CAPITULO 1

REPRESENTACIONES FORMALES DEL CONOCIMIENTO

- **Aspectos Generales de la Representación del Conocimiento**
- **Lógica de Proposiciones y Lógica de Predicados**
- **Ingeniería del Conocimiento y Lógica Formal**
- **Evaluación y Resolución en Lógica Formal**
- **Introducción a Otras Lógicas**
- **Resumen**
- **Textos Básicos**

1. REPRESENTACIONES FORMALES DEL CONOCIMIENTO

Uno de los puntos que habíamos sólo esbozado en el capítulo anterior era el de la representación del conocimiento. Resulta evidente que si un programa de inteligencia artificial debe explotar eficazmente un conjunto determinado de conocimientos de un dominio concreto, al menos parte de la potencia de dicho programa vendrá determinada por la eficacia y la consistencia del esquema que hayamos elegido para representar el conocimiento.

1.1. Aspectos Generales de la Representación del Conocimiento

En cualquier dominio de aplicación nos vamos a encontrar siempre con dos tipos de entidades diferentes:

Los hechos, o verdades del dominio
Las representaciones de los hechos

Las representaciones de los hechos son las estructuras internas que los programas de inteligencia artificial manipulan, y se corresponden con las verdades del dominio.

Para manipular computacionalmente "hechos", necesitamos definir un conjunto de procedimientos que conviertan tales hechos en representaciones. Una vez ejecutado nuestro programa de IA, necesitamos nuevos procedimientos que conviertan las representaciones internas en hechos comprensibles para nosotros. El proceso global configura lo que se denomina fase de *codificación – decodificación* (Figura 1.1)

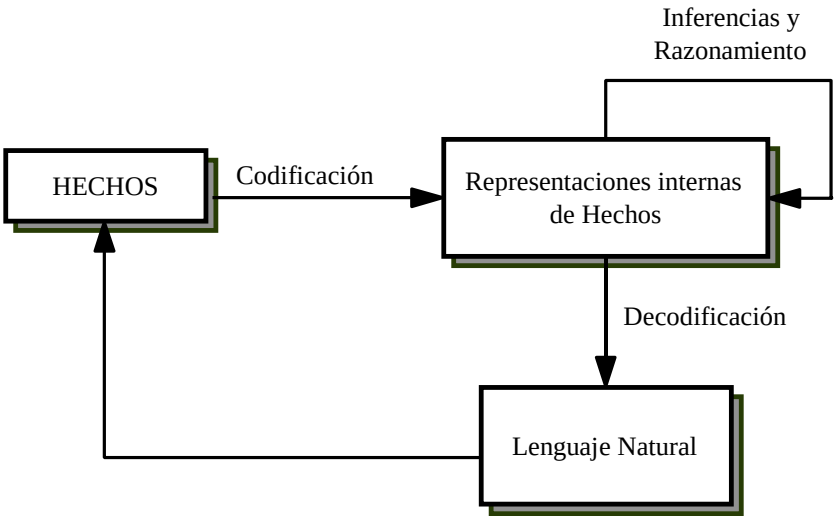


Figura 1.1 El ciclo básico de codificación – decodificación

Cualquier procedimiento para representar conocimiento tiene que reunir un conjunto mínimo de condiciones, entre las que destacamos:

Transparencia: Concepto que hace referencia a si podemos o no identificar fácilmente el conocimiento representado.

Naturalidad: Concepto que hace referencia a si podemos o no representar el conocimiento en su forma original.

Claridad: Concepto que hace referencia a si podemos o no representar directamente el conocimiento.

Eficiencia: O facilidad relativa con la cual se puede acceder a conocimientos específicos durante la ejecución.

Adecuación: O capacidad del esquema de representación elegido para representar todos los conocimientos y tipos de conocimiento que requiere el sistema.

Modularidad: O capacidad del esquema de representación para fragmentar los distintos tipos de conocimiento del sistema.

Al margen de otras consideraciones, hemos de tener presente que el conocimiento involucrado en los distintos procesos de razonamiento puede estar detallado de muy diversas formas.

Así, hablamos de "principios primarios", o bloques de construcción básicos sobre los que se basa el dominio en cuestión. A partir de tales principios deben poder ser desarrollados otros principios más específicos, nuevos teoremas y reglas de acción. Éstos, a su vez, forman la base necesaria para derivar conocimientos adicionales.

Un aspecto importante a tener en cuenta es la flexibilidad con la que podemos manejar el conocimiento. Como regla general, cuanto mayor sea el nivel del conocimiento barajado, menor será su flexibilidad. El conocimiento de alto nivel es potente pero poco flexible. Por el contrario, el conocimiento de bajo nivel es flexible pero poco potente.

Los esquemas de representación del conocimiento existentes pueden clasificarse en una cualquiera de las siguientes categorías:

Métodos Declarativos
Métodos Procedimentales

Los esquemas declarativos hacen énfasis en la representación del conocimiento como una acumulación de hechos estáticos, a los que se añade cierta información limitada que describe cómo se va a emplear el mencionado conocimiento. Por el contrario, los esquemas procedimentales enfatizan la representación del conocimiento en forma de estructuras dinámicas, que describen procedimientos de utilización de los conocimientos. En realidad, los problemas interesantes de inteligencia artificial suelen requerir distintas proporciones de ambas filosofías en la representación del conocimiento del dominio.

1.2. Lógica de Proposiciones y Lógica de Predicados

Ambas, la lógica de proposiciones y la lógica de predicados, se suelen englobar en lo que podríamos llamar "lógica formal". Como esquema de representación del conocimiento, la lógica formal permite derivar conocimiento a partir de conocimiento ya existente a través de procesos deductivos. De este modo, en lógica formal una aseveración es cierta si podemos demostrar que se puede deducir a partir de otras aseveraciones que se sabe que son ciertas.

