Comunicaciones a la Academia

presentadas en las Sesiones Científicas

La verificación estructural de sistemas basados en conocimiento

LUIS M. LAITA*
Académico Correspondiente

Conferencia pronunciada en la Academia
el día 5 de Abril de 1.995

1. Los sistemas basados en conocimiento. Introducción.

Todos hemos visto películas, o leído novelas y artículos, que nos presentan robots y ordenadores "inteligentes", que se comportan como humanos. Se habla de "armas inteligentes", de "edificios inteligentes", etc. Asistimos entonces a una atribución a las máquinas y objetos, de un rasgo típicamente humano, la inteligencia.

Esta atribución, desde luego exagerada, tiene sin embargo algún fundamento; la respuesta a la pregunta ¿se comportan inteligentemente las máquinas?, es parcialmente positiva.

Todo el mundo reconoce que las máquinas ya han sustituído al hombre en la gestión rápida y eficiente de cantidades ingentes de datos, como todos observamos, por ejemplo, cuando vamos a sacar dinero al banco.

Pero, curiosamente, también están sustituyendo al hombre en la realización de tareas que no sólo usan memoria, sino que se consideran típicas del comportamiento inteligente, como sacar conclusiones de premisas dadas, hacer analogías, sugerir decisiones, resolver problemas no triviales, e incluso

* Facultad de Informática. Universidad Politécnica de Madrid.

realizar descubrimientos. Ello hace que estemos inmersos sin darnos cuenta en una auténtica revolución científica y técnica que está cambiando modos de pensar y actuar.

Estas tareas "inteligentes" las realizan los ordenadores, no ciegamente ni por arte de magia sino porque el hombre ha introducido en ellos unos sistemas que han venido en llamarse "Sistemas de Conocimiento".

Hay varios tipos de Sistemas de Conocimiento, y aquí sólo se hace referencia a uno como ilustración.

El sistema consta en primer lugar, de un conjunto, que puede ser muy grande, de afirmaciones del tipo

"Si ocurren A, no B, ..., L, etc. entonces ocurre M ",

que se escribe mediante la fórmula lógica

$$"A \wedge \neg B \wedge ... \wedge L \rightarrow M."$$

Estas afirmaciones no son arbitrarias, sino el resultado de integrar cuidadosamente en asertos compactos, un conocimiento sobre un determinado campo, suministrado por entrevistas a expertos en ese campo, por consulta de libros especializados, por experiencias, etc. Este tipo de afirmaciones recibe el nombre de "reglas de producción". Además de las reglas de producción, hay otras que se llaman "restricciones" del tipo $\alpha \wedge \beta \rightarrow$ Falso, que indican que los expertos consideran que las condiciones α y β nunca pueden darse juntas.

El sistema tiene asociado un mecanismo para sacar conclusiones a partir de las afirmaciones anteriores y de algunos hechos conocidos. Este mecanismo tiene dos formas de actuar, "hacia adelante" y "hacia atrás". Veamos un ejemplo.

Sea dada la Base de reglas

1. $B \wedge D \wedge E \rightarrow F$
2. $G \wedge D \rightarrow A$
3. $C \wedge F \rightarrow A$
4. $B \rightarrow X$
5. $D \rightarrow E$
6. $C \wedge D \wedge B \wedge X \wedge A \rightarrow H$
7. $C \rightarrow D$
8. $X \wedge C \rightarrow A$
9. $X \wedge B \rightarrow D$

El "disparo hacia adelante" se produce si se dan unos hechos externos, como por ejemplo *B* y *C*.

Aplicando la regla lógica de inferencia llamada "modus ponens".

$$\frac{\alpha, \alpha \rightarrow \beta}{\beta}$$

se obtiene, de la regla 4, *B,C,X*; de la regla 8, *A,B,C,X*; de la regla 9, *A,B,C,X,D*; por último, de la 6, se obtiene *H*. *H* ha sido alcanzada por disparo hacia adelante a partir de los hechos *B* y *C*.

Ilustremos el "disparo hacia atrás" tomando como ejemplo la misma base, en la que por simplicidad cambiamos la regla 6 por la regla 6': $X \wedge A \rightarrow H$. El esquema de la figura recoge en un árbol con nodos "AND" y "OR" cómo están dispuestas las reglas.

La regla lógica que aquí se aplica es la de "Modus Tollens":

$$\frac{\neg \beta, \alpha \rightarrow \beta}{\neg \alpha}$$

Supongamos que *H* no fuera alcanzable. Por aplicación sucesiva de Modus Tollens obtendríamos $\neg X$ y $\neg B$, lo cual contradice que *B* es un hecho dado. El mismo razonamiento debe llevarse a cabo por todas las ramas hasta cerrarlas en *B* o en *C*, que son los hechos dados. Los nodos "AND" deben cerrarse todos, y para cerrar un nodo "OR" basta cerrar una de sus ramas.

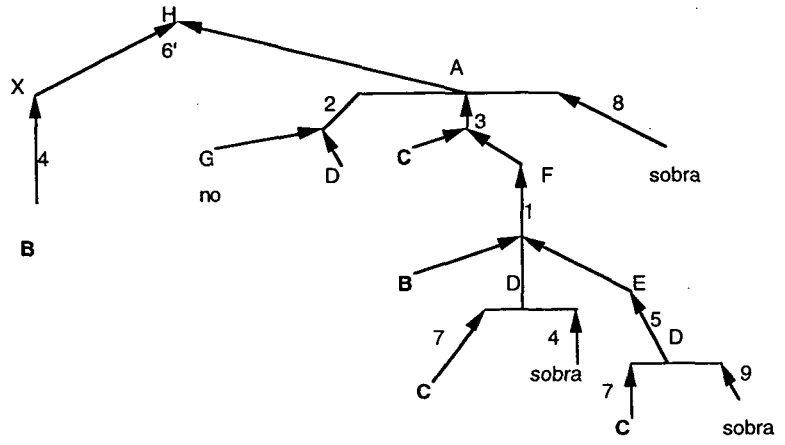

Figura 1

Es en este contexto donde aparece el problema de la verificación, con el consiguiente recurso a la lógica y las matemáticas. En este trabajo significará primordialmente detección de contradicciones en el disparo de reglas (hay otros problemas, circularidad, redundancia, subsunción, inalcanzabilidad de metas etc. que no vamos a tratar).

2. El Problema de la Verificación de Sistemas Expertos.

Durante, y después de la construcción de un sistema experto, es necesario un cuidadoso proceso de evaluación. El motivo no es trivial, ya que los sistemas expertos se están hoy día usando en el manejo y control de sistemas industriales muy complejos, en medicina, y en aeronáutica y astronáutica, por citar unos ejemplos. Si el sistema, que puede constar de miles de reglas, no está bien diseñado, entonces puede dar lugar a fallos con consecuencias dramáticas.

La Evaluación a la que nos referimos, tiene dos aspectos, la "Validación", y la "Verificación". La validación consiste en comprobar si se ha construído el sistema que cumple las especificaciones para las que fue diseñado. La verificación comprueba si el sistema está bien construído.

El problema de la verificación comenzó a tratarse cuando se presentó un sistema experto en medicina llamado MYCIN (Shorliffe 1976). Desde ese año hasta 1988 aparecieron esporádicamente algunos trabajos.

Debido al desarrollo explosivo de los sistemas expertos, y de la frecuente aparición de fallos en ellos, la comunidad científica se hizo más y más consciente de la necesidad de elaborar criterios y teorías de verificación. Ello ha hecho que a partir de 1988 se hayan celebrado seis "Workshops" de la "AAAI" (American Association for Artificial Intelligence), dos simposiums "EUROVAV" (VAV es "validation and verification), y cinco reuniones internacionales aprovechando reuniones de IJCAI (International Joint Commitee for Artificial Intelligence) y de ECAI (European Commitee for Artificial Intelligence). En total, se han presentado más de cuatrocientos artículos en siete años, lo que representa un desarrollo casi sin precedentes para tratarse de un tema muy específico.

Casi por casualidad, se nos encargó a un reducido grupo de Profesores de la Universidad Politécnica de Madrid, actuar como consultores en lógica en un proyecto ESPRIT dedicado a la validación y verificación. Desde el primer momento vimos que se trataba de un problema importante, y también vimos que, salvo en contadas ocasiones no se había abordado matemáticamente.

Se formó un grupo de Profesores de las Universidades Politécnica, Complutense, Sevilla, y Universidad Pública de Navarra, grupo que tengo el honor de coordinar, y que se ha dedicado y continúa dedicándose a la elaboración de un modelo matemático de verificación.

Se presentaba la alternativa de abordar la verificación de sistemas basados en conocimiento, bien mediante un estudio paralelo a la verificación de programas, bien mediante la propuesta de un modelo completamente nuevo. Habiendo escogido la segunda aproximación al problema, se intenta en este trabajo, tratar, en primer lugar la verificación de programas, para pasar a dar luego una descripción del modelo de verificación de Bases de Conocimiento en el que estamos trabajando. Se ha escogido un método de exposición divulgativo, sin entrar en el aparato matemático más que cuando es necesario.

