

Tema 3: Formas Normales y Algoritmo DPLL

Dpto. Ciencias de la Computación Inteligencia Artificial
UNIVERSIDAD DE SEVILLA

Lógica Informática
(Tecnologías Informáticas)

Contenido

Introducción

Formas normales

Equivalencia lógica

Sustitución

Formas normales

Algoritmos para SAT y TAUT

Tablero y Formas Normales

Forma clausal

Algoritmo DPLL

Estructura del algoritmo

Extracción de Modelos

Introducción

Formas normales

Equivalencia lógica

Sustitución

Formas normales

Algoritmos para SAT y
TAUT

Tablero y Formas Normales

Forma clausal

Algoritmo DPLL

Estructura del algoritmo

Extracción de Modelos

- ▶ En este tema presentaremos 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.

Introducción

Formas normales

Equivalencia lógica

Sustitución

Formas normales

Algoritmos para SAT y
TAUT

Tablero y Formas Normales

Forma clausal

Algoritmo DPLL

Estructura del algoritmo

Extracción de Modelos

- ▶ 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, \dots, p_n , puede considerarse una expresión algebraica (similar a un polinomio), que define una función booleana $H_F : \{0, 1\}^n \rightarrow \{0, 1\}$.
- ▶ 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 $F \equiv G$.

[Introducción](#)[Formas normales](#)[Equivalencia lógica](#)[Sustitución](#)[Formas normales](#)[Algoritmos para SAT y
TAUT](#)[Tablero y Formas Normales](#)[Forma clausal](#)[Algoritmo DPLL](#)[Estructura del algoritmo](#)[Extracción de Modelos](#)

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 insatisficible, 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 .
- ▶ $B = p \rightarrow (q \wedge r) \vee \neg s$, $A = q \wedge r$, $A' = t \rightarrow \neg r$

$$\overbrace{p \rightarrow \underbrace{(q \wedge r)}_A \vee \neg s}^B$$

$$\overbrace{p \rightarrow \underbrace{(t \rightarrow \neg r)}_{A'} \vee \neg s}^{B_{\{A/A'\}}}$$

Sustitución: ejemplos

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

- ▶ $B_{\{r \vee s / p\}} = p \rightarrow q \rightarrow p.$
- ▶ $B_{\{q \wedge r / q\}} = p \rightarrow q \rightarrow r \vee s.$
- ▶ $B_{\{q \rightarrow r \vee s / p \wedge r\}} = p \rightarrow p \wedge r.$
- ▶ $B_{\{p \rightarrow q / p\}} = p \rightarrow q \rightarrow r \vee s.$

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

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

Introducción

Formas normales

Equivalencia lógica

Sustitución

Formas normales

Algoritmos para SAT y
TAUT

Tablero y Formas Normales

Forma clausal

Algoritmo DPLL

Estructura del algoritmo

Extracción de Modelos

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 $\Sigma = \{L_1, \dots, L_n\}$ 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.

Introducción

Formas normales

Equivalencia lógica

Sustitución

Formas normales

Algoritmos para SAT y

TAUT

Tablero y Formas Normales

Forma clausal

Algoritmo DPLL

Estructura del algoritmo

Extracción de Modelos

- ▶ 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:

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

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

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

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:

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

[Introducción](#)[Formas normales](#)[Equivalencia lógica](#)[Sustitución](#)[Formas normales](#)[Algoritmos para SAT y](#)[TAUT](#)[Tablero y Formas Normales](#)[Forma clausal](#)[Algoritmo DPLL](#)[Estructura del algoritmo](#)[Extracción de Modelos](#)

Ejemplo:

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

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

Hemos eliminado literales repetidos en una misma cláusula gracias a la equivalencia: $A \vee A \equiv A$.

(En la FND utilizaríamos la equivalencia $A \wedge A \equiv A$).

Introducción

Formas normales

Equivalencia lógica

Sustitución

Formas normales

Algoritmos para SAT y
TAUT

Tablero y Formas Normales

Forma clausal

Algoritmo DPLL

Estructura del algoritmo

Extracción de Modelos

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 : $G = G_1 \vee \dots \vee G_n$

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

si en G_i no ocurren literales complementarios

entonces devolver SI; **parar**

devolver NO

Procedimiento FNC

Entrada: $F \in PROP$

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

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

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

si en G_i no ocurren literales complementarios

entonces devolver NO; **parar**

devolver SI

Introducción

Formas normales

Equivalencia lógica

Sustitución

Formas normales

Algoritmos para SAT y
TAUT

Tablero y Formas Normales

Forma clausal

Algoritmo DPLL

Estructura del algoritmo

Extracción de Modelos

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.

