FROM FIRST PRINCIPLES IN ELECTRONIC TROUBLESHOOTING".
ACADEMIC PRESS-M.J.COOMBS. 1984

[CHAN-85]
B.CHANDRASEKARAN;D.W.MILLER;
"AN A.I. APPROACH TO SENSOR CONFLICT DETECTION".
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR
POWER PLANT OPERATION AND CONTROL. 1985

[CHRI-85]
A.M.CHRISTIE; P.B.DEEGAN;R.E.PROVINCE; H.L.SANTOS;
"AN EXPERT SYSTEM FOR REAL-TIME DIAGNOSTICS AND CONTROL".
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR
POWER PLANT OPERATION AND CONTROL. 1985

[CHU-80]
T.L.CHU;G.APOSTOLAKIS;
"METHODS FOR PROBABILISTIC ANALYSIS OF NON-COHERENT FAULT TREES".
IEEE TRANSACTIONS ON RELIABILITY. VOL. R-29, N.5 1980

[CLEL-87]
R.S.CLELAND;
"A BOILER TUBE FAILURE ANALYSIS EXPERT SYSTEM".
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[CUES-86]
M.CUESTA
"NUEVE PARAMETROS PARA EVALUAR EL MANTENIMIENTO PREVENTIVO DE
COMPONENTES A POTENCIAL FIJO"
ASOCIACION DE INGENIEROS AERONAUTICOS.REVISTA DE INGENIERIA
AERONAUTICA Y ASTRONAUTICA.NUMERO: 278-279 1986

[DEMP-68]
A.DEMPSTER;
"A GENERALIZATION OF BAYESIAN INFERENCE".
JOURNAL OF ROY. STATIS. SCI. SER. B-30. 1968

[DIXO-85]
B.W.DIXON; M.F.HINTON;
"EVENT TREE ANALYSIS USING A.I. TECHNIQUES".
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR
POWER PLANT OPERATION AND CONTROL. 1985

[EPRI-85-A]
COMBUSTION ENGINEERING INC.
"IMPROVED ON-LINE OPERATIONAL SUPPORT USING SUCESS PATH MONITORING"
EPRI 1985

[EPRI-85-B]
EPRI
"FUNCTIONAL SPECIFICATIONS FOR AI SOFTWARE TOOLS FOR ELECTRIC POWER
APLICATIONS"
EPRI 1985

[FAUS-86]
A.FAUS;
"FILOSOFIA DEL MANTENIMIENTO AERONAUTICO"
ASOCIACION DE INGENIEROS AERONAUTICOS.REVISTA DE INGENIERIA
AERONAUTICA Y ASTRONAUTICA.NUMERO:278-279 1986

[FUSS-72]
J.B.FUSSEL;W.E.VESELY;
"A NEW METHODOLOGY FOR OBTAINING CUT SETS FOR FAULT TREES".
TRANS.ANS;VOL 15,262. 1972

[GABR-85]
J.R.GABRIEL;
"FACTORS AFFECTING MANUFACTURING COST OF COMPLEX SYSTEMS AND THE ROLE OI
NEW TECHNOLOGIES IN REDUCING THESE COSTS".
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR
POWER PLANT OPERATION AND CONTROL. 1985

[GORD-85]
J.GORDON;E.SHORTLIFFE;
"A METHOD FOR MANAGING EVIDENTIAL REASONING IN A HIERARCHICAL HYPOTHESIS
SPACE".
ARTIFICIAL INTELLIGEN, 26. 1985

[GRAH-82]
N.GRAHAM;
"INTRODUCTION TO COMPUTER SCIENCE"
WEST PUBLUSHING CO. 1982

[GUAR-85]
S.B.GUARRO; D.OKRENT;
"USE OF LOGIC FLOWGRAPH MODELS IN A COMPUTER AIDED PROCESS ANALYSIS AND
MANAGEMENT SYSTEM".
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR
POWER PLANT OPERATION AND CONTROL. 1985

[HADD-86]
P. HADDAWAY;
"IMPLEMENTATION OF AN EXPERIMENTS WITH A VARIABLE PRECISION LOGIC INFERENCE SYSTEM".
PROCEEDINGS OF FITH NATIONAL CONFERENCE OF A.I. 1986

[HAJE-87]
B.J.HAJEK;S.HASHEMI;R.BHATNAGER;D.W.MILLER;J.STASENKO;
"EXPERT SYSTEM APPLICATION TO FAULT DIAGNOSIS AND PROCEDURE SYNTHESIS".
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[HASH-87]
S.HASHEMI;W.F.PUNCH III; B.J.HAJEK;
"CSRL APPLICATION TO NUCLEAR POWER PLANT DIAGNOSIS AND SENSOR DATA VALIDATION".
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[HAUP-86]
U.HAUPTMANS;
"ANALISIS DE ARBOLES DE FALLOS"
EDITORIAL BELLATERRA.BARCELONA. 1986

[HECK-86]
D.E. HECKERMAN; E.J.HORVITZ; C.P.LANGLOTZ;
"A FRAMEWORK FOR COMPARING ALTERNATIVE FORMALISMS FOR PLAUSIBLE REASONING"
PROCEEDINGS OF FITH NATIONAL CONFERENCE OF A.I. 1986

[HICK-85]
D.H.HICKMAN;E,H,SHORTLIFFE;M.B.BISCHOFF;A.C.SCOTT;C.D.JACOBS;
"A STUDY OF THE TREATMENT ADVICE OF A COMPUTER-BASED CANCER CHEMOTHERAF PROTOCOL ADVISOR".
ANNALS OF INTERNAL MEDICINE,103. 1985

[ILLA-86]
J. ILLANA;
"EL MANTENIMIENTO EN LA AVIACION MILITAR:PERSPECTIVA ACTUAL"
ASOCIACION DE INGENIEROS AERONAUTICOS.REVISTA DE INGENIERIA AERONAUTICA Y ASTRONAUTICA.NUMERO: 278-279 1986

[KESS-87]
T.KESSLER;G.LIU;B.FROGNER;D.GLOSKI;R.LUCIER;
"INSPECTR:AN INTERACTIVE POWER PLANT PERFORMANCE DIAGNOSTIC ASSISTANT"
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[KINO-85]
M.KINOSHITA; K.FUKUNISHI; N.YAMADA;J.TAUJI;
"AN EVEN-DRIVEN CONTROL METHOD FOR AUTOMATIC OPERATION AND CONTROL OF
NUCLEAR POWER PLANTS".
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR
POWER PLANT OPERATION AND CONTROL. 1985

[KISN-85]
R.A.KISNER;
"AUTOMATING LARGE-SCALE REACTOR SYSTEMS".
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR
POWER PLANT OPERATION AND CONTROL. 1985

[KITA-87]
Y.KITAMURA;M.MIZUNO;T.KOJIMA;
"AN EXPERT SYSTEM FOR SUPPORTING COAL FIRED PLANT OPERATION".
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[KLEE-86]
J. DE KLEER; B.C.WILLIAMS;
"REASONING ABOUT MULTIPLE FAULTS"
PROCEEDINGS OF FITH NATIONAL CONFERENCE OF A.I. 1986

[KOCH-87]
C.G.KOCH;B.A.ISLE;A.W.BULLER;
"INTELLIGENCE USER INTERFACE FOR EXPERT SYSTEMS APPLIED TO GAS TURBINE
MAINTENANCE AND TROUBLESHOOTING".
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[KOEN-74]
B.V.KOEN;A.CARNINO;
"RELIABILITY CALCULATION WITH LIST PROCESSING TECHNIQUE".
IEEE TRANSACTIONS ON RELIABILITY;VOL.R-23;N.1; 1974

[KOIK-86]
N.NOIKE; T.MAESHIRO; T.GOTOH; M.KUNUGI; T.HIROKAWA; N.WADA;
"A REAL-TIME EXPERT SYSTEM FOR POWER SYSTEM FAULT ANALYSIS"
ELECTRIC POWER RESEARCH INSTITUTE 1986

[KOLO-84]
J.L.KOLODNER;
"HACIA UNA COMPRENSION DEL PAPEL DE LA EXPERIENCIA EN LA EVOLUCION Y TRANSFORMACION DE UN NOVEL EN UN EXPERTO".
ACADEMIC PRESS-M.J.COOMBS. 1984

[KUJA-85]
E.KUJAWSKI; I.M.JACOBS; A.M.SMITH; J.E.MOTT; B.K.H.SUN;
"SIGNAL VALIDATION SOFTWARE UTILIZING A GENERALIZED DECISION ESTIMATOR FOR COMMON CAUSE FAILURES".
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR POWER PLANT OPERATION AND CONTROL. 1985

[LIDS-87]
L.M. LIDSKY;
"EXPERT SYSTEM TOOLS".
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[MARI-86]
P. MARION;
"LA SEGURIDAD EN AVIACION CIVIL.LA IMPORTANCIA DEL FACTOR HUMANO"
ASOCIACION DE INGENIEROS AERONAUTICOS.REVISTA DE INGENIERIA AERONAUTICA Y ASTRONAUTICA.NUMERO: 283 1986

[MART-82]
H.F.MARTZ; R.A.WALLER;
"BAYESIAN RELIABILITY ANALYSIS".
NUEVA YORK.CHICHESTER.BRISBANE.TORONTO.SINGAPUR. 1982

[MELL-84]
W.V.MELLE; B.G.BUCHANAN; E.H.SHORTLIFFE;
"RULE BASED EXPERT SYSTEMS;THE MYCIN'S EXPERIMENTS OF THE STANFORD HEURISTIC PROGRAMMING PROJECT"
ADDISON WESLEY 1984

[MICH-85]
D.MICHIE;
"TAMMING ENGINEERING COMPLEXITY:HOW A.I. FITS IN".
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR
POWER PLANT OPERATION AND CONTROL. 1985

[MICH-86]
R.S.MICHALSKI;P.H.WINSTON;
"VARIABLE PRECISION LOGIC".
ARTIFICIAL INTELLIGENCE NO.29. NORTH HOLLAND. 1986

[MOOR-85]
R.L.MOORE; L.B.HAWKINSON;C.G.KNICKERBOCKER; M.E.LEVIN;
"EXPERT SYSTEMS IN REAL-TIME ENVIRONMENTS".
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR
POWER PLANT OPERATION AND CONTROL. 1985

[MORR-86]
P.H.MORRIS; R.A.NADO;
"REPRESENTING ACTIONS WITH AN ASSUMPTION BASED THRUTH MAINTENANCE SYSTE
(ATMS)"
PROCEEDINGS OF FITH NATIONAL CONFERENCE OF A.I. 1986

[NAU-84]
D.NAU;J.REGGIA;
"RELATIONSHIP BETWEEN DEDUCTUVE AND ABDUCTIVE INFERENCE IN KNOWLEDGE -
BASED DIAGNOSTIC PROBLEM SOLVING".
PROC.FIRST INTEL. WORKSHOP ON EXPERT DATABASE SYSTEMS;KERSCHBERGE L.
KIAWA ISLAND SC. 1984

[NEUS-87]
C.H.NEUSCHAEFER; P.W.RZASA; E.FILSHTEIN; R.L.BURRINGTON; R.DONAIS;
"APPLICATION OF C-E'S GENERIC DIAGNOSTIC SYSTEM TO POWER PLANT
DIAGNOSTICS".
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[NILS-81]
N.J.NILSSON;
"PROBLEM SOLVING METHODS IN A.I."
MC. GRAW-HILL BOOK CO. 1981

[OKAM-85]
M.OKAMOTO; N.OGURA; N.KATO; K.MUTO; K.KIMERA; M.TANI; Y.MASUDA;
K.TAKAHASHI;
"THE DEVELOPMENT AND EVALUATION OF PRESSURIZED WATER REACTOR ADVANCED
CONTROL ROOM CONCEPTS IN JAPAN".
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR
POWER PLANT OPERATION AND CONTROL. 1985

[PARG-87]
P.PARGE; J.STUART; J.VINSON;
"THE DEVELOPMENT AND APPLICATION OF TURBOMAC;AN EXPERT MACHINERY
DIAGNOSTIC SYSTEM".
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[PARR-81]
G.W.PARRY; P.W.WINTER;
"CARACTERIZATION AND EVALUATION OF UNCERTAINTY IN PROBABILISTIC RISK
ANALYSIS".
NUCLEAR SAFETY.VOL.22 NUMERO 1. 1981

[PAZO-80]
J. PAZOS SIERRA
"TECNICAS DE OPTIMIZACION DE SISTEMAS"
ED. CREI 1980

[PAZO-87]
J.PAZOS SIERRA
"INTELIGENCIA ARTIFICIAL"
ED. PARANINFO 1987

[PAZZ-86]
M.J.PAZZANI;
"REFINING THE KNOWLEDGE BASE OF A DIAGNOSTIC EXPERT SYSTEM: AN
APPLICATION OF FAILURE-DRIVEN LEARNING"
PROCEEDINGS OF FIFTH NATIONAL CONFERENCE OF A.I. 1986

[PEAR-86]
J.PEARL;
"ON EVIDENTIAL REASONING IN A HIERRARCHY OF HIPOTHESIS".
ARTICIAL INTELLIGENCE JOURNAL, 26:1. 1986

[PENG-86]
Y.PENG;
"A FORMALITATION OF PARSIMONIUS COVERING AND PROBABILISTIC REASONING IN
ABDUCTUVE DIAGNOSTIC INFERENCE"
TECHNICAL REPORT TR-1615(PH.D.DISSERTATION. dEP. OF COMPUTER SCIENCE.
UNIVERSITY OF MARYLAND. 1986

[PENG-86-A]
Y. PEMG;
"A FORMALIZATION OF PARSIMONIUS COVERING AND PROBABILISTIC REASONING IN
ABDUCTIVE DIAGNOSTIC INFERENCE".
TECHNICAL REPORT TR-1615 (TESIS DOCTORAL) DEP. OF COMPUTER
SCIENCE.UNIVERSITY OF MARYLAND. 1986

[PENG-86-B]
Y.PENG;J.REGGIA;
"A PROBABILISTIC CAUSAL MODEL FOR DIAGNOSTIC PROBLEM SOLVING".
SUBMITTED FOR PUBLICATION. 1986

[PERE-86]
E.PEREZ JIMENEZ;
"LA INSPECCION DEL ESTADO EN MATERIA DE AERONAVEGABILIDAD"
ASOCIACION DE INGENIEROS AERONAUTICOS.REVISTA DE INGENIERIA
AERONAUTICA Y ASTRONAUTICA.NUMERO: 278-279 1986

[POPL-73]
H.POPLE;
"ON THE MECHANIZATION OF ABDUCTIVE LOGIC".
PROC. OF INTERNATIONAL JOINT CONFERENCE ON A.I." IJCAI. 1973

[RAIM-86]
O.RAIMAN;
"ORDER OF MAGNITUDE REASONING"
PROCEEDINGS OF FITH NATIONAL CONFERENCE OF A.I. 1986

[RATN-86]
D.RATNER; I.POHL;
"JOINT AND LPA:COMBINATION OF APROXIMATION AND SEARCH"
PROCEEDINGS OF FITH NATIONAL CONFERENCE OF A.I. 1986

[REGG-81]
J.REGGIA;
"KNOWLEDGE-BASED DECISON SUPORT SYSTEMS:DEVELOPMENTS TROUGH KMS".
TECHNICAL REPORT TR-1121 (TESIS DOCTORAL). DEPARTEMENT OF COMPUTER SCIENCE. UNIVERSITY OF MARYLAND. 1981

[REGG-84]
J.A.REGGIA; D.S.NAU; P.Y.WANG;
"DIAGNOSIS EXPERT SYSTEMS BASED ON SET COVERING MODEL".
ACADEMIC PRESS-M.J.COOMBS. 1984

[REGG-84]
J.REGGIA;D.NAU;
"AN ABDUCTIVE NON MONOTONIC LOGIC".
PROC. WORKSHOP OF NON MONOTONIC REASONING. AAAI. 1984

[REGG-85]
J.REGGIA;D.NAU;P.WANG;Y.PENG;
"A FORMAL MODEL OF DIAGNOSTIC INFERENCE".
INFORMATION SCIENCES, 37 1985

[REGG-86]
J.A.REGGIA; Y.PENG;
"PLAUSIBILITY OF DIAGNOSIS HYPOTHESES"
PROCEEDINGS OF FITH NATIONAL CONFERENCE OF A.I. 1986

[REIT-85]
R. REITER;
"A THEORY OF DIAGNOSIS FROM FIRST PRINCIPLES"
TR-187/86.DEP. OF COMPUTER SCIENCE. UNIVERSITY OF TORONTO. 1985

[RIES-81]
C.K.RIESBECK;
"FAILURE DRIVEN REMINDING FOR INCREMENTAL LEARNING."
PROCEEDINGS OF INTERNATIONAL JOINT CONFERENCE OF A.I. 1981

[RIES-84]
C.K.RIESBECK;
"KNOWLEDGE REORGANIZATION AND STILE OF REASONING".
ACADEMIC PRESS-M.J.COOMBS. 1984

[RIOS-67]
S.RIOS;
"METODOS ESTADISTICOS"
ED. CASTILLO 1967

[RODR-86]
G.RODRIGUEZ; G.MARTINEZ; P.RIVERA;
"FAULT TREE ANALYSIS APPROACH TO BUILD KNOWLEDGE BASES FOR ENGINEERING APPLICATIONS"
INSTITUTE FOR ELECTRICAL RESEARCH.MEXICO 1986

[ROMA-87]
H.T.ROMAN; F.A.MARIAN; R.J.MEARS;
"EXPERT SYSTEMS APPLICATIONS AT PSE&G - A TASK FORCE APPROACH".
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[SCHW-84]
M.SCHWBARBLAT;J.ARELLANO;G.MARTINEZ;
"FAULT TREE/EVENT SEQUENCE METHODOLOGY AND DATA BASE MANAGEMENT SYSTEMS AS A TOOL FOR THE SAFE OPERATION OF COMPLEX SYSTEMS".
INTERNATIONAL CONFERENCE ON POWER PLANT SIMULATION.CUERNAVACA.MEXICO.

[SHAF-76]
G.SHAFER;
"A MATHEMATICAL THEORY OF EVIDENCE".
PRINCETON UNIVERSITY PRESS. 1976

[SHAF-86]
G.SHAFER;
"PROBABILITY JUDGEMENT IN ARTIFICIAL INTELLIGENCE IN UNCERTAINTY IN A.I.".
NORTH-HOLLAND. 1986

[SHAN-82]
R.C.SHANK;
"DINAMIC MEMORY:A THEORY OF LEARNING IN PEOPLE AND COMPUTERS".
CAMBRIDGE:CAMBRIDGE UNIVERSITY PRESS. 1982

[SHAO-86]
G.SHAO; HUNG LIU;
"CAUSAL AND PLAUSIBLE REASONING IN EXPERT SYSTEMS"
PROCEEDINGS OF FITH NATIONAL CONFERENCE OF A.I. 1986

[SHOR-76]
E.F.SHORTLIFFE;
"COMPUTER-BASED MEDICAL CONSULTATIONS:MYCIN".
NEW YORK:AMERICAN ELSEVIER. 1976

[SHOR-86]
E.H.SHORTLIFFE; C.P.LANGLOTZ; L.M.FAYAN;
"USING DECISION THEORY TO JUSTIFY HEURISTICS"
PROCEEDINGS OF FITH NATIONAL CONFERENCE OF A.I. 1986

[SHUB-82]
H.SHUBIN;J.ULRICH;
"IDT: AN INTELLIGENT DIAGNOSTIC TOOL"
PROC. NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE.AAAI. 1982

[SIMO-86]
B.A.SIMON FRENCH;
"DECISION THEORY: AN INTRODUCTION TO THE MATHEMATICS OF RATIONALITY".
ELLIS HORWOOD LTD. ENGLAND. 1986

[SNOW-86]
P.SNOW;
"BAYESIAN INFERENCE WITHOUT POINT ESTIMATES"
PROCEEDINGS OF FITH NATIONAL CONFERENCE OF A.I. 1986

[SPIE-69]
M.R.SPIEGEL
"ESTADISTICA"
MC. GRAW-HILL 1969

[SPUR-85]
A.J.SPURGIN; R.L.BEVERIDGE;
"LESSONS FROM OPERATOR RESPONSES TO ACCIDENTS".
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR POWER PLANT OPERATION AND CONTROL. 1985

[SWAI-83]
A.D.SWAIN;H.E.GUTTMANN;
"HANDBOOK ON HUMAN RELIABILITY ANALYSIS WITH EMPHASIS ON NUCLEAR POWER PLANT APPLICATIONS.FINAL REPORT."
US-NUCLEAR REGULATORY COMMISION NUREG/CR-1278.WASHINGTON. 1983

[TANA-85]
M.TANAKA; T.OGINO; T.MIKI; M.FUJII;
"ADVANCED OPERATOR DECISION AIDS FOR NUCLEAR POWER PLANTS".
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR POWER PLANT OPERATION AND CONTROL. 1985

[VESE-81]
W.E.VESELY ET AL.
"FAULT TREE HANDBOOK".
NUREG-0492 1981

[WAKA-85]
J.WAKABAYASHI; A.GOFUKU; F.OKAZAKI; S.TASHIMA;
"APPLICATION OF PROJECTIVE OPERATOR TECHNIQUE FOR THE DISTURBANCE IDENTIFICATION OF NUCLEAR POWER PLANTS".
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR POWER PLANT OPERATION AND CONTROL. 1985

[WARL-73]
J. WARLETTA
"FIABILIDAD"
INTA 1973

[WEIS-84]
C.R.WEISBIN;G. de SAUSURRE;J.BARHEM;E.M.OBLOW;J.C.WHITE;
"MINIMAL CUT-SET METHODOLOGY FOR ARTIFICIAL INTELLIGENCE APPLICATIONS"
FIRST CONFERENCE ON A.I. APPLICATIONS.DENVER. 1984

[WILL-86]
B.C.WILLIAMS;
"DOING-TIME:PUTTING QUALITATIVE REASONING ON FIRMER GROUND"
PROCEEDINGS OF FITH NATIONAL CONFERENCE OF A.I. 1986

[XERO-86]
"JORNADAS RANK XEROX SOBRE SISTEMAS EXPERTOS"
XEROX.MADRID 1986

[YAKO-77]
S.J.YAKOWITZ
"COMPUTATIONAL PROBABILITY AND SIMULATION"
1977

[YEN-86]
J.YEN;
"A REASONING MODEL BASED ON AN EXTENDED DEMPSTER-SHAFER THEORY"
PROCEEDINGS OF FITH NATIONAL CONFERENCE OF A.I. 1986

[YOSH-85]
k.YOSHIDA; M.YOKOBAYASHI; T.AOYAGI; Y.SHINOHARA; A.KOHSAKA;
"DEVELOPMENT AND VERIFICATION OF AN ACCIDENT DIAGNOSTIC SYSTEM FOR
NUCLEAR POWER PLANT BY USING A SIMULATOR".
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR
POWER PLANT OPERATION AND CONTROL. 1985

[ZADE-79]
L.A.ZADEH;
"A THEORY OF APROXIMATE REASONING".
MACHINE INTELLIGENCE.VOL.9.PAG.149-194. 1979

[ZIPF-82]
G.A.ZIPF ET AL.
"A COMPARISON OF SELECTED METHODS TO EVALUATE RELIABILITY CHARACTERISTICS OF LARGE TECHNICAL SYSTEMS"
SEMINAR ON "RELIABILITY ANALISIS OF COMPLEX SYSTEMS" 1982

OTRA BIBLIOGRAFIA CONSULTADA

[ALTY-86]
J.L.ALTY;M.J.COOMBS;
"SISTEMAS EXPERTOS"
DIAZ DE SANTOS.MADRID. 1986

[ANTI-87]
R.C.ANTINOJA;S.B.MC MILLAN;T.C.WIESNER;C.J.COOK;
"WELDER:AN EXPERT SYSTEM WELDING ENGINEER'S AIDE"
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[AOKI-87]
S.AOKI;K.KAWAI;H.MASUYAMA;H.TAKAOKA;
"AI-AIDED OPERATION GUIDANCE SYSTEM IN THERMAL POWER STATION".
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[BERK-85]
A.A.BERK;
"LISP:EL LENGUAJE DE LA I.A."
ANAYA 1985

[BERN-85]
J.A.BERNAD;D.D.LENNING;
"EXPERIMENTAL EVALUATION OF THE REACTIVITY CONSTRAINT APPROACH FOR THE
CLOSED-LOOP CONTROL OF REACTOR POWER OVER RANGE OF DIFFERENTIAL REACTI'
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR
POWER PLANT OPERATION AND CONTROL. 1985

[BERT-87]
W.BERTCH;R.BRANDOM;M.HUNT;
"EXPERT SYSTEM FOR TURBINE THERMAL PERFORMANCE DIAGNOSTICS."
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[BRAT-86]
I.BRATKO;
PROLOG:PROGRAMMING FOR A.I."
INTERNATIONAL COMPUTER SCIENCE SERIES 1986

[BURG-87]
V.BURGUETE;
"LA INFORMATICA EN LA GESTION DEL MANTENIMIENTO AERONAUTICO:DE LOS COMIENZOS A LA INTELIGENCIA ARTIFICIAL"
ASOCIACION DE INGENIEROS AERONAUTICOS.REVISTA DE INGENIERIA AERONAUTICA Y ASTRONAUTICA.NUMERO: 290-291 1987

[CAIN-87]
D.G.CAIN;
"SELECTING EXPERT SYSTEMS APPLICATIONS"
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[CARL-87]
J.CARLSON;M.COFFMAN;
"TU ELECTRIC EXPERIENCE WITH ON-LINE GENERATOR MONITORING AND DIAGNOSTICS"
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[COOM-84]
M.J.COOMBS
"DEVELOPMENTS IN EXPERT SYSTEMS"
ACADEMIC PRESS-M.J.COOMBS 1984

[CUEN-85]
J.CUENA;
"LOGICA INFORMATICA"
ALIANZA EDITORIAL 1985

[CUEN-86]
J.CUENA;
"INTELIGENCIA ARTIFICIAL.SISTEMAS EXPERTOS".
ALIANZA EDITORIAL. 1986

[DERM-85]
P.HARMOND; D.KING;
"ARTIFICIAL INTELLIGENCE APPLICATIONS IN BUSINESS:EXPERT SYSTEMS"
JHON WILEY&SONS,INC 1985

[DERM-86]
J.MCDERMOTT; L.ESHELMAN;
"MOLE:A KNOWLEDGE ACQUISITION TOOL THAT USES ITS HEAD"
PROCEEDINGS OF FITH NATIONAL CONFERENCE OF A.I. 1986

[DOHN-87]
C.DOHNER; S.J.ACIERNO;
"GAS TURBINE EXPERT SYSTEM-AN OVERVIEW"
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[FALL-86]
T.C.FALL;
"EVIDENTIAL REASONING WITH TEMPORAL ASPECTS"
PROCEEDINGS OF FITH NATIONAL CONFERENCE OF A.I. 1986

[FAUG-85]
W.S.FAUGHT; B.ROTHLEDER;
"A PROTOTYPE FUEL-SHUFFLING SYSTEM USING A KNOWLEDGE-BASED TOOL KIT"
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR
POWER PLANT OPERATION AND CONTROL 1985

[FELK-85]
L.FELKEL; H.POLKE; H.SCHULLER; H.EISGRUBER; H. HOFFMAN;
"APPLICATION OF AN ADVANCED SOFTWARE CONCEPT FOR OPERATOR AIDS IN
NUCLEAR POWER PLANTS"
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR
POWER PLANT OPERATION AND CONTROL 1985

[FINN-87-A]
G.A.FINN; F.P.WHITTUM; R.P.BONE;
"AN EXPERT SYSTEM FOR TECHNICAL SPECIFICATIONS"
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[FINN-87-B]
G.A.FINN; J.R.HALL;
"AN EXPERT SYSTEM FOR ROTATING EQUIPMENT VIBRATION DIAGNOSIS".
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[FRAN-85]
M.V.FRANK; S.A.EPSTEIN;T.CASEY;
"APPLICATION OF ARTIFICIAL INTELLIGENCE TO IMPROVE PLANT AVAILABILITY"
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR
POWER PLANT OPERATION AND CONTROL 1985

[FREN-86]
S.FRENCH;
"DECISION THEORY:AN INTRODUCTION TO THE MATHEMATICS OF RATIONALITY".
ELLIS HORWOOD LTD.ENGLAND. 1986

[FROG-87]
B.FROGNER; S.HUANG; D.SHIEH; T.KESSLER; G.LIU; A.E.TOME;
S.M.DIVAKARUNI;
"EPRI DEMOSTRATION OF DECISION TREE APPROACH TO HEAT RATE DIAGNOSTICS
FOR FOSSIL POWER PLANTS".
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[GOLD-83]
"GOLDEN COMMON LISP.MANUAL.VERSION 1.01"
GOLD HILL COMPUTER INC 1983

[GOME-87-A]
P.GOMEZ-ARROYO;
"LA TECNOLOGIA AERONAUTICA Y EL DESARROLLO DE LA CIENCIA COGNITIVA"
ASOCIACION DE INGENIEROS AERONAUTICOS.REVISTA DE INGENIERIA
AERONAUTICA Y ASTRONAUTICA.NUMERO: 290-291 1987

[GOME-87-B]
P.GOMEZ-ARROYO
"SISTEMA DE ROUTING-CONTROL DE IBERIA"
ASOCIACION DE INGENIEROS AERONAUTICOS.REVISTA DE INGENIERIA
AERONAUTICA Y ASTRONAUTICA.NUMERO: 290-291 1987

[HARM-85]
J.M.DERMONT;
"ARTIFICIAL INTELLIGENCE APPLICATIONS FOR BUSINESS:EXPERT SYSTEMS"
ABLEX PUBLISHING CO. 2TH. EDITION. 1985

[HASL-84]
D.W.HASLING; W.J.CLANCEY;G.RENNELS;
"STRATEGIES EXPLANATIONS TO CONSULTING SYSTEM IN DIAGNOSIS"
ACADEMIC PRESS-M.J.COOMBS 1984

[HOLD-85]
A.HOLD; D.BERAHA; O.LUPA'S;
"PRESSURIZED WATER REACTOR AND BOILING WATER REACTOR REAL-TIME FOR CORE SURVEILLANCE AND DIGITAL CONTROL"
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR POWER PLANT OPERATION AND CONTROL 1985

[HOLL-85]
E.HOLLNAGEL;
"COGNITIVE SYSTEMS ENGINEERING AS A DESIGN TOOL"
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR POWER PLANT OPERATION AND CONTROL 1985

[HONG-85]
HONG-NIAM JOW; J.H.HOOPS; C.K.WHITNEY;
"INFORMATION PRIORITIZATION FOR TACTICAL SUPPORT OF NUCLEAR PLANT OPERATIONS IN OFF-NORMAL SITUATIONS"
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR POWER PLANT OPERATION AND CONTROL 1985

[KISN-85-A]
R.A.KISNER;
"ANALYTICAL EVALUATION OF COMPUTER-BASED DECISION AIDS"
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR POWER PLANT OPERATION AND CONTROL 1985

[LAMB-84]
B.A.LAMBIRD; D.LAVINE; L.N.KANAL;
"DISTRIBUTED ARCHITECTURE AND PARALLEL NONDIRECTIONAL SEARCH FOR KNOWLEDGE-BASED CARTOGRAPHIE FEATURE EXTRACTION SYSTEMS"
ACADEMIC PRESS-M.J.COOMBS 1984

[LANG-87]
R.B.LANG;
"TRIBES-A CPC/CEAC TRIP BUFFER EXPERT SYSTEM"
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[LI-86]
G.LI; B.W.WAH;
"MANIP-2:A MULTICOMPUTER ARCHITECTURE FOR EVALUATING LOGIC PROGRAMS"
PROCEEDINGS OF THE INTERNATIONAL CONFERENCE OF PARALLEL PROCESSING.
IEEE 1985

[LIPN-85]
M.H.LIPNER; R.A.MUNDY; A.J.IMPINK; C.E.MEYER;
"A COMPUTERIZED EMERGENCY PROCEDURE SYSTEM FOR A PRESSURIZED WATER
REACTOR"
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR
POWER PLANT OPERATION AND CONTROL 1985

[MART-86]
C.MARTINEZ ARNAIZ;
"LA SEGURIDAD EN LAS ESTRUCTURAS AERONAUTICAS"
ASOCIACION DE INGENIEROS AERONAUTICOS.REVISTA DE INGENIERIA
AERONAUTICA Y ASTRONAUTICA.NUMERO: 283 1986

[MAUR-72]
W.D.MAURER
"THE PROGRAMMER'S INTRODUCTION TO LISP"
1972

[MEAR-87]
R.J.MEARS;
"MINIMIZING MAINTENANCE AND OUTAGE TIME WITH AN AUTOMATED TAGGING
SYSTEM"
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[MICH-86A]
R.S.MICHALSKI; I.MOZETIC; J.HONG; N.LAWAC;
"THE MULTI-PURPOSE INCREMENTAL LEARNING SYSTEM AQ15 AND ITS TESTING
APPLICATION TO THREE MEDICAL DOMAINS"
PROCEEDINGS OF FITH NATIONAL CONFERENCE OF A.I. 1986

[MONT-85]
K.MONTA; S.FUKUTOMI; M.ITOCH; I.TAI;
"DEVELOPMENT OF A COMPUTERIZED OPERATOR SUPPORT SYSTEM FOR BOILING WATE
REACTOR POWER PLANTS"
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR
POWER PLANT OPERATION AND CONTROL 1985

[NASE-87]
J.NASER; R.COLLEY; J.GAISER; T.BROOKMIRE; S.ENGLE;
"A FUEL INSERT SHUFFLE PLANNER EXPERT SYSTEM "
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[OLSE-87]
L.OLSEN;
"THE DEVELOPMENT OF PILOT ALARM RESPONSE EXPERT SYSTEMS AT CONSOLIDATED
EDISON"
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[PACK-87]
R.W.PACK; J.R.NASCAL; R.A.MANCINI;
"THE TIP EXPERT SYSTEM FOR HEAT RATE IMPROVEMENT"
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[PAPI-85]
B.PAPIN; P.BERNAD; J.L.TYRAN; G.ZWINGELSTEIN; P.BAJARD;
"ON-LINE SURVEILLANCE OF PRESSURIZED WATER REACTORS BY PROCESS ANALYSIS
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR
POWER PLANT OPERATION AND CONTROL 1985

[PETR-87]
W.PETRICK; K.NG; C.STUART; D.CAIN;
"A PRODUCTION SYSTEM FOR COMPUTERIZED EMERGENCY PROCEDURES TRACKING"
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[POWE-86]
C.A.POWELL; C.K.PICKERING; K.T.WESCOURT;
"SYSTEM INTEGRATION OF KNOWLEDGE-BASED MAINTENANCE AIDS"
PROCEEDINGS OF FITH NATIONAL CONFERENCE OF A.I. 1986

[RAHM-87]
S.RAHMAN;R.BHATNAGAS;
"AN EXPERT SYSTEM BASED ALGORITHM FOR SHORT TERM LOAD FORECAST"
IEEE/PES WINTER MEETING.NEW ORLEANS.LOUISIANA 1987

[RINE-87]
S.C.RINEHART; G.H.KOCH;
"COREX: AN EXPERT SYSTEM FOR CORROSION ANALYLIS AND TREATMENT"
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[ROMB-85]
C.T.ROMBAUGHT; R.A.KOCHENDARFER; K.G.KARGOL;
"ADVANCED IN USING DIGITAL COMPUTERS TO MONITOR AND CONTROL THE POWER
PEAKING IN NUCLEAR REACTORS"
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR
POWER PLANT OPERATION AND CONTROL 1985

[SANT-86]
J.L.SANTAMARIA;
"CONFERENCIA INTERNACIONAL SOBRE PROTECCION Y SEGURIDAD EN EL TRANSPORT
AEREO"
ASOCIACION DE INGENIEROS AERONAUTICOS.REVISTA DE INGENIERIA
AERONAUTICA Y ASTRONAUTICA.NUMERO: 283 1986

[SAYL-87]
C.H.M.SAYLOR; S.L.LIPTON;
"EXPERT SYSTEM TECHNOLOGY-THE IMPLEMENTATION PROCESS"
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[SEBO-85]
D.E.SEBO; B.W.DIXON; M.A.BRAY;
"A REACTOR SAFETY ASSESMENT SYSTEM"
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR
POWER PLANT OPERATION AND CONTROL 1985

[SHOR-83]
E.H.SHORTLIFFE;C.P.LANGLOTZ;
"ADAPTACION DE UN SISTEMA DE CONSULTA DE CRITICA DE PLANES DE USUARIO"
ACADEMIC PRESS-M.J.COOMBS 1984

[SMIT-85]
D.E.SMITH; S.E.SHEEMAN
"A CONTROL SYSTEM VERIFIER USING AUTOMATED REASONING SOFTWARE"
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR POWER PLANT OPERATION AND CONTROL 1985

[SOPO-87]
D.M.SOPOCY; A.R.GLAZER; J.A.MONTANNS;
"EXPERT SYSTEMS APPLICATIONS IN WATER TREATMENT"
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[STEF-87]
A.STEFANINI; M.GALLANTI;
"EXPERT SYSTEMS APPLICATIONS IN POWER GENERATION AND DISTRIBUTION: A SURVEY ON THE ONGOING PROJECTS AT CISE"
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[STRA-85]
R.C.STRATTON;G.G.TOWN;
"AN A.I. PROGRAM IN A COMPUTER APPLICATION SUPPORTING NUCLEAR REACTOR OPERATIONS"
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR POWER PLANT OPERATION AND CONTROL 1985

[SUDD-85]
A.L.SUDDUTH;
"UTILITY PERSPECTIVES ON A.I. APPLICATION TO NUCLEAR POWER"
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR POWER PLANT OPERATION AND CONTROL 1985

[SUDD-87]
A.L.SUDDUTH;
"A.I. IN PROCESS DESIGN AND OPERATION"
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[SUN-87]
B.K.H.SUN; J.A.NASER; D.G.CAIN;
"EPRI PROJECTS:TECHNICAL PROGRESS UPDATE"
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[SYST-84-A]
SYSTEMS DESIGNERS PLC SOFTWARE TECHNOLOGY CENTRE
"POPLOG:A MULTIPURPOSE,MULTILANGUAGE PROGRAM DEVELOPMENT,ENVIRONMENT"
STC.SURREY 1984

[SYST-84-B]
SYSTEMS DESIGNERS PLC SOFTWARE TECHNOLOGY CENTRE
"SAGE:AN EXPERT SYSTEM DEVELOPMENT SHELL"
STC.SURREY 1984

[SYST-84-C]
SYSTEMS DESIGNERS PLC SOFTWARE TECHNOLOGY CENTRE
"ENVISAGE:AN ADVANCED EXPERT SYSTEM DEVELOPMENT SHELL"
STC.SURREY 1984

[TAKA-85]
T.KIGUCHI; H.MOTODA; N.YAMADA; K.YOSHIDA;
"A KNOWLEDGE BASED SYSTEM FOR PLANT DIAGNOSIS"
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR
POWER PLANT OPERATION AND CONTROL 1985

[TAYL-86]
T.TAYLOR; J.WALL; D.LUBKEMAN;
"KNOWLEDGE-BASED EXPERT SYSTEMS FOR POWER ENGINEERING PROBLEMS"
ELECTRIC POWER RESEARCH CENTER.RALEIGHT.NORTH CAROLINA. 1986

[TOUC-87]
R.A.TOUCHTON; A.DALE; D.G.CAIN;
"REALM: AN EXPERT SYTEM FOR CLASSIFYING EMERGENCIES"
SEMINAR ABOUT EXPERT SYSTEMS APPLICATIONS IN POWER PLANTS. 1987

[UNDE-85]
W.E.UNDERSTOOD; R.E.SWANSON;
"CAUSAL REPRESENTATION AND EXPLANATION OF NUCLEAR POWER PLANT OPERATION"
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR
POWER PLANT OPERATION AND CONTROL 1985

[WALM-85]
S.J.WALMSLEY;
"A.I. METHODOLOGIES APPLIED TO CONSTRUCTION MANAGEMENT DECISION SUPPORT SYSTEMS"
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR POWER PLANT OPERATION AND CONTROL 1985

[WASH-85]
T.WASHIO; M.KITAMURA; K.KATAJIMA; K.SIJIYAMA;
"SEMANTIC NETWORK APPROACH TO AUTOMATED FAILURE DIAGNOSIS IN NUCLEAR POWER PLANT"
INTERNATIONAL TOPICAL MEETING ON COMPUTER APPLICATIONS FOR NUCLEAR POWER PLANT OPERATION AND CONTROL 1985

[WINS-84]
P.H.WINSTON;B.K.P.HORN
"LISP"
ADDISON-WESLEY.2TH. EDITION 1984

[WOLL-86]
B.F.WOLLEMBERG;
"FEASIBILITY STUDY FOR AN ENERGY MANAGEMENT SYSTEM INTELLIGENT ALARM PROCESSOR"
IEEE TRANSACTIONS ON POWER SYSTEM.VOL.PWRS-1.NO.2 1986

[WOOL-86]
b.WOOLF; D.BLEGEN; J.H.JANSEN; A.VERLOOP;
"TEACHING A COMPLEX INDUSTRIAL PROCESS"
PROCEEDINGS OF FITH NATIONAL CONFERENCE OF A.I. 1986

[ZIMM-84]
A.C.ZIMMER;
"A MODEL FOR THE INTERPRETATION OF VERBAL PREDICTIONS"
ACADEMIC PRESS-M.J.COOMBS 1984

A N E X O 1

A.1.1.- INTRODUCCION

En este anexo se va a resolver de forma comentada un caso concreto, simple, pero suficientemente representativo como para que aparezcan aplicadas todas las metodologías propuestas en esta tesis.

