

---

# DEDUCCIÓN: NATURAL Y AUTOMÁTICA



*Faraón Llorens Largo*

Diciembre de 2002



# Resumen Lección

## Dedución:

### Natural y Automática

para optar al Concurso nº C 1246 para proveer la plaza DF02479  
Convocatoria publicada en B.O.E. de 8 de diciembre de 2001

#### Catedrático de Escuela Universitaria

Perfil: *Lógica Computacional*  
*Razonamiento*  
para Ingeniería Informática

Área de Conocimiento: *Ciencia de la Computación e Inteligencia*  
*Artificial*

Impartida por:  
*Faraón Llorens Largo*



Departamento de Ciencia de la Computación e Inteligencia Artificial

**UNIVERSIDAD DE ALICANTE**



---

## RESUMEN

---

La Lógica de Predicados pone a nuestra disposición un lenguaje que nos permitirá formalizar expresiones del conocimiento humano haciendo explícitos los objetos y las relaciones, así como sus restricciones. Además nos proporciona un método, la deducción matemática, para obtener nuevo conocimiento a partir del antiguo. Es por ello que la lógica se convierte en una asignatura presente en los primeros cursos de distintas titulaciones, especialmente de informática, ya que proporciona una base formal de trabajo [Garrido, 1991], [Reeves y Clarke, 1993], [Llorens y Castel, 1999] y [Tymoczko y Henle, 2002]. En la actualidad existe un variado número de herramientas que pueden servir de ayuda en el aprendizaje de la lógica [Goldson, Reeves, y Bornat, 1993], [Barwise y Etchemendy, 2000], [TTL, 2000] y [Llorens y Mira, 2002].

El razonamiento es el proceso cognitivo por medio del cual utilizamos y aplicamos nuestro conocimiento, permitiéndonos pasar de una información a otra relacionada con esta. Sin la posibilidad de hacer inferencias, el sistema de procesamiento de información se vería obligado a tener que definir todas las situaciones puntuales y específicas con las que se tenga que enfrentar. A las sentencias de las cuales partimos en el proceso de razonamiento se les llama premisas y a la sentencia a la cual llegamos se le denomina conclusión. Las premisas junto a la conclusión forman el argumento. El concepto lógico de deducción correcta dice que de premisas verdaderas debemos obtener conclusión verdadera, es decir, no podemos aceptar que las premisas sean verdaderas y la conclusión falsa. La lógica nos proporciona métodos de cálculo que nos permiten inferir, por simple manipulación sintáctica, nuevas fórmulas a partir de las conocidas [Socher-Ambrosius y Johann, 1997]. Uno de estos métodos es la Deducción Natural, cuyo mecanismo está muy cercano al razonamiento intuitivo del ser humano. Así, de forma sencilla, a partir de las fórmulas dadas como premisas y con el único apoyo de unas reglas básicas, obtenemos determinadas conclusiones. Podemos utilizar una herramienta didáctica diseñada específicamente para enseñar a los estudiantes a realizar deducciones naturales, el Asistente para Deducción Natural (ADN) [Llorens y Mira, 2000a], [Mira, 2000] y [Llorens y Mira, 2000b].

Además podemos intentar abordar el tema de la automatización de la deducción. Para ello debemos realizar pequeñas modificaciones a esta técnica (de notación, de reglas a aplicar y de estrategia a utilizar) que nos lo permita. Todos estos desarrollos teóricos dieron lugar a la aparición de la programación lógica, como paradigma de resolución de problemas basado en la lógica [Dodd, 1990]. Prolog es el lenguaje de programación lógica más utilizado [Bratko, 1990], [Clocksin y Mellish, 1987], [Giannesini y otros, 1989], [Callear, 1994] y [Deransart, Ed-Dbali, y Cervoni, 1996].



## CONTENIDO

Resumen.....	1
Contenido.....	3
Introducción .....	4
1 Representación del Conocimiento y Razonamiento .....	4
2 Argumentación.....	4
3 Razonamiento Intuitivo y Formal.....	5
4 Razonamiento Inductivo y Deductivo .....	6
5 Lenguaje .....	6
Deducción Natural .....	9
1 Cálculo Lógico.....	9
2 Reglas Básicas.....	9
3 Estrategias de Derivación .....	11
3.1 Razonamiento Hipotético .....	12
3.2 Prueba Exhaustiva.....	12
3.3 Refutación.....	13
4 Deducción <i>versus</i> Computación.....	13
5 Visualización de Deducciones Lógicas .....	14
6 Asistente para Deducción Natural .....	16
Deducción Automática .....	19
1 Reflexiones para la Automatización.....	19
2 Universo de Herbrand y Unificación .....	19
3 Forma Clausal.....	20
4 Sistemas de Refutación.....	21
5 Regla de Resolución.....	22
6 Programación Lógica y Prolog.....	25
Ejemplo.....	27
1 Formalización .....	27
2 Deducción Natural.....	28
3 Deducción Automática.....	29
3.1 Refutación y Forma Clausal.....	29
3.2 Regla de Resolución con Unificación .....	31
4 Programación Lógica.....	32
4.1 Programa Lógico .....	32
4.2 SLD-Resolución .....	32
4.3 Prolog.....	33
Bibliografía.....	35

---

## INTRODUCCIÓN

---

# 1 REPRESENTACIÓN DEL CONOCIMIENTO Y RAZONAMIENTO

¿Qué es *conocimiento*? podríamos decir que es el hecho o la condición de conocer algo que hemos adquirido por experiencia o asociación, por observación o deducción. Los seres humanos vamos adquiriendo conocimiento al ver, oír, tocar, sentir y oler el mundo que nos rodea. Y ese conocimiento lo vamos almacenamos de alguna manera en nuestro cerebro.

Cuando pretendemos simular el funcionamiento de nuestra mente para poder trabajar con conocimiento primero deberemos formalizarlo y representarlo de una determinada manera. Existen distintos modelos de *representación de conocimiento*, cada una con sus ventajas e inconvenientes. Pero de cualquier forma, una buena representación del conocimiento debe ser fácilmente entendible por los humanos y al mismo tiempo no ambigua y de fácil manipulación.

El *razonamiento* es el proceso cognitivo por medio del cual utilizamos y aplicamos nuestro conocimiento, permitiéndonos pasar de una información a otra relacionada con esta, es “la capacidad del hombre para manejar sus ideas”<sup>1</sup>. Sin la posibilidad de hacer inferencias, un sistema de procesamiento de información se vería obligado a tener que definir todas las situaciones puntuales y específicas con las que se tendría que enfrentarse. No se puede separar razonamiento de representación del conocimiento, van íntimamente ligados y cada uno condiciona al otro. A las sentencias de las cuales partimos en el proceso de razonamiento se les llama *premisas* y a la sentencia a la cual llegamos se le denomina *conclusión*. Las premisas junto a la conclusión forman el argumento.

## 2 ARGUMENTACIÓN

Las estructuras argumentales son la clave del razonamiento. En esencia un *argumento* es una promesa, de forma que si las premisas del argumento son verdaderas, el argumento *correcto* garantiza la verdad de la conclusión. Nuestro

---

<sup>1</sup> “Homo cybersapiens. La inteligencia artificial y la humana”, de Tirso de Andrés.



objetivo será *construir argumentos* correctos. Pero no es fácil componer argumentos, estos han de ser presentados de forma cuidadosa ya que todo argumento informal es siempre mejorable tras un análisis cuidadoso. El primer paso para entender un argumento es determinar su conclusión. Si el argumento tiene fuerza, los distintos enunciados propuestos como premisas deben apoyar la conclusión.

El *análisis de los argumentos* es esencial. Desde la antigüedad el hombre se ha preocupado de someter las ideas a discusión. En esencia, lo que hay detrás de estos análisis es el sentido común. Por un lado si no estamos de acuerdo con las conclusiones debemos exponer sus deficiencias para *refutarlos*. Por el contrario, si los argumentos son nuestros, debemos aprender sus debilidades para reforzarlos. Hay dos aspectos que pueden hacer débiles los argumentos. El primero son las suposiciones iniciales, ya que el argumento no vale para nada si las premisas están equivocadas. El otro son las inferencias sobre las que se apoya, ya que si son cuestionables el argumento entero se estropea. Sólo se puede discutir verdaderamente si se está de acuerdo en un mínimo. Con ese mínimo aceptado, uno puede decir, el otro replicar y así ir afinando los argumentos. Vamos a tratar ahora de establecer ese mínimo.

### 3 RAZONAMIENTO INTUITIVO Y FORMAL

**L**os humanos hemos desarrollado muchos modos de pensar, persiguiendo conseguir formas de elaborar pensamientos más precisos y adecuados a la realidad. Algunos de ellos semánticamente muy ricos, pero imprecisos y extraordinariamente polisémicos, por lo que eran inmanejables artificialmente. Se buscaron entonces modelos más precisos, buscando rigor conceptual y metodológico. En el enfoque simbólico se utilizan símbolos para representar el conocimiento y el estado del mundo y se simula el proceso cognitivo mediante manipulación de esos símbolos. La *lógica formal* pone a nuestra disposición un lenguaje que nos permitirá formalizar expresiones del conocimiento humano haciendo explícitos los objetos y las relaciones, así como sus restricciones. Además nos proporciona un método, la deducción matemática, para obtener nuevo conocimiento a partir del antiguo.

El concepto lógico de *deducción correcta* dice que de premisas verdaderas debemos obtener conclusión verdadera, es decir, no podemos aceptar que las premisas sean verdaderas y la conclusión falsa. La lógica nos proporciona métodos de cálculo que nos permiten inferir, por simple manipulación sintáctica, nuevas fórmulas a partir de las conocidas. Uno de estos métodos es la deducción natural, cuyo mecanismo está muy cercano al razonamiento intuitivo del ser humano. Así, de forma sencilla, a partir de las fórmulas dadas como premisas y con el único apoyo de unas reglas básicas, obtenemos determinadas conclusiones.

## 4 RAZONAMIENTO INDUCTIVO Y DEDUCTIVO

**C**lásicamente se distingue entre dos tipos de razonamiento. En el razonamiento deductivo se parte de unas premisas para alcanzar una conclusión que necesariamente se debe seguir de ellas, mientras que en el razonamiento inductivo alcanzaremos una conclusión que vendrá más o menos apoyada por las premisas. Veámoslo más detenidamente.

En un *razonamiento deductivo* la conclusión se sigue necesariamente de las premisas, o lo que es lo mismo, un razonamiento deductivo es correcto si es imposible que las premisas sean verdaderas y la conclusión falsa. Es por ello que decimos que el razonamiento deductivo es un proceso que va de lo general a lo particular o hacia abajo. Las conclusiones deductivas son en cierto modo tautológicas (cierto o falso, todo o nada) y únicamente reflejan información contenida en las premisas.

En cambio, en un *argumento inductivo* las premisas apoyan o sugieren la conclusión. Por ello decimos que es un proceso por el que se llega a lo general a partir de lo particular o hacia arriba. Diremos que un razonamiento inductivo es fuerte si es improbable que la conclusión sea falsa cuando las premisas sean verdaderas. Hablamos por tanto de fuerza del argumento y esto es cuestión de grados. Por tanto, las conclusiones inductivas son probabilísticas y van más allá de la propia evidencia expresada en las premisas. La lógica inductiva es la lógica de la ciencia, de forma que si un experimento se lleva a cabo con frecuencia y la mayoría de las veces se obtienen cierto resultado, podemos decir con cierto grado de seguridad que dicho resultado seguirá apareciendo si el experimento se repite a menudo bajo circunstancias que sean esencialmente las mismas. Es una herramienta que permite medir la garantía que ofrece una evidencia, manejando y controlando la incertidumbre.

## 5 LENGUAJE

**A**l pretender formalizar las expresiones del conocimiento humano y dado que ese conocimiento lo adquirimos, en gran medida, y transmitimos por medio del lenguaje, debemos trabajar con él. Pero el *lenguaje natural* que habitualmente utilizamos es ambiguo y engorroso, y por ello, para facilitar la manipulación formal del conocimiento necesitamos de un *lenguaje artificial*. Una característica esencial de los lenguajes artificiales es su simplicidad y regularidad, en contraste con la vaguedad e irregularidad de los lenguajes naturales (los que hablan los humanos). Las dos clases principales de lenguajes artificiales son los lenguajes formales de la lógica y los lenguajes de programación. Aquí nos ocuparemos de los primeros, aunque sin perder de vista los segundos, ya que la mayoría de reflexiones son extrapolables.