3. Verificación lógica de Programas [1]

La programación convencional es más conocida por la mayoría que la programación en sistemas de conocimiento, y presentar una muestra de cómo puede construirse una teoría para verificar programas, es una manera intuitiva de mostrar que la informática en general puede y debe basarse en teorías.

Por simplicidad, los programas que se consideran acaban dando una salida en tiempo finito. Justamente antes y después de ejecutarse el programa se especifica mediante asertos los valores que tienen las variables del programa. Esos valores se denominan "estados".

Sea por ejemplo *P* el programa simple:

$$x, y := x+1,\ y-2$$

Suponemos que antes de ejecutarlo las variables tienen asignado el conjunto de estados:

$$[A] = \left[\{(x,2),(y,6)\}, \{(x,4),(y,3)\}\right]$$

Al ejecutarse *P* se obtiene el conjunto de estados:

$$[B] = \left[\{(x,3),(y,3)\}\ \{(x,5),(y,1)\}\right]$$

Un programa como P en el que se especifican los estados de entrada [A] y salida [B], denotado por "[A] P [B]", se denomina "programa anotado". [A] es una precondición y [B] una postcondición.

Es importante hacer notar que las condiciones [A] y [B] pueden reescribirse como asertos formales de un lenguaje lógico: así [A] puede reescribirse como: $\left[(x = y \wedge y = 6) \vee (x = 4 \wedge y = 3)\right]$.

En resumen, las especificaciones de los programas pueden ser consideradas como afirmaciones en la teoría de conjuntos, o como asertos de la lógica. Esta doble consideración permite un tratamiento lógico-matemático muy interesante.

Describamos brevemente ese tratamiento, para lo cual empezanmos con dos definiciones.

Definición 1. *Un programa anotado [A] P [B] es correcto cuando todo estado de [A], al ejecutarse por P da lugar a un estado de [B]; indistintamente puede decirse que [A] P [B] es correcto cuando todo estado que haga verdadero a [A] al ejecutarse por P da lugar a un estado que hace verdadero [B].* En el primer caso nos referimos a conjuntos de estados, y en el segundo a asertos lógicos.

Definición 2. *La precondición más débil, denotada por* $wp(P,R)$ *de un programa P que acaba en tiempo finito en una postcondición R, es el conjunto de todos los estados tales que al ejecutarse por P dan lugar a estados de (o que hacen verdadera a) R.*

Es inmediato demostrar, a partir de estas definiciones, *que un programa anotado* $[Q]P[R]$ *es correcto si y sólo si se verifica la fórmula de la teoría de conjuntos* $Q \subseteq wp(P,R)$ *o equivalentemente, la fórmula lógica,* $Q \rightarrow wp(P,R))$.

Este resultado traslada el problema informático de la verificación (corrección) de programas a un contexto matemático, ya que existen métodos efectivos para determinar el operador *WP* en algunos lenguajes de programación. Por ejemplo para el mandato asignación $y:=t$, con post-condición R, es, $wp(y:=t,R) = R_y[t]$, que es la fórmula que resulta de sustituir en R, la variable y por el término t.

Como simple ilustración de aplicación de esta teoría de esta vamos a estudiar el siguiente ejemplo. Se trata de examinar dos programas que aparentemente consiguen el mismo efecto: partiendo de una inicialización para las variables x,y: $x:=X$, $y:=Y$ (donde X e Y varían sobre "Verdadero" y "Falso"), se intenta llegar a la postcondición $[x=Y \wedge y = X]$, es decir, conseguir el intercambio de valores.

Primer programa: P1

$x := X$

$y := Y$

$t := x$

$x := y$

$y := x$

Segundo programa: P2

$x := X$

$y := Y$

$t := x$

$y := t$

$x := y$

La postcondición R es $x \leftrightarrow Y \wedge y \leftrightarrow X$

Hallando sucesivamente $wp(P,R)$ para los mandatos de P1, y aceptando que la precondición más débil del mandato asignación es $wp(y:=t,R) = R_y[t]$ (cambiar y por t queda)

$wp(y:=t,R) = (x=Y \wedge t = X)$,

y similarmente,

$wp(x:= y, x=Y \wedge t=X) = (y=Y \wedge t= X)$

----- etc.

Ejecutando todo el proceso se obtiene al final del mismo, como precondición, la expresión:

$$wp(P1,R) = (Y \leftrightarrow Y \wedge X \leftrightarrow X).$$

Haciendo lo mismo con el programa P2 se obtiene:

$$wp(P2,R) = (X \leftrightarrow Y \wedge X \leftrightarrow X).$$

La pregunta a proponer es entonces: ¿son P1 y P2 correctos?, y si lo son, ¿cuál de los dos es el programa recomendable?.

La precondición $wp(P1,R)=(Y\leftrightarrow Y\wedge X\leftrightarrow X)$ es siempre verdadera (V) para cualesquiera de los dos valores V (verdad) y F (falsedad), que puedan tomar X e Y. Como cualquier proposición Q verifica $Q\rightarrow V$, resulta que cualquier precondición (cualquier fórmula que contenga las variables del trozode programa examinado, *x, y, X, Y*) implica $wp(P1,R)$. Dicho de otra forma $[Q]P1[R]$ es correcto siempre para cualquier precondición Q.

En cuanto a $wp(P2,R)=(X\leftrightarrow Y\wedge X\leftrightarrow X)$, sólo es verdadera cuando X coincide con Y (cuando ambos sean V o ambos F), lo cual conlleva que $[Q]P2[R]$ es correcto sólo cuando la precondición requiera que X e Y coincidan.

Este es un ejemplo de aplicación del operador Wp y de la fórmula $Q\rightarrow wp(P,R)$, a la determinación de la corrección o incorrección de un programa (ver [1]).

Las fórmulas para el operador WP de los mandatos asignación, "Si,..., entonces", "Case" y la fórmula de la concatenación son fórmulas exactas: así, la fórmula de la concatenación de dos programas P1; P2 es:

$$Wp(P1;P2(,R)=wp\rightarrow wp(P1,\ wp(p2,R)$$

El mandato iterativo es más complicado, y aunque su operador WP puede hallarse, es tan complicado el resultado, que espreferible usar el WP para determinar caminos auxiliares, como hallar invariantes y cotas. Como ejemplo de la complejidad creciente de la fórmula de Wp, se demuestra que el operador Wp del mandato alternativo es:

$$Wp(If,R)=BB\ cand\ \big((B1\rightarrow Wp(P1\rightarrow,R)\big)\wedge\big(B1\rightarrow Wp(<p1,R)\big)...\wedge\big(B1\rightarrow Wp(P1,R)\big)$$

donde las letras "Bi" representan las guardas o condiciones del mandato alternativo, "BB" es la disyunción de ellas, y "cand" es la conjunción de una lógica trivalente que expresa la condición de estar o no definidas las guardas.

Con el ejemplo y las consideraciones subsiguientes, hemos intentado sugerir la conclusión, de que la verificación de programas no es solamente una tarea "ad hoc" sino un proceso que cuenta con una teoría lógico-algebraica que la justifica. Por supuesto, la aplicación de sólo la teoría lleva a complicaciones innecesarias, por ello es de sentido común usarla solamente cuando sea necesario o al menos útil. La verificación de sistemas de conocimiento debe estar también basada en algún tipo de teoría.

4. Una formulación algebraica para la verificación de programas [2].

En un trabajo, publicado por la Real Academia, hemos estudiado la estructura algebraica del operador WP, mostrando que un programa P es correcto si y sólo si P y WP son funtores adjuntos entre dos categorías preorden especiales, cuyos objetos son conjuntos de estados de entrada y salida.

Bajando el grado de abstracción, y siguiendo un tratamiento debido a Manes y Arbib, se pueden considerar los Programas como funciones entre un conjunto de estados de entrada y un conjunto de estados de salida.

Tomemos como ejemplo el "mandato" iterativo de los lenguajes como Pascal, "While A do P"; mientras el aserto A sea verdadero ejecutar el mandato P. P es entonces una función f.

Si A no es verdadero, es decir si el estado de entrada x no pertenece a A, el mandato no se ejecuta. Esto se puede representar por el esquema "cero iteraciones" de la figura 1.

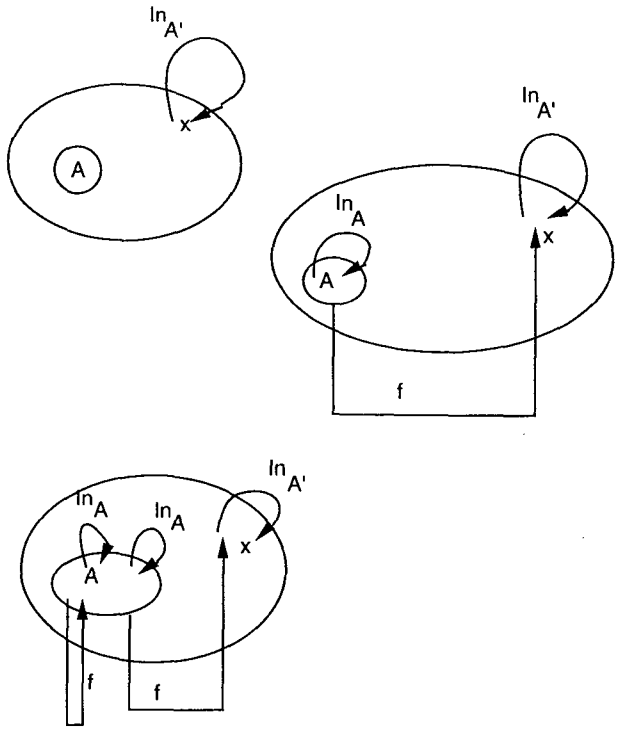

Figura 2

Nótese que hemos cambiado el elemento x por una función inclusión $in_A(x)=x$, y como sólo si x estuviera en A las iteraciones podrían continuar, el programa termina ahí. Del mismo modo si el programa realiza una o dos iteraciones, los esquemas serían los correspondientes en la figura 1.

