Lógica Proposicional y de Primer Orden

Fernando Sancho Caparrini

SVRAI – MULCIA
Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla

24 de octubre de 2022

Problema básico

Dados un conjunto de **afirmaciones** (hechos, hipótesis,...), $\mathcal{BC}$, y una afirmación A, **decidir** si A ha de ser **necesariamente cierta** cuando todas las afirmaciones de $\mathcal{BC}$ lo son.

La **Lógica** proporciona **formulaciones precisas** de este problema y **diferentes soluciones**. Para abordar este problema formalmente hemos de:

1. Diseñar un **lenguaje** para expresar las afirmaciones (*representación*).
2. Concretar qué entendemos por **afirmación cierta**.
3. Proporcionar mecanismos efectivos para razonar que garanticen la **corrección de las deducciones**.

A lo largo de este curso estudiaremos estas cuestiones en los dos casos más comunes: la **Lógica Proposicional**, y la **Lógica de Primer Orden**.

Lógica Proposicional

Lógica Proposicional

Características generales de la Lógica Proposicional (LP):

- Sus expresiones (fórmulas) representan *afirmaciones* (hechos, oraciones) que pueden considerarse **verdaderas o falsas**.
- Se construyen a partir de expresiones básicas otras más complejas usando operadores (**conectivas**).
- Las conectivas se corresponden con formas sencillas de construir afirmaciones complejas en el lenguaje natural partiendo de otras más sencillas:
  - Conjunción: "...tal ... **y** ...cual..."
  - Disyunción: "...tal ... **o** ...cual ..."
  - Implicación "**Si** ...tal ... **entonces**...cual ..."
  - Negación: "**No** es cierto que tal ..."
- Sólo permite analizar las formas de razonamiento ligadas a este tipo de construcciones.

Argumentos y formalización en LP: Ejemplos

Ejemplo 1: *Si el tren llega a las 7 y no hay taxis en la estación, entonces Juan llegará tarde a la reunión. Juan no ha llegado tarde a la reunión. El tren llegó a las 7.* **Por tanto,** *habían taxis en la estación.*

Ejemplo 2: *Si hay corriente y la lámpara no está fundida, entonces está encendida. La lámpara no está encendida. Hay corriente.* **Por tanto,** *la lámpara está fundida.*

- **Simbolización**:

Simb.	Ejemplo 1	Ejemplo 2
p	el tren llega a las 7	hay corriente
q	hay taxis en la estación	la lámpara está fundida
r	Juan llega tarde a la reunión	la lámpara está encendida

- **Razonamiento (informal)**: Si p y no q, entonces r. No r. p. Por tanto, q.
- **Razonamiento formal (¿cómo?)**: $\{p \wedge \neg q \rightarrow r,\ \neg r,\ p\} \models q$.

El lenguaje de la Lógica Proposicional

El *lenguaje de la Lógica Proposicional* consta de:

1. Un conjunto numerable de **variables proposicionales**:
   $VP = \{p, p_0, p_1, \ldots, q, q_0, q_1, \ldots, r, r_0, \ldots\}$
2. Operadores básicos de contrucción, **Conectivas lógicas**:
   - De aridad 1: $\neg$ (negación).
   - De aridad 2: $\vee$ (disyunción), $\wedge$ (conjunción), $\rightarrow$ (condicional) y $\leftrightarrow$ (bicondicional).
3. **Símbolos auxiliares**: "(" y ")" (para leer fácilmente las expresiones).

Fórmulas

El conjunto de las *fórmulas proposicionales*, **PROP**, es el menor conjunto de expresiones que verifica:

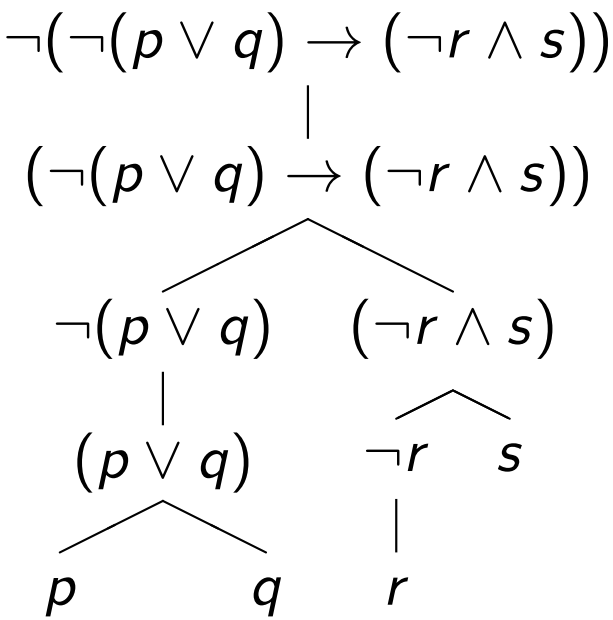
- $VP \subseteq \mathbf{PROP}$,
- Es cerrado bajo las conectivas, es decir:
  - Si $F \in \mathbf{PROP}$, entonces $\neg F \in \mathbf{PROP}$.
  - Si $F, G \in \mathbf{PROP}$, entonces $(F \vee G)$, $(F \wedge G)$, $(F \rightarrow G)$, $(F \leftrightarrow G) \in \mathbf{PROP}$.

- La sintaxis del lenguaje pretende **evitar la ambigüedad en la interpretación de las fórmulas** (esa es la función de los símbolos auxiliares).

Cuestión: ¿Existe ambigüedad de lectura (como ocurre a veces en el lenguaje humano)?

Árboles de formación

- Asociamos a cada fórmula un **árbol de formación** (esencialmente único) que describe el modo en que se construye la fórmula a partir de otras más sencillas.
- **Ejemplo**:



- Las fórmulas que aparecen en el árbol de formación de una fórmula F se denominan **subfórmulas** de F.

Reducción de paréntesis

Para facilitar la lectura de las fórmulas adoptaremos los siguientes convenios de notación:

- **Omitiremos** los paréntesis externos.
- Daremos a las conectivas una **precedencia de asociación**. De **mayor a menor preferencia están ordenadas por**:

$$\neg, \wedge, \vee, \rightarrow$$

  Ejemplo: $F \wedge G \rightarrow \neg F \vee G$ es $((F \wedge G) \rightarrow (\neg F \vee G))$.
- **Siempre se dejarán los paréntesis para $\leftrightarrow$.**
- Cuando una conectiva se usa repetidamente, se asocia por la derecha: $F \vee G \vee H$ es $(F \vee (G \vee H))$.

Principio de Inducción sobre fórmulas

Gracias a la definición de **PROP** si deseamos probar que toda fórmula proposicional satisface cierta propiedad Ψ, podemos probarlo por **inducción sobre fórmulas**.

Para ello probamos:

1. **Caso base:** Todos los elementos de VP tienen la propiedad Ψ.
2. **Paso de inducción:**
   1. Si $F \in$ **PROP** tiene la propiedad Ψ, entonces $\neg F$ tiene la propiedad Ψ.
   2. Si $F, G \in$ **PROP** tienen la propiedad Ψ, entonces las fórmulas $(F \vee G)$, $(F \wedge G)$, $(F \rightarrow G)$ y $(F \leftrightarrow G)$ también tienen la propiedad Ψ.

Semántica: Funciones de verdad

- Los elementos del conjunto $\{0, 1\}$ se llaman **valores de verdad**. Se dice que 0 es el valor **falso** y el 1 es el valor **verdadero**.
- El *significado* de una conectiva se determina mediante su **función de verdad** (una *función booleana*):
  - $H_{\neg}(i) = \begin{cases} 1, & \text{si } i = 0; \\ 0, & \text{si } i = 1. \end{cases}$
  - $H_{\vee}(i,j) = \begin{cases} 0, & \text{si } i = j = 0; \\ 1, & \text{en otro caso.} \end{cases}$
  - $H_{\wedge}(i,j) = \begin{cases} 1, & \text{si } i = j = 1; \\ 0, & \text{en otro caso.} \end{cases}$
  - $H_{\rightarrow}(i,j) = \begin{cases} 0, & \text{si } i = 1, j = 0; \\ 1, & \text{en otro caso.} \end{cases}$
  - $H_{\leftrightarrow}(i,j) = \begin{cases} 1, & \text{si } i = j; \\ 0, & \text{en otro caso.} \end{cases}$

Valoraciones

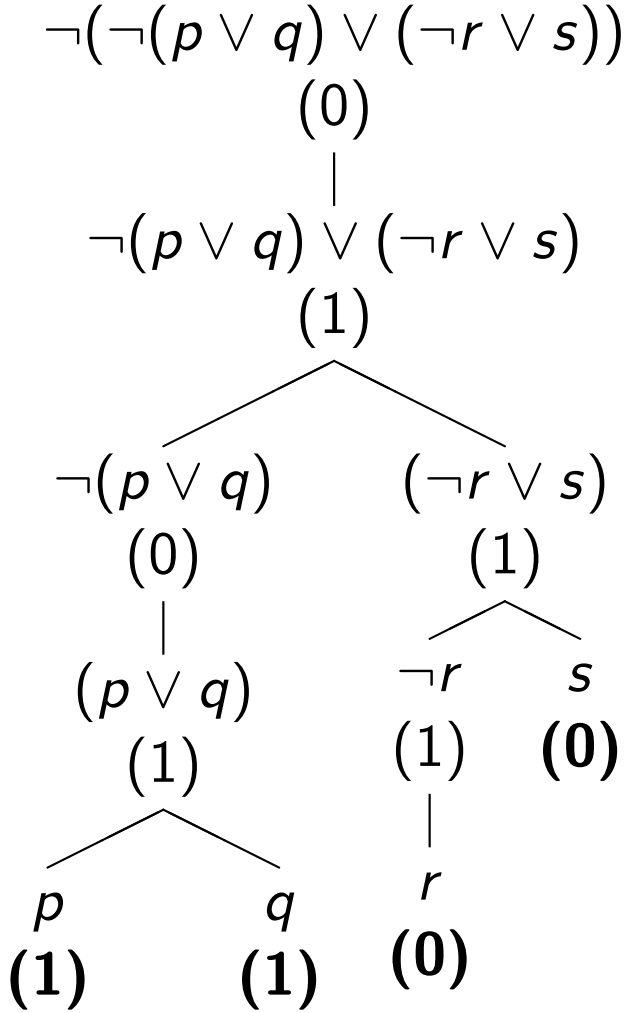
- Las variables proposicionales se interpretan mediante una **valoración de verdad** (o **interpretación**), una aplicación

$$v : VP \to \{0, 1\}$$

- **Se prueba por inducción** que podemos extender cada valoración, v, **de forma única**, al conjunto de todas las fórmulas verificando que para toda fórmula F se verifique:
  - $v(\neg F) = H_{\neg}(v(F))$.
  - $v((F \vee G)) = H_{\vee}(v(F), v(G))$.
  - $v((F \wedge G)) = H_{\wedge}(v(F), v(G))$.
  - $v((F \to G)) = H_{\to}(v(F), v(G))$.
  - $v((F \leftrightarrow G)) = H_{\leftrightarrow}(v(F), v(G))$.
- Se dice que $v(F)$ es el **valor de verdad** de F respecto de v.

