Tema DA-8: Implementación en Prolog de los tableros semánticos

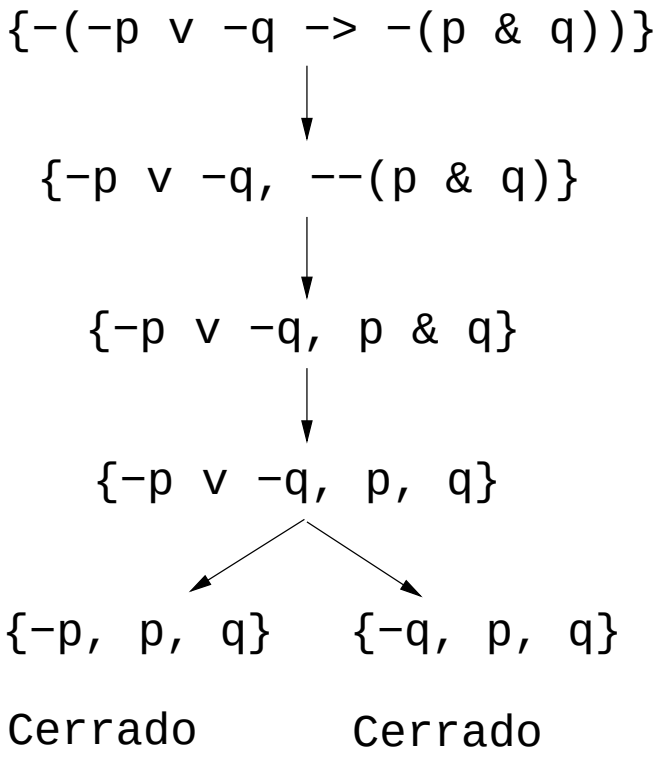
José A. Alonso Jiménez
Miguel A. Gutiérrez Naranjo

Dpto. de Ciencias de la Computación e Inteligencia Artificial

Universidad de Sevilla

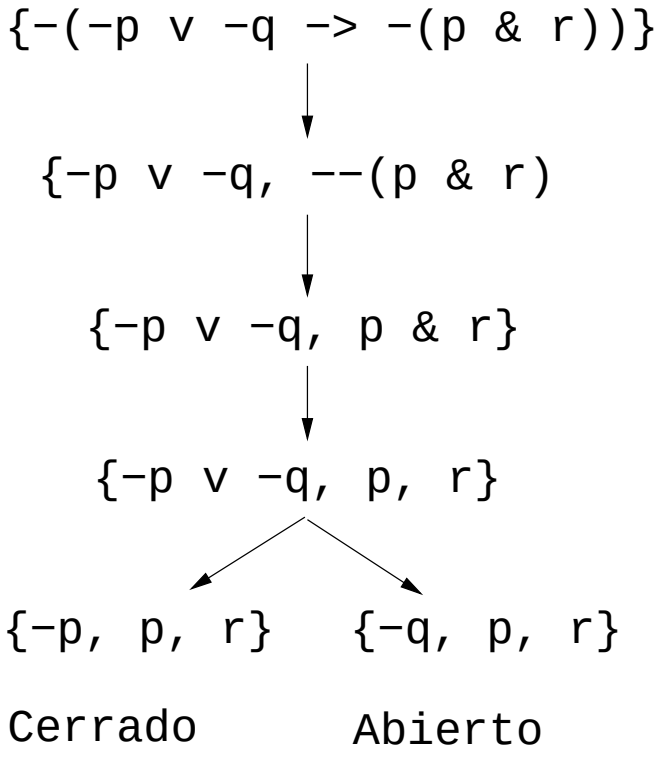
Demostración por tableros semánticos

- **Demostración de fórmula válida:**
  - **Demostración:**
    $\neg p \vee \neg q \longrightarrow \neg(p \wedge q)$ **es válida**
    syss $\{\neg(\neg p \vee \neg q \longrightarrow \neg(p \wedge q))\}$ **es inconsistente**
    syss $\{\neg p \vee \neg q, \neg\neg(p \wedge q)\}$ **es inconsistente**
    syss $\{\neg p \vee \neg q, p \wedge q\}$ **es inconsistente**
    syss $\{p, q, \neg p \vee \neg q\}$ **es inconsistente**
    syss $\{p, q, \neg p\}$ **es inconsistente y**
    $\{p, q, \neg q\}$ **es inconsistente**
  - **Tablero semántico:**