Ello nos lleva a una fórmula

$$(While\ A\ do\ f)(x)=(in_{A'}+in_{A'}.f.in_A+in_{A'}.f.in_A.f.in_A+...)(x),\ \text{es decir,}$$

$$(While\ A\ do\ f)=\sum_{n=0}^{\infty} in_{A'}.(f.in_A)^n$$

donde los símbolos "." y + son, respectivamente, la composición de funciones, y una suma especial que corresponde a la disyunción exclusiva de la lógica, pero que cumple las propiedades de la suma ordinaria de la aritmética. De este modo se reemplazan los organigramas ordinarios a organigramas algebraicos, como muestra la figura ".

De forma parecida se pueden asignar fórmulas, conteniendo sólo productos y sumas, a todos los mandatos de los lenguajes de programación estructurada, lo cual permite pasar a esquemas del álgebra, esquemas que pertenecen a la programación.

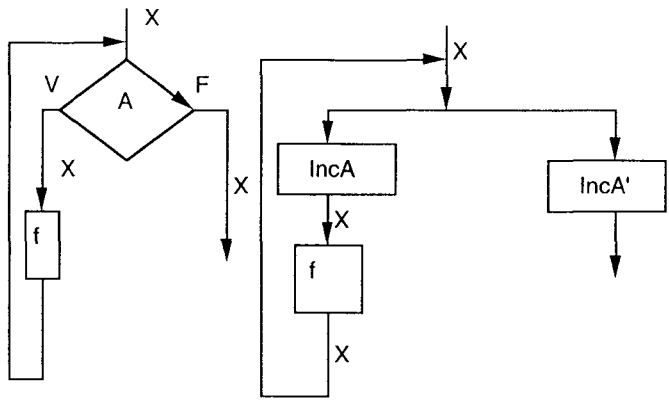

Figura 3

La fórmula que expresa la corrección de un programa anotado $\{Q\}\ P\ \{R\}$, se obtiene como sigue.

Que la condición $Q \rightarrow WP$ se cumple, es igual que decir, denotando por "F" la fórmula "While" sumatorio de arriba, $\forall x\left(x \in Q \rightarrow F(x) \in R\right)$, lo que en notación algebraica equivale a $"in_R.(F).in_Q = in_R"$.

Este tratamiento algebraico sugiere considerar las reglas de una base de conocimiento como programas "guardados" por funciones de inclusión, y aplicar las fórmulas algebraicas correspondientes. Sin embargo no hemos encontrado de momento la forma de expresar mediante programas los consecuentes de las reglas. Esta imposibilidad, junto con la práctica seguridad de que un tratamiento diferente a la verificación de programas puede ser más eficaz, nos ha llevado a presentar un modelo independiente para la verificación de sistemas expertos.

5. Verificación de sistemas expertos, introducción.

Como ilustración empezamos describiendo un método de vertificación, basado en las llamadas "redes de Petri", y propuesto por Meseguer en [6]. Aunque hay otros métodos basados en redes de Petri hemos escogido éste por su claridad y sencillez.

Una red de Petri consta de "lugares", representados por círculos, y "transiciones", representadas por líneas y arcos. Puede haber arcos desde un lugar a una transición y desde una transición a un lugar. Los lugares desde los que salen arcos a una transición se llaman "lugares entrada" a la transición, y los lugares a los que llegan arcos desde la transición se denominan "lugares salida" de la transición. Nótese que un lugar puede ser entrada y salida a la vez.

En los lugares puede haber "marcas". Se conviene en que un lugar "puede ejecutarse" cuando contiene tantas marcas (al menos) como arcos salen de él. Una transición "se dispara" cuando todos sus lugares entrada se han ejecutado. Cuando una transición se dispara se produce el siguiente efecto: aparecen en cada uno de sus lugares salida tantas marcas nuevas como arcos van a parar a él desde la transición. Entonces, en un lugar que sea a la vez entrada y salida de una transición quedan tantas marcas M' como las que había al principio M menos tantas cuantas indique el número S de arcos que van a la transición mas tantas cuantas indique el número E de arcos que vienen desde la transición: $M' = M - S + E$.

Si se considera toda la red de Petri y todos sus lugares, en vez de M se tiene una matriz de una fila $(M_1, M_2, ..., M_n)$ que contiene los marcados iniciales de todos los lugares dados en un orden. Del mismo modo se tiene en lugar de $-S+E$ una matriz D cuyo elemento a_{k1} es la diferencia entre los arcos que van desde la transición 1 al lugar k y los que van desde el lugar k a la

transición 1. El marcado final es una matriz $(M_1', M_2', ..., M_n')$ que se halla sumando $(M_1, M_2, ..., M_n)$ con una serie de productos en el que el primer elemento es una matriz de una fija ej = $(0,...,1,...0)$ con el 1 en el lugar j-ésimo, que es la etiqueta de la transición j-ésima, y el segundo elemento es la matriz D.

Vamos a utilizar estos conceptos para modelizar la verificación de bases de conocimiento.

Sea por ejemplo el sistema de reglas

$\neg A \wedge B \rightarrow X$ $\quad$ $\neg B \wedge C \rightarrow Y$

$\neg A \wedge B \rightarrow Y$ $\quad$ $D \wedge \neg A \rightarrow Z$

$A \wedge C \rightarrow X$ $\quad$ $A \wedge Y \rightarrow X$

$A \wedge C \rightarrow Y$ $\quad$ $B \wedge Y \wedge Z \rightarrow \neg X$

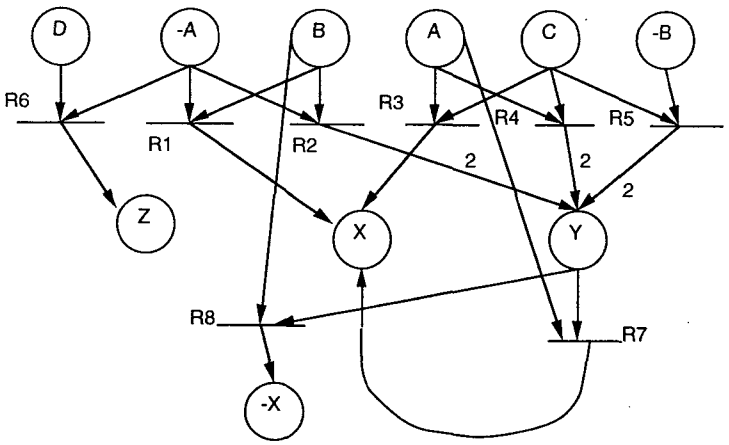

Figura 4

Cada regla se representa por una transición, y cada elemento de las premisas y conclusiones de las reglas, por un lugar (sin repeticiones). Desde cada lugar correspondiente a una premisa se traza un arco a la transición de la regla correspondiente. Desde cada regla-transición se trazan tantos arcos a la conclusión correspondiente cuantas sean las reglas que tienen esa conclusión como premisa de otra regla. Para no complicar la figura hemos marcado con "2" los arcos que van a parar a *Y*, indicando en cada caso que son dos arcos.

"Hechos posibles" son todos los elementos que apareciendo en las premisas de las reglas no aparecen en las conclusiones, en nuestro ejemplo: $A, \neg A, B, \neg B, C, D$. Con estos hechos se pueden formar los siguientes conjuntos maximales que no contengan un elemento y su negación.

$$BH_1 = \{A, B, C, D\}$$
$$BH_2 = \{A, \neg B, C, D\}$$
$$BH_3 = \{\neg A, B, C, D\}$$
$$BH_4 = \{\neg A, \neg B, C, D\}$$

La consistencia se estudia con respecto a cada uno de esos conjuntos (*BH* significa "base de hechos"). Escogiendo por ejemplo BH_1 se da un marcado inicial a cada lugar como sigue. Si se trata de un lugar correspondiente a un elemento de BH_1, el número de marcas es igual al número de arcos que salen de él mas uno: en todos los demás lugares el número de marcas es cero. Obtenemos para BH_1 el marcado $M = (4,0,4,0,4,2,0,0,0,0)$ correspondiente a $(A, \neg A, B, \neg .BC, D, X, \neg X, Y, Z)$.

Partiendo de los hechos BH_1 solo se puede llegar a inconsistencia si se obtiene $Xy\neg X$, es decir cuando en los lugares X y $\neg X$ hay marcas, por ejemplo una en cada uno.

