

**Tema DA–4:
Lógica proposicional:
Tableros semánticos**

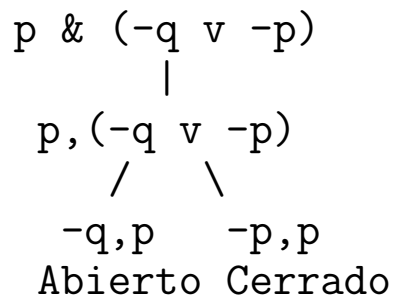
**José A. Alonso Jiménez
Miguel A. Gutiérrez Naranjo**

Dpto. de Ciencias de la Computación e Inteligencia Artificial

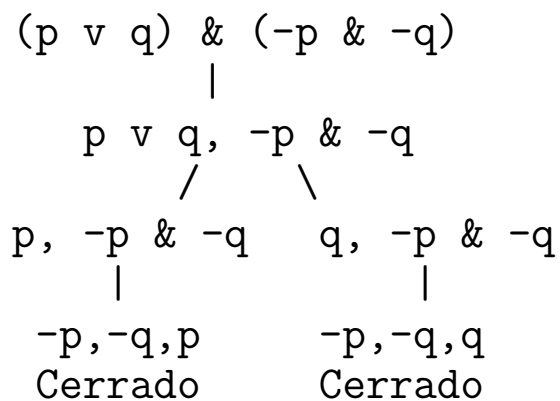
UNIVERSIDAD DE SEVILLA

Ejemplos de tableros semánticos

- Tablero semántico de $p \wedge (\neg q \vee \neg p)$



- Tablero semántico de $(p \vee q) \wedge (\neg p \wedge \neg q)$



Representación de las conectivas

- **Conectivas**

- Las conectivas lógicas son $-$, $\&$, \vee , \Rightarrow , \Leftrightarrow .
- La precedencia es $-$, $\&$, \vee , \Rightarrow , \Leftrightarrow .
- Las conectivas binarias asocian por la derecha.

- **Ejemplos:**

?- $p \& -q \vee r \Rightarrow s = ((p \& -q) \vee r) \Rightarrow s$.

Yes

?- $p \& -q \vee r \Rightarrow s = (p \& (-q \vee r)) \Rightarrow s$.

No

?- $p \& q \& r = p \& (q \& r)$.

Yes

?- $p \& q \& r = (p \& q) \& r$.

No

- **Declaración**

```
:- op(610, fy, -).      % negación
:- op(620, xfy,&).     % conjunción
:- op(630, xfy,v).     % disyunción
:- op(640, xfy,=>).    % implicación
:- op(650, xfy,<=>).   % equivalencia
```

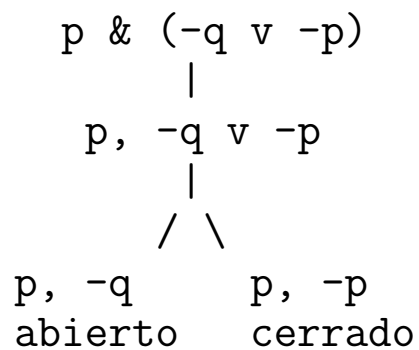
Representación de tableros

- Representación de tableros

- Los tableros se representan por términos

- Ejemplo:

Tablero:



Término:

$$\begin{array}{l} t([p \ \& \ (-q \ \vee \ -p)], \\ \quad t([p, \ -q \ \vee \ -p], \\ \qquad t([-q, \ p], \\ \qquad \quad \text{abierto}, \\ \qquad \quad \text{vacío}), \\ \qquad t([-p, \ p], \\ \qquad \quad \text{cerrado}, \\ \qquad \quad \text{vacío})), \\ \text{vacío}) \end{array}$$

Construcción de tableros

- `tablero(+F,-Tab)`

- **Especificación:** `tablero(+F,-Tab)` se verifica si `Tab` es el tablero de la fórmula `F`.

- **Ejemplo:**

```
?- tablero(p & (-q v -p),T).
T = t([p& (-q v-p)],
      t([p, -q v-p],
        t([-q, p], abierto, vacio),
        t([-p, p], cerrado, vacio)),
      vacio)
```

- **Definición:**

```
tablero(F,Tab) :-
    Tab = t([F], _, _),
    expande_tablero(Tab).
```