Demostración por tableros semánticos

- **Demostración de fórmula no válida:**
  - **Demostración:**
    $\neg p \vee \neg q \longrightarrow \neg(p \wedge r)$ **es válida**
    syss $\{\neg(\neg p \vee \neg q \longrightarrow \neg(p \wedge r))\}$ **es inconsistente**
    syss $\{\neg p \vee \neg q, \neg\neg(p \wedge r)\}$ **es inconsistente**
    syss $\{\neg p \vee \neg q, p \wedge r\}$ **es inconsistente**
    syss $\{p, q, \neg p \vee \neg r\}$ **es inconsistente**
    syss $\{p, q, \neg p\}$ **es inconsistente y**
    $\{p, q, \neg r\}$ **es inconsistente**
  - **Tablero semántico:**



Notación uniforme

- **Literales**
  - **Un *literal* es un átomo o la negación de un átomo.**
  - **`literal(+F)` se verifica si la fórmula `F` es un literal.**
  - **Def. de `literal`:**

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

- **Dobles negaciones**
  - **La fórmula F es una *doble negación* si existe una fórmula G tal que F es de la forma $\neg\neg G$.**
  - **Entonces $\models F \leftrightarrow G$.**
- **Fórmulas alfa**
  - **Las fórmulas *alfa*, junto con sus componentes, son las indicadas en la siguiente tabla**

$A_1 \wedge A_2$	A_1	A_2
$\neg(A_1 \rightarrow A_2)$	A_1	$\neg A_2$
$\neg(A_1 \vee A_2)$	$\neg A_1$	$\neg A_2$

  - **Si F es una fórmula alfa y sus componentes son F_1 y F_2, entonces $\models F \leftrightarrow F_1 \wedge F_2$.**
  - **`alfa(+A,-A1,-A2)` se verifica si `A` es una fórmula alfa y sus componentes son `A1` y `A2`.**
  - **Def. de `alfa`:**

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

Notación uniforme

- **Fórmulas beta**
  - **Las fórmulas *beta*, junto con sus componentes, son las indicadas en la siguiente tabla**

$B_1 \vee B_2$	B_1	B_2
$B_1 \rightarrow B_2$	$\neg B_1$	B_2
$\neg(B_1 \wedge B_2)$	$\neg B_1$	$\neg B_2$
$B_1 \leftrightarrow B_2$	$B_1 \wedge B_2$	$\neg B_1 \wedge \neg B_2$
$\neg(B_1 \leftrightarrow B_2)$	$B_1 \wedge \neg B_2$	$\neg B_1 \wedge B_2$

  - **Si F es una fórmula beta y sus componentes son F_1 y F_2, entonces $\models F \leftrightarrow F_1 \vee F_2$.**
  - **`beta(+B,-B1,-B2)` se verifica si `B` es una fórmula beta y sus componentes son `B1` y `B2`.**
  - **Def. de `beta`:**

```
beta(B1 v B2,       B1,        B2).
beta(B1 => B2,      -B1,       B2).
beta(-(B1 & B2),    -B1,       -B2).
beta(B1 <=> B2,     B1 & B2,   -B1 & -B2).
beta(-(B1 <=> B2), B1 & -B2, -B1 & B2).
```

Completación de tableros

- **Un tablero correspondiente a un conjunto de fórmulas S es un árbol construido mediante las reglas:**

  (I) El árbol cuyo único nodo tiene como etiqueta S es un tablero de S.

  (C) Sea $\mathcal{T}$ un tablero de S y S_1 la etiqueta de una hoja de $\mathcal{T}$.

  (C.1) Si S_1 cerrado (es decir, que contiene una fórmula y su negación), entonces el árbol obtenido añadiendo como hijo de S_1 el nodo etiquetado con `cerrado` es un tablero de S.

  (C.2) Si S_1 es abierto (es decir, es un conjunto de literales que no contiene una fórmula y su negación), entonces el árbol obtenido añadiendo como hijo de S_1 el nodo etiquetado con `abierto` es un tablero de S.

