UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES

PLANTEL ARAGON

HERRAMIENTA BASADA EN REDES PROBABILISTICAS PARA EL DESARROLLO DE SISTEMAS EXPERTOS CON INCERTIDUMBRE

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

P R E S E N T A :

JESUS NORIEGA ROJAS

San Juan de Aragón, Edo. de Méx., 1994

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

VNIVERSIDAD NACIONAL
AVFNOMA DE
MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
ARAGÓN
DIRECCION

JESUS NORIEGA ROJAS
P R E S E N T E

En contestación a su solicitud de fecha 18 de marzo del año en curso, relativa a la autorización que se le debe conceder para que el señor profesor ING JUAN GASTALDI PEREZ pueda dirigirle el trabajo de Tesis denominado **"HERRAMIENTA BASADA EN REDES PROBABILISTICAS PARA EL DESARROLLO DE SISTEMAS EXPERTOS CON INCERTIDUMBRE"**, con fundamento en el punto 6 y siguientes del Reglamento para Exámenes Profesionales en esta Escuela, y toda vez que la documentación presentada por usted reune los requisitos que establece el precitado Reglamento; me permito comunicarle que ha sido aprobada su solicitud

Aprovecho la ocasión para reiterarle mi distinguida consideración

ATENTAMENTE
"POR MI RAZA HABLARA EL ESPIRITU"
San Juan de Aragón, Edo de Méx , Marzo 24 de 1993
EL DIRECTOR

M en I CLAUDIO C MERRIFIELD CASTRO

c c p Lic Alberto Ibarra Rosas, Jefe de la Unidad Académica
c c p Ing Juan Gastaldi Pérez, Jefe de Carrera de Ingeniería en Computación
c c p Ing Manuel Martínez Ortiz, Jefe del Departamento de Servicios Escolares
c c p Ing Juan Gastaldi Pérez, Asesor de Tesis

CCMC/AIR/ lcg

RESUMEN

Esta tesis surge de la necesidad de contar con una herramienta para el desarrollo de sistemas basados en conocimiento o sistemas expertos para diversas aplicaciones en el monitoreo y control de redes eléctricas. Algunas de las aplicaciones, como el diagnóstico de fallas, presentan incertidumbre en el conocimiento y los datos, por lo que es importante contar con una herramienta que maneje dicha incertidumbre en una forma adecuada (esta incertidumbre se debe a información incierta, imprecisa e inconfiable). Una técnica de reciente desarrollo que permite la representación y razonamiento con incertidumbre son las redes probabilísticas o redes bayesianas.

Se desarrolló una herramienta (Shell) basada en la técnica de redes probabilísticas. La herramienta le permite al usuario definir su modelo probabilístico en una forma amigable al contar con una interfaz gráfica en base a ventanas y menús. También facilita el uso del modelo en tiempo real al ser acoplada a un sistema de adquisición de datos. Para ello se contempla el desarrollo de las técnicas de propagación, aprendizaje paramétrico y evaluación.

El conocimiento es representado en una red probabilística. Esta se obtiene con ayuda del experto del área en donde será aplicada la herramienta, definiendo cada uno de los nodos y la relación de dependencias existentes entre ellos. Las probabilidades condicionales y *a priori* son obtenidas como una razón (número de casos), las cuales pueden ser obtenidas de datos del dominio (aprendizaje paramétrico). Adicionalmente se presenta la forma de evaluar los resultados derivados de la herramienta, es decir las probabilidades *a posteriori*, por medio de la *medida de Brier* y la capacidad predictiva del sistema (error promedio).

La herramienta fue desarrollada utilizando una metodología de desarrollo de *software* orientada a objetos con el lenguaje de programación C++. Cuenta con una interfaz gráfica amigable desarrollada con el paquete *Interface Architect* de *Hewlett Packard.*

También se presenta la aplicación de la herramienta en diversas áreas como: medicina y diagnóstico de fallas en redes eléctricas de distribución. Este trabajo fue desarrollado en las instalaciones del Instituto de Investigaciones Eléctricas en Cuernavaca, Morelos.

DEDICATORIAS

A Dios:

Por darme la vida y la fuerza necesaria para realizar este trabajo.

A mis Padres:

María Esther Rojas de Noriega
Jesus Noriega Cabrera

Como una muestra de amor y gratitud por brindarme una formación profesional, siempre con cariño, comprensión y ejemplo de superación.

A mis Hermanos:

Ana Luisa Noriega Rojas
Juan Antonio Noriega Rojas

Por su cariño y apoyo incondicional.

A mi Novia:

Tomasa Rodriguez Robles

Por todo su amor, comprensión y ayuda recibida durante el desarrollo de este trabajo. Como un inicio de todos nuestros planes.

AGRADECIMIENTOS

Quiero agradecer especialmente al Dr. Luis Enrique Sucar Succar, por su asesoría en la realización de este trabajo y por su amistad y apoyo recibido a lo largo de este período.

A todos los profesores de la Escuela Nacional de Estudios Profesionales "Aragón" por su ayuda en mi formación profesional, en especial al Ing. Juan Gastaldi Pérez por las facilidades obtenidas en la asesoría de este trabajo.

Agradezco al M.I. Oscar J. González Gómez y al Ing. Ramón Durán Hernández por su amistad y apoyo brindados para la realización de este trabajo.

A los investigadores y personal del Instituto de Investigaciones Eléctricas por la ayuda recibida durante mi estancia de becario.

A mis amigos Leticia, María Elena, Ismael, José Antonio y Francisco por su ayuda y amistad en todo momento.

A mi familia: a mi padre, a mi madre y a mis hermanos, por todo el apoyo y confianza que han tenido en mí durante el desarrollo de la tesis.

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

INTRODUCCION

INTELIGENCIA ARTIFICIAL Y SISTEMAS EXPERTOS

La Inteligencia Artificial **(IA)** es una rama de las ciencias de la computación que estudia los principios que hacen posible la inteligencia y, el diseño de máquinas inteligentes. Es decir, el estudio y diseño de sistemas que muestren características asociadas con la inteligencia humana, como pueden ser: comprensión del lenguaje natural, percepción, aprendizaje, razonamiento, manipulación, resolución de problemas, etc. La *IA* busca la simulación del comportamiento humano esto es, el descubrimiento de técnicas que permitan diseñar y programar máquinas las cuales emulen y extiendan nuestras capacidades mentales. Es aquí donde surgen los Sistemas Expertos **(SE)**, como un intento por imitar a un experto humano en algún dominio específico, siendo ésta el área de *IA* que ha tenido mayor éxito en aplicaciones prácticas. Los *SE* son capaces de resolver problemas que requieren conocimiento de un experto en algún área particular. Poseen el conocimiento de un dominio específico, por esta razón también son llamados *sistemas basados en conocimiento*. Entre las aplicaciones típicas de Sistemas Expertos se tienen tareas como diagnostico médico, localización de fallas en cierto tipo de equipos e interpretación de datos.

Una característica adicional que con frecuencia se requiere en los sistemas expertos, es su habilidad para trabajar con incertidumbre [Bratko 86]. La información correspondiente con algún problema a resolver, en ocasiones es incompleta, imprecisa y las relaciones en el dominio del problema son aproximadas. Lo anterior requerirá llevar a cabo algún tipo de razonamiento probabilístico.

SISTEMAS EXPERTOS

Un *Sistema Experto* (SE) es un sistema de computadora capaz de representar y razonar acerca de algún campo de conocimiento, con una vista para resolver problemas y dar consejos". Se distingue de otro tipo de programas de inteligencia artificial en que:

- Trabaja con material complejo que requiere una cantidad considerable de experiencia humana.

- Debe exhibir un alto desempeño en términos de velocidad y seguridad, de forma que sea una herramienta útil.

- Debe ser capaz de explicar y justificar soluciones y recomendaciones para convencer al usuario de que su razonamiento es correcto [Jackson 86].

Los principales componentes de un Sistema Experto son:

1.- Base de Conocimiento.
2.- Máquina o Motor de Inferencia.
3.- Memoria de Trabajo o Base de Hechos.
4.- Interfaz con el Usuario.
5.- Interfaz con el Experto.

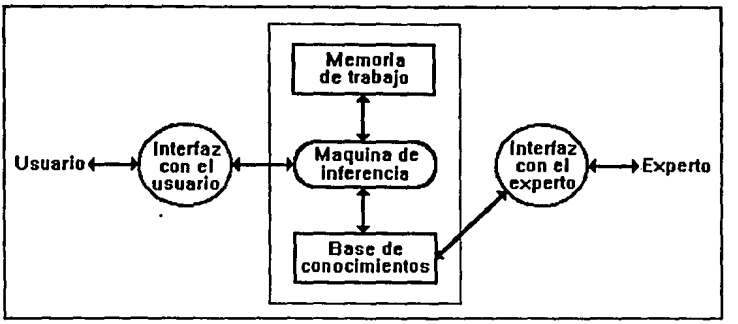


Figura 1. Componentes de un S.E.

En la figura 1 se muestra la interconexión de estos componentes dentro del sistema experto [Buchanan 84].

La *base de conocimiento (BC)*, comprende el conocimiento el cual es específico para una aplicación dada. Incluye información acerca del dominio, reglas que describen las relaciones o fenómenos en el dominio y posiblemente métodos, heurísticas e ideas para resolver el problema. El conocimiento tiene que ser representado de alguna forma, tal que pueda ser incluido en el sistema. Está representación puede realizarse en forma de *reglas de producción, árboles, gráficos, redes semánticas, prototipos o marcos de referencia (frames), lógica de predicados y redes probabilísticas.*

La *máquina o motor de inferencia*, utiliza el conocimiento almacenado en la base de conocimientos para resolver problemas (lleva a cabo el razonamiento). En adición a esta interacción con la base de conocimientos, la máquina de inferencia también registra los hechos conocidos acerca del problema en una base de datos llamada *memoria de trabajo (MT) o base de hechos*, la cual es actualizada conforme se tiene nueva información.

La *Interfaz con el usuario*, provee la comunicación entre el usuario y el sistema, permitiéndole al usuario observar el proceso de solución de problemas aplicado por la máquina de inferencia. En ocasiones es conveniente ver a la máquina de inferencia y a la interfaz como un módulo, usualmente llamada herramienta de sistemas expertos (*shell*) o simplemente herramienta [Bratko 86]. La base de conocimiento es única para cada dominio particular mientras que la máquina de inferencia puede ser común para aplicaciones similares.

La *Interfaz con el experto* realiza varias funciones entre estas se tienen: la inclusión del conocimiento del experto en la **BC** (adquisición del conocimiento), el mantenimiento del conocimiento con el fin de modificar e incrementar la **BC** de una forma sencilla y la validación y depuración del conocimiento al detectar inconsistencias y errores.

REPRESENTACION DEL CONOCIMIENTO

Una representación ha sido definida como un conjunto de reglas sintácticas y semánticas que permiten describir objetos [Winston 84].

En el campo de Sistemas Expertos, la representación del conocimiento implica alguna forma sistemática de codificación de lo que un experto conoce acerca de algún dominio. Esta representación requiere de una organización adecuada del conocimiento, de manera que este sea accesible y fácil de aplicar.

Los principales puntos para la evaluación de un lenguaje de representación son: lógica adecuada, poder heurístico y una notación conveniente. Lógica adecuada significa que el formalismo debe ser capaz de expresar el conocimiento que se desea representar. Poder heurístico denota que el lenguaje debe ser expresivo con reglas (sintácticas y semánticas) bien definidas que son aplicadas para representar el conocimiento, de manera que la construcción e interpretación del conocimiento sea el deseado. Notación conveniente es una virtud porque muchas de las aplicaciones de sistemas expertos requieren la codificación de grandes cantidades de conocimiento. Las expresiones resultantes deben ser fáciles de escribir y de leer, y debe ser posible entender su significado sin conocer como las interpreta la computadora.

Los principales formalismos que se han utilizado en el diseño de Sistemas Expertos son: *reglas de producción, objetos estructurados y lógica de predicados*. Estos formalismos son llamados "Sistemas de inferencia de modelo-dirigido" [Waterman 78].

REGLAS DE PRODUCCION

Las reglas de producción también son llamadas reglas de condición-acción o reglas de situación-acción. Su uso principal es en la codificación de asociaciones empíricas entre patrones de datos presentados al sistema y acciones que el sistema debe ejecutar como una consecuencia. Un Sistema de Producción consiste de un conjunto de reglas (algunas veces llamado Memoria

de Producción), un Interprete que decide como y cuando aplicar tales reglas, y una memoria de trabajo que puede retener datos, metas, o resultados intermedios.

Las reglas consisten de pares premisa-acción, por ejemplo:

Si P_1 & ... & P_n,
Entonces Q_1 & ... & Q_m,

Las cuales se leen *Si premisas* P_1 & ... & P_n son verdaderas entonces ejecutar acciones Q_1 & ... & Q_m. Las **P** son llamadas condiciones y a las **Q** se les conoce como conclusiones.

Las condiciones son usualmente tripletas del tipo:

"Objeto-Atributo-Valor" por ejemplo: (Pedro,edad,36)

Lo anterior significa, que el "valor" del "atributo" edad para el "objeto" Pedro es 36. Uno se puede imaginar una regla que incluya está condición como:

[regla 1] Si (Pedro edad 36) &
(Pedro empleado no)
Entonces (Pedro reclama BD)

BD=Beneficios de desempleado.

La regla anterior no es general, ya que se refiere a una persona y a cierta edad específica. Si se quisiera ampliar la sentencia para cualquier persona entre 15 y 65 años de edad, entonces será necesario introducir variables, las cuales no denoten objetos particulares o valores. Estas variables se igualarán con sus valores correspondientes y serán límite de ellos [Jackson 86].

La función básica de la memoria de trabajo (MT) es retener datos, en la forma de tripleta objeto-atributo-valor. Estos datos son usados por el interprete para conducir las reglas, en el sentido que, la ausencia o presencia de datos en la memoria de trabajo disparará algunas reglas, por satisfacer su patrón de activación.

Suponiendo que se tienen algunas tripletas en la memoria de trabajo, en el siguiente ciclo el interprete buscará para ver cuales reglas en la memoria de producción tienen condiciones las cuales pueden ser satisfechas. Si una condición contiene constantes, entonces es satisfecha sólo si una expresión idéntica está presente en la memoria de trabajo. Si una condición contiene una o más variables, entonces es satisfecha sólo si existe una expresión en la memoria de trabajo la cual se iguale en un camino que es consistente con la manera en la cual otras condiciones en la misma regla ya han sido igualadas.

El interprete para un conjunto de reglas de producción puede ser descrito en términos de lo que es llamado ciclo reconocer-actuar, este consiste de los pasos siguientes:

1.- Igualar los patrones de reglas contra los elementos de la memoria de trabajo.

2.- Si hay más de una regla que puede disparar, decidir cual aplicar; esto es llamado *resolución de conflicto.*

3.- Aplicar la regla. Posteriormente volver al paso uno.

En el punto 2, el sistema tiene un conjunto de parejas, que consisten de reglas y las variables ligadas derivadas de la igualación de patrones; esas parejas son llamadas instanciadas. La resolución de conflicto del sistema se encarga de determinar cual regla disparar. Es posible diseñar un conjunto de reglas tal que, para todas las configuraciones de datos, solo una regla se pueda disparar. Tales conjuntos de reglas son llamados "deterministas".

Los mecanismos de resolución de conflicto varían de un sistema a otro. A continuación se muestran tres mecanismos los cuales pueden ser usados en combinación, éstos son:

a) **Repetición.** Una regla no debe ser disparada más de una vez sobre el mismo dato. Así se eliminan del conflicto el conjunto de parejas instanciadas que ya han sido ejecutadas antes.

b) **Reglas Recientes.** Esta estrategia clasifica las pareja instanciadas en términos del tiempo de los elementos que toman parte en la igualación. Así las reglas que usan datos más recientes son preferidas a aquellas que tienen datos más antiguos.

c) **Reglas específicas.** Se prefieren parejas instanciadas las cuales son derivadas de reglas más específicas, ya que son más difíciles de satisfacer.

Las reglas de producción pueden ser conducidas hacia delante o hacia atrás. Es posible encadenar hacia delante de las condiciones conocidas como verdaderas, con respecto a las conclusiones que los hechos permiten establecer, igualando datos en la memoria de trabajo contra las premisas de la regla. También es posible encadenar hacia atrás de una conclusión que se desea establecer, en dirección de las condiciones necesarias para que sea verdad, observando si son soportadas por los hechos. En este caso, se igualan metas en la memoria de trabajo con la parte de acción de una regla.

OBJETOS ESTRUCTURADOS

El término "objetos estructurados" [Nilsson 82] se refiere a un esquema de representación cuya construcción fundamental es en bloques. Esta representación es análoga a nodos y arcos en teoría de gráficos o campos (slots) y contenidos (fillers) en estructura de registros. Todas las representaciones de objetos estructurados, son esencialmente agrupaciones de información de manera más o menos natural, que pueden ser aplicados a propósitos particulares.

Gráficos, Arboles y Redes

La terminología de teoría de gráficos a sido importada al área de la computación e IA para describir cierto tipo de datos estructurados. Se asume que existen entidades primitivas llamadas *nodos y ligas*. Los nodos son fuente y destino de ligas y usualmente tienen etiquetas para distinguirse entre ellos. Las ligas pueden o no tener etiquetas, dependiendo si hay diferentes tipos de ligas. Los nodos también son llamados vértices y las ligas arcos, bordes o líneas (a pesar de los diversos nombres, en este trabajo haremos referencia a ellos como nodos y ligas).

Si **N** es un conjunto de nodos, entonces algún subconjunto de **N x N** es una gráfica general, **G**. Si el orden de las parejas en **N x N** es material, **G** es una gráfica dirigida. Si se adiciona la restricción que las parejas deben contener nodos distintos, el resultado es una gráfica simple. Si **G** es una gráfica simple con **n** nodos, **n-1** ligas, sin circuitos, y además entre cualquier par de nodos sólo hay una trayectoria, entonces **G** es un árbol. Es usual designar a uno de los nodos como la raíz del árbol. La estructura de las ramas la forman los sucesores del nodo raíz. Los nodos que no tienen sucesores son llamados terminales u hojas de el árbol, mientras el resto son no-terminales.

En teoría de gráficos, una red es una gráfica dirigida pesada, es decir una gráfica dirigida con etiquetas numéricas asociadas a las ligas. Las etiquetas indican relaciones arbitrarias entre los nodos. Suponiendo que **L** es un conjunto de ligas etiquetadas y **N** es un conjunto de nodos, un subconjunto de **N x L x N** forma una red. Esto implica que dos nodos pueden estar conectados por más de una liga. Lo anterior representa el hecho que dos objetos se unen en más de una relación.

En general las gráficas, árboles y redes se utilizan en sistemas expertos para representar las relaciones existentes entre los hechos que forman el conocimiento en algún área dada, determinando así la estructura del mismo. No sólo señalan las relaciones directas implícitas en el conocimiento, también permiten establecer una clasificación del mismo conforme al criterio del experto.

Redes Semánticas

Son usadas para representar información la cual puede ser organizada en base a pequeñas estructuras, las cuales describen conceptos (u objetos) relacionados a un mismo hecho. Se denominan así porque originalmente se utilizaron para representar el significado de expresiones del lenguaje natural. El término "redes asociativas" esta más de acuerdo con respecto a la manera en que son utilizadas.

El uso de redes para la representación del conocimiento empezó con los trabajos de Quillian [68] sobre el entendimiento del lenguaje. Quillian fue de los primeros en sugerir que los aspectos relevantes de la memoria humana pueden ser modelados en términos de una red de nodos representando conceptos.

La información es estructurada en una red, en la cual los nodos representan conceptos, los cuales están interconectados por ligas que representan las relaciones entre ellos. La red forma una jerarquía de conceptos relacionados, en la cual cada uno es descrito en términos de otros conceptos y del tipo de liga que los relaciona. Un tipo importante de liga es una liga llamada **ISA** (es una) la cual especifica que un concepto o clase es un subconjunto de otro concepto. Esta relación permite la herencia de propiedades de clases a subclases, así no hay necesidad de representar todas las propiedades de un concepto explícitamente, ya que pueden ser inferidas de otros concepto. En redes grandes esto puede traer un ahorro considerable de almacenamiento, esta característica es llamada "economía cognitiva" [Jackson 86]. Otros tipos de ligas incluyen conjunción, disjunción y otras relaciones definidas para dominios diferentes. El proceso de inferencia involucra el desplazamiento en la red a lo largo de las ligas asociadas, usando la herencia de información de los niveles altos hacia abajo a lo largo de las ligas **ISA**.

Prototipos o Marcos de Referencia

Prototipos (Frames) fueron introducidos por Minsky [75] como una estructura para agrupar información. Minsky describe los *prototipos* como "estructuras de datos para la representación de situaciones de estereotipo". Este esquema surge de la idea de que la codificación conceptual en el cerebro humano tiene que ver con las propiedades notorias asociadas con objetos que son de alguna manera sobresalientes de su clase. Tales objetos son llamados *prototipos u objetos prototipos*.

Un Prototipo es descrito como una estructura de registro dividido en dos partes: campos de alto nivel y campos de bajo nivel. Los primeros son fijos y representan propiedades que son verdaderas para esa situación. Los campos de bajo nivel o terminales, son variables y contienen datos para instancias específicas de una situación típica. Asociado a un campo terminal pueden existir un conjunto de condiciones que deben ser satisfechas. Los campos terminales normalmente están rellenos con asignaciones por omisión (*default*).

El razonamiento esta basado en la localización del prototipo que mejor represente una situación dada y en la asignación de valores a los campos terminales [Sucar 92] . El proceso de igualación es controlado por la información en el prototipo y por los datos existentes.

LÓGICA DE PREDICADOS

La lógica ha sido estudiada por mucho tiempo como un lenguaje formal para representar razonamiento humano. Un tipo particular de lógica llamada cálculo de predicados de primer orden ha sido aplicado a sistemas expertos [Hammond 83]. Algunos Teoremas de *IA* usan una forma restringida de lógica de predicados, en la que todas las fórmulas son representadas como cláusulas de Horn y las inferencias están basadas en el principio de resolución [Robinson 65]. Las cláusulas de Horn son un esquema sintáctico restringido en el cual las expresiones lógicas tienen la forma:

$$P <- Q_1, \ldots Q_n,$$

donde **P** y Q_i son átomos y los Q_i están implícitamente asociados. Esta es una expresión condicional, con **P** como su consecuencia y la conjunción $Q_1 \& \ldots \& Q_n$ como su antecedente. Los átomos son predicados de la forma:

$$P(t_1, \ldots, t_m)$$

donde **P** es un predicado y los t_i son términos que pueden ser constantes, variables o funciones. Las expresiones del cálculo de predicados pueden ser transformadas a cláusulas de Horn por una transformación sintáctica [Jackson 86]. Así la base de conocimientos estará representada como un conjunto de cláusulas de Horn representando reglas de la forma: **Si** $Q_1 \& \ldots \& Q_n$ **Entonces P** [Sucar 92]. Es importante considerar que el criterio de Horn tiene dos puntos significativos: sólo puede tenerse una conclusión por regla y una regla no puede aparecer negada.

A diferencia de muchos sistemas deductivos los cuales utilizan una docena de reglas o más, en *IA* la comprobación de teoremas utiliza sólo una regla de inferencia llamada "principio de resolución" [Robinson 65]. En este para probar que la meta **P** sigue de un conjunto de proposiciones **S**, asumimos $\neg$**P** y formamos la conjunción **S** & $\neg$**P** y se trata de derivar una contradicción. Si lo anterior se logra exitosamente, se ha aprobado **P**, aunque si falla no puede decirse que se probo $\neg$**P**.

INCERTIDUMBRE

Los sistemas basados en conocimiento son aplicados en muchas áreas donde la información es incierta, imprecisa e inconfiable. Dicha incertidumbre surge de una variedad de fuentes, incluyendo datos incompletos o incorrectos, conocimiento impreciso o carencias del mecanismo de representación. Dado que la incertidumbre se encuentra en la mayoría de los dominios de interés de la vida diaria, surge la necesidad de desarrollar técnicas que permitan manejarla de una manera adecuada, ya que muchos sistemas expertos se encontrarán con este tipo de información. Del manejo adecuado de la información con incertidumbre depende el buen desempeño del sistema y la credibilidad en las decisiones que de él se deriven.

Se han propuesto diversos paradigmas para el manejo de incertidumbre en sistemas expertos. Tres de las teorías más completas para propagación de creencia son: *probabilidad subjetiva, teoría de Dempster-Shafer y lógica difusa* (también conocida como lógica borrosa), así como algunas técnicas desarrolladas específicamente para sistemas expertos como *MYCIN* con *factores de certeza y PROSPECTOR* con *método subjetivo Bayesiano*. Cada una de las técnicas mencionadas ha tenido un buen desempeño dentro de el área específica para la que fueron creadas, pero presentan problemas al extenderse a otros dominios. Estas técnicas tiene sus limitantes, ya que en algunas se suponen independencias condicionales o mutua exclusividad de los datos, otras requieren que se lleven a cabo funciones de ajuste *Ad Hoc* (funciones para aquello de que se trata). En algunas técnicas se requiere almacenar gran cantidad de información y/o realizar cálculos complejos, siendo esta una restricción con respecto a los requerimientos de memoria y cómputo.

Una técnica desarrollada recientemente, la cual elimina algunas de las deficiencias de las técnicas mencionadas, se conoce como *Redes Probabilísticas o Redes Bayesianas*. Estas permiten la representación y el razonamiento con incertidumbre en base a teoría de probabilidad, obteniendo así una mayor consistencia en el cálculo de las probabilidades *a posteriori* de las variables de interés.

OBJETIVO DE LA TESIS

Este trabajo surge de la necesidad de contar con una herramienta para el desarrollo de sistemas basados en conocimiento (sistemas expertos) para diversas aplicaciones en el monitoreo y control de redes eléctricas. Algunas de estas aplicaciones, como el diagnóstico de fallas, presentan incertidumbre en el conocimiento y los datos, por lo que es importante disponer de una herramienta que maneje dicha incertidumbre en forma adecuada.

El objetivo de la presente tesis es el desarrollar una herramienta para el manejo de incertidumbre basada en la técnica de redes probabilísticas, consiguiendo con ello una adecuada representación y razonamiento con incertidumbre, fundamentado en teoría de probabilidad. Dicha herramienta permitirá al usuario definir su modelo probabilístico de una forma amigable, al contar con una interfaz al usuario gráfica. También facilitará el uso del modelo en tiempo-real al ser acoplado al sistema de adquisición de datos. Para ello se aplicarán las técnicas para propagación de probabilidades, aprendizaje y evaluación dentro de la herramienta.

DESCRIPCION DE LA TESIS

Ya hemos hablado de algunos formalismos para la representación del conocimiento. En el capítulo siguiente, hablaremos de diversas técnicas para el manejo de incertidumbre y las características de cada una de ellas, así como algunas de sus desventajas.

En el capítulo 2 presentamos la técnica de redes probabilísticas, mostrando los fundamentos teóricos de ésta, así como un ejemplo donde se aplica la teoría. Adicionalmente se muestra la manera de actualizar las probabilidades condicionales y *a priori* al realizar aprendizaje paramétrico, y la forma de evaluar los resultados derivados de la herramienta.

En el capítulo 3 realizamos la descripción de la herramienta para su aplicación en sistemas expertos con incertidumbre, recalcando las características deseadas en ella para trabajar con redes probabilísticas.

El capítulo 4 presenta la especificación de la herramienta haciendo énfasis en los aspectos de interfaz al usuario de tal forma que el ambiente de trabajo sea amigable.

En el capítulo 5 presentamos el diseño de la herramienta utilizando el *método de Booch* para desarrollo de *software* orientado a objetos. Además describimos brevemente aquellos

aspectos de la metodología que son necesarios para llevar a cabo el diseño de la herramienta.

El capítulo 6 contiene pruebas y ejemplos de aplicación de la herramienta, mostrando la estructura de la red probabilística, las probabilidades condicionales, probabilidades *a priori* y los cálculos requeridos para obtener los resultados.

Finalmente se presentan las conclusiones a este trabajo.

CAPITULO 1

MANEJO DE INCERTIDUMBRE EN SISTEMAS EXPERTOS

1.1. INTRODUCCION

Los sistemas basados en conocimiento ó sistemas expertos son aplicados en áreas donde la información es incierta, tales como medicina, reconocimiento del lenguaje, visión, diagnóstico de fallas, entre otras. Por lo que esta información incierta, llamémosla incertidumbre, debe ser manejada adecuadamente.

La incertidumbre surge de diversas fuentes, entre estas se encuentran: información incompleta e inconfiable, conocimiento contradictorio e impreciso y carencia en el poder descriptivo de los formalismos de representación del conocimiento. Lo primero se debe a limitaciones prácticas cuando se adquiere la información. El conocimiento impreciso y contradictorio, surge de una definición inadecuada de los conceptos del dominio y de la dificultad de obtener un consenso en los puntos de vista de los expertos de los cuales se obtiene el conocimiento. Por último la carencia descriptiva en los formalismos, provoca una mala traducción del conocimiento del experto a un lenguaje formal.

Todo buen sistema experto debe ser capaz de manejar incertidumbre, ya que cualquier dominio de interés contiene datos que por lo general son inexactos, incompletos o imprecisos. Se han propuesto diversos paradigmas para el manejo de incertidumbre en sistemas expertos, unos cualitativos y otros cuantitativos. Los cualitativos o no-numéricos se dividen en formales y heurísticos. Los formales están constituidos por **lógicas no-monotónicas**, donde es posible retractarse de algunas hipótesis que ya se habían dado por ciertas al recibir nueva información que describa mejor el problema. En los heurísticos tenemos la **teoría de endoso de Cohen** [Cohen 85] donde se manejan razones para creer o no en una hipótesis.

Los métodos cuantitativos o numéricos incluyen: *teoría de probabilidad subjetiva* [Finetti 74 y Lindley 87], *lógica difusa* (también es conocida como lógica borrosa) [Zadeh 78], *teoría de Dempster-Shafer* [Shafer 76], *factores de certeza* [Buchanan 75] y *métodos Bayesianos subjetivos (Prospector)* [Duda 76]. En lógica difusa se expresa la incertidumbre en términos lingüísticos por medio de intervalos difusos. La *teoría de Dempster-Shafer* permite medir evidencias inciertas en intervalos de creencia y combinarlas a través de la *regla de Dempster.* En *factores de certeza* se asigna un factor el cual denota la creencia en una hipótesis, dada la evidencia observada. Por último *Prospector* maneja incertidumbre apoyándose en teoría de decisión Bayesiana y en la regla de Bayes.

1.2 METODOS NO NUMERICOS

Aunque existen diversos métodos no numéricos para el manejo de incertidumbre, únicamente mencionaremos el método basado en la teoría de *endoso de Cohen*, ya que es el que tiene mayor relevancia respecto al método que utilizamos en este trabajo. Para un estudio de otros métodos referirse a McDermott [80], Doyle [79] y DeKleer[86].

1.2.1. Teoría de endoso

La idea fundamental de la *teoría de endoso* [Cohen 85] se basa en el hecho de que el conocimiento acerca de situaciones inciertas debe influenciar el funcionamiento del sistema. Un paso importante hacia esta meta es el diseño de la tarea de resolución, para decidir que hacer cuando una evidencia necesaria falta. Una tarea de resolución es aquella que reduce la incertidumbre al obtener más información. Aunado a lo anterior, se tiene la idea que la incertidumbre debe afectar las decisiones de control. Cuando un sistema basado en endosos programa tareas en su agenda, lo hace preguntando que aportará cada una de ellas para incrementar su certeza. De esta manera las decisiones de control pueden no resolver los problemas de incertidumbre, pero aprovechan el conocimiento con incertidumbre para facilitar la resolución.

Todas las razones para creer (o no creer) en una hipótesis son representadas en estructuras llamadas *endosos*. Los endosos son prototipos, como estructuras de conocimiento, representando razones para creer (endosos positivos) o no creer (endosos negativos) en una hipótesis particular. Son asociadas varias veces con proposiciones y reglas de inferencia durante el razonamiento.

Cinco clases de endosos son importantes para el razonamiento con incertidumbre en sistemas basados en reglas [Cohen 85]:

- **Endoso de reglas.** Razones para creer (o no creer) en reglas de inferencia.
- **Endoso de datos.** Razones para creer (o no creer) en datos puros.
- **Endoso de tareas.** Argumentos acerca de la evidencia que se producirá al ejecutar la tarea, usada por el programa de tareas.
- **Endoso de conclusiones.** Razones para creer (o no) en unas conclusiones. Estas son combinaciones de una regla de endoso y de relaciones detectadas, tal como corroboración entre conclusiones.
- **Endoso de resolución.** Registros de métodos de aplicación para resolver incertidumbre.

El razonamiento llevado a cabo por estos endosos se asemeja al razonamiento de muchos sistemas expertos. Así, se comienza tratando de concluir una meta y entonces se encadena hacia atrás por medio de su regla base. Conforme el razonamiento avanza, nuevos grupos de endosos

se desarrollan, a partir de las razones para creer (o no creer) en sus conclusiones. Estos nuevos endosos proporcionan justificaciones para las conclusiones. Es importante que los endosos afecten la agenda ya que la teoría de endoso esta orientada hacia los efectos de la incertidumbre sobre el funcionamiento. Estos efectos son dobles. Primero el sistema utiliza endosos para decidir si una proposición es cierta, preguntando si los endosos de una conclusión-submeta fueron bastante buenos para garantizar el uso de la conclusión para afirmar su meta padre. Segundo, el sistema usa tareas de resolución para reducir la incertidumbre obteniendo más información por medio de endosos. El principio de estas tareas es que los endosos negativos son vistos como problemas a ser resueltos. Así proveen el razonamiento requerido para disminuir la incertidumbre.

En adición a reglas para decidir cuando una proposición es suficientemente cierta para una tarea y reglas para resolver incertidumbre, la teoría de endoso tiene una regla simple para combinar los endosos y propagarlos sobre inferencias. Esta regla tiene dos problemas: las razones para creer o no creer en una premisa no siempre son endosos de la conclusión y la regla puede conducir a grandes cuerpos de endosos después de unas cuantas inferencias. Cohen y su grupo de la Universidad de Massachusetts están explorando nuevos métodos, tales como reglas de combinación semántica para endosos, que intentan resolver estos y otros problemas relacionados.

1.3. METODOS NUMERICOS

Antes de introducir las técnicas numéricas, se examinan algunos conceptos básicos de probabilidad. Para la demostración de las ecuaciones que se mencionan en la siguiente sección referirse a [Neapolitan 90] y [Abramson 90], o a cualquier libro de probabilidad estándar.

1.3.1. Teoría de probabilidad

Considerando un experimento que tiene un conjunto de alternativas dadas, las cuales son mutuamente exclusivas y exhaustivas. Al conjunto de alternativas o resultados se le conoce como espacio de prueba ó espacio muestral y es denotado por Ω. Mutuamente exclusivo y exhaustivo significa, que al menos una de las alternativas debe ocurrir y ellas no pueden ocurrir simultáneamente.

