Tema 11: Implementación en Prolog de la transformación a cláusulas

José A. Alonso Jiménez

Jose-Antonio.Alonso@cs.us.es
http://www.cs.us.es/∼jalonso

Dpto. de Ciencias de la Computación e Inteligencia Artificial

Universidad de Sevilla

Equivalencia lógica

- ***F* y *G* son *equivalentes* si $\models F \leftrightarrow G$.**
- **Decisión de la equivalencia:**
  - `es_equivalente(+F,+G)` **se verifica si las fórmulas** `F` **y** `G` **son equivalentes.**
  - **Ejemplos:**

```
?- es_equivalente(-(p & q), -p v -q).
Yes
?- es_equivalente(-(p & q), -p & -q).
No
```

  - **Def. de** `es_equivalente`**:**

```
es_equivalente(F,G) :-
   es_tautologia(F <=> G).
```

Forma normal negativa

- **Forma normal negativa:**
  - **Una fórmula *está en forma normal negativa (FNN)* si no contiene las conectivas $\rightarrow$, $\leftrightarrow$ y la negación no se aplica a subfórmulas compuestas.**
  - **Ejemplos:**
    * **$(\neg p \vee q) \wedge (\neg q \vee p)$ está en FNN**
    * **$(p \rightarrow q) \wedge (q \rightarrow p)$ no está en FNN**
    * **$\neg(p \wedge q)$ no está en FNN**
  - **Una fórmula *G es una forma normal negativa (FNN)* de la fórmula F si G está en forma normal negativa y es equivalente a F.**
  - **Ejemplo: una FNN de $p \leftrightarrow q$ es $(\neg p \vee q) \wedge (\neg q \vee p)$.**

Cálculo de forma normal negativa

- **Aplicando a una fórmula F los siguientes pasos se obtiene una forma normal negativa de F:**

  1. **Eliminar las equivalencias usando la relación**

     $$A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A) \quad (1)$$

  2. **Eliminar las implicaciones usando la equivalencia**

     $$A \rightarrow B \equiv \neg A \vee B \quad (2)$$

  3. **Interiorizar las negaciones usando las equivalencias**

     $$\neg(A \wedge B) \equiv \neg A \vee \neg B \quad (3)$$
     $$\neg(A \vee B) \equiv \neg A \wedge \neg B \quad (4)$$
     $$\neg\neg A \equiv A \quad (5)$$

Cálculo de forma normal negativa

- `fnn(+F,?G)` **se verifica si** `G` **es una forma normal negativa de la fórmula** `F`**.**
- **Ejemplos:**

```
?- fnn(p <=> q,F).
F = (-p v q) & (-q v p)
?- fnn(p v -q => r, F).
F = (-p & q) v r
?- fnn(p & (q => r) => s, F).
F = (-p v (q & -r)) v s
```

- **Def. de** `fnn`**:**

```
fnn(F, G) :-
   elimina_equivalencias(F,F1),
   elimina_implicaciones(F1,F2),
   interioriza_negaciones(F2,G).
```

Cálculo de forma normal negativa

- `elimina_equivalencias(+F,?G)` **se verifica si** `G` **es la fórmula obtenida eliminando las equivalencias de la fórmula** `F`**, usando la reducción (1).**
- **Def. de** `elimina_equivalencias`**:**

```
elimina_equivalencias(A <=> B, (A1 => B1) & (B1 => A1)) :- !,
   elimina_equivalencias(A, A1),
   elimina_equivalencias(B, B1).
elimina_equivalencias(A, B) :-
   A =.. [Op|L1], !,
   maplist(elimina_equivalencias,L1,L2),
   B =.. [Op|L2].
elimina_equivalencias(A, A).
```

Cálculo de forma normal negativa

- `elimina_implicaciones(+F,?G)` **se verifica si** `G` **es la fórmula obtenida eliminando las implicaciones de la fórmula** `F`**, usando la reducción (2).**
- **Def. de** `elimina_implicaciones`**:**

```
elimina_implicaciones(A => B, -A1 v B1) :- !,
   elimina_implicaciones(A, A1),
   elimina_implicaciones(B, B1).
elimina_implicaciones(A, B) :-
   A =.. [Op|L1], !,
   maplist(elimina_implicaciones,L1,L2),
   B =.. [Op|L2].
elimina_implicaciones(A, A).
```