Se obtiene entonces contradicción si se llega a un marcado final del tipo $M' = (X_1, 0, X_2, 0, X_3, X_{4,1,1,} X_5, X_6)$.

Se considera el sistema $M' = M + dD$ donde d es una suma de matrices $e(j)$ características de las transiciones (tantos sumandos como transiciones se han disparado), y D es la matriz cuyo elemento *dij* resulta de restar el número de veces en que plaza *j* es salida de una transición *i* y el número de veces en que es salida. Queda entonces,

$$M' = N + [d_1(1,0,0,\ldots,0) + d_2(0,1,0,\ldots,0) + \ldots + d_n(0,0,\ldots,1)]$$

Las variables del sistema son los elementos de d, que toman valores 0 y 1 puesto que se supone que las reglas se disparan una sola vez. La X_i son parámetros que sólo pueden tomar valores positivos al corresponder a marcados.

Si el sistema tiena solución para valores positivos de las X_i entonces se alcanza contradicción.

Existen otros métodos formales de verificación, entre los cuales nos limitamos a mencionar, el método de verificación basado en restricciones (Rousset [9], el método de verificación basado en relaciones en grafos de D.L. Nazareth y M.H. Kennedy, [7] y el método de verificación basado en clasificaciones de T.A. Nguyen. [8]

Los métodos apuntados sugieren diferentes formas de afrontar los problemas de verificación desde un punto de vista teórico. Existen también muchos métodos "ad hoc", y otros basados en otras técnicas.

Las ventajas de los métodos basados en redes de Petri, y en grafos son las siguientes. (1) Usan construcciones bien conocidas, (2) Estas construcciones son muy intuitivas ei el número de reglas no es muy grande. Sin embargo, estos métodos pueden también ser considerados como "paradigma" de las limitaciones que casi todos los métodos de verificación tienen, entre las que mencionamos, (1) La complejidad crece fuera de control cuando el sistema experto es muy grande, (2) Circularidades especialmente, pero también otros problemas de verificación diferentes al estudio de la consistencia, no pueden tratarse (3) Funcionan para Bases de Conocimiento proposicionales, (4) No contemplan incertidumbre ni control (5) La "flecha" de las reglas es la implicación material, y no se contempla la idea de que pueda haber una lógica subyacente en la que la flecha represente "relaciones relevantes" entre antecedentes y consecuentes. (6) Carecen de la semántica para expresar tiempo.

Falta por tanto por elaborar un modelo que obvie esos inconvenientes, es decir, que acepte que las reglas puedan contener predicados y variables

$$P(x_1,...,x_n) \wedge \neg Q(x_1,...,x_n) \wedge .. \rightarrow R(x_1,...,x_n) \vee ... \vee S(x_1,...,x_n),$$

y que la lógica subyacente pueda diferir de la bivalente.

Además, la complejidad de algoritmo correspondiente que hemos mencionado antes, es un problema que requiere que en cada caso se hagan simplificaciones ad hoc. Dicho de otra forma: parece difícil que se pueda construir un método de verificación que valga para todos los sistemas.

A continuación describimos, sin entrar en la formulación matemática, un modelo formal que acoge reglas que usan predicados y que es independiente del tipo de lógica subyacente al sistema. Es un método formal desarrollado por el autor de estas lineas y otros (ver [2]), y publicado en un número especial del "International Journal of Intelligent Systems", que recoge los veinte trabajos sobre Verificación que han sido juzgados más importantes de entre los más de trescientos que se han venido presentando en los "Workshops" anuales sobre verificación, de la American Association for Artificial Intelligence, celebrados desde 1988 al presente.

El conjunto de reglas se considera como el conjunto de axiomas de una teoría T. Tal teoría tiene un lenguaje formal formado por los predicados $P,Q,...,R$ de las reglas, además de por las variables $x_1,...,x_n$ y los símbolos de la lógica.

Con tal lenguaje se construye el conjunto de todas las fórmulas posibles y a este conjunto se le denomina "campo deductivo". La teoría T es un

subconjunto del campo deductivo y T contiene a su vez un subconjunto de fórmulas que son axiomas de la lógica. Este conjunto es modular en el sentido de que puede introducirse en él una lógica u otra.

Dentro del campo deductivo se definen unos conjuntos

$$E^P = \{\alpha | T \vdash p \rightarrow \alpha\}$$
$$E_q = \{\beta | T \vdash \beta \rightarrow q\}$$

donde $\vdash$ es el símbolo para indicar que $p \rightarrow a$ y $b \rightarrow q$ son teoremas en la teoría T. Estos conjuntos reciben el nombre de "m-filtros" y "m-ideales", donde "m" significa "metaalgebraico", porque reflejan, cuando T es la teoría de anillos conmutativos, las propiedades de los ideales.

Estos conjuntos se utilizan para expresar las condiciones de consistencia. Nótese que si p es una conjunción de hechos de un conjunto maximal de que no contenga un hecho y su negación. E^P expresa el conjunto de las fórmulas qaue en T son implicadas por p.

Mediante combinaciones de estos conjuntos E^P, E_q se obtienen condiciones teóricas sobre la base de conocimientos. En el trabajo mencionado [2] y en [3], se presentan varios criterios para consistencia en razonamiento adelante y un criterio para razonamiento hacia atrás.

Para simplificar nuestra descripción, nos vamos a limitar a la lógica bivalente. Si la lógica subyacente al sistema es la clásica bivalente, se puede sustituir el campo deductivo por un álgebra de Boole. Vamos a explicar este caso porque al ser más simple, puede servir de introducción al estudio del trabajo general.

Se forma primero el álgebra de todas las clases de equivalencia formadas con el lenguaje de la base de conocimiento, que para mayor simplificación, consideramos proposicional. Esta álgebra recibirá el nombre de "álgebra asociada al lenguaje de la base de conocimiento".

Suponer , para simplificar, que la base de reglas contiene una sola regla "$a \rightarrow b$". El álgebra de Boole B' asociada al lenguaje de esta base aparece en la figura, donde las flechas representan el símbolo de ordenación $\leq$.

Si ahora introducimos la regla $a \rightarrow b$ de la Base de Conocimiento, estamos afirmando que esa fórmula es cierta, es decir, $a \rightarrow b = 1$, o sea, $\neg a \vee b = 1$, con lo que queda $a \wedge \neg b = 0$.

Un elemento a de un álgebra de Boole B', distinto del "0", es un átomo si para cualquier elemento $\beta \in B$, si $\beta \leq \alpha$, entonces $\beta = 0$, o $\beta = \alpha$. Por tanto los átomos del álgebra de la figura son, $a \wedge b$, $a \wedge \neg b$, $\neg a \wedge b$, $\neg a \wedge \neg b$.

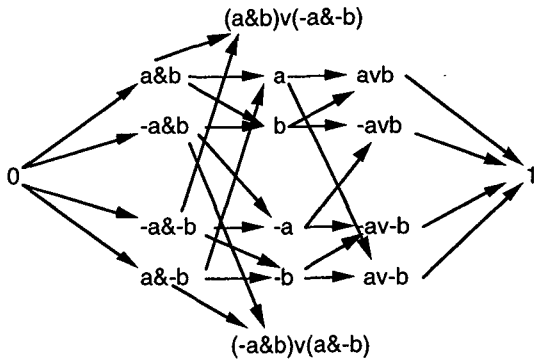

Figura 5

Entonces, al introducir $a \rightarrow b$, el átomo $a \neg b$ colapsa en el cero y desaparecen algunas flechas como se muestra en la figura, que llamaríamos "álgebra asociada a la Base de Conocimiento" (no meramente al lenguaje de la base).

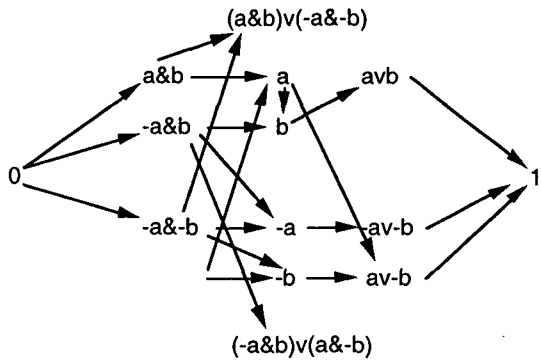

Figura 6

Si además tenemos en cuenta que de la hipótesis $a \rightarrow b$, se demuestran equivalencias tales como $a \wedge b = a$, $a \vee b = b$, resulta al final el siguiente esquema, que es el álgebra B asociada a la base de conocimientos dada $a \rightarrow b$.

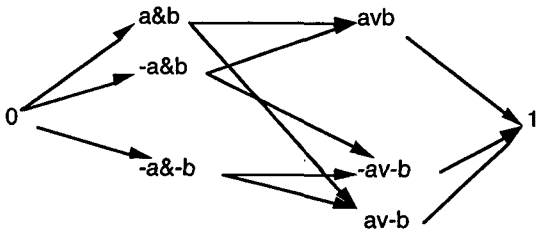

Figura 7

Si la base estuviera formada por las reglas:

$$a \rightarrow c,$$
$$b \rightarrow a,$$
$$b \rightarrow \neg c,$$

el lenguaje correspondiente sería $\{a,b,c\}$.