Construcción de tableros

- `expande_tablero(t(+Flas, -Izq, -Dcha))`
 - **Especificación:** se verifica si `Izq` y `Dcha` son los sub-tableros completos correspondiente a la raíz `Flas`. La reglas de expansión son:
 1. Si `Flas` es una lista de literales cerrada (i.e. contiene un literal y su negación), entonces `Izq` = cerrado y `Dcha` = vacío.
 2. Si `Flas` es una lista de literales (no cerrada), entonces `Izq` = cerrado y `Dcha` = vacío.
 3. Si `Flas` = `[--A|L]`, entonces `Izq` es el tablero de `[A|L]` y `Dcha` = vacío.
 4. Si `Flas` = `[A|L]` y `A` es una fórmula alfa (de componentes `A1` y `A2`), entonces `Izq` es el tablero de `[A1,A2|L]` y `Dcha` = vacío.
 5. Si `Flas` = `[B|L]` y `B` es una fórmula beta (de componentes `B1` y `B2`), entonces `Izq` es el tablero de `[B1|L]` y `Dcha` es el tablero de `[B2|L]`.
 6. En otro caso, el primer elemento de `Flas` es un literal y el tablero es el de la lista obtenida poniendo el primer elemento de `Flas` al final.

Construcción de tableros

- Definición:

```
expande_tablero(t(Flas, cerrado, vacio)) :-      % 1
    lista_de_literales(Flas),
    cerrada(Flas), !.
expande_tablero(t(Flas, abierto, vacio)) :-      % 2
    lista_de_literales(Flas), !.
expande_tablero(t([-(-A)|Flas], Izq, vacio)) :-  % 3
    !,
    Izq = t([A|Flas], _, _),
    expande_tablero(Izq).
expande_tablero(t([A | Flas], Izq, vacio)) :-    % 4
    alfa(A, A1, A2), !,
    Izq = t([A1, A2 | Flas], _, _),
    expande_tablero(Izq).
expande_tablero(t([B | Flas], Izq, Dcha)) :-     % 5
    beta(B, B1, B2), !,
    Izq = t([B1 | Flas], _, _),
    Dcha = t([B2 | Flas], _, _),
    expande_tablero(Izq),
    expande_tablero(Dcha).
expande_tablero(t([Fla | Flas], Izq, Dcha)) :-  % 6
    append(Flas, [Fla], NFlas),
    expande_tablero(t(NFlas, Izq, Dcha)).
```

Construcción de tableros

- `lista_de_literales(+L)`

- **Especificación:** se verifica si L es una lista de literales.

- **Ejemplos:**

```
?- lista_de_literales([p, - q, r]).  
Yes  
?- lista_de_literales([p, - q, r v s]).  
No
```

- **Definición**

```
lista_de_literales([]).  
lista_de_literales([A|L]) :-  
    literal(A),  
    lista_de_literales(L).
```

- `literal(+F)`

- **Especificación:** se verifica si la fórmula F es un literal.

- **Ejemplos:**

```
?- literal(p).  
Yes  
?- literal(- p).  
Yes  
?- literal(p => q).  
No
```


Construcción de tableros

- **Definición:**

```
literal(F) :- atom(F).  
literal(-F) :- atom(F).
```

- **cerrada(+L)**

- **Especificación:** se verifica si L es una lista cerrada (i.e. que contiene una fórmula y su negación).

- **Ejemplos:**

```
?- cerrada([p => q, r, - (p => q)]).  
Yes  
?- cerrada([p => q, r, - (p => r)]).  
No
```

- **Definición:**

```
cerrada(L) :-  
    member(-F,L),  
    member(F,L).
```

- **alfa(+A, -A1, -A2)**

- **Especificación:** se verifica si A es una fórmula alfa y sus componentes son A1 y A2.

- **Definición:**

```
alfa(A1 & A2, A1, A2).  
alfa(-(A1 => A2), A1, -A2).  
alfa(-(A1 v A2), - A1, -A2).  
alfa(A1 <=> A2, A1 => A2, A2 => A1).
```

Construcción de tableros

- $\text{beta}(+B, -B1, -B2)$
 - **Especificación:** se verifica si B es una fórmula beta y sus componentes son B1 y B2.
 - **Definición:**
 - $\text{beta}(B1 \vee B2, B1, B2)$.
 - $\text{beta}(B1 \Rightarrow B2, -B1, B2)$.
 - $\text{beta}(-(B1 \& B2), -B1, -B2)$.
 - $\text{beta}(-(B1 \Leftrightarrow B2), -(B1 \Rightarrow B2), -(B2 \Rightarrow B1))$.

Teoremas por tableros

- `es_teorema_tab(+F)`

- **Especificación:** se verifica si la fórmula F es teorema (mediante tableros)

- **Ejemplos:**

```
?- es_teorema_tab((p => q) v (q => p)).
```

```
Yes
```

```
?- es_teorema_tab((p => q) & (q => p)).
```

```
No
```

- **Definición:**

```
es_teorema_tab(F) :-  
    tablero(-F,Tab),  
    tablero_cerrado(Tab).
```

- `tablero_cerrado(+Tab)`

- **Especificación:** se verifica si Tab es un tablero cerrado (i.e. tiene todas sus ramas cerradas).