Los lenguajes formales son reglas para generar estructuras lógicas. En la definición de ese lenguaje formal empezaremos por determinar el alfabeto (conjunto de símbolos) que utilizaremos y las frases (*fórmulas bien formadas*) que podremos construir con combinaciones autorizadas de esos símbolos. Así, ante una sentencia del lenguaje natural buscaremos sus componentes:

- Qué se afirma, es decir, las propiedades y relaciones que aparecen en la sentencia y que representaremos en forma de *predicados*.
- De quienes se afirma, es decir, los objetos o individuos a los que hace referencia la sentencia y que representaremos por medio de *términos*.

Cuando los predicados se aplican a un solo término se trata de propiedades o características de dicho objeto; cuando hacen referencia a varios sujetos suelen representar relaciones. A su vez, definiremos y delimitaremos los mundos sobre los que trabajamos. Los objetos deben pertenecer a un dominio genérico o *universo del discurso*, y pueden ser:

- *Constantes*: representan objetos concretos del dominio.
- *Variables*: permiten referenciar cualquier elemento del universo.
- *Funciones*: denotan objetos referenciados indirectamente, en función de otros objetos.

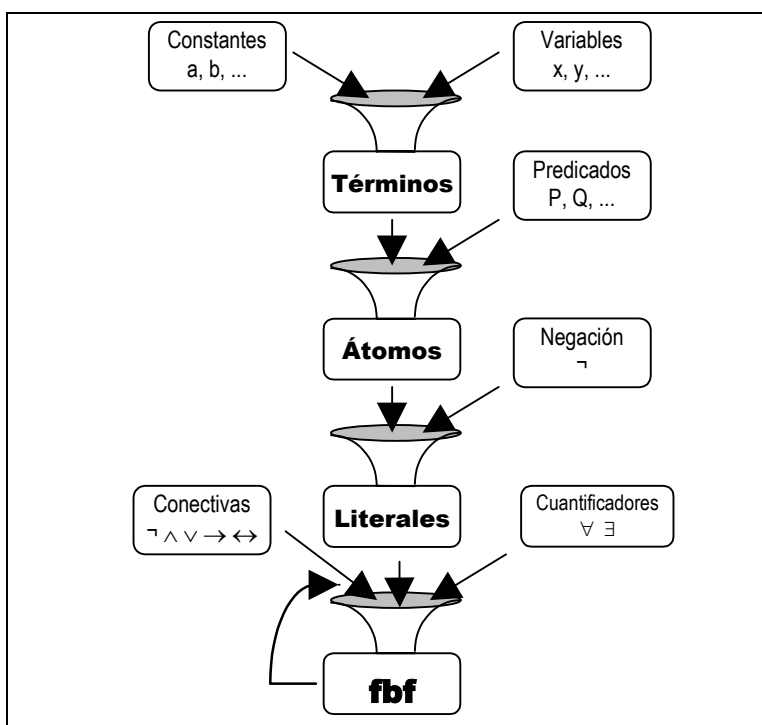


Figura 1: El lenguaje formal de la lógica

Así, con un predicado y los términos implicados en dicho predicado podemos formar *fórmulas atómicas* o elementales. Combinando dichas fórmulas atómicas mediante las conectivas lógicas (conjunción " $\wedge$ ", disyunción " $\vee$ ", negación " $\neg$ " e implicación " $\rightarrow$ ") obtendremos fórmulas más complejas, que llamaremos fórmulas moleculares. Por otro lado, si queremos expresar la cantidad de objetos que satisfacen alguna condición utilizaremos los cuantificadores. En la lógica clásica se definen únicamente dos, aunque se pueden extender. Así, el

cuantificador universal " $\forall$ " (para todo) hace referencia a todos los elementos del universo del discurso y el existencial " $\exists$ " (existe) indica que por lo menos existe un individuo del universo que satisface el enunciado.

Finalmente, los paréntesis se utilizan para clarificar las fórmulas. También nos puede ayudar a ver claramente la estructura sintáctica de las mismas la representación de las fórmulas en forma de árbol etiquetado. El árbol sintáctico de una fórmula lógica nos permite distinguir claramente cuál es el operador principal y las prioridades entre ellos. Esto nos ayudará en la comprensión del significado de la fórmula y en la posterior manipulación sintáctica cuando tengamos que aplicar las reglas de la deducción.

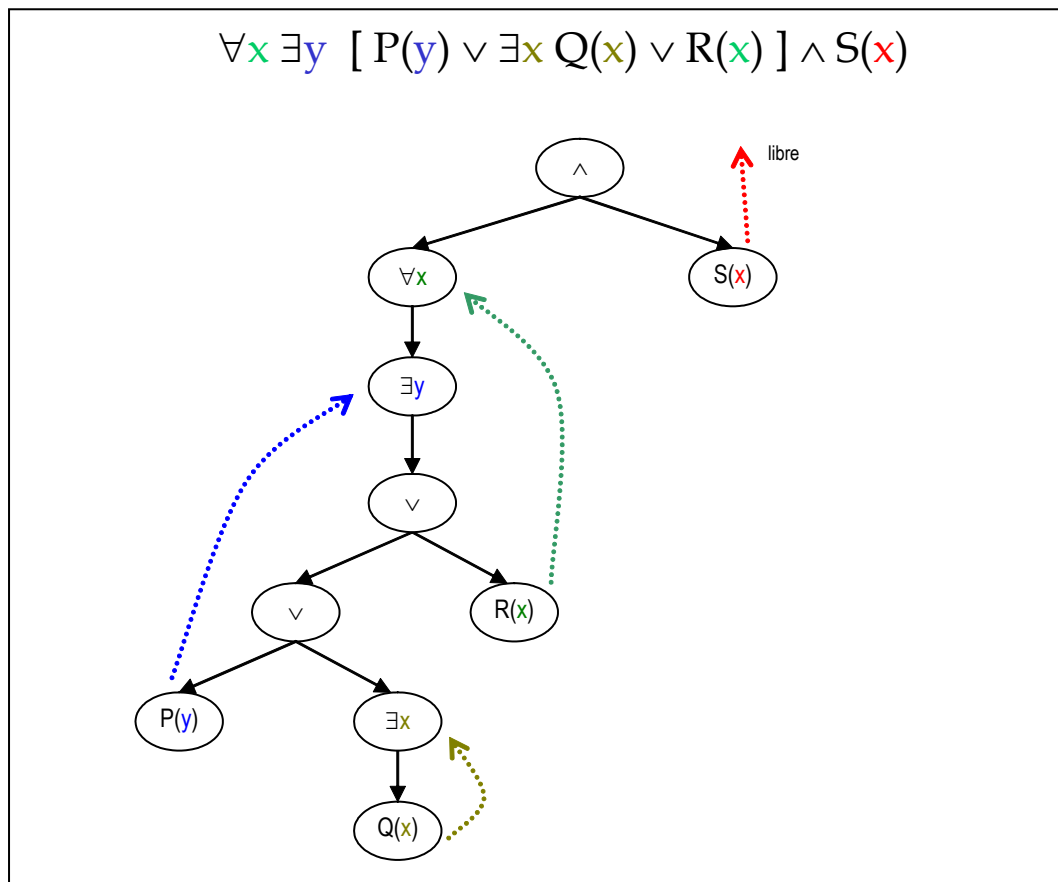


Figura 2: Árbol sintáctico

---

## DEDUCCIÓN NATURAL

---

### 1 CÁLCULO LÓGICO

Una idea básica de la lógica es que todos los argumentos, sin importar lo complejos que sean, se forman mediante el encadenamiento de inferencias muy simples. Así que se trata de desmenuzar el salto cognitivo que va de las premisas a la conclusión en pequeños pasos que hemos acordado como correctos. De esta forma, si asumimos las premisas y cada paso elemental que damos lo justificamos de forma adecuada, iremos obteniendo nuevas fórmulas lógicas que podemos asumir como conclusiones derivadas de las premisas.

Tendremos una deducción correcta cuando consigamos una *secuencia finita de fórmulas*, donde cada una de las fórmulas ha sido obtenida mediante la aplicación de alguna regla de inferencia. Las fórmulas iniciales de esa secuencia serán las premisas de las que partimos y la última fórmula obtenida la conclusión. A esta técnica se le denomina *deducción natural* porque es similar a la forma en que las personas razonan, avanzando paso a paso de las premisas de un argumento a la conclusión del mismo.

### 2 REGLAS BÁSICAS

Por tanto, antes de empezar el juego debemos determinar cuales son esas justificaciones aceptadas por todos, es decir las *reglas de inferencia* simples. Para determinar las reglas básicas de la deducción natural nos basaremos en la idea de montar y desmontar fórmulas lógicas, de forma que desmontando las premisas en sus componentes básicas (fórmulas atómicas) podamos montar la conclusión a partir de ellas. Para operar de esta manera dispondremos de dos reglas (una de introducción o entrada y otra de eliminación o salida) para cada uno de los símbolo lógico (conectivas y cuantificadores). Si la regla básica incorpora en su consecuencia una conectiva o cuantificador que no aparece en las fórmulas sobre las que se aplica será una regla de introducción; si elimina de su consecuencia una conectiva o cuantificador que aparece en sus fórmulas origen será una regla de eliminación.

	regla introducción	regla eliminación
$\wedge$ (conjunción)	IC	EC
$\vee$ (disyunción)	ID	ED*
$\neg$ (negación)	IN*	EN
$\rightarrow$ (implicación)	II*	EI*
$\forall$ (cuantificador universal)	IU	EU
$\exists$ (cuantificador existencial)	IE	EE

Figura 3: Tabla de reglas básicas de inferencia

\* Estas reglas se conocen con nombres propios: ED (prueba por casos), IN (reducción al absurdo), II (teorema de deducción) y EI (modus ponens)

Se intuye que si disponemos de procedimientos para añadir o quitar los distintos símbolos lógicos, podremos transformar por pura manipulación sintáctica las fórmulas que constituyen las premisas en la fórmula lógica que queremos obtener como conclusión. Desde un punto de vista de ingeniería, se trataría de desmontar las fórmulas lógicas que tenemos como premisas hasta obtener sus componentes básicas (fórmulas atómicas) y volver a montarlas en la configuración adecuada a la fórmula lógica objetivo. Las reglas serían las herramientas que nos permitirían montar y desmontar dichas fórmulas lógicas.

CONJUNCIÓN

Aceptar

IC

$$\frac{P \quad Q}{P \wedge Q}$$

EC

$$\frac{P \wedge Q}{P} \quad \frac{P \wedge Q}{Q}$$

DISYUNCIÓN

Aceptar

ID

$$\frac{P}{P \vee Q} \quad \frac{Q}{P \vee Q}$$

ED

(Prueba por Casos)  

$$\frac{P \vee Q \quad \left[ \begin{array}{l} P \\ R \\ Q \\ R \end{array} \right]}{R}$$

NEGACIÓN

Aceptar

IN

(Reducción al Absurdo)  

$$\frac{\left[ \begin{array}{l} P \\ Q \wedge \neg Q \end{array} \right]}{\neg P}$$

EN

$$\frac{\neg \neg P}{P}$$

IMPLICACIÓN

Aceptar

TD

(Teorema de la deducción)  

$$\frac{\left[ \begin{array}{l} P \\ Q \end{array} \right]}{P \rightarrow Q}$$

MP

(Modus Ponendo Ponens)  

$$\frac{P \rightarrow Q \quad P}{Q}$$

CUANTIFICACIÓN UNIVERSAL

Aceptar

IU

(con restricciones)  

$$\frac{P(a)}{\forall x P(x)}$$

EU

$$\frac{\forall x P(x)}{P(a)}$$

CUANTIFICACIÓN EXISTENCIAL

Aceptar

IE

$$\frac{P(a)}{\exists x P(x)}$$

EE

(con restricciones)  

$$\frac{\exists x P(x) \quad \left[ \begin{array}{l} P(a) \\ Q \end{array} \right]}{Q}$$

Figura 4: Reglas básicas

Por tanto, conforma vayamos obteniendo fórmulas las escribiremos en líneas consecutivas que numeraremos. Cada línea de nuestra deducción (y por tanto la fórmula lógica escrita en ella) estará justificada por la aplicación de una regla básica a alguna o algunas fórmulas de líneas anteriores. Dicha justificación la podremos en la columna de la derecha. Por ejemplo, en la figura del ejemplo, la justificación de la línea 10 que dice ED 1, 2-5, 6-9 significa que esa fórmula la hemos obtenido porque tenemos en la línea 1 una fórmula cuya conectiva principal es una disyunción; suponemos en la línea 2 una de las dos alternativas y obtenemos cierta conclusión condicionada en la línea 5; suponemos en la línea 6 la otra alternativa y obtenemos la misma conclusión condicionada en la línea 9; y llegamos en la línea 10 a que dicha conclusión ya no esta condicionada, porque queda justificada por la regla de eliminación del disyuntor o prueba por casos. En la siguiente figura podemos ver el ejemplo de la deducción natural del argumento  $\exists x (P(x) \wedge \forall y (Q(y) \rightarrow R(x,y))) \Rightarrow \forall y (Q(y) \rightarrow \exists x (P(x) \wedge R(x,y)))$ , que podría corresponder a demostrar que “cualquier número natural tiene otro menor que él” a partir de la premisa “existe un número natural menor que todos los demás”.