A.1.2.- CONSTRUCCION DE UN ARBOL DE FALLOS A PARTIR DE REGLAS

Se parte del siguiente conjunto de reglas:

(1) X1 ⟶ A

(2) X6 ⟶ A

(3) X7 ⟶ A

(4) A ⟶ B

(5) X8 ⟶ B

(6) B $\wedge$ X9 $\wedge$ X40 ⟶ C

(7) $\bar{X}9 \wedge \bar{X}40$ ⟶ D

(8) C ⟶ E

(9) X40 ⟶ E

(10) E $\wedge$ $\bar{X}36 \wedge$ X39 ⟶ F

(11) F ⟶ G

(12) X31 ⟶ G

(13) X40 ⟶ G

Donde en principio, tanto los X_i como las letras, indican estados de componentes y subsistemas, del sistema a estudiar.

Aplicando el procedimiento propuesto se establecen las siguientes conclusiones:

1) Se localiza la G como concepto que no se encuentra en la parte izquierda de ninguna regla.
2) Se comprueba que está más de una vez ⟶ es puerta OR ⟶ es el primer nivel y es el SND. Se le denominará OR 1.1.
3) Se comprueba que F que es antecedente de G está a la derecha de la regla (11) luego F es una puerta. Asimismo se comprueba que X31 y X40 que son antecedentes de G no están a la derecha de ninguna regla por lo que se deduce que X31 y X40 son SB.

Se vuelve a aplicar el paso 2) a la nueva puerta.

2) Solo está una vez.
3) Tiene varios conceptos a su izquierda, por tanto es una puerta AND del segundo nivel, y que se denominará AND 2.1. Hasta ahora tenemos:

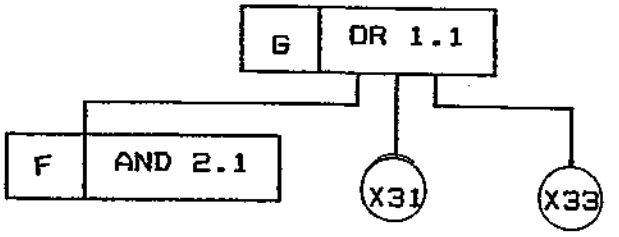


Fig. A1-1

5) De las partes izquierda de F, se comprueba que E se encuentra a la derecha de las reglas (8) y (9) por tanto es una puerta OR del nivel 3 y que denominaremos OR 3.1. Se comprueba asimismo que X36 y X39 no están a la derecha de ninguna regla, por lo que tienen que ser SB, quedando ahora el árbol así:

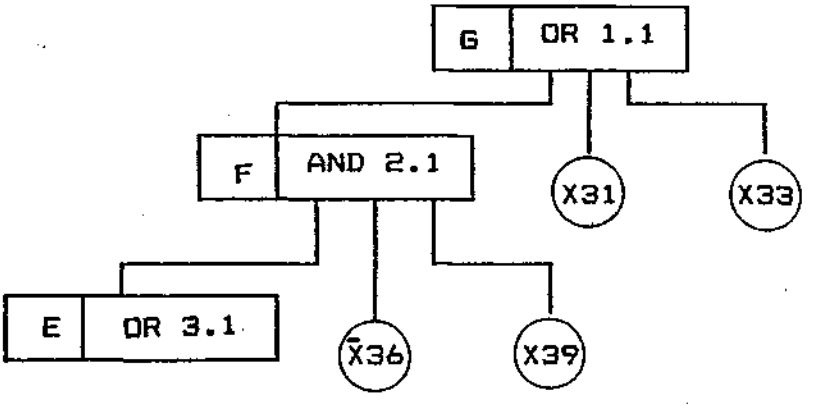


Fig. A1.2

Reiterando este proceso, se llega a la construcción del árbol de fallos que se muestra en la Fig. A1-3 y en el que se han añadido las etiquetas correspondientes a los SB y a una puerta.

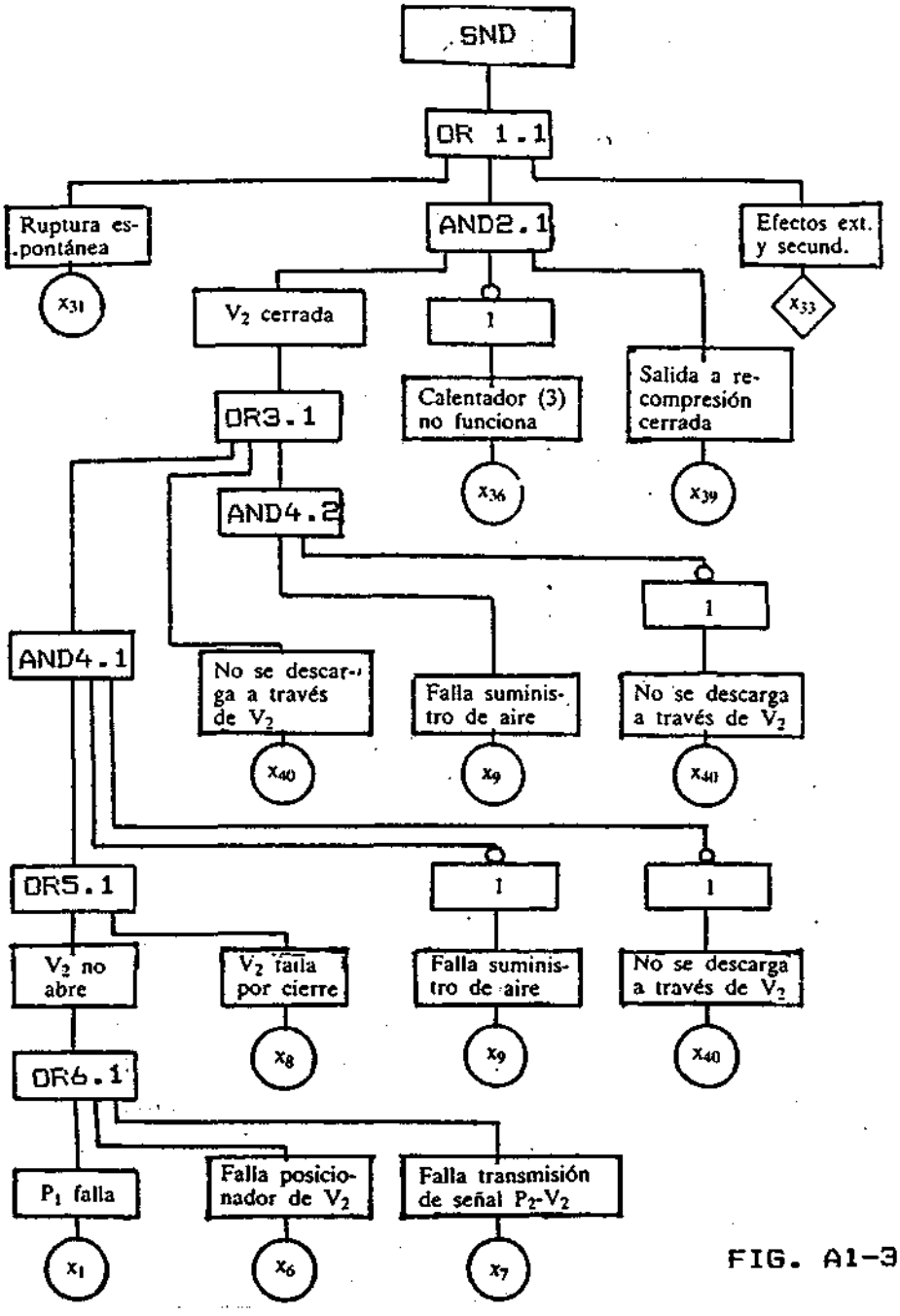


FIG. A1-3

Mediante este método el árbol ha quedado construido, deduciéndose además que el conjunto de reglas presentado es consistente a un solo árbol de fallos, puesto que solo se detecta un SND. Asimismo, se comprueba que no queda ninguna regla sin aplicar, por lo que todo el conocimiento en forma de reglas es útil, y finalmente se comprueba que no hay ninguna regla que encontrándose su parte derecha una sola vez tuviera un solo antecedente, de lo que se deduce que no hay ninguna puerta falsa.

A.1.3.- ESTRUCTURACION Y REPRESENTACION DEL ARBOL POR NIVELES

La estructuración por niveles es un procedimiento que como se ha visto en la construcción del árbol puede hacerse a la vez que éste. Si se parte de un árbol construido basta con aplicar la metodología propuesta, llegándose a establecer los niveles a los que pertenece cada puerta, y por tanto al mismo árbol mostrado en la fig. A1-3.

Representación : Se indicará en cada nivel, cada puerta, y a continuación, las entradas que concurran a ella desde el nivel inmediato inferior.

NIVEL 6

(OR 6.1 = X1 X6 X7)

NIVEL 5

(OR 5.1 = OR 6.1 X8)

NIVEL 4

(AND 4.1 = OR 5.1 $\bar{X}9$ $\bar{X}40$) (AND 4.2 = X9 $\bar{X}40$)

NIVEL 3

(OR 3.1 = AND 4.1 AND 4.2 X40)

NIVEL 2

(AND 2.1 = OR 3.1 $\bar{X}36$ X39)

NIVEL 1

(OR 1.1 = AND 2.1 X31 X33)

El árbol queda así representado estando ya preparado para aplicar el procedimiento de depuración.

A.1.4.- DEPURACION

Siguiendo las instrucciones de depuración se observa que en el nivel 5, la puerta OR 5.1 contiene a la puerta OR 6.1, por lo que ésta última al ser del mismo tipo que la de la cabecera, se sustituirá por su contenido en niveles anteriores, quedando la puerta OR 5.1 así: (OR 5.1 = X1 X6 X7 X8) con lo que se observa que en realidad los niveles del árbol han quedado reducidos en una unidad, toda vez que, el resto de las puertas en cada nivel no presenta características para ser depuradas. La representación del árbol depurado pasa entonces a ser:

NIVEL 5

(OR 5.1 = X1 X6 X7 X8)

NIVEL 4

(AND 4.1 = OR 5.1 $\bar{X}9$ $\bar{X}40$) (AND 4.2 = X9 $\bar{X}40$)

NIVEL 3

(OR 3.1 = AND 4.1 AND 4.2 X40)

NIVEL 2

(AND 2.1 = OR 3.1 $\bar{X}36$ X39)

NIVEL 1

(OR 1.1 = AND 2.1 X31 X33)

A.1.5.- FORMACION DE CM EN CADA NIVEL

Se aplicarán todos los pasos según la metodología propuesta, y que se describen a continuación:

NIVEL 5

a) Se revisa la puerta de nivel N=5. Es puerta OR.

b) Como la puerta es OR, se forman tantos C.S. como S.B. haya, es decir:

$(C.S.)_{N=5}$ = (X1) (X6) (X7) (X8) [OR 5.1] (cuatro).

NIVEL 4

c) Se revisa la primera puerta.Es tipo C. [AND 4.1]

5) Se extrae la puerta contenida = OR 5.1

1) Es puerta OR. Se forman como C.S. los que concurran a ella provenientes del nivel inferior.

(X1) (X6) (X7) (X8)

3) Se comprueba que ninguno contiene a otro.
Se comprueba que ninguno tiene elementos repetidos.
Se comprueba que ninguno tiene un S.B. y su negado.

7) Se añaden los S.B. $\bar{X}9$ $\bar{X}40$ a los anteriores.Nos queda:

($\bar{X}9$ $\bar{X}40$ X1) ($\bar{X}9$ $\bar{X}40$ X6) ($\bar{X}9$ $\bar{X}40$ X7) ($\bar{X}9$ $\bar{X}40$ X8)

3) Sin variación.

4) Cuatro.

Se revisa la segunda puerta.[AND 4.2]. Tipo A.

b) Se forma el C.S. (X9 $\bar{X}$40).

d) ($\bar{X}$9 $\bar{X}$40 X1) ($\bar{X}$9 $\bar{X}$40 X6) ($\bar{X}$9 $\bar{X}$40 X7) ($\bar{X}$9 $\bar{X}$40 X8).
(Cuatro) [AND 4.1]
(X9 $\bar{X}$40) [AND 4.2] (Uno).

NIVEL 3

c) Revisión de puertas. Solo una. [OR 3.1] Tipo C.

5) Se extraen las puertas contenidas. [AND 4.1] [AND 4.2]

1) Se forman:

($\bar{X}$9 $\bar{X}$40 X1) ($\bar{X}$9 $\bar{X}$40 X6) ($\bar{X}$9 $\bar{X}$40 X7) ($\bar{X}$9 $\bar{X}$40 X8)
(X9 $\bar{X}$40)

3) Se comprueba que ninguno contiene a otro.
" " " tiene S.B. repetidos.
" " " " un S.B. y su negado.

6) Se añade a los anteriores el S.B. (X40)

3) Sin variación.

4) Cinco.

d) ($\bar{X}9$ $\bar{X}40$ X1) ($\bar{X}9$ $\bar{X}40$ X6) ($\bar{X}9$ $\bar{X}40$ X7) ($\bar{X}9$ $\bar{X}40$ X8) (X9 $\bar{X}40$) (X40) [OR 3.1] (Seis).

NIVEL 2

c) Revisión de puertas. Unica. [AND 2.1] Tipo C.

5) Se extrae OR 3.1.

2) Al ser única, las combinaciones forman la misma lista de OR 3.1.

3) Sin variación.

7) Se añaden a cada uno de los anteriores los S.B. $\bar{X}36$ X39 y tras aplicar 3), 4), y d), quedan:

($\bar{X}36$ X39 $\bar{X}9$ $\bar{X}40$ X1) ($\bar{X}36$ X39 $\bar{X}9$ $\bar{X}40$ X6)
($\bar{X}36$ X39 $\bar{X}9$ $\bar{X}40$ X7) ($\bar{X}36$ X39 $\bar{X}9$ $\bar{X}40$ X8)
($\bar{X}36$ X39 X9 $\bar{X}40$) ($\bar{X}36$ X39 X40) [AND 2.1] (Seis)

NIVEL 1

c) Revisión de puertas. Unica. [OR 1.1]. Tipo C.

5) Se extrae AND 2.1

1) Los C.S. son los de [AND 2.1]

3) Sin variación.

6) Se añaden a los anteriores los S.B. (X31) (X33).
Aplicando después 3), 4), y d) queda:

($\bar{X}$36 X39 $\bar{X}$9 $\bar{X}$40 X1) ($\bar{X}$36 X39 $\bar{X}$9 $\bar{X}$40 X6)
($\bar{X}$36 X39 $\bar{X}$9 $\bar{X}$40 X7) ($\bar{X}$36 X39 $\bar{X}$9 $\bar{X}$40 X8)
($\bar{X}$36 X39 X9 $\bar{X}$40) ($\bar{X}$36 X39 X40) (X31) (X33)
[OR 1.1] (Ocho).

En éste, se han conseguido deducir todos los C.M. del sistema estudiado.

Una vez deducidos los CM puede optarse ya por la resolución de incertidumbre de cara a ver cuál es en ese momento el que más aporta a la no-fiabilidad del sistema en ese momento.

A.1.5.- RESOLUCION DE INCERTIDUMBRE EN CM

Antes de poder resolver la incertidumbre, que en este caso y dado el tamaño del árbol, la del tipo estructural apenas aporta nada, habrá que disponer de los datos de fiabilidad y plantear una serie de cuestiones previas que permitan realizar los cálculos necesarios. La consecución del método requiere entonces el disponer de:

- El parámetro de tasa de fallos de cada SB.
- Etiqueta de identificación de cada SB ó puerta en caso de tenerla.
- Tiempo de funcionamiento de cada componente.

Se observa entonces que cada SB lleva asociado el tiempo transcurrido desde su anterior revisión ó sustitución. Se evita de esta forma el concepto de T_{ob} ó Tiempo de observación, que suele afectar a todo un sistema ó subsistema. Asimismo se parte de una distribución de fiabilidad exponencial con lo que se sabe en cada momento cual es la no fiabilidad atribuida a un SB.

Partimos entonces de los siguientes parámetros de fiabilidad:

	TMF(h)	θ (h^{-1})	T_{ob}	No-fiabilidad
X1	$9{,}03.10^{3}$	$1{,}10.10^{-4}$	1000	0,104
X6	$2{,}14.10^{4}$	$4{,}67.10^{-5}$	1000	0,045
X7	$9{,}63.10^{3}$	$1{,}03.10^{-4}$	1000	0,097
X8	$4{,}87.10^{4}$	$2{,}05.10^{-5}$	400	0,008
X9	$6{,}00.10^{3}$	$1{,}66.10^{-4}$	1000	0,152
X31	$1{,}00.10^{7}$	$1{,}00.10^{-7}$	1000	0,00010
X33	$4{,}38.10^{3}$	$2{,}28.10^{-4}$	600	0,0013
X36	$6{,}00.10^{3}$	$1{,}66.10^{-4}$	1000	0,152
X39	$1{,}00.10^{3}$	$1{,}00.10^{-3}$	200	0,181
X40	$6{,}00.10^{3}$	$1{,}66.10^{-4}$	1000	0,152
	DATOS FIJOS		DATOS DINAMICOS	

Esta tabla contiene la lista de los SB que intervienen en este SND. Las dos columnas de DATOS FIJOS, contienen en realidad el mismo parámetro, el TMF ó Tiempo Medio hasta el Fallo, ó bien su inversa que es la θ ó Tasa de Fallos. Se ha denominado datos fijos a estos parámetros toda vez que son introducidos una sola vez.

Aplicando la metodología propuesta para la resolución de incertidumbre, se deducirá que la hipótesis de fallo, será aquel CM que presente una no fiabilidad más alta. Se presenta entonces:

$(\bar{X}36\ X39\ \bar{X}9\ \bar{X}40\ X1) = \min [\ (1-0,152)\ 0,181\ (1-0,152)\ (1-0,152)\ 0,104\ ] = \underline{0,104}$

y de la misma forma:

$(\bar{X}36\ X39\ \bar{X}9\ \bar{X}40\ X6) = \underline{0,045}$

$(\bar{X}36\ X39\ \bar{X}9\ \bar{X}40\ X7) = \underline{0,097}$

$(\bar{X}36\ X39\ \bar{X}9\ \bar{X}40\ X8) = \underline{0,008}$

$(\bar{X}36\ X39\ X9\ \bar{X}40) = \underline{0,152}$

$(\bar{X}36\ X39\ X40) = \underline{0,152}$

$(X31) = \underline{0,0001}$

$(X33) = \underline{0,0013}$

por tanto se deduce que la hipótesis de fallo es la que tiene una no-fiabilidad 0,152 y que corresponde a la presencia simultánea de los SB $(\bar{X}36\ X39\ X40)$ ó bien, de los $(\bar{X}36\ X39\ X9\ \bar{X}40)$. Como ya se ha dicho, cada SB lleva asociado una etiqueta identificativa que en este caso serán:

X1 = "EL EVAPORADOR DE ETILENO ESTA FUNCIONANDO".

X6 = "POSICIONADOR DE VALVULA V_2 FALLA".

X7 = "TRANSMISION DE SEÑAL P_1-V_2 ESTA FALLANDO".

X8 = "VALVULA V_2 FALLA POR CIERRE".

X9 = "EL SUMINISTRO DE AIRE A INSTRUMENTOS ESTA FALLANDO".

X31= "RUPTURA ESPONTANEA DE RECIPIENTE O TUBOS".

X33= "EFECTOS EXTERNOS Y SECUNDARIOS".

X36= "EL CALENTADOR NO ESTA FUNCIONANDO".

X39= "SALIDA A RECOMPRESION CERRADA".

X40= "NO HAY DESCARGA A TRAVES DE V_2.

En base a estos identificadores se establecerá la conclusión del diagnóstico, que en este caso sería:

EL CALENTADOR FUNCIONA, LA SALIDA A RECOMPRESION ESTA CERRADA, Y HAY DESCARGA A TRAVES DE LA VALVULA V2,---- ó bien,----

EL CALENTADOR FUNCIONA, LA SALIDA A RECOMPRESION ESTA CERRADA, Y FALLANDO EL SUMINISTRO DE AIRE A INSTRUMENTOS, NO HAY DESCARGA A TRAVES DE V2, de donde el operador establecería las acciones a tomar.

A.1.6.- CONSTRUCCION DE CONJUNTOS VIRTUALES Y RESOLUCION DE INCERTIDUMBRE EN LOS MISMOS

Atacamos en este ejemplo de forma conjunta las dos cuestiones de la extracción de los CV y eliminación de la incertidumbre, puesto que no afectamos a la claridad de la exposición, siendo además más representativo del proceso real.

NIVEL 1

(CM).....(X31) con no-fiabilidad = 0,0001

(CM).....(X33) " " = 0,0013

El fallo más probable deducible en este instante es pues X33 y por tanto se muestra:

"El fallo más probable es debido a efectos externos y secundarios".

NIVEL 2

(CV).....($\bar{X}$36 X39); Q_e= min ($\bar{X}$36 X39) = min ((1-X36) X39) = min (0,848 0,181)= 0,181

Afecta a 6 CM de un total de 8.

Evaluación del peso: el peso de los CM será siempre el mismo, es decir 1/8 siendo 8 el número total de CM del sistema. Sin embargo el de los CV será según el nivel donde se encuentre. Entonces el parámetro H que será el determinante de la elección será:

(CV) ($\bar{X}$36 X39).....H= 0,181.6/8 = 0,135

(CM) (X31).....H= 0,0001.1/8 = $1{,}28.10_{-5}$

(CM) (X33).....H= 0,0013.1/8 = $1{,}62.10_{-4}$

eligiéndose por tanto el CV, y deduciendo que:

"El fallo es debido al menos a que:
EL CALENTADOR ESTA FUNCIONANDO Y LA SALIDA A RECOMPRESION ESTA CERRADA".

Repitiendo el proceso en cada nivel se tiene:

NIVEL 3

(CM) ($\bar{X}$36 X39 X9 $\bar{X}$40) Q_s = 0,152

(CM) ($\bar{X}$36 X39 X40) Q_s = 0,152

(CV) ($\bar{X}$36 X39 $\bar{X}$9 $\bar{X}$40) Q_s = 0,181 (Afecta a 4 CM)

H : 0,152.1/8 = 0,019

H : 0,152.1/8 = 0,019

H : 0,181.4/8 = 0,090

Deduciéndose entonces que:

" El fallo es debido al menos a que:
EL CALENTADOR ESTA FUNCIONANDO, LA SALIDA A RECOMPRESION ESTA CERRADA, Y NO FALLANDO EL SUMINISTRO DE AIRE DE INSTRUMENTOS, HAY DESCARGA A TRAVES DE V_{2}".

NIVEL 4

(CM) ($\bar{X}36$ X39 $\bar{X}9$ $\bar{X}40$ X1) Q_{1} = 0,104

(CM) ($\bar{X}36$ X39 $\bar{X}9$ $\bar{X}40$ X6) Q_{6} = 0,045

(CM) ($\bar{X}36$ X39 $\bar{X}9$ $\bar{X}40$ X7) Q_{7} = 0,097

(CM) ($\bar{X}36$ X39 $\bar{X}9$ $\bar{X}40$ X8) Q_{8} = 0,008

H : 0,104.1/8 = 0,013

H : 0,045.1/8 = 0,005

H : 0,097.1/8 = 0,012

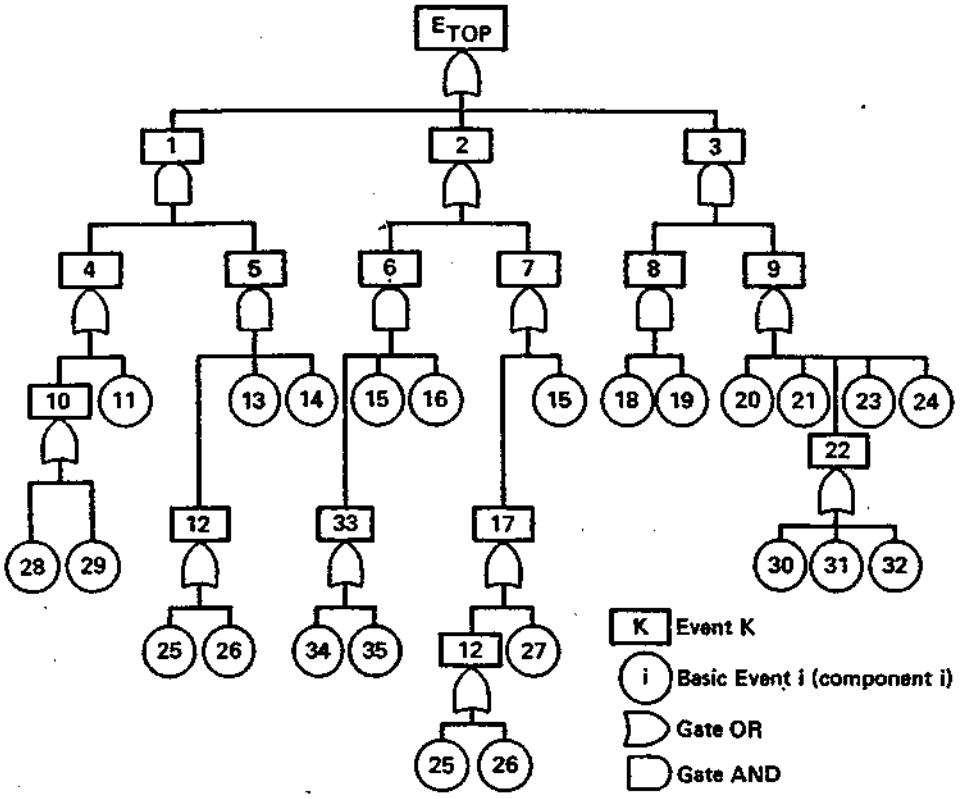
H : 0,008.1/8 = 0,001

Aquí se observa ya como cualquiera de ellos, siendo ya mínimo y por tanto un modo de fallo completo, tiene un peso menor que los CM deducidos en el nivel 3 y por tanto el sistema vuelve a ellos y deduce los mismos que se habían deducido antes en el cálculo del CM menos fiable, como era de esperar.

De comprobar el operador que alguno, y basta con uno, de los SB no se produce, toda la rama queda anulada a partir de ese nivel y el sistema pasaría entonces a discernir por la rama del nivel inmediato superior, cuáles serían los CV de mayor peso para seguir el diagnóstico por ellos.

A.2.- Comentario sobre el apartado H del planteamiento del problema

El problema propuesto por Combustion Engineering Inc. parte de un ejemplo basado en el árbol de fallos mostrado en la Fig. A1-4 y manejado por un conjunto de reglas en número de 24, y que denomina S.E. Lo cierto es que estas reglas constituyen un "control" sobre el conjunto de conexiones lógicas que ligan a los S.B. entre sí y de cara al SND. En el ejemplo por ellos propuesto se realizan una serie de inferencias que llevan a una determinada conclusión.



Diagnostic Search by Deductive Reasoning (Example)

Step No.	Rule No.	Applicable Proposition	Acquired Knowledge
1	2	I} {-}	E_7(-)
2	13	G_7(OR)ΛE_7(-)	E_{17}(-)ΛE [}(-)
3	19	$E_{15}^{\}} \equiv E_{15}^{\}} \Lambda G_7(OR)\Lambda G_6(AND)\Lambda E_7$(-)	E_6(-)
4	10	E_6(-)ΛE_7(-)	E_2(-)
5	13	G_{17}(OR)ΛE_{17}(-)	E_{12}(-)
6	19	$E_{12}^{\}} \equiv E_{12}^{\}} \Lambda G_{17}(OR)\Lambda G_5(AND)\Lambda E_{17}$(-)	E_5(-)
7	14	G_1(AND)ΛE_5(-)	E_1(-)
8	12	G_{TOP}(OR)ΛE_{TOP}(+	E_3(+)
9	22	G_3(AND)ΛE_3(+)	E_8(+)ΛE_9(*)
10	15	G_8(AND)ΛE_8(+)	E_{18}(+)ΛE_{19}(+)

FIG. A1-4 : Ejemplo de árbol de fallos propuesto por C-E y aplicación de las reglas.

En este ejemplo no se maneja el concepto de C.M. y esto hace que según sea la señal de un sensor que en este caso es colocado sobre la puerta 7 hayan de realizarse el conjunto de inferencias cada vez que se tenga que realizar el diagnóstico.

En este ejemplo las puertas se denominan por números dentro de un rectángulo y los SB por números dentro de un círculo. Una vez introducido el árbol, la aplicación de las 24 reglas que representan paso a paso las deducciones extraibles de cada puerta les lleva a deducir que dos SB son críticos para el sistema y que son el 18 y el 19.

El problema parte de una situación de detección de fallo en el SND y de no fallo en la puerta 7. Para mayor detalle, consultar [NEUS-87]. Si atacamos este problema con la metodología propuesta en esta tesis, en primer lugar se tendrían los CM ya almacenados y que en este caso son:

(X36) (X26) (X27) (X15) (X30 X18 X19) (X31 X18 X19)
(X32 X18 X19) (X20 X18 X19) (X21 X18 X19)
(X28 X18 X19) (X24 X18 X19)

El mero hecho de que no hay fallo en la puerta 7 elimina en el árbol depurado los CM :

(X25) (X26) (X27) (X15)

quedándo los otros seis modos de fallo posibles. En el método propuesto en esta tesis, esta labor se realiza en un solo paso, pues al evaluar los CV se tiene:

NIVEL 1: (X25) (X26) (X27) (X15)

Al introducir la señal del sensor en 7 estos CM ó CV del primer nivel han sido eliminados.

NIVEL 2

(CV)....(X13 X14) ≡ Sus SB no están en los CM, por tanto queda eliminado.

(CV)....(X18 X19) ≡ Sus SB sí están en los CM, por tanto es CV.

(CV)....(X15 X16) ≡ Sus SB no están en los CM por tanto queda eliminado.

Este es un caso que resulta representativo de la eliminación de incertidumbre por información de tipo estructural.

Se ve pues que, en tan solo dos pasos se llega a la misma conclusión que C-E, demostrándose asi que la deducción de los CV permite llegar a la conclusión con gran rapidez. Todas las cuestiones apuntadas, quedan reflejadas en el conjunto de programas que se muestran a continuación, (anexo 2) y sus resultados de aplicación a dos ejemplos, en el anexo 3.

ANEXO 2

```
{DSK}<LISPFILES>ARBOLES>EST-ARBOL-FALLOS.LSP;15     30-Nov-87 11:51:40              Page 1

(FILECREATED "30-Nov-87 11:51:40" {DSK}<LISPFILES>ARBOLES>EST-ARBOL-FALLOS.LSP;15 27189

      changes to:  (VARS EST-ARBOL-FALLOSCOMS)
                   (FNS CONSTRUYE.NOMBRE.NEGADO FADD.ARBOL DESACTIVAR FADD.NODO FREMOVE.NODO
                        TODAS.PROPIEDADES.NODO PERTENECE GENERAR.NOMBRE.ASOCIADO FGET.NODO FPUT.NODO)

      previous date: "23-Nov-87 18:54:27" {DSK}<LISPFILES>ARBOLES>EST-ARBOL-FALLOS.LSP;1)


(PRETTYCOMPRINT EST-ARBOL-FALLOSCOMS)

(RPAQQ EST-ARBOL-FALLOSCOMS ((* Funciones de manejo de la estructura ARBOL y los NODOS del mismo
                                (PRIMITIVAS BASICAS)
                                ,)
                             (FNS FGET.ARBOL FPUT.ARBOL FADD.ARBOL FREMOVE.ARBOL TODO.ARBOL
                                PERTENECE)
                             (* Funciones de manejo de la informacion de los nodos
                                (especifico para arboles de fallo))
                             (FNS ACTIVAR DESACTIVAR FGET.NODO FPUT.NODO FADD.NODO FREMOVE.NODO
                                CARACTERISTICA.NODO TODO.NODO TODAS.PROPIEDADES.NODO EXISTE.NODO)
                             (* Funciones de generacion de nombres)
                             (FNS GENERAR.NOMBRE GENERAR.NOMBRE.ASOCIADO CONSTRUYE
                                CONSTRUYE.ASOCIADO CONSTRUYE.NOMBRE.NEGADO)
                             (* Funciones de manipulacion de la estructura
                                (adicion y eliminacion de nodos del arbol)
                                Se definen tres procedimientos distintos de insercion de nodos en
                                el arbol ,correspondientes a cada uno de los diferentes tipos de
                                nodos que puedo encontrar en un arbol de fallos)
                             (FNS ANADIR.BASICO.NUEVO ANADIR.PUERTA.NUEVA ANADIR.NODO.ASOCIADO
                                CREAR.NUEVO.ASOCIADO MODIFICA.PROPS.ASOCIADO SUPRIMIR.NODO
                                ELIMINAR.NODO INC.LISTA.ELIM)))
```

(* Funciones de manejo de la estructura ARBOL y los NODOS del mismo (PRIMITIVAS BASICAS) ,)

```
(DEFINEQ

(FGET.ARBOL
  [LAMBDA (ARBOL PROPIEDAD)                                    (* edited: "10-Nov-87 10:08")

        (* * Obtiene la propiedad especificada de la estructura arbol)


    (GETPROP ARBOL PROPIEDAD])

(FPUT.ARBOL
  [LAMBDA (ARBOL CARACTERISTICA VALOR)                         (* edited: "10-Nov-87 10:08")

        (* * Incluye como valor para la propiedad del arbol especificado el que aparece en la lista de argumentos)


    (PUTPROP ARBOL CARACTERISTICA VALOR])

(FADD.ARBOL
  [LAMBDA (ARBOL PROPIEDAD VALOR)                              (* edited: "27-Nov-87 17:05")

        (* * Anade un valor determinado a los ya existentes en alguna de las caracteristicas del arbol especificado)


    (COND
      ((NULL VALOR))
      ((ATOM VALOR)
        (ADDPROP ARBOL PROPIEDAD VALOR))
      (T (PUTPROP ARBOL PROPIEDAD (APPEND VALOR (GETPROP ARBOL PROPIEDAD])

(FREMOVE.ARBOL
  [LAMBDA (ARBOL PROPIEDAD)                                    (* edited: "10-Nov-87 10:08")

        (* * Elimina la propiedad especificada del arbol (propiedad especifica del arbol, no de sus nodos))


    (REMPROP ARBOL PROPIEDAD])
```

```
(TODO.ARBOL
  [LAMBDA (ARBOL)                                    (* edited: "29-Oct-87 03:48")

          (* * Obtiene todas las caracteristicas del arbol especificado. Las devuelve en formato de lista de propiedad)


    (GETPROPLIST ARBOL])

(PERTENECE
  [LAMBDA (NODO ARBOL)                               (* edited: "27-Nov-87 17:10")

          (* * La funcion comprueba la pertenencia del nodo al arbol devolviendo un valor T o NIL)


    (LET ((CARACT-ARBOL (FGET.ARBOL ARBOL (QUOTE CARACTERISTICA)))
          (CARACT-NODO (CARACTERISTICA.NODO NODO)))

          (* * Se considera que pertenece si el identificador del arbol coincide con el del nodo, (previamente Se comprueba
          que el nodo especificado existe))


         (AND (EXISTE.NODO NODO)
              (EQ CARACT-ARBOL CARACT-NODO])
)


(* Funciones de manejo de la informacion de los nodos (especifico para arboles de fallo))

(DEFINEQ

(ACTIVAR
  [LAMBDA (NODO)                                     (* edited: "16-Nov-87 11:43")

          (* * Esta funcion realiza la activacion del nodo. Esta operacion, de acuerdo con la implementacion adoptada para la
          estructura ARBOL consiste simplemente en especificar el valor T para la propiedad del nodo ACTIVO)


    (DESACTIVAR NODO)                                (* Inicializacion del nodo)
    (FPUT.NODO NODO (QUOTE ACTIVO?)
               T])

(DESACTIVAR
  [LAMBDA (NODO)                                     (* edited: "27-Nov-87 17:05")

          (* * Esta funcion elimina el nodo del arbol. (= Desactiva el nodo))


    (SETPROPLIST NODO NIL])

(FGET.NODO
  [LAMBDA (NODO PROPIEDAD)                           (* edited: "25-Nov-87 14:46")

          (* * Obtiene el valor de la propiedad del nodo especificado. No realiza comprobaciones de pertenencia o existencia
          del nodo. Estas deberan realizarse antes de efectuar la llamada)


          (* * En la especificacion por defecto del SELECTQ, se recupera la informacion del nodo asociado)


    (SELECTQ PROPIEDAD
             ((PADRE NEGADO ACTIVO? ASOCIADO)

          (* * Las especificaciones aparecen en el propio nodo)


               (GETPROP NODO PROPIEDAD))
             ((IDENTIFICACION ID-NEGADO)

          (* * La especificacion aparecera, o bien en el nodo real si es un suceso basico, o bien en el nodo asociado.)


               (if (BASICO? NODO)
```

```
                then (FGET.COMPONENTE (NODO.REAL NODO)
                                      PROPIEDAD)
              else (GETPROP (NODO.ASOCIADO NODO)
                            PROPIEDAD)))
      ((PARAMETRO TIEMPO)
```

(* Las especificaciones aparecen en el nodo real. Solo son significativas para los suceso basicos)*

```
       (FGET.COMPONENTE (NODO.REAL NODO)
                        PROPIEDAD))
     (GETPROP (NODO.ASOCIADO NODO)
              PROPIEDAD])
```

(FPUT.NODO

```
  [LAMBDA (NODO PROPIEDAD VALOR)                    (* edited: "25-Nov-87 14:45")
```

(Incluye como valor de la propiedad del nodo el especificado. No realiza comprobaciones de pertenencia al arbol o existencia del nodo. Estas comprobaciones deberan realizarse antes de efectuar la llamada)*

```
   (SELECTQ PROPIEDAD
            ((PADRE NEGADO ACTIVO? ASOCIADO)
```

(* Las especificaciones aparecen en el propio nodo)*

```
       (PUTPROP NODO PROPIEDAD VALOR))
     ((IDENTIFICACION ID-NEGADO)
```

(* La especificacion aparecera, o bien en el nodo real si es un suceso basico, o bien en el nodo asociado. Las modificaciones sobre los nodos reales no son permitidas utilizando estas primitivas de manejo de estructura)*

```
       (if (BASICO? NODO)
           then (PRINTOUT T " OPERACION NO PERMITIDA " T)
         else (PUTPROP (NODO.ASOCIADO NODO)
                       PROPIEDAD VALOR)))
     ((PARAMETRO TIEMPO)
```

(* Las especificaciones aparecen en el nodo real. Solo son significativas para los suceso basicos)*

```
       (PRINTOUT T " OPERACION NO PERMITIDA " T))
     (PUTPROP (NODO.ASOCIADO NODO)
              PROPIEDAD VALOR])
```

(FADD.NODO

```
  [LAMBDA (NODO PROPIEDAD VALOR)                    (* edited: "27-Nov-87 17:08")
```

(* Anade los valores especificado a los ya existentes de la propiedad del nodo)*

(* !!! Cuidado con el tratamiento especial de las listas !!!)*

```
   (COND
     ((NULL VALOR))
     ((ATOM VALOR)
```

(* Se anade con ADDPROP)*

```
      (SELECTQ PROPIEDAD
               ((PADRE NEGADO ACTIVO? ASOCIADO)
```

(* Las especificaciones aparecen en el propio nodo)*

```
          (ADDPROP NODO PROPIEDAD VALOR))
        ((IDENTIFICACION PARAMETRO TIEMPO ID-NEGADO)
```

(* Operaciones de adicion no permitidas)*

```
                    (PRINTOUT T " OPERACION NO PERMITIDA " T))
                  (ADDPROP (NODO.ASOCIADO NODO)
                           PROPIEDAD VALOR)))
      (T
```

(* Anade todos los elementos de la lista)*

```
        (FPUT.NODO NODO PROPIEDAD (APPEND VALOR (FGET.NODO NODO PROPIEDAD])
```

(FREMOVE.NODO
```
  [LAMBDA (NODO PROPIEDAD)                                    (* edited: "27-Nov-87 17:08")
```

(* Elimina una propiedad del nodo especificado del arbol)*

```
    (SELECTQ PROPIEDAD
             ((PADRE NEGADO ACTIVO? ASOCIADO)
```

(* Las especificaciones aparecen en el propio nodo)*

```
               (REMPROP NODO PROPIEDAD))
             ((IDENTIFICACION ID-NEGADO)
```

(* La especificacion aparecen o bien en el nodo real si es un suceso basico, o bien en el nodo asociado. Las modificaciones sobre los nodos reales no son permitidas utilizando estas primitivas de manejo de la estructura NODO)*

```
               (IF (BASICO? NODO)
                   THEN (PRINTOUT T " OPERACION NO PERMITIDA " T)
                 ELSE (REMPROP (NODO.ASOCIADO NODO)
                               PROPIEDAD)))
             ((PARAMETRO TIEMPO)
```