La más sencilla de las lógicas formales es la *lógica de proposiciones*, en la que los hechos del mundo real se representan como proposiciones lógicas, que son *fórmulas bien definidas* o *fórmulas bien formadas* (respectivamente, FBDs o FBFs).

Por ejemplo, en lógica de proposiciones, la declaración "Sócrates es un hombre" se denota por "SOCRATESHOMBRE". Análogamente, la declaración "Aristóteles es un hombre" se denota por "ARISTOTELESHOMBRE". Aunque más adelante volveremos sobre la representación del conocimiento mediante lógica de proposiciones, parece claro que, por construcción, la lógica de proposiciones presenta varios problemas. Así, ¿cómo podemos representar eficazmente varios ejemplos de una misma entidad? ¿cómo podemos resolver el problema de la cuantificación?

La representación del conocimiento mediante lógica de proposiciones no es versátil. Surge así la necesidad de utilizar, siempre desde la óptica formal, la *lógica de predicados*.

En lógica de predicados el conocimiento se representa como declaraciones lógicas que son FBFs o FBDs. Pasamos así de estructuras del tipo "SOCRATESHOMBRE" a estructuras del tipo "hombre(SOCRATES)".

Los componentes básicos de un esquema de representación del conocimiento basado en lógica de predicados son:

Alfabeto
Lenguaje formal
Conjunto de enunciados básicos o axiomas
Reglas inferenciales

Al respecto, los axiomas describen fragmentos de conocimiento, y las reglas inferenciales se aplican a los axiomas para tratar de deducir nuevos enunciados verdaderos.

Alfabeto

En cualquier lenguaje formal, el alfabeto es el conjunto de símbolos a partir de los que se construyen los enunciados. En lógica de predicados el alfabeto está constituido por los siguientes elementos:

Predicados
Variables

Funciones
Constantes
Juntores
Cuantificadores
Delimitadores

Los predicados representan relaciones en el dominio de discurso. Un predicado es verdadero si los elementos involucrados verifican la relación especificada. Los predicados, y los términos que identifican los elementos relacionados, se utilizan para definir las *fórmulas atómicas* o *átomos*. Algunos ejemplos de átomos son los siguientes:

hombre (JUAN)
masalto (JUAN, PEPE)
masalto (JUAN, padre (PEPE))

Las variables, por su parte, representan conjuntos de constantes.

Las funciones describen elementos y los identifican como el resultado único de la aplicación de una transformación entre otros elementos del dominio [e.g., padre(JUAN), madre(padre(JUAN)), asesino(x))].

Por último, las constantes representan los elementos específicos del dominio de discurso (i.e., MILU, IDEFIX, RINTINTIN).

Con todos los elementos anteriormente descritos deberíamos ser capaces de construir fórmulas atómicas susceptibles de una adecuada representación. A continuación deberemos utilizar la *semántica* para dotar de significado al conjunto específico de símbolos definidos en el dominio, y establecer su correspondencia en el universo de los hechos:

perro (MILU) Milú es un perro
p(A) Mi coche es nuevo

Otros elementos del alfabeto son los *juntores*, que nos permiten representar declaraciones compuestas. Entre ellos se encuentran los siguientes:

AND: Con el cual, para que una FBD sea cierta, todos y cada uno de los componentes relacionados han de ser ciertos.

OR: Con el cual, al menos uno de los componentes relacionados ha de ser cierto para que la FBD correspondiente sea cierta.

NOT: Juntor que cambia el estado lógico de una expresión.

$\rightarrow$: Que establece relaciones de implicación entre expresiones.

= : Juntor que indica la equivalencia lógica entre dos FBDs.

Otro elemento importante del alfabeto son los *cuantificadores*, que surgen como procedimiento para resolver uno de los problemas mencionados en la lógica de proposiciones, y entre los que destacamos los siguientes:

Cuantificador Universal: $\forall x$

Establece que la FBD es cierta para todos los valores que puede tomar "*x*"

$\forall x$ [PERSONA (x) $\rightarrow$ NECESITA_AIRE (x)]

Cuantificador Existencial: $\exists x$

Establece que existe al menos un "*x*" que hace verdadera a la FBD.

$\exists x$ [PROPIETARIO (x, COCHE) and PROPIETARIO (x, BARCO)]

En ambos casos la variable genérica "*x*" asociada al cuantificador se denomina *variable cuantificada*, mientras que el *alcance* del cuantificador es la FBD que le sigue.

El último elemento importante del alfabeto son los *delimitadores*, elementos como "," y "()", necesarios para obtener representaciones correctas del conocimiento.

Lenguaje formal

El lenguaje formal asociado a la lógica de predicados es el conjunto de todas las FBDs que se pueden construir legalmente a partir del alfabeto.

Se puede definir inductivamente una FBD del siguiente modo:

Cualquier fórmula atómica es FBD.

Si *F* y *G* son FBDs, entonces también son FBDs las siguientes:

not F F and G F or G $F \rightarrow G$

Si "x" es una variable, y *F* es una FBD, entonces también son FBDs las siguientes:

$(\forall x)\ F$ $(\exists x)\ F$

El conjunto de FBDs que seamos capaces de construir sobre un dominio concreto constituye el *lenguaje formal* asociado. De este modo, y considerando la definición anterior de FBD, la expresión:

$$(\exists x)\ [(\forall x)[P(x,y) \text{ and } Q(x,y) \text{ and } R(x,x) \rightarrow R(x,y)]]$$

es una FBD en algún dominio no determinado (i.e., no vulnera ninguna de las condiciones que deben cumplir las FBDs si "*x*" e "*y*" son variables).