Siendo Ω un conjunto finito de puntos de prueba y F un conjunto de eventos relativos a Ω. Para cada $E \in F$, corresponde un número real **P(E)**, llamado *probabilidad de E*. Este número es obtenido por dividir el número de alternativas equiposibles favorables a un evento **E**, entre el número total de alternativas equiposibles. La probabilidad de un evento es denotada por **P(E)** y debe satisfacer lo siguiente

1.- $\mathbf{P(E)} \geq 0$ para $E \in F$.

2.- $P(\Omega) = 1$.

3.- Si E_1 y E_2 son subconjuntos disjuntos de F, entonces

$$P(E_1 \cup E_2) = P(E_1) + P(E_2)$$

De acuerdo a (3) la probabilidad de que al menos uno de estos eventos ocurra es la suma de las probabilidades individuales, o,

$$P(A_1 \cup A_2 \cup ... \cup A_K) = \sum_{i=1}^{K} P(A_i) \qquad (1.1)$$

La probabilidad de algún evento, siempre estará entre 0 y 1, es decir $0 \leq P(E) \leq 1$, donde $P(E)$ es un número real. Por definición, cuando $P(E)=0$, el evento E nunca ocurre y cuando $P(E)=1$, el evento E debe ocurrir. El complemento de E, ($\neg E$) contiene la colección de todos los eventos elementales de Ω excepto E. Dado que E y $\neg E$ son mutuamente exclusivas y $E \cup \neg E = \Omega$ de la ecuación (1.1) tenemos

$$\begin{aligned} P(E) + P(\neg E) &= P(E \cup \neg E) \\ &= P(\Omega) \\ &= 1 \end{aligned} \qquad (1.2)$$

Escribiendo esta ecuación como $P(\neg E)=1-P(E)$ tenemos una forma fácil de calcular $P(\neg E)$ de $P(E)$.

Todas las probabilidades están basadas en información. Cuando un espacio de probabilidad es creado, llamamos a las probabilidades las cuales están basadas en información inicial probabilidades "*a priori*" (se utiliza el término *a priori* para indicar que son independientes de experiencia). Las probabilidades basadas en información adicional son llamadas probabilidades condicionales.

Suponer que E_1, $E_2 \in F$, además $P(E_1)>0$. La probabilidad de E_2 dado que E_1 ocurre, se escribe $P(E_2|E_1)$ y es llamada la probabilidad condicional de E_2 dado E_1, la cual es definida como sigue.

$$P(E_2|E_1) = \frac{P(E_1 \cap E_2)}{P(E_1)} \qquad (1.3)$$

El término $E_1 \cap E_2$, es la probabilidad que E_1 y E_2 ocurran simultáneamente y es llamada probabilidad conjunta.

Si alguna de las siguientes condiciones se cumple:

1.- $P(E_1) = 0$ o $P(E_2)=0$

2.- $P(E_2|E_1) = P(E_2)$.

se dice que E_2 es independiente de E_1.

Claramente E_2 es independiente de E_1, si al conocer que E_1 ha ocurrido, no cambia la probabilidad de E_2 y viceversa. Así E_1 es independiente de E_2 si E_2 es independiente de E_1.

Aunado a lo anterior E_1 y E_2 son independientes si y sólo si:

$$P(E_1 \cap E_2) = P(E_1)P(E_2) \tag{1.4}$$

La prueba se deriva de la ecuación (1.3) y de la hipótesis de independencia.

La ecuación (1.3), conduce a un poderoso teorema, llamado *teorema de Bayes* en honor a su creador, Thomas Bayes (1702-1761). Antes de definir este teorema es necesario determinar los siguientes conceptos. Permitiendo a Ω ser un espacio de prueba y $\{E_1,E_2,\ldots,E_n\}$ ser un conjunto de eventos tal que para $i \neq j$,

$$E_i \cap E_j = \varnothing \quad y \quad \bigcup_{i=1}^{n} E_i = \Omega \tag{1.5}$$

De acuerdo con (1.5) se dice que los eventos en $\{E_1,E_2,\ldots,E_n\}$ son mutuamente exclusivos y exhaustivos. Considerando además que $1 \leq i \leq n$ y que $P(E_i) > 0$. Para algún $E \in F$, se tiene:

$$P(E) = \sum_{i=1}^{n} P(E|E_i)P(E_i) \tag{1.6}$$

Demostración. Debido a que los E_i's son exhaustivos, tenemos que

$$E = (E \cap E_1) \cup (E \cap E_2) \cup \ldots \cup (E \cap E_n).$$

y debido a que los E_i's son mutuamente exclusivos, por la ecuación (1.1) tenemos:

$$P(E) = (E \cap E_1) + (E \cap E_2) + \ldots + (E \cap E_n).$$

Así la prueba de la ecuación (1.6) se obtiene de la igualdad anterior y de (1.3).

Teorema de Bayes. Considerando el espacio de muestra Ω y un conjunto de eventos mutuamente exclusivos y exhaustivos en F $\{E_1,E_2,\ldots,E_n\}$, tal que para $1 \leq i \leq n$, $P(E_i) > 0$. Además, para algún $E \in F$ tal que $P(E) > 0$, tenemos para $1 \leq j \leq n$ que:

$$P(E_j|E) = \frac{P(E|E_j)P(E_j)}{\sum_{i=1}^{n} P(E|E_i)P(E_i)} \tag{1.7}$$

La demostración resulta de las ecuaciones (1.3) y (1.6).

Si E y E' son dos eventos tal que P(E) y P(E') son ambos positivos, entonces la siguiente igualdad sigue de la ecuación (1.3):

$$P(E|E') = \frac{P(E'|E)P(E)}{P(E')} \tag{1.8}$$

Esta ecuación es llamada algunas veces teorema de Bayes. Aquí se le referencia como "definición de probabilidad condicional".

1.3.2. Probabilidad subjetiva

Después de analizar la teoría de probabilidad básica, entramos de lleno a la aplicación de probabilidad subjetiva en sistemas expertos. Esta se basa en la teoría ya mencionada, principalmente en el teorema de Bayes.

Para entender mejor el desarrollo de esta teoría cambiaremos la notación en términos de hipótesis (**H**) y evidencias (**E**).

Si **H** y **E** son dos conjuntos disjuntos, **E** puede ser expresada como:

$$E = (E \cap H) \cup (E \cap \neg H)$$

de aquí tenemos que,

$$\begin{aligned} P(E) &= P((E \cap H) \cup (E \cap \neg H)) \\ &= P(E \cap H) + P(E \cap \neg H) \\ &= P(E|H)P(H) + P(E|\neg H)P(\neg H) \end{aligned}$$

Cambiando la ecuación (1.8) a hipótesis y evidencias y considerando el valor anterior de **P(E)**, obtenemos:

$$P(H|E) = \frac{P(E|H)P(H)}{P(E|H)P(H) + P(E|\neg H)P(\neg H)} \qquad (1.9)$$

La ecuación (1.9) es la base de probabilidad para el manejo de incertidumbre y proporciona una forma de obtener la probabilidad condicional de **H** dado **E** de la probabilidad condicional de **E** dado **H**.

Consideremos un caso en el que todas las reglas en el **SE** están expresadas en la forma: **Si H es verdadera Entonces E será observada con probabilidad P** ". Si **H** es observada, esta regla establece que la probabilidad que el evento **E** ocurra es **P**. Si **H** es desconocida y **E** es observada, la ecuación (1.9) muestra como calcular la probabilidad considerando que **H** es verdadera. La ecuación (1.9) relaciona una hipótesis (**H**) a un fragmento de evidencia de soporte (**E**), y también relaciona la evidencia observada a una hipótesis insubstanciada.

En sistemas expertos, las probabilidades que son requeridas para resolver un problema, son proporcionadas por el experto humano y almacenadas en la base de conocimiento. Estas probabilidades incluyen las probabilidades *a priori* para todas las posibles hipótesis (**P(H)**) y las probabilidades condicionales para la observación de un fragmento de evidencia dada una hipótesis (**P(E|H)**). En diagnóstico médico, por ejemplo, un médico especializado debe proveer las probabilidades *a priori* de todas las posibles enfermedades para un dominio médico particular. Además proporciona las probabilidades condicionales para la observación de un síntoma dada una enfermedad específica, para todas las parejas de síntomas y enfermedades (asumiendo que todos los síntomas son condicionalmente independientes dada una enfermedad). El usuario da al experto información acerca de las observaciones (la presencia de ciertos síntomas), y el sistema calcula $P(H_i|E_j...E_k)$ para todas las hipótesis $(H_i...H_m)$ de acuerdo con

los síntomas proporcionados ($E_j...E_k$) y las probabilidades almacenadas [Abramson 90]. La probabilidad $P(H_i|E_j...E_k)$ es llamada probabilidad *a posteriori* de la hipótesis H_i tras la observación de $E_j,...,E_k$.

La ecuación (1.9) es limitada ya que cada pieza de evidencia solo afecta a una hipótesis. En la teoría mencionada en la sección anterior se cuenta con una ecuación que permite trabajar con una evidencia y múltiples hipótesis (mutuamente exclusivas y exhaustivas). Así, reescribiendo la ecuación (1.7) en términos de hipótesis y evidencias, tenemos:

$$P(H_i|E) = \frac{P(E|H_i)P(H_i)}{\sum_{k=1}^{m} P(E|H_K)P(H_K)} \qquad (1.10)$$

Considerando ahora evidencias múltiples e hipótesis múltiples (mutuamente exclusiva y exhaustivas), tenemos:

$$P(H_i|E_1E_2...E_n) = \frac{P(E_1E_2...E_n|H_i)P(H_i)}{\sum_{k=1}^{m} P(E_1E_2...E_n|H_k)P(H_k)} \qquad (1.11)$$

Desafortunadamente la ecuación (1.11) es impráctica para muchas aplicaciones, ya que el denominador, requiere que se conozcan las probabilidades condicionales de todas las posibles combinaciones de evidencias para todas las hipótesis. Para aligerar la carga se asumen independencias condicionales entre fragmentos de evidencia dada una hipótesis lo cual provoca que (1.11) se reduzca a:

$$P(H_i|E_1E_2...E_n) = \frac{P(E_1|H_i)P(E_2|H_i)...P(E_n|H_i)P(H_i)}{\sum_{k=1}^{m} P(E_1|H_k)P(E_2|H_k)...P(E_n|H_k)P(H_k)} \qquad (1.12)$$

1.3.2.1. Actualización. La creencia es propagada en un sistema por medio de la regla de Bayes, al calcular todas las probabilidades *a posteriori* de las hipótesis, dada la evidencia observada. Las probabilidades *a posteriori* proporcionan información acerca de la verdad de una hipótesis.

A continuación se muestra un ejemplo para ilustrar el procedimiento [Abramson 90]. Suponer que en un sistema hay tres hipótesis mutuamente exclusivas y exhaustivas H_1, H_2 y H_3, las cuales tienen probabilidades *a priori* $P(H_1)$, $P(H_2)$ y $P(H_3)$, respectivamente. Entonces observamos dos fragmentos de evidencia condicionalmente independientes E_1 y E_2, que apoyan estas hipótesis en diferentes grados. La tabla 1-1 muestra las probabilidades condicionales y *a priori* para todas las hipótesis y evidencias en este ejemplo.

	H_1	H_2	H_3
$P(H_i)$	0.5	0.3	0.2
$P(E_1 \mid H_i)$	0.4	0.8	0.3
$P(E_2 \mid H_i)$	0.7	0.9	0.0

Tabla 1-1. Probabilidades *a priori* y Condicionales.

Cuando la evidencia es obtenida, la creencia en una hipótesis se incrementa si la evidencia la apoya y se decrementa si la evidencia se opone a ella. Suponer que se observa $\mathbf{E_1}$ antes que $\mathbf{E_2}$. Observando $\mathbf{E_1}$, es posible calcular las probabilidades *a posteriori* para las hipótesis de acuerdo a la ecuación (1.10) y considerando a i=1,2,3. Así,

$$P(H_1|E_1) = \frac{0.4x0.5}{0.4x0.5+0.8x0.3+0.3x0.2} = 0.40$$

$$P(H_2|E_1) = \frac{0.8x0.3}{0.4x0.5+0.8x0.3+0.3x0.2} = 0.48$$

$$P(H_3|E_1) = \frac{0.3x0.2}{0.4x0.5+0.8x0.3+0.3x0.2} = 0.12$$

Después que $\mathbf{E_1}$ es observada, la creencia en las hipótesis $\mathbf{H_1}$ y $\mathbf{H_3}$ se decrementa, mientras que la creencia en $\mathbf{H_2}$ se incrementa. Después de observar $\mathbf{E_2}$, calculamos las probabilidades *a posteriori* utilizando la fórmula (1.11) y considerando que $\mathbf{E_1}$ y $\mathbf{E_2}$ son condicionalmente independientes dado $\mathbf{H_i}$, lo cual produce.

$$P(H_i|E_1E_2) = \frac{P(E_1|H_i)xP(E_2|H_i)xP(H_i)}{\sum_{k=1}^{3} P(E_1|H_k)xP(E_2|H_k)xP(H_k)} \qquad i=1,2,3. \tag{1.13}$$

De (1.13) tenemos.

$$P(H_1|E_1E_2) = \frac{0.4x0.7x0.5}{0.4x0.7x0.5+0.8x0.9x0.3+0.3x0.0x0.2} = 0.393$$

$$P(H_2|E_1E_2) = \frac{0.8x0.9x0.3}{0.4x0.7x0.5+0.8x0.9x0.3+0.3x0.0x0.2} = 0.607$$

$$P(H_3|E_1E_2) = \frac{0.3x0.0x0.2}{0.4x0.7x0.5+0.8x0.9x0.3+0.3x0.0x0.9} = 0.0$$

Aunque las hipótesis iniciales fueron $\mathbf{H_1}$, $\mathbf{H_2}$ y $\mathbf{H_3}$, sólo $\mathbf{H_1}$, $\mathbf{H_2}$ se mantienen bajo consideración después que $\mathbf{E_1}$ y $\mathbf{E_2}$ fueron observadas; $\mathbf{H_2}$ es más probable que $\mathbf{H_1}$.

La principal dificultad en la aplicación de teoría de probabilidad subjetiva, es el alto

número de probabilidades que deben ser obtenidas para la construcción de una base de conocimientos funcional, además de la poca aceptación de la suposición de independencia y mutua exclusividad.

1.3.3. Teoría de Dempster-Shafer

La *teoría Dempster-Shafer* (**TDS**) fue desarrollada por Arthur Dempster [67] en los 60's y extendida por Glen Shafer [76] en los 70's. Esta teoría fue motivada por las dificultades que se encontraron al tratar de representar la ignorancia, y de la idea que la suma de la creencia subjetiva asignada a un evento y su negación deben sumar 1. La **TDS** no fija probabilidad a la negación de una hipótesis, una vez que la probabilidad de su ocurrencia es conocida, como ocurre en teoría de probabilidad ya que $P(H)+P(\neg H)=1$. Shafer considera que una evidencia la cual favorece parcialmente una hipótesis, no debe ser interpretada como que también apoya su negación [Shafer 76].

El concepto básico en la **TDS** es el marco de discernimiento, Θ, definido como un conjunto exhaustivo de eventos mutuamente exclusivos. Por ejemplo en un diagnóstico médico sobre ictericia [Gordon 85], en el cual existen cuatro posibles enfermedades: hepatitis (Hep), cirrosis (Cirr), cálculo biliar (Cal) y cáncer de páncreas (Pan), Θ tiene cuatro elementos. En **TDS**, Θ se asemeja al espacio de prueba Ω de teoría de probabilidad. La diferencia se encuentra, que en teoría Dempster-Shafer el número de hipótesis posibles es $|\ 2^{\Theta}\ |$, mientras que en teoría de probabilidad es $|\ \Omega\ |$. En el ejemplo tenemos $2^{\Theta}=16$ elementos, representando todos los posibles subconjuntos de Θ. En esta teoría la observación de una evidencia se opone a una hipótesis, sólo si la evidencia apoya la negación de la hipótesis. Así la evidencia que se opone a la hipótesis {Hep} (hepatitis y sólo hepatitis) es equivalente a la evidencia que apoya la hipótesis {Cirr,Cal,Pan} (todo excepto hepatitis).

Siendo **A** un subconjunto de Θ, el número de probabilidad básica de **A**, denotado **m(A)**, es la probabilidad asignada al conjunto **A**. La cantidad **m(A)**, puede interpretarse como la porción de creencia total asignada exactamente a **A**. Las funciones **P(A)** y **m(A)** difieren en que **A** es un elemento simple en teoría de probabilidad, mientras que en **TDS A** puede contener varios elementos. La función **m**, asigna a todos los elementos de Θ un valor entre 0 y 1. Esta función es llamada asignación de probabilidad básica (*apb*) si satisface dos propiedades:

1.- El número de probabilidad básico de un evento nulo es 0,

$$m(\varnothing)=0$$

2.- La suma de los números de probabilidad básica de todos los subconjuntos de Θ es 1,

$$\sum_{A\subseteq\Theta} m(A) = 1$$

En el ejemplo una posible asignación de probabilidad básica será: m({Hep})=0.3, m({Cirr})=0.2, m({cal})=0.1, m({Hep,Cirr})=0.4.

La creencia de **A**, **Bel(A)**, mide la cantidad total de creencia en **A**, no la cantidad

asignada precisamente a **A** por la asignación de probabilidad básica. Matemáticamente, puede ser expresada como.

$$Bel(A) = \sum_{B \subset A} m(B). \qquad (1.13)$$

La función **Bel** es llamada función de creencia si satisface lo siguiente:

1.- La creencia en una hipótesis nula es 0, o **Bel(∅)=0.**

2.- La creencia en Θ es 1, o **Bel(Θ)=1.**

3.- La suma de las creencias de **A** y ¬**A** debe ser menor o igual a 1, o **Bel(A)+Bel(¬A)≤1** (**A** significa **A** y sólo **A**, mientras que ¬**A** significa todo excepto **A**).

Cuando un conjunto contiene sólo un elemento, **Bel** es igual a **m**, pero **Bel** es mayor o igual a **m** para los conjuntos que contienen más de un elemento.

En **TDS** los únicos subconjuntos de Θ de interés son aquellos que tienen una asignación de probabilidad diferente de 0, cada uno de esos subconjuntos es llamado elemento focal de la función de creencia **Bel**. La unión de todos los elementos focales para una función de creencia es llamada *esencia*. En el ejemplo, la *esencia* es {Hep, Cirr, Cal} y

$$\text{Bel}(\{\text{Hep,Cirr}\}) = \text{m}(\{\text{Hep}\}) + \text{m}(\{\text{Cirr}\}) + \text{m}(\{\text{Hep,Cirr}\}) = 0.9.$$

En teoría de probabilidad, una distribución de probabilidad *a priori* uniforme, representa completa ignorancia en esos hechos. En ella no existe manera de distinguir entre ignorancia e instancias en donde la información conocida sugiere una distribución uniforme. Por el contrario **TDS** expresa la ignorancia explícitamente. Si todos los elementos focales son simples, no existe ignorancia con respecto a su ocurrencia; si algún elemento focal contiene más de un elemento, entonces existe algo de ignorancia. En **TDS** la creencia acerca de la negación de una hipótesis no depende de la creencia en la hipótesis misma, la única restricción es **Bel(A)+Bel(¬A)≤1.**

Para combinar los grados de creencia basados sobre distintos grupos de evidencia, se calcula la suma ortogonal por medio de la *regla de Dempster*, obteniendo un nuevo grado de creencia basada en la evidencia combinada:

$$[m_1 \oplus m_2](A) = \sum_{(A_i \cap B_j)=A} m_1(A_i) m_2(B_j) \qquad (1.14)$$

Podemos interpretar esta fórmula en la forma siguiente: si un grupo de evidencia ($\mathbf{E_1}$) asigna creencia $\mathbf{m_1}$ a un subconjunto $\mathbf{A_1}$ y otro grupo de evidencia ($\mathbf{E_2}$) asigna creencia $\mathbf{m_2}$ a otro subconjunto $\mathbf{B_1}$, el producto ($\mathbf{m_1}$*$\mathbf{m_2}$) da la creencia combinada en su intersección ($\mathbf{A_1} \cap \mathbf{B_1}$) y la creencia total en algún subconjunto **A** es la suma de las creencias asignadas en esta forma. Surge una dificultad si la intersección es conjunto vacío, entonces alguna creencia será asignada a él y la suma de las ***apb*** no será 1. Esto se soluciona descartando el conjunto vacío y extendiendo el resto, de tal forma que la suma de todas las ***apb*** sea 1. Así la ecuación (1.14) es reemplazada por:

$$[m_1 \oplus m_2] = \sum_{(A_i \cap B_j)=A} \frac{m_1(A_i)m_2(B_j)}{1-K} \quad A \neq \varnothing \tag{1.15}$$

donde:

$$K = \sum_{(A_i \cap B_j)=\varnothing} m_1(A_i)m_2(B_j) \tag{1.16}$$

Por ejemplo suponer que se tienen dos funciones de creencia m_1 y m_2 las cuales están definidas en un marco de discernimiento Θ representadas por [Gordon 85]:

$m_1(\{Hep\})=0.8$
$m_1(\{\Theta\})=0.2$
$m_2(\{Cirr\})=0.5$
$m_2(\{Hep,Cirr\})=0.2$
$m_2(\{\Theta\})=0.3$

La suma ortogonal de las dos funciones se puede observar en la tabla 2-1 al aplicar la ecuación 1.14.

M_1 \ M_2	{Cirr}(0.5)	{Hep,Cirr}(0.2)	Θ(0.3)
{Hep}(0.8)	Θ(0.4)	{Hep}(0.16)	{Hep}(0.24)
Θ(0.2)	{Cirr}(0.1)	{Hep,Cirr}(0.04)	Θ(0.06)

Tabla 1-2. Ilustración de la regla de combinación de Dempster.

Aplicando 1.16 para obtener K y eliminando el conjunto vacío con 1.15 tenemos que:

$$K = 0.4$$

$$m_1 \oplus m_2(Hep) = \frac{0.16+0.24}{1-0.4} = 0.6667$$

$$m_1 \oplus m_2(Cirr) = \frac{0.1}{1-0.4} = 0.1666$$

$$m_1 \oplus m_2(Hep,Cirr) = \frac{.04}{1-0.4} = 0.0667$$

$$m_1 \oplus m_2(\theta) = \frac{0.06}{1-0.4} = 0.1$$

Se puede observar en los resultados anteriores que la suma de las creencias combinadas da un valor de uno y es posible concluir que la combinación de las dos funciones apoya más a {Hep} dado su valor de 0.6667.

Una restricción importante de la **TDS** es el requerimiento de representar todas las posibles hipótesis y que estas deben de ser mutuamente exclusivas y exhaustivas. Esto puede ser

una dificultad en algunas aplicaciones ya que los requerimientos computacionales son altos, creciendo exponencialmente conforme el número de posibilidades se incrementa. Para ver ejemplos de la aplicación de esta teoría referirse a Gordon [85].

1.3.4. Lógica difusa

La *teoría de conjuntos difusos* (estos conjuntos también son conocidos como conjuntos *borrosos*) desarrollada por Zadeh [65], condujo al desarrollo de la teoría de *posibilidad* (también conocida como teoría de la evidencia) ó *lógica difusa* [Zadeh 78], la cual a sido propuesta como una alternativa para representar conocimiento con incertidumbre. Esta teoría surge de la necesidad de representar de alguna manera, información que es inexacta o vaga.

En cualquier sistema experto la base de conocimientos contiene conocimiento humano el cual es impreciso. Con frecuencia las relaciones entre hipótesis son vagamente definidas. Al representar conocimiento perteneciente a algún dominio, en ocasiones se utilizan términos tales como probable o muy probable, para describir ocurrencias de evidencia e hipótesis. Por ejemplo, "Si los síntomas son **X's**, entonces es muy probable que la enfermedad sea **Y**" [Abramson 90]. Cuando este tipo de experiencia es codificado dentro de probabilidades, la imprecisión se pierde y la idea es representada con un valor específico. Zadeh desarrolló su teoría de posibilidad para expresar estos términos vagos con precisión y exactitud. La *Teoría de posibilidad* reemplaza la lógica binaria de probabilidad por lógica multivaluada. En teoría de probabilidad un evento ocurre o no. En teoría de posibilidad se permiten diversos grados de ocurrencia. La *Teoría de posibilidad* está basada en teoría de conjuntos difusos. A continuación se da una breve explicación de está teoría. Para un estudio completo referirse a Zadeh [65].

Siendo **U** un conjunto de objetos, un conjunto difuso es una clase de objetos con un grado de membresía en **U**. Suponer que **A** es un subconjunto de **U**, caracterizado por una función de membresía $\mu_A(u)$, la cual asocia un número real entre 0 y 1 ([0,1]) a cada elemento $u \in U$. El valor $\mu_A(u)$ representa el grado de membresía de **u** en **A**, donde un valor de $\mu_A(u)$ cercano a 1.0 indica el grado mayor de pertenencia de **u** en **A**. La diferencia entre un conjunto difuso y un conjunto normal (un conjunto en el cual un objeto puede pertenecer o no al conjunto) es el rango de posibles valores para μ_A. Estas dos teorías también difieren en la asignación del valor fundamental, ya que altamente posible no implica altamente probable, y viceversa. Además no existe restricción en la suma de las posibilidades, mientras que la suma de las probabilidades debe ser igual a 1.

Para ilustrar las propiedades de conjuntos difusos, considerar el siguiente ejemplo.

Siendo **U** un conjunto de números enteros, $U = \{1,2,3,...\}$ y **A** el conjunto difuso de números pequeños [Zadeh 78].

Una caracterización subjetiva de **A** pueden ser :

u	1	2	3	4	5	6
$\mu_A(u)$	1.0	1.0	0.8	0.6	0.4	0.2

Un conjunto difuso es vacío si y sólo si su función de membresía es cero para todos los elementos en **U**. Dos conjuntos difusos **A** y **B** son iguales si y sólo si $\mu_A(u)=\mu_B(u)$ para todo **u** en **U**. El complemento de un conjunto difuso **A** es denotado por $\neg$**A** y su función de membresía es definida por $\mu_{\neg A}=1-\mu_A$. Un conjunto difuso **A** está contenido en un conjunto difuso **B** (**A** es subconjunto de **B**) si y sólo si $\mu_A(u) \leq \mu_B(u)$ para todo $u \in U$. La unión de dos conjuntos difusos **A** y **B** (con funciones de membresías μ_A y μ_B, respectivamente) es un conjunto difuso **C**, con $\mu_C(u)=\max[\mu_A(u),\mu_B(u)]$ para todo $u \in U$. Similarmente la intersección de dos conjuntos difusos es otro conjunto difuso **C**, $\mu_C(u)= \min[\mu_A(u),\mu_B(u)]$ para todo $u \in U$.

El paradigma sugerido para propagar la creencia, conocido como **teoría de posibilidad**, trabaja principalmente con distribuciones. Las distribuciones de posibilidad están relacionadas a las funciones de membresía difusa. Si **A** es un subconjunto difuso de **U**, caracterizado por su función de membresía μ_A:U$\rightarrow$[0,1]. La proposición "X es A" asocia una distribución de posibilidad Π_X tal que Π_X=**A**. De la misma manera, la función de distribución de posibilidad asociada con **X**, π_X es igual a la función de membresía de **A**, es decir $\pi_X=\mu_A$.

Regresando al ejemplo de números pequeños, **A** puede ser escrito como [Zadeh 78].

$$A= 1/1 + 1/2 + 0.8/3 + 0.6/4 + 0.4/5 + 0.2/6$$

donde + denota la unión de conjuntos difusos y el término 0.8/3 significa una posibilidad de 0.8 de que 3 es un número entero pequeño. Dados estos valores para **A**, la proposición "X es un número entero pequeño" asocia a **X** con la distribución de posibilidad Π_X=**A**. La *medida de posibilidad*, Poss{$x \in A$}, es la posibilidad que un valor x pertenece a **A** y es expresado como:

$$Poss\{x \in A\} = \max_{u \in A}[\pi_x(u)] \qquad (1.17)$$

Dado que las proposiciones pueden ser compuestas y cuantificadas con términos tales como "muy posible o poco posible", la teoría debe permitir la transición del español a un sistema más formal, incluyendo reglas que permitan componer y cuantificar sentencias. Para ello *teoría de posibilidad* cuenta con las siguientes reglas:

Reglas modificadoras. Estas reglas definen el impacto en las distribuciones de posibilidad provocadas por modificadores en español (por ejemplo Muy, Bastante, Poco etc,) en una proposición de la forma "X es A".

Reglas de Composición. Si **A** y **B** son proposiciones, entonces **A** x **B** denota una proposición que es una composición de **A** y **B**. Los modos de composición usualmente usados son: " Y, O e Implicación lógicos".

Reglas de Calificación de Verdad. Estas reglas definen las modificaciones a la distribución de posibilidad provocadas por modificadores de verdad, (Muy Verdadero y Más o menos Verdadero). Una proposición sería " Es τ que X es A". donde τ representa el cuantificador en la proposición.

En *teoría de posibilidad*, las inferencias son trazadas por medio de una versión generalizada del *modus ponen*. Así si **A**, **A***, **B** y **B*** son sentencias difusas, donde el operador

(*) puede ser un adjetivo modificador, el *modus ponen* generalizado presentará.

Premisa: X es A*.
Implicación: Si X es A entonces Y es B
Conclusión: Y es B*.

Por ejemplo,

Premisa: Este Mango está muy maduro.
Implicación: Si un Mango está maduro,
entonces el Mango está bueno.
Conclusión: Este mango esta muy bueno.

Aunque la teoría de conjuntos difusos ha sido aplicada en sistemas expertos, existen problemas computacionales y semánticos. Stalling [77] argumenta que el acercamiento Bayesiano ofrece mayor flexibilidad computacional y hace uso de más información que el acercamiento difuso. De la misma manera que la **TDS**, la interpretación de los valores finales no es muy clara a menos que se transformen a términos lingüísticos; similarmente no hay una justificación clara de los operadores y de las reglas de inferencia. Algunos investigadores consideran que la teoría de posibilidad debe ser usada para representar otro tipo de incertidumbre diferente de probabilidad [Sucar 92].

1.3.5. Factores de certeza

Factores de certeza (**FC**) fueron desarrollados para el manejo de incertidumbre en *MYCIN* [Shortliffe 75], un sistema experto basado en reglas para el diagnóstico y tratamiento de enfermedades infecciosas. Los diseñadores decidieron no seguir teoría de probabilidad ya que "los expertos se resisten a expresar sus procesos de razonamiento en términos probabilísticos coherentes" y el uso de la regla de Bayes "requiere grandes cantidades de datos válidos y numerosas aproximaciones y suposiciones" [Shortliffe 75].

En un sistema experto basado en *factores de certeza*, la base de conocimiento consiste de un conjunto de reglas de la forma: **Si <evidencia> Entonces (FC) <hipótesis>**, donde **FC** denota la creencia en una hipótesis, dada la evidencia observada.

Antes que alguna combinación o propagación de evidencia se realice, dos funciones intermedias deben ser calculadas. Estas funciones son: **MB(h,e)** y **MD(h,e)**. La primera mide el grado en el cual la creencia en una hipótesis **h** será incrementada si **e** fuera observada. La segunda el grado en el cual la incredulidad en **h** será incrementada por la observación de la misma evidencia **e**. Ellas son definidas como sigue:

$$MB[h,e]=\begin{cases} 1 & Si\ p(h)=1 \\ \dfrac{\max[P(h|e),P(h)] - P(h)}{\max[1,0] - P(h)} & otro\ caso \end{cases} \quad (1.18)$$

$$MD[h,e] = \begin{cases} 1 & Si\ P(h)=0 \\ \dfrac{\min[P(h|e),P(h)] - P(h)}{\min[1,0] - P(h)} & otro\ caso \end{cases} \quad (1.19)$$

Los valores de **MB[h,e]** y **MD[h,e]** se ubican en el rango entre 0 y 1. La evidencia es propagada al calcular los **FC** con **MB** y **MD**, donde,

$$FC = \frac{MB - MD}{1 - \min[MB,MD]} \quad (1.20)$$

El valor de **FC** puede estar en un rango de -1 a 1; **FC < 0** indica la confirmación de la negación de **h** y **FC > 0** indica la confirmación de **h**.

Cuando dos o más reglas afectan a una misma hipótesis, los **FC** individuales obtenidos de esas reglas, son combinados produciendo un **FC** combinado para la hipótesis dada.

$$FC_{combinado}(X,Y) = \begin{cases} X + Y\,(1 - X) & Si\ X\ y\ Y>0. \\ \dfrac{X + Y}{1 - \min[|X|,|Y|]} & Si\ X\ o\ Y<0. \\ -CF_{combinado}(-X,-Y) & Si\ X\ y\ Y<0. \end{cases} \quad (1.21)$$

Desde que **FC** fueron introducidos en *MYCIN*, han sido utilizados para representar incertidumbre en sistemas expertos basados en reglas. Adams [76] encontró que algunas suposiciones implícitas consideradas en **FC** algunas veces no son válidas, cómo la suposición de independencia entre hipótesis.

Las mejores características de **FC** son que representan, combinan y propagan los efectos de fuentes múltiples de evidencia, en términos de las creencias comunes o incredulidad en cada hipótesis. El modelo original fue diseñado sólo para representar los cambios en las creencias inducidas por alguna evidencia, más que un grado absoluto de creencia [Horvitz 86]. **FC** no requiere que se le asignen probabilidades *a priori*. Heckerman y Horvitz [87] señalan que **FC** no pueden representar algunas clases de dependencias entre creencias inciertas de una manera eficiente y natural.

1.3.6. Método bayesiano subjetivo (PROSPECTOR)

Prospector [Duda 76] es un sistema experto utilizado en geología para evaluar la riqueza mineral en un sitio de exploración. Se basa en teoría de decisión Bayesiana y en el teorema de Bayes. En este método se usan razones de probabilidad llamadas *odds* probabilísticos. Se utiliza una red de inferencia para representar las relaciones existentes en el conjunto de reglas las cuales determinan el conocimiento. Una red de inferencia representa una colección de reglas

como una estructura. En ella cada nodo representa una proposición y los arcos una regla de inferencia.

Las reglas toman la forma **Si <evidencia> Entonces <hipótesis> (LS,LN)**, donde **LS** mide el soporte a favor de una hipótesis y **LN** mide la oposición a ella [Abramson 90]. El experto proporciona los valores de **LS** y **LN** para las reglas y los *odds a priori* para las hipótesis y evidencias. En operación el usuario indica que evidencia fue observada, entonces los *odds* son actualizados y propagados a través de la red de inferencia hasta obtener los *odds a posteriori* de la hipótesis final.

La ventaja de utilizar razones de probabilidad en lugar de probabilidades condicionales y *a priori*, se debe a que el experto no tiene que proporcionar la probabilidad exacta y es más fácil asignar una razón de probabilidad. Dado que las probabilidades condicionales y *a priori* pueden ser recuperadas de las dos razones de probabilidad (**LS** y **LN**), este método puede usar el teorema de Bayes para propagar la evidencia.