(* Las especificaciones aparecen en el nodo real. Solo son significativas para los suceso basicos)*

```
               (PRINTOUT T " OPERACION NO PERMITIDA " T))
             (REMPROP (NODO.ASOCIADO NODO)
                      PROPIEDAD])
```

(CARACTERISTICA.NODO
```
  [LAMBDA (NODO)                                              (* edited: "10-Nov-87 10:03")
```

(* Obtiene la caracteristica del arbol al que pertenece el nodo)*

```
    (LET ((LETRAS (UNPACK NODO)))
         (PACK (CONS (CAR LETRAS)
                     (CONS (CADR LETRAS])
```

(TODO.NODO
```
  [LAMBDA (NODO)                                              (* edited: "29-Oct-87 03:49")
```

(* Devuelve todas las caracteristicas del nodo especificado en formato de lista de propiedad)*

```
    (GETPROPLIST NODO])
```

(TODAS.PROPIEDADES.NODO
```
  [LAMBDA (NODO)                                              (* edited: "27-Nov-87 17:09")
```

(* Devuelve los nombres de todas las caracteristicas especificadas para ese nodo)*

```
    (LET ((TODAS (PROPNAMES NODO)))
         [if (BASICO? NODO)
             then
```

(* Anade las caracteristicas del componente del sistema asociado al suceso basico)*

```
                (SETQ TODAS (APPEND TODAS (QUOTE (TIEMPO PARAMETRO IDENTIFICACION]
        [if (NOT (NODO.REFERENCIAL? NODO))
            then (SETQ TODAS (APPEND TODAS (PROPNAMES (NODO.ASOCIADO NODO]

        (* * Devuelve el valor de todas)

    TODAS])

(EXISTE.NODO
  [LAMBDA (NODO)                                        (* edited: "23-Nov-87 17:51")

        (* * Comprueba si existe algun nodo con el nombre especificado (no comprueba si pertenece al arbol.
        Esta operacion la realiza la funcion PERTENECE) La existencia del nodo se ha especificado utilizando una propiedad
        : ACTIVO? Esta implementacion es redundante, ya que bastaria comprobar en base al nombre del nodo si se encuentra
        dentro del rango de los nodos creados para el arbol considerado y no ha sido eliminado. Esta comprobacion asi
        planteada es bastante costosa en tiempo, por lo que se prefiere esta implementacion aun a costa de utilizar
        informacion redundante)

    (FGET.NODO NODO (QUOTE ACTIVO?])
)

(* Funciones de generacion de nombres)

(DEFINEQ

(GENERAR.NOMBRE
  [LAMBDA (ARBOL TIPO PADRE)                            (* edited: " 2-Nov-87 02:54")

        (* * Genera el nombre del siguiente nodo correspondiente a un arbol determinado. El argumento tipo especifica si el
        nombre a generar debe ser el de una puerta o el de un suceso basico. Padre es un argumento especial que ha de
        tenerse en cuenta al generar el nombre del nuevo nodo)

    (LET* [(TIPO-BORRADAS (SELECTQ TIPO
                                (PUERTA (QUOTE PUERTAS-BORRADAS))
                                (BASICO (QUOTE BASICOS-BORRADOS))
                                (PRINTOUT T "ERROR EN ARGUMENTO TIPO DE GENERAR.NOMBRE" T)))
           (TIPO-ULTIMO (SELECTQ TIPO
                               (PUERTA (QUOTE ULTIMA-PUERTA))
                               (BASICO (QUOTE ULTIMO-BASICO))
                               NIL))
           (IDENTIF-TIPO (SELECTQ TIPO
                               (PUERTA (QUOTE P))
                               (BASICO (QUOTE B))
                               NIL))
           (NOMB-BORRADOS (FGET.ARBOL ARBOL TIPO-BORRADAS))
           (IDENTIF-ARBOL (FGET.ARBOL ARBOL (QUOTE CARACTERISTICA)))
           (SIGUIENTE (ADD1 (FGET.ARBOL ARBOL TIPO-ULTIMO]
          (if (OR PADRE (NULL NOMB-BORRADOS))
              then

          (* * He de generar un nombre completamente nuevo)

                    (FPUT.ARBOL ARBOL TIPO-ULTIMO SIGUIENTE)
                    (CONSTRUYE IDENTIF-ARBOL IDENTIF-TIPO SIGUIENTE PADRE)
            else

          (* * Utilizo uno de los nombres que han sido previamente descartados)

                  (FPUT.ARBOL ARBOL TIPO-BORRADAS (CDR NOMB-BORRADOS))
                  (CAR NOMB-BORRADOS])

(GENERAR.NOMBRE.ASOCIADO
  [LAMBDA (NODO-ASOCIADO)                               (* edited: "27-Nov-87 17:11")

        (* * Genera el nombre correspondiente a un nodo asociado al especificado como argumento)
```

```
    (LET ((BORRADAS (FGET.NODO NODO-ASOCIADO (QUOTE BORRADOS)))
          (SIGUIENTE))
         (if (NULL BORRADAS)
             then (SETQ SIGUIENTE (ADD1 (OR (FGET.NODO NODO-ASOCIADO (QUOTE
                                                                        NUMERO-ASOCIADOS))
                                            0)))
                  (FPUT.NODO NODO-ASOCIADO (QUOTE NUMERO-ASOCIADOS)
                             SIGUIENTE)

          (* * Anade 1 al numero de asociados al nodo)


          (* * Construye el nombre correspondiente)


                  (CONSTRUYE.ASOCIADO NODO-ASOCIADO SIGUIENTE)
           else (FPUT.NODO NODO-ASOCIADO (QUOTE BORRADOS)
                           (CDR BORRADAS))

          (* * Devuelve el primer elemento de los borrados)


                 (CAR BORRADAS])

(CONSTRUYE
  [LAMBDA (ARBOL NOMBRE NUMERO PADRE)                    (* edited: "10-Nov-87 10:12")

          (* * Genera un atomo correspondiente al nombre de un nodo perteneciente al arbol especificado y al tipo
          especificado)


    (PACK (APPEND (UNPACK ARBOL)
                  (UNPACK NOMBRE)
                  (UNPACK NUMERO)
                  (if PADRE
                      then (CONS (QUOTE %.)
                                 (UNPACK PADRE])

(CONSTRUYE.ASOCIADO
  [LAMBDA (ASOCIADO INDICE)                              (* edited: "23-Nov-87 13:54")

          (* * Genera el nombre de un asociado a partir del nombre del nodo referencial y el numero de asociado
          correspondiente)


    (PACK (APPEND (UNPACK ASOCIADO)
                  (LIST (QUOTE %.))
                  (UNPACK INDICE])

(CONSTRUYE.NOMBRE.NEGADO
  [LAMBDA (NODO)                                         (* edited: "27-Nov-87 19:59")

          (* * Genera un nombre para referirse al negado del nodo especificado)


          (* * Se obtiene anteponiendo una N al nombre del nodo)


    (PACK (CONS (QUOTE N)
                (UNPACK NODO])
)
```

(* Funciones de manipulacion de la estructura (adicion y eliminacion de nodos del arbol) Se definen tres procedimientos distintos de insercion de nodos en el arbol ,correspondientes a cada uno de los diferentes tipos de nodos que puedo encontrar en un arbol de fallos)

```
(DEFINEQ
```

```
(ANADIR.BASICO.NUEVO
  [LAMBDA (ARBOL PADRE COMPONENTE-REAL VALOR-NEGACION)        (* edited: "27-Nov-87 14:00")

          (* * Esta funcion anade un nodo de tipo basico al arbol especificado, inicializandolo con las caracteristicas
          expresadas en la lista de parametros)


          (* * Devuelve el nodo anadido)

    (LET [(NUEVO-NODO (GENERAR.NOMBRE ARBOL (QUOTE BASICO]

          (* * Activacion del nodo)

          (ACTIVAR NUEVO-NODO)

          (* * Padre del nodo)

          (FPUT.NODO NUEVO-NODO (QUOTE PADRE)
                     PADRE)

          (* * Tipo del nodo)

          (FPUT.NODO NUEVO-NODO (QUOTE TIPO)
                     (QUOTE BASICO))

          (* * Suceso basico del sistema asociado)

          (FPUT.NODO NUEVO-NODO (QUOTE REAL)
                     COMPONENTE-REAL)

          (* * Negado ?)

          (FPUT.NODO NUEVO-NODO (QUOTE NEGADO)
                     VALOR-NEGACION)

          (* * Lo anado como hijo del padre especificado)

          (if (EQ PADRE (QUOTE INICIO))
              then                                          (* Nodo inicial del arbol)
                   (FPUT.ARBOL ARBOL (QUOTE INICIAL)
                               NUEVO-NODO)
            else (FADD.NODO PADRE (QUOTE HIJOS.BASICOS)
                            NUEVO-NODO))

          (* * Devuelve el nodo creado)

    NUEVO-NODO])

(ANADIR.PUERTA.NUEVA
  [LAMBDA (ARBOL TIPO-PUERTA PADRE IDENTIFICACION)        (* edited: "27-Nov-87 14:00")

          (* * Anade al arbol un nodo de tipo PUERTA (AND u OR) con las caracteristicas especificadas en la lista de
          argumentos)


          (* * Devuelve el nodo anadido)

    (LET [(NUEVO-NODO (GENERAR.NOMBRE ARBOL (QUOTE PUERTA]

          (* * Activacion del nodo)
```

```
        (ACTIVAR NUEVO-NODO)

        (* * Padre del nodo)

        (FPUT.NODO NUEVO-NODO (QUOTE PADRE)
                   PADRE)

        (* * Tipo del nodo)

        (FPUT.NODO NUEVO-NODO (QUOTE TIPO)
                   TIPO-PUERTA)

        (* * Identificacion)

        (FPUT.NODO NUEVO-NODO (QUOTE IDENTIFICACION)
                   IDENTIFICACION)

        (* * Lo anado como hijo del padre especificado)

        (if (EQ PADRE (QUOTE INICIO))
            then                                        (* Nodo inicial del arbol)
                 (FPUT.ARBOL ARBOL (QUOTE INICIAL)
                             NUEVO-NODO)
          else (FADD.NODO PADRE (QUOTE HIJOS.PUERTAS)
                          NUEVO-NODO))

        (* * Devuelve el nodo creado)

    NUEVO-NODO])

(ANADIR.NODO.ASOCIADO
  [LAMBDA (ARBOL PADRE NODO-ASOC NEGACION)              (* edited: "27-Nov-87 14:11")

        (* * Anade un nodo "asociado" al arbol. Puede estar asociado a un nodo de tipo basico, o bien de tipo puerta.
        Solo en el caso de que sea basico tendra sentido el ultimo argumento)


        (* * Devuelve el nombre del nodo incluido en el arbol)

   (LET* ((TIPO-HIJOS (CLASIFICA.TIPO TIPO))
          (NUEVO-NODO))
         [if (SUBARBOL.REAL? NODO-ASOC)
             then

        (* * El nodo asociado es un subarbol. Anado un nuevo asociado)

                 (SETQ NUEVO-NODO (GENERAR.NOMBRE.ASOCIADO NODO-ASOC))
                 (CREAR.NUEVO.ASOCIADO NUEVO-NODO NODO-ASOC PADRE NEGACION)
           else

        (* * Dado que no es un subarbol he de crear una estructura de Subarbol para el nuevo nodo. Tendra como padres al
        anterior (Subarbol) y al nuevo NODO-PADRE)

               (LET [(NODO-1 (GENERAR.NOMBRE.ASOCIADO NODO-ASOC))
                     (PADRE-ANTERIOR (FGET.NODO NODO-ASOC (QUOTE PADRE]
                    (SETQ NUEVO-NODO (GENERAR.NOMBRE.ASOCIADO NODO-ASOC))

        (* * Creo los nodos asociados)

                 (CREAR.NUEVO.ASOCIADO NODO-1 NODO-ASOC PADRE-ANTERIOR (FGET.NODO
                                                NODO-ASOC
                                                (QUOTE NEGADO)))
                 (CREAR.NUEVO.ASOCIADO NUEVO-NODO NODO-ASOC PADRE NEGACION)
```

```
        (* * Elimino las caracteristicas correspondientes a los asociados del nodo de estructura basica)

                        (MODIFICA.PROPS.ASOCIADO NODO-ASOC ARBOL)

        (* * Modifico en la lista de hijos del subarbol el nombre correspondiente a elemento, sustituyendolo por el nuevo
        nombre generado correspondiente al padre NODO-EXTINGUIR)

                        (FPUT.NODO PADRE-ANTERIOR TIPO-HIJOS (SUBST NODO-1 NODO-ASOC
                                                                    (FGET.NODO PADRE-ANTERIOR
                                                                               TIPO-HIJOS]

        (* * Anado el nuevo nodo como hijo del padre correspondiente)

        (if (EQ PADRE (QUOTE INICIO))
            then                                        (* Nodo inicial del arbol)

        (* * ERROR -
        IMPOSIBLE)

                    (SETQ NUEVO-NODO NIL)
          else (FADD.NODO PADRE (CLASIFICA.TIPO (FGET.NODO NUEVO-NODO (QUOTE TIPO)))
                          NUEVO-NODO))

        (* * Devuelve el nodo creado)

      NUEVO-NODO])

(CREAR.NUEVO.ASOCIADO
  [LAMBDA (NUEVO-NODO NODO-ASOCIADO PADRE VALOR-NEGACION)    (* edited: "10-Nov-87 09:43")

        (* * Crea un nuevo nodo asociado al especificado en la lista de parametros e inicializa los valores caracteristicos
        de este nodo: PADRE ASOCIADO y NEGADO. La primera operacion que realiza es la activacion del nodo)

    (ACTIVAR NUEVO-NODO)
    (FPUT.NODO NUEVO-NODO (QUOTE ASOCIADO)
               NODO-ASOCIADO)
    (FPUT.NODO NUEVO-NODO (QUOTE PADRE)
               PADRE)
    (FPUT.NODO NUEVO-NODO (QUOTE NEGADO)
               VALOR-NEGACION])

(MODIFICA.PROPS.ASOCIADO
  [LAMBDA (NUEVO-NODO-REFERENCIAL)                    (* edited: "10-Nov-87 09:43")

        (* * Elimina las propiedades del nodo que ha pasado a tener estructura "BASICA". Nos referiremos a este tipo de
        nodo con el nombre de REFERENCIALES, entendiendo por esto el conjunto de caracteristicas de un nodo que no dependen
        del padre de ese nodo. Las caracteristicas especificas de los nodos asociados a los referenciales son : PADRE
        ASOCIADO ACTIVO? y NEGADO)

    (FREMOVE.NODO NUEVO-NODO-REFERENCIAL (QUOTE PADRE))
    (FREMOVE.NODO NUEVO-NODO-REFERENCIAL (QUOTE NEGADO))
    (FREMOVE.NODO NUEVO-NODO-REFERENCIAL (QUOTE ASOCIADO])

(SUPRIMIR.NODO
  [LAMBDA (ARBOL NODO AFECTA-COMPONENTES)             (* edited: "27-Nov-87 16:17")

        (* * Elimina el nodo especificado del arbol y libera el espacio de memoria. Solo elimina el nodo, no los sucesores.
        Realiza las comprobaciones necesarias para el tratamiento de subarboles y sucesos basicos iguales con distintos
        nodos padres)

        (* * La operacion de eliminar un nodo se especifica como "DESACTIVAR" el nodo. Esta operacion libera el espacio de
        memoria reservado para este nodo y lo elimina como nodo del arbol.)

    (LET* ((NUMERO-ASOCIADOS (FGET.NODO NODO (QUOTE NUMERO-ASOCIADOS)))
```

```
 (PADRE (FGET.NODO NODO (QUOTE PADRE)))
 (TIPO (FGET.NODO NODO (QUOTE TIPO)))
 (TIPO-HIJO (CLASIFICA.TIPO TIPO)))
(if (NULL NUMERO-ASOCIADOS)
    then
```

(* Es un nodo que no presenta ningun nodo asociado. Basta con eliminar el nodo)*

(* Desactivacion del nodo negado si es basico)*

```
            (PRINTOUT T .TAB 10 "ELIMINO EL NODO" ,, NODO T)
            (INC.LISTA.ELIM ARBOL NODO)
            [if (BASICO? NODO)
                then (DESACTIVAR (NOMBRE.NODO.NEGADO NODO))
                       (if AFECTA-COMPONENTES
                           then (QUITA.ASOCIADO.REAL (NODO.REAL NODO]
            (DESACTIVAR NODO)
  elseif (EQ NUMERO-ASOCIADOS 1)
    then
```

(* Se trata de un nodo que no presenta ningun asociado, pero aparece estructurado como un nodo referencial y un unico asociado (equivalente a un nodo no referencial) Se ha de eliminar tambien en este caso el nodo referencial)*

(* Esta condicion junto con la anterior equivalen a la comprobacion (NOT (SUBARBOL? NODO) Se evaluan los dos condicionantes por separado por razones de eficiencia y claridad))*

(* Desactivacion del nodo negado si es basico)*

```
            (LET ((AUTENTICO.NODO (NODO.ASOCIADO NODO)))
                 (PRINTOUT T .TAB 10 "ELIMINO EL NODO" ,, AUTENTICO.NODO T)
                 (INC.LISTA.ELIM ARBOL AUTENTICO.NODO)
                 [if (BASICO? AUTENTICO.NODO)
                     then (DESACTIVAR (NOMBRE.NODO.NEGADO AUTENTICO.NODO))
                            (if AFECTA-COMPONENTES
                                then (QUITA.ASOCIADO.REAL (NODO.REAL AUTENTICO.NODO]
                 (DESACTIVAR AUTENTICO.NODO)
                 (DESACTIVAR NODO))
  else
```

(* El nodo especificado es un subarbol (es decir, uno de los asociados a un nodo referencial) Elimina exclusivamente el asociado, no el referencial. Modifica convenientemente las caracteristicas del nodo referencial)*

```
          (PRINTOUT T .TAB 10 "ELIMINO EL NODO" ,, NODO ,, "COMO HIJO DE" ,, PADRE T)
          (FPUT.NODO NODO (QUOTE NUMERO-ASOCIADOS)
                     (SUB1 NUMERO-ASOCIADOS))
          (INC.LISTA.ELIM ARBOL NODO)
          (DESACTIVAR NODO))
```

(* Borra el nodo especificado. Se ha de eliminar tambien como hijo del padre correspondiente.*
Se ha preferido incluir esta operacion en esta funcion para asociar asi todas las operaciones relacionadas con la eliminacion de un nodo del arbol. Podria realizarse antes de efectuar la llamada a FREMOVE, resultando algo mas eficiente pero mucho menos transparente)

```
(if (EXISTE.NODO PADRE)
    then
```

(* Se comprueba la posibilidad de que el padre haya sido previamente eliminado)*

```
            (FPUT.NODO PADRE TIPO-HIJO (REMOVE NODO (FGET.NODO PADRE TIPO-HIJO])

(ELIMINAR.NODO
  [LAMBDA (ARBOL NODO AFECTA-COMPONENTES)                    (* edited: "27-Nov-87 16:56")
```

```
        (* * Elimina el nodo especificado, y todos los sucesores afectados del arbol)


        (* * El tercer parametro especifica si al eliminar un suceso basico debe o no verse afectados los componentes)


    (LET ((RESTO-POR-ELIMINAR (LIST NODO))
          (NODO-ELIMINAR)
          (NUEVOS-A-ANADIR))
         (until (NULL RESTO-POR-ELIMINAR)
            do

        (* * Bucle de eliminacion de nodo y sucesores)


                (SETQ NODO-ELIMINAR (CAR RESTO-POR-ELIMINAR))
                [SETQ NUEVOS-A-ANADIR (if (SUBARBOL? NODO-ELIMINAR)
                                          then NIL
                                        else (APPEND (FGET.NODO NODO-ELIMINAR (QUOTE
                                                                               HIJOS.PUERTAS))
                                                     (FGET.NODO NODO-ELIMINAR (QUOTE
                                                                               HIJOS.BASICOS]

        (* * Elimina el nodo)


                (SUPRIMIR.NODO ARBOL NODO-ELIMINAR AFECTA-COMPONENTES)

        (* * Actualiza el valor de RESTO-POR-ELIMINAR anadiendo todos los hijos del nodo eliminado.
        Si es un subarbol no se anade ninguno. Basta con mantener la estructura basica del subarbol eliminando el nodo como
        asociado a ese subarbol. No se sigue eliminando por esta rama. Estas consideraciones se hacen al calcular el valor
        de NUEVOS-A-ANADIR)


                (SETQ RESTO-POR-ELIMINAR (APPEND NUEVOS-A-ANADIR (CDR RESTO-POR-ELIMINAR]))

(INC.LISTA.ELIM
  [LAMBDA (ARBOL NODO)                                    (* edited: "21-Nov-87 11:42")

        (* * Anade a la propiedad correspondiente del arbol los nodos que han sido eliminados)


    (if (SUBARBOL? NODO)
        then

        (* * Incluyo en la propiedad correspondiente del nodo asociado el nodo borrado)


              (FADD.NODO NODO (QUOTE BORRADOS)
                         NODO)
      else (LET [(TIPO-ELIM (SELECTQ (FGET.NODO NODO (QUOTE TIPO))
                                     ((AND OR)
                                       (QUOTE PUERTAS-BORRADAS))
                                     (BASICO (QUOTE BASICOS-BORRADOS))
                                     (PRINTOUT T T T .TAB 5 "REVISAR TIPO DEL NODO " NODO]
             (FADD.ARBOL ARBOL TIPO-ELIM (NODO.ASOCIADO NODO])
)
(DECLARE: DONTCOPY
  (FILEMAP (NIL (1681 3753 (FGET.ARBOL 1691 . 1916) (FPUT.ARBOL 1918 . 2198) (FADD.ARBOL 2200 . 2624) (
FREMOVE.ARBOL 2626 . 2891) (TODO.ARBOL 2893 . 3159) (PERTENECE 3161 . 3751)) (3852 11145 (ACTIVAR 3862
 . 4354) (DESACTIVAR 4356 . 4583) (FGET.NODO 4585 . 5807) (FPUT.NODO 5809 . 7038) (FADD.NODO 7040 .
7996) (FREMOVE.NODO 7998 . 9072) (CARACTERISTICA.NODO 9074 . 9394) (TODO.NODO 9396 . 9646) (
TODAS.PROPIEDADES.NODO 9648 . 10358) (EXISTE.NODO 10360 . 11143)) (11193 14828 (GENERAR.NOMBRE 11203
 . 12727) (GENERAR.NOMBRE.ASOCIADO 12729 . 13662) (CONSTRUYE 13664 . 14104) (CONSTRUYE.ASOCIADO 14106
 . 14479) (CONSTRUYE.NOMBRE.NEGADO 14481 . 14826)) (15108 27167 (ANADIR.BASICO.NUEVO 15118 . 16487) (
ANADIR.PUERTA.NUEVA 16489 . 17701) (ANADIR.NODO.ASOCIADO 17703 . 20088) (CREAR.NUEVO.ASOCIADO 20090 .
20674) (MODIFICA.PROPS.ASOCIADO 20676 . 21400) (SUPRIMIR.NODO 21402 . 24975) (ELIMINAR.NODO 24977 .
26424) (INC.LISTA.ELIM 26426 . 27165)))))
STOP
```

```
{DSK}<LISPFILES>ARBOLES>EST-ARBOL-BUSQUEDA.LSP;1     30-Nov-87 11:52:11                Page 1

(FILECREATED "30-Nov-87 11:52:11" {DSK}<LISPFILES>ARBOLES>EST-ARBOL-BUSQUEDA.LSP;1 4889

      changes to:  (VARS EST-ARBOL-BUSQUEDACOMS))


(PRETTYCOMPRINT EST-ARBOL-BUSQUEDACOMS)

(RPAQQ EST-ARBOL-BUSQUEDACOMS ((* Procedimientos de manipulacion de la estructura de arbol de
                                  busqueda)
                               (* Funciones de manejo de la estructura ARBOL y los NODOS del mismo
                                  (PRIMITIVAS BASICAS)
                                  ,)
                               (FNS FGET.ARBOL FPUT.ARBOL FADD.ARBOL FREMOVE.ARBOL TODO.ARBOL
                                    PERTENECE)
                               (* Funciones de manejo de la informacion de los nodos
                                  (especifico para arboles de busqueda))
                               (FNS ACTIVAR DESACTIVAR FGET.BUSQUEDA FPUT.BUSQUEDA FADD.BUSQUEDA
                                    FREMOVE.BUSQUEDA)))
```

(* Procedimientos de manipulacion de la estructura de arbol de busqueda)

(* Funciones de manejo de la estructura ARBOL y los NODOS del mismo (PRIMITIVAS BASICAS) ,)

```
(DEFINEQ

(FGET.ARBOL
  [LAMBDA (ARBOL PROPIEDAD)                                    (* edited: "10-Nov-87 10:08")

        (* * Obtiene la propiedad especificada de la estructura arbol)

    (GETPROP ARBOL PROPIEDAD])

(FPUT.ARBOL
  [LAMBDA (ARBOL CARACTERISTICA VALOR)                         (* edited: "10-Nov-87 10:08")

        (* * Incluye como valor para la propiedad del arbol especificado el que aparece en la lista de argumentos)

    (PUTPROP ARBOL CARACTERISTICA VALOR])

(FADD.ARBOL
  [LAMBDA (ARBOL PROPIEDAD VALOR)                              (* edited: "27-Nov-87 17:05")

        (* * Anade un valor determinado a los ya existentes en alguna de las caracteristicas del arbol especificado)

    (COND
      ((NULL VALOR))
      ((ATOM VALOR)
        (ADDPROP ARBOL PROPIEDAD VALOR))
      (T (PUTPROP ARBOL PROPIEDAD (APPEND VALOR (GETPROP ARBOL PROPIEDAD])

(FREMOVE.ARBOL
  [LAMBDA (ARBOL PROPIEDAD)                                    (* edited: "10-Nov-87 10:08")

        (* * Elimina la propiedad especificada del arbol (propiedad especifica del arbol, no de sus nodos))

    (REMPROP ARBOL PROPIEDAD])

(TODO.ARBOL
  [LAMBDA (ARBOL)                                              (* edited: "29-Oct-87 03:48")

        (* * Obtiene todas las caracteristicas del arbol especificado. Las devuelve en formato de lista de propiedad)

    (GETPROPLIST ARBOL])
```

```
(PERTENECE
  [LAMBDA (NODO ARBOL)                                    (* edited: "27-Nov-87 17:10")

        (* * La funcion comprueba la pertenencia del nodo al arbol devolviendo un valor T o NIL)


    (LET ((CARACT-ARBOL (FGET.ARBOL ARBOL (QUOTE CARACTERISTICA)))
          (CARACT-NODO (CARACTERISTICA.NODO NODO)))

        (* * Se considera que pertenece si el identificador del arbol coincide con el del nodo, (previamente Se comprueba
        que el nodo especificado existe))


         (AND (EXISTE.NODO NODO)
              (EQ CARACT-ARBOL CARACT-NODO])
)
```

(* Funciones de manejo de la informacion de los nodos (especifico para arboles de busqueda))

```
(DEFINEQ

(ACTIVAR
  [LAMBDA (NODO)                                          (* edited: "16-Nov-87 11:43")

        (* * Esta funcion realiza la activacion del nodo. Esta operacion, de acuerdo con la implementacion adoptada para la
        estructura ARBOL consiste simplemente en especificar el valor T para la propiedad del nodo ACTIVO)


    (DESACTIVAR NODO)                                     (* Inicializacion del nodo)
    (FPUT.NODO NODO (QUOTE ACTIVO?)
               T])

(DESACTIVAR
  [LAMBDA (NODO)                                          (* edited: "27-Nov-87 17:05")

        (* * Esta funcion elimina el nodo del arbol. ( = Desactiva el nodo))


    (SETPROPLIST NODO NIL])

(FGET.BUSQUEDA
  [LAMBDA (NODO PROPIEDAD)                                (* edited: "27-Nov-87 14:26")

        (* * Obtiene la informacion especificada de un nodo del arbol de busqueda pasado como argumento a la funcion)


    (GETPROP NODO PROPIEDAD])

(FPUT.BUSQUEDA
  [LAMBDA (NODO PROPIEDAD VALOR)                          (* edited: "27-Nov-87 14:28")

        (* * Incluye en el nodo especificado la informacion pasada como argumento)


    (PUTPROP NODO PROPIEDAD VALOR])

(FADD.BUSQUEDA
  [LAMBDA (NODO PROPIEDAD VALOR)                          (* edited: "27-Nov-87 14:32")

        (* * Anade a la informacion ya presente en el nodo sobre la propiedad especificada, el nuevo valor propuesto)


    (ADDPROP NODO PROPIEDAD VALOR])

(FREMOVE.BUSQUEDA
  [LAMBDA (NODO PROPIEDAD)                                (* edited: "27-Nov-87 14:30")

        (* * Elimina del nodo especificado la informacion pasada como argumento a la funcion)


    (REMPROP NODO PROPIEDAD])
)
```

```
(DECLARE: DONTCOPY
  (FILEMAP (NIL (900 2972 (FGET.ARBOL 910 . 1135) (FPUT.ARBOL 1137 . 1417) (FADD.ARBOL 1419 . 1843) (
FREMOVE.ARBOL 1845 . 2110) (TODO.ARBOL 2112 . 2378) (PERTENECE 2380 . 2970)) (3074 4867 (ACTIVAR 3084
 . 3576) (DESACTIVAR 3578 . 3805) (FGET.BUSQUEDA 3807 . 4081) (FPUT.BUSQUEDA 4083 . 4328) (
FADD.BUSQUEDA 4330 . 4610) (FREMOVE.BUSQUEDA 4612 . 4865)))))
STOP
```

```
{DSK}<LISPFILES>ARBOLES>FUNCIONES-ARBOL-FALLOS.LSP;17     30-Nov-87 11:52:22                Page 1

(FILECREATED "30-Nov-87 11:52:22" {DSK}<LISPFILES>ARBOLES>FUNCIONES-ARBOL-FALLOS.LSP;17 1205

      changes to:  (VARS FUNCIONES-ARBOL-FALLOSCOMS)
                   (FNS NEGADO.FORMATEADO? NODO.REAL GENERAR.TODOS.PADRES FORMATEA ESPECIFICA.ESTADO)

      previous date: "23-Nov-87 19:07:14" {DSK}<LISPFILES>ARBOLES>FUNCIONES-ARBOL-FALLOS.LSP;1)


(PRETTYCOMPRINT FUNCIONES-ARBOL-FALLOSCOMS)

(RPAQQ FUNCIONES-ARBOL-FALLOSCOMS ((* Funciones para facilitar la utilizacion de las primitivas de
                                      manejo de la estructura ARBOL. Definicion de MACROS para
                                      aumentar la legilibilidad del programa)
                                   (FNS BASICO? NEGADO? NEGADO.FORMATEADO? ULTIMO.HIJO? SUBARBOL?
                                      SUBARBOL.REAL? NODO.REFERENCIAL? SUBARBOL-NO-ANALIZADO?)
                                   (* Funciones de ayuda. Permiten una mayor legilibilidad del
                                      programa Son esencialmente funciones para obtener informacion
                                      del arbol)
                                   (FNS NODO.REAL NODO.ASOCIADO NOMBRE.NODO.NEGADO NEGACION
                                      FORMATEA CLASIFICA.TIPO CLASIFICA.TIPO.P/B MIEMBRO.NEGADO
                                      MIEMBRO CAMBIAR.NOMBRES GENERAR.TODOS.PADRES)
                                   (* Funciones relacionadas con el estado en que se encuentra un
                                      nodo)
                                   (FNS ESTADO.REAL INVIERTE.SI.NEGADO INVIERTE.ESTADO
                                      ESPECIFICA.ESTADO)))
```

(* Funciones para facilitar la utilizacion de las primitivas de manejo de la estructura ARBOL. Definicion de MACROS para aumentar la legilibilidad del programa)

```
(DEFINEQ

(BASICO?
  [LAMBDA (NODO)                                        (* edited: "11-Nov-87 11:37")

        (* * Comprueba si un nodo es de tipo basico)


    (EQ (QUOTE BASICO)
        (FGET.NODO NODO (QUOTE TIPO])

(NEGADO?
  [LAMBDA (SUCESO)                                      (* edited: "10-Nov-87 10:00")

        (* * Comprueba si el suceso basico especificado esta o no negado)


    (FGET.NODO SUCESO (QUOTE NEGADO])

(NEGADO.FORMATEADO?
  [LAMBDA (NODO)                                        (* edited: "27-Nov-87 20:02")

        (* * Comprueba si un nodo esta negado. El nombre del nodo viene dado siguiendo el convenio de anteponer una N si el
        nodo esta negado)


    (EQ NODO (NOMBRE.NODO.NEGADO NODO])

(ULTIMO.HIJO?
  [LAMBDA (NODO)                                        (* edited: "23-Nov-87 17:52")

        (* * Devuelve un valor Booleano especificando si el nodo especificado es el ultimo hijo del padre)


        (* Si el ultimo hijo del padre coincide con el pasado como argumento devolvera el valor T. Al llegar al nodo
        inicial del arbol, este tiene declarado como padre el nombre simbolico "INICIO" que no presenta ningun hijo, con lo
        cual queda asegurado el final de la propagacion del calculo de minimos)


    (EQ NODO (CAR (LAST (FGET.NODO (FGET.NODO NODO (QUOTE PADRE))
                                   (QUOTE HIJOS.PUERTAS])
```

```
(SUBARBOL?
  [LAMBDA (NODO)                                    (* edited: "10-Nov-87 10:00")

        (* * Devuelve T o NIL dependiendo de si el nodo pasado como argumento es o no un Subarbol)


    (LET [(NUM-ASOCIADOS (FGET.NODO NODO (QUOTE NUMERO-ASOCIADOS]
         (if NUM-ASOCIADOS
             then (GREATERP NUM-ASOCIADOS 1)
           else NIL])

(SUBARBOL.REAL?
  [LAMBDA (NODO)                                    (* edited: "20-Nov-87 15:35")

        (* * Devuelve T o NIL dependiendo de si el nodo pasado como argumento es o no un Subarbol (Tiene algun nodo
        asociado, aun cuando solo sea uno))


    (LET [(NUM-ASOCIADOS (FGET.NODO NODO (QUOTE NUMERO-ASOCIADOS]
         (if NUM-ASOCIADOS
             then (GEQ NUM-ASOCIADOS 1)
           else NIL])

(NODO.REFERENCIAL?
  [LAMBDA (NODO)                                    (* edited: "11-Nov-87 09:30")

        (* * Comprueba si el nodo propuesto es un nodo referencial (coincide con su asociado) o bien es un asociado a un
        subarbol)


    (EQ NODO (NODO.ASOCIADO NODO])

(SUBARBOL-NO-ANALIZADO?
  [LAMBDA (NODO)                                    (* edited: " 4-Nov-87 11:09")

        (* * Comprueba si el nodo especificado es un subarbol del arbol de fallos que ya ha sido analizado.
        Devuelve T si aun no ha sido analizado)


    (NOT (FGET.NODO NODO (QUOTE ANALIZADO?])
)


(* Funciones de ayuda. Permiten una mayor legilibilidad del programa Son esencialmente
funciones para obtener informacion del arbol)

(DEFINEQ

(NODO.REAL
  [LAMBDA (NODO)                                    (* edited: "27-Nov-87 20:03")

        (* * Esta funcion se aplica a los sucesos basicos. Si el nodo es de tipo Puerta el nodo real se considera que
        coincide con el asociado (Puede plantearse tambien este concepto para las puertas si se considera util) En el caso
        de los sucesos basicos nos permite asociar el nodo que aparece en el arbol de fallos con las especificaciones
        reales de un determinado componente del sistema, (que podra aparecer por tanto en diversos arboles de fallo))


        (* * Esta asociacion puede hacerse o bien en cada nodo (repetido para todos los nodos asociados) o bien unicamente
        en el nodo referencial. Se ha elegido esta ultima opcion)


    (FGET.NODO NODO (QUOTE REAL])

(NODO.ASOCIADO
  [LAMBDA (NODO)                                    (* edited: "10-Nov-87 10:10")

        (* * Obtiene el nombre del nodo asociado. LLamamos nodo asociado a aquel con las mismas caracteristicas pero
        distinto padre. El tratamiento es analogo para subarboles o sucesos basicos)


        (* * Aparece otra version de esta funcion (NODO.ASOCIADO.3) en la que se calcula el nodo asociado en base al nombre
```

generado para ese nodo. Dada la frecuencia con la que se utiliza esta funcion se ha preferido implementarla como una propiedad del nodo. Esta informacion es redundante pero mucho mas rapido que calcular los PACK y UNPACK del nombre del nodo para calcular el asociado. Si el valor de la propiedad es NIL indica que el nodo asociado es el mismo)

```
    (OR (GETPROP NODO (QUOTE ASOCIADO))
        NODO])
```

(NOMBRE.NODO.NEGADO
[LAMBDA (NODO) *(* edited: "20-Nov-87 15:54")*

(* Devuelve el nombre utilizado para referirse al negado. Solo tiene sentido utilizarlo para basicos)*

```
    (FGET.NODO NODO (QUOTE NOMBRE-NEGADO])
```

(NEGACION
[LAMBDA (NODO) *(* edited: "21-Nov-87 11:27")*

(* Niega un nodo. La negacion se realiza sobre un nodo cuyo nombre sigue el convenio de anteponer una N si aparece negado)*

```
    (LET ((NOM-NEG (NOMBRE.NODO.NEGADO NODO)))
         (if (EQ NODO NOM-NEG)
             then (NODO.ASOCIADO NODO)
           else NOM-NEG])
```

(FORMATEA
[LAMBDA (NODO) *(* edited: "24-Nov-87 09:56")*

(* Modifica el formato del nodo anteporniendo una N si esta negado, y devolviendo el referencial si se trata de una asociado)*

```
    (if (NEGADO? NODO)
        then (NOMBRE.NODO.NEGADO NODO)
      else (NODO.ASOCIADO NODO])
```

(CLASIFICA.TIPO
[LAMBDA (TIPO) *(* edited: " 2-Nov-87 02:53")*

(* Devuelve la especificacion del tipo de hijos al que hacemos referencia)*

```
    (SELECTQ TIPO
             ((AND OR)
               (QUOTE HIJOS.PUERTAS))
             (BASICO (QUOTE HIJOS.BASICOS))
             (PROMPTPRINT "!!! REVISAR TIPO DEL NODO . CLASIFICA.NODO HA FALLADO !!! "])
```

(CLASIFICA.TIPO.P/B
[LAMBDA (TIPO) *(* edited: " 2-Nov-87 02:53")*

(* Devuelve la especificacion del tipo de hijos al que hacemos referencia)*

```
    (SELECTQ TIPO
             ((AND OR)
               (QUOTE PUERTA))
             (BASICO (QUOTE BASICO))
             (PROMPTPRINT "!!! REVISAR TIPO DEL NODO . CLASIFICA.NODO HA FALLADO !!! "])
```