Con este lenguaje construiríamos el álgebra de Boole B asociada a ese lenguaje, cuyos elementos son las clases de equivalencia lógica que se pueden formar a partir de todas las fórmulas bien formadas con ese lenguaje. Como se ha dicho antes, el preorden "$\leq$", se expresan aquí mediante una flecha, que traduce la implicación. Ejemplo de "flechas" son $a \wedge b \wedge c \rightarrow\ a \wedge b \rightarrow b$.

En particular, los átomos del álgebra de Boole, que también denotaremos por la letra B, asociada al lenguaje de la base de conocimiento dada en último lugar, son:

$a \wedge b \wedge c$
$a \wedge b \wedge \neg c$
$a \wedge \neg b \wedge c$
$a \wedge \neg b \wedge \neg c$
$\neg a \wedge b \wedge c$
$\neg a \wedge b \wedge \neg c$
$\neg a \wedge \neg b \wedge c$
$\neg a \wedge \neg b \wedge \neg c.$

Cada elemento del álgebra puede escribirse, usando formas normales disjuntivas completas, como una disyunción de átomos. Así (la clase) "a" es (la clase) $(a \wedge b \wedge c) \vee (a \wedge b \wedge \neg c) \vee (a \wedge \neg c) \vee (a \wedge \neg b \wedge \neg c)$. Esto es en realidad

hacer corresponder a cada elemento a del álgebra, el conjunto de sus átomos, mediante una función $F: B \to P(A)$, donde $P(A)$ es el conjunto potencia del conjunto A de todos los átomos de B. Esta función envía cada elemento de B al conjunto de conjunciones que forman la forma normal disjuntiva completa que equivale lógicamente a ese elemento.

Considerando los átomos escritos en el orden arriba indicado, se puede asignar "códigos" $(1,0,0,0,0,0,0,0)\, a\, (a \wedge b \wedge c), (0,1,0,0,0,0,0,0)\, a\, (a \wedge b \wedge \neg c), \ldots,$ $(0,0,0,0,0,0,1,0)\, a\, (a \wedge \neg b \wedge \neg c), (0,0,0,0,0,0,0,1)\, a\, (\neg a \wedge \neg b \wedge \neg c)$ .Esta asignación permite entonces codificar "a" mediante un vector $(1,1,1,1,0,0,0,0)$, que informa qué átomos corresponden a a mediante la función f. Codificando todos los literales que aparecen en la base de conocimiento, obtendríamos:

$$\begin{array}{cc} a & (1,1,1,1,0,0,0,0) \\ \neg a & (0,0,0,0,1,1,1,1) \\ b & (1,1,0,0,1,1,0,0) \\ \neg b & (0,0,1,1,0,0,1,1) \\ c & (1,0,1,0,1,0,1,0) \\ \neg c & (0,1,0,1,0,1,0,1) \end{array}$$

Si en el álgebra B se introduce alguna fórmula α, del mismo lenguaje que el que da lugar a B, y que se considera verdadera, o lo que es lo mismo, se identifican las clases a y 1, algunos átomos dejan de serlo, y colapsan en el cero. Así por ejemplo, si en B identificamos la clase 1 con la clase $\neg a \vee c$ (que es la fórmula $a \to c$, primera regla de nuestra base de conocimiento), desaparecen los átomos segundo y cuarto. Saber qué atomos colapsan es fácil, y se determina de igual forma que en el caso en que sólo teníamos dos elementos a y b en el lenguaje, es decir, si $\neg a \vee c = 1$, entonces $a \wedge \neg c = 0$, con lo que los átomos que contienen $a \wedge \neg c$, que son precisamente el segundo y el cuarto, colapsan en el cero. Entonces $a \to c$, que es equivalente a su forma normal $(a \wedge b \wedge c) \vee (a \wedge \neg b \wedge c) \vee (a \wedge \neg b \wedge \neg c) \vee (\neg a \wedge b \wedge c) \vee (\neg a \wedge b \wedge \neg c) \vee (\neg a \wedge \neg b \wedge c) \vee$ $\vee (\neg a \wedge \neg b \wedge \neg c)$, puede codificarse por $(1,0,1,0,1,1,1,1)$. Haciendo lo mismo con las otras reglas se obtiene la siguiente codificación.

$$\begin{array}{cc} a \to c & (1,0,1,0,1,1,1,1) \\ b \to a & (1,1,1,1,0,0,1,1) \\ b \to \neg c & (0,1,1,1,0,1,1,1) \end{array}$$

Nótese que la codificación de por ejemplo la regla $a \rightarrow c$, expresada mediante la disyunción correspondiente, $\neg a \vee c$, resulta de examinar elemento a elemento los códigos de $\neg a$ y de c, y obtener un código que escriba 0 en el lugar *i* del vector código si en los lugares *i* de $\neg a$ y de c, está escrito un cero, y que escriba 1 en el lugar *j*, si en el lugar *j* de $\neg a$ y/o *c* hay un uno.

Del mismo modo se hallaría el código correspondiente a la fórmula que resulta de la conjunción, que denotaremos or "BR" (base de reglas), de las reglas de la base de conocimiento. Como antes, se determina su forma normal disyuntiva completa, que explicita qué átomos forman la fórmula, que en nuestro ejemplo da lugar al código:

$$\text{BR} \qquad (0,0,1,0,0,0,1,1)$$

Nótese que este último código resulta de multiplicar los de las tres reglas, elemento a elemento, o también de determinar qué átomos colapsan a cero examinando toda la base de reglas BR.

Los hechos $a, \neg a, \neg b$, etc. o las conjunciones de esos hechos tienen su código. Cualquiera de estos códigos que tuviera la forma $(x,x,0,x,x,x,0,0)$, con x variando sobre 0 y 1, añadido en conjunción a BR, daría lugar a una fórmula de código $(0,0,0,0,0,0,0,0)$, que indica que todos los átomos han colapsado en cero, obteniéndose el álgebra $\{0\}$. Si esto ocurre, hay conflicto en disparo hacia adelante de las reglas con respecto al (los hecho(s) dado(s).

Resumiendo, se define una función $F: B \rightarrow P(A)$, que asigna a cada elemento de *B*, los átomos que vienen determinados por la forma normal disyuntiva completa de ese elemento. Al introducir BR hay átomos que colapsan, y entonces se escriben el código correspondiente, y el código que debería tener un hecho o una conjunción de hechos para que toda el álgebra colapse a cero. Si tal hecho o conjunción existen, hay conflicto en el disparo hacia adelante.

El método descrito tiene el fundamento lógico de la equivalencia de cada fórmula con su forma normal, y el hecho de la existencia de la función *F*.

Tiene también una explicación puramente algebraica; presentamos a continuación.

6. Modelo formal

En esta sección B y B' representan las mismas estructuras que en el párrafo anterior. *F* es también la función ahí definida.

6.1. Notación.

Sea α una fórmula bien formada en el lenguaje de la base de conocimientos, E^{α} denota el filtro principal generado por α en B': $E^{\alpha} = \{\beta \in C' \mid \alpha \to \beta\}$ (la flecha representa la relación de proorden en el álgebra). Identificando α con su forma normal, y ésta con $F(\alpha)$, $E^{\alpha} = E^{F(\alpha)}$. De forma similar, el conjunto E_{α} representa el ideal principal $E_{\alpha}\left(= E_{F(\alpha)}\right) = \{\beta \mid \beta \to \alpha\}$.

6.2. Definición.

Representemos por $\alpha_1, \alpha_2, ..., \alpha_m$, las reglas y restricciones de integridad de un sistema basado en conocimiento KBS. El "álgebra de Boole asociada a KBS" es el álgebra cociente $B = B' / E^{\alpha 1 \wedge \alpha 2 \wedge ... \wedge \alpha m}$, o equivalente, $B = B' / E^{F(\alpha_1) \wedge F(\alpha_2) \wedge ... \wedge F(\alpha_m)}$.

6.3. Nota explicativa.

El ejemplo dado arriba: "$\alpha_1 = a \to b$" ilustra esta definición. Los representantes de las clases en $B = B' / E^{(a \to b)}$ son los elementos del álgebra de la figura 7. Dos fórmulas σ_1 y σ_2 pertenecen a la misma clase en $B = B' / E^{(a \to b)}$ si la "resta" $\sigma_1 \leftrightarrow \sigma_2$ está en $E^{(a \to b)}$, como por ejemplo $\sigma_1 = b$ y $\sigma_2 = a \vee b$, porque $(a \to b) \to (a \leftrightarrow a \vee b)'$.

6.4. Nota

No todas las flechas en $(a \to b) \to (a \leftrightarrow a \vee b)$ son de la misma naturaleza. La flecha que separa los paréntesis es $\leq$, mientras que las demás constituyen lo que se denomina "pseudocomplemento". El pseudocomplemento de a relativo a b es el elemento $\neg a \vee b$, y se representa por $a \Rightarrow b$. Por tanto, la fórmula de arriba es un orden $(a \Rightarrow b) \leq \left((a \Rightarrow (a \vee b)) \wedge ((a \vee b) \Rightarrow a)\right)$, entre elementos, y así, la fórmula $B = B' / E^{(a \to b)}$ es $B = B' / E^{a \Rightarrow b}$.