Valor de verdad (con árboles de formación)

Supongamos $v(p) = v(q) = \mathbf{1}$ y $v(r) = v(s) = \mathbf{0}$. Usando el árbol de formación para calcular (de abajo a arriba) el valor $v(\neg(\neg(p \vee q) \vee (\neg r \vee s))$:



Tablas de verdad

Como se ve con el árbol, el valor de verdad de una fórmula F respecto de una v **está determinado por los valores de verdad de las subfórmulas** de F.

Ejemplo: si $v(p) = v(q) = 0$ y $v(r) = 1$, entonces

$$\begin{aligned} v(\neg((p \to q) \vee r)) &= H_\neg(H_\vee(v(p \to q), v(r))) = \\ &= H_\neg(H_\vee(H_\to(v(p), v(q)), 1)) = 0 \end{aligned}$$

Fijada v podemos presentar el cálculo de F mediante una tabla que recorre los valores de sus subfórmulas:

p	q	r	$p \to q$	$(p \to q) \vee r$	$\neg((p \to q) \vee r)$
0	0	1	1	1	0

Una **tabla de verdad** para F es una tabla similar que contiene **una fila por cada posible valoración** que asigne valores a las variables proposicionales que aparecen en F.

Ejemplo

p	q	r	$(p \vee q)$	$\neg(p \vee q)$	$(p \to r)$	$\neg(p \vee q) \to (p \to r)$
V	V	V	V	F	V	V
V	V	F	V	F	F	V
V	F	V	V	F	V	V
V	F	F	V	F	F	V
F	V	V	V	F	V	V
F	V	F	V	F	V	V
F	F	V	F	V	V	V
F	F	F	F	V	V	V

Validez y satisfactibilidad (I)

- Decimos que una fórmula F es **válida en** v, o que v es un **modelo** de F, si $v(F) = 1$.
  - Notación: $v \models F$.
  - Una valoración v es *modelo* de un conjunto de fórmulas U, $v \models U$, si v es modelo de todas las fórmulas de U.
- Una fórmula F es una **tautología** (o **válida**) si es válida para toda valoración (notación $\models F$).
- Una fórmula F es **satisfactible** (o consistente) si existe una valoración que es modelo de F. En caso contrario diremos que es **insatisfactible** (o inconsistente).
  - Análogamente, un conjunto de fórmulas U es satisfactible (o consistente) si existe una valoración que es modelo de U. En caso contrario diremos que es insatisfactible (o inconsistente).
- Una fórmula F es **contingente** si es consistente pero no tautología

Clasificaciones de fórmulas

<table>
<tr><td colspan="3">Todas las fórmulas</td></tr>
<tr><td>Tautologías</td><td>Contigentes</td><td>Contradicciones</td></tr>
<tr><td>Verdadera en todas las interpretaciones

(ej. $p \vee \neg p$)</td><td>Verdadera en algunas interpretaciones y falsa en otras

(ej. $p \rightarrow q$)</td><td>Falsa en todas las interpretaciones

(ej. $p \wedge \neg p$)</td></tr>
<tr><td colspan="2">Safisfacibles</td><td>Insatisfacibles</td></tr>
<tr><td colspan="3">Todas las fórmulas</td></tr>
</table>

Validez y satisfactibilidad (II)

Relación entre conceptos:

Lema. Para cada $F \in$ **PROP** se verifica:

- Si F es un tautología entonces F es satisfactible.
- F es una tautología si y sólo si $\neg F$ insatisfactible.

Ejemplos:

- Son tautologías: $(p \vee \neg p)$ y $((p \to q) \to p) \to p$.
- $p \wedge \neg p$ es insatisfactible y, por tanto, $\neg(p \wedge \neg p)$ es una tautología.
- $(p \to q) \to p$ es satisfactible pero no es una tautología.

Consecuencia Lógica

- Una fórmula F es **consecuencia lógica** de un conjunto de fórmulas U, si todo modelo de U es modelo de F. Es decir,
  para toda valoración, v,

$$v \models U \quad \Longrightarrow \quad v \models F$$

  - Notación: $U \models F$.
  - La relación de consecuencia lógica permite formular el problema básico en el marco de la lógica proposicional.

Relación entre consecuencia lógica, consistencia y validez:

Proposición. Sea $\{F_1, \ldots F_n\} \subseteq$ **PROP**. Son equivalentes:

- $\{F_1, \ldots, F_n\} \models F$
- $F_1 \wedge \cdots \wedge F_n \rightarrow F$ es un tautología.
- $\{F_1, \ldots, F_n, \neg F\}$ es insatisfactible.

Ejemplos

$\{p \to q, q \to r\} \models p \to r$

	p	q	r	$p \to q$	$q \to r$	$p \to r$
I_1	0	0	0	1	1	1
I_2	0	0	1	1	1	1
I_3	0	1	0	1	0	1
I_4	0	1	1	1	1	1
I_5	1	0	0	0	1	0
I_6	1	0	1	0	1	1
I_7	1	1	0	1	0	0
I_8	1	1	1	1	1	1

$\{p\} \not\models p \wedge q$

p	q	$p \wedge q$
1	1	1
1	0	0
0	1	0
0	0	0

Algoritmos de decisión (I)

Dado un conjunto de fórmulas proposicionales, U, un **algoritmo de decisión** para U es un algoritmo que dada $A \in PROP$, devuelve `SI` cuando $A \in U$, y `NO` si $A \notin U$.

Casos especialmente interesantes:

- **SAT** $= \{A \in PROP : \ A$ es satisfactible$\}$
- **TAUT** $= \{A \in PROP : \ A$ es una tautología$\}$
- Fijado $U \subseteq PROP$, la **Teoría de** U es

$$\mathcal{T}(U) = \{A \in PROP : \ U \models A\}$$

Un algoritmo de decisión para $\mathcal{T}(U)$ propociona una respuesta al Problema Básico expuesto al principio del tema.

Algoritmos de decisión (II)

Problema Básico:
Diseñar un algoritmo que, dado un conjunto finito de fórmulas proposicionales, U, y una fórmula F, **decida** si $U \models F$.

El problema se reduce a decidir la **satisfactibilidad** de una cierta fórmula (o si se prefiere, la **validez** de otra) por la proposición anterior. Por tanto,

- La construcción de tablas de verdad induce un algoritmo (ineficiente) para decidir la consecuencia lógica.
- El Problema Básico es resoluble algorítmicamente, aunque no se conoce ninguna solución eficiente y se duda de la existencia de algoritmos de decisión eficientes para este problema, ya que ...
- ... determinar la satisfactibilidad de una fórmula proposicional es un problema **NP-completo**.

Algoritmos de decisión (III)

Problema Básico (bis):
Obtener un algoritmo eficiente que, dado un conjunto finito de fórmulas proposicionales, U, y una fórmula F, decida si $U \models F$.

Observaciones:

- **Este problema es equivalente** al de obtener un algoritmo eficiente para determinar la satisfactibilidad de una fórmula proposicional.
- Se trata de un **problema abierto**, que posiblemente tendrá una respuesta negativa (se cree que no existen algoritmos eficientes para resolver SAT).
- **Para propósitos prácticos puede bastar con algoritmos eficientes para alguna clase especial de fórmulas.**

Limitaciones de la lógica proposicional

- La lógica proposicional posee un semántica sencilla y existen algoritmos de decisión para sus problemas básicos, como SAT o la consecuencia lógica.
- Sin embargo, la expresividad de la lógica proposicional es bastante limitada.
- **Existen problemas cuya descripción mediante lógica proposicional es complicada**, pues requieren un gran número de fórmulas o fórmulas de gran tamaño.
- Más aún, **existen formas de razonamiento válidas que no pueden ser expresadas mediante la lógica proposicional**, por ejemplo:
  - Todos los hombres son mortales
  - Sócrates es un hombre.
  - Por tanto, Sócrates es mortal.
- La **Lógica de Primer Orden** extiende a la Lógica Proposicional proporcionando mayor expresividad.

Ejemplo de argumentación

- Problema de los animales: Se sabe que
  1. Los animales con pelo o que dan leche son mamíferos.
  2. Los mamíferos que tienen pezuñas o que rumian son ungulados.
  3. Los ungulados de cuello largo son jirafas.
  4. Los ungulados con rayas negras son cebras.