Cálculo de forma normal negativa

- `interioriza_negaciones(+F,?G)` **se verifica si** `G` **es la fórmula obtenida interiorizando las negaciones de la fórmula** `F` **(que no tiene => ni <=>) de forma que las negaciones se apliquen sólo sobre símbolos proposicionales. Las reglas de interiorización son las reducciones (3), (4) y (5).**
- **Def. de** `interioriza_negaciones`**:**

```
interioriza_negaciones(-(A & B), A1 v B1) :- !,
   interioriza_negaciones(-A, A1),
   interioriza_negaciones(-B, B1).
interioriza_negaciones(-(A v B), A1 & B1) :- !,
   interioriza_negaciones(-A, A1),
   interioriza_negaciones(-B, B1).
interioriza_negaciones((-(-A)),  A1) :- !,
   interioriza_negaciones(A, A1).
interioriza_negaciones(A, B) :-
   A =.. [Op|L1], !,
   maplist(interioriza_negaciones,L1,L2),
   B =.. [Op|L2].
interioriza_negaciones(A, A).
```

Forma normal conjuntiva

- **Forma normal conjuntiva:**
  - **Una fórmula está en *forma normal conjuntiva (FNC)* si es una conjunción de disyunciones de literales; es decir, es de la forma $(L_{1,1} \vee \ldots \vee L_{1,n_1}) \wedge \ldots \wedge (L_{m,1} \vee \ldots \vee L_{m,n_m})$.**
  - **Ejemplos:**
    - **$(\neg p \vee q) \wedge (\neg q \vee p)$ está en FNC**
    - **$(\neg p \vee q) \wedge (q \rightarrow p)$ no está en FNC**
  - **Una fórmula *G es una forma normal conjuntiva (FNC)* de la fórmula F si G está en forma normal conjuntiva y es equivalente a F.**
  - **Ejemplo: Una FNC de $\neg(p \wedge (q \rightarrow r))$ es $(\neg p \vee q) \wedge (\neg p \vee \neg r)$.**

Cálculo de forma normal conjuntiva

- **Aplicando a una fórmula F los siguientes pasos se obtiene una forma normal conjuntiva de F:**

  1. **Calcular una forma normal negativa de F.**
  2. **Interiorizar las disyunciones usando la propiedad distributiva de la disyunción sobre la conjunción**

$$A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C) \quad (6)$$
$$(A \wedge B) \vee C \equiv (A \vee C) \wedge (B \vee C) \quad (7)$$

- **`fnc(+F,?G)` se verifica si `G` es una forma normal conjuntiva de la fórmula `F`.**
- **Ejemplos:**

```
?- fnc(p & (q => r), F).
F = p& (-q v r)
?- fnc(-(p & (q => r)), F).
F = (-p v q)& (-p v-r)
```

- **Def. de `fnc`:**

```
fnc(F,G) :-
   fnn(F,F1),
   interioriza_disyunciones(F1,G).
```

Cálculo de forma normal conjuntiva

```
interioriza_disyunciones(A v (B & C), ABC) :- !,
   interioriza_disyunciones(A v B, AB),
   interioriza_disyunciones(A v C, AC),
   interioriza_disyunciones(AB & AC, ABC).
interioriza_disyunciones((A & B) v C, ABC) :- !,
   interioriza_disyunciones(A v C, AC),
   interioriza_disyunciones(B v C, BC),
   interioriza_disyunciones(AC & BC, ABC).
interioriza_disyunciones(A, B) :-
   A =.. [Op|L],
   maplist(interioriza_disyunciones,L,L1),
   ( L1 = L ->
     B =.. [Op|L1]
   ; % not(L1 = L) ->
     B1 =.. [Op|L1],
     interioriza_disyunciones(B1,B)).
interioriza_disyunciones(A, A).
```

Transformación a cláusulas

- **Cláusulas**
  - Una *cláusula* es un conjunto finito de literales.
  - Variables sobre cláusulas: $C, C_1, C_2, \ldots$
  - El *valor de una cláusula* C en una interpretación I es

$$I(C) = \begin{cases} 1, & \text{si existe un } L \in C \text{ tal que } I(L) = 1 \\ 0, & \text{en caso contrario.} \end{cases}$$

  - Una fórmula F y una cláusula C son *equivalentes* si $I(F) = I(C)$ para cualquier interpretación I.
  - Si $C = \{L_1, L_2, \ldots, L_n\}$, entonces C es equivalente a $L_1 \vee L_2 \vee \ldots \vee L_n$.
  - La *cláusula vacía* es el conjunto vacío de cláusulas y se representa por $\square$.
  - En cualquier interpretación I, $I(\square) = 0$.