(MIEMBRO.NEGADO
[LAMBDA (NODO LISTA-CONSIDERA) *(* edited: "21-Nov-87 11:19")*

(* Comprueba si el negado del nodo asociado al basico que aparece como argumento pertenece a la lista especificada como segundo argumento)*

```
    (if (BASICO? NODO)
        then (MEMB (NEGACION NODO)
                   LISTA-CONSIDERA)
```

```
      else

           (* * No tiene sentido para las puertas)


             NIL])

(MIEMBRO
  [LAMBDA (NODO LISTA-CONSIDERA)                          (* edited: "23-Nov-87 18:00")

           (* * Comprueba si el nodo asociado al suceso pertenece a la lista especificada como argumento)


    (MEMB (NODO.ASOCIADO NODO)
          LISTA-CONSIDERA)
    (PRINTOUT T " CUIDADO UTILIZA MIEMBRO CON : " NODO , , LISTA-CONSIDERA T])

(CAMBIAR.NOMBRES
  [LAMBDA (LISTA-NOMBRES)                                 (* edited: "20-Nov-87 16:01")

           (* * Modifica la lista de nombres incluyendo los nombres de los nodos asociados y anadiendo el prefijo N si el
           suceso basico aparece negado)


    (for ELEMENTO in LISTA-NOMBRES collect (if (NEGADO? ELEMENTO)
                                               then (NOMBRE.NODO.NEGADO ELEMENTO)
                                             else (NODO.ASOCIADO ELEMENTO])

(GENERAR.TODOS.PADRES
  [LAMBDA (NODO)                                          (* edited: "27-Nov-87 20:04")

           (* * Genera todos los padres correspondientes al nodo especificado (al hablar de todos se refiere a los de todos
           los nodos asociados))


    (if (SUBARBOL.REAL? NODO)
        then (LET ((ASOCIADO (NODO.ASOCIADO NODO))
                   (NUMERO (FGET.NODO NODO (QUOTE NUMERO-ASOCIADOS)))
                   (CONTADOR 0)
                   (TODOS)
                   (INDICE 1)
                   (NUEVO-ASOCIADO))
                  (until (EQP CONTADOR NUMERO)
                     do (SETQ NUEVO-ASOCIADO (CONSTRUYE.ASOCIADO ASOCIADO INDICE))
                        [if (EXISTE.NODO NUEVO-ASOCIADO)
                            then (SETQ TODOS (CONS (FGET.NODO NUEVO-ASOCIADO
                                                              (QUOTE PADRE))
                                                   TODOS)
                                       (SETQ CONTADOR (ADD1 CONTADOR]
                        (SETQ INDICE (ADD1 INDICE)))

           (* * Devuelve todos los padres)


             TODOS)
      else

           (* * Devuelve el padre del nodo referencial)


        (LIST (FGET.NODO (NODO.ASOCIADO NODO)
                         (QUOTE PADRE])
)


(* Funciones relacionadas con el estado en que se encuentra un nodo)

(DEFINEQ

(ESTADO.REAL
  [LAMBDA (NODO)                                          (* edited: "10-Nov-87 14:34")

           (* * Devuelve el estado real del nodo teniendo en cuenta la posibilidad de que el nodo se encuentre negado)
```

```
    (INVIERTE.SI.NEGADO NODO (FGET.NODO NODO (QUOTE ESTADO])

(INVIERTE.SI.NEGADO
  [LAMBDA (NODO ESTADO)                                    (* edited: "23-Nov-87 13:59")

          (* * Comprueba si el nodo especificado esta negado. Si lo esta, la interpretacion del estado propuesto es la
          inversa)


    (if (NEGADO.FORMATEADO? NODO)
        then (INVIERTE.ESTADO ESTADO)
      else (ESPECIFICA.ESTADO ESTADO])

(INVIERTE.ESTADO
  [LAMBDA (ESTADO)                                         (* edited: "23-Nov-87 13:51")

          (* * Funcion utilizada para invertir un estado)


    (SELECTQ ESTADO
             (FALLO (QUOTE NO-FALLO))
             (NO-FALLO (QUOTE FALLO))
             (QUOTE DESCONOCIDO])

(ESPECIFICA.ESTADO
  [LAMBDA (ESTADO)                                         (* edited: "23-Nov-87 13:49")

          (* * Devuelve DESCONOCIDO si no hay informacion del estado en que se encuentra el nodo)


    (if (NULL ESTADO)
        then (QUOTE DESCONOCIDO)
      else ESTADO])
)
(DECLARE: DONTCOPY
  (FILEMAP (NIL (1421 4416 (BASICO? 1431 . 1672) (NEGADO? 1674 . 1910) (NEGADO.FORMATEADO? 1912 . 2236)
(ULTIMO.HIJO? 2238 . 2920) (SUBARBOL? 2922 . 3309) (SUBARBOL.REAL? 3311 . 3759) (NODO.REFERENCIAL?
3761 . 4069) (SUBARBOL-NO-ANALIZADO? 4071 . 4414)) (4559 10620 (NODO.REAL 4569 . 5404) (NODO.ASOCIADO
5406 . 6300) (NOMBRE.NODO.NEGADO 6302 . 6592) (NEGACION 6594 . 7005) (FORMATEA 7007 . 7380) (
CLASIFICA.TIPO 7382 . 7799) (CLASIFICA.TIPO.P/B 7801 . 8208) (MIEMBRO.NEGADO 8210 . 8672) (MIEMBRO
8674 . 9033) (CAMBIAR.NOMBRES 9035 . 9504) (GENERAR.TODOS.PADRES 9506 . 10618)) (10698 12035 (
ESTADO.REAL 10708 . 11017) (INVIERTE.SI.NEGADO 11019 . 11408) (INVIERTE.ESTADO 11410 . 11721) (
ESPECIFICA.ESTADO 11723 . 12033)))))
STOP
```

```
{DSK}<LISPFILES>ARBOLES>ARBOL-FALLOS.LSP;16     30-Nov-87 19:05:28              Page 1

(FILECREATED "30-Nov-87 19:05:29" {DSK}<LISPFILES>ARBOLES>ARBOL-FALLOS.LSP;16 16816

      changes to:  (VARS ARBOL-FALLOSCOMS)
                   (FNS ELIMINAR.NODO CONSTRUIR.ARBOL SALVAR.ARBOL ANADIR.BASICO ANADIR.NUEVO.NODO
                        ANADIR.ASOCIADO SUPRIMIR.NODO ANADIR.NODO ANADIR.NODO.INICIAL)

      previous date: "23-Nov-87 19:27:39" {DSK}<LISPFILES>ARBOLES>ARBOL-FALLOS.LSP;1)


(PRETTYCOMPRINT ARBOL-FALLOSCOMS)

(RPAQQ ARBOL-FALLOSCOMS ((* Procedimientos de creacion y eliminacion (manipulacion como estructura
                                                                      completa)
                            de los arboles de fallo. Todos estos procedimientos estan orientados
                            especialmente a un dialogo interactivo con el usuario para obtener la
                            informacion necesaria)
                           (* ARBOLES DE FALLO)
                           (FNS CONSTRUIR.ARBOL INICIAR.ARBOL MODIFICA.ARBOL ELIMINAR.ARBOL
                                SALVAR.ARBOL CARGAR.ARBOL GENERAR.NOMBRE.FICHERO)
                           (FNS ANADIR.NODO CONSULTA.DATOS.NODO CAMBIA.DATOS.NODO)))
```

(* Procedimientos de creacion y eliminacion (manipulacion como estructura completa) de los arboles de fallo. Todos estos procedimientos estan orientados especialmente a un dialogo interactivo con el usuario para obtener la informacion necesaria)

(* ARBOLES DE FALLO)

```
(DEFINEQ

(CONSTRUIR.ARBOL
  [LAMBDA NIL                                          (* edited: "27-Nov-87 17:01")

        (* * Construye una estructura de tipo arbol a partir de las especificaciones dadas por el usuario)


    (LET* ((SALIDA (CREATEW (QUOTE (371 303 593 466))
                            " PROCESO DE CREACION DEL ARBOL" 4 T))
           (ANTERIOR (TTYDISPLAYSTREAM SALIDA))
           (ARBOL-FALLOS)
           (NODO-ANADIDO)
           (FIN NIL))
          (OPENW T)
          (CLEARW T)
          (INICIALIZA.ASOCIADOS.REALES)

          (* * Borra el activo actualmente)


          (* * ARBOL-FALLOS es la variable que indica el arbol de fallos activo actualmente)


          (* * Inicializacion de los parametros del arbol)


          (PRINTOUT T T T 15 " CARACTERISTICAS DEL ARBOL " T)
          [SETQ ARBOL-FALLOS (INICIAR.ARBOL (PREGUNTA " NOMBRE DEL NUEVO ARBOL ====>"
                                                      (QUOTE FUNCION)
                                                      (QUOTE (TRUE)))
                                            (PREGUNTA
                                              " CARACTERISTICA DEL ARBOL ( 2 LETRAS ) ====> "
                                                      (QUOTE FUNCION)
                                                      (QUOTE (DOS.LETRAS?)))
                                            (PREGUNTA "IDENTIFICACION DEL ARBOL ===>"
                                                      (QUOTE FUNCION-O-NIL)
                                                      (QUOTE (STRINGP]
```

```
        (* * Definicion de los nodos del arbol. Comienza con el nodo inicial. Se exige que sea de tipo PUERTA el nodo
        inicial)


        (PRINTOUT T T T .TAB 10 "ESPECIFICACIONES DEL NODO INICIAL" T T)
        [SETQ NODO-ANADIDO (ANADIR.PUERTA.NUEVA ARBOL-FALLOS (PREGUNTA
                                                              "TIPO DEL NODO INICIAL AND/OR ===>"
                                                                    (QUOTE PERTENENCIA)
                                                                    (QUOTE (AND OR)))
                                                    (QUOTE INICIO)
                                                    (PREGUNTA
                                                          "IDENTIFICACION DEL NUEVO NODO ===>"
                                                              (QUOTE FUNCION-O-NIL)
                                                              (QUOTE (STRINGP]
        (if NODO-ANADIDO
            then                                              (* Todo ha ido bien)
                 (PRINTOUT T T T .TAB 20 "NUEVO NODO ANADIDO : " NODO-ANADIDO T)
          else                                                (* Problemas de algun tipo)
                (PRINTOUT T T T .2 "ERROR EN ADICION DEL NODO. NO ES POSIBLE CONTINUAR " T)
                (SETQ FIN T))
        (until FIN
           do

        (* * Bucle de adicion de nodos al arbol)


                (CLEARW T)
                (PRINTOUT T T .TAB 15 "NODO INICIAL : " (FGET.ARBOL ARBOL-FALLOS (QUOTE INICIAL)
                                                                    ))
                (PRINTOUT T T T .TAB 10 "ESPECIFICACIONES DEL NUEVO NODO" T)
                (SETQ NODO-ANADIDO (ANADIR.NODO ARBOL-FALLOS))
                (if NODO-ANADIDO
                    then                                      (* Todo ha ido bien)
                         (PRINTOUT T T T .TAB 20 "NUEVO NODO ANADIDO : " NODO-ANADIDO T)
                         (CLEARBUF T T)
                         (CONTROL T)
                         (PRINTOUT T T T .TAB 10 "PULSA INTRO PARA CONTINUAR. F PARA FINALIZAR ")
                         (SETQ FIN (EQ (READC)
                                       (QUOTE F)))
                         (CONTROL NIL)
                  else (PRINTOUT T T T .2 "ERROR EN ADICION DEL NODO. NO ES POSIBLE CONTINUAR " T)
                       (SETQ FIN T)))

        (* * Salva el arbol al fichero correspondiente)


        (SALVAR.ARBOL ARBOL-FALLOS)
        (CLOSEW (TTYDISPLAYSTREAM ANTERIOR))

        (* * Devuelve el nombre del arbol creado)


    ARBOL-FALLOS])

(INICIAR.ARBOL
  [LAMBDA (ARBOL CARACTERISTICA IDENTIFICACION)                (* edited: "27-Nov-87 12:05")

        (* * Esta funcion inicializa los parametros fundamentales del arbol. Devuelve el nombre del arbol creado)


    (FPUT.ARBOL ARBOL (QUOTE CARACTERISTICA)
                CARACTERISTICA)
    (FPUT.ARBOL ARBOL (QUOTE IDENTIFICACION)
                IDENTIFICACION)
    (FPUT.ARBOL ARBOL (QUOTE ULTIMA-PUERTA)
                0)
    (FPUT.ARBOL ARBOL (QUOTE ULTIMO-BASICO)
                0)
    (FPUT.ARBOL ARBOL (QUOTE TIPO)
                (QUOTE NORMAL))
    (FPUT.ARBOL ARBOL (QUOTE PUERTAS-BORRADAS)
                NIL)
    (FPUT.ARBOL ARBOL (QUOTE BASICOS-BORRADOS)
                NIL)
```

```
    (FPUT.ARBOL ARBOL (QUOTE INICIAL)
                NIL)

        (* * Devuelve el nombre del arbol creado)


    ARBOL])

(MODIFICA.ARBOL
  [LAMBDA (ARBOL)                                        (* edited: "23-Nov-87 17:27")
    (LET ((MENU-PETICIONES (create MENU
                                   TITLE ← " TIPO DE OPERACION A REALIZAR "
                                   ITEMS ←(QUOTE ((" ANADIR NUEVO " (QUOTE ANADIR)

                                                  " INCLUYE UN NUEVO COMPONENTE DEL SISTEMA ")
                                                  (" ELIMINAR " (QUOTE ELIMINAR)

                                                " ELIMINAR LOS DATOS CORESPONDIENTE A UNA COMPONENTE ")
                                                  ("CONSULTA" (QUOTE CONSULTA)

                                                "OBTIENE LA INFORMACION SOBRE LOS DATOS DE UNA COMPONENTE ")
                                                  (" MODIFICACION" (QUOTE MODIFICA)

                                                      "MODIFICA LOS DATOS DE LA COMPONENTE ")
                                                  (" FIN DE SESION " (QUOTE FINALIZAR)

                                                           " FINALIZA LA SESION DE AYUDA ")))
                                   CENTERFLG ← T
                                   ITEMHEIGHT ← 40
                                   ITEMWIDTH ← 300
                                   MENUBORDERSIZE ← 3))
          (FIN))
         (until FIN do (SELECTQ (MENU MENU-PETICIONES)
                                  (ANADIR (ANADIR.NODO ARBOL))
                                  (ELIMINAR (ELIMINAR.NODO ARBOL (PREGUNTA
                                                                   " NODO A ELIMINAR ====> "
                                                                   (QUOTE FUNCION)
                                                                   (QUOTE (PERTENECE ARBOL)))
                                                                 T))
                                  [CONSULTA (CONSULTA.DATOS.NODO (PREGUNTA
                                                                   " NODO A CONSULTAR =====> "
                                                                   (QUOTE FUNCION)
                                                                   (QUOTE (PERTENECE ARBOL]
                                  [MODIFICA (CAMBIA.DATOS.NODO ARBOL
                                                               (PREGUNTA
                                                                 " NODO A MODIFICAR ====> "
                                                                         (QUOTE FUNCION)
                                                                         (QUOTE
                                                                           (PERTENECE
                                                                             ARBOL]
                                  (FINALIZAR (SETQ FIN T))
                                  (RINGBELLS])

(ELIMINAR.ARBOL
  [LAMBDA (ARBOL)                                        (* edited: "13-Nov-87 17:31")

        (* * Elimina de memoria principal la informacion correspondiente al arbol especificado)


    (LET [(INICIAL (FGET.ARBOL ARBOL (QUOTE INICIAL]
         (if INICIAL
             then (ELIMINAR.NODO ARBOL INICIAL)
                  (DESACTIVAR ARBOL])

(SALVAR.ARBOL
  [LAMBDA (ARBOL-FALLOS)                                 (* edited: "27-Nov-87 17:02")

        (* * Salva la informacion contenida en el arbol propuesto. Recorre todos los nodos del arbol en propundidad)


    (LET [(TIPO-ARBOL (FGET.ARBOL ARBOL-FALLOS (QUOTE TIPO]
         (if [AND ARBOL-FALLOS (MEMB TIPO-ARBOL (QUOTE (NORMAL DEPURADO]
             then                                        (* Arbol de formato correcto)
                  (LET ((NOMBRE-FICHERO (OPENSTREAM (GENERAR.NOMBRE.FICHERO ARBOL-FALLOS
```

```
                                                                TIPO-ARBOL)
                                          (QUOTE OUTPUT)
                                          (QUOTE OLD/NEW)))
                    [RESTO-POR-CONSIDERAR (LIST (FGET.ARBOL ARBOL-FALLOS (QUOTE INICIAL]
                    (NODO-ACTUAL)
                    (NUEVOS-A-ANADIR))

(* * Especificaciones del arbol)

                    (PRINT (LIST ARBOL-FALLOS (TODO.ARBOL ARBOL-FALLOS))
                           NOMBRE-FICHERO)
                    [until (NULL RESTO-POR-CONSIDERAR)
                       do

(* * Bucle de recorrido del arbol en profundidad)

                          (SETQ NODO-ACTUAL (CAR RESTO-POR-CONSIDERAR))
                          (if (SUBARBOL-NO-ANALIZADO? NODO-ACTUAL)
                              then [if (NOT (NODO.REFERENCIAL? NODO-ACTUAL))
                                       then (LET ((NUM (FGET.NODO NODO-ACTUAL
                                                                  (QUOTE
                                                                    NUMERO-ASOCIADOS)))
                                                  (NODO-ASOCIADO (NODO.ASOCIADO
                                                                   NODO-ACTUAL)))

(* * Salva el nodo asociado (referencial) al fichero)

                                                  (PRINT (LIST NODO-ASOCIADO
                                                               (TODO.NODO
                                                                 NODO-ASOCIADO))
                                                         NOMBRE-FICHERO)
                                                  (if (GREATERP NUM 1)
                                                      then (FPUT.NODO NODO-ACTUAL
                                                                      (QUOTE
                                                                        ANALIZADO?)
                                                                      T)
                                                           (FPUT.NODO NODO-ACTUAL
                                                                      (QUOTE
                                                                    NUMERO-NO-ANALIZADOS)
                                                                      (SUB1 NUM]

(* * Salva las especificaciones del negado)

                                                  (if (BASICO? NODO-ACTUAL)
                                                      then (LET ((NOMBRE.NEG (NOMBRE.NODO.NEGADO
                                                                               NODO-ACTUAL)))
                                                                (PRINT (LIST NOMBRE.NEG (TODO.NODO
                                                                                         NOMBRE.NEG))
                                                                       NOMBRE-FICHERO)))
                                                  [SETQ NUEVOS-A-ANADIR (APPEND (FGET.NODO
                                                                                  NODO-ACTUAL
                                                                                  (QUOTE
                                                                                    HIJOS.PUERTAS))
                                                                                (FGET.NODO
                                                                                  NODO-ACTUAL
                                                                                  (QUOTE
                                                                                    HIJOS.BASICOS]
                                       else [LET [(NUM (FGET.NODO NODO-ACTUAL (QUOTE
                                                                              NUMERO-NO-ANALIZADOS]
                                                 (if (EQ 1 NUM)
                                                     then (FREMOVE.NODO NODO-ACTUAL (QUOTE
                                                                                      ANALIZADO?))
                                                          (FREMOVE.NODO NODO-ACTUAL (QUOTE
                                                                              NUMERO-NO-ANALIZADOS))
                                                      else (FPUT.NODO NODO-ACTUAL (QUOTE
                                                                              NUMERO-NO-ANALIZADOS)
                                                                      (SUB1 NUM]
                                            (SETQ NUEVOS-A-ANADIR NIL))

(* * Salva el nodo actual)
```

```
                                (PRINT (LIST NODO-ACTUAL (TODO.NODO NODO-ACTUAL))
                                       NOMBRE-FICHERO)

        (* * Actualiza el valor de RESTO-POR-CONSIDERAR anadiendo todos los hijos del nodo considerado.)

                                (SETQ RESTO-POR-CONSIDERAR (APPEND NUEVOS-A-ANADIR (CDR
                                                                                    RESTO-POR-CONSIDERAR]
                            (CLOSEF NOMBRE-FICHERO))
          else                                                  (* Arbol de formato incorrecto)
                    (RINGBELLS)
                    (PRINTOUT T .TAB 15
                      " ARBOL DE FALLOS INEXISTENTE O TIPO INCORRECTO . REVISAR EL TIPO DEL ARBOL "
                            T])

(CARGAR.ARBOL
  [LAMBDA (NOMBRE-ARBOL TIPO AFECTA-ASOCIADOS)                  (* edited: "19-Nov-87 12:43")

        (* * Carga del fichero correspondiente la descripcion del arbol especificado)


    (if AFECTA-ASOCIADOS
        then (INICIALIZA.ASOCIADOS.REALES))
    (LET* ([EXISTE? (NLSETQ (OPENSTREAM (GENERAR.NOMBRE.FICHERO NOMBRE-ARBOL TIPO)
                                        (QUOTE INPUT]
           (NOMBRE-FICHERO (CAR EXISTE?))
           (EXPRESION)
           (NODO))
          (if EXISTE?
              then (until (EOFP NOMBRE-FICHERO)
                          do (SETQ EXPRESION (READ NOMBRE-FICHERO))
                             (SETQ NODO (CAR EXPRESION))
                             (SETPROPLIST NODO (CADR EXPRESION))
                             (if (AND AFECTA-ASOCIADOS (BASICO? NODO))
                                 then (PON.ASOCIADO.REAL (NODO.REAL NODO)
                                                         NODO))
                             (SKIPSEPRS NOMBRE-FICHERO))
                   (CLOSEF NOMBRE-FICHERO)
                   T
            else (RINGBELLS)
                 (PROMPTPRINT "NO EXISTE INFORMACION PARA EL ARBOL DE FALLOS ESPECIFICADO ")
                 NIL])

(GENERAR.NOMBRE.FICHERO
  [LAMBDA (NOMBRE-ARBOL TIPO)                                   (* edited: "29-Oct-87 03:45")

        (* * Genera el nombre del fichero correspondiente al arbol de fallos que ha sido editado. El tipo especifica si el
        nombre que desea generarse corresponde a un arbol normal o a uno depurado)


    (PACKFILENAME (QUOTE NAME)
                  (CONCAT (MKSTRING NOMBRE-ARBOL)
                          (SELECTQ TIPO
                                   (NORMAL "-ARBOL")
                                   (DEPURADO "-DEPURADO")
                                   (PROMPTPRINT
                                         " !!!!CUIDADO CON EL TIPO DEL ARBOL !!! ERROR !!!")))
                  (QUOTE EXTENSION)
                  "ARB"])
)
(DEFINEQ

(ANADIR.NODO
  [LAMBDA (ARBOL)                                               (* edited: "27-Nov-87 14:14")

        (* * Anade un nodo al arbol especificado)


    (LET ([TIPO (PREGUNTA "TIPO DEL NODO (AND/OR/BASICO) ===>  " (QUOTE PERTENENCIA)
                          (QUOTE (AND OR BASICO]
          [PADRE (PREGUNTA "PADRE DEL NODO ===> " (QUOTE FUNCION)
                           (QUOTE (PERTENECE ARBOL]
```

```
        (NUEVO-NODO)
        (NUEVO?))
       (SELECTQ (CLASIFICA.TIPO.P/B TIPO)
                [PUERTA [SETQ NUEVO? (PREGUNTA "NUEVO NODO / NODO ASOCIADO (N/A) ==> "
                                               (QUOTE PERTENENCIA)
                                               (QUOTE (N A))
                                               (QUOTE ((N T)
                                                       (A NIL]
                        (SETQ NUEVO-NODO (if NUEVO?
                                             then [ANADIR.PUERTA.NUEVA
                                                   ARBOL TIPO PADRE (PREGUNTA
                                                     "IDENTIFICACION DEL NUEVO NODO ====> "
                                                     (QUOTE FUNCION-O-NIL)
                                                     (QUOTE (STRINGP]
                                           else (ANADIR.NODO.ASOCIADO
                                                 ARBOL PADRE (PREGUNTA
                                                   " NOMBRE DEL NODO ASOCIADO ===> "
                                                   (QUOTE FUNCION)
                                                   (QUOTE (PERTENECE ARBOL)))
                                                 NIL]
                [BASICO (LET [[REAL (PREGUNTA "COMPONENTE ASOCIADO DEL SISTEMA ===> "
                                              (QUOTE FUNCION)
                                              (QUOTE (EXISTE.COMPONENTE]
                              (NEGADO (PREGUNTA "NEGADO ? (S/N) ====> " (QUOTE PERTENENCIA)
                                                (QUOTE (S N))
                                                (QUOTE ((S T)
                                                        (N NIL]
                             (SETQ NUEVO? (NOT (YA.NOMBRADO.REAL? REAL)))
                             (if NUEVO?
                                 then (SETQ NUEVO-NODO (ANADIR.BASICO.NUEVO ARBOL PADRE
                                                                            REAL NEGADO))

        (* * Lo incluimos en la lista de asociados a componentes)

                                          (PON.ASOCIADO.REAL REAL NUEVO-NODO)
                                   else (SETQ NUEVO-NODO (ANADIR.NODO.ASOCIADO ARBOL PADRE
                                                                               (
                                                                      OBTEN.ASOCIADO.REAL
                                                                        REAL)
                                                                      NEGADO]
                [PRINTOUT T " ERROR EN CLASIFICA TIPO EN ANADIR.NODO"))

        (* * Devuelve el nodo insertado en el arbol)


    NUEVO-NODO])

(CONSULTA.DATOS.NODO
  [LAMBDA (NODO)                                        (* edited: "16-Nov-87 12:14")

        (* * Muestra toda la informacion disponible del nodo especificado)


    (PRINTOUT T .TAB 20 " NODO ====> " , NODO T T T)
    (for PROPIEDAD in (TODAS.PROPIEDADES.NODO NODO)
       do (PRINTOUT T .TAB 5 PROPIEDAD . , .TAB 25 (FGET.NODO NODO PROPIEDAD)
                    T])

(CAMBIA.DATOS.NODO
  [LAMBDA (ARBOL NODO)                                  (* edited: "18-Nov-87 14:14")

        (* * Modifica alguna de los valores para las propiedades del nodo)


    (LET ((PROPS)
          (PROP-MODIFICAR)
          (NUEVO-VALOR)
          (ANTERIOR-VALOR))
         (SETQ PROPS (QUOTE (IDENTIFICACION NEGADO TIPO)))
         (PRINTOUT T .TAB 7 "NODO ===> " NODO T T)
         (PRINTOUT T .TAB 7 " PROPIEDADES MODIFICABLES ==== > ")
         (for ELEMENTO in PROPS do (PRINTOUT T .TAB 20 ELEMENTO T))
         (SETQ PROP-MODIFICAR (PREGUNTA " PROPIEDAD A MODIFICAR ===> " (QUOTE PERTENENCIA)
```

```
                                        PROPS))
        (SETQ ANTERIOR-VALOR (FGET.NODO NODO PROP-MODIFICAR))
        (PRINTOUT T T .TAB 10 "  ANTERIOR VALOR ===> " ANTERIOR-VALOR T)
        (SETQ NUEVO-VALOR (PREGUNTA " NUEVO VALOR ====> " (SELECTQ PROP-MODIFICAR
                                                                (IDENTIFICACION
                                                                  (QUOTE FUNCION))
                                                                (NEGADO (QUOTE
                                                                          PERTENENCIA))
                                                                (TIPO (QUOTE PERTENENCIA))
                                                                NIL)
                                          (SELECTQ PROP-MODIFICAR
                                                   (IDENTIFICACION (QUOTE (STRINGP)))
                                                   (NEGADO (QUOTE (T NIL)))
                                                   [TIPO (if (MEMB ANTERIOR-VALOR
                                                                   (QUOTE (AND OR)))
                                                             then (QUOTE (AND OR))
                                                           else (QUOTE (BASICO]
                                                   NIL)))
        (FPUT.NODO NODO PROP-MODIFICAR NUEVO-VALOR])
)
(DECLARE: DONTCOPY
  (FILEMAP (NIL (1234 12775 (CONSTRUIR.ARBOL 1244 . 4589) (INICIAR.ARBOL 4591 . 5412) (MODIFICA.ARBOL
5414 . 7110) (ELIMINAR.ARBOL 7112 . 7496) (SALVAR.ARBOL 7498 . 11084) (CARGAR.ARBOL 11086 . 12137) (
GENERAR.NOMBRE.FICHERO 12139 . 12773)) (12776 16794 (ANADIR.NODO 12786 . 14910) (CONSULTA.DATOS.NODO
14912 . 15330) (CAMBIA.DATOS.NODO 15332 . 16792)))))
STOP
```

```
{DSK}<LISPFILES>ARBOLES>MENUS.LSP;14     30-Nov-87 11:53:09              Page 1

(FILECREATED "30-Nov-87 11:53:10" {DSK}<LISPFILES>ARBOLES>MENUS.LSP;14 10741

      changes to:  (VARS MENUSCOMS)
                   (FNS AYUDAS AYUDAS-SUCESOS AYUDAS-ARBOLES CONFIRMAR)

      previous date: "23-Nov-87 19:32:01" {DSK}<LISPFILES>ARBOLES>MENUS.LSP;1)


(PRETTYCOMPRINT MENUSCOMS)

(RPAQQ MENUSCOMS ((* Menus de ayuda, procedimientos orientados a la introduccion de datos por parte
                     del usuario)
                  (FNS AYUDAS AYUDAS-ARBOLES AYUDAS-SUCESOS)
                  (* Comunicacion con el usuario)
                  (FNS PREGUNTA DOS.LETRAS? CONFIRMAR ESPERAR)
                  (* Funciones auxiliares)
                  (FNS DOS.LETRAS?)))


(* Menus de ayuda, procedimientos orientados a la introduccion de datos por parte del usuario)

(DEFINEQ

(AYUDAS
  [LAMBDA NIL                                          (* edited: "30-Nov-87 11:03")

        (* * Procedimiento de ayuda a la edicion. Incluye procedimientos para iniciar o modificar los valores de los
        sucesos basicos. Tambien procedimientos para la edicion de arboles de fallo, reglas, y compilacion de reglas)


    (LET ((MENU-PETICIONES (create MENU
                                   TITLE ← " TIPO DE OPERACION A REALIZAR "
                                   ITEMS ←(QUOTE (("SUCESOS BASICOS" (QUOTE SUCESOS)

              " PROCEDIMIENTOS RELACIONADOS CON LOS PARAMETROS DE LOS COMPONENTES DEL SISTEMA ")
                                                  ("ARBOLES DE FALLO" (QUOTE ARBOLES)

                                 " PROCEDIMIENTOS RELACIONADOS CON LA EDICION DE ARBOLES DE FALLOS ")
                                                  ("REGLAS DE GENERACION" (QUOTE REGLAS)

                                      "PROCEDIMIENTOS RELACIONADOS CON LA EDICION DE REGLAS ")
                                                  (" FIN DE SESION " (QUOTE FINALIZAR)

                                                    " FINALIZA LA SESION DE AYUDA Y EDICION")))
                                   CENTERFLG ← T
                                   ITEMHEIGHT ← 40
                                   ITEMWIDTH ← 300
                                   MENUBORDERSIZE ← 3))
          (FIN))
         (until FIN do (SELECTQ (MENU MENU-PETICIONES)
                                  (SUCESOS (AYUDAS-SUCESOS))
                                  (ARBOLES (AYUDAS-ARBOLES))
                                  (REGLAS (RINGBELLS)
                                          (PRINTOUT T .TAB 7 "NO IMPLEMENTADO ACTUALMENTE " T))
                                  (FINALIZAR (SETQ FIN T))
                                  (RINGBELLS])

(AYUDAS-ARBOLES
  [LAMBDA NIL                                          (* edited: "27-Nov-87 12:21")

        (* * Menus y operaciones relacionadas con la edicion y modificacion de los arboles de fallo)


        (* * ARBOL-FALLOS es la variable que indica el arbol de fallos activo actualmente)


    (LET ((MENU-PETICIONES (create MENU
                                   TITLE ← " TIPO DE OPERACION A REALIZAR "
                                   ITEMS ←(QUOTE ((" CONSTRUIR NUEVO " (QUOTE NUEVO)

                                                  "CONSTRUCCION DE UN NUEVO ARBOL DE FALLOS")
                                                  (" MODIFICACIONES  Y   CONSULTAS "
```

```
                                                    (QUOTE MODIFICA)

                "OPERACIONES COMUNES CON EL ARBOL DE FALLOS EXISTENTE: ANADIR NODOS, BORRAR.... ")
                                                    (" DEPURACION DEL ARBOL " (QUOTE DEPURAR)

                                                  " DEPURA EL ARBOL DE FALLOS ACTIVO ACTUALMENTE ")
                                                    (" SALVAR " (QUOTE SALVAR)

                    " GUARDA EN UN FICHERO LA INFORMACION DEL ARBOL DE FALLOS ACTIVO ACTUALMENTE ")
                                                    (" CARGAR " (QUOTE CARGAR)

" OBTIENE LA INFORMACION DEL ARBOL DE FALLOS ESPECIFICADO, QUE SE CONVIERTE EN EL NUEVO ARBOL DE FALLOS
O. LA INFORMACION DEL ANTERIOR ARBOL ACTIVO SE PIERDE "
                                                                    )
                                                    ("BORRAR " (QUOTE BORRAR)

                                                     " BORRA EL ARBOL DE FALLOS ESPECIFICADO ")
                                                    (" FIN DE SESION " (QUOTE FINALIZAR)

                                                              " FINALIZA LA SESION DE AYUDA ")))
                                    CENTERFLG ← T
                                    ITEMHEIGHT ← 40
                                    ITEMWIDTH ← 300
                                    MENUBORDERSIZE ← 3))
        (ARBOL-FALLOS)
        (COMPONENTES)
        (FIN))
       (CARGAR.COMPONENTES)
       (until FIN do (SELECTQ (MENU MENU-PETICIONES)
                              [NUEVO (if (CONFIRMAR
                  " PERDERAS LA INFORMACION REFERENTE AL ARBOL DE FALLOS ACTIVO ACTUALMENTE ")
                                         then (ELIMINAR.ARBOL ARBOL-FALLOS)
                                              (SETQ ARBOL-FALLOS (CONSTRUIR.ARBOL]
                              (MODIFICA (MODIFICA.ARBOL ARBOL-FALLOS))
                              (DEPURAR (DEPURAR ARBOL-FALLOS))
                              (SALVAR (SALVAR.ARBOL ARBOL-FALLOS))
                              [CARGAR (LET ((CONTINUAR T))
                                           (if (AND ARBOL-FALLOS (NOT (CONFIRMAR

                  " PERDERAS LA INFORMACION REFERENTE AL ARBOL DE FALLOS ACTIVO ACTUALMENTE ")))
                                                    then (SETQ CONTINUAR NIL))
                                                         (* Existe un arbol de fallos activo actualmente)
                                               (if CONTINUAR
                                                   then (ELIMINAR.ARBOL ARBOL-FALLOS)
                                                        [SETQ ARBOL-FALLOS
                                                          (PREGUNTA "NOMBRE DEL ARBOL ===> "
                                                                    (QUOTE FUNCION)
                                                                    (QUOTE (TRUE]
                                                        (CARGAR.ARBOL
                                                          ARBOL-FALLOS
                                                          (PREGUNTA
                                                            " NORMAL O DEPURADO  N/D ===> "
                                                                    (QUOTE PERTENENCIA)
                                                                    (QUOTE (N D))
                                                                    (QUOTE ((N NORMAL)
                                                                            (D DEPURADO]
                              (BORRAR (if (CONFIRMAR NIL)
                                          then (ELIMINAR.ARBOL ARBOL-FALLOS)))
                              (FINALIZAR (ELIMINAR.ARBOL ARBOL-FALLOS)
                                         (BORRAR.COMPONENTES)
                                         (SETQ FIN T))
                              (RINGBELLS])

(AYUDAS-SUCESOS
  [LAMBDA NIL                                                    (* edited: "30-Nov-87 11:04")

    (* * Funcion de ayuda al manejo de la estructura utilizada para almacenar la informacion correspondiente a los
    diferentes componentes del sistema (fiabilidad, tiempo de funcionamiento))


    (* * La variables global "COMPONENTES" contiene la estructura de componentes del sistema completo)
```

```
    (LET ((MENU-PETICIONES (create MENU
                                  TITLE ← " TIPO DE OPERACION A REALIZAR "
                                  ITEMS ←(QUOTE ((" INICIALIZAR COMPONENTES " (QUOTE NUEVO)

                        " ESPECIFICACION DE LOS DATOS PARA LOS COMPONENTES DE UN NUEVO SISTEMA ")
                                                (" MODIFICACIONES "
                                                  (QUOTE MODIFICA)

                "OPERACIONES COMUNES CON EL ARBOL DE FALLOS EXISTENTE: ANADIR NODOS, BORRAR.... "
                                                  (SUBITEMS (" INICIALIZA TODOS LOS TIEMPOS "
                                                              (QUOTE TIEMPO2)

                    "INICIALIZACION DE LOS DATOS DE TIEMPO DE FUNCIONAMIENTO DE TODO EL SISTEMA ")))
                                                (" FIN DE SESION " (QUOTE FINALIZAR)

                                                                  " FINALIZA LA SESION DE AYUDA ")))
                                  CENTERFLG ← T
                                  ITEMHEIGHT ← 40
                                  ITEMWIDTH ← 300
                                  MENUBORDERSIZE ← 3))
          (FIN)
          (COMPONENTES))

          (* * Carga la informacion disponible actualmente sobre los componentes del sistema)


          (CARGAR.COMPONENTES)
          (until FIN do (SELECTQ (MENU MENU-PETICIONES)
                                  (NUEVO (if (CONFIRMAR
                          " PERDERAS LA INFORMACION SOBRE LOS ACTUALES COMPONENTES DEL SISTEMA ")
                                             then (BORRAR.COMPONENTES)
                                                  (CONSTRUIR.COMPONENTES)))
                                  (MODIFICA (MODIFICA.COMPONENTES))
                                  (TIEMPO2 (if (CORROBORA)
                                               then

          (* * Inicializacion del sistema : Coloca todos los tiempos a 0 horas)


                                                    (INICIALIZA.TIEMPOS.SISTEMA)))
                                  (FINALIZAR (ACTUALIZA.COMPONENTES)
                                             (BORRAR.COMPONENTES)
                                             (SETQ FIN T))
                                  (RINGBELLS])
)


(* Comunicacion con el usuario)

(DEFINEQ

(PREGUNTA
  [LAMBDA (STRING TIPO-RESTRICCION ARGUMENTOS-RESTRICCION LISTA-FORMATO)
                                                        (* edited: "23-Nov-87 18:29")

          (* * Plantea una pregunta al usuario, y espera a leer la respuesta. Realiza la validacion de la respuesta en base
          al tipo de restriccion. El tercer argumento son los parametros necesarios dependiendo del tipo de restriccion para
          realizar esta validacion)


    (LET ((VALOR-LEIDO)
          (OTRO-FORMATO)
          (CORRECTO))
         (until CORRECTO
            do (CLEARBUF T T)
               (PRINTOUT T T .TAB 10 STRING ,,)
               (SETQ VALOR-LEIDO (READ))
               (SELECTQ TIPO-RESTRICCION
                        ((QUOTE PERTENENCIA)
                          (SETQ CORRECTO (MEMB VALOR-LEIDO ARGUMENTOS-RESTRICCION)))
                        [(QUOTE FUNCION)
                          (SETQ CORRECTO (APPLY (CAR ARGUMENTOS-RESTRICCION)
                                                (CONS VALOR-LEIDO (for ELEMENTO
```

```
                                                                    in (CDR
                                                                ARGUMENTOS-RESTRICCION)
                                                                    collect
                                                                     (EVAL ELEMENTO]
                              [(QUOTE FUNCION-O-NIL)
                                (SETQ CORRECTO (OR (NULL VALOR-LEIDO)
                                                   (APPLY (CAR ARGUMENTOS-RESTRICCION)
                                                          (CONS VALOR-LEIDO
                                                                (for ELEMENTO
                                                                   in (CDR
                                                                     ARGUMENTOS-RESTRICCION)
                                                                   collect (EVAL ELEMENTO]
                              (PROMPTPRINT " !!! ERROR EN EL TIPO DE RESTRICCION !!! "))
                       (if (NOT CORRECTO)
                           then (RINGBELLS)))
                (SETQ OTRO-FORMATO (ASSOC VALOR-LEIDO LISTA-FORMATO))
                (if OTRO-FORMATO
                    then (CADR OTRO-FORMATO)
                  else VALOR-LEIDO])

(DOS.LETRAS?
  [LAMBDA (CARACTERISTICA)                                  (* edited: " 2-Nov-87 01:49")

        (* * Comprueba si la caracteristica propuesta para el arbol tiene 2 letras. No se impone otra limitacion)


    (EQP 2 (LENGTH (UNPACK CARACTERISTICA])

(CONFIRMAR
  [LAMBDA (STRING)                                          (* edited: "25-Nov-87 09:59")

        (* * Pide conformidad al usuario sobre el aspecto detallado por el argumento)


    (LET ((VALOR)
          (CORRECTO))
         (if STRING
             then (PRINTOUT T .TAB 5 STRING T T T))
         (RINGBELLS)
         (until CORRECTO
            do (PRINTOUT T T .TAB 10 "ESTAS SEGURO DE DESEAR CONTINUAR ?  (S/N) ")
               (SETQ VALOR (READ))
               [SETQ CORRECTO (MEMB VALOR (QUOTE (S N]
               (if (NOT CORRECTO)
                   then (RINGBELLS)))
         (SELECTQ VALOR
                  (S T)
                  (N NIL)
                  (PRINTOUT T "REVISA CONFIRMAR . VALOR ADMITIDO IMPOSIBLE"])

(ESPERAR
  [LAMBDA NIL                                               (* edited: "25-Nov-87 10:02")
    (CLEARBUF T T)
    (CONTROL T)
    (READC)
    (CONTROL NIL])
)


(* Funciones auxiliares)

(DEFINEQ

(DOS.LETRAS?
  [LAMBDA (CARACTERISTICA)                                  (* edited: " 2-Nov-87 01:49")

        (* * Comprueba si la caracteristica propuesta para el arbol tiene 2 letras. No se impone otra limitacion)


    (EQP 2 (LENGTH (UNPACK CARACTERISTICA])
)
(DECLARE: DONTCOPY
  (FILEMAP (NIL (715 7436 (AYUDAS 725 . 2166) (AYUDAS-ARBOLES 2168 . 5363) (AYUDAS-SUCESOS 5365 . 7434))
```

```
 (7477 10383 (PREGUNTA 7487 . 9182) (DOS.LETRAS? 9184 . 9474) (CONFIRMAR 9476 . 10190) (ESPERAR 10192
 . 10381)) (10417 10719 (DOS.LETRAS? 10427 . 10717)))))
STOP
```

```
(FILECREATED "30-Nov-87 11:52:51" {DSK}<LISPFILES>ARBOLES>DEPURACION.LSP;13 28936

      changes to:  (FNS MARCA.ANALIZADO DEPURAR COMPROBACION-INICIAL EXPANDIR DEPURAR-PODAR
                        PODAR.NODO PODA.AND.OR PODA.AND.AND PODA.OR.OR PODA.OR.AND
                        PROPAGA.SUPRESION.OR PROPAGA.SUPRESION.AND IGUALAR.TIPOS.PUERTAS
                        TRATAR.SUBARBOLES COMPROBAR.LISTA MIRA.LISTA)
                   (VARS DEPURACIONCOMS)

      previous date: "23-Nov-87 19:39:36" {DSK}<LISPFILES>ARBOLES>DEPURACION.LSP;1)


(PRETTYCOMPRINT DEPURACIONCOMS)

(RPAQQ DEPURACIONCOMS ((* Procedimientos relacionados con la depuracion y podas de arboles de fallo)
                       (FNS DEPURAR POSIBLE.DEPURAR COMPROBACION-INICIAL EXPANDIR DEPURAR-PODAR)
                       (* Procedimientos relacionados con las podas realizadas sobre arboles de
                          fallo)
                       (FNS PODAR.NODO PODA.AND.OR PODA.AND.AND PODA.OR.OR PODA.OR.AND
                            PROPAGA.SUPRESION.OR PROPAGA.SUPRESION.AND)
                       (* Procedimientos relacionados con la depuracion de arboles de fallo)
                       (FNS DEPURAR.NODO IGUALAR.TIPOS.PUERTAS REMODELAR.NODO TRATAR.SUBARBOLES
                            TRATAR.HIJOS COMPROBAR.LISTA MIRA.LISTA)
                       (* Procedimientos de ayuda)
                       (FNS MARCA.ANALIZADO ELIMINA.MARCAS)))
```