  Se observa un animal que tiene pelos, pezuñas y rayas negras. Por consiguiente, se concluye que el animal es una cebra.
- Formalización:

$$\{\ \text{tiene\_pelos} \vee \text{da\_leche} \rightarrow \text{es\_mamífero},$$
$$\text{es\_mamífero} \wedge (\text{tiene\_pezuñas} \vee \text{rumia}) \rightarrow \text{es\_ungulado},$$
$$\text{es\_ungulado} \wedge \text{tiene\_cuello\_largo} \rightarrow \text{es\_jirafa},$$
$$\text{es\_ungulado} \wedge \text{tiene\_rayas\_negras} \rightarrow \text{es\_cebra},$$
$$\text{tiene\_pelos} \wedge \text{tiene\_pezuñas} \wedge \text{tiene\_rayas\_negras}\}$$
$$\models \text{es\_cebra}$$

Problemas lógicos: veraces y mentirosos

- Enunciado: En una isla hay dos tribus, la de los veraces (que siempre dicen la verdad) y la de los mentirosos (que siempre mienten). Un viajero se encuentra con tres isleños A, B y C y cada uno le dice una frase
  1. A dice "B y C son veraces syss C es veraz"
  2. B dice "Si A y C son veraces, entonces B y C son veraces y A es mentiroso"
  3. C dice "B es mentiroso syss A o B es veraz"

  Determinar a qué tribu pertenecen A, B y C.
- Simbolización: a: "A es veraz", b: "B es veraz", c: "C es veraz".
- Formalización:
  $F_1 = a \leftrightarrow (b \wedge c \leftrightarrow c)$, $F_2 = b \leftrightarrow (a \wedge c \rightarrow b \wedge c \wedge \neg a)$ y
  $F_3 = c \leftrightarrow (\neg b \leftrightarrow a \vee b)$.
- Modelos de $\{F_1, F_2, F_3\}$:
  Si I es modelo de $\{F_1, F_2, F_3\}$, entonces $I(a) = 1, I(b) = 1, I(c) = 0$.
- Conclusión: A y B son veraces y C es mentiroso.

Limitación expresiva de la lógica proposicional

- **Ejemplo 1**: *Si Sevilla es vecina de Cádiz, entonces Cádiz es vecina de Sevilla. Sevilla es vecina de Cádiz. Por tanto, Cádiz es vecina de Sevilla*
  - Representación en lógica proposicional:
    $$\{SvC \rightarrow CvS, \quad SvC\} \models CvS$$
- **Ejemplo 2**: *Si una ciudad es vecina de otra, entonces la segunda es vecina de la primera. Sevilla es vecina de Cádiz. Por tanto, Cádiz es vecina de Sevilla*
  - Representación en lógica proposicional: **Imposible**
  - Veremos la representación en **lógica de primer orden**:

$$\{\forall x\, \forall y\, [vecina(x, y) \rightarrow vecina(y, x)], vecina(Sevilla, Cadiz)\} \models vecina(Cadiz, Sevilla)$$

Lógica de Primer Orden

Ejemplo (I)

Consideremos las siguientes afirmaciones:

1. Marco era pompeyano.
2. Todos los pompeyanos eran romanos.
3. Cada romano, o era leal a César, o le odiaba.
4. Todo el mundo es leal a alguien.
5. La gente sólo intenta asesinar a aquellos a quienes no es leal.
6. Marco intentó asesinar a César.
7. Todo pompeyano es leal a su padre.

¿Podemos deducir a partir de esta información que Marco era leal a César? ¿Podemos deducir que Marco odiaba a César? ¿Era César el padre de Marco?

Ejemplo (II)

- Podemos formalizar las afirmaciones observando que todas ellas expresan propiedades de los elementos de un cierto conjunto de individuos (romanos) y las relaciones que se dan entre ellos.
- Introduzcamos símbolos para expresar estas relaciones y para referirnos a los individuos de los que estamos hablando:
  - $P(x)$ expresa "x es pompeyano".
  - $R(x)$ expresa "x es romano".
  - $L(x, y)$: "x es leal a y".
  - $O(x, y)$: "x odia a y".
  - $IA(x, y)$: "x intentó asesinar a y".
  - Por último, parece natural introducir una función f que para cada x, devuelve el padre de x, $f(x)$.

Ejemplo (III)

Ahora podemos formalizar los enunciados anteriores:

1. $P(\textbf{Marco})$ expresa "Marco es pompeyano"
2. $\forall x\,(P(x) \rightarrow R(x))$
   - "Todos los pompeyanos son romanos"
3. $\forall x\,(R(x) \rightarrow (L(x, \textbf{Cesar}) \vee O(x, \textbf{Cesar}))$
   - "Todo romano es leal a César o le odia"
4. $\forall x\,\exists y\,L(x, y)$
   - "Todo el mundo es leal a alguien".
5. $\forall x\,\forall y\,(IA(x, y) \rightarrow \neg L(x, y))$
   - "La gente sólo intenta asesinar a aquellos a quienes no es leal".
6. $IA(\textbf{Marco}, \textbf{Cesar})$
   - "Marco intentó asesinar a César".
7. $\forall x\,(P(x) \rightarrow L(x, f(x)))$
   - "Todo pompeyano es leal a su padre".

Ejemplo (IV)

- Para las preguntas podemos escribir:
  a. $L(\textbf{Marco}, \textbf{Cesar})$: Marco es leal a César.
  b. $O(\textbf{Marco}, \textbf{Cesar})$: Marco odia a César.
- Sin embargo, no podemos expresar que "Marco es el padre de César" sin considerar algún símbolo más.
- Una posibilidad es añadir a nuestro lenguaje el símbolo "$=$" para expresar al igualdad entre dos objetos. De este modo tendríamos:
  - $f(\textbf{Marco}) = \textbf{Cesar}$: César es el padre de Marco.
- Como puede verse, hemos ampliado el conjunto de símbolos disponibles en la lógica proposicional.
- El conjunto de símbolos introducidos constituye lo que denominamos un **Lenguaje de Primer Orden**.

Lenguajes de Primer Orden

- Un **lenguaje de primer orden** (LPO) L consta de:
  - Símbolos lógicos (comunes a todos los lenguajes):
    1. Un conjunto de **variables**: $V = \{x_0, x_1, \dots\}$.
    2. **Conectivas lógicas**: $\neg$, $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$.
    3. **Cuantificadores**: $\exists$ (existencial), $\forall$ (universal).
    4. **Símbolos auxiliares**: "(", ")ƨ ","
  - Símbolos no lógicos (propios de cada lenguaje):
    1. Un conjunto L_C de **constantes**.
    2. Un conjunto de **símbolos de función** $L_F = \{f_0, f_1, \dots\}$, cada uno con su aridad.
    3. Un conjunto de **símbolos de predicados** $L_P = \{p_0, p_1, \dots\}$, cada uno con su aridad.

  Los conjuntos V, L_F, L_C y L_P son disjuntos
- Los símbolos de predicado de aridad 0 actúan como símbolos proposicionales.
- El símbolo $=$ **no es un predicado común** a todos los LPO. Cuando está incluido diremos que se trata de un **Lenguaje de Primer Orden con igualdad**.

Ejemplos

- En el ejemplo de los romanos se introdujo el lenguaje:

$$LR = \{\underbrace{\textbf{Marco}, \textbf{Cesar}}_{\text{constantes}}, \underbrace{P, L, O, R, IA}_{\text{símb. predicado}}, \underbrace{f}_{\text{símb. función}}\}$$

  P, R y f tienen aridad 1. L, O y IA tienen aridad 2.
- El lenguaje de la Aritmética (números naturales):

$$LA = \{\underbrace{\mathbf{0}, \mathbf{1}}_{\text{constantes}}, \overbrace{<, =}^{\text{símb. predicado}}, \underbrace{\cdot, +}_{\text{símb. de función}}\}$$

  $<$, $+$ y $\cdot$ tienen aridad 2.
- Un lenguaje para el parentesco:

$$LP = \{\underbrace{padre\_de, madre\_de, hijo, hermano, casados}_{\text{símb. predicado}}\}$$

  Todos de aridad 2.

Términos

- Los **términos** de un lenguaje L se definen como:
  1. Las variables y las constantes son términos.
  2. Si $t_1, \ldots, t_n$ son términos y f es un símbolo de función de L de aridad n, entonces $f(t_1, \ldots, t_n)$ es un término.
- Los términos son expresiones que me permiten hablar de los **objetos del mundo**.
- Ejemplos:
  - Son términos del lenguaje LR:

    $$\textbf{Marco}, \textbf{Cesar}, f(x), f(\textbf{Cesar}), f(f(\textbf{Cesar})), \ldots$$

  - Son términos del lenguaje de la Aritmética:

    $$\mathbf{0}, +(x, y), \cdot(x, +(y, \mathbf{1})), \ldots$$

    Utilizando la notación infija tradicional se escriben

    $$x + y,\ x \cdot (y + \mathbf{1})$$

Fórmulas

- Las fórmulas son expresiones que permiten hablar de **veracidad y falsedad**.
- Las **fórmulas atómicas** de L son las expresiones $p(t_1, \ldots, t_n)$, donde p es un símbolo de predicado de aridad n y $t_1, \ldots, t_n$ son términos.
- Las **fórmulas** de L se definen como sigue:
  - Las fórmulas atómicas de L son fórmulas de L.
  - Si F y G son fórmulas de L, entonces $\neg F$, $(F \vee G)$ $(F \wedge G)$, $(F \rightarrow G)$ y $(F \leftrightarrow G)$ también lo son.
  - Si x es una variable y F es una fórmula de L, entonces $\exists x\, F$ y $\forall x\, F$ también son fórmulas.

Ejemplos

- En LA, $\neg\exists x(x \cdot \mathbf{0} = y)$
- En LP, $\exists x(padre\_de(x, y) \wedge padre\_de(x, z))$.
  Pero $\exists x\ padre\_de(padre\_de(x, y), z)$, **NO es una fórmula**.
- En LR,

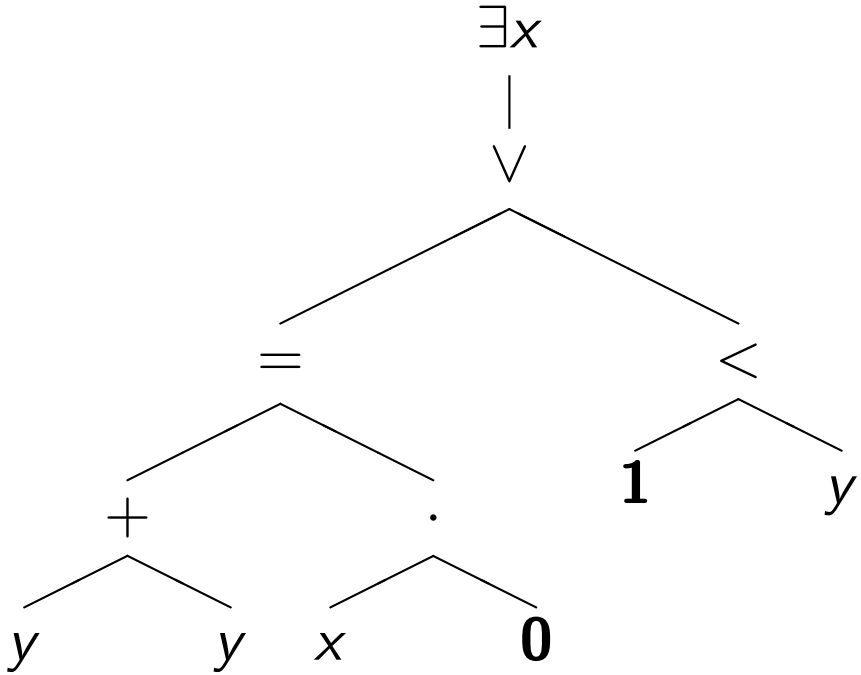
$$\forall x\, \exists y\, L(x, y)$$

$$\forall x\, (R(x) \rightarrow (L(x, \mathbf{Cesar}) \vee O(x, \mathbf{Cesar})))$$

- **Notación**: Para facilitar la lectura de las fórmulas y reducir el número de paréntesis adoptamos los mismos convenios que para la lógica proposicional:
  - Omitiremos los paréntesis externos.
  - Daremos a las conectivas una precedencia de asociación.
    De mayor a menor, están ordenadas por: $\neg, \wedge, \vee, \rightarrow$.
  - Se dejan los paréntesis para la conectiva $\leftrightarrow$.