Los dos números asociados con cada regla son llamados razones de probabilidad y representan la fuerza o peso de cada regla. La razón de probabilidad de observar **E** dado que **H** es verdadera, es definida como [Abramson 90]:

$$LS(H,E) = \frac{P(E|H)}{P(E|\neg H)} \tag{1.22}$$

En general los *odds a priori* de una hipótesis **H** están determinados por:

$$Odds(H) = O(H) = \frac{P(H)}{P(\neg H)} = \frac{P(H)}{1-P(H)} \tag{1.23}$$

Mientras que los *odds a posteriori* de **H** dado **E** son calculados por:

$$O(H|E) = \frac{P(H|E)}{P(\neg H|E)} \tag{1.24}$$

Combinando las definiciones anteriores, la relación entre *odds* y razones de probabilidad establece que:

$$O(H|E) = LS(H,E) \; x \; O(H) \tag{1.25}$$

La ecuación anterior permite actualizar los *odds* de **H** dado que **E** fue observada. Valores altos de **LS** ($LS >> 1$) indican que la observación de **E** incrementa enormemente la probabilidad *a posteriori* de **H**. En otras palabras la regla soporta la hipótesis.

Similarmente la razón de probabilidad de observar $\neg$**E** dado que **H** es verdadera es definida como [Abramson 90]:

$$LN(H,E) = \frac{P(\neg E|H)}{P(\neg E|\neg H)} \tag{1.26}$$

En forma análoga:

$$O(H|E) = LN(H,E) \times O(H) \quad (1.27)$$

Valores bajos de LN ($0 \leq LN << 1$) decrementan la probabilidad *a posteriori* de **H** dado $\neg$**E**.

Las evidencias son combinadas utilizando las reglas siguientes:

1.- **Conjunción.** La probabilidad de observar **n** piezas de evidencia es igual a la probabilidad de observar la menos probable de ellas:

$$P(E_1 \cap E_2 \cap ... \cap E_n) = \min[P(E_1), P(E_2), ..., P(E_n)] \quad (1.28)$$

2.- **Disjunción.** La probabilidad de observar al menos una evidencia, de un conjunto de **n** piezas de evidencia es igual a la probabilidad de observar la más probable de ellas:

$$P(E_1 \cup E_2 \cup ... \cup E_n) = \max[P(E_1), P(E_2), ..., P(E_n)] \quad (1.29)$$

La creencia es propagada por el cálculo de los *odds a posteriori* de una hipótesis (**H**), dada la observación de alguna evidencia (**E**). Utilizando los *odds a posteriori* es posible recobrar la probabilidad *a posteriori* **P(H|E)** como sigue:

$$P(H|E) = \frac{O(H|E)}{1 + O(H|E)} \quad (1.30)$$

A continuación se presenta un ejemplo para ilustrar el esquema de propagación[1].

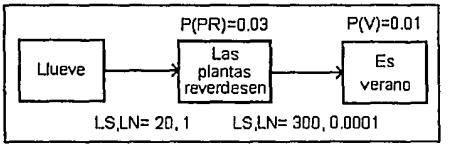


Figura 1-1. Ejemplo de propagación en Prospector.

En la figura 1-1, (20,1) representa los valores de **LS** y **LN** en la regla **Si llueve Entonces las plantas reverdesen (LS,LN)**. Suponer que el usuario confirma que llueve. Esta evidencia incrementa los *odds* de **PR** por un factor de 20 como sigue.

Los *odds a priori* de **PR** son: 0.03/(1-0.03)= 0.0309.

Los *odds a posteriori* son: 0.0309x20= 0.618.

Pasando los *odds* de **PR** a una probabilidad tenemos:

0.618/(1+0.618)= 0.382.

[1] Ejemplo basado en Duda [79]

Lo anterior incrementa los *odds* de **V** por un factor de 300, pero influenciado por el incremento de la probabilidad de **PR** se obtiene un factor de:

$$300x(0.382-0.03)/(1-0.03)= 108.866$$

De manera semejante, calculando los *odds a priori* de **V** se obtiene: 0.0101 y al calcular los *odds a posteriori* de **V**, sólo que ahora utilizando el factor obtenido anteriormente, resulta: 1.10. Pasando este valor a probabilidad, se obtiene un valor en **V** de 0.5238. La propagación continuará de esta manera a través de la red, hasta alcanzar la hipótesis final.

Este método sufre de problemas y limitaciones parecidos a *factores de certeza* en *MYCIN*, debido a la suposición de independencia. Duda [76] señala, que esta suposición no es valida si hay múltiples trayectorias de un nodo a otro. La estructura de la red esta restringida, ya que sólo produce probabilidades coherentes si forma un árbol [Heckerman 86].

1.4. CONCLUSIONES

En este capítulo hemos analizado algunas técnicas para el manejo de incertidumbre en sistemas expertos. Podemos decir que aunque estas técnicas operan adecuadamente en el dominio para el que fueron creadas, surgen problemas al tratar de extenderlas a otros dominios de aplicación.

Al analizar las técnicas descritas en este capítulo observamos diversas desventajas en las mismas, en algunas se suponen independencias condicionales y mutua exclusividad de los datos. Otras llevan a cabo funciones de ajuste *Ad Hoc* (funciones para aquello de que se trate) para lograr consistencia en sus decisiones, aunque tales prácticas son con frecuencia necesarias, deben de ser evitadas cuando sea posible. Además en unas existe poca claridad en los resultados obtenidos, reflejándose en una mala interpretación de los mismos, así como una gran cantidad de requerimientos computacionales para el funcionamiento adecuado en otras técnicas. Debido a esto se buscaron nuevas técnicas, desarrollando así las redes probabilísticas, con las cuales se obtiene mayor consistencia en el manejo de incertidumbre de acuerdo a teoría de probabilidad.

CAPITULO 2

REDES PROBABILISTICAS

2.1. INTRODUCCION

En el capítulo anterior mencionamos algunos métodos utilizados, para el manejo de incertidumbre en Sistemas Expertos. Aunque tales métodos fueron utilizados exitosamente en el área para la que fueron diseñados, poseen algunas deficiencias debido a la forma en que manejan el conocimiento con incertidumbre; principalmente al ser extendidos a dominios diferentes para los que son creados. Sin embargo, no dejan de ser importantes en el desarrolló de una disciplina que permita realizar razonamiento probabilístico, con conocimiento que por lo general contiene incertidumbre.

Entre los aspectos que les han sido criticados se encuentran: las suposiciones de independencia condicional, mutua exclusividad y exhaustividad en los hechos que conforman el conocimiento; la gran cantidad de información que se requiere almacenar en ciertos métodos para llevar a cabo algún esquema de propagación. Otro aspecto se debe a la utilización de funciones de ajuste *Ad-Hoc*, para obtener consistencia en el conocimiento (cuando se lleva a cabo dicho ajuste, con frecuencia se suponen justificaciones inductivas y su validez depende del buen desempeño del sistema).

En la búsqueda de un formalismo que trabaje con incertidumbre, el cual evite algunas de las inconsistencias antes mencionadas, Pearl [82, 86 y 88] desarrolló un método basado en un modelo de razonamiento humano. A las probabilidades utilizadas aquí las llamo creencias y al método propagación de creencia. Pearl [86] argumenta que "la estructura fundamental del conocimiento humano puede ser representada por gráficas de dependencias y que el trazo mental de ligas en estos gráficos son los pasos básicos en las interrogantes y actualización del conocimiento". Pearl ha demostrado que al representar información en una base de conocimiento mediante redes probabilísticas, se construye una base de conocimientos probabilística consistente, sin la necesidad de imponer suposiciones de independencia condicional.

En este capítulo discutiremos el método de representación basado en redes probabilísticas, ya que en este se fundamenta el objetivo de la presente tesis. Primero veremos las técnicas de razonamiento (propagación) para ciertas estructuras de redes. Posteriormente, hablaremos de un método para realizar un tipo de aprendizaje conocido como "aprendizaje paramétrico" dentro de este marco de trabajo, así como la forma de evaluar los resultados derivados del sistema, con respecto a un patrón definido por el experto.

Antes de hablar acerca de redes probabilísticas, es necesario definir algunos conceptos que son de importancia para el estudio de las mismas.

2.2. VARIABLES PROPOSICIONALES Y PROBABILIDAD CONJUNTA

Se dice que **A** es una variable proposicional finita en Ω, si contiene un número finito de eventos, los cuales son mutuamente exclusivos y exhaustivos. La variable **A** se representa como:

$$\mathbf{A}=\{a_1, a_2,\ldots, a_{n-1}\}.$$

Las variables proposicionales son denotadas por letras mayúsculas, a diferencia de sus miembros, los cuales se denotan con minúsculas.

Para representar la intersección de eventos de varias variables proposicionales se utiliza la ",", en lugar del símbolo lógico $\cap$ o el símbolo de intersección $\wedge$. Así si A, B, y C son variables proposicionales, el evento (a_i,b_j,c_k) denota $a_i \wedge b_j \wedge c_k$. De manera semejante cuando nos referimos a variables en vez de valores específicos, el evento (A,B,C) denota $(A \wedge B \wedge C)$.

Considerando un conjunto de n variables proposicionales finitas $\{X_1,X_2,..,X_n\}$, definidas en el mismo espacio de probabilidad Ω, la probabilidad de que todas ellas ocurran, es decir $P(X_1 \wedge X_2 \wedge ,.., \wedge X_n)$ es conocida como **"probabilidad conjunta"** y se representa como:

$$P(X_1 \wedge X_2 \wedge,...,\wedge X_n) = P(X_1,X_2,...,X_n) \qquad (2.1)$$

Cuando se tienen las probabilidades conjuntas de un conjunto de variables proposicionales finitas $\{X_1,X_2,..,X_n\}$, corriendo a través de todas sus posibles combinaciones de valores, a la totalidad de probabilidades conjuntas se le llama **distribución de probabilidad conjunta.**

2.3. REDES PROBABILISTICAS

Las Redes probabilísticas, conocidas también como *redes Bayesianas, redes causales o redes de creencia,* fueron introducidas por Pearl [82, 86 y 88], como una representación gráfica de dependencias probabilísticas para razonamiento probabilístico en sistemas expertos. Una red probabilística **(RP)** es un gráfico acíclico dirigido **(GAD)**, en el cual cada nodo representa una variable proposicional y cada arco la relación que existe entre ellos. La variable al final de un arco es dependiente de la variable que se encuentra en el origen del mismo. Por ejemplo **C** es dependiente de **A** en la red probabilística de la figura 2-1. El gráfico en la figura 2-1 representa la distribución de probabilidad conjunta de las variables **A, B, C, D, E y F** como:

$$P(A,B,C,D,E,F)=P(E|C)P(F|C,D)P(C|A)P(D|A,B)P(A)P(B) \qquad (2.2)$$

La ecuación (2.2) es obtenida al aplicar la regla de la cadena y de la información de dependencias representadas en la red.

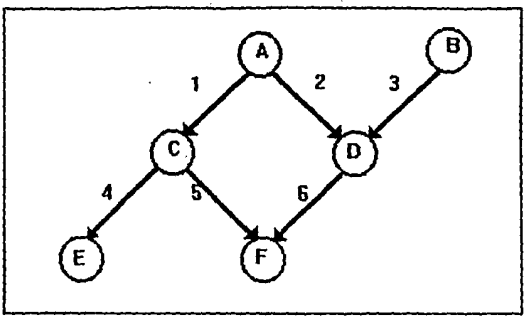


Figura 2-1. Red Probabilística.

Una red probabilística, se asemeja a una base de reglas donde cada nodo representa una variable, evidencia o hipótesis y las reglas son substituidas por las dependencias representadas por las ligas. Un grupo de flechas apuntando a un nodo, representan un conjunto de reglas relacionando las variables al comienzo de la liga con la variable al final, cuantificada por su respectiva probabilidad condicional. Por ejemplo en la figura 2-1, la liga 5 y 6 representan la regla :

$$Si\ C_i,\ D_j\ entonces\ F_k.$$

cuantificada en términos probabilísticos como la matriz de probabilidad condicional **P** donde: $P_{ijk}=P(F_k|C_i,D_j)$.

A diferencia de muchos sistemas expertos, en los cuales la red de inferencia va de la evidencia a la hipótesis (*Prospector* es uno de ellos), en redes probabilísticas van de la condición a la consecuencia o de causas a efectos, de donde se deriva el nombre de *redes causales*.

La topología de una red probabilística proporciona información directa acerca de las dependencias existentes entre las variables que conforman la red. En particular, representa cuales variables son condicionalmente independientes dadas otras variables. Por definición, **A** es condicionalmente independiente de **B**, dado **C**, si:

$$P(A|B,C) = P(A|C) \qquad (2.3)$$

Esto se representa gráficamente por el nodo **C** separando al nodo **A** de **B** en la red. En general **C** será un subconjunto de nodos de la red de tal forma que si lo removemos, se formarán dos subconjuntos de nodos **A** y **B** desconectados. Es decir, no habrá una trayectoria entre algún nodo en **A** y algún nodo en **B**. En el gráfico de la figura 2-1, **E** será condicionalmente independiente de **A, B, D y F** dado **C**.

Al conocer la distribución de probabilidad condicional de cada variable proposicional dados sus padres, es posible calcular la distribución de probabilidad conjunta de todas las variables proposicionales en la red. Así el poder de redes probabilísticas se fundamenta en la

posibilidad de obtener la distribución de probabilidad conjunta, a partir de las probabilidades condicionales por medio de la siguiente ecuación.

$$P(v_1,v_2,...,v_n) = P(v_n|c(v_n))P(v_{n-1}|c(v_{n-1}))...P(v_2|c(v_2))P(v_1) \quad (2.4)$$

donde $c(v_n)$ representa los padres de v_n. La ecuación (2.4) es obtenida de una manera similar a la ecuación (2.2), la diferencia se debe a que (2.4) es expresada en términos generales.

La importancia de la distribución de probabilidad conjunta se debe a que ella determina la estructura del **GAD.** De acuerdo a lo anterior una red probabilística está determinada por un **GAD** y por la distribución de probabilidades condicionales de cada variable dados sus padres. Se tienen las siguientes relaciones entre la distribución de probabilidad conjunta, la distribución de probabilidad condicional, el **GAD** y redes probabilísticas:

1.- Por definición una red probabilística contiene un **GAD** y una distribución de probabilidad conjunta únicos. De aquí una red probabilística también determina una distribución de probabilidad condicional única de las variables dados sus padres (recordar que la ecuación 2.4. establece que la distribución de probabilidad conjunta puede ser obtenida de la distribución condicional).

2.- Todo **GAD**, cuyos vértices son variables proposicionales, junto con la distribución condicional de las variables dados sus padres, determinan una red probabilística única.

3.- Toda distribución conjunta de un conjunto de variables proposicionales, tiene al menos una red probabilística en la cual está representada.

Una base de conocimientos representada como una red probabilística, es usada para razonar acerca de las consecuencias de datos de entrada específicos, mediante lo que es llamado **razonamiento probabilístico.** Este consiste en asignar un valor a ciertas variables y propagar sus efectos a través de la red, actualizando la probabilidad de las otras variables. La propagación es consistente con teoría de probabilidad, basada en la aplicación de cálculo Bayesiano y en las dependencias representadas en la red. La propagación de probabilidades en una red es un problema complejo pero existen diversos algoritmos para llevarla a cabo [Sucar 92]. Estos algoritmos están restringidos a cierto tipo de estructuras.

Pearl [82] desarrolló un método para propagación de probabilidades en redes las cuales forman un árbol, donde cada nodo tiene únicamente una liga que llega a el o un padre. Este método fue extendido posteriormente a poliárboles [Kim 83], donde un nodo puede tener varios padres pero sólo existe una posible trayectoria entre un par de nodos, es decir una red conectada simple. Los métodos mencionados anteriormente no funcionan en redes multi-conectadas, con trayectorias múltiples entre nodos. Para este tipo de red no existe un algoritmo de propagación eficiente, pero Pearl [88] ha sugerido un esquema de razonamiento para redes en las cuales el número de trayectorias no es muy grande. Recientemente Spiegelhalter [86a] y Lauritzen [88] desarrollaron una técnica basada en teoría de gráficas la cual funciona eficientemente cuando la red no es muy densa.

2.4. PROPAGACION DE PROBABILIDAD EN ARBOLES

A continuación se presenta la teoría utilizada en el método de Pearl para la propagación de probabilidad en árboles.

En un árbol probabilístico cada nodo tiene un sólo padre, excepto un nodo llamado raíz el cual únicamente tiene hijos. La figura 2-2.(a) muestra un árbol probabilístico.

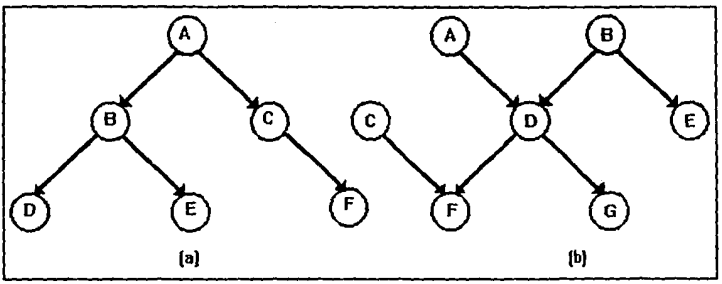


Figura 2-2. a) Arbol probabilístico, b) Poliárbol.

Cada nodo representa una variable multivaluada discreta y cada arco dirigido es cuantificado por una matriz de probabilidad condicional, $P(B|A)$, donde $P(B|A)_{ij} = P(B_i|A_j)$.

Consideremos una red probabilística la cual es un **GAD. V** es un conjunto de variables proposicionales y **W** un subconjunto de éstas variables las cuales tienen un valor asignado (los valores asumidos por esas variables son conocidos. Para referirnos a estas variables diremos que son asignadas). Para $\mathbf{B} \in \mathbf{V}$, denotamos.

$\mathbf{W}^-_{\mathbf{B}}$ representa el conjunto **W** cuyo padre es **B**.

$\mathbf{W}^+_{\mathbf{B}}$ representa $\mathbf{W}-\mathbf{W}^-_{\mathbf{B}}$.

Por ejemplo, suponiendo que en el gráfico de la figura 2-2(a) las variables asignadas son $W=\{C,D,E\}$. Entonces $W^-_B=\{D,E\}$ y $W^+_B=\{C\}$.

Dado un subconjunto de variables **W** asignadas, la probabilidad *a posteriori* de alguna variable **B** por el teorema de Bayes será.

$$P(b_i|W) = \frac{P(W|b_i)P(b_i)}{P(W)} \qquad (2.5)$$

donde $\mathbf{B} \notin \mathbf{W}$.

Dadas las dependencias representadas en el árbol, **B** divide el mismo en dos subárboles independientes, uno formado por todos los descendientes de **B** y otro por los nodos restantes. Así **W** será descompuesta en dos: $\mathbf{W_B}^-$ para los datos cuyo padre es **B** y $\mathbf{W_B}^+$ para los datos contenidos en el resto de la red. Por lo tanto (2.5) puede escribirse como.

$$P(b_i|W) = \frac{P(W_B^-, W_B^+|b_i)P(b_i)}{P(W)} \tag{2.6}$$

Dado que los dos subárboles son condicionalmente independientes dado **B** tenemos que

$$P(W_B^-, W_B^+|b_i) = P(W_B^-|b_i)P(W_B^+|b_i) \tag{2.7}$$

Sustituyendo (2.7) en (2.6):

$$P(b_i|W) = \frac{P(W_B^-|b_i)P(W_B^+|b_i)P(b_i)}{P(W)} \tag{2.8}$$

Considerando que:

$$P(W_B^+|b_i) = \frac{P(b_i|W_B^+)P(W_B^+)}{P(b_i)} \tag{2.9}$$

Y

$$P(W) = P(W_B^+ \cup W_B^-) \tag{2.10}$$

la ecuación (2.8) queda como:

$$P(b_i|W) = \frac{P(W_B^-|b_i)P(b_i|W_B^+)P(W_B^+)}{P(W_B^- \cup W_B^+)} \tag{2.11}$$

y la ecuación (2.11) se convierte en:

$$P(b_i|W) = \alpha P(W_B^-|b_i)P(b_i|W_B^+) \tag{2.12}$$

donde α es una constante de normalización. La ecuación (2.12) permite actualizar las probabilidades de cada nodo en la red, al combinar las evidencia que viene de sus descendientes con la evidencia de sus padres. Esta ecuación muestra que no se requieren probabilidades *a priori*, excepto para el nodo raíz **A** para el cual:

$$P(a_i|W_A^+) = P(a_i) \tag{2.13}$$

Si definimos los siguientes términos [Pearl 86]:

$$\lambda(b_i) = P(W_B^-|b_i) \tag{2.14}$$

$$\pi(b_i) = P(b_i|W_B^+) \tag{2.15}$$

La ecuación (2.12) puede ser escrita como:

$$P(b_i|W) = \alpha\lambda(b_i)\pi(b_i) \tag{2.16}$$

El vector de valores $\lambda(b_i)$ es llamado "valor lambda de **B** y es denotado por $\lambda(B)$. De manera semejante el vector de valores $\pi(b_i)$ es llamado valor π de **B** y se denota como $\pi(B)$.

Dado que los hijos son condicionalmente independientes dados sus padres, $\lambda(b_i)$ puede ser calculada de sus descendientes como:

$$\lambda(b_i) = \prod_k P(W_B^{K-}|b_i) = \prod_k \lambda_k(b_i) \tag{2.17}$$

donde **k** denota el subconjunto de todos los hijos de **B** con **m** posibles valores y **Π** representa el producto de todos los mensajes λ de **b**. Por lo anterior W_B^{k-} representa la evidencia de el subárbol unido a el **k-esimo** hijo de **B**. Condicionando cada término en (2.17) con respecto a los **m** posibles valores de el **k-esimo** hijo de **B**, denotado por **S**, obtenemos:

$$\lambda(b_i) = \prod_k \left[\sum_j^m P(W_B^{k-}|b_i,S_j^k)P(S_j^k|b_i)\right] \tag{2.18}$$

donde $W_B^{k-}=W_S^-$. Dado que **B** es condicionalmente independiente de las evidencias que son descendientes de **S**, dado **S**, obtenemos:

$$P(W_S^-|b_i,S_j^K) = P(W_S^-|S_j^k) = \lambda(S_j^k) \tag{2.19}$$

Sustituyendo (2.19) en (2.18) tenemos:

$$\lambda(b_i) = \prod_k \left[\sum_j^m P(S_j^k|b_i)\lambda(S_j^k)\right] \tag{2.20}$$

donde $B \notin W$ y:

Si $B \in W$ y b_i es el valor asignado $\lambda(b_i)=1$.
Si $B \in W$ y b_i no es el valor asignado $\lambda(b_i)=0$.

La sumatoria representa $\lambda_{S^k}(b_i)$ la cual será definida posteriormente.

La ecuación (2.20) determina como calcular "el soporte de diagnóstico" de un nodo, de sus descendientes. De manera similar a $\lambda(b_i)$, es posible derivar una expresión para $\pi(b_i)$. Primero $\pi(b_i)$ se condiciona sobre los posibles valores de su padre **A**.

$$\pi(b_i) = \sum_j^m P(b_i|W_B^+,a_j)P(a_j|W_B^+) \qquad (2.21)$$

En la ecuación (2.21) se puede eliminar W_B^+ de el primer término, debido a la independencia condicional. El segundo término, representa la probabilidad *a posteriori* de **A** dadas todas las evidencias, excepto los datos correspondientes a las evidencias cuyo padre es **B**. Utilizando (2.16) y (2.17) en el segundo término de (2.21) se obtiene:

$$\pi(b_i) = \sum_j^m P(b_i|a_j)\left[\alpha\pi(a_j)\prod_C \lambda_C(a_j)\right] \qquad (2.22)$$

Donde **C** constituye todos los hijos de **A** excepto **B**, es decir los hermanos de **B**. El término entre corchetes representa $\pi_B(a_j)$ el cual será definido posteriormente.

Las ecuaciones (2.16), (2.20) y (2.22) constituyen las bases para el mecanismo de propagación en un árbol probabilístico. Para ello sólo se requiere almacenar los vectores λ y π en cada nodo, actualizándolos con los parámetros correspondientes de sus vecinos, fijando simultáneamente la matriz de probabilidad condicional **P** para cada nodo. Esto puede ser realizado por un mecanismo en base a mensajes [Pearl 86a] con el cual cada nodo se comunica con sus vecinos (padres e hijos). Inicialmente la red esta en equilibrio. Cuando algunos nodos les asignamos un valor (son conocidos), la información es propagada a través de la red, cada nodo envía mensajes a sus padres e hijos, en dos direcciones:

a) Propagación de abajo hacía arriba. Cada nodo **B** envía un mensaje a su padre **A**. A este se le conoce como el mensaje de **B** a **A** y se define como:

$$\lambda_B(a_j) = \sum_i^m P(b_i|a_j)\lambda(b_i) \qquad (2.23)$$

La razón de que a $\lambda_B(A)$ se le llame mensaje se debe a que su valor es obtenido de información acerca de **B** y es significativo para **A**.

b) **Propagación de arriba a abajo.** Cada nodo **A** envía un mensaje a cada uno de sus hijos **B**. Para algún hijo **K** el mensaje es definido como sigue:

$$\pi_B(a_j) = \alpha\pi(a_j)\prod_{C\neq B} \lambda_C(a_j) \qquad (2.24)$$

donde $A \notin W$ y:

Si $A \in W$ y es asignada por a_j $\pi_B(a_j)=1$.

Si $A \in W$ y no es asignada por a_j $\pi_B(a_j)=0$.

A este mensaje se le conoce como el mensaje de **A** a **B**, ya que su valor es obtenido de información acerca de **A** (el valor π y su mensaje λ de sus otros hijos) y es significativo para **B**.

Una ecuación adicional para calcular el mensaje $\pi_B(a_j)$, la cual evita calcular todos los mensajes lambda de los hijos de **A** en (2.24), es la siguiente:

$$\pi_B(a_j) = \alpha \frac{P(a_j|W)}{\lambda_B(a_j)} \tag{2.25}$$

donde **B** no ha sido asignada y no es la raíz y **A** es el padre de **B**.

Demostración:

De la ecuación (2.16) tenemos que:

$$P(a_j|W) = \alpha \lambda(a_j)\pi(a_j) \tag{2.26}$$

De acuerdo con (2.17):

$$\lambda(a_j) = \prod_k \lambda_{S^k}(a_j) \tag{2.27}$$

donde **S** denota el **k** hijo de **A**. Por lo que (2.26) queda como:

$$p(a_j|W) = \alpha \pi(a_j)\left[\prod_k \lambda_{S^k}(a_j)\right] \tag{2.28}$$

Sustituyendo la ecuación (2.28) en (2.25), esta queda comprobada.

Cada nodo usa esta información para actualizar sus parámetros locales y actualizar su probabilidad *a posteriori* si lo requiere. Por ejemplo en la red de la figura 2-2a, si **D** y **E** fueran asignadas, ellas enviarían un mensaje λ a **B**, el cual enviaría un mensaje λ a **A** y mensajes π a todos sus hijos. Cuando los mensajes llegan al nodo raíz comienza la propagación de arriba a abajo hasta llegar a los nodos hoja donde la propagación termina.

Cabe mencionar que cuando una variable es la raíz su valor π es:

$$\pi(a_j) = P(a_j) \tag{2.29}$$

lo cual se deriva de (2.13) y (2.15). Similarmente el valor λ de una variable que es una hoja de la ecuación (2.14) es:

$$\lambda(b_i) = 1 \tag{2.30}$$

debido a que $\mathbf{W_B^-} = \varnothing$.

Para un estudio detallado de la teoría mencionada referirse a Pearl [86] y Neapolitan [90].

2.5. ILUSTRACION DEL METODO DE PROPAGACION DE PROBABILIDADES EN ARBOLES

La teoría mencionada en la sección anterior, permite determinar la probabilidad de un conjunto de variables en la red dado que algunas han sido asignadas, a través de la comunicación entre vértices vecinos por medio de mensajes. Adicionalmente el método tiene la ventaja de que no importa el orden en el cual las variables son asignadas.

Inicialmente la red esta en un estado inicial $\mathbf{E_1}$. Cuando una variable **X** es asignada, el método puede ser aplicado a la red (en estado $\mathbf{E_1}$) para determinar la probabilidad condicional de cada una de las variables dada la variable asignada. Esto deja la red en un estado $\mathbf{E_2}$. Cuando una segunda variable **Y** es asignada, el método puede ser aplicado a la red en estado $\mathbf{E_2}$ determinando así las probabilidades condicionales de cada una de las variables en la red dada la asignación de las dos variables. Este proceso continua para todas las variables que sean asignadas. El resultado obtenido en las probabilidades *a posteriori* no depende de el orden en el cual las variables son asignadas.

Para ilustrar el método de propagación veremos un ejemplo sencillo tomado de Neapolitan [90]. Este considera las variables proposicionales **A, B, C y D** con los siguientes valores posibles.

$\mathbf{a_1}$ = El esposo engaña a su mujer.
$\mathbf{a_2}$ = El esposo no engaña a su mujer.
$\mathbf{b_1}$ = El esposo cena con otra.
$\mathbf{b_2}$ = El esposo no cena con otra.
$\mathbf{c_1}$ = El esposo es visto cenando con otra.
$\mathbf{c_2}$ = El esposo no es visto cenando con otra
$\mathbf{d_1}$ = Una mujer extraña llama por teléfono.
$\mathbf{d_2}$ = Una mujer extraña no llama por teléfono.

La figura 2-3 representa la relación entre las variables proposicionales y sus respectivas probabilidades condicionales, por lo que la figura forma una red probabilística.

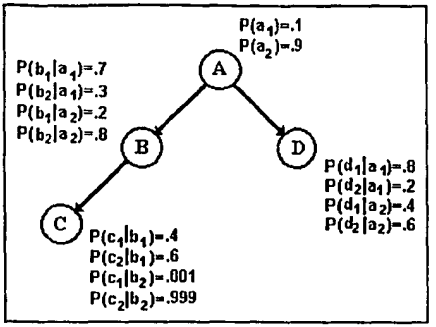


Figura 2-3. Estado de la red probabilística antes de la asignación de alguna variable.

Para comprender mejor el método de propagación, en primer lugar veremos como calcular la probabilidad *a priori* de cada variable cuando ninguna ha sido asignada.

Recordar que cada variable pasa un mensaje λ a sus padres y un mensaje π a cada uno de sus hijos. El gráfico es un gráfico pesado con estos valores residentes en los arcos. Además recordar que cada variable tiene un valor λ y un valor π asociado a ella. La red consiste de las probabilidades condicionales en cada variable, los valores λ y π de las mismas y los mensajes λ y π en cada arco. La red probabilística completa se muestra en la figura 2-4.

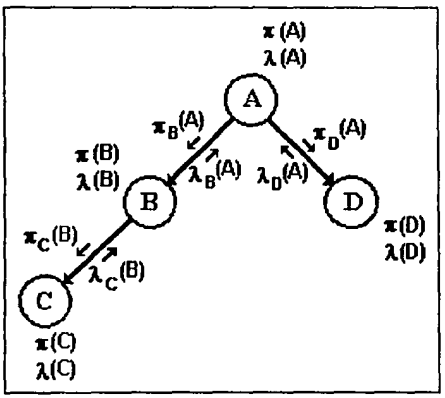


Figura 2-4. Red probabilística con los valores λ y π y los mensaje λ y π señalados.

El método utiliza las siguientes ecuaciones. La ecuación (2.20) permite calcular el valor λ de una variable con los mensajes que ella recibe de sus hijos. La ecuación (2.22) establece el valor π de una variable con los mensajes π que ella recibe de su padre. La ecuación (2.16)

es utilizada para calcular la probabilidad condicional de una variable a partir de sus valores λ y π.

A continuación veremos como calcular las probabilidades *a priori* de cada variable y la propagación por medio de los mensajes λ y π. Denotamos el conjunto de variables asignadas como **W** y consideramos que para el cálculo de las probabilidades *a priori* **W** es el conjunto vacío. Para comenzar calcularemos P(B|W)

De (2.30) tenemos que:

$$\lambda(c_1)=1 \quad y \quad \lambda(c_2)=1$$

$$\lambda(d_1)=1 \quad y \quad \lambda(d_2)=1$$

Conforme a la ecuación (2.23) calculamos los mensajes λ de **C** y **D**:

$$\lambda_C(b_1)=P(c_1|b_1)\lambda(c_1)+P(c_2|b_1)\lambda(c_2)=(.4)1+(.6)1=1$$

$$\lambda_C(b_2)=P(c_1|b_2)\lambda(c_1)+P(c_2|b_2)\lambda(c_2)=(.001)1+(.999)1=1$$

$$\lambda_D(a_1)=P(d_1|a_1)\lambda(d_1)+P(d_2|a_1)\lambda(d_2)=(.8)1+(.2)1=1$$

$$\lambda_D(a_2)=P(d_1|a_2)\lambda(d_1)+P(d_2|a_2)\lambda(d_2)=(.4)1+(.6)1=1$$

Cuando **B** recibe un mensaje λ de su hijo **C**, cambiará el valor λ de **B**, el cual esta determinado por la ecuación (2.20) recordando que la sumatoria en esta ecuación representa el mensaje que un hijo envía a su padre.

$$\lambda(b_1)=\lambda_C(b_1)=1$$
$$\lambda(b_2)=\lambda_C(b_2)=1$$

El valor π de la raíz **A** lo podemos obtener de (2.29), por lo que:

$$\pi(a_1)=P(a_1)=.1$$
$$\pi(a_2)=P(a_2)=.9$$

Con la ecuación (2.24) obtenemos el mensaje π que **B** recibe de **A**.

$$\pi_B(a_1)=\pi(a_1)\lambda_D(a_1)=(.1)1=.1$$
$$\pi_B(a_2)=\pi(a_2)\lambda_D(a_2)=(.9)1=.9$$

El valor π de **B** lo calcularemos por medio de la ecuación (2.22).

$$\pi(b_1)=P(b_1|a_1)\pi_B(a_1)+P(b_1|a_2)\pi_B(a_2)=(.7).1+(.2).9=.25$$
$$\pi(b_2)=P(b_2|a_1)\pi_B(a_1)+P(b_2|a_2)\pi_B(a_2)=(.3).1+(.8).9=.75$$

finalmente la P(B|W) la podemos obtener de (2.16) recordando que $\mathbf{W}=\varnothing$.

$$P(b_1|W)=P(b_1)=\alpha\lambda(b_1)\pi(b_1)=\alpha(1)(.25)=\alpha(.25)$$
$$P(b_2|W)=P(b_2)=\alpha\lambda(b_2)\pi(b_2)=\alpha(1)(.75)=\alpha(.75)$$

Dado que $P(b_1)+P(b_2)=1$, podemos deshacernos de la constante de normalización α (la normalización se refiere al hecho de que las probabilidades deben sumar uno, esto se consigue dividiendo cada una de las probabilidades entre la sumatoria de todas las probabilidades) al obtener:

$$P(b_1)=\frac{\alpha(.25)}{\alpha(.25)+\alpha(.75)}=.25$$
$$P(b_2)=\frac{\alpha(.75)}{\alpha(.25)+\alpha(.75)}=.75$$

observando que la constante α se elimina al calcular P(B|W).