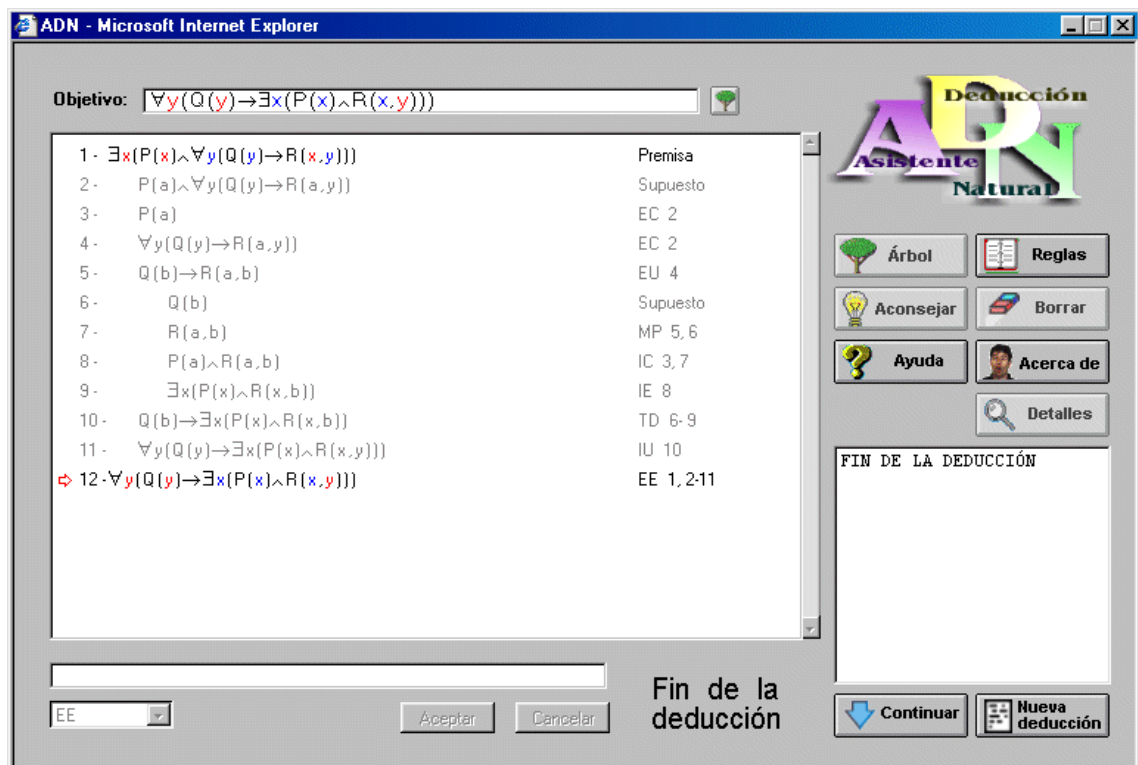


Figura 5: Ejemplo de Deducción

### 3 ESTRATEGIAS DE DERIVACIÓN

Una vez establecidas las reglas del juego vamos a trazar estrategias para tener éxito en el mismo y lograr nuestros objetivos. Uno de los aspectos cruciales

de la deducción natural es el de las *subpruebas* (subdeducciones o subderivaciones). En cualquier paso de nuestra deducción podemos introducir un supuesto provisional, que debe ser cancelado en alguna línea posterior. Desde el supuesto hasta la cancelación tendremos una subdeducción. Los supuestos provisionales son una herramienta muy potente ya que nos permiten suponer lo que nosotros queramos. Pero debemos pagar un alto precio por ello: para poder finalizar una demostración deberemos haber cancelado todas las suposiciones que hayamos abierto. De esta forma, la suposición y posterior cancelación de supuestos provisionales se convierte en una pieza clave de las deducciones naturales. La utilización de subdeducciones como estrategia de trabajo nos permite modularizar nuestras deducciones, planteándonos subobjetivos más sencillos que el objetivo final, y que en su conjunto nos lleven a la conclusión que buscamos.

En general, para planificar en cada momento nuestro siguiente paso tenemos dos opciones:

1. Empezar por el principio, examinando las premisas y la información que ya tenemos, determinando que podemos hacer con ellas.
2. Empezar por el final, examinando la conclusión a la que queremos llegar, estableciendo qué debemos hacer para alcanzarla.

Vamos a pasar a examinar con mayor detalle algunas de las estrategias que de que disponemos para ello.

### 3.1 Razonamiento Hipotético

En el *razonamiento hipotético* suponemos  $A$  (hipótesis) y entonces, usando la información de que disponemos, deducimos  $B$ . Este razonamiento no demuestra  $B$ , únicamente lo que demuestra es la implicación, es decir, si se cumple  $A$  entonces se deberá cumplir  $B$  ( $A \rightarrow B$ ). Es conocida también como *prueba directa*. De esta forma cuando la conclusión a la que queramos llegar tenga la implicación como conectiva principal, podemos suponer el antecedente de la misma e intentar llegar al consecuente. Una vez logrado cancelamos el supuesto con la regla de II (o más conocida como *teorema de deducción* o TD).

### 3.2 Prueba Exhaustiva

Pero hablando de hipótesis, habrá momentos en que tengamos que considerarlas todas, no desdeñar ninguna. Es lo que llamamos *prueba exhaustiva*. De esta forma si tenemos una disyunción, es decir, distintas alternativas con la garantía de que al menos una de ellas es cierta, podemos ir considerándolas una a una y demostrar que podemos llegar a una misma conclusión en cualquiera de los casos. En ese momento estamos en condiciones de afirmar dicha conclusión. Es lo que se conoce como *prueba por casos* (o eliminación de la disyunción ED). Podemos decir que es una *prueba exhaustiva explícita* ya que analizamos todos los casos posibles. Pero ¿qué podemos hacer cuando el número de casos sea muy grande o incluso infinito? Utilizaremos un rodeo lógico. Si sabemos que alguien,



al menos, cumple una cierta propiedad, supondremos un individuo genérico e intentaremos llegar a una conclusión que no dependa de ese individuo. De esa forma queda demostrada la conclusión. Lo podemos considerar como una *prueba exhaustiva implícita*, ya que únicamente trabajamos con un caso genérico. Esta estrategia la aplicaremos cuando tengamos una fórmula cuantificada existencialmente y aplicaremos la regla de eliminación del existencial (EE).

### 3.3 Refutación

Como último recurso, existe una estrategia que siempre se puede intentar: la *reducción al absurdo* o prueba por contradicción. Es bastante simple y muy utilizada, al mismo tiempo que controvertida. Permite establecer la verdad de una proposición demostrando que la proposición contraria conduce a un absurdo. Para demostrar  $A$ , asumimos  $\neg A$  e intentamos llegar a una contradicción (algo y su negación).

## 4 DEDUCCIÓN *VERSUS* COMPUTACIÓN

**P**odemos considerar la deducción como una forma de computación, ya que ¿un programa no es una deducción en la que a partir de unas entradas (premisas) debemos obtener unas salidas determinadas (conclusiones)? Se pretende presentar un tema marcadamente no informático para los estudiantes utilizando una metodología que enganche y motive a los alumnos de informática. La idea básica es que la deducción es una forma de computación. Este enfoque metodológico permite acercar ambos campos: lógica y programación. Se trata de abordar el tema de la deducción, comparándolo con la programación. De esta forma, por un lado, mantenemos el carácter formal de la lógica de manera que el alumno se acostumbra a trabajar con rigor y de forma abstracta. Por otro lado, su visión computacional motivará al alumno y, al bajar en nivel de abstracción, le ayudará a comprenderlo mejor.

Una deducción la podemos ver como un algoritmo que partiendo de unos valores de entrada (premisas) obtiene unas determinadas salidas (conclusiones) utilizando un conjunto dado de instrucciones (reglas). Y ¿qué es un programa de ordenador sino una deducción? En ambos casos se trata de ser capaces, con la única ayuda de las herramientas formales que se nos proporcionan (un lenguaje formal y unas reglas de inferencia, en un caso, y un conjunto de instrucciones en el otro) de resolver un problema que se nos plantea.

Las reglas básicas conformarían el conjunto de instrucciones básicas de nuestro lenguaje de programación. Podemos ver a las reglas derivadas como los procedimientos que nos podemos definir en cualquier lenguaje de programación de forma que nos permiten ahorrar líneas de código. Siguiendo con nuestra comparación con la computación, podemos considerar la columna de las justificaciones como las instrucciones de nuestro programa, de forma que

siguiendo esos pasos podemos realizar cualquier deducción que tenga la misma estructura. La columna de las fórmulas lógicas conformaría la traza del algoritmo (deducción) para una entrada concreta (si cambiamos las premisas, cambiarían las fórmulas). Las premisas serían los parámetros de entrada al programa y la conclusión la salida obtenida.

Deducción	Computación
premisas	valores de entrada
conclusión	valores de salida
reglas básicas	conjunto de instrucciones del lenguaje
fórmulas de las líneas de derivación	traza del programa
justificaciones de las líneas de derivación	líneas (instrucciones) del programa
reglas derivadas	procedimientos
subdeducciones	modularización

Figura 6: **Comparativa deducción - computación**

Pero no olvidemos que la realidad es la inversa, y hay que hacerla notar a los alumnos: la programación es una forma de demostración matemática. Por acercar el tema de la deducción a la realidad que conocen no debemos perder de vista el carácter formal y fundamental de esta disciplina.

## 5 VISUALIZACIÓN DE DEDUCCIONES LÓGICAS

Voy a visualizar una deducción lógica, es decir, hacer visible mediante cierta representación la estructura de nuestras sentencias y los pasos lógicos seguidos para obtener la conclusión. La visualización es importante debido a su fuerza expresiva y nos sirve para alcanzar cierta comprensión acerca del contenido y las relaciones. Recurrimos a la representación gráfica en forma de árbol para organizar de forma espacial el conocimiento que tenemos en forma de fórmulas lógicas y representamos el flujo lógico mediante manipulaciones del árbol.

A continuación se pueden ver los pasos de una animación realizada en Flash, que visualiza la demostración lógica del ejemplo que estoy utilizando (figura 4). Aunque se pierden los efectos dinámicos de la misma, puede servirnos de ilustración. La animación se puede descargar desde internet, en el apartado ejemplos del sitio web del asistente para deducción natural al que le dedico el siguiente apartado.