Árboles de formación

Análisis sintáctico de la expresión $\exists x(y + y = x \cdot \mathbf{0} \vee \mathbf{1} < y)$



O también:

$$\exists x(y + y = x \cdot \mathbf{0} \vee \mathbf{1} < y)$$
$$(y + y = x \cdot \mathbf{0} \vee \mathbf{1} < y)$$
$$y + y = x \cdot \mathbf{0} \qquad \mathbf{1} < y$$

Las fórmulas de los nodos se denominan **subfórmulas**

Estancias libres y ligadas

- Una **estancia ligada** de una variable x en una fórmula F es una aparición de x en una subfórmula del tipo $\exists x\, F$ o $\forall x\, F$. En otro caso, diremos que es una **estancia libre**.
  - **Variable libre** en F: Al menos una estancia libre.
  - **Variable ligada** en F: Al menos una estancia ligada.
- $\exists x \forall y\, (x \cdot y = z \cdot \mathbf{1})$ no es cerrada (z es libre).
- $\exists x\, (\forall y\, (x \cdot y = \mathbf{1}) \vee x \cdot y = x)$ no es cerrada.
  - La variable y aparece libre **y** ligada.
  - Aunque sintácticamente es correcto, no escribiremos fórmulas en las que una misma variable aparezca libre y ligada. Usaremos en su lugar la fórmula

$$\exists x\, (\forall y\, (x \cdot y = \mathbf{1}) \vee (x \cdot z = x))$$

Semántica

- **Objetivo**: Dotar de significado a los términos y fórmulas de un lenguaje de primer orden.
  - Términos cerrados: elementos del universo.
  - Significado de las fórmulas: propiedades sobre los elementos del universo.
- Una *L*-**estructura** (o **interpretación**) $\mathcal{M}$, consta de:
  - Un conjunto no vacío $M \neq \emptyset$ (el **universo** de la estructura).
  - Una interpretación en M para cada símbolo de L:
    1. Para cada constante c, $c^{\mathcal{M}} \in M$.
    2. Para cada función, f, de aridad $n > 0$, $f^{\mathcal{M}} : M^n \to M$.
    3. Para cada predicado, p, de aridad $n > 0$, $p^{\mathcal{M}} : M^n \to \{0, 1\}$ (equiv., $p^{\mathcal{M}} \subseteq M^n$).
    4. Si L es un LPO *con igualdad* la interpretación de $=$ es

$$\{(a, a) : \ a \in M\}$$

- Si no hay confusión, escribiremos M en vez de $\mathcal{M}$, p^M en lugar de $p^{\mathcal{M}}$, etc.

Ejemplos (I)

- Para LP, sea $\mathcal{M}_1$ la estructura dada por:
  - Universo: $M_1 = \{Pedro,\ Pablo,\ Ana,\ Laura\}$.
  - $padre\_de^{M_1} = \{(Pablo, Ana), (Pedro, Pablo)\}$.
  - $madre\_de^{M_1} = \{(Ana, Laura)\}$.
  - $hermano^{M_1} = \emptyset$.
  - $casados^{M_1} = \emptyset$.
- Para LP, consideremos $\mathcal{M}_2$ dada por:
  - Universo: $M_2 = \{0, 1, 2, 3, 4, 5, 6\}$.
  - $padre\_de^{M_2} \equiv$ ser múltiplo de.
  - $madre\_de^{M_2} \equiv$ ser menor.
  - $hermano^{M_2} \equiv$ primos entre sí.
  - $casados^{M_2} = \emptyset$.

Ejemplos (II)

- Para LA, sea $\mathcal{M}_3$ dada por:
  - Universo: $M_3 = \mathbb{N}$
  - $\mathbf{0}^{M_3} = 0$.
  - $\mathbf{1}^{M_3} = 1$.
  - La función $+^{M_3}$ es la suma de números naturales.
  - La función $\cdot^{M_3}$ es el producto de números naturales.
  - $=^{M_3}$ es la igualdad entre números naturales.
  - $<^{M_3}$ es el orden entre números naturales.
- Para LA, sea $\mathcal{M}_4$ dada por:
  - Universo: $M_4 = \mathbb{Q}$
  - $\mathbf{0}^{M_4} = \frac{1}{2}$.
  - $\mathbf{1}^{M_4} = 2$.
  - La función $+^{M_4}$ es la diferencia de números racionales.
  - La función $\cdot^{M_4}$ está dada por $p \cdot^{M_4} q = p$.
  - $=^{M_4}$ es la igualdad entre números naturales.
  - $<^{M_4}$ es el orden entre números racionales.

Interpretación de términos (I)

- Dada una L–estructura $\mathcal{M}$, a cada término t de L, **sin variables**, le corresponde un elemento de M, que denotamos por $t^{\mathcal{M}}$ (su **interpretación** en $\mathcal{M}$):
  - Si $t \equiv c$ una constante, entonces $t^{\mathcal{M}} = c^{\mathcal{M}} \in M$.
  - Si $t \equiv f(t_1, \ldots, t_n)$, entonces $t^{\mathcal{M}} = f^{\mathcal{M}}(t_1^{\mathcal{M}}, \ldots, t_n^{\mathcal{M}})$.

- Ejemplos:

$$\begin{aligned}((\mathbf{0}\cdot\mathbf{1})+\mathbf{1})^{M_3} &= ((\mathbf{0}\cdot\mathbf{1})^{M_3} +^{M_3} \mathbf{1}^{M_3}) \\ &= (\mathbf{0}^{M_3} \cdot^{M_3} \mathbf{1}^{M_3}) + 1 \\ &= (0\cdot 1)+1 = 1\end{aligned}$$

$$\begin{aligned}((\mathbf{0}\cdot\mathbf{1})+\mathbf{1})^{M_4} &= ((\mathbf{0}\cdot\mathbf{1})^{M_4} +^{M_4} \mathbf{1}^{M_4}) \\ &= (\mathbf{0}^{M_4} \cdot^{M_4} \mathbf{1}^{M_4}) - 2 \\ &= (\tfrac{1}{2} \cdot^{M_4} 2) - 2 \\ &= \tfrac{1}{2} - 2 = -\tfrac{3}{2}\end{aligned}$$

Interpretación de términos (II)

- Asociamos a cada L–estructura, $\mathcal{M}$, un lenguaje $L(\mathcal{M})$, que tiene todos los símbolos de L y, además, una constante $\underline{a}$ por cada elemento $a \in M$.
- La interpretación de los símbolos de $L(\mathcal{M})$ en $\mathcal{M}$ es la misma para los símbolos de L, y para cada $a \in M$,

$$\underline{a}^{\mathcal{M}} = a$$

- Ahora podemos calcular $t^{\mathcal{M}}$ para todo término de $L(\mathcal{M})$ sin variables:

$$\begin{aligned} ((\underline{2} \cdot \underline{5}) + \mathbf{1})^{M_3} &= ((\underline{2} \cdot \underline{5})^{M_3} +^{M_3} \mathbf{1}^{M_3}) \\ &= (\underline{2}^{M_3} \cdot^{M_3} \underline{5}^{M_3}) + 1 \\ &= (2 \cdot 5) + 1 = 11 \end{aligned}$$

$$\begin{aligned} ((\underline{2}^{M_4} \cdot \underline{5}^{M_4}) + \mathbf{1})^{M_4} &= ((x \cdot y)^{M_4} +^{M_4} \mathbf{1}^{M_4}) \\ &= (\underline{2}^{M_4} \cdot^{M_4} \underline{5}^{M_4}) - 2 \\ &= 2 - 2 = 0 \end{aligned}$$

Interpretación de fórmulas (I)

Dada una L–estructura $\mathcal{M}$, decimos que una fórmula F cerrada de $L(\mathcal{M})$ se **satisface** en $\mathcal{M}$, $\mathcal{M} \models F$, si:

- Si F es $p(t_1, \ldots, t_n)$ (atómica), entonces $\mathcal{M} \models F$ sii $(t_1^{\mathcal{M}}, \ldots, t_n^{\mathcal{M}}) \in p^{\mathcal{M}}$.
- Si F es $F_1 \vee F_2$, entonces $\mathcal{M} \models F$ sii se verifica que

$$\mathcal{M} \models F_1 \quad \text{ó} \quad \mathcal{M} \models F_2$$

- Las conectivas $\wedge$, $\rightarrow$ y $\leftrightarrow$ se tratan de manera similar.
- Si F es $\neg F_1$, entonces $\mathcal{M} \models F$ sii no se tiene $\mathcal{M} \models F_1$.
- Si F es $\exists x F_1(x)$, entonces $\mathcal{M} \models F$ sii

$$\text{existe } b \in M \text{ tal que } \mathcal{M} \models F_1(\underline{b})$$

- Si F es $\forall x F_1(x)$, entonces $\mathcal{M} \models F$ sii

$$\text{para todo } b \in M, \text{ se tiene } \mathcal{M} \models F_1(\underline{b})$$

Interpretación de fórmulas (II)

- En particular, la definición anterior nos permite precisar cuándo una fórmula cerrada de L, F, es válida en $\mathcal{M}$ (o bien que $\mathcal{M}$ es un modelo de F) y escribir $\mathcal{M} \models F$.
- Si F tiene variables libres, por definición,

$$\mathcal{M} \models F(x_1, \ldots x_n) \iff \mathcal{M} \models \forall x_1 \cdots \forall x_n F(x_1, \ldots x_n)$$

- Si Σ es un conjunto de fórmulas de un lenguaje L y $\mathcal{M}$ una estructura para L, decimos que $\mathcal{M}$ **es un modelo de** Σ, si

$$\text{para toda fórmula } F \in \Sigma, \quad \mathcal{M} \models F.$$

Ejemplos

En $\mathcal{M}_1$:

- Universo: $M_1 = \{Pedro,\ Pablo,\ Ana,\ Laura\}$.
- $padre\_de^{M_1} = \{(Pablo, Ana), (Pedro, Pablo)\}$.
- $madre\_de^{M_1} = \{(Ana, Laura)\}$.
- $hermano^{M_1} = \emptyset$, $casados^{M_1} = \emptyset$.