  (C.3) Si S_1 contiene una doble negación $\neg\neg F$, entonces el árbol obtenido añadiendo como hijo de S_1 el nodo etiquetado con $(S_1 - \{\neg\neg F\}) \cup \{F\}$ es un tablero de S.

  (C.4) Si S_1 contiene una fórmula alfa F de componentes F_1 y F_2, entonces el árbol obtenido añadiendo como hijo de S_1 el nodo etiquetado con $(S_1 - \{F\}) \cup \{F_1, F_2\}$ es un tablero de S.

  (C.5) Si S_1 contiene una fórmula beta F de componentes F_1 y F_2, entonces el árbol obtenido añadiendo como hijos de S_1 los nodos etiquetados con $(S_1 - \{F\}) \cup \{F_1\}$ y $(S_1 - \{F\}) \cup \{F_2\}$ es un tablero de S.

Completación de tableros

- **Un tablero semántico de S es *completo* si no se le puede aplicar ninguna de las reglas de expansión; es decir, todas sus hojas son abiertas o cerradas.**
- **`tablero_completo(+S,-Tab)` se verifica si `Tab` es un tablero completo del conjunto de fórmulas `S`.**
- **Ejemplo:**

```
?- tablero_completo([- (-p v -q => - (p & r))],T).
T = t([- (-p v -q => - (p & r))],
      t([-p v -q, - -(p & r)],
        t([p & r, -p v -q],
          t([p, r, -p v -q],
            t([-p, p, r], cerrado, vacio),
            t([-q, p, r], abierto, vacio)),
          vacio),
        vacio),
      vacio)
Yes
```

- **Representación: `t(S,Izq,Dcha)` representa el tablero de raíz `S`, rama izquierda `Izq` y rama derecha `Dcha`.**
- **Def. de `tablero_completo`:**

```
tablero_completo(S,Tab) :-
   completacion(t(S,_Izq,_Dcha),Tab).
```

Completación de tableros

- `completacion(+Tab1,-Tab2)` **se verifica si** `Tab2` **es una completación (i.e. un tablero completo) del tablero** `Tab1`**.**
- **Def. de** `completacion`**:**

```
completacion(t(Flas,Izq1,Dcha1),t(Flas,Izq3,Dcha3)) :-
   paso(t(Flas,Izq1,Dcha1),t(Flas,Izq2,Dcha2)), !,
   completacion(Izq2,Izq3),
   completacion(Dcha2,Dcha3).
completacion(Tab,Tab).
```

- `paso(+Tab1,-Tab2)` **se verifica si** `Tab2` **es un tablero obtenido aplicando una regla de completación al tablero** `Tab1`**.**
- **Def. de** `paso`

```
paso(t(S1,_,_),t(S1, cerrado, vacio)) :-     % C1
   cerrada(S1), !.
paso(t(S1,_,_),t(S1, abierto, vacio)) :-     % C2
   lista_de_literales(S1), !.
paso(t(S1,_,_),t(S1, Izq, vacio)) :-         % C3
   regla_doble_negacion(S1, S2), !,
   Izq = t(S2, _, _).
paso(t(S1,_,_),t(S1, Izq, vacio)) :-         % C4
   regla_alfa(S1, S2), !,
   Izq = t(S2, _, _).
paso(t(S1,_,_),t(S1, Izq, Dcha)) :-          % C5
   regla_beta(S1, S2, S3),
   Izq  = t(S2, _, _),
   Dcha = t(S3, _, _).
```

Completación de tableros

- `cerrada(+S)` **se verifica si** `S` **es una lista cerrada (i.e. que contiene una fórmula y su negación).**
- **Def. de** `cerrada`**:**

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

- `lista_de_literales(+S)` **se verifica si** `S` **es una lista de literales.**
- **Def. de** `lista_de_literales`**:**

```
lista_de_literales([]).
lista_de_literales([F|S]) :-
   literal(F),
   lista_de_literales(S).
```

- `regla_doble_negacion(+S1,-S2)` **que se verifica si** `S1` **es una lista de fórmulas que contiene una doble negación** `--F` **y** `S2` **es la lista de fórmulas obtenidas sustituyendo en** `S1` **la fórmula** `--F` **por** `F`**.**
- **Ejemplos:**