(* Procedimientos relacionados con la depuracion y podas de arboles de fallo)

```
(DEFINEQ

(DEPURAR
  [LAMBDA (ARBOL-FALLOS)                                    (* edited: "27-Nov-87 20:08")
```

(* Algoritmo de DEPURACION El algoritmo realiza tambien una poda eliminando un tipo especial de conjuntos de separacion, los conjuntos minimos imposibles, y las redundancias en los conjuntos minimos. Esta poda se realiza para aumentar la eficiencia al calcular los conjuntos minimos y realizar un diagnostico de fallo.)*

(* Comprueba primero si la estructura propuesta para el analisis se corresponde con la esperada. Tambien comprueba si el arbol ha sido depurado previamente)*

```
    (if (POSIBLE.DEPURAR ARBOL-FALLOS)
        then
```

(Se comprueba si es necesario anadir una puerta OR para comenzar a depurar. La depuracion no comenzara si se detecta alguna situacion anomala, como podria ser el que el arbol estuviera constituido por un unico suceso Basico o que el tipo de la puerta inicial no fuese reconocible)*

```
            (if (COMPROBACION-INICIAL ARBOL-FALLOS)
                then
```

(* Variables utilizadas en la depuracion de cada nodo. Inicializacion)*

(* La lista AND-OR contiene la lista de Podas. RESTO.POR.EXPLORAR indica los nodos que aun no han sido depurados)*

```
                    [LET* [(NODO.DEPURA (FGET.ARBOL ARBOL-FALLOS (QUOTE INICIAL)))
                           [LISTA.AND.OR (QUOTE ((AND NIL)
                                                 (OR NIL]
                           (NUEVOS.A.ANADIR)
                           (RESTO.POR.EXPLORAR (LIST (LIST NODO.DEPURA LISTA.AND.OR]
                          (until (NULL RESTO.POR.EXPLORAR)
                             do
```

(* Bucle mientras queden nodos por depurar)*

```
                                 (SETQ NODO.DEPURA (CAAR RESTO.POR.EXPLORAR))
```

```
                                    (SETQ NUEVOS.A.ANADIR NIL)
                                    (if (EXISTE.NODO NODO.DEPURA)
                                        then          (* No fue eliminado como consecuencia de propagacion
                                                         de redundancias)
                                              (if (SUBARBOL-NO-ANALIZADO? NODO.DEPURA)
                                                  then
                                                       (* Iniciamos el analisis del nodo)
                                                        (SETQ LISTA.AND.OR (CADAR
                                                                            RESTO.POR.EXPLORAR))
                                                        (DEPURAR-PODAR ARBOL-FALLOS
                                                                       NODO.DEPURA
                                                                       LISTA.AND.OR)
                                                        (if (EXISTE.NODO NODO.DEPURA)
                                                            then
                                                       (* No fue eliminado en la depuracion, o en la poda)
                                                                (SETQ NUEVOS.A.ANADIR
                                                                  (EXPANDIR NODO.DEPURA
                                                                            LISTA.AND.OR))
                                                                (MARCA.ANALIZADO
                                                                            NODO.DEPURA))
                                                  else
                                                       (* Revisamos las marcas del subarbol ya analizado)
                                                       (ELIMINA.MARCAS NODO.DEPURA)))

(* * Recorrido en profundidad del arbol)

                                    (SETQ RESTO.POR.EXPLORAR (APPEND NUEVOS.A.ANADIR
                                                                     (CDR
                                                                       RESTO.POR.EXPLORAR]

(* * Especifica el nuevo tipo del arbol)

                          (FPUT.ARBOL ARBOL-FALLOS (QUOTE TIPO)
                                      (QUOTE DEPURADO))

(* * Calcula los modos de fallo minimos asociados al arbol de fallos)


(* * Se ha preferido realizar el calculo de minimos de forma separada a proceso de depuracion del arbol.
Podrian haberse realizado simultaneamente al recorrer el arbol en profundidad a medida que este va siendo depurado,
pero se ha considerado mas practico realizarlos en momentos separados (Son procesos con la suficiente entidad como
para analizarlos por separado) Este calculo de minimos con el arbol ya depurado, puede permitir realizar una
estimacion del comportamiento del metodo de diagnostico propuesto para el arbol de fallos concreto, dado que es
posible calcular el numero de conjuntos de separacion que no han sido eliminados en el arbol de fallos, y que por
tanto pueden diificultar el diagnostico de fallo)

                          (MINIMOS ARBOL-FALLOS)

(* * Fin de la depuracion)

                          (RINGBELLS)
                          (PRINTOUT T .TAB 15 ARBOL-FALLOS , , (QUOTE DEPURADO)
                                    T)

(* * Opciones tras la conclusion con el arbol depurado)

                          [LET ((OPCIONES (create MENU
                                                  TITLE ← "  OPCIONES "
                                                  ITEMS ←(QUOTE ((MOSTRAR (QUOTE MOSTRAR)

                                                      "MOSTRAR GRAFICAMENTE EL ARBOL DEPURADO ")
                                                                                 (
" SALVAR - ELIMINAR - FINALIZAR " (QUOTE FIN)
                                                  "FINALIZAR LAS OPERACIONES")))
                                                  CENTERFLG ← T
                                                  ITEMHEIGHT ← 40
                                                  ITEMWIDTH ← 300
                                                  MENUBORDERSIZE ← 3))
```

```
                                (FIN))
                               (until FIN do (SELECTQ (MENU OPCIONES)
                                                      (MOSTRAR (PRINTOUT T T
                                                         " NO IMPLEMENTADO: MOSTRAR.ARBOL "
                                                                  T T)
                                                            (RINGBELLS))
                                                      (FIN (SALVAR.ARBOL ARBOL-FALLOS)
                                                           (ELIMINAR.ARBOL ARBOL-FALLOS)
                                                           (SETQ FIN T))
                                                      (RINGBELLS]
                      else (if (EQ (QUOTE DEPURADO)
                                   (FGET.ARBOL ARBOL-FALLOS (QUOTE TIPO)))
                               then (PRINTOUT T
                                              " EL ARBOL DE FALLOS ESPECIFICADO YA HA SIDO DEPURADO ")
                             else (PRINTOUT T
                            " EL ARBOL DE FALLOS ESPECIFICADO NO PRESENTA UNA ESTRUCTURA RECONOCIBLE "))
                             (RINGBELLS])
```

(POSIBLE.DEPURAR

```
  [LAMBDA (ARBOL)                                        (* edited: "10-Nov-87 09:28")
```

(* Comprueba si es posible depurar la estructura de datos propuesta para el analisis El tipo NORMAL expresa esta posibilidad)*

```
    (EQ (FGET.ARBOL ARBOL (QUOTE TIPO))
        (QUOTE NORMAL])
```

(COMPROBACION-INICIAL

```
  [LAMBDA (ARBOL-FALLOS)                                 (* edited: "27-Nov-87 20:09")
```

(* Comprueba si es necesario anadir una puerta OR al arbol para realizar la depuracion. Comprueba ciertos tipos de situaciones anomalas en el arbol a analizar)*

```
    (LET* [(PUERTA-INICIAL (FGET.ARBOL ARBOL-FALLOS (QUOTE INICIAL)))
           (TIPO-PUERTA (FGET.NODO PUERTA-INICIAL (QUOTE TIPO]
          (SELECTQ TIPO-PUERTA
                   (AND
```

(* Anade un puerta OR como puerta inicial del arbol de fallos)*

```
                        (LET ((NUEVA-PUERTA (GENERAR.NOMBRE ARBOL-FALLOS (QUOTE PUERTA)))
                              (ID-NUEVA (FGET.NODO PUERTA-INICIAL (QUOTE IDENTIFICACION)))
                              (HIJ-NUEVOS (LIST PUERTA-INICIAL)))
                             (FPUT.ARBOL ARBOL-FALLOS (QUOTE INICIAL)
                                         NUEVA-PUERTA)
```

(* Inicializa el nuevo nodo)*

```
                             (ACTIVAR NUEVA-PUERTA)
                             (FPUT.NODO NUEVA-PUERTA (QUOTE PADRE)
                                        (QUOTE INICIO))
                             (FPUT.NODO NUEVA-PUERTA (QUOTE TIPO)
                                        (QUOTE OR))
                             (FPUT.NODO NUEVA-PUERTA (QUOTE HIJOS.PUERTAS)
                                        HIJ-NUEVOS)
                             (FPUT.NODO NUEVA-PUERTA (QUOTE IDENTIFICACION)
                                        ID-NUEVA)
                             (FPUT.NODO PUERTA-INICIAL (QUOTE PADRE)
                                        NUEVA-PUERTA)
                          T))
                   (BASICO (PRINTOUT T "EL ARBOL PROPUESTO CONSTA" T " DE UN UNICO SUCESO BASICO "
                                     T)
                           NIL)
                   (OR (PRINTOUT T " NO SE REALIZA NINGUNA MODIFICACION  SOBRE EL NODO INICIAL "
                                 T)
                       T)
                   (PRINTOUT T "EL TIPO DE LA PUERTA NO ES RECONOCIBLE" T])
```

(EXPANDIR

```
  [LAMBDA (NODO LISTA-ANTERIOR)                          (* edited: "27-Nov-87 20:09")
```

```
        (* * Esta funcion genera todos los sucesores del nodo especificado con su correspondiente lista de podas)

    (LET* [(HIJOS (FGET.NODO NODO (QUOTE HIJOS.PUERTAS)))
           (NUEVOS (FGET.NODO NODO (QUOTE HIJOS.BASICOS)))
           (TIPO (FGET.NODO NODO (QUOTE TIPO)))
           (LISTA (CADR (ASSOC TIPO LISTA-ANTERIOR)))
           (NUEVA-LISTA (COPY LISTA-ANTERIOR))
           (LISTA-INICIAL (QUOTE ((AND NIL)
                                  (OR NIL]
          (PUTASSOC TIPO (LIST (APPEND (CAMBIAR.NOMBRES NUEVOS)
                                       LISTA))
                       NUEVA-LISTA)
          (for ELEMENTO in HIJOS collect (if (SUBARBOL? ELEMENTO)
                                             then

        (* * Inicia una nueva lista de podas al tratarse de un subarbol.)

                                                    (LIST ELEMENTO LISTA-INICIAL)
                                             else

        (* * Anade los nuevos elementos a la lista de podas de los hijos del nodo que ha sido analizado)

                                                    (LIST ELEMENTO NUEVA-LISTA])

(DEPURAR-PODAR
  [LAMBDA (ARBOL-FALLOS NODO LISTA-AND-OR)                    (* edited: "27-Nov-87 20:10")
    (LET [(HIJOS-POR-EXPLORAR (FGET.NODO NODO (QUOTE HIJOS.PUERTAS)))
          (TIPO (FGET.NODO NODO (QUOTE TIPO]

        (* * HIJOS-POR-EXPLORAR contiene las puertas "hijas" de la actual que aun no han sido analizadas.
        TIPO puede tomar los valores AND OR o BASICO.)



        (* * Realiza las podas: Elimina los nodos innecesarios del arbol)

       (PODAR.NODO ARBOL-FALLOS NODO TIPO LISTA-AND-OR)
       (if (EXISTE.NODO NODO)
           then

        (* * Depura el nodo: (si no ha sido previamente eliminado al realizar la poda) Elimina las puertas del mismo tipo
        como hijos de la puerta analizada actualmente)

                  (DEPURAR.NODO ARBOL-FALLOS NODO HIJOS-POR-EXPLORAR TIPO])
)


(* Procedimientos relacionados con las podas realizadas sobre arboles de fallo)

(DEFINEQ

(PODAR.NODO
  [LAMBDA (ARBOL-FALLOS NODO TIPO LISTA-AND-OR)               (* edited: "27-Nov-87 20:11")

        (* * Realiza la poda del nodo especificado en la lista de parametros en base a la informacion suministrada en la
        lista de podas y en funcion del tipo del nodo)


    (LET [(PADRE (FGET.NODO NODO (QUOTE PADRE)))
          (SUCESOS (FGET.NODO NODO (QUOTE HIJOS.BASICOS]
         (SELECTQ TIPO
                  (AND (PODA.AND.OR ARBOL-FALLOS NODO (CADR (ASSOC (QUOTE OR)
                                                                   LISTA-AND-OR))
                                    SUCESOS)
                       (if (EXISTE.NODO NODO)
                           then (PODA.AND.AND ARBOL-FALLOS NODO (CADR (ASSOC
                                                                      (QUOTE AND)
```

```
                                                                    LISTA-AND-OR))
                                                SUCESOS)))
                    (OR (PODA.OR.OR ARBOL-FALLOS NODO (CADR (ASSOC (QUOTE OR)
                                                                   LISTA-AND-OR))
                                    SUCESOS)
                        (if (EXISTE.NODO NODO)
                            then (PODA.OR.AND ARBOL-FALLOS NODO (CADR (ASSOC (QUOTE
                                                                              AND)
                                                                             LISTA-AND-OR))
                                              SUCESOS)))
                    (PRINTOUT T " ERROR EN EL TIPO DEL NODO " , , , NODO))
```

(* Analiza los efectos de la poda sobre el nodo)*

(* Comprueba si hemos eliminado todos los hijos del nodo (el nodo puerta no tiene ya sentido))*

```
(if [AND (EXISTE.NODO NODO)
         (NULL (FGET.NODO NODO (QUOTE HIJOS.PUERTAS)))
         (NULL (FGET.NODO NODO (QUOTE HIJOS.BASICOS]
    then                                        (* Nodo tipo puerta sin hijos)
         (SUPRIMIR.NODO ARBOL-FALLOS NODO))
```

(* Comprueba si el nodo ha sido eliminado (por cualquiera de las razones consideradas en la poda) y propaga las consecuencias de esta eliminacion)*

```
(if (NOT (EXISTE.NODO NODO))
    then (SELECTQ TIPO
                  (AND (PROPAGA.SUPRESION.AND ARBOL-FALLOS PADRE))
                  (OR (PROPAGA.SUPRESION.OR ARBOL-FALLOS PADRE))
                  (PRINTOUT T " ERROR EN EL TIPO DEL NODO " , , , NODO
                            " AL INTENTAR PROPAGAR LA SUPRESION "])
```

(PODA.AND.OR

```
 [LAMBDA (ARBOL-FALLOS NODO LISTA-OR BASICOS)          (* edited: "27-Nov-87 20:12")
```

(* Realiza la poda de un nodo AND (comprueba si se debe eliminar o no))*

```
  (COND
    ((NULL BASICOS)
```

(* Fin de las comprobaciones. No realiza poda)*

```
     NIL)
    ((MEMB (FORMATEA (CAR BASICOS))
           LISTA-OR)
```

(* Realiza la poda eliminando el nodo)*

```
     (ELIMINAR.NODO ARBOL-FALLOS NODO))
    (T
```

(* Comprueba los siguientes)*

```
     (PODA.AND.OR ARBOL-FALLOS NODO LISTA-OR (CDR BASICOS])
```

(PODA.AND.AND

```
 [LAMBDA (ARBOL-FALLOS NODO LISTA-AND SUCESOS)         (* edited: "27-Nov-87 20:12")
```

(* Elimina aquellos hijos del nodo AND que no influyen en el funcionamiento del sistema. Elimina el nodo si da lugar a minimos imposibles)*

```
  (LET ((HIJO (CAR SUCESOS)))
       (COND
         ((NULL SUCESOS)
```

```
        (* * Fin de la poda)


          NIL)
         ((MIEMBRO.NEGADO (FORMATEA HIJO)
                          LISTA-AND)

        (* * Elimina el nodo. Da lugar a conjuntos minimos imposibles (suponiendo los estados de los sucesos FALLA o
        NO-FALLA))


          (ELIMINAR.NODO ARBOL-FALLOS NODO))
         ((MEMB (FORMATEA HIJO)
                LISTA-AND)

        (* * Elimina el suceso basico como hijo y sigo comprobando)


          (ELIMINAR.NODO ARBOL-FALLOS HIJO)
          (PODA.AND.AND ARBOL-FALLOS NODO LISTA-AND (CDR SUCESOS)))
         (T

        (* * Conserva el suceso y sigue comprobando)


            (PODA.AND.AND ARBOL-FALLOS NODO LISTA-AND (CDR SUCESOS])

(PODA.OR.OR
  [LAMBDA (ARBOL-FALLOS NODO-ACTUAL LISTA-OR BASICOS)        (* edited: "27-Nov-87 20:13")

        (* * Elimina los hijos correspondientes (Darian lugar a conjuntos de separacion))


    (COND
      ((NULL BASICOS)

        (* * * Fin de la poda)


       NIL)
      ((MEMB (FORMATEA (CAR BASICOS))
             LISTA-OR)

        (* * Elimina el suceso basico como hijo y sigue comprobando)


       (ELIMINAR.NODO ARBOL-FALLOS (CAR BASICOS))
       (PODA.OR.OR ARBOL-FALLOS NODO-ACTUAL LISTA-OR (CDR BASICOS)))
      (T

        (* * Incluye el suceso y sigue comprobando)


       (PODA.OR.OR ARBOL-FALLOS NODO-ACTUAL LISTA-OR (CDR BASICOS])

(PODA.OR.AND
  [LAMBDA (ARBOL-FALLOS NODO LISTA-AND SUCESOS)              (* edited: "27-Nov-87 20:14")

        (* * Elimina alguno de los hijos del nodo OR si genera conjuntos de separacion inconsistentes)


        (* * Se ha suprimido la comprobacion de si alguno de los hijos aparece en la lista de podas, por no considerarlo
        excesivamente util en este caso. No puedo eliminar directamente el hijo, ya que debo combinarlo, aun cuando dara
        lugar a redundancia en uno de los minimos, ya que ese elemento aparecera dos veces. El analisis para eliminar este
        tipo de redundancias se realiza posteriormente, y el sustituir el hijo por NIL complicaria el tratamiento, y no
        aporta una mejora notable en la eficiencia del metodo de diagnostico)


    (LET ((HIJO (CAR SUCESOS)))
         (COND
           ((NULL SUCESOS)

        (* * Fin de la poda)
```

```
        NIL)
       ((MIEMBRO.NEGADO (FORMATEA HIJO)
                        LISTA-AND)
```

(* Elimina el nodo. Da lugar a conjuntos minimos imposibles (suponiendo los estados de los sucesos FALLA o NO-FALLA))*

```
        (ELIMINAR.NODO ARBOL-FALLOS HIJO))
       (T
```

(* Conserva el suceso y sigue comprobando)*

```
          (PODA.AND.AND ARBOL-FALLOS NODO LISTA-AND (CDR SUCESOS])
```

(PROPAGA.SUPRESION.OR

```
  [LAMBDA (ARBOL NODO)                                    (* edited: "27-Nov-87 20:15")
```

(* El nodo eliminado era OR el padre es de tipo AND. Dado que este hijo no genera conjuntos minimos, el padre de tipo AND tampoco tiene asociado conjuntos minimos significativos al realizar el diagnostico.*
Elimina por tanto el subarbol completo dependiente del padre (nodo pasado como argumento,) y propaga la supresion al padre)

```
    (LET [(PADRE (FGET.NODO NODO (QUOTE PADRE]
         (ELIMINAR.NODO ARBOL NODO)
         (PROPAGAR.SUPRESION.AND PADRE])
```

(PROPAGA.SUPRESION.AND

```
  [LAMBDA (ARBOL NODO)                                    (* edited: "27-Nov-87 20:15")
```

(* El nodo eliminado era AND, el padre es de tipo OR. Comprueba si hemos eliminado todos los hijos del padre y si es asi, lo elimina y sigue propagando)*

```
    (if [AND (NULL (FGET.NODO NODO (QUOTE HIJOS.BASICOS)))
             (NULL (FGET.NODO NODO (QUOTE HIJOS.PUERTAS]
        then                                              (* Nodo sin hijos)
             (LET [(PADRE (FGET.NODO NODO (QUOTE PADRE]
                  (SUPRIMIR.NODO NODO)
                  (PROPAGA.SUPRESION.OR PADRE])
)
```

(* Procedimientos relacionados con la depuracion de arboles de fallo)

```
(DEFINEQ
```

(DEPURAR.NODO

```
  [LAMBDA (ARBOL-FALLOS NODO-EXPLORAR HIJOS TIPO)         (* edited: "29-Oct-87 14:56")
```

(* Depura el nodo especificado comprobando cada uno de sus hijos de tipo puerta (AND u OR) y eliminandolo si es del mismo tipo que el)*

```
    (IGUALAR.TIPOS.PUERTAS ARBOL-FALLOS NODO-EXPLORAR HIJOS TIPO)
```

(* Elimino inconsistencias y redundancias en los sucesos basicos)*

```
    (COMPROBAR.LISTA ARBOL-FALLOS NODO-EXPLORAR])
```

(IGUALAR.TIPOS.PUERTAS

```
  [LAMBDA (ARBOL-FALLOS NODO-EXPLORAR HIJOS TIPO)         (* edited: "27-Nov-87 20:17")
```

(* Elimina todas aquellas puertas hijas de la especificada que sean del mismo tipo. Al eliminarlas, todos los hijos de las puertas eliminadas deben incluirse como hijos de la puerta del nivel superior)*

```
    (COND
```

```
      ((NULL HIJOS)

          (* * Fin depuracion de nodo)


        (PRINTOUT T .TAB 15 (QUOTE NODO)
                  , NODO-EXPLORAR , , (QUOTE DEPURADO)
                  T))
      (T [COND
           [(EQ TIPO (FGET.NODO (CAR HIJOS)
                                (QUOTE TIPO)))

          (* * Elimina la puerta del mismo tipo)


             (SETQ HIJOS (APPEND (REMODELAR.NODO ARBOL-FALLOS NODO-EXPLORAR (CAR HIJOS))
                                 (CDR HIJOS]
           (T

          (* * No se realiza modificacion)


              (SETQ HIJOS (CDR HIJOS]

          (* * Prosigue la depuracion con el siguiente hijo)


         (IGUALAR.TIPOS.PUERTAS ARBOL-FALLOS NODO-EXPLORAR HIJOS TIPO])

(REMODELAR.NODO
  [LAMBDA (ARBOL-FALLOS NODO-PADRE NODO-EXTINGUIR)          (* edited: "17-Nov-87 11:14")

          (* * Elimina el nodo especificado del arbol y reconfigura la estructura de este)


    (LET ((NUEVAS-PUERTAS (FGET.NODO NODO-EXTINGUIR (QUOTE HIJOS.PUERTAS)))
          (NUEVOS-BASICOS (FGET.NODO NODO-EXTINGUIR (QUOTE HIJOS.BASICOS)))
          (ANTIGUOS-HIJOS)
          (ES-SUBARBOL? (SUBARBOL? NODO-EXTINGUIR)))
         (if ES-SUBARBOL?
             then (SETQ ANTIGUOS-HIJOS (FGET.NODO NODO-PADRE (QUOTE HIJOS.PUERTAS)))
                  (TRATAR.SUBARBOLES NODO-PADRE NODO-EXTINGUIR NUEVAS-PUERTAS (QUOTE
                                                                              HIJOS.PUERTAS))
                  (TRATAR.SUBARBOLES NODO-PADRE NODO-EXTINGUIR NUEVOS-BASICOS (QUOTE
                                                                              HIJOS.BASICOS))
           else (TRATAR.HIJOS NODO-PADRE NUEVAS-PUERTAS NUEVOS-BASICOS))
         (SUPRIMIR.NODO ARBOL-FALLOS NODO-EXTINGUIR)

          (* * Devuelve el valor de los nuevos nodos que debe analizar ahora.Deben devolverse los nuevos nombres de los nodos
          si estos han sido modificados al realizar el tratamiento. Estas modificaciones se introducen si el nodo a eliminar
          es un subarbol. En caso contrario los nuevos nodos que deben analizarse (depurarse) coinciden con los hijos de tipo
          AND u OR del nodo remodelado)


         (if ES-SUBARBOL?
             then (for ELEMENTO in (FGET.NODO NODO-PADRE (QUOTE HIJOS.PUERTAS))
                       collect ELEMENTO unless (MEMBER ELEMENTO ANTIGUOS-HIJOS))
           else NUEVAS-PUERTAS])

(TRATAR.SUBARBOLES
  [LAMBDA (NODO-PADRE NODO-EXTINGUIR HIJOS TIPO-HIJOS)      (* edited: "27-Nov-87 20:19")

          (* * Esta funcion reconstruye el arbol de forma que al ser el nodo que debe ser eliminado un Subarbol, construye
          estructuras de subarboles para cada uno de los hijos. Cada uno de estos subarboles tendra 2 nodos asociados, en uno
          de ellos el padre sera el antiguo subarbol, y para el otro nodo el padre sera el nodo NODO.PADRE.
          Debe realizarse un tratamiento especial para aquellos hijos que sean tambien subarboles. Consideramos subarbol si
          el numero de asociados es mayor que 1 Luego en ningun caso el subarbol sera eliminado)


    (LET ((VALOR-NEGACION)
          (NODO-ASOC))
         (for ELEMENTO in HIJOS
            do (SETQ VALOR-NEGACION (FGET.NODO ELEMENTO (QUOTE NEGADO)))
               (SETQ NODO-ASOC (NODO.ASOCIADO ELEMENTO))
```

```
            (if (NODO.REFERENCIAL? ELEMENTO)
                then
```

(* Dado que no es un subarbol he de crear una estructura de Subarbol para el nuevo nodo. Tendra como padres al anterior (Subarbol) y al nuevo NODO-PADRE)*

```
                        [LET ((NODO-1 (GENERAR.NOMBRE.ASOCIADO NODO-ASOC))
                              (NODO-2 (GENERAR.NOMBRE.ASOCIADO NODO-ASOC)))
```

(* Crea los nodos asociados)*

```
                             (CREAR.NUEVO.ASOCIADO NODO-1 ELEMENTO NODO-EXTINGUIR
                                                   VALOR-NEGACION)
                             (CREAR.NUEVO.ASOCIADO NODO-2 ELEMENTO NODO-PADRE VALOR-NEGACION)
```

(* Elimina las caracteristicas correspondientes a los asociados del nodo de estructura basica)*

```
                             (MODIFICA.PROPS.ASOCIADO ELEMENTO)
```

(* Anade el nuevo nodo a la lista de hijos del nuevo padre)*

```
                             (FADD.NODO NODO-PADRE TIPO-HIJOS NODO-2)
```

(* Modifica en la lista de hijos del subarbol el nombre correspondiente a elemento, sustituyendolo por el nuevo nombre generado correspondiente al padre NODO-EXTINGUIR)*

```
                             (FPUT.NODO NODO-EXTINGUIR TIPO-HIJOS (SUBST NODO-1 ELEMENTO
                                                                         (FGET.NODO
                                                                           NODO-EXTINGUIR
                                                                             TIPO-HIJOS]
                  else                                          (* Anado un nuevo asociado)
                       (LET ((NUEVO-NODO (GENERAR.NOMBRE.ASOCIADO NODO-ASOC)))
                            (CREAR.NUEVO.ASOCIADO NUEVO-NODO NODO-ASOC NODO-PADRE VALOR-NEGACION)
```

(* Incluye el nuevo Hijo)*

```
                             (FADD.NODO NODO-PADRE TIPO-HIJOS NUEVO-NODO])
```

```
(TRATAR.HIJOS
  [LAMBDA (NODO-PADRE NUEVAS-PUERTAS NUEVOS-BASICOS)        (* edited: "10-Nov-87 09:43")
```

(* Este procedimiento efectua el tratamiento de aquellos hijos de un nodo que debe ser eliminado al efectuar la depuracion. Se trata aqui el caso de que el nodo a eliminar no sea un subarbol)*

```
    (for ELEMENTO in NUEVAS-PUERTAS
       do (FPUT.NODO ELEMENTO (QUOTE PADRE)
                     NODO-PADRE)                            (* Cambio el padre)
          (FADD.NODO NODO-PADRE (QUOTE HIJOS.PUERTAS)
                     ELEMENTO)                              (* Anado el nuevo Hijo)
          )
    (for ELEMENTO in NUEVOS-BASICOS
       do (FPUT.NODO ELEMENTO (QUOTE PADRE)
                     NODO-PADRE)                            (* Cambio el padre)
          (FADD.NODO NODO-PADRE (QUOTE HIJOS.BASICOS)
                     ELEMENTO)                              (* Anado el nuevo Hijo)
          ])

(COMPROBAR.LISTA
  [LAMBDA (ARBOL-FALLOS NODO-EXPLORAR)                      (* edited: "27-Nov-87 20:20")
```

(* Elimina inconsistencias y redundancias de la lista de sucesos basicos "hijos" del nodo que esta siendo analizado)*

```
    (LET* ((LISTA-BASICOS (FGET.NODO NODO-EXPLORAR (QUOTE HIJOS.BASICOS)))
           (VALORES-BASICOS (CAMBIAR.NOMBRES LISTA-BASICOS)))
          (MIRA.LISTA ARBOL-FALLOS NODO-EXPLORAR LISTA-BASICOS VALORES-BASICOS)
```

```
        (* * Realiza las comprobaciones sobre la lista de hijos del nodo de tipo BASICO)


        (if (EXISTE.NODO NODO-EXPLORAR)
            then

        (* * Si el nodo no ha sido eliminado tras las comprobaciones anteriores, realiza las mismas comprobaciones para los
        hijos del nodo de tipo PUERTA)


                    (LET* ((LISTA.PUERTAS (FGET.NODO NODO-EXPLORAR (QUOTE HIJOS.PUERTAS)))
                           (VALORES.PUERTAS (CAMBIAR.NOMBRES LISTA.PUERTAS)))
                          (MIRA.LISTA ARBOL-FALLOS NODO-EXPLORAR LISTA.PUERTAS VALORES.PUERTAS])

(MIRA.LISTA
  [LAMBDA (ARBOL-FALLOS NODO-EXPLORAR LISTA-NODOS LISTA-NODOS-ASOCIADOS)
                                                        (* edited: "27-Nov-87 20:21")

        (* * Comprueba si aparece repetido alguno de los nodos de LISTA-NODOS. Si en la lista aparece el mismo nodo
        afirmado y negado, eliminamos el nodo: si es una puerta de tipo AND no dara fallo nunca luego no infuye en las OR
        de niveles superiores. Si es de tipo OR dara fallo siempre, lo que puede interpretarse como fallo de diseno o error
        en el arbol, en cualquier caso no influye en las puertas AND de niveles superiores en el momento de determinar la
        causa del fallo (Partimos de que esta parte del sistema estaria fallando siempre))


    (LET ((HIJO (CAR LISTA-NODOS))
          (HIJO-ASOCIADO (CAR LISTA-NODOS-ASOCIADOS))
          (RESTO-ASOCIADOS (CDR LISTA-NODOS-ASOCIADOS)))
         (COND
           ((NULL LISTA-NODOS)

        (* * Fin comprobaciones)


            NIL)
           ((MIEMBRO.NEGADO HIJO-ASOCIADO RESTO-ASOCIADOS)

        (* * Elimina el nodo. Aparece un elemento afirmado y negado a la vez. Finalizan las comprobaciones)


            (ELIMINAR.NODO ARBOL-FALLOS NODO-EXPLORAR)
            (PRINTOUT T T .TAB 20 "COMPRUEBE EL NODO " , , NODO-EXPLORAR ,, T T .TAB 10
" POSIBLE ERROR EN CONSTRUCCION DEL ARBOL : APARECE COMO HIJO DE UN MISMO NODO OTRO NODO AFIRMADO Y NEGAD
LA VEZ "
                      T T .TAB 15 "EL NODO HA SIDO ELIMINADO PARA PROSEGUIR EL ANALISIS " T T .TAB
                      18 " INCONSISTENCIA EN LA ESTRUCTURA "))
           ((MEMB HIJO-ASOCIADO RESTO-ASOCIADOS)

        (* * Aparece un elemento repetido. Se elimina)


            (ELIMINAR.NODO ARBOL-FALLOS HIJO)
            (PRINTOUT T .TAB 10 "EL NODO" ,, HIJO-ASOCIADO ,, "APARECE REPETIDO COMO HIJO DE " ,,
                      NODO-EXPLORAR ,, T .TAB 10 "HA SIDO ELIMINADA UNA DE LAS OCURRENCIAS " T)

        (* * Prosigue la comprobacion)


            (MIRA.LISTA ARBOL-FALLOS NODO-EXPLORAR (CDR LISTA-NODOS)
                        RESTO-ASOCIADOS))
           (T

        (* * Prosigue la comprobacion. Conserva el elemento)


             (MIRA.LISTA ARBOL-FALLOS NODO-EXPLORAR (CDR LISTA-NODOS)
                         RESTO-ASOCIADOS])
)


(* Procedimientos de ayuda)
```

```
(DEFINEQ

(MARCA.ANALIZADO
  [LAMBDA (NODO)                                        (* edited: "30-Nov-87 10:04")

          (* * Marca el nodo especificado como analizado, si resulta util para el analisis del arbol)


    (if (SUBARBOL? NODO)
        then                                            (* Es necesario marcar)
              (FPUT.NODO NODO (QUOTE ANALIZADO?)
                         T)
              (FPUT.NODO NODO (QUOTE NUMERO-NO-ANALIZADOS)
                         (SUB1 (FGET.NODO NODO (QUOTE NUMERO-ASOCIADOS])

(ELIMINA.MARCAS
  [LAMBDA (NODO)                                        (* edited: "17-Nov-87 11:55")

          (* * Analiza la marca del nodo y la elimina o modifica de acuerdo con el tipo de marca)


    (LET [(NUM-NO-ANALIZADOS (FGET.NODO NODO (QUOTE NUMERO-NO-ANALIZADOS]
         (IF (EQP 1 NUM-NO-ANALIZADOS)
             THEN (FREMOVE.NODO NODO (QUOTE ANALIZADO?))
                  (FREMOVE.NODO NODO (QUOTE NUMERO-NO-ANALIZADOS))
           ELSE (FPUT.NODO NODO (QUOTE NUMERO-NO-ANALIZADOS)
                           (SUB1 NUM-NO-ANALIZADOS])
)
(DECLARE: DONTCOPY
  (FILEMAP (NIL (1272 10657 (DEPURAR 1282 . 6650) (POSIBLE.DEPURAR 6652 . 6995) (COMPROBACION-INICIAL
6997 . 8697) (EXPANDIR 8699 . 9792) (DEPURAR-PODAR 9794 . 10655)) (10746 17872 (PODAR.NODO 10756 .
12906) (PODA.AND.OR 12908 . 13511) (PODA.AND.AND 13513 . 14561) (PODA.OR.OR 14563 . 15269) (
PODA.OR.AND 15271 . 16591) (PROPAGA.SUPRESION.OR 16593 . 17221) (PROPAGA.SUPRESION.AND 17223 . 17870))
 (17951 27754 (DEPURAR.NODO 17961 . 18444) (IGUALAR.TIPOS.PUERTAS 18446 . 19479) (REMODELAR.NODO 19481
 . 21051) (TRATAR.SUBARBOLES 21053 . 23543) (TRATAR.HIJOS 23545 . 24469) (COMPROBAR.LISTA 24471 .
25528) (MIRA.LISTA 25530 . 27752)) (27791 28914 (MARCA.ANALIZADO 27801 . 28331) (ELIMINA.MARCAS 28333
 . 28912)))))
STOP
```

```
{FLOPPY}MINIMOS.LSP;1      1-Dec-87 11:11:23              Page 1

(FILECREATED " 1-Dec-87 11:11:23" {DSK}<LISPFILES>ARBOLES>MINIMOS.LSP;31 11995

      changes to:  (VARS MINIMOSCOMS)
                   (FNS MINIMOS CALCULAR.MINIMOS CALC-MIN-AND COMBINACIONES
                        ELIMINA.CONJUNTOS.SEPARACION CALC-MIN-OR ELIMINA.MINIMOS.ASOCIADOS
                        SALVAR.MINIMOS PROPAGAR.MINIMOS ANADE.VIRTUAL CONJUNTOS.SEPARACION)

      previous date: "20-Nov-87 17:10:34" {DSK}<LISPFILES>ARBOLES>MINIMOS.LSP;14)


(PRETTYCOMPRINT MINIMOSCOMS)

(RPAQQ MINIMOSCOMS ((* Procedimientos relacionados con el calculo de conjuntos minimos)
                    (FNS MINIMOS PROPAGAR.MINIMOS CALCULAR.MINIMOS CALC-MIN-AND ANADE.VIRTUAL
                         COMBINACIONES ELIMINA.CONJUNTOS.SEPARACION CALC-MIN-OR
                         MARCAR.MINIMOS.CALCULADOS ELIMINA.MINIMOS.ASOCIADOS SALVAR.MINIMOS)))


(* Procedimientos relacionados con el calculo de conjuntos minimos)

(DEFINEQ

(MINIMOS
  [LAMBDA (ARBOL)                                          (* edited: "30-Nov-87 10:21")

        (* * Recorre el arbol especificado en orden central, (profundidad) y realiza el calculo de minimos)


    (PRINTOUT T T T .TAB 20 " CALCULANDO MINIMOS" T T T)
    (if [AND ARBOL (EQ (QUOTE DEPURADO)
                       (FGET.ARBOL ARBOL (QUOTE TIPO]
        then                                               (* Iniciamos el calculo)
             (LET [(NODO.ACTUAL)
                   (SIGUIENTES)
                   (RESTAN (LIST (FGET.ARBOL ARBOL (QUOTE INICIAL]
                  [until (NULL RESTAN)
                     do (SETQ NODO.ACTUAL (CAR RESTAN))
                        (SETQ SIGUIENTES (FGET.NODO NODO.ACTUAL (QUOTE HIJOS.PUERTAS)))
                        (if (NULL SIGUIENTES)
                            then

        (* * Calcula los minimos del nodo terminal (solo hijos basicos), y propaga el calculo hacia niveles superiores)


        (* * Calcula los minimos asociados al padre, borra (es posible almacenar si se considera necesario) lo almacenado
        para los hijos y propaga hacia la raiz del arbol)


        (* * Calcula los minimos desde el ultimo nivel al primero. (Del nivel inferior hacia los niveles superiores) Este
        metodo de calculo viene expresado por la condicion : (NULL SIGUIENTES) que indica que he llegado al ultimo nivel.)


                                (PROPAGAR.MINIMOS NODO.ACTUAL))
                            (SETQ RESTAN (APPEND SIGUIENTES (CDR RESTAN]

        (* * Guarda los minimos calculados)


                        (SALVAR.MINIMOS ARBOL))
        else (PRINTOUT T .TAB 10 "ARBOL ESPECIFICADO INEXISTENTE, O NO DEPURADO PREVIAMENTE " T])

(PROPAGAR.MINIMOS
  [LAMBDA (NODO)                                           (* edited: "21-Nov-87 01:45")

        (* * Calcula los minimos asociados a una puerta en funcion de los minimos calculados para sus sucesores)


    (CALCULAR.MINIMOS NODO)

        (* * Si se considera interesante almacenar los minimos asociados a cada puerta es posible plantearlo)
```

```
    (if (ULTIMO.HIJO? NODO)
        then                                          (* Propago al padre)
             (PROPAGAR.MINIMOS (FGET.NODO NODO (QUOTE PADRE])

(CALCULAR.MINIMOS
  [LAMBDA (NODO)                                      (* edited: "30-Nov-87 10:09")

        (* * Calcula los conjuntos minimos asociados al nodo especificado. En la version actual el valor calculado no es
        almacenado. Se plantea la posibilidad de almacenarlo en un fichero indexado por el numero de orden del nodo.
        Esto permitiria recuperar esta informacion asociada a cada nodo)


    (SELECTQ (FGET.NODO NODO (QUOTE TIPO))
             (AND (CALC-MIN-AND NODO))
             (OR (CALC-MIN-OR NODO))
             (PRINTOUT T " REVISA EL TIPO DEL NODO: " NODO " ERROR AL CALCULAR LOS MINIMOS " T))

        (* * Marca el nodo para indicar que los minimos han sido calculados. Esto sera utilizado para saber en que momento
        se puede prescindir de la informacion de minimos asociados a este nodo)


    (MARCAR.MINIMOS.CALCULADOS NODO])

(CALC-MIN-AND
  [LAMBDA (NODO)                                      (* edited: "30-Nov-87 10:11")

        (* * Calcula los minimos asociados a un nodo de tipo AND a partir de los minimos de sus sucesores)


    (LET ((CANDIDATOS)
          (VALOR))
         (for ELEMENTO in (FGET.NODO NODO (QUOTE HIJOS.PUERTAS))
            do (SETQ CANDIDATOS (CONS (FGET.NODO ELEMENTO (QUOTE MINIMOS.ASOCIADOS))
                                      CANDIDATOS))
               (ELIMINA.MINIMOS.ASOCIADOS ELEMENTO))

        (* * Calcula los minimos a partir de los minimos asociados a las puertas OR descendientes del nodo AND considerado.
        Una vez calculado, anade el virtual (descendientes de tipo BASICO) a cada combinacion calculada)


        [SETQ VALOR (ANADE.VIRTUAL (COMBINACIONES CANDIDATOS)
                                   (CAMBIAR.NOMBRES (FGET.NODO NODO (QUOTE HIJOS.BASICOS]

        (* * Elimina los conjuntos de separacion)


        (SETQ VALOR (ELIMINA.CONJUNTOS.SEPARACION VALOR))

        (* * Incluye los valores en el nodo)


        (FPUT.NODO NODO (QUOTE MINIMOS.ASOCIADOS)
                   VALOR)
        (FPUT.NODO NODO (QUOTE MODOS-ASOCIADOS)
                   (LENGTH VALOR])

(ANADE.VIRTUAL
  [LAMBDA (ELEMENTOS VIRTUAL)                         (* edited: "23-Nov-87 13:20")

        (* * Anade a los conjuntos minimos calculados el virtual correspondiente)


    (if (NULL ELEMENTOS)
        then (LIST VIRTUAL)
      else (for MINIMO in ELEMENTOS collect (UNION.J VIRTUAL MINIMO])

(COMBINACIONES
  [LAMBDA (LISTA-COMBINANDOS)                         (* edited: "30-Nov-87 10:13")

        (* * Calcula las combinaciones entre los combinandos que aparecen en el argumento. Cada uno de los combinandos
        consta a su vez de varias sublistas que se corresponden con todos los minimos asociados a un nodo determinado.
        Deben calcularse todas las combinaciones entre minimos correspondientes a diferentes nodos.
        En la version que aparece a continuacion no se detectan conjuntos redundantes dentro de los de separacion, este
```

proceso de calculo de los minimos entre los de separacion se calcula en una etapa posterior)

```
(COND
  ((NULL (CDR LISTA-COMBINANDOS))                    (* No se combinan con nada)
    (CAR LISTA-COMBINANDOS))
  (T (LET ((VALOR (COMBINACIONES (CDR LISTA-COMBINANDOS)))
           (RESULTADO))
          [for COMBINANDO in (CAR LISTA-COMBINANDOS)
             do (for YA-CALCULADO in VALOR do (SETQ RESULTADO (CONS (UNION.J
                                                                      COMBINANDO
                                                                      YA-CALCULADO)
                                                                    RESULTADO]
```