A continuación calcularemos P(A) recordando que $\lambda_D(A)=(1,1)$, donde estamos utilizando una notación vectorial para establecer que $\lambda_D=(a_1)=1$ y $\lambda_D=(a_2)=1$. Determinando el mensaje λ que **B** envía a **A**, de la ecuación (2.23) tenemos:

$$\lambda_B(a_1)=P(b_1|a_1)\lambda(b_1)+P(b_2|a_1)\lambda(b_2)=(.7)1+(.3)1=1$$
$$\lambda_B(a_2)=P(b_1|a_2)\lambda(b_1)+P(b_2|a_2)\lambda(b_2)=(.2)1+(.8)1=1$$

Debido a lo anterior su valor λ de **A** se ve afectado por el mensaje que **A** recibe de **B** por lo que:

$$\lambda(a_1)=\lambda_B(a_1)\lambda_D(a_1)=(1)1=1$$
$$\lambda(a_2)=\lambda_B(a_2)\lambda_D(a_2)=(1)1=1$$

Finalmente P(A) será:

$$P(a_1)=P(a_1|W)=\alpha\lambda(a_1)\pi(a_1)=\alpha(1)(.1)=\alpha(.1)$$
$$P(a_2)=P(a_2|W)=\alpha\lambda(a_2)\pi(a_2)=\alpha(1)(.9)=\alpha(.9)$$

después de normalizar tenemos P(A)=(.1,.9).

En la práctica no es necesario calcular el valor de P(A) puesto que **A** es la raíz. Su probabilidad ya estaba indicada en la red, pero se obtuvo por motivos de explicación.

De la forma semejante a P(A) y P(B) se calculan las probabilidades *a priori* de C y D, obteniendo los resultados siguientes:

P(C)=(.10075, .89925)
P(D)=(.44, .56)

Después de calcular los valores anteriores la red queda en el estado mostrado en la figura 2-5. Este es el estado de la red antes de que algunas variables sean asignadas. Note que todos los valores λ y mensajes λ son iguales a 1. En general los valores λ y los mensajes λ toman el valor 1 cuando no existen variables asignadas como ha sido el caso hasta aquí.

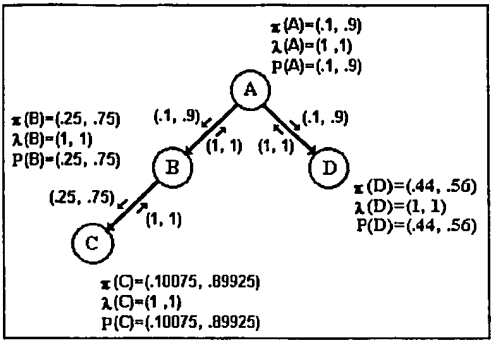


Figura 2-5. Estado de la red antes de la asignación de alguna variable.

Suponer que **B** es asignada por $\mathbf{b}_1$. Debido a (2.20), este hecho cambia el valor λ de **B** a (1,0) y debido a (2.24) el mensaje π de **B** a **C** cambia a (1,0). Ahora $\mathbf{W}=\{\mathbf{B}\}$ y ello afectará la P(E|W) para cada variable **E** en la red. Es decir, al asignar **B,** los valores λ y π de las otras variables se ven afectados por medio de la propagación de los nuevos mensajes λ y π, obteniendo así una nueva P(E|W).

El padre de B en este caso **A** obtiene un nuevo valor λ debido al nuevo mensaje λ de **B** (ec.2.20.). Sin embargo el valor π de **A** no cambia. En la misma forma, si **A** tuviera un padre, el nuevo valor λ de **A** causaría que **A** envíe un nuevo mensaje λ a su padre, cambiando así el valor λ de su padre, mientras su valor π se mantiene constante. Así cuando una variable es asignada, los valores λ de todos sus ancestros cambian debido a que reciben nuevos mensajes λ, continuando así la propagación hasta alcanzar la raíz, pero los valores π no cambian. El nuevo mensaje λ de **B** a **A** también afecta a sus otros hijos de **A**. Cada uno de ellos recibe un nuevo mensaje π debido a (2.24) y consecuentemente sus valores π cambian debido a (2.22). Si su hijo de **A** tiene hijos, el cambio en su valor π causa un cambio en el mensaje π que el envía a sus hijos. La propagación continua de esta manera hasta llegar a las hojas de el árbol. Finalmente el nuevo mensaje π de **B** a **C** causa que el valor π de **C** cambie (con respecto a 2.22). Si **C** tuviera hijos la propagación de los mensajes π continuaría hacía abajo de el árbol hasta alcanzar las hojas.

Hemos visto que las probabilidades *a posteriori* basadas en la asignación de **B**, pueden determinarse de la propagación de los mensajes λ y π. Cuando la propagación termina la red alcanza un nuevo estado basado en la asignación de **B**. Antes de analizar la asignación de más de una variable veremos lo que ocurre en el esquema de propagación cuando **B** es asignada.

Suponiendo que a **B** se le asigna $\mathbf{b}_1$. Entonces $\mathbf{W}=\{\mathbf{B}\}$, lo cual provoca que:

$$\lambda(B)=(1,0)$$
$$P(B|W)=(1,0)$$

B envía un nuevo mensaje π a **C**:

$$\pi_C(B)=(1,0)$$

por consiguiente **C** obtiene un nuevo valor π:

$$\pi(c_1)=P(c_1|b_1)\pi_C(b_1)+P(c_1|b_2)\pi_C(b_2)=(.4)1+(.001)0=.4$$
$$\pi(c_2)=P(c_2|b_1)\pi_C(b_1)+P(c_2|b_2)\pi_C(b_2)=(.6)1+(.999)0=.6$$

entonces la P(C|W) será:

$$P(c_1|W)=\alpha\lambda(c_1)\pi(c_1)=\alpha(1)(.4)=\alpha(.4)$$
$$P(c_2|W)=\alpha\lambda(c_2)\pi(c_2)=\alpha(1)(.6)=\alpha(.6)$$

Normalizando obtenemos P(C|W)=(.4, .6). **B** envía un nuevo mensaje λ a **A**:

$$\lambda_B(a_1)=P(b_1|a_1)\lambda(b_1)+P(b_2|a_1)\lambda(b_2)=(.7)1+(.3)0=.7$$
$$\lambda_B(a_2)=P(b_1|a_2)\lambda(b_1)+P(b_2|a_2)\lambda(b_2)=(.2)1+(.8)0=.2$$

Por lo que **A** obtiene un nuevo valor λ:

$$\lambda(a_1)=\lambda_B(a_1)\lambda_C(a_1)=(.7)1=.7$$
$$\lambda(a_2)=\lambda_B(a_2)\lambda_C(a_2)=(.2)1=.2$$

El nuevo valor de la P(A|W) será:

$$P(a_1|W)=\alpha\lambda(a_1)\pi(a_1)=\alpha(.7)(.1)=\alpha(.07)$$
$$P(a_2|W)=\alpha\lambda(a_2)\pi(a_2)=\alpha(.2)(.9)=\alpha(.18)$$

Normalizando tenemos:

$$P(a_1|W)=\frac{\alpha(.07)}{\alpha(.07)+\alpha(.18)}=.28$$
$$P(a_2|W)=\frac{\alpha(.18)}{\alpha(.07)+\alpha(.18)}=.72$$

Ahora **A** envía un nuevo mensaje π a **D**. Es posible utilizar (2.24) para calcular este mensaje. Sin embargo si **A** tuviera muchos hijos se requeriría la multiplicación de muchos mensajes. Es esos casos la utilización de la ecuación (2.25) evita el producto de todos los mensajes, así podemos utilizar la P(A|W) para determinar el nuevo mensaje π de **A** a **D**. De acuerdo con (2.25) tenemos:

$$\pi_D(a_1)=\alpha\frac{P(a_1|W)}{\lambda_D(a_1)}=\alpha\frac{.28}{1}=\alpha(.28)$$
$$\pi_D(a_2)=\alpha\frac{P(a_2|W)}{\lambda_D(a_2)}=\alpha\frac{.72}{1}=\alpha(.72)$$

por lo que normalizando $\pi_D(A)=(.28, .72)$. El nuevo valor π de **D** esta dado por:

$$\pi(d_1)=P(d_1|a_1)\pi_D(a_1)+P(d_1|a_2)\pi_D(a_2)=(.8).28+(.4).72=.512$$
$$\pi(d_2)=P(d_2|a_1)\pi_D(a_1)+P(d_2|a_2)\pi_D(a_2)=(.2).28+(.6).72=.488$$

Por último calculando P(D|W) y normalizando obtenemos

$$P(D|W)=(.512, .488)$$

La red se encuentra ahora en un estado basado en la asignación de **B**. Este estado es mostrado en la figura 2.6.

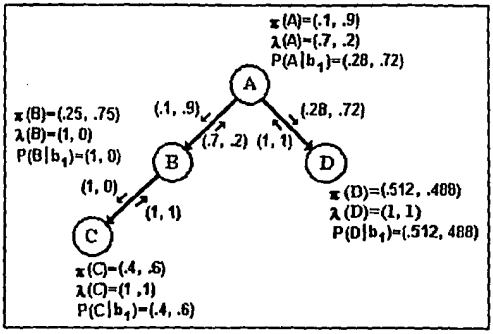


Figura 2.6. Estado de la red después que B fue asignada.

Ahora veremos la asignación de más de una variable continuando con el ejemplo anterior y la consideración que **B** ya fue asignada.

Suponer que **D** es asignada por $\mathbf{d}_2$. Por lo que W={B,D} y

$$\lambda(D)=(0,1)$$
$$P(D|W)=(0,1)$$

D enviará nuevos mensajes λ a **A**:

$$\lambda_D(a_1)=P(d_1|a_1)\lambda(d_1)+P(d_2|a_1)\lambda(d_2)=(.8)0+(.2)1=.2$$
$$\lambda_D(a_2)=P(d_1|a_2)\lambda(d_1)+P(d_2|a_2)\lambda(d_2)=(.4)0+(.6)1=.6$$

Cuando **A** recibe nuevos mensajes λ de **D**, **A** obtiene un nuevo valor λ.

$$\lambda(a_1)=\lambda_B(a_1)\lambda_D(a_1)=(.7).2=.14$$
$$\lambda(a_2)=\lambda_B(a_2)\lambda_D(a_2)=(.2).6=.12$$

Por lo que la P(A|W) será:

$$P(a_1|W)=\alpha\lambda(a_1)\pi(a_1)=\alpha(.14)(.1)=\alpha(.014)$$
$$P(a_2|W)=\alpha\lambda(a_2)\pi(a_2)=\alpha(.12)(.9)=\alpha(.108)$$

Normalizando produce P(A|W)=(.1148, .8852).

Puesto que la propagación ya ha alcanzado la raíz ahora se propagará de arriba hacía abajo de manera que **A** enviará mensajes π a sus otros hijos. En este caso el otro hijo de **A** es **B**, por lo que el mensaje π de **A** a **B** será.

$$\pi_B(a_1)=\frac{P(a_1|W)}{\lambda_B(a_1)}=\frac{.1148}{.7}=.164$$

$$\pi_B(a_2)=\frac{P(a_2|W)}{\lambda_B(a_2)}=\frac{.8852}{.2}=4.426$$

Debemos observar que aunque **A** envía un mensaje π a **B**, los valores λ, π y la P(B|W) no cambiarán, esto se debe a que **B** ya había sido asignada y por lo tanto ella y sus descendientes dejarán de ser influenciados por información proveniente de su padre, **A**. Entonces la propagación termina. El estado de la red se muestra en la figura 2-7.

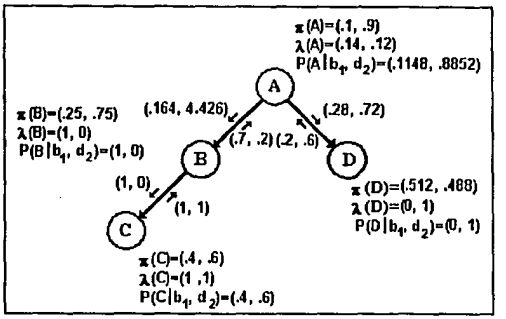


Figura 2-7. Estado de la red después de asignar B y D.

2.6. PROPAGACION DE PROBABILIDAD EN POLI-ARBOLES

El método de propagación desarrollado en las secciones previas solo trabaja en árboles, por lo que esta restringido a una causa por variable. Una extensión del método de propagación para redes un poco más complejas llamadas "*poliárboles*" fue propuesta por Kim y Pearl [83]. En un poliárbol cada nodo puede tener múltiples padres como se observa en la figura 2-2b. Por ejemplo en medicina un síntoma puede ser causado por más de una enfermedad. Si se conoce la presencia de cierto síntoma y a su vez se sabe de la presencia de una enfermedad **X**, esta enfermedad explicará el síntoma, por consiguiente hace las otras enfermedades menos probables. El método requiere que el GAD sea una red conectada simple, es decir, que sólo exista una trayectoria entre un par de nodos dados.

El mecanismo de propagación se deriva en una forma similar al de árboles. Considerando que algún nodo **B** en la red tiene **n** padres y **m** hijos. Este nodo divide la red en un subgráfico superior, que contiene las evidencias que vienen de cada uno de sus padres: $W^+_{B1},\ldots W^+_{Bn}$; y en **m** subgráficos inferiores, con la evidencia $W^-_{B1},\ldots, W^-_{Bm}$, para el subgráfico que corresponde a cada hijo. Por el teorema de Bayes y el hecho que los hijos son condicionalmente

independientes, obtenemos (en forma similar a 2.12) una expresión para calcular la probabilidad *a posteriori* de **B** dada la evidencia **W**.

$$P(b_i|W) = \alpha P(b_i|W_{B1}^+,...W_{Bn}^+)P(W_{B1}^-|b_i)...P(W_{Bm}^-|b_i) \qquad (2.31)$$

Condicionando (2.31) sobre los posibles valores de cada padre $\mathbf{A^1},..., \mathbf{A^n}$, obtenemos:

$$P(b_i|W) = \alpha\left[\sum_{j1}...\sum_{jn} P(b_i|A_{j1}^1,...,A_{jn}^n)P(A_{j1}^1|W_{B1}^+)...P(A_{jn}^n|W_{Bn}^+)\right] x\ P(W_{B1}^-|(b_i)...P(W_{Bm}^-|b_i) \qquad (2.32)$$

y usando la definición previa para λ y π llegamos a:

$$P(b_i) = \alpha\left[\sum_{j1}...\sum_{jn} P(b_i|A_{j1}^1,...,A_{jn}^n)\pi(A_{j1}^1)...\pi(A_{jn}^n)\right] x\ \lambda_1(b_i)...\lambda_m(b_i) \qquad (2.33)$$

La ecuación previa muestra que las probabilidades *a posteriori* para cada variable pueden ser actualizadas localmente, por medio de los vectores π que vienen de cada padre y de los vectores λ de cada hijo. Es posible desarrollar un esquema similar al de propagación de probabilidad en árboles, basado en la comunicación por medio de mensajes en un poliárbol utilizando (2.33). Cuando algunos nodos son asignados las probabilidades se propagan hasta llegar a todas las raíces, entonces se propaga hacía abajo hasta alcanzar las hojas, donde la propagación termina y la red consigue un nuevo estado. Esto requiere de un número finito de actualizaciones donde el equilibrio de la red es obtenido en un tiempo que es proporcional al diámetro de la red. La diferencia principal con el método de árboles es que se requiere la probabilidad condicional conjunta de un nodo dados sus padres: $\mathbf{P(b_i|A_j,...,A_n)}$.

2.7. APRENDIZAJE PARAMETRICO

Una de las principales ventajas de utilizar teoría de probabilidad para representar conocimiento con incertidumbre, se debe a que las probabilidades pueden ser estimadas de datos empíricos y mejoradas conforme se procesen más casos, a esta mejora se le llama *aprendizaje paramétrico*. El proceso se inicia con información de el experto, es decir la estructura de la red, y con la obtención de las distribuciones de probabilidad *a priori* y condicional de datos existentes.

Se requiere estimar la distribución de probabilidad *a priori* para el nodo raíz $\mathbf{P(a_i)}$ denotada por **P(A)**, y la distribución de probabilidad condicional de cada nodo dado su padre $\mathbf{P(b_j|a_i)}$ denotada por **P(B|A)**. Asumimos que cada nodo representa una variable multivaluada discreta con **N** posibles valores, **P(A)** será almacenada como un vector de **N** dimensiones y **P(B|A)** como una matriz de **N x N** dimensiones. Cada entrada en P(A) y P(B|A) será almacenada como una razón entera.

Dada una observación y el valor actual de cada nodo, es trivial actualizar los parámetros del sistema incrementando las probabilidades *a priori* y las matrices de probabilidad condicional en uno. Si se observa $\mathbf{A_k}$ y $\mathbf{B_l}$, entonces las probabilidades serán actualizadas de la forma siguiente [Sucar 92]:

Para la probabilidad *a priori*.

$$P(a_i) = \frac{N(a_i) + 1}{\sum_{j=0}^{n} N(a_j) + 1} \qquad Para\ i=k. \tag{2,34}$$

$$P(a_i) = \frac{N(a_i)}{\sum_{j=0}^{n} N(a_j) + 1} \qquad Para\ i\neq k. \tag{2.35}$$

Donde: N = Número de casos, k= índice del valor asignado, n=total de valores.

Para las probabilidades condicionales.

$$P(b_j|a_i) = \frac{N(b_j|a_i) + 1}{\sum_{m=0}^{n} N(b_m|a_i) + 1} \qquad Para\ i=k\ y\ j=l. \tag{2.36}$$

$$P(b_j|a_i) = \frac{N(b_j|a_i)}{\sum_{m=0}^{n} N(b_m|a_i) + 1} \qquad Para\ i\neq k\ y\ j=l. \tag{2.37}$$

Donde: N= Número de casos, k= índice del valor del padre, l= índice del valor asignado.

Existen otros métodos alternativos para llevar a cabo el aprendizaje. Uno de ellos sugiere comenzar con todos los valores puestos a cero y actualizarlos con cada observación. De esta forma los parámetros probabilísticos son aprendidos por el sistema y su desarrollo es mejorado conforme más datos son acumulados. Otra alternativa sugiere comenzar con información estimada por el experto como un valor inicial, representada también como una razón, mejorándola con datos estadísticos. Este acercamiento es útil en casos en los cuales no se tienen bastantes datos para obtener una estimación inicial.

2.8. EVALUACION DE RESULTADOS

En esta sección se presenta la forma de evaluar los resultados obtenidos al aplicar el método de propagación. La evaluación se realiza midiendo la diferencia entre las predicciones del sistema (resultados) y la valoración del experto. Esta evaluación se obtiene al calcular la diferencia cuadrada entre la probabilidad *a posteriori* y el valor actual de acuerdo al experto a el cual se asigna una probabilidad igual a 1. Sumando el error cuadrado para todos los ejemplos se obtiene la medida de la capacidad predictiva del sistema llamada *medida de Brier* la cual se define como [Sucar 92]:

$$B = \sum_i (1 - P_i)^2 \qquad (2.38)$$

Dividiendo la medida de Brier entre el número de ejemplos (N) obtenemos una estimación para el error predictivo promedio el cual definimos como:

$$mB = \frac{B}{N} = \frac{\sum_i (1 - P_i)^2}{N} \qquad (2.39)$$

Obteniendo de esta manera estimación de la veracidad de los resultados derivados del sistema.

2.9. CONCLUSIONES

En este capítulo analizamos la teoría en la cual se fundamenta el método de propagación de probabilidades propuesto por Pearl, basado en teoría de probabilidad. Hemos visto que este método es adecuado para la representación y razonamiento con incertidumbre ya que:

- Permite representar claramente las relaciones existentes en el conocimiento.
- No supone independencias condicionales injustificadas.
- Evita la utilización de técnicas de ajuste, al tener un esquema de propagación fundamentado en teoría de probabilidad, principalmente en las relaciones representadas en la estructura de la red y en las distribuciones de probabilidad condicional.

Aunque este método está restringido a árboles y poliárboles, donde funciona eficientemente, existen otros métodos para estructuras más complejas.

Entre las principales ventajas del método, se tienen los pocos requerimientos de almacenamiento y su eficiencia en términos de tiempo. Si **n** es el número máximo de alternativas en una variable, se requerirá almacenar $\mathbf{n^2}$ probabilidades condicionales, dos vectores n-dimensionales para los valores λ y π en cada nodo y dos vectores n-dimensionales en cada arco para los mensajes correspondientes. Por lo que un árbol requerirá almacenar $\mathbf{n^2+2n}$ valores en cada nodo y **2n** valores en cada arco.

Con respecto al tiempo, si **m** es el número máximo de hijos que puede tener una variable, el número de multiplicaciones para actualizar una variable será: **n** para calcular el mensaje π, $\mathbf{n^2}$ **para el mensaje** λ, $\mathbf{n^2}$ **para calcular el valor** π, **nm** para el valor λ y **n** para la nueva probabilidad. De aquí el número total de multiplicaciones para actualizar una variable es $\mathbf{2n^2+nm+2n}$. Es decir el tiempo de actualización es lineal con respecto al número de variables en la red.

Ejemplos de sistema experto que utilizan este método son: Binfort, Levitt y Mann [87] para el razonamiento con geometría en visión mecánica (*ADRIES* y *SUCCESSOR*), Spielgelhalter [86b] en medicina y Sucar [91] en visión para endoscopía.

Después de analizar los métodos de representación de conocimiento y razonamiento probabilísticos, en los capítulos siguientes veremos la implementación de estos métodos tomando como base las técnicas de desarrollo de *software* orientadas a objetos.

DESCRIPCION DE LA HERRAMIENTA PARA SISTEMAS EXPERTOS CON INCERTIDUMBRE

3.1. INTRODUCCION

En este capítulo realizaremos la descripción de la herramienta para manejo de incertidumbre en sistemas expertos. Nos enfocaremos a la técnica de redes probabilísticas, ya que permiten la representación y razonamiento con incertidumbre, contemplando la aplicación de las técnicas para propagación de probabilidades, aprendizaje paramétrico y evaluación dentro de la herramienta.

Este trabajo surge de la necesidad de contar con sistemas expertos para diversas aplicaciones en el monitoreo y control de redes eléctricas. Algunas de estas aplicaciones como el diagnóstico de fallas, presentan incertidumbre en el conocimiento y los datos, por lo que es importante el contar con una herramienta que maneje dicha incertidumbre en una forma adecuada. Es importante señalar que aunque la idea de esta herramienta surge para su aplicación en redes eléctricas, esta puede ser aplicada en una gran diversidad de áreas en las cuales exista incertidumbre en el conocimiento por lo que tiene una aplicación múltiple. Unicamente requiere la especificación de la red probabilística particular para cada dominio de aplicación.

Esta herramienta ha sido elaborada bajo una metodología de desarrollo de software orientada a objetos. Esta se divide en cuatro fases principales: Descripción, Especificación, Diseño y Pruebas. En este capítulo se cubre la fase de descripción, el capítulo 4 la especificación, el capítulo 5 el diseño, y en el capítulo 6 se presentan varios ejemplos de aplicación en que ha sido probada la herramienta. La herramienta se desarrolló en una máquina HP-9000 modelo 710, con disco de 420 megabytes y 16 megabytes en RAM.

3.2. DEFINICION DE LOS REQUERIMIENTOS

A continuación veremos brevemente cada uno de los puntos de importancia con los que la herramienta debe contar y en la próxima sección profundizaremos en cada uno de ellos.

3.2.1. Representación

La herramienta debe permitir la representación y el manejo de conocimiento con incertidumbre en forma coherente, para ello se planea utilizar la técnica de redes probabilísticas

ya que cuenta con una base teórica bien fundamentada.

3.2.2. Entrada/Salida

Este punto se refiere a la entrada y salida de información a la base de conocimiento, es decir la información de la red probabilística. Estas pueden ser de tres formas:

- Información entrada del teclado.
- Almacenamiento de información en disco.
- Lectura de información de disco.

3.2.3. Despliegue

Puesto que la herramienta debe de estar en contacto directo con el usuario, esta debe mostrar la información con la cual esta trabajando cuando el usuario así lo desee, dándole un panorama general del estado de la red probabilística. Esta información será particular a cada nodo, mostrando los datos referentes a él.

3.2.4. Lectura de valores

Para que la herramienta lleve a cabo alguna de sus funciones de operación, requiere la lectura de valores de los nodos que conforman la red probabilística. La lectura se podrá realizar de dos formas: del teclado o de un archivo externo. Esta última permitirá trabajar con datos procedentes de un sistema de adquisición de datos.

3.2.5. Funciones

Este es uno de los puntos de mayor interés en la herramienta ya que se contempla que esta lleve a cabo tres funciones diferentes. Estas son:

- Propagación de probabilidades.
- Aprendizaje paramétrico.
- Evaluación.

La función de propagación permitirá llevar a cabo tres tipos diferentes de propagación. En cada uno de ellos se deben realizar las funciones de aprendizaje y evaluación correspondiente con ese

tipo de propagación. Los tipos de propagación son los siguientes [2]:

a) **Propagación evidencial.** Asignar valores a las hojas y propagar su efecto hacia la raíz. Se aplica a estructuras de tipo árbol. Para realizar este tipo de propagación en primer lugar se debe inicializar la red, después asignar un valor a todos los nodos hojas y finalmente propagar hacía arriba del árbol hasta alcanzar el nodo raíz obteniendo así su probabilidad *a posteriori*. Para la propagación se requieren los valores λ de cada nodo y los mensajes λ que un nodo hijo manda a sus padres. Adicionalmente para el cálculo de la probabilidad *a posteriori* del nodo raíz se debe contar con el valor π del mismo, el cual es igual al valor de su probabilidad *a priori*. Para el resto de los nodos no se requiere su valor π. La figura 3-1 ilustra este tipo de propagación.

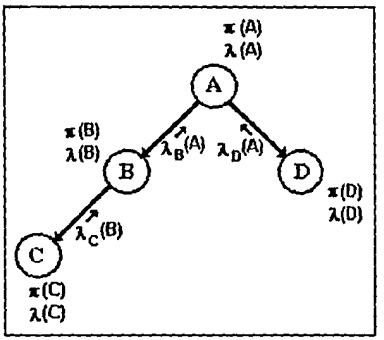


Figura 3-1. Propagación evidencial.

b) **Propagación bidireccional.** Asignar un valor a cualquier variable sin importar si es raíz, intermedia u hoja y propagar sus efectos hacia la raíz y hacia las hojas. Se aplica a estructuras de tipo árbol. Antes de llevar a cabo alguna propagación es necesario realizar el proceso de inicialización. En este tipo de propagación será posible asignarle un valor a cualquier nodo sin importar si es raíz, intermedio u hoja, sin existir límite en la cantidad de variables a las que se le puede asignar un valor, así como en el orden en que ellas son conocidas. La propagación se debe realizar hacia arriba y hacia abajo de la variable cuyo valor es conocido, hasta alcanzar el nodo raíz y los nodos hojas. Debido a lo anterior el método resulta un poco más complejo que para propagación evidencial, este requiere que se calcule lo siguiente: Los valores λ y π para cada nodo, los mensajes λ que un nodo hijo manda a su padre y los mensajes π que un nodo padre manda a sus hijos (ver figura 3-2).

---

[2] Para un estudio más detallado de las bases teóricas de los puntos mencionados en esta sección y en la anterior referirse al Capítulo 2.

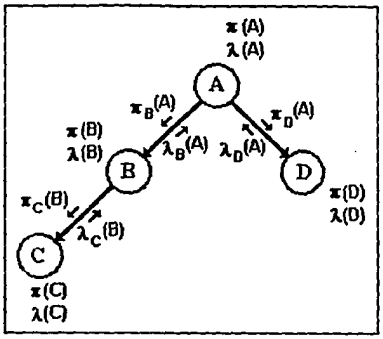


Figura 3-2. Propagación bidireccional.

La herramienta debe calcular las probabilidades *a posteriori* para todos los nodos del árbol probabilístico dadas las variables conocidas y no sólo para el nodo raíz.

c) **Propagación de probabilidad en poliárboles.** Propagación evidencial y bidireccional en estructuras de tipo poliárbol. El método es similar a los anteriores, la única diferencia se encuentra en el cálculo de los mensajes λ y de los valores π y del hecho que un nodo puede tener varios padres.

3.2.6. Variables

De acuerdo a sus múltiples aplicaciones, la herramienta debe trabajar con diferentes tipos de variables. Conforme a lo anterior la herramienta permitirá el manejo de tres tipos diferentes de variables en una misma aplicación. Estos son:

a) *Variables binarias.* Tendrán únicamente el valor de cero o el valor de uno.

b) *Variables continuas.* Pueden tener valores en un rango determinado.

c) *Variables discretas.* Pueden tener diversos valores discretos asignados por el usuario.

3.2.7. Validación

Validación de la información de entrada a la herramienta y de la relación existente entre las diversas variables que conforman la red probabilística.

3.2.8. Edición

Una vez que la herramienta tiene cargada una red probabilística, debe contar con la posibilidad de añadir más elementos a la estructura de la red, así como la posibilidad de borrar algunos nodos ya existentes.

3.2.9. Interfaz hombre/máquina

Desarrollo de la herramienta en un ambiente gráfico adecuado el cual permita una buena comunicación y acoplamiento entre el usuario y la herramienta.

3.2.10. Interfaz control supervisorio

El sistema podrá interactuar con un sistema de control supervisorio de manera que se generen soluciones en tiempo-real. Por esta razón se contempla la lectura de valores de archivos externos.

3.3. ANALISIS DE LOS REQUERIMIENTOS

Hasta aquí hemos visto brevemente las características deseadas en la herramienta. En esta sección profundizaremos en la descripción de cada una de ellas, así como en su validación teórica (para un estudio más detallado de ésta referirse al capítulo anterior).

3.3.1. Representación

Dadas las características del conocimiento del mundo real y la incertidumbre que existe en el mismo, surgen problemas en su representación dentro de un sistema basado en conocimiento (sistema experto). Una técnica desarrollada recientemente la cual permite representar este tipo de conocimiento en una forma confiable son las redes probabilísticas.

Existen diversos algoritmos para efectuar propagación de probabilidad utilizando la técnica de redes probabilísticas, dependiendo del tipo de estructuras. Pearl [82] desarrolló un método para realizar propagación de probabilidad en redes cuya estructura forma un árbol (figura 3-3a), posteriormente este método fue extendido a estructuras más complejas llamadas poli-árboles [Kim 83] (figura 3-3b). El método de Pearl será aplicado en este trabajo para llevar a cabo la función de propagación.

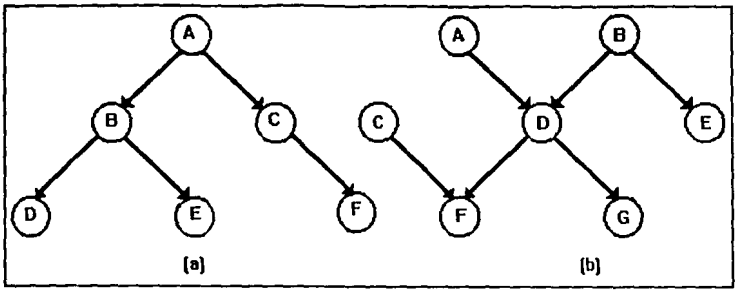


Figura 3-3. a) Un árbol y b) poli-árbol probabilísticos.

3.3.2. Entrada/Salida

Las tres formas en las que se puede accesar información a las bases que contienen el conocimiento dentro de la herramienta son las siguientes:

a) Información entrada del teclado. Esta opción obtiene la información requerida por la herramienta del teclado y es borrada cuando el usuario sale de la herramienta.

b) Almacenamiento de información en disco. Esta opción permitirá guardar información concerniente con la estructura de la red probabilística así como de las variables que la conforman en disco. Esta información constituye la base de conocimientos quedando almacenada para ser utilizada en una ejecución futura.

c) Lectura de información de disco. Esta opción permite leer la información de la base de conocimientos, que previamente fue almacenada (inciso anterior).

La información será almacenada en diferentes archivos, uno que contendrá la estructura de la red probabilística, y uno con la información correspondiente a cada variable (nodo).

3.3.3. Despliegue

Como se menciono en el punto 3.2.3., la herramienta debe de estar en comunicación directa con el usuario de tal forma que le muestre el estado de la red, siempre y cuando esta ya se encuentre cargada. El usuario podrá observar la información de una variable en estado inicial o después de llevar a cabo alguna de las funciones de la herramienta. La información desplegada es tomada de los dos archivos mencionados en el punto anterior y de cálculos realizados por el sistema. Esta información será la siguiente:

- Nombre de la variable.
- Nivel del nodo.

- Tipo de nodo (Raíz, Intermedio u hoja).
- Tipo de variable (Binaria, continua o discreta).
- Número de elementos.
- Número de elementos del padre.
- Rango(máximo y mínimo).
- Intervalo.
- Nombre del padre.
- Nombre de los hijos.
- Valores de la variable.
- Probabilidades *a priori*, condicionales y *a posteriori*.
- Valor Lambda.
- Valor Pi.
- Indice del valor asignado.

3.3.4. Lectura de Valores

De acuerdo con 3.2.4, antes que la herramienta lleve a cabo alguna de sus funciones requiere leer los valores con los cuales va a trabajar. En una primera etapa cuando los datos son cargados del teclado, el sistema preguntará automáticamente el valor que será asignado a cada nodo, esto para todos los nodos hojas de la red probabilística. El sistema debe verificar que este valor sea válido para el nodo dado. Cuando la información es leída de un archivo externo únicamente se cargan los valores asignados a cada nodo. Estos valores son para todos los nodos hojas.

En esta primera etapa la propagación se realiza de las hojas hacia el nodo raíz, es decir, en forma evidencial. En una segunda etapa, será posible asignar un valor a cualquier nodo de la red sin importar si es nodo hoja o no. En este caso será necesario checar si el nodo existe y si su valor es válido. Aquí se llevará a cabo una propagación bidireccional es decir hacia arriba y hacia abajo del nodo cuyo valor es conocido. En esta etapa también será posible leer valores de archivos externos.

3.3.5. Funciones

Las principales funciones que el sistema va ha realizar son: propagación, aprendizaje y evaluación. A continuación se describe cada una de ellas. Para un estudio más completo de las misma referirse al capítulo anterior.

PROPAGACION. Esta función permite realizar propagación de probabilidades utilizando el método de Pearl. Este método requiere que la red sea un GAD y contar con información referente a cada nodo tal como: nombre de el nodo, quien es su padre, si tiene hijos, sus valores π y λ, el mensaje λ que un hijo manda a un padre, el mensaje π que un padre manda a un hijo, las probabilidades condicionales para los nodos hojas e intermedios y la probabilidad *a priori* para el nodo raíz. Tanto los valores como los mensajes λ y π son calculados conforme el método se va desarrollando, sin requerirse que éstos sean almacenados en archivos.