Introducción

Formas normales

Equivalencia lógica

Sustitución

Formas normales

Algoritmos para SAT y
TAUT

Tablero y Formas Normales

Forma clausal

Algoritmo DPLL

Estructura del algoritmo

Extracción de Modelos

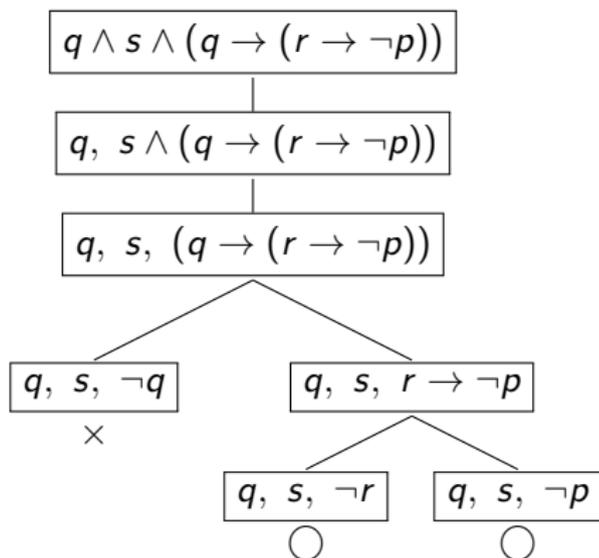
Tableros completos y FND

- ▶ Un tablero completo para una fórmula F puede utilizarse para obtener una FND de F .
- ▶ Si T es un tablero completo para F , procedemos como sigue:
 1. Si U_1, \dots, U_k son los conjuntos de literales que etiquetan las hojas abiertas de T , formamos para cada U_j una conjunción, C_j , con todos los literales de U_j .
 2. Una FND de F se obtiene formando la disyunción

$$C_1 \vee \dots \vee C_k$$

Tableros y FND: Ejemplo

Si F es la fórmula $q \wedge s \wedge (q \rightarrow (r \rightarrow \neg p))$



Una FND de F es $(q \wedge s \wedge \neg r) \vee (q \wedge s \wedge \neg p)$.

- ▶ Para obtener una FNC de una fórmula F :
 - ▶ Si G es una FND de $\neg F$, aplicando a $\neg G$ las leyes de De Morgan y eliminación de negaciones dobles transformamos $\neg G$ en una FNC de F .
- ▶ Por tanto, para obtener una FNC de F seguimos el siguiente procedimiento:
 1. Calculamos un tablero completo para $\neg F$.
 2. Si U_1, \dots, U_k son los conjuntos de literales que etiquetan las hojas abiertas del tablero completo para $\neg F$, formamos para cada U_j una disyunción D_j con los literales complementarios de los literales de U_j .
 3. Una FNC de F es la conjunción

$$D_1 \wedge \dots \wedge D_k$$

Tableros y FNC: Ejemplos

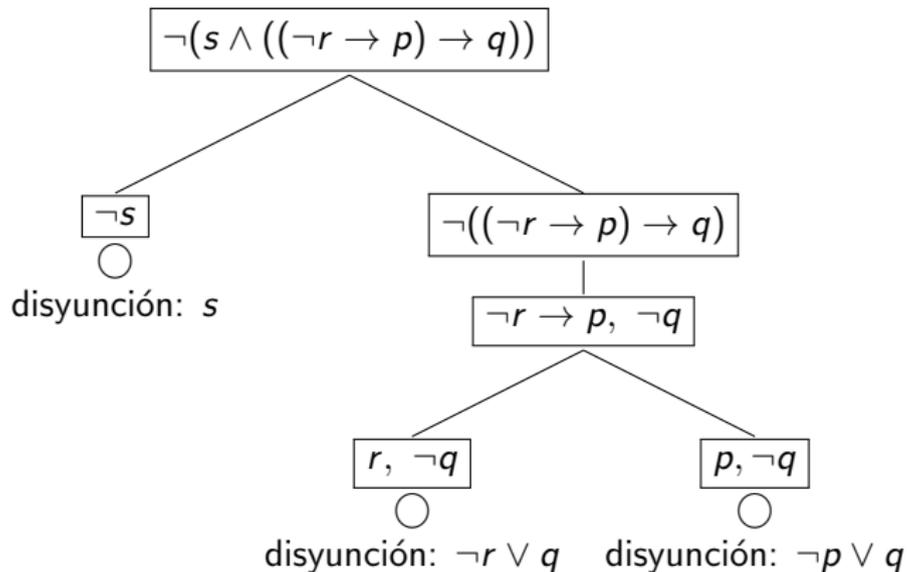
Sea F la fórmula $s \wedge ((\neg r \rightarrow p) \rightarrow q)$.