Se tiene:

- $\mathcal{M}_1 \models \exists x(padre\_de(\underline{Pablo}, x) \wedge madre\_de(x, \underline{Laura}))$
- $\mathcal{M}_1 \models \neg\exists x\ padre\_de(x, \underline{Laura})$
- $\mathcal{M}_1 \models \forall x\, \forall y\, \forall z\, (padre(x, z) \wedge madre(y, z) \rightarrow \neg casados(x, y))$.
- $\mathcal{M}_1 \models hermano(x, y) \leftrightarrow hermano(y, x)$
- $\mathcal{M}_1 \not\models \exists x\ padre\_de(x, y)$

Ejemplos (II)

En $\mathcal{M}_2$:

- Universo: $M_2 = \{0, 1, 2, 3, 4, 5, 6\}$.
- $padre\_de^{M_2} \equiv$ ser múltiplo de.
- $madre\_de^{M_2} \equiv$ ser menor.
- $hermano^{M_2} \equiv$ primos entre sí, $casados^{M_2} = \emptyset$.

Se tiene:

- $\mathcal{M}_2 \models \exists x(padre\_de(\underline{4}, x) \wedge madre\_de(x, \underline{3}))$
- $\mathcal{M}_2 \models \exists x\ padre\_de(x, \underline{3})$
- $\mathcal{M}_2 \models hermano(x, y) \leftrightarrow hermano(y, x)$
- $\mathcal{M}_2 \models \exists x \forall y\ padre\_de(x, y)$ $[x = 0]$
- ¿Se tiene $\mathcal{M}_2 \models hermano(x, y) \rightarrow \neg padre\_de(x, y)$?

Validez y Consistencia

- Una fórmula F de L sin variables libres (cerrada) es **satisfactible** si existe una L–estructura $\mathcal{M}$ tal que

$$\mathcal{M} \models F$$

- Un conjunto de fórmulas cerradas Σ de un lenguaje L es **consistente** si existe una L–estructura, $\mathcal{M}$, tal que

$$\text{para toda formula } F \in \Sigma, \quad \mathcal{M} \models F$$

- Una fórmula F es **lógicamente válida** si para toda estructura $\mathcal{M}$ se tiene que $\mathcal{M} \models F$ (Notación: $\models F$).
  - Ejemplo: $\forall x P(x) \vee \exists x \neg P(x)$

Consecuencia lógica y equivalencia

- Diremos que una fórmula F es **consecuencia lógica** de un conjunto de fórmulas cerradas Σ, $(\Sigma \models F)$, si para toda L–estructura $\mathcal{M}$ se tiene que

$$\text{si } \mathcal{M} \models \Sigma, \text{ entonces } \mathcal{M} \models F$$

  - Es decir, si todo modelo de Σ es modelo de F.

- Los problemas de la consistencia, consecuencia lógica y la validez para la lógica primer orden, **no son decidibles**.

Formas Normales y DPLL

Introducción

- Presentaremos ahora mecanismos para transformar las fórmulas de las lógicas estudiadas y conseguir expresiones *equivalentes* que muestran algunas ventajas para aplicar algoritmos.
- Comenzaremos dando procedimientos para transformar fórmulas proposicionales.
- Y daremos un algoritmo para decidir SAT que es la base de la mayoría de algoritmos modernos para ese problema.

Formas Normales en LP

- Comenzaremos dando una visión *algebraica* de las fórmulas proposicionales.
- Si identificamos cada conectiva con su función de verdad, entonces cada fórmula proposicional F, que contenga sólo las variables proposicionales $p_1, \ldots, p_n$, puede considerarse una expresión algebraica (similar a un polinomio), que define una función booleana $H_F : \{0,1\}^n \to \{0,1\}$.
- Estas expresiones algebraicas pueden manipularse de manera similar al modo en que reescribimos una expresión aritmética para simplificarla.
- Con estos procedimientos conseguiremos dar dos formas simples para cada fórmula: la *Forma Normal Conjuntiva* (FNC) y la *Forma Normal Disyuntiva* (FND).
- Pero antes debemos explicitar qué entendemos por **fórmulas equivalentes**.

Equivalencia lógica en LP

Definición.

Dos fórmulas F_1, F_2 son **equivalentes** ($F_1 \equiv F_2$) si, para toda valoración v, se tiene que $v(F_1) = v(F_2)$.

Es fácil comprobar que:

- $F_1 \equiv F_2$ si F_1 y F_2 tienen los mismos modelos.
- $F_1 \equiv F_2$ si y sólo si $F_1 \models F_2$ y $F_2 \models F_1$

Ejemplos:

- $F \notin SAT$, $G \notin SAT$, entonces $F \equiv G$.
- $F \in TAUT$, $G \in TAUT$, entonces $F \equiv G$.

Equivalencias (I)

Sean A, $B \in PROP$. Se tienen las siguientes equivalencias:

- **Conmutatividad**:

$$A \vee B \equiv B \vee A$$
$$A \wedge B \equiv B \wedge A$$

- **Asociatividad**:

$$A \vee (B \vee C) \equiv (A \vee B) \vee C$$
$$A \wedge (B \wedge C) \equiv (A \wedge B) \wedge C$$

- **Distributividad**:

$$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$$
$$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$$

- **Doble negación**:

$$\neg\neg A \equiv A$$

- **Leyes de De Morgan**:

$$\neg(A \wedge B) \equiv \neg A \vee \neg B$$
$$\neg(A \vee B) \equiv \neg A \wedge \neg B$$

Equivalencias (II)

- **Idempotencia**:

$$A \vee A \equiv A$$
$$A \wedge A \equiv A$$

- **Absorción**:

$$A \vee (A \wedge B) \equiv A$$
$$A \wedge (A \vee B) \equiv A$$

- **Leyes de tautología**: Si A es una tautología, entonces

$$A \wedge B \equiv B$$
$$A \vee B \equiv A$$

- **Leyes de inconsistentes**: Si A es insatisfactible, entonces

$$A \wedge B \equiv A$$
$$A \vee B \equiv B$$

Sustitución

- Dadas $A, A', B \in PROP$ si A es una subfórmula de B, la sustitución de A por A' en B es la fórmula que se obtiene al cambiar cada aparición de A en B por A'.
  - Notación: $B_{\{A/A'\}}$.
  - Si A no es una subfórmula de B, por definición $B_{\{A/A'\}}$ es B.
- $B = p \to (q \wedge r) \vee \neg s$, $A = q \wedge r$, $A' = t \to \neg r$

$$\overbrace{p \to \underbrace{(q \wedge r)}_{A} \vee \neg s}^{B} \qquad \overbrace{p \to \underbrace{(t \to \neg r)}_{A'} \vee \neg s}^{B_{\{A/A'\}}}$$

Sustitución: ejemplos

Si $B = p \to q \to r \vee s$ entonces:

- $B_{\{r \vee s/p\}} = \quad p \to q \to p.$
- $B_{\{q \wedge r/q\}} = \quad p \to q \to r \vee s.$
- $B_{\{q \to r \vee s/p \wedge r\}} = \quad p \to p \wedge r.$
- $B_{\{p \to q/p\}} = \quad p \to q \to r \vee s.$

Si $C = (p \to q) \vee (r \to p \to q)$, entonces:

- $C_{\{p \to q/t\}} = \quad t \vee (r \to t).$
- $C_{\{p \to q/t \to p \to q\}} = (t \to p \to q) \vee (r \to (t \to p \to q)).$

El Teorema de Sustitución

Teorema de Sustitución. Si $A, A', B \in PROP$ y $A \equiv A'$ entonces $B_{\{A/A'\}} \equiv B$.

- Este teorema permite manipular "algebraicamente" una fórmula para obtener otra fórmula más simple y equivalente a ella. Este proceso es similar al empleado en la simplificación de expresiones algebraicas.
- **Ejemplo:**

$$\begin{aligned} B \vee (A \wedge (A \rightarrow B)) &\equiv B \vee (A \wedge (\neg A \vee B)) \\ &\equiv B \vee (A \wedge \neg A) \vee (A \wedge B) \\ &\equiv B \vee (A \wedge B) \equiv B \end{aligned}$$

Literales proposicionales

- Una fórmula F es un **literal** si es una variable proposicional o la negación de una variable proposicional.
- Dos literales, L_1 y L_2, son **complementarios** si uno de ellos es la negación del otro. L^c denota el literal complementario de L.

Lema. Sea $\Sigma = \{L_1, \ldots, L_n\}$ un conjunto de literales. Entonces:

1. $\bigvee_{i=1}^{n} L_i \in TAUT \iff \Sigma$ contiene un par de literales complementarios.
2. $\bigwedge_{i=1}^{n} L_i \notin SAT \iff \Sigma$ contiene un par de literales complementarios.