El método esta dividido en dos partes: la primera se refiere a la inicialización del árbol probabilístico dejando al mismo en un estado inicial E_1. La segunda parte se refiere a un proceso de actualización. Este se lleva a cabo después de que a una ó más variables se les ha asignado un valor, determinando así la probabilidad *a posteriori* de las demás variables, dadas las variables cuyo valor es conocido, dejando la red en un estado E_2. Si a una segunda variable se le asigna un valor, se continua el método partiendo ahora de la red en estado E_2, determinando así la probabilidad *a posteriori* de las otras variables de la red dadas las variables conocidas. El resultado obtenido de la asignación de valores a las variables, no depende del orden en el cual sus valores son asignados.

Cabe mencionar que este procedimiento es el mismo para propagación evidencial y propagación bidireccional. La diferencia está en que en el primero se le asignan valores a todos los nodos hojas y no existen mensajes π de un padre a un hijo. En el segundo es posible asignar valores a cualquier nodo. Una extensión de este método se aplica a poliárboles y difiere únicamente en el número de padres que un nodo puede tener.

APRENDIZAJE. Una de las principales ventajas de utilizar teoría de probabilidad para representar conocimiento con incertidumbre es que las probabilidades pueden ser estimadas de datos empíricos, y mejoradas conforme se tienen más casos [Sucar 93]. El proceso se inicia con información del experto, es decir, la estructura de la red y las distribuciones de probabilidad *a priori* y condicional. Debido a que cada nodo representa una variable multivaluada con **N** posibles valores, **P(A)** es almacenada como un vector de **N** dimensiones y **P(B|A)** como una matriz de **N x N** dimensiones.

Dada una observación y el valor actual de cada nodo, es trivial actualizar los parámetros del sistema, incrementando las probabilidades *a priori* y las matrices de probabilidad condicional. Para ello las probabilidades son almacenadas como fracciones (número de casos). Las probabilidades serán actualizadas conforme a las ecuaciones mencionadas en la sección 2.7.

EVALUACION. Esta función muestra la evaluación de la parte probabilística del sistema, midiendo la diferencia entre la predicción del sistema y la valoración del experto. Esta evaluación se realiza al calcular la diferencia cuadrada entre la probabilidad *a posteriori* y el valor actual de acuerdo al experto, para lo cual se asigna una probabilidad igual a uno. Una extensión de esta función permitirá cambiar este valor igual a 1 por un valor obtenido del experto en el momento de llevar a cabo la evaluación, el cual puede ser diferente de 1 si el experto así lo desea. Sumando el error cuadrado para todos los ejemplos se obtiene la medida de la capacidad predictiva del sistema llamada *medida de Brier* y dividiendo esta entre el número de ejemplos se obtiene el *error predictivo promedio*. Ambos términos ya fueron definidos en la sección 2.8.

Esta función permitirá observar las estadísticas para un nodo y sus valores. El sistema despliega información para cada uno de los valores del nodo proporcionando la siguiente información: el número de veces que un valor ha sido elegido, el total de resultados correctos, el porcentaje de resultados correctos, el error cuadrado y el promedio del error cuadrado. La información mencionada anteriormente podrá ser almacenada y leída de un archivo. De esta forma, cuando se tengan más ejemplos, las estadísticas puedan ser actualizadas con la

información reciente; siempre y cuando no se haya llevado a cabo alguna función de aprendizaje, en tal caso las estadísticas deben ser borradas. La función de evaluación varía un poco con respecto al tipo de propagación que se realice, ya que para propagación evidencial únicamente se evaluará el nodo raíz y en los otros casos será posible evaluar cualquier nodo.

La figura 3-4 ilustra el funcionamiento de la herramienta. Incluye las principales funciones con las que cuenta la herramienta así como el almacenamiento y carga de la información.

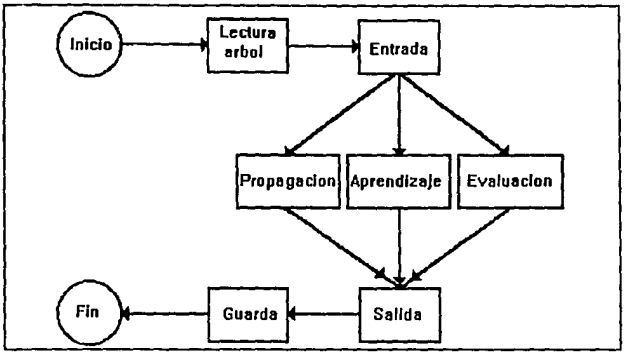


Figura 3-4. Diagrama de flujo del funcionamiento de la herramienta.

3.3.6. Variables

La herramienta permitirá el manejo de tres tipos diferentes de variables en una misma aplicación. Estos tipos permitirán modelar en forma discretizada cualquier tipo de variable que se presente en diversas aplicaciones. Estos son:

a) *Variables binarias.*

b) *Variables continuas.*

c) *Variables discretas.*

Los tres tipos difieren en el rango de valores que cada uno puede tener. Así las *variables binarias* pueden tener únicamente dos valores, uno o cero. Las *variables continuas* se definen dentro de un rango para cada variable y cualquier valor dentro del rango, puede ser asignado a la variable. Por último, las *variables discretas* tienen valores que son definidos previamente por el usuario y cualquier valor de los definidos puede ser asignado a la variable correspondiente.

Debido a los diversos tipos de variables que se van a utilizar, la herramienta deberá verificar que un valor asignado a una variable sea del tipo correspondiente a ella, es decir si la variable es de tipo binaria solamente se deben aceptar valores de cero o uno. De manera semejante si la variable es discreta, la herramienta debe comprobar que el valor asignado a esta variable pertenezca a los valores predeterminados. Si es de tipo continua, el valor debe de estar dentro del rango de valores definidos.

3.3.7. Validación

La herramienta debe de validar toda la información entrada por el usuario y en caso de existir algún error debe de desplegar un mensaje informándole al usuario el error que esta cometiendo. Entre las cosas que se deben validar están las siguientes:

- En primer lugar se debe verificar la existencia del archivo principal, es decir el que guarda la estructura del árbol probabilístico. Se tomará en cuenta que el nombre del archivo podrá tener máximo 10 caracteres y el sistema automáticamente le asignara la extensión. De la misma manera se debe verificar que el nombre de las variables (nodos) no se exceda de 10 caracteres.

- En la construcción de la red probabilística, cuando la herramienta solicite información acerca de los padres de una variable, se debe validar la existencia de los nodos padres y en caso de que estos no existieran la herramienta deberá mandar un mensaje.

- Cuidar que los nodos no se excedan en el número de hijos que pueden tener.

- Cuando se solicite el nombre de algún nodo para realizar con él alguna de las funciones de la herramienta, se debe verificar que este nodo pertenezca a la red probabilística que este en uso, si este no es el caso, la herramienta deberá desplegar un mensaje de error que manifieste la inexistencia de la variable.

- Cuando la herramienta cargue información de un archivo externo, se debe checar que las variables que conforman la red probabilística, concuerden con las variables almacenadas en los archivos externos, en caso de que no coincidan la herramienta debe de indicar esta discrepancia.

- Como se mencionó en 3.3.6., se debe vigilar que los valores asignados a una variable durante la ejecución de la herramienta concuerden con el tipo de valor que corresponda a la variable y en caso de no concernir la herramienta debe detectarlo y desplegar un mensaje indicando este error.

3.3.8. Edición

Una vez que la herramienta tiene cargada una red probabilística será posible modificar la estructura de la red, esto se podrá realizar de dos formas: adicionando más elementos (nodos) a la estructura de la red y borrando nodos ya existentes de la misma.

3.3.9. Interfaz hombre/máquina

La herramienta debe permitir un alto nivel de comunicación entre ella y el usuario de tal forma que el usuario pueda definir su modelo probabilístico en forma amigable. Aunado a lo anterior, la herramienta debe ser fácil de operar. Para ello se contempla la implementación de una interfaz hombre/máquina con el paquete "INTERFACE ARCHITECT" [Interface 92]. Dicho paquete es un constructor de interfases de usuario de segunda generación el cual crea interfases en un estilo consistente con OSF/MOTIF 1.1. [Heller 91]. Esta interfaz será descrita en detalle en el siguiente capítulo.

3.3.10. Interfaz control supervisorio

Una de las finalidades de ésta herramienta, es que pueda interactuar con sistemas de control supervisorio y auxiliar en diversas aplicaciones en el monitoreo y control de redes eléctricas. Para realizar esta interacción se contempla la creación de librerías que contengan el código necesario para llevar a cabo la propagación de probabilidades. Los archivos externos mencionados en 3.2.4, serán la interfaz entre las librerías y el sistema de control supervisorio. Análogo a esto, las librerías tendrán acceso a la estructura del árbol probabilístico, la cual se encuentra almacenada en el archivo correspondiente.

El **SCADA** (Sistema de Control y Adquisición de Datos) almacenará los valores de las variables en un archivo externo a la herramienta, estos valores pasaran a las librerías mencionadas, que auxiliándose del archivo que contiene la estructura del árbol, asignarán los valores a las variables correspondientes. El sistema calculará las probabilidades *a posteriori* y las almacenará en otro archivo externo de salida de donde el **SCADA** las tomará para su aplicación. La figura 3-5 muestra este proceso. Dicha librería consiste de 3 funciones básicas:

1.- Asignar valores a una variable.
2.- Propagación. Propagar los efectos de las variables conocidas a través de la red.
3.- Resultados. Obtener las probabilidades *a posteriori* de las variables de interés.

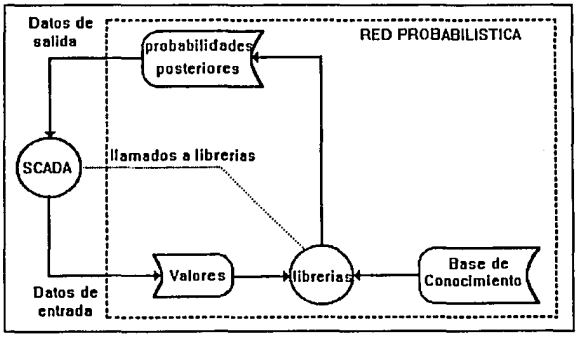


Figura 3-5. Interacción con el sistema de adquisición y control (SCADA).

3.4. CONCLUSIONES

En este capítulo hemos descrito las características con las que debe de contar la herramienta para su aplicación en sistemas expertos. Determinamos tres tipos diferentes de propagación de probabilidades con las cuales la herramienta va a trabajar así, como otras dos funciones adicionales (aprendizaje y evaluación) que complementan la herramienta. Mencionamos la forma en la cual la herramienta va a adquirir información, primero para formar, almacenar y cargar una base de conocimientos representada como una red probabilística, y posteriormente, como recibirá valores para llevar a cabo alguna de las funciones del sistema. Hemos visto la potencialidad de la herramienta para su aplicación en diversas áreas, al permitir manejar diferentes tipos de variables. También mencionamos la importancia de contar con una interfaz hombre/máquina amigable, ya que de ella dependerá el buen uso de la herramienta reflejándose en un funcionamiento adecuado de la misma.

En el capítulo siguiente llevaremos a cabo la especificación de la herramienta. En esta daremos énfasis a los aspectos de interfaz al usuario para cada una de las funciones mencionadas, así como en los comandos con los que contará la herramienta.

CAPITULO 4

ESPECIFICACION DE LA HERRAMIENTA

4.1. INTRODUCCION

La herramienta para desarrollo de redes probabilísticas debe permitir un alto nivel de comunicación con el usuario, de tal forma que el usuario pueda definir su modelo probabilístico en forma amigable. Asimismo la herramienta debe ser fácil de operar. Lo anterior dependerá, en gran parte, del desarrollo de una interfaz hombre/máquina adecuada (de aquí en adelante la llamaremos interfaz con el usuario). Para ello se especifica una interfaz gráfica en base a ventanas y menús de acuerdo a los *estándares* de interfaz con el usuario.

En este capítulo llevaremos a cabo la especificación de la herramienta, en particular de la interfaz con el usuario, detallando cada uno de los comandos con los que contará la herramienta. Desglosaremos cada uno de los comandos referentes de cada menú, asimismo mostraremos las ventanas que el sistema desplegará para dicho comando y señalaremos algunas características de interés. En primer lugar mostraremos el menú principal con cada uno de sus menús describiendo la función de cada uno. Posteriormente iremos desglosando cada sub-menú con cada uno de sus comandos, para todas las funciones de la interfaz con el usuario.

4.2. MENU *PRINCIPAL*

El menú principal será la ventana inicial que la herramienta mostrará al usuario cuando éste la ponga en marcha. En la parte superior de esta ventana se mostrará el nombre de la herramienta, debajo del nombre estará la barra de menús la cual contendrá los siguientes menús:

1.- *Archivo.*

2.- *Desplegar.*

3.- *Funciones.*

4.- *Editar.*

5.- *Datos.*

6.- *Salida.*

La figura 4-1. muestra la ventana principal con sus menús.

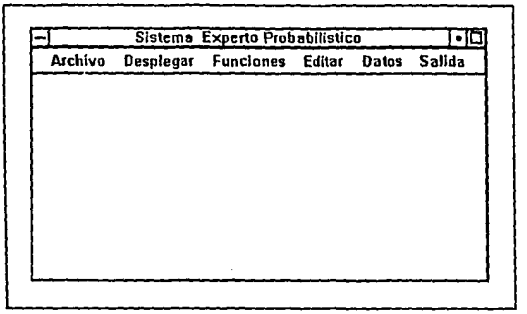


Figura 4-1. Ventana *principal*.

A continuación describiremos cada uno de estos comandos.

Archivo. El menú archivo permitirá realizar operaciones relacionadas con los archivos que contienen la estructura de la red probabilística, tales como cargar una estructura o almacenarla.

Desplegar. Este menú cuenta con comandos que muestran la estructura de la red probabilística, así como toda la información correspondiente a cada nodo de la red.

Funciones. En éste menú están incluidas las funciones que va a realizar el sistema. Dichas funciones ya fueron descritas en 3.3.5.

Editar. Este menú permitirá modificar la estructura de la red probabilística de dos formas: añadiendo nodos a la red o borrando nodos existentes de la misma.

Datos. Como se menciono en 3.3.4., la herramienta podrá leer valores tanto del teclado como de archivos externos. Por medio de esté menú el usuario le indicará a la herramienta como obtendrá los datos.

Salida. El menú salida retornará al usuario al sistema, o re-iniciar todas las variables dentro de la herramienta borrando la base de conocimientos actual.

4.3. MENU *ARCHIVO*

Este menú contiene los comandos que la herramienta podrá realizar sobre los archivos, estos comandos se muestran en la figura 4-2.

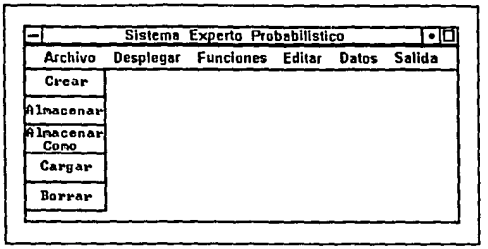


Figura 4-2. Comandos del menú *Archivo*.

4.3.1. Comando *Crear*

Este comando obtiene información del usuario para construir la red probabilística y posteriormente poder realizar alguna de las funciones del sistema. En primer lugar la herramienta debe conocer cual será el nombre de la red probabilística y el número de nodos que esta va a tener, para ello la herramienta desplegará la ventana mostrada en la figura 4-3.

Figura 4-3. Ventana del comando *Crear*.

En segundo lugar este comando mostrará una ventana por medio de la cual la herramienta recibirá los datos referentes a cada nodo en la red (figura 4-4). Dicha ventana contará con tres áreas de texto para entrar el nombre de la variable (o nodo), nombre del padre, y número de elementos. Para que la herramienta registre los dos primeros datos, el usuario deberá oprimir un botón el cual salvará los datos entrados.

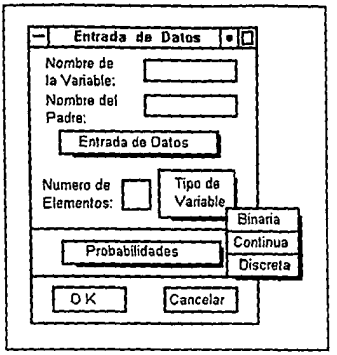


Figura 4-4. Ventana para *entrar datos.*

Como se puede apreciar en la figura 4-4, ésta ventana contiene un menú de opciones titulado *tipo de variable.* Este menú permitirá elegir diferentes tipos de variables con los que la herramienta podrá trabajar, los que están descritos en 3.3.6. Cuando el usuario seleccione una variable binaria la herramienta automáticamente pone el valor de dos en el área de texto para el *número de elementos.* Si la variable es continua el usuario debe especificar un rango, para ello utiliza la ventana mostrada en la figura 4-5a. Si la variable es discreta el usuario deberá entrar cada uno de los valores correspondientes a esa variable, para ello ocupa la ventana de la figura 4-5b.

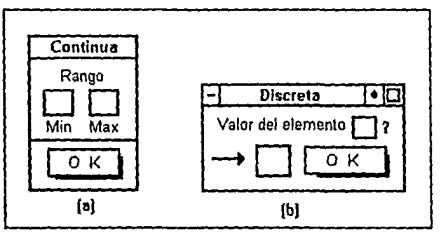


Figura 4-5. Lectura de valores. a) *Variable continua.* b) *Variable discreta.*

La ventana de la figura 4-4 también cuenta con un botón denominado *probabilidades.* Por medio de este botón el usuario entrará las probabilidades correspondientes a cada variable, ya que al oprimirlo presentará una de las ventanas contenidas en la figura 4-6. Si el nodo es raíz exhibirá la ventana (a), si es cualquier otro nodo la ventana (b).

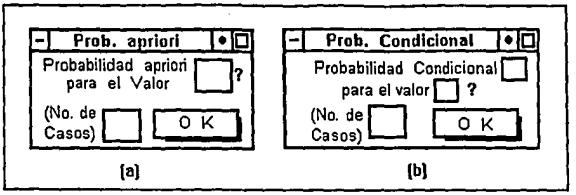


Figura 4-6. Probabilidades, a) *a priori* y b) condicionales.

4.3.2. Comando *Almacenar*

Este comando almacenará toda la información concerniente a la red probabilística en dos tipos diferentes de archivos, uno de ellos contendrá la estructura de la red probabilística y tendrá el nombre especificado en la ventana **crear** para la red probabilística con la extensión **".pt"**. La información correspondiente a cada nodo se almacenará en un archivo propio de ese nodo con la extensión **".var"**. Cuando el usuario desee almacenar una red probabilística la herramienta mostrará la ventana que se muestra en la figura 4-7, la cual será únicamente para confirmar o cancelar la operación.

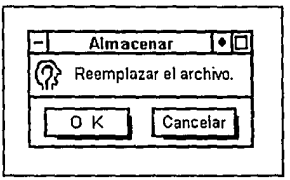


Figura 4-7. Ventana para el comando *Almacenar*.

4.3.3. Comando *Almacenar Como*

Este comando es parecido al anterior, difiere en que aquí el usuario le indica el nombre del archivo con el que se almacenará la red probabilística, pero el archivo referente a cada nodo es el mismo que en el comando anterior. Para llevar a cabo esta operación la herramienta mostrará la ventana de la figura 4-8.

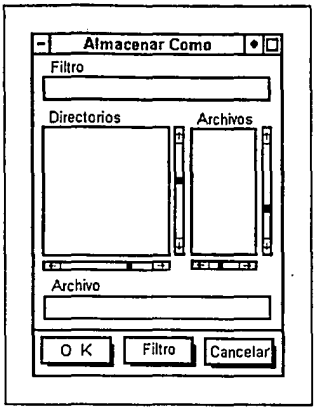


Figura 4-8. Ventana del comando *Almacenar Como*.

Como se menciono en el punto anterior, la información estará almacenada en dos tipos diferentes de archivos. El archivo que contiene la estructura de la red tendrá la siguiente información: el número total de nodos, nombre y padres de cada nodo a excepción del nodo raíz, ya que este no tiene padre. El otro tipo de archivo contendrá información relativa a los nodos del árbol probabilístico, existiendo un archivo para cada uno de ellos. Tendrá almacenada la siguiente información: tipo de variable (binaria, continua o discreta), número de elementos, rango o valores y las probabilidades *a priori* o condicionales dependiendo si el nodo es raíz o no. Para variables continuas se almacenará un rango el cual consiste de un valor máximo y un valor mínimo. Para variables discretas se almacenaran sus valores, uno para cada elemento de la misma. La figura 4-9 muestra la estructura de estos archivos.

Estructura del arbol- <nombre>.pt	
Numero de nodos	
Nombre variable	Nombre del padre
Nodo 0	
Nodo 1	Nodo 0
...	...
Nodo n-1	Nodo i

Informacion de una variable- <nom-var>.var		
Tipo de Variable		
Numero de elementos	Rango o valores	
Probabilidades (apriori o condicionales)		
P(1,1)	...	P(1,n)
...	...	...
P(m,1)	...	P(m,n)

Figura 4-9. Formato de los archivos para la red y variables.

4.3.4. Comando *Cargar*

El comando cargar, le permitirá al usuario recuperar información almacenada previamente en archivos como los que se muestran en la figura 4-9. Cuando el usuario entra a este comando deberá de indicarle a la herramienta que archivo va a cargar, seleccionando alguno de los archivos que contenga la estructura de la red. Para ello la herramienta mostrará la ventana que aparece en la figura 4-10.

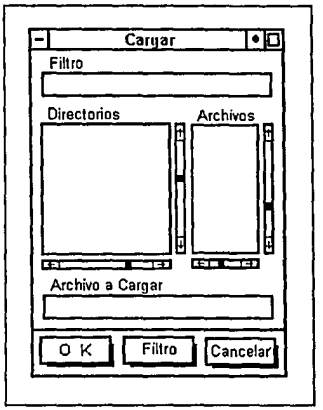


Figura 4-10. Ventana para el comando *Cargar*.

4.3.5. Comando *Borrar*

Esta opción borrará una red probabilística, para ello la herramienta elimina tanto el archivo que contiene la estructura de la red como el archivo correspondiente a cada nodo. El usuario le indicará el nombre de la red probabilística a borrar a través de una ventana como la mostrada en la figura 4-10, sólo que con el título de borrar. Una vez que la herramienta ha registrado que red se va a borrar, desplegará una ventana de mensaje pidiéndole al usuario que confirme la acción, esto únicamente se debe a razones de seguridad.

4.4. MENU *DESPLEGAR*

Este menú permitirá observar la información concerniente a una red probabilística. El menú esta formado por dos comandos, uno para observar la estructura de la red y otro para observar la información correspondiente a cada variable. La figura 4-11 muestra estos comandos.

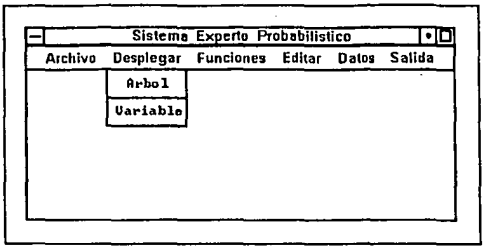


Figura 4-11. Comandos del menú *Desplegar*.

Cuando el usuario desee ver la información referente a la estructura de la red probabilística, está información se mostrará en una ventana similar a la mostrada en la figura 4-8. Esta ventana tendrá la etiqueta de hijos en lugar de directorios y la de padres por la de archivos. No contará con la etiqueta de filtro ni con el área de texto para filtro y en la parte donde se despliega el archivo mostrará el nombre del nodo.

Si el usuario deseara ver información acerca de alguna variable en especial la herramienta mostrará en primer lugar una ventana de selección para que el usuario elija cual es la variable de la que quiere obtener información. Dicha ventana se muestra en la figura 4-12.

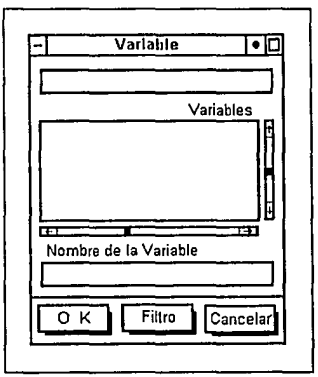


Figura 4-12. Ventana para seleccionar una variable.

Una vez que el usuario ya ha seleccionado una variable la herramienta presentará su información en una ventana como la de la figura 4-13.

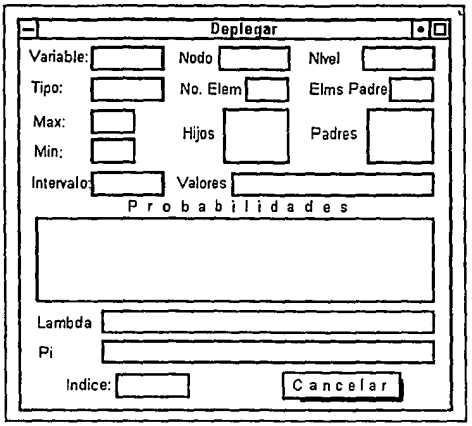


Figura 4-13. Ventana para desplegar la información de una *Variable*.

Una parte de esta información, es obtenida de los archivos de la red probabilística y la otra parte es calculada por la herramienta.

4.5. MENU *FUNCIONES*

Antes de llevar a cabo alguna de las funciones con las que cuenta la herramienta (estas funciones fueron comentadas en 3.3.5.) se le debe de indicar de donde tomará los valores con los que ejecutará alguna de las funciones. Como se vió en 3.3.4, esto puede ser del teclado o de un archivo externo. Si los valores son obtenidos de un archivo externo, la herramienta únicamente preguntará el nombre del archivo, en caso contrario mostrará las ventanas para lectura de valores que se presentan en esta sección. Los submenús con los que cuenta éste menú se pueden observar en la figura 4-14, y se describen a continuación.

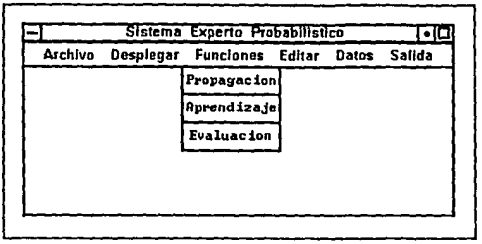


Figura 4-14. Submenú del menú *Funciones*.

4.5.1. Comando *Propagación*

La función de propagación debe de realizar los tipos de propagación que se muestran en la figura 4-15. Estos tipos ya fueron discutidos en la sección 3.2.5.

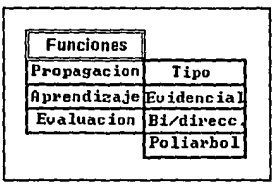


Figura 4-15. Tipos de propagación.

Cuando se lleve a cabo propagación evidencial la herramienta requerirá los valores de todos los nodos hojas con los cuales va a realizar la propagación, para ello desplegará una ventana como la que se muestra en la figura 4-16. En el área de texto que lleva como título "Nodo hoja:" la herramienta desplegará el nombre del nodo para el cual requiere el valor.

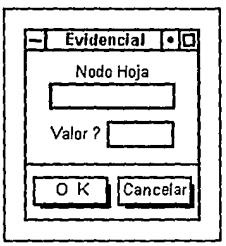


Figura 4-16. Ventana para el comando *Evidencial*.

Si se realiza propagación bidireccional o en poliárboles, la herramienta presentará la ventana que aparece en la figura 4-17. En estos casos el usuario podrá indicarle a que nodo le desea asignar un valor y el valor mismo. Recordar que en estos dos tipos de propagación el usuario podrá asignarle valores a todos los nodos que el desee sin importar si son nodos hojas o intermedios, a diferencia de propagación evidencial donde sólo es posible asignarle valores a los nodos hojas.

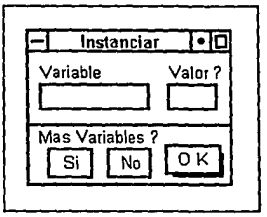


Figura 4-17. Ventana para realizar *propagación bidireccional y en poliárboles*.

4.5.2. Comando *Aprendizaje*

Cuando se realice la función de aprendizaje la herramienta mostrará una ventana como la de la figura 4-18. En el área de texto titulada "Variable" la herramienta desplegará el nombre de cada uno de los nodos en la red, desde la raíz hasta las hojas y el usuario entrará su valor en el área de texto asignada para ello.

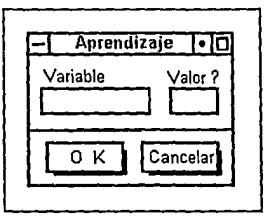


Figura 4-18. Ventana para la función de *Aprendizaje*.

4.5.3. Evaluación

Esta función realiza la evaluación de los resultados arrojados por la herramienta, para ello cuenta con los comandos que se muestran en la figura 4-19.

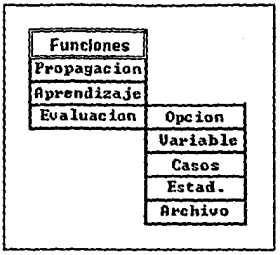


Figura 4-19. Comandos de la función *Evaluación*.

El comando *Variable* desplegará una ventana de selección en la cual el usuario seleccionará cual es la variable que desea evaluar, esto únicamente cuando se lleve a cabo propagación bidireccional y en poliárboles, ya que en propagación evidencial siempre se evalúa el nodo raíz. Dicha ventana será como la que aparece en la figura 4-12.

En el comando de casos la herramienta presentará una ventana como la que se exhibe en la figura 4-20. En esta ventana la herramienta desplegará la variable que va a evaluar y en el área de texto inferior el usuario deberá indicar su valor.

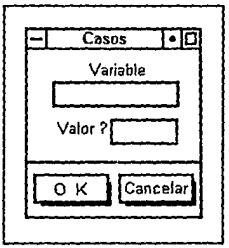


Figura 4-20. Ventana para el comando *Casos*.

Para desplegar los resultados obtenidos de la función de evaluación, la herramienta mostrará una ventana donde estará toda la información derivada de ella para la variable evaluada. Esta información se mostrará en una ventana como la de la figura 4-21.

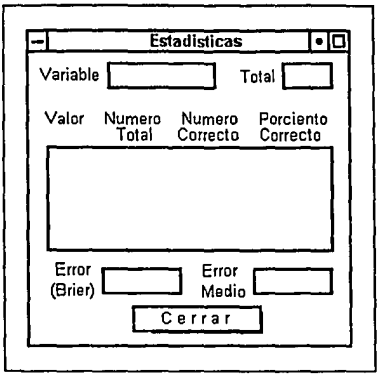


Figura 4-21. Ventana para el comando *Estadísticas*.

El comando archivo permitirá almacenar y cargar estadísticas. Para ello el usuario tendrá que determinar el nombre del archivo en el cual se almacenará o se recuperará la información. Esto se realizará por medio de dos ventanas, como las que se muestran en las figuras 4-8 y 4-10. La información que se almacenará en estos archivos será la que se muestra en la figura 4-21. El nombre de estos archivos llevará la extensión ".sts". Almacenar los datos permitirá recuperar las estadísticas obtenidas de eventos anteriores y actualizarlas con nuevos valores. Dado que este archivo tendrá un formato estándar será posible cargar este archivo para su uso en otras aplicaciones. La figura 4-22 muestra la estructura del archivo que almacenará los datos estadísticos.

Nombre de la variable		Total de casos	
Valor	Numero total	No. correcto	% Correcto
Error cuadratico		Error cuadratico promedio	

Figura 4-22. Formato del archivo de estadísticas.

4.6. MENU *EDITAR*

El menú editar permitirá modificar la estructura de la red probabilística por medio de dos comandos, los cuales se muestran en la figura 4-23.

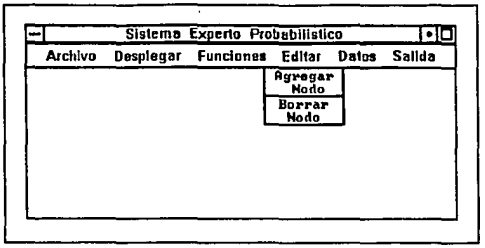


Figura 4-23. Comandos del menú *Editar*.

4.6.1. Comando *Agregar nodo*

Este comando permitirá adicionar elementos a la estructura de una red probabilística. Para ello el usuario debe de indicar el nombre del nodo y el nombre del padre del mismo, mediante la ventana mostrada en la figura 4-24.

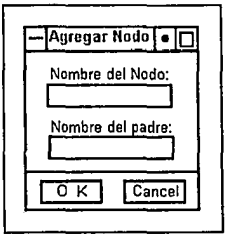


Figura 4-24. Ventana para el comando *Agregar nodo*.

Adicionalmente, el usuario podrá borrar nodos de la estructura de la red. Para ello tendrá que indicarle a la herramienta que nodo desea borrar. Para realizar esto, la herramienta desplegará una ventana de selección con todos los nodos que conforman la red probabilística y el usuario seleccionará un nodo de ahí. Si el nodo seleccionado tiene hijos sus hijos también se borrarán. La figura 4-25 ilustra esta ventana.

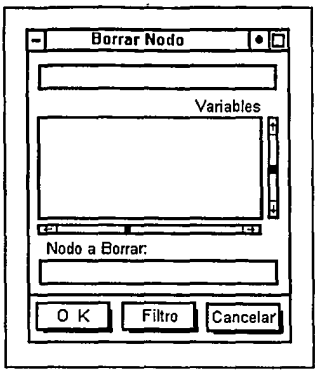


Figura 4-25. Ventana para el comando *Borrar Nodo*.

4.7. MENU *DATOS*

Como se mencionó en 4.5., se le debe de indicar a la herramienta de donde va tomar los datos con los cuales ejecutará alguna de sus funciones. La herramienta puede tomar los datos del teclado o de un archivo externo. Estas dos opciones se pueden apreciar en la figura 4-26.

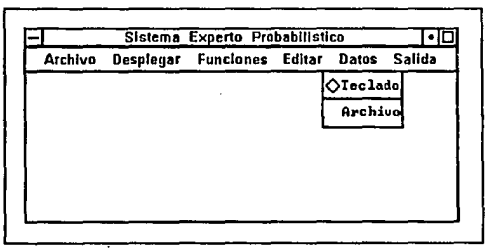


Figura 4-26. Comandos del menú *Datos*.

Si los valores son obtenidos de un archivo externo la herramienta únicamente preguntará el nombre del archivo del cual tomará los datos, para ello desplegará una ventana como la que se muestra en la figura 4-27.

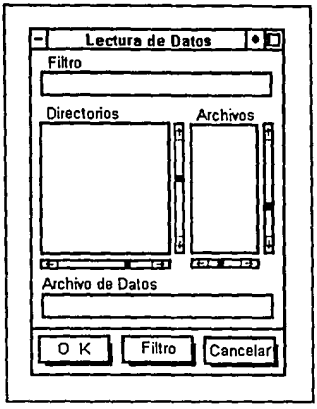


Figura 4-27. Ventana para seleccionar un archivo del cual se cargarán los datos.

Puesto que los valores de los nodos se obtendrán de archivos externos, deben ser almacenados con una estructura determinada. Dicha estructura se ilustra en la figura 4-28a. El sistema también debe ser capaz de almacenar los resultados a los que llega, es decir, la probabilidad *a posteriori* para las variables de interés. La figura 4-28b muestra la estructura de este archivo. Estos archivos serán utilizados cuando el sistema interactúe con un sistema de control supervisorio.