(* Devuelve el resultado obtenido)*

```
          RESULTADO])
```

```
(ELIMINA.CONJUNTOS.SEPARACION
  [LAMBDA (MINIMOS)                                  (* edited: "30-Nov-87 10:14")
```

(* Calcula los conjuntos minimos, una vez calculados los conjuntos de separacion)*

```
    (LET ((RESULTADO NIL))
```

(* En RESULTADO van siendo almacenados los elementos considerados no de separacion, a medida que se analiza la lista de minimos)*

```
        (for ELEMENTO in MINIMOS do (SETQ RESULTADO (TRATAR.INCLUSIONES RESULTADO ELEMENTO)
                                                   ))
     RESULTADO])
```

```
(CALC-MIN-OR
  [LAMBDA (NODO)                                     (* edited: "30-Nov-87 10:16")
```

(* Calcula los minimos asociados a un nodo de tipo OR a partir de los minimos de sus sucesores.
Elimina los conjuntos de separacion)*

```
    (LET ([MINIMOS (for ELEMENTO in (FGET.NODO NODO (QUOTE HIJOS.BASICOS))
                      collect (CAMBIAR.NOMBRES (LIST ELEMENTO]
          (CANDIDATOS)
          (VALOR-DEVOLVER)
          (FIN))
         (for ELEMENTO in (FGET.NODO NODO (QUOTE HIJOS.PUERTAS))
            do
```

(* Une todos los minimos asociados a cada una de las puertas descendientes del nodo)*

```
               (SETQ CANDIDATOS (APPEND (FGET.NODO ELEMENTO (QUOTE MINIMOS.ASOCIADOS))
                                        CANDIDATOS))
               (ELIMINA.MINIMOS.ASOCIADOS ELEMENTO))
         (if MINIMOS
             then                                    (* Presenta algun descendiente basico)
                  (for NUEVO-CANDIDATO in CANDIDATOS
                     do (SETQ VALOR-DEVOLVER (LIST NUEVO-CANDIDATO))
                        [for ELEMENTO in MINIMOS until FIN
                           do
```

(* Comprueba primero la longitud por considerarlo mas rapido que comprobar la inclusion en los dos sentidos.
La mayor eficiencia de uno u otro metodo dependera en muchos casos del numero de conjuntos de separacion, y de la composicion y ordenacion de estos)*

```
                                (if (GREATERP (LENGTH NUEVO-CANDIDATO)
                                              (LENGTH ELEMENTO))
                                    then (if (INCLUYE NUEVO-CANDIDATO ELEMENTO)
                                             then     (* No lo incluimos en los minimos)
                                                  (SETQ VALOR-DEVOLVER MINIMOS)
                                                  (SETQ FIN T)
```

```
                                                    else (SETQ VALOR-DEVOLVER (CONS ELEMENTO
                                                                                    VALOR-DEVOLVER)))
                                                else (if (NOT (INCLUYE ELEMENTO NUEVO-CANDIDATO))
                                                         then (SETQ VALOR-DEVOLVER (CONS ELEMENTO
                                                                                         VALOR-DEVOLVER]
                                    (SETQ MINIMOS VALOR-DEVOLVER))
                 else (SETQ MINIMOS CANDIDATOS))

        (* * Incluye los valores en el nodo)


        (FPUT.NODO NODO (QUOTE MINIMOS.ASOCIADOS)
                   MINIMOS)
        (FPUT.NODO NODO (QUOTE MODOS-ASOCIADOS)
                   (LENGTH MINIMOS])

(MARCAR.MINIMOS.CALCULADOS
  [LAMBDA (NODO)                                                    (* edited: "17-Nov-87 12:03")

        (* * Marca el nodo para indicar que los minimos correspondientes han sido calculados. Est marca es necesaria para
        propagar el calculo hacia los niveles superiores)


    (FPUT.NODO NODO (QUOTE PADRES-FALTAN)
               (OR (FGET.NODO NODO (QUOTE NUMERO-ASOCIADOS))
                   1])

(ELIMINA.MINIMOS.ASOCIADOS
  [LAMBDA (NODO-ACTUAL)                                             (* edited: "30-Nov-87 10:17")

        (* * Elimina las propiedades MINIMOS-ASOCIADOS y NUMERO-NO-ANALIZADOS introducidas para realizar el calculo y
        propagacion de los minimos. Comprueba si el nodo es o no un subarbol para realizar el tratamiento adecuado)


    (LET [(NUMERO-NO-ANALIZADOS (FGET.NODO NODO-ACTUAL (QUOTE PADRES-FALTAN]
         (if (EQ NUMERO-NO-ANALIZADOS 1)
             then (FREMOVE.NODO NODO-ACTUAL (QUOTE PADRES-FALTAN))

        (* * En esta version se eliminan los minimos asociados a cada nodo. (FREMOVE.NODO NODO-ACTUAL
        (QUOTE MINIMOS.ASOCIADOS)))


          else (FPUT.NODO NODO-ACTUAL (QUOTE PADRES-FALTAN)
                                      (SUB1 NUMERO-NO-ANALIZADOS])

(SALVAR.MINIMOS
  [LAMBDA (ARBOL)                                                   (* edited: "30-Nov-87 10:21")

        (* * Salva los minimos del arbol especificado al fichero correspondiente)


    (if ARBOL
        then                                                        (* Especificacion correcta)
             (LET* [(FICHERO (OPENSTREAM (PACKFILENAME (QUOTE NAME)
                                                       (CONCAT "MINIMOS-" (MKSTRING ARBOL))
                                                       (QUOTE EXTENSION)
                                                       "ARB")
                                         (QUOTE OUTPUT)
                                         (QUOTE OLD/NEW)))
                    (INICIAL (FGET.ARBOL ARBOL (QUOTE INICIAL)))
                    (VALOR (FGET.NODO INICIAL (QUOTE MINIMOS.ASOCIADOS]
                   (PRINT VALOR FICHERO)
                   (FREMOVE.NODO INICIAL (QUOTE MINIMOS.ASOCIADOS))

        (* * Especifica como caracteristica del arbol el numero total de modos de fallo)


                   (FPUT.ARBOL ARBOL (QUOTE NUM-MODOS-TOTALES)
                               (LENGTH VALOR))
                   (CLOSEF FICHERO)
             T)
      else NIL])
)
(DECLARE: DONTCOPY
```

```
  (FILEMAP (NIL (835 11973 (MINIMOS 845 . 2524) (PROPAGAR.MINIMOS 2526 . 3101) (CALCULAR.MINIMOS 3103 .
4021) (CALC-MIN-AND 4023 . 5255) (ANADE.VIRTUAL 5257 . 5613) (COMBINACIONES 5615 . 6837) (
ELIMINA.CONJUNTOS.SEPARACION 6839 . 7396) (CALC-MIN-OR 7398 . 9674) (MARCAR.MINIMOS.CALCULADOS 9676 .
10114) (ELIMINA.MINIMOS.ASOCIADOS 10116 . 10947) (SALVAR.MINIMOS 10949 . 11971)))))
STOP
```

(FILECREATED "30-Nov-87 11:54:28" {DSK}<LISPFILES>ARBOLES>CONJUNTOS.LSP;14 6537

changes to: (VARS CONJUNTOSCOMS)
(FNS GENERAR.TODAS.COMBINACIONES INCLUYE INCLUIDO TRATAR.INCLUSIONES UNION.J)

previous date: "23-Nov-87 20:07:52" {DSK}<LISPFILES>ARBOLES>CONJUNTOS.LSP;1)

(PRETTYCOMPRINT CONJUNTOSCOMS)

(RPAQQ CONJUNTOSCOMS ((* Operaciones sobre conjuntos representados en formato de lista)
(FNS GENERAR.TODAS.COMBINACIONES INCLUYE ELIMINA.INCLUIDOS INCLUIDO
TRATAR.INCLUSIONES UNION.J)))

(* Operaciones sobre conjuntos representados en formato de lista)

(DEFINEQ

(GENERAR.TODAS.COMBINACIONES
[LAMBDA (COMBINANDOS) (* edited: "30-Nov-87 10:28")
(LET ((COMBINACIONES)
(VALOR)
(FIN)
(ANTERIOR (GETPROPLIST (QUOTE $$COMPARACIONES)))
(VIRTUALES))
(SETPROPLIST (QUOTE $$COMPARACIONES)
NIL)
(COND
((NULL (CDR COMBINANDOS))

(* * Restaura el estado)

(SETPROPLIST (QUOTE $$COMPARACIONES)
ANTERIOR)

(* * No se considera posible que aparezcan hijos repetidos. Esto debera haber sido eliminado previamente)

(for ELEMENTO in (CAR COMBINANDOS) collect (LIST ELEMENTO)))
(T (SETQ VALOR (GENERAR.TODAS.COMBINACIONES (CDR COMBINANDOS)))
(for COMBINACION in VALOR
do (SETQ FIN NIL)
(SETQ COMBINACIONES NIL)
[for ELEMENTO in (CAR COMBINANDOS) until FIN
do (COND
((MEMB ELEMENTO COMBINACION)

(* * Elimina los conjuntos de separacion. Solo se conserva la COMBINACION anterior sin anadir ningun otro elemento)

(SETQ FIN T)

(* * Elimina los conjuntos redundantes en VIRTUALES (Aquellos con los que ya se ha combinado el elemento ELEMENTO))

(SETQ VIRTUALES (ELIMINA.INCLUIDOS COMBINACION VIRTUALES))

(* * Anade la combinacion a la lista de combinaciones que incluian ese elemento (COMPARACIONES))

(ADDPROP (QUOTE $$COMPARACIONES)
ELEMENTO COMBINACION)

(* * No se combina combinacion con nada (Generaria conjuntos redundantes))

(SETQ COMBINACIONES (LIST COMBINACION)))
((INCLUIDO (GETPROP (QUOTE $$COMPARACIONES)
ELEMENTO)
(CONS ELEMENTO COMBINACION)))
((MIEMBRO.NEGADO ELEMENTO COMBINACION))

```
                              (T (SETQ COMBINACIONES (CONS (CONS ELEMENTO COMBINACION)
                                                           COMBINACIONES]
                       (SETQ VIRTUALES (NCONC COMBINACIONES VIRTUALES)))
              (SETPROPLIST (QUOTE $$COMPARACIONES)
                           ANTERIOR)
              VIRTUALES])

(INCLUYE
  [LAMBDA (ELEM1 ELEM2)                                    (* edited: "30-Nov-87 10:33")

          (* * Comprueba si ELEM1 incluye a ELEM2. Se considera tambien el caso de igualdad en el que devuelve T)


          (* * Version iterativa)


     (if (NULL ELEM2)
         then

          (* * Por conveniencia. !! Cuidado con este convenio !!!)


             NIL
       else (LET ((RESULTADO-OBTENIDO NIL))
                 (for ELEMENTO in ELEM2 until RESULTADO-OBTENIDO
                    do (if (NOT (MEMB ELEMENTO ELEM1))
                           then (SETQ RESULTADO-OBTENIDO T)))

          (* * El reultado vendra dado por REULTADO-OBTENIDO, pero su inverso, ya que si no lo incluye RESULTADO-OBTENIDO
          sera T)


                 (NOT RESULTADO-OBTENIDO])

(ELIMINA.INCLUIDOS
  [LAMBDA (COMBINACION TODAS)                              (* edited: " 6-Nov-87 08:44")

          (* * Elimina de TODAS, todos los elementos que incluyan COMBINACION)


     (for ELEMENTO in TODAS collect ELEMENTO unless (INCLUYE ELEMENTO COMBINACION])

(INCLUIDO
  [LAMBDA (ELEMENTO ELEM-COMPROBAR)                        (* edited: "30-Nov-87 10:34")

          (* * Comprueba si alguno de los elementos de ELEMENTO esta incluido en ELEM-COMPROBAR)


     (LET ((FIN))
          (for SUBCONJUNTO in ELEMENTO until FIN do (if (INCLUYE ELEM-COMPROBAR
                                                                 SUBCONJUNTO)
                                                        then (SETQ FIN T)))

          (* * Devuelve el valor de Fin que indica si incluye o no a alguno)


      FIN])

(TRATAR.INCLUSIONES
  [LAMBDA (ACTUALES NUEVO-A-CONSIDERAR)                    (* edited: "30-Nov-87 10:36")

          (* * Comprueba si el nuevo conjunto considerado debe o no ser incluido en el conjunto ACTUALES de los considerados
          actualmente. Este nuevo conjunto puede modificar la composicion de ACTUALES)


          (* * Esta funcion es analoga a la funcion utilizada al realizar el diagnsotico (CONJUNTOS.SEPARACION) Se ha
          preferido realizar una nueva funcion dado que la forma de obtener los conjuntos que debe comparar es diferente en
          ambas, al igual que el resultado esperado. pero ambas funciones son analogas en cunato a la forma de trabajar)


     (LET ((FIN)
           (VALOR-DEVOLVER (LIST NUEVO-A-CONSIDERAR)))
```

```
        [for ELEMENTO in ACTUALES until FIN
           do
```

(* Comprueba primero la longitud por considerarlo mas rapido que comprobar la inclusion en los dos sentidos. La mayor eficiencia de uno u otro metodo dependera en muchos casos del numero de conjuntos de separacion, y de la composicion y ordenacion de estos)*

```
                (if (GREATERP (LENGTH NUEVO-A-CONSIDERAR)
                              (LENGTH ELEMENTO))
                    then (if (INCLUYE NUEVO-A-CONSIDERAR ELEMENTO)
                             then (SETQ VALOR-DEVOLVER ACTUALES)
                                  (SETQ FIN T)
                           else (SETQ VALOR-DEVOLVER (CONS ELEMENTO VALOR-DEVOLVER)))
                  else (if (NOT (INCLUYE ELEMENTO NUEVO-A-CONSIDERAR))
                           then (SETQ VALOR-DEVOLVER (CONS ELEMENTO VALOR-DEVOLVER]
    VALOR-DEVOLVER])

(UNION.J
  [LAMBDA (ELEM1 ELEM2)                                    (* edited: "30-Nov-87 10:36")
```

(* Devuelve un conjunto conteniendo todos los elmentos que aparecen en alguno de los propuestos. Se diferencia de la suministrada por el sistema, en que utiliza MEMB para las comparaciones)*

```
    (LET ((RESULTADO ELEM2))
         [for ELEMENTO in ELEM1 do (if (NOT (MEMB ELEMENTO RESULTADO))
                                       then (SETQ RESULTADO (CONS ELEMENTO RESULTADO]
     RESULTADO])
)
(DECLARE: DONTCOPY
  (FILEMAP (NIL (607 6515 (GENERAR.TODAS.COMBINACIONES 617 . 2746) (INCLUYE 2748 . 3557) (
ELIMINA.INCLUIDOS 3559 . 3865) (INCLUIDO 3867 . 4363) (TRATAR.INCLUSIONES 4365 . 5978) (UNION.J 5980
 . 6513)))))
STOP
```

(FILECREATED "30-Nov-87 11:53:19" {DSK}<LISPFILES>ARBOLES>DIAGNOSTICO.LSP;17 50023

changes to: (VARS DIAGNOSTICOCOMS)
(FNS DIAGNOSTICO.FALLO GENERAR.OTRO.NODO AVANZAR GENERAR.POSIBLES AMPLIAR.ARBOL IMPRIMIR.DATOS.NUEVOS CALCULAR.NO.FIABILIDAD CALCULAR.NO.FIABILIDAD.PUERTA F.CALCULA RETROCEDER POSIBLE.FORMATO CONSIDERA.INFORMACION MODIFICAR.FIABILIDADES PROPAGAR.PADRE PROPAGAR.HIJOS PROPAGA.SIEMPRE PROPAGA.SI.TODOS FINALIZAR CALCULAR.MODO.TOTAL ANADIR.DATOS.MINIMOS CONJUNTOS.SEPARACION FINALIZAR.BUSQUEDA IMPRIMIR.DATOS.ANADIDOS IMPRIME.PUERTA IMPRIME.UNA IMPRIME.BASICO INICIAR.BUSQUEDA IMPRIME.ELEMENTO IMPRIMIR.DATOS.FINALES ANADIR ELEMENTO.AFECTADO)

previous date: "23-Nov-87 20:00:54" {DSK}<LISPFILES>ARBOLES>DIAGNOSTICO.LSP;1)

(PRETTYCOMPRINT DIAGNOSTICOCOMS)

```
(RPAQQ DIAGNOSTICOCOMS ((* Procedimientos para la realizacion de un diagnostico del fallo detectado)
                        (FNS DIAGNOSTICO.FALLO GENERAR.OTRO.NODO INICIAR.BUSQUEDA AVANZAR
                             GENERAR.POSIBLES AMPLIAR.ARBOL ANADIR IMPRIMIR.DATOS.NUEVOS
                             IMPRIMIR.DATOS.ANADIDOS IMPRIME.PUERTA IMPRIME.UNA IMPRIME.BASICO
                             IMPRIME.ELEMENTO)
                        (* Calculo de no fiabilidades)
                        (FNS CALCULAR.NO.FIABILIDAD CALCULAR.NO.FIABILIDAD.BASICO
                             CALCULAR.NO.FIABILIDAD.PUERTA TMF.INVERSO F.CALCULA)
                        (* Procedimientos de introduccion de datos sobre la causa del fallo por
                           parte del operador)
                        (FNS RETROCEDER POSIBLE.FORMATO ELEMENTO.AFECTADO CONSIDERA.INFORMACION
                             ELIMINA.CONSIDERA MODIFICAR.FIABILIDADES PROPAGAR.PADRE
                             ULTIMO.POSIBLE? PROPAGAR.HIJOS PROPAGA.SIEMPRE PROPAGA.SI.TODOS
                             NO.CUMPLEN.RESTO)
                        (* Procedimientos de finalizacion tras haber encontrado el conjunto minimo
                           menos fiable)
                        (FNS FINALIZAR CALCULAR.MODO.TOTAL ANADIR.DATOS.MINIMOS
                             CONJUNTOS.SEPARACION IMPRIMIR.DATOS.FINALES)
                        (* Funciones de eleccion y finalizacion)
                        (FNS ORDENAR MAS.PROBABLE FINALIZAR.BUSQUEDA)))
```

(* Procedimientos para la realizacion de un diagnostico del fallo detectado)

```
(DEFINEQ
```

(DIAGNOSTICO.FALLO

```
  [LAMBDA (FALLO-DETECTADO)                                    (* edited: "30-Nov-87 10:41")
```

(* Realiza el diagnostico del fallo detectado obteniendo el modo de fallo mas probable. Es posible obtener tambien los sucesivos modos de fallo ordenados en funcion de la probabilidad de fallo)*

```
    (LET* ((FIN)
           (CORRECTO NIL)
           (OPCION)
           (CONTINUAR? (create MENU
                              TITLE ← " DECISION DE FINALIZACION "
                              ITEMS ←(QUOTE ((SIGUIENTE-MODO-DE-FALLO (QUOTE SEGUIR)
                                                                      " PROSEGUIR DIAGNOSTICO DE FALLO ")
                                             (FINALIZAR-DIAGNOSIS (QUOTE PARAR)
                                                                  "FINALIZAR DIAGNOSTICO")))
                              CENTERFLG ← T
                              ITEMHEIGHT ← 40
                              ITEMWIDTH ← 300
                              MENUBORDERSIZE ← 3))
           (TIPO-CONTINUACION (create MENU
                                      TITLE ← " TIPO DE REANUDACION "
                                      ITEMS ←(QUOTE ((NUEVA-INFORMACION (QUOTE NUEVA-INFO)
                                                                        " INTRODUCIR NUEVOS DATOS ")
                                                     (OTROS-MODOS (QUOTE OTROS-MODOS)
                                                                  " PROSIGUE LA BUSQUEDA DE OTROS MODOS DE FALLO ")))
                                      CENTERFLG ← T
```

```
                                    ITEMHEIGHT ← 40
                                    ITEMWIDTH ← 300
                                    MENUBORDERSIZE ← 3))
        (NUEVA-SALIDA (CREATEW (QUOTE (0 50 605 800))
                               (FGET.NODO FALLO-DETECTADO (QUOTE IDENTIFICACION))
                               2))
        (NUEVOS-DATOS (CREATEW (QUOTE (760 53 361 711))
                               " INTRODUZCA LA NUEVA INFORMACION" 2 T))
        (ANTERIOR-SALIDA (TTYDISPLAYSTREAM NUEVA-SALIDA))
        (INT-ANTERIOR)
        (ARBOL-BUSQUEDA (QUOTE ARBOL-BUSQUEDA))
        (SITUACION-ACTUAL)
        (ABIERTA)
        (POSIBLES)
        (VALOR)
        (ULTIMO-EXPANDIDO)
        (NUEVO-CONSIDERADO))
       (CLEARW T)
       (OPENW T)
       [if (CARGAR.ARBOL FALLO-DETECTADO (QUOTE DEPURADO))
           then
```

(* Es posible realizar el diagnostico a partir de la informacion existente)*

```
                   (INICIAR.BUSQUEDA FALLO-DETECTADO ARBOL-BUSQUEDA)
                   [SETQ ABIERTA (ORDENAR (FGET.NODO (FGET.ARBOL ARBOL-BUSQUEDA
                                                                 (QUOTE INICIAL))
                                                     (QUOTE SIGUIENTES]
                   (if (NULL ABIERTA)
                       then                          (* (NULL ABIERTA) en el instante de iniciar la
                                                        busqueda)
```

(* Abierta esta vacia el iniciar la busqueda)*

```
                                (PRINTOUT T .TAB 20
" NO DISPONGO DE INFORMACION ADICIONAL SOBRE LA CAUSA DEL FALLO PARA PODER REALIZAR UN DIAGNOSTICO "
                                       T)
                                (RINGBELLS)
                          else                       (* Es posible avanzar y obtener mas informacion)
```

(* NLSETQGAG corresponde a una variable del sistema que modifica el comportamiento en el caso de producirse un interrupcion dentro de un NLSETQ)*

```
                                (SETQ INT-ANTERIOR NLSETQGAG)
                                (SETQ NLSETQGAG T)
                                (until FIN
                                   do
```

(* Bucle de busqueda del modo de fallo mas probable (de menor no fiabilidad). 1.- Selecciono el siguiente nodo 2.- Genera los nuevos sucesores 3.- Reordena la lista abierta y amplia el arbol de busqueda.*
La busqueda puede ser dirigida por el usuario a medida que este introduzca nueva informacion sobre el estado de los componentes del sistema. El sistema suministra informacion sobre las posibilidades que esta considerando , organizando la busqueda de acuerdo a la estructura propuesta de conjuntos virtuales.)

```
                                   (SETQ ULTIMO-EXPANDIDO NUEVO-CONSIDERADO)
                                   (SETQ NUEVO-CONSIDERADO (CAR ABIERTA))
                                   (CLEARBUF T T)
                                   (SETQ VALOR (NLSETQ (AVANZAR NUEVO-CONSIDERADO
                                                                ULTIMO-EXPANDIDO)))
```

(* Inhibe las interrupciones del usuario)*

```
                                   (INTERRUPTCHAR T)
                                   (CLRPROMPT)
                                   (if (NULL VALOR)
                                       then
```

(* Se ha producido interrupcion por parte del usuario. El usuario quiere introducir nueva informacion para orientar la busqueda)*

```
                                (SETQ ABIERTA (RETROCEDER ARBOL-BUSQUEDA ABIERTA
                                                          FALLO-DETECTADO))
                        else                    (* NO ES NULL VALOR)
```

(* Sigue avanzando en la busqueda de la causa del fallo)*

(* Anade al arbol de busqueda los nuevos nodos. Comprueba las redundancias, e irregularidades por inconsistencias. Devuelve los nodos que realmente deben ser considerados en la nueva lista frontera (abierta))*

```
                        (SETQ POSIBLES (AMPLIAR.ARBOL (CAR VALOR)
                                                      NUEVO-CONSIDERADO
                                                      ARBOL-BUSQUEDA
                                                      FALLO-DETECTADO))
                        (if POSIBLES
                            then        (* DEBE PROSEGUIRSE LA BUSQUEDA)
```

(* Reordena Abierta de acuerdo con el metodo de busqueda, tras anadir los nuevos nodos incorporados al arbol de busqueda como resultado de la expansion del nodo de mayor no fiabilidad)*

```
                                [SETQ ABIERTA (ORDENAR (APPEND
                                                         POSIBLES
                                                         (CDR ABIERTA]
                        else            (* POSIBLES ES NULL)
```

(* Condicion de terminacion. Se ha obtenido el modo de fallo mas probable)*

```
                        (SETQ ABIERTA (FINALIZAR (CDR ABIERTA)
                                                 NUEVO-CONSIDERADO
                                                 ARBOL-BUSQUEDA
                                                 FALLO-DETECTADO))
                        (SETQ CORRECTO NIL)
                        (until CORRECTO
                           do (SELECTQ (MENU CONTINUAR?)
                                       (PARAR (SETQ CORRECTO T)
                                              (SETQ FIN T))
                                       (SEGUIR
                                         (SELECTQ (MENU
                                                    TIPO-CONTINUACION)
                                                  (NUEVA-INFO
                                                    (SETQ CORRECTO T)
                                                    (SETQ ABIERTA
                                                      (RETROCEDER
                                                        ARBOL-BUSQUEDA
                                                        ABIERTA
                                                      FALLO-DETECTADO)))
                                                  (OTROS-MODOS
                                                    (SETQ CORRECTO T)
                                                    (CLEARW T)
                                                    (PRINTOUT T T .TAB 10
                              " INICIAMOS LA BUSQUEDA DE OTRO MODO DE FALLO  "
                                                                        T))
                                                  (RINGBELLS)))
                                       (RINGBELLS)))
                        (if (AND (NOT FIN)
                                 (NULL ABIERTA))
                            then        (* FINALIZACION)
```

(* Finalizacion. Se han analizado todos los modos de fallo posibles)*

```
                                        (RINGBELLS)
                                        (PRINTOUT T T .TAB 5
                    " ANTE LA INFORMACION INTRODUCIDA CABA PENSAR QUE NO HAY FALLO EN EL SISTEMA "
                                                T T .TAB 10
                                " Y QUE  POR TANTO EL FALLO ES DE LA SENAL DE DETECCION "
                                                T T T)
```

```
                                                    (ESPERAR)
                                                    (SETQ FIN T]
```

(* Restaura el sistema a la situacion inicial)*

```
        (CLOSEW (TTYDISPLAYSTREAM ANTERIOR-SALIDA))
        (ELIMINAR.ARBOL FALLO-DETECTADO)
        (FINALIZAR.BUSQUEDA ARBOL-BUSQUEDA)
        (SETQ NLSETQQAG INT-ANTERIOR])
```

(GENERAR.OTRO.NODO
[LAMBDA (ARBOL) *(* edited: "30-Nov-87 10:42")*

(* Genera el nombre correspondiente a otro nodo del arbol de busqueda)*

```
   (LET [(ULTIMO (ADD1 (FGET.ARBOL ARBOL (QUOTE ULTIMO]
        (FPUT.ARBOL ARBOL (QUOTE ULTIMO)
                    ULTIMO)
        (PACK (APPEND (UNPACK (QUOTE NODO))
                      (UNPACK ULTIMO])
```

(INICIAR.BUSQUEDA
[LAMBDA (FALLO-DETECTADO ARBOL-BUSQUEDA) *(* edited: "25-Nov-87 14:55")*

(* Inicializa el arbol de busqueda para iniciar la busqueda del modo de fallo mas probable causante del fallo)*

```
   (FPUT.ARBOL ARBOL-BUSQUEDA (QUOTE IDENTIFICACION)
               (FGET.ARBOL FALLO-DETECTADO (QUOTE IDENTIFICACION)))
   (FPUT.ARBOL ARBOL-BUSQUEDA (QUOTE ULTIMO)
               0)
   (LET* [(NUEVO-NODO (GENERAR.OTRO.NODO ARBOL-BUSQUEDA))
          (OTRO-NODO)
          (NODO-INICIAL (FGET.ARBOL FALLO-DETECTADO (QUOTE INICIAL)))
          (HIJOS-INICIAL (CAMBIAR.NOMBRES (APPEND (FGET.NODO NODO-INICIAL (QUOTE
                                                                  HIJOS.PUERTAS))
                                                  (FGET.NODO NODO-INICIAL (QUOTE
                                                                  HIJOS.BASICOS]
          (ACTIVAR NUEVO-NODO)
          (FPUT.ARBOL ARBOL-BUSQUEDA (QUOTE INICIAL)
                      NUEVO-NODO)
          (FPUT.NODO NUEVO-NODO (QUOTE PUERTAS-INCLUIDAS)
                     (CAMBIAR.NOMBRES (LIST NODO-INICIAL)))
          (FPUT.NODO NUEVO-NODO (QUOTE PADRE)
                     (QUOTE INICIO))
          (FPUT.NODO NODO-INICIAL (QUOTE ESTADO)
                     (QUOTE FALLO))
          (FPUT.NODO NUEVO-NODO (QUOTE NO-FIABILIDAD)
                     1)
          (for ELEMENTO in HIJOS-INICIAL
             do (SETQ OTRO-NODO (GENERAR.OTRO.NODO ARBOL-BUSQUEDA))
                (ACTIVAR OTRO-NODO)
                (FPUT.NODO OTRO-NODO (QUOTE PADRE)
                           NUEVO-NODO)
                (ANADIR (LIST ELEMENTO)
                        OTRO-NODO)
                (FPUT.NODO OTRO-NODO (QUOTE NO-FIABILIDAD)
                           (CALCULAR.NO.FIABILIDAD OTRO-NODO FALLO-DETECTADO))
                (FADD.NODO NUEVO-NODO (QUOTE SIGUIENTES)
                           OTRO-NODO])
```