Formas normales en LP

- Una fórmula está en **Forma Normal Conjuntiva (FNC)** si es una conjunción de disyunciones de literales:

$$F = \bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{m_i} L_{i,j} \right)$$

- Una fórmula está en **Forma Normal Disjuntiva (FND)** si es una disyunción de conjunciones de literales:

$$F = \bigvee_{i=1}^{n} \left(\bigwedge_{j=1}^{m_i} L_{i,j} \right)$$

Formas normales en LP (II)

Corolario.

- Una fórmula en FNC es una tautología si y solo si cada una de sus disyunciones es una tautología.
- Una fórmula en FND es insatisfactible si y solo si cada una de sus conjunciones es insatisfactible.

Teorema.
Para toda fórmula $G \in PROP$:

- existe F_1 en **FNC** tal que $F_1 \equiv G$.
- existe F_2 en **FND** tal que $F_2 \equiv G$.

Paso a forma normal

Procedimiento para transformar G a FNC:

1. Eliminar todas las implicaciones usando:

$$A \to B \equiv \neg A \vee B \quad \text{y} \quad A \leftrightarrow B \equiv (A \to B) \wedge (B \to A)$$

2. Trasladar las negaciones, mediante las leyes de Morgan:

$$\neg(A \wedge B) \equiv \neg A \vee \neg B \qquad \text{y} \qquad \neg(A \vee B) \equiv \neg A \wedge \neg B$$

3. Eliminar negaciones dobles usando

$$\neg\neg A \equiv A$$

4. Convertir a FNC utilizando la ley distributiva:

$$A \vee (B_1 \wedge B_2) \equiv (A \vee B_1) \wedge (A \vee B_2)$$

Para pasar a FND, en 4 utilizamos la ley distributiva:

$$A \wedge (B_1 \vee B_2) \equiv (A \wedge B_1) \vee (A \wedge B_2)$$

Ejemplo:

Obtener una FNC de $(\neg p \wedge q) \rightarrow (q \vee r) \rightarrow p$:

$$
\begin{aligned}
&(\neg p \wedge q) \rightarrow (q \vee r) \rightarrow p \qquad \Rightarrow \\
&\Rightarrow \quad \neg(\neg p \wedge q) \vee ((q \vee r) \rightarrow p) \\
&\Rightarrow \quad \neg(\neg p \wedge q) \vee \neg(q \vee r) \vee p \\
&\Rightarrow \quad \neg\neg p \vee \neg q \vee (\neg q \wedge \neg r) \vee p \\
&\Rightarrow \quad p \vee \neg q \vee ((\neg q \vee p) \wedge (\neg r \vee p)) \\
&\Rightarrow \quad (p \vee \neg q \vee \neg q \vee p) \wedge (p \vee \neg q \vee \neg r \vee p) \\
&\Rightarrow \quad (\neg q \vee p) \wedge (\neg q \vee \neg r \vee p)
\end{aligned}
$$

Hemos eliminado literales repetidos en una misma cláusula gracias a la equivalencia: $A \vee A \equiv A$.
(En la FND utilizaríamos la equivalencia $A \wedge A \equiv A$).

Algoritmo de satisfactibilidad mediante FND

Procedimiento FND

Entrada: $F \in PROP$

Salida: `SI`, si $F \in SAT$; `NO`, en caso contrario.

Calcular una FND de F: $G = G_1 \vee \cdots \vee G_n$

para $i = 1$ **hasta** n

si en G_i no ocurren literales complementarios

entonces devolver `SI`; parar

devolver `NO`

Algoritmo de validez mediante FNC

Procedimiento FNC

Entrada: $F \in PROP$

Salida: `SI`, si $F \in TAUT$; `NO`, en caso contrario.

Calcular una FNC de F: $G = G_1 \wedge \cdots \wedge G_n \leftarrow FNC(F)$

para $i = 1$ **hasta** n

si en G_i no ocurren literales complementarios

entonces devolver `NO`; parar

devolver `SI`

Ejemplos

- $F_1 = (p \wedge q) \rightarrow (q \wedge r) \vee p$.
  Una FNC de F_1 es:

$$(\neg p \vee \neg q \vee q \vee p) \wedge (\neg p \vee \neg q \vee r \vee p)$$

  Es una tautología (y, por tanto, satisfactible).
- $F_2 = \neg(p \vee q) \vee (p \rightarrow q)$.
  Una FND de F_2 es:

$$(\neg p \wedge \neg q) \vee \neg p \vee q$$

  Por tanto, F_2 es satisfactible.

  Una FNC de F_2 es:

$$(\neg p \vee q) \wedge (\neg q \vee \neg p \vee q)$$

  Por tanto, F_2 no es tautología.

Cláusulas en LP

- Una **cláusula** es una disyunción de literales $L_1 \vee \cdots \vee L_n$.
- Dada una valoración v y una cláusula $L_1 \vee \cdots \vee L_n$:

$$v \models L_1 \vee \cdots \vee L_n \iff \text{Existe } i = 1, \ldots, n \text{ tal que } v \models L_i$$

  Por tanto, el valor de verdad de $L_1 \vee \cdots \vee L_n$ no depende ni del orden en que aparecen los literales ni de posibles repeticiones de literales.
- En consecuencia, identificamos la cláusula $L_1 \vee \cdots \vee L_n$ con el conjunto de literales $\{L_1, \ldots L_n\}$.
- Caso especial: la **cláusula vacía**, correspondiente al conjunto de literales vacío. La denotamos por $\square$.
- Por definición, para toda valoración, v, se tiene que $v(\square) = 0$, es decir, $\square$ es una contradicción.

Forma clausal en LP

Teorema.
Para toda fórmula $F \in PROP$ existe un conjunto de cláusulas $\{C_1, \ldots, C_n\}$ tal que para toda valoración, v,

$$v \models F \iff v \models \{C_1, \ldots, C_n\}$$

$\{C_1, \ldots, C_n\}$ se denomina una **forma clausal** de F.

Demotración: Podemos obtener una forma clausal a partir de una FNC, ya que si

$$(L_{1,1} \vee \cdots \vee L_{1,n_1}) \wedge \cdots \wedge (L_{m,1} \vee \cdots \vee L_{m,n_m})$$

es la FNC, basta escribirla en forma clausal como:

$$\{\{L_{1,1}, \ldots, L_{1,n_1}\}, \ldots, \{L_{m,1}, \cdots, L_{m,n_m}\}\}$$

Forma clausal en LP (II)

Corolario.

En el caso de un conjunto de fórmulas U existe un conjunto de cláusulas $\{C_1, \ldots, C_n\}$ tal que para toda valoración, v,

$$v \models U \iff v \models \{C_1, \ldots, C_n\}$$

$\{C_1, \ldots, C_n\}$ se denomina una **forma clausal** de U.

Nota: Podemos obtener una forma clausal de un conjunto U uniendo formas clausales de las fórmulas de U. Por ejemplo:

$$U = \{\underbrace{p \to q}_{(1)}, \underbrace{(p \vee \neg q) \wedge (r \to \neg p)}_{(2)}, \underbrace{p \wedge \neg r}_{(3)}\}$$

Una forma clausal de U es:

$$\{\underbrace{\neg p \vee q}_{(1)}, \underbrace{p \vee \neg q, \neg r \vee \neg p}_{(2)}, \underbrace{p, \neg r}_{(3)}\}$$

El algoritmo DPLL

- Determinar la **satisfactibilidad** de un conjunto de cláusulas, por lo que requiere una fase de preprocesamiento.
- Es un refinamiento (Davis, Logemann y Loveland, 1962) de un algoritmo propuesto por Davis y Putnam (1960).
- Es la base de muchos "SAT solvers": programas para determinar la satisfactibilidad de un conjunto de fórmulas proposicionales (habitualmente, cláusulas).
- Puede utilizarse como algoritmo de decisión, y también como generador de modelos.
- Hace una búsqueda sistemática por medio de los posibles valores de las variables proposicionales.
- Al igual que los Tableros Semánticos, suele representarse gráficamente mediante un árbol binario.

Estructura del algoritmo

- Podemos distinguir dos partes en el algoritmo:
  1. **Propagación de unidades**: simplifica el conjunto de cláusulas.
  2. **División**: organiza la búsqueda de una valoración que muestre que el conjunto es satisfactible.
- El algoritmo puede describirse de manera recursiva:

Procedimiento *DPLL*
Entrada: S conjunto de cláusulas
Salida: `SI`, si $S \in SAT$; `NO`, en caso contrario
Si $S = \emptyset$ **devolver** `SI`
Si $\square \in S$ **devolver** `NO`
Si S tiene unidades:
devolver $DPLL(Propaga\_unidades(S))$
Si no:
$\{S_1,\ S_2\} = Division(S)$
devolver $DPLL(S_1) \vee DPLL(S_2)$

Propagación de unidades

- Se elige una cláusula unitaria $L \in S$ y se aplican consecutivamente las dos reglas siguientes:
  1. **Subsunción unitaria**. Se eliminan de S todas las cláusulas subsumidas por L, es decir, que contengan el literal L (inluida la propia cláusula L).
  2. **Resolución unidad**. Se elimina el literal complementario L^c de todas las cláusulas de S.
- El proceso se repite hasta que no queden unidades en S.

Procedimiento *Propaga_unidades*
Entrada: S conjunto de cláusulas
Salida: Conjunto de cláusulas
  Mientras S tenga unidades:
    Selecciona L unidad de S
    $S = \{C \ : L \notin C, \ C \in S\}$
    $S = \{C - \{L^c\} \ : L^c \in C, \ C \in S\}$
  devolver S'

División

- Tras el proceso de propagación, si el algoritmo no ha parado, entonces S no contiene cláusulas unitarias.
- Se elige un literal L que aparezca en una cláusula de S y se construyen los conjuntos: $S \cup \{L\}$ y $S \cup \{L^c\}$.