Transformación a cláusulas

- **Fórmulas clausales**
  - Las ***fórmulas clausales*** son las fórmulas obtenidas mediante las siguientes reglas
    * Si F es un literal, entonces F es una fórmula clausal
    * Si F y G son fórmulas clausales, entonces $F \vee G$ es una fórmula clausal.
  - Ejemplos:
    * p, $\neg p$ y $\neg p \vee (q \vee \neg r)$ son fórmulas clausales
    * $\neg p \vee (q \wedge \neg r)$ no es una fórmula clausal

Transformación a cláusulas

- **Transformación de fórmulas clausales en cláusulas**
  - **El siguiente procedimiento transforma fórmulas clausales en cláusulas equivalentes:**

$$\text{Cláusula}(\boldsymbol{F}) = = \begin{cases} \{\boldsymbol{F}\}, & \text{si } \boldsymbol{F} \text{ es un literal;} \\ \text{Cláusula}(\boldsymbol{F}_1) \cup \text{Cláusula}(\boldsymbol{F}_2), & \text{si } \boldsymbol{F} = (\boldsymbol{F}_1 \vee \boldsymbol{F}_2) \end{cases}$$

  - `cláusula(+F,-C)` **se verifica si** `C` **es una cláusula equivalente a la fórmula clausal** `F`
  - **Ejemplos:**

```
?- cláusula(p,C).                    =>  C = [p]
?- cláusula(-p,C).                   =>  C = [-p]
?- cláusula((-p v r) v (-p v q),C).  =>  C = [q, r, -p]
```

  - **Def. de** `cláusula`**:**

```
cláusula(F1 v F2, S) :- !,
   cláusula(F1, S1),
   cláusula(F2, S2),
   append(S1, S2, S3),
   sort(S3,S).
cláusula(L, [L]).
```

Transformación a cláusulas

- **Conjuntos de cláusulas**
  - El *valor de un conjunto de cláusulas* S en una interpretación I es

$$I(S) = \begin{cases} 1, & \text{si para toda } C \in S,\ I(C) = 1 \\ 0, & \text{en caso contrario.} \end{cases}$$

  - Una interpretación I es *modelo de un conjunto de cláusulas* S si $I(S) = 1$.
  - Un conjunto de cláusulas es *inconsistente* si no tiene modelos.
  - Una fórmula F y un conjunto de cláusulas S son *equivalentes* si $I(F) = I(S)$ para cualquier interpretación I.
  - Si $S = \{C_1, C_2, \ldots, C_n\}$, entonces S es equivalente a $C_1 \wedge C_2 \wedge \ldots \wedge C_n$.

Transformación a cláusulas

- **Transformación de fórmulas en FNC a conjunto de cláusulas**
  - **El siguiente procedimiento transforma fórmulas en forma normal conjuntiva en conjuntos cláusulas equivalentes. Cláusulas–FNC(F) es**
    * **Cláusulas–FNC(F_1) $\cup$ Cláusulas–FNC(F_2), si $F = F_1 \wedge F_2$**
    * **{Cláusula(F)}, en caso contrario**
  - **`cláusulas_FNC(+F,?S)` se verifica si `S` es un conjunto de cláusulas equivalente a la fórmula en forma normal conjuntiva `F`. Por ejemplo,**

```
?- cláusulas_FNC(p & (-q v r), S).          =>  S = [[p], [r, -q]]
?- cláusulas_FNC((-p v q) & (-p v -r), S).  =>  S = [[q, -p], [-p, -r]]
```

  - **Def. de `cláusulas_FNC`:**

```
cláusulas_FNC(A1 & A2, S) :- !,
   cláusulas_FNC(A1, S1),
   cláusulas_FNC(A2, S2),
   union(S1, S2, S).
cláusulas_FNC(A, [S]) :-
   cláusula(A, S).
```

Transformación a cláusulas

- **Transformación de fórmulas a conjunto de cláusulas**
  - `cláusulas(+F,?S)` **se verifica si** `S` **es un conjunto de cláusulas equivalente a la fórmula** `F`**.**
  - **Ejemplos:**

```
?- cláusulas(p & (q => r),S).
S = [[p], [r, -q]]
```

  - **Def. de** `cláusulas`**:**

```
cláusulas(F,S) :-
   fnc(F,F1),
   cláusulas_FNC(F1,S).
```

Transformación a cláusulas

- **Transformación de conjuntos de fórmulas a conjunto de cláusulas**
  - **El conjunto de fórmulas S_1 y el conjunto de cláusulas S_2 son *equivalentes* si $I(S_1) = I(S_2)$ para cualquier interpretación I.**
  - `cláusulas_conjunto(+CF,-CC)` **se verifica si** `CC` **es un conjunto de cláusulas equivalente al conjunto de fórmulas** `CF`**.**
  - **Ejemplo:**

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

  - **Def. de** `cláusulas_conjunto`**:**

```
cláusulas_conjunto([],[]).
cláusulas_conjunto([F|CF],CC) :-
   cláusulas(F,CC1),
   cláusulas_conjunto(CF,CC2),
   union(CC1,CC2,CC).
```

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. 4.2: Cláusulas**
- **Chang, C.–L. y Lee, R.C.–T. *Symbolic Logic and Mechanical Theorem Proving* (Academic Press, 1973)**
- **Fitting, M. *First-Order Logic and Automated Theorem Proving (2nd ed.)* (Springer, 1995)**
- **Schöning, U. *Logic for Computer Scientists* (Birkäuser, 1989)**