(AVANZAR
[LAMBDA (SIGUIENTE ULTIMO-CONSIDERADO) *(* edited: "30-Nov-87 10:43")*

(* Genera los nodos sucesores del especificado. Se entiende por nodos sucesores los avances en el calculo del modo de fallo a traves de la estructura de conjuntos virtuales. La presentacion de estos datos al usuario varia dependiendo de si el modo de fallo considerado incluye a la ultima suposicion, o por el contrario ha considerado un nuevo modo de fallo. En principio no se utilizara una presentacion grafica que puede plantearse (se avanzaria simultaneamente en el grafico del arbol de fallos))*

```
    (* * Permite las interrupciones del usuario para introducir nueva informacion)


  (CLEARBUF T T)
  (INTERRUPTCHAR (QUOTE P)
                 (QUOTE ERROR))
  (PROMPTPRINT " PULSA -P- PARA INTRODUCIR NUEVOS DATOS ")
  (if (EQUAL ULTIMO-CONSIDERADO (FGET.NODO SIGUIENTE (QUOTE PADRE)))
      then (IMPRIMIR.DATOS.ANADIDOS SIGUIENTE)
    else (IMPRIMIR.DATOS.NUEVOS SIGUIENTE))

    (* * Devuelve el valor de los posibles caminos a considerar para proseguir la busqueda)


  (GENERAR.POSIBLES SIGUIENTE])

(GENERAR.POSIBLES
  [LAMBDA (NODO)                                    (* edited: "30-Nov-87 10:45")

    (* * Genera todos los posibles sucesores de un nodo en el arbol de busqueda. Esta generacion de sucesores implica
    un nuevo nivel de virtuales. En el calculo del modo de fallo a traves de la estructura de virtuales, avanzamos un
    nuevo paso)


    (* * Esta funcion es especialmente critica al realizar un diagnostico de fallo. Son posibles numerosas versiones e
    implementaciones, y en muchos casos su mayor o menor eficiencia dependera de la topologia particular del arbol de
    fallos. Es posible revisar y modificar esta funcion si se requiere por cuestiones de velocidad o replanteamiento en
    la forma de realizar el analisis)


  (LET* ((PUERTAS-OR)
         (NUEVO-VALOR)
         (COMBINANDOS)
         (NUM-COMBINANDOS)
         (FIN NIL))

    (* * Calcula las puertas OR asociadas al nodo del arbol de busqueda)


    (for ELEMENTO in (FGET.NODO NODO (QUOTE PUERTAS-INCLUIDAS))
       do (SETQ PUERTAS-OR (UNION.J (CAMBIAR.NOMBRES (FGET.NODO ELEMENTO
                                                                (QUOTE
                                                                   HIJOS.PUERTAS)))
                                    PUERTAS-OR)))

    (* * Calcula los elementos que debo combinar para generar los nuevos virtuales (virtuales correspondientes al nivel
    actual))


    [for ELEMENTO in PUERTAS-OR
       do [SETQ NUEVO-VALOR (CAMBIAR.NOMBRES (APPEND (FGET.NODO ELEMENTO
                                                                (QUOTE
                                                                   HIJOS.PUERTAS))
                                                     (FGET.NODO ELEMENTO
                                                                (QUOTE
                                                                   HIJOS.BASICOS]
            (if (NOT (NULL NUEVO-VALOR))
                then (SETQ COMBINANDOS (CONS NUEVO-VALOR COMBINANDOS]

    (* * Si hay combinandos genero todas las posibles combinaciones. Estas combinaciones se corresponden con los
    virtuales a este nivel)


    (if COMBINANDOS
        then                                        (* Hay algo que combinar)
             (GENERAR.TODAS.COMBINACIONES COMBINANDOS])

(AMPLIAR.ARBOL
  [LAMBDA (NUEVOS-ELEMENTOS PADRE ARBOL FALLO-DETECTADO)    (* edited: "30-Nov-87 10:45")

    (* * Amplia el arbol de busqueda incluyendo los nuevos elementos. Genera los nodos correspondientes para incluir
    cada uno de los virtuales generados)
```

```
    (LET ((NUEVO-NODO)
          (NUEVOS-DEVUELTOS))
         (for ELEMENTO in NUEVOS-ELEMENTOS
            do

         (* * Inicializacion de las caracteristicas del nodo)


                (SETQ NUEVO-NODO (GENERAR.OTRO.NODO ARBOL))
                (ACTIVAR NUEVO-NODO)
                (FPUT.NODO NUEVO-NODO (QUOTE PADRE)
                           PADRE)
                (ANADIR ELEMENTO NUEVO-NODO)
                (FPUT.NODO NUEVO-NODO (QUOTE NO-FIABILIDAD)
                           (CALCULAR.NO.FIABILIDAD NUEVO-NODO FALLO-DETECTADO))
                (FADD.NODO PADRE (QUOTE SIGUIENTES)
                           NUEVO-NODO)
                (SETQ NUEVOS-DEVUELTOS (CONS NUEVO-NODO NUEVOS-DEVUELTOS)))

         (* * Devuelve los nuevos nodos del arbol)


     NUEVOS-DEVUELTOS])

(ANADIR
  [LAMBDA (VALORES NODO)                                    (* edited: "24-Nov-87 09:09")

         (* * Anade los hijos (VALORES) correspondientes a NODO)


    (for ELEMENTO in VALORES
       do (SELECTQ (CLASIFICA.TIPO.P/B (FGET.NODO ELEMENTO (QUOTE TIPO)))
                   (PUERTA (FADD.NODO NODO (QUOTE PUERTAS-INCLUIDAS)
                                      ELEMENTO))
                   (BASICO (FADD.NODO NODO (QUOTE BASICOS-INCLUIDOS)
                                      ELEMENTO))
                   (PRINTOUT T " FALLO EN ANADIR EN EL TIPO DEL NODO " NODO T))

         (* * Especifico en el arbol de fallos del FALLO-DETECTADO el nodo del arbol de busqueda)


           (FADD.NODO ELEMENTO (QUOTE NODOS-CONSIDERADOS)
                      NODO])

(IMPRIMIR.DATOS.NUEVOS
  [LAMBDA (ELEMENTO)                                        (* edited: "30-Nov-87 10:46")

         (* * Imprime la informacion correspondiente a la nueva suposicion considerada actualmente)


    (LET [(TODOS (LIST ELEMENTO))
          (PADRE (FGET.NODO ELEMENTO (QUOTE PADRE]
         [until (EQ PADRE (QUOTE INICIO))
            do (SETQ TODOS (CONS PADRE TODOS))
               (SETQ PADRE (FGET.NODO PADRE (QUOTE PADRE]
         (ESPERAR)
         (CLEARW T)
         (PRINTOUT T .TAB 2
           " SUPOSICIONES SOBRE ELEMENTOS DEL SISTEMA QUE ESTAN FUNCIONANDO DE FORMA INCORRECTA "
                 T T T T T)

         (* * Inicializa la lista historica de SITUACION-ACTUAL)


         (INICIALIZAR.SITUACION)

         (* * Imprime la informacion correspondiente a todos los nodos del camino)


         (* * El primer nodo es el inicial, que es de tipo OR. No es necesario incluirlo, ya que el primer descendiente
         sacara la informacion correspondiente a este nodo)
```

```
      (for VALOR in (CDR TODOS) do (IMPRIMIR.DATOS.ANADIDOS VALOR])

(IMPRIMIR.DATOS.ANADIDOS
  [LAMBDA (NUEVO-ELEMENTO)                                    (* edited: "26-Nov-87 13:14")

        (* * Imprime los datos correspondientes al nuevo elemento)


    (TERPRI)
    (TERPRI)
    (for ELEMENTO in (FGET.NODO NUEVO-ELEMENTO (QUOTE PUERTAS-INCLUIDAS))
       do (IMPRIME.PUERTA ELEMENTO))
    (for ELEMENTO in (FGET.NODO NUEVO-ELEMENTO (QUOTE BASICOS-INCLUIDOS))
       do (IMPRIME.BASICO ELEMENTO])

(IMPRIME.PUERTA
  [LAMBDA (ELEMENTO)                                          (* edited: "26-Nov-87 13:27")

        (* * Imprime la informacion correpondiente al elemento ELEMENTO)


    (LET [[BASICOS (CAMBIAR.NOMBRES (FGET.NODO ELEMENTO (QUOTE HIJOS.BASICOS]
          (PUERTAS (CAMBIAR.NOMBRES (FGET.NODO ELEMENTO (QUOTE HIJOS.PUERTAS]
         (IMPRIME.UNA (FGET.NODO ELEMENTO (QUOTE PADRE))
                      (CONSTANT POSICION.OR)
                      " CAUSADO AL MENOS POR : "
                      (CONSTANT (NOT SUPOSICION-NUEVA)))
         (IMPRIME.UNA ELEMENTO (CONSTANT POSICION.AND)
                      " CAUSADO AL MENOS POR : "
                      (CONSTANT SUPOSICION-NUEVA))
         (for VALOR in BASICOS do (IMPRIME.ELEMENTO VALOR (CONSTANT POSICION.BASICO)))
         (for VALOR in PUERTAS do (IMPRIME.UNA VALOR (CONSTANT POSICION.BASICO)
                                                NIL SUPOSICION-NUEVA])

(IMPRIME.UNA
  [LAMBDA (PUERTA N1 STRING NUEVO)                            (* edited: "26-Nov-87 13:20")

        (* * Imprime los datos referentes a la puerta especificada (AND u OR), precedido de un mensaje indicado por el
        STRING)


        (* * N1 expresa la posicion donde debe imprimirse la informacion sobre el elemento.)


        (* * El argumento NUEVO nos indica si la informacion que estamos mostrando sobre una determinada suposicion, es o
        no nueva (y ha sido mostrada anteriormente por el sistema))


    (LET ((INFORMACION (FGET.NODO PUERTA (QUOTE IDENTIFICACION)))
          (ESTADO (ESTADO.REAL PUERTA)))
         (if INFORMACION
             then (PRINTOUT T T .TAB N1 INFORMACION)
                  (if STRING
                      then (PRINTOUT T STRING))
                  (if NUEVO
                      then (PRINTOUT T ,, .TAB POSICION.NUMERO (ANADIR.A.SITUACION PUERTA)))
                  (if (EQ ESTADO (QUOTE FALLO))
                      then (PRINTOUT T ,, "COMPROBADO"))
                  (TERPRI)
                  (TERPRI])

(IMPRIME.BASICO
  [LAMBDA (ELEMENTO)                                          (* edited: "26-Nov-87 13:25")

        (* * Imprime informacion sobre sobre un basico, supuesto como causa de fallo de una puerta OR de un nivel superior,
        y la informacion de esta puerta OR)


    (IMPRIME.UNA (FGET.NODO ELEMENTO (QUOTE PADRE))
                 POSICION.OR " CAUSADO POR : " (CONSTANT (NOT SUPOSICION-NUEVA)))
    (IMPRIME.ELEMENTO ELEMENTO (CONSTANT POSICION.AND])
```

```
(IMPRIME.ELEMENTO
  [LAMBDA (ELEMENTO-BASICO N)                    (* edited: "25-Nov-87 14:51")

          (* * Imprime la informacion correspondiente al suceso basico, que forma parte del virtual de este nivel)


    (LET [(INFORMACION (FGET.NODO ELEMENTO-BASICO (QUOTE IDENTIFICACION)))
          (INF-NEGAD (FGET.NODO ELEMENTO-BASICO (QUOTE ID-NEGADO]
         (if INFORMACION
             then (TERPRI)
                  (if (NEGADO.FORMATEADO? ELEMENTO-BASICO)
                      then (if (EQ INF-NEGAD (QUOTE NADA))
                               then (PRINTOUT T .TAB N "NO ")
                                    (PRINTOUT T .TAB (PLUS N 4)
                                              INFORMACION)
                           else (PRINTOUT T .TAB N INF-NEGAD))
                    else (PRINTOUT T .TAB N INFORMACION))
                  (PRINTOUT T ,, .TAB (CONSTANT POSICION.NUMERO)
                            (ANADIR.A.SITUACION ELEMENTO-BASICO))
                  (if (EQ (ESTADO.REAL ELEMENTO-BASICO)
                          (QUOTE FALLO))
                      then (PRINTOUT T ,, "COMPROBADO"))
                  (TERPRI)
                  (TERPRI])
)


(* Calculo de no fiabilidades)

(DEFINEQ

(CALCULAR.NO.FIABILIDAD
  [LAMBDA (NODO FALLO-DETECTADO)                 (* edited: "30-Nov-87 10:47")

          (* * Calcula la probabilidad de fallo asociado a un nodo en funcion de la no fiabilidad del nodo del que desciende
          en el arbol de busqueda, y de los valores de fiabilidad asociados a los elementos (puerta y basicos) incluidos en
          el nodo especificado)


    (LET ((PUERTAS-INCLUIDAS (FGET.NODO NODO (QUOTE PUERTAS-INCLUIDAS)))
          (BASICOS-INCLUIDOS (FGET.NODO NODO (QUOTE BASICOS-INCLUIDOS)))
          (NUEVA-NO-FIABILIDAD (FGET.NODO (FGET.NODO NODO (QUOTE PADRE))
                                          (QUOTE NO-FIABILIDAD)))
          (TOTAL-MODOS-ASOCIADOS 1))

          (* * Calcula el numero total de modos de fallo asociados al nodo)


         [for ELEMENTO in PUERTAS-INCLUIDAS do (SETQ TOTAL-MODOS-ASOCIADOS
                                                     (TIMES TOTAL-MODOS-ASOCIADOS
                                                            (FGET.NODO ELEMENTO (QUOTE
                                                                                  MODOS-ASOCIADOS]
         (for ELEMENTO in BASICOS-INCLUIDOS until (LEQ NUEVA-NO-FIABILIDAD 0)
            do (SETQ NUEVA-NO-FIABILIDAD (MIN (CALCULAR.NO.FIABILIDAD.BASICO ELEMENTO)
                                              NUEVA-NO-FIABILIDAD)))
         (for ELEMENTO in PUERTAS-INCLUIDAS until (LEQ NUEVA-NO-FIABILIDAD 0)
            do (SETQ NUEVA-NO-FIABILIDAD (MIN (CALCULAR.NO.FIABILIDAD.PUERTA ELEMENTO)
                                              NUEVA-NO-FIABILIDAD)))

          (* * Devuelve el valor apropiado)


         (QUOTIENT (TIMES NUEVA-NO-FIABILIDAD TOTAL-MODOS-ASOCIADOS)
                   (FGET.ARBOL FALLO-DETECTADO (QUOTE NUM-MODOS-TOTALES])

(CALCULAR.NO.FIABILIDAD.BASICO
  [LAMBDA (SUCESO)                               (* edited: "10-Nov-87 10:46")

          (* * Calcula la no fiabilidad asociada a un suceso basico)


    (SELECTQ (FGET.NODO ELEMENTO (QUOTE ESTADO))
```

```
            (FALLO 1)
            (NO-FALLO 0)
            (F.CALCULA (TMF.INVERSO ELEMENTO)
                       (FGET.NODO ELEMENTO (QUOTE TIEMPO])

(CALCULAR.NO.FIABILIDAD.PUERTA
  [LAMBDA (PUERTA)                                    (* edited: "30-Nov-87 10:48")

        (* * Calcula la no fiabilidad asociada a una puerta en funcion de los sucesos basicos hijos de esa puerta
        (y del propio estado de la puerta como es logico))


    (LET ([HIJOS (CAMBIAR.NOMBRES (FGET.NODO PUERTA (QUOTE HIJOS.BASICOS]
          (NUEVO-VALOR 1))
         [SELECTQ (FGET.NODO PUERTA (QUOTE ESTADO))
                  (FALLO (SETQ NUEVO-VALOR 1))
                  (NO-FALLO (SETQ NUEVO-VALOR 0))
                  (for ELEMENTO in HIJOS until (LEQ NUEVO-VALOR 0)
                     do (SETQ NUEVO-VALOR (MIN (CALCULAR.NO.FIABILIDAD.BASICO ELEMENTO)
                                               NUEVO-VALOR]

        (* * Devuelve el valor calculado)


     NUEVO-VALOR])

(TMF.INVERSO
  [LAMBDA (SUCESO)                                    (* edited: "23-Nov-87 13:59")

        (* * Calcula el parametro lambda asociado al suceso basico pasado como argumento (parametro lambda = inverso del
        tiempo medio de fallo (h)))


    (if (NEGADO.FORMATEADO? SUCESO)
        then (DIFFERENCE 1 (FGET.NODO SUCESO (QUOTE PARAMETRO)))
      else (FGET.NODO SUCESO (QUOTE PARAMETRO])

(F.CALCULA
  [LAMBDA (PARAMETRO-LAMBDA TIEMPO-UTILIZACION)       (* edited: "30-Nov-87 10:48")

        (* * Calcula la no fiabilidad de un elemento en funcion del inverso del tiempo medio de fallo y el tiempo de
        funcionamiento. Utiliza la Distribucion exponencial)



        (* * e = 2.718282)


    (DIFFERENCE 1 (EXPT 2.718282 (MINUS (TIMES PARAMETRO-LAMBDA TIEMPO-UTILIZACION])
)


(* Procedimientos de introduccion de datos sobre la causa del fallo por parte del operador)

(DEFINEQ

(RETROCEDER
  [LAMBDA (ARBOL-BUSQUEDA ABIERTA FALLO-DETECTADO)    (* edited: "30-Nov-87 10:49")

        (* * Lee la nueva informacion introducida por el usuario y modifica las estructuras de acuerdo con esa nueva
        informacion)


    (LET ((ANTERIOR-TTY (TTYDISPLAYSTREAM NUEVOS-DATOS))
          (FIN)
          (DATO)
          (NUEVO-ESTADO))
         (CLEARW T)
         (OPENW T)
         (PRINTOUT T T T .TAB 10 " INTRODUZCA LA NUEVA INFORMACION " T)
         [until FIN
            do [SETQ DATO (PREGUNTA "ELEMENTO AFECTADO ====> " (QUOTE FUNCION-O-NIL)
                                    (QUOTE (POSIBLE.FORMATO FALLO-DETECTADO]
```

```
            (if (NULL DATO)
                then (SETQ FIN T)
              else (SETQ DATO (ELEMENTO.AFECTADO DATO))

    (* * Muestra el estado actual para pedir confirmacion)


                    (if (NEGADO.FORMATEADO? DATO)
                        then (PRINTOUT T "NO ES CIERTO QUE "))
                    (PRINTOUT T (FGET.NODO DATO (QUOTE IDENTIFICACION))
                              T)
                    (PRINTOUT T T .TAB 10 "ESTADO ACTUAL : " ,, (ESTADO.REAL DATO)
                              T)

    (* * Pide confirmacion de la nueva informacion)


                    (if (CONFIRMAR NIL)
                        then [SETQ NUEVO-ESTADO (INVIERTE.SI.NEGADO
                                                 DATO
                                                 (PREGUNTA " ES CIERTA LA AFIRMACION ANTERIOR (S/N) ====> "
                                                           (QUOTE PERTENENCIA)
                                                           (QUOTE (S N))
                                                           (QUOTE ((S FALLO)
                                                                   (N NO-FALLO]
                                 (SETQ ABIERTA (CONSIDERA.INFORMACION DATO NUEVO-ESTADO
                                                                      ABIERTA ARBOL-BUSQUEDA
                                                                      FALLO-DETECTADO]
        (CLOSEW (TTYDISPLAYSTREAM ANTERIOR-TTY))
        (CLEARBUF T T)
    ABIERTA])

(POSIBLE.FORMATO
  [LAMBDA (DATO ARBOL-FALLOS)                          (* edited: "30-Nov-87 10:49")

    (* * Comprueba si es correcto el formato propuesto para los nuevos datos)


    (* * En la version actual se han considerado dos posibles formatos para los datos. En el primero se especifica
    directamente un nodo del arbol de fallos sobre el que queremos introducir alguna informacion.
    En el segundo se especifica una de las suposiciones del sistema que sera validada o rechazada)


   (OR (CONSIDERADO DATO)
       (PERTENECE DATO ARBOL-FALLOS])

(ELEMENTO.AFECTADO
  [LAMBDA (DATO-INTRODUCIDO)                           (* edited: "24-Nov-87 09:22")

    (* * Considera los diversos formatos admitidos para introducir nueva informacion, y calcula el elemento del arbol
    de fallos (FALLO-DETECTADO) afectado)


    (* * En la version actual del sistema se consideran dos unicos formatos)


   (if (NUMBERP DATO-INTRODUCIDO)
       then

    (* * El usuario ha introducido la informacion utilizando la lista historica de la situacion actual considerada)


           (FGET.SITUACION DATO-INTRODUCIDO)
     else

    (* * El usuario ha especificado directamente el nodo al que hace referencia del arbol de fallos)


       (FORMATEA DATO-INTRODUCIDO])

(CONSIDERA.INFORMACION
  [LAMBDA (NODO-FALLO NUEVO-ESTADO ABIERTA ARBOL FALLO-DETECTADO)
```

```
                                                    (* edited: "30-Nov-87 10:52")

        (* * Considera la nueva informacion introducida por el usuario, y modifica el arbol de busqueda
        (y como consecuencia la estructura de abierta) en base a esta informacion. Devuelve la nueva lista de nodos
        frontera (aun no analizados))


    (LET ((AFECTADOS)
          (ORDENAR?)
          (NODOS-CONSIDERAR))
         [if (NEQ NUEVO-ESTADO (ESTADO.REAL NODO-FALLO))
             then

        (* * Se ha de considerar la nueva informacion)


                 (FPUT.NODO NODO-FALLO (QUOTE ESTADO)
                            NUEVO-ESTADO)
                 (SETQ NODOS-CONSIDERAR (LIST NODO-FALLO))
                 [until (NULL NODOS-CONSIDERAR)
                    do

        (* * Bucle de consideracion de la nueva informacion)


                           (SETQ AFECTADOS (FGET.NODO (CAR NODOS-CONSIDERAR)
                                                      (QUOTE NODOS-CONSIDERADOS)))
                           (SELECTQ NUEVO-ESTADO
                                    [NO-FALLO

        (* * Suprime las ramas del arbol de busqueda que ya no conducen a un modo de fallo)


                                                (for ELEMENTO in AFECTADOS
                                                   do (SETQ ABIERTA (ELIMINA.CONSIDERA ARBOL
                                                                                       ELEMENTO
                                                                                       ABIERTA T]
                                            (FALLO

        (* * Modifica las no fiabilidades de los nodos del arbol en base a esta nueva informacion, por tanto se ha de
        reordenar abierta)


                                                  (SETQ ORDENAR? T)
                                                  (for ELEMENTO in AFECTADOS
                                                     do (MODIFICAR.FIABILIDADES ARBOL ELEMENTO
                                                                                FALLO-DETECTADO)))
                                          (PRINTOUT T "ESTADO IMPOSIBLE NODO : " NODO-FALLO
                                                    " EN CONSIDERA.INFORMACION "
                                                    T))

        (* * Inclusion de los nuevos nodos a considerar)


                           (SETQ NODOS-CONSIDERAR (APPEND (PROPAGAR.PADRE NODO-FALLO)
                                                          (PROPAGAR.HIJOS NODO-FALLO)
                                                          (CDR NODOS-CONSIDERAR]

        (* * Si las no fiabilidades de los nodos se han visto afectadas, reordena ABIERTA.)


                    (if ORDENAR?
                        then (SETQ ABIERTA (ORDENAR ABIERTA]

        (* * El valor devuelto por la funcion es la nueva lista de nodos frontera)


      ABIERTA])

(ELIMINA.CONSIDERA
  [LAMBDA (ARBOL ELEMENTO ABIERTA AFECTA-ABIERTA?)          (* edited: "23-Nov-87 13:46")

        (* * Elimina del arbol de busqueda las ramas innecesarias (y por tanto modifica tambien la lista de nodos frontera)
        Estas ramas innecesarias surgen como consecuencia de modos de fallo descartados al comprobarse el no fallo de
```

alguno de sus componentes)

(* El ultimo parametro especifica si debe verse o no afectado la lista de nodos frontera)*

```
[if (EXISTE.NODO ELEMENTO)
    then (LET ((RESTO.POR.ELIMINAR (LIST ELEMENTO))
               (NODO.ELIMINAR)
               (NUEVOS.A.ANADIR))
              (until (NULL RESTO.POR.ELIMINAR)
                 do
```

(* Bucle de eliminacion)*

```
                    (SETQ NODO.ELIMINAR (CAR RESTO.POR.ELIMINAR))
                    (SETQ NUEVOS.A.ANADIR (FGET.NODO NODO.ELIMINAR (QUOTE SIGUIENTES)))
                    (if (AND AFECTA-ABIERTA? (NULL NUEVOS.A.ANADIR))
                        then
```

(* Es un nodo que se encuentra en la frontera de expansion)*

```
                              (SETQ ABIERTA (DREMOVE NODO.ELIMINAR ABIERTA)))
                    (DESACTIVAR NODO.ELIMINAR)
                    (SETQ RESTO.POR.ELIMINAR (APPEND NUEVOS.A.ANADIR (CDR
                                                          RESTO.POR.ELIMINAR]
```

(* Devolvemos la nueva lista frontera como resultado de la funcion)*

```
ABIERTA])
```

(MODIFICAR.FIABILIDADES

```
[LAMBDA (ARBOL ELEMENTO FALLO-DETECTADO)                    (* edited: "30-Nov-87 10:52")
```

(* Propaga las nuevas fiabilidades a traves del arbol de busqueda, tras haber sido comprobada la veracidad de una de las suposiciones hechas por el sistema)*

```
(if (EXISTE.NODO ELEMENTO)
    then (LET ((RESTO.MODIFICAR (LIST ELEMENTO))
               (NODO.MODIFICAR)
               (NUEVA.NO.FIABILIDAD))
              (until (NULL RESTO.MODIFICAR)
                 do (SETQ NODO.MODIFICAR (CAR RESTO.MODIFICAR))
                    (SETQ NUEVA.NO.FIABILIDAD (CALCULAR.NO.FIABILIDAD NODO.MODIFICAR
                                                                      FALLO-DETECTADO))
                    (if (GREATERP NUEVA.NO.FIABILIDAD (FGET.NODO NODO.MODIFICAR
                                                                 (QUOTE NO-FIABILIDAD)
                                                                 ))
                        then
```

(* Bastaria comprobar si son iguales, la comprobacion de mayor se realiza por motivos de precision en el calculo. Las no fiabilides solo pueden crecer tras comprobar que uno de los elementos esta fallando, de ahi que compruebe con mayor. Si es mayor sigue propagando)*

```
                              (SETQ RESTO.MODIFICAR (APPEND (FGET.NODO NODO.MODIFICAR
                                                                       (QUOTE
                                                                        SIGUIENTES))
                                                            (CDR RESTO.MODIFICAR)))
                        else (SETQ RESTO.MODIFICAR (CDR RESTO.MODIFICAR])
```

(PROPAGAR.PADRE

```
[LAMBDA (NODO)                                              (* edited: "30-Nov-87 10:54")
```

(* Propaga hacia los antecesores la nueva informacion correspondiente al nuevo estado en que se encuentra uno de los hijos)*

```
(LET ((PADRES (GENERAR.TODOS.PADRES NODO))
```

```
        (ESTADO (ESTADO.REAL NODO))
        (TODOS))

        (* * Si el estado del nodo no es ni FALLO ni NO-FALLO, es decir es indeterminado, no es posible la propagacion
        (ultimo caso considerado del SELECTQ))


       (SELECTQ ESTADO
                [FALLO (for ELEMENTO in PADRES
                          do (SELECTQ (FGET.NODO ELEMENTO (QUOTE TIPO))
                                      [OR

        (* * Propaga el fallo a los antecesores)


                                          (if (PROPAGA.SIEMPRE ELEMENTO ESTADO)
                                              then (SETQ TODOS (CONS ELEMENTO TODOS]
                                      [AND

        (* * Si todos los sucesores del antecesor han fallado ya, propaga el fallo al antecesor)


                                           (if (PROPAGA.SI.TODOS
                                                 ELEMENTO
                                                 [REMOVE NODO
                                                         (APPEND (FGET.NODO
                                                                   ELEMENTO
                                                                   (QUOTE
                                                                    HIJOS.BASICOS))
                                                                 (FGET.NODO
                                                                   ELEMENTO
                                                                   (QUOTE
                                                                    HIJOS.PUERTAS]
                                                 ESTADO)
                                               then (SETQ TODOS (CONS ELEMENTO TODOS]
                                      (PRINTOUT T "ERROR TIPO DEL NODO " NODO
                                                " PROPAGAR.PADRE "
                                                T]
                [NO-FALLO (for ELEMENTO in PADRES
                             do (SELECTQ
                                    (FGET.NODO ELEMENTO (QUOTE TIPO))
                                    [OR

        (* * Si se ha comprobado ya que todos los sucesores del antecesor no fallan, propaga el no fallo al antecesor)


                                        (if (PROPAGA.SI.TODOS
                                              ELEMENTO
                                              [REMOVE NODO (APPEND (FGET.NODO
                                                                     ELEMENTO
                                                                     (QUOTE
                                                                       HIJOS.BASICOS))
                                                                   (FGET.NODO
                                                                     ELEMENTO
                                                                     (QUOTE
                                                                       HIJOS.PUERTAS]
                                              ESTADO)
                                            then (SETQ TODOS (CONS ELEMENTO TODOS]
                                    [AND

        (* * Propaga el no fallo a los antecesores)


                                         (if (PROPAGA.SIEMPRE ELEMENTO ESTADO)
                                             then (SETQ TODOS (CONS ELEMENTO TODOS]
                                    (PRINTOUT T "ERROR TIPO DEL NODO " NODO " PROPAGAR.PADRE " T]
                NIL)

        (* * Devuelve todos los nodos hacia los que hemos propagado el nuevo estado del nodo)


    TODOS])

(ULTIMO.POSIBLE?
```

```
[LAMBDA (ELEMENTOS ESTADO)                                   (* edited: "12-Nov-87 11:02")

        (* * Comprueba si todos los elementos salvo uno se encuentran en el estado especificado. Si es asi devuelve ese
        elemento, en caso contrario devuelve NIL)


    (LET ((VALOR NIL)
          (FIN NIL))
         [FOR POSIBLE IN ELEMENTOS UNTIL FIN DO (IF (NEQ (ESTADO.REAL POSIBLE)
                                                         ESTADO)
                                                    THEN (IF (NULL VALOR)
                                                             THEN
                                                     (* Es el primero que encuentro)
                                                                 (SETQ VALOR POSIBLE)
                                                             ELSE
                                                     (* Ya encontre mas de uno)
                                                                 (SETQ FIN T)
                                                                 (SETQ VALOR NIL]

        (* * Devolvemos Valor)


     VALOR])

(PROPAGAR.HIJOS
  [LAMBDA (NODO)                                             (* edited: "30-Nov-87 10:55")

        (* * Propaga el nuevo estado en el que se encuentra el nodo hacia sus sucesores)


    (LET ((ESTADO (ESTADO.REAL NODO))
          (TIPO-NODO (FGET.NODO NODO (QUOTE TIPO)))
          [HIJOS (APPEND (FGET.NODO NODO (QUOTE HIJOS.BASICOS))
                         (FGET.NODO NODO (QUOTE HIJOS.PUERTAS]
          (TODOS))
         [if [OR (AND (EQ ESTADO (QUOTE FALLO))
                      (EQ TIPO-NODO (QUOTE AND)))
                 (AND (EQ ESTADO (QUOTE NO-FALLO))
                      (EQ TIPO-NODO (QUOTE OR]
            then

        (* * Propaga el mismo estado a todos los hijos)


                 [for ELEMENTO in HIJOS do (if (PROPAGAR.SIEMPRE ELEMENTO ESTADO)
                                               then (SETQ TODOS (CONS ELEMENTO TODOS]
          elseif [OR (AND (EQ ESTADO (QUOTE FALLO))
                          (EQ TIPO-NODO (QUOTE OR)))
                     (AND (EQ ESTADO (QUOTE NO-FALLO))
                          (EQ TIPO-NODO (QUOTE AND]
            then

        (* * Comprueba si se da el suficiente numero de restricciones en los hijos para efectuar la propagacion)


                 (LET [(AFECTADO (ULTIMO.POSIBLE? HIJOS (INVIERTE.ESTADO ESTADO]
                      (if AFECTADO
                          then

        (* * Solo es posible propagar si todos los sucesores salvo uno se encuentran en el estado inverso al antecesor)


                                  (if (PROPAGA.SIEMPRE AFECTADO ESTADO)
                                      then (SETQ TODOS (CONS AFECTADO TODOS]

        (* * Devuelve como resultado todos los nodos hacia los que se ha propagado)


     TODOS])

(PROPAGA.SIEMPRE
  [LAMBDA (ELEMENTO-A-PROPAGAR NUEVO-ESTADO)                 (* edited: "30-Nov-87 10:56")

        (* * Propago hacia el elemento especificado en la lista de parametros, esta propagacion vendra como consecuencia de
```

```
        un nuevo estado detectado para alguno de sus antecesores o sucesores)


    (LET ((ESTADO-NODO (ESTADO.REAL ELEMENTO-A-PROPAGAR)))
         (if (NEQ NUEVO-ESTADO ESTADO-NODO)
             then (FPUT.NODO ELEMENTO-A-PROPAGAR (QUOTE ESTADO)
                             (INVIERTE.SI.NEGADO ELEMENTO-A-PROPAGAR NUEVO-ESTADO))

        (* * Se avisa al usuario para prevenir posibles errores o inconsistencias en la informacion introducida)


                  (if (MEMB ESTADO-NODO (QUOTE (FALLO NO-FALLO)))
                      then (RINGBELLS)
                           (PRINTOUT T T T "EL ESTADO DEL NODO : " ELEMENTO-A-PROPAGAR
                    "HA SIDO MODIFICADO POR CONSISTENCIA CON LA ULTIMA INFORMACION INTRODUCIDA "
                                     T " ESTADO ANTERIOR : " ESTADO-NODO T "ESTADO ACTUAL : "
                                     (ESTADO.REAL ELEMENTO-A-PROPAGAR)
                                     T " INTRODUZCA NUEVA INFORMACION SI DESEA CORREGIR " T))

        (* * Devuelve T ya que se ha efectuado la propagacion hacia el elemento propuesto)


                  T
          else

        (* * Devuelve NIL ya que no se ha efectuado la propagacion)


                NIL])

(PROPAGA.SI.TODOS
  [LAMBDA (ELEMENTO-A-PROPAGAR RESTO-CONDICIONANTE ESTADO)    (* edited: "30-Nov-87 10:57")

        (* * Propaga el ESTADO al nuevo elemento, siempre que todos los elemento de RESTO-CONDICIONANTE, se encuentren ya
        en ese estado)



        (* * La comparacion de la igualdad del estado anterior y el nuevo se hace dos veces. Unicamente se ha pretendido
        evitar el comprobar el resto de los elementos si no es estrictamente necesario)


    (if (OR (EQ ESTADO (ESTADO.REAL ELEMENTO-A-PROPAGAR))
            (NO.CUMPLEN.RESTO RESTO-CONDICIONANTE ESTADO))
        then

        (* * No se efectua la propagacion, por lo tanto devuelve NIL)


            NIL
      else

        (* * Propagacion)


           (PROPAGA.SIEMPRE ELEMENTO-A-PROPAGAR ESTADO])

(NO.CUMPLEN.RESTO
  [LAMBDA (ELEMENTOS-A-COMPROBAR ESTADO)                    (* edited: "10-Nov-87 14:30")

        (* * Comprueba si todos los elementos propuestos se encuentran en el estado especificado)


    (LET ((RESULTADO NIL))
         (for ELEMENTO in ELEMENTOS-A-COMPROBAR until RESULTADO
            do (if (NEQ ESTADO (ESTADO.REAL ELEMENTO))
                   then (SETQ RESULTADO T)))

        (* * El valor a devolver viene dado por RESULTADO)


     RESULTADO])
)
```

```
(* Procedimientos de finalizacion tras haber encontrado el conjunto minimo menos fiable)

(DEFINEQ

(FINALIZAR
  [LAMBDA (FRONTERA MODO-FALLO ARBOL-BUSQUEDA FALLO-DETECTADO)
                                                        (* edited: "30-Nov-87 10:58")

        (* * Ha sido calculado un modo de fallo completo. Este procedimiento comprueba aquellos modos de fallo con la misma
        no fiabilidad que el calculado. Detecta los conjuntos de separacion)


    (LET* ((TODOS-MODOS (LIST MODO-FALLO))
           (MAX-NO-FIABILIDAD (FGET.NODO MODO-FALLO (QUOTE NO-FIABILIDAD)))
           (SIGUIENTE (CAR FRONTERA))
           [OTRA-NO-FIABILIDAD (if (NULL FRONTERA)
                                   then

        (* * Tomamos un valor -1 ya que nunca una no fiabilidad podra ser menor que este valor. Las no fiabilidades
        oscilaran entre 0 y 1 al tratarse de probabilidades de no fallo)


                                                -1
                                   else (FGET.NODO SIGUIENTE (QUOTE NO-FIABILIDAD]
           (ULTIMO-CONSIDERADO))

        (* * Calcula el conjunto minimo o modo de fallo considerado)


        (FPUT.NODO MODO-FALLO (QUOTE MODO-FALLO)
                   (CALCULAR.MODO.TOTAL MODO-FALLO))
        [until (LESSP OTRA-NO-FIABILIDAD MAX-NO-FIABILIDAD)
           do

        (* * Se analizan analizar todos los modos de fallo con la misma no fiabilidad que el considerado)


                (CLEARBUF T T)
                (SETQ VALOR (NLSETQ (AVANZAR SIGUIENTE ULTIMO-CONSIDERADO)))

        (* * Inhibe las interrupciones del usuario)


                (INTERRUPTCHAR T)
                (CLRPROMPT)
                [if (NULL VALOR)
                    then

        (* * Interrupcion para introducir nueva informacion)


                           (SETQ FRONTERA (RETROCEDER ARBOL-BUSQUEDA FRONTERA))
                  else (SETQ POSIBLES (AMPLIAR.ARBOL (CAR VALOR)
                                                     SIGUIENTE ARBOL-BUSQUEDA FALLO-DETECTADO)
                             )
                       (if (NULL POSIBLES)
                           then                          (* Calcula el modo de fallo completo)
                                (FPUT.NODO SIGUIENTE (QUOTE MODO-FALLO)
                                           (CALCULAR.MODO.TOTAL SIGUIENTE))
                                (SETQ TODOS-MODOS (CONJUNTOS.SEPARACION TODOS-MODOS
                                                                        SIGUIENTE))
                                (SETQ FRONTERA (CDR FRONTERA))
                         else (SETQ FRONTERA (ORDENAR (APPEND POSIBLES (CDR FRONTERA]
                (SETQ ULTIMO-CONSIDERADO SIGUIENTE)
                (SETQ SIGUIENTE (CAR FRONTERA))
                (SETQ OTRA-NO-FIABILIDAD (if (NULL FRONTERA)
                                             then -1
                                           else (FGET.NODO SIGUIENTE (QUOTE NO-FIABILIDAD]
        (IMPRIMIR.DATOS.FINALES TODOS-MODOS)

        (* * Devuelve la nueva lista frontera para proseguir la busqueda si el usuario lo desea)
```

```
      FRONTERA])

(CALCULAR.MODO.TOTAL
  [LAMBDA (NODO)                                        (* edited: "30-Nov-87 10:59")

        (* * Calcula el modo de fallo (conjunto minimo) asociado a un nodo especifico del arbol de busqueda)


    (LET ((PADRE (FGET.NODO NODO (QUOTE PADRE)))
          (RESULTADO))
         [until (EQ PADRE (QUOTE INICIO))
            do

        (* * El ultimo nodo corresponde al nodo OR inicial. No se incluye)


                (SETQ RESULTADO (ANADIR.DATOS.MINIMOS NODO RESULTADO))
                (SETQ NODO PADRE)
                (SETQ PADRE (FGET.NODO NODO (QUOTE PADRE]

        (* * Devuelve el resultado calculado)


     RESULTADO])

(ANADIR.DATOS.MINIMOS
  [LAMBDA (NODO RESULTADO)                              (* edited: "30-Nov-87 11:00")

        (* * Este procedimiento se emplea en el calculo del modo de fallo total. Analiza un nodo determinado del arbol de
        busqueda y anade los sucesos basicos asociados con ese nodo al minimo total calculado)


    (LET ((VALOR-DEVOLVER RESULTADO))

        (* * Analiza los basicos asociados al nodo)


        [for ELEMENTO in (FGET.NODO NODO (QUOTE BASICOS-INCLUIDOS))
           do (if (NOT (MEMB ELEMENTO RESULTADO))
                  then (SETQ VALOR-DEVOLVER (CONS ELEMENTO VALOR-DEVOLVER]

        (* * Analiza las puertas asociadas al nodo)


        [for PUERTA in (FGET.NODO NODO (QUOTE PUERTAS-INCLUIDAS))
           do (for ELEMENTO in (CAMBIAR.NOMBRES (FGET.NODO PUERTA (QUOTE HIJOS.BASICOS)))
                 do (if (NOT (MEMB ELEMENTO VALOR-DEVOLVER))
                        then (SETQ VALOR-DEVOLVER (CONS ELEMENTO VALOR-DEVOLVER]

        (* * RESULTADO)


     VALOR-DEVOLVER])

(CONJUNTOS.SEPARACION
  [LAMBDA (ACTUALES NUEVO-MODO-CONSIDERAR)              (* edited: "30-Nov-87 11:01")

        (* * Comprueba si el nuevo modo de fallo NUEVO-MODO-CONSIDERAR incluye alguno de los ya considerados
        (que deben ser suprimidos) o es incluido por alguno de ellos (por lo que no debe incluirse) Esta funcion es tambien
        critica dado el tiempo gastado en comprobar la inclusion de dos listas. La version aqui presentada admite
        revisiones para aumentar la eficiencia)


    (LET ((FIN)
          (VALOR-BASICOS (FGET.NODO NUEVO-MODO-CONSIDERAR (QUOTE MODO-FALLO)))
          (VALOR-COMPRUEBA)
          (VALOR-DEVOLVER (LIST NUEVO-MODO-CONSIDERAR)))
         [for ELEMENTO in ACTUALES until FIN
            do (SETQ VALOR-COMPRUEBA (FGET.NODO ELEMENTO (QUOTE MODO-FALLO)))

        (* * Comprueba primero la longitud por considerarlo mas rapido que comprobar la inclusion en los dos sentidos.
        La mayor eficiencia de uno u otro metodo dependera en muchos casos del numero de conjuntos de separacion, y de la
        composicion y ordenacion de estos)
```

```
                (if (GREATERP (LENGTH VALOR-BASICOS)
                              (LENGTH VALOR-COMPRUEBA))
                    then (if (INCLUYE VALOR-BASICOS VALOR-COMPRUEBA)
                             then (SETQ VALOR-DEVOLVER ACTUALES)
                                  (SETQ FIN T)
                           else (SETQ VALOR-DEVOLVER (CONS ELEMENTO VALOR-DEVOLVER)))
                  else (if (NOT (INCLUYE VALOR-COMPRUEBA VALOR-BASICOS))
                           then (SETQ VALOR-DEVOLVER (CONS ELEMENTO VALOR-DEVOLVER]
    VALOR-DEVOLVER])

(IMPRIMIR.DATOS.FINALES
  [LAMBDA (TODOS-MODOS-FALLO)                          (* edited: "25-Nov-87 14:38")

        (* * Imprime una informacion resumen de todos los modos de fallo calculados con la misma no fiabilidad, una vez que
        han sido eliminados los conjuntos de separacion)


    (LET ((INDICE 1)
          (INFORMACION))
         (ESPERAR)
         (CLEARW T)
         (INICIALIZAR.SITUACION)
         (PRINTOUT T T .TAB 5
                   " LISTADO DE TODOS LOS MODOS DE FALLO DETECTADOS CON IDENTICA NO FIABILIDAD "
                   T)
         (for MODO in TODOS-MODOS-FALLO
            do (PRINTOUT T T .TAB 5 "MODO " INDICE T)
               (SETQ INDICE (ADD1 INDICE))
               (for ELEMENTO in (FGET.NODO MODO (QUOTE MODO-FALLO))
                  do (IMPRIME.ELEMENTO ELEMENTO 10])
)


(* Funciones de eleccion y finalizacion)

(DEFINEQ

(ORDENAR
  [LAMBDA (LISTA-NODOS-EXPANDIR)                       (* edited: "10-Nov-87 10:39")

        (* * Ordena la lista de nodos a expandir (lista ABIERTA), de forma que el siguiente nodo a expandir sea el que nos
        conduzca a un modo de fallo de mayor NO FIABILIDAD (mayor probabilidad de fallo) Al realizar esta ordenacion,
        introducimos un factor de correccion: El numero de minimos incluidos en un nodo (cada nodo representa un conjunto
        virtual para los nodos del arbol de busqueda de niveles inferiores), de esta forma daremos preferencia al realizar
        la busqueda a aquellos caminos donde aparezcan implicados un mayor numero de modos de fallo
        (conjuntos minimos))


    (SORT LISTA-NODOS-EXPANDIR (QUOTE MAS.PROBABLE])

(MAS.PROBABLE
  [LAMBDA (ELEMENTO-1 ELEMENTO-2)                      (* edited: "17-Nov-87 13:41")

        (* * Devuelve T si el primer elemento es mas fiable que el segundo. Los datos de fiabilidad son recuperables a
        partir de la estructura ARBOL-FALLOS)


    (GEQ (FGET.NODO ELEMENTO-1 (QUOTE NO-FIABILIDAD))
         (FGET.NODO ELEMENTO-2 (QUOTE NO-FIABILIDAD]

(FINALIZAR.BUSQUEDA
  [LAMBDA (ARBOL-BUSQUEDA)                             (* edited: "30-Nov-87 11:01")

        (* * Elimina toda la informacion del arbol de busqueda, liberando el espacio de memoria)


        (* * Los dos parametros NIL al final de la lista de parametros indican que no es necesario que al eliminar el arbol
        modifique la lista de nodos frontera, dado que se elimina todo el arbol)
```

```
    (ELIMINA.CONSIDERA ARBOL-BUSQUEDA (FGET.ARBOL ARBOL-BUSQUEDA (QUOTE INICIAL))
                       NIL NIL])
)
(DECLARE: DONTCOPY
  (FILEMAP (NIL (2103 21282 (DIAGNOSTICO.FALLO 2113 . 8901) (GENERAR.OTRO.NODO 8903 . 9334) (
INICIAR.BUSQUEDA 9336 . 11069) (AVANZAR 11071 . 12309) (GENERAR.POSIBLES 12311 . 14425) (AMPLIAR.ARBOL
 14427 . 15420) (ANADIR 15422 . 16153) (IMPRIMIR.DATOS.NUEVOS 16155 . 17300) (IMPRIMIR.DATOS.ANADIDOS
17302 . 17805) (IMPRIME.PUERTA 17807 . 18723) (IMPRIME.UNA 18725 . 19757) (IMPRIME.BASICO 19759 .
20255) (IMPRIME.ELEMENTO 20257 . 21280)) (21322 25100 (CALCULAR.NO.FIABILIDAD 21332 . 22975) (
CALCULAR.NO.FIABILIDAD.BASICO 22977 . 23383) (CALCULAR.NO.FIABILIDAD.PUERTA 23385 . 24206) (
TMF.INVERSO 24208 . 24663) (F.CALCULA 24665 . 25098)) (25201 41152 (RETROCEDER 25211 . 26912) (
POSIBLE.FORMATO 26914 . 27525) (ELEMENTO.AFECTADO 27527 . 28309) (CONSIDERA.INFORMACION 28311 . 30614)
 (ELIMINA.CONSIDERA 30616 . 32005) (MODIFICAR.FIABILIDADES 32007 . 33384) (PROPAGAR.PADRE 33386 .
35890) (ULTIMO.POSIBLE? 35892 . 36789) (PROPAGAR.HIJOS 36791 . 38480) (PROPAGA.SIEMPRE 38482 . 39805)
(PROPAGA.SI.TODOS 39807 . 40632) (NO.CUMPLEN.RESTO 40634 . 41150)) (41250 48189 (FINALIZAR 41260 .
43947) (CALCULAR.MODO.TOTAL 43949 . 44630) (ANADIR.DATOS.MINIMOS 44632 . 45748) (CONJUNTOS.SEPARACION
45750 . 47370) (IMPRIMIR.DATOS.FINALES 47372 . 48187)) (48239 50001 (ORDENAR 48249 . 49041) (
MAS.PROBABLE 49043 . 49452) (FINALIZAR.BUSQUEDA 49454 . 49999)))))
STOP
```

```
{DSK}<LISPFILES>ARBOLES>COMPONENTES.LSP;43     30-Nov-87 11:54:09              Page 1

(FILECREATED "30-Nov-87 11:54:10" {DSK}<LISPFILES>ARBOLES>COMPONENTES.LSP;43 21047

      changes to:  (VARS COMPONENTESCOMS)
                   (FNS EXISTE.COMPONENTE TODO.SOBRE.COMPONENTES CONSTRUIR.COMPONENTES
                        CARGAR.COMPONENTES INICIALIZA.TIEMPOS.SISTEMA ACTUALIZA.TIEMPOS.COMPONENTES
                        ELIMINAR.COMPONENTE CAMBIA.COMPONENTE CONSULTA.COMPONENTE
                        INICIAR.SIGUIENTE.COMPONENTE ANADIR.COMPONENTE FREMOVE.COMPONENTE.PROP
                        AYUDAS-SUCESOS ANADIR.A.SITUACION INICIALIZA.ASOCIADOS.REALES
                        PON.ASOCIADO.REAL OBTEN.ASOCIADO.REAL YA.NOMBRADO.REAL?)

      previous date: "20-Nov-87 19:10:34" {DSK}<LISPFILES>ARBOLES>COMPONENTES.LSP;27)


(PRETTYCOMPRINT COMPONENTESCOMS)

(RPAQQ COMPONENTESCOMS ((* Procedimientos de manejo de los componentes del sistema)
                        (* Primitivas basicas)
                        (FNS ACTIVAR.COMPONENTE FGET.COMPONENTE FPUT.COMPONENTE
                             FGET.COMPONENTE.PROP FPUT.COMPONENTE.PROP FREMOVE.COMPONENTE.PROP
                             EXISTE.COMPONENTE TODO.COMPONENTE TODO.SOBRE.COMPONENTES
                             GENERAR.SIGUIENTE.COMPONENTE ACTUALIZA.COMPONENTES
                             INICIALIZAR.COMPONENTES)
                        (* Funciones de manejo de estructura)
                        (FNS CONSTRUIR.COMPONENTES CARGAR.COMPONENTES BORRAR.COMPONENTES
                             MODIFICA.COMPONENTES INICIALIZA.TIEMPOS.SISTEMA
                             ACTUALIZA.TIEMPOS.COMPONENTES INICIAR.SIGUIENTE.COMPONENTE
                             ANADIR.COMPONENTE ELIMINAR.COMPONENTE CONSULTA.COMPONENTE
                             CAMBIA.COMPONENTE)
                        (* Funciones de manejo de una estructura ligada a los componentes : los
                           sucesos basicos asociados a cada componente en el arbol activo
                           actualmente)
                        (FNS INICIALIZA.ASOCIADOS.REALES OBTEN.ASOCIADO.REAL PON.ASOCIADO.REAL
                             QUITA.ASOCIADO.REAL YA.NOMBRADO.REAL?)))


(* Procedimientos de manejo de los componentes del sistema)


(* Primitivas basicas)

(DEFINEQ

(ACTIVAR.COMPONENTE
  [LAMBDA (COMPONENTE)                                    (* edited: "12-Nov-87 12:24")

          (* * Activa la componente especificada. El convenio elegido para activacion y desactivacion pretende seguir la
          mismas directrices del resto del sistema, y recuperar esa informacion borrada, lo que puede considerarse de
          utilidad)


    (FPUT.COMPONENTE COMPONENTE (QUOTE ACTIVO)
                     T])

(FGET.COMPONENTE
  [LAMBDA (COMPONENTE PROPIEDAD)                          (* edited: "12-Nov-87 12:17")

          (* * Obtiene la informacion del nodo especificado)


    (CADR (ASSOC PROPIEDAD (CAR (NTH COMPONENTES COMPONENTE])

(FPUT.COMPONENTE
  [LAMBDA (COMPONENTE PROPIEDAD VALOR)                    (* edited: "12-Nov-87 12:19")

          (* * Incluye informacion sobre una componente determinada. Primitiva de manejo de la estructura)


    (RPLACA [CDR (ASSOC PROPIEDAD (CAR (NTH COMPONENTES COMPONENTE]
            VALOR])

(FGET.COMPONENTE.PROP
```

```
[LAMBDA (PROPIEDAD)                                    (* edited: "12-Nov-87 11:50")

        (* * Obtiene el valor de una propiedad generica de los componentes del sistema)


        (* * COMPONENTES es la variable global utilizada para los componentes del sistema)

    (GETPROP (QUOTE COMPONENTES)
             PROPIEDAD])

(FPUT.COMPONENTE.PROP
  [LAMBDA (PROPIEDAD VALOR)                            (* edited: "12-Nov-87 12:29")

        (* * Coloca el valor de una propiedad generica de los componentes del sistema)


        (* * COMPONENTES es la variable global utilizada para los componentes del sistema)

    (PUTPROP (QUOTE COMPONENTES)
             PROPIEDAD VALOR])

(FREMOVE.COMPONENTE.PROP
  [LAMBDA (PROPIEDAD)                                  (* edited: "23-Nov-87 14:55")

        (* * Elimina una propiedad generica de los componentes del sistema)


    (REMPROP (QUOTE COMPONENTES)
             PROPIEDAD])

(EXISTE.COMPONENTE
  [LAMBDA (COMPONENTE-SISTEMA)                         (* edited: "30-Nov-87 11:06")

        (* * Se comprueba en base a la propiedAD ACTIVO. Este metodo permite conservar la informacion de los componentes,
        aun cuando hayan sido borrados, para otro tipo de consultas)


    (AND (NUMBERP COMPONENTE-SISTEMA)
         (GREATERP COMPONENTE-SISTEMA 0)
         (LEQ COMPONENTE-SISTEMA (FGET.COMPONENTE.PROP (QUOTE ULTIMO)))
         (FGET.COMPONENTE COMPONENTE-SISTEMA (QUOTE ACTIVO])

(TODO.COMPONENTE
  [LAMBDA (COMPONENTE)                                 (* edited: "12-Nov-87 12:20")

        (* * Obtiene todas las propiedades de uno de los sucesos basicos del sistema)


        (* * COMPONENTES es la variable global del sistema que almacena la informacion relativa a los componentes del
        mismo)


    (CAR (NTH COMPONENTES COMPONENTE])

(TODO.SOBRE.COMPONENTES
  [LAMBDA NIL                                          (* edited: "30-Nov-87 11:06")

        (* * Obtiene toda la informacion sobre propiedades de los componentes del sistema)


    (GETPROPLIST (QUOTE COMPONENTES])

(GENERAR.SIGUIENTE.COMPONENTE
  [LAMBDA NIL                                          (* edited: "12-Nov-87 11:53")

        (* * Genera el numero correlativo correspondiente al siguiente componente del sistema, y actualiza ese valor)


    (LET [(SIGUIENTE (ADD1 (FGET.COMPONENTE.PROP (QUOTE ULTIMO]
```

```
        (FPUT.COMPONENTE.PROP (QUOTE ULTIMO)
                              SIGUIENTE)
    SIGUIENTE])

(ACTUALIZA.COMPONENTES
  [LAMBDA NIL                                          (* edited: "17-Nov-87 19:41")

        (* * Realiza la actualizacion de los datos sobre los componentes del sistema)


    (LET [(ULTIMO (OR (FGET.COMPONENTE.PROP (QUOTE ULTIMO))
                      0))
          (FICHERO (OPENSTREAM (PACKFILENAME (QUOTE NAME)
                                             "COMPONENTES-SISTEMA"
                                             (QUOTE EXTENSION)
                                             "ARB")
                               (QUOTE OUTPUT)
                               (QUOTE OLD/NEW]
          (FREMOVE.COMPONENTE.PROP (QUOTE VALUE))
          (PRINT (TODO.SOBRE.COMPONENTES (QUOTE COMPONENTES))
                 FICHERO)
          (PRINT COMPONENTES FICHERO)
          (CLOSEF FICHERO])

(INICIALIZAR.COMPONENTES
  [LAMBDA NIL                                          (* edited: "13-Nov-87 14:42")

        (* * Inicializa las componentes del sistema. La variable global COMPONENTES contiene esta informacion)


    (SETQ COMPONENTES NIL)
    (SETPROPLIST (QUOTE COMPONENTES)
                 NIL)
    (FPUT.COMPONENTE.PROP (QUOTE ULTIMO)
                          0])
)


(* Funciones de manejo de estructura)

(DEFINEQ

(CONSTRUIR.COMPONENTES
  [LAMBDA NIL                                          (* edited: "30-Nov-87 11:07")

        (* * Construye la estructura de componentes del sistema)


    (LET* ((SALIDA (CREATEW (QUOTE (371 303 593 466))
                            " ESPECIFICACION DE LOS SUCESOS BASICOS DEL SISTEMA " 4 T))
           (ANTERIOR (TTYDISPLAYSTREAM SALIDA))
           (FIN))
          (OPENW T)
          (CLEARW T)

          (* * Inicializacion de los parametros correspondientes a las componentes del sistema)


          (INICIALIZAR.COMPONENTES)
          (until FIN
             do

          (* * Bucle de adicion de componentes)


                 (CLEARW T)
                 (ANADIR.COMPONENTE)
                 (PRINTOUT T T T .TAB 10 "PULSA INTRO PARA CONTINUAR. F PARA FINALIZAR ")
                 (CLEARBUF T T)
                 (CONTROL T)
                 (SETQ FIN (EQ (READC)
                               (QUOTE F)))
                 (CONTROL NIL))
```