Procedimiento División

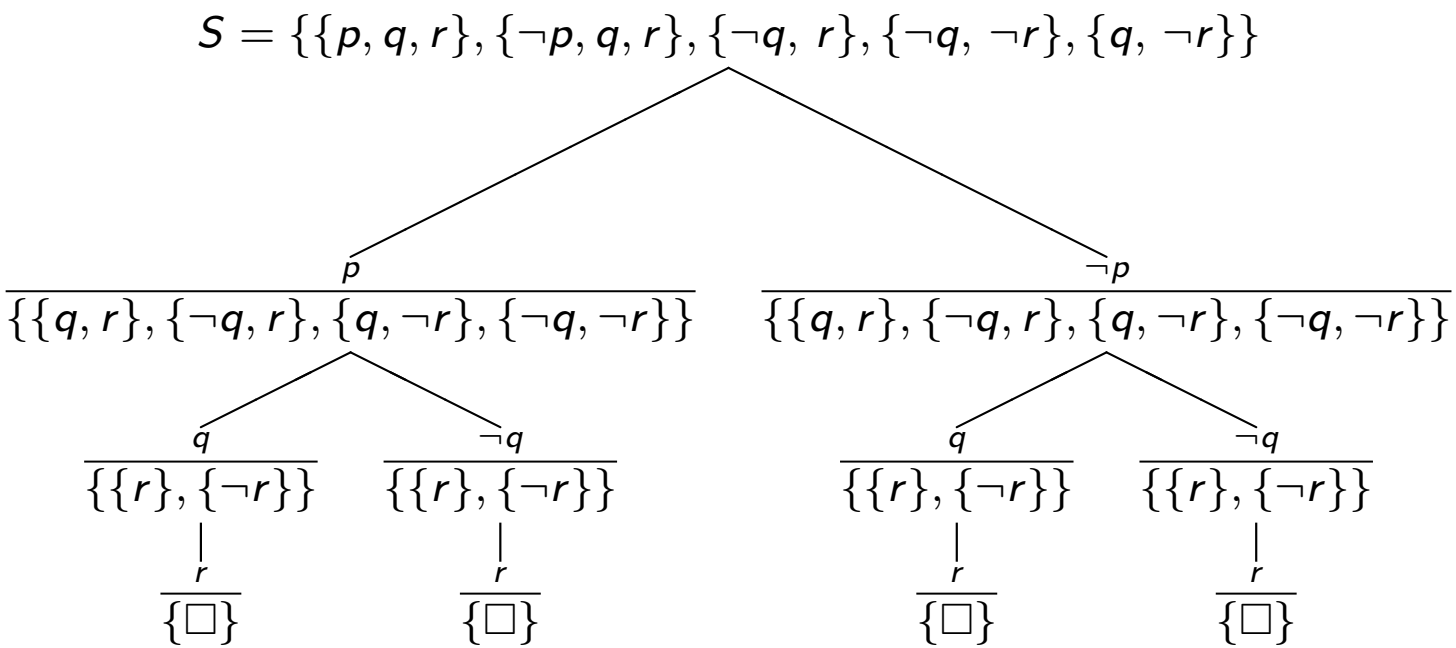
Entrada: S conjunto de cláusulas

Salida: Dos conjuntos de cláusulas

Selecciona $C \in S$, $L \in C$

devolver $\{S \cup \{\{L\}\},\ S \cup \{\{L^c\}\}\}$

Ejemplo



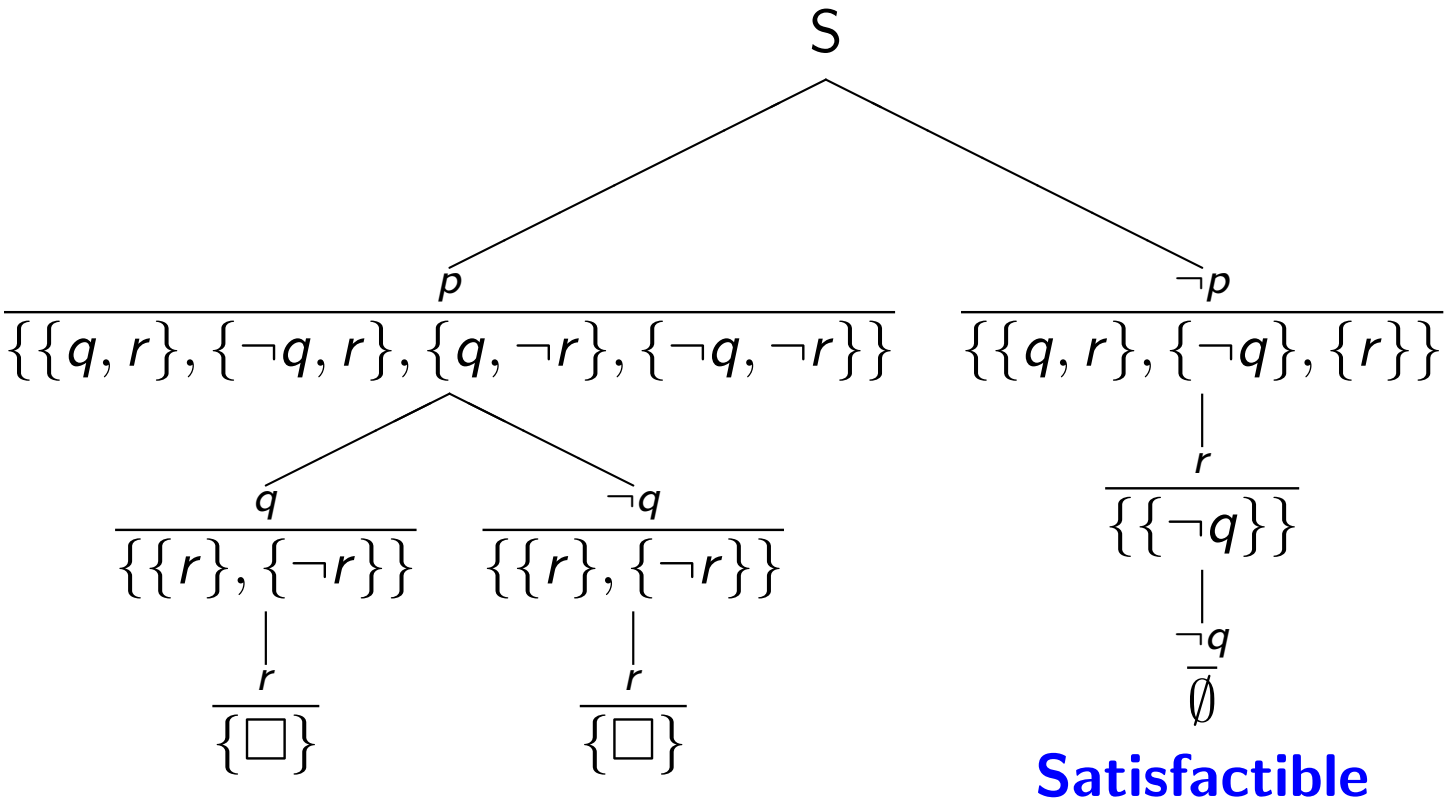
Por tanto, S es **insatisfactible**.

Extracción de modelos

- Si el algoritmo DPLL responde afirmativamente (S es satisfactible, y en consecuencia tiene modelos), entonces cada rama que llega a $\emptyset$ proporciona modelos.
- Para obtener el modelo de una rama, basta anotar la elección de unidades que se ha hecho a lo largo de la misma (ya sea por propagación, o por división):
  - Si una unidad aparece positiva, la valoración para la variable asociada será 1.
  - Si una unidad aparece negativa, la valoración para la variable asociada será 0.
  - Las variables que no intervienen pueden tener cualquier valor de verdad asignado.
- Además, por la forma en que trabaja DPLL, ramas distintas proporcionan siempre modelos distintos (algo que no pasaba con tableros semánticos).

Ejemplo

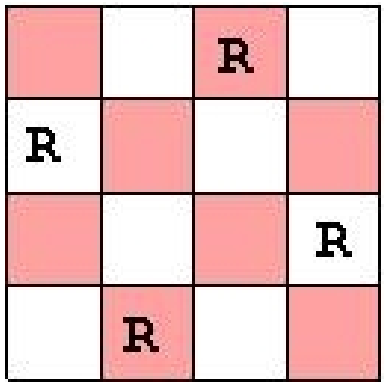
$$S = \{\{p, q, r\}, \{\neg p, q, r\}, \{p, \neg q\}, \{p, r\}, \{\neg p, \neg q, r\}, \{\neg p, q, \neg r\}, \{\neg p, \neg q, \neg r\}\}$$

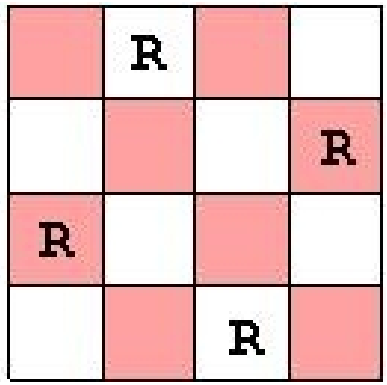


Un modelo de S sería: $v(p) = 0$, $v(r) = 1$, $v(q) = 0$

Problema de las 4 reinas (Proposicional)

Enunciado: Calcular las formas de colocar 4 reinas en un tablero de 4x4 de forma que no se ataquen entre sí (no haya más de una reina en cada fila, columna o diagonal).