6.5. Consistencia en razonamiento hacia adelante

6.5.1. Nota. Para simplificar, consideramos como "hechos" para razonar, cualquier elemento que esté el antecedente de alguna regla pero que no forme parte del consecuente de ninguna regla.

6.5.2. Definición (Criterio de consistencia en razonamiento hacia adelante)

Supongamos que los símbolos $\alpha_1,\alpha_2,...,\alpha_m$, representan las reglas y las restricciones de integridad. A su vez $\{\beta_1,\beta_2,...,\beta_l\}$, representaa un conjunto consistente de hechos. La base de conocimientos es consistente en razonamiento adelante si y sólo si $E^{F(\alpha_1)\wedge F(\alpha_2)\wedge...\wedge F(\alpha_n)\wedge F(\beta_1)\wedge F(\beta_2)\wedge F(\beta_1)} \neq B'$ (es decir, si es un filtro propio en B'.

5.6.3. Nota explicativa. Si el filtro generado por la conjunción de reglas, restricciones, y hechos, coincide con B', cualquier fórmula del lenguaje de L, en particular, α y $\neg\alpha$, están inmplicadas por la mencionada conjunción (recordar la definición de filtro), lo cual significa inconsistencia. Nótese que en tal caso, si $E^{F(\alpha_1)\wedge F(\alpha_2)\wedge...\wedge F(\alpha_n)\wedge F(\beta_1)\wedge F(\beta_2)\wedge...\wedge F(\beta_1)} = B'$, entonces, $F(\alpha_1)\wedge F(\alpha_2)\wedge...\wedge F(\alpha_n)\wedge F(\beta_1)\wedge F(\beta_2)\wedge...\wedge F(\beta_1)=0$. La conjunción de reglas, restricciones, y hechos lleva a la contradicción. Esto es lo que justifica el algoritmo que hemos expuesto informalmente arriba. En este algoritmo y su justificación, el autor de estas líneas ha trabajado con los Profesores Angélica de Antonio, Aurora Pérez, y Luis de Ledesma.

Interpretación algebraica del modelo formal.

Esta sección se desarrolla con detalle en [4] y [5], aquí damos solamente un resumen.

Se considera la *k*-álgebra $A=R/I$ donde R es el anillo $(Z/Z2)[x1,x2,...,xn]$, e I es el ideal $<x1^2-x1,...,xn^2-xn>$ generado por $x1^2-x1,...,xn^2-xn$. Añadiendo una operación "$+^*$" definida así: $a+^*b=a+b-ab$, un preorden definido "$a\leq b$ si y sólo si $a.b=a$", y un complemento $1+a$ de a, A adquiere estructura de un álgebra de Boole.

Si B' representa el mismo conjunto que en la sección anterior, la función,

$$\Phi:(B',\vee,\wedge,\neg,\rightarrow)\rightarrow(A,+^*,...,1+,\leq),$$

definida así: si $\Phi(A)=a,\Phi(B)=b$, entonces $\Phi(A)=1+a$, $\Phi(A\vee B)=a+^*b$, y $\Phi(A\wedge B)=a.b$, es un isomorfismo que preserva el orden.

Se puede demostrar que Φ preserva los ideales, y que, además estos ideales son exactamente los de la *k*-álgebra A.

Por otro lado una fórmula del tipo regla o restricción, $a\rightarrow b(\neg a\vee b)$, se traduce por $a.(1+b)$, y un hecho $(d\leftrightarrow 1$ ó $\neg d\leftrightarrow 0)$ por $d+1$. Entonces, a un

ideal del tipo E_α, donde α es el complemento de la conjunción de reglas y restricciones $a \rightarrow b$ y hechos *d*, es decir, es de la forma:

$$E_{<\neg \text{regla } 1 \vee \neg \text{regla } 2 \vee \ldots \vee \neg \text{restr } 1 \vee \neg \text{restr } 2 \vee \ldots \neg \text{hecho } 1 \vee \neg \text{hecho } 2 \vee \ldots >}.$$

le corresponde mediante Φ un ideal del tipo:

$$<\left(\Phi(\text{regla}1)+1\right),\left(\Phi(\text{regla}2)+2\right),\ldots,\left(\Phi(\text{restr}1)+1\right),\left(\Phi(\text{restr}2)+1\right),\ldots,\left(\Phi(\text{hecho}1)+1\right),$$
$$\left(\Phi(\text{hecho}2)+1\right),\ldots>.$$

Recordemos que si $E_\alpha = B'$ hay contradicción en la base de reglas a partir de las restricciones y hechos dados. Esto se traduce algebraicamente a la condición

$$\left(\left(\Phi(regla1)+*\left(\Phi(regla2)+2\right)+*\ldots+*\left(\Phi(restr1)+1\right)+*\left(\Phi(restr2)+1\right)+*\ldots\left(\Phi(hecho1)+1\right)+*\ldots\right.\right.$$
$$\left.\left(\Phi(hecho2)+1\right)+*\ldots\right)=1.$$

Lo más importante de esta descripción algebraica es que permite una "implantación" en los lenguajes de computación algebraica REDUCE y MAPLEV.

Referencias

[1] GRIES, D., *The Science of Programming*. Springer-Verlag 1981.

[2] LAITA, L.M., COUTO, J., LEDESMA, L., FERNÁNDEZ MARGARIT, A.; *A formal model for Knowledge Based Systems Verification*. International Journal of Intelligent Systems, 9, 9, 769-786, 1994.

[3] LAITA, L.M., RAMÍREZ, B., DE LEDESMA, L., RISCOS, A., *A Formal Model for Verification of of Dynamic Consistency of KBSs*. Computers and Mathematics, with Applications, 29, 5, 81-96, 1995.

[4] LEDESMA, L., ROANES LOZANO, E., ROANES MACÍAS, E., *An interpretation of the propositional Boolean Algebra as a k-algebra, effective calculus*. J. Cammpbell y J. Calmet (eds),Proceedings of the Second International Conference on Artificial Intelligence and Symbolic Mathematical Computing. AISMC-2. Springer Verlag (en prensa).

[5] Verification of Knowledge Based Systems: an Albraic Interpretation. Effective Calculus. Proceedings of the Workshop on Validation and Verification, International Joint Committe for Artificial Intelligence, 1995. En prensa.

[6] MESEGUER, P., Detección de Inconsistencias en bases de reglas utilizando redes de Petri. *Actas de la III reunión AEPIA-89*, 1989, pp. 81-89.

[7] NAZARETH, D.L., KENNEDY, M.H., *Static and dynamic verification of rule-based systems using digraphs*. Preprint School of Business Administration, U. of Wisconsin-Milwankee.

[8] NGUYEN, A., and al Knowledge Bases Verification. *AI Magazine*, Summer 1987 pp. 67-75.

[9] ROUSSET, M.C., On the consistency of Knowledge Bases: The Covadis System. *Proceedings of EAIC-88*, Pitman, 1988.

Bibliografía sobre Verificación de Bases de Conocimiento

AGARWAL, R., & TANNIRU, M., (1992). A Petri-net based approach for verifying the integrity of production systems. International Journal of Man-Machine Studies 36 (3), 447-468.

AGUSTI-CULLEL, J., ESTEVA, F., GARCÍA, P. and GODÓ, L., (1990). Formalizing multiple-value logics as institutions, in Uncertainty in Knowledge Bases. Bouchon-Meunier,. Yager, and Zadeh, Eds. Lecture Notes in Computer Science. Springer-Verlag. Berlin, 521.

ANDERT, E.P., (1992). Integrated knowledge-based system design and validation for solving problems in uncertain environments, International Journal of Man-Machine Studies, 36 (2), 357-373.

AYEL, M., (1988). Protocols for Coherence Checking in Expert System Knowledge Bases. Proc. of ECAI-88, Munchen, 220-225.

AYEL, M., & LAURENT, J.P., (1989a). Coherence testing of knowledge bases: Congress Applications of Artificial Intelligence VII, March 1989, Orlando.

AYEL, M., & LAURENT, J.P., (1989b). Off-line Coherence Checking for Knowledge Based Systems. Second Workshop on Validation, Verification & Test, IJCAI-89.

AYEL, M., & ROUSSET, M.C., (1990). La coherence dans les bases de connaissances. Cepadues Editions.

AYEL, M., & LAURENT, J.P., Eds., (1991). Validation, Veriification and Test of Knowledge-based Systems, New York: John Wiley and Sons.

BECKER, L., DUKWORTH, J., LAZNOVSKI, A., GREEN, P., (1994). Automated Test Generation and Evaluation for Real-Time Expert Systems. International Journal of Intelligent Systems, 9 (8), 659-682.

BELANCHE, L, CORTÉS, U., (1991). The Nought Atributes in Knowledge Based Systems. EUROVAV-91, 77-102.

BELLMAN, K., Ed., (1989). Proceedings of the AAAI-89 Workshop on Knowledge-Based System Verification, Validation, and Testing, St. Paul, Minnesota.