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

- **Def. de** `regla_doble_negacion`**:**

```
regla_doble_negacion(S1, [F|S2]) :-
   member(- -F, S1), !,
   delete(S1, - -F, S2).
```

Completación de tableros

- **regla_alfa(+S1,-S2) que se verifica si S1 es una lista de fórmulas que contiene una fórmula alfa A y S2 es la lista de fórmulas obtenidas sustituyendo en S1 la fórmula A 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
```

- **Def. de regla_alfa(+S1,-S2):**

```
regla_alfa(S1, [A1,A2|S2]) :-
   member(A, S1),
   alfa(A, A1, A2), !,
   delete(S1, A, S2).
```

- **regla_beta(+S1,-S2,-S3) que se verifica si S1 es una lista de fórmulas que contiene al menos una fórmula beta B y S2 es la lista de fórmulas obtenidas sustituyendo en S1 la fórmula B por una de sus componentes y S3, 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
```

- **Def. de regla_beta:**

```
regla_beta(S1, [B1|RS1], [B2|RS1]) :-
   member(B, S1),
   beta(B, B1, B2),
   delete(S1, B, RS1).
```

Tableros cerrados

- **Tableros cerrados**
  - **Un tablero es *cerrado* si todas sus hojas están etiquetadas con `cerrado` o `vacio`.**
  - **`es_cerrado(+Tab)` se verifica si `Tab` es un tablero cerrado.**
  - **Def. de `es_cerrado`:**

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

Teorema por tableros

- **Una fórmula F es un *teorema (mediante tableros semánticos)* si tiene una *prueba mediante tableros*; es decir, si $\{\neg F\}$ tiene un tablero completo cerrado.**
- **Prueba mediante tableros**
  - `prueba(+F,-Tab)` **se verifica si** `Tab` **es una prueba (mediante tableros semánticos) de la fórmula** `F`**.**
  - **Ejemplos:**

```
?- prueba(-p v -q => -(p & q),T).
T = t([- (-p v-q => - (p & q))],
      t([-p v-q, - -(p & q)],
        t([p & q, -p v -q],
          t([p, q, -p v -q],
            t([-p, p, q], cerrado, vacio),
            t([-q, p, q], cerrado, vacio)),
          vacio),
        vacio),
      vacio)
?- prueba(-p v -q => -p,T).
No
```

  - **Def. de** `prueba`**:**

```
prueba(F,Tab) :-
   tablero_completo([-F],Tab), !,
   es_cerrado(Tab).
```

Teorema por tableros

- **Teorema mediante tableros**
  - `es_teorema(+F)` **se verifica si la fórmula** `F` **es teorema (mediante tableros semánticos).**
  - **Ejemplos:**

```
?- es_teorema(-p v -q => -(p & q)).
Yes
?- es_teorema(-p v -q => -(p & r)).
No
```

  - **Def. de** `es_teorema`**:**

```
es_teorema(F) :-
   prueba(F,_Tab).
```

- **El cálculo de tableros semánticos es adecuado y completo; es decir, una fórmula es válida syss es teorema mediante tableros semánticos.**