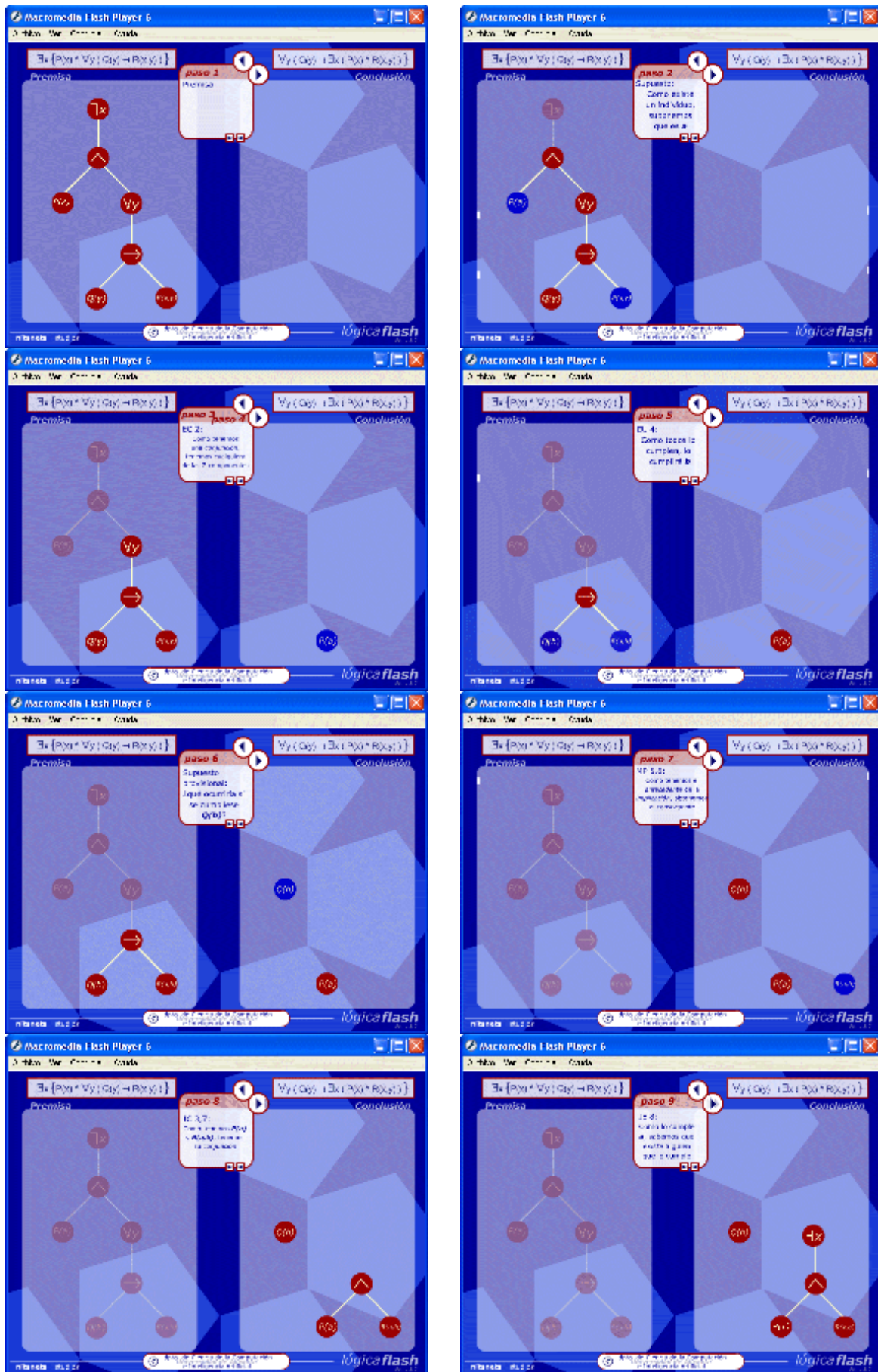


Figura 7: Visualización de una deducción (I)

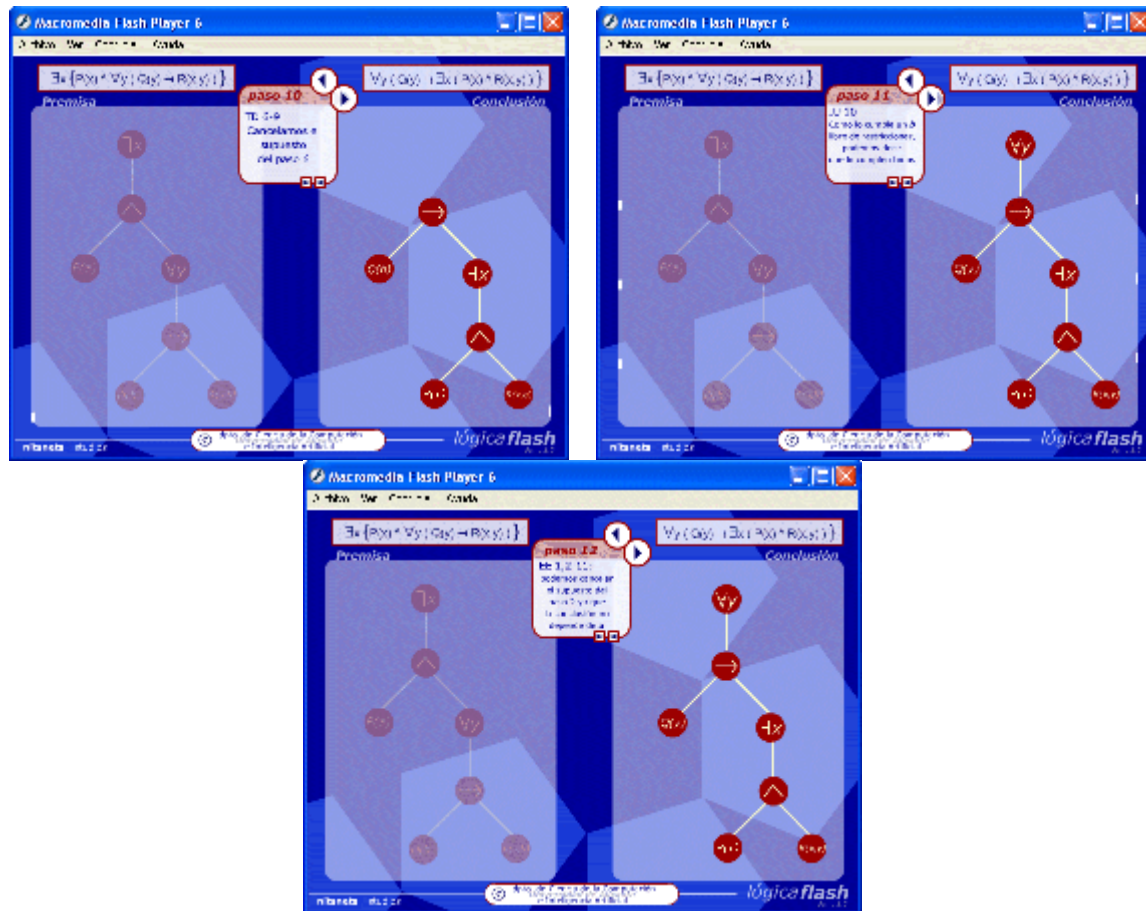


Figura 8: Visualización de una deducción (II)

## 6 ASISTENTE PARA DEDUCCIÓN NATURAL

En la actualidad existe un variado número de herramientas que pueden servir de ayuda en el aprendizaje de la lógica. El *Asistente para Deducción Natural* (ADN) es una herramienta didáctica de apoyo para el aprendizaje de la técnica de deducción natural diseñada y desarrollada en el departamento de Ciencia de la Computación e Inteligencia Artificial de la Universidad de Alicante. Esta herramienta pretende ser un instrumento didáctico que ayude a los estudiantes a escribir fórmulas lógicas bien formadas y a realizar deducciones correctamente. No se trata de un sistema que construya demostraciones de forma automática; simplemente es un asistente que supervisará y guiará al estudiante en el proceso, ayudándole a elaborar sus propias deducciones. Comprueba si la fórmula es sintácticamente correcta (fbf) y si se ha obtenido de forma adecuada (aplicación de las reglas básicas). Además, posee algunas herramientas de apoyo, como el visor de árboles sintácticos de las fórmulas, informe detallado de los errores, visor de reglas básicas, aconsejador y ayuda en línea.

ADN puede ser ejecutado con cualquier navegador, ya que se trata de un applet escrito en Java que se encuentra disponible en internet en el sitio web:

<http://www.dccia.ua.es/logica/ADN>

En la Figura 9 se muestra la pantalla principal del ADN y las partes en las que se divide la misma:

1. Editor de la fórmula objetivo: nos permite editar la fórmula que será el objetivo de la deducción.
2. Cuerpo de la deducción (*pizarra*): en esta zona se irán visualizando los pasos de la deducción. Se pueden observar tres partes (dispuestas en columnas):
  - Numeración de las líneas, para poder hacer referencia a ellas.
  - Fórmulas lógicas que vamos obteniendo. Las sangrías indican que entramos en un nuevo supuesto (hacia la derecha) o que lo cancelamos (vuelta a la izquierda), configurando lo que llamamos subdeducciones.
  - Justificación de la fórmula obtenida mediante la aplicación de una regla básica a una o más fórmulas anteriores.
3. Editor de fórmulas: se utiliza para insertar nuevas fórmulas dentro de la deducción y decir de qué fórmulas han derivado las mismas.
4. Área de opciones del programa: aquí tenemos los botones que acceden a las diferentes funciones del asistente: Árbol, Aconsejar, Ayuda, Reglas, ...
5. Ventana de información: en esta área se muestra información al usuario.

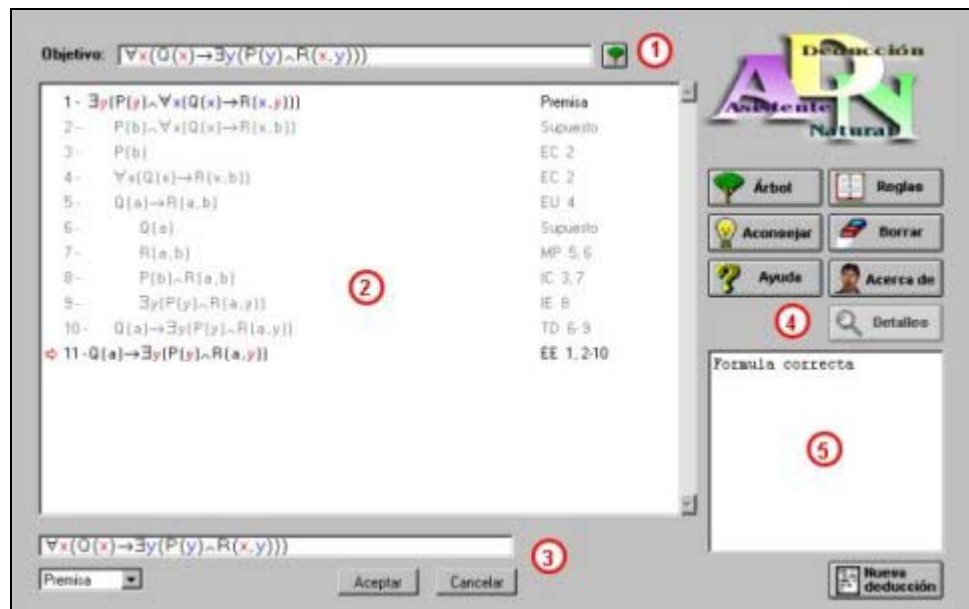


Figura 9: Ventana ADN

El ADN dispone de una herramienta muy útil a la hora de realizar una deducción. Esta herramienta es el Aconsejador. El aconsejador analiza las fórmulas de la deducción e intenta guiarnos hacia el objetivo o decirnos qué reglas podemos aplicar. En cualquier momento podemos poner en marcha el aconsejador, y se puede dejar visible durante toda la deducción, con lo que se irá

actualizando conforme añadamos o eliminemos fórmulas. Aunque no siempre nos llevará a la solución, ya que la deducción natural en lógica de primer orden no es un problema decidible, si que puede servirnos de apoyo. Esto nos lleva directamente al siguiente punto en el cual reflexionamos sobre como automatizar la técnica de deducción.



## DEDUCCIÓN AUTOMÁTICA

### 1 REFLEXIONES PARA LA AUTOMATIZACIÓN

La automatización del razonamiento es un sueño muy antiguo desde los diagramas lógicos de Ramón Llull hasta el autómatas construido por relés que jugaba finales de ajedrez construido por Torres Quevedo. Pero no se ha hecho realidad hasta la aparición de los ordenadores. En 1956 Newell y Simon desarrollaron el “Logic Theorist” que demostró treinta y ocho teoremas de los primeros cincuenta y dos que aparecían en el capítulo segundo de los *Principia Matemática* de Russell y Whitehead. Desde entonces la inteligencia artificial ha avanzado mucho y por caminos no presentidos.

Si intentamos automatizar el proceso de deducción natural nos encontramos con una serie de inconvenientes, que vamos a ir solucionando, aunque algunas veces sea a costa de restringir la potencia de la lógica. La siguiente tabla muestra los problemas con que nos podemos encontrar para automatizar la técnica de deducción natural y las propuestas que han ido surgiendo para su resolución y que desarrollaremos en los siguientes apartados.