BROWNSTON, L., FARREL, R., KENT, E., & MARTIN, N., (1985). Programming Expert Systems in OPS5. Addison-Wesley.

BUCHANAN, B.G., & SHORTLIFFE, E.H., (1984). The problem of evaluation. In B.G. Buchanan & E.H. Shortliffe (Eds.), Rule-Based Expert Systems, Reading, Massdachusetts: Addison Wesley, 571-588.

BYLANDER, T. and CHANDRASEKARAN, B., (1987). Generic tasks for knowledge-based reasoning: The 'right' level of abstraction for knowledge acquisition. Int. J. International Journal of Man-Machine Studies. 26, 231-244.

CHANDRASEKARAN, B. (1983). On evaluating AI systems for medical diagnosis, AI Magazine, 4 (2), 34-38.

CHANDRASEKARAN, B., (1986). Generic tasks in knowledge-based reasoning: Highlevel building blocks for expert system design. IEEE Expert, 1, 23-30.

CLANCEY, W.J., and BOCK, C., (1985). Representing Control Knowledge as Abstract Tasks and Metarules. Stanford Knowledge Systems Laboratory Memo No. KSL 85-16.

CRAGUN, B.J., & STEUDEL, H.J., (1987). A decision-table-based processor for checking completeness and consistency in rule-based expert systems, International Journal of Man-Machine Studies, 26 (5), 633-648.

CULBERT, C., Ed., (1990). Proceedings of the AAAI-90 Workshop on Knowledge-Based System Verification, Validation, and Testing, Boston, Massachusetts.

DAVIS, R., (1976). Use of Meta Level Knowledge in the Construction and Maintenance of Large Knowledge Bases, Ph. D. dissertation, Computer Science Department, Stanford University, Stanford, California.

EVERTZ, R., MOTTA, E. (1991). The abstract interpretation of Hybrid Rule/Frame-Based Systems, EUROVAV-91, 39-54.

EUROVAV 91 (P. Jenkins of Logica LTD, UK and E. Plaza of CEAB, Spain, Eds), (1991). The following is a list of the authors of articles in the Proceedings. (i) Theoretical Foundations: T. Hoppe, T. and P. Meseguer, C. Martin-Mattei, L. Laita, J. Couto, and L. de Ledesma, R. Evertz and E. Motta, D. Pearce, S. Talbot and M. Ayel, L. Belanche and U. Cortés, M. van Someren. (ii) Tools and Techniques. S. Petitjean, L. Brunessaux, and J. Vaudet, T. Bench-Capon and F. Coenen, O'Mengshoel, F. Polat and H. Guvenir, P. Meseguer, V. Canivell and A. del Moral, N. Shadbolt, C. Sierra, J. Agustí-Cullell, and E. Plaza, M. Sebag and M. Shoenauer. (iii) Industrial requirements. M. Grisoni and H. Howells, P. Krause, L. Brunessaux, W. Berendt and S. Lambert, D. Byrne and J. Dewar, and P. Newcomb.

EUROVAV-93 (Cardeñosa, J., of UPM. and Meseguer, P. of CEAB, Spain, Eds), (1993), Contains twenty two articles divided in six sections, plus two guest papers. The following is a list of the authors.
(i) Guest Papers. J. Liebowitz, B. Wielinga.
(ii) ESPRIT Projects on Validation. J. Cardeñosa and N. Juristo, A. Rouge, J.L. Lapique, F. Brossier, and Y. Lozinguez, S. Craw,
(iii) Validation Methods and Techniques. J. Casamayor, F. Marqués, and H.A. Decker, G. Antoniou and V. Sperschneider, B. Wenderler, L. Vignolet, and L. Talbot, S. Renault.
(iv) Knowledge Modelling and Validation. C. Haouche and I. Lamsade, C. Duursma, and R. Schrooten. D. Cañamero, S. Geldof, and A. McTyre.
(v) Formal Approaches to Validation, L.M. Laita, L., Ledesma, A. Pérez, B. Ramírezx, H. Herre, P. Hors and M.C. Rousset, D. Kinkielele.
(vi) Validation and Machine Learning. H. Lounis, D. Borrajo and A. de Antonio.
(vii) Industrial Approaches to Validation. R. Phelps, and W. Aerts, W.C. Vicat, P. Brezillon, and C. Nottola, R. Blondeau, L. Gibet.

FONTAINE, D., LE BEUX, P., & STRAUSS, A., (1988). Un systeme interactif pour le maintienance de la coherence d'une base de regles. In Proceedings Les Systemes Experts et leurs Applications, Avignon, 49-61.

GASCHNIG, J.,, KLAHR, P., POPLE, H., SHORTLIFFE, E.H. & TERRY, A., (1983). Evaluation of expert systems: Issues and case studies. In F. Hayes-Roth, D.A. Waterman, & P. Klahr (Eds.), Building Expert Systems, Reading, Mass. Addison Wesley.

GEORGEFF, M.P., (1982). Procedural control in production systems. Artif. Intell. 18, 175-201.

GINSBERG, A., (1987). A new approach to checking knowledge bases for inconsistency and redundancy. In Proceedings of the Seventh Annual National Conference on Artificial Intelligence (pp. 585-589). Minneapolis.

GINSBERG, A., (1988). Knowledge-Base Reduction: A New Approach to Checking Knowledge Bases for Inconsistency & Redundancy. Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI'88, St. Paul, MN), 585-589.

GINSBERG, A., SHOLOM, M., & POLITAKES, P., (1988). Automatic knowledge base refinement for classification systems. Artificial Intelligence, 35, 197-226.

GOLD, D.I., PLANT, R.T., (1994). Towards the Formal Specification of an OPS5 Production System Arquitecture. International Journal of Intelligent Systems, Vol. 9, 8, 739-768.

GREEN, C.J., & REYES, M.M. (1987). Verification and validation of expert systems, Proceedings of the Western Conference on Expert Systems (WESTEX-87), Anaheim, California, pp. 38-43.

GROUNDWATER, E.H., DONNELL, M.L., & ARCHER, M.A., (1987). Approaches to verification and validation of expert systems for nuclear power plants. Technical Report NP-5236, Electric Power Research Institute.

GUPTA, U., De., (1991). Validating and Verifying Knowledge-Based Systems, IEEE Computer Society Press, Los Alamitos, California.

HAMILTON, D., KELLEY, K., CULBERT, C., (1991) KBS V&V-State of -the-Practice and implications for V&V standards, Proc. of the 1991 Workshopp on Knowledge-Based Systems Verification, Validation, and Testing, Cal.

HIGHLAND, F., KORNMAN, B., (1994). A Design Language and the Use of Cleanroom Methodology for Knowledge-Based System Development. International Journal of Intelligent Systems, vol. 9 (9), 787-808.

HOOPE, T. and MESEGUER, P., (1991). "On the terminology of VVT". European Workshop on the Verification and Validation of Knowledge Based Systems. EUROVAV-91, Cambridge UK.

JACOB, R.J.K., & FROSCHER, J.N., (1990). A software engineering methodology for rule-based systems. IEEE Transactions on Knowledge and Data Engineering, 2 (2), 173-189.

DE KLEER, J., (1986). An assumption-based truth-maintenance system (TMS). Artificial Intelligence, 28, 127-224.

KLINE, P.J. and DOLINS, S.B., (1986). Problem features that influence the design of expert systems. Proc. AAAI86. Los Angeles, 956-962.

KRAUSE, P.J., BYERS, P.,, HAJNAL, S. and FOX, J., (1990). The use of object-oriented process specification for the verification and validation of decision support systems, in ECAI'90 Workshop on Validation, Verification and Test of KBS. Stockholm, Sweden.

LAITA, L.M., COUTO, J., LEDESMA, L., FERNÁNDEZ MARGARIT, A., (1994). A Formal Model for Knowledge-Based Systems Verification. International Journal of Intelligent Systems, vol. 9, 9, 769-786.

LAITA, L.M., (1995). Verification of Knowledge Based Systems, Encyclopedia of Computer Science and Technology. Por aparecer próximamente.

LANE, N.E., (1986). Global issues in evaluation of expert systems, Proceeding of the 1986 IEEE International Conference on Systems, Man, and Cybernetics, Atlanta, Georgia, 121-125.

LAURENT, J.P., (1992). Proposals for a valid terminology in KBS validation. Proceedings of the European Conference on Artificial Intelligence, Vienna, 829-835.

LIEBOWITZ, J., (1986). Useful approach for evaluating expert systems, Expert Systems, 3 (2), 86-96.

LIU, L.K., & DILLON, T., (1991). An approach towards the verification of expert systems using numerical Petri nets. International Journal of Intelligent Systems, 6, 255-276.

LOISEAU, S., (1989). La description et la détection des incohérences dans les bases de règles. Proceedings of the International Conference on Expert Systems and their Applications, Avignon, 231-246.

MAES, R., & VAN DIJK, J.E., (1988). On the Role of Ambiguity and Incompleteness in the Design of Decision Tables and Rule-Based Systems. The Computer Journal, 31 (6), 481-489.