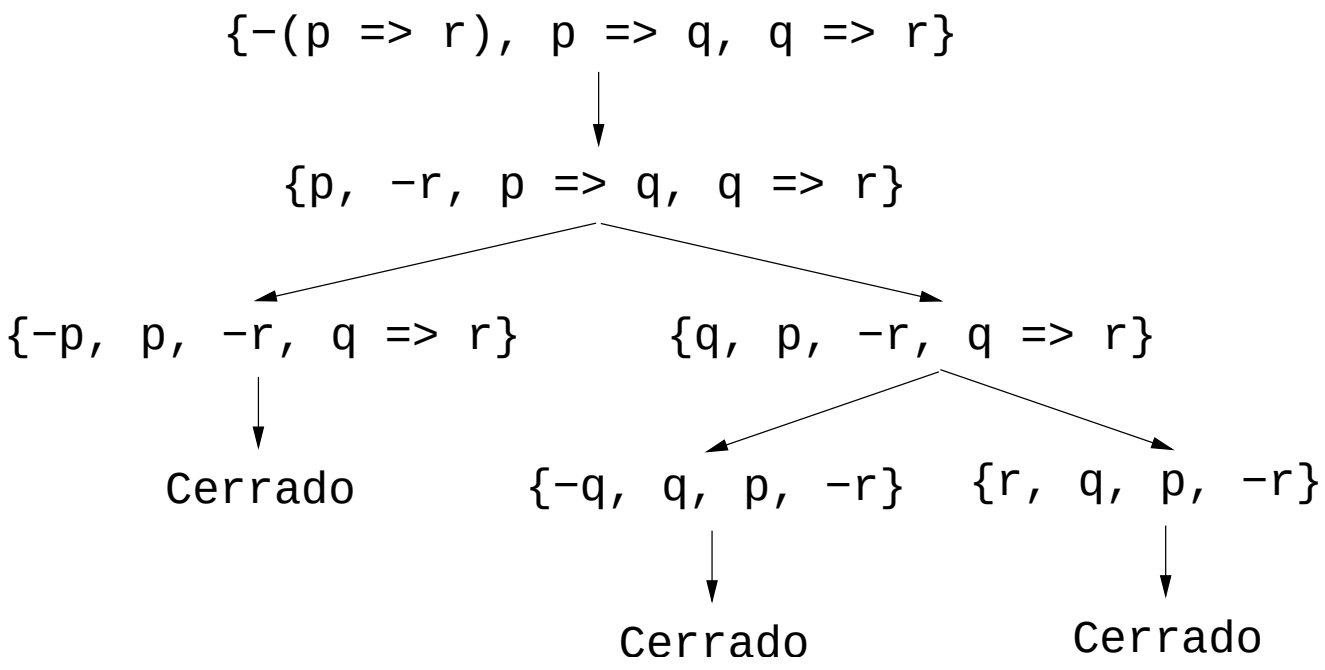
Deducción por tableros

- **La fórmula F es *deducible (mediante tableros semánticos)* a partir del conjunto de fórmulas S si existe una prueba mediante tableros de que F a partir de S; es decir, existe un tablero completo cerrado de $S \cup \{\neg F\}$.**

- **Deducción por tableros**
  - `prueba_deducible_tab(+S,+F,-Tab)` **se verifica si** `Tab` **es una prueba por tableros semánticos de que la fórmula** `F` **es deducible del conjunto de fórmulas** `S`**.**
  - **Ejemplos:**

```
?- prueba_deducible_tab([p => q, q => r], p => r,T).
T = t([- (p => r), p => q, q => r],
      t([p, -r, p => q, q => r],
        t([-p, p, -r, q => r], cerrado, vacio),
        t([q, p, -r, q => r],
          t([-q, q, p, -r], cerrado, vacio),
          t([r, q, p, -r], cerrado, vacio))),
      vacio)
Yes
?- prueba_deducible_tab([p => q, q => r], p <=> r,T).
No
```

Deducción por tableros



- **Def. de** `prueba_deducible_tab`**:**

```
prueba_deducible_tab(S,F,Tab) :-
   tablero_completo([-F|S],Tab), !,
   es_cerrado(Tab).
```

- **Deducibilidad por tableros**
  - `es_deducible_tab(+S,+F)` **se verifica si la fórmula** `F` **es deducible (mediante tableros) del conjunto de fórmulas** `S`**.**
  - **Def. de** `es_deducible_tab`**:**

```
es_deducible_tab(S,F) :-
   prueba_deducible_tab(S,F,_Tab).
```

- **La fórmula** F **es consecuencia de** S **syss** F **es deducible mediante tableros a partir de** S**.**

Bibliografía

- **Alonso, J.A. y Borrego, J.** ***Deducción automática (Vol. 1: Construcción lógica de sistemas lógicos)*** **(Ed. Kronos, 2002)**
  `www.cs.us.es/~jalonso/libros/da1-02.pdf`
  - **Cap. 4.1: Tableros semánticos**
- **Ben–Ari, M.** ***Mathematical Logic for Computer Science (2nd ed.)*** **(Springer, 2001)**
  - **Cap. 2: Propositional Calculus: Formulas, Models, Tableaux**
- **Fitting, M.** ***First-Order Logic and Automated Theorem Proving (2nd ed.)*** **(Springer, 1995)**
- **Nerode,A. y Shore, R.A.** ***Logic for Applications*** **(Springer, 1997)**