Formalización: r_{ij} $(1 \leq i, j \leq 4)$ indica que hay una reina en la fila i columna j

Problema de las 4 reinas (Proposicional)

En cada fila hay una reina:

$$r_{11} \vee r_{12} \vee r_{13} \vee r_{14}, \;\; r_{21} \vee r_{22} \vee r_{23} \vee r_{24}$$

$$r_{31} \vee r_{32} \vee r_{33} \vee r_{34}, \;\; r_{41} \vee r_{42} \vee r_{43} \vee r_{44}$$

Si en una casilla hay reina, entonces no hay más reinas en su fila, su columna y sus diagonales:

$$r_{11} \rightarrow (\neg r_{12} \wedge \neg r_{13} \wedge \neg r_{14}) \wedge (\neg r_{21} \wedge \neg r_{31} \wedge \neg r_{41}) \wedge (\neg r_{22} \wedge \neg r_{33} \wedge \neg r_{44})$$

$$r_{12} \rightarrow (\neg r_{11} \wedge \neg r_{13} \wedge \neg r_{14}) \wedge (\neg r_{22} \wedge \neg r_{32} \wedge \neg r_{42}) \wedge (\neg r_{21} \wedge \neg r_{23} \wedge \neg r_{34})$$

$$r_{13} \rightarrow (\neg r_{11} \wedge \neg r_{12} \wedge \neg r_{14}) \wedge (\neg r_{23} \wedge \neg r_{33} \wedge \neg r_{43}) \wedge (\neg r_{31} \wedge \neg r_{22} \wedge \neg r_{24})$$

$$r_{14} \rightarrow (\neg r_{11} \wedge \neg r_{12} \wedge \neg r_{13}) \wedge (\neg r_{24} \wedge \neg r_{34} \wedge \neg r_{44}) \wedge (\neg r_{23} \wedge \neg r_{32} \wedge \neg r_{41})$$

$$r_{21} \rightarrow (\neg r_{22} \wedge \neg r_{23} \wedge \neg r_{24}) \wedge (\neg r_{11} \wedge \neg r_{31} \wedge \neg r_{41}) \wedge (\neg r_{32} \wedge \neg r_{43} \wedge \neg r_{12})$$

$$r_{22} \rightarrow (\neg r_{21} \wedge \neg r_{23} \wedge \neg r_{24}) \wedge (\neg r_{12} \wedge \neg r_{32} \wedge \neg r_{42}) \wedge (\neg r_{11} \wedge \neg r_{33} \wedge \neg r_{44} \wedge \neg r_{13} \wedge \neg r_{31})$$

$$r_{23} \rightarrow (\neg r_{21} \wedge \neg r_{22} \wedge \neg r_{24}) \wedge (\neg r_{13} \wedge \neg r_{33} \wedge \neg r_{43}) \wedge (\neg r_{12} \wedge \neg r_{34} \wedge \neg r_{14} \wedge \neg r_{32} \wedge \neg r_{41})$$

$$r_{24} \rightarrow (\neg r_{21} \wedge \neg r_{22} \wedge \neg r_{23}) \wedge (\neg r_{14} \wedge \neg r_{34} \wedge \neg r_{44}) \wedge (\neg r_{13} \wedge \neg r_{33} \wedge \neg r_{42})$$

Problema de las 4 reinas (Proposicional)

Si en una casilla hay reina, entonces no hay más reinas en su fila, su columna y sus diagonales:

$$r_{31} \rightarrow (\neg r_{32} \wedge \neg r_{33} \wedge \neg r_{34}) \wedge (\neg r_{11} \wedge \neg r_{21} \wedge \neg r_{41}) \wedge (\neg r_{42} \wedge \neg r_{13} \wedge \neg r_{22})$$

$$r_{32} \rightarrow (\neg r_{31} \wedge \neg r_{33} \wedge \neg r_{34}) \wedge (\neg r_{12} \wedge \neg r_{22} \wedge \neg r_{42}) \wedge (\neg r_{21} \wedge \neg r_{43} \wedge \neg r_{14} \wedge \neg r_{23} \wedge \neg r_{41})$$

$$r_{33} \rightarrow (\neg r_{31} \wedge \neg r_{32} \wedge \neg r_{34}) \wedge (\neg r_{13} \wedge \neg r_{23} \wedge \neg r_{43}) \wedge (\neg r_{11} \wedge \neg r_{22} \wedge \neg r_{44} \wedge \neg r_{24} \wedge \neg r_{42})$$

$$r_{34} \rightarrow (\neg r_{31} \wedge \neg r_{32} \wedge \neg r_{33}) \wedge (\neg r_{14} \wedge \neg r_{24} \wedge \neg r_{44}) \wedge (\neg r_{12} \wedge \neg r_{23} \wedge \neg r_{43})$$

$$r_{41} \rightarrow (\neg r_{42} \wedge \neg r_{43} \wedge \neg r_{44}) \wedge (\neg r_{11} \wedge \neg r_{21} \wedge \neg r_{31}) \wedge (\neg r_{14} \wedge \neg r_{23} \wedge \neg r_{32})$$

$$r_{42} \rightarrow (\neg r_{41} \wedge \neg r_{43} \wedge \neg r_{44}) \wedge (\neg r_{12} \wedge \neg r_{22} \wedge \neg r_{32}) \wedge (\neg r_{31} \wedge \neg r_{24} \wedge \neg r_{33})$$

$$r_{43} \rightarrow (\neg r_{41} \wedge \neg r_{42} \wedge \neg r_{44}) \wedge (\neg r_{13} \wedge \neg r_{23} \wedge \neg r_{33}) \wedge (\neg r_{21} \wedge \neg r_{32} \wedge \neg r_{34})$$

$$r_{44} \rightarrow (\neg r_{41} \wedge \neg r_{42} \wedge \neg r_{43}) \wedge (\neg r_{14} \wedge \neg r_{24} \wedge \neg r_{34}) \wedge (\neg r_{11} \wedge \neg r_{22} \wedge \neg r_{33})$$

Problema de las 4 reinas (Proposicional)

Búsqueda de modelos con Mace4:

$$r_{11} : 0,\ r_{12} : 0,\ r_{13} : 1,\ r_{14} : 0$$

$$r_{21} : 1,\ r_{22} : 0,\ r_{23} : 0,\ r_{24} : 0$$

$$r_{31} : 0,\ r_{32} : 0,\ r_{33} : 0,\ r_{34} : 1$$

$$r_{41} : 0,\ r_{42} : 1,\ r_{43} : 0,\ r_{44} : 0$$

$$r_{11} : 0,\ r_{12} : 1,\ r_{13} : 0,\ r_{14} : 0$$

$$r_{21} : 0,\ r_{22} : 0,\ r_{23} : 0,\ r_{24} : 1$$

$$r_{31} : 1,\ r_{32} : 0,\ r_{33} : 0,\ r_{34} : 0$$

$$r_{41} : 0,\ r_{42} : 0,\ r_{43} : 1,\ r_{44} : 0$$

Problema de las N Reinas (Primer Orden)

En este caso, en un LPO definimos un predicado $R(x, y)$ que indica si hay una reina en la posición (x, y). Restricciones:

- En cada fila al menos hay una reina: $\forall x \exists y R(x, y)$
- En cada fila a lo sumo hay una reina: $R(x, y_1) \wedge R(x, y_2) \rightarrow y_1 = y_2$
- En cada columna a lo sumo hay una reina:

$$R(x_1, y) \wedge R(x_2, y) \rightarrow x_1 = x_2$$

- En cada diagonal principal a lo sumo hay una reina:

$$R(x_1, y_1) \wedge R(x_2, y_2) \wedge (x_2 + -x_1 = y_2 + -y_1) \rightarrow x_1 = x_2 \wedge y_1 = y_2$$

- En cada diagonal secundaria a lo sumo hay una reina:

$$R(x_1, y_1) \wedge R(x_2, y_2) \wedge (x_1 + -x_2 = y_2 + -y_1) \rightarrow x_1 = x_2 \wedge y_1 = y_2$$

Hemos de considerar las propiedades habituales en las operaciones aritméticas.

Sudoku

Problema: Resolver sudokus.

- Un sudoku estás dado por un tablero de 9×9 casillas, dividido a su vez en 3×3 bloques de tamaño 3×3.

	1	8				7		
			3			2		
	7							
				7	1			
6							4	
3								
4			5					3
	2			8				
							6	

- Cada casilla puede contener un número del 1 al 9.
- Algunas casillas se rellenan inicialmente y hay que rellenar el resto de modo que cada fila, columna y bloque contenga todos los dígitos.

Sudoku (Primer Orden)

Una solución será una función $f : [1,9] \times [1,9] \rightarrow [1,9]$ de forma que $f(x,y)$ indica el valor que hay que escribir en la casilla que ocupa la posición (x,y).
Restricciones del problema:

- El dominio de trabajo es $[1,9]$:
- En cada fila todos los elementos son distintos:

$$\forall x \forall y_1 \forall y_2 (f(x,y_1) = f(x,y_2) \rightarrow y_1 = y_2)$$

- En cada columna todos los elementos son distintos:

$$\forall y \forall x_1 \forall x_2 (f(x_1,y) = f(x_2,y) \rightarrow x_1 = x_2)$$

- En cada fila hay un elemento de cada tipo: $\forall x \forall z \exists y (f(x,y) = z)$
- En cada columna hay un elemento de cada tipo: $\forall y \forall z \exists x (f(x,y) = z)$

Sudoku (Primer Orden)

Una solución será una función $f : [1,9] \times [1,9] \rightarrow [1,9]$ de forma que $f(x,y)$ indica el valor que hay que escribir en la casilla que ocupa la posición (x,y).
Restricciones del problema:

- En cada bloque todos los elementos son distintos:

$$\forall x_1 \forall y_1 \forall x_2 \forall y_2 (mi(x_1,x_2) \wedge mi(y_1,y_2) \wedge f(x_1,y_1) = f(x_2,y_2)$$

$$\rightarrow x_1 = x_2 \wedge y_1 = y_2)$$

donde *mi* es una **relación de equivalencia**, y además verifica:

$$mi(1,2) \wedge mi(2,3) \wedge mi(4,5) \wedge mi(5,6) \wedge mi(7,8) \wedge mi(8,9) \wedge$$

$$\wedge \neg mi(1,4) \wedge \neg mi(4,7) \wedge \neg mi(1,7)$$

Sudoku (Primer Orden)

Una solución será una función $f : [1,9] \times [1,9] \rightarrow [1,9]$ de forma que $f(x,y)$ indica el valor que hay que escribir en la casilla que ocupa la posición (x,y).
Hay que añadir información inicial: $f(1,1) = 1,\ f(2,2) = 2,\ \ldots$

1			2			3		
	2			3			4	
		3			4			5
6			4			5		
	7			5			6	
		8			6			7
8			0			7		
	0			1			8	
		1			2			4

Sudoku (Primer Orden)

Una solución será una función $f : [1,9] \times [1,9] \rightarrow [1,9]$ de forma que $f(x,y)$ indica el valor que hay que escribir en la casilla que ocupa la posición (x,y).

1	4	5	2	8	0	3	7	6
7	2	6	5	3	1	8	4	0
0	8	3	7	6	4	1	2	5
6	1	0	4	2	7	5	3	8
3	7	4	1	5	8	0	6	2
2	5	8	3	0	6	4	1	7
8	6	2	0	4	3	7	5	1
4	0	7	6	1	5	2	8	3
5	3	1	8	7	2	6	0	4