- **Definición:**

```
tablero_cerrado(t(_,Izq,Dcha)) :-  
    tablero_cerrado(Izq),  
    tablero_cerrado(Dcha).  
tablero_cerrado(cerrado).  
tablero_cerrado(vacio).
```

Consecuencia por tableros

- `es_consecuencia_tab(+S,+F)`

- **Especificación:** se verifica si la fórmula F es consecuencia del conjunto de fórmulas S .

- **Ejemplos:**

```
?- es_consecuencia_tab([p => q, q => r], p => r).
```

```
Yes
```

```
?- es_consecuencia_tab([p => q, q => r], p <=> r).
```

```
No
```

- **Definición:**

```
es_consecuencia_tab(S,F) :-  
  Tab = t([-F|S],_,_),  
  expande_tablero(Tab),  
  tablero_cerrado(Tab).
```

Construcción mejorada de tableros

- Mejora: Aplicar las reglas alfas antes que las betas.
- `expande_tablero(t(+Flas, -Izq, -Dcha))`
 - Especificación: se verifica si Izq y Dcha son los sub-tableros completos correspondiente a la raíz Flas. La reglas de expansión son:
 1. Si Flas es una lista cerrada (i.e. contiene una fórmula y su negación), entonces Izq = cerrado y Dcha = vacio.
 2. Si Flas es una lista de literales (no cerrada), entonces Izq = cerrado y Dcha = vacio.
 3. Si Flas tiene una fórmula alfa, entonces Izq es el tablero de la lista Flas sustituyendo una fórmula alfa de Flas por sus componentes y Dcha = vacio.
 4. Si Flas tiene una fórmula beta, entonces Izq es el tablero de la lista Flas sustituyendo una fórmula beta de Flas por una de sus componentes y Dcha por la otra.

Construcción mejorada de tableros

- **Definición:**

```
expande_tablero(t(Flas, cerrado, vacio)) :-          % 1
    cerrada(Flas), !.
expande_tablero(t(Flas, abierto, vacio)) :-         % 2
    lista_de_literales(Flas), !.
expande_tablero(t(Flas, Izq, vacio)) :-            % 3
    regla_alfa(Flas, Flas1), !,
    Izq = t(Flas1, _, _),
    expande_tablero(Izq).
expande_tablero(t(Flas, Izq, Dcha)) :-              % 4
    regla_beta(Flas, Flas1, Flas2),
    Izq = t(Flas1, _, _),
    Dcha = t(Flas2, _, _),
    expande_tablero(Izq),
    expande_tablero(Dcha).
```

- **regla_alfa(+Flas, -Flas1)**

- **Especificación:** se verifica si *Flas* es una lista de fórmulas que contiene al menos una fórmula alfa y *Flas1* es la lista de fórmulas obtenidas sustituyendo una fórmula alfa de *Flas* por sus componentes.

- **Ejemplos:**

```
?- regla_alfa([p & (q v r), p => r],L).
L = [p, q v r, p => r]
?- regla_alfa([p v (q v r), p => r],_L).
No
```

Construcción mejorada de tableros

- **Definición:**

```
regla_alfa(Flas, [A1,A2|Flas1]) :-  
    member(A, Flas),  
    alfa(A, A1, A2), !,  
    delete(Flas, A, Flas1).  
regla_alfa(Flas, [A1|Flas1]) :-  
    member(A, Flas),  
    A = -(-A1),  
    delete(Flas, A, Flas1).
```

- **regla_beta(+Flas, -Flas1, -Flas2)**

- **Especificación:** se verifica si *Flas* es una lista de fórmulas que contiene al menos una fórmula beta y *Flas1* es la lista de fórmulas obtenidas sustituyendo una fórmula beta de *Flas* por una de sus componentes y *Flas2*, por la otra.

- **Ejemplos:**

```
?- regla_beta([p & (q v r), p => r],L1,L2).  
L1 = [-p, p & (q v r)]  
L2 = [r, p & (q v r)]  
?- regla_beta([p & (q v r), -(p => r)],_L1,_L2).  
No
```

- **Definición:**

```
regla_beta(Flas, [B1|RFlas], [B2|RFlas]) :-  
    member(B, Flas),  
    beta(B, B1, B2),  
    delete(Flas, B, RFlas).
```

Bibliografía

- Ben–Ari, M. *Mathematical logic for computer science (2nd ed.)* (Springer–Verlag, 2001)
 - Cap. 2 “Propositional calculus: formulas, models, tableaux”.
- Fitting, M. *First-Order Logic and Automated Theorem Proving (2nd Ed.)* (Springer–Verlag, 1996)
 - Cap. 2 “Propositional Logic”.