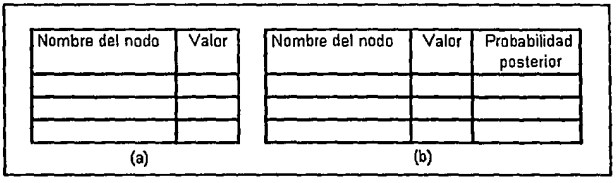


Figura 4-28. Formato de los archivos para cargado de archivos externos (enlace a otros sistemas). En (a) se muestra el formato para valores de variables (entrada), y en (b) para los resultados.

Una vez que los valores ya han sido cargados, será posible llevar a cabo las funciones propias del sistema. La figura 4-29 muestra la relación de los archivos mencionados.

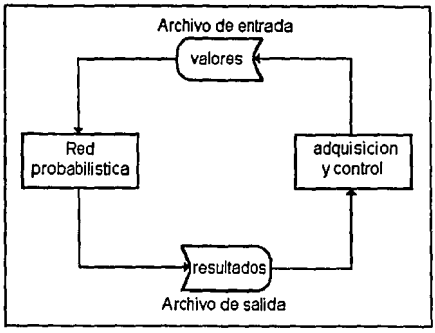


Figura 4-29. Interacción con otros sistemas a través de archivos.

4.8. MENU *SALIDA*

El menú salida contará con dos comandos. Uno de ellos para salir de la herramienta y otro para re-iniciar. La figura 4-30 muestra estos comandos.

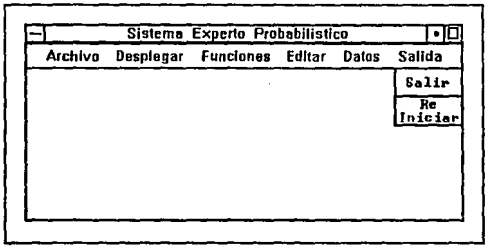


Figura 4-30. Comandos del menú *Salida*.

Es importante recordar que las ventanas que presentará la herramienta para los dos comandos mencionados serán únicamente para confirmar o para cancelar la operación. El comando *Salir* regresará al usuario al sistema. Para ello la herramienta mostrará una ventana como la que se muestra en la figura 4-31.

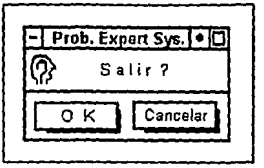


Figura 4-31. Ventana para el comando *Salir*.

El comando *Re-iniciar* borrará la base de conocimientos que este cargada en la herramienta y limpiará todas las variables, de tal forma que el usuario pueda comenzar con otra base de conocimientos. Para ello la herramienta presentará una ventana como la que se muestra en la figura 4-32.

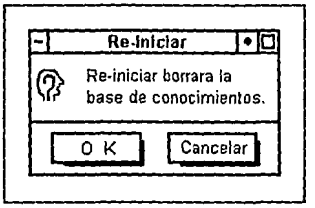


Figura 4-32. Ventana para el comando *Re-iniciar*.

4.9. MENSAJES DE ERROR

Como se menciono en 3.3.7, la herramienta debe verificar la validez de la información entrada por el usuario y si esta es errónea deberá desplegar un mensaje de error. La herramienta presentará dicho error en una ventana de mensaje con un botón que el usuario tendrá que oprimir después de leer la información. Un bosquejo de está ventana, con un posible mensaje de error se presenta en la figura 4-33.

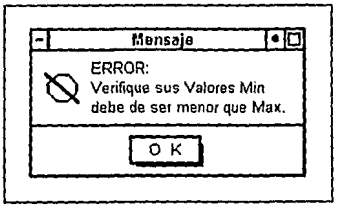


Figura 4-33. Ventana para desplegar Mensajes.

4.10. CONCLUSIONES

Al principio de éste capítulo recalcamos la importancia de la comunicación que debe de existir entre el usuario y la herramienta, para lograr con ello un buen uso de la misma. Por lo anterior decidimos ir desglosando cada uno de los aspectos primordiales con los que deseamos que cuente la herramienta en menús y estos a su vez en comandos, tratando que cada comando cuente con una función específica. Al especificar cada uno de los comandos a su vez mostramos una ventana, tratando de que en ella se refleje la información requerida por la herramienta o se despliegue la información acerca de dicha función. Así hemos definido seis funciones principales: *Archivo, Desplegar, Funciones, Editar, Datos y Salida.* La interfaz con el usuario ha sido desarrollada bajo X-Windows/Unix utilizando la herramienta para diseño de interfases "Interface Architect" (Ver Apéndice).

Ahora que ya hemos especificado claramente los comandos con los que contará la herramienta pasaremos al diseño de la misma. Dicho diseño se basa en las especificaciones presentadas en los capítulos 3 y 4, utilizando una metodología orientada a objetos. El diseño de la herramienta se detalla en el siguiente capítulo.

CAPITULO 5

DISEÑO DE LA HERRAMIENTA

5.1. INTRODUCCION

Como hemos mencionado en capítulos anteriores, la herramienta ha sido desarrollada utilizando una metodología orientada a objetos. La metodología orientada a objetos es una nueva forma de analizar y resolver problemas más grandes y complejos, dividiendo a los mismos en subgrupos de partes relacionadas al problema. Estos subgrupos son traducidos a unidades llamadas objetos. En esencia la programación orientada a objetos implica la creación de objetos, los cuales combinan y encapsulan tanto los datos como el código que opera sobre ellos.

En este capítulo llevaremos a cabo el diseño de la herramienta. Para ello nos basaremos en el método de Booch [Booch 91] para desarrollo de software orientado a objetos, llevando a cabo la implantación en el lenguaje de programación C++ [Stroustrup 86]. Hemos elegido desarrollar la herramienta utilizando una metodología orientada a objetos, ya que esta tiene una correspondencia con la forma en que se lleva a cabo el esquema de propagación, donde los nodos se asemejan a los objetos y la comunicación entre ellos es por medio de mensajes.

En la siguiente sección definiremos algunos conceptos importantes dentro de este marco de desarrollo, y posteriormente, comenzaremos con el diseño. Conforme avancemos en el diseño, describiremos brevemente los conceptos utilizados de la metodología y los aspectos correspondientes a estos concepto dentro de la herramienta. Es importante mencionar que en este capítulo sólo describiremos aquellos puntos de la metodología que son utilizados para realizar el diseño, sin pretender ser exhaustivos. Para un estudio completo de esta metodología referirse a Booch [91].

5.2. DEFINICIONES BASICAS

Objeto. Un *objeto* es una entidad lógica que contiene *datos*, y el código que manipula esos datos denominado *procedimiento*. Dentro de un objeto, parte del código y/o de los datos puede ser privado para ese objeto, y no será accesible directamente para nada de lo que haya fuera de ese objeto. De esta manera, un objeto proporciona un nivel significativo de protección contra agentes externos que pudieran modificar o utilizar incorrectamente las partes privadas del objeto [Schild 91].

Clase. Una *clase* es una descripción para producir objetos de un mismo tipo. Una clase esta formada por los *procedimientos* y los *datos* que definen las características comunes a todos los objetos de esa clase. La clave de la programación orientada a objetos está en abstraer los

procedimientos y datos comunes a un conjunto de objetos y almacenarlos en una clase [Ceballos 93].

Herencia. La *herencia* es el proceso mediante el cual un objeto puede adquirir las propiedades de otro objeto. Esto es importante ya que sirve como base para el concepto de clasificación. Si no se utiliza herencia, cada objeto tendría que definir explícitamente todas sus características. Sin embargo, cuando se utiliza herencia, los objetos sólo necesitan definir aquellas características que lo hacen único dentro de su clase. El mecanismo de herencia es el que hace posible que un objeto sea un caso concreto de uno más general [Schild 91].

5.3. DISEÑO DE LA HERRAMIENTA UTILIZANDO EL METODO DE BOOCH

En esta sección aplicaremos el método de Booch [91] para llevar a cabo el diseño de la herramienta, partiendo de las consideraciones mencionadas en los capítulos 3 y 4. Entre los puntos que debemos considerar para llevar a cabo el diseño están los siguientes:

- ¿Que clases existen y como están relacionadas?
- ¿Como están estructurados los objetos y como colaboran entre ellos?
- ¿Donde será definida cada clase y cada objeto?

Para contestar estas preguntas nos apoyaremos en los siguientes diagramas:

- *Diagramas de clase.* Un diagrama de clase es aquel que muestra las clases utilizadas y sus relaciones.
- *Diagramas de objetos.* Hay dos tipos de diagramas de objetos: diagramas de instancia-objetos y diagramas de escenario. Un diagrama de instancia-objeto tiene como propósito ilustrar la estructura física de un objeto. Un diagrama de escenario ilustra el comportamiento funcional de un conjunto de objetos.
- *Diagramas de módulo.* Es un tipo de diagrama que muestra los subsistemas presentes y sus interacciones.

Para cada elemento gráfico de la notación, hay una especificación que proporciona información detallada de los aspectos representados en el diagrama. Cada diagrama representa una vista del modelo del sistema en desarrollo, y deben de ser consistentes entre ellos.

5.3.1. Diagrama de clases

Un *diagrama de clase* es usado para mostrar la existencia de las clases y sus relaciones en el sistema. Representa una vista de la estructura de clases dentro del sistema. Un diagrama de clase puede ser adornado con ciertos elementos de su especificación colocados dentro de un compartimiento. Adicionalmente, para cada clase en el diagrama de clases existe una *especificación de la clase* la cual se utiliza para declarar todos los miembros de la clase.

Un *atributo* denota una propiedad de los objetos de una clase. Este puede tener un nombre, una clase o ambos. Un *método* puede tener solamente nombre y se distingue de un atributo al agregar "()".

Las relaciones entre clases y objetos son usadas para definir interacciones entre ellos. Existen varios tipos de relaciones de clases, la más elemental es la relación de *asociación*, la cual es denotada por una línea simple. Esta relación especifica que hay alguna relación entre dos clases sin llegar a conocer su naturaleza exacta. Un tipo de relación más compleja es la relación *tiene*. Para representar la relación *tiene* se utiliza un círculo relleno unido a una línea simple donde el círculo representa la asociación. Considerando que tenemos dos clases llamadas **X** y **Y**, dada una instancia de la clase **X**, la relación *tiene* denota que es posible que la clase **X** tenga varias instancias de la clase **Y**. El número de instancias es especificado por la cardinalidad de la relación. La *cardinalidad* se especifica en las puntas de una relación *tiene* entre dos clases, denotando el número de objetos destino que pueden ser alcanzados por un objeto fuente dado. Es posible especificar un número exacto o un rango de números.

Adicionalmente una relación de clases puede incluir ciertos símbolos para indicar el tipo de acceso. Cuando no se especifica algún acceso especial se presupone un *acceso público*, con una línea pequeña interceptando la línea que muestra la relación se representa un *acceso protegido* y con dos líneas paralelas un *acceso privado*.

Otro aspecto importante son las *utilerías de clase*, estas permiten representar una colección de subprogramas libres. Un subprograma libre es una función que no esta directamente asociada con una clase.

Utilizando la notación mencionada y analizando los requerimientos existentes en nuestra aplicación llegamos al diagrama de clases mostrado en la figura 5-1. En dicha figura podemos observar que nuestra aplicación cuenta con dos clases (cada una se representa como una burbuja amorfa punteada) llamadas: *ptree* y *pnode* y con una utilería de clase denominada *interfaz* (esta se representa como una clase pero además lleva una sombra).

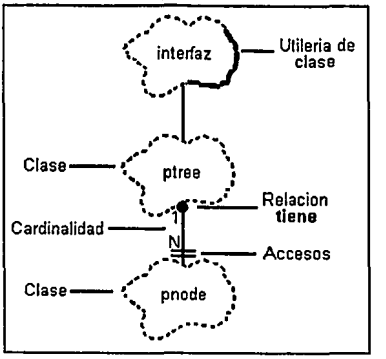


Figura 5-1. Diagrama de clases.

Entre las dos clases mencionadas existe una relación *tiene*, donde *ptree* tiene varias instancias de la clase *pnode*, es decir la dirección de la asociación va de la clase *ptree* a la clase *pnode*, esto es indicado por el circulo relleno en el extremo de la clase *ptree*. La cardinalidad determina que tenemos un objeto de la clase *ptree* y hasta **N** objetos de la clase *pnode*. Para la clase *ptree* tenemos un acceso público, por el contrario *pnode* tiene un acceso privado representado por las dos líneas paralelas (figura 5-1).

Para cada clase en el diagrama de clases debe existir una *especificación de clase*, la cual sirve para capturar las características de la clase. A continuación definimos algunos conceptos utilizados para la especificación tanto de clases como de objetos.

Acceso. Es el mecanismo para controlar el acceso a la estructura o comportamiento de una clase. Un artículo (ya sea clase o un objeto) público es accesible para todas las clases. Un artículo privado es accesible sólo para su clase y para amigos de su clase.

Superclase. Es una clase de la cual otras clases heredan propiedades.

Espacio. Es una expresión que denota la cantidad de almacenamiento consumida por cada instancia de la clase.

Persistencia. Es la propiedad de un objeto por la cual su existencia trasciende en tiempo y/o en espacio. Es decir, indica el tiempo de vida de una instancia de la clase, esta puede ser transitoria si su tiempo de vida corresponde al tiempo de vida del agente que lo creó y persistente si su tiempo de vida trasciende al agente que lo creó.

Concurrencia. Es la propiedad que distingue a un objeto activo de uno que no lo es.

Para un estudio más profundo de esta metodología referirse a Booch[91]. La especificación de clase para las clases del diagrama anterior se muestran a continuación.

Especificación de la clase (ptree).

Nombre: ptree.

Responsabilidades: Lleva a cabo operaciones relacionadas con la red probabilística en general sin llegar a conocer la información perteneciente a cada nodo.

Acceso: Público.

Cardinalidad: 1.

Superclase: Ninguna.

Miembros:

Atributos:

nodes (entero). Número de nodos en el árbol.

levs (entero). Número de niveles en el árbol.

node[Mnodes] (phead). Arreglo de estructuras de nodos probabilísticos, limitada a un número máximo de "Mnodes".

sts (stats). Estructura para calcular y almacenar las estadísticas, al realizar la función de evaluación.

Métodos:

pt_built. Definición de un árbol probabilístico incluyendo las variables. Esto puede ser del teclado o cargando un archivo que ya tenga una estructura almacenada.

pt_show. Despliega el estado de las variable en un árbol probabilístico.

pt_showplus. Despliega información adicional de una variable.

pt_nid. Obtiene el índice de un nodo particular en el árbol.

pt_store. Almacena la estructura de un árbol probabilístico en un archivo.

pt_nset. Obtiene el índice del valor asignado.

pt_nshow. Despliega información de una variable.

pt_bupro. Propagación de probabilidades en forma evidencial. Requiere los valores de los nodos hoja.

pt_learn. Actualiza las probabilidades en el árbol de acuerdo a una observación.

pt_save. Guarda las matrices de probabilidad después de haber realizado la función de aprendizaje.

pt_stats. Realiza estadísticas para llevar a cabo una evaluación de acuerdo a una observación dada.

pt_savestat. Guarda las estadísticas en un archivo.

pt_loadstat. Obtiene las estadísticas almacenadas en un archivo para continuar trabajando con las mismas.

Espacio:

Mrange=20.
Mnodes=10.
stats≈250.
phead≈34+pnode.
objeto=4+Mnodes(34+pnode)+250≈
≈4+10(34+1250)+250≈13100 bytes.

Persistencia: Persistente.

Concurrencia: Secuencial.

Especificación de la clase (pnode).

Nombre: pnode.

Responsabilidades: Lleva a cabo operaciones relacionadas con cada uno de los nodos que conforman la red probabilística.

Acceso: Privado.

Cardinalidad: 1..N

Superclase: ptree.

Miembros:

Atributos:

name[15] (char). Nombre de la variable.

type (ntype). Tipo de variable: binaria, continua o discreta.

num (int). Número de elementos en una variable.

max (int). Valor máximo en un rango (para variables continuas).

min (int). Valor mínimo en un rango (para variables continuas).

nump (int). Número de elementos del padre de una variable.

ssize (double). Longitud de cada intervalo en un rango.

value[Mrange] (int). Valores para cada elemento en una variable (continuas y discretas).

prior[Mrange] (int). Vector de probabilidades *a priori* para los elementos del nodo raíz (número de casos-entero).

tprior (int). Sumatoria de las probabilidades *a priori.*

condp[Mrange][Mrange] (int). Arreglo de probabilidades condicionales para los nodos no raíz (número de casos-entero).

tcondp[Mrange] (int). Sumatoria de las probabilidades condicionales por columna.

la[Mrange] (double). Valores actuales del vector λ.

laprev (int). Indice del valor asignado a una variable.

pi[Mrange] (double). Valores actuales del vector π.

postp[Mrange] (double). Probabilidades *a posteriori.*

parent (* pnode). Apuntador al nodo padre.

sons[Msons] (* pnode). Arreglo de apuntadores a los hijos.

Métodos:

upson. Pone la dirección del padre en un nodo hijo.

init. Inicializa variables ya sea con información entrada del teclado o de un archivo.

printplus. Obtiene información adicional acerca de una variable probabilística para ser desplegada en pantalla.

set. Asigna un valor a un elemento de una variable específica.

obt_index. Obtiene el índice para un valor a evaluar en una variable.

vali. Regresa el índice de un valor conocido el cual fue asignado a una variable.

stodata. Almacena los datos correspondientes con una variable probabilística en un archivo.

pro_post. Calcula las probabilidades *a posteriori* para el nodo raíz.

pro_lam. Calcula mensajes λ para un nodo padre.

pro_reset. Inicializa vector λ.

pro_update. Actualiza el vector λ con mensajes λ de sus hijos.

pro_all. Obtiene el vector λ final con mensajes λ de todos sus hijos.

ln_update. Actualiza las probabilidades *a priori*/condicionales después de una observación, asumiendo que a cada nodo y a su padre se les ha asignado un valor.

ln_save. Guarda las nuevas probabilidades después de haber realizado aprendizaje, en archivos.

Espacio:

Mrange= 20.
Mnodes= 10.
Msons= 10.
objeto≈
15+2+8+4+40+40+2+800+40+80+2+80+80+4+40≈
≈1250 bytes

Persistencia: Transitoria.

Concurrencia: Secuencial.

Adicionalmente en la figura 5-1 podemos observar una utilería de clase denominada *interfaz* la cual tiene una relación indirecta con la clase *ptree*. En nuestro caso dicha utilería está representada por la interfaz al usuario, es decir el menú principal y todos sus comandos. La tabla 5-1 muestra los menús, comandos y los nombres de archivos de las ventanas utilizadas en dicha interfaz.

Menú	Comando	Nombre del Archivo(s)
Archivo	Crear.	Crear. dialogShell10. dialogShell11. dialogShell12. dialogShell13. dialogShell14. dialogShell15.
	Almacenar.	questionDialog1.
	Almacenar Como	fileSelectionDialog13.
	Cargar.	fileSelectionDialog8.
	Borrar.	fileSelectionDialog1. questionDialog2.
Desplegar	Arbol.	----------
	Variable.	fileSelectionDialog7. dialogShell1.
Funciones	Propagación.	*Evidencial* fileSelectionDialog2. (A) dialogShell4. (T) *Bidireccional* fileSelectionDialog9. (A) dialogShell2. (T) *Poli-árboles* fileSelectionDialog3. (A) dialogShell9. (T)
	- - - - - - -	- - - - - - - - - - - - - -
	Aprendizaje.	fileSelectionDialog10. (A) dialogShell3. (T)

Tabla 5-1. Menús, comandos y nombres de archivos utilizados en la interfaz al usuario.

Menú	Comando	Nombre del Archivo(s).
Funciones	Evaluación.	*Variable* fileSelectionDialog11.(A) *Casos.* fileSelectionDialog4. (A) dialogShell8. (T) *Estadísticas.* dialogShell5 *Cargar estadísticas.* fileSelectionDialog5 *Guardar Estadísticas.* fileSelectionDialog12.
Editar	Agregar Nodo. Borrar Nodo.	dialogShell6. fileSelection6. questionDialog3.
Datos	Archivo Ext. Teclado.	Ninguno. Ninguno.
Salida	Salir. Re-iniciar.	exitDialog. dialogShell7.

Tabla 5-1 (cont). Menús, comandos y nombres de archivos utilizados en la interfaz al usuario.

En la tercer columna podemos observar información adicional. Cuando los nombres de archivos en un mismo comando no se encuentran alineados, significa que el archivo de más a la izquierda, llama a los archivo que se encuentran recorridos con respecto a éste. Los nombres escritos en cursiva denotan comandos y los archivos que crean las ventanas son los que se encuentran debajo de él (por ejemplo, en el menú evaluación *casos* se refiere a este comando y dialogShell8 al archivo que genera la ventana para este comando). Finalmente las letras **T** y **A** se refieren a la forma en que la herramienta leerá los valores que les asignará a los nodos de la red. Si los valores se obtienen de archivos externos, se ocupa la ventana creada con el archivo que tenga la letra **A**, en caso contrario el que tenga la letra **T**.

5.3.2. Diagrama de objetos

Un *diagrama de objeto* es usado para mostrar la existencia de objetos y su relación en el modelo del sistema. Un objeto es una instancia de una clase. Un objeto lo podemos representar por medio de una burbuja amorfa, sólo que para diferenciarlo de una clase la burbuja es continua.

En adición a un diagrama de objetos existe la especificación de los objetos. En la figura 5-2. se muestra el diagrama de objetos involucrados en el sistema. En nuestra aplicación hemos definido un objeto de clase *ptree* y varios de clase *pnode* (los objetos de clase *pnode* están representados en la figura, como si un objeto estuviera detrás de otro).

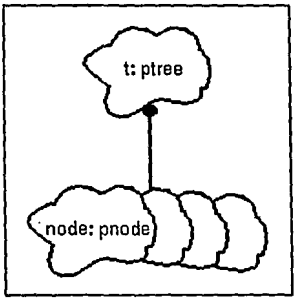


Figura 5-2. Diagrama de objetos.

Especificación de los objetos.

Nombre: t.

Responsabilidades: Red probabilística creada a partir de información entrada del teclado o al cargarla de un archivo ya existente, contiene "Mnodes" nodos o variables probabilísticas. En particular este objeto contiene todas las operaciones realizadas sobre la estructura de la red probabilística.

Clase: ptree.

Calificación: El objeto representa a la clase.

Persistencia: Persistente.

****

Nombre: node[0],...,node[Mnodes]

Responsabilidades: Este objeto representa todos los nodos probabilísticos pertenecientes a la red "t". Cada uno guarda una variable probabilística. Este objeto realiza todas las operaciones concernientes con los nodos de la red.

Clase: pnode.

Calificación: Representa una instancia de la clase.

Persistencia: Transitorio.

5.3.3. Diagrama de transición de estados

El funcionamiento dinámico asociado con ciertas clases se puede observar por medio de un diagrama de transición de estados. Estos diagramas muestran los estados obtenidos de las clases dadas, los eventos que causan una transición de un estado a otro y las acciones que resultan de un cambio de estado. Un estado se representa con un rectángulo con las puntas redondeadas con el nombre del estado. El comienzo de un diagrama de estados se especifica con un círculo vacío con una flecha y el final del diagrama, con un círculo grueso y una flecha. Asimismo, el diagrama de transición de estados puede ser apoyado por medio de la especificación de estados. Las acciones en un diagrama de estados pueden ser asociadas con un estado y expresadas en la especificación de estados. En la figura 5-3 se muestra el diagrama de transición de estados de la herramienta, y en seguida la especificación de cada uno.

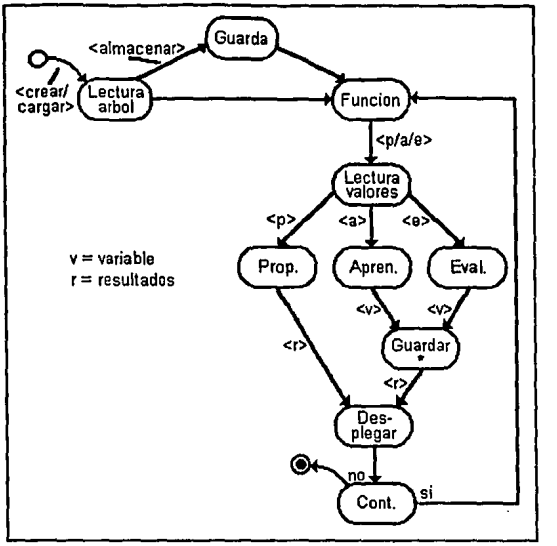


Figura 5-3. Diagrama de estados.

Especificación de estados.

Nombre	Acciones.
1) Lectura árbol.	Lectura de la estructura y de las variables que conforman la red. Esto puede ser del teclado o de algún archivo. Después de obtener la red, se puede guardar en un archivo, ó entrar de lleno a las funciones de la herramienta.
2) Guarda.	Almacena la estructura de la red así como la información de las variables que la conforman en un archivo.
3) Función.	Se selecciona alguna de las funciones del sistema para su ejecución posterior (p/propagación, a/aprendizaje, e/evaluación).
4) Lectura de valores.	Entrada de valores para los nodos de la red, conforme a la función seleccionada.
5) Propagación.	Lleva a cabo la propagación, actualizando la información de los nodos no asignados.

6) Aprendizaje.	Actualización de las probabilidades *a priori* y condicionales conforme a los valores proporcionados en el punto 4.
7) Evaluación.	Comparación de las probabilidades *a posteriori* de algún nodo de la red con un valor determinado y cálculo de estadísticas.
8) Guardar.	Guarda la información resultante de aprendizaje y evaluación en archivos.
9) Desplegar.	Desplegar el estado de los nodos de la red probabilística, así como estadísticas derivadas de la función de evaluación.
10) Continuar.	Se puede realizar otra función de la herramienta o salir de la misma.

5.3.4. Diagrama de módulos

Un diagrama de módulos es usado para mostrar la distribución de clases y objetos en módulos en el modelo físico de un sistema. Un diagrama de módulos simple, representa todo o parte de la arquitectura modular de un sistema. La única relación que se puede tener entre dos módulos es una dependencia de compilación representada por una línea dirigida. En una forma semejante a los demás diagramas mencionados, este diagrama puede contar con una especificación de módulos. La figura 5-4. muestra los módulos involucrados en nuestro diseño (adicionalmente en la figura se especifica la utilización de cada uno de los símbolos).

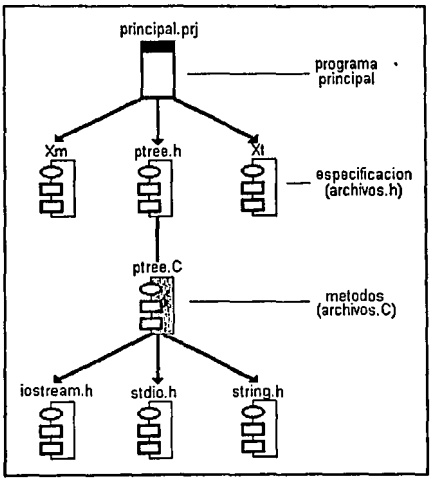


Figura 5-4. Diagrama de módulos de acuerdo al método de Booch.

Especificación de módulos.

Nombre: principal.c

Responsabilidades: Este módulo lleva a cabo el control de las operaciones a realizar de acuerdo a los comandos seleccionados por el usuario. Contiene el menú principal así como todas las ventanas que serán desplegadas por este.

Acceso: público.

Declaraciones: Este módulo contiene todos los archivos de la interfaz que fueron mostrados en la tabla 5-1.

****

Nombre: ptree.h

Responsabilidades: Contiene algunas funciones intermedias para realizar la liga entre el módulo que contiene la interfaz al usuario y el módulo que contiene la implementación de redes probabilísticas.

Acceso: público.

Declaraciones: Las funciones encontradas en este módulo no llevan a cabo ninguna operación especial, únicamente ligan la interfaz con el programa de redes probabilísticas.

****

Nombre: ptree.C

Responsabilidades: Aplicación de árboles probabilísticos en base a objetos. Incluye las funciones para realizar propagación, aprendizaje paramétrico y evaluación de resultados, así como el almacenamiento de esta información en archivos.

Acceso: público.

Declaraciones:

Clases
- pnode.
- ptree.

Objetos
- t.
- node[0],...,node[Mnode].

Tipos
- ntype - tipos de variables (enum).
- htype - tipos de nodos (enum).
- mtype - tipos de entrada (enum).
- pmess - estructura para mensajes.
- stats - estructura para estadísticas.
- estructura para nodos en un árbol.

Los demás módulos son librerías de *C++, X Windows y Motif.*

5.4. CONCLUSIONES

En este capítulo realizamos el diseño de la herramienta utilizando una metodología de desarrollo de software orientada a objetos. Aunque nos enfocamos más a la parte del diseño relacionada con redes probabilísticas y sus funciones adicionales, es importante señalar que el paquete *Interface Architect* genera las ventanas utilizando también programación orientada a objetos. Hemos visto algunos conceptos de importancia de la programación orientada a objetos, así como la metodología de Booch para el diseño de software orientado a objetos. Especificamos los puntos de esta metodología relacionados con el diseño de nuestra aplicación. Al realizar el diseño, observamos como se facilita la resolución del problema al dividirlo en clases y objetos. Cada uno tiene características determinadas y realiza funciones específicas, existiendo una interrelación entre estas.

Después de haber realizado el análisis, especificación y diseño de la herramienta pasaremos a la elaboración de la misma. Para el desarrollo de la interfaz al usuario utilizamos el paquete *Interface Architect* y la aplicación de la técnica de redes probabilísticas la efectuamos en el lenguaje de programación *C++*. En el siguiente capítulo veremos algunos ejemplos de aplicación de la herramienta.

CAPITULO 6

PRUEBAS Y EJEMPLOS DE APLICACION DE LA HERRAMIENTA

6.1. INTRODUCCION

Como se mencionó en capítulos anteriores, este trabajo surge de la necesidad de contar con sistemas expertos para su aplicación en el monitoreo y control de redes eléctricas. Sin embargo esta herramienta y, en general, la técnica de redes probabilísticas, puede ser aplicado en otros dominios en los cuales exista incertidumbre en el conocimiento, por lo que tiene una aplicación múltiple. Unicamente se requiere la red probabilística correspondiente a cada dominio de aplicación.

En el presente capítulo veremos algunas aplicaciones de la herramienta. En primer lugar presentamos un ejemplo ilustrativo aplicado a medicina. En este ejemplo realizaremos propagación, aprendizaje y evaluación con los nodos que conforman la red probabilística. Para observar la variación de las probabilidades *a posteriori*, realizaremos la función de propagación asignando diferentes valores a los nodos hojas. Primero asignaremos valores a los nodos suponiendo que el enfermo presenta todos los síntomas, obteniendo de este caso las probabilidades *a posteriori*. Después asignaremos a los nodos hojas valores contrarios a los anteriores, obteniendo las probabilidades *a posteriori* para los nuevos valores. Realizaremos la evaluación de los resultados obtenidos para el caso anterior y posteriormente la evaluación de las probabilidades *a posteriori* iniciales. Asimismo llevaremos a cabo aprendizaje paramétrico con las variables de la red.

En un segundo ejemplo mostraremos una aplicación de la herramienta para el monitoreo y control de redes eléctricas. En este ejemplo definiremos la red probabilística perteneciente a este dominio con los nodos que la conforman, y desglosaremos brevemente una parte de la red. Aquí no asignaremos probabilidades *a priori* o condicionales a los nodos que conforman la red, esto se debe a que para una aplicación así, la herramienta tomaría las probabilidades del experto y de datos estadísticos ó unicamente de datos estadísticos, las cuales se irán actualizando con cada observación. De esta forma las probabilidades son mejoradas por la herramienta conforme se tiene más información (función de aprendizaje).

6.2. APLICACION DE LA HERRAMIENTA EN MEDICINA

En ésta sección veremos un ejemplo sencillo de aplicación de la herramienta en el área de medicina para el diagnóstico de *cancer metastático*. Presentamos en primer lugar una red probabilística pequeña correspondiente a este dominio de aplicación[3] y unas tablas que muestran el número de casos para los diversos valores. Posteriormente mostramos las probabilidades *a priori* y condicionales para los diversos valores de los nodos según corresponda. Finalmente llevaremos a cabo la función de propagación para el caso donde el paciente tiene todos los síntomas y el caso contrario, así como la evaluación de resultados y aprendizaje paramétrico.

6.2.1. Red probabilística y probabilidad asociada

La figura 6-1 muestra la red probabilística perteneciente a esta aplicación y los números de casos correspondientes.

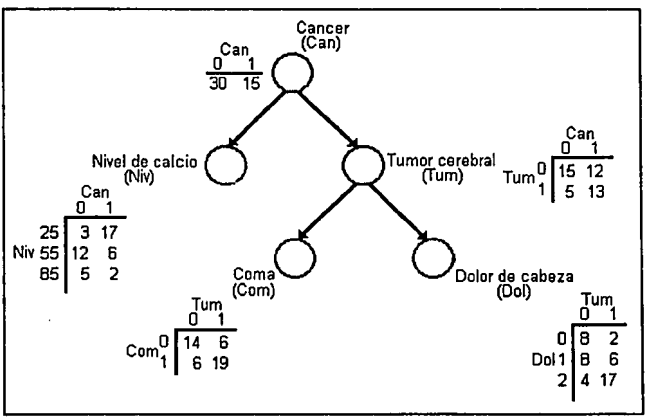


Figura 6-1. Red probabilística y tablas con los número de casos.

A continuación presentamos las probabilidades *a priori* y condicionales pertenecientes a la red de la figura 6-1.

Probabilidades *a priori* para **cancer** (nodo raíz):

$P(\mathbf{Cancer}_0) = 0.666667$ $P(\mathbf{Cancer}_1) = 0.333333$

[3] Los ejemplos mostrados en este capítulo son ilustrativos por lo que las redes probabilísticas definidas son muy sencillas. En una aplicación real la estructura de las redes sería más compleja.

Probabilidades condicionales para **nivel de calcio:**

$P(Niv_0|Can_0) = 0.15$ $P(Niv_0|Can_1) = 0.68$

$P(Niv_1|Can_0) = 0.6$ $P(Niv_1|Can_1) = 0.24$

$P(Niv_2|Can_0) = 0.25$ $P(Niv_2|Can_1) = 0.08$

Probabilidades condicionales para **tumor cerebral:**

$P(Tum_0|Can_0) = 0.75$ $P(Tum_0|Can_1) = 0.48$

$P(Tum_1|Can_0) = 0.25$ $P(Tum_1|Can_1) = 0.52$

Probabilidades condicionales para **coma:**

$P(Com_0|Tum_0) = 0.7$ $P(Com_0|Tum_1) = 0.24$

$P(Com_1|Tum_0) = 0.3$ $P(Com_1|Tum_1) = 0.76$

Probabilidades condicionales para **dolor de cabeza:**

$P(Dol_0|Tum_0) = 0.4$ $P(Dol_0|Tum_1) = 0.08$

$P(Dol_1|Tum_0) = 0.4$ $P(Dol_1|Tum_1) = 0.24$

$P(Dol_2|Tum_0) = 0.2$ $P(Dol_2|Tum_1) = 0.68$

6.2.2. Propagación

Ahora vamos a realizar la propagación de probabilidades. Para ello mostraremos las operaciones realizadas durante el cálculo de las probabilidades *a posteriori*, así como los resultados obtenidos, los cuales han sido comprobados con la herramienta.