Inconveniente	Solución
Dominio (infinitos)	Universo de Herbrand
Manejo cuantificadores	Unificador más general
Notación múltiple	Normalización: forma clausal
Estrategias de demostración	Refutación o reducción al absurdo
Elevado número de reglas	Regla de resolución

Figura 10: Reflexión sobre automatización

### 2 UNIVERSO DE HERBRAND Y UNIFICACIÓN

En el Cálculo de Proposiciones, una fórmula con  $n$  variables proposicionales distintas, con valores de verdad en el conjunto  $\{V, F\}$ , tiene  $2^n$  interpretaciones. En el lenguaje lógico que estamos utilizando disponemos del

concepto de variable para referenciar a un elemento cualquiera del dominio. Estos dominios pueden tener infinitos elementos. Para reducir el tratamiento Herbrand planteó un modelo para el análisis de validez de una fórmula basado en la insatisfactibilidad y definió un universo abstracto el *Universo de Herbrand*. Demostró que si una fórmula no es satisfactible en este universo entonces tampoco lo será en otros dominios. El universo de Herbrand asociado a una fórmula es el conjunto de todos los términos sin variables que se pueden construir con las constantes y los símbolos de función del alfabeto de la fórmula.

Por otro lado hemos visto que para manipular los cuantificadores mediante las reglas de inferencia, debíamos decidir que constantes utilizábamos. Este cambio de las variables por constantes o particularización se llama *sustitución*. Debemos tener en cuenta que todas las ocurrencias de una variable deben ser sustituidas por el mismo término (constante, variable o función) y que las sustituciones siempre se hacen de poner términos donde había variables.

Llamamos *unificación* al proceso de encontrar sustituciones de términos que hagan idénticas ciertas fórmulas lógicas. El unificador no tiene porque ser único. En este caso debemos ahora pensar cuál es el unificador que más nos conviene. Nos interesa un unificador lo más simple posible para dejar la fórmula lo más abierta posible para posteriores sustituciones. A esta sustitución la llamaremos el *unificador más general* (umg).

### 3 FORMA CLAUSAL

Poder escribir la misma información de dos maneras distintas no es aconsejable cuando intentamos hacer un tratamiento automático. Y sin embargo sabemos que podemos escribir una sentencia con diferentes fórmulas lógicas equivalentes. Por otra parte si hacemos un estudio semántico de las conectivas lógicas nos damos cuenta que no son imprescindibles todas las que hemos visto, es decir, podemos reducir el conjunto de conectivas lógicas. Una propuesta es utilizar únicamente la negación, la conjunción y la disyunción. Si además ordenamos la aparición de estas, de forma que las conjunciones estén lo más externamente posible y la negación en el interior, obtenemos fórmulas más adecuadas para un tratamiento automático.

Utilizaremos como base la notación en *forma clausal* que siendo más sencilla que la notación vista es igual de poderosa. Cualquier fórmula lógica puede ser normalizada transformándola a forma clausal, que consiste en una colección (conjunción) de cláusulas, donde cada cláusula es a su vez una disyunción de literales, siendo estos literales fórmulas atómicas afirmadas o negadas.

Para transformar a forma clausal partiremos de una fórmula bien formada de la lógica que no tenga variables libres y seguiremos los pasos enumerados en la figura.



1. Reducir conectivas lógicas: eliminación de los implicadores y coimplicadores que puedan aparecer en la fórmula original.
2. Normalizar negadores: interiorización de los negadores de manera que cada negador quede directamente adosado a una fórmula atómica.
3. Normalizar variables: si es necesario, renombrar las variables de forma que cada cuantificador tenga como índice una variable con nombre distinto (variables distintas, nombres distintos).
4. Eliminar cuantificadores existenciales: para lo cual utilizaremos las constantes y las funciones de skolem.
5. Forma prenexa: adelantar los cuantificadores universales a la cabeza de la fórmula.
6. Eliminar cuantificadores universales: llegados a este punto, sabemos que todas las variables que aparecen están cuantificadas universalmente, por tanto para simplificar la notación no escribiremos los símbolos  $\forall$ .
7. Exteriorizar conjuntores: distribuir la disyunción respecto a la conjunción para dejar fuera los conjuntores y dentro los disyuntores.
8. Extraer las cláusulas: eliminación de conjunciones (separación en cláusulas) y obtención de las distintas cláusulas.
9. Normalizar variables: si es necesario, volvemos a renombrar las variables.

Figura 11: Transformación a forma clausal

## 4 SISTEMAS DE REFUTACIÓN

Los sistemas de demostración automática están concebidos para producir demostraciones por reducción al absurdo o *refutaciones*. Esta estrategia ya la hemos comentado al hablar de la deducción natural. Enunciado de una manera más usual de la demostración automática diremos que un sistema de refutación sigue los siguientes pasos:

- Partimos de un conjunto de fórmulas bien formadas  $C$  (premisas) a partir del cual deseamos demostrar cierta fbf objetivo  $O$  (conclusión).
- Negamos la fbf objetivo ( $\neg O$ ) y añadimos ésta al conjunto de fórmulas  $C$
- Si encontramos una contradicción, entonces podemos concluir que la fórmula objetivo  $O$  se deduce del conjunto de fórmulas  $C$ .

## 5 REGLA DE RESOLUCIÓN

Un hecho primordial en el desarrollo de la demostración automática de teoremas fue el descubrimiento del *principio de resolución* por J. Alan Robinson, que nos dice cómo puede derivarse una nueva proposición a partir de otras dos dadas.

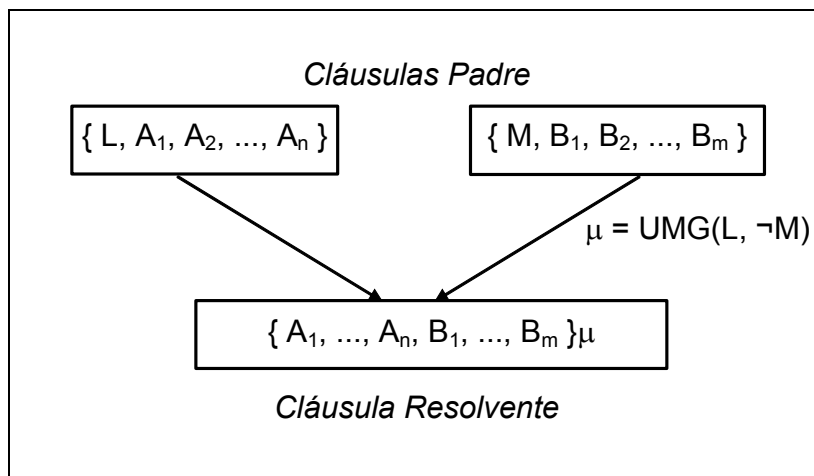


Figura 12: **Regla de resolución**

Se basa en la notación clausal y tiene la característica de encontrar las contradicciones, lo que la hace especialmente adecuada para hacer demostraciones aplicando la técnica de refutación. Pero ahora vamos a restringir el tipo de cláusulas a utilizar. Trabajaremos únicamente con *cláusulas de Horn definidas*, es decir, aquellas que tienen como máximo un átomo afirmado. Esto nos permitirá una mejor y más eficiente implementación de la comprobación de teoremas. Podemos diferenciar los siguientes tipos de cláusulas de Horn:

1. Hipótesis o cláusulas con un literal no negado, llamadas cláusulas con *cabeza*, que pueden ser a su vez *hechos* si no tienen literales negados y *reglas* con literales negados.
2. *Preguntas* o cláusulas con ningún literal no negado, llamadas cláusulas sin *cabeza* o *decapitadas*.

Hay trabajos demostrando que todo problema resoluble con cláusulas generales, tiene un modelo equivalente resoluble en Cláusulas de Horn, por lo que las restricciones de notación no conllevan una pérdida de capacidad de representación, y en cambio, sí que facilitan la formulación. Por tanto la regla de resolución quedaría:

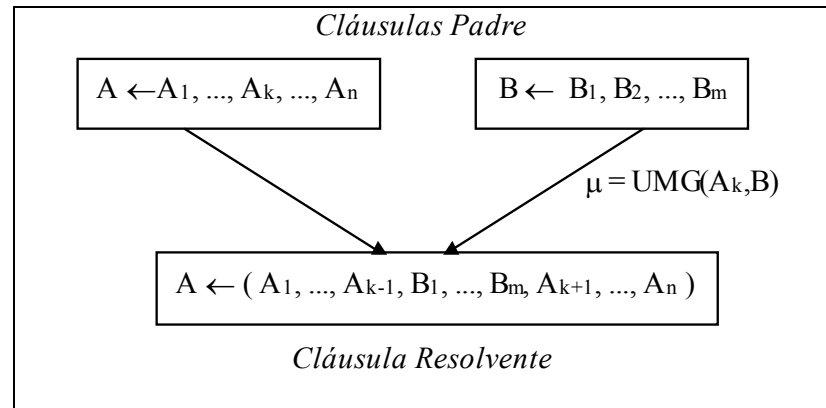


Figura 13: Regla de resolución con cláusulas de horn definidas

El resolvente obtenido al aplicar la regla de resolución no es único, ya que puede haber más de una manera de escoger los literales a unificar ( $L$  y  $M$ ). Además, la elección de las cláusulas padre a resolver de entre el conjunto de cláusulas disponible tampoco es única. Estas elecciones son importantes y dan lugar a las *estrategias de resolución*. Hay por tanto dos decisiones a tomar al hacer la resolución:

- 1) la selección de las cláusulas a unificar y
- 2) la selección de los átomos a unificar dentro de esas cláusulas.

El punto 1 se conoce como *regla de búsqueda* y el punto 2 como *regla de selección*. Vamos en lo que sigue a estudiar qué estrategias utilizaremos en nuestras demostraciones automáticas.

Como ya hemos dicho, a partir de este momento vamos a trabajar únicamente con cláusulas de horn definidas, es decir, aquellas que tienen como máximo un átomo afirmado. Distintos estudios han demostrado que todo problema representado en cláusulas de horn será resoluble si tenemos sólo una cláusula decapitada, y el resto con cabeza, es decir, si sólo existe una pregunta. Esto es fácil de ver de forma intuitiva:

- Al menos una cláusula decapitada: al aplicar el método de resolución a dos cláusulas de Horn con cabeza, obtenemos como resultado otra cláusula con cabeza, con lo que no podríamos derivar la cláusula vacía.

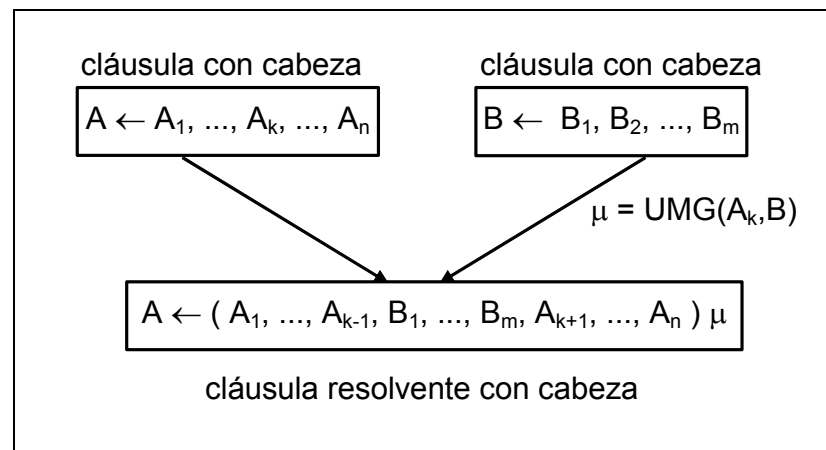


Figura 14: Regla de resolución con cláusulas con cabeza

- Sólo una cláusula decapitada: de existir varias cláusulas decapitadas, cualquier prueba por resolución de una nueva cláusula puede ser convertida en una prueba usando como mucho una de ellas. Al resolver con la primera cláusula decapitada, obtenemos una nueva cláusula decapitada, y por tanto no necesitamos resolver con las otras cláusulas decapitadas.

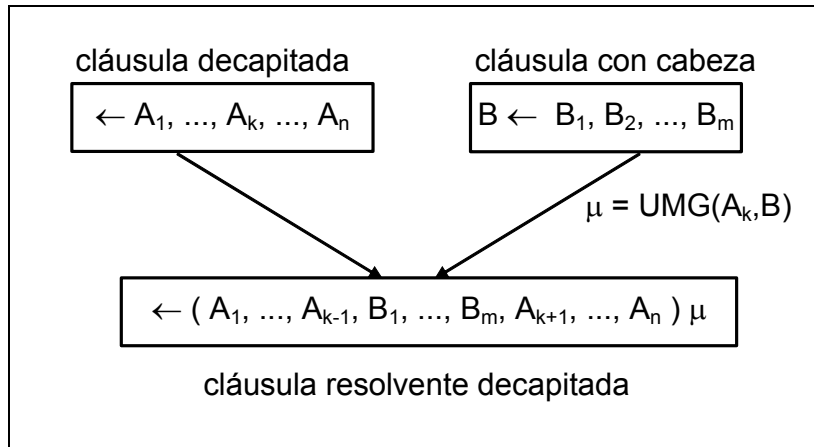


Figura 15: Regla de resolución con una cláusula decapitada

Basándonos en este resultado definiremos una estrategia de resolución que partirá de la cláusula sin cabeza y trata de resolver cláusulas con cabeza con cláusulas objetivo (sin cabeza). Esta estrategia se le conoce como *resolución lineal*.