1. Una FND de $\neg F$ es $G \equiv \neg s \vee (r \wedge \neg q) \vee (p \wedge \neg q)$
2. Ahora $\neg G$ proporciona una FNC de F :

$$\begin{aligned} F \equiv \neg G &\equiv \neg(\neg s \vee (r \wedge \neg q) \vee (p \wedge \neg q)) \\ &\equiv \neg\neg s \wedge \neg(r \wedge \neg q) \wedge \neg(p \wedge \neg q) \\ &\equiv s \wedge (\neg r \vee \neg\neg q) \wedge (\neg p \vee \neg\neg q) \\ &\equiv s \wedge (\neg r \vee q) \wedge (\neg p \vee q) \end{aligned}$$

Tableros y FNC: Ejemplo (II)

Si F es la fórmula $s \wedge ((\neg r \rightarrow p) \rightarrow q)$. Calculamos un tablero completo para $\neg F$:



Una FNC de F es $s \wedge (\neg r \vee q) \wedge (\neg p \vee q)$.

Cláusulas en LP

- ▶ Una **cláusula** es una disyunción de literales

$$L_1 \vee \cdots \vee L_n.$$

- ▶ Dada una valoración v y una cláusula $L_1 \vee \cdots \vee L_n$:

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

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 $\{L_1, \dots, L_n\}$.
- ▶ 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 $\{C_1, \dots, C_n\}$ tal que para toda valoración, v ,

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

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

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

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

es la FNC, basta escribirla en forma clausal como:

$$\{\{L_{1,1}, \dots, L_{1,n_1}\}, \dots, \{L_{m,1}, \dots, L_{m,n_m}\}\}$$

[Introducción](#)[Formas normales](#)[Equivalencia lógica](#)[Sustitución](#)[Formas normales](#)[Algoritmos para SAT y TAUT](#)[Tablero y Formas Normales](#)[Forma clausal](#)[Algoritmo DPLL](#)[Estructura del algoritmo](#)[Extracción de Modelos](#)

Forma clausal en LP (II)

Corolario.

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

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

$\{C_1, \dots, C_n\}$ 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:

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

Una forma clausal de U es:

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

Introducción

Formas normales

Equivalencia lógica

Sustitución

Formas normales

Algoritmos para SAT y
TAUT

Tablero y Formas Normales

Forma clausal

Algoritmo DPLL

Estructura del algoritmo

Extracción de Modelos

- ▶ 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.

Tableros vs DPLL

Tienen características propias que distinguen claramente ambos métodos:

▶ **Algoritmo DPLL:**

- ▶ No trabaja con fórmulas arbitrarias sino sobre conjuntos de **cláusulas**. Es preciso “preprocesar” el conjunto de fórmulas, pasándolo a forma clausal.
- ▶ Está entre los algoritmos más eficientes para la lógica proposicional, pero no se extiende fácilmente a otras lógicas.

▶ **Tableros semánticos:**

- ▶ Trabaja directamente sobre el conjunto de fórmulas proposicionales.
- ▶ No es tan eficiente como DPLL, pero es muy flexible y puede adaptarse a otras lógicas (lógica de primer orden, descriptivas, modales, etc.), donde resulta útil en el estudio teórico.