Una vez obtenida la red probabilística, le asignaremos a **Dol** (dolor de cabeza) el valor de 2, así tenemos:

$$\lambda(Dol) = (0,0,1)$$

Calculando el mensaje λ que **Dol** envía a su padre **Tum** (tumor cerebral) de 2.23, tenemos:

$$\lambda_{Dol}(Tum_0)=P(Dol_0|Tum_0)\lambda(Dol_0)+P(Dol_1|Tum_0)\lambda(Dol_1)+P(Dol_2|Tum_0)\lambda(Dol_2)=$$
$$=0.4(0)+0.4(0)+0.2(1)=0.2$$
$$\lambda_{Dol}(Tum_1)=P(Dol_0|Tum_1)\lambda(Dol_0)+P(Dol_1|Tum_1)\lambda(Dol_1)+P(Dol_2|Tum_1)\lambda(Dol_2)=$$
$$=0.08(0)+0.24(0)+0.68(1)=0.68$$

Por lo que $\lambda_{Dol}(\mathbf{Tum})=(0.2, 0.68)$.

Asignándole a **Com** (Coma) el valor de 1 tenemos:

$$\lambda(\mathbf{Com})=(0,1).$$

De 2.23 calculamos el mensaje que este envía a su padre **Tum**:

$$\lambda_{Com}(Tum_0)=P(Com_0|Tum_0)\lambda(Com_0)+P(Com_1|Tum_0)\lambda(Com_1)$$
$$=0.7(0)+0.3(1)=0.3$$
$$\lambda_{Com}(Tum_1)=P(Com_0|Tum_1)\lambda(Com_0)+P(Com_1|Tum_1)\lambda(Com_1)$$
$$=0.24(0)+0.76(1)=0.76$$

Así $\lambda_{Com}(\mathbf{Tum})=(0.3, 0.76)$.

Calculando el valor λ de **Tum** (tumor cerebral) con 2.17 tenemos:

$$\lambda(Tum_0)=\lambda_{Dol}(Tum_0)\lambda_{Com}(Tum_0)=0.2(0.3)=0.06$$
$$\lambda(Tum_1)=\lambda_{Dol}(Tum_1)\lambda_{Com}(Tum_1)=0.68(0.76)=0.5168$$

Por lo que $\lambda(\mathbf{Tum})=(0.06, 0.5168)$

Suponiendo que le asignamos a **Niv** (nivel de calcio) un valor de 25 (consideramos que este sería el nivel de calcio que tendría un enfermo de cancer metastático) tenemos:

$$\lambda(\mathbf{Niv})=(1,0,0).$$

El mensaje λ que **Niv** envía a **Can**, de 2.23, será:

$$\lambda_{Niv}(Can_0)=P(Niv_0|Can_0)\lambda(Niv_0)+P(Niv_1|Can_0)\lambda(Niv_1)+P(Niv_2|Can_0)\lambda(Niv_2)=$$
$$=0.15(1)+0.6(0)+0.25(0)=0.15$$
$$\lambda_{Niv}(Can_1)=P(Niv_0|Can_1)\lambda(Niv_0)+P(Niv_1|Can_1)\lambda(Niv_1)+P(Niv_2|Can_1)\lambda(Niv_2)=$$
$$=0.68(1)+0.24(0)+0.08(0)=0.68$$

Así tenemos $\lambda_{Niv}(\mathbf{Can})=(0.15, 0.68)$

Ahora calcularemos el mensaje λ que **Tum** envía a su padre **Can**.

$$\lambda_{Tum}(Can_0) = P(Tum_0|Can_0)\lambda(Tum_0) + P(Tum_1|Can_0)\lambda(Tum_1)$$
$$= 0.75(0.06) + 0.25(0.5168) = 0.1742$$
$$\lambda_{Tum}(Can_1) = P(Tum_0|Can_1)\lambda(Tum_0) + P(Tum_1|Can_1)\lambda(Tum_1)$$
$$= 0.48(0.06) + 0.52(0.5168) = 0.297536$$

Por lo que λ_{Tum}**(Can) = (0.1742, 0.297536).**

Calculando el valor λ de **Can** con 2.17, resulta:

$$\lambda(Can_0) = \lambda_{Niv}(Can_0)\lambda_{Tum}(Can_0) = 0.15(0.1742) = 0.02613$$
$$\lambda(Can_1) = \lambda_{Niv}(Can_1)\lambda_{Tum}(Can_1) = 0.68(0.297536) = 0.202324$$

Quedando **λ(Can) = (0.02613, 0.202324).**

Calculando las probabilidades *a posteriori* de 2.16, tenemos:

$$P(Can_0|W) = \alpha\,\pi(Can_0)\lambda(Can_0) = \alpha\, 0.666667(0.02613) = 0.01742\alpha$$
$$P(Can_1|W) = \alpha\,\pi(Can_1)\lambda(Can_1) = \alpha\, 0.333333(0.202324) = 0.067441\alpha$$

Normalizando

$$P(Can_0|W) = \frac{.01742\alpha}{.01742\alpha + 0.067441\alpha} = 0.205277$$

$$P(Can_1|W) = \frac{.067441\alpha}{0.01742\alpha + 0.067441\alpha} = 0.794723$$

Esto quiere decir, asumiendo que la red fuera correcta, que dados los síntomas el paciente tiene cerca de un 80% (0.79) de probabilidad de tener *cancer.*

Una vez más realizaremos propagación de probabilidades, sólo que ahora le asignaremos a las variables valores contrarios a los asignados anteriormente (el paciente no tiene los síntomas).

Asignándole a **Dol** un valor de 0, tenemos:

λ(Dol) = (1,0,0)

Calculando el mensaje λ que **Dol** envía a **Tum,** tenemos:

$$\lambda_{Dol}(Tum_0) = 0.4(1) + 0.4(0) + 0.2(0) = 0.4$$
$$\lambda_{Dol}(Tum_1) = 0.08(1) + 0.24(0) + 0.68(0) = 0.08$$

Por lo que λ_{Dol}**(Tum) = (0.4, 0.08).**

Si a **Com** le asignamos un valor de 0, tenemos:

λ(Com) = (1, 0)

El mensaje λ que **Com** envía a **Tum** es:

$$\lambda_{Com}(Tum_0)=0.7(1)+0.3(0)=0.7$$
$$\lambda_{Com}(Tum_1)=0.24(1)+0.76(0)=0.24$$

Quedando λ_{Com}**(Tum) = (0.7, 0.24)**.

El valor λ de **Tum** de acuerdo a los mensajes calculados es el siguiente:

$$\lambda(Tum_0)=0.4(0.7)=0.28$$
$$\lambda(Tum_1)=0.08(24)=0.0192$$

Así tenemos que **λ(Tum) = (0.28, 0.0192)**.

Suponiendo que le asignamos a **Niv** el valor de 50 obtenemos:

λ(Niv) = (0, 1, 0)

El mensaje λ que **Niv** envía a su padre **Can** será:

$$\lambda_{Niv}(Can_0)=0.15(0)+0.6(1)+0.25(0)=0.6$$
$$\lambda_{Niv}(Can_1)=0.68(0)+0.24(1)+0.08(0)=0.24$$

De acuerdo a lo anterior λ_{Niv}**(Can) = (0.6, 0.24)**.

Calculando el mensaje λ que **Tum** envía a su padre **Can**:

$$\lambda_{Tum}(Can_0)=0.75(0.28)+0.25(0.0192)=0.2148$$
$$\lambda_{Tum}(Can_0)=0.48(0.28)+0.52(0.0192)=0.144384$$

Por lo que λ_{Tum}**(Can) = (0.2148, 0.144384)**.

Obteniendo el valor λ de **Can** resulta:

$$\lambda(Can_0)=0.6(0.2148)=0.12888$$
$$\lambda(Can_1)=0.24(0.144384)=0.034652$$

Así tenemos que **λ(Can) = (0.12888, 0.034652)**

Calculando las probabilidades *a posteriori* de **Can,** obtenemos:

$$P(Can_0|W)=\alpha 0.666667(0.12888)=\alpha 0.08592$$
$$P(Can_1|W)=\alpha 0.333333(0.034652)=\alpha 0.011551$$

Normalizando

$$P(Can_0|W)=\frac{0.08592\alpha}{0.08592\alpha+0.011551\alpha}=0.881493$$

$$P(Can_1|W)=\frac{0.011551\alpha}{0.08592\alpha+0.011551}=0.118507$$

Ahora la probabilidad de tener *cancer* se ha reducido a aproximadamente el 12%.

6.2.3. Evaluación

A continuación presentamos la evaluación de las probabilidades *a posteriori* resultantes de la función de propagación. En primer lugar presentamos la evaluación para el caso donde el paciente no tiene los síntomas (caso anterior). Esta parte de los resultados se muestran en la tabla 6-1 en una manera semejante a como los despliega la herramienta. Para la otra parte se presenta la forma de obtenerlos. Considerando las probabilidades *a posteriori* obtenidas y evaluando **Can** para el valor de 0 obtenemos los resultados de la tabla 6-1.

Valor	Número Total	Numero Correcto	%
0	1	1	100
1	0	0	0

Tabla 6-1. Resultado parcial de la evaluación de las probabilidades *a posteriori*.

La *medida de Brier* es calculado conforme a 2.38, y el *error promedio* de acuerdo a 2.39:

$$M.\ Brier=(1-0.881493)^2=0.014044$$

$$E.\ promedio=\frac{0.014044}{1}=0.014044$$

Así obtenemos un error total y un error promedio de 1.4 %.

Ahora tomaremos las probabilidades *a posteriori* obtenidas cuando el paciente tiene todos los síntomas (caso inicial) para realizar la evaluación dándole a **Can** en valor de 1. La tabla 6-2 muestra los resultados mostrados por la herramienta y posteriormente se muestra la *medida de Brier y el error promedio.*

Valor	Número Total	Numero Correcto	%
0	1	1	100
1	1	1	100

Tabla 6-2. Resultado parcial de la evaluación de las probabilidades *a posteriori*.

$$M.\ Brier = 0.014044 + (1 - .794723)^2 = 0.056183$$

$$E.\ promedio = \frac{0.056183}{2} = .028092$$

De los resultados anteriores tenemos que la herramienta tiene 5.6% de error total y 2.8% de error promedio, conforme estos valores se acerque más a cero se obtiene mayor credibilidad en los resultados obtenidos al disminuir el porcentaje de error.

En los dos casos asumimos que el diagnóstico del sistema coincide con el del experto (correcto).

6.2.4. Aprendizaje paramétrico

Para finalizar este ejemplo realizaremos aprendizaje paramétrico con las variables de la red probabilística. Para llevar a cabo esta función vamos a asignarle un valor a cada variable de la red desde la raíz hasta las hojas, actualizando las probabilidades *a priori* y las matrices de probabilidad condicional correspondientes de acuerdo a las ecuaciones mostradas en 2.7.

Asignándole a **Can (Cancer)** el valor de 0, obtenemos las siguientes probabilidades *a priori.*

$P(\mathbf{Cancer}_0) = 0.673913$ $P(\mathbf{Cancer}_1) = 0.32609$

Las probabilidades condicionales son las siguientes:

Si **Niv (Nivel de calcio)** toma el valor de 25:

$P(\mathbf{Niv}_0 | \mathbf{Can}_0) = 0.190476$ $P(\mathbf{Niv}_0 | \mathbf{Can}_1) = 0.68$

$P(\mathbf{Niv}_1 | \mathbf{Can}_0) = 0.571429$ $P(\mathbf{Niv}_1 | \mathbf{Can}_1) = 0.24$

$P(\mathbf{Niv}_2 | \mathbf{Can}_0) = 0.238095$ $P(\mathbf{Niv}_2 | \mathbf{Can}_1) = 0.08$

Dándole a **Tum (Tumor cerebral)** el valor de 1:

$P(\mathbf{Tum}_0 | \mathbf{Can}_0) = 0.714286$ $P(\mathbf{Tum}_0 | \mathbf{Can}_1) = 0.48$

$P(\mathbf{Tum}_1 | \mathbf{Can}_0) = 0.285714$ $P(\mathbf{Tum}_1 | \mathbf{Can}_1) = 0.52$

Asignándole a **Com (coma)** el valor de 0 tenemos:

$P(Com_0 | Tum_0) = 0.7$ $P(Com_0 | Tum_1) = 0.269231$

$P(Com_1 | Tum_0) = 0.3$ $P(Com_1 | Tum_1) = 0.730769$

Si **Dol (dolor de cabeza)** toma el valor de 0 obtenemos:

$P(Dol_0 | Tum_0) = 0.4$ $P(Dol_0 | Tum_1) = 0.115385$

$P(Dol_1 | Tum_0) = 0.4$ $P(Dol_1 | Tum_1) = 0.230769$

$P(Dol_2 | Tum_0) = 0.2$ $P(Dol_2 | Tum_1) = 0.653846$

Al comparar estas probabilidades con las de la sección 6.2.1, observamos el cambio producido por esta observación.

6.3. APLICACION PARA EL MONITOREO Y CONTROL DE REDES ELECTRICAS

Durante varios años la industria eléctrica ha dedicado un gran esfuerzo para mejorar la eficiencia y confiabilidad de los sistemas de generación y transmisión de energía. Debido a los cambios tecnológicos, las empresas eléctricas se encuentran en la posibilidad de mejorar la eficiencia de las redes de distribución.

Los sistemas de adquisición de datos en tiempo real (**SCADA- Sistema de Control y Adquisición de Datos**) han sido muy importantes en aumentar la confiabilidad del sistema eléctrico y han permitido el uso óptimo de las inversiones en instalaciones de generación, transmisión y distribución. Un sistema **SCADA** actual además de contar con las funciones básicas de telemedición, telecontrol, adquisición de datos y alarmas. Debe de incluir funciones avanzadas como:

- Localización y confinamiento de fallas.
- Análisis de alternativas de reconfiguración automática y restablecimiento del sistema en condiciones de fallas severas o desastres.
- Herramientas de ayuda para apoyar a los operadores del sistema en la toma de decisiones fuera de línea y en tiempo real para mantener la continuidad y seguridad de operación de la red.

La herramienta desarrollada en esta tesis puede ser utilizada para realizar las funciones adicionales que requiere un sistema **SCADA** en el sector eléctrico. Por ejemplo, se puede utilizar para apoyar en la localización y diagnóstico de fallas. Esto se puede hacer a partir de conocer en un momento dado el estado de los dispositivos y protecciones que obtiene el **SCADA.** A continuación se presenta un ejemplo de aplicación en el diagnóstico de fallas en líneas eléctricas,

para ello consideremos la línea eléctrica de la figura 6-2.

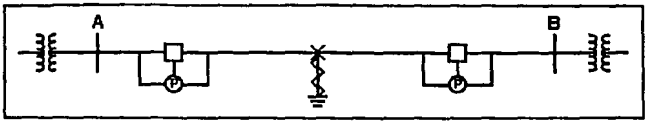


Figura 6-2. Línea eléctrica.

En primer lugar se debe de contar con una red probabilística que contenga el conocimiento correspondiente a este dominio. Puesto que en una línea pueden ocurrir diferentes fallas y cada una de ellas dispone de diversas protecciones, podemos definir un diagrama con estas relaciones en un primer nivel. Las posibles fallas que pueden ocurrir en una línea son las siguientes:

FA - Una fase abierta.
FFA - Dos fases abiertas.
LT - Línea a tierra.
LLT - Dos líneas a tierra.
LLLT - Tres líneas a tierra.
LL - Falla entre dos líneas.
LLL - Falla entre tres líneas

La figura 6-3 ilustra estas fallas y algunas de las protecciones relacionadas.

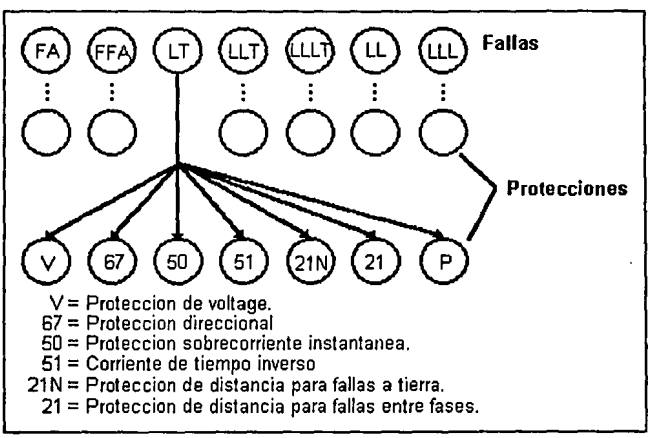


Figura 6-3. Diagrama de fallas y protecciones.

Aunque en el diagrama anterior sólo se muestran las protecciones para una falla de *línea*

a tierra, las fallas restantes contienen sus propias protecciones. Sin embargo, las distintas fallas pueden accionar una misma protección.

Considerando la falla de *línea a tierra* y basándonos en sus protecciones, determinaremos una red probabilística para la protección **50** que es de *sobrecorriente instantánea*. La red obtenida se muestra en la figura 6-4.

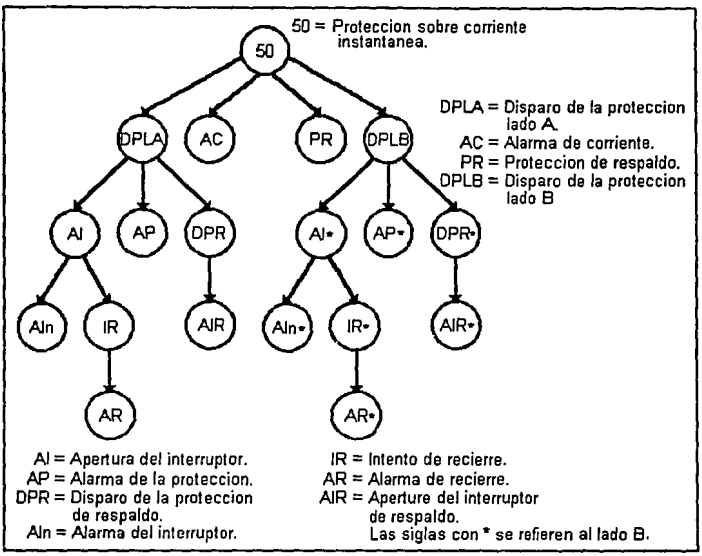


Figura 6-4. Red probabilística para la protección de sobrecorriente instantánea.

Así como la red de la figura 6-4, ha sido determinada para la protección de *corriente instantánea*, las demás protecciones cuentan con su propia red probabilística y para una aplicación real se debe de contar con diversas redes para todas las protecciones existentes, que incluyan las diferentes fallas, todas unidas en una red única. Cada uno de los nodos mostrados en la figura 6-4 debe contar con sus respectivas distribuciones de probabilidad condicional. Aunque en este ejemplo no se especifican dichas probabilidades éstas deben ser determinadas para llevar a cabo el esquema de propagación. La determinación de las probabilidades no es una tarea sencilla. En la práctica se pueden obtener de datos estadísticos que contengan la ocurrencia de disparo de las alarmas, apertura de interruptores, etc.

Una vez determinada la estructura de la red probabilística y sus distribuciones de probabilidad, la herramienta puede ser usada para determinar el tipo de falla. Para esto recibirá del **SCADA** información acerca de la activación de las distintas alarmas del sistema, que corresponden a los nodos hoja de la red probabilística. La herramienta propagará sus efectos

hacía arriba hasta alcanzar las protecciones y de ahí hasta los nodos raíz correspondientes a los distintos tipos de fallas. De tal forma que después de procesar la información, la herramienta determinará la probabilidad *a posteriori* para cada clase de falla, de donde el operador puede saber cual es la falla (más probable) que ocurrió en la red eléctrica.

6.4. CONCLUSIONES

En este capítulo presentamos dos ejemplos ilustrativos de la aplicación de la herramienta en dos áreas distintas: medicina y diagnóstico de fallas. Gracias a estos ejemplo es posible visualizar la múltiple aplicación de la herramienta, en particular en dominios que contienen incertidumbre en el conocimiento. Para la aplicación de la herramienta en distintos dominios, únicamente se requiere la definición particular de la red probabilística correspondiente a ese dominio.

En el ejemplo de medicina presentamos los cálculos realizados por la herramienta para llegar a las probabilidades *a posteriori*. Realizamos propagación en forma evidencial con dos conjuntos de valores diferentes: uno cuando el paciente presenta todos los síntomas y uno cuando no presenta ninguno. En este ejemplo es posible observar el proceso de propagación, así como las diferencias en las respuestas derivadas con diferentes datos.

En el segundo ejemplo, mostramos una aplicación para el diagnóstico de fallas. En este ejemplo definimos la estructura de la red probabilística desglosando únicamente una rama de la aplicación dada la magnitud de está área. Es importante señalar que en este ejemplo la estructura de la red fue determinada con la ayuda de un experto en el área. Aunque no se especificaron las probabilidades condicionales y *a priori*, estás pueden ser adquiridas con la ayuda del experto y de datos estadísticos, mejorándolas con la función de aprendizaje. Este ejemplo se puede tomar como un principio en una futura aplicación de la herramienta para el diagnóstico de fallas y el procesamiento de alarmas en control supervisorio de redes eléctricas.

CONCLUSIONES

INTRODUCCION

La contribución principal de esta tesis es el desarrollo de una herramienta genérica para el manejo de incertidumbre en base a redes probabilísticas. Inicialmente analizamos diversas técnicas para el manejo de incertidumbre en Sistemas Expertos. Al estudiar las diferentes técnicas para el manejo de incertidumbre, señalamos algunas deficiencias en las mismas. Dichas deficiencias se reducen con el uso redes probabilísticas, consiguiendo con ello mayor consistencia en la manipulación de incertidumbre conforme a teoría de probabilidad, resolviendo algunos problemas en el manejo de incertidumbre en Sistemas Expertos.

HERRAMIENTA BASADA EN REDES PROBABILISTICAS PARA SISTEMAS EXPERTOS

Al utilizar redes probabilísticas como la base teórica para el desarrollo de la herramienta para sistemas expertos, no sólo se obtiene la forma de representar conocimiento con incertidumbre también la manera de razonar con el. La representación del conocimiento la obtenemos en la especificación de la estructura de la red y en las distribuciones de probabilidad condicional y *a priori*. El razonamiento con incertidumbre lo conseguimos al utilizar el método de propagación de probabilidades de Pearl [82 y 86].

La estructura de la red es proporcionada por el experto del área donde será utilizada la herramienta, definiendo cada uno de los nodos y la relación de dependencias existente entre ellos. Las probabilidades condicionales y *a priori* son proporcionadas como una razón (número de casos). Estas probabilidades pueden ser obtenidas de diversas maneras. De valores asignados por el experto, los cuales son mejorados conforme se tengan casos adicionales (aprendizaje paramétrico). De valores iniciados desde cero mejorándolos con los nuevos casos que se presenten. Adicionalmente se podrán obtener las probabilidades de datos estadísticos utilizando algún programa externo.

Se definieron tres tipos diferentes de propagación: evidencial, bidireccional y en poliárboles, así como dos funciones adicionales: aprendizaje paramétrico y evaluación. La herramienta realiza propagación de probabilidades en forma evidencial, es decir de las hojas hacía la raíz. Para ello se asignarán valores a los nodos hojas propagando sus efectos a través de la estructura de la red hasta llegar al nodo raíz y obtener su probabilidad *a posteriori*. La herramienta, por medio de la función de evaluación, realiza la evaluación de la probabilidad *a posteriori* al determinar la diferencia cuadrada entre ésta y la valoración del experto para lo cual se considera un valor de uno. Además, calcula la capacidad predictiva del sistema conocida como *medida de Brier* y el *error promedio* [Sucar 92]. La función de aprendizaje se realiza incrementando en uno las probabilidades condicionales y *a priori* conforme a los valores para

cada nodo de la red probabilística.

Un aspecto que le da a la herramienta mayor flexibilidad para ser utilizada en múltiples aplicaciones, se debe a que permite trabajar con tres tipos diferentes de variables, estos son: *variables binarias* (las cuales toman los valores de uno o cero), *variables continuas* (en estas especificamos un rango y dividimos el rango en intervalos) y *variables discretas* (se trabaja con valores enteros). Aunque la herramienta trabaja con valores continuos estos valores son discretizados, es decir, los diferentes intervalos existentes en un rango, son representados cada uno por un valor entero. De no hacerlo así, para utilizar este tipo de variables se tendrían que usar distribuciones de probabilidad.

El utilizar una metodología de desarrollo de *software* orientada a objetos para desarrollar la herramienta facilita el análisis y resolución del problema al dividirlo en clases y objetos. Además, la programación orientada a objetos es "natural" para este caso, ya que existe una similitud entre objetos y su comunicación por medio de mensajes de la programación orientada a objetos, y los nodos y mensajes de redes probabilísticas.

Adicionalmente se consideró de gran importancia el contar con una buena comunicación entre el usuario y el experto, para ello se desarrollo una interfaz amigable. Dicha interfaz contiene un conjunto de ventanas para que el usuario realice con ella diversas operaciones, desde funciones particulares requeridas para redes probabilísticas, hasta operaciones sobre archivos. La interfaz con el usuario cuenta además con ventanas para realizar propagación bidireccional y en poliárboles. Aunque estas funciones no se efectúan con la herramienta, las ventanas requeridas para ello se han incluido en la interfaz, faltando únicamente el código para llevar a cabo estas funciones y ligarlas con las ventanas. Asimismo la herramienta cuenta con ventanas para recibir información de archivos externos provenientes de un *SCADA* (Sistema de Control y Adquisición de Datos) para su aplicación en tiempo real.

Entre las ventajas de la herramienta para su aplicación en tiempo-real se tienen los pocos requerimientos de almacenamiento y su eficiencia en términos de tiempo, ya que el tiempo de actualización es lineal con respecto al número de variables en la red.

En general la herramienta realiza propagación de probabilidades en forma evidencial, así como las funciones de aprendizaje y evaluación correspondientes. Cuenta con una interfaz con el usuario la cual incluye además de las funciones mencionadas, operaciones auxiliares para el esquema de propagación, así como operaciones sobre archivos. La interfaz contiene un total de 35 ventanas.

TRABAJOS FUTUROS

A continuación presentamos algunos aspectos derivados de este trabajo que pueden desarrollarse en futuros trabajos de investigación.

La herramienta esta disponible para incluir funciones para realizar propagación en forma bidireccional y en poliárboles. Para realizar propagación bidireccional será necesario anexar funciones que involucren cálculos para obtener el valor π de cada nodo y el mensaje π que un padre envía a sus hijos para todos los nodos de la red. Para efectuar propagación en poliárboles será necesario desarrollar el método propuesto por Kim y Pearl [83] el cual es derivado en forma similar al presentado en este trabajo. Adicionalmente será necesario incluir en la herramienta técnicas que permitan trabajar en redes multiconectadas, incrementando con ello la complejidad en las relaciones de dependencia. En este caso la red probabilística estará formada por varias estructuras como las que han sido mencionadas aquí.

La función de aprendizaje puede ser mejorada al incluir programas intermedios entre la herramienta y archivos de datos estadísticos, que permitan obtener las probabilidades condicionales y *a priori* de los mismos, de tal forma que el experto únicamente valide la credibilidad de dichas probabilidades.

La función de evaluación tendrá que ser mejorada al realizar otros tipos de propagación, ya que será necesario evaluar cualquier nodo de la red y no sólo el nodo raíz. Esta función también se puede mejorar, al cambiar el valor del experto, que en general se toma como un uno para todos los casos, por un valor de probabilidad (número real entre $0 \leq x \leq 1$) estimado por el experto al momento de realizar la evaluación.

Es importante mencionar la necesidad de adicionar a la herramienta funciones que permitan llevar acabo la graficación de la estructura de la red probabilística, es decir, que gráficamente se despliegue en la pantalla cada uno de los nodos existentes, así como las relaciones de dependencias entre ellos, representadas por las ligas.

Finalmente se propone la aplicación de la herramienta para el monitoreo y control de redes eléctricas, para ello será necesario la adquisición del conocimiento. La red probabilística se obtendrá con ayuda del experto, y las probabilidades condicionales y *a priori* de datos estadísticos y de la experiencia del experto. La herramienta será utilizada en tiempo real al recibir información de archivos externos provenientes de un **SCADA.** Estos serán procesados con la herramienta y, una vez que esta ha determinado las probabilidades *a posteriori*, las enviará de regreso al **SCADA.** Dicha información ayudará al operador a tomar decisiones para la solución de problemas, como el diagnóstico de fallas y procesamiento de alarmas, en forma rápida y eficiente.

HERRAMIENTA PARA LA CONSTRUCCIÓN DE INTERFASES AL USUARIO (INTERFACE ARCHITECT)

HP Interface Architect 2.0 [Interface 92] es una herramienta interactiva para la construcción de interfases al usuario. Permite la creación de una interfaz en forma rápida y sencilla, ya que contiene un conjunto de ventanas estándares para su aplicación directa en la construcción de una interfaz con el usuario (tales como ventanas para la selección de archivos, entrada de información, despliegue de mensajes, etc), además de contar con comandos que le permiten al usuario diseñar sus propias ventanas conforme a los requerimientos de su aplicación.

Interface Architect (nos referiremos a ella como *Architect*) permite refinar, especificar el funcionamiento y probar la interfaz dentro de la herramienta (esto para funciones en **C**) . *Architect* es compatible con *OSF/Motif 1.1.* [Heller 91] ya que la interfaz es creada en este estilo. Adicionalmente el paquete genera el código de la interfaz, en *"C" ANSI* [Rosler 86] o de *Kernighan & Ritchie* [Kernighan 78] y *OSF/Motif 1.1*, por lo que el diseñador no necesita conocer mucho sobre *OSF/Motif 1.1.*

Interface Architect [Interface 92] puede ser usada en una gran diversidad de aplicaciones, desde aplicaciones existentes conducidas a través de entradas por el teclado, hasta aplicaciones con una manipulación directa por medio de ventanas. *HP Interface Architect 2.0.* fue desarrollada por *Hewlett-Packard (HP)* en sociedad con *Visual Edge Software, Ltd*, de Montreal Canada. *Visual Edge* es especialista en el diseño, desarrollo, mercado y soporte de herramientas para desarrollo de *software*. *HP Interface Architect* esta basada en *UIM/X* (user interface management) desarrollada también por *Visual Edge.*

DESCRIPCION DE INTERFACE ARCHITECT 2.0

Architect es un constructor de interfases al usuario gráficas de segunda generación (*GUI Graphical Users Interface*), el cual mejora la productividad del programador al permitirle a los desarrolladores de *software* crear, modificar, probar y generar código interactivamente para la parte de interfaz con el usuario de su aplicación.

Architect está completamente integrado con las herramientas de *OSF/Motif 1.1.* Contiene un editor *WYSIWYG* (Lo que vez es lo que obtienes) para elegir gráficamente algunos de los *widgets* de el *OSF/Motif 1.1.* (un widget es cada uno de los componente básico de una interfaz tales como botones, barras deslizables, menús, etc) y diseñar su interfaz. Posee un

constructor/interprete en C. Con *Architect* se puede crear código C de enlace entre la interfaz al usuario y una aplicación. Se tiene acceso en línea a funciones compiladas, así como a un conjunto de librerías que contiene funciones de utilidad. Permite modificar una interfaz mientras la aplicación principal esta en ejecución. Es posible modificar la interfaz e inmediatamente ver los efectos sobre la aplicación. Genera código C libre de errores, así como el programa *main* (programa principal) y el *makefile* (archivo de comandos de compilación y liga) correspondiente a la aplicación.

CARACTERISTICAS DE INTERFACE ARCHITECT

- *Architect* contiene un conjunto de ventanas preconstruidas para facilitar el tiempo y esfuerzo requerido para desarrollar una interfaz estándar. Este conjunto de ventanas también sirve como una introducción a la amplia variedad de técnicas de *OSF/Motif 1.1.*

- Contiene un editor *WYSIWYG* fácil de usar, el cual permite crear *widgets* desde la paleta del usuario, o de diversos menús.

- Dispone de varios editores especializados, integrados a través del uso de una metáfora de arrastrar y soltar.

- El editor de propiedades de *widgets* permite editar las propiedades de múltiples *widgets* al mismo tiempo. Contiene un selector de color, fuentes, mapa de pixels y llamadas a funciones. Este editor posee una lista de opciones para elegir la deseada, resultando muy sencillo asignar valores a los recursos (un recurso es una propiedad que define el funcionamiento y apariencia de un widget).

- Contiene un editor de menús para crear menús *popup*, menús *pulldown*, menús de *opciones*[4], colocar etiquetas, aceleradores, nemónicos y hacer llamadas a funciones.

- Permite escribir código C correspondiente a las funciones que serán llamadas en un editor de funciones especializado. *Architect* guarda el código de las funciones como parte de la definición del proyecto, sin necesidad de modificar el código C generado, cada vez que la GUI es modificada. Se pueden utilizar llamadas a funciones de X [O'Reilly 89], Xm [Heller 91] o Xt [Nye 90] todas ellas ligadas dentro de *Architect*.

---

[4] Un menú *popup* es aquel que se despliega en la pantalla o en una interfaz, por períodos cortos de tiempo mientras el usuario presiona el botón derecho del ratón. Un menú *pulldown* siempre está visible en una interfaz. Se utiliza generalmente en barras de menú, de tal forma que cuando el usuario presiona el botón izquierdo del ratón se despliega una lista de comandos. Un menú de opciones también permanece visible sólo que se representa como un botón con una etiqueta. Cuando este botón es activado con el ratón, se despliega una lista de opciones permitiendo seleccionar alguna, cambiando la descripción de la etiqueta por la seleccionada.

- Tiene editores para especificar el funcionamiento de ventanas de aplicación (por ejemplo, el área donde se dibujará en un programa de dibujo). El usuario gráficamente elige eventos y los liga con el código C de las acciones ha realizar.

- Permite entrar expresiones de C como el valor de una propiedad.

- Cuenta con librerías de funciones para llevar a cabo la programación del funcionamiento. Las librerías de *Architect* reducen significativamente la complejidad y el tamaño del código, comparado a la programación en *OSF/Motif 1.1*.

- Posee un interprete de C integrado. El usuario puede probar su interfaz con su aplicación en ejecución. El interprete puede mezclar código compilado e interpretado, así es posible llamar funciones compiladas, o compilar la interfaz para su ejecución.

- *Architect* genera código C de *Kernigan & Richie* [Kernighan 78] ó *"C" ANSI* [Rosler 86], y código *OSF/Motif 1.1*, puro ó código de las librerías de *Architect*. Así los desarrollos de *software* quedan protegidos y se asegura su portabilidad.

FASES EN EL DESARROLLO DE UNA INTERFAZ

Fase Uno: Presentación

La primera fase del desarrollo de una interfaz es la presentación (la elaboración de la apariencia de la aplicación). Con *Architect* es posible esbozar una interfaz con una manipulación directa de un editor *WYSIWYG*. *Architect* soporta la edición de todos los objetos de *OSF/Motif 1.1.* como son: *Shells*, *widgets*, *gadgets*, *dialogos* y *recursos*.