MAHABALA, H.N., RAVI PRAKASH, G., et al., (1987). Expert System for Selection of Drill: A Case Study four use of Metaknowledge and Consistency Checks. Proceedings of the II International Conference on Applications of AI in Engineering, Boston, August, 371-386.

MAREK, W., (1986). Completeness and consistency in knowledge based systems, Proceedings of the First International Conference on Expert Database Systems, Clarleston, South Carolina, 119-126.

MEHROTRA, M., (1991). Rule groupings: a software engineering approach towards verification of expert systems. NASA Contractor Report 4372, Washington, DC.

MESEGUER, P. (1990). A new method to checking rule bases for inconsistency: A petri net approach. Proceedings of ECAI90, Stockholm, 437-442.

MESEGUER, P., (1991b). Verification of Multi-Level Rule-Based Expert Systems. Research Report, IIIA-91/10, CEAB.

MESEGUER, P., (1992). Incremental Verification of Rule-Based Expert Systems. Proceedings of the ECAI'92, 840-844.

MESEGUER, P., VERDAGUER, A., (1993). Verification of Multi-Level Rule-Based Expert Systems, Theory and Practice. International Journal of Expert Systems, 6 (2), 163-192.

MESEGUER, P., PLAZA, E., (1994). The VALID Project: Goals, Development, and Results. International Journal of Intelligent Systems, (9), 9, 867-892.

MILLER, L.A., Ed., (1992). Procveedings of the AAAI-92 Workshop on Knowledge-Based System Verification, Validation, and Testing, San José, California.

MORELL, L.J., (1987). Use a metaknowledge in the verification of knowledge-based systems. Proceedings of the IEA-AEI, 847-857.

MURATA, T. and MATSUYAMA, K., (1988). Inconsistency check of a set of clauses using Petri net reductions, Journal of the Franklin Institute, 325 (I), 73-93.

NAZARETH, D.L., (1989). Issues in the verification of knowledge in rule-based systems, International Journal of Man-Machine Studies, 30 (3), 255-271.

NAZARETH, D.L., (1991). Investigating the applicability of Petri nets for rule-based system verification, Working Paper, School of Business Administration, University of Wisconsin-Milwaukee. To appear in IEEE Transactions on Knowledge and Data Enrineering.

NAZARETH, D.L., & KENNEDY, M.H., (1991). Verification of rule-based knowledge using directed graphs, Knowledge Acquisition, 3, 339-360.

NAZARETH, D.L., & KENNEDY, M.H. (1993). Knowledge-Based System Verification, Validation, and Testing: the Evolution of a Discipline, International Journal of Expert Systems, 6 (2), 143-162.

NGUYEN, T.A., PERKINS, W.A., LEFFEY, T.J. & PECORA, D., (1985). Checking expert system knowledge bases for consistency and completeness, Proceedings of the IJCAI85, Los Angeless, California, 375-378.

NGUYEN, T.A., (1987). Verifying consistency of production systems, Proceedings of the IEEE87 Conference on Artificial Intelligence Applications, Orlando, Florida, 4-8.

O'KEEFE, R.M., BALCI, O., 6 SMITH, E.P., (1987). Validating expert system performance, IEEE Expert, 2 (4), 81-90.

O'LEARY, D.E., (1987). Validation of expert systems with applications to auditing and accounting expert systems. Decision Sciences, 18 (3), 468-486.

O'LEARY, D.E., (1988). Methods of validating expert systems, Interfaces, 18 (6), 72-79.

O'LEARY, D.E., (1990). Verification of Frame and Semantic Network Knowledge Bases. Proc. of the 5th Knowledge Acquisition for Knowledge Based Systems Workshop. Banff. Canada.

O'LEARY, D.E., Ed., (1991). Proceedings of the AAAI-91 Workshop on Knowledge-Based System Verification testing, Anaheim, California.

O'LEARY, D.E., (1994). De Verification and Validation of Intelligent Systems: Five Years of AAAI Workshops. International Journal of Intelligent Systems, 9 (8), 653-659.

O'KEARY, D.E., (1994), Towards a Theory of Verification and Validation: Artifacts. International Journal of Intelligent Systems, 9 (9), 853-866.

PIPARD, E., (1988). Detection d'inconsistances et d'incomplétudes dans les bases de régles: le système INDE. Les Systèmes Experts et leurs Applications, Avignon, 13-23.

PLANT, R. Ed., (1994). Proceedings of the 1994 AAAI Workshop on Knowledge-Based System Verification, Validation and Testing. Seatle.

PLAZA, E., (1989). KAIROS: a framework for specifying control and metacontrol in indeterministic declarative systems. GRIAL research report.

PREECE, A.D., De., (1993). Proceedings of the AAAI-93 Workshop on Knowledge-Based System Verification testing, Washington.

PREECE, A.D., and SUEN, C.Y., (1993). Guest Editors for the Special Issue on Verification and Validation of the International Journal of Expert Systems 6, (2 and 3).

PREECE, A.D., SHINGHAL, R., (1994). Foundation and Application of Knowledge Base Verification. International Journal of Intelligent Systems, 9 (8), 683-702.

PUURONEN, S., (1987). A Tabular Rule-Checking Method. Seventh International Workshop on Expert Systems and their Applications, 257-268.

RAVI PRAKASH, G., SUBRAHMANIAN, E., & MAHABALA, H.N., (1991). A Methodology for Systematic Verification of OPS5-based AI Applications. Proceedings of IJCAI-91, Sydney, 3-8.

RAVI PRAKASH, G. MAHABALA, N.H., (1993). SVEPOA, A Tool to Aid Verification and Validation of OPS5-Based AI Applications. International Journal of Expert Systems, 6 (2), 143-162.

ROUSSET, M.C., (1987). Sur la Validité de bases de coinnaisances: le système COVADIS. Proc. Les systèmes experts et leurs applications, Avignon, 269-282.

ROUSSET, M.C., (1988a). On the consistency of knowledge bases: the COVADIS system. In European Conference on Artificial Intelligence, ECAI 88, Munich 79-84.

ROUSSET, M.C., (1988b). "On the consistency of knowledge bases: The COVADIS system". Computat. Intell. 4, 166-170.

SHAW, M.L.G., & WOODWARD, J.B., (1988). Validation in a knowledge support system: consistency and construing with multiple experts. International Journal of Man-Machine Studies, 29, 329-350.

SHIRLEY, R.S., (1987). Some lessons learned using expert systems for process control. In Proceedings of the American Control Conference, 2, 1342-1346.

SHORTLIFFE, E.H., (1976). Computer-Based Medical Consultations: MYCIN, New York: Elsevier.

SIERRA, C., (1989). MILORD: Arquitectura multinevell per a sistemes experts en classificacio, PhD dissertation. Universitat Politécnica de Catalunya.

STACHOWITZ, R.A., COMBS, J.B., & CHANG, C.L., (1987). Validation of Knowledge-Based Systems. Proceedings 2nd AIAA/NASA/USAF Symposium on Automation, Robotics and Advances Computing for the National Space Program 1-10.

SUWA, M., SCOTT, A.C., & SHORTLIFFE, E.H., (1982). An approach to verifying completeness and consistency in a rule-based expert system, AI Magazine, 3 (4), 16-21.

TALBOT, S., AYEL, M., (1991), Consistency and Rules subject to Exceptions, EUROVAV-91, 69-76.

TERANO, T., (1992). A case study of expert system evaluation using a checklist-based guideline. Proc. AAAI-92 Workshop on VV&T Expert Systems. San José, CA.

TERANO, T. and KOBAYASHI, S., (1989). Problem analyses, tool evaluation, and verification & validation study: Three steps for knowledge-based systems development methodology. IJCAI-89 Workshop on VV & T of Knowledge-Based systems. Detroit, MI.

VALIENTE, G., (1992). On knowledge bases redundancy under uncertain reasoning. In Proceedings International Conference on Information Processing.

VALIENTE, G., (1993). Verification of Knowledge Base Redundancy and Subsumption using Graph Transformations. International Journal of Expert Systems, 6 (3), 341-356.

VERDAGUER, A., (1989).PNEUMON-IA: Devolupament i validacio d'un sistema expert d'ajuda al diagnostic medic. PhD dissertation. Universitat Autónoma de Barcelona.

WIELINGA, B.J., TH. SCHREIBER, A. and BREUKER, J.A., (1992). KADS: A Modelling Approach to Knowledge Engineering. Knowledge Acquisition, 4, 5-53.

YAGER, R.R. and LARSEN, H.L., (1991). On discovering potential inconsistencies in validating uncertain knowledge bases by reflecting on the input. Proceedings of the AAAI91 VV&T Woorkshop.

ZLATREVA, N.P., (1994). A Framework for Verification, Validation, and Refinement of Knowledge Bases: The VVR System. International Journal of Intelligent Systems, 9 (8), 703-738.

Reconocimientos. Deseo agradecer el importante apoyo recibido de los Excelentísimos Sres. Académicos, D. Darío Maravall, D. José A. Girón, D. Miguel de Guzmán, D. Gregorio Millán, y en especial, de D. Sixto Ríos García. Agradezco a la Real Academia de Ciencias la oportunidad que se me ha concedido para exponer este trabajo.