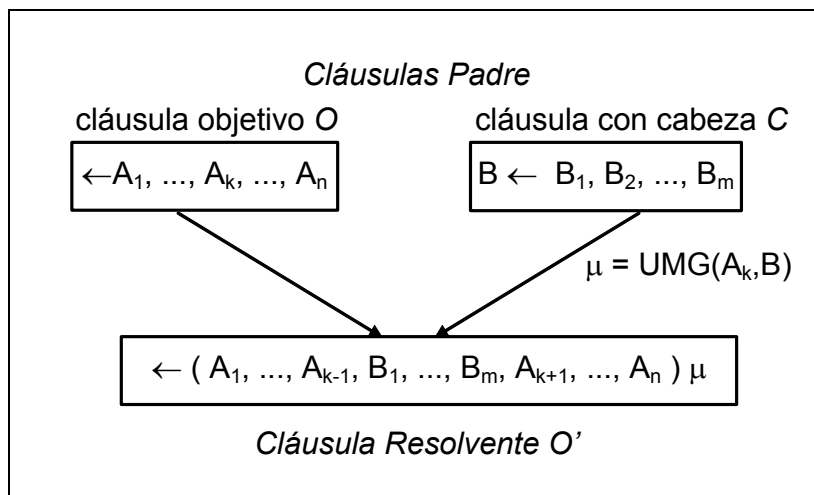


Figura 16: Resolución lineal con cláusulas de horn definidas

Como regla de selección utilizaremos la de unificar el átomo más a la izquierda de la cláusula objetivo. Así, la regla de resolución quedaría:

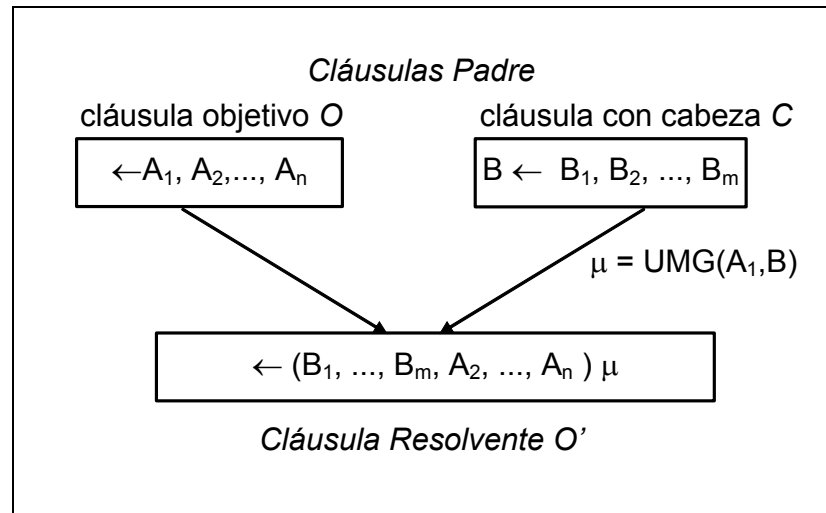


Figura 17: Resolución lineal con cláusulas de horn definidas con elección del átomo más a la izquierda

A esta regla la llamamos SLD-Resolución porque:

- utilizamos cláusulas de Horn Definidas
- utilizamos la estrategia de búsqueda Lineal: primero en profundidad
- utilizamos una regla de Selección: primero a la izquierda

Al utilizar resolución lineal el programa lógico estará razonando hacia atrás desde el objetivo propuesto hasta que encuentre el modo de terminar utilizando las cláusulas del programa, es decir se parte de las metas a conseguir. Este método de razonamiento se denomina también *razonamiento dirigido al objetivo*. Esto resultará adecuado para sistemas interrogadores.

Una importante propiedad formal que presenta la resolución es la de ser de *refutación completa*, es decir, si un conjunto de cláusulas no es *coherente*, la resolución podrá deducir de ellas la cláusula vacía. Así, si nuestras cláusulas (programa lógico) son coherentes, sólo tenemos que añadirles la negación de lo que queremos probar. Si por resolución deducimos la cláusula vacía, nuestro objetivo quedará probado.

## 6 PROGRAMACIÓN LÓGICA Y PROLOG

Esta elección de una estrategia fija (SLD-Resolución) en el control de la búsqueda hace que el programador se despreocupe del control y se centre en la especificación del problema a resolver. Empleada como un lenguaje para comunicarse con los ordenadores, la lógica representa un formalismo de nivel superior y más orientado a la persona que otros lenguajes de programación. Toda esta base teórica dio como resultado la aparición de una clase de lenguajes de programación, la *programación lógica*.

Para poder acercarnos en lo posible a los lenguajes de programación, a partir de ahora, para las fórmulas lógicas en forma clausal utilizaremos la siguiente notación específica de la programación lógica (es una notación más intuitiva y cercana a una interpretación procedimental):

1. Como es una colección de cláusulas, las escribiremos en secuencia. Recordemos que el orden es irrelevante ya que se trata de conjunciones (propiedades asociativa y conmutativa de la conjunción).
2. Cada cláusula es una colección (disyunción) de literales, que son fórmulas atómicas afirmadas o negadas:
  - Escribiremos primero las fórmulas atómicas no negadas llamadas *cabeza de la cláusula*, seguidas de las negadas, *cuerpo de la cláusula*, separados ambos grupos por el símbolo especial " $\leftarrow$ " (si).
  - Las fórmulas atómicas no negadas (cabeza de la cláusula) irán separadas por el símbolo ";" (o), y las negadas (cuerpo de la cláusula) serán escritas sin el negador ( $\neg$ ) y separadas por el símbolo "," (y).

Como la notación de la programación lógica es engorrosa para introducir en el ordenador ya que los símbolos lógicos no aparecen en el teclado, modificaremos la sintaxis para el lenguaje de programación *prolog*:

- Las variables se identifican porque empiezan por mayúsculas.
- El símbolo lógico  $\leftarrow$  se escribe como :-
- El objetivo (pregunta) no forma parte del programa y se introduce desde la interfaz de prolog.
- Las cláusulas se separan por punto (.). De esta forma una cláusula puede escribirse en más de una línea.

## EJEMPLO

## 1 FORMALIZACIÓN

**A** continuación vemos un ejemplo que realiza todos los pasos comentados y que nos ayudará a entenderlos mejor. El sentido común no dice que el siguiente argumento es correcto, pero vamos a demostrarlo formalmente.

- El marido de la hermana del padre de alguien es su tío
- Dos personas son hermanos si tienen los mismos padres
- Mi padre se llama David
- Los padres de David son Eva y Blas
- Los padres de Ana son Eva y Blas
- Ana está casada con Carlos

*Luego, al menos tengo un tío*

Como paso previo a la demostración vamos a formalizar el argumento utilizando el lenguaje de la lógica de primer orden. Para ello emplearemos los siguientes símbolos:

- *Predicados:*

$C(x,y)$	$x$ (marido) está casado con $y$ (esposa)
$H(x,y)$	$x$ e $y$ son hermanos
$P(x,y)$	$x$ es el padre de $y$
$M(x,y)$	$x$ es la madre de $y$
$T(x,y)$	$x$ es el tío de $y$
- *Constantes:*

yo	yo
a	Ana
b	Blas
c	Carlos
d	David
e	Eva

El dominio en el que trabajamos, y por tanto en el que tomarán valores las variables y al que pertenecen las constantes, es el de las personas:  $D = \{\text{personas}\}$ .

Con esta interpretación para los símbolos, la formalización de las fórmulas que forman nuestro argumento quedará:

- $\forall x \forall w \{ \exists y \exists z [ C(x,y) \wedge H(y,z) \wedge P(z,w) ] \rightarrow T(x,w) \}$

- $\forall x \forall y \{ \exists z \exists w [ P(z,x) \wedge P(z,y) \wedge M(w,x) \wedge M(w,y) ] \rightarrow H(x,y) \}$
  - $P(d,yo)$
  - $M(e,d) \wedge P(b,d)$
  - $M(e,a) \wedge P(b,a)$
  - $C(c,a)$
- Luego,  $\exists x T(x,yo)$

Para finalizar con esta introducción veamos gráficamente la estructura de las fórmulas de las dos primeras premisas:

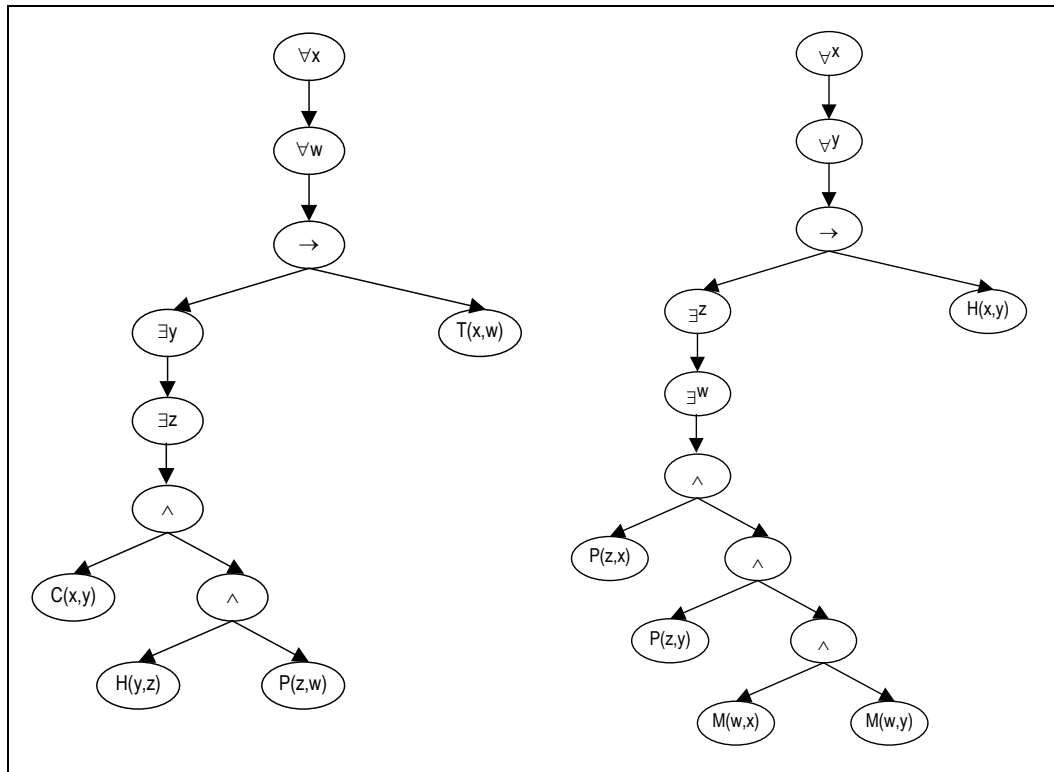


Figura 18: Estructura sintáctica fórmulas ejemplo

## 2 DEDUCCIÓN NATURAL

1	$\forall x \forall w \{ \exists y \exists z [ C(x,y) \wedge H(y,z) \wedge P(z,w) ] \rightarrow T(x,w) \}$	Premisa
2	$\forall x \forall y \{ \exists z \exists w [ P(z,x) \wedge P(z,y) \wedge M(w,x) \wedge M(w,y) ] \rightarrow H(x,y) \}$	Premisa
3	$P(d,yo)$	Premisa
4	$M(e,d) \wedge P(b,d)$	Premisa
5	$M(e,a) \wedge P(b,a)$	Premisa
6	$C(c,a)$	Premisa
7	$\forall w \{ \exists y \exists z [ C(c,y) \wedge H(y,z) \wedge P(z,w) ] \rightarrow T(c,w) \}$	EU 1
8	$\exists y \exists z [ C(c,y) \wedge H(y,z) \wedge P(z,yo) ] \rightarrow T(c,yo)$	EU 7
9	$\forall y \{ \exists z \exists w [ P(z,a) \wedge P(z,y) \wedge M(w,a) \wedge M(w,y) ] \rightarrow H(a,y) \}$	EU 2
10	$\exists z \exists w [ P(z,a) \wedge P(z,d) \wedge M(w,a) \wedge M(w,d) ] \rightarrow H(a,d)$	EU 9
11	$M(e,d) \wedge P(b,d) \wedge M(e,a) \wedge P(b,a)$	IC 4,5