Existen sin embargo expresiones más sencillas que, de acuerdo con la definición inductiva dada anteriormente, no pueden ser consideradas FBDs. Algunas de tales expresiones son:

not f(A)
$(\forall P)$ P(A)
$(\exists f)$ f(A)

Las expresiones anteriores *no son FBDs en lógica de predicados.* Las lógicas que no permiten cuantificación sobre predicados o funciones se denominan *lógicas de primer orden*.

Reglas de inferencia

La inferencia en lógica formal es el proceso de generar nuevas FBDs a partir de FBDs ya existentes mediante la aplicación de las llamadas *reglas de inferencia*. De tales reglas la más común es la llamada "*modus ponens*", que se puede expresar simbólicamente del siguiente modo:

$$[P1 \text{ and } (P1 \rightarrow P2)] \rightarrow P2$$

expresión que viene a decir que "si sabemos que P1 implica P2, y sabemos que P1 es verdad, podemos inferir que P2 es verdad".

Otra regla de inferencia común es la "*especialización universal*", que se utiliza para generar la FBD "*f*(INDIVIDUO)" a partir de la FBD $(\forall x)[f(x)]$. La especialización universal puede expresarse simbólicamente del siguiente modo:

$$\text{INDIVIDUO and } (\forall x)[f(x)] \rightarrow f(\text{INDIVIDUO})$$

El *modus ponens* nos permite, por ejemplo, inferir la existencia de "fuego" al observar la presencia de "humo" (Figura 1.2). Por el contrario, la *especialización universal* nos permite inferir que "las hortensias necesitan agua para vivir", ya que "las hortensias son plantas" y "todas las plantas necesitan agua para vivir".

Además de estas dos reglas básicas de inferencia existen otras, como la *sustitución*, el *modus tollens*, etc, todas ellas capaces de generar nuevas FBDs a partir de FBDs ya existentes. Con estas reglas, con tal que los axiomas construidos sean válidos, la inferencia sintáctica es siempre posible.

Desgraciadamente, el que una inferencia sea posible no quiere decir que el resultado tenga o no el menor interés. Por el contrario, en lógica formal es muy común que las inferencias realizadas conduzcan a información absolutamente irrelevante y, además, no se nos asegura la obtención de nueva información útil en un tiempo prudencial.

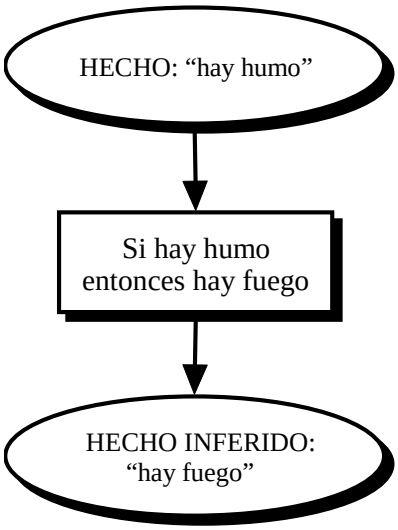


Figura 1.2 Un ejemplo esquemático de “modus ponens”

1.3. Ingeniería del Conocimiento y Lógica Formal

Como ya se ha comentado, una de las tareas de la *ingeniería del conocimiento* es la de estructurar y codificar conocimiento para que éste sea utilizado de forma eficiente por un programa de inteligencia artificial. Aunque más adelante tendremos ocasión de profundizar en los aspectos esenciales de la ingeniería del conocimiento, veremos a continuación algunas de sus características fundamentales en relación con la lógica formal.

Cuando empleamos un esquema de representación del conocimiento basado en lógica formal, el proceso básico de ingeniería del conocimiento consta de las siguientes fases:

Comprensión e identificación del conocimiento relevante del dominio.
Formalización de los enunciados correspondientes.
Análisis o fragmentación de los enunciados en sus partes constituyentes.
Establecimiento de la simbología adecuada para representar elementos y relaciones.
Construcción de las FBDs

Estos cinco pasos (denominados respectivamente *identificación, formalización, descomposición, traducción* y *recomposición*), configuran la fase de *codificación*, mediante la cual pretendemos obtener representaciones del conocimiento del dominio manejables desde una perspectiva computacional. Veamos algunos ejemplos:

Codificar en lógica de predicados la declaración: “Milú es un perro foxterrier blanco”

(1) Descomposición: Milú es un perro
Milú es un foxterrier
Milú es blanco

(2) Traducción: PERRO (MILU)

FOXTERRIER (MILU)
BLANCO (MILU)

(3) Recomposición: FBD = PERRO (MILU) and
FOXTERRIER (MILU) and
BLANCO (MILU)

Codificar en lógica de predicados la declaración: “Anselmo, que es trompetista, ni juega al fútbol, ni toca el acordeón”

En este ejemplo aparece ya un primer problema de interpretación. Al respecto, interesa encontrar estructuras lo más “simétricas” posibles. La pregunta es ¿hasta qué punto serían aceptables las siguientes transformaciones?:

Anselmo no juega al fútbol = Anselmo no es futbolista
Anselmo no toca el acordeón = Anselmo no es acordeonista

En algunos casos, aunque podamos perder matices, tales transformaciones serán aceptables; en otros casos, sin embargo, no podremos considerar que las expresiones correspondientes sean equivalentes. En este ejemplo asumiremos que los matices que perdemos no son importantes.

(1) Descomposición: Anselmo es trompetista
Anselmo no juega al fútbol
Anselmo no toca el acordeón

equivale a: Anselmo es trompetista
Anselmo no es futbolista
Anselmo no es acordeonista

(2) Traducción: TROMPETISTA (ANSELMO)
no FUTBOLISTA (ANSELMO)
no ACORDEONISTA (ANSELMO)

(3) Recomposición: FBD = TROMPETISTA(ANSELMO) and
not FUTBOLISTA (ANSELMO) and
not ACORDEONISTA (ANSELMO)

Estos ejemplos sugieren que el proceso de codificación, más o menos complicado, está adecuadamente determinado. Puede ser que se pierda algún matiz; sin embargo, siempre deberemos ser capaces de encontrar una traducción en lenguaje formal satisfactoria para un determinado enunciado en lenguaje natural.