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 (incluida 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'

- ▶ 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 *Division*

Entrada: S conjunto de cláusulas

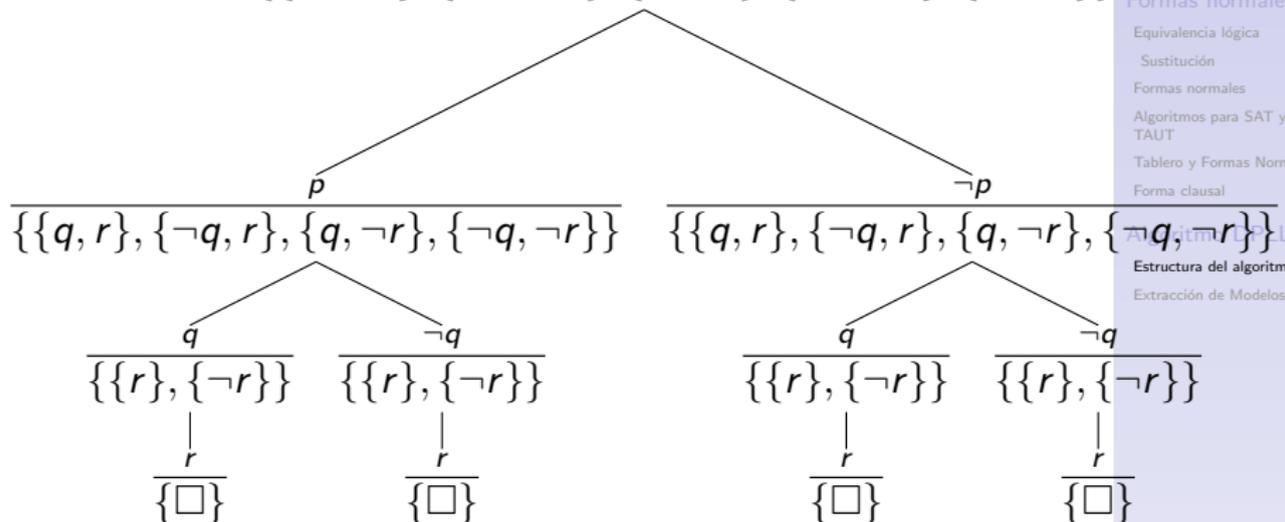
Salida: Dos conjuntos de cláusulas

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

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

Ejemplo

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

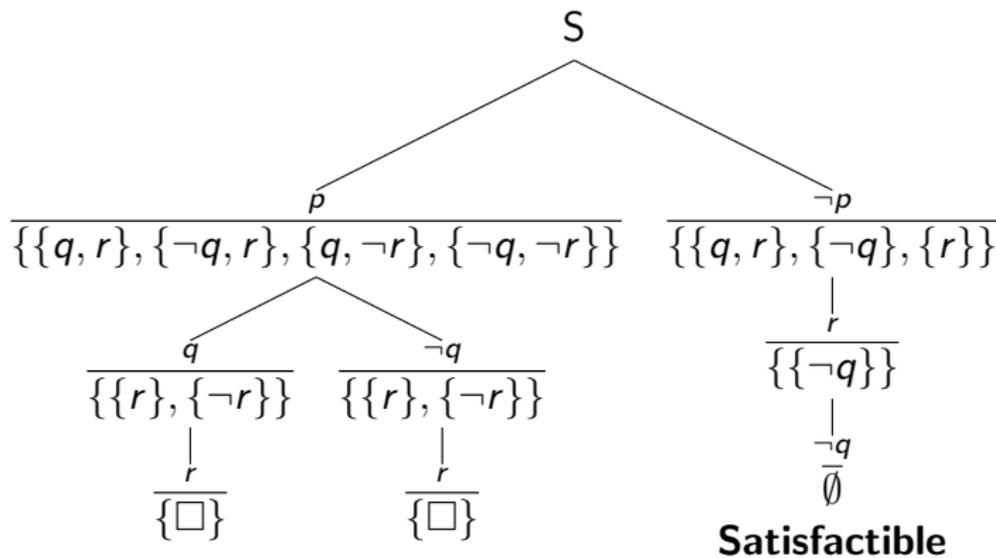


Por tanto, S es **insatisfactible**.

- ▶ 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 viene dado por:

$$v(p) = 0 \quad v(r) = 1 \quad v(q) = 0$$

Introducción

Formas normales

Equivalencia lógica

Sustitución

Formas normales

Algoritmos para SAT y
TAUT

Tablero y Formas Normales

Forma clausal

Algoritmo DPLL

Estructura del algoritmo

Extracción de Modelos