12	$P(b,a) \wedge P(b,d) \wedge M(e,a) \wedge M(e,d)$	Co 11
13	$\exists w [ P(b,a) \wedge P(b,d) \wedge M(w,a) \wedge M(w,d) ]$	IE 12
14	$\exists z \exists w [ P(z,a) \wedge P(z,d) \wedge M(w,a) \wedge M(w,d) ]$	IE 13
15	$H(a,d)$	MP 10,14
16	$C(c,a) \wedge H(a,d)$	IC 6,15
17	$C(c,a) \wedge H(a,d) \wedge P(d,yo)$	IC 16,3
18	$\exists z [ C(c,a) \wedge H(a,z) \wedge P(z,yo) ]$	IE 17
19	$\exists y \exists z [ C(c,y) \wedge H(y,z) \wedge P(z,yo) ]$	IE 18
20	$T(c,yo)$	MP 8,19
21	$\exists x T(x,yo)$	IE 20

### 3 DEDUCCIÓN AUTOMÁTICA

#### 3.1 Refutación y Forma Clausal

Para aplicar la técnica de refutación tomamos las premisas afirmadas y la conclusión negada, e intentamos encontrar una contradicción. Para buscar la contradicción utilizaremos la regla de resolución. Por tanto, para poder aplicar la regla de resolución, deberemos previamente transformar las fórmulas a forma clausal:

**Premisa 1:**  $\forall x \forall w \{ \exists y \exists z [ C(x,y) \wedge H(y,z) \wedge P(z,w) ] \rightarrow T(x,w) \}$

1ª Eliminar implicadores y coimplicadores:

$$\forall x \forall w \{ \neg \exists y \exists z [ C(x,y) \wedge H(y,z) \wedge P(z,w) ] \vee T(x,w) \}$$

2ª Normalizar negadores:

$$\forall x \forall w \{ \forall y \neg \exists z [ C(x,y) \wedge H(y,z) \wedge P(z,w) ] \vee T(x,w) \}$$

$$\forall x \forall w \{ \forall y \forall z \neg [ C(x,y) \wedge H(y,z) \wedge P(z,w) ] \vee T(x,w) \}$$

$$\forall x \forall w \{ \forall y \forall z [ \neg C(x,y) \vee \neg H(y,z) \vee \neg P(z,w) ] \vee T(x,w) \}$$

3ª Normalizar variables:

$$\forall x \forall w \{ \forall y \forall z [ \neg C(x,y) \vee \neg H(y,z) \vee \neg P(z,w) ] \vee T(x,w) \}$$

4ª Eliminar cuantificadores existenciales:

$$\forall x \forall w \{ \forall y \forall z [ \neg C(x,y) \vee \neg H(y,z) \vee \neg P(z,w) ] \vee T(x,w) \}$$

5ª Forma prenexa:

$$\forall x \forall w \forall y \forall z \{ [ \neg C(x,y) \vee \neg H(y,z) \vee \neg P(z,w) ] \vee T(x,w) \}$$

6ª Eliminar cuantificadores universales:

$$[ \neg C(x,y) \vee \neg H(y,z) \vee \neg P(z,w) ] \vee T(x,w)$$

7ª Poner en FNC:

$$\neg C(x,y) \vee \neg H(y,z) \vee \neg P(z,w) \vee T(x,w)$$

**Premisa 2:**  $\forall x \forall y \{ \exists z \exists w [ P(z,x) \wedge P(z,y) \wedge M(w,x) \wedge M(w,y) ] \rightarrow H(x,y) \}$

1ª Eliminar implicadores y coimplicadores:

$$\forall x \forall y \{ \neg \exists z \exists w [ P(z,x) \wedge P(z,y) \wedge M(w,x) \wedge M(w,y) ] \vee H(x,y) \}$$

2ª Normalizar negadores:

$$\forall x \forall y \{ \forall z \neg \exists w [ P(z,x) \wedge P(z,y) \wedge M(w,x) \wedge M(w,y) ] \vee H(x,y) \}$$

$$\forall x \forall y \{ \forall z \forall w \neg [ P(z,x) \wedge P(z,y) \wedge M(w,x) \wedge M(w,y) ] \vee H(x,y) \}$$

$$\forall x \forall y \{ \forall z \forall w [ \neg P(z,x) \vee \neg P(z,y) \vee \neg M(w,x) \vee \neg M(w,y) ] \vee H(x,y) \}$$

3ª Normalizar variables:

$$\forall x \forall y \{ \forall z \forall w [ \neg P(z,x) \vee \neg P(z,y) \vee \neg M(w,x) \vee \neg M(w,y) ] \vee H(x,y) \}$$

4ª Eliminar cuantificadores existenciales:

$$\forall x \forall y \{ \forall z \forall w [ \neg P(z,x) \vee \neg P(z,y) \vee \neg M(w,x) \vee \neg M(w,y) ] \vee H(x,y) \}$$

5ª Forma prenexa:

$$\forall x \forall y \forall z \forall w \{ [ \neg P(z,x) \vee \neg P(z,y) \vee \neg M(w,x) \vee \neg M(w,y) ] \vee H(x,y) \}$$

6ª Eliminar cuantificadores universales:

$$[ \neg P(z,x) \vee \neg P(z,y) \vee \neg M(w,x) \vee \neg M(w,y) ] \vee H(x,y)$$

7ª Poner en FNC:

$$\neg P(z,x) \vee \neg P(z,y) \vee \neg M(w,x) \vee \neg M(w,y) \vee H(x,y)$$

Premisa 3:  $P(d,yo)$

Premisa 4:  $M(e,d) \wedge P(b,d)$

Premisa 5:  $M(e,a) \wedge P(b,a)$

Premisa 6:  $C(c,a)$

Conclusión negada:  $\neg \exists x T(x,yo)$

1ª Eliminar implicadores y coimplicadores:

$$\neg \exists x T(x,yo)$$

2ª Normalizar negadores:

$$\forall x \neg T(x,yo)$$

3ª Normalizar variables:

$$\forall x \neg T(x,yo)$$

4ª Eliminar cuantificadores existenciales:

$$\forall x \neg T(x,yo)$$

5ª Forma prenexa:

$$\forall x \neg T(x,yo)$$

6ª Eliminar cuantificadores universales:

$$\neg T(x,yo)$$

7ª Poner en FNC:

$$\neg T(x,yo)$$

Por lo que hemos obtenido el siguiente conjunto de cláusulas:

- C<sub>1</sub>:  $\neg C(x_1,y_1) \vee \neg H(y_1,z_1) \vee \neg P(z_1,w_1) \vee T(x_1,w_1)$
- C<sub>2</sub>:  $\neg P(z_2,x_2) \vee \neg P(z_2,y_2) \vee \neg M(w_2,x_2) \vee \neg M(w_2,y_2) \vee H(x_2,y_2)$
- C<sub>3</sub>:  $P(d,yo)$
- C<sub>4</sub>:  $M(e,d)$
- C<sub>5</sub>:  $P(b,d)$
- C<sub>6</sub>:  $M(e,a)$
- C<sub>7</sub>:  $P(b,a)$
- C<sub>8</sub>:  $C(c,a)$
- C<sub>9</sub>:  $\neg T(x_3,yo)$

### 3.2 Regla de Resolución con Unificación

Ahora ya podemos aplicar la regla de resolución a este conjunto de cláusulas para buscar la contradicción. Tendremos también en cuenta el concepto de unificación y escribiremos al lado del árbol de refutación las sustituciones (el unificador más general - umg) que utilizemos. Los valores que podrán tomar las distintas variables estarán en el Universo de Herbrand:

$$H = \{a, b, c, d, e, yo\}$$

Vamos a utilizar la representación en forma de árbol para visualizar la manera en que se van obteniendo nuevas cláusulas hasta la producción de la cláusula NADA.

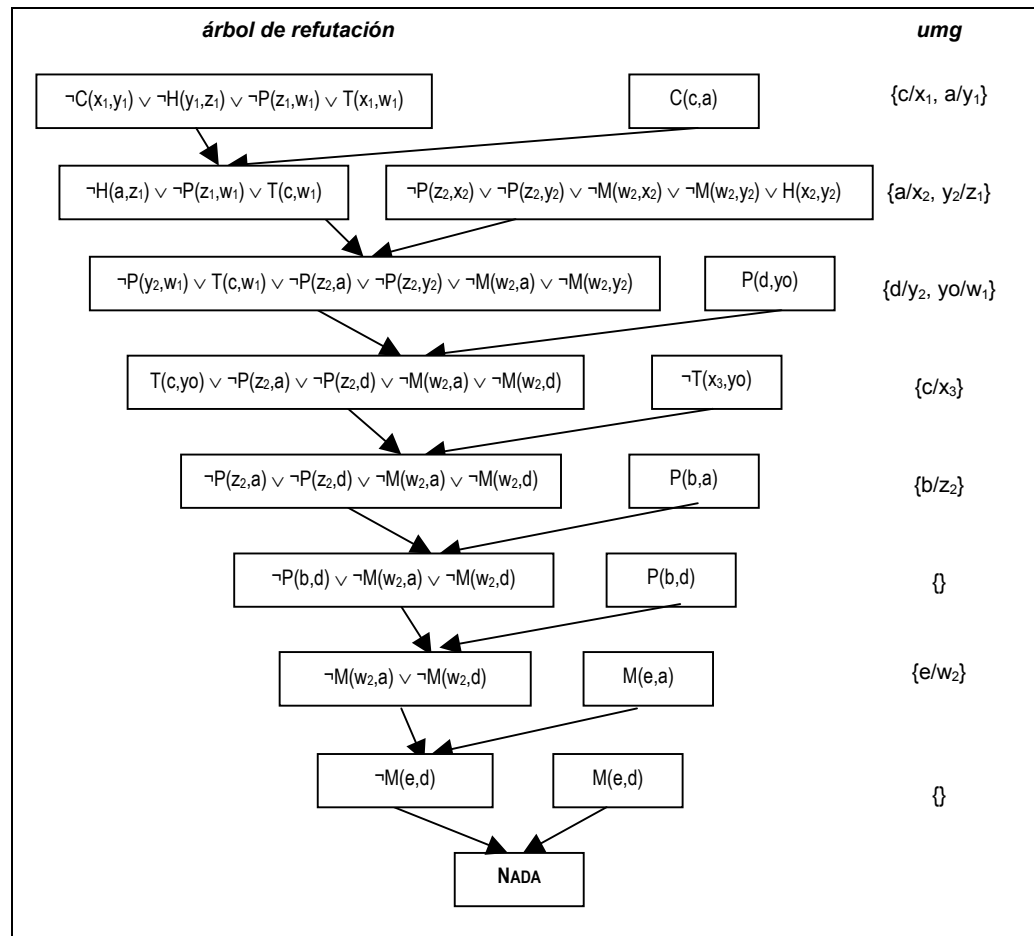


Figura 19: Árbol de refutación con regla de resolución

Como hemos derivado la cláusula NADA, podemos afirmar que el conjunto de cláusulas es insatisfacible, es decir, que no podemos encontrar ninguna interpretación que las haga todas verdaderas al mismo tiempo. Como base de conocimiento procede de afirmar las premisas y negar la conclusión, lo que indica es que no se pueden dar premisas ciertas y conclusión falsa, es decir que si las premisas son verdaderas, la conclusión también lo será. Dicho de otra manera, que la deducción es CORRECTA. Además, la sustitución que nos ha permitido derivar la cláusula NADA es:

$$\begin{aligned} & \{c/x_1, a/y_1\} \circ \{a/x_2, y_2/z_1\} \circ \{d/y_2, yo/w_1\} \circ \{c/x_3\} \circ \{b/z_2\} \circ \{e/w_2\} = \\ & = \{c/x_1, a/x_2, c/x_3, a/y_1, d/y_2, d/z_1, b/z_2, yo/w_1, e/w_2\} \end{aligned}$$

Por tanto, no sólo hemos deducido que  $\exists x T(x,y_0)$ , sino que también hemos obtenido un valor para ese alguien ( $x$ ). Como la variable de la forma clausal que correspondía era  $x_3$ , y como el *umg* nos dice que tenemos que sustituir la  $x_3$  por  $c$ , podemos afirmar que  $c$  (Carlos) es mi tío.