Veamos a continuación qué ocurre con la fase de decodificación, en la que haremos uso de la semántica para interpretar los enunciados de las correspondientes FBDs.

Interpretar en lenguaje natural la siguiente declaración escrita como FBD:

$$(\forall x)[\text{not DEPORTE}\ (x) \rightarrow \text{ENGORDA}\ (x)]$$

Literalmente: “para todo elemento -*x*- se verifica que si no deporte -*x*- entonces engorda -*x*-”

Existen varias declaraciones en lenguaje natural que corresponden a la FBD anterior:

Todo el que no hace deporte engorda
La inactividad produce aumento de peso
Quien lleva una vida sedentaria suele engordar

De forma más o menos libre, todas las frases anteriores podrían traducirse según la FBD del enunciado. La decodificación (al igual que la codificación), suele ir siempre acompañada de cierta imprecisión, ambigüedad o inexactitud. Los esfuerzos deben dirigirse hacia la minimización de la imprecisión asociada a los lenguajes formales.

Veamos más ejemplos:

Convertir a FBD la declaración: “No recuerdo si Carmen encontró una pulsera o un collar, pero lo cierto es que se puso muy contenta”

Este ejemplo pone de manifiesto varias cuestiones importantes:

(a) ¿Podemos prescindir de elementos de la frase original y, sin embargo, obtener una FBD apropiada?. En el ejemplo, ¿podemos eliminarnos de la declaración (como primera persona, sujeto de la acción), sin perder matices importantes? ¿qué es lo verdaderamente informativo, el que nos acordemos o no de un determinado hecho, o el hecho en sí?

(b) ¿Es importante el tiempo de la acción o, por el contrario, podemos asumir la acción desde una perspectiva atemporal?

(c) ¿Podemos asumir implicaciones en el sentido de relaciones causa-efecto no evidentes en la frase?

(d) ¿Podemos perder matices cuantitativos?

Estas y otras cuestiones del mismo tipo deberán ser planteadas antes de proceder a codificar una declaración determinada y compleja. Para la declaración de nuestro ejemplo, una posible solución sería la siguiente:

(1)	Descomposición:	Carmen se puso muy contenta Carmen se encontró una pulsera Carmen se encontró un collar
(2)	Traducción:	CONTENTA(CARMEN) ENCONTRAR_PULSERA(CARMEN) ENCONTRAR_COLLAR(CARMEN)
(3)	Recomposición:	[ENCONTRAR_PULSERA(CARMEN) or ENCONTRAR_COLLAR(CARMEN)] $\rightarrow$

$\rightarrow$ CONTENTA(CARMEN)

En este proceso de codificación hemos considerado que lo verdaderamente importante es el hecho en sí, por lo que nos hemos autoeliminado como sujeto de la acción. También hemos perdido matices cuantitativos (muy contenta), y hemos convertido la acción en atemporal (sustituyendo pretéritos por infinitivos). Por último, hemos asumido la implicación al considerar cierta la relación causa-efecto entre antecedente y consecuente (suponemos que Carmen se puso muy contenta por su hallazgo). El que tales suposiciones sean permisibles o no dependerá del caso concreto y del contexto que estemos tratando.

Al margen de estos problemas de interpretación intrínsecamente asociados a la semántica, existen otros problemas derivados de la interpretación de frases en su contexto. Analicemos los dos ejemplos que siguen a continuación:

A todo el mundo le gusta el arte o el deporte
En los Estados Unidos todos son o republicanos o demócratas

Ambas frases, estructuralmente idénticas, presentan no obstante importantes diferencias conceptuales debidas a cuestiones de contexto. Así, mientras la primera frase no excluye la posibilidad de que haya alguien interesado a la vez por el arte y por el deporte, no tendría sentido que hubiese un solo estadounidense que fuese simultáneamente republicano y demócrata. La traducción de ambas frases debe efectuarse teniendo en cuenta (y especificando) la naturaleza del juntor "OR":

Caso A: OR Exclusivo

Indica la posibilidad de una y sólo una de las cláusulas involucradas en la declaración.

Frase 1: $(\forall x)$[PERSONA $(x) \rightarrow$ GUSTA_ARTE(x) or GUSTA_DEPORTE(x) or [GUSTA_ARTE(x) and GUSTA_DEPORTE(x)]]

Frase 2: $(\forall x)$[ESTADOUNIDENSE $(x) \rightarrow$ REPUBLICANO (x) or DEMOCRATA (x)]

Caso B: OR Inclusivo

Indica la posibilidad de al menos una de las cláusulas involucradas en la declaración.

Frase 1: $(\forall x)$[PERSONA $(x) \rightarrow$ GUSTA_ARTE(x) or GUSTA_DEPORTE(x)]

Frase 2: $(\forall x)$[ESTADOUNIDENSE $(x) \rightarrow$ REPUBLICANO (x) or DEMOCRATA (x) and not [REPUBLICANO (x) and DEMOCRATA (x)]]

Dependiendo de las fases de identificación y formalización, a la hora de codificar conocimiento en lógica formal deberemos decidir qué carácter le queremos dar al juntor "OR" para obtener las mejores representaciones en el dominio. Lo que es claro es que, en una misma aplicación, no se pueden mezclar juntores "OR" de distinta naturaleza.

1.4. Evaluación y Resolución en Lógica Formal

Veíamos anteriormente que la lógica formal permite derivar declaraciones nuevas, y ciertas, a partir de un conjunto de axiomas o principios básicos (i.e., verdades en nuestro dominio de discurso). En cualquier caso, la verdad de una declaración está relacionada con el proceso de interpretación. En lógica de proposiciones la verdad de una proposición compleja puede determinarse a partir de los valores de la Tabla 1.1, conocida con el nombre de *tabla de verdad*.

X	**Y**	**X and Y**	**X or Y**	**X → Y**	**not X**	**X = Y**
si	si	si	si	si	no	si
si	no	no	si	no	no	no
no	no	no	no	si	si	si
no	si	no	si	si	si	no

Tabla 1.1 Tabla de verdad

Obsérvese que la definición de verdad para la implicación (→) no es intuitivamente clara. Una implicación es verdadera siempre que el antecedente sea falso o siempre que el consecuente sea verdadero. En realidad, la verdad de una implicación está referida a la propia implicación, pero no presupone nada sobre la verdad de cualquiera de sus componentes. Esta característica nos obligaría a concluir como cierto que:

PERRO (MILU) → GATO (MILU)

En lógica formal la determinación de la verdad de una fórmula compleja supone la reducción sucesiva de la declaración, desde dentro hacia afuera, utilizando convenientemente la tabla de verdad.

Analicemos el siguiente ejemplo:

Dados los siguientes axiomas: "Algún P es falso" y "Q y R son verdaderos" concluir algo acerca de la verdad de la declaración:

[[P or (Q and R)] and not P] → (Q and P)

Resolviendo la expresión desde dentro hacia afuera:

(1) Q es verdadero
R es verdadero
——————————→ Q y R es verdadero

(2) [P or (verdadero)]

⟶ verdadero

(3) [(verdadero) and not P]
⟶ verdadero (hay al menos un P falso)

(4) (Q and P)
⟶ falso (hay al menos un P falso)

(5) verdadero → falso
⟶ falso (la implicación es falsa)

Podemos concluir que la declaración completa es falsa dados los axiomas.

Este procedimiento de evaluación de declaraciones es muy sensible a la dificultad de los dominios. El proceso se complica enormemente aún en dominios relativamente sencillos. Veamos el siguiente ejemplo:

Sea el siguiente conjunto de declaraciones:

Marco fue un hombre que nació en Pompeya

Todos los pompeyanos eran romanos

César era un gobernante al que muchos romanos le profesaban lealtad; otros, sin embargo, no le eran leales

Todo el mundo es leal a alguien

Cuando alguien intenta asesinar a un gobernante es porque no le es leal

Marco intentó asesinar a César

... intentar averiguar si Marco era leal a César.

Codificación:

(1) HOMBRE (MARCO)

(2) POMPEYANO (MARCO)

(3) $(\forall x)$ [POMPEYANO (x) → ROMANO (x)]

(4) GOBERNANTE (CESAR)

(5) $(\forall x)$ [ROMANO (x) → [LEAL_A (x,CESAR) or ODIA_A(x,CESAR)] and not [LEAL_A (x,CESAR) and ODIA_A (x,CESAR)]]

(6) $(\forall x)$ $(\exists y)$ LEAL_A (x,y)

(7) $(\forall x)$ $(\forall y)$ [PERSONA (x) and GOBERNANTE (y) and INTENTA_ASESINAR $(x,y) \rightarrow$ not LEAL_A (x,y)]

(8) INTENTA_ASESINAR (MARCO,CESAR)

Con todas las limitaciones vistas anteriormente, ésta podría ser una codificación adecuada del conjunto de axiomas previamente expuesto. Trataremos ahora de responder a la pregunta planteada. Para empezar el proceso partiremos de una hipótesis razonable (hipótesis de trabajo) que será la declaración "Marco no fue leal a César = not LEAL_A (MARCO,CESAR)". Asumiremos que la hipótesis es cierta y comenzaremos las sustituciones:

En (7)

PERSONA (MARCO) and GOBERNANTE (CESAR) and

INTENTA_ASESINAR (MARCO,CESAR) $\rightarrow$ not LEAL_A (MARCO, CESAR)

Dado que hemos asumido que la hipótesis es cierta, la implicación es verdadera; por lo tanto, nos quedaremos con:

PERSONA (MARCO) and GOBERNANTE (CESAR) and INTENTA_ASESINAR (MARCO,CESAR)

Por eliminación en (4)

PERSONA (MARCO) and INTENTA_ASESINAR (MARCO, CESAR)

Por eliminación en (8)

PERSONA (MARCO)

Esta última cláusula es irresoluble ya que entre los axiomas no existe ninguno que nos permita decidir sobre su veracidad. El proceso ha fracasado por falta de conocimiento. Una posible solución a este problema consiste en, una vez detectada la deficiencia, incluir nuevos axiomas que nos permitan proseguir el proceso de evaluación. En este caso la inclusión del axioma "$(\forall x)$ [HOMBRE $(x) \rightarrow$ PERSONA(x)]" resolvería el problema.

De todas formas, parece claro que la deficiencia surge en la fase de codificación, en donde la ambigüedad del lenguaje natural, la multitud de representaciones más o menos equivalentes y la carencia de sentido común, nos conducen muchas veces a callejones sin salida.

El método propuesto para evaluar expresiones no es muy operativo computacionalmente ya que nos obliga a realizar diversas sustituciones. Lo ideal sería disponer de un procedimiento de demostración que llevase a cabo, en una única operación, la variedad de procesos involucrados en un razonamiento basado en declaraciones de la lógica formal. Surgen así los procedimientos de *resolución,* que

operan sobre declaraciones previamente normalizadas según el método que veremos a continuación.

La resolución obtiene demostraciones por medio de la denominada *refutación*, o prueba consistente en tratar de encontrar que la negación de una declaración produce una contradicción axiomática. Este enfoque es radicalmente diferente a la técnica hasta ahora empleada de *demostración hacia atrás*, desde los teoremas hasta los axiomas.

De todas formas, antes de aplicar la resolución por refutación hemos de tener todo nuestro conocimiento estructuralmente *normalizado*. Para ello trataremos de construir la denominada *forma normalizada conjuntiva de Davis* que, en esencia, trata de simplificar las FBDs y separar los cuantificadores del resto de la fórmula. El procedimiento para obtener la forma normalizada conjuntiva de Davis es el siguiente:

(1) Eliminar las implicaciones

(2) Reducir el número de negaciones

(3) Normalizar las variables para que cada cuantificador esté ligado a una única variable

(4) Obtener la fórmula normalizada *PRENEX*, constituida por un prefijo de cuantificadores seguido por una matriz libre de cuantificadores

(5) Eliminar los cuantificadores existenciales

(6) Abandonar el prefijo

(7) Convertir la matriz en una conjunción de disyunciones

(8) Identificar las cláusulas

(9) Normalizar las variables por separado en el conjunto de cláusulas generadas en el paso anterior

Para realizar las transformaciones necesarias para obtener la forma normalizada conjuntiva de Davis es útil considerar las equivalencias de la Tabla 1.2 o *tabla de equivalencias*.

P1 $\rightarrow$ P2	not P1 or P2
P1 or (P2 and P3)	(P1 or P2) and (P1 or P3)
P1 and (P2 or P3)	(P1 and P2) or (P1 and P3)
P1$\rightarrow$ P2	not P2 $\rightarrow$ not P1
not (not P1)	P1
not (P1 or P2)	not P1 and not P2
not (P1 and P2)	not P1 or not P2
not $\forall$x P(x)	$\exists$x not P(x)
not $\exists$x P(x)	$\forall$x not P(x)
P1 or Falso	P1
P1 or Verdadero	Verdadero
P1 and Falso	Falso
P1 and Verdadero	P1
P1 or not P1	Verdadero
P1 and not P1	Falso

Tabla 1.2 Tabla de equivalencias

La eliminación de implicaciones se realiza considerando que:

$$P1 \rightarrow P2 \quad = \quad \text{not P1 or P2}$$

Por otra parte, la eliminación de negaciones se consigue empleando not [not P] = P, las leyes de DeMorgan, y las equivalencias entre cuantificadores.

La normalización de variables para que cada cuantificador esté ligado a una única variable es posible puesto que las variables no tienen valores concretos. En otras palabras, podremos asignar más de un identificador a una variable sin que la correspondiente FBD se vea alterada (e.g.,$\forall x$ P(x) or $\forall x$ Q(x) = $\forall x$ P(x) or $\forall y$ Q(y))

La eliminación de cuantificadores existenciales es posible ya que debe existir al menos un valor que pueda sustituir a la variable cuantificada existencialmente, y que haga verdadera a la fórmula. Así, podemos eliminar el cuantificador sustituyendo la variable por una referencia a una función que genere el valor deseado. Dado que no siempre conocemos el valor que hace verdadera a la fórmula, debemos crear un nuevo nombre de función para cada sustitución. De este modo no hacemos ninguna afirmación sobre tales funciones y sólo indicamos que deben existir. Así, la fórmula:

$$(\exists y)\ \text{PRESIDENTE}\ (y) = \text{PRESIDENTE}\ (S1)$$

donde S1 es una función sin argumentos que hace verdad a PRESIDENTE. En el caso de cuantificadores existenciales afectados por cuantificadores universales, los valores que satisfagan al predicado pueden depender de los valores de las variables cuantificadas universalmente, por lo tanto debemos generar funciones con el mismo número de argumentos que el número de cuantificadores universales que afectan a la expresión. Así:

$$(\forall x)\ (\exists y)\ \text{PADRE\_DE}\ (y,x) = (\forall x)\ \text{PADRE\_DE}\ (S2(x),x)$$

Tales funciones se denominan *funciones de SKOLEM*, si tienen argumentos, y *constantes de SKOLEM*, si carecen de argumentos.

Una vez conseguida una fórmula PRENEX, en la que los únicos cuantificadores sean universales, el prefijo puede ser simplemente ignorado sin que por ello la declaración se vea afectada.

Para convertir la matriz resultante en una conjunción de disyunciones tendremos que emplear las propiedades:

asociativa (i.e., P1 or (P2 or P3) = (P1 or P2) or P3), y
distributiva (i.e., [(P1 and P2) or P3 = (P1 or P3) and (P2 or P3)]).

De este modo, la fórmula:

(INVIERNO and BOTAS) or (VERANO and SANDALIAS)

puede sufrir las siguientes tranformaciones:

(INVIERNO and BOTAS) or (VERANO and SANDALIAS) =

= [INVIERNO or (VERANO and SANDALIAS)] and
[BOTAS or (VERANO and SANDALIAS)] =

= (INVIERNO or VERANO) and (INVIERNO or SANDALIAS) and
(BOTAS or VERANO) and (BOTAS or SANDALIAS)

Una vez obtenidas e identificadas las cláusulas correspondientes, la normalización de variables por separado supone renombrar las variables de forma que no haya dos cláusulas que hagan referencia a la misma variable. El resultado final es una disyunción de *literales*.

Analicemos el siguiente ejemplo: “Obtener la forma normalizada conjuntiva de Davis para la siguiente expresión: *Todos los romanos que Marco conocía odiaban a César, y los que no le odiaban creían que odiar es de locos”*

Comprensión e Identificación:

La declaración anterior es análoga a la siguiente: “Todos los romanos conocidos por Marco, o bien odiaban a César, o bien creían que odiar es de locos”

Descomposición y Traducción:

“Todos los romanos conocidos por Marco...”
$(\forall x)$ [ROMANO (x) and CONOCE (x, MARCO)]

“...odiaban a César...” (los romanos a quienes Marco conoce)
ODIA (x, CESAR)

"Creen que odiar es de locos..."
$(\forall y)\ [(\exists z)\ \text{ODIA}\ (y,z)] \rightarrow \text{CREE\_LOCO}\ (x,y)$

Recomposición:

$(\forall x)$ [ROMANO (x) and CONOCE (x, MARCO)] $\rightarrow$

[ODIA (x, CESAR) or $[(\forall y)\ [(\exists z)\ \text{ODIA}\ (y,z)] \rightarrow \text{CREE\_LOCO}\ (x,y)]]$

En la expresión anterior el juntor "or" tiene carácter exclusivo.

Aplicamos ahora el procedimiento descrito[42] para representar a la FBD en forma clausal.

(1) Eliminación de implicaciones:

$(\forall x)$ not [ROMANO (x) and CONOCE (x,MARCO)] or [ODIA (x,CESAR) or $[(\forall y)$ not $[(\exists z)\ \text{ODIA}\ (y,z)]$ or CREE_LOCO (x,y)]]

(2) Eliminación de negaciones:

$(\forall x)$ [not ROMANO (x) or not CONOCE (x,MARCO)] or [ODIA (x,CESAR) or $[(\forall y)\ (\forall z)$ not ODIA (y,z) or CREE_LOCO (x,y)]]

(4) Obtención de la fórmula normalizada PRENEX:

$(\forall x)\ (\forall y)\ (\forall z)$ [not ROMANO (x) or not CONOCE (x,MARCO)] or [ODIA (x,CESAR) or [not ODIA (y,z) or CREE_LOCO (x,y)]]

(6) Abandonamos el prefijo:

[not ROMANO (x) or not CONOCE (x,MARCO)] or [ODIA (x,CESAR) or [not ODIA (y,z) or CREE_LOCO (x,y)]]

(7) Convertimos la matriz en una conjunción de disyunciones:

not ROMANO (x) or not CONOCE (x,MARCO) or ODIA (x,CESAR) or not ODIA (y,z) or CREE_LOCO (x,y)

La expresión resultante es una disyunción con 5 literales pertenecientes a una única cláusula, ya que no hay conjunción alguna.

En este momento estamos ya en condiciones de aplicar el método de *resolución por refutación*. Tal método no es más que un proceso iterativo simple en el cual, en cada caso, se resuelven dos cláusulas llamadas *cláusulas padres* produciéndose una nueva

[42] Nótese que en este ejemplo algunos pasos del procedimiento general no son necesarios. No obstante, para mayor claridad mantenemos la numeración utilizada en la descripción del método.

cláusula inferida. Para mayor claridad ilustraremos el método de resolución empleando la *lógica de proposiciones* como esquema de representación.

Consideremos una proposición “S” y un conjunto de axiomas “F”. En primer lugar debemos convertir todas las proposiciones “F” a forma clausal. A continuación negaremos la hipótesis “S” que queremos investigar, la convertiremos a forma clausal y la añadiremos al conjunto de cláusulas anterior. Por último, seleccionaremos dos cláusulas padre y las resolveremos juntas, repitiendo el proceso de resolución de cláusulas hasta que encontremos una cláusula vacía (indicativa de una contradicción), o hasta que no podamos continuar.

Consideremos el siguiente conjunto de axiomas:

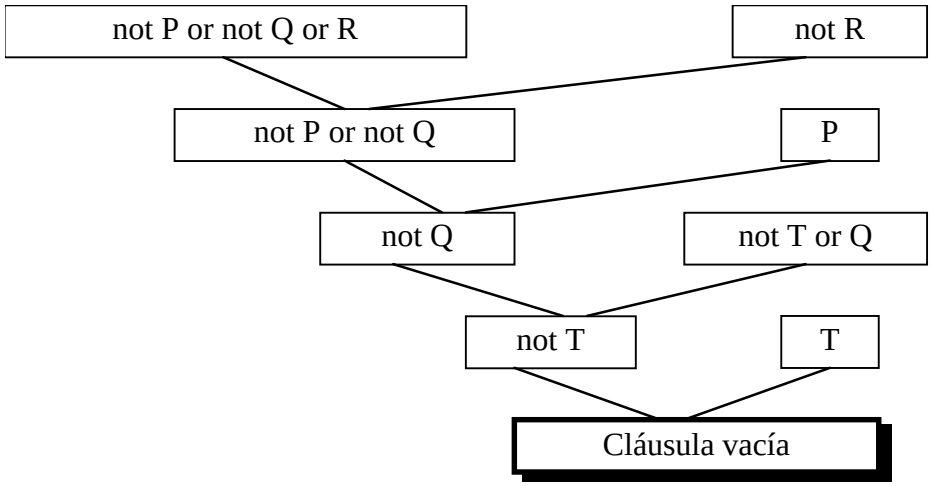
P
(P and Q) $\rightarrow$ R
(S or T) $\rightarrow$ Q
T

y demostremos “R” por resolución.

Conversión a forma clausal e inclusión de la hipótesis negada:

P
not P or not Q or R
not S or Q
not T or Q
T
not R

Resolución:



El hecho de encontrar una cláusula vacía indica una contradicción. Dado que hemos partido de un conjunto de axiomas (que son ciertos), al cual hemos incorporado la negación de la hipótesis que queremos verificar, la hipótesis debe ser cierta.

Este procedimiento es esencialmente el mismo que el método de resolución empleado en lógica de predicados. Sin embargo, la resolución en lógica de predicados es algo más compleja debido a la necesidad de considerar todas las formas posibles de sustituir valores en las variables.

1.5. Introducción a Otras Lógicas

Las lógicas formales de primer orden son útiles para encarar problemas en una amplia variedad de dominios. Sin embargo hay multitud de campos interesantes en los que, simplemente, los esquemas de representación del conocimiento basados en lógica formal de primer orden son totalmente inadecuados.

Así, información que contiene grados relativos de magnitudes (i.e., más grande que,...), información incierta o información imprecisa, heurísticas amplias, etc, constituyen problemas no tratables desde la óptica de estas lógicas formales.

Para tratar de resolver los problemas que acabamos de mencionar han sido propuestos diversos enfoques, entre los que cabe considerar los siguientes:

Las lógicas no monótonas

El razonamiento probabilístico

La lógica difusa

Los modelos de credibilidad

Algunos de estos modelos serán tratados en temas posteriores. Mencionaremos aquí, no obstante, parte de las características fundamentales de la lógica no monótona.

La lógica no monótona permite la eliminación del conocimiento. De este modo, la verosimilitud de una afirmación puede estar basada en la falta de confianza en alguna otra afirmación.

Los sistemas tradicionales basados en lógica formal son monótonos en el sentido de que el número de declaraciones verdaderas se incrementa estrictamente en el transcurso de los procesos inferenciales. Pueden añadirse nuevas declaraciones al sistema, y pueden demostrarse nuevas relaciones, sin embargo, ninguno de tales acontecimientos invalidará nunca una declaración demostrada o conocida previamente. Por ello, algunas de las ventajas de tales sistemas son:

Cuando se añade una nueva declaración, no es necesario realizar ningún análisis de consistencia.

Dada una declaración que acaba de ser demostrada, no es necesario recordar las declaraciones en las que se ha basado la demostración, ya que no hay riesgo de que tales declaraciones desaparezcan.

Desgraciadamente, los sistemas monótonos tienen problemas cuando trabajan con información incompleta, entornos cambiantes y generación de supuestos, situaciones todas ellas típicas de problemas del mundo real.

Los sistemas no monótonos basan su estrategia en el hecho de que, normalmente, nunca se dispone de toda la información que es útil para resolver un problema. Sin embargo, cuando dicha información falta, siempre pueden hacerse suposiciones sensatas mientras no se presente ninguna evidencia contradictoria.

La construcción de tales suposiciones origina lo que se denomina *razonamiento por defecto*, y es un caso particular de lógica no monótona. Su no-monotonicidad le viene del hecho de que la inclusión de una evidencia nueva puede forzar la eliminación de conocimiento hasta entonces considerado “cierto” o “válido”.

Una descripción computacional precisa del razonamiento por defecto debe establecer una conexión entre la falta de conocimiento específico “X” y una conclusión “Y”:

Si X no es conocido,
Entonces concluir Y

Pero en casi todos los sistemas sólo una pequeña porción de “elementos conocidos” son almacenados explícitamente, los demás elementos deben poder ser demostrados a partir de verdades explícitas.

Así, una definición mejor del razonamiento por defecto es:

Si X no puede demostrarse,
Entonces concluir Y

Continuando con este planteamiento, y desde la óptica de la lógica formal... ¿cómo sabemos que X no puede demostrarse? En efecto, para un X arbitrario nunca podremos asegurar si tal X puede o no ser demostrado. Hemos de replantearnos la definición del siguiente modo:

Si X no puede demostrarse en un determinado instante,
Entonces concluir Y

Pero ahora la definición ya no depende de cuestiones estrictamente lógicas. Depende más bien de la potencia de computación en el tiempo asignado, y de la eficacia de los procesos de búsqueda definidos en el sistema. Como consecuencia, nos es imposible realizar planteamientos estrictamente formales.

La necesidad del razonamiento por defecto, que surge de la carencia de información completa, nos obliga a utilizar sistemas cuya conducta no pueda caracterizarse fácilmente.

1.6. Resumen

En este capítulo nos enfrentamos por primera vez con el problema de la representación del conocimiento. Tras una breve discusión sobre la fase de codificación-decodificación se mencionan algunas de las características y condiciones que debe reunir todo esquema de representación del conocimiento en IA. A continuación se distinguen los dos grandes tipos de esquemas empleados: métodos declarativos y métodos procedimentales. El deseo de poder formalizar la representación del conocimiento nos lleva a la introducción de la lógica de proposiciones. Sin embargo, las limitaciones de la lógica de proposiciones, que no nos permite representar varios ejemplos de una misma entidad, ni cuantificar, nos lleva a preferir la lógica de predicados como esquema de representación formal. A continuación se describen los componentes básicos de la lógica de predicados y se proponen algunos ejemplos. Dado que se está considerando la lógica de predicados como un esquema de representación del conocimiento, inmediatamente después nos planteamos la relación entre la ingeniería del conocimiento y la lógica formal. Así, encontramos que el proceso básico de ingeniería del conocimiento, cuando empleamos lógica de predicados debe pasar por las siguientes fases: identificación, formalización, descomposición, traducción y recomposición. Pero el fin último que se persigue cuando representamos conocimiento en IA es obtener inferencias nuevas; es decir, ampliar nuestro conocimiento a partir del conocimiento ya existente. De este modo introducimos el concepto de "evaluación". No obstante, la necesidad de efectuar múltiples sustituciones complica mucho los procesos inferenciales, aún en problemas pequeños. Por ello se describe a continuación el procedimiento de resolución, mediante el cual, partiendo de una expresión previamente normalizada, se puede comprobar fácilmente la veracidad de una declaración por "refutación". Concluimos el tema con una breve mención a otras lógicas que se pueden emplear, y hacemos énfasis especial en los esquemas no monótonos que nos permiten razonar "por defecto".

1.7. Textos Básicos

Nilsson, "Principios de Inteligencia Artificial", Díaz de Santos, eds., 1987

Rich, "Inteligencia Artificial", G.Gili, eds., 1988

Rich, Knight, "Inteligencia Artificial", McGraw-Hill, eds., 1994

Rolston, "Principios de Inteligencia Artificial y Sistemas Expertos", McGraw-Hill, eds., 1990