Los *Shells* son *widgets* de alto nivel que contienen un manejador de *widgets* (hijos). Su trabajo es manejar a los componentes primitivos tales como *etiquetas, widgets de texto, barras deslizables* y *botones.*

Los *gadgets* son similares a los *widgets* pero difieren en dos cosas: ofrecen un alto desempeño porque no tienen una ventana **X** y, debido a lo anterior heredan propiedades de sus padres (por ejemplo color de fondo) ofreciendo una menor flexibilidad.

Un *diálogo* es una ventana de alto nivel que puede ser manipulada por un manejador de *widgets*. Generalmente indican un estado de la aplicación. Existen 5 tipos de *diálogos* estos son: de información, error, interrogación, precaución y de trabajo.

Fase Dos: Adicionando funcionamiento

La segunda fase del desarrollo de la interfaz es la especificación del funcionamiento. Mientras que la presentación envuelve especificar el estado inicial estático, la especificación del funcionamiento involucra colocar todos los elementos dinámicos de la interfaz. Especificar y

probar el funcionamiento requiere aproximadamente del 60% al 80% del tiempo requerido para el desarrollo de una interfaz con el usuario.

Los principales componentes de la especificación del funcionamiento son los siguientes:

- Especificación de las funciones que serán llamadas. Acciones que ocurren cuando el usuario presiona un botón, mueva la barra de desplazamiento o elija alguna entrada del menú.

- Especificación del funcionamiento de la ventana de la aplicación, es decir, acciones que ocurren en la ventana de la aplicación. Por ejemplo, el área donde el usuario dibuja líneas y círculos en un programa de dibujo. El desarrollador especifica que acción se realizará cuando el usuario presione un botón del ratón y lo arrastre.

- Especificación de los elementos dinámicos de el estado inicial. Estos elementos dependen dinámicamente del estado de la aplicación. Por ejemplo, la colocación de un mensaje en una caja de diálogo, que puede ser diferente cada vez que el diálogo es desplegado.

Architect soporta tres tipos de especificación del funcionamiento: llamadas a funciones de una aplicación escritas en **C**. Llamadas a funciones **Xm, Xt, y X**. En adición, se pueden hacer llamadas a *widgets* propios, los cuales también pueden ser ligados con *Architect.*

Para facilitar la especificación del funcionamiento, *Architect* contiene un conjunto de librerías. Las librerías desarrollan tareas complejas, tales como conversión de recursos, entradas para mapas de color, manejo automático de hijos y manejo de casos especiales de arreglos geométricos. Adicionalmente las librerías llevan a cabo el chequeo de errores, notificación y recuperación.

El tercer componente del funcionamiento de una interfaz es el estado inicial dinámico de la misma. Cuando se despliega una interfaz, típicamente del 15% al 20% de los recursos en muchas interfases dependen del estado de la aplicación. Por ejemplo, realizar ciertos cambios cuando un nuevo archivo es cargado o colocar un nuevo argumento arriba de la línea de comandos. *Architect* permite especificar y probar estas "*Interfases del usuario paramétricas*".

Para la construcción de una interfaz paramétricas, el usuario puede especificar una expresión de **C** como el valor inicial de algún recurso, entonces la expresión de **C** es evaluada cuando el *widget* es creado. La expresión de **C** puede ser alguna expresión legal de **C** que regrese el tipo correcto: por ejemplo, una llamada a una función (con argumentos) dentro de la aplicación o una variable global de la aplicación. El interprete de *Architect* puede accesar y llamar código compilado de la aplicación en desarrollo.

Fase Tres: Probando y refinando la interfaz

La construcción de una interfaz al usuario es sólo la mitad de la batalla. Probar y refinar la interfaz es igualmente importante y consume tiempo. Con el constructor/interprete de **C** es posible construir, modificar y probar una interfaz con la aplicación en ejecución, pero sin el ciclo de compilación-liga-depuración.

El Interprete de **C** puede mezclar código compilado y código interpretado. Este puede modificar las variables globales compiladas y aun pasar la dirección de una rutina interpretada, dentro de una rutina compilada para la ejecución.

Architect soporta un *enfoque incremental y un enfoque modular* para el desarrollo de *software*. Con el primero se puede usar el interprete de **C** para desarrollar rápidamente una interfaz y cuando se este satisfecho, compilar la interfaz individualmente para su ejecución. El enfoque modular de desarrollo permite entrar el código de la interfaz en funciones, separando de la interfaz el código de la aplicación.

Fase Cuatro: Generación de Código

Cuando se ha acompletado una interfaz, *Architect* genera el código **C** y los archivos de recursos **X** correspondientes. Es posible colocar algunos recursos de ciertos *widget* de la interfaz, dentro de un archivo de recursos de tal forma que el usuario pueda personalizar su interfaz con el usuario.

EJEMPLO PARA LA CREACION DE UNA INTERFAZ

A continuación presentamos un ejemplo para la creación de una interfaz al usuario. Este ejemplo consta de dos ventanas. La primera de ellas (lleva como título *Ejemplo liga IA/C++*) contiene un área de texto titulada *Nombre*, donde el usuario puede escribir un nombre, y una segunda área donde dicho nombre aparece en mayúsculas cuando el usuario presiona un botón etiquetado como *Convertir*. Además se cuenta con un botón con la etiqueta *Salir*. Este botón despliega una segunda ventana con el título *Información*, la cual contiene avisos para el usuario. Si el usuario presiona el botón titulado *OK* se termina el ejemplo y aparece un mensaje de despedida en la pantalla. Si se presiona el botón etiquetado como *Desplegar* aparece en la pantalla el nombre entrado por el usuario, sólo que ahora en minúsculas. La figura I muestra estas ventanas.

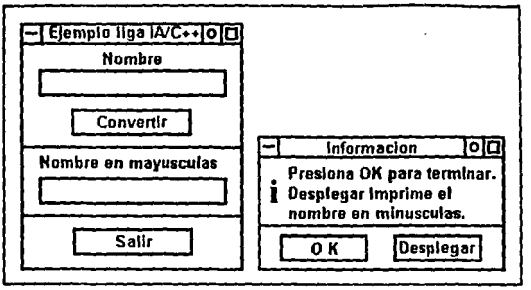


Figura I. Ventanas para el ejemplo de liga IA/C++.

Adicionalmente en este ejemplo se presenta un aspecto de interés que es el ligar funciones escritas en C++ con *Interface Architect*, dichas funciones son las que hacen la conversión de mayúsculas y minúsculas.

Fase uno "Presentación". En esta fase se definen los componentes de cada una de las ventanas, es decir, se hace un bosquejo de estas (Figura I). Para comprender la constitución de las ventanas, presentamos su estructura en forma de un árbol de *widgets* (figura II).

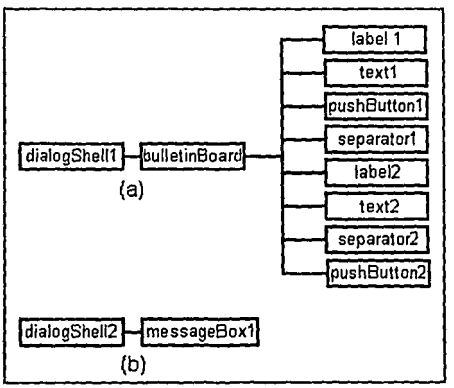


Figura II. Arbol de widgets para las ventanas del ejemplo de creación de una interfaz.

Fase dos "Funcionamiento". En esta fase se especifica el código para cada una de las acciones que se van a realizar cuando se presione uno de los botones de las ventanas. El botón titulado *Convertir* tiene el código siguiente:

```
{
  nombre= XmTextGetString(UxGetWidget(text1));
  nombre=uno(nombre);
  XmTextSetString(UxGetWidget(text2), nombre);
}
```

Para el botón etiquetado como *Salir*, el código es el siguiente:

```
{
  UxPopupInterface(dialogShell2, no_grab);
}
```

A continuación presentamos el código de la ventana que tiene la etiqueta de *Información*. El botón titulado *OK* tiene el código siguiente:

```
{
  int i, x, y;
  x=0;
  printf("\n");
  for (i=19;i<28;i=i+3){
     x=0;
     while(x<i){
        printf(" ");
        x++;
     }
  if (i<25) printf("Bye\n");
  else printf("Bye ...\n\n");
  }
}
printf("\n\n");
exit();
}
```

El código del botón *Desplegar* se muestra a continuación:

```
{
  nombre=dos(nombre);
  printf("\n");
  printf("El nombre en minúsculas es: %s\n ", nombre);
  UxPopdownInterface(dialogShell2);
}
```

Asimismo presentamos el código correspondiente con las funciones de C++ llamadas *uno* y *dos* las cuales son utilizadas cuando se oprimen los botones: *Convertir* y *Desplegar.*

```
/*****************************************************
*              archivo: programa.C
*          Contiene las funciones de C++
*****************************************************/

# include <stdio.h>
# include <ctype.h>
extern "C"{
   extern char* uno(char *);
   extern char* dos(char *);
}

/*****************************************************
*                  subrutina uno
*****************************************************/

char* uno(char *p)
{
   int i=0;
   for(i=0; p[i]; i++)
      p[i]=toupper(p[i]);
   return(&p[0]);
}

/*****************************************************
*                  subrutina dos
*****************************************************/

char* dos(char *p)
 {
   int i=0;
   for(i=0;p[i];i++)
      p[i]=tolower(p[i]);
   return(&p[0]);
 }
```

Fase tres "Probando y Refinando la Interfaz". En esta fase probamos las dos ventanas con *Architect*, sólo que no realizamos las llamadas a las funciones de C++ ya que el interprete no lo permite. Este interpreta únicamente código en *"C" Ansi o Kernighan & Ritchie* (Ver la parte de "Problemas y Limitaciones" y liga con C++).

Fase cuatro "Generacion del Código". La última fase es la generación del código. Como ya hemos mencionado *Architect* genera el código sin la necesidad de la intervención del usuario. A continuación presentamos el código para la generación del programa principal del ejemplo.

```
/*--------------------------------------------------------------
 * This is the project main program file for Xt generated
 * code. You may add application dependent source code
 * at the appropriate places.
 *
 * Do not modify the statements preceded by the dollar
 * sign ($), these statements will be replaced with
 * the appropriate source code when the main program is
 * generated.
 *
 * $Date: 92/01/26 13:01:54 $              $Revision: 1.11.36.1 $
 *--------------------------------------------------------------*/

#ifdef XOPEN_CATALOG
#include <locale.h>
#endif

#include <X11/Intrinsic.h>
#include <X11/StringDefs.h>

/*---------------------------------------------------
 * UxXt.h needs to be included only when compiling a
 * stand-alone application.
 *---------------------------------------------------*/
#ifndef DESIGN_TIME
#include "UxXt.h"
#endif /* DESIGN_TIME */

XtAppContext        UxAppContext;
Widget              UxTopLevel;
Display             *UxDisplay;
int             UxScreen;

/*-----------------------------------------------
 * Insert application global declarations here
 *-----------------------------------------------*/

main(argc,argv)
        int     argc;
        char    *argv[];
{ _main();
{
```

```
/*-------------------------------------------------------
 * Declarations.
 * The default identifier - mainIface will only be declared
 * if the interface function is global and of type swidget.
 * To change the identifier to a different name, modify the
 * string mainIface in the file "xtmain.dat". If "mainIface"
 * is declared, it will be used below where the return value
 * of PJ_INTERFACE_FUNCTION_CALL will be assigned to it.
*-------------------------------------------------------*/
      Widget mainIface;
/*-------------------------------
 * Interface function declaration
 *-------------------------------*/
      Widget  create_dialogShell1();
/*--------------------
 * Initialize program
 *--------------------*/

#ifdef XOPEN_CATALOG
      setlocale(LC_ALL, "");
#endif

      UxTopLevel = XtAppInitialize(&UxAppContext, "prueba",
                                   NULL, 0, &argc, argv, NULL, NULL, 0);

      UxDisplay  = XtDisplay(UxTopLevel);
      UxScreen = XDefaultScreen(UxDisplay);

/* We set the geometry of UxTopLevel so that dialogShells
   that are parented on it will get centered on the screen
   (if defaultPosition is true). */
      XtVaSetValues(UxTopLevel,
                    XtNx, 0,
                    XtNy, 0,
                    XtNwidth, DisplayWidth(UxDisplay, UxScreen),
                    XtNheight, DisplayHeight(UxDisplay, UxScreen),
                    NULL);

/*-------------------------------------------------------
 * Insert initialization code for your application here
 *-------------------------------------------------------*/
 /*-----------------------------------------------------------------
 * Create and popup the first window of the interface.  The
 * return value can be used in the popdown or destroy functions.
 * The Widget return value of  PJ_INTERFACE_FUNCTION_CALL will
 * be assigned to "mainIface" from  PJ_INTERFACE_RETVAL_TYPE.
*-------------------------------------------------------------*/
```

```
 mainIface = create_dialogShell1();
 UxPopupInterface(mainIface, no_grab);
/*----------------------
 * Enter the event loop
 *----------------------*/
      XtAppMainLoop (UxAppContext);
}
}
```

A continuación se presenta el código para el *makefile* (archivo de comandos para realizar la compilación y liga del programa).

```
#########################################################################      #
MAKEFILE FOR STAND-ALONE XT CODE APPLICATION.
#
#   EXECUTABLE      is the name of the executable to be created
#   MAIN            is the .c file containing your main() function
#   INTERFACES      is a list of the generated C code files
#   APP_OBJS        is a (possibly empty) list of the object code
#   files that form the non-interface portion of #   your application
#
#   In the first three statements, the variables on the right
#   of the equal sign will be replaced with their corresponding
#   values when the makefile is automatically generated.
#
#   This template is used for makefiles which do not reference
#   the Ux runtime library.
#
#      $Date: 92/02/28 19:41:28 $Revision: 1.27.36.4 $
#########################################################################

EXECUTABLE      = prueba
MAIN            = prueba.c
INTERFACES      = dialogShell1.c \ dialogShell2.c

APPL_OBJS       = UxXt.o programa.o
UX_DIR          = /usr/architect2.0
UX_LIBPATH      = $(UX_DIR)/lib
X_LIBS          = -lXm -lXt -lX11

X_LIBPATH       = -L/usr/lib/X11R4
MOTIF_LIBPATH   = -L/usr/lib/Motif1.1
X_CFLAGS        = -I/usr/include/X11R4
MOTIF_CFLAGS    = -I/usr/include/Motif1.1

# flags for HP-MC680*00 machines (series 300, 400)
S300_FLAGS      = -Wc,-Ns10000,-Np300,-Nw500 -Wp,-H400000
```

```
# flags for HP-PA machines (series 600, 700, 800)
S800_FLAGS      = -Wp,-H400000 -z

# uncomment the following line for ANSI-C code generation
#PROTOTYPES      = -Aa -D_HPUX_SOURCE
# uncomment the following line for K&R C code generation
PROTOTYPES      = -D_NO_PROTO
CFLAGS          = -DSYSV $(PROTOTYPES) -DXOPEN_CATALOG \
                  $(X_CFLAGS) $(MOTIF_CFLAGS) \
                  $(S800_FLAGS)
LIBPATH       = $(X_LIBPATH) $(MOTIF_LIBPATH)
LIBS          = $(X_LIBS) -lm -lPW


OBJS = $(MAIN:.c=.o) $(INTERFACES:.c=.o) $(APPL_OBJS)

# ****************************************************************
#     Change linking of final executable from c to C++
$(EXECUTABLE): $(OBJS)
        CC $(OBJS) $(LIBPATH) $(LIBS) -o $(EXECUTABLE)
        @echo "done"
```

Como el código mostrado anteriormente, *Architect* genera código para las dos ventanas diseñadas en el ejemplo y el proyecto completo.

PROBLEMAS Y LIMITACIONES.

Interface Architect apesar de ser una herramienta muy poderosa encontramos algunos problemas y limitaciones durante el desarrollo de la interfaz. A continuación enumeramos algunos de ellos.

1.- Existen pequeñas discrepancias en la información contenida en los manuales y algunas librerías de *Architect*.

2.- En ocasiones puede existir una diferencia en la forma en que funciona una interfaz y su aplicación en modo de prueba, y el funcionamiento de la aplicación completa con todos los archivos compilados. Esto se debe en la mayoría de los casos, a que en modo de prueba no es necesario declarar variables utilizadas en diferentes ventanas como globales y al generar el código compilado se detecta un error por falta de definición de dichas variables.

3.- Cuando se declara una variable global en un archivo y en otros se declara la misma variable como global externa, *Architect* ya no permite modificar la declaración. Si esto fuera necesario se tendrá que salir de *Architect* y modificar las declaraciones en los

archivos de las interfases con extensión **.i**, y posteriormente entrar de nuevo a *Architect* y cargar otra vez el proyecto.

4.- Cuando se declara una variable en el editor de declaraciones como variable específica, dicha variable entra en el contexto de estructura (es un arreglo de variables pertenecientes a una ventana dada) y por esta razón el nombre de esta ya no podrá ser utilizada por ninguna ventana en la aplicación.

5.- Después de un tiempo de estar trabajando con *Architect* esta pierde velocidad y si se llegasen a hacer varias operaciones rápidamente la máquina puede llegar a perder el control de la memoria, parar la aplicación y el sistema completo. Siendo necesario reiniciar el sistema.

6.- Una de las principales limitantes de *Architect* se debe a que el interprete no puede trabajar con **C++**. Adicionalmente si las funciones de la aplicación del usuario están en **C++** *Architect* no permite trabajar con sus funciones propias (funciones **Ux**, por ejemplo *UxGetWidget*) siendo necesario utilizar funciones de *OSF/Motif 1.1.* o *Xt.* A continuación se describen los pasos que hay que realizar para poder trabajar con **C++** e *Interface Architect.*

USO DE C++ CON HP INTERFACE ARCHITECT

Para llamar rutinas escritas en **C++** de ventanas generadas con *Interface Architect* se deben realizar los pasos siguientes:

1) Desarrollar la interfaz al usuario y las rutinas de **C++**.

2) Utilizar el generador de código de architect (*uxcgen*) para producir el código *Xm/Xt.*

3) Modificar la función *main.*

4) Modificar el *makefile.*

5) Generar el código para la interfaz.

6) Compilar la aplicación. (opcional).

7) Probar la aplicación.

Notas:

- No se puede llamar o accesar rutinas de **C++** con **Architect** estando en modo de prueba, de hacerlo así se despliega un mensaje de error.

- Puesto que este procedimiento requiere la generación de código *Xm/Xt,* no se deben hacer

llamadas a librerías de Architect tales como *UxPutResource* o *UxSendSubproc* y en caso de ser necesario ocupar alguna, utilizar funciones de *OSF/Motif 1.1 o de X toolkit* que sean semejantes.

Las únicas funciones **Ux** que se pueden utilizar son:

```
UxPutInterface(nombre_interface, tipo_grab);
UxPopdownInterface(nombre_interface);
```

A continuación se desglosan los puntos mencionados anteriormente.

1) Desarrollar la interfaz al usuario y las rutinas de *C++*.

- Crear la interfaz para la aplicación.

- Todas las rutinas de C++, referenciadas dentro de *architect*, deben ser declaradas externas en el archivo en el que se encuentran. En el ejemplo presentado las funciones se declaran en el archivo *programa.C*. Las declaraciones son:

```
extern "C"{
    extern char* uno(char *);
    extern char* dos(char *);
}
```

Adicionalmente se debe de incluir un archivo ".h", en las ventanas donde se utilizan las funciones de C++. El archivo tiene las siguientes declaraciones:

```
/*******************************************************
* Archivo cabecera utilizado para programa.C
*       archivo: programa.h
* A continuacion se declaran las rutinas de C++
*******************************************************/
char *uno( );
char *dos( );
```

Lo anterior para el caso que el código **C** generado sea *K&R*. En caso de que el código sea **ANSI-C**, las funciones se declararan como sigue:

```
char *uno(char*);
char *dos(char*);
```

2) Seleccionar generación de código *Xm/Xt*.

La generación de código *Xm/Xt* es seleccionado de el menú vía:

"Options" ------> *"Write Code"* ------> *"Xm/Xt Code"*.

3) Modificar la función *main*().

Es necesario incluir una rutina *main*() de C++ compatible con la función *main()* original. Una forma fácil de construirla es la siguiente:

- Abrir el *"Xt Main Program"* accesado de el *"Program Layout Editor"* el cual es accesado de el menú vía:

"Edit" ------> *"Program Layout"*

- Adicionar lo siguiente a el *"Xt Main Program"*.

```
_main();
{
```

La primer línea va después de la llave "{" de *main*, quedando:

```
main(argc, argv)
int argc;
int char *argv[];
{ _main();
{
```

La última línea de este archivo debe ser "}". Lo anterior se debe a que el compilador hace una referencia a *main*, pero usa *_main()* en lugar de *main()*. Ver el archivo de generación del programa principal del ejemplo (fase cuatro)

4) Modificaciones a el *makefile*.

- Adicionar el archivo objeto que contiene las rutinas de C++ a el *makefile*. En este caso el archivo *programa.o* fue adicionado en la línea de declaraciones que contiene *APPL_OBJS=*. El código será compilado con el compilador apropiado (cc) para objetos cuyo archivo fuente termine en .c y el compilador de (CC) para objetos cuyo archivo fuente termine en .C.

- Cambiar el compilador usado, para ligar el ejecutable de el compilador de C a el compilador de C++.

Para cambiar el compilador en el *makefile*, la línea

$ (CC) $(OBJS) $(LIBPATH) $(LIBS) -o $(EXECUTABLE)

al final del archivo, debe ser igual a:

CC $(OBJS) $(LIBPATH) $(LIBS) -o $(EXECUTABLE)

Lo anterior se puede observar en el *makefile* del ejemplo mostrado en la fase cuatro.

5) Generar el código C para su interfaz. En este caso se generan los archivos *dialogShell1.c, dialogShell2.c,* el archivo principal del proyecto *prueba.c* y el *makefile prueba.mk.* Los archivos anteriores se generan con *Architect* vía:

"File" ------> "Write C code as"

La opción *"Write C code as"* también crea el ejecutable si el usuario así lo desea, en caso contrario el usuario tendrá que generarlo posteriormente como se menciona en el siguiente inciso.

Igualmente se puede utilizar la opción *"write C code".* A diferencia de la anterior esta opción no solicita un nombre para los archivos a generar.

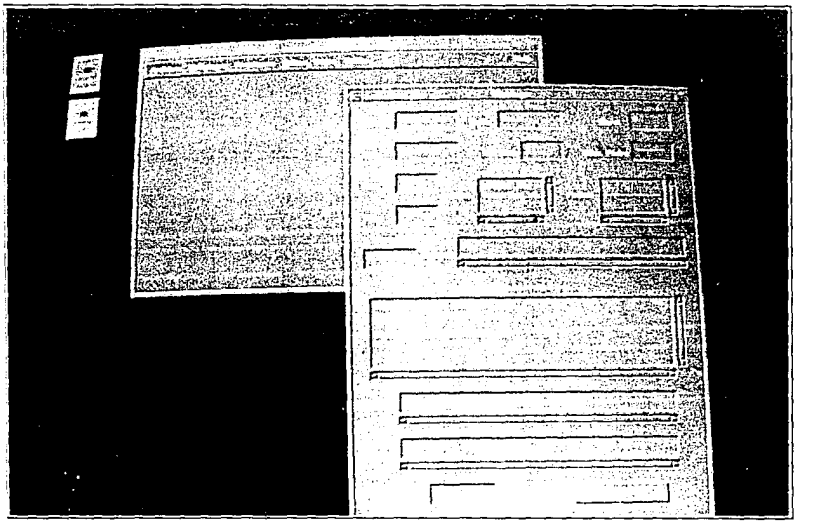
6) Compilar la aplicación. El último paso es compilar la aplicación, si aun no se a generado el ejecutable con la opción *"Write C code as"* de el menú *"File".* Esta operación se lleva a cabo utilizando el comando *make,* de la siguiente forma.

$ make -f prueba.mk

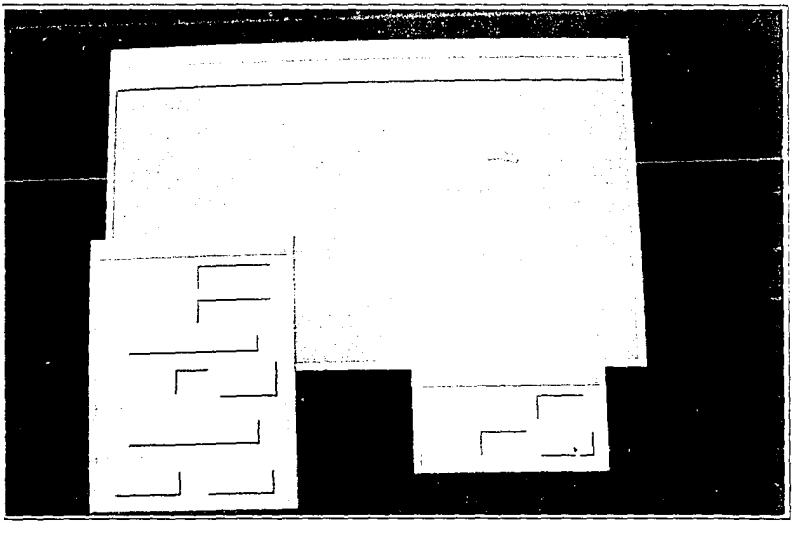
7) Probar la aplicación.

VENTANAS CREADAS CON INTERFACE ARCHITECT PARA LA HERRAMIENTA

A continuación se presentan algunas de las ventanas generadas con *Interface Architect* para la herramienta desarrollada en este trabajo. Estas se obtuvieron mediante fotografías del monitor de una computadora **HP 900 serie 700 modelo 710,** en la cual se desarrollo este trabajo.



Esta fotografía muestra el menú principal y la ventana para despliegue de información.



Esta fotografía muestra el menú principal, la ventana para entrada de datos y la ventana para la asignación de las probabilidades *a priori*.

Abramson, B. y Ng. K. [1990], "Uncertainty Management in Expert Systems". *IEEE Expert*, Vol. 5, No. 2, pp. 29-48.

Adams, J.B. [1976], "A Probability Model of Medical Reasoning and the MYCIN Model", *Math. Biosciences*, Vol. 32, pp. 177-186.

Binfort, T. O., Levitt, T. S. and Mann W. [1989], "Bayesian Inference in Model-Based Machine Vision", *Proceeding of the third AAAI Workshop on Uncertainty in Artificial Intelligence*, Seattle, Washington.

Booch, Grady. [1991], *Object-Oriented Design with Applications*, Benjamin/Cummings, Menlo Park, Calif.

Bratko, I. [1986], *Prolog Programming For Artificial Intelligence*, Addison-Wesley, Wokinham, England.

Buchanan, B.G. and Shortliffe, E.H. [1984], *Rule-Based Expert Systems: The MYCIN Experiments of the Standford Heuristic Programming Project*,B.G. Addison-Wesley, Reading, Mass.

Ceballos, F. J. [1991], *Curso de Programación C++, Programación Orientada a Objetos*, Rama y Addison-Wesley, Madrid, España.

Cohen, P.R. [1985], *Heuristic Reasoning about Uncertainty: An AI Approach*, Pitman, Boston, Mass.

De Kleer, J. [1986], "An Assumption-Based TMS", *Artificial Intelligence*, Vol. 28, pp. 127-162.

Dempster, P, [1967] "Upper and Lower Probabilities Induced by a Multivalued Mapping", *Annals. Math. Statistics*, Vol. 38, No. 2, pp. 325-339.

Doyle, J. [1979], "A Truth Maintenance System", *Artificial Intelligence*, Vol. 12, pp. 231-272.

Duda, R.O., Hart P.E., and Nilsson, N.L. [1976] "Subjetive Bayesian Methods for a Rule-Based Inference System", *Proc. Nat'l Computer Conf*, Vol. 45, pp. 1075-1082.

Duda, R.O., Gasching, J. y Hart, P.E. [1979], "Model Design in Prospector Consultant System For Mineral Exploracion", en *Expert Systems in the Micro-Electronic Age*, D.Michie, Edinburgh Univ. Press, Edinburgh, U.K., pp. 153-167.

Finetti de, B. [1974], *Theory of Probability*, John Wiley & Sons, New York, N.Y.

Gordon R.A. y Shortliffe, E.H. [1985], "A Method for Managing Evidential Reasoning in a Hierarchical Hypotesis Space", *Artificial Intelligence*, Vol. 26, No. 3, pp. 323-357.

Hammond, P. [1983], "APES: A Prolog Expert System Shell", *Internal Report*, Department of Computing, Imperial College, London, UK.

Heckerman, D.E. [1986], "Probability Interpretation for MYCIN's Certainty Factors", en *Uncertainty in Artificial Inteligence*, L.N. Kanal and J.F. Lemmer (eds), Elsevier Science Publishers, New York, N.Y., pp. 167-196.

Heckerman D.E. y Horvitz, E.J. [1987], "On the Expressiveness of Rule-Based Systems for Reasoning under Uncertainty", *Proc. Sixth, Nat'l. Conf. AI*, Morgan Kaufman, Palo Alto, Calif, pp. 121-126.

Heller, Dan. [1991], *Motif Programming Manual*, Vol 6, O'Reilly & Associates, Sebatopol, CA, USA.

Horvitz E.J. y Heckerman, D.E. [1986], "The Inconsistent Use of Measures of Certainty in AI Research", en *Uncertainty in Artificial Intelligence*, L.N. Kanal and J.F. Lemmer, (eds.), Elsevier Science Publisher, New York, N.Y., pp. 137-152.

Interface Architect 2.0, [1992], *Architect 2.0. Developers Guide* Hewlett-Packard & Visual Edge, Palo Alto, CA. USA.

Jackson, Peter, [1986]. *Introducction to Expert Systems*, Addison-Wesley, Reading, Massachusetts.

Kernighan, and Ritchie, [1978], *The C Programming Language*, Prentice-Hall, Englewood Cliffs, NJ.

Kim, J. and Pearl, J. [1983], "A Computational Model for Combined Causal and Diagnostic Reasoning in Inference Systems", *Proc. IJCAI*, pp. 190-193.

Laurlxzen, S. L. and Spiegelhalter, D. J. [1988], "Local Computations with Probabilities on Graphical Structures and their Application to Expert Systems", *J. Royal Statistical Society*, Series B, Vol. 50, No. 2, pp. 157-224.

Lindley, D.V. [1987], "The probability Approach to the Treatment of Uncertainty in AI and Expert Systems", *Statistical Sciece*, Vol.2, No. 1, pp. 17-24.

McCarthy, J. [1980], "Circumscription: A Form of Non-Monotonic Reasoning", *Artificial Intelligence*, Vol. 13, pp. 27-39.

McDermott, D., y Doyle, J. [1980], "Non-Monotonic Logic", *Artificial Intelligence*, Vol. 13. pp 41-72.

Minsky, M. [1975] "A Framework for Representing Knowledge", in P. Winston (Ed.), *The Psychology of Computer Vision*, Mc Graw-Hill, New York, N.Y. pp. 211-277.

Neapolitan, R. E. [1990]. *Probabilistic Reasoning in Expert Systems: Theory and Algorithms*, John Wiley & sons, New York, N.Y.

Nilson, N.J. [1982], *Principles of Artificial Intelligence* Springer-Verlag, Heidelberg, Germany.

Nye, Adrian, and Tim O' Reilli, [1990], *X Toolkit Intrinsics Programming Manual*, Vol 4, O'Reilly & associates, Sebatopol, CA, USA.

Pearl, J., [1982], "Reverend-Bayes on Inference Engines: A Distribuited Hierarchical Approach", *Proc. National Conference on Artificial Intelligence*, pp. 133-136.

Pearl, J. [1985], "How to Do with Probabilities What People Say You Can't", *Proc. Second Conf. AI Applications, IEEE Computer Society Press*, Los Alamitos, Calif., December, pp 6-12.

Pearl, J. [1986], "On Evidential Reasoning in a Hierarchy of Hypothesis", *Artificial Intelligence*, Vol. 28, pp. 9-15.

Pearl, J. [1988], *Probabilistic Reasoning in Intelligence Systems*, Morgan Kaufmann, San Mateo, Calif.

Quillian, M.R. [1968] "Semantic Memory", en *Semantic Information Processing*, M. Minsky (Ed,), MIT Press, Cambridge, Mass., pp. 227-270.

Reiter, R. [1980], "A Logic For Default Reasoning", *Artificial Intelligence*, Vol. 13, pp. 81-132.

Robinson, J.A. [1965], "A Machine-Oriented Logic Based on the Resolution Principle", *Journal of the ACM*, Vol 12, pp. 23-41.

Rosler, L. [1986], *Preliminary Draft Proposed Standard-The Languaje*, X3 Secretariat, Computer and Business Equipment, Washington, D.C.

Schild, H. [1991], *Aplique Turbo C++*, Osborne/MacGraw-Hill, México.

Shafer, G. [1976], *A Mathematical Theory of Evidence*, Princeton Univ. Press. Princeton, N.J.

Shortliffe, E.H. and Buchanan, B.G. [1975], "A Model of Inexact Reasoning in Medicine", *Mathematical Biosciencies*, Vol 23, pp. 351-379.

Spiegelhalter, D. J. [1986a], "Probabilistic Reasoning in Predictive Expert Systems", en *Uncertainty in Artificial Intelligence*, L.N. Kanal and J.F. Lemmer (Eds.), North-Holland, Amsterdam, pp. 47-68.

Spiegelhalter, D. J., [1986b], "A Statistical View of Uncertainty in Expert System", en *Artificial Intelligence and Statistics*, W.A. Gale (Ed.), Addison-Wesley, Reading, Mass.

Stalling, W. [1977], "Fuzzy Set Theory Versus Bayesian Statistics", *IEEE Trans. Systems, Man, and Cybernetics*, Vol. 7, No. 3, pp. 216-219.

Stroustrup Bjarne, [1986], *The C++ Programming Language*, Addison-Wesley, New Jersey.

Sucar, L. E., Gilles, D. F. and Gilles D.A., [1991], "Handling Uncertainty in Knowledge-Based Computer Vision", en *Symbolic and Quantitative Approaches to Uncertainty*, (ECSQAU-91) R. Kruse and P. Siegel (Eds), Springer-Verlag, Berlin. pp. 328-332.

Sucar. L. E. [1992], *Probabilistic Reasoning in Knowledge-Based Vision Systems, Tésis Doctoral*, Imperial College of Science, Technology, and Medicine, Londres, Inglaterra, 1992.

Sucar, L. E., Gilles, D. F. and Gilles D.A., [1993], "Objective Probabilities in Expert Systems", en *Artificial Intelligence*, Elsevier, Science Publisher, New York, N.Y., pp. 187-207.

Waterman, D.A. and Hayes-Roth, F. [1978], *Pattern Directed Inference Systems*, Academic Press, New York: .

Zadeh, L.A. [1965], "Fuzzy Sets", *Information and Control*, Vol. 8, No. 3, pp. 338-353.

Zadeh, L.A. [1978], "Fuzzy Sets as a Basis for a Theory of Possibility", *Fuzzy Sets and Systems*, Vol. 1, No. 1, pp. 3-28.