## 4 PROGRAMACIÓN LÓGICA

### 4.1 Programa Lógico

Utilizando la sintaxis de la programación lógica (que es más intuitiva que la forma clausal) tendremos el siguiente programa lógico:

- C<sub>1</sub>:  $T(x_1, w_1) \leftarrow C(x_1, y_1), H(y_1, z_1), P(z_1, w_1)$
- C<sub>2</sub>:  $H(x_2, y_2) \leftarrow P(z_2, x_2), P(z_2, y_2), M(w_2, x_2), M(w_2, y_2)$
- C<sub>3</sub>:  $P(d, y_0) \leftarrow$
- C<sub>4</sub>:  $M(e, d) \leftarrow$
- C<sub>5</sub>:  $P(b, d) \leftarrow$
- C<sub>6</sub>:  $M(e, a) \leftarrow$
- C<sub>7</sub>:  $P(b, a) \leftarrow$
- C<sub>8</sub>:  $C(c, a) \leftarrow$
- C<sub>9</sub>:  $\leftarrow T(x_3, y_0)$

### 4.2 SLD-Resolución

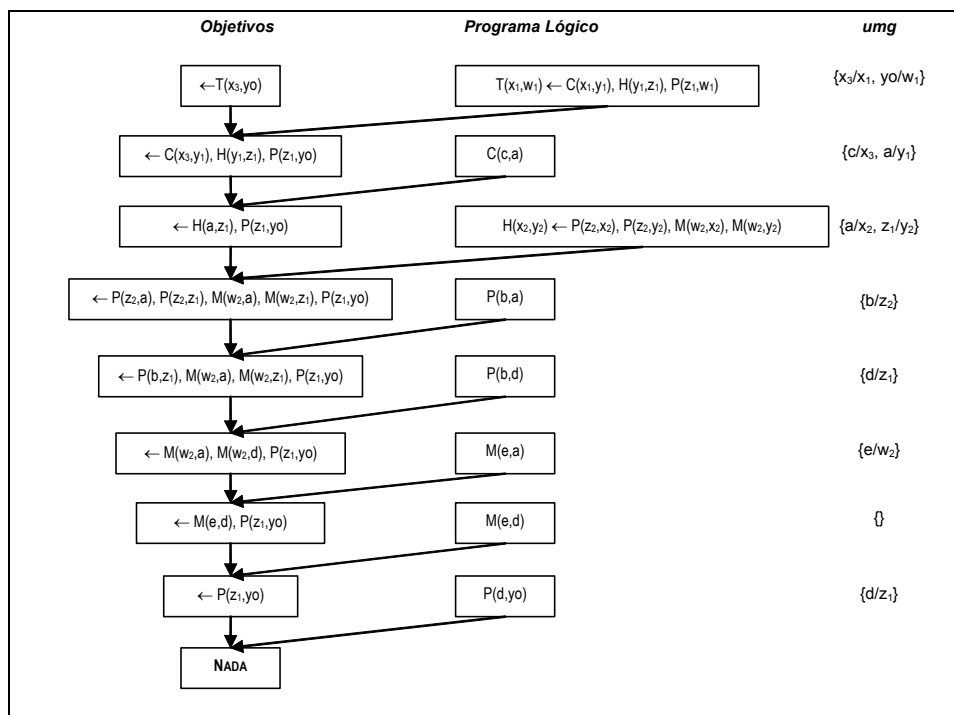


Figura 20: Árbol de refutación programa lógico

### 4.3 Prolog

Utilizando la sintaxis del lenguaje de programación prolog, tendremos la siguiente base de conocimientos (hechos y reglas):

```
tio(X1,W1) :-
    casados(X1,Y1),
    hermanos(Y1,Z1),
    padre(Z1,W1).
hermanos(X2,Y2) :-
    padre(Z2,X2),
    padre(Z2,Y2),
    madre(W2,X2),
    madre(W2,Y2) .
padre(david,yo).
madre(eva,david).
padre(blas,david).
madre(eva,ana).
padre(blas,ana).
casados(carlos,ana).
```

Y dada la siguiente *pregunta* :  
**?- tio(X,yo).**

Prolog respondería:  
**X=carlos**

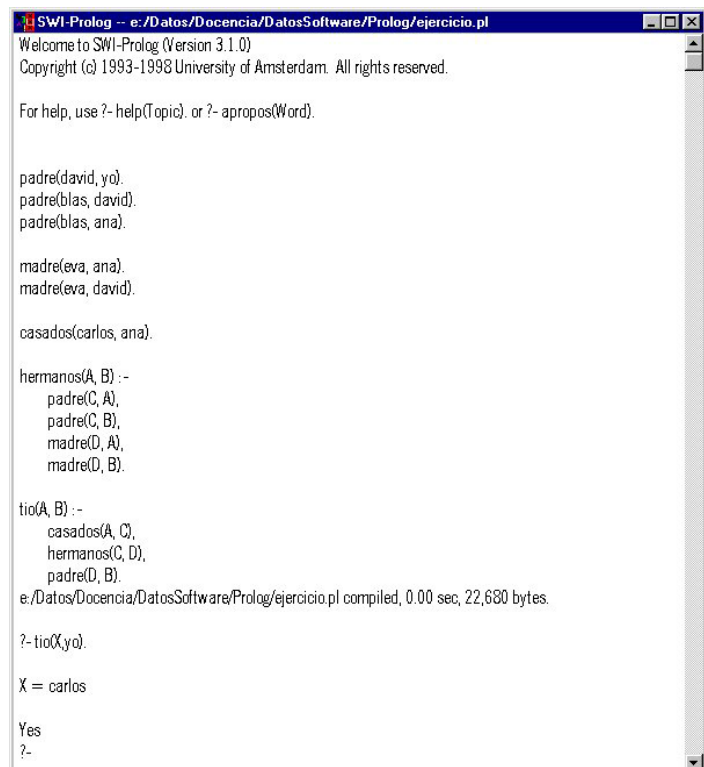
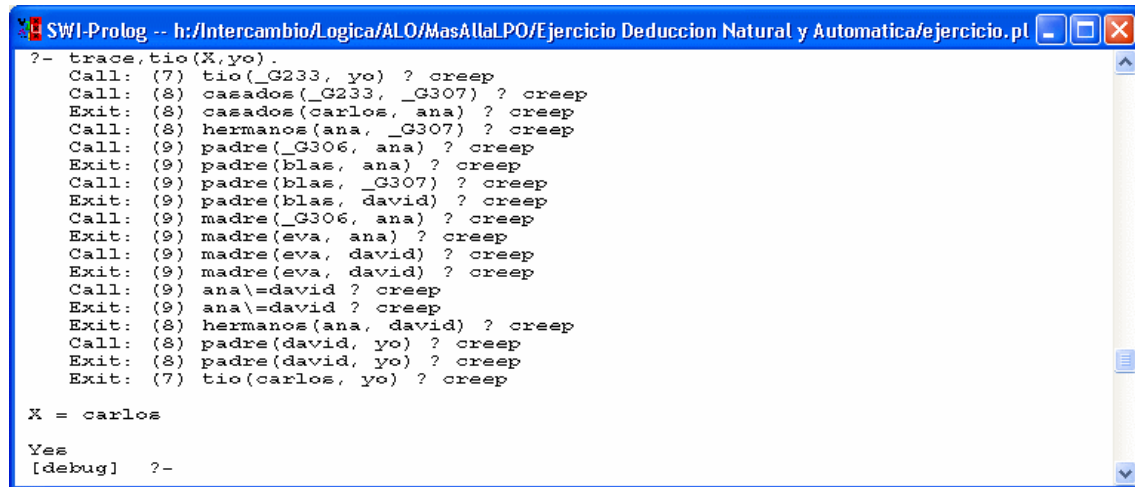


Figura 21: Base de conocimiento prolog

Si activamos la traza de prolog vemos los pasos que ha realizado para obtener la respuesta:



```

?- trace, tio(X, yo).
Call: (7) tio(_G233, yo) ? creep
Call: (8) casados(_G233, _G307) ? creep
Exit: (8) casados(carlos, ana) ? creep
Call: (8) hermanos(ana, _G307) ? creep
Call: (9) padre(_G306, ana) ? creep
Exit: (9) padre(blas, ana) ? creep
Call: (9) padre(blas, _G307) ? creep
Exit: (9) padre(blas, david) ? creep
Call: (9) madre(_G306, ana) ? creep
Exit: (9) madre(eva, ana) ? creep
Call: (9) madre(eva, david) ? creep
Exit: (9) madre(eva, david) ? creep
Call: (9) ana\=david ? creep
Exit: (9) ana\=david ? creep
Exit: (8) hermanos(ana, david) ? creep
Call: (8) padre(david, yo) ? creep
Exit: (8) padre(david, yo) ? creep
Exit: (7) tio(carlos, yo) ? creep

X = carlos
Yes
[debug] ?-

```

Figura 22: Taza prolog

## BIBLIOGRAFÍA

- Barwise, J. y Etchemendy, J. (2000). *Language, Proof and Logic*. Seven Bridges Press / CSLI Publications, Text / Software Package
- Bratko, I. (1990). *PROLOG Programming for Artificial Intelligence*. 2ª, Addison-Wesley,
- Callear, D. (1994). *Prolog Programming for Students. With Expert Systems and Artificial Intelligence Topics*. Continuum,
- Clocksin, W. F. y Mellish, C. S. (1987). *Programación en Prolog*. Gustavo Gili,
- Deransart, P., Ed-Dbali, A., y Cervoni, L. (1996). *Prolog: The Standard. Reference Manual*. Springer-Verlag, New York.
- Dodd, T. (1990). *Prolog. A Logical Approach*. Oxford University Press,
- Garrido, M. (1991). *Lógica Simbólica*. 2ª ed., Editorial Tecnos,
- Giannesini, F., Kanoui, H., Pasero, R., y van Caneghem, M. (1989). *Prolog*. Addison-Wesley Iberoamericana,
- Goldson, D., Reeves, S. y Bornat, R. (1993). "A Review of Several Programs for the Teaching of Logic", *The Computer Journal*, vol. 36, no. 4, 1993.
- Llorens, F. (2001). *Sistemas de Razonamiento y Conocimiento Distribuido. Agentes Inteligentes*. Ramón Rizo (director). Doctor Ingeniero (PhD), Tesis Doctoral. Dpto. Ciencia de la Computación e Inteligencia Artificial, Universidad de Alicante.
- Llorens, F. y Castel, M. J. (1999). *Lógica de Primer Orden*. 2ª edición, Imprime: Ramón Torres Gosálvez, Alicante. Dpto. Ciencia de la Computación e Inteligencia Artificial, Universidad de Alicante
- Llorens, F. y Mira, S. (2000a), "ADN (Asistente para Deducción Natural) Natural Deduction Assistant," en *Proceedings of the First International Congress on Tools for Teaching Logic*, Universidad de Salamanca, Cursos Extraordinarios, Salamanca, pp. 65-70.
- Llorens, F. y Mira, S. (2000b), "Herramienta para la enseñanza de la *Deducción Natural*," en *VI Jornadas sobre la Enseñanza Universitaria de la Informática*, Servicio de Publicaciones, Universidad de Alcalá, Alcalá de Henares, pp. 496-502.
- Llorens, F. y Mira, S. (2002), "ADN: una herramienta para la enseñanza de la *Deducción Natural*," en *Aportaciones de la Didáctica de la Matemática a diferentes perfiles profesionales*, M. C. Penalva, G. Torregrosa y J. Valls (eds.), Universidad de Alicante, Alicante, pp. 447-460.
- Mira, S. (2000). *Asistente para Deducción Natural (ADN)*. Faraón Llorens (director). PFC Ingeniero en Informática. Dpto. Ciencia de la Computación e Inteligencia Artificial, Escuela Politécnica Superior, Universidad de Alicante.
- Reeves, S. y Clarke, M. (1993). *Logic for Computer Science*. Reprinted. First printed 1990, International Computer Science Series, Addison-Wesley Publishing,

- Socher-Ambrosius, R. y Johann, P. (1997). *Deduction Systems*. Graduate Texts in Computer Science, Springer-Verlag, New York.
- TTL (June de 2000). "First International Congress on Tools for Teaching Logic", June de 2000, University of Salamanca, Spain,
- Tymoczko, T. y Henle, J. (2002). *Razón, dulce razón. Una guía de campo de la lógica moderna*. Editorial Ariel, Barcelona.