(* Salva las componentes al fichero correspondiente)*

```
(ACTUALIZA.COMPONENTES)
(CLOSEW (TTYDISPLAYSTREAM ANTERIOR])
```

(CARGAR.COMPONENTES
[LAMBDA NIL *(* edited: "30-Nov-87 11:07")*

(* Recupera la informacion referente a los componentes del sistema,)*

```
(LET* ([EXISTE? (NLSETQ (OPENSTREAM (QUOTE COMPONENTES-SISTEMA.ARB)
                                    (QUOTE INPUT]
        (NOMBRE-FICHERO (CAR EXISTE?)))
       (if EXISTE?
           then
```

(* Recupera la informacion de los componentes del sistema)*

```
                (SETPROPLIST (QUOTE COMPONENTES)
                             (READ NOMBRE-FICHERO))
                (SKIPSEPRS NOMBRE-FICHERO)
                (SETQ COMPONENTES (READ NOMBRE-FICHERO))
                (CLOSEF NOMBRE-FICHERO)
```

(* Arranque correcto del sistema)*

```
                T
  else
```

(* Arranque del sistema imposible)*

```
                NIL])
```

(BORRAR.COMPONENTES
[LAMBDA NIL *(* edited: "16-Nov-87 11:13")*

(* Elimina la informacion sobre los componentes del sistema)*

```
(SETQ COMPONENTES NIL)
(SETPROPLIST (QUOTE COMPONENTES)
             NIL])
```

(MODIFICA.COMPONENTES
[LAMBDA NIL *(* edited: "18-Nov-87 10:29")*

(* Realiza operaciones de acceso y modificacion de la informacion de los componentes del sistema a peticion de los ususarios)*

```
(LET ((MENU-PETICIONES (create MENU
                               TITLE ← " TIPO DE OPERACION A REALIZAR "
                               ITEMS ←(QUOTE ((" ANADIR NUEVO " (QUOTE ANADIR)
                                                " INCLUYE UN NUEVO COMPONENTE DEL SISTEMA ")
                                               (" ELIMINAR " (QUOTE ELIMINAR)
                                                " ELIMINAR LOS DATOS CORESPONDIENTE A UNA COMPONENTE ")
                                               ("CONSULTA" (QUOTE CONSULTA)
                                                "OBTIENE LA INFORMACION SOBRE LOS DATOS DE UNA COMPONENTE ")
                                               (" MODIFICACION" (QUOTE MODIFICA)
                                                   "MODIFICA LOS DATOS DE LA COMPONENTE ")
                                               (" FIN DE SESION " (QUOTE FINALIZAR)
                                                        " FINALIZA LA SESION DE AYUDA ")))
                               CENTERFLG ← T
```

```
                                    ITEMHEIGHT ← 40
                                    ITEMWIDTH ← 300
                                    MENUBORDERSIZE ← 3))
          (FIN))
         (until FIN do (SELECTQ (MENU MENU-PETICIONES)
                                    (ANADIR (ANADIR.COMPONENTE))
                                    (ELIMINAR (ELIMINAR.COMPONENTE (PREGUNTA
                                                                    " NUMERO DEL COMPONENTE ===> "
                                                                       (QUOTE FUNCION)
                                                                       (QUOTE (
                                                                    EXISTE.COMPONENTE)))
                                                                  T))
                                    [CONSULTA (CONSULTA.COMPONENTE (PREGUNTA
                                                                    " NUMERO DEL COMPONENTE ===> "
                                                                       (QUOTE FUNCION)
                                                                       (QUOTE (
                                                                    EXISTE.COMPONENTE]
                                    [MODIFICA (CAMBIA.COMPONENTE (PREGUNTA
                                                                    " NUMERO DEL COMPONENTE ===> "
                                                                       (QUOTE FUNCION)
                                                                       (QUOTE (
                                                                    EXISTE.COMPONENTE]
                                    (FINALIZAR (SETQ FIN T))
                                    (RINGBELLS])

(INICIALIZA.TIEMPOS.SISTEMA
  [LAMBDA NIL                                          (* edited: "30-Nov-87 11:09")

        (* * Inicializa los tiempos de funcionamiento)

    (LET [(ULTIMO (FGET.COMPONENTE.PROP (QUOTE ULTIMO]

        (* * Actualiza todas las componentes)

        (for COMPONENTE from 1 to ULTIMO do (FPUT.COMPONENTE COMPONENTE (QUOTE TIEMPO)
                                                             0))

        (* * Marca la ultima actualizacion)

        (FPUT.COMPONENTE.PROP (QUOTE ULTIMA-ACTUALIZACION)
                              (DATE])

(ACTUALIZA.TIEMPOS.COMPONENTES
  [LAMBDA NIL                                          (* edited: "30-Nov-87 11:10")

        (* * Actualiza el tiempo de funcionamiento de los componentes en funcion del tiempo transcurrido desde la ultima
        actualizacion, y del tiempo que lleva funcionado)

    (LET ((ULTIMO (FGET.COMPONENTE.PROP (QUOTE ULTIMO)))
          (HORAS-TRANSCURRIDAS (CALCULA.TIEMPO.DESDE.ULTIMA)))

        (* * Actualiza todas las componentes)

        (for COMPONENTE from 1 to ULTIMO do (FPUT.COMPONENTE COMPONENTE (QUOTE TIEMPO)
                                                             (PLUS (FGET.COMPONENTE
                                                                      COMPONENTE
                                                                      (QUOTE TIEMPO))
                                                                   HORAS-TRANSCURRIDAS)))

        (* * Marca la ultima actualizacion)

        (FPUT.COMPONENTE.PROP (QUOTE ULTIMA-ACTUALIZACION)
                              (DATE])

(INICIAR.SIGUIENTE.COMPONENTE
  [LAMBDA (PARAMETRO IDENTIFICACION IDENT-NEGADO)      (* edited: "25-Nov-87 11:16")

        (* * Inicializa los valores de un nuevo componente)
```

```
    (LET [(COMPONENTE (BQUOTE ((PARAMETRO (\, PARAMETRO))
                               (IDENTIFICACION (\, IDENTIFICACION))
                               (ID-NEGADO (\, IDENT-NEGADO))
                               (TIEMPO 0)
                               (ACTIVO NIL]
          (if (NULL COMPONENTES)
              then (SETQ COMPONENTES (LIST COMPONENTE))
            else (RPLACD (LAST COMPONENTES)
                         (LIST COMPONENTE])
```

(ANADIR.COMPONENTE
[LAMBDA NIL *(* edited: "25-Nov-87 11:16")*

(* Anade un nuevo componente al sistema)*

```
    (LET [(NUEVO (GENERAR.SIGUIENTE.COMPONENTE))
          [OPCION (PREGUNTA
                  " TIEMPOS MEDIOS HASTA EL FALLOS O INVERSO (PARAMETRO LAMBDA) =====> ( T/I )  "
                          (QUOTE PERTENENCIA)
                          (QUOTE (T I]
          (PARAMETRO (PREGUNTA " PARAMETRO DEL NUEVO COMPONENTE ===> " (QUOTE FUNCION)
                             (QUOTE (NUMBERP]
          (SETQ PARAMETRO (SELECTQ OPCION
                                   (T (QUOTIENT 1.0 PARAMETRO))
                                   (I PARAMETRO)
                                   (PRINTOUT T .TAB 5 "REVISA EL PROGRAMA " T .TAB 15
                                  " ANADIR.COMPONENTE ADMITIO UNA OPCION DE TIEMPOS NO CONSIDERADA ")))
          (INICIAR.SIGUIENTE.COMPONENTE PARAMETRO (PREGUNTA
                                    " DETALLES (EXPLICACION) ACERCA DEL NUEVO COMPONENTE ===>"
                                                  (QUOTE FUNCION-O-NIL)
                                                  (QUOTE (STRINGP)))
                                      (PREGUNTA " EXPLICACION SI NEGADO EL COMPONENTE ====> "
                                                (QUOTE FUNCION-O-NIL)
                                                (QUOTE STRINGP)))
          (ACTIVAR.COMPONENTE NUEVO)
          (PRINTOUT T T .TAB 10 "NUEVO COMPONENTE " NUEVO T])
```

(ELIMINAR.COMPONENTE
[LAMBDA (COMPONENTE PREGUNTA?) *(* edited: "30-Nov-87 11:10")*

(* Suprime la informacion del suceso basico del sistema especificado)*

(* La informacion en realidad no es suprimida por este procedimiento, queda simplemente inaccesible a traves de las primitivas de manejo de la estructura. Se prefiere conservar esta informacion para estudios posteriores de los componentes del sistema (estadisticos, o similares))*

```
    (if PREGUNTA?
        then (if (CONFIRMAR NIL)
                 then (FPUT.COMPONENTE COMPONENTE (QUOTE ACTIVO)
                                       NIL))
      else (FPUT.COMPONENTE COMPONENTE (QUOTE ACTIVO)
                            NIL])
```

(CONSULTA.COMPONENTE
[LAMBDA (COMPONENTE) *(* edited: "25-Nov-87 11:44")*

(* Muestra la informacion actualizada de un componente)*

```
    (if (EXISTE.COMPONENTE COMPONENTE)
        then (PRINTOUT T T .TAB 10 " SUCESO BASICO " , , COMPONENTE ,, T T T)
             (for PROPIEDAD in (QUOTE (IDENTIFICACION PARAMETRO TIEMPO ID-NEGADO))
                do (PRINTOUT T T .TAB 10 PROPIEDAD , , .TAB 30 (FGET.COMPONENTE COMPONENTE
                                                                                PROPIEDAD)
                                  T))
      else (PRINTOUT T .TAB 10 " EL COMPONENTE ESPECIFICADO HA SIDO ELIMINADO" T])
```

(CAMBIA.COMPONENTE

```
[LAMBDA (COMPONENTE)                                    (* edited: "30-Nov-87 11:11")

        (* * Actualiza la informacion correspondiente a una de los sucesos basicos)


    (LET ((MENU-PETICIONES (create MENU
                                   TITLE ← " PROPIEDAD A MODIFICAR "
                                   ITEMS ←(QUOTE ((" PARAMETRO " (QUOTE PARAMETRO)

                                                   "MODIFICACION DE LOS DATOS DE FIABILIDAD ")
                                                   (" IDENTIFICACION "
                                                     (QUOTE IDENTIFICACION)

                                                  " MODIFICA LA IDENTIFICACION DEL SUCESO BASICO "
                                                     (SUBITEMS (" AFIRMACION  " (QUOTE
                                                                                 IDENTIFICACION)

                                               " MODIFICA LA EXPLICACION ACERCA DEL SIGNIFICADO DEL COMPONENTE ")
                                                                    (" NEGACION  " (QUOTE ID-NEGADA)

" MODIFICA LA EXPLICACION SOBRE LA NEGACION DEL CONCEPTO ESPECIFICADO POR EL COMPONENTE. POR DEFECTO SE FO
 ANTEPONIENDO UN NO A LA AFIRMADA "
                                                                                          )))
                                                   (" TIEMPOS " (QUOTE TIEMPO)

                                                     "INICIALIZACION DE LOS DATOS DE TIEMPO DE FUNCIONAMIENTO ")
                                                   ("NO MODIFICACION " (QUOTE FINALIZAR)

                                                      "FINALIZA LA SESION DE MODIFICACIONES ")))
                                   CENTERFLG ← T
                                   ITEMHEIGHT ← 40
                                   ITEMWIDTH ← 300
                                   MENUBORDERSIZE ← 3))
          (FIN))
         (CONSULTA.COMPONENTE COMPONENTE)
         (until FIN do (SELECTQ (MENU MENU-PETICIONES)
                                [PARAMETRO (PRINTOUT T T .TAB 5 "DATOS ANTERIORES: "
                                                   (FGET.COMPONENTE COMPONENTE
                                                                    (QUOTE PARAMETRO))
                                                   T)
                                           (FPUT.COMPONENTE COMPONENTE (QUOTE PARAMETRO)
                                                            (PREGUNTA "NUEVO DATO ===> "
                                                                      (QUOTE FUNCION)
                                                                      (QUOTE (NUMBERP]
                                [IDENTIFICACION (PRINTOUT T T .TAB 5 "DATOS ANTERIORES: "
                                                          (FGET.COMPONENTE COMPONENTE
                                                                           (QUOTE
                                                                           IDENTIFICACION))
                                                          T)
                                                (FPUT.COMPONENTE COMPONENTE (QUOTE
                                                                             IDENTIFICACION)
                                                                 (PREGUNTA
                                                                       "NUEVO DATO ===> "
                                                                            (QUOTE FUNCION)
                                                                            (QUOTE
                                                                              (STRINGP]
                                [ID-NEGADA (PRINTOUT T T .TAB 5 "DATOS ANTERIORES: "
                                                   (FGET.COMPONENTE COMPONENTE
                                                                    (QUOTE ID-NEGADO))
                                                   T)
                                           (FPUT.COMPONENTE COMPONENTE (QUOTE ID-NEGADO)
                                                            (PREGUNTA "NUEVO DATO ===> "
                                                                      (QUOTE
                                                                          FUNCION-O-NIL)
                                                                      (QUOTE (STRINGP]
                                (TIEMPO (if (CONFIRMAR NIL)
                                            then

        (* * Para mantener la coherencia de la informacion sobre la ultima actualizacion, actualiza todas.
        (Necesario revisar sobre un sistema real, una vez decidido el sistema de actualizacion de los tiempos seleccionado)
)

                                                                    (ACTUALIZA.TIEMPOS.COMPONENTES)
```

```
                                                    (FPUT.COMPONENTE COMPONENTE (QUOTE
                                                                                  TIEMPO)
                                                                                 0)))
                                          (TIEMPO2 (if (CORROBORA)
                                                       then

        (* * Inicializacion del sistema : Coloca todos los tiempos a 0 horas)

                                                                  (INICIALIZA.TIEMPOS.SISTEMA)))
                                          (FINALIZAR (SETQ FIN T))
                                          (RINGBELLS])
)
```

(* Funciones de manejo de una estructura ligada a los componentes : los sucesos basicos asociados a cada componente en el arbol activo actualmente)

```
(DEFINEQ

(INICIALIZA.ASOCIADOS.REALES
  [LAMBDA NIL                                          (* edited: "23-Nov-87 14:47")

        (* * Inicializa la lista de especificaciones de nodos asociados a los componentes del sistema del arbol que voy a
        cargar)


    (LET ((ASOCIADOS (CONS)))
         (for INDICE from 1 to (FGET.COMPONENTE.PROP (QUOTE ULTIMO))
            do (TCONC ASOCIADOS (QUOTE NADA)))
         (FPUT.COMPONENTE.PROP (QUOTE ASOCIADOS-REALES)
                               (CAR ASOCIADOS])

(OBTEN.ASOCIADO.REAL
  [LAMBDA (NUMERO-REAL)                                (* edited: "23-Nov-87 14:52")

        (* * Obtiene el nodo del arbol de fallo ya asociado al componente del sistema)


        (* * Esta informacion se encuentra almacenada bajo la propiedad ASOCIADOS-REALES de los componentes del sistema)


    (CAR (NTH (FGET.COMPONENTE.PROP (QUOTE ASOCIADOS-REALES))
              NUMERO-REAL])

(PON.ASOCIADO.REAL
  [LAMBDA (REAL NODO)                                  (* edited: "23-Nov-87 14:50")

        (* * Coloca como asociado a la componente del sistema en el arbol actualmente activo el nodo especificado)


    (RPLACA (NTH (FGET.COMPONENTE.PROP (QUOTE ASOCIADOS-REALES))
                 REAL)
            (NODO.ASOCIADO NODO])

(QUITA.ASOCIADO.REAL
  [LAMBDA (REAL)                                       (* edited: "23-Nov-87 14:53")

        (* * Coloca como asociado a la componente del sistema en el arbol actualmente activo el nodo especificado)


    (RPLACA (NTH (FGET.COMPONENTE.PROP (QUOTE ASOCIADOS-REALES))
                 REAL)
            (QUOTE NADA])

(YA.NOMBRADO.REAL?
  [LAMBDA (NUMERO-REAL)                                (* edited: "23-Nov-87 14:52")

        (* * Comprueba si el componente del sistema especificado ya tiene algun otro componente del arbol de fallos
        asociado)
```

(* Esta informacion se encuentra almacenada bajo la propiedad ASOCIADOS-REALES de los componentes del sistema)*

```
    (NEQ (QUOTE NADA)
         (CAR (NTH (FGET.COMPONENTE.PROP (QUOTE ASOCIADOS-REALES))
                   NUMERO-REAL])
)
(DECLARE: DONTCOPY
  (FILEMAP (NIL (1833 6652 (ACTIVAR.COMPONENTE 1843 . 2291) (FGET.COMPONENTE 2293 . 2554) (
FPUT.COMPONENTE 2556 . 2888) (FGET.COMPONENTE.PROP 2890 . 3268) (FPUT.COMPONENTE.PROP 3270 . 3653) (
FREMOVE.COMPONENTE.PROP 3655 . 3924) (EXISTE.COMPONENTE 3926 . 4472) (TODO.COMPONENTE 4474 . 4873) (
TODO.SOBRE.COMPONENTES 4875 . 5143) (GENERAR.SIGUIENTE.COMPONENTE 5145 . 5563) (ACTUALIZA.COMPONENTES
5565 . 6259) (INICIALIZAR.COMPONENTES 6261 . 6650)) (6699 18621 (CONSTRUIR.COMPONENTES 6709 . 7794) (
CARGAR.COMPONENTES 7796 . 8629) (BORRAR.COMPONENTES 8631 . 8916) (MODIFICA.COMPONENTES 8918 . 10804) (
INICIALIZA.TIEMPOS.SISTEMA 10806 . 11377) (ACTUALIZA.TIEMPOS.COMPONENTES 11379 . 12254) (
INICIAR.SIGUIENTE.COMPONENTE 12256 . 12804) (ANADIR.COMPONENTE 12806 . 14037) (ELIMINAR.COMPONENTE
14039 . 14779) (CONSULTA.COMPONENTE 14781 . 15398) (CAMBIA.COMPONENTE 15400 . 18619)) (18779 21025 (
INICIALIZA.ASOCIADOS.REALES 18789 . 19328) (OBTEN.ASOCIADO.REAL 19330 . 19771) (PON.ASOCIADO.REAL
19773 . 20141) (QUITA.ASOCIADO.REAL 20143 . 20505) (YA.NOMBRADO.REAL? 20507 . 21023)))))
STOP
```

```
{DSK}<LISPFILES>ARBOLES>SIMULACION.LSP;35     30-Nov-87 11:53:47              Page 1

(FILECREATED "30-Nov-87 11:53:47" {DSK}<LISPFILES>ARBOLES>SIMULACION.LSP;35 9248

      changes to:  (VARS SIMULACIONCOMS)
                   (FNS S-JL INICIALIZAR.SISTEMA USUARIO)

      previous date: "20-Nov-87 19:10:48" {DSK}<LISPFILES>ARBOLES>SIMULACION.LSP;19)


(PRETTYCOMPRINT SIMULACIONCOMS)

(RPAQQ SIMULACIONCOMS ((* Procedimientos orientados a la realizacion de una simulacion del
                          funcionamiento del sistema para un caso real)
                       (FNS S-JL INICIALIZAR.SISTEMA APAGAR.SISTEMA INFORMACION.SISTEMA)
                       (* Funciones de manejo de tiempos)
                       (FNS TIEMPOS.CORRECTOS? CALCULA.TIEMPO.DESDE.ULTIMA CALCULA.TIEMPO.RESTA
                            USUARIO)))
```

(* Procedimientos orientados a la realizacion de una simulacion del funcionamiento del sistema para un caso real)

```
(DEFINEQ

(S-JL
  [LAMBDA NIL                                          (* edited: "30-Nov-87 11:16")
```

(* Funcion de arranque del sistema. Realiza las tareas de vigilancia y se encarga de la actualizacion de los tiempos de funcionamiento de los componentes)*

(* Se ha preferido detectar los fallos de funcionamiento en la simulacion no utilizar un sistema de interrupciones, sino realizar un pooling de los posibles fallos criticos del sistema global.*
Puede ser mas aconsejable la opcion de interrupciones en un sistema real. Esta funcion solo es valida para la realizacion de la simulacion)

(* El sistema podria haberse planteado como un bucle infinito, pero se ha preferido dar una opcion de apagar el sistema)*

```
    (LET* ((FIN NIL)
           (FALLO-1 NIL)
           (FALLO-2 NIL)
           (COMPONENTES)
           (TIEMPO-RESTA 3600)
           (MENU-SIMULACION (create MENU
                                    TITLE ← " SIMULACION "
                                    ITEMS ←(QUOTE ((" RUPTURA EN SECCION 1 " (QUOTE FALLO1)
                                                                  " FUNCION CRITICA ESPECIFICADA ")
                                                   (" RUPTURA EN SECCION 3" (QUOTE FALLO2)
                                                                  " FUNCION CRITICA ESPECIFICADA ")
                                                   (" INFORMACION DEL ESTADO DEL SISTEMA "
                                                     (QUOTE INFO)
                                                  " MUESTRA INFORMACION DEL ESTADO, TIEMPO DE FUNCIONAMIENTO ... ")
                                                   ("FINALIZAR SIMULACION " (QUOTE FIN)
                                                            " FINALIZA LA SIMULACION - DESCONECTA EL SISTEMA ")))
                                    CENTERFLG ← T
                                    ITEMHEIGHT ← 40
                                    ITEMWIDTH ← 300
                                    MENUBORDERSIZE ← 3))
           (NUEVA-SALIDA (CREATEW (QUOTE (750 465 300 300))
                                  "          ESTADO DEL SISTEMA" 2 T))
           (ANTERIOR-SALIDA (TTYDISPLAYSTREAM NUEVA-SALIDA)))
```

(* Inicializa el sistema (carga la informacion de los componentes, y del tiempo de funcionamiento, actualiza los tiempos de estos componentes))*

(* Se prefiere mantener la informacion de los componentes (y tiempos) en memoria, aun cuando no es estrictamente necesario, podrian ser cargadas, al igual que los arboles de fallo en el momento en que sean requeridos)*

```
(if (INICIALIZAR.SISTEMA)
    then (if (TIEMPOS.CORRECTOS?)
             then
```

(* Calcula el tiempo que falta hasta la siguiente actualizacion)*

```
                  (SETQ TIEMPO-RESTA (CALCULA.TIEMPO.RESTA))
```

(* Realiza la actualizacion de tiempos)*

```
                  (ACTUALIZA.TIEMPOS.COMPONENTES)
```

(* Salva la nueva informacion)*

```
                  (ACTUALIZA.COMPONENTES)
                  (CLEARW T)
                  (OPENW T)
```

(* Todo ha ido bien en la inicializacion. Arranca el sistema)*

```
                  (until FIN
                     do (during TIEMPO-RESTA timerUnits (QUOTE SECONDS)
                                until FIN
                                do
```

(* Pooling de comprobacion de fallos en el sistema)*

```
                                    (if FALLO-1
                                        then (DIAGNOSTICO.FALLO (QUOTE SECCION-1))
                                             (SETQ FALLO-1 NIL)
                                             (CLEARW T))
                                    (if FALLO-2
                                        then (DIAGNOSTICO.FALLO (QUOTE SECCION-3))
                                             (SETQ FALLO-2 NIL)
                                             (CLEARW T))
```

(* Introduccion de datos para poder realizar la simulacion. (puede plantearse leyendo los datos de un fichero. Es otra opcion de simulacion))*

```
                                    (if (USUARIO)
                                        then (CLEARBUF T T)
                                             (SELECTQ (MENU MENU-SIMULACION)
                                                      (FALLO1 (SETQ FALLO-1 T))
                                                      (FALLO2 (SETQ FALLO-2 T))
                                                      (INFO (INFORMACION.SISTEMA))
                                                      (FIN
```

(* Operaciones necesarias, antes de apagar el sistema. En la version actual unicamente se elimina la informacion de los componentes del sistema Podrian incluirse funciones tales como arrancar otro sistema de seguridad, o guardar cierta informacion del estado en que queda el sistema)*

```
                                                           (SETQ FIN T))
                                                      NIL)))
```

(* No se permiten interrupciones por fallo del sistema durante la actualizacion de los tiempos de funcionamiento de los componentes. Esto, en un sistema con un gran numero de componentes, puede no ser permisible, debido al riesgo que conlleva. Caben al menos dos soluciones, o bien realizar la actualizacion por grupos de componentes, con un tiempo de actualizacion permisible o bien permitir interrupciones (no seria valido el sistema de pooling utilizado actualmente) pero asegura la restauracion de los tiempos al estado en que se encontraban antes de comenzar la actualizacion que fue interrumpida)*

```
        (* * Actualiza cada hora. Calculo el tiempo que falta hasta la siguiente actualizacion)

                                        (SETQ TIEMPO-RESTA (CALCULA.TIEMPO.RESTA))

        (* * Realizo la actualizacion de tiempos)

                                        (ACTUALIZA.TIEMPOS.COMPONENTES)

        (* * Salva la nueva informacion)

                                        (ACTUALIZA.COMPONENTES))

        (* * Restaura el sistema)

                                (CLOSEW (TTYDISPLAYSTREAM ANTERIOR-SALIDA))
                        else (CLOSEW (TTYDISPLAYSTREAM ANTERIOR-SALIDA))
                               (PRINTOUT T T .TAB 10 " IMPOSIBLE ARRANCAR EL SISTEMA " T T .TAB 10 T T
                                       .TAB 10
                                     " INFORMACION ERRONEA SOBRE FECHA DE ULTIMA ACTUALIZACION "
                                       T T .TAB 16 "REVISE HORA DEL SISTEMA " T))
          else (RINGBELLS)
                 (CLOSEW (TTYDISPLAYSTREAM ANTERIOR-SALIDA))
                 (PRINTOUT T T .TAB 10 " IMPOSIBLE ARRANCAR EL SISTEMA " T T .TAB 10
                           " NO FUE ENCONTRADA INFORMACION DE COMPONENTES DEL SISTEMA "
                           T T T .TAB 10
                           " COMPRUEBE LA EXISTENCIA DEL FICHERO COMPONENTES-SISTEMA.ARB "
                           T])

(INICIALIZAR.SISTEMA
  [LAMBDA NIL                                                    (* edited: "30-Nov-87 11:16")

        (* * Recupera la informacion referente a los componentes del (sistema. No se ha considerado ninguna otra operacion
        de inicializacion))

    (CARGAR.COMPONENTES])

(APAGAR.SISTEMA
  [LAMBDA NIL                                                    (* edited: "16-Nov-87 11:14")

        (* * Operaciones necesarias a realizar antes de desconectar el sistema)

        (* * En la version actual de simulacion unicamente se ha previsto eliminar la informacion mantenida sobre los
        componentes del sistema)

    (BORRAR.COMPONENTES])

(INFORMACION.SISTEMA
  [LAMBDA NIL                                                    (* edited: "16-Nov-87 11:13")

        (* * Muestra algun dato significativo del funcionamiento del sistema)

    (PRINTOUT T .TAB 5 " NO DISPONIBLE INFORMACION ACTUALMENTE " T)
    (PRINTOUT T .TAB 8 "UTILIDAD NO IMPLEMENTADA "])
)


(* Funciones de manejo de tiempos)

(DEFINEQ

(TIEMPOS.CORRECTOS?
  [LAMBDA NIL                                                    (* edited: "13-Nov-87 14:52")
```

(* Comprueba si la hora actual es mayor que la hora de la ultima actualizacion, para comprobar si es posible los calculos de tiempos. Es preciso comprobar la consistencia de la informacion de tiempos)*

```
    (IGREATERP (IDATE (DATE))
               (IDATE (FGET.COMPONENTE.PROP (QUOTE ULTIMA-ACTUALIZACION])
```

```
(CALCULA.TIEMPO.DESDE.ULTIMA
  [LAMBDA NIL                                          (* edited: "18-Nov-87 10:22")
```

(* Calcula el tiempo (en horas) transcurrido desde la ultima actualizacion)*

```
    (QUOTIENT [DIFFERENCE (IDATE (DATE))
                          (IDATE (FGET.COMPONENTE.PROP (QUOTE ULTIMA-ACTUALIZACION]
              3600])
```

```
(CALCULA.TIEMPO.RESTA
  [LAMBDA NIL                                          (* edited: "18-Nov-87 14:04")
```

(* Calcula el tiempo que falta hasta el momento en que debe realizarse la siguiente actualizacion)*

```
    (LET ((VALOR (REMAINDER [DIFFERENCE (IDATE (DATE))
                                        (IDATE (FGET.COMPONENTE.PROP (QUOTE
                                                                     ULTIMA-ACTUALIZACION]
                            3600)))
         (if (ZEROP VALOR)
             then 3600
           else VALOR])
```

```
(USUARIO
  [LAMBDA NIL                                          (* edited: "24-Nov-87 10:10")
```

(* Comprueba si el usuario quiere introducir informacion para realizar o terminar la simulacion)*

```
    (KEYDOWNP (QUOTE A])
)
(DECLARE: DONTCOPY
  (FILEMAP (NIL (776 7583 (S-JL 786 . 6546) (INICIALIZAR.SISTEMA 6548 . 6857) (APAGAR.SISTEMA 6859 .
7252) (INFORMACION.SISTEMA 7254 . 7581)) (7627 9226 (TIEMPOS.CORRECTOS? 7637 . 8099) (
CALCULA.TIEMPO.DESDE.ULTIMA 8101 . 8468) (CALCULA.TIEMPO.RESTA 8470 . 8966) (USUARIO 8968 . 9224)))))
STOP
```

```
{DSK}<LISPFILES>ARBOLES>SITUACION.LSP;14     30-Nov-87 11:54:02              Page 1

(FILECREATED "30-Nov-87 11:54:02" {DSK}<LISPFILES>ARBOLES>SITUACION.LSP;14 3159

      changes to:  (VARS SITUACIONCOMS)
                   (FNS ANADIR.A.SITUACION)

      previous date: "23-Nov-87 20:25:14" {DSK}<LISPFILES>ARBOLES>SITUACION.LSP;1)


(PRETTYCOMPRINT SITUACIONCOMS)

(RPAQQ SITUACIONCOMS ((* Primitivas basicas y funciones de manejo de la estructura utilizada para
                         registrar la situacion actual de suposiciones hechas por el sistema al
                         realizar un diagnostico)
                      (FNS FGET.SITUACION ANADIR.A.SITUACION FGET.SITUACION.PROP
                           FPUT.SITUACION.PROP INICIALIZAR.SITUACION CONSIDERADO)))
```

(* Primitivas basicas y funciones de manejo de la estructura utilizada para registrar la situacion actual de suposiciones hechas por el sistema al realizar un diagnostico)

```
(DEFINEQ

(FGET.SITUACION
  [LAMBDA (PROPIEDAD)                                   (* edited: "16-Nov-87 11:28")

        (* * Obtiene de la situacion actual la propiedad o el dato especificado)


        (* * Situacion actual es la variable global utilizada para recordar las suposiciones hechas actualmente por el
        sistema)


    (CAR (NTH SITUACION-ACTUAL PROPIEDAD])

(ANADIR.A.SITUACION
  [LAMBDA (ELEMENTO)                                    (* edited: "30-Nov-87 11:12")

        (* * Anade a las suposiciones actuales el nuevo elemento. Devuelve el elemento)


    (if (NULL SITUACION-ACTUAL)
        then (SETQ SITUACION-ACTUAL (LIST ELEMENTO))
      else (RPLACD (LAST SITUACION-ACTUAL)
                   (LIST ELEMENTO)))
    (FPUT.SITUACION.PROP (QUOTE ULTIMO)
                         (ADD1 (FGET.SITUACION.PROP (QUOTE ULTIMO])

(FGET.SITUACION.PROP
  [LAMBDA (PROPIEDAD)                                   (* edited: "11-Nov-87 10:01")

        (* * Recupera el valor de una determinada propiedad de la situacion actual)


    (GETPROP (QUOTE SITUACION-ACTUAL)
             PROPIEDAD])

(FPUT.SITUACION.PROP
  [LAMBDA (PROPIEDAD VALOR)                             (* edited: "11-Nov-87 10:02")

        (* * Incluye una determinada propiedad en la situacion actual del diagnostico)


    (PUTPROP (QUOTE SITUACION-ACTUAL)
             PROPIEDAD VALOR])

(INICIALIZAR.SITUACION
  [LAMBDA NIL                                           (* edited: "16-Nov-87 11:37")

        (* * Inicializa la informacion sobre las consideraciones y suposiciones hechas por el sistema.
        La variable global SITUACION-ACTUAL contiene esta informacion)
```

```
    (SETQ SITUACION-ACTUAL NIL)
    (SETPROPLIST (QUOTE SITUACION-ACTUAL)
                 NIL)
    (FPUT.SITUACION.PROP (QUOTE ULTIMO)
                         0])
```

(CONSIDERADO

```
  [LAMBDA (DATO)                                        (* edited: "16-Nov-87 11:32")
```

(* Comprueba si el dato especificado (en la version actual como un numero) es una de las suposiciones que ha hecho el sistema)*

```
    (AND (NUMBERP DATO)
         (GREATERP DATO 0)
         (LEQ DATO (FGET.SITUACION.PROP (QUOTE ULTIMO])
)
(DECLARE: DONTCOPY
  (FILEMAP (NIL (799 3137 (FGET.SITUACION 809 . 1209) (ANADIR.A.SITUACION 1211 . 1719) (
FGET.SITUACION.PROP 1721 . 1999) (FPUT.SITUACION.PROP 2001 . 2288) (INICIALIZAR.SITUACION 2290 . 2747)
 (CONSIDERADO 2749 . 3135